Studies in Computational Intelligence 432

Samee Ullah Khan
Joanna Kołodziej
Juan Li
Albert Y. Zomaya (Eds.)

Evolutionary Based Solutions for Green Computing

Springer

Studies in Computational Intelligence 432

For further volumes:
http://www.springer.com/series/7092

Samee Ullah Khan, Joanna Kołodziej, Juan Li,
and Albert Y. Zomaya (Eds.)

Evolutionary Based Solutions for Green Computing

Springer

Editors
Samee Ullah Khan
Department of Electrical
and Computer Engineering
North Dakota State University
Fargo
USA

Joanna Kołodziej
Institute of Computer Science
Cracow University of Technology
Cracow
Poland

Juan Li
Department of Computer Science
North Dakota State University
Fargo
USA

Albert Y. Zomaya
High Performance Computing
and Networking Center
School of Information Technologies
University of Sydney
Sydney
Australia

ISSN 1860-949X
ISBN 978-3-642-30658-7
DOI 10.1007/978-3-642-30659-4
Springer Heidelberg New York Dordrecht London

e-ISSN 1860-9503
e-ISBN 978-3-642-30659-4

Library of Congress Control Number: 2012940775

To our Families and Friends with Love and Gratitude

Preface

Over the last decade we have witnessed a growing interest of computing service providers in designing intelligent models and methodologies and upgrading the existing infrastructures to high-performance computing systems that can meet the increasing demands of powerful newer applications. In parallel, almost in concert, computing manufacturers have consolidated and moved from the stand-alone servers to rack mounted blades. Highly parametrized large-scale distributed computing systems can may be composed of made up of a large amount number of various components (computers, databases, etc) and must provide a wide range of services, not limited just to high performance computing platforms. Geographically distributed users may have a limited access to the system's services and resources and different, often conflicting, requirements. Moreover, Also the information and data processed in such dynamic environments may be incomplete, imprecise, fragmentary, and overloading. All of the above mentioned issues will necessitate the development of intelligent resource management techniques. On the other hand, the complex multi-level management systems may directly lead to increasing of the electricity usage in such systems. An optimal energy utilization has reached to a point that many information technology (IT) managers and corporate executives are all up in arms to identify scalable solution that can reduce electricity consumption (so that the total cost of operation is minimized) of their respective large-scale computing systems and simultaneously improve upon or maintain the current throughput of the system.

There are two fundamental problems that must addressed when considering a holistic solution that will deliver an energy-efficient resource allocation in high-performance computing, namely, the (a) total cost of operation (TCO) and (b) heat dissipation. However, both of these problems can be resolved by energy-efficient (re) allocation and management of resources. If absolutely nothing is done, then the power consumption per system is bound to increase which in turn would increase electricity and maintenance costs. Most of the electricity consumed by the high-performance processors is converted to heat, which can have a severe impact on the overall performance of a tightly packed system in racks (or blade chassis). Therefore, it is imperative to gather the consent of researchers to muster their efforts

in proposing unifying solutions that are practical and applicable in the domain of high-performance computing systems.

It is interesting to observe that Still not so large of a family of energy-aware metaheuristic- and, in particular, evolutionary-based optimization methods are presented in literature. Basically, classical single population genetic strategies are used as the energy optimizers. The adaptation of such methodologies for solving the optimization problems in large-scale dynamic environments requires an application of highly specialized genetic operators. The energy consumed by the system is usually just one component of a multi-objective fitness function. In such a case, the Multi-objective Genetic Algorithm (MOGA) framework seems to be a key solution to tackle the complexity of the optimization process. Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) algorithms are useful in generating the optimal paths and tree structures in graph-based models of networks, multi-processor machines and parallel applications. Finally, just few approaches in grid and cloud scheduling show that island, hierarchical and cellular parallel GAs can essentially speed up the optimization process and improve the exploration and exploitation of the large search space.

This book herewith presents the key ideas in the analysis, implementation, and evaluation of the next generation of intelligent techniques for the formulation and solution of complex green computing and in general green IT problems. The intelligent evolutionary and general metaheuristic-based solutions for energy optimization in data processing, scheduling, resource allocation and communication in modern grid, could and network computing are presented along with several important conventional technologies to cover the hot topics from the fundamental theory of the "green computing" concept and to describe the basic architectures of systems. Its eight chapters are informally structured into three main parts:

I. ***Where we are?:*** The energy effective scheduling, data processing, communication and resource allocation issues in large-scale distributed computing systems have been intensively studied over the last years. However, still not so many examples of the implementation of the energy-aware evolutionary-based methodologies can be found in the literature. Abbasi et al. in Chapter 1 provide a comprehensive survey on state-of-the-art in evolutionary-driven green computing in Distributed Cyber Physical (DCP) infrastructures. They briefly characterize the main types of the DCp systems, namely Data Centers (DCs), Computational Grids and Clouds, Wireless Sensor Networks (WSNs), and Body Sensor Networks (BSN), and specify the global optimization problems related to the cumulative energy utilization. The effectiveness of selected technologies is illustrated in the case studies in of each of the each considered environment.
II. ***Energy Awareness in High Performance Computing Systems:*** Modern High Performance Computing (HPC) systems are designed as the multi-level platforms for providing various services for the users. The users are usually associated in with numerous virtual organizations for the authorization of their access to the system services and resources. Therefore, an effective power supply and energy consumption management in today's grid, cluster, and cloud

infrastructures strongly depends on the structure of all the physical devices, middleware and virtual user services layers in the system architecture. Kipp et al. in Chapter 2 focus on the problem of the optimization of the energy utilization in the provisioning of HPC services through Internet. The authors formulate the problem instance according to the Quality of Service (QoS) requirements of the users and service providers, through resource consolidation and migration techniques, or by adjusting the state of physical nodes. It seems however that in HPC systems in most of the energy is still consumed by the servers and local clusters of physical devices. In Chapter 3 Enokido et al. analyze the formal model of optimal power supply for servers in macro and micro scales in HPC environments. Their analysis and achieved results can be recommended as a fundamental methodology for the systems designers. The problem of the equipment of the grid systems with low-power computing devices with modular power-supply modules is discussed in Chapter 4. The authors formulate the scheduling problem in computational grids as the multi-objective global optimization task with energy consumption and secure resource allocation as the main scheduling criteria. The effectiveness of single- and multi-population genetic-based schedulers has been justified in comprehensive empirical analysis. Finally, Shen and Zhang in Chapter 5 define the generic model of simple shadow price technique for task deadline constraint scheduling problem. This model can be easily adapted to various distributed computing environments.

III. ***Energy Aware Management in Many Core Systems with Wireless Communication:*** The last three chapters of the book are dedicated to the thermal and power management in many core systems with wireless communication. Kudithipudi et al. in Chapter 6 discuss the specific design challenges in monitoring the temperature in large-scale multi-core systems. They present the methods of exploitation of the multi-level optimizations at runtime in response to the dynamic behavior of the processor's workloads and related thermal distribution in such systems. The low-energy wireless communication techniques in many-core systems are defined by Ganguly et. al in Chapter 7. In Chapter 8 Marcelloni and Veccio present the evolutionary solution for two fundamental problems of data compression and node localization in Wireless Sensor Networks with multi-core servers.

Acknowledgements

We are grateful to all the contributors of this book, for their willingness to work on this interdisciplinary book project. We thank the authors for their interesting proposals of the book chapters, their time, efforts and their research results, which makes this volume an interesting complete monograph of the latest research advances and technology development on the next generation distributed and complex emergent intelligent decision systems. We also would like to express our sincere thanks to the

reviewers, who have helped us to ensure the quality of this volume. We gratefully acknowledge their time and valuable remarks and comments.

Our special thanks go to Prof. Janusz Kacprzyk, editor-in-chief of the Springer's Studies in Computational Intelligence Series, Dr. Thomas Ditzinger, Holger Schaepe, and all editorial team of Springer Verlag for their patience, valuable editorial assistance and excellent cooperative collaboration in this book project.

Finally, we would like to send our warmest gratitude message to our friends and families for their patience, love, and support in the preparation of this volume.

We strongly believe that this book ought to serve as a reference for students, researchers, and industry practitioners interested or currently working in the evolving interdisciplinary area of green computing using emergent natural heuristics paradigm.

Fargo, Cracow, and Sydney
January 2012

Samee Ullah Khan
Joanna Kołodziej
Jen Juan Li
Albert Y. Zomaya

Contents

List of Contributors

Ailixier Aikebaier
Seikei University, Japan
e-mail: alisher.akber@computer.org

Ionut Anghel
Technical University of Cluj-Napoca, Romania
e-mail: ionut.anghel@cs.utcluj.ro

Benjamin Belzer
Washington State University, USA
e-mail: belzer@eecs.wsu.edu

Dan Chen
China University of Geosciences Wuhan, China; e-mail: Danjj43@gmail.com

Tudor Cioara
Technical University of Cluj-Napoca, Romania
e-mail: Tudor.CIOARA@cs.utcluj.ro

Ayse Coskun
Boston University, USA
e-mail: acoskun@bu.edu

Tomoya Enokido
Rissho University, Japan
e-mail: eno@ris.ac.jp

Mariagrazia Fugini
Politecnico di Milano, Italy
e-mail: fugini@elet.polimi.it

Amlan Ganguly
Rochester Institute of Technology, USA
e-mail: amlan.ganguly@rit.edu

Takuro Inoue
Seikei University, Japan
e-mail: akuro.burton@gmail.com

Tao Jiang
HLRS, Germany
e-mail: jiang@hlrs.de

Michael Jonas
Arizona State University, USA
e-mail: MJonas@asu.edu

Samee Ullah Khan
Department of Electrical and Computer Engineering,
North Dakota State University, Fargo, ND 58108, USA;
e-mail: samee.khan@ndsu.edu

Alexander Kipp
HLRS, Germany
e-mail: kipp@hlrs.de

Joanna Kołodziej
Institute of Computer Science, Cracow University of Technology, 31-155 Cracow, Poland
e-mail: jkolodziej@uck.pk.edu.pl

Dhireesha Kudithipudi
Rochester Institute of Technology, USA
e-mail: dxkeec@rit.edu

Jia Liu
HLRS, Germany
e-mail: liu@hlrs.de

Francesco Marcelloni
Department of Information Engineering, University of Pisa, Italy
e-mail: f.marcelloni@iet.unipi.it

Daniel Moldovan
Technical University of Cluj-Napoca, Romania
e-mail: Daniel.MOLDOVAN@cs.utcluj.ro

Alireza Nojeh
University of British Columbia, Canada
e-mail: anojeh@ece.ubc.ca

Partha Pande
Washington State University, USA
e-mail: pande@eecs.wsu.edu

Qinru Qiu
Syracuse University, USA
e-mail: qiqiu@syr.edu

Ioan Salomie
Technical University of Cluj-Napoca, Romania
e-mail: Ioan.Salomie@cs.utcluj.ro

Gang Shen
Georgia State Univeristy, USA
e-mail: gshen1@student.gsu.edu

Makoto Takizawa
Seikei University, Japan
e-mail: makoto.takizawa@computer.org

Massimo Vecchio
University of Vigo, Spain
e-mail: massimo@gts.uvigo.es

Georgios Varsamopoulos
Arizona State University, USA
e-mail: Georgios.Varsamopoulos@asu.edu

Lizhe Wang
Center for Earth Observation and Digital Earth; Chinese Academy of Sciences; Beijing; China
e-mail: LZWang@ceode.ac.cn

Yanqing Zhang
Georgia State University, USA
e-mail: yzhang@cs.gsu.edu

Albert Y. Zomaya
Advanced Network Research Group, School of Information Technologies, University of Sydney, Sydney, NSW 2006, Australia;
e-mail: zomaya@it.usyd.edu.au

Chapter 1
Evolutionary Green Computing Solutions for Distributed Cyber Physical Systems

Zahra Abbasi, Michael Jonas, Ayan Banerjee,
Sandeep Gupta, and Georgios Varsamopoulos

Abstract. Distributed Cyber Physical Systems (DCPSs) are networks of computing systems that utilize information from their physical surroundings to provide important services such as smart health, energy efficient cloud computing, and smart grids. Ensuring their *green operation*, which includes energy efficiency, thermal safety, and long term uninterrupted operation increases the scalability and sustainability of these infrastructures. Achieving this goal often requires researchers to harness an understanding of the interactions between the computing equipment and its physical surroundings. Modeling these interactions can be computationally challenging with the resources on hand and the operating requirements of such systems. To overcome these computational difficulties researchers have utilized Evolutionary Algorithms (EAs), which employ a randomized search to find a near optimal solution comparatively quickly and with compelling performance compared to heuristics in many domains. In this chapter we review several EA solutions for green DCPSs. We introduce three representative DCPS examples including Data Centers (DCs), Wireless Sensor Networks (WSNs), and Body Sensor Networks (BSN) and discuss several green computing problems and their EA based solutions.

Zahra Abbasi
Arizona State University
e-mail: Zahra.Abbasi@asu.edu

Michael Jonas
Arizona State University
e-mail: MJonas@asu.edu

Ayan Banerjee
Arizona State University
e-mail: ABanerj3@asu.edu

Sandeep Gupta
Arizona State University
e-mail: Sandeep.Gupta@asu.edu

Georgios Varsamopoulos
Arizona State University
e-mail: Georgios.Varsamopoulos@asu.edu

S.U. Khan et al. (Eds.): Evolutionary Based Solutions for Green Computing, SCI 432, pp. 1–28.
springerlink.com

1.1 Introduction

Green computing generally refers to the efficient use of resources in computing in conjunction with minimizing environmental impact, and maximizing economic viability. Similar to any other resource consumption problem the goal of green computing is to (i) use fewer hazardous materials, (ii) maximize the efficiency of all resource use in computing systems during their lifetime, and (iii) reuse as many resources as possible and to dispose what can not be recycled responsibly. This chapter focuses primarily on the second item above, where evolutionary techniques are used to achieve energy efficiency and safe computing in Distributed Cyber Physical Systems (DCPSs).

Research and industry continue to drive forward green computing paradigms such as making the use of computers as energy-efficient as possible [1–3]. Such solutions can be applied in different levels of computing systems from the low hardware level such as low power electronics to the high software level such as scheduling algorithms. This chapter will discuss green computing techniques that rely solely on workload and equipment management software. Energy aware workload placement in data centers [4–6], energy aware routing in WSNs, and power management and duty cycling are all examples of solutions of this type. The goal is to incorporate effective green computing parameters in decision making for workload assignment and power management of computing nodes in DCPSs.

A distributed computing system is a network of computing nodes that interact with each other in order to achieve a common goal. DCPSs are distributed systems in which computing systems interact with the physical environment based on information from the physical and cyber space. We introduce three representative DCPS examples including Data Centers (DCs), Wireless Sensor Networks (WSNs), and Body Sensor Networks (BSNs). Green computing is important in such systems to increase scalability and sustainability [3,7–10].

Generally resource assignment in computation systems are NP hard discrete combinatorial optimization problems, where finding an optimal solution (what resource should be assigned to what job at what time) among a finite set of discrete feasible options would require a brute force search. [11] However a brute force search is usually infeasible due to either the scale of these systems such as in data centers, the small amount of resources on hand in WSNs, or the performance impact of such an effort. Green DCPS designers must try to find the optimal balance of energy-latency tradeoff, where the system performance is sacrificed to gain energy efficiency. Only in the case where the benefits of green computing such as increased system lifetime, increased system reliability, lower cost of ownership, and improved safety outweigh the cost on system performance are such solutions likely to be used. In the domains we survey we show several examples where the benefits of green DCPS design outweigh the costs and where EA based solutions are the best solution available for the unique problems they face at present.

Finding the balance described above is made even more challenging in DCPSs because of the nonlinear behavior of many physical systems. Generally this

nonlinearity increases the solution time to unacceptable thresholds if not handled elegantly. This adds an additional factor of complexity researchers must contend with before green DCPS design solutions are viable.

Evolutionary Algorithms (EA) are efficient, nature-inspired planning and optimization methods based on the principal of natural evolution and genetics [12]. These techniques are well-known for their ability to find satisfactory solutions within a reasonable amount of time. This makes them especially suited for solving the complex problems of green computing in DCPS. The evolutionary solutions find a near-optimal solution by an evolutionary based randomized search technique that uses the objective function as a metric. Recent literature leverages this capability to provide satisfactory green computing solutions by tackling the aforementioned challenges [1–4,13–15]. The genetic algorithm, differential evolutionary, and multi-objective optimizations are proposed to tackle the energy-latency tradeoff problem in data centers and WSNs [1–4,15,16].

The rest of this chapter is organized as follows. Section 1.2 provides an overview of the DCPS domains surveyed and describes the challenges and benefits of applying green computing to these domains. Section 1.3 describes the general form of EA solutions and the criteria of domains they are best suited for. Section 1.4 describes EA approaches to the DCPS domains introduced and discusses the results. We discuss the proposed EA approaches and conclude future work in Section 1.5.

1.2 Green Computing in DCPS Domains

For a domain to be a DCPS, it must integrate the physical processes in the environment with those of the software [17]. An understanding of the interactions between computational equipment and the physical environment can be leveraged to improve the energy efficient of the system in ways that would be impossible using cyber approach alone. Such interactions are either intentional for the sake of functionality/management, or unintentional which come from the natural interaction between computing system and the physical surrounding. The heat dissipated by computing nodes is an example of unintentional interactions while energy harvesting is an intentional interaction of green computing. However, unintentional interactions are not beyond some measure of control. A cyber physical solution can both be aware of and manage unintentional interactions whereever there is a benefit to doing so.

Generally speaking green computing refers to the efficient use of resources in computing in conjunction with minimizing environmental impact, and maximizing economic viability. In this chapter we survey a number of green computing solutions whose benefits include:

- **Maximizing Energy Efficiency by Workload Management**: In any heterogeneous environment there is the potential for some computing equipment to be

more efficient at computation than other equipment. Alternatively, even in a homogeneous environment if computing equipment has supporting infrastructure then there is the potential for some computing equipment to require less support than others. In both of these cases workload management software can be used to improve energy efficiency as long as other system objectives are satisfied.

- **Maximizing Energy Scavenging and Minimizing Energy Cost for Cost Sustainability**: Green designers want to scavenge green energy sources wherever possible. A cyber physical solution can adapt the computation load of the computing nodes to the available energy profile, or assign jobs among computing nodes proportionally to their available green energy.
- **Minimizing Hazards on Computing Systems and Their Physical Surroundings**: Thermal safety is crucial for any computing system. The computing systems in a data center need to run below their redline temperature or else the hardware can be damaged and SLA violations can occur. A cyber physical solution needs to be aware of the safety requirements of the system. An example of where this comes into play is where schedules can avoid system overheating in data centers.

Achieving the above green computing goals in DCPSs, which are either big energy consumers due to their large scale or subject to limited battery size or both, is desirable. However, due to the complex dynamics of the physical environment as well as multiple computing performance objectives, exact solutions with low computation cost are unlikely to be found. In the rest of this section we introduce a variety of DCPS domains and cover the challenges and benefits of applying green computing solutions to them.

1.2.1 Data Centers

Data centers are prominent examples of high power DCPSs where green computing efforts focus on various forms of workload management. Essentially workload management attempts to perform computation in the data center when and where the cost is the lowest and to conserve as much energy as possible on the supporting infrastructure. There are several factors that can be capitalized on: (i) cooling system efficiency can be improved by managing the heat distribution among servers, (ii) computation efficiency can be improved by using the most energy efficienct equipment in the data center and by scaling the server power consumption mode to the input workload (e.g., DVFS), and (iii) electricity costs can be lowered by utilizing the spatio-temporal variation of electricity costs and renewable energy source availability [1, 5, 15–25]. A green computing solution for data centers can consist of a combination of the aforementioned factors as shown by Fig. 1.1.

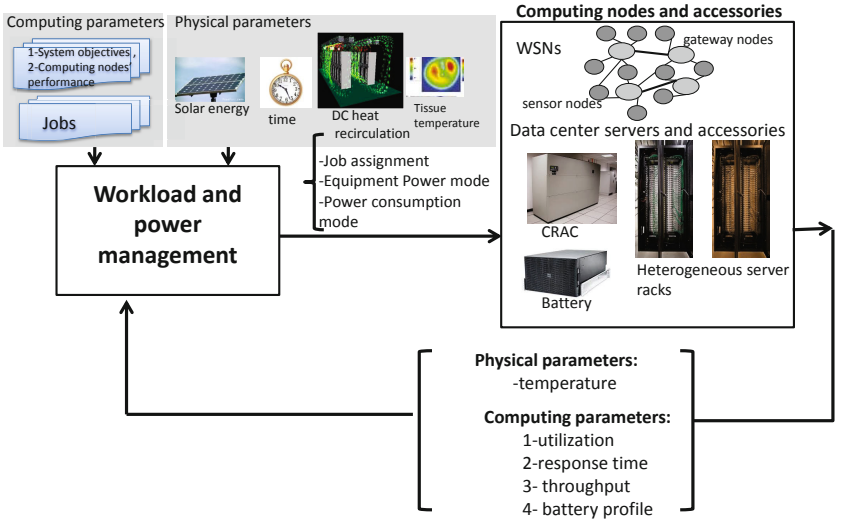


Fig. 1.1. A diagram for cyber physical solution of green computing in data centers and WSNs through workload and power management

1.2.2 WSNs

WSNs are DCPSs that are utilized to monitor the physical environment (e.g., temperature and humidity), send the gathered data to a base station where it is stored, and actuate controlled operations in the physical environment [26]. The main design objective of a WSN is sustainability. Management software is usually designed to be responsive and exploitive of the cyber physical interactions as well as the cyber-related opportunities [3, 9, 17]. WSN software managers can exploit cyber physical interactions in several ways:(i) workload schedulers can exploit the sensors' heterogeneous energy profiles, their accessible communication types, and their job types to increase the lifetime of the system. (ii) green energy harvesting; sensor nodes have limited batteries and the lifetime of the system can be improved by using green energy when and where it is available. [9, 13, 14, 27–29]

1.2.3 BSNs

BSNs are essentially WNSs that are specialized in health data monitoring. As opposed to a focus on sustainability, green computing in BSNs focuses on minimizing hazards caused by the DCPS on its physical surroundings (e.g. people). Management software in BSNs is designed to minimize the temperature impact of the

computing nodes to avoid burning surrounding tissue. BSNs have unique challenges because of this focus and their low computational capacity.

1.2.4 Problems Statements for Green Computing in DCPS

A DCPS can be modeled by the set $S = \{s_1, s_2, \ldots, s_i, \ldots, s_n\}$ of computing nodes, where $|S| = n$, denotes the number of nodes. The computing nodes are targeted to perform a workload consisting of the set J of tasks where $J = \{j_1, j_2, \ldots j_k, \ldots j_m\}$. Examples of jobs in DCPS are batch jobs in high performance computing(HPC) data centers [4], transactional traffic at web data centers [30], and packet routing in WSNs [29].

1.2.4.1 Computing Energy Aware Workload Scheduling in DCPS

Software schemes exist that address green computing in DCPS by taking into account only the cyber behavior while ignoring the physical behavior [1,13,14]. Such schemes usually trade energy for other system performance objectives such as the jobs response time. The performance objectives of a distributed system are either jobs' perceived performance (i.e. latency of a job) or the cumulative performance of all jobs (e.g., the average latency of all jobs and throughput of the system) or both. These objectives are not usually independent. A tradeoff may be required to achieve the desired objectives of each. The multiple objectives in such a system can be represented by a vector $\boldsymbol{o}$ containing all jobs perceived performance objectives $o(J)$, the system's perceived performance objectives $o(S)$, and the system's energy objectives $o(E)$. Green computing in such an environment can be expressed as a multi-objective optimization problem:

$$\text{Objective: optimize } [o(J) \text{ and } o(S) \text{ and } o(E)]. \tag{1.1}$$

An example of such optimization is the joint optimization of energy consumption cost, i.e., $o(E)$, which can be the actual monetary value for the energy cost, performance violation cost, i.e., $o(J)$ which can be the SLA revenue lost (SLA utility cost model), and the system migration cost, i.e., $o(S)$, where the migration cost is uncured for any system state change due to any new decisions.

1.2.4.2 Thermal Aware Workload Scheduling in DCPS

Computing nodes dissipate heat while running tasks. To ensure thermal safety, the computing nodes' temperature should never exceed their safe operating temperature which is either specified by the manufacturer or dependant on the environment in which they are placed. Let T be the temperature vector of nodes. Thermal safety aware scheduling and communication problems deal with distributing the tasks, J,

among n nodes such that thermal safety conditions of all nodes are met [3–5]. This problem is complicated by the usual energy-latency tradeoff in system design and is especially complicated in DCPSs because of the complex, non-Markovian, physical models involved [3, 5]. For example, in data centers, an idle node that has been idle for a long period of time will heat up more slowly than an idle node that only recently stopped running tasks. Thermal safety aware workload placement requires minimizing the worst case (hot spots). [3]. Formally this problem is expressed as follows:

$$\begin{array}{ll} \text{minimize maximize}_i & T_i \\ \text{subject to} & T_i = f(J), \forall i \\ & J \text{ are assigned and performance constrains of } J \text{ is met,} \end{array} \tag{1.2}$$

where f is a function that relates the temperature of each node to the job assignment of the node itself and all other nodes. The question is how to assign jobs to each node. Algorithmically, this is similar to Min-Max Resource Allocation problem (MMRA) [31] which is known to be NP complete. The complexity of the problem comes from the scale of nodes as well as the dependency among nodes for both performance and heat dissipation.

1.2.4.3 Energy Harvesting and Cost Management

Energy costs can denote either actual monetary value per unit of energy or the type of energy i.e., green or brown (green energy denotes renewable energy such as solar whereas the brown energy denotes other types of energy such as nuclear). The goal of energy cost management is to provide sustainable computing by consuming low-cost energy. To achieve sustainability, a cost aware power management technique is required to balance the DCPS nodes' load and performance by matching the spatio-temporal energy consumption of the DCPS to the corresponding available low-cost energy profile. The general solution for sustainability can be stated as follows: (i) buffer energy in batteries whenever the low cost energy is available and drain energy out of batteries when it is not, (ii) duty cycle computation nodes to adapt the energy consumption to the available energy profile, and (iii) migrate jobs to the nodes where the low-cost energy is available. The combination of these operations can govern the energy usage of a DCPS to minimize the energy cost. However, in practice, the solution should address the following challenges: (i) uncertainty associated with the availability of low cost energy, (ii) overhead and constraints of job migration, (iii) violation of jobs' performance requirements from duty cycling, and (iv) workload spikes of an unpredictable nature which prevents optimal duty cycling.

To formally define the problem, let the quadruple (s_i, j_k, v, t) be the job assignment vector which states the job j_k at time t is assigned to the node s_i which run under v power consumption mode or duty cycling policy. The energy cost optimization problem can be modeled as a discrete time system, where at each time t, the problem determines the job assignment and power consumption source of the computing nodes. Low-cost energy and workload in most cases exhibits seasonal

behavior (e.g., cyclic Internet workload behavior), hence the cost optimization problem can be managed in time window T_W, where the availability of green energy and workload is predictable as follows:

$$\begin{array}{ll} \text{minimize} & \sum_{t=1}^{T_W} \sum_{i=1}^{n} \sum k = 1^{m} EnergyCost((s_i, j_k, v, t), B_i, p_i^{AC}) \\ \text{subject to} & \text{low-cost energy profile} \\ & \text{Job assignment constraints} \\ & \text{Job performance constraints} \\ & \text{Job migration constraints} \\ & \text{Nodes available power management modes/duty cycling,} \end{array} \tag{1.3}$$

where the *EnergyCost* is a function that maps a job assignment vector to the energy cost according to the available low-cost energy profile at each node (denoted by B_i) and other energy source cost denoted by p_i^{AC}. The complexity of the problem is two fold: (i) the nonlinear nature of the power management function or duty cycling function (e.g., discrete voltage scale of CPU), and (ii) the nonlinear power consumption cost. The power consumption cost of a computing node consists of both idle power and dynamic power. Dynamic power is a function of job assignments while idle power is independent of job assignment. The complexity of the problem is exacerbated when the stochastic nature of low-cost energy is taken into account.

An examples of such a problem in data centers is workload and server management across data centers [6, 30, 32–34] where the objective is to distribute workload across data centers according to their current electricity cost and type (green/brown). In WSNs managing energy harvesting for sustainability is an example of this problem [10].

1.3 Overview on EA

There are various types of evolutionary algorithms in which the common underlying idea is as follows: given a population of individuals, a *guided random selection* causes natural selection (i.e., survival of the fittest) and this causes a rise in the fitness of population to achieve the desired goal [12]. The EA is targeted to optimize a a metric function. For that, the algorithm creates a set of candidate solutions in a randomized way and applies the metric function as a fitness measure. Based on this fitness, some of the better candidates are chosen to seed the next generation that are produced by recombination/or mutation of the current generation. This process is iterated until a solution with desirable quality is found or a computational limit is reached. Such a process is very effective in solving complex optimization problems due to following reasons [12]: First, the use of a population of solutions helps the EA avoid becoming ”trapped” at a local optimum, when an even better optimum may be found outside the vicinity of the current solution. Second, the complexity of the objective function does not increase the complexity of the process, since the model does not need to know about the nature of problems and feasible solutions.

The main drawback of EA is that it has no concept of an ”optimal solution”. A solution is ”better” only in comparison to the ”currently known” solutions. For this reason, EAs are employed on problems where it is difficult or impossible to test for optimality. There are various classes of EA solutions. We review the following three classes of EA which are utilized in green computing problems.

- Genetic Algorithms (GA): GA models genetic evolution and consists of the following main processes [35]: (i) initialization where the initial population are either generated randomly or seeded toward areas where optimal solutions are likely to be found, (ii) selection where some of the population is selected to generate next generation. Individual solutions are usually evaluated through a fitness function, (iii) reproduction where the next generation of population is produced through genetic operators: crossover (also called recombination), and/or mutation. For each new solution to be produced, i.e., child, a pair or several of ”parent” solutions are selected. The above methods of crossover and mutation, are then used to create new solutions which typically share many of the characteristics of their ”parents”, finally (v) termination: the generational process (e.g., selection and reproduction procedure) is repeated until a termination condition, e.g., a time limit or sufficient fitness has been reached.
- Differential Evolution (DE): DE is used for multidimensional real-valued functions. It is identical to GA except for the reproduction mechanism. The core idea of DE is to adapt the search step inherently along the evolutionary process in a manner that trades off exploitation and exploration [36].
- Multi objective genetic algorithm (MOGA): While mono-objective optimization problems have a unique optimal solution, a multi-objective problem usually has a set of solutions known as the Pareto optimal set. The image of this set in the objective space is denoted as the Pareto front as shown in Fig. 1.2. MOGA adapts evaluation and selection operators of GA to allow the convergence of the evolution process to the best Pareto front and to maintain some diversity of the potential solutions [1,37]. A similar strategy is required in multi-objective differential evolution.
- Swarm Intelligence (SI): SI is a population based stochastic optimization technique, inspired by social behavior of certain insect species [38]. It leverages the complex and often intelligent behavior through the complex interaction of thousands of autonomous swarm members. Such interactions are based on primitive instincts which are performed in communication through the environment. An example is the pheromone through which ants communicate. The pheromones is a chemical substances that can be sensed by ants as they travel along trails. Ants are attracted by pheromones and tend to go after trails with high pheromone concentrations. The pheromone concentrations increases when the attracted ants lay more of the same on the same trail. Ant Colony Optimization (ACO) is a class of SI that works by simulating this pheromone-depositing and pheromone-following behavior of ants [39]. Given this, the functioning of an ACO algorithm can be summarized as follows. A set of computational concurrent and asynchronous agents (a colony of ants) moves through states of the problem corresponding to intermediate solutions of the problem to solve. Their move is based

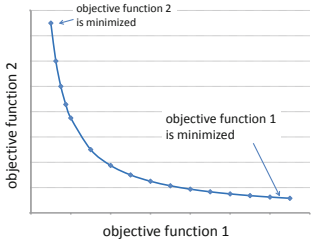


Fig. 1.2. A sample Pareto front graph

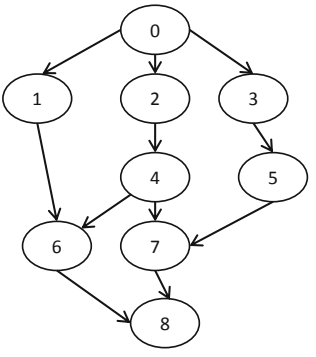


Fig. 1.3. A DAG representing a set of dependent tasks

on a stochastic local decision policy that come from two parameters, trails and attractiveness (visibility). The value of trail is iteratively updated when an ant completes a solution which directs the move of the future ants.

1.4 EA Applications for Green Computing in DCPS

All the aforementioned problems are intractable combinatorial optimization problems, for which the optimal solution cannot be achieved in a computationally efficient way. For this reason, several literatures propose to use heuristic and metaheuristic solutions including EA based solutions to solve them. In this section we review case studies that show how EA based solutions are utilized to solve green computing related problems in data centers and WSNs. Theoretical performance bounds for EA solutions are rarely found in practice. Before we can discuss the performance of the proposed schemes we first need to give a brief overview of heuristic solutions. We then discuss EA based solutions in more detail in terms of (i) how the green computing problem is modeled using existing EA schemes, and (ii) how the performance of EA solutions compares to heuristic solutions.

1.4.1 Survey on Evolutionary-Based Solutions for Energy Aware Workload Scheduling in High Performance Computing (HPC) Data Centers

HPC data centers are oriented toward parallel computation-intensive applications, e.g. simulations or large data set processing, that require many processors and may run for long periods of time. The job scheduling of HPC data centers is defined as

the process of allocating the set J of m tasks to the set S of n machines where each processor is allowed to operate at different voltage levels. The set job J may denote either dependent or independent jobs. The latter may enforce a precedence to the jobs execution with respect the dependency (see Fig. 1.3). The objective is either minimization of power or obtaining a desired tradeoff among all systems and jobs objectives (e.g., makspan, QoS, and power).

In order to manage energy, the research community incorporates DVFS into task scheduling. Some examples of such efforts are as follows: Zahng et al. formulates the problem and shows its NP-hardness proof [40]. Chen et al. model scientific applications with Directed Acyclic Graphs (DAG) (see Fig. 1.3) and propose to reduce energy consumption by leveling down the processor supply voltage during execution of non-critical tasks [20]. Kimura et al. and Wang et al. use the same idea of extending task execution time by using slack times for non-critical jobs [21] and [22]. Freeh et al. and Lim et al. propose to apply DVFS opportunistically during the communication phases of high performance computing such as MPI [23, 24]. Gruian et al. proposes a list based low energy scheduling algorithm which smartly enhances task-graphs (ETG) and energy gain in list based scheduling [25]. Finally, Lee et al. studies energy aware scheduling for parallel independent tasks on heterogeneous computing systems for which an energy-conscious scheduling (ECS) heuristic is proposed [41]. The algorithm uses a "Relative Superiority (RA)" metric that measures the relative energy efficiency between every two task assignments and their corresponding processors and voltage levels. At first the algorithm looks for feasible task assignments and execution orders and then uses the RA metric to choose the most energy efficient task assignment from the feasible solutions.

Consider the following three parameters for task scheduling: the task j, the processor p, and the voltage v. The RA of (j, p, v) and (j, p', v') is calculated as follows:

$$RA(j,p,v,p',v') = -\left[\left[\frac{E(j,p,v)-E(j,p',v')}{E(j,p,v)}\right] + \left[\frac{EFT(j,p,v)-EFT(j,p',v')}{EFT(j,p,v)-min(EST(j,p,v),EST(j,p',v'))}\right]\right] \tag{1.4}$$

$E(j, p, v)$ denotes the total energy consumption of executing job j at the processor p and the voltage v. Also $EFT(j, p, v)$, and $EST(j, p, v)$ denote the earliest finish time and the earliest start time of the job j at the given processor and voltage respectively.

There has been some work that uses EA based solutions such as GA [16], ACO [42], and multi-objective GA [1] to solve the energy aware task scheduling problem. A brief description of the aforementioned EA based solutions and their performance evaluation is given below.

Mezmaz et al. [1] enhance their previous heuristic scheme ECS [41] by creating a hybrid of GA and ECS [41] which is described below. GA is used to provide feasible task scheduling, and ECS is used to provide energy aware task assignment. A chromosome, i.e., solution, is composed of N genes in which each gene is a scheduling solution for a task, a processor, and a voltage scaling on the processor, e.g., (j, p, v). The output of GA is the Pareto optimal set of tasks in which the precedence constraints are respected. In other words, GA builds the task part of a solution. The

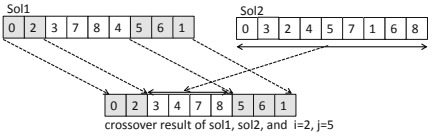


Fig. 1.4. Crossover operation of the GA based scheduling algorithm as proposed in [1]

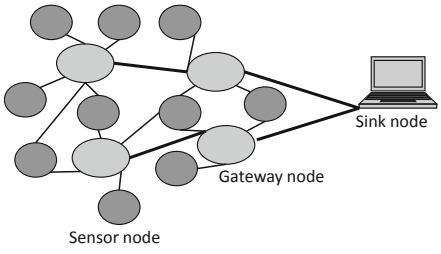


Fig. 1.5. Gateway (cluster) based WSN

crossover operator in the proposed GA uses two solutions to generate a new solution. Assume two task ordering solutions, sol_1 and sol_2. The crossover operator generates sol'_1 and sol'_2 by mixing the original solutions such that precedence is satisfied. For example to generate sol'_1, it selects two random integers x and y, such that $0 \leq x \leq y \leq |J| = m$, and all tasks of sol_1 before x and after y are copied into sol'_1 in order. Then all tasks of sol_2 that are not yet in sol'_1 is copied to the positions of sol'_1 located between x and y in order (see Fig. 1.4). When the partial solutions are ready ECS [41] is called to calculate the RA metric of the given task over all processors and voltage scaling. ECS builds the remaining processor and voltage parts of the partial solutions (i.e.,p and v) provided by the mutation and crossover operators of the GA. The fitness operator of the GA is called once the task processor and voltage parts of each gene of the solution are known. The role of this operator is to calculate the energy consumption and makespan of each solution. Additionally, the authors propose a heuristic, multi-star, which uses parallelization to reduce the execution time of the proposed GA scheme.

The proposed scheme has been implemented using ParadisEO [43]. This software platform provides tools for the design of parallel meta-heuristics for multi-objective optimization. Experiments have been performed on a grid of three clusters containing 714 cores in total. The approach has been evaluated with the Fast Fourier Transformation task graph which is a real-world application. Experiments show the proposed GA based meta-heuristic reduces energy consumption by 47.5% and the completion time by 12% compared to another meta heuristic based scheduling solution [44]. Furthermore, the multi-star approach is on average 13 times faster than the previously proposed GA parallelization approach [45].

Shen et al. also uses GA to solve the energy aware task scheduling problem by using DVFS on the set of independent jobs [16]. the authors proposed Shadowed GA (SGA) to overcome GA's low search speed,. SGA uses a two-measurement system: fitness values (such as the total energy consumption) are used to evaluate overall solutions (chromosomes) and shadow prices are used to evaluate the components of a solution (genes). In this way, shadow prices relatively measure the solutions' fitness value with each change of a component. The intuition behind using shadow prices is make GA operations more intelligent. GA can use shadow prices to directly

compare components, their relationships, and their contribution toward a better solution. For energy aware task scheduling, the shadow price is defined as the average energy consumption per instruction for each processor. SGA adds one more operation to the operations provided by the classic GA mutation and crossover operation. For example to select a sub population, one of the following operations is randomly performed:

- Classic mutation operation (Move). Randomly select two processors and move one randomly selected task from one processor to the other.
- Classic crossover operation. Exchange two randomly selected tasks between two randomly selected processors.
- Shadow priced guided mutation operation. Mutate and exchange tasks according to the shadowed prices.

Using the above operation the evolution will reduce the processor's shadow price by moving tasks among them. The authors show that SGA saves more energy than GA and can find solutions in shorter time than GA in their simulation study.

Chen et al. proposed an Ant Colony System (ACS) algorithm for a large-scale workflow scheduling problem in computational grids [42]. The scheduling algorithm is designed for workflow applications in which multiple user-specified performance parameters are desired such as reliability, time, and cost. The cost parameter can be interpreted as energy cost; therefore this work can be utilized for the energy cost management. The authors propose some heuristics to optimize the performance goals and an adaptive scheme is designed to enable artificial ants to select heuristics based on pheromone values. The performance of the algorithm is compared with deadline-MDP [46] which can only tackle the cost optimization task with deadline constraints. The results show the ACS algorithm manages to decrease the cost by 10–20% compared to the Deadline-MDP approach.

1.4.1.1 Summary of Results

These studies show that EA based solutions can yield higher performance than heuristic solutions [1,42]. Literature indicates that EA performance can be significantly improved by parallelization and by incorporating application specific heuristics in modeling [1,16]. The results show that EA based solutions are applicable to the scheduling domain. However to decide on their application in practice, a comprehensive study comparing heuristic solutions to EA solutions and analyzing the cost-performance tradeoffs involved would be useful. Current research lacks this evaluation.

1.4.2 Survey on Energy Efficient Routing Problem for WSNs

Consider a set S of connected sensor nodes randomly distributed in a region where each node has a limited battery life used primarily for transmitting data. Each node

collects data which needs to be delivered to the gateway nodes which have more processing power and larger transmission ranges [29]. Energy efficient routing in such systems is usually designed to maximize the lifetime of the system. Note that the minimum energy solution is not the same as the maximum-lifetime solution because if all traffic is routed through the most energy efficient path the nodes in that path will be exhausted quickly. In WSNs it is more important to route traffic such that the energy consumption is balanced among the nodes in proportion to their available energy than to minimize the total energy consumption. Therefore, the general problem statement for the energy efficient routing problem is as follows: Given a set of feasible routes, for every node, find the routes and the frequency that each route should be used to maximize the lifetime of the nodes. Mathematically, such routing problem are formalized as integer programming [29] and are NP-complete. WSN nodes need to make quick decisions to ensure low overhead. For that reason, heuristics and meta-heuristics are proposed to find routes. A survey of meta-heuristic solutions for the WSN routing problem is also given in [26].

Some examples of heuristic solutions are as follows. Heinzelman et al. proposes Low Energy Adaptive Clustering Hierarchy (LEACH) algorithm which is probably one of the most referenced protocols in the sensor networks area [9]. The algorithm evenly balances the energy load among the sensor nodes by using a randomized rotation of local cluster base stations. Power-Efficient GAthering inSensor Information Systems (PEGASIS) [27] is another heuristic scheme that enhances the energy saving benefit of LEACH by enforcing that each node communicates only with a close neighbor and takes turns transmitting to the base station. Shah and Rabaey [28] propose using a set of sub-optimal paths occasionally to increase the lifetime of the network. These paths are chosen by means of a probability function which depends on the energy consumption of each path. Gupta et al. propose an algorithm for making a network of sensors into clusters in such a way that less energy constrained gateway nodes act as clusterheads [47]. The load is then balanced among these gateways.

There are also some EA based solutions for the energy efficient routing problem in the literature including ACO [48–50], DE [2], GA [13,14].

Bashya et al. [48] propose an ant inspired Collaborative Routing Algorithm for Wireless Sensor Network Longevity (CRAWL) that is shown to have high performance when nodes have non-uniform energy supply.

Camil et al. [49] propose the Energy Efficient Ant Based Routing (EEABR) algorithm that uses artificial ants to find routing paths between the sensor nodes and the sink nodes, which are optimized in terms of distance and energy levels. To this end, the authors define the visibility function as the inverse of the total energy consumption of each node, and the trail pheromone as the traffic rate in the routing table at each node. The trail pheromone is calculated to denote both the energy levels and the length of the paths. Therefore, the selection probability of nodes is determined by the tradeoff between ACO visibility which is our desire to use nodes with more energy and actual trail intensity which is our desire to minimize transmission energy. This scheme is compared to authors' other ACO routing scheme that disregards the visibility above, and the results show that EEABR's performance is superior.

Hang et al. take a different approach in utilizing ACO to achieve energy efficiency in the WSNs routing problem [50]. The paper adopts the time series model, ARMA, to analyze dynamic tendencies in data traffic and to deduce the construction of the load factor, which can help to reveal the future energy status of sensors in WSNs. The load factor is then used to guide the pheromone update rule such that artificial ants foresee the local energy state of networks and then corresponding actions are adaptively taken to enhance the energy efficiency in routing construction. The simulation results show that the proposed ACO scheme can indeed balance the total energy cost for data transmission.

The routing problem for WSNs where all nodes are capable of reaching each other is also studied in [2], where the authors utilize the Multi-Objective Differential Evolution (MODE) approach to reduce energy as well as transmission delay in WSNs. The proposed scheme is based on nonlinear power attenuation in which the signal power fails by a factor of $r^{-\alpha}$, where r denotes the distance from the sender and α denotes the power attenuation factor. Due to this nonlinear power attenuation, relaying a signal using intermediate nodes may result in lower power consumption than direct communication at the potential cost of latency. Therefore, a set of Pareto optimal routings with respect to multiple objectives (i.e., reducing energy and latency) can be identified, where the multiple candidates represent different possible tradeoffs between energy consumption and communication latency. The authors propose a discrete version of MODE where the decision variable is represented by a multi-dimensional vector variable e.g., $x = [x_1, x_2, \ldots, x_n]$, $x_i \in \chi_i$, where χ_i is a set of discrete values that delimit the possible value that the corresponding x_i can have. Since the differential vector of traditional DEs can no longer be interpreted for the discrete case the probabilistic operator is necessary. This operator includes a greedy, mutation, and perturbation probabilities. To model the routing problem the authors define a chromosome as a path from a source node to a destination node. This path is represented as a sequence of network node IDs with the source node in the first locus and the destination node in the last locus. The length of chromosomes varies depending upon the source node, the destination node, and the path. In order to facilitate the evolutionary search, an expansion operator is introduced to systematically generate the initial population and implement the reproduction operation which utilizes a set of heuristic rules based on power relationships among related nodes. Through a simulation study the article shows the ability of multi-objective DE to come up with a Pareto front of multiple routes between a given source and destination.

Authors in [13] apply GA to maximize network longevity in terms of time to first node death. The objective is to generate a transmission schedule which consists of a collection of transmission rounds. A transmission schedule denotes how data is collected from each sensor and propagated to the base station. It represents a collection of routing paths that the network will follow to maximize its lifetime. Every routing path is a tree that is called the aggregation tree. The most efficient path is not the best answer, since the continually using this path would cause some nodes to die early than others. Therefore, the author finds a collection of aggregation trees, where each tree is used for a fixed number of rounds, so that the energy consumption

is balanced among all nodes in the network. Solutions are a set of aggregation trees and their frequency. Therefore, the set of solutions is given to the GA. The transmission schedule is the frequencies of all aggregation trees representing a chromosome and each gene corresponds to a specific aggregation tree frequency. The frequency of an aggregation tree is represented as bits of their binary values. Finally, the fitness function is designed to increase the lifetime of the system. The simulation results show that the GA algorithm finds the near optimal solutions faster than the linear programing model of the problem.

Energy efficient routing for WSNs is also studied in [14] where the authors propose two tier routing. They assume two tier workload gathering in a sensor network, where there are some higher power relay nodes which may form a network among themselves to route data towards the base station. Then they propose a GA based solution for scheduling the data gathering of relay nodes. The problem is modeled such that each chromosome in the initial population corresponds to a valid routing scheme represented by relay node numbers. The fitness value for each individual is computed as the initial power of every relay node over the maximum energy dissipated over all relay nodes for the routing scheme defined by the chromosome. Selection of individuals is carried out using the Roulette-Wheel selection method and a uniform crossover operation. Mutation is performed such that the nodes that dissipate the maximum energy due to transferring data are selected as critical nodes. Then routes that use the critical nodes are changed to reduce their load. The alternative nodes for each critical node lie within transmission range of the source node such that the Euclidean distance between the source and the end destination is greater than the distance between the source node and the alternative node. Through experimental study, the authors show an average life time extension of 200% compared to the network that uses the minimum transmission energy model and minimum hop routing model discussed in [9] and [47], respectively.

1.4.2.1 Summary of Results

Some of the results [13, 14, 49] show that EA based solutions can find better solutions compared to existing heuristic solutions. Also, various EA based solutions are leveraged in literature, but there is no study comparing their relative performances. Also there is no evaluation based study that compares the cost-performance tradeoff of EA based solutions with heuristic solutions.

1.4.3 Survey on Applications of EA for Thermal Aware Job Scheduling in HPC Data Centers

Thermal aware job scheduling for HPC data centers has been proposed in some works to increase their energy efficiency [4, 5, 15, 51, 52]. The main idea is to increase support infrastructure efficiency such as CRAC and to provision workload on the

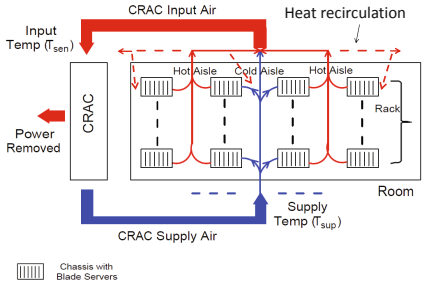


Fig. 1.6. Demonstration of cold-hot aisle server arrangement and heat recirculation within the data center room

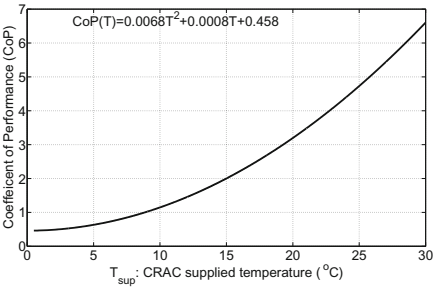


Fig. 1.7. A CoP graph example

most energy efficient servers. Moore *et al.* [5, 51], and Bash and Forman [52] show that thermal aware workload placement can save energy in data centers. Tang *et al.* [15] and Mukherjee *et al.* [4] model the heat recirculation and propose heuristic and GA based solutions for spatio and spatio-temporal thermal aware job scheduling, respectively. This section first gives an overview of the concept of thermal aware scheduling in data centers and then reviews related heuristics and GA based solutions proposed in the literature.

1.4.3.1 Thermal Aware Job Scheduling for Data Centers: An Overview

Thermal aware job scheduling in data centers deals with minimization of the heat recirculation among servers. Fig. 1.6 shows the layout of a typical data center. flaws in this layout leads a heat recirculating between servers instead of flowing to the *Computer Room Air Conditioners*CRAC. The heat recirculation among servers comes from the physical layout of contemporary data centers in which computing servers are organized in rows of racks and the rack arrangement is such that either the front or back panels of every two racks in rows are facing each other. In this arrangement, called the hot aisle / cold aisle arrangement, the air flow makes *recirculation* of the hot air from the air outlet of the computing servers into their inlet air. The heat recirculation forces data center operators to operate their *computer room air conditioners* (CRACs) to supply cold air, denoted as T_{sup}, at a much lower temperature than the servers' redline temperature, T_{red}, which is the maximum safe operating temperature of the servers. The amount of heat recirculation depends on the physical layout/airflow of data center room. Servers do not equally contribute to the heat recirculation. The heat recirculation among servers is modeled as the N by N matrix, D, where each element of D, d_{ij}, is the ratio of server j's outlet temperature to the inlet temperature of server i [15]. Let $\boldsymbol{p}$, be the power vector of all servers and T_{in}be the inlet temperature vector of all servers such that:

$$T_{in} = T_{sup} + D\boldsymbol{p}. \tag{1.5}$$

Since workload directly affects the power, workload assignment directly affects the temperature distribution. The CRAC cooling energy can be modeled by its *coefficient of performance* (CoP), which is the ratio of the heat removed (i.e., computing energy) over the energy required to remove that heat (i.e., cooling energy). A higher CoP means more energy efficient cooling, because CoP monotonically increases with T_{sup} (see Fig. 1.7). However, according to Eq. 1.5, the T_{sup} can be *at most* equal to the maximum inlet temperature of servers.

Since servers' heat generation directly depends on their workload, thermal aware job scheduling can be used to minimize heat recirculation among servers, which allows CRAC to operate at a higher temperature, and therefore efficiency.

1.4.3.2 Heuristic and GA Based Solutions for Thermal Aware Job Scheduling in Data Centers

Heuristic solutions for thermal aware job scheduling are proposed to minimize the total heat recirculation (i.e., MinHR, and LRH) [4, 5] and to distribute workload to the servers inversely proportional to the servers' inlet temperature [5, 53].

Tang *et al.* [15] and Mukherjee *et al.* utilize GA for thermal aware job scheduling in HPC data centers and propose XInt-GA, and SCINT respectively [4, 15]. Through a simulation study of realistic job traces, the authors find that GA based solutions(i.e., XInt-GA, and SCINT) have much higher energy-saving benefits than the aforementioned heuristic solutions but also take much longer to complete (a couple of hours compared to a fraction of a second). A brief overview of XInt-GA and SCINT is given below.

XInt-GA is a GA based solution for spatio thermal aware job scheduling in a virtualized and homogeneous data center where all servers are capable of running all jobs at the same speed. In such a data center model, minimizing heat recirculation is a combinatorial min-max optimization problem, which is NP-complete' [31]. Consider a job assignment vector $\mathbf{c}$ where each element, c_i, specifies how many processors of a server i are assigned to the job. The thermal aware job assignment problem can be summarized as follows:

$$\begin{aligned} &\min \max_i \ T_{in,i} \\ &\text{s.t.:} \quad C_{tot} - \textstyle\sum_{i=1}^{N} c_i \\ &\qquad \boldsymbol{T}_{in} = T_{red} - D(\boldsymbol{a} \odot \boldsymbol{c}) + D\boldsymbol{b} \\ &\qquad c_i \leq m_i, \end{aligned} \tag{1.6}$$

where C_{tot} the denotes total number of processor that the jobs require, $\boldsymbol{b}$ denotes the servers' idle power vector, and a denotes the power consumption vector of the servers for each of their processors. In IXint-GA, the initial set of feasible solutions is any job assignment vectors that meets the performance requirement of jobs. A gene is each element of the job assignment vector, and the fitness function is the resulting peak inlet temperature of that job assignment. The new generation is selected

by a probability-based Roulette Wheel selection process that is a. ** *definition or citation for the Routlette Wheel.*

The article [15] comparatively evaluates the performance of the XInt-GA algorithm against two other algorithms called XInt-SQP and MinHR [5] using a simulated small-scale data center. XInt-SQP uses sequential quadratic programming to the solve the minimax formulation of Eq. 1.6 in the real number domain. Which means that the job assignment vector, **c**, should be relaxed to a continuous variable, i.e., more than one job is assigned to a processor. The solution is further discretized to find a close integer and feasible solution. MinHR is a heuristic solutions that minimizes total heat recirculation instead of minimizing the maximum heat recirculation. Results show that XInt-GA algorithm achieves maximum cost saving (up to 30% depending on the data center load) among all other solutions. *Discuss XInt-SQP*

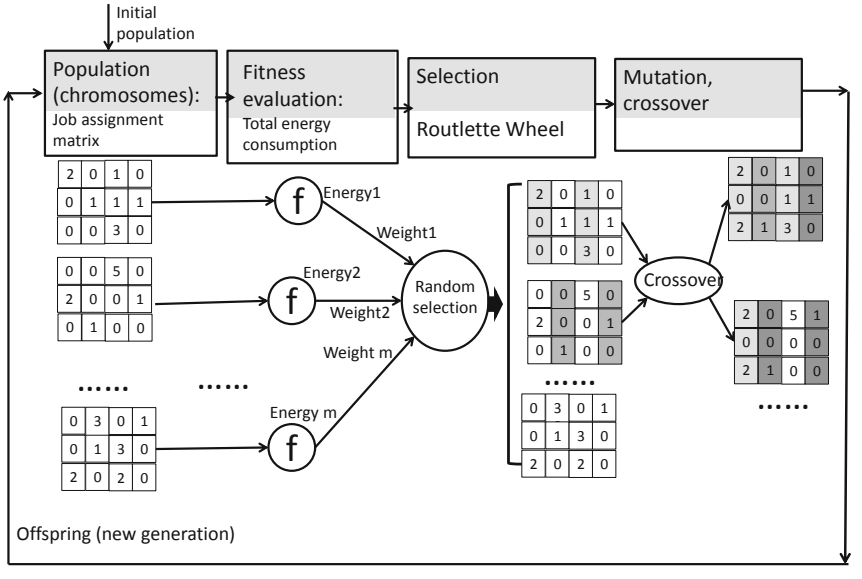


Fig. 1.8. The diagram for the XInt-GA algorithm as proposed in [15]

Mukharejee et al. [4] introduce Spatio-Temporal thermal aware job scheduling, an alternative to [15] introduced above, which decides both the job execution start time and the job to server assignments. By leveraging slack in the execution schedule of the data center, jobs can be postponed as much as their deadlines allow to make a smoother schedule in which the most energy efficient equipment is utilized as much as possible. This can be combined with an understanding of the heterogeneity of equipment to schedule jobs on the most energy efficient equipment. Intuitively, certain equipment is more energy efficient per computation than other equipment in

any heterogeneous environment. By using the most efficient equipment as much as possible energy is saved both by spending less per computation and by cooling less because all energy spent emerges as heat and thus would need to be cooled. Combining these factors synergistically provides a form of scheduling with a super-linear energy saving effect.

Consider a discrete time slot system for scheduling. In such a system, the optimal schedule is only possible to achieve when we have perfect knowledge of job runtimes. In practice, because the actual job runtime is only known when the job is completed, an optimal schedule is impossible to achieve. For example, if in our initial job schedule the most efficient equipment in the data center is fully utilized and then a job on that equipment completes before our expectations, it may not be possible to fill in the new slack time on this equipment optimally while preserving SLA's. Had we had a perfect job runtime estimate in advance we could have created a different schedule in which a job that was executed in parallel may have been able to be postponed to fill in this slack time in a more efficient way.

Formally this problem can be defined to minimize the total energy consumption as follows:

$$\begin{aligned} \text{min: } & \sum_{t=1}^{T}\left(\sum_{i=1}^{N}(\boldsymbol{a}\odot\boldsymbol{c}_{i,t})+\boldsymbol{b}\right)+\frac{\sum_{i=1}^{N}(\boldsymbol{a}\odot\boldsymbol{c}_{i,t})+\boldsymbol{b})}{CoP(T_{red}-max_i D(\boldsymbol{a}\odot\boldsymbol{c}_{i,t}+\boldsymbol{b}))}\right)\tau \\ \text{s.t.: } & \sum_{k=1^K} c_{i,k,t} = c_{i,t} \le m_i \ \forall i \le N, \text{ and, } t \le T \text{ [capacity constraint]} \\ & \sum_{i=1}^{N} c_{i,k,t} \ge C_{tot,k} \text{ [performance constraint]} \\ & \text{[jobs' deadline constraint]} \\ & \text{[jobs' arrival time constraint]}, \end{aligned} \tag{1.7}$$

where k denotes the job index, K denotes total number of jobs, and τ denotes the time interval for decision making. The first and second terms in the objective function denote the computing energy and cooling energy, respectively. The deadline and arrival time constraints are omitted above for the sake of brevity. Because this problem is NP-hard, the offline optimal solution cannot be solved in polynomial time. Therefore, a GA based solution, SCheduling to minimize thermal cross-INTerference (SCINT), is proposed to heuristically solve the problem offline. In this way, the initial population is seeded from the basic feasible job scheduler (e.g., FIFO). The fitness function is the total energy consumption. The crossover (mating) and mutation are performed to produce new solutions and randomize genes within an individual, respectively. The new generation is selected from the current pool of solutions using Roulette Wheel selection algorithm.

Mukharejee et al [4] provide results through a simulated data center of ASU's HPCI data center whose heat recirculation matrix is obtained by a CFD simulation software. The performance of SCINT is compared to other heuristic solutions including online EDF-LRH (Earliest Deadline First Least Recirculated Heat) using job submission logs from the data center above. EDF-LRH uses heuristic and rank servers statically by using LRH metric to minimize total heat recirculation. LRH ranks servers according to their contribution on the heat recirculation statically when they are assumed to be fully utilized. Jobs are assigned to the available servers whose LRH rank is the lowest. Results show that SCINT saves up to 40% more

energy than naive non-energy aware job scheduling (i.e., FCFS) and 23% more energy than EDF-LRH. However, SCINT is an offline algorithm that requires up to several hours to process compared to the several milliseconds required by EDF-LRH.

1.4.3.3 Summary of Results

Mukharejee et al [4] clearly show the cost-performance tradeoff for the GA based solution compared to the heuristic solutions. The paper also shows the GA application on evaluating heuristic solutions. In other words, since exact optimal solution due to nonlinearity of objective function, discrete variables, and NP-hardness of the problem cannot be found without a brute-force search , GA based solution is leveraged to evaluate the heuristic solutions. Results on both studies [4, 15] indicate that GA based solution yield significant energy-saving benefit in the high running duration cost. This is a motivation for the future research in this area to find out fast GA based solutions

1.4.4 Survey on Thermal Aware Communication Scheduling in Implanted Biosensor Networks

For some types of WSNs such as BSNs, thermal safety is very important. In BSNs, communication in implanted biosensor networks which are used for health monitoring can be organized into clusters where most of the communication takes place. However, clusters that run for a long time can generate enough heat to damage surrounding tissue. Tang et al. [3] propose a set of heuristic solutions and a GA based solution to ensure the thermal safety of tissues where biosensor networks are implanted. The authors propose a cluster leader rotation to prevent potential thermal tissue overheating. The thermal effects of implanted biosensors are modeled by calculating Specific Absorption Rate (SAR) and discritizing the temperature increase over space and time by Finite Difference Time Domain (FDTD). The model captures the heat interferences among nodes. In summary, the problem is defined as: given a control volume, the known location of implanted sensors, and the related properties, how does one schedule the rotation sequence to minimize temperature increases. To model the temperature dynamics of biosensors, the paper considers that temperature increase of tissues is affected by three major sources as follows:

- Heating caused by RF powering. The authors assume sensor nodes are recharged by an external RF power source and this recharging process will heat surrounding tissue.
- Radiation from the sensor node's antenna. The tissue surrounding the antenna partially absorbs the transmission energy of the antenna.

- Power dissipation by the sensor node circuitry. As computation is performed, heat is dissipated.

Given the above heat sources, the authors model temperature over space and time. FDTD is used to discretize the temperature variation equation over space (using small 2-D cells) and time, such that the temperature of each cell is a function of its past temperature as well as the past temperature of its neighbors i.e., the temperature at point (i, j) at time $m+1$ is a function of the temperature at points (i, j), $(i+1, j)$, $(i, j+1)$, $(i-1, j)$, and $(i, j-1)$ at time m.

The FDTD based model above is computationally expensive. To address this, the authors propose a simplified model that estimates the maximum possible temperature for each cluster leader rotation. This increase is the Temperature Increase Potential (TIP) metric. In this model the temperature increase rate of a node is a function of its euclidean distance from the current leader and the time since it was previously the leader. The FDTD numerical result is used to train the simplified model. Then the TIP metric is used to calculate the summation of influence of node i's leadership on all other nodes. Given the TIP metric, finding a rotation of leaders that minimizes the maximum temperature is still a NP-hard problem similar to the Traveling Sales Man (TSP) problem. Polynomial time solutions to this problem do not exist. Therefore, the authors propose a GA based method to find a near optimal rotation as follows:

The chromosome of each individual is the rotation sequence. The TIP metric of each individual is used as the fitness measure. Therefore, a thermally safer rotation is more likely to survive in the GA evolution process. The initial sequences are chosen randomly from a poll of the feasible solutions. A random crossover is used in which individuals are randomly selected. Chromosomes are swapped to generate two new individuals for the next generation. For the mutation process, an individual with low probability (low fitness values) is selected and two genes in the rotation are randomly selected and swapped to create a new rotation. The new population consists of a weighted random selection of the previous individuals and the newly generated children where the individuals with higher fitness values are more likely to survive. Authors provide no evaluation of the GA algorithm compared to any heuristic.

1.4.5 Survey on Energy Harvesting and Cost Management in Data Centers

The electricity price across data centers varies over time and location. Also many data centers leverage built-in renewable energy sources to contribute in the green computing [6]. Recently research community propose to leverage the aforementioned spatio-temporal variation of electricity price, and energy supply type(i.e., green/brown) by workload and server management across data centers for web type jobs [6, 30, 32–34, 54], as well as batch type job [55]. However due to nonlinearity

of energy cost model which depends on both idle power of servers and dynamic power [30, 32, 33], job migration overhead [33, 55] and the jobs delay requirement which prevent them to run at particular data centers [30, 33, 34], the online form of the problem is found to be integer programming, which is generally NP-hard. For that the literatures propose heuristics and meta-heuristics to solve it.

Qureshi et al. [34] use heuristics to quantify the potential economic gain of considering electricity price at the location of computation. Authors in [30] approximate the across data center workload management problem with min-cost flow model. The authors further improve their scheme by proposing a joint optimization of power control and electricity price in [54]. They develop a nonlinear optimization problem which control voltage scaling of servers as as well as number of active servers and solve it by the general benders decomposition method. [33, 55] develop online greedy solutions to tackle the job migration overhead in the problem. Finally Le et al [32], use the simulated annealing method to solve the across data center server and workload management problem.

1.4.5.1 Summary of Results

The literature identify the opportunity to increase energy efficiency of data centers through energy cost management across data centers. They model the problem and report the significant cost saving of this problem using realistic data. However efficient solution to the problem has not been developed yet. Proposed heuristics and meta-heuristic solution has not been compared in term of cost-saving performance and time efficiency/computation complexity. EA would potentially help the performance of the problem solution that can be considered as future work.

1.5 Conclusion

This chapter gives an overview on green computing problem in DCPS that are solved using evolutionary techniques. a taxonomy of the researches in this area is given in Table. 1.1. Green computing solutions for DCPS can be achieved through energy and thermal aware workload scheduling among nodes. However, such problems turn out to be NP-hard, where the optimal solution cannot be calculated in a time efficient way. Therefore, literatures develop heuristic and meta-heuristic solutions using EA. Generally, EA based solutions are shown to have higher performance than their heuristic counterpart, but they may take longer time to complete.

The current study shows that research community leverage successfully EA based solutions to model thermal and energy aware workload distribution in DCPS. However, the study also shows that much works is left for future work in this area. The theoretical guarantee does not usually exist for the performance of EA based

Table 1.1. A taxonomy of researches on the evolutionary techniques application to the green computing in DCPS

domain	distributed system	problem	articles	scheme
cyber space	data center	workload scheduling	[1,16,42]	meta-heuristic (i.e.,GA,ACO)
	data center	workload scheduling	[20–25,40,41]	heuristic
	WSN	routing problem	[2,13,14,48–50]	meta-heuristic (GA,ED,ACO)
	WSN	routing problem	[9,27,28,47]	heuristic
cyber physical	data center	thermal aware workload sched.	[4,15]	meta-heuristic (GA)
	data center	thermal aware workload sched.	[4,5,53]	heuristic
	WSN	thermal aware workload sched. and comm.	[3]	meta-heuristic (GA)
	data center	energy cost management across data centers	[32]	meta-heuristic (simulated annealing)
	data center	energy cost management across data centers	[30,33,34,55]	heuristic

solutions. Hence, to ensure the performance of EA schemes in practice, a comprehensive simulation/experimental study is required which examine the performance of solutions over various type of input data. Most of proposed solutions are not properly evaluated. In other words, many of the literature provide results according to a limited input data. Many of them lack from a comprehensive comparison study which exactly show how much better performance the EA based solutions achieve for what value of cost compared to heuristic solutions or other EA solutions. Various types of EA solutions are utilized for the green computing problems in DCPS, but there is no study that compare the performance of them (in terms of the optimality of solution, and the computation time efficiency of schemes with respect to each other. To sum up, the reviewed studies show that green computing problem can be successfully modeled with EA approaches, but the results are far from approaching the proper utilization of the opportunities that EA provide. Further, many studies show that the performance of EA solutions can be improved significantly by incorporating the application-specific heuristics in the EA decision making process. Future study should emphasize on such schemes to achieve more time efficient and higher performance EA based solutions than existing ones.

Recently energy harvesting and energy cost management in DCPS is proposed by some literature. These problems are very complex to tackle the uncertainty associated with the green energy source, and nonlinear utility function to adapt the energy consumption to the limited low-cost/green energy profile while maintaining the workload quality of service. The application of EA for such problems is the other open research area.

References

1. Mezmaz, M., Melab, N., Kessaci, Y., Lee, Y., Talbi, E., Zomaya, A., Tuyttens, D.: A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. Journal of Parallel and Distributed Computing (2011)
2. Xue, F., Sanderson, A., Graves, R.: Multi-objective routing in wireless sensor networks with a differential evolution algorithm. In: Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control, ICNSC 2006, pp. 880–885 (2006)
3. Tang, Q., Tummala, N., Gupta, S., Schwiebert, L.: Communication scheduling to minimize thermal effects of implanted biosensor networks in homogeneous tissue. IEEE Transactions on Biomedical Engineering 52(7), 1285–1294 (2005)
4. Mukherjee, T., Banerjee, A., Varsamopoulos, G., Gupta, S.K.S., Rungta, S.: Spatio-temporal thermal-aware job scheduling to minimize energy consumption in virtualized heterogeneous data centers. Computer Networks (June 2009), http://dx.doi.org/10.1016/j.comnet.2009.06.008
5. Moore, J., Chase, J., Ranganathan, P., Sharma, R.: Making scheduling "cool": temperature-aware workload placement in data centers. In: ATEC 2005: Proceedings of the Annual Conference on USENIX Annual Technical Conference, pp. 5–5. USENIX Association, Berkeley (2005)
6. Liu, Z., Lin, M., Wierman, A., Low, S.H., Andrew, L.L.H.: Greening geographical load balancing. In: Proc. ACM SIGMETRICS. ACM, San Jose (2011)

7. Quick start guide to increase data center energy efficiency, General Services Administration (GSA) and the Federal Energy Management Program (FEMP). Tech. Rep. (September 2010)
8. Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E.: A survey on sensor networks. IEEE Communications Magazine 40(8), 102–114 (2002)
9. Heinzelman, W., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, p. 10 (2002)
10. Kansal, A., Hsu, J., Srivastava, M., Raghunathan, V.: Harvesting aware power management for sensor networks. In: Proceedings of the 43rd annual Design Automation Conference, pp. 651–656. ACM (2006)
11. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-completeness. WH Freeman & Co. (1979)
12. Eiben, A., Smith, J.: Introduction to evolutionary computing. Springer (2003)
13. Islam, O., Hussain, S.: An intelligent multi-hop routing for wireless sensor networks. In: 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops, WI-IAT 2006 Workshops, pp. 239–242 (2006)
14. Bari, A., Wazed, S., Jaekel, A., Bandyopadhyay, S.: A genetic algorithm based approach for energy efficient routing in two-tiered sensor networks. Ad Hoc Networks 7(4), 665–676 (2009)
15. Tang, Q., Gupta, S.K.S., Varsamopoulos, G.: Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach. IEEE Trans. Parallel Distrib. Syst. 19(11), 1458–1472 (2008)
16. Shen, G., Zhang, Y.: A shadow price guided genetic algorithm for energy aware task scheduling on cloud computers. In: Advances in Swarm Intelligence, pp. 522–529 (2011)
17. Banerjee, A., Venkatasubramanian, K., Mukherjee, T., Gupta, S.: Ensuring safety, security and sustainability of mission-critical cyber physical systems. In: Proceeding on Special Issue on Cyber-Physical Systems (2011)
18. Gupta, S.K.S., Mukherjee, T., Varsamopoulos, G., Banerjee, A.: Research directions in energy-sustainable cyber-physical systems. Elsevier Comnets Special Issue in Sustainable Computing (SUSCOM), Invited Paper 1(1), 57–74 (2011)
19. Kant, K.: Data center evolution: A tutorial on state of the art, issues, and challenges. Computer Networks 53(17), 2939–2965 (2009)
20. Chen, G., Malkowski, K., Kandemir, M., Raghavan, P.: Reducing power with performance constraints for parallel sparse applications. In: 19th IEEE International on Parallel and Distributed Processing Symposium, 2005. Proceedings, p. 8 (2005)
21. Kimura, H., Sato, M., Hotta, Y., Boku, T., Takahashi, D.: Emprical study on reducing energy of parallel programs using slack reclamation by dvfs in a power-scalable high performance cluster. In: 2006 IEEE International Conference on Cluster Computing, pp. 1–10 (2006)
22. Wang, L., von Laszewski, G., Dayal, J., Wang, F.: Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with dvfs. In: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 368–377 (2010)
23. Freeh, V., Lowenthal, D.: Using multiple energy gears in mpi programs on a power-scalable cluster. In: Proceedings of the tenth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 164–173 (2005)
24. Lim, M., Freeh, V., Lowenthal, D.: Adaptive, transparent frequency and voltage scaling of communication phases in mpi programs. In: Proceedings of the ACM/IEEE, 2006 Conference, SC 2006, pp. 14–14 (2006)
25. Gruian, F., Kuchcinski, K.: Lenes: task scheduling for low-energy systems using variable supply voltage processors. In: Proceedings of the 2001 Asia and South Pacific Design Automation Conference, pp. 449–455 (2001)

26. Kulkarni, V., Forster, A., Venayagamoorthy, G.: Computational intelligence in wireless sensor networks: A survey. IEEE Communications Surveys & Tutorials (99), 1–29 (2011)
27. Lindsey, S., Raghavendra, C.: Pegasis: Power-efficient gathering in sensor information systems. In: IEEE Aerospace Conference Proceedings, vol. 3, pp. 3–1125 (2002)
28. Shah, R., Rabaey, J.: Energy aware routing for low energy ad hoc sensor networks. In: 2002 IEEE Wireless Communications and Networking Conference, WCNC 2002, vol. 1, pp. 350–355 (2002)
29. Chang, J., Tassiulas, L.: Maximum lifetime routing in wireless sensor networks. IEEE/ACM Transactions on Networking 12(4), 609–619 (2004)
30. Rao, L., Liu, X., Xie, L., Liu, W.: Minimizing electricity cost: optimization of distributed internet data centers in a multi-electricity-market environment. In: 2010 IEEE Proceedings of INFOCOM, pp. 1–9 (2010)
31. Yu, G., Kouvels, P.: On min-max optimization of a collection of classical discrete optimization problems. Optimization Theory and Applications 98(1), 221–242 (1998)
32. Le, K., Bilgir, O., Bianchini, R., Martonosi, M., Nguyen, T.: Managing the cost, energy consumption, and carbon footprint of Internet services. SIGMETRICS Perform. Eval. Rev. 38(1), 357–358 (2010)
33. Abbasi, Z., Mukherjee, T., Varsamopoulos, G., Gupta, S.K.: Dynamic hosting management of web based applications over clouds. In: International Conference on High Performance Computing Conference(HiPC 2011) (December 2011)
34. Qureshi, A., Weber, R., Balakrishnan, H., Guttag, J., Maggs, B.: Cutting the electric bill for Internet-scale systems. In: Proc. ACM SIGCOMM, pp. 123–134 (2009)
35. Mitchell, M.: An introduction to genetic algorithms. The MIT press (1998)
36. Price, K., Storn, R., Lampinen, J.: Differential evolution: a practical approach to global optimization. Springer (2005)
37. Schaffer, J.: Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the 1st International Conference on Genetic Algorithms, pp. 93–100. L. Erlbaum Associates Inc. (1985)
38. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm intelligence: from natural to artificial systems, vol. (1). Oxford University Press, USA (1999)
39. Maniezzo, V., Carbonaro, A.: Ant colony optimization: an overview. In: Essays and Surveys in Metaheuristics, pp. 21–44 (2001)
40. Zhang, L., Li, K., Zhang, Y.: Green task scheduling algorithms with speeds optimization on heterogeneous cloud servers. In: Proceedings of the 2010 Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing, pp. 76–80. IEEE Computer Society (2010)
41. Lee, Y., Zomaya, A.: Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling. In: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 92–99 (2009)
42. Chen, W., Zhang, J.: An ant colony optimization approach to a grid workflow scheduling problem with various qos requirements. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 39(1), 29–43 (2009)
43. Liefooghe, A., Jourdan, L., Talbi, E.: A unified model for evolutionary multi-objective optimization and its implementation in a general purpose software framework. In: IEEE Symposium on Computational Intelligence in Miulti-Criteria Decision-Making, MCDM 2009, pp. 88–95. IEEE (2009)
44. Lee, Y., Zomaya, A.: A novel state transition method for metaheuristic-based scheduling in heterogeneous computing systems. IEEE Transactions on Parallel and Distributed Systems, 1215–1223 (2008)
45. Cohoon, J., Hegde, S., Martin, W., Richards, D.: Punctuated equilibria: a parallel genetic algorithm. In: Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and their Application, pp. 148–154. L. Erlbaum Associates Inc. (1987)

46. Yu, J., Buyya, R., Tham, C.: Cost-based scheduling of scientific workflow application on utility grids. In: Proceedings of the First International Conference on e-Science and Grid Computing, pp. 140–147. IEEE Computer Society (2005)
47. Gupta, G., Younis, M.: Load-balanced clustering of wireless sensor networks. In: IEEE International Conference on Communications, ICC 2003, vol. 3, pp. 1848–1852 (2003)
48. Bashyal, S., Venayagamoorthy, G.: Collaborative routing algorithm for wireless sensor network longevity. In: 3rd IEEE International Conference on Intelligent Sensors, Sensor Networks and Information, ISSNIP 2007, pp. 515–520 (2007)
49. Camilo, T., Carreto, C., Silva, J., Boavida, F.: An energy-efficient ant-based routing algorithm for wireless sensor networks. Ant Colony Optimization and Swarm Intelligence, 49–59 (2006)
50. Huang, R., Chen, Z., Xu, G.: Energy-aware routing algorithm in wsn using predication-mode. In: 2010 IEEE International Conference on Communications, Circuits and Systems (ICCCAS), pp. 103–107 (2010)
51. Moore, J., Chase, J., Ranganathan, P.: Weatherman: Automated, online, and predictive thermal mapping and management for data centers. In: IEEE International Conference on Autonomic Computing (ICAC), pp. 155–164 (June 2006)
52. Bash, C., Forman, G.: Cool job allocation: Measuring the power savings of placing jobs at cooling-efficient locations in the data center. HP Laboratories Palo Alto, Tech. Rep. HPL-2007-62 (August 2007)
53. Sharma, R., Bash, C., Patel, C., Friedrich, R., Chase, J.: Balance of power: Dynamic thermal management for internet data centers. IEEE Internet Computing, 42–49 (2005)
54. Rao, L., Liu, X., Ilic, M., Liu, J.: Mec-idc: joint load balancing and power control for distributed internet data centers. In: Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems, pp. 188–197. ACM (2010)
55. Buchbinder, N., Jain, N., Menache, I.: Online Job-Migration for Reducing the Electricity Bill in the Cloud. In: Domingo-Pascual, J., Manzoni, P., Palazzo, S., Pont, A., Scoglio, C. (eds.) NETWORKING 2011, Part I. LNCS, vol. 6640, pp. 172–185. Springer, Heidelberg (2011)

Chapter 2
Energy-Aware Provisioning of HPC Services through Virtualised Web Services

Alexander Kipp, Tao Jiang, Jia Liu, Mariagrazia Fugini, Ionut Anghel, Tudor Cioara, Daniel Moldovan, and Ioan Salomie

Abstract. In this chapter we present a support infrastructure for Virtual Organisations allowing for complex IT service products delivery and distribution in a controlled yet open way. The provision of services under such paradigms usually requires a specialised environment and knowledge base and tools. In particular, in HPC (High Performance Computing) environments, a detailed knowledge of the available systems and of their behaviour and performances is of essential importance since it offers a way to avoid the downgrade of performances and the related increase of costs. Furthermore, HPC environments are characterised by a very high demand of energy. However, most users of such infrastructures are not aware of the technologies they are using, enforcing the involvement of according experts in order to avoid a non-optimal usage of the resources, and thus a waste of time, energy and money. We show examples of such a complex IT product by describing a sample process in the field of medical computation - cancellous bone simulation - and illustrate how such a complex product can be provided in an easy-to-use and energy-efficient fashion via a service virtualisation infrastructure.

Alexander Kipp, Tao Jiang, Jia Liu
High Performance Computing Centre Stuttgart, Germany
e-mail: {kipp,jiang,liu}@hlrs.de

Mariagrazia Fugini
Politecnico di Milano, Italy
e-mail: fugini@elet.polimi.it

Ionut Anghel, Tudor Cioara, Daniel Moldovan and Ioan Salomie
Technical University of Cluj-Napoca, Romania
e-mail: {ionut.anghel,Tudor.CIOARA,Daniel.MOLDOVAN,
Ioan.Salomie}@cs.utcluj.ro

S.U. Khan et al. (Eds.): Evolutionary Based Solutions for Green Computing, SCI 432, pp. 29–53.
springerlink.com

2.1 Introduction

Beside the use of the Internet as a source of information, also the use of the Web as an interaction platform is getting popular for the provisioning of services of many sorts, such as supply chain services, telecom services, e-commerce services, or energy provisioning services, to mention just a few. Furthermore, with the trend towards cloud computing, the Internet has gained additional interest from stakeholders as a communication and interoperability platform allowing provisioning of almost any kind of resources, thanks to the availability of data and communication standards. By taking into account these infrastructures which allow integrating a wide range of services and resources, possibly distributed all over the world, the concept of Virtual Organisations (VO) has become widely used since the beginning of the 21th century. The VO model, as defined in [12], foresees to handle virtual resources, typically made available via the Internet, in order to provide aggregated and complex products consisting of different orchestrated services. This approach has been considered in a large variety of research projects, e.g. Akogrimo [19], [47]; BEinGRID [27]; TrustCoM [38] and BREIN [44]. In particular, the capability provided by the VO paradigm to provide dynamic and self-healing environments, able to react autonomously to unforeseen events, such as the underperformance or the failure of a subcontracting service provider, has been one of the drivers of these research projects. However, the management of such dynamic environments is still regarded as one of the most challenging tasks, due to the technical difficulties related to their ability to fully exploit adaptability and self-awareness [13]. These challenges are related for example to the various aspects to be considered when a service provider is replaced by another e.g., due to a service failure. Finding a compatible substitute service, replacing it under the correct contractual issues, and having it connected correctly to the service network are some of these challenging tasks [14]. This is even more crucial in those tasks which are related to internal management of resources within an organization. In such direction, Enterprise Resource Planning (ERP) and Enterprise Application Integration (EAI) are becoming more and more important for an efficient usage of the internal resources, whilst facing similar challenges when operating a VO consisting of sole external services.

The conceptual and practical solution for most of the above issues has been offered by the Service Oriented Architecture (SOA) paradigm, which mapped the well established and proven ideas of providing standardized interfaces - abstract-ing from the underlying realisation of functionality - from the world of electric engineering to the world of software engineering.

In order to put such an approach at work, web service technologies have been seen as the enabler for a real SOA in a distributed environment based on web technologies. However, this approach has been successful only in a limited way, since, due to a continuous growth in the field of web service standards, the original goal, namely to decouple interface definitions from service implementation, which enables service consumers to invoke the interface methods, has not been fully achieved as yet due to both technical difficulties, bound to control of distributed resources, security and availability issues, and the difficulty to construct web service repository

(in the UDDI style) able to become used in world-wide commercial environments. The use of such VOs based on web services has instead become more accepted and used in scientific environments, where resource sharing is used for example to conduct large experiments needing specialized resources (e.g., high performance computing resources or large databases of samples and data). So far, only technically - and scientific oriented people - are keen to realize and use software processes based on web services in a VO style, although the cloud model based on a pool of web services to be invoked in a VO fashion is be-coming of interest for the commercial and enterprise world as well. In particular, the different IT experiences of users show that currently available environments are not satisfying their concrete requirements and needs. We show examples of such a complex IT product by describing a sample process in the field of medical computation - cancellous bone simulation - and illustrate how such a complex product can be provided in an easy-to-use fashion via a service virtualisation infrastructure whilst analyzing the corresponding energy-consumption footprint. This kind of simulation is a very complex process, requiring a huge amount of detailed knowledge about the computing infrastructure and infrastructure setup. Through this example, we show how such a complex process can be provided as an IT service to be provided and consumed according to a web service VO paradigm, where the virtualisation of the underlying infrastructure and environment enables the invocation of the simulation process within a common workflow language, like e.g. BPEL. We analyze the energy consumption of the environment and steer the deployment and execution of the simulation accordingly. In particular, this allows users with little IT skill to consume such complex services with limited burden needed to acquire knowledge about the used environment. In order to achieve energy efficient process workflow within the virtualised environment, we describe our approach of energy-aware provisioning and allocation of virtualised HPC resources. In particular, we describe how this is achieved by taking into account constraints about the Quality of Service (QoS) requirements, through resource consolidation and migration techniques, or by adjusting the state of physical nodes by changing e.g., the P-State of a CPU.

Therefore, we apply the Global Control Loop of the GAMES project infrastructure [15] to our service virtualisation environment, in order to allow for a transparent steering of the job deployment by taking into account the energy efficiency of the entire processes.

This chapter is organized as follows. Section 2.2 analyzes the capabilities of common workflows description languages in the scientific domain. Section 2.3 depicts our service virtualisation infrastructure allowing for the integration of complex service and IT infrastructures via virtualised service endpoints. Section 2.4 presents the processing of the GAMES Global Control Loop component, as well as the coupling of this component with our service virtualisation infrastructure allowing for a transparent execution of scientific workflows, whilst taking into account the corresponding energy efficiency. Section 2.5 illustrates a sample workflow where our enhanced virtualisation infrastructure is used, whilst Section 2.6 gives the evaluation of the Global Control Loop showing the potential impact of this component on a classical cloud environment. Finally, Section 2.7 concludes this chapter.

2.2 Scientific Workflows with Common Workflow Description Languages

In this section, we analyse the requirements for scientific workflows, and highlight in which mode the "de-facto" standard for workflows in eBusiness, namely the BPEL (Business Process Execution Language) workflow language, can be used to enact and deploy HPC scientific processes.

2.2.1 Requirements for Scientific Workflow Environments

Scientists are typically specialized in their application domain while not being (and often are not interested in being) aware of the underlying IT and service infrastructure being used for their experiments and simulations [35]. However, since some of the current application environments require a certain knowledge of the IT infrastructure and software environments, in particular in HPC and simulation environments, in order to set up the experiment, to select the useful resources (data and programs) from the platform or to select a trusted partner in cooperation during an experiment, scientists are often forced to gain this acquaintance. This basic knowledge is typically not sufficient to execute and follow up the simulation processes, or to face specific problems related to the IT infrastructure, e.g. in case the IT environment does not behave as expected (for example, when a failure or an exception arises). This obliges scientists to ask for support from IT experts, which typically causes delays and inconveniences for scientists and IT experts. Therefore, it is recommended to decouple the IT infrastructure from the application logic, so reflecting the separation between the scientific knowledge about the applications from the IT knowledge about service provisioning. In particular, this decoupling allows for an improved professional management of the IT infrastructure environment, whilst scientists can concentrate on research aspects related to the studied scientific domain.

Never the less, scientists are still interested in controlling the workflow execution, since, during the simulation, in particular in case of unforeseen phenomena, it is of interest to suspend the execution, or to change some parameters and later resume the simulation. In addition, in case of unforeseeable phenomena, it is also of interest to extract intermediate and temporary results of the computation.

Beside this, during execution of a simulation process, a refinement of the workflow structure might be required, due to the intermediate simulation results. This modification/evolution of the process execution mode can be due to an unpredicted behaviour of specific components or elements related to the application domain, rather than to the IT infrastructure.

Another issue to be considered in the context of scientific simulation environments is that these processes are often long-running processes. Furthermore, the infrastructure of such HPC computing jobs is replaced periodically due to the fast development and improvement in the IT hardware sector. This development,

allowing for higher performance computing power with less energy consumption, typically requires to replace the IT architecture after approximately 5 years, since systems need to adjust their energy saving requirements, e.g., trivially, due to the continuous rising of electric power prices and to the increased interest of the community in energy saving and in environmental impact. Hence, the adaptation of the IT infrastructure is a common evolution, implying also an evolution of the simulation environment due to changed hardware (e.g., new IT infrastructures allowing for better performance for a specific job with better energy efficiency require adaptation in the scientific process structure avoiding unnecessary cycles or invocation of more performing software services). In HPC environments, heterogeneous resources are used in order to face the different requirements of varying computing jobs, causing an additional overhead for the scientists in order to determine the optimal configuration of a specific scientific job.

The idea of evolving scientific workflows is not new. Actually, various Workflow Management Systems (WfMS) are available which support changing the process structure under varying constraints and exceptions. However, these approaches are typically specialized and limited to specific domains, causing problems in case of collaborations within inter-disciplinary domains, like e.g. Taverna [34], Pegasus [8] or Triana [39]. GriCol has been established as a reference language for scientific workflows [5]. However, GriCol exhibits the same gaps as the mentioned frameworks. In these environments, scalability is also not well addressed nor choreographies are fully considered, in particular with respect to cross-domain collaborations [16].

Another peculiarity of scientific workflows it that their results are often achieved by several executions of the simulation process with varying parameters (parameter study) in a trial-and-error manner [42]. This computation mode is due to the fact that for most real-world problems it is difficult to find a mathematical model to describe the entire problem statement. Therefore, simulations, in particular parameter studies, are executed to discover the optimal solution by trying various different settings. Once a result is achieved, it is also important to allow for reproducibility of the achieved results, so allowing for result traceability.

2.2.2 *Applying Common Workflow Description Languages to the Scientific Computing Domain*

Simulation processes follow a well defined flow, which can be described using common workflow languages [16]. In particular, the de-facto standard for web service based workflows, namely BPEL, is in general usable for scientific workflows. The major target of common workflow environments is the automation of well defined processes, which maps theoretically to the principle of parameter studies. These studies are reflected by the execution of the same process with varying input parameters, until a specific goal is achieved.

Typically, within scientific processes large amounts of data have to get processed [30], so the according handling is a fundamental issue [9]. However, since BPEL mainly concentrates on the control flow but not on the data flow, BPEL shows some gaps with this respect. Namely, BPEL is missing an according support for the handling of large data sets at workflow level, pipelining, the sharing of data between different and potentially distributed instances, as well as different abstraction views on the according data sets [40]. In addition, these data sets are often of heterogeneous nature, so, in order to allow for the usage of a workflow engine in this context, the explicit passing of links between these data sets, as well as the potential mapping of the different data types, has to be integrated in a transparent way.

Within such environments, scientists typically act as modeller, user, administrator and analyst of the workflow [16]. Often, the development of scientific workflows is typically done in a "try-and-error" manner, whilst these classical workflows describe the control flow of an application / process not considering the according data flow. The major gap here is that common workflow languages do not provide sensible mechanism to handle large datasets, which have to get processed during execution of the scientific workflow [40], [29].

The mentioned workflow technologies are already established in the eBusiness domain, where the major focus relies on the description of well known processes. Due to the well-structured proceeding within simulation environments, allowing for an adaptation of these common established workflows technologies in this context, the scientific community is getting more and more interested in WfMSs [10], [43]. These WfMSs, in particular in combination with a SOA based environment, allow for the execution of experiments within highly heterogeneous environments. In particular in HPC environments this is a very challenging issue, since within these environments typically several different machines for different job types are involved. This is due to the fact that specific simulations tend to run on specific hardware and software platforms differently, in particular with respect to the execution performance. So, it would make sense to execute the different phases of a simulation process on different compute machines, possibly with different settings. This shows once again that, for an optimal usage of the platforms, a good knowledge of the underlying infrastructure is important. However, most scientists do not have - and are not interested in - this knowledge. They typically prefer to focus on the development of their simulation processes. Hence, an environment allowing scientists to concentrate on the design and on the execution flow of the simulation process, rather than on technical details of the IT platform, is essential to face this issue.

Flexibility is another crucial issue in scientific workflows. Due to changes in environments and to the unpredictable and dynamic behaviour of the application, an environment for such simulation processes must be able to allow for a transparent usage of platform resources. This required flexibility has two main consequences: on one hand, an adjustment of the workflow itself might be necessary, due to received intermediary results, e.g. showing that specific branches of a simulation should be taken into account in more detail. On the other hand, it might be necessary to adjust the service infrastructure, for instance because the execution monitoring shows that some processing steps could be performed in a more efficient way if being deployed

on specific systems or reveals that some steps could be adapted for energy efficiency in the runtime environment (e.g., by turning the CPU or memory to work in energy-saving mode). From a technical perspective, this issue can be faced by providing the HPC resources in a SOA like fashion, enabling the usage of resources without the need to take into account the corresponding technological details. Therefore, an interoperable access to resources via web service technologies, namely by referring to the WSDL and SOAP standards, would allow obtaining an interoperable and abstract access to resources by hiding, at least to some degree, the technical peculiarities of the underlying infrastructure. In the following, these resources can be orchestrated with state-of-the art workflow languages, such as BPEL.

The transparent usage of workflow and service infrastructures requires automated data transformation and service discovery. This degree of platform transparency for the end user is highly demanded from common web service- based environments, and is however not yet provided by all service systems. As discussed before, conventional scientific workflow languages are mainly data driven and are typically provided within a hybrid language, which sometimes requires some deep knowledge of the technical details.

In summary, scientific environments can benefit from the robustness of proven, common workflow technologies, in particular by taking advantage of the proven and grown integrated mechanisms for recovery, exception handling and flow control. However, most common scientific environments allow only for changing the input parameters, but not the server infrastructure.

BPEL, as a service-based business specification language, is widely accepted in industry and research, and has many benefits in our context of HPC applications. First, it enables the integration of legacy applications via web service interfaces. Furthermore, a broad tool support is provided, allowing for the visual design of workflows and thus making it easy to build complex workflows. In particular the visual support allows also users with little IT background to model according processes.

As it has been shown in [1], BPEL can be applied for scientific workflows. In order to fill the gaps with respect to handling of large data sets in scientific workflow environments, an extension to BPEL has been announced, namely BPEL-D [21]. The BPEL-D extension allows for the description of data dependencies [21] within BPEL processes. However, these dependencies are translated into standard BPEL before execution, so that these dependencies are not available during runtime. This makes the approach not usable within dynamic environments at all. Another approach to handle big amounts of data in BPEL processes is described in [48]. This approach allows for the integration of references to datasets across web services, and enabling to refer only to data sets which are relevant for a specific simulation, thus allowing transferring only the needed data sets to the compute nodes. However, the approach does not describe how these data sets should be transferred, in particular nor for the transport protocol to be used for the data transfer neither for the data source to be accessed. Typically, this data set is protected: hence for remote access the data sets specifications about the use of data sets have to be provided as well. An architecture for scientific workflow is described in [16]. The reference

implementation currently relies on the apache AXIS2 framework and does not allow for an abstraction of the integrated resources and thus, does not allow for the dynamic, transparent evolution of scientific workflows. Therefore, the realisation of an extended service bus allowing for the invocation of virtual web services has to be integrated as described in Section 2.3.

2.3 Virtualisation Infrastructure

In this section the architecture of the virtualisation gateway is introduced. A detailed description of the entire virtualisation gateway architecture, as well as the realisation and evaluation, can be found in [24] and [25]. As mentioned before, there is a concrete need for service virtualisation and thus consequently for an abstraction layer. This abstraction layer operates as an intermediary service between consumer and the actual service logic by intercepting and analyzing invocations and mapping them to the corresponding service(s), allowing for a significant increase in the flexibility and dynamics of complex, distributed environments. As we have shown in [20], the gateway infrastructure allows also to encapsulate and enable for a dynamic replacement of resources within a complex airport environment, by applying software agent technologies to represent the corresponding resources and enable them to communicate via the virtualisation gateway infrastructure. This mapping can also involve the necessary transformation steps as the virtualisation gateway does not focus on a specific interface description.

2.3.1 General Architecture

Since the gateway acts as a single entry point to not only local service logics, but potentially also to the whole virtual collaboration environment, it also provides the functionality to encapsulate services.

By using "double-blind virtualisation" mechanisms, i.e., by deploying the gateway at both, the consumer and service provider side, it is possible to change resources and providers without affecting the invoking applications. Figure 2.1 presents a sample deployment of the Gateway infrastructure. The dashed line denotes an indirect link (as the gateway extends the message channel).

Figure 2.2 shows the structure of such a gateway. This structure enables service provider to *encapsulate* and *hide* their infrastructure in a way that also allows for *virtualisation of products*. With the gateway being extensible, it provides the basis to non-invasively enact *security, privacy* and *business policies* related to message transactions. With the strong SOA approach pursued by the virtualisation gateway, the structure furthermore meets the requirements of *minimal impact* and *maximum deployment flexibility*; through its filters, it supports the *standardized messaging support*. The gateway is furthermore constructed in a way that allows for

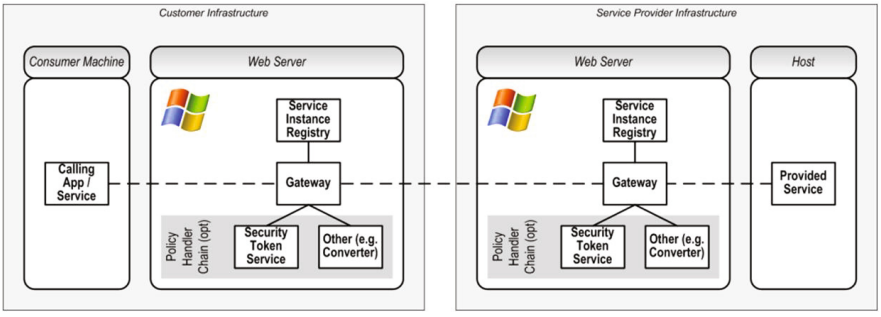


Fig. 2.1. Sample deployment of the Gateway infrastructure

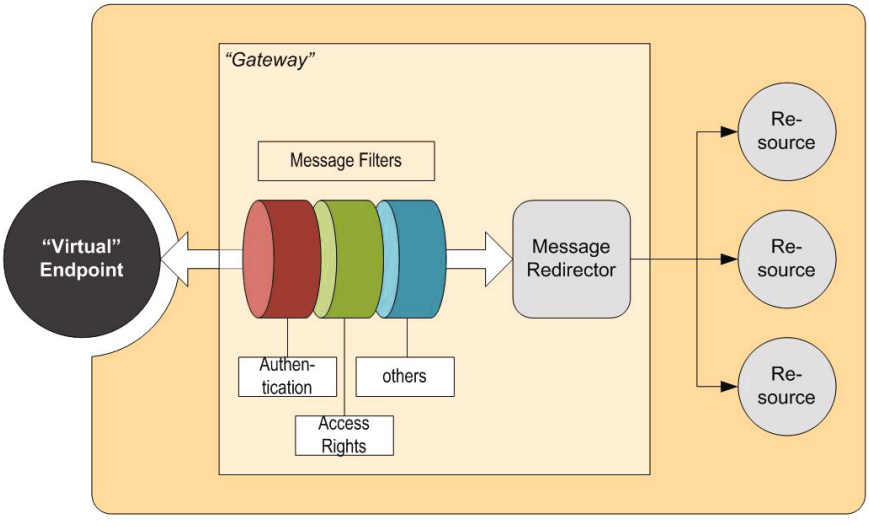


Fig. 2.2. Gateway Structure

participation in *multiple collaborations* at the same time without requiring reconfiguration of the underlying infrastructure.

As an effect, this virtualisation technology acts as an enhanced messaging infrastructure which can serve the following purposes:

- *Policy enforcement*: The gateway can enforce policies upon message transactions since it allows the definition of criteria that must be fulfilled before a potential consumer is authorized to access a specific service. To this end, policy decision points can be inserted into the message chain (see below). For example, it is possible to differentiate service consumers based on their reputation, thus estimating the relative trustworthiness, the effort worth investing etc.

- *Message security, identity and access management*: In an ideal world, all deployed client applications and web services support the corresponding specifications like WS-Security, WS-Trust and WS-Federation. Ideally, each client application should be able to fetch security tokens that are necessary for service access, and each deployed service should be able to authorize an incoming request using a claims-based security model with fine-grained authorization. Unfortunately, many applications in production today do not yet adhere to these principles, and the gateway can serve as a migration path towards broader adoption of the claims-based access model. The customer-side gateway can authenticate internal requestors, request security tokens on their behalf and protect (i.e. encrypt) outgoing messages. A service-side gateway can act as a policy-enforcement point to authenticate and authorize incoming calls. For example the gateway can establish a secure connection to the service consumer while the concrete client application does not support any secure data transmission.
- *Protocol translation*: Not only the web service protocol specification is subject to constant changes, but also a broad scope of protocols serve similar needs, thus leading to recurring interoperability issues between customers and providers. With static interfaces, the consumer has to ensure that his/her service invocation is compatible with the specifications of the provider.
- *Transformation*: Since the gateway provides an universal interface for the underlying services a transformation has to be done before the message is forwarded to the corresponding service. Transformation actions can be done as a following step of the protocol translation, in particular regarding the transformation of service specific parameters.
- *Filtering and information leakage protection*: The gateway can detect and remove private information from a request, offering a hook to install information leakage detection and prevention mechanisms.
- *Load balancing & fail over*: The gateway can act as a load balancer. If e.g. one service is currently heavy in use the gateway may decide to forward requests to this service to an equivalent one.
- *Routing*: Using a similar concept to DNS, the gateway allows for dynamic routing of messages depending on registered and available providers (load balancing). This allows on-the-fly replacement of providers, given that no current request is still in progress (in which case the according data would be lost). Since the gateway exposes virtual endpoints to the consumer, he / she does not have to adapt the process' logic, if the providers change.
- *Login monitoring*: Often it is interesting for a service provider to see which version of a service is still used by the customers. Via the gateway this information is also available.

The gateway of a service provider acts as the virtualisation endpoint of the services exposed by the respective organization. Its main task consists in intercepting incoming and outgoing messages to enforce a series of policies related to access right restrictions, secure authentication, transformation etc. thus ensuring that both provider and collaboration specific policies are maintained in transactions. Figure 2.3 shows the gateway from a customer perspective.

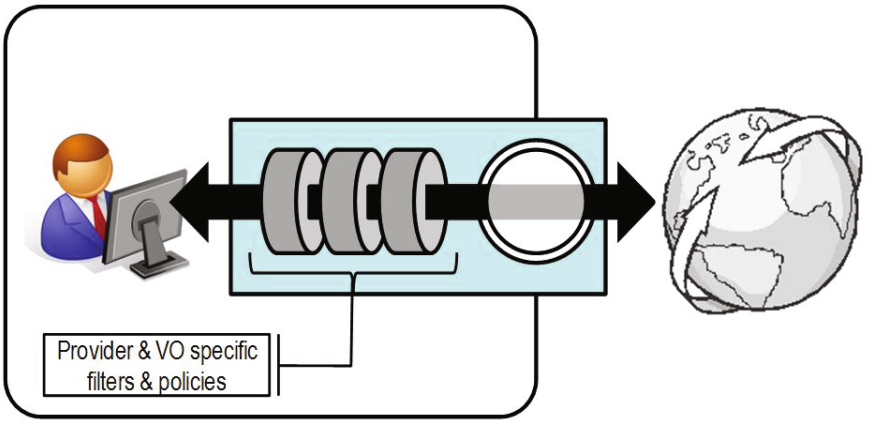


Fig. 2.3. The Gateway Principle from the Customer Perspective

As a virtual endpoint, the gateway is capable of redirecting messages from virtual addresses to actual, physical locations (local or in the Internet), thus simplifying endpoint management from client side, i.e., applications / client services are not affected by changes in the collaboration, such as replacement of providers etc. An intrinsic handler in the gateway channel hence consists in an endpoint resolution mechanism that identifies the actual destination of a given message.

In order to announce a service, the service provider publishes a virtual service interface definition (WSDL). Incoming calls are intercepted by the gateway in order to transform these messages according to the service implementation's requirements. To this end the gateway accesses a knowledge base containing all the necessary information like e.g. the mapping of the virtual name to a concrete service endpoint and the transformation of method names and parameters. As the gateway can route to any number of physical endpoints, this approach allows the provider to expose multiple WSDL documents for the same service, as well as the other way around, namely to expose multiple services via a single (virtual) web service interface. This allows not only to distinguish between various (potentially user specific) instances of a single web service, as currently exploited by cloud service farms, but enables also the service provider to dynamically move service instances between resources, as well as to compose new services through combination of existing applications, with no need to rewrite the application logic. It is hence also possible to provide services dynamically - e.g., by announcing new virtual interfaces, or by altering the service logic with no need to change the interface.

These aspects simplify service management significantly, as the issue of having to cater for individual instances and their representation is removed. For example, if the approach is used in a cloud like fashion, updates initiated at the breeding site would automatically spread across the network over time.

2.3.2 Applying the Gateway Infrastructure to Different Domains

In this section we describe how the gateway can be applied in order to face the challenging issue about the provisioning of complex IT service products. In particular, we depict how the Gateway Infrastructure can be applied to concrete environments following a hierarchical approach. This proceeding allows for a concentration of expertise within the related domains, whilst enabling expert from these domains to use and tailor the services of the other domains easily. Such an approach is depicted in Figure 2.4. In this context Gateway Infrastructure has been applied to the following domains:

- The service customer;
- The service provider;
- The service infrastructure (acting as a local gateway not being foreseen for external usage).

In this environment, the service customer has designed a local workflow in order to orchestrate different services provided locally or remotely. As depicted, all invocations are done via virtual service endpoints, which are intercepted by the proper customer gateway. In the subsequent step, the virtualisation gateway logic decides to which destination and in which format the message should get routed. Since these invocations are done via virtual service endpoints, the change of a concrete service endpoint does not affect the customer's workflow at all, since, besides the transparent routing of the according messages, the gateway allows for a transparent message transformation as well. So, in case of changing a service endpoint for a virtual service endpoint, the gateway enables for the integration of rules in order to map the common invocations to the new service interface. In particular, for the designer of the customer workflow, it is irrelevant if the service is executed locally or remotely. The process designer can concentrate on the design of the process flow logic with no need to take into account the concrete service infrastructure.

The service provider domain is divided into two sub-domains, since the service provider delivers both complex IT services and the infrastructure. Of course, the presented approach allows also for the integration of an external infrastructure provider. In that case, the Service Infrastructure Gateway would be located at the third party infrastructure service provider. However, this does not affect the workflow orchestration from the side of the domain expert or the service provider, since also the invocation of the service infrastructure is encapsulated within the gateway infrastructure by using the announced virtual service endpoints.

The complex IT service product is provided by allowing a domain expert (in our case, an expert of medical simulations) to design a workflow reflecting the necessary steps by orchestrating the needed services. All these services are integrated in the workflow via virtual service endpoints, so reflecting the functionality of a service and how this service has to be invoked. No other technical details have to be considered at this stage for the workflow designer.

Finally, the Service Infrastructure Gateway allows mapping of requests to the service infrastructure.

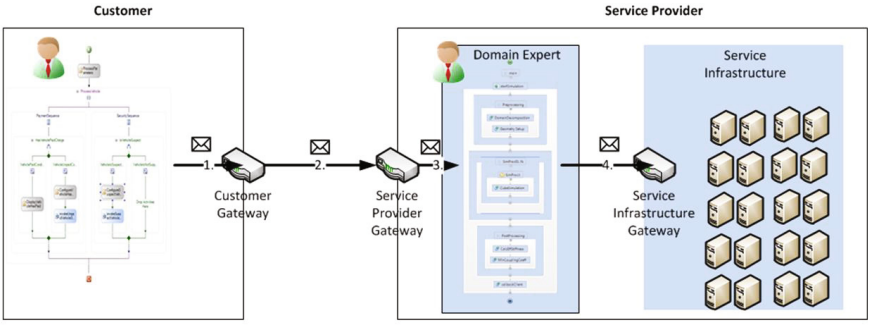


Fig. 2.4. Applying the virtualisation infrastructures to different domains

In the following, we present the processing of this hierarchical gateway infrastructure by illustrating a concrete service invocation done from the service customer.

I. As part of the customers' workflow reflecting the simulation of a specific region of a human body, a cancellous bone simulation process should be triggered. The customers' workflow engine invokes an according virtual service endpoint.
II. The corresponding message is intercepted by the customer's gateway which forwards this request to the suitable service provider, which has a contract with the customers' company for this kind of simulations. The workflow designers as well as the execution environment are not aware of which service provider is going to process the step in this workflow. Typically, service customers don't want to get bothered with these details, but rather want to receive the experiment's results.
III. The domain expert of the service provider designs a process for the simulation of cancellous bones by applying the necessary functions, applications and data flows (in the correct order). By taking advantage of the internal Service Infrastructure Gateway, the domain expert does not have to take into account the technical details of the service infrastructure. The domain expert is enabled to just orchestrate virtual service endpoints within the simulation process, which is internally deployed. Upon receipt of a request for a simulation from the customer, it is checked whether the customer is allowed to process this request. Then, the invocation is forwarded to the workflow engine hosting the requested service.
IV. Finally, the activities of the local workflow of the service provider are transmitted via the Service Infrastructure to the proper compute nodes.

Fig. 2.5 depicts in detail how the mapping of requests to the service infrastructure is handled. As described in detail in [24] and [25], the virtualisation gateway has been realised using .NET 3.5, by implementing the entire routing and managing logic. However, most IT infrastructure environments do not allow for the deployment of .NET services. So, in order to steer the invocation of application and services on the IT Service Infrastructure, an additional Cluster Controller has been realized. The

Cluster Controller is implemented as a light-weighted gSOAP [46] application, allowing for the execution of local system operations which can be triggered via a web service interface. Furthermore, the Cluster Controller allows for passing references to data sets to the compute nodes and clusters, respectively. These references include:

- The data transfer protocol and the location of the required data set.
- Optionally, a query allowing for the selection of specific elements of the dataset. This reduces the amount of data to be transferred to the compute nodes.
- The account information required to access the data set.

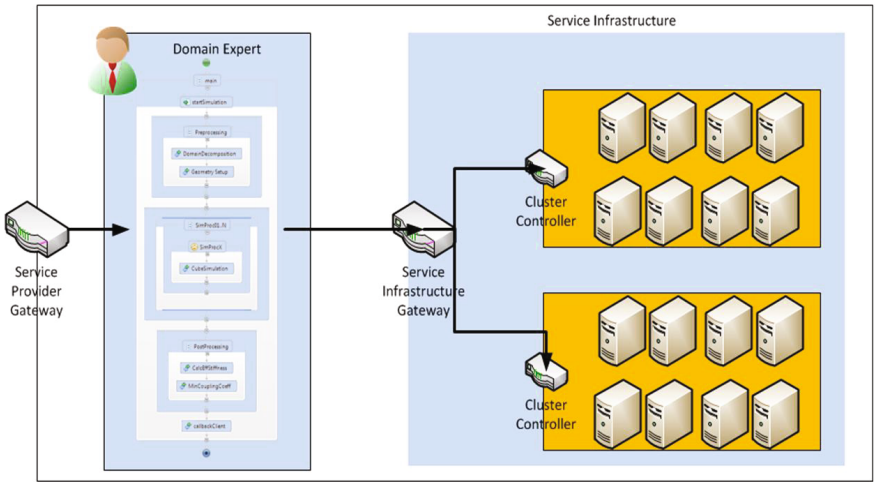


Fig. 2.5. Applying gateways to handle the local service infrastructure

These references allow for an intelligent management of large data sets since gateways implement a self-managed routing of the needed information and messages following a SOA based approach.

2.4 Energy-Aware Job Scheduling and Deployment

As described in Section 2.2, common workflow technologies established in various domains can be applied in order to provide and integrate HPC resources in terms of the SOA. In particular, BPEL facilitates the orchestration, recovery, exception handling and control of scientific workflows. However, the increased complexity in HPC domain, workflows employ heterogeneous centralized/distributed IT resources, such as CPU, memory, network and data storage in a massive way. Consequently, complex workflows consume a significant amount of energy. *Green*

workflows are envisioned to minimise the environmental burden whilst maximizing energy efficiency of workflows without sacrificing performance constraints and specific QoS requirements. Since the energy consumption is determined by the underlying hardware allocated by the workflow, effective heuristics and mechanism for dynamic adaption of job scheduling and deployment according to the current resource utilization are necessary to achieve an better energy efficiency. For instance workloads of tasks within a workflow can be consolidated and then unused resources can be switched to a low-power state, low-performance level or even be turned off in order to save energy. In the following we are going to introduce the Global Control Loop which is proposed and developed within the GAMES project and is fully competent for above mentioned energy-aware resource allocation for workflow.

In virtualised service centres, usually the computing resources are over-provisioned to be able to handle peak loads. The Global Control Loop exploits this feature by sensing the workload changes, proactively detecting the service centre over provisioned IT computing resources and identifying/executing the appropriate actions to adapt IT computing resources usage to the incoming workload. The Global Control Loop aims to: (i) analyze the service centre energy/performance data to detect those situations in which the design time defined Green and Key Performance Indicators (GPIs/KPIs [22]) levels are not reached, (ii) decide on the adaptation actions to enforce the GPIs/KPIs and (iii) act to execute those adaptation actions.

The Global Control Loop lies on top of the Service Provider resource virtualisation Infrastructure, as described in Section 3. Therefore, the Global Control loop interacts with the GAMES Monitoring Infrastructure, which is described in detail in [26], and with the Service Instance Registry of the Virtualisation Gateway infrastructure in order to steer the deployment and the parameterisation of the according HPC jobs. This procedure allows the Global Control Loop to apply two types of adaptation actions: (i) workload consolidation actions such as job deployment and migration and (ii) dynamic power management actions at server level such as wake-up or turn-off server. For example if a Customer issues a request for a virtualised service to the Provider Infrastructure, the Provider-side Global Control Loop checks for the best suitable service for the corresponding request, by taking into account the energy efficiency of the execution of the entire workflow, and configures the Service Instance Registry accordingly allowing for the transparent execution and setup of the best suitable services in a transparent fashion.

The Global Control Loop energy aware adaptation decision process is implemented using a MAPE (Monitoring, Analysis, Planning/Deciding and Execution) based self-adapting control feedback loop (see Fig. 2.6).

The *Monitoring* phase is responsible for collecting and representing in a programmatic manner the energy/performance related data. The service centre energy/performance related data are collected using the Integrated GAMES Monitoring Tools and stored in the Aggregated Database. The data from the Aggregated Database is heterogeneous and difficult to be automatically process. To solve these problems the EACM (Energy Aware Context Model) model has been developed to represent the

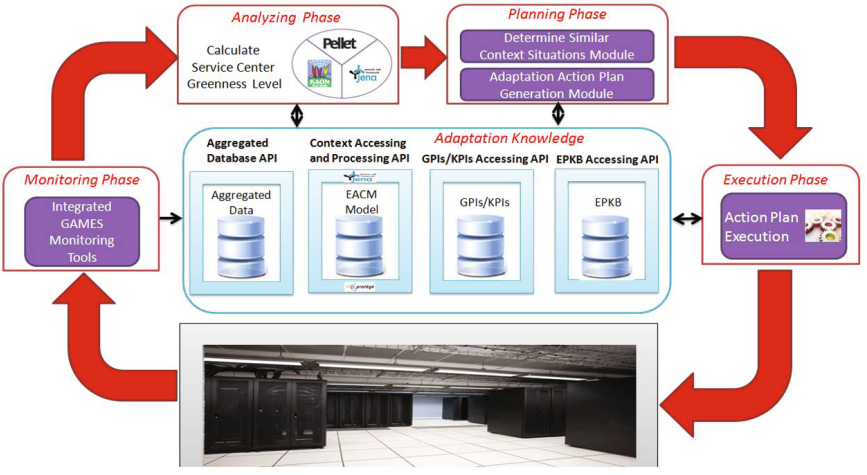


Fig. 2.6. Global Control Loop self-adaptation MAPE phases

energy/performance related data in an executable manner by means of ontologies (see Fig. 2.7).

In the EACM model, the energy/performance related context data is represented using three main concepts: Context Resources, Context Actions and Context Policies (more details can be found in [36]). The *Context Resource* concept defines the physical or virtual entities that generate and/or process the energy related context data. The *Context Actions* define the set of adaptation actions that may be executed by the Global Control Loop at run time to enforce the service centre energy efficiency goals. *Context Policies* define the service centre energy efficiency goals through a design time defined set of GPIs/KPIs.

To measure the service centre greenness level we have defined the concept of service centre context situation entropy (E). The entropy is a metric which establishes the current service centre context situation degree of complying with all the defined GPIs/KPIs. The entropy value of a service centre situation (S) is calculated using the following relation:

$$E_S = \sum_i w_{(P_i)} \sum_j w_{(r_{ij})} \cdot v_{(r_{ij})} \tag{2.1}$$

where: (i) $w_{(P_i)}$ is the weight of GPIs/KPIs policy i, (ii) $w_{(r_{ij})}$ is the weight of the service centre context resource j in the GPIs/KPIs policy i and (iii)$v_{(r_{ij})}$ is the deviation between the value recorded by the context resource j and the accepted value defined by policy i.

The entropy value is used to trigger the Global Control Loop planning phase as follows: if the entropy value is above a predefined threshold, the service centre greenness level is acceptable and adaptation actions are not required, otherwise the adaptation planning process is started.

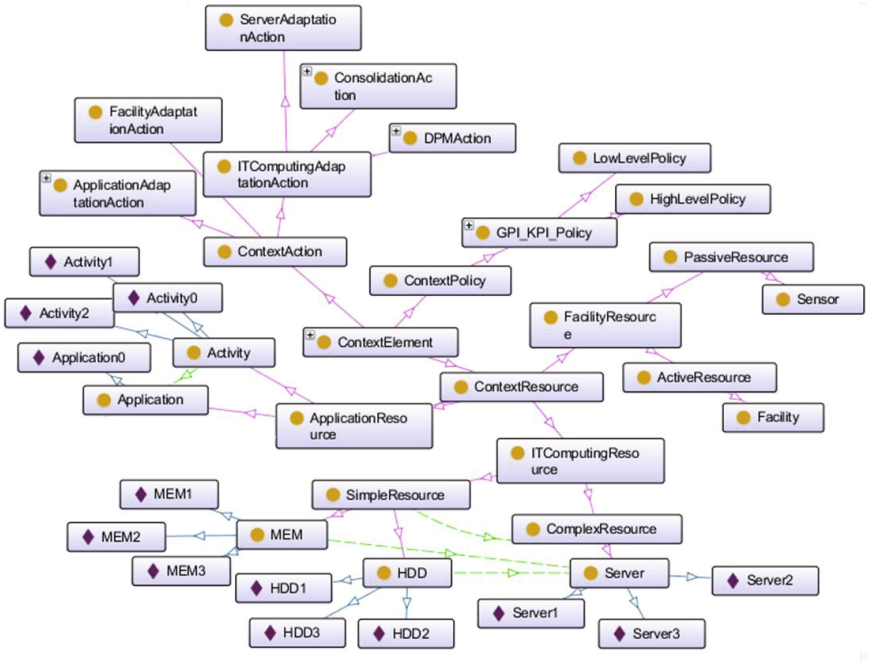


Fig. 2.7. Modelling the service centre energy/performance related context data using the EACM model ontology. Circle-marked boxes represent the EACM model concepts while diamond-marked boxes represent the concepts instances

The *Analysis* phase is responsible for interpreting energy/performance data provided by the monitoring phase, aiming to detect those situations in which the GPIs/KPIs levels are not reached and an adaptation process is needed to be executed. To evaluate the GPIs/KPIs policies, they are represented in XML and automatically converted in SWRL rules using our policy representation/evaluation model proposed in [4]. The SWRL representation of the GPIs/KPIs rules is then injected into the EACM model instance ontology implementation and evaluated using reasoning engines (see Fig. 2.8).

The *Planning* phase is responsible for deciding on the adaptation action plans that should be executed. In this phase the EPKB (Energy Practice Knowledge Base) is searched for identifying equivalent situations in which similar values for the GPIs/KPIs were determined. If an equivalent situation is found, the same adaptation action plan is taken.

Otherwise, the sequence of actions is determined through a reinforcement learning algorithm. The algorithm starts from the current service centre context situation and builds a decision tree by simulating the execution of all available adaptation actions using rewards / penalties (see Fig 2.9). In our approach to simulate the execution of an action implies creating the service centre context situation (represented by an EACM instance) that corresponds to the service centre state that will be

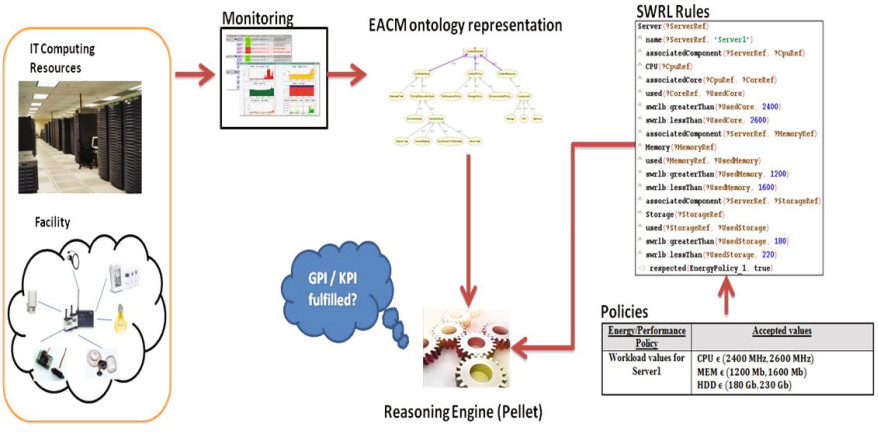


Fig. 2.8. The GPIs/KPIs policies evaluation

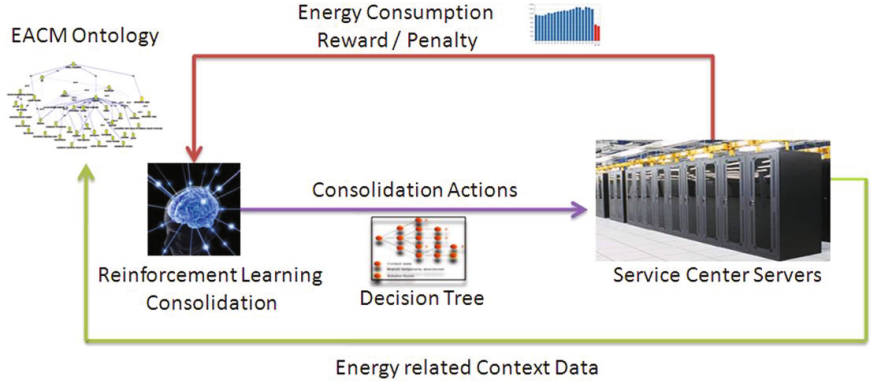


Fig. 2.9. The reinforcement learning-based adaptation actions planning process

generated as a result of the action enforcement. A tree node stores the EACM instance describing the service centre energy related context situation and its associated entropy value. A tree path between two nodes $Node_0$ and $Node_n$ defines the sequence of actions that executed starting from $Node_0$ stored context situation generates the new service centre context situation stored by node $Node_n$. For the global control loop, the minimum entropy path in the reinforcement learning decision tree represents the best sequence of adaptation actions to be executed.

The *Execution* or *Acting* phases are responsible for applying the actions determined by the planning process.

As shown above, the Global Control Loop was designed with purpose of identifying and performing the most appropriate power and performance management strategies to adapt provisioned resources to the incoming workload. To take the

advantages of it and make the provisioning of HPC services energy-aware, the Global Control Loop is going to be integrated with the virtualisation infrastructure by coupling it with the Service Infrastructure Gateway located at the service infrastructure domain. The interaction between them is described below:

- The knowledge base used by Global Control Loop is extended as a service instance registry to store the resource demands, QoS requirements and GPI/KPI threshold of activities of the local workflow of the service provider as well as the description of underlying infrastructure
- At the beginning of the simulation workflow additional steps are needed, in which the deployment request of corresponding jobs and response will be exchanged between the local workflow engine and Global Control Loop
- The simulation jobs will be deployed and the infrastructure will be dynamically adapted by Global Control Loop in an energy efficient manner.
- The invocation of the according service infrastructure stay untouched by using the virtual service endpoints which will be redirect to the real resource by Service Infrastructure Gateway.

In order to improve the concept and evaluate the effectiveness of the Global Control Loop, we conducted a BPEL process consisting of a set of executions with interactions with the Global Control Loop and invocations of different services with different resource consumption footprints, to investigate the various impact factors on energy consumption. The details are described in next section.

2.5 An Example: HPC Workflow

As described in Sec. 2.2, BPEL can be used to describe scientific workflows as well; however, the realisation of this kind of workflows with BPEL is still facing some major gaps. Therefore, we described in Section 2.3 how our virtualisation infrastructure can enable the usage of BPEL for such workflows in order to fill these gaps. In this section we provide a workflow description of our HPC usage scenario, described in detail in [6] and [7].

We are not going to debate about how the activities have to get modelled for a scientific environment. This is described in detail in [40]. Within our HPC use case, we use BPEL to control the setup and execution of a scientific workflow management system, as presented e.g. in [41].

The following BPEL process has been developed using the integrated Eclipse BPEL designer [45]. This is an example of how standard graphical editors can be used to orchestrate complex processes and make them available as easy-to-use services for external customers. By proceeding in this way, scientists can concentrate on the simulation with no need to take into account the underlying infrastructure. Figure 2.10 is the graphical representation of this BPEL process from the viewpoint of the Eclipse BPEL designer in the cancellous bone simulation.

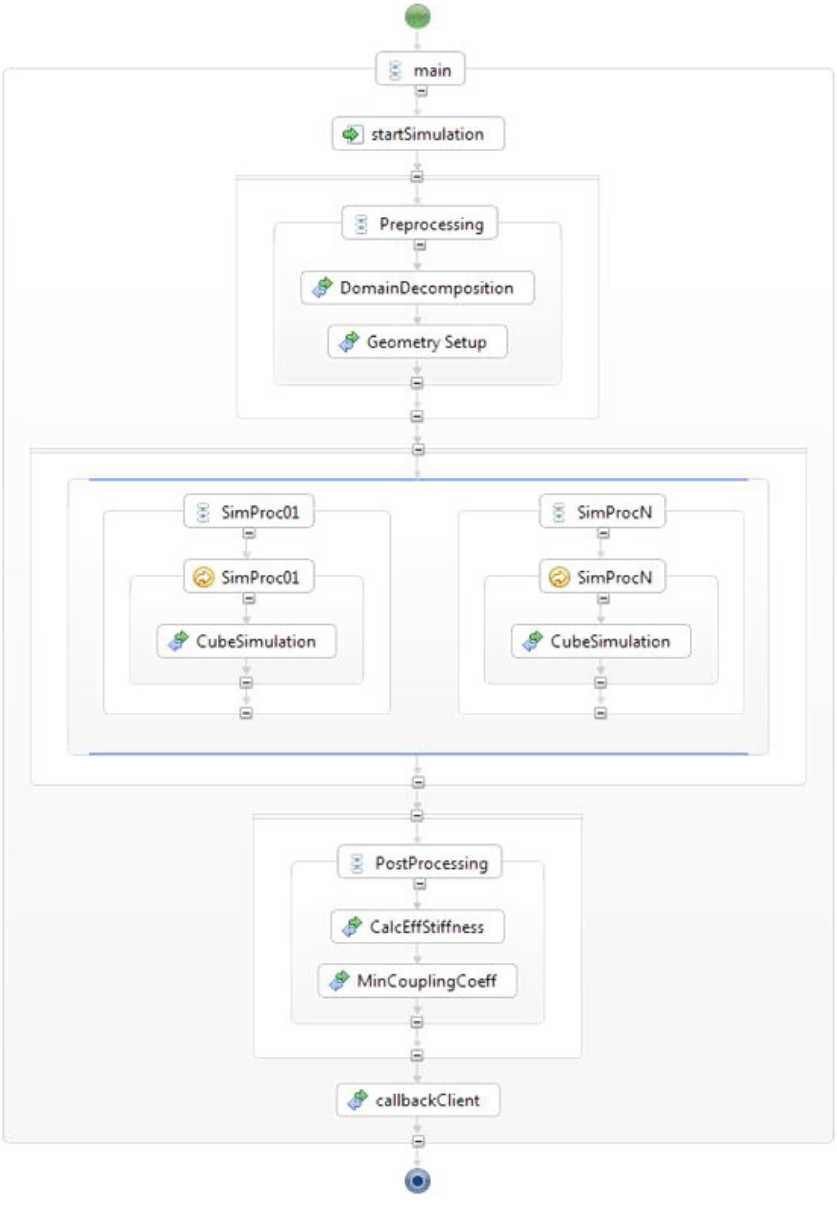


Fig. 2.10. BPEL process for a cancellous bone simulation

The process is started by receiving the invocation message via the web service interface by triggering the *startSimualtion* method. Subsequently, the defined activities of the simulation process are executed, namely:

- PreProcessing;
- Simulation;
- PostProcessing.

of the simulation results.

The required functionality of each step is provided by separate applications, which can be invoked via a virtual service interface. These invocations are mapped in the next step into concrete service endpoints, or might be forwarded to an external party. However, the scientist designing this process does not have to be aware of the location of these services. He can just orchestrate the required functions in the BPEL process; the virtualisation infrastructure takes care that the invocations are sent to the correct services.

2.6 Evaluation

In order to determine the optimization potentials of the Global Control Loop, a BPEL process invoking different services with different resource consumption footprints has been applied on the GAMES testbed [23], by using an OpenNebula Cloud Computing environment. Therefore, Figure 2.11 depicts the comparison of the energy consumption of these two execution modes.

We computed and show here also the standard error of the energy consumption of multiple test runs, which is the standard deviation of the sample means and thus gives a measure of their spread. About 68% of all sample means will be within one standard error of the population mean (and 95% within two standard errors) [31]. The standard error indicates the accuracy of the sample mean as compared with the population mean. The smaller the standard error, the less the spread and the more likely that any sample mean is close to the population mean. As shown in Figure 11, there are two positive peaks at about 0:27 and 1:33, which are due to the deployment and boot of 5 VMs. The two negative peaks at about 1:00 and 1:57 are due to the un-deployment of the VMs. In GAMES enabled execution, the power consumption is obviously lower than in normal execution, since the compute nodes which are not needed were turned off and waked up again when newly-deployed jobs came by the Global Control Loop. It has to be noted that in our test runs only 5 VMs were deployed on the testbed, and maximum 5 nodes out of 18 nodes were used to host the VMs and run the services. Obviously, the improvement could have been made more significant if the testbed nodes had been fully utilized, but this will be performed in future tests.

To summarize, the Global Control Loop tackles the service centre high energy consumption problem by consolidating the service centre workload so that its computing resources are more efficiently utilized and as consequence the total energy

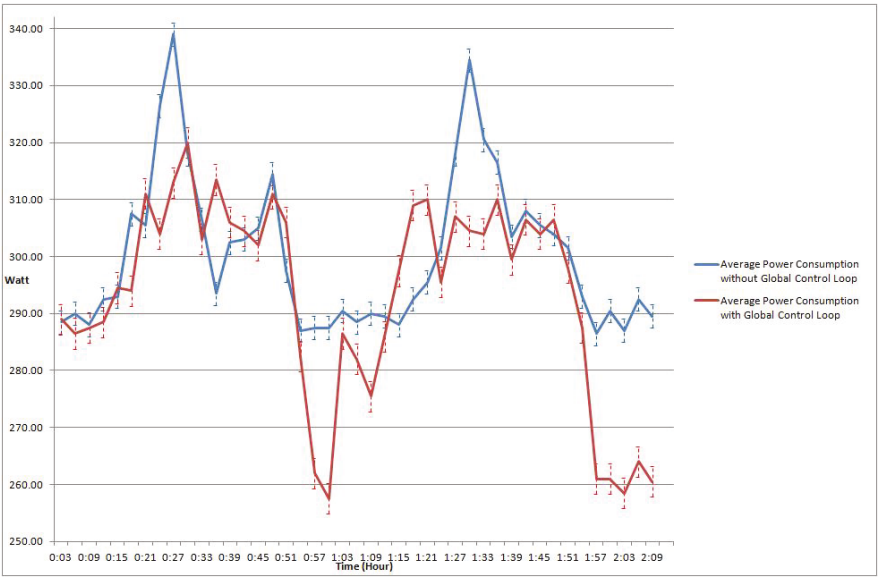


Fig. 2.11. Average power consumption comparison

consumption decreases. The Global Control Loop implementation for these first evaluations does not check and turn off servers that do not execute workload before any consolidation actions are taken. Even in these circumstances the results of the first trial are promising showing the energy saving potential of the Global Control Loop.

2.7 Concluding Remarks

In this chapter, we have presented a new approach enabling the provisioning of complex HPC services in an easy-to-use and transparent way, whilst taking into account the according energy-efficiency of the concrete jobs being executed. We have described how our service virtualisation infrastructure can be coupled with the GAMES framework, allowing for a transparent steering mechanism taking into account the current energy consumption of the corresponding services.

As demonstrated by the first evaluation of the GAMES framework within a classical cloud environment, the presented Global Control Loop can have a significant impact on the entire centre energy consumption. In the next steps, this Global Control Loop is going to be refined and coupled with the Virtualisation Gateway Infrastructure.

Another direction of research is the study of how the structure of workflow processes can be varied (process evolution), considering how the changes can make a

process execution more energy efficient, still maintaining the original functionality and QoS required for the process.

Finally, as the approach now considers only one single business process per time running in the data centre, future work will investigate how more complex situations can be managed, where several simulation processes or more complex HPC activities run concurrently.

References

1. Akram, A., Meredith, D., Allan, R.: Evaluation of BPEL to scientific workflows. In: Proc. of Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2006), pp. 269–274 (2006), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1630828
2. Astrova, I., Koschel, A., Kruessmann, T.: Comparison of enterprise ser-vice buses based on their support of high availability. In: Proc. of the 2010 ACM Symposium on Applied Computing - SAC 2010 (2010), http://portal.acm.org/citation.cfm?doid=1774088.1774605
3. Chappel, D.A.: Enterprise Service Bus: Theory in Practice, 1st edn. (Hendrickson M). O'Reilly Media (2004)
4. Cioara, T., Anghel, I., Salomie, I.: A Policy-based context aware self-management model. In: Proc. of the 11th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Romania, (2009)
5. Currle-Linde, N., Adamidis, P., Resch, M., Bos, F., Pleiss, J.: GriCoL: A language for scientific grids. In: Proc. of 2006 Second IEEE International Conference on e-Science and Grid Computing (e-Science 2006), pp. 62–62 (2006), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4031035
6. D7.2 - Validation Scenarios. GAMES report, http://www.green-datacenters.eu
7. D7.4 - First GAMES Trial Results. GAMES report, http://www.green-datacenters.eu
8. Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Patil, S., Su, M.-H., Vahi, K., Livny, M.: Pegasus: Mapping Scientific Workflows onto the Grid. In: Dikaiakos, M.D. (ed.) AxGrids 2004. LNCS, vol. 3165, pp. 11–20. Springer, Heidelberg (2004)
9. Deelman, E., Chervenak, A.: Data management challenges of data-intensive scientific workflows. In: Proc. of 2008 8-th IEEE International Symposium on Cluster Computing and the Grid (CCGRID), pp. 687–692 (2008), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4534284
10. Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-Science: an overview of workflow system features and capabilities. Future Generation Computer Systems 25(5), 528–540 (2009), http://linkinghub.elsevier.com/retrieve/pii/S0167739X08000861
11. Degenring, A.: Enterprise Service Bus. JavaSPEKTRUM 4, 16–18 (2005)
12. Dimitrakos, T., Golby, D., Kearney, P.: Towards a trust and contract management framework for dynamic virtual organisations. In: eAdoption and the Knowledge Economy: eChallenges 2004 (2004), http://epubs.cclrc.ac.uk/bitstream/701/E2004-305-TRUSTCOM-OVERVIEW-FINAL%5B1%5D.pdf
13. Dustdar, S., Dorn, C., Li, F., Baresi, L., Cabri, G., Pautasso, C., Zambonelli, F.: A roadmap towards sustainable self-aware service systems. In: Proc. of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2010), pp. 10–19. ACM (2010)

14. Friedrich, G., Fugini, M., Mussi, E., Pernici, B., Tagni, G.: Exception han-dling for repair in service-based processes. IEEE Transactions on Soft-ware Engineering 36(2), 198–215 (2010)
15. GAMES project website, http://www.green-datacenters.eu
16. Görlach, K., Sonntag, M., Karastoyanova, D., Leymann, F., Reiter, M.: Conventional simulation workflow technology for scientific simulation. In: Yang, X., Wang, L., Jie, W. (eds.) Guide to e-Science, pp. 0–31 (2011)
17. Gosain, S., Malhotra, A., El Sawy, O.A.: Coordinating for flexibility in e-business supply chains. Journal of Management Information Systems 21(3), 7–45 (2004)
18. Graf, S., Hommel, W.: Organisationsübergreifendes Management von Föderations-Sicherheitsmetadaten auf Basis einer Service-Bus-Architektur. In: Informationsmanagement in Hochschulen, 4th edn., pp. 233–246 (2010), dx.doi.org/10.1007/978-3-642-04720-6_20
19. Jähnert, J., Mandic, P., Cuevas, A.: A prototype and demonstrator of Akogrimo's architecture: An approach of merging grids, SOA, and the mobile Internet. Computer Communications 33(11), 1304–1317 (2010), http://linkinghub.elsevier.com/retrieve/pii/S014036641000112X
20. Karaenke, P., Schuele, M., Micsik, A., Kipp, A.: Inter-organizational Interoperability through Integration of Multiagent, Web Service, and Semantic Web Technologies. In: Fischer, K., Müller, J.P., Levy, R. (eds.) ATOP 2009 and ATOP 2010. LNBIP, vol. 98, pp. 55–75. Springer, Heidelberg (2012)
21. Khalaf, R.: Note on Syntactic Details of Split BPEL-D Business Processes. Institut für Architektur von Anwendungssystemen, Stuttgart, 13 (2007), http://elib.uni-stuttgart.de/opus/volltexte/2007/3230/index.html
22. Kipp, A., Jiang, T., Fugini, M., Salomie, I.: Layered Green Performance Indicators. Future Generation Computer Systems 28(2), 478–489 (2012), http://linkinghub.elsevier.com/retrieve/pii/S0167739X11000860
23. Kipp, A., Liu, J., Jiang, T.: Testbed architecture for generic, energy-aware evaluations and optimisations. In: Proc. of the First International Conference on Advanced Communications and Computation (INFOCOMP 2011), Barcelona (2011) (article in press)
24. Kipp, A., Schubert, L.: Electronic business interoperability. In: Kajan, E. (ed.) Electronic Business Interoperability: Concepts, Opportunities and Challenges, pp. 153–184. IGI Global (2011), http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-60960-485-1
25. Kipp, A., Schubert, L., Geuer-Pollmann, C.: Dynamic service encapsulation. In: Proc. of the First International Conference on Cloud Computing, Munich (2009), http://www.cloudcomp.eu/
26. Kipp, A., Schubert, L., Liu, J.: Energy Consumption Optimisation in HPC Service Centres. In: Proc. of the Second International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering, Ajaccio, Corsica, France, pp. 1–16 (2011), http://www.ctresources.info/ccp/paper.html?id=6281
27. Kipp, A., Wesner, S., Schubert, L.: A new approach for classifying Grids, 13 (2007), http://www.it-tude.com/grid_classification.html
28. Li, G., Muthusamy, V., Jacobsen, H.: A distributed service-oriented architecture for business process execution. ACM Transactions on the Web 4(1), 1–33 (2010), http://portal.acm.org/citation.cfm?doid=1658373.1658375
29. Ludäscher, B., Altintas, I., Berkley, C.: Scientific workflow man-agement and the Kepler system. Concurrency and Computation: Practice and Experience 18(10), 1039–1065 (2006), doi.wiley.com/10.1002/cpe.994
30. Ludäscher, B., Weske, M., McPhillips, T., Bowers, S.: Scientific Workflows: Business as Usual? In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 31–47. Springer, Heidelberg (2009)
31. Harvill, L.M.: Standard Error of Measurement. East Tennesse State University, http://www.ncme.org/pubs/items/16.pdf

32. Modafferi, S., Mussi, E., Maurino, A., Pernici, B.: A framework for provisioning of complex e-services. In: Proc. of the IEEE International Conference on Services Computing (SCC 2004), pp. 81–90. IEEE Press (2004), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1357993
33. Moser, O., Rosenberg, F., Dustdar, S.: Non-intrusive monitoring and ser-vice adaptation for WS-BPEL. In: Proc. of the 17th International Conference on World Wide Web - WWW 2008, p. 815 (2008), http://portal.acm.org/citation.cfm?doid=1367497.1367607
34. Oinn, T., Greenwood, M., Addis, M.: Taverna: lessons in creating a workflow environment for the life sciences. Concurrency and Computation: Practice and Experience 18(10), 1067–1100 (2006), doi.wiley.com/10.1002/cpe.993
35. Roure, D.D., Goble, C., Stevens, R.: The design and realisation of the Virtual Research Environment for social sharing of workflows. Future Generation Computer Systems 25(5), 561–567 (2009)
36. Salomie, I., Cioara, T., Anghel, I., Moldovan, D., Copil, G., Plebani, P.: An Energy Aware Context Model for Green IT Service Centers. In: Maximilien, E.M., Rossi, G., Yuan, S.-T., Ludwig, H., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6568, pp. 169–180. Springer, Heidelberg (2011)
37. Schmidt, M., Hutchison, B., Lambros, P., Phippen, R.: The Enterprise Service Bus: making service-oriented architecture real. IBM Systems Journal 44(4), 781–797 (2005), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5386706
38. Schubert, L., Wesner, S., Dimitrakos, T.: Secure and Dynamic Virtual Organizations for Business. In: Cunningham, P., Cunningham, M. (eds.) Proc. of the 15th ISPE International Conference on Concurrent Engineering (CE 2008), Collaborative Product and Service Life Cycle Management for a Sustainable World, vol. 2. IOS Press (2005)
39. Shields, M., Taylor, I.: Programming scientific and distributed workflow with Triana services. Concurrency and Computation: Practice & Experience - Special Issue on "Workflow in Grid Systems" 18(10), 1–10 (2006)
40. Sonntag, M., Currle-Linde, N., Görlach, K., Karastoyanova, D.: Towards simulation workflows with BPEL: deriving missing features from GRICOL. In: Proc. of the 21st IASTED International Conference on Modelling and Simulation MS 2010. ACTA Press (2010)
41. Sonntag, M., Karastoyanova, D., Deelman, E.: BPEL4Pegasus: Combining Business and Scientific Workflows. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 728–729. Springer, Heidelberg (2010)
42. Sonntag, M., Karastoyanova, D.: Next generation interactive scientific experimenting based on the workflow technology. In: Proc. of the 21st IASTED International Conference Modelling and Simulation (MS 2010). ACTA Press (2010)
43. Taylor, I.J., Deelman, E., Gannon, D.B.: Workflows for e-Science - Scientific Workflows for Grids. Springer (2007)
44. Taylor, S., Surridge, M., Laria, G., Ritrovato, P., Schubert, L.: Business Collaborations in Grids: The BREIN Architectural Principals and VO Model. In: Altmann, J., Buyya, R., Rana, O.F. (eds.) GECON 2009. LNCS, vol. 5745, pp. 171–181. Springer, Heidelberg (2009)
45. The BPEL Designer Project, http://www.eclipse.org/bpel/
46. The gSOAP Toolkit, http://www.cs.fsu.edu/~engelen/soap.html
47. Wesner, S.: Towards an architecture for the mobile grid (Architektur für ein Mobiles Grid). Information Technology 47(6), 336–342 (2005), http://www.oldenbourg-link.com/doi/abs/10.1524/itit.2005.47.6.336
48. Wieland, M., Gorlach, K., Schumm, D., Leymann, F.: Towards reference passing in web service and workflow-based applications. In: Proc. of 2009 IEEE International Enterprise Distributed Object Computing Conference, pp. 109–118 (2009), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5277706

Chapter 3
Macro Level Models of Power Consumption for Servers in Distributed Systems

Tomoya Enokido, Takuro Inoue, Alisher Aikebaire, and Makoto Takizawa

Abstract. In order to realize green eco-society, the total electric power consumption of information systems has to be reduced. In this chapter, we discuss novel models of the power consumption of a server in distributed systems. We first classify distributed applications of information systems into three types: computation (CP), communication (CM), and storage (ST) types of applications. We first measure at macro level the power consumption of a whole server where the types of applications are performed. We do not consider how much electric power each hardware component like CPU consumes at micro level. The power consumption of a server depends on not only hardware component but also software component. Based on the experimentations, we imply power consumption models of a server to perform types CP, CM, and ST of application processes at macro level. Then, we propose algorithms to select a server in a set of servers so that not only QoS (Quality of Service) requirement like response time and deadline constraints hold but also the total power consumption of the servers is reduced based on the power consumption models. We evaluate the selection algorithms compared with traditional algorithms like round robin algorithms in terms of the total power consumption and satisfiability of QoS requirements.

Tomoya Enokido
Faculty of Business Administration, Rissho University, 4-2-16, Osaki, Shinagawa-ku, Tokyo, 141-8602, Japan
e-mail: eno@ris.ac.jp

Takuro Inoue
Department of Computers and Information Science, Faculty of Sience and Technology, Seikei University, 3-3-1 Kichijoji-kitamachi, Musashino-shi, Tokyo 180-8633, Japan
e-mail: takuro.burton@gmail.com

Ailixier Aikebaier
Department of Computers and Information Science, Faculty of Sience and Technology, Seikei University, 3-3-1 Kichijoji-kitamachi, Musashino-shi, Tokyo 180-8633, Japan
e-mail: alisher.akber@computer.org

Makoto Takizawa
Department of Computers and Information Science, Faculty of Sience and Technology, Seikei University, 3-3-1 Kichijoji-kitamachi, Musashino-shi, Tokyo 180-8633, Japan
e-mail: makoto.takizawa@computer.org

S.U. Khan et al. (Eds.): Evolutionary Based Solutions for Green Computing, SCI 432, pp. 55–94.
springerlink.com

3.1 Introduction

In scalable information systems like peer-to-peer (P2P) [10, 35] and cloud computing systems [24, 44], huge number of server computers (servers) are interconnected and consume electric power. In order to realize distributed applications, various types of distributed algorithms are so far developed. For example, algorithms for allocating computation resources like CPU and memory to processes and for synchronizing multiple conflicting processes are discussed in terms of cost and performance [19, 37]. On the other hand, it is more significant to realize energy-aware information systems for reducing the total electrical power consumption as discussed in Green IT [26, 34]. In one approach to reducing the power consumption of information systems, the power consumption of each hardware device like CPU and memory is tried to be reduced. For example, energy-efficient CPUs are produced by Intel [28] and AMD [1]. Even if the power consumption of each hardware device could be made clear, it is difficult to estimate the total power consumption of a whole computer since the power consumption depends on what kinds of software components like operating systems and application processes are performed. In this chapter, we rather consider how much electronic power a whole server consumes at a *macro* level. We would like to discuss the novel macro-level approach to reducing the power consumption of a server. We first measure how much electric power a whole server consumes to perform application by using the power meter UWMeter [40] where the power consumption of a server can be measured every one hundred milliseconds [J/0.1 sec]. There are wide range of distributed applications like Web applications [4, 7], database applications [21], and Google file systems [23]. We classify applications into three types: computation (CP), communication (CM), and storage (ST) types of applications [16–18, 30]. In the CP applications like scientific computing, a server mainly consumes CPU resource to process requests issued by clients. In the CM applications, a server mainly consumes network resources. Here, a large volume of data is transmitted to a client like file transfer protocol (FTP) [36] applications. In the ST applications, storage resources like hard disk drives are mainly consumed. Here, processes read and write files in storage drives like hard disk drives.

In this chapter, we discuss how to reduce the total power consumption of servers to perform the types of application processes in the client-server system. Here, a client first issues a request to a load balancer. The load balancer selects a server in the servers pool and then forwards the request to the server. On receipt of the request, the server performs the request as an application process and sends a response to the client. Servers mainly consume the power to handle requests like Web page request compared with clients. It is critical to reduce the power consumption of the servers.

First, we discuss a pair of a simple computation model and a simple power consumption model [16] of a server for the CP applications, which is derived from the experimental results. The simple power consumption model is a two-state model. Here, the electric power of a server is maximally consumed if at least one application process is performed. If no process is performed, i.e. the server is in idle state, the server consumes the minimum power. In addition, the computation rate

of a server is maximum, i.e. with maximum clock cycle if at least one process is performed. This is the simple computation model. Based on the simple computation model, we discuss how to estimate time when every current application process terminates on a server. We discuss a *power consumption laxity based* (*PCLB*) algorithm to select a server for each request from a client in a pool of possible servers. In the PCLB algorithm, a server which consumes the minimum power consumption is selected for each request from a client. We evaluate the PCLB algorithm in terms of total power consumption and elapse time compared with the basic round-robin (RR) algorithm [31].

Next, we discuss the power consumption model of a server to perform the CM applications where servers transmit files to clients. Here, there are two types of file transmission models: server bound and client bound models. In the server-bound model, the total amount of maximum receipt rates of all the current clients from a server is larger than the effective transmission rate of the server. Hence, the more number of clients transmit requests to the server, the longer it takes to transmit files to the clients. On the other hand, the total amount of maximum receipt rates of all the clients is smaller than the effective transmission rate of the server in the client bound model. Hence, every client can receive a file at the maximum receipt rate. From the experimental results, we obtain the significant property of the power consumption model [18] that the power consumption rate of a server to transmit files is lineally increased for the total transmission rate. Based on the file transmission model and the power consumption model, we propose an *extended power consumption-based* (*EPCB*) algorithm to select a server in a pool of possible FTP servers. Here, a server with minimum power consumption to transmit files to every client is selected for each request from a client. Then, we evaluate the EPCB algorithms in terms of the total power consumption and elapse time compared with the RR algorithm.

Finally, we discuss the power consumption model of a server to perform application processes in the ST applications. We measure the power consumption rate of a server where files in storage drives are manipulated by application processes. We consider three types C (computation), R (read), and W (write) of processes. In C processes, only computation is performed as discussed in the CP applications. That is, CPU resources are mainly consumed. In R and W processes, files in storage drives like hard disk drives (HDDs) are read and written, respectively. We define a power consumption model of a server where processes read and write files in the types of storage drives. Here, we obtain a power consumption model of a storage server which is similar to the simple two-state power consumption model. That is, the server consumes the maximum power if at least one process is performed.

In section 3.2, we overview the related studies. In section 3.3, we present a system model which we consider in this chapter. In section 3.4, we present the experimental results on the power consumption of a server. In section 3.5, we discuss the computation model and power consumption model of a server for CP, CM, and ST applications. In section 3.6, we discuss algorithms to select a server for each type of applications. In section 3.7, we evaluate the selection algorithms for each type of applications.

3.2 Related Studies

In the green IT technologies [26, 34], the total electric power consumption of information systems has to be reduced. Various hardware technologies like low-power consumption CPUs [2, 29] are developed. For example, AMD produced Opteron processors [2] where the power efficiency is improved. Intel produced Xeon processor 5600 series [29] where power consumption can be automatically regulated so that server performance is adjusted for traffic.

In wireless sensor networks [32, 42] and mobile ad hoc networks [14, 27], routing algorithms to reduce the power consumption of each node are discussed. Especially, it is discussed how to deploy sensor nodes to reduce the power consumption of the battery of each sensor node.

A cloud computing system [24, 44] is composed of a huge number of server computers like Google file systems [23]. Various types of load balancing algorithms are discussed for distributed systems. Bevilacqua discussed a method to obtain efficient load balancing for data parallel applications through dynamic data assignment and a simple priority mechanism on a heterogeneous cluster system [6]. Colajanni *et al*. discussed a load balancing algorithm which adjusts the time-to-live values of the DNS based scheduler for geographically distributed heterogeneous Web servers [12]. The Linux Virtual Server (LVS) is a software to build a cluster system and some types of load balancing algorithms are implemented like round-robin and least-connection algorithms [31]. By these algorithms, the computation is maximized but the power consumption is not considered. On the other hand, Rajamani *et al*. identified the key factors in the system-workload context which impact energy saving policies in a cluster system and discussed how to reduce the total power consumption of servers by turning off unnecessary servers [38]. Yang *et al*. proposed an optimized method to find the required number of Web servers in a cluster system [43]. Biancini *et al*. discussed how to reduce the power consumption of a data center with a cluster of homogeneous servers [8] and heterogeneous servers [25] by turning off servers which are not required for handling a collection of requests. Meisner *et al*. proposed the PowerNap where the state of a server can be changed between a high-performance active state and a near-zero power idle state based on the existence of a request to reduce the total power consumption of enterprise blade servers [33]. In the approach, if there is no request in a server, the state of the server components is changed to deep sleep mode and every activity is suspended until new request arrival. However, it is difficult to dynamically turn on and off servers in some types of systems like mission critical systems. The approach can be adopted for only cluster systems where every server is managed in a centralized manner. In addition, computers cannot be turned off by other persons different from the owners in other types of computation models like peer-to-peer (P2P) and grid models. In this chapter, we consider a distributed system like P2P systems where there is no centralized coordination. We do not consider the approach to turning off servers to reduce the total power consumption of servers. We rather take an approach to selecting energy-efficient servers in networks.

There are various types of disk products with different specifications like solid state drives (SSDs) and hard disk drives (HDDs). Carrera *et al*. discussed four approaches to reduce the disk energy by using the different disk products in network servers [11]. In the paper, energy consumption can be reduced without performance degradation of network servers in the multi-speed disks. However, the multi-speed disks are not still general products. In this chapter, we consider general SSD and HDD drives. We consider three types C, R, and W of application processes which manipulate files in the storage drives. Then, we define a power consumption model of a server where processes read and write files in types of storage drive. Based on the power consumption model of a storage server, we discuss how to select a server in a set of storage servers for each request from clients.

Fan *et al*. discussed the aggregate power usage characteristics of large collections of servers for different three types of large-scale applications: Webserch, Webmail, and Mapreduce [20]. The paper assumes that CPUs and memory dominate the total power consumption of a server and the proposed model mainly uses CPU utilization to estimate the total power consumption of a server. Then, a strategy for maximizing the amount of computation equipments which can be deployed at a data center with given power capacity is proposed by using the CPU voltage-frequency scaling and the change of the idle power consumption. Barroso *et al*. [9] discussed how to achieve the energy-proportional computing. In the paper, it is shown to be necessary to improve the energy-proportional profiles in memory and disk subcomponents like CPUs [2, 29] to achieve the energy-proportional computing.

In this chapter, we classify applications into CP, CM, and ST types of applications. We consider the total power consumption of a server to perform the three types of application processes. We do not consider the power consumption of each component of a server to reduce the total power consumption of the server. In one approach, we would like to propose a macro-level model to show the power consumption of a server to perform application processes. First, we measure the power consumption of servers and then make clear what parameters mainly dominate the power consumption of servers.

3.3 System Model

3.3.1 Servers and Clients

Let S be a set of multiple servers s_1, ..., s_n ($n \geq 1$) each of which supports clients with the same service as shown in Figure 3.1. For example, each server holds a replica of data object d in the CM applications. Let C be a set of clients c_1, ..., c_m ($m \geq 1$) which issue requests to servers in the server set S to obtain the service. The servers and clients are interconnected in a network. We assume the network is reliable and synchronous [22]. That is, no message is lost and every message is delivered to the destination process in constant time in the network.

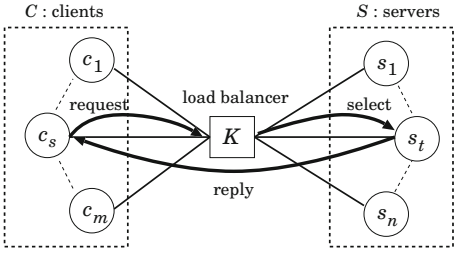


Fig. 3.1. System model

A client c_s first issues a request to a load balancer K [6,12,31] as shown in Figure 3.1. The load balancer K selects one server s_t in the server set S and forwards the the request to the server s_t. Here, the server s_t spends computation, communication, and storage resources to perform the request. For example, the CPU resources are consumed to encode multimedia data [39] and do the scientific computation in the CP applications. In the storage (ST) applications, storage drives are manipulated. On receipt of a request, the server s_t creates a process p_i. The process p_i is performed on the server s_t. Here, a term *process* means an application process on a server to handle a request issued by a client in this chapter. A process which consumes the CPU resource by computation is referred to as *computation* (CP) process. A process which consumes the communication resource by transmitting data is referred to as *communication* (CM) process. A process which reads and writes files in storage drives is referred to as *storage* (ST) process.

3.3.2 Processes in Servers

Processes are performed on a server s_t. A process being performed on a server s_t is *current* at time τ. A process which already terminates before time τ is *previous* at time τ. Let P be a set of application processes p_1, ..., p_l ($l \geq 1$) to be performed on servers in the set server S.

Let $CP_t(\tau)$ be a set of current processes on a server s_t at time τ. $N_t(\tau)$ shows the number $|CP_t(\tau)|$ of current processes on the server s_t. If $CP_t(\tau) = \{p_i\}$, a process p_i is exclusively performed on a server s_t at time τ. That is, only one process p_i is performed and no other process on the server s_t at time τ. If $p_i \in CP_t(\tau)$ and $N_t(\tau) > 1$, a process p_i is *interleaved* with another process $p_j \in CP_t(\tau)$ ($i \neq j$) at time τ. That is, multiple processes are concurrently performed at time τ. Let $CP(\tau)$ show a set of current processes on every server in the server set S at time τ, $CP(\tau) = \cup_{p_t \in S} CP_t(\tau)$.

Each process p_i starts on a server s_t at time st_{ti} and terminates at time et_{ti}. A process p_i *precedes* another process p_j on a server s_t ($p_i \rightarrow_t p_j$) iff $et_{ti} < st_{tj}$. A

process p_i is *interleaved* with another process p_j ($p_i \mid_t p_j$) iff $et_{tj} \geq et_{ti} \geq st_{ti}$ or $et_{tj} \geq st_{ti} \geq st_{tj}$. A process p_i is *connected* with another process p_j ($p_i \leftrightarrow_t p_j$) iff (1) p_i is interleaved with p_j ($p_i \mid_t p_j$) or (2) p_i is interleaved with some process p_k which is connected with p_j ($p_i \mid_t p_k$ and $p_k \leftrightarrow_t p_j$). A schedule sch_t of processes p_1, ..., p_m shows the execution history of the processes on a server s_t. A schedule sch_t is a partially ordered set of processes performed on a server s_t in the precedent relation $\rightarrow_t$ and in the connected relation $\leftrightarrow_t$.

A knot $K_t(p_i)$ is a closure subset of the processes in a schedule sch_t which are connected with a process p_i, i.e. $K_t(p_i) = \{p_j \mid p_j \leftrightarrow_t p_i\}$. The schedule sch_t is divided into pairwise disjointing knots K_{t1}, ..., K_{tl_t}. Let p_j and p_k are a pair of processes in a knot $K_t(p_i)$ where p_j first starts at time st_{tj} and p_k lastly terminates at time et_{tk}. The execution time TK_t of the knot $K_t(p_i)$ is et_{tk} - st_{tj}. Let $KP_t(\tau)$ be a current knot which is a set of current or previous processes which are connected with at least one current process in the set $CP_t(\tau)$ at time τ.

Figure 3.2 shows a schedule sch_t of four processes p_1, p_2, p_3, and p_4 on a server s_t. Here, the process p_1 is interleaved with the process p_2 ($p_1 \mid_t p_2$). In addition, $p_2 \mid_t p_3$, $p_2 \mid_t p_4$, and $p_3 \mid_t p_4$. Figure 3.3 shows how the processes are interleaved with each other. Here, a node shows a process and an edge between nodes p_i and p_j indicates the interleaved relation $p_i \mid_t p_j$. The process p_1 is connected with the processes p_2, p_3, and p_4 ($p_1 \leftrightarrow_t p_2$, $p_1 \leftrightarrow_t p_3$, and $p_1 \leftrightarrow_t p_4$). Thus, all the processes are connected with each other. Here, a knot $K_t(p_1)$ is a set $\{p_1, p_2, p_3, p_4\}$ of the processes. $K_t(p_1) = K_t(p_2) = K_t(p_3) = K_t(p_4)$. Suppose only the process p_4 is performed on the server s_t at time τ as shown in Figure 3.2. $CP_t(\tau) = \{p_4\}$. A process p_1 first starts at time st_{t1} and a process p_4 lastly terminates at time et_{t4} in the knot $K_t(p_4)$. Then, the execution time TK_t of the knot $K_t(p_4)$ is et_{t4} - st_{t1}. No process is performed before time st_{t1} and after time et_{t4} while at least one process is performed between time st_{t1} and st_{t4}.

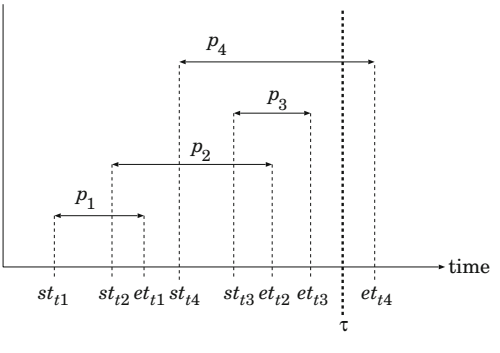


Fig. 3.2. Knot

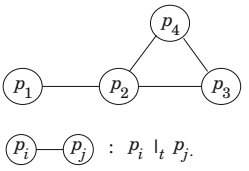


Fig. 3.3. Interleaved relation

3.4 Experimentations

3.4.1 CP Applications

3.4.1.1 Experimental Environment

We first measure the power consumption of a server s_1 to perform CP application processes. Figure 3.4 shows the experimental environment. The server s_1 is interconnected with three clients c_1, c_2, and c_3 in one-Gbps networks. The server s_1 holds data d whose size is 544 Mega bytes [MB] long. Table 3.1 shows the specification of the server s_1. Each client c_s issues a request to the server s_1. On receipt of the request, the server s_1 compresses the data d to a file f by using the gzip format [13].

Table 3.1. Server s_1

Server	s_1
Number of CPUs	1
Number of cores / CPU	1
CPU	AMD Athlon 1648B (2.7GHz)
Memory	4,096MB
Disk	150GB, 7200rpm
NIC	Broadcom Gbit Ether (1Gbps)

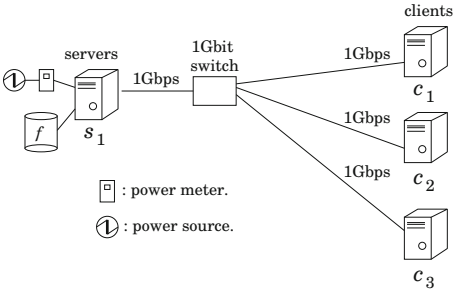


Fig. 3.4. Experimental environment

3.4.1.2 Concurrent Execution of Requests

We measure how much electric power the server s_1 consumes to perform the following three types of experimentations by using the power meter UWMeter [40]:

1. A client c_1 issues a request to the server s_1.
2. A pair of clients c_1 and c_2 concurrently issue requests to the server s_1.
3. Three clients c_1, c_2, and c_3 concurrently issue requests to the server s_1.

Figure 3.5 shows the power consumption rate [W] for experimentation time [sec]. Table 3.2 summarizes the total execution time and the power consumption rate obtained in the experimental results. It takes 44 [sec], 93 [sec] and 146 [sec] for all the concurrent processes to compress the data in experimentations 1, 2, and 3, respectively. It takes 2.1 times and 3.3 times longer time to compress the data in the experimentations 2 and 3, respectively, than the experimentation 1. Any process is performed on a server with the maximum clock frequency. The more number of processes are concurrently performed, the larger computation resource is consumed and the longer it takes to perform all the concurrent processes due to overhead like process switch.

The server s_1 consumes the electric power to compress the data at rates 119 [W], 120 [W], and 121 [W] in the experimentations 1, 2, and 3, respectively. Here, if at least one compression process is performed on a server, the electric power is maximally consumed on the server s_1.

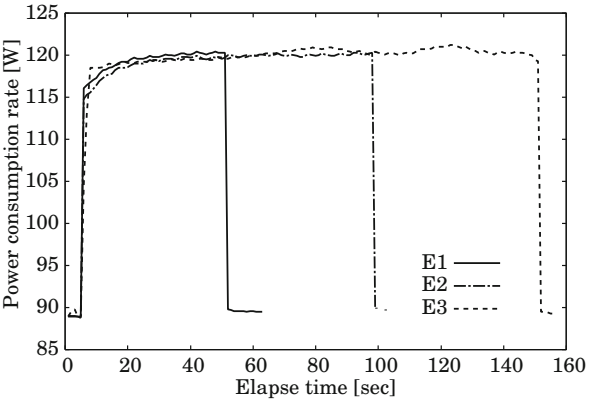


Fig. 3.5. Concurrent executions of requests

Table 3.2. Concurrent executions of requests

Experimentations	Time [sec]	Power consumption rate [W]
1	44	119
2	93	120
3	146	121

3.4.2 CM Applications

3.4.2.1 Experimental Environment

We measure how much electric power a computer spends to transmit files to other computers. As shown in Figure 3.6, a pair of servers s_1 and s_2 are interconnected with a pair of clients c_1 and c_2 in 1Gbps networks. The server s_1 is equipped with a one-core CPU as shown in Table 3.3. The server s_2 is composed of a pair of two-core CPUs. That is, the bandwidth b_{ts} from each server s_t to every client c_s is 1Gbps ($t = 1, 2$). Each client c_s downloads a file f from one of the servers. The size of the file f is 43,051,806 bytes long. Here, we measure the total power consumption of the servers s_1 and s_2.

For each server s_t, we consider two types of experimentations, one-client ($1C_t$) and two-client ($2C_t$) environments ($t = 1, 2$). In the $1C_t$ environment, one client, say c_1 downloads the file f from the server s_t. In the $2C_t$ environment, a pair of the clients c_1 and c_2 concurrently download the file f from the server s_t.

3.4.2.2 Experimental Results

A server s_t consumes the electric power to transmit files to clients. In the environment $1C_1$, the server s_1 transmits a file f to one client, say c_1 at rate tr_{11}. Here, the server s_1 is composed of one one-core CPU. The maximum transmission rate $Maxtr_1$ is 160 [Mbps] in the network of bandwidth b_{11} = 1G [bps]. In the $2C_1$

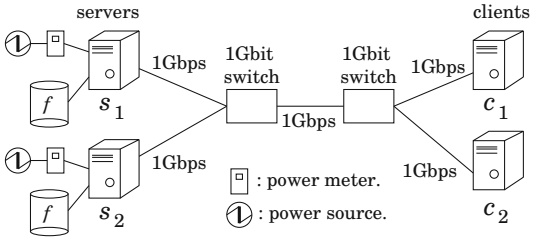


Fig. 3.6. Experimental environment

Table 3.3. Servers

Server	s_1	s_2
Number of CPUs	1	2
Number of cores / CPU	1	2
CPU	AMD Athlon 1648B (2.7GHz)	AMD Opteron 270 (2GHz)
Memory	4,096MB	4096MB
Disk	150GB, 7200rpm	74GB, 10000rpm x 2 RAID1
NIC	Broadcom Gbit Ether (1Gbps)	Nvidia Ether Controler (1Gbps)

environment, the server s_1 concurrently transmits the file f to a pair of clients c_1 and c_2. Here, $tr_1 = tr_{11} + tr_{12}$. Figure 3.7 (1) shows the power consumption rate of the server s_1 for the total transmission rate tr_1. At the higher rate tr_1 the server s_1 transmits the file f, the larger amount of power the server s_1 consumes. We obtain an approximated formula $PC_1(tr)$ to show the power consumption rate of a server s_1 for total transmission rate tr [Mbps] by applying the least-squares method on the experimental results. In Figure 3.7 (1), the bold dotted line shows the approximated power consumption of the server s_1 where one client downloads the file f from the server s_1. The dotted line shows the approximated power consumption of the server s_1 where a pair of clients c_1 and c_2 concurrently download the file f from the server s_1. Let $PC_1^1(tr)$ and $PC_1^2(tr)$ be the power consumption rates of the server s_1 in $1C_1$ and $2C_1$, respectively, at total transmission rate tr. In the single-CPU server s_1, the power consumption rate $PC_1(tr)$ is proportional to the total transmission rate tr.

- $1C_1 : PC_1^1(tr) = 0.11tr + 4.15$ [W].
- $2C_1 : PC_1^2(tr) = 0.12tr + 4.43$ [W].

Next, we consider another server s_2 with a pair of two-core CPUs. Here, the maximum transmission rate $Maxtr_2$ of the server s_2 is 450 [Mbps]. We measure the power consumption rate for the total transmission rate tr_2 for the environments $1C_2$ and $2C_2$. Figure 3.7 (2) shows the power consumption rate [W/sec] of the server s_2 for the total transmission rate tr. Figure 3.7 (2) shows that at the higher rate the server s_2 transmits, the larger power consumption s_2 consumes. The approximated formulas $PC_2^1(tr)$ and $PC_2^2(tr)$ of the power consumption rate of the server s_2 for total transmission rate tr [Mbps] are given in the environments $1C_2$ and $2C_2$:

- $1C_2 : PC_2^1(tr) = 0.02tr + 3.02$ [W].
- $2C_2 : PC_2^2(tr) = 0.03tr + 3.34$ [W].

The increase rate of the power consumption of the server s_2 in the environment $2C_2$ is about 1.5 times larger than the environment $1C_2$. Compared with the one-CPU environment $1C_t$, the power consumption rate is not so much increased for the increase of transmission rate in the two-CPU environment $2C_t$.

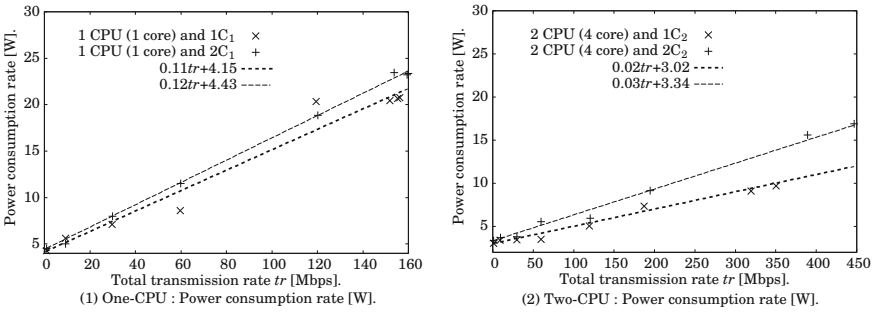


Fig. 3.7. Power consumption rate [W]

Following the experiments, the power consumption rate $PC_t(tr)$ of a server s_t is lineally increased for transmission rate $tr_t(\tau)$ ($0 \leq tr_t(\tau) \leq Maxtr_t$) as follows:

$$PC_t(tr) = \beta_t(m) \cdot \delta_t \cdot tr_t(\tau) + minE_t. \tag{3.1}$$

Here, δ_t is the power consumption to transmit one Mbits [W/Mb] for the $1C_t$ environment. δ_t depends on a server type s_t. As shown in Figure 3.7, the more number of clients, the larger amount of electric power of the server s_t is consumed. $\beta_t(m)$ shows how much power consumption is increased for the number m of clients, $\beta_t(m) \geq 1$ and $\beta_t(m) \geq \beta_t(m - 1)$. There is a fixed point $maxm_t$ such that $\beta_t(maxm_t - 1) \leq \beta_t(maxm_t) = \beta_t(maxm_t + h)$ for $h > 0$. $minE_t$ gives the minimum power consumption rate of the server s_t where no file is transmitted. $\beta_t(maxm_t) \cdot \delta_t \cdot Maxtr_t + minE_t$ gives the maximum power consumption rate $maxE_t$ of the server s_t.

3.4.3 ST Applications

3.4.3.1 Experimental Environments

We show the experimental results on the power consumption of a server to perform processes in ST application. We first measure how much amount of electric power a server consumes to perform processes in the ST applications. Here, the processes make an access to secondary storage drives. We make clear what factors dominate the power consumption of a server based on the experimental results.

Processes which read and write files in hard disk drive (HDD) and solid state drive (SSD) are performed on a server with the Linux (Cent OS). We measure the power consumption rate [W] of a server by using the electric power meter Metaprotocol UWMeter [40] every one hundred milliseconds [J/0.1sec]. We consider a server s_t with CPU (Intel®Core™i7) and 500 GB memory as shown in Table 1 where the types of processes are performed.

Table 3.4. Server and drives

Server	CPU : Intel(R) Core(TM) i7 Memory : 6.00GB DDR3 3 x 2048 OS : Cent OS 64 bit
HDD	Capacity : 500GB Rotation : 7,200 rpm Read speed : 121.45MB/s Write speed : 106.27MB/s Interface : Serial ATA300
SSD	Capacity : 120GB Read Speed : 197.02MB/s Write Speed : 93.89MB/s Interface : Serial ATA300

We consider three types of processes to be performed on a server s_t, **computation** (C), **read** (R), and **write** (W) processes. R and W processes just read and write files, respectively. In C processes, only CPU resource is consumed as discussed in the CP applications. R and W processes read and write data of one [GB] in a storage drive by using **read** and **write** system calls, respectively. If multiple R and W processes are concurrently performed on the server, every pair of processes p_i and p_j access to different files in a same storage drive. As discussed in the section 4.1, the total power consumption of a server is shown to be proportional to the size of data to be read and written. In one read/write system call, a data unit of 1024[B] is read and written. The maximum size of data which can be read and written in one system call is 1024 [B]. In the paper [15], the simple power consumption model is proposed to perform C processes on a server. If at least one C process is performed on a server, the server spends the maximum power consumption. The server spends the minimum power if no process is performed, i.e. the server is in an idle state.

We consider the following types of environments where the total number m (≥ 0) of C, R, and W processes are concurrently performed on the server s_t:

[Environments]

I. $C_t\ (m)$: Only C processes are concurrently performed.
II. $R_t\ (m)$: Only R processes are concurrently performed.
III. $W_t\ (m)$: Only W processes are concurrently performed.
IV. $CR_t\ (m)$: Only R processes are concurrently performed with a C process.
V. $CW_t\ (m)$: Only W processes are concurrently performed with a C process.
VI. $RW_t\ (m,w)$: R and W processes are concurrently performed. But no C process is performed. Here, write ratio w shows the ratio of the number of W processes to m.
VII. $CRW_t\ (m,w)$: R and W processes are concurrently performed with a C process where w shows the W process ratio.

Here, $RW_t(m,0) = R_t(m)$ and $RW_t(m,1) = W_t(m)$. $CRW_t(m,0) = CR_t(m)$ and $CRW_t(m,1) = CW_t(m)$. For examples, $RW_t(10,0.5)$ means that five R processes and five W processes are concurrently performed on the server s_t.

3.4.3.2 Experimental Results

First, we measure the average execution time $AER_t(m)$ and $AEW_t(m)$ [sec] of each process in the environments $R_t(m)$ and $W_t(m)$, respectively, where the number m (≥ 0) of processes are concurrently performed on a server s_t. Figures 3.8 shows the execution time $AER_t(m)$ and $AEW_t(m)$ for m. Here, $AER_t(m)$ is constant for $m \leq 128$ and exponentially increased for $m \geq 129$.

It takes a longer time to write data than read, $AER_t(m) < AEW_t(m)$. In our experimental results on the server s_t, it takes 30% longer time to write data in HDD than read, i.e. $AEW_t(1) = 1.3 \cdot AER_t(1)$ for HDD. $AEW_t(1) = 1.1 \cdot AER_t(1)$ for SSD.

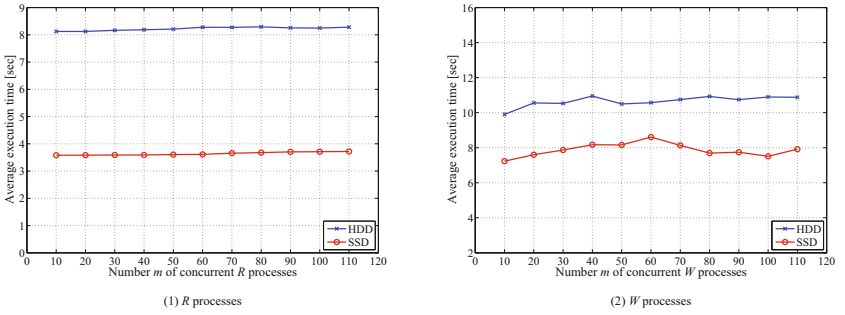


Fig. 3.8. Execution time

Since the experimental results for C processes are presented in section 4.1. We show the experimental results on R and W processes. Figure 3.9 (1) shows the power consumption rates $e_{tC}(\tau)$, $e_{tW}(\tau)$, $e_{tR}(\tau)$, and $e_{tRW}(\tau)$ [W] of the server s_t to perform processes in the environments $C_t(m)$, $W_t(m)$, $R_t(m)$, and $RW_t(m,w)$ for $m = 10$ and $w = 0.5$, respectively. As shown in Figure 3.9 (1), the power consumption rates are independent of the total number m of concurrent processes. Furthermore, the power consumption rate $e_{tA}(\tau)$ of a server s_t is maximum if at least one process is performed at time τ in a type $A(\in \{C,R,W\})$ environment. The server s_t consumes the minimum power consumption $minE_t$ if no process is performed, i.e. the server s_t is in idle state. According to Figure 3.9 (1), the minimum power consumption rate $minE_t$ is 101 [W] (10.1 [J/0.1sec]). Let $maxC_t$, $maxW_t$, and $maxR_t$ indicate the maximum power consumption rates of the server s_t where only C, W, and R processes are concurrently performed, respectively. Let $N_t(\tau)$ be the number m of processes concurrently performed on the server s_t at time τ. The power consumption rate $e_{tA}(\tau)$ [W] of the server s_t is obtained for each environment type $A \in \{C, R, W\}$ from the experimental results shown in Figures 3.9 as follows:

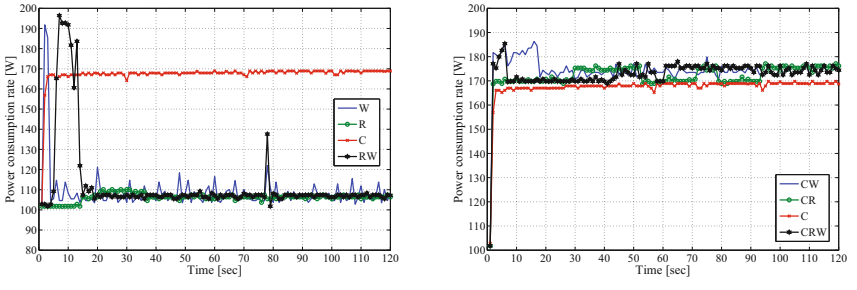


Fig. 3.9. Power consumption rates

$$e_{tC}(\tau) = \begin{cases} maxC_t & \text{if at least one } C \text{ process is performed on the server } s_t \text{ at} \\ & \text{time } \tau \text{ in an environment } C. \\ minE_t & \text{otherwise, i.e. } N_t(\tau) = 0 \text{ and no process is performed on } s_t. \end{cases}$$

$$e_{tW}(\tau) = \begin{cases} maxW_t & \text{if at least one } W \text{ process is performed on the server } s_t \text{ at} \\ & \text{time } \tau \text{ in an environment } W. \\ minE_t & \text{otherwise, i.e. } N_t(\tau) = 0. \end{cases}$$

$$e_{tR}(\tau) = \begin{cases} maxR_t & \text{if at least one } R \text{ process is performed on the server } s_t \text{ at} \\ & \text{time } \tau \text{ in an environment } R. \\ minE_t & \text{otherwise, i.e. } N_t(\tau) = 0. \end{cases}$$

$$e_{tRW}(\tau) = \begin{cases} maxRW_t & \text{if at least one } R\text{process and at least one } W \text{ process} \\ & \text{are performed on } s_t \text{ at time } \tau \text{ in an environment } RW. \\ minE_t & \text{otherwise.} \end{cases}$$

Here, $minE_t < maxW_t = maxR_t = maxRW_t = 108$ [W] $\leq maxC_t = 168$ [W].

As discussed section 4.1, the power consumption rate $e_{tC}(\tau)$ of the server s_t is maximum if at least one C process is performed on the server s_t at time τ. The power consumption rate $e_{tC}(\tau)$ is minimum, i.e. $e_{tC}(\tau) = minE_t$ if no process is performed on the server s_t at time τ. Hence, we consider the environments $CR_t(m)$, $CW_t(m)$, and $CRW_t(m, 0.5)$ where the same number of R and W processes are performed on a server s_t concurrently with a C process, respectively. Figure 3.9 (2) shows the power consumption rates $e_{tCR}(\tau)$, $e_{tCW}(\tau)$, and $e_{tCRW}(\tau)$ of the server s_t at time τ in the environments $CR_t(m)$, $CW_t(m)$, and $CRW_t(m, 0.5)$ for $m = 10$, respectively. Here, $m = 10$. According to Figure 3.9 (2), $maxCRW_t = maxCR_t = maxCW_t = 1.1 \cdot maxC_t$. In the environment $CRW_t(m, 0.5)$, the larger amount of power is consumed than the environment $C_t(m)$. The minimum power consumption rate $minE_t$ and the maximum power consumption rates $maxR_t$, $maxC_t$, and $maxCRW_t$ of the server s_t are 101, 109, 168, and 176 [W], respectively.

3.5 Power Consumption Models

3.5.1 CP Applications

In this section, we define a pair of the computation model and the power consumption model of a server to perform processes in computation (CP) type applications.

3.5.1.1 Computation Model

A. Computation Rate of a Server

Processes which consume CPU resource are performed on a server s_t. First, we discuss a computation model of a server to perform processes. We consider the following parameters to define the computation model of a server s_t:

- T_{ti} = total computation time of a process p_i on a server s_t [sec].
- $minT_{ti}$ = minimum computation time of a process p_i which is exclusively performed on a server s_t ($1 \leq minT_{ti} \leq T_{ti}$) [sec].
- $maxT_i = max(minT_{1i}, ..., minT_{ni})$.
- $minT_i = min(minT_{1i}, ..., minT_{ni})$.
- $F_{ti} = 1 / T_{ti}$ = average computation rate (ACR) of a process p_i on a server s_t ($0 < F_{ti} \leq 1 / minT_{ti} \leq 1$) [1/sec].
- $maxF_{ti} = 1 / minT_{ti}$ = maximum ACR of a process p_i on a server s_t.
- $maxF_i = max(maxF_{1i}, ..., maxF_{ni})$.
- $minF_i = min(maxF_{1i}, ..., maxF_{ni})$.

If a process p_i is exclusively performed, i.e. only the process p_i is performed without any other process, on the fastest server s_t and the slowest server s_u, $minT_{ti} < minT_{ui}$, $minT_i = minT_{ti}$, and $maxT_i = minT_{ui}$. The more number of processes are performed, the longer it takes to perform each of the processes on a server s_t. Let $\alpha_t(\tau)$ indicate the *computation degradation rate* of a server s_t at time τ ($0 \leq \alpha_t(\tau) \geq 1$) [1/sec] [Figure 3.10]. Here, $\alpha_t(\tau_1) \leq \alpha_t(\tau_2) \leq 1$ if $N_t(\tau_1) \leq N_t(\tau_2)$ for every pair of different time τ_1 and τ_2. $\alpha_t(\tau) = 1$ if $N_t(\tau) \leq 1$. In this chapter, $\alpha_t(\tau)$ is assumed to be $\varepsilon^{N_t(\tau)-1}$ where $0 \leq \varepsilon_t \leq 1$. Suppose it takes 0.5 [sec] to exclusively perform a process p_i on a server s_t. Here, $T_{ti} = minT_{ti} = 0.5$ [sec]. Here, $F_{ti} = maxF_{ti} = 1/0.5 = 2$ [1/sec]. Suppose that another process p_u where $minT_{tu} = 0.5$ starts at the same time st as the process p_i. If $\varepsilon_t = 0.98$, $\alpha_t(\tau) = \varepsilon_t = 0.98$ since $N_t(\tau) = 2$ for $\tau \geq st$. Hence, $T_{tu} = 1.02 \cdot minT_{tu} = 0.51$ [sec] and $F_{tu} = 1/0.51 = 1.96$. We define the normalized computation rate (NCR) $f_{ti}(\tau)$ of a process p_i on a server s_t as follows:

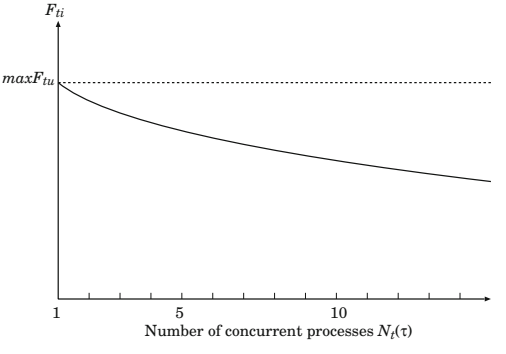


Fig. 3.10. Degradation rate

[Normalized Computation Rate (NCR)]

$$f_{ti}(\tau) = \begin{cases} \alpha_t(\tau) \cdot maxF_{ti}/maxF_i \ [1/\text{sec}]. \\ \alpha_t(\tau) \cdot minT_i/minT_{ti} \quad [1/\text{sec}]. \end{cases} \tag{3.2}$$

The maximum *NRC* $maxf_{ti}$ shows $maxF_{ti}$ / $maxF_i$. $0 \leq f_{ti}(\tau) \leq maxf_{ti} \leq 1$. $f_{ti}(\tau)$ shows how many number of steps of a process p_i are performed on a server s_t at time τ. $maxf_t$ is defined to be the maximum *NCR* of a server s_t. Here, $maxf_{ti}(\tau) \leq maxf_t$ for every process p_i and time τ. Next, suppose that a process p_i starts and terminates on a server s_t at time st_{ti} and et_{ti}, respectively. Here, $T_{ti} = et_{ti}$ - st_{ti}.

$$\int_{st_{ti}}^{et_{ti}} f_{ti}(\tau)d\tau = minT_i \cdot \int_{st_{ti}}^{et_{ti}} \frac{\alpha_t(\tau)}{minT_{ti}} = minT_i. \tag{3.3}$$

If a process p_i is exclusively performed on the server s_t, $T_{ti} = et_{ti}$ - $st_{ti} = minT_{ti}$. If other processes are performed with the process p_i, i.e. $\alpha_t(\tau) < 1$, $T_{ti} = et_{ti}$ - st_{ti} = $minT_{ti}$ / $\alpha > minT_{ti}$ where $\alpha = \int_{st_{ti}}^{et_{ti}} \alpha_t(\tau)\ d\tau$ (< 1). Here, $minT_i$ shows the total amount of computation to be spent by a process p_i. If a process p_i includes more number of computation steps than another process p_u, $minT_i > minT_u$. The more number of processes are concurrently performed with the process p_i at time τ, the smaller $f_{ki}(\tau)$ is. Suppose a process p_i starts at time st'_{ki} and terminates at time et'_{ki}. Here, $T_{ki} = et'_{ki}$ - $st'_{ki} > minT_i$ while $\int_{st'_{ki}}^{et'_{ki}} f_t(\tau)\ d\tau = minT_i$. The computation laxity $cl_{ti}(\tau)$ shows how much computation a server s_t has to spend to perform up a process p_i at time τ: $cl_{ti}(st_{ti}) = minT_i$ and $cl_{ti}(et_{ti}) = 0$. If the process p_i would be exclusively performed on the server s_t, p_i is expected to terminate at time $et_{ti} = \tau + cl_{ti}(\tau)$.

B. Simple Computation Model

From the experimental results presented in the preceding section, we obtain a simple computation model [16] where each server s_t satisfies the following properties:

I. $maxf_{ti} = maxf_{tu} = maxf_t$ for every pair of different processes p_i and p_j.
II. $\Sigma_{p_i \in P_t(\tau)}\ f_{ti}(\tau) = \alpha_t(\tau) \cdot maxf_t$.

The first condition shows that any process is performed on a server with the maximum clock frequency. The second condition means that the more number of processes are concurrently performed, the larger computation resource is consumed. $\alpha_t(\tau) \leq 1$. Here, let K_t be a knot in a schedule sch_t of a server s_t, where st and et are the starting time and termination time, respectively. The execution time TK_t of the knot K_t is $\Sigma_{p_i \in K_t}\ minT_{ti}$ / α where $\alpha = \int_{st}^{et} \alpha_t(\tau)\ d\tau$ / (et - st).

Let us consider three processes p_1, p_2, and p_3 on a server s_t as shown in Figure 3.11 (1). First, suppose that the processes are serially performed. Here, a knot K_t is $\{p_1, p_2, p_3\}$. The execution time TK_t of the knot K_t is et_{t3} - $st_{t1} = minT_{t1} + minT_{t2} + minT_{t3}$ since $\alpha_t(\tau) = 1$ for $st_{t1} \leq \tau \leq et_{t3}$.

Next, three processes p_1, p_2, and p_3 are concurrently performed on the server s_t as shown in Figure 3.11 (2). Here, a knot K_t is $\{p_1, p_2, p_3\}$. The process p_1 first

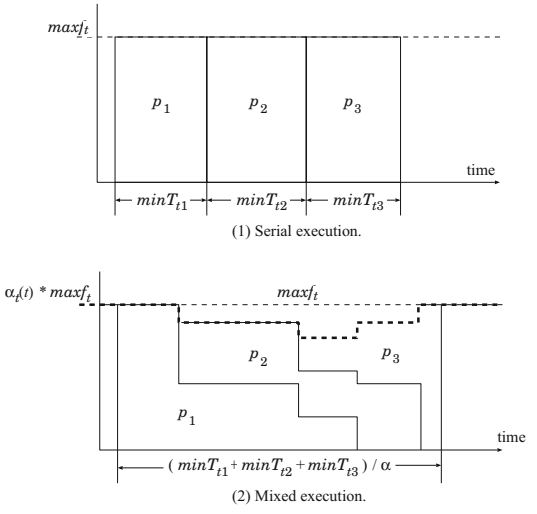


Fig. 3.11. Execution time of current knot

starts at time st_{t1}. At time st_{t2}, a process p_2 starts. Here, $\alpha_t(\tau) = \varepsilon_t^{2-1} = \varepsilon_t$ for $st_{t2} \leq \tau \leq st_{t3}$. Then, at time st_{t3}, a process p_3 starts. Here, $\alpha_t(\tau) = \varepsilon_t^{3-1} = \varepsilon_t^2$ for $st_{t3} \leq \tau \leq et_{t1}$. Here, $TK_t = et_{t3} - st_{t1} = (minT_{t1} + minT_{t2} + minT_{t3}) / \alpha$. $\alpha = [(st_{t2} - st_{t1}) + (st_{t3} - st_{t2})\varepsilon_t + (et_{t1} - st_{t3})\varepsilon_t^2 + (et_{t2} - et_{t1})\varepsilon_t + (et_{t3} - et_{t2})] / (et_{t3} - st_{t1})$.

It depends on the scheduling algorithm how much the *NCR* $f_{ti}(\tau)$ of each current process p_i is at time τ. In this chapter, we assume CPU resource is equally allocated to every current process as follows:

$$f_{ti}(\tau) = \alpha_t(\tau) \cdot maxf_t \ / \ |CP_t(\tau)|. \tag{3.4}$$

C. Estimated Termination Time

We discuss how to estimate a termination time of a current knot $KP_t(\tau) = \{p_{t1}, ..., p_{tl_t}\}$ of processes on a server s_t, where the starting time is st. Suppose a process p_i starts on a server s_t at time τ, i.e. $st_{ti} = \tau$. Let $EV_t(\tau)$ be a set of starting time and termination time of processes which start or terminate in the current knot $KP_t(\tau)$ between time st and current time τ on the server s_t. Here, each element e_l in the set $EV_t(\tau)$ is referred to as an *event* on the knot $KP_t(\tau)$. Events in $EV_t(\tau)$ are sorted in the ascending order. $EV_t(\tau) = \langle e_1, ..., e_l \rangle$ where $e_1 \leq \cdots \leq e_l$. For example, $EV_t(\tau) = \langle st_{t1}, st_{t2}, st_{t3}, et_{t3}, \tau \rangle$ in Figure 3.12. If there is no current knot $KP_t(\tau)$ on a server s_t at time τ, a process p_i is the first process on the new current knot $KP_t(\tau)$ and $EV_t(\tau) = \langle st_{t1} \rangle$ where $st_{t1} = \tau$. Let int_k be time interval between an event e_k and a succeeding event e_{k+1} in $EV_t(\tau)$, i.e. $int_k = e_{k+1} - e_k$ ($1 \leq k \leq l - 1$ and $l \geq 2$). $int_1 = 0$ if $l = 1$. Let AC_{tik} be the amount of computation of a process p_i which is performed on a server s_t during the interval int_k. $N_t(int_k)$ indicates the number of processes which are being performed in an interval int_k. AC_{tik} is given as follows:

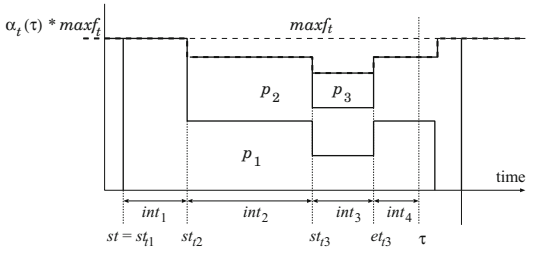


Fig. 3.12. Events in a current knot

$$AC_{tik} = (\varepsilon_t^{N_t(int_k)-1} \cdot maxf_t) \;/\; N_t(int_k) \cdot int_k. \tag{3.5}$$

For example, the number $N_t(int_2)$ of concurrent processes during the time interval int_2 is 2 in Figure 3.12. Then, AC_{t12} of a process p_1 in the time interval int_2 on the server s_t is $(\varepsilon_t \cdot maxf_t) / 2 \cdot int_2$.

Let $INT_t(\tau)$ be a set of time intervals $\{int_1, ..., int_{l-1}\}$ between time st and time τ on the current knot $KP_t(\tau)$. The total amount $TAC_{ti}(\tau)$ of computation which is performed on a server s_t between time st and τ for each process p_i in $KP_t(\tau)$ is calculated by the following procedure:

AmountOfPerformed($INT_t(\tau)$, $KP_t(\tau)$) {
 $TAC_t = \phi$;
 for every process p_i **in** $KP_t(\tau)$ {
  $TAC_{ti} = 0$;
  for every time interval int_k **in** $INT_t(\tau)$ {
   if $p_i \in N_t(int_k)$, $TAC_{ti} = TAC_{ti} + AC_{tik}$;
  }
 $TAC_t = TAC_t + \{TAC_{ti}\}$;
 }
 return(TAC_t);
}

In the procedure **AmountOfPerformed()**, the total amount TAC_{ti} of computation performed by each process p_i in $KP_t(\tau)$ between time st and τ on a server s_t is stored in a set TAC_t. Here, $TAC_t = TAC_{t1}, ..., TAC_{tl_t}$. Let $TAC_t[i]$ show TAC_{ti}.

The total amount of computation to be performed to terminate a process p_i is $maxf_t \cdot minT_{ti}$ on a server s_t. The residual computation quantity RCQ_{ti} shows how much computation has to be spent to terminate a process p_i on the server s_t after time τ. RCQ_{ti} is given as follows:

$$RCQ_{ti} = (maxf_t \cdot minT_{ti}) - TAC_{ti}. \tag{3.6}$$

A tuple $\langle i, RCQ_{ti} \rangle$ is a pair of an identifier i of a process p_i and a residual computation quantity RCQ_{ti} of the process p_i at time τ. RCQ_{ti} of each process p_i in $KP_t(\tau)$ can be calculated as follows:

CalcRCQ($INT_t(\tau)$, $KP_t(\tau)$) {
 $RCQ_t = \phi$;
 TAC_t = **AmountOfPerformed**($INT_t(\tau)$, $KP_t(\tau)$);
 for every process p_i **in** $KP_t(\tau)$ {
  $TAC_{ti} = TAC_t[i]$; /* TAC_{ti} of p_i in TAC_t; */
  $RCQ_{ti} = (maxf_t \cdot minT_{ti})$ - TAC_{ti};
  if $RCQ_{ti} > 0$, $RCQ_t = RCQ_t \cup \{\langle i, RCQ_{ti} \rangle\}$;
  /* If $RCQ_{ti} \leq 0$, p_i has terminated before time τ. */
 }
 return(RCQ_t);
}

In the procedure **CalcRCQ()**, every pair of an identifier i and a residual computation quantity RCQ_{ti} of a process p_i which has to be performed after time τ is stored in RCQ_t. Let $ET_t(\tau)$ be an estimated termination time of a current knot $KP_t(\tau)$ where a server s_t is selected for a process p_j at time τ. $ET_{p_j}(\tau)$ shows an estimated termination time of the process p_j on a server s_t at time τ. A pair of $ET_t(\tau)$ and $ET_{p_j}(\tau)$ are calculated as follows:

EstimateTermination($INT_t(\tau)$, $KP_t(\tau)$, p_j) {
 RCQ_t = **CalcRCQ**($INT_t(\tau)$, $KP_t(\tau)$);
 $RCQ_t = RCQ_t \cup \{\langle j, maxf_t \cdot minT_{tj} \rangle\}$; /*$\langle j, RCQ_{ti} \rangle$ of p_j is added into RCQ_t.*/
 SORT(RCQ_t); /*RCQ_t is sorted in ascending order of RCQ_{ti}.*/
 $ET_t(\tau) = \tau$;
 while ($RCQ_t \neq \phi$) { /* ····· (1) */
  $N_t(\tau) = |RCQ_t|$;
  $maxf_t(\tau) = \varepsilon_t^{N_t(\tau)-1} \cdot maxf_t$;
  $EstT_t = \phi$;
  for every element $\langle i, RCQ_{ti} \rangle$ **in** RCQ_t {
   $EstT_{ti} = (maxf_t(\tau)/N_t(\tau))^{-1} \cdot RCQ_{ti}$;
   $EstT_t = EstT_t \cup \{\langle i, EstT_{ti} \rangle\}$;
  }
  $minEstT_{ti}$ = **MIN**($EstT_t$);
  $ET_t(\tau) = ET_t(\tau) + minEstT_{ti}$;
  for every element $\langle i, EstT_{ti} \rangle$ **in** $EstT_t$ {
   if $EstT_{ti} = minEstT_{ti}$, {
    if $i = j$, $ET_{p_j}(\tau) = ET_t(\tau)$;
    $RCQ_t = RCQ_t$ - $\{\langle i, RCQ_{ti} \rangle\}$;
   }
   else {
    $\langle i, RCQ_{ti} \rangle$ = **search**(i, RCQ_t);

```
      newRCQ_ti = RCQ_ti - (minEstT_ti · maxf_t(τ) / N_t(τ));
      ⟨i, RCQ_ti⟩ = ⟨i, newRCQ_ti⟩;
    }
   }
  }
  return(⟨ET_t(τ), ET_pj(τ)⟩);
}
```

Suppose a server s_t is selected for a process p_4 at time τ in Figure 3.13. Here, we assume that $RCQ_t = \{\langle p_1,4\rangle, \langle p_4,8\rangle, \langle p_2,16\rangle\}$ according to the procedure **EstimateTermination()**. In addition, $maxf_t = 1$ and $\varepsilon_t = 0.99$. Here, the **while** loop is performed three times [see line (1)]. In the first round, $maxf_t(\tau) = \varepsilon_t^{3-1} \cdot 1 = 0.99^2 \cdot 1 = 0.98$, $EstT_t = \{\langle p_1,12.5\rangle, \langle p_4,25\rangle, \langle p_2,50\rangle\}$, and $ET_t(\tau) = \tau + 12.5$, respectively, as shown in Figure 3.13 (1). Then, RCQ_t is changed to $\{\langle p_4,4\rangle, \langle p_2,12\rangle\}$. In the second round, $maxf_t(\tau) = \varepsilon_t^{2-1} \cdot 1 = 0.99 \cdot 1 = 0.99$, $EstT_t = \{\langle p_4,8.1\rangle, \langle p_2,24.3\rangle\}$ and $ET_t(\tau) = \tau + 12.5 + 8 = \tau + 20.6$, respectively, as shown in Figure 3.13 (2). Here, the process p_4 is the new process for which the server s_t is selected at time τ. Therefore, $ET_{p_4}(\tau) = \tau + 20.6$. Then, RCQ_t is changed to $\{\langle p_2,8\rangle\}$. In the third round, $maxf_t(\tau) = \varepsilon_t^0 \cdot 1 = 1$, $EstT_t = \{\langle p_2,8\rangle\}$, and $ET_t(\tau) = \tau + 12.5 + 8.1 + 8 = \tau + 28.6$, respectively, as shown in Figure 3.13 (3). Hence, the estimated termination time $ET_t(\tau)$ of a current knot $KP_t(\tau)$ is $\tau + 28.6$ and the estimated termination time $ET_{p_4}(\tau)$ of the process p_4 is $\tau + 20.6$.

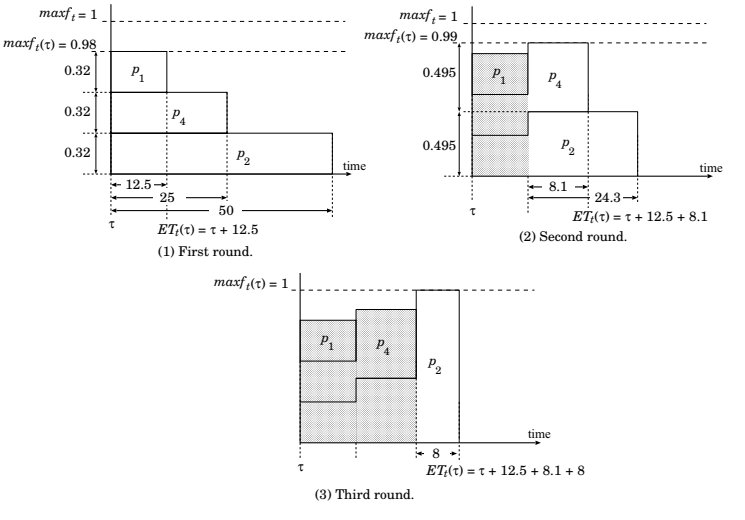


Fig. 3.13. Estimated termination time

3.5.1.2 Power Consumption Model

We discuss the power consumption model of a server s_t for CP applications. Here, we consider the following parameters:

- $E_t(\tau)$ = electric power consumption rate of a server s_t at time τ [W] ($t = 1$, â€¦, n).
- $maxE_t$ = maximum electric power consumption rate of a server s_t.
- $minE_t$ = minimum electric power consumption rate of a server s_t.
- $maxE = max(maxE_1, ..., maxE_n)$.
- $minE = min(minE_1, ..., minE_n)$.

We define the normalized power consumption rate ($NPCR$) $e_t(\tau)$ of a server s_t at time τ as follows:

$$e_t(\tau) = E_t(\tau) \,/\, maxE \ \ (\leq 1). \tag{3.7}$$

Let $mine_t$ and $maxe_t$ show the minimum power consumption rate $minE_t \,/\, maxE$ and the maximum power consumption rate $maxE_t \,/\, maxE$ of a server s_t, respectively. Here, $e_t(\tau)$ shows a ratio of how much a server s_t consumes electric power to a server which mostly consumes the electric power at time τ. If the fastest server s_t maximumly spends the electric power with the maximum clock frequency, $e_t(\tau) = maxe_t = 1$. We define a simple power consumption model for a server s_t as follows:

$$e_t(\tau) = \begin{cases} maxe_t \text{ if } N_t(\tau) \geq 1. \\ mine_t \text{ otherwise.} \end{cases} \tag{3.8}$$

This means, if at least one process is performed on a server s_t, the electric power is maximally consumed on the server s_t. If no process is performed on the server s_t, $e_t(\tau) = mine_t$. The normalized power consumption $NPC_t(\tau_1, \tau_2)$ [Ws] of a server s_t from time τ_1 to time τ_2 is given as follows:

$$NPC_t(\tau_1, \tau_2) = \int_{\tau_1}^{\tau_2} e_t(\tau) d\tau. \tag{3.9}$$

3.5.2 CM Applications

In this section, we discuss the transmission and power consumption models for communication (CM) applications.

3.5.2.1 Transmission Models

First, let $tr_{ts}(\tau)$ be a transmission rate of a server s_t to transmit a file f_s to a client c_s at time τ. Suppose a server s_t concurrently sends files f_1, ..., f_m to a set $CT_t(\tau)$ of clients c_1, ..., c_m at transmission rates $tr_{t1}(\tau)$, ..., $tr_{tm}(\tau)$ ($m \geq 1$), respectively, at

time τ. b_{ts} shows the maximum bandwidth [bps] between a server s_t and a client c_s. Let $Maxtr_t$ be the maximum transmission rate [bps] of the server s_t ($\leq b_{ts}$). Here, the total transmission rate $tr_t(\tau)$ of the server s_t at time is given as $tr_t(\tau) = tr_{t1}(\tau) + \cdots + tr_{tm}(\tau)$. Here, $0 \leq tr_t(\tau) \leq Maxtr_t$.

Each client c_s receives messages from servers at receipt rate $rr_s(\tau)$ at time τ. Let $Maxrr_s$ indicate the maximum receipt rate of the client c_s. We assume each client c_s receives a file from at most one server at rate $Maxrr_s$ ($= rr_s(\tau)$). The server s_t allocates each client c_s with transmission rate $tr_{ts}(\tau)$ so that $tr_{ts}(\tau) \leq Maxrr_s$ at time τ.

Let TR_{ts} be the total transmission time of a file f_s from a server s_t to a client c_s. If the server s_t sends files to other clients concurrently with the client c_s, the transmission time TR_{ts} is increased. Let $minTR_{ts}$ show the minimum transmission time $|f_s| / min(Maxrr_s, Maxtr_t)$ [sec] of a file f_s from a server s_t to a client c_s where $|f_s|$ indicates the size [bit] of the file f_s. $TR_{ts} \geq minTR_{ts}$. Suppose the server s_t starts and ends transmitting a file f_s to the client c_s at time st and et, respectively. Here, $= |f_s|$ and the transmission time TR_{ts} is (et - st). If the server s_t sends only the file f_s to the client c_s at time τ, $tr_{ts}(\tau) = min(Maxrr_s, Maxtr_t)$ [bps]. The transmission laxity $lt_{ts}(\tau)$ is $|f_s|$ - $\int_{\tau}^{et} tr_{ts}(\tau)\, d\tau$ [bit] at time τ, i.e. how many bits of a file f_s the server s_t still has to transmit to the client c_s at time τ. First, we consider a model where a server s_t satisfies the following properties:

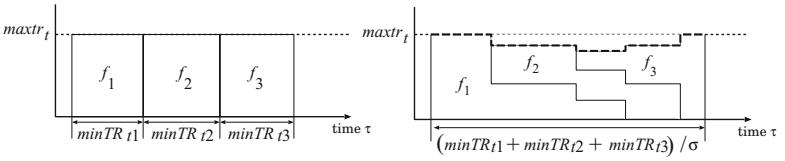
[Server-bound model] If $Maxrr_1 + \cdots + Maxrr_m \geq \sigma_t(\tau) \cdot Maxtr_t, \Sigma_{c_s \in CT_t(\tau)}\, tr_{ts}(\tau) = \sigma_t(\tau) \cdot Maxtr_t$ for every time τ.

Here, $\sigma_t(\tau)$ (≤ 1) is the transmission degradation factor of a server s_t. Here, we assume $\sigma_t(\tau) = \gamma^{1-|CT_t(\tau)|}$ ($0 < \gamma \leq 1$) at time τ. The more number of clients a server s_t transmits, the longer it takes. The effective transmission rate of the server s_t is $\sigma_t(\tau) \cdot Maxtr_t$.

Suppose a server s_t sends three files f_1, f_2, and f_3 to clients c_1, c_2, and c_3, respectively. First, suppose that the server s_t serially sends the files f_1, f_2, and f_3 to the clients c_1, c_2, and c_3, i.e. $et_{t1} = st_{t2}$ and $et_{t2} = st_{t3}$ as shown in Figure 3.14 (1). Here, the transmission time TR_t is et_{t3} - $st_{t1} = minTR_{t1} + minTR_{t2} + minTR_{t3}$. Next, suppose the server s_t first starts transmitting the file f_1 at time st_{t1} and terminates transmitting the file f_3 at time et_{t3} as shown in Figure 3.14 (2). At time st_{t2}, the server s_t starts transmitting the file f_2. Here, $CT_t(\tau) = 2$ and $\sigma_t(\tau) = \gamma^{1-2} = \gamma^{-1}$ for $st_{t2} \leq \tau \leq st_{t3}$. Then, at time st_{t3}, the server s_t starts transmitting the file f_3. Here, $CT_t(\tau) = 3$ and $\sigma_t(\tau) = \gamma^{1-3} = \gamma^{-2}$ for $st_{t3} \leq \tau \leq et_{t1}$. Here, $TR_t = et_{t3}$ - $st_{t1} = (minTR_{t1} + minTR_{t2} + minTR_{t3}) / \sigma$. $\sigma = [(st_{t2}$ - $st_{t1}) + (st_{t3}$ - $st_{t2})\gamma + (et_{t1}$ - $st_{t3})\gamma^2 + (et_{t2}$ - $et_{t1})\gamma + (et_{t3}$ - $et_{t2})] / (et_{t3}$ - $st_{t1})$.

On the other hand, we consider another environment where some client c_s cannot receive a file from a server s_t at maximum transmission rate $Maxtr_t$, i.e. $Maxrr_s < Maxtr_t$. Hence, the transmission rate $tr_t s(\tau)$ is the maximum receipt rate $Maxrr_s$.

[Client-bound model] If $Maxrr_1 + \cdots + Maxrr_m \leq \sigma_t(\tau) \cdot Maxtr_t, \Sigma_{c_s \in CT_t(\tau)}\, tr_{ts}(\tau) = Maxrr_1 + \cdots + Maxrr_m$ for every time τ.



(1) Serial transmission. (2) Parallel transmission.

Fig. 3.14. Transmission time

3.5.2.2 Power Consumption Model

Let $PT_t(\tau)$ show the electric power consumption rate [W] of a server s_t for transmitting files to clients at time τ ($t = 1, ..., n$). $maxE_t$ and $minE_t$ indicate the maximum and minimum electric power consumption of a server s_t, respectively. Here, $minE_t \leq PT_t(\tau) \leq maxE_t$. $maxE$ and $minE$ show $max(maxE_1, ..., maxE_n)$ and $min(minE_1, ..., minE_n)$, respectively. Here, we assume that only file transfer applications are performed on each server s_t. The electric power consumption rate $PT_t(\tau)$ of a server s_t at time τ is given as follows:

$$PT_t(\tau) = PC_t(tr_t(\tau)) = \beta_t(|CT_t(\tau)|) \cdot \delta_t \cdot tr_t(\tau) + minE_t. \tag{3.10}$$

The power consumption $TPC_t(\tau_1, \tau_2)$ [Ws] of a server s_t from time τ_1 to time τ_2 is given as follows:

$$TPC_t(\tau_1, \tau_2) = \int_{\tau_1}^{\tau_2} PT_t(\tau) d\tau. \tag{3.11}$$

3.5.3 ST Applications

In this section, we discus the models for storage (ST) application.

3.5.3.1 Access Model

In the ST applications, each process p_i is characterized in terms of a process type $tp_i \in \{C, R, W\}$ and amount l_i [bit] computation and access. If a process p_i is a C process, l_i stands for the total amount of computation of a C process p_i. Let ET_{ti} denote the execution time to perform a process p_i on a server s_t. Suppose a C process p_i is performed on a server s_t without any other process. Here, it takes MET_{ti} [sec] to perform the process p_i. The more number of processes are performed, the longer it takes to perform each process p_i, i.e. $MET_{ti} \leq ET_{ti}$. The maximum computation rate $maxF_{ti}$ of a process p_i on a server s_t is $1/MET_{ti}$ [1/sec]. It is noted $maxF_{ti} = maxF_{tj}$ for every pair of processes p_i and p_j. $maxF_t$ shows the maximum

computation rate of the server s_t, i.e. $maxF_t = maxF_{ti}$ for every process p_i. The computation rate $F_{ti}(\tau)$ of a process p_i on a server s_t at time τ shows how many percentages of the total computation of the process p_i is performed at time τ as discussed in the preceding subsection. If the process p_i is performed without any process on the server s_t, $F_{ti}(\tau) = 1/MET_{ti}$ [1/sec] which is the maximum computation rate $maxF_{ti}$. The maximum computation rate $F_t(\tau)$ of the server s_t is $maxF_{ti}$ for process p_i, $F_t(\tau) = maxF_t$. In this paper, we assume the computation rate is fairly allocated to every concurrent process at time τ, i.e. $F_{ti}(\tau) = \alpha_t \cdot maxF_t/|NC_t(\tau)|$ for every process p_i in $NC_t(\tau)$. Here, $\alpha_t = \epsilon$ where $0 < \epsilon \leq 1$.

In an R or W process, l_i indicates the size of a file to be read or written. An R process p_i reads a file of size s_i while a W process p_j writes a file of size l_j [bit] in a storage drive of a server s_t. The read access rate AR_{ti} and write access rate AW_{tj} [bps] of the R and W processes p_i and p_j on the server s_t are given as l_i/ET_{ti} [bps] and l_j/ET_{tj} [bps], respectively. In the experimental results shown in Figure 3.8, the read rate AR_{ti} of an R process p_i is 8.22 M [bps] and the write rate AW_{sj} of a W process p_j is 10.65 M [bps] for HDD. It is also implied from the experimental results that $AR_{ti} = AR_{tj}$ for every pair of R processes p_i and p_j and $AW_{ti} = AW_{tj}$ for every pair of W processes p_i and p_j even if multiple processes are concurrently performed on the server s_t. AR_t and AW_t indicate the maximum read and write rates of the server s_t to read and write files, respectively. Let $a_{tR}(\tau)$ and $a_{tW}(\tau)$ be the read and write rates for R and W processes to read and write files on a server s_t at time τ, respectively.

According to the experimental results, the read access rate $a_{tR}(\tau)$ and write access rate $a_{tW}(\tau)$ are obtained as follows:

$$a_{tR}(\tau) = \begin{cases} AR_t & \text{if at least one } R \text{ process is performed on a server } s_t \text{ at time } \tau. \\ 0 & \text{otherwise.} \end{cases}$$

$$a_{tW}(\tau) = \begin{cases} AW_t & \text{if at least one } W \text{ process is performed on a server } s_t \text{ at time } \tau. \\ 0 & \text{otherwise.} \end{cases}$$

Multiple processes are concurrently performed on a server s_t. We discuss how long it takes to perform processes on a server s_t. Let K_t be a knot of processes $p_{t1}, \cdots, p_{tm}$ on a server s_t. The starting time st_{t1} of the process p_1 is the minimum and the ending time et_{tm} is the maximum in the knot K_t. That is, no process in the knot K_t is performed before time st_{t1} and after time et_{tm}. At least one process in the knot K_t is performed anytime between time st_{t1} and et_{tm}. The execution time T_t to perform every process in the knot K_t is $et_{tm} - st_{t1}$.

Suppose processes p_1, p_2, p_3, p_4, and p_5 are interleaved, i.e. concurrently performed as shown in Figure 3.15. Here, a knot $K_t(p_1) = \{p_1, p_2, p_3, p_4, p_5\}$. The execution time T_t of the knot $K_t(p_1)$ is $et - st$. Next, suppose the processes are serially performed, i.e. the process p_1 starts at time t'_0 and ends at time t'_1 and the process p_2 starts at t'_1 and ends at t'_2 as shown in Figure 3.16. The execution time ET_1 to perform the process p_1 with no other concurrent process is $t'_1 - t'_0$. $ET_2 = t'_2 - t_1$, $ET_3 = t'_3 - t'_2$, $ET_4 = t'_4 - t'_3$, and $ET_5 = t'_5 - t'_4$. Here, the execution time $T_t = et$ - st

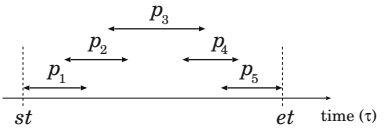


Fig. 3.15. Interleaved execution of processes

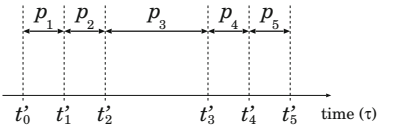


Fig. 3.16. Serial execution of processes

of the knot of Figure 3.15 is the same as $ET_1 + ET_2 + ET_3 + ET_4 + ET_5$ where the processes are serially performed as shown in Figure 3.16. This means, the execution time of a knot $\{p_{t1}, \cdots, p_{tm}\}$ is the same, i.e. independently of what schedule the processes $p_{t1}, \cdots, p_{tm}$ are performed in.

3.5.3.2 Power Consumption Model

We obtain the power consumption model by abstracting most dominating parameters of the power consumption of a server s_t from the experimental results presented in the preceding section. Let $CP_t(\tau)$ be a collection of processes which are being performed on a server s_t at time τ. Let $CRP_t(\tau)$, $CWP_t(\tau)$, and $CCP_t(\tau)$ ($\subseteq CP_t(\tau)$) be pairwise disjointing collections of R, W, and C processes, respectively, which are concurrently performed on a server s_t at time τ. Here, $CP_t(\tau) = CRP_t(\tau) \cup CWP_t(\tau) \cup CCP_t(\tau)$. Let $N_t(\tau)$, $NC_t(\tau)$, $NR_t(\tau)$, and $NW_t(\tau)$ be the number of processes, $|CP_t(\tau)|$, $|CCP_t(\tau)|$, $|CRP_t(\tau)|$, $|CP_t(\tau)|$, and $|CWP_t(\tau)|$, respectively.

The power consumption rate $e_{tA}(\tau)$ [W] of a server s_t to perform processes at time τ is given for an environment $A \in \{C, W, R, RW, CR, CW, CRW\}$ as follows:

$$e_{tA}(\tau) = \begin{cases} maxR_t & \text{if only and at least one } R \text{ process is performed on a server } s_t \text{ at time } \tau \\ maxC_t & \text{if only and at least one } C \text{ process is performed on a server } s_t \text{ at time } \tau \\ maxCR_t & \text{if only and at least one } C \text{ process and at least one } R \text{ process are performed on a server } s_t \text{ at time } \tau \\ minE_t & \text{if no process is performed on a server } s_t \text{ at time } \tau. \end{cases}$$

That is, the power consumption rate $e_{tA}(\tau)$ is a binary function which takes either $maxA_t$ or $minE_t$ depending on the environment A as shown in Figure 3.17. The server s_t consumes the minimum power consumption rate $minE_t$ if no process is performed on the server s_t. The minimum power consumption rate $minE_t$ is independent of an environment type A but depends on the server s_t.

Let D_t show difference to the maximum power consumption rate $maxR_t$ ($= maxW_t = maxRW_t$) to $minE_t$, i.e. $D_t = maxR_t - minE_t$. Let C_t be $maxC_t - minE_t$. The maximum power consumption rates [W] are given in terms of $minE_t$, D_t, and C_t as follows [Figure 3.18] :

- $maxC_t = maxE_t + C_t$
- $maxR_t \ (= maxW_t = maxRW_t) = maxE_t + D_t$.
- $maxCR_t \ (= maxCW_t = maxCWR_t) = minE_t + D_t + C_t$.

The power consumption rate $e_t(\tau)$ [W] of a server s_t at time τ is thus given as follows :

$$e_t(\tau) = minE_t + d_t(\tau) \cdot D_t + c_t(\tau) \cdot C_t. \tag{3.12}$$

Here, $d_t(\tau)$ and $c_t(\tau)$ show whether an R or W process exists or not and a C process exists or not, respectively, as follows :

$$d_t(\tau) = \begin{cases} 1 & \text{if } NR_t(\tau) \geq 1 \text{ or } NW_t(\tau) \geq 1. \\ 0 & \text{otherwise.} \end{cases}$$

$$c_t(\tau) = \begin{cases} 1 & \text{if } NC_t(\tau) \geq 1 \\ 0 & \text{otherwise.} \end{cases}$$

The power consumption rate $e_t(\tau)$ of a server s_t at time τ depends on the factors $d_t(\tau)$ and $c_t(\tau)$. $d_t(\tau) = 1$ if at least one R or W process is performed on the server s_t at time τ. If neither R nor W processes are performed, $d_t(\tau) = 0$. $c_t(\tau) = 1$ if at least one C process is performed on the server s_t at time τ. Otherwise, $c_t(\tau) = 0$.

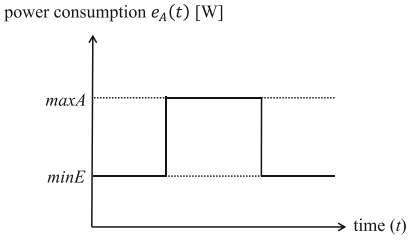


Fig. 3.17. Power consumption rate

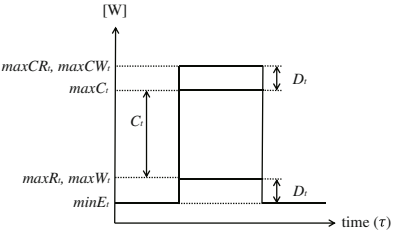


Fig. 3.18. Maximum power consumption rate

According to the experimental results, $minE_t = 100$ [W], $D_t = 8$ [W], and $C_t = 68$ [W] in the server s_t whose specification is shown in Table 1.

The total power consumption E_t (s_t, e_t) [Ws] of a server s_t from time st to time et is given as follows:

$$E_t(s_t, e_t) = \int_{st}^{et} e_t(\tau) \ d\tau. \tag{3.13}$$

For example, if only R processes are performed on a server s_t from time st to time et, the power power consumption E_t (s_t, e_t) $= maxR_t \cdot (et - st) = (minE_t + D_t) \cdot (et - st)$ [Ws]. If only C and at least one process is performed on a server s_t from time st to time et, E_t (s_t, e_t) $= maxC_t \cdot (et - st) = (minE_t + C_t) \cdot (et - st)$.

Figure 3.19 shows the power consumption rate $e_t(\tau)$ [W] of a server s_t at time τ where types of processes are concurrently performed as shown in Figure 3.15. First, the server s_t consumes the minimum power consumption rate $minE_t$ since no process is performed. Then, an R process p_1 starts at time st. $e_t(st) = maxR_t = minE_t + D_t$. Next, the W process p_2 starts at time st. $e_t(\tau) = maxRW_t = maxR_t$ $(st \leq \tau < \tau_1)$. Then, a C process p_3 starts at time t_1. Here, $e_t(\tau) = maxCRW_t = minE_t + D_t + C_t$ $(\tau_1 \leq \tau < \tau_2)$ since three types R, W, and C processes p_1, p_2, and p_3 are concurrently performed, respectively. The R process p_1 ends at time t_2. $e_t(\tau) = maxC_t$ [W] since only a C process is performed $(\tau_2 \leq \tau < \tau_3)$. At time t_3, a W process p_4 starts and then an R process p_6 starts. $e_t(\tau) = maxCRW_t$ $(\tau_3 \leq \tau < \tau_4)$. The C process p_3 ends at time t_4. $e_t(\tau) = maxRW_t$ $(\tau_4 \leq \tau < et)$. All the processes end at time et and the power consumption $e_t(\tau) = minE_t$ $(\tau \geq et)$. The hatching area shows the total amount of electric power consumed on the server s_t in Figure 3.19. The total power consumption E_t (s_t, e_t) of the server s_t to perform the processes p_1, p_2, p_3, p_4, and p_5 is $\int_{st}^{et} e_t(\tau)d\tau = \int_{st}^{\tau_1} e_{tRW}(\tau)d\tau + \int_{\tau_1}^{\tau_2} e_{tCRW}(\tau)d\tau + \int_{\tau_2}^{\tau_3} e_{tC}(\tau)d\tau + \int_{\tau_3}^{\tau_4} e_{tCRW}(\tau)d\tau + \int_{\tau_4}^{\tau_5} e_{tRW}(\tau)d\tau =$

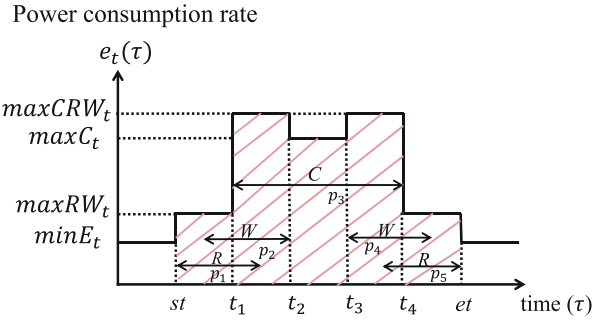


Fig. 3.19. Power consumption rate

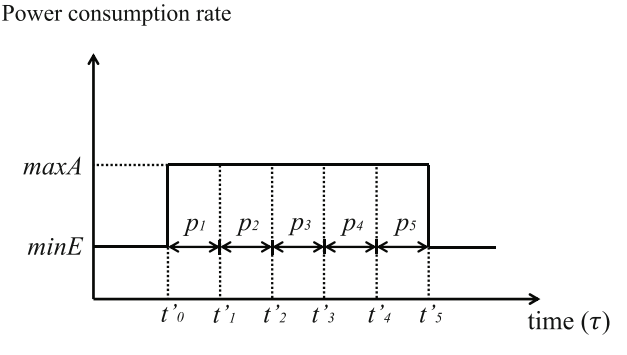


Fig. 3.20. Power consumption rate

$minE_t \cdot (et - st) + D_t \cdot (\tau_2 - st) + C_t \cdot (\tau_3 - \tau_2) + (D_t + C_t) \cdot (\tau_4 - \tau_3) + D_t \cdot (\tau_5 - \tau_4) =$
$minE_t \cdot (et - st) + D_t \cdot (\tau_2 - st + \tau_5 - \tau_3) + C_t \cdot (\tau_4 - \tau_1)$ [Ws].

Figure 3.20 shows the power consumption rate of the server s_t for the serial schedule of Figure 3.16.

3.6 Server Selection Algorithms

A client has to obtain the required service from one server in a set S of possible servers. We discuss algorithms to select one server in the server set S for each type of application.

3.6.1 CP Applications

Suppose a process p_i is issued to a load balancer K at time τ. First, the *power consumption laxity* (*PCL*) $le_t(\tau)$ [Ws] of each server s_t is estimated [Figure 3.21]:

$$le_t(\tau) = maxe_t \cdot (ET_t(\tau) - \tau). \tag{3.14}$$

In the *power consumption laxity-based* (PCLB) algorithm, a server s_t which consumes the minimum power consumption is selected for a process p_i, i.e. a server s_t whose $le_t(\tau)$ is minimum is selected in the server set S. At time τ, a server s_t is selected for a process p_i by the following procedure **PCLB**:

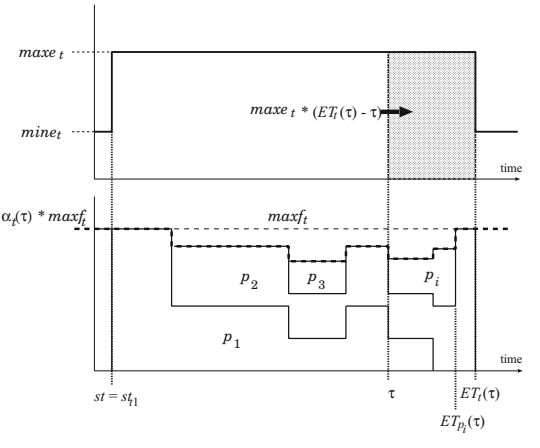


Fig. 3.21. Power consumption laxity (PCL)

PCLB(τ, S, p_i) {
 $LE_t = \phi$;
 for each server s_t **in** S {
  $\langle ET_t(\tau), ET_{p_i}(t) \rangle$ = **EstimateTermination**($INT_t(\tau)$, $KP_t(\tau)$, p_i);
  $le_t(\tau) = maxe_t \cdot (ET_t(\tau)$ - $\tau)$;
  $LE_i = LE_i + \{le_i(t)\}$;
 }
 $server = s_t$ where $le_t(\tau)$ is minimum in LE_t;
 return($server$);
}

3.6.2 CM Applications

3.6.2.1 Transmission Rates

At time τ, the maximum transmission rate $maxtr_t(\tau)$ of a server s_t depends on the transmission degradation factor $\sigma_t(\tau) = \gamma^{1-|CTt(\tau)|}$ of the server s_t, i.e. the number of clients to which the server s_t concurrently transmits files at time τ. Each time a new request is issued by a client c_s and a current request for a client c_s is terminated at time τ, $CT_t(\tau) = CT_t(\tau) \cup \{c_s\}$ and $CT_t(\tau) = CT_t(\tau)$ - $\{c_s\}$, respectively. Here, the maximum transmission rate $maxtr_t(\tau)$ of a server s_t at time τ is calculated as $\gamma^{1-|CTt(\tau)|} \cdot Maxtr_t$. Here, $0 < \gamma \leq 1$. In order to more efficiently utilize the total transmission rate $maxtr_t(\tau)$, the transmission rate $tr_{ts}(\tau)$ for a client c_v is calculated as follows:

CalcTR(s_t, c_v) {
 $V = 0$; $R = 0$; $TS = maxtr_t(\tau) / CT_t(\tau)$;
 for each client c_s **in** $CT_t(\tau)$ {
  if $Maxrr_s \leq TS$ { $tr_t(\tau) = TS$ and $R = R + (TS$ - $Maxrr_s)$; }
  else { $tr_t(\tau) = Maxrr_s$ and $V = V + (Maxrr_s$ - $TS)$; }
 }
 for each client c_s **in** $CT_t(\tau)$ {
  if $tr_{ts}(\tau) < Maxrr_s$ { $tr_{ts}(\tau) = tr_{ts}(\tau) + V \cdot (Maxrr_s$ - $tr_{ts}(\tau)) / R$; }
 }
 return($tr_{tv}(\tau)$);
}

In the **CalcTR()** algorithm, the unused part of the maximum transmission rate of the server s_t for the client c_s (= $tr_ts(\tau)$ - $Maxrr_s$) can be used for other client.

3.6.2.2 Extended Power Consumption-Based (EPCB) Algorithm

We discuss how a load balancer K selects a server s_t for a client c_s in the server set S. In the *extended power consumption-based* (EPCB) algorithm, a server s_t is selected for the client c_s where the power consumption to transmit files to clients is the smallest. $lt_{ts}(\tau)$ shows the transmission laxity of a file f_s on a server s_t as discussed in the preceding subsection. Here, the total transmission laxity of a server s_t at time τ is $\Sigma_{c_s \in CT_t(\tau)} lt_{ts}(\tau)$. Here, the estimated change $EPT_{ts}(\tau)$ [Ws] of power consumption of a server s_t for transmitting a file f to a client c_s at time τ when s_t starts transmitting the file f is defined as follows:

$$\begin{aligned} EPT_{ts}(\tau) &= (\Sigma_{c_s \in CT_t(\tau)} lt_{ts}(\tau) \ / \ tr_{ts}(\tau)) \cdot \beta_t(|CT_t(\tau)|) \cdot \delta_t \cdot tr_{ts}(\tau) \\ &= \Sigma_{c_s \in CT_t(\tau)} lt_{ts}(\tau) \cdot \beta_t(|CT_t(\tau)|) \cdot \delta_t. \end{aligned} \tag{3.15}$$

Here, a server s_t is selected for a client c_s in the EPCB algorithm by using $EPT_{ts}(\tau)$ at time τ as follows:

EPCB(c_s, τ) {
  $server = \phi$; $EPT = 0$;
  for each s_t **in** S {
    $EPT_{ts}(\tau) = \Sigma_{c_s \in CT_t(\tau)} lt_{ts}(\tau) \cdot \beta_t(|CT_t(\tau)|) \cdot \delta_t$
    if $server = \phi$, {
      $server = s_t$;
      $EPT = EPT_{ts}(\tau)$;
    }
    else {
      if $EPT > EPT_{ts}(\tau)$, {
        $EPT = EPT_{ts}(\tau)$;
        $server = s_t$;
      }
    }
  return($server$);
}

For example, there are a pair of servers s_1 and s_2. The power consumption coefficients δ_1 and δ_2 to transmit one [Mbit] for one client of servers s_1 and s_2 are 0.09 and 0.07 [W/Mbit], respectively. The server s_1 is selected by a client c_1 ($CT_1(\tau) = \{c_1\}$) and s_2 is selected by a client c_2 ($CT_2(\tau) = \{c_2\}$) at time τ, respectively. Suppose a client c_3 issues a new request to transmit a file f whose size is one Gbytes to a load balancer K at time τ. The transmission laxities $lt_{11}(\tau)$ and $lt_{22}(\tau)$ are 0.1 and 0.9 [GByte], respectively. Suppose the increasing rates $\beta_1(2)$ and $\beta_2(2)$ of the power consumption of the servers s_1 and s_2 are 1.2 and 1.1, respectively. Here, a server s_t which has the minimum value of the formula $\Sigma_{c_s \in CT_t(\tau)} lt_{ts}(\tau) \cdot \beta_t(|CT_t(\tau)|) \cdot \delta_t$ is selected in the EPCB algorithm. Here, $\Sigma_{c_s \in CT_1(\tau)} lt_{1s}(\tau) \cdot \beta_1(|CT_1(\tau)|) \cdot \delta_1 = 1.1 \cdot 1.2 \cdot 0.09 = 0.119$. $\Sigma_{c_s \in CT_2(\tau)} lt_{2s}(\tau) \cdot \beta_2(|CT_2(\tau)|) \cdot \delta_2 = 1.9 \cdot 1.1 \cdot 0.07 = 0.146$ [Ws]. Therefore, the server s_1 is selected for a client c_3 in the EPCB algorithm.

3.6.3 ST Applications

We discuss how to select a server in a set of storage servers for each request from clients. Suppose there are a set S of servers $s_1, \cdots, s_n$ $(n \geq 1)$. A client c_i first issues a data manipulation request r_i to a load balancer L as shown in Figure 3.22. The load balancer L selects one server s_t in the server set S and forwards the request r_i to the server s_t. Then, a process p_{ti} is created for the request r_i and is performed on the server s_t. The server s_t sends a reply of the request r_i to the client c_i on completion of the process p_{ti}.

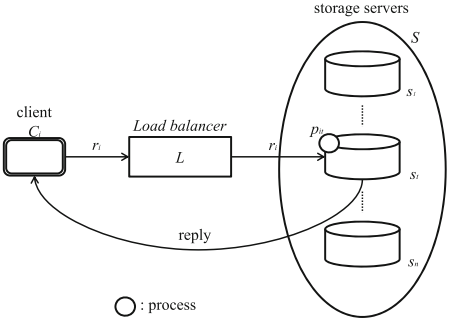


Fig. 3.22. Servers and clients

Each server s_t satisfies the access rate model if a smaller number of processes are performed than some number $maxN_t$. A server s_t is *overloaded* at time τ if $N_t(\tau) > maxN_t$. If overloaded, the access rate model does not hold. According to the experimental results, $maxN_t$ is 120 for the server s_t of Table 1. If a server s_t is overloaded, the load balancer L selects another server s_u which is not overloaded. In one algorithm based on the round robin (RR) algorithm [41], the load balancer L selects a server s_t in the server set S as follows:

I. First, $t = 1$.
II. If a server s_t is not overloaded, select the server s_t.
III. If s_t is overloaded, $t = t + 1$ and go to 2.

The load balancer L forwards the request r_i to the server s_t. Then, the request r_i is performed as a process p_{ti} on the server s_t. In the algorithm, it is critical how the servers are totally ordered in the server set S. The servers are totally ordered in the maximum power consumption rate as $maxCRW_1 \leq maxCRW_2 \leq \cdots \leq maxCRW_n$. That is, if a server s_t consumes the smaller electric power than another server s_u, $t < u$, i.e. the server s_t precedes s_u $(s_t \rightarrow s_u)$.

We assume every process is one of the application process types R, W, and C. A pair of servers p_t and p_u in the server set S are ordered in the following two types of relations $\rightarrow_{RW}$ and $\rightarrow_C$:

I. A server s_t precedes another server s_u with respect to R and W processes ($s_t \rightarrow_{RW} s_u$) if $maxR_t < maxR_u$ or $AR_t \geq AR_u$ if $maxR_t = maxR_u$, i.e. $maxW_t = maxW_u$.
II. A server s_t precedes another server s_u with respect to C processes ($s_t \rightarrow_C s_u$) iff $maxC_t < maxC_u$ or s_t is faster than s_u if $maxC_t = maxC_n$.

For example, a server s_t precedes another server s_u ($s_t \rightarrow_{RW} s_u$) if the maximum power consumption rate $maxR_t$ of the server s_t to read a file is smaller than $maxR_u$. Next, suppose $maxR_t = maxR_u$. If the maximum read access rate AR_t of the server s_t is larger than AR_u, $s_t \rightarrow_{RW} s_u$. If the maximum power consumption rate $maxC_t$ is smaller than $maxC_u$ ($maxC_t < maxC_u$), a server s_t precedes the server s_u ($s_t \rightarrow_C s_u$). If a pair of the servers s_t and s_u consume the same power ($maxC_t = maxC_u$), $s_t \rightarrow s_u$ if s_t is faster than s_u. Let S_{RW} and S_C be a pair of ordered sets $\langle S, \rightarrow_{RW} \rangle$ and $\langle S, \rightarrow_C \rangle$, respectively.

The load balancer L selects a server s_t in the ordered sets S_{RW} and S_C as follows:

I. Initially, $S'_{RW} = S_{RW}$ and $S'_C = S_C$.
II. $A = RW$ if a request r_i is R or W, $A = C$ if r_i is C.
III. Select a server s_t such that $s_t \rightarrow_A s_u$ for every other server s_u in the set S'_A.
IV. If the server s_t is not overloaded, select the server s_t and $S'_A = S_A - \{s_t\}$.
V. Otherwise (s_t is overloaded), $S'_A = S'_A - \{s_t\}$. go to (3).

If the server s_t selected for a request type A at step (2) is overloaded, a server s_u such that $s_t \rightarrow_A s_u$ but three is no server s_v such that $s_t \rightarrow_A s_v \rightarrow_A s_u$ is selected in the set S'_A. Then, the request r_i from a client c_i is forwarded to the server s_u.

As discussed in the power consumption model, if both C and R/W processes are performed on a server s_t, the most of power $maxCRW_t$ is consumed on the server s_t.

Hence, if C processes are performed on the server s_t, an R/W request is not issued to the server s_t, $S'_{RW} = S'_{RW} - \{s_t\}$ another server is selected in S'_{RW} at step (3). Otherwise, the server s_t is taken. Thus, C and R/W requests are not sent to the same server.

The server sets S'_{RW} and S'_C are collections of servers which are not overloaded and to be selected as RW and C servers, respectively. If a server s_t is selected in the server sets S'_{RW} and S'_C according to the selection algorithm and is overloaded, the server s_t is removed from the sets S'_{RW} and S'_C, respectively. For each overloaded server s_t, if RW and C processes are ended, s_t is added to the set S'_{RW} and S'_C again. $S'_{RW} = S_{RW} \cup \{S_t\}$ and $S'_C = S'_C \cup \{S_t\}$, respectively.

3.7 Evaluation

We evaluate the server selection algorithms for CP and CM applications in terms of the total power consumption.

3.7.1 CP Applications

We evaluate the PCLB algorithm in terms of total power consumption compared with the RR algorithm through the simulation. There are ten servers s_1, ..., s_{10} as shown in Table 3.5, $S = \{s_1, ..., s_{10}\}$. The servers in the server set S are ordered with respect to the processing speed, i.e. maximum computation rate. Here, the server s_1 is the fastest server and the server s_{10} is the slowest server. The *NCR* $maxf_t$ of each server s_t is randomly selected between 1 and 0.7 [1/sec]. The maximum computation rate $maxf_1$ of the fastest server s_1 is 1.0 and $maxf_{10}$ of the slowest server s_{10} is 0.7. The computation degradation rate ε_t of each server s_t is randomly selected between 0.99 and 0.95. The minimum power consumption rate $minE_t$ of each server s_t is randomly selected between 85 and 150 [W]. The maximum power consumption rate $minE_t$ of each server s_t is randomly selected between 1.2 times and 1.4 times of minimum power consumption rate $minE_t$. For example, the maximum *NPCR* $maxe_1$ = 1 and minimum *NPCR* $mine_1$ = 0.76 for the fastest server s_1.

Totally 600 number of processes, $P = \{p_1, ..., p_{600}\}$ are issued to the servers in the server set S for 1,000 seconds. The minimum computation time $minT_i$ of each process p_i on the fastest server is randomly selected between 2 and 10 [sec] as a real number. For example, if $minT_i$ of a process p_i is 5.24 [sec], it takes six seconds for the fastest server s_1 to perform the process p_i. Here, $minT_{ti} = minT_i / maxf_t$ for each server s_t. The starting time of each process is randomly selected between 0 and 1,000 [sec] in the simulation time. In simulations, the PCLB and RR algorithms are performed on the same traffic pattern. The maximum number of processes starting at the same time is 5 and the average number is 0.6 at each second.

Table 3.5. Servers

Server	s_1	s_2	s_3	s_4	s_5
$maxf_t$	1.000	0.998	0.859	0.840	0.839
$minE_t$ ($mine_t$)	149 (0.76)	141 (0.72)	133 (0.68)	129 (0.65)	108 (0.55)
$maxE_t$ ($maxe_t$)	197 (1)	186 (0.94)	160 (0.81)	156 (0.79)	136 (0.69)

Server	s_6	s_7	s_8	s_9	s_{10}
$maxf_t$	0.833	0.729	0.716	0.713	0.700
$minE_t$ ($mine_t$)	98 (0.5)	139 (0.71)	147 (0.75)	108 (0.55)	96 (0.49)
$maxE_t$ ($maxe_t$)	125 (0.63)	182 (0.92)	196 (0.99)	136 (0.69)	123 (0.62)

In the RR algorithm, the servers are ordered as s_1, ..., s_{10}. A request is first issued to the fastest server s_1. If the server s_1 is overloaded, requests are issued to the server s_2. Thus, requests are issued to the servers in the set S.

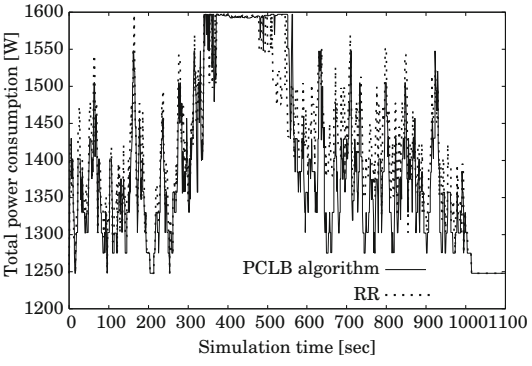


Fig. 3.23. Total power consumption [Ws]

Figure 3.23 shows the total power consumption [Ws] of the servers in the server set S at each time τ. Table 3.6 shows the total power consumption which is obtained by subtracting the power consumption of servers in the idle state from the total power consumption during the simulation, i.e. the power consumption for only performing totally 600 processes. In the PCLB algorithm, a server s_t which minimally consumes the power consumption for each process p_i is selected. The total power consumption of the PCLB algorithm is 167,608 [Ws]. The total power consumption in the RR algorithm is 190,938 [Ws]. In the PCLB algorithm, the power consumption can be more reduced than the RR algorithm.

Table 3.6. Total power consumption [Ws]

	PCLB	RR
Total power consumption [Ws]	167,608	190,938

Table 3.7 shows the total simulation time to terminate the total number 600 of processes in the PCLB and RR algorithms. The termination times of all the processes in the PCLB and RR algorithms are the same 1,014 [sec]. The difference of the termination time between PCLB and RR algorithms is neglectable.

Table 3.7. Termination time [sec]

	PCLB	RR
Termination time [sec]	1,014	1,014

From the evaluation results, we conclude the total power consumption can be more reduced in the PCLB algorithm than the RR algorithm and the difference of termination time between the PCLB and RR algorithms is neglectable. Therefore, the PCLB algorithm is more useful than the RR algorithm.

3.7.2 CM Applications

We evaluate the EPCB algorithm in terms of the total power consumption and elapse time compared with the RR algorithm through the simulation. In the evaluation, there are five servers $S = \{s_1, s_2, s_3, s_4, s_5\}$ as shown in Table 3.8. The power consumption coefficient δ_t of each server s_t is randomly selected between 0.02 and 0.11 [W/Mb] based on the experimental results. The increasing rate of the power consumption $\beta_t(m)$ for the number m of clients is randomly selected between 1.09 and 1.5. The minimum power consumption rate $minE_t$ of each server s_t is randomly selected between 3 and 4 [W]. The maximum transmission rate $Maxtr_t$ is randomly selected between 150 and 450 [Mbps]. Each server s_t has a replica of a file f of one giga-byte.

Table 3.8. Types of servers

Servers	α	$\beta(m)$	$minE$ [W]	$Maxtr$ [Mbps]
s_1	0.03	1.259	3.39	406
s_2	0.05	1.195	3.17	401
s_3	0.03	1.285	3.12	249
s_4	0.09	1.117	3.90	231
s_5	0.02	1.162	3.02	171

Totally 100 clients download the file f from one server s_t in the server set S. The maximum receipt rate $Maxrr_s$ of each client c_s is randomly selected between 1 and 100 [Mbps]. Each client c_s issues a transfer request of the file f to a load balancer K at time st_s. Here, the starting time st_s is randomly selected between 1 and 3,600 [sec] at the simulation time. Each client c_s issues one request at time st_s in the simulations of the EPCB and RR algorithms.

Figure 3.24 shows the total power consumption rates [W] of the servers s_1, ..., s_5. Table 3.9 shows the total power consumptions of the servers in the EPCB and RR algorithms. The total power consumptions of the EPCB and RR algorithms are 204,973 and 1,058,238 [Ws], respectively. The total amount of power consumption in the RR algorithm is larger than the EPCB algorithm. In the EPCB algorithm, a server s_t is selected for a client c_s, whose power consumption is the smallest to transmit a file f to the client c_s.

Table 3.10 shows the elapse time in the EPCB and RR algorithms. The elapse time are 9,134 [sec] and 44,178 [sec] in the EPCB and RR algorithms, respectively. The elapse time of the RR algorithm is longer than the EPCB algorithm.

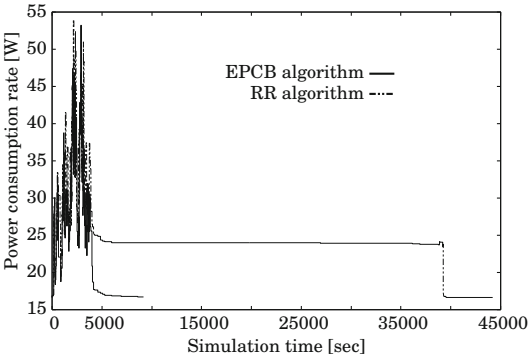


Fig. 3.24. Total power consumption rate [W]

Table 3.9. Total power consumption [Ws]

EPCB	RR
204,973 [Ws]	1,058,238 [Ws]

Table 3.10. Elapse time

EPCB	RR
9,134 [sec]	44,178 [sec]

From the evaluation results, we consider the total power consumption can be more reduced in the EPCB algorithm than the RR algorithm. In addition, the elapse time can be more reduced in the EPCB algorithm than the RR algorithm. Therefore, the EPCB algorithm is more useful than the RR algorithm.

3.8 Conclusion

In this chapter, we discussed the power consumption models of a server to perform types of application processes based on the experimental results. First, we classified applications on a cluster system into three types: computation (CP), communication (CM), and storage (ST) types of applications. Then, we measured the power consumption of a whole server to perform the types of application. We defined computation, communication, and power consumption models for each type of application based on the measurement of a power consumption of servers. The power consumption models for the CP, CM, and ST applications are two-state ones. Here, a server consumes the maximum electric power as long as at least one process is performed. Otherwise, the minimum power is consumed. According to the computation, communication, and power consumption models, we proposed algorithms to select one of servers in a server cluster for the CP and CM types of applications so that not only application requirements are satisfied but also the total power consumption of servers can be reduced. We evaluated the proposed algorithms for the CP and CM applications in terms of the total power consumption and elapse time compared with the RR algorithm. The evaluation results showed the proposed algorithms can reduce the total power consumption of servers.

References

1. Advanced Micro Devices, Inc.: Power Efficiency Technology Leadership from AMD. Available via DIALOG, http://sites.amd.com/us/business/it-solutions/power-efficiency/Pages/power-efficiency.aspx (cited August 10, 2011)
2. Advanced Micro Devices, Inc.: ACP - The Truth About Power Consumption Starts Here. Available via DIALOG, http://www.amd.com/us/Documents/43761D-ACP_PowerConsumption.pdf (cited November 8, 2011)
3. Aikebaier, A., Enokido, T., Takizawa, M.: Energy-Efficient Computation Models for Distributed Systems. In: The 12th International Conference on Network-Based Information Systems (NBiS 2009), pp. 424–431 (2009)
4. Apache HTTP Server Version 2.0 Documentation. Available via DIALOG, http://httpd.apache.org/docs/2.0/en/ (cited August 8, 2011)
5. Apache Module mod_deflate. Available via DIALOG, http://httpd.apache.org/docs/2.0/en/mod/mod_deflate.html (cited August 8, 2011)
6. Bevilacqua, A.: A Dynamic Load Balancing Metthod on a Heterogeneous Cluster of Workstations. Informatica 23(1), 49–56 (1999)
7. Bianchini, R., Carrera, E.V.: Analytical and Experimental Evaluation of Cluster-Based Network Servers. World Wide Web 3(4), 215–229 (2000)
8. Bianchini, R., Rajamony, R.: Power and Energy Management for Server Systems. Computer 37(11), 68–76 (2004)
9. Barroso, L.A., Hölzle, U.: The Case for Energy-Proportional Computing. IEEE Computer 40(12), 33–37 (2007)
10. Buford, F.J., Yu, H., Lua, K.E.: P2P Networking and Applications. Morgan Kaufmann, MA (2009)
11. Carrera, E.V., Pinheiro, E., Bianchini, R.: Conserving Disk Energy in Network Servers. In: The 17th Annual International Conference on Supercomputing (ICS 2003), pp. 86–97 (2003)
12. Colajanni, M., Cardellini, V., Yu, P.S.: Dynamic Load Balancing in Geographically Distributed Heterogeneous Web Servers. In: The 18th IEEE International Conference on Distributed Computing Systems (ICDCS 1998), pp. 295–303 (1998)
13. Deutsch, P.: RFC: 1952 GZIP file format specification version 4.3. Available via DIALOG, http://www.ietf.org/rfc/rfc1952.txt (cited November 11, 2011)
14. Durresi, A., Durresi, M., Paruchuri, V., Barolli, L.: Ad Hoc Communications for Emergency Conditions. In: The 25th IEEE International Conference on Advanced Information Networking and Applications (AINA 2011), pp. 787–794 (2011)
15. Enokido, T., Aikebaier, A., Misbah Deen, S., Takizawa, M.: Power Consumption-based Server Selection Algorithms for Communication-based Systems. In: The 13th International Conference on Network-based Information Systems (NBiS 2010), pp. 201–208 (2010)
16. Enokido, T., Aikebaier, A., Takizawa, M.: A Model for Reducing Power Consumption in Peer-to-Peer Systems. IEEE Systems Journal 4(2), 221–229 (2010)
17. Enokido, T., Aikebaier, A., Takizawa, M.: Process Allocation Algorithms for Saving Power consumption in Peer-to-Peer Systems. IEEE Trans. on Industrial Electronics 58(6), 2097–2105 (2011)
18. Enokido, T., Suzuki, K., Aikebaier, A., Takizawa, M.: Algorithms for Reducing the Total Power Consumption in Data Communication-based Applications. In: The 24th IEEE International Conference on Advanced Information Networking and Applications (AINA 2010), pp. 142–149 (2010)
19. Enokido, T., Takizawa, M.: A Purpose-based Synchronization Protocol for Secure Information Flow Control. Journal of Computer Systems Science and Engineering (JCSSE) 25(2), 25–32 (2010)

20. Fan, X., Weber, W., Barroso, L.A.: Power Provisioning for a Warehouse-sized Computer. In: The 34th International Symposium on Computer Architecture (ICSA 2007), pp. 13–23 (2007)
21. Ferraiolo, D.F., Kuhn, D., Chandramouli, R.: Role-Based Access Control. Artech Hous, Norwood (2007)
22. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of Distributed Consensus with One Faulty Process. Journal of the ACM 32(2), 374–382 (1985)
23. Ghemawat, S., Gobioff, H., Leung, S.: The Google File System. In: The 19th ACM Symposium on Operating Systems Principles (SOSP 2003), pp. 29–43 (2003)
24. Grossman, R.L.: The Case for Cloud Computing. IT Professional 11(2), 23–27 (2009)
25. Heath, T., Diniz, B., Carrera, E.V., Meira, W.J., Bianchini, R.: Energy Conservation in Heterogeneous Server Clusters. In: The 10th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP 2005), pp. 186–195 (2005)
26. Hemmert, S.: Green HPC: From Nice to Necessity. Computing in Science and Engineering 12(6), 8–10 (2010)
27. Ikeda, M., Kulla, E., Hiyama, M., Barolli, L., Takizawa, M.: Experimental Results of a MANET Testbet in Indoor Stairs Environment. In: The 25th IEEE International Conference on Advanced Information Networking and Applications (AINA 2011), pp. 779–786 (2011)
28. Intel Power Management Technology. Available via DIALOG, http://www.intel.com/content/www/us/en/power-management/power-management-technologies-for-processor-graphics-display-and-memory-paper.html (cited August 10, 2011)
29. Intel Corporation.: Intel Xeon Processor 5600 Series : The Next Generation of Intelligent Server Processors. Available via DIALOG, http://www.intel.com/content/www/us/en/processors/xeon/xeon-5600-brief.html (cited November 8, 2011)
30. Inoue, T., Ikeda, M., Enokido, T., Aikebaier, A., Takizawa, M.: A Power Consumption Model for Storage-based Applications. In: The Fifth International Conference on Complex, Intelligent, and Software Intensive Systems, CISIS 2011 (2011)
31. Job Scheduling Algorithms in Linux Virtual Server. Available via DIALOG, http://www.linuxvirtualserver.org/docs/scheduling.html (cited August 29, 2011)
32. Liu, X., Zhao, H., Li, X.: EPC: Energy-aware Probability-based Clustering Algorithm for Correlated Data Gathering in Wireless Sensor Networks. In: The 25th IEEE International Conference on Advanced Information Networking and Applications (AINA 2011), pp. 419–426 (2011)
33. Meisner, D., Gold, B.T., Wenisch, T.F.: PowerNap: Eliminating Server Idle Power. In: The 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2009), pp. 205–216 (2009)
34. Murphy, R., Sterling, T., Dekate, C.: Advanced Architectures and Execution Models to Support Green Computing. Computing in Science and Engineering 12(6), 38–47 (2010)
35. Perera, G.: New Search Paradigms and Power Management for Peer-to-Peer File Sharing. VDM Verlag, Saarbrucken (2008)
36. Postel, J., Reynolds, J.: RFC: 959 File transfer protocol (FTP), Available via DIALOG, http://www.ietf.org/rfc/rfc959.txt (cited November 7, 2011)
37. Rahimi, S.K., Haug, F.S.: Distributed Database Management Systems. John Wiley & Sons, Hoboken (2010)
38. Rajamani, K., Lefurgy, C.: On Evaluating Request-Distribution Schemes for Saving Energy in Server Clusters. In: 2003 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS 2003), pp. 111–122 (2003)
39. Steinmetz, R., Nahrstedt, K.: Multimedia Systems. Springer, New York (2004)
40. UWmeter. Available via DIALOG, http://www.metaprotocol.com/UWmeter/Features.html (cited July 5, 2011)
41. Weighted Round Robin (WRR), Available via DIALOG, http://www.linuxvirtualserver.org/docs/scheduling.html (cited December 7, 2011)

42. Yang, T., Mino, G., Spaho, E., Barolli, L., Durresi, A., Xhafa, F.: A Simulation System for Multi Mobile Events in Wireless Sensor Networks. In: The 25th IEEE International Conference on Advanced Information Networking and Applications (AINA 2011), pp. 411–418 (2011)
43. Yang, Y., Xiong, N., Aikebaier, A., Enokido, T., Takizawa, M.: Minimizing Power Consumption with Performance Efficiency Constraint in Web Server Clusters. In: The 12th International Conference on Network-Based Information Systems (NBiS 2009), pp. 45–51 (2009)
44. Zhang, L., Zhou, Q.: CCOA: Cloud Computing Open Architecture. In: IEEE International Conference on Web Services, pp. 607–616 (2009)

Chapter 4
Energy and Security Awareness in Evolutionary-Driven Grid Scheduling

Joanna Kołodziej, Samee U. Khan, Lizhe Wang, Dan Chen, and Albert Y. Zomaya

Abstract. Ensuring the energy efficiency, thermal safety, and security-awareness in today's large-scale distributed computing systems is one of the key research issues that leads to the improvement of the system scalability and requires researchers to harness an understanding of the interactions between the system external users and the internal service and resource providers. Modeling these interactions can be computationally challenging especially in the infrastructures with different local access and management policies such as computational grids and clouds. In this chapter, we approach the independent batch scheduling in Computational Grid (CG) as a three-objective minimization problem with *Makespan*, *Flowtime* and energy consumption in risky and security scenarios. Each physical resource in the system is equipped with Dynamic Voltage Scaling (DVS) module for optimizing the cumulative power energy utilized by the system. The effectiveness of six genetic-based single- and multi-population grid schedulers has been justified in comprehensive empirical analysis.

Joanna Kołodziej
Institute of Computer Science, Cracow University of Technology, ul. Warszawska 24, 31-155 Cracow, Poland
e-mail: jkolodziej@uck.pk.edu.pl

Samee U. Khan
Department of Electrical and Computer Engineering, North Dakota State University, ND 58108, USA
e-mail: samee.khan@ndsu.edu

Lizhe Wang
Center for Earth Observation and Digital Earth, Chinese Academy of Sciences, Beijing, China
e-mail: LZWang@ceode.ac.cn

Dan Chen
China University of Geosciences Wuhan, China
e-mail: Danjj43@gmail.com

Albert Y. Zomaya
School of Information Technologies, University of Sydney, Sydney, NSW 2006, Australia
e-mail: albert.zomaya@sydney.edu.au

S.U. Khan et al. (Eds.): Evolutionary Based Solutions for Green Computing, SCI 432, pp. 95–138.
springerlink.com

4.1 Introduction

The concept of today's grid systems grown far beyond the original models of high performance computing centers and networks. The modern Computational Grids (CGs) are designed as the complex infrastructures that are made up of hundreds or thousands of various components (computers, databases, etc) and numerous user-oriented services and procedures. The management of such infrastructures remains challenging problem, especially when additional personalization of the communication protocols and security of the transferred data and information are the crucial issues. We can observe significant disproportion of resource availability and resource provisioning in such systems and the increase in energy consumption to match resource provisioning in light of the computational demands [25]. Therefore, the ability of the efficient scheduling of the grid applications and the allocation of the resources in a desired configuration of all system components in a scalable and robust manner is essential in today's grid computing.

The intelligent grid schedulers should efficiently optimize the standard scheduling objectives, such as *Makespan*, *Flowtime* and resource utilization, but also can meet the security requirements of the grid users and can minimize the energy consumed by all of the system components. Energy-efficient and security-aware scheduling in CGs becomes complex endeavors due to the multi-constraints and different optimization criteria and different priorities of the resource owners. The computational intractability of the problems calls for heuristic approaches as effective means for designing risk-resilient energy-aware grid schedulers by trading-off among the different scheduling requirements, constraints and scenarios. Recent literature leverages the capability of Genetic Algorithms (GAs) to provide satisfactory green computing solutions as well as security-aware scheduling by tackling the aforementioned scheduling challenges.

This chapter addresses the *Independent Batch Job Scheduling* problem in CGs, where tasks are processed in a batch mode, there are no dependencies among tasks, and two standard scheduling objective functions, namely *Makespan* and *Flowtime*, are minimized along with security and cumulative energy consumption in the system. A comprehensive empirical analysis of wide range of single- and multi-population genetic-based metaheuristics has been provided in static and dynamic grid environments by using the grid simulator.

The following notation for tasks and machines in independent grid scheduling is introduced from this point forward will be used throughout the chapter:

- n – the number of tasks in a batch;
- m – the number of machines available in the system for the execution of a given batch of tasks;
- $N = \{1, \ldots, n\}$ – the set of tasks' labels;
- $M = \{1, \ldots, m\}$ – the set of machines' labels.

The rest of the chapter is organized as follows. The generic architecture of the security-aware grid is defined in Section 4.2. The main scheduling attributes along with the definitions of scheduling problems, scheduling scenarios and main objectives are specified in Section 4.4. The generic framework of the single-population grid schedulers and various combinations of the genetic operators are presented in Section 4.5. The empirical evaluation of 18 types of simple genetic schedulers is provided in Section 4.6. Sec. 4.7 describes the main concepts and the empirical analysis of the effectiveness of four types of multi-population genetic strategies in grid scheduling. Related work is discussed in Section 4.8. The chapter is concluded in Section 4.9.

4.2 Generic Model of Secure Grid Cluster

Grid system is usually modelled as a multi-layer architecture with the hierarchical management system that consist of two or three levels depending on the system knowledge, access to data and the system services and resources. The whole architecture is composed of the local computational clusters, as it is presented in Fig. 4.1, and it is defined as a compromise between the centralized and decentralized resources and service managements.

The exemplary two-level architecture is a simple Meta-Broker model presented in [14]. In this model the Meta-Broker (MB) operates in the inter-site global level and is responsible for the processing of all applications and information submitted by the grid users to the resource administrators. He also collects the feedback information from the resource owners and send the results of the computations to the users.

A three-level grid system example is presented by Kwok et al. in [33]. I this model there is a central global authority (central scheduler) who is responsible for the final optimal assignment of the tasks submitted to the system and the communication with the external grid users. The local system managers communicate with the resource administrators and resource owners and specify the "grid sites reputation indexes" that are further forwarded to the global scheduler. All the machines and resource providers work at the local intra-site level.

The hierarchical structure of the system management must be additionally combined with the multi-layer structure of the main grid components [10], namely: (1) grid "fabric" layer, (2) grid core middleware, (3) grid user layer, and (4) applications layer. The grid "fabric" layer is composed of the grid resources, services, and local resource management systems. The grid core middleware provides services related to security and access management, remote job submission, storage, and resource information and scheduling. The grid user layer contains all of the grid service end-users and system entities, and plays the most important role in ensuring multi-criterion scheduling and resource management efficiency.

The roles of the meta-scheduler and meta-brokers in grid clusters are different when security is considered as additional criterion in the scheduling process. The meta-scheduler must analyze the security requirements defined as security demand

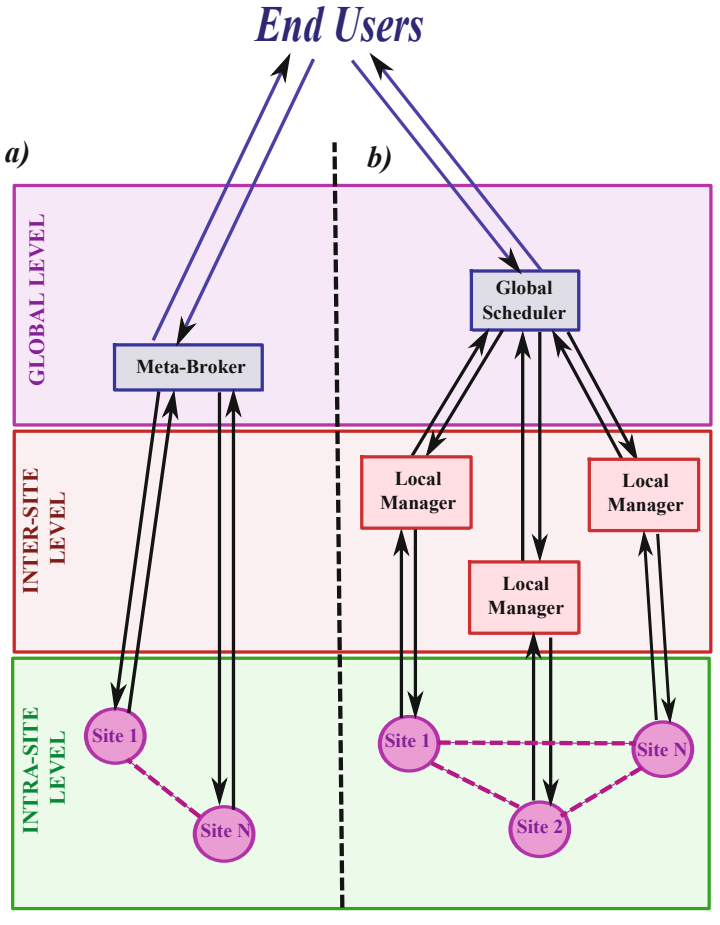


Fig. 4.1. Two-level (a) and three-level (b) hierarchical architecture of clusters in Computational Grids

parameters for the execution of tasks and requests of the CG users for trustful resources available within the system. The system brokers analyze "trust level" indexes of the machines received from the resource managers and send proposals to the scheduler. Moreover, the brokers also control the resource allocation and communication between CG users and resource owners.

The trust level and security demand parameters are the results of the aggregation of several scheduling and system attributes [47]. The main attributes can be specified as follows:

- **Security Demand (Tasks) Attributes:**
  - data integration;
  - task sensitivity;

  - access control;
  - peer authentication;
  - task execution environment;
- **Trust Level (Machines) Attributes:**
  - *Behavioral Attributes:*
    - prior task execution success rate;
    - cumulative grid cluster utilization;
  - *Intrinsic Security Attributes:*
    - firewall capabilities;
    - intrusion detection capabilities;
    - intrusion response capabilities.

The aggregation of the above mentioned attributes into a single-valued scalar parameters can be realized by using the fuzzy logic-based methods as it is demonstrated by Song et al. in [46]. In this model the machines and users' applications are characterized by the *security demand vector* $SD = [sd_1, \ldots, sd_n]$ and *trust level vector* $TL = [tl_1, \ldots, tl_m]$. The values of the sd_j and tl_i parameters are real fractions within the range [0,1] with 0 representing the lowest and 1 the highest security requirements for a task execution and the most risky and fully trusted machine, respectively. A task can be successfully completed at a resource when a *security assurance condition* is satisfied. That is to say that $sd_j \leq tl_i$ for a given (j,i) task-machine pair. The model of Song et al. is applied for the characteristic of machines and tasks in grid scheduling problem defined in the following section (Sec. 4.3). The process of matching sd_j with tl_i is similar to that of a real-life scenario where users of some portals, such as Yahoo!, are required to specify the security level of the login session.

4.3 Scheduling Problems in Computational Grids

There are numerous types of scheduling problems that can be specified in highly parametrized computational environments such as computational grids. A particular type of the problem can be defined with respect to different properties of the underlying grid environment and various requirements of the users, namely **(a)** type of the environment, **(b)** type of the grid architecture, **(c)** task processing policy, and **(d)** tasks' interrelations. To achieve the desired performance of the system all this information must be "embedded" into the scheduling mechanism [2], [29]. The aforementioned four scheduling attributes are presented in Fig. 4.2.

Depending on the type of the grid environment, the scheduling may be realized as the *static* or *dynamic* process. In the case of the static scheduling the number of the submitted applications and the available resources remain constant in a considered time interval, while in the dynamic scenario the resources may be added or removed from the system in an unpredictable way.

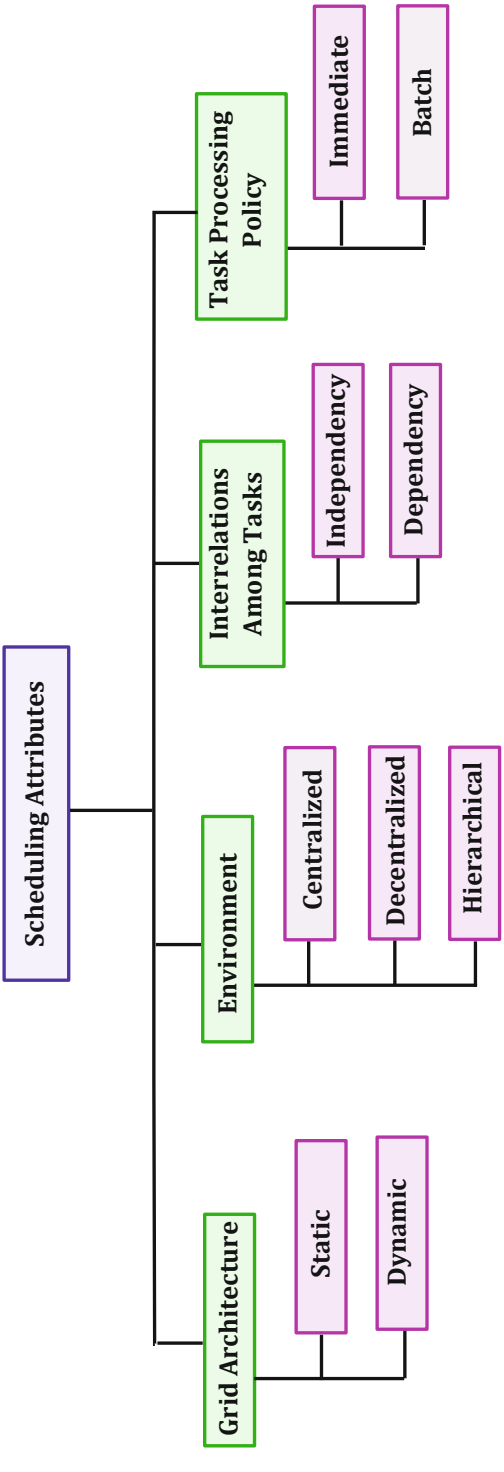


Fig. 4.2. Main scheduling attributes in Computational Grids

The resource management and scheduling can be organized in centralized, decentralized, or hierarchical modes. In *centralized model*, there is a central scheduler with a full knowledge of the system. In *decentralized model*, local schedulers interact with each other to manage the tasks submitted to the system. Finally, in the *hierarchical model*, there is a central meta-scheduler, which interacts with local managers (brokers) in order to define the optimal schedules.

The tasks in the system can be processed according to the *immediate* policy, where the tasks are scheduled as soon as they are entered into the system, or *batch* mode, where submitted tasks are grouped into batches and the scheduler assigns each batch to the resources.

Finally, tasks may be *independently* submitted and calculated in the grid system or considered as parallel applications with priority constraints and *interrelations* among the application components (usually modeled by a Directed Acyclic Graph (DAG)).

4.3.1 Problems Notation and Classification

To our best knowledge there is no standard unified notation for classification of the scheduling problems in grids. Fibich et al. proposed in [13] a simple extension of the Graham's [17] and Brucker's [9] classifications of conventional scheduling problems in order to adapt this methodology to the dynamic grid environment. Based on the idea presented in [13] and [27] and the main scheduling attributes specified in the previous section, the basic notation for the grid scheduling problem instances in CGs can be defined as follows:

$$\alpha|\beta|\gamma \tag{4.1}$$

where α characterizes the resource layer and grid architecture type, β specifies the processing characteristics and the constraints, and γ denotes the scheduling criteria.

(α) Resource Characteristics and Grid Architecture Type

According to the standard resource notation [17] the grid computational resources can be classify as Rm machines[1] with possible different speeds for different jobs. grid architecture type can be denoted by C for centralized, D for decentralized and $H(i)$ for hierarchical system, where i denotes the system levels. The following notation

$$Pm, H(2) \tag{4.2}$$

[1] For single CPU machine α=1; identical machines in parallel infrastructure – Pm, machines in parallel with different speeds – Qm, unrelated machines in parallel –Rm, and (flow—open—job) shop resources (Fm, Om, Jm)

is used for the representation of the 2-level hierarchical grid architecture with parallel identical machines in the system.

(β) Tasks Processing Mode, Tasks Interrelations, Static and Dynamic Scheduling Modes and Scheduling Constraints

The following notation can be used for setting the grid tasks' attributes and scheduling modes (see also Fig. 4.2):

- b – batch mode;
- im – immediate mode;
- dep – dependency among tasks ;
- $indep$ – independent scheduling;
- sta – static scheduling;
- dyn – dynamic scheduling;

The scheduling may be conducted under various constraints specified by all the grid users. The typical constraints include budget and deadline (d_j) limits for a given task j. An instant of the independent batch scheduling in dynamic mode with a limited deadline can be denoted as follows:

$$b, indep, dyn, d_j. \tag{4.3}$$

(γ) Scheduling Criteria and Objectives

The problem of scheduling tasks in CG is multi-objective in its general setting as the quality of the solutions can be measured using several criteria.

Two basic models are utilized in multi-objective optimization: hierarchical and simultaneous modes. In the *simultaneous mode* (*sim*) all objectives are optimized simultaneously while in the *hierarchical* (*hier*) case, the objectives are sorted *a priori* according to their importance in the model. The process starts by optimizing the most important objective and when further improvements are impossible, the second objective is optimized under the restriction of keeping unchanged (or improving) the optimal values of the first, and so on. It is very hard in grid scheduling to define or efficiently approximate the Pareto front, especially in dynamic scheduling, where it may extend very fast together with the scale of the grid and the number of the submitted tasks.

The basic scheduling criteria can be defined as global optimization criteria and include: *Makespan*, *Flowtime*, resource utilization, load balancing, matching proximity, turnaround time, total weighted completion time, lateness, weighted number of tardy jobs, weighted response time, etc [50].

In this chapter we consider two standard optimization criteria for grid scheduling, namely *Makespan* and *Flowtime*, that can be defined as follows:

- *Makespan* – is defined as the finishing time of the latest task and can be calculated by the following formulae:

$$C_{max} = \min_{S \in Schedules} \left\{ \max_{j \in Tasks} C_j \right\}, \tag{4.4}$$

where C_j denotes the time when task j is finalized, $Tasks$ denotes the set of all tasks submitted to the grid system and $Schedules$ is the set of all possible schedules generated for the considered batch of the tasks (or current number of tasks submitted to the system at a given moment); and

- *Flowtime*, that is expressed as the sum of finalization times of all the tasks. It can be defined in the following way:

$$C_{sum} = \min_{S \in Schedules} \left\{ \sum_{j \in Tasks} C_j \right\}. \tag{4.5}$$

The notation

$$hier, (C_{max}, C_{sum}) \tag{4.6}$$

means that *Makespan* and *Flowtime* are optimized in the hierarchical mode.

Makespan and *Flowtime* define the deadline and *Quality of Service (QoS)* criteria that are usually considered in most of the grid scheduling problems. To express the security and energy awareness, some additional scheduling objective functions must be specified. In Section 4.4 we define the *Independent Batch Scheduling* problem and *Expected Time to Compute* matrix model. Three main scheduling scenarios specified by setting the security and energy scheduling attributes are discussed. The main scheduling objectives are expressed in terms of completions times of machines.

4.4 Independent Batch Scheduling Problem, Scheduling Scenarios and Objective Functions

Independent Batch Scheduling is one of the simplest and fundamental scheduling models in computational grids. It is assumed in this model that: (a) tasks are grouped into batches; (b) tasks can be executed independently in a hierarchically structured static or dynamic grid environments. According to the notation introduced in Sec. 4.3.1 (see formula (4.1) an instance of the independent batch grid scheduling problem can be expressed as follows:

$$RmH(3)\left[\{b, indep, (stat, dyn), hier\}\right](objectives)), \tag{4.7}$$

where:

- Rm – Graham's notation references that tasks are mapped into (parallel) resources of various speed[2]
- $H(3)$ – denotes 3-level hierarchical grid management system;
- b – designates that the task processing mode is "batch mode"
- $indep$ – denotes "independency" as the task interrelation
- (sta, dyn) – indicates that both static and dynamics grid scheduling modes are considered
- $hier$ – references that the scheduling objectives are optimized in hierarchical mode
- $objectives$ – denotes the set of the considered scheduling objective functions.

In the case of energy-aware scheduling presented in this chapter there are three $objective$ functions, namely:

- C_{max} – denotes *Makespan* that is the dominant scheduling objective;
- C_{sum} – denotes *Flowtime* as the *Quality of Service (QoS)* objective;
- $E_I(E_{II})$ – denotes total energy consumption (E_I or E_{II} is selected depending on the scheduling scenario (see Sec. 4.4.3))

The schedules are realized in the ***secure*** and ***risky*** modes specified according to the security requirements defined for scheduling by the grid users or/and the system administrators and service and resource provides.

4.4.1 Expected Time to Compute (ETC) Matrix Model Adapted to Energy and Security Aware Scheduling in Grids

Independent batch scheduling in CG can be modelled by using the *Expected Time to Compute (ETC)* matrix model [4]. In the conventional version of this model tasks and machines are characterized by the following parameters:

(a) **Task** j:

- wl_j – workload parameter expressed in Millions of Instructions (MI)– $WL = [wl_1, \ldots, wl_n]$ is a *workload vector* for all tasks in the batch;

(b) **Machine** i:

- cc_i – computing capacity parameter expressed in *Millions of Instructions Per Second (MIPS)*, this parameter is a coordinate of a *computing capacity vector*, which is denoted by $CC = [cc_1, \ldots, cc_m]$;
- $ready_i$ – ready time of i, which expresses the time needed for the reloading of the machine i after finishing the last assigned task, a *ready times vector* for all

[2] In independent grid scheduling it is usually assumed that each task may be assigned just to one machine.

machines is denoted by
$ready\_times = [ready_1, \ldots, ready_m]$.

Tasks can be considered as monolithic applications or meta-task with no dependencies among the components. The term "machine" is related to a single or multiprocessor computing unit or even to a local small-area network.

For each pair (j,i) of task-machine labels, the coordinates of WL and CC vectors are usually generated by using some probability distributions (we have used the Gaussian distribution [34] for the purpose of the empirical analysis presented later on in this chapter) and are used for an approximation of the completion time of the task j on machine i. This completion time is denoted by $ETC[j][i]$ and can be calculated in the following way:

$$ETC[j][i] = \frac{wl_j}{cc_i}. \tag{4.8}$$

All $ETC[j][i]$ parameters are defined as elements of an ETC matrix, $ETC = \left[ETC[j][i]\right]_{n\times m}$, which is the main structure in ETC model. The elements in the rows of the ETC matrix define the estimated completion times of a given task on different machines, and elements in the column of the matrix are interpreted as approximate times of the completion of different tasks on a given machine.

This model is useful for a formal specification of *Makespan* and *Flowtime* criteria, that can be expressed in terms of completion times of machines. A *completion time completion*$[i]$ of the machine i is defined as the sum of the ready time parameters for this machine and a cumulative execution time of all tasks actually assigned to this machine, that is to say:

$$completion[i] = ready_i + \sum_{j\in Task(i)} ETC[j][i], \tag{4.9}$$

where $Task(i)$ is the set of tasks assigned to the machine i.

The *Makespan* C_{max} is expressed as the maximal completion time of all machines, that is:

$$C_{max} = \max_{i\in M} completion[i]. \tag{4.10}$$

The cumulative *Flowtime* for a given schedule is defined as a sum of workflows of the sequences of tasks on machines, that is to say:

$$C_{sum} = \sum_{i\in M} \left[ready_i + \sum_{j\in Sorted[i]} ETC[j][i] \right] \tag{4.11}$$

where $Sorted[i]$ denotes a set tasks assigned to the machine i sorted in ascending order by the corresponding ETC values.

The completion times of machines and the times of successful executions of tasks on these machines can change if the security and energy optimization conditions are additionally considered. Therefore, the conventional ETC matrix model must be

extended by incorporating the additional characteristics of tasks and machines in the system, and the modification of the methodologies of the generation of the ETC matrix.

4.4.2 Security Conditions

In security-aware scheduling all the system "actors", namely grid end-users, grid schedulers, cluster service and resource providers, system administrators and resource owners may specify their own requirements for the secure execution of tasks on the grid machines and secure access to the grid data and services. The most important users' requirements can be expressed in the following way [28]:

- **Access to Remote Data**: The input and output data specified by the user may be stored remotely. Therefore, users will need to provide the location of the remote data. If a ubiquitous wide-area file system is in operation on the grid, the user would only have to care about the location of files and data with respect to some root location under which they are stored.
- **Resource Specification**: The user may specify his individual requirements for the resources necessary in optimizing the execution times and costs of scheduling and computing tasks. The user may wish to target particular types of resources (e.g. SMP machines), but should not be concerned with the type of resource management on the grid, nor with the resource management systems on individual resources on the grid.
- **Resource reliability**: In some cases, the machines within the grid system could be unavailable due to high system dynamics or special policies of the resource owners. The user should be informed about the resource reliability in order to reduce the cost of possible resource failures or the abortion of executed tasks. In the case of resource failure the system administrators can activate re-scheduling or task migration procedures, and pre-emption policies.
- **Trustfulness of Resources**: The user may be required to allocate his tasks in the most trustful resources. Therefore the user should be able to verify the trust indexes of the resources and estimate the security demands for his tasks on the available resources.
- **Standardized authentication and authorization mechanisms requirements**: The users will likely utilize a standardized certificate authentication scheme. The certificates can be digitally signed by a certificate authority, and kept in the user's repository, which is recognized by the resources and resource owners. It is desirable for a certificate to be automatically created by the user's interface application during task submission.

The successful execution of tasks submitted to CGs may be impossible (or interrupted) if such requirements are very strong and the access to the resources is limited. On the other hand, the grid cluster or the grid resource may be not accessible to the global meta-scheduler or grid user when being infected with intrusions

or by malicious attacks. It means that some, even very simple, authorization and authentication protocols and some anti-viruses protection mechanism are needed for efficient scheduling especially in the dynamic environment. In such cases the machines and task should be additionally characterized by the trust level tl_i and security demands sd_j parameters as it was specified in Sec. 4.2.

Let us denote by Pr_f a *Machine Failure Probability* matrix, that defines the probabilities of failures of the machines during the tasks executions $P_f[j][i]$ for each task-machine pair (j,i). These probabilities can be calculated by using the negative exponential distribution function, that is to say:

$$Pr_f[j][i] = \begin{cases} 0 & , \ sd_j \leq tl_i \\ 1 - e^{-\alpha(sd_j - tl_i)} & , \ sd_j > tl_i \end{cases} \tag{4.12}$$

where α is interpreted as a failure coefficient and is a global parameter of the model.

The scheduler may initialize his work in two different modes: (a) *secure mode*, where he analyzes the elements of the *Machine Failure Probability* matrix in order to minimize the failure probabilities for task-machine pairs; and (b) *risky mode*, in which he performs an "ordinary" scheduling without any preliminary analysis of the security conditions, aborts the task scheduling in the case of machine failure, and reschedule this task at another resource.

Secure Mode

In this scenario all of the security and resource reliability conditions are verified for all task-machine pairs (j,i). The main goal of the meta-scheduler is to design an optimal schedule for which, beyond the other criteria, the probabilities of failures of the machines during the tasks executions will be minimal. It is assumed that additional "cost" of the verification of security assurance condition for a given task-machine pair may delay the predicted execution time of the task on the machine. This cost is approximately proportional to the probability of failure of the machine during the task execution. The completion time of the machine i in the secure mode is denoted by $completion^s[i]$ and can be calculated as follows:

$$completion^s[i] = ready_i + \sum_{\{j \in Tasks(i)\}} (1 + Pr_f[j][i]) ETC[j][i]). \tag{4.13}$$

The *Makespan* and *Flowtime* in this case can be expressed as follows:

$$C^s_{max} = \max_{i \in M} completion^s[i]. \tag{4.14}$$

$$C^s_{sum} = \sum_{i \in M} \left[ready_i + \sum_{j \in Sorted[i]} \left(1 + Pr_f[j][i]\right) \cdot ETC[j][i] \right]. \tag{4.15}$$

Risky Mode

In this scenario all secure and failing conditions are ignored. The scheduling process is realized as a two-step procedure. First, the scheduling is performed just by analyzing the **conventional** *ETC* matrix. If failures of machines are observed, then the unfinished tasks are temporarily moved into the backlog set of tasks. The tasks from this set are re-scheduled according the rules specified for the secure mode. The total completion time of machine i ($i \in M$) in this case can be defined as follows:

$$completion^r[i] = completion[i] + completion^s_{res}[i], \tag{4.16}$$

where $completion[i]$ is calculated by using the Eq. (4.9), for tasks primarily assigned to the machine i , and $completion^s_{res}[i]$ is the completion time of machine i calculated by using the Eq. (4.13) for rescheduled tasks, i.e. the tasks re-assigned to the machine i from the other resources.

The Makespan and Flowtime in this mode can be calculated in the following way:

$$C^r_{max} = \max_{i \in M} completion^r[i]. \tag{4.17}$$

$$C^r_{sum} = \sum_{i \in M} \left[ready_i + \sum_{j \in Sorted[i]} ETC[j][i] + \sum_{j \in Sorted_{res}[i]} \left(1 + Pr_f[j][i]\right) \cdot ETC[j][i] \right]. \tag{4.18}$$

In the hierarchical optimization mode (see Eq. (4.7), parameter *hier*) where *Makespan* is defined as the dominant scheduling criterion, the *Flowtime* objective function should be minimized in both secure and risky scenarios subject to the following constraints:

- in the secure mode

$$ready_i + \sum_{j \in Sorted[i]} \left(1 + Pr_f[j][i]\right) \cdot ETC[j][i] \leq C^s_{max}) \; \forall i \in M; \tag{4.19}$$

- in the risky mode

$$ready_i + \sum_{j \in Sorted[i]} ETC[j][i] + \sum_{j \in Sorted_{res}[i]} \left(1 + Pr_f[j][i]\right) \cdot ETC[j][i] \leq C^r_{max} \; \forall i \in M. \tag{4.20}$$

Although the probabilities of machines' failures are expected to be higher in the risky than in the secure mode, there is certainly no guarantee of the successful execution of all tasks in the security scenario. It can be observed that if the *security assurance condition* is satisfied for each task-machine pair (i.e. $sd_j \leq tl_i$ for $i \in M, j \in N$), the completion times of machines in both *secure* and *risky* modes are identical with the completion times defined for standard independent scheduling problem, where it is assumed that each task *must* be successfully executed on each machine and no security requirements are analyzed.

4.4.3 Energy Model

The energy model presented in this chapter is based on the *Dynamic Voltage and Frequency Scaling (DVFS)* technique [25] that is used for adjusting the voltage supplies and frequencies of the grid computational nodes.

Each machine in the grid is equipped with a DVFS module for scaling its supply voltage and operating frequency. It has been assumed that the frequency of the machine is proportional to its processing speed [37]. It follows from the Eq. (4.24) that the reduction of the supply voltage and frequency is directly correlated to the reduction of the energy utilization. Table 4.1 shows the typical parameters for 16 DVFS levels and three main "energetic" categories for machines.

Table 4.1. DVFS levels for three machine classes

	Class I		Class II		Class III	
Level	**Volt.**	**Rel.Freq.**	**Volt.**	**Rel.Freq.**	**Volt.**	**Rel.Freq.**
0	1.5	1.0	2.2	1.0	1.75	1.0
1	1.4	0.9	1.9	0.85	1.4	0.8
2	1.3	0.8	1.6	0.65	1.2	0.6
3	1.2	0.7	1.3	0.50	1.9	0.4
4	1.1	0.6	1.0	0.35		
5	1.0	0.5				
6	0.9	0.4				

The energetic class of machine i, $(i \in M)$ is denoted by s^i and it is represented by the vector $Vr_{(i)}$ of DVFS levels, that can be specified as follows:

$$Vr_{(i)} = \left[(v_{s_0}(i), f_{s_0}(i)); \ldots; (v_{s_{l(max)}}(i), f_{s_{l(max)}}(i))\right]^T \tag{4.21}$$

where $v_{s_l}(i)$ refers to the voltage supply for machine i at level s_l, $f_{s_l}(i)$ is a scaling parameter for the frequency of the machine at the same level s_l, and l_{max} is the number of levels in the class s^i. The parameters $\{f_{s_0}(i), \ldots, f_{s_{l(max)}}(i)\}$ lie in the [0,1] range and should be interpreted as the relative frequencies of the machine i from class s^i at the $s_0, \ldots, s_{l(max)}$ DVFS levels.

The reduction of the machine frequency and its supply voltage can lead to the extension of the computational times of the tasks executed on that machine. For a given "task-machine" pair (j,i), the completion times for the task j on machine i at different DVFS levels in the class s^i can be interpreted as the coordinates of a vector $\widehat{ETC}[j][i]$ that is defined in the following way:

$$\widehat{ETC}[j][i] = \left[\frac{1}{f_{s_0}(i)} \cdot ETC[j][i], \ldots, \frac{1}{f_{s_{l(max)}}(i)} \cdot ETC[j][i]\right], \tag{4.22}$$

where $ETC[j][i]$ are the expected completion times for task j on machine i calculated by using the conventional ETC matrix model.

In energy-aware scheduling the completion times calculated for each pair (j,i) of task–machine labels in conventional ETC matrix (see Eq. (4.8)) should be replaced by the $\widehat{ETC}[j][i]$ vectors, that is to say:

$$\widehat{ETC} = \left[\widehat{ETC}[j][i][s_l]\right]_{n\times m\times s_{l(max)}}, \tag{4.23}$$

where $\widehat{ETC}[j][k][s_l]$ is the time necessary for the completion of the task j on machine i at the level s_l.

The DVFS model is based on the power consumption model employed in complementary metal-oxide semiconductor (CMOS) logic circuits [6]. In CMOS model, the capacitive power Pow_{ji} utilized by the machine i for computing the task j it is expressed as follows:

$$Pow_{ji} = A \cdot C \cdot v^2 \cdot f, \tag{4.24}$$

where A is the number of switches per clock cycle, C is the total capacitance load, v is the supply voltage and f is the frequency of the machine. The energy consumed per machine i for the computation of task j is defined as the result of the following integration:

$$E_{ji} = \int_0^{completion[j][i]} Pow_{ji}(t)dt, \tag{4.25}$$

where $completion[j][i]$ is a completion time of the task j on machine i.

Based on Equation (4.23) the energy used for finishing the task j on machine i at level s_l can be defined as follows:

$$E_{ji}(s_l) = \gamma \cdot (f_{s_l}(i))_j \cdot f \cdot [(v_{s_l}(i))_j]^2 \cdot \widehat{ETC}[j][i][s_l], \tag{4.26}$$

where $\gamma = A \cdot C$ is a constant parameter for a given machine class, $(v_{s_l}(i))_j$ is a voltage supply value for class s^i and machine i at level s_l for computing task j, and $(f_{s_l}(i))_j$ is a corresponding relative frequency for machine i.

Therefore the computational times for each possible pair (j,i) at the level s_l can be calculated as follows:

$$\begin{aligned} Tim_{\{j,i,s_l\}} &= \gamma \cdot (f_{s_l}(i))_j \cdot f \cdot [(v_{s_l}(i))_j]^2 \cdot (f_{s_1}(i))_j \cdot ETC[j][i] = \\ &= \gamma \cdot f \cdot [(v_{s_l}(i))_j]^2 \cdot ETC[j][i] \end{aligned} \tag{4.27}$$

The cumulative energy utilized by the machine i for the completion of all tasks from the batch that are assigned to this machine, is defined in the following way:

$$\begin{aligned} E_i = \sum_{\substack{j\in Tasks(i)\\ l\in\hat{L}_j}} \{Tim_{\{j,i,s_l\}}\} + \gamma \cdot f \cdot [v_{s_{max}}]^2 \cdot ready_i + \gamma \cdot f_{s_{min}}(i) \cdot f \cdot \\ \cdot [v_{s_{min}}(i)]^2 \cdot Idle[i] = \gamma \cdot f \cdot \sum_{\substack{j\in Tasks(i)\\ l\in\hat{L}_i}} ([(v_{s_l}(i))_j]^2 \cdot ETC[j][i]) + \\ + [v_{s_{max}}(i)]^2 \cdot ready_i + f_{s_{min}}(i) \cdot [v_{s_{min}}(i)]^2 \cdot Idle[i] \end{aligned} \tag{4.28}$$

where $Idle[i]$ denotes an idle time of machine i, and $\hat{L}_i$ denotes a subset of DVFS levels used for the tasks assigned to machine i[3].

Finally, an average cumulative energy utilized by the grid system for completion of all tasks in the batch is defined as:

$$E_{batch} = \frac{\sum_{i=1}^{m} E_i}{m} \tag{4.29}$$

This model is used in the following section for specification of two "energetic" scheduling scenarios.

4.4.3.1 Energy-Aware Scheduling Scenarios

It is assumed machines supplied with DVFS modules can work in two following modes:

I. **Max-Min Mode**, where each machine works at the **maximal** DVFS level during the execution and computation of tasks and switch into idle mode after the execution of all tasks assigned to this machine;
II. **Modular Power Supply Mode**, where each machine can work at **different** DVFS levels during the task executions and can then enter into idle mode.

The procedures for calculation and optimization *Makespan*, *Flowtime* and cumulative energy utilized by the system, are different in the aforementioned scheduling scenarios and also additionally in secure and risky modes.

In **Max-Min Mode** the formulas for the completion time, *Makespan* and *Flowtime* are the same as in Eqs. (4.16), (4.17) and (4.18) in the risky mode; and Eqs. (4.13), (4.14) and (4.15) in the secure mode.

The idle time for machine i working in **Max-Min Mode** can be expressed as the difference between the *Makespan* and the completion time of this machine, that is to say:

$$Idle^r[i] = C^r_{max} - completion^r[i] \tag{4.30}$$

in the risky mode, and

$$Idle^s[i] = C^s_{max} - completion^s[i] \tag{4.31}$$

in the secure mode.

In the case of the machine with the maximal completion time (*Makespan*), the idle factor is zero.

In **Modular Power Supply Mode**, for each task-machine pair, the DSV level s_l must be specified. The formulas for computing the completion times in secure and risky scenarios at the level s^i can be defined as follows:

$$completion^s_{Il}[i] = ready_i + \sum_{j \in Tasks(i)} \frac{1}{f_{s_l}(i)} \cdot \left(1 + P_f[j][i]\right) \cdot ETC[j][i]. \tag{4.32}$$

[3] All additional machine frequency transition overheads do not bear down on the overall ETC model with an active "energetic" module and are ignored.

in the secure mode, and

$$completion^{r}_{II}[i] = completion_{II}[i] + completion^{s}_{res,II}[i] \tag{4.33}$$

in the risky mode, where $completion^{s}_{res,II}[i]$ is the completion time calculated for rescheduled tasks on machine i, and $completion_{II}[i]$ is calculated as follows:

$$completion_{II}[i] = ready_i + \sum_{j \in Tasks(i)} \frac{1}{f_{s_l}(i)} \cdot ETC[j][i]. \tag{4.34}$$

These formulas are using for the specification of the *Makespan* $\left(C^{r}_{max}\right)_{II}$ and $\left(C^{s}_{max}\right)_{II}$ in both security and risky modes. The *Flowtime* can be calculated from the modified Eq. (4.18) and (4.15) by replacing the $ECT[j][i]$ components by $\frac{1}{f_{s_l}(i)} \cdot ETC[j][i]$.

Finally, the formulas for idle times can be expressed as follows:

$$Idle^{s}_{II}[i] = \left(C^{s}_{max}\right)_{II} - completion^{s}_{II}[i] \tag{4.35}$$

$$Idle^{r}_{II}[i] = \left(C^{r}_{max}\right)_{II} - completion^{r}_{II}[i] \tag{4.36}$$

The average energy consumed in the system in **Min-Max Mode** and risky scenario is defined as follows:

$$\begin{aligned} E^{r}_{I} &= \tfrac{1}{m} \cdot \textstyle\sum_{i=1}^{m} \gamma \cdot completion^{r}[i] \cdot f \cdot [v_{s_{max}}(i)]^2 + \\ &+ \tfrac{1}{m} \cdot \textstyle\sum_{i=1}^{m} \gamma \cdot f_{s_{min}}(i) \cdot [v_{s_{min}}(i)]^2 \cdot Idle^{r}[i] \end{aligned} \tag{4.37}$$

Similar formula can be defined for the energy utilized in the secure mode E^{s}_{I}. It can be realized by replacing the $completion^{r}[i]$ and $Idle^{r}[i]$ in Eq. (4.37) by $completion^{s}[i]$ and $Idle^{s}[i]$ (see Eq. (4.13) and (4.31)).

In **Modular Power Supply Mode** the average cumulative energy is given by Eq. (4.29):

$$E_{II} = \frac{\sum_{i=1}^{m} E_i}{m} \tag{4.38}$$

where E_i is calculated by modification the Eq. (4.28) by using the Eq. (4.33)–(4.36).

The optimization procedure is two-steps. First the *Makespan* and *Flowtime* are minimized. Then, keeping the first two objectives at the optimal levels, the cumulative energy utilized in the system is also minimized.

4.5 Security-Aware Genetic-Based Batch Schedulers

Heuristic methods are well known from their robustness and have been applied successfully to solve scheduling problems and general combinatorial optimization problems in a variety of fields [3], [2], [14], [29]. In grid scheduling these methods can tackle the various scheduling attributes and additional energy and security aspects.

The heuristic scheduling methods are usually classified into three main groups, namely (1) calculus-based (greedy algorithms and ad-hoc methods); (2) stochastic (guided and non-guided methods); and (3) enumerative methods (dynamic programming and branch-and-bound algorithm). In this work we focus on genetic-based methods that classified as population-based stochastic schedulers. Alg. 1 defines a generic framework for genetic algorithm (GA) designed for grid scheduling. This framework is based on the general model of the conventional single-population GA dedicated [40].

Algorithm 1. A template of the genetic engine for *HGS-Sched*

1: Generate the initial population P^0 of size μ; $e = 0$
2: Evaluate P^0;
3: **while** not termination-condition **do**
4: Select the parental pool T^e of size λ; $T^t := Select(P^e)$;
5: Perform crossover procedure on pars of individuals in T^e with probability p_c; $P_c^e := Cross(T^e)$;
6: Perform mutation procedure on individuals in P_c^e with probability p_m; $P_m^e := Mutate(P_c^e)$;
7: Evaluate P_m^e ;
8: Create a new population P^{e+1} of size μ from individuals in P^e and P_m^e ; $P^{e+1} := Replace(P^e; P_m^e)$
9: $e := e + 1$;
10: **end while**
11: **return** Best found individual as solution;

The parameter e in Alg. 1 is the counter of the generations in GA (e counts the number of loops in GA). The populations in this algorithms are encoding by using the following two methodologies:

- **Direct Encoding:** A schedule vector S is expressed in the form:

$$S = [i_1, \ldots, i_n]^T, \tag{4.39}$$

where $i_j \in M$ denotes the number of the machine on which the task labeled by j is executed.

- **Permutation-based Encoding:** A schedule vector is defined as follows:

$$Sch = [Sch_1, \ldots, Sch_n]^T, \tag{4.40}$$

where $Sch_i \in N, i = 1, \ldots, n$. Schedule Sch is the vector of labels of tasks assigned to the machines. For each machine the labels of the tasks assigned to this machine are sorted in ascending order by the completion times of the tasks.

In permutation-based representation some additional information about the numbers of tasks assigned to each machine is required. The total total number of tasks assigned to a machine i is denoted by $\widetilde{Sch_i}$ and is interpreted as the i-th coordinate of an assignment vector $\widetilde{Sch} = [\widetilde{Sch_1}, \ldots, \widetilde{Sch_m}]^T$, which defines in fact the loads

of grid machines. The *direct representation* is used for encoding schedules in the base populations P^e and P^{e+1}, and *permutation representation* is used in P^e_c and P^t_m populations.

The initial population in Alg. 1 is generated by using the *Minimum Completion Time + Longest Job to Fastest Resource - Shortest Job to Fastest Resource MTC + LJFR-SJFR* method, in which all but two individuals are generated randomly. Those two individuals are created by using the *Longest Job to Fastest Resource - Shortest Job to Fastest Resource (LJFR-SJFR)* and *Minimum Completion Time (MCT)* heuristics [51]. In LJFR-SJFR method initially the number of m tasks with the highest workload are assigned to the available m machines sorted in ascending order by the computing capacity criterion. Then the remaining unassigned tasks are allocated to the fastest available machines. In the MCT heuristics, a given task is assigned to the machine yielding the earliest completion time. The detailed definition of those procedures may be found in [11].

Alg. 1 was adapted to the CG scheduling problem through an implementation of specialized encoding methods and genetic operators. The operators from the following set were used in experiments presented in this book:

- **Selection operators:** *Linear Ranking*;
- **Crossover operators:** *Partially Mapped Crossover (PMX)* and *Cycle Crossover (CX)*;
- **Mutation operators:** *Move*, *Swap* and *Rebalancing*;
- **Replacement operators:** *Steady State*, *Elitist Generational*, *Struggle*.

All the above mentioned operators are commonly used in the genetic meta-heuristics dedicated to solving combinatorial optimization problems [12]. The detailed definition and examples may be found in [5].

In *Linear Ranking* method a selection probability for each individual in a population is proportional to the rank of the individual. The rank of the worst individual is defined as zero, while the best rank is defined as $pop\_size - 1$, where pop_size is the size of the population.

The following crossover and mutation operators are implemented for permutation-based representation of the schedules. In *Partially Matched Crossover (PMX)* [15] a segment of one parent-chromosome is mapped to a segment of the other parent-chromosome (corresponding positions) and the remaining genes are exchanged according to the mapping 'relationship' of tasks and machines specified by the concrete scheduling rules. Tn *Cycle Crossover (CX)* [42], first, a cycle of alleles is identified. The crossover operator leaves the cycles unchanged, while the remaining segments in the parental strings are exchanged.

In *Move* mutation a task is moved from one machine to another one. Although the task can be appropriately chosen, this mutation strategy tends to unbalance the number of jobs per machine. It is realized by the modification of two coordinates in the vector $\tilde{u}$ of the schedule code in permutation-based representation. The main idea of the *Rebalancing* method is to improve the solution (by rebalancing the machine loads) and then mutate it. In rebalancing procedure, first, the most overloaded

machine is selected. Two tasks j and $\hat{j}$ are identified in the following way: j is assigned to another machine i', $\hat{j}$ is assigned to i and $ETC[j][i'] \leq ETC[\hat{j}][i]$. Then the assignments are interchanged for tasks j and $\hat{j}$. In *Swap* mutation the indexes of two selected tasks in the schedule representation are swapped.

A base population for a new GA loop in Alg. 1 may be defined by using the *Elitist generational* replacement method, where a new population contains two best solutions from the old base population and the rest are the newly generated offsprings. In the *Steady State* replacement method, the set of the best offsprings (the number of elements in this set is fixed) replaces the worst solutions in the old base population. The main drawback of this methods is that it can lead to the premature convergence of the algorithms in some local solutions. The *Struggle* replacement mechanism can be an effective tool for avoiding too fast of a scheduler's convergence to the local optima. In such method, new generations of individuals are created by replacing a part of the population by the individuals most similar – if this replacement minimizes the fitness value. The definition of the struggle replacement procedure requires a specification of the appropriate *similarity measure*, which indicates the degree of the similarity among two GA's chromosomes. We use in this work the *Mahalanobis distance* [36] for measuring the distances between schedules according to the following formula:

$$sim_e(S^1; S^2) = \sqrt{\sum_{j=1}^{n} \frac{(S^1[j] - S^[j])^2}{\sigma_P^2}} \tag{4.41}$$

where σ_P is the standard deviation of the $S^1[j]$ over the population P.

The struggle strategy has shown to be very effective in solving several large-scale multi-objective problems (see e.g., [7], [18]). However, the computational cost can be very high, because of the need to calculate distances among all off springs in resulting population and the individuals in the base population for the current GA loop. To reduce the execution time of the struggle procedure we use a *hash technique*, in which the hash table with the *task-resource allocation* key is created. The value of this key is calculated as the sum of the absolute values of the subtraction of each position and its precedent in the direct representation of the schedule vector (reading the schedule vector in a circular way).

Eighteen GA variants with all possible combinations of these operators are defined in Table 4.2.

All these algorithms have been empirically evaluated by using the grid simulator. The results of this empirical analysis are presented in the next section.

4.6 Empirical Evaluation of Genetic Grid Schedulers

In this section we present a simple empirical analysis of the efficiency of 18 implementations of single-population GA schedulers defined in the previous section in the minimization of the scheduling objective function specified in Sec. 4.4. The

Table 4.2. Eighteen variants of single-population GA-based schedulers

Scheduler	Crossover method	Mutation method	Replacement method
GA-PMX-M-SS	Partially Matched (PMX)	Move	Steady State
GA-PMX-M-EG	Partially Matched (PMX)	Move	Elitist Generational
GA-PMX-M-ST	Partially Matched (PMX)	Move	Struggle
GA-PMX-S-SS	Partially Matched (PMX)	Swap	Steady State
GA-PMX-S-EG	Partially Matched (PMX)	Swap	Elitist Generational
GA-PMX-S-ST	Partially Matched (PMX)	Swap	Struggle
GA-PMX-R-SS	Partially Matched (PMX)	Rebalancing	Steady State
GA-PMX-R-EG	Partially Matched (PMX)	Rebalancing	Elitist Generational
GA-PMX-R-ST	Partially Matched (PMX)	Rebalancing	Struggle
GA-CX-M-SS	Cycle (CX)	Move	Steady State
GA-CX-M-EG	Cycle (CX)	Move	Elitist Generational
GA-CX-M-ST	Cycle (CX)	Move	Struggle
GA-CX-S-SS	Cycle (CX)	Swap	Steady State
GA-CX-S-EG	Cycle (CX)	Swap	Elitist Generational
GA-CX-S-ST	Cycle (CX)	Swap	Struggle
GA-CX-R-SS	Cycle (CX)	Rebalancing	Steady State
GA-CX-R-EG	Cycle (CX)	Rebalancing	Elitist Generational
GA-CX-R-ST	Cycle (CX)	Rebalancing	Struggle

relative performance of all schedulers has been quantified with the following four metrics:

- *Makespan* – the dominant scheduling criterion which can be calculated in various way depending on the security and energetic criteria (see Sec. 4.4.3);
- *Mean_Flowtime* – mean *Flowtime* calculated as follows:

$$Mean\_Flowtime = \frac{C_{sum}}{m} \tag{4.42}$$

where C_{sum} similar to *Makespan*, can be calculated in various way depending on the security and energetic criteria (see Sec. 4.4.3);

- a relative energy consumption improvement rate expressed as follows:

$$Im(E) = \frac{E_I - E_{II}}{E_{batch}} \cdot 100\%, \tag{4.43}$$

where E_{II} and E_I are defined in Eq. (4.29) and Eq. (4.37) respectively;

- *FailureRate* $Fail_r$ parameter defined as follows:

$$Fail_r = \frac{n_{failed}}{n} \cdot 100\% \tag{4.44}$$

where n_{failed} is the number of unfinished tasks, which must be rescheduled

For providing the experiments and for simulating the grid environment we used the *Sim-G-Batch* grid simulator [28], that is based on the discrete event-based model *HyperSim-G* [52], and facilitates the evaluation of different scheduling heuristics under a variety of scheduling criteria across several grid scenarios. These scenarios are defined by the configuration of security conditions for scheduling and the access to the grid resources, grid size, energy utilization parameters, and system dynamics. The simulator allows the flexible activation or deactivation of all of the scheduling criteria and modules, as well as works with a mixture of meta-heuristic schedulers.

The main concept and general flow of the energy-aware and security version of the *Sim-G-Batch* simulator is presented in Fig. 4.3.

The basic set of the input data for the simulator includes:

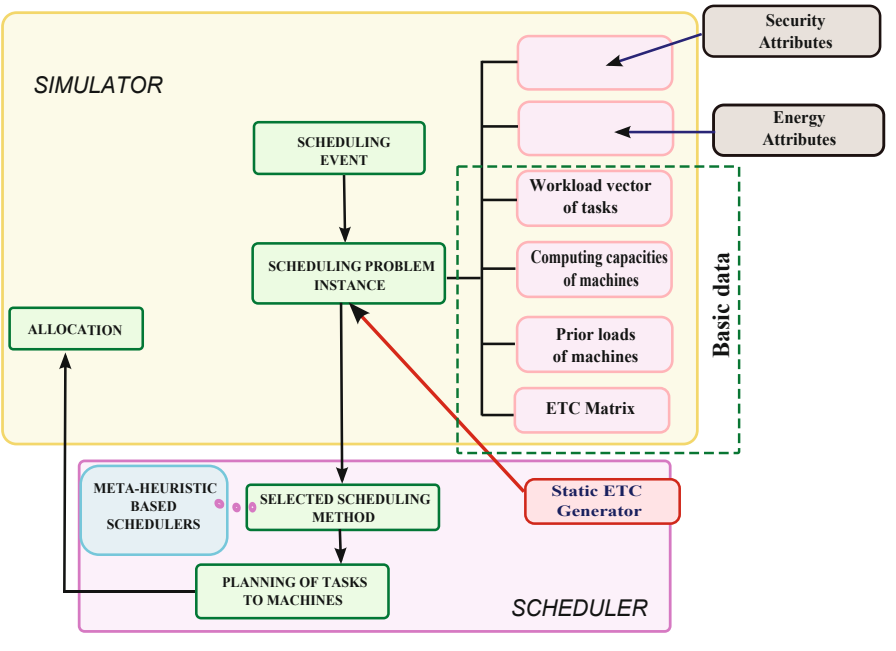


Fig. 4.3. General Flowchart of Security- and Energy-Aware *Sim-G-Batch* Simulator Linked to Scheduling

Table 4.3. Values of key parameters of the grid simulator in static and dynamic cases

	Medium	**Very Large**
Static case		
Nb. of hosts	64	256
Resource cap.	$N(5000, 875)$	
Total nb. of tasks	1024	4096
Workload of tasks	$N(250000000, 43750000)$	
Dynamic case		
Init. hosts	64	256
Max. hosts	70	264
Min. hosts	50	240
Resource cap.	$N(5000, 875)$	
Add host	$N(562500, 84375)$	$N(437500, 65625)$
Delete host	$N(625000, 93750)$	
Init. tasks	768	3072
Total tasks	1024	4096
Workload	$N(250000000, 43750000)$	
Both cases		
Security demands sd_j	$U[0.6; 0.9]$	
Trust levels tl_i	$U[0.3; 1]$	
Failure coefficient α	3	

- the workload vector of tasks,
- the computing capacity vector of machines,
- the vector of prior loads of machines, and
- the ETC matrix of estimated execution times of tasks on machines.

This set is extended by SD and Tl vectors (see Sec. 4.2) for the specification of the main security parameters and the following "energy" attributes:

- machine categories specification parameters (number of classes, maximal computational capacity value, computational capacity ranges interval for each class, machine operational speed parameter for each class, etc.);
- DVFS levels matrix for machine categories.

The simulator is highly parametrized in order to reflect numerous realistic grid scenarios. All schedulers are decoupled from the simulator main body and are implemented as the external libraries. The performance of all considered GA-based metaheuristics has been analyzed in two grid size scenarios: Medium with 64 hosts and 1024 tasks, and Very Large with 256 hosts and 4096 tasks. The capacity of the resources and the workload of tasks are randomly generated by Gaussian distributions.

The sample values of key input parameters used in the experiments for the simulator are presented in Table 4.3[4].

[4] The following notation $U(a,b)$ and $N(a,b)$ is used for uniform and Gaussian probability distributions, respectively.

Table 4.4. GA setting for large static and dynamic benchmarks

Parameter	Elitist Generational/Struggle	Steady State
degree of branches (t)	0	
period_of_metaepoch (α)	$1/2*n/10$	$(2*n/10$
nb_of_metaepochs	10	
population size (pop_size)	$\lceil(\log_2 n)^2 - \log_2 n\rceil$	$4*(\log_2 n - 1)$
intermediate pop.	$pop\_size - 2$	$(pop\_size)/3$
cross probab.	0.8	1.0
mutation probab.	0.2	
max_time_to_spend	40 sec. (*static*) / 75 sec. (*dynamic*)	

The machines can work at 16 DVFS levels and can be categorized into three "energetic" resource classes, Class I, Class II, and Class III. The class identifiers have been selected randomly for the machines. The values of supply voltages and relative machine frequencies at all DVFS levels are specified in Table 4.1. The key parameters for the genetic schedulers are presented in Table 4.4.

4.6.1 Results

Each experiment was repeated 30 times under the same configuration of operators and parameters. The results for *Makespan*, *Flowtime*, failure rate and relative energy consumption improvement rate averaged in 30 runs of the simulator are presented in Tables 4.5–4.11.

It follows from the results that the quality of scheduling strongly depends on the proper combination of crossover and mutation operations. In all considered instances the *PMX* crossover together with *Move* mutation give the worst results and the combination of *CX* crossover with *Rebalancing* mutation seems to be the most effective in the most of the instances. Indeed, in the static case $GA-CX-R-ST$ algorithm ranks first in about 75% of instances and $GA-CX-R-SS$ algorithm is the best in the remaining 25%of the total instances. In the dynamic scenario the situation is similar: $GA-CX-R-ST$ algorithm achieves the best results in about 62.5% of instances and $GA-CX-R-SS$ – in the remaining 27.5%. The $GA-CX-R-EG$ algorithm ranks in all cases as the second or third best scheduler. I can be observed that in the groups of algorithm with the same mutation and crossover operators, the *Struggle* replacement mechanism has the best positive impact on the algorithm performance. The average failure rates for the considered algorithms are in range 37% – 54%, and the energy improvement rate is in range 8% – 36%.

Table 4.5. Average *Makespan* values for eighteen GA-based schedulers in **secure** scenario [±*s.d.*], (*s.d.* = standard deviation)

	Static		Dynamic	
Strategy	**Medium**	**Very Large**	**Medium**	**Very Large**
GA-PMX-M-SS	4165583.135 [± 690668.957]	4288022.184 [± 824291.171]	4294782.139 [± 911521.151]	4466888.391 [± 560336.505]
GA-PMX-M-EG	4177553.793 [± 190066.130]	4301234.468 [± 451450.111]	4279562.342 [± 375819.231]	4485562.553 [± 750553.639]
GA-PMX-M-ST	4157425.782 [± 379356.512]	4295435.433 [± 515425.756]	4278653.544 [± 489524.252]	4435334.722 [± 364469.977]
GA-PMX-S-SS	4126691.373 [± 7592355.678]	4268733.837 [± 575235.535]	4202645.677 [± 5733466.268]	4418525.678 [± 953456.457]
GA-PMX-S-EG	4148405.275 [± 43761.015]	4285671.797 [± 605962.237]	4216301.090 [± 53486.153]	4421427.663 [± 701992.116]
GA-PMX-S-ST	4133822.183 [± 380836.642]	4261593.241 [± 634705.248]	4199261.431 [± 555304.633]	439189.653 [± 711735.974]
GA-PMX-R-SS	4099722.422 [± 939509.617]	4209834.534 [± 505720.705]	4122903.467 [± 149437.882]	4332736.028 [± 563164.075]
GA-PMX-R-EG	4111018.744 [± 837452.949]	4217551.633 [± 540610.017]	4161359.893 [± 590881.527]	4375643.075 [± 748754.806]
GA-PMX-R-ST	4096915.832 [± 653833.933]	4202740.834 [± 468878.756]	4109927.347 [± 997874.784]	4313319.834 [± 752973.821]
GA-CX-M-SS	4057635.783 [± 711365.726]	4175408.673 [± 796525.362]	4034538.827 [± 933600.267]	4284402.632 [± 684131.232]
GA-CX-M-EG	4067320.534 [± 654607.977]	4184339.256 [± 582147.023]	4090546.734 [± 751364.567]	4303769.235 [± 952170.387]
GA-CX-M-ST	404176222.362 [± 987536.453]	4167744.674 [± 803122.735]	4009105.874 [± 679674.456]	4241630.356 [± 728082.735]
GA-CX-S-SS	3954447.634 [± 886725.393]	4125317.764 [± 671196.560]	39872916.546 [± 998157.753]	4205709.634 [± 564696.828]
GA-CX-S-EG	4004965.356 [± 918998.520]	4156258.356 [± 713229.061]	3990734.764 [± 594713.746]	4222631.465 [± 957740.279]
GA-CX-S-ST	3940265.563 [± 989958.718]	4098563.182 [± 455636.448]	41614537.863 [± 56051.579]	4219725.327 [± 988429.144]
GA-CX-R-SS	392372.215 [± 384195.783]	405397.326 [± 697330.373]	3985763.378 [± 581510.863]	4152623.367 [± 712805.735]
GA-CX-R-EG	3912183.536 [± 341357.564]	4095293.987 [± 981563.633]	4093277.356 [± 754356.222]	4197426.546 [± 774433.987]
GA-CX-R-ST	**3907233.453 [± 552722.245]**	**3982865.453 [± 827733.653]**	**3900433.453 [± 368863.563]**	**4100641.722 [± 704973.453]**

Table 4.6. Average *Makespan* values for eighteen GA-based schedulers in **risky** scenario [$\pm s.d.$], ($s.d.$ = standard deviation)

	Static		Dynamic	
Strategy	**Medium**	**Very Large**	**Medium**	**Very Large**
GA-PMX-M-SS	4183064.456 [± 303594.681]	41996544.866 [± 633541.835]	4272342.367 [± 523861.891]	4351579.387 [± 955889.372]
GA-PMX-M-EG	4196660.637 [± 795391.867]	4207559.346 [± 673312.285]	4250769.729 [± 570063.523]	4378943.912 [± 663276.562]
GA-PMX-M-ST	41666682.922 [± 557522.757]	4182062.259 [± 659041.716]	4242506.997 [± 564267.992]	4342520.961 [± 435640.377]
GA-PMX-S-SS	4150255.659 [± 364204.730]	4156125.373 [± 612692.293]	4239582.165 [± 566626.175]	4323745.969 [± 685834.282]
GA-PMX-S-EG	4149105.038 [± 440578.157]	416379.853 [± 472235.032]	4220166.253 [± 837511.681]	4309346.443 [± 515978.598]
GA-PMX-S-ST	4133090.502 [± 271571.051]	4177107.429 [± 473557.935]	4198308.287 [± 492853.075]	4294855.678 [± 430583.789]
GA-PMX-R-SS	4126676.780 [± 728966.192]	4118808.678 [± 649630.565]	4164756.208 [± 56766.709]	4254796.820 [± 580607.466]
GA-PMX-R-EG	4129602.691 [± 270662.077]	4109778.902 [± 913428.430]	4182790.083 [± 761523.938]	4270973.780 [± 100219.228]
GA-PMX-R-ST	4063022.741 [± 430715.830]	4077928.331 [± 410172.944]	4158207.631 [± 611305.330]	4232768.084 [± 499217.899]
GA-CX-M-SS	4096232.073 [± 554669.949]	4093218.110 [± 602133.299]	4114422.323 [± 721311.202]	4188744.263 [± 672746.995]
GA-CX-M-EG	4109712.181 [± 869717.384]	4099788.995 [± 453095.636]	4141965.814 [± 817008.729]	4209289.242 [± 636853.293]
GA-CX-M-ST	4054429.908 [± 464589.755]	4070157.151 [± 446428.587]	4127500.870 [± 870899.646]	4198327.481 [± 631520.079]
GA-CX-S-SS	4022572.196 [± 516330.313]	4032408.549 [± 590815.709]	4090392.967 [± 898536.995]	4129281.695 [± 917216.254]
GA-CX-S-EG	4015442.255 [± 405905.662]	4033181.392 [± 650601.916]	4106452.250 [± 982489.800]	4164853.468 [± 785931.696]
GA-CX-S-ST	4006934.189 [± 436482.946]	4010678.953 [± 516992.488]	4039201.986 [± 816514.067]	4140525.379 [± 633513.047]
GA-CX-R-SS	3986872.198 [± 806632.171]	3972483.354 [± 618035.814]	4024651.986 [± 436431.848]	4109893.365 [± 747400.818]
GA-CX-R-EG	3999111.928 [± 692024.001]	4002987.835 [± 467779.063]	4089790.735 [± 632965.292]	4091857.736 [± 791598.934]
GA-CX-R-ST	**3955725.386 [± 911937.937]**	**39677356.846 [± 879753.092]**	4030546.634 [± 850836.928]	**4038634.546 [± 932835.453]**

Table 4.7. Average *Flowtime* values for eighteen GA-based schedulers in **secure** scenario [$\pm s.d.$], ($s.d.$ = standard deviation)

	Static		Dynamic	
Strategy	**Medium**	**Very Large**	**Medium**	**Very Large**
GA-PMX-M-SS	2193567211.836 [± 128370293.469]	8309597733.826 [± 291748525.988]	2399334723.236 [± 411586354.582]	8401592320.927 [± 889213844.016]
GA-PMX-M-EG	2206734272.356 [± 135504556.429]	8326428289.314 [± 13744410.910]	2404322051.418 [± 439654339.019]	8410389312.587 [± 989122999.052]
GA-PMX-M-ST	2182232499.779 [± 218224825.514]	8297351808.190 [± 207359074.643]	2357752153.775 [± 476389087.741]	8395922739.597 [± 606291276.896]
GA-PMX-S-SS	2158140985.025 [± 180302378.918]	8261813008.415 [± 240332696.510]	2316860418.277 [± 593778909.154]	8367766411.152 [± 755794904.240]
GA-PMX-S-EG	2175542598.454 [± 162987312.683]	828568131.930 [± 369229026.877]	2332618330.048 [± 45367872.253]	8388110055.663 [± 507151151.344]
GA-PMX-S-ST	2147247760.724 [± 211590465.457]	8271025623.997 [± 453137510.918]	2320515589.851 [± 711100452.764]	8373206007.045 [± 784801239.205]
GA-PMX-R-SS	21142225326.145 [± 240067553.885]	8249643747.642 [± 253050662.960]	2286055243.299 [± 577523722.350]	8338732539.636 [± 741000998.450]
GA-PMX-R-EG	2136071183.723 [± 143892082.090]	8258614783.371 [± 233840747.951]	2292713481.576 [± 377833946.994]	835216627.023 [± 699126484.932]
GA-PMX-R-ST	2117878614.800 [± 196255094.628]	8227564670.521 [± 225294411.337]	2283126157.824 [± 543390178.534]	8322161405.019 [± 736033399.656]
GA-CX-M-SS	2105676446.564 [± 18596758.737]	8195359732.256 [± 282655389.940]	2263563557.141 [± 539894530.830]	8294397268.110 [± 790327708.149]
GA-CX-M-EG	2125655077.793 [± 142293067.762]	8238431289.893 [± 212224990.911]	2274715011.673 [± 604531703.308]	8306547838.652 [± 812904982.726]
GA-CX-M-ST	21111848829.461 [± 110335889.154]	8204221008.299 [± 254605827.121]	2238467253.243 [± 513415557.850]	8272449832.295 [± 822664629.425]
GA-CX-S-SS	2084338418.886 [± 165598016.948]	8179625719.400 [± 250682728.242]	2204403080.180 [± 430849430.300]	8242964769.102 [± 600614818.712]
GA-CX-S-EG	2097868914.479 [± 17614361.064]	8190763974.690 [± 334857074.927]	2227164271.644 [± 473657175.059]	8260315502.581 [± 569231789.391]
GA-CX-S-ST	2078132035.030 [± 120473769.993]	8166781141.473 [± 250660461.860]	2195675260.221 [± 577482551.646]	8253290991.790 [± 769767684.495]
GA-CX-R-SS	2060575340.426 [± 123314394.677]	8138208217.698 [± 772885985.554]	**2177096342.984 [± 489373983.094]**	**8211987409.256 [± 999265736.534]**
GA-CX-R-EG	2066071183.009 [± 103578848.015]	8143852396.768 [± 272146853.672]	2196929745.169 [± 533701714.987]	8235408309.560 [± 795752579.317]
GA-CX-R-ST	**2035128734.875 [± 55234984.937]**	**8109087253.984 [± 339791854.937]**	2199758911.356 [± 436450229.234]	8221050176.985 [± 708270871.387]

Table 4.8. Average *Flowtime* values for eighteen GA-based schedulers in **risky** scenario [$\pm s.d.$], ($s.d.$ = standard deviation)

	Static		Dynamic	
Strategy	**Medium**	**Very Large**	**Medium**	**Very Large**
GA-PMX-M-SS	2343724728.245 [± 598538835.360]	8321846250.347 [± 431346593.814]	2413245472.632 [± 417986755.794]	8644534678.245 [± 404482983.284]
GA-PMX-M-EG	2389239424.349 [± 129097532.581]	8347268532.324 [± 432955978.981]	2436061767.548 [± 701148099.631]	8660435684.376 [± 664054585.165]
GA-PMX-M-ST	2307355142.051 [± 126906586.696]	8308727439.878 [± 256156869.233]	2403766189.879 [± 469026059.421]	8605175251.450 [± 693219859.321]
GA-PMX-S-SS	2263649999.867 [± 268317752.960]	8262608448.916 [± 216787358.248]	2360160544.425 [± 656197729.295]	8502620979.464 [± 851502055.485]
GA-PMX-S-EG	2286784630.430 [± 184783757.268]	8292058041.397 [± 499461667.383]	2393131389.346 [± 454805534.425]	8531596508.841 [± 573812983.895]
GA-PMX-S-ST	2237247760.724 [± 244213856.135]	8271025623.997 [± 434633012.674]	2377515589.851 [± 788991696.341]	8513206007.045 [± 719513808.715]
GA-PMX-R-SS	2226429310.580 [± 255806730.308]	8229357685.290 [± 290693183.985]	2333974209.902 [± 519849125.536]	8472673073.563 [± 743945636.397]
GA-PMX-R-EG	2213613356.567 [± 167571940.967]	8254728732.731 [± 283645546.676]	2348829866.347 [± 526061100.837]	8493687750.198 [± 784339554.900]
GA-PMX-R-ST	2151930817.794 [± 134663685.381]	8184860096.988 [± 242728273.618]	2320407124.381 [± 528711756.114]	8466099314.428 [± 728490457.382]
GA-CX-M-SS	2188284638.934 [± 162112136.850]	8199577483.835 [± 257276315.811]	2239455642.778 [± 387064110.517]	8426829568.357 [± 537095310.247]
GA-CX-M-EG	2193593276.050 [± 163978670.974]	8211445673.176 [± 173050142.794]	2296573568.612 [± 621630992.871]	8457381627.647 [± 596961998.050]
GA-CX-M-ST	2132978230.586 [± 129547599.824]	8166440897.331 [± 182827503.146]	2270646610.327 [± 616490962.213]	8432573052.527 [± 885691236.077]
GA-CX-S-SS	2126523158.454 [± 228154433.283]	8161517451.690 [± 277432894.907]	2222289600.939 [± 552469744.108]	8365526025.239 [± 773215992.243]
GA-CX-S-EG	2113772162.066 [± 187159613.747]	8154839647.617 [± 326230384.931]	2233669538.605 [± 620551577.626]	8392873916.020 [± 859689519.119]
GA-CX-S-ST	2106340445.440 [± 138247946.733]	8146356979.846 [± 239873730.726]	2218990182.691 [± 525735729.121]	8345528446.458 [± 947608913.414]
GA-CX-R-SS	2076534164.177 [± 181982147.103]	**81127123653.774 [± 19799961.957]**	221654378.495 [± 48988254.993]	**8291082340.932 [± 834522826.826]**
GA-CX-R-EG	2098943746.287 [± 143403467.472]	8138965387.563 [± 176627923.813]	2208567534.205 [± 564619337.921]	8339386800.606 [± 730340037.273]
GA-CX-R-ST	**2020734590.928 [± 2393872683.029]**	8122230062.891 [± 217213422.096]	**2168923672.647 [± 8893693282.361]**	8310330051.728 [± 711071173.232]

Table 4.9. Average $Fail_r$ values for eighteen GA-based schedulers in **secure** scenario [$\pm s.d.$], ($s.d.$ = standard deviation)

	Static		Dynamic	
Strategy	**Medium**	**Very Large**	**Medium**	**Very Large**
GA-PMX-M-SS	44.67 [± 7.9]	49.40 [± 8.9]	43.81 [± 8.12]	47.12 [± 6.05]
GA-PMX-M-EG	47.83 [± 9.82]	47.01 [± 5.54]	44.80 [± 8.19]	48.63 [± 7.50]
GA-PMX-M-ST	43.547 [± 7.91]	47.95 [± 7.56]	40.77 [± 5.22]	48.37 [± 6.46]
GA-PMX-S-SS	42.99 [± 9.92]	48.68 [± 6.75]	44.04 [± 9.79]	47.22 [± 8.56]
GA-PMX-S-EG	43.48 [± 7.61]	49.67 [± 6.09]	44.16 [± 8.53]	47.66 [± 7.99]
GA-PMX-S-ST	41.82 [± 7.98]	48.93 [± 9.23]	43.26 [± 5.20]	49.11 [± 7.73]
GA-PMX-R-SS	43.09 [± 9.37]	48.55 [± 5.70]	43.90 [± 9.43]	48.74 [± 5.65]
GA-PMX-R-EG	43.07 [± 8.37]	46.47 [± 5.41]	43.69 [± 5.92]	48.93 [± 7.48]
GA-PMX-R-ST	42.96 [± 5.53]	47.39 [± 6.88]	43.06 [± 9.72]	48.74 [± 5.52]
GA-CX-M-SS	42.78 [± 7.15]	47.93 [± 7.25]	42.53 [± 9.25]	48.10 [± 6.93]
GA-CX-M-EG	41.96 [± 6.54]	48.39 [± 8.22]	42.94 [± 7.3]	49.11 [± 9.57]
GA-CX-M-ST	41.64 [± 9.77]	47.91 [± 8.64]	42.12 [± 6.79]	48.41 [± 7.24]
GA-CX-S-SS	42.54 [± 5.66]	45.65 [± 7.71]	43.28 [± 6.15]	48.65 [± 9.69]
GA-CX-S-EG	42.84 [± 9.55]	44.26 [± 9.30]	40.45 [± 9.71]	48.22 [± 9.52]
GA-CX-S-ST	38.66 [± 9.59]	46.08 [± 7.58]	39.47 [± 6.5]	48.27 [± 8.98]
GA-CX-R-SS	39. 13 [± 8.57]	43.34 [± 9.78]	41.67 [± 8.15]	46.12 [± 7.18]
GA-CX-R-EG	39.72 [± 6.56]	43.75 [± 6.77]	41.03 [± 9.46]	44.97 [± 7.84]
GA-CX-R-ST	**37.24 [± 4.27]**	**42.20 [± 6.92]**	**38.33 [± 9.88]**	**43.05 [± 7.88]**

Table 4.10. Average $Fail_r$ values for eighteen GA-based schedulers in **risky** scenario [$\pm s.d.$], ($s.d.$ = standard deviation)

	Static		Dynamic	
Strategy	**Medium**	**Very Large**	**Medium**	**Very Large**
GA-PMX-M-SS	44.29 [± 6.82]	51.99 [± 6.43]	45.66 [± 5.29]	53.65 [± 9.52]
GA-PMX-M-EG	44.68 [± 7.95]	52.07 [± 6.68]	45.50 [± 6.02]	53.72 [± 9.92]
GA-PMX-M-ST	44.92 [± 7.82]	51.80 [± 6.83]	45.49 [± 9.94]	53.42 [± 4.35]
GA-PMX-S-SS	44.02 [± 9.43]	51.88 [± 7.39]	45.39 [± 6.68]	53.21 [± 6.83]
GA-PMX-S-EG	43.28 [± 10.02]	51.99 [± 7.37]	45.12 [± 8.93]	53.09 [± 8.87]
GA-PMX-S-ST	43.94 [± 8.21]	51.77 [± 5.36]	44.98 [± 5.83]	52.94 [± 7.66]
GA-PMX-R-SS	43.91 [± 8.99]	51.18 [± 5.84]	44.64 [± 5.67]	52.54 [± 5.87]
GA-PMX-R-EG	43.99 [± 6.29]	51.99 [± 9.34]	44.83 [± 7.69]	52.53 [± 10.02]
GA-PMX-R-ST	43.64 [± 8.55]	52.03 [± 9.64]	43.93 [± 9.34]	52.01 [± 5.38]
GA-CX-M-SS	44.01 [± 9.22]	51.93 [± 8.33]	43.14 [± 7.74]	51.88 [± 9.95]
GA-CX-M-EG	43.95 [± 8.69]	51.29 [± 7.36]	43.02 [± 8.17]	52.09 [± 5.73]
GA-CX-M-ST	42.99 [± 10.13]	50.92 [± 8.63]	42.85 [± 8.70]	51.98 [± 7.92]
GA-CX-S-SS	42.22 [± 5.98]	50.78 [± 5.83]	42.99 [± 6.89]	51.96 [± 9.67]
GA-CX-S-EG	41.97 [± 7.03]	49.63 [± 9.92]	42.93 [± 8.28]	51.62 [± 8.32]
GA-CX-S-ST	41.08 [± 7.55]	50.20 [± 10.22]	43.03 [± 9.53]	51.93 [± 8.76]
GA-CX-R-SS	41.85 [± 8.06]	49.83 [± 9.53]	42.73 [± 6.72]	51.01 [± 7.47]
GA-CX-R-EG	40.59 [± 6.92]	49.21 [± 9.02]	4240 [± 7.83]	50. 39 [± 9.93]
GA-CX-R-ST	**39.88 [± 9.04]**	**47.92 [± 8.88]**	41.92 [± 9.24]	**50.38 [± 7.98]**

Table 4.11. Average $Im(E)$ values for eighteen GA-based schedulers in **secure** scenario [$\pm s.d.$], ($s.d.$ = standard deviation)

	Static		Dynamic	
Strategy	**Medium**	**Very Large**	**Medium**	**Very Large**
GA-PMX-M-SS	17.65 [± 2.49]	13.95 [± 2.94]	15.99 [± 4.1]	12.01 [± 1.89]
GA-PMX-M-EG	17.99 [± 3.55]	15.24 [± 1.44]	16.44 [± 2.39]	13.10 [± 1.82]
GA-PMX-M-ST	18.10 [± 2.18]	15.37 [± 2.07]	16.35 [± 2.47]	12.95 [± 1.93]
GA-PMX-S-SS	18.03 [± 3.09]	15.61 [± 2.40]	16.31 [± 3.59]	13.06 [± 5.50]
GA-PMX-S-EG	18.73 [± 4.26]	15.85 [± 3.62]	16.32 [± 3.45]	13.80 [± 2.50]
GA-PMX-S-ST	19.67 [± 4.98]	16.10 [± 4.51]	16.25 [± 3.11]	13.73 [± 2.78]
GA-PMX-R-SS	19.65 [± 2.40]	16.49 [± 2.53]	16.86 [± 3.7]	13.38 [± 2.74]
GA-PMX-R-EG	19.56 [± 3.73]	16.36 [± 3.26]	17.28 [± 4.01]	15.11 [± 4.28]
GA-PMX-R-ST	19.76 [± 3.75]	18.63 [± 2.29]	19.04 [± 5.99	15.77 [± 3.74]
GA-CX-M-SS	20.564 [± 3.85]	19.53 [± 2.82]	18.63 [± 1.99]	15.94 [± 3.74]
GA-CX-M-EG	21.23 [± 5.89]	20.43 [± 2.12]	18.36 [± 6.04]	17.98 [± 3.12]
GA-CX-M-ST	23.84 [± 3.98]	22.12 [± 4.39]	18.37 [± 5.18]	16.28 [± 4.02]
GA-CX-S-SS	25.70 [± 6.33]	20.83 [± 7.80]	19.27 [± 4.85]	19.02 [± 6.00]
GA-CX-S-EG	27.72 [± 2.32]	21.90 [± 7.34]	20.82 [± 4.29]	19.99 [± 4.65]
GA-CX-S-ST	26.87 [± 6.83]	21.35 [± 5.25]	20.37 [± 5.77]	20.23 [± 7.06]
GA-CX-R-SS	26.99 [± 5.89]	21.02 [± 5.32]	22.87 [± 4.19]	20.39 [± 6.37]
GA-CX-R-EG	28.14 [± 4.22]	21.63 [± 5.30]	27.91 [± 8.01]	20.19 [± 5.83]
GA-CX-R-ST	**35.38 [± 5.83]**	**28.15 [± 3.79]**	**30.93 [± 4.92]**	**32.05 [± 5.28]**

Table 4.12. Average $Im(E)$ values for eighteen GA-based schedulers in **risky** scenario $[\pm s.d.]$, ($s.d.$ = standard deviation)

	Static		Dynamic	
Strategy	**Medium**	**Very Large**	**Medium**	**Very Large**
GA-PMX-M-SS	10.43 [± 1.59]	9.98 [± 1.43]	8.12 [± 1.41]	8.06 [± 1.40]
GA-PMX-M-EG	11.13 [± 1.12]	10.23 [± 1.43]	8.42 [± 1.70]	8.66 [± 1.66]
GA-PMX-M-ST	11.07 [± 0.99]	10.08 [± 1.25]	8.34 [± 1.46]	8.60 [± 1.69]
GA-PMX-S-SS	13.63 [± 1.82]	12.62 [± 1.418]	11.60 [± 1.65]	11.05 [± 1.85]
GA-PMX-S-EG	14.86 [± 1.84]	12.92 [± 1.49]	11.93 [± 1.45]	11.31 [± 1.57]
GA-PMX-S-ST	15.37 [± 2.44]	13.71 [± 4.43]	12.37 [± 1.781]	11.325 [± 1.75]
GA-PMX-R-SS	15.96 [± 2.55]	14.29 [± 2.90]	14.33 [± 3.51]	13.47 [± 3.74]
GA-PMX-R-EG	16.95 [± 1.67]	15.44 [± 2.83]	15.28 [± 3.52]	14.03 [± 4.70]
GA-PMX-R-ST	18.41 [± 3.76]	17.83 [± 2.42]	15.34 [± 4.21]	15.08 [± 3.92]
GA-CX-M-SS	18.41 [± 2.62]	16.99 [± 2.571]	14.22 [± 3.82]	13.00 [± 2.78]
GA-CX-M-EG	18.25 [± 3.97]	17.04 [± 3.05	14.83 [± 2.21]	13.40 [± 3.59]
GA-CX-M-ST	19.03 [± 2.94]	17.01 [± 3.28]	15.01 [± 3.64]	14.21 [± 1.85]
GA-CX-S-SS	19.26 [± 2.78]	18.83 [± 2.77]	17.45 [± 5.58]	15.21 [± 4.73]
GA-CX-S-EG	19.74 [± 1.87]	18.36 [± 3.29]	18.91 [± 3.22]	16.57 [± 2.81]
GA-CX-S-ST	19.34 [± 2.13]	18.35 [± 2.39]	18.78 [± 3.52]	16.98 [± 3.94]
GA-CX-R-SS	20.66 [± 2.98]	18.98 [± 1.97]	19.96 [± 2.99]	17.91 [± 3.82]
GA-CX-R-EG	20.17 [± 3.65]	19.22 [± 4.29]	20.06 [± 3.02]	17.98 [± 5.00]
GA-CX-R-ST	**21.77 [± 2.48]**	**20.20 [± 2.2]**	**21.33 [± 3.52]**	**18.03 [± 3.71]**

By summarizing the results for all the objective functions, the *GA-CX-R-ST* algorithm is the best in the minimization of all considered objectives. It should be also observed that in the risky mode almost all of the achieved results are worse than in the secure mode. It means that in the case of ignoring all security conditions the high failure rates increase the energy utilized by the system, and of course extend the deadlines for completing the particular tasks and the whole schedules.

4.7 Multi-population Genetic Grid Schedulers

Due to sheer size of the grid and the dynamics of the grid environment, the exploration of the optimization landscapes generated for the grid scheduling problems remains challenging task for simple genetic-based grid schedulers, especially in the case of many different scheduling criteria (not just *Makespan* and *Flowtime*). In such case the effectiveness of the algorithms may be improved by executing them simultaneously on many populations as parallel processes. All single-population GAs presented in Sec. 4.5 may be used as main genetic mechanisms in multi-population genetic meta-heuristics. In this work, we consider two following multi-population genetic-based risk resilient grid schedulers:

- ***HGS-Sched (R)*** – multi-population hierarchical *HGS-Sched* scheduler working in the risky mode;
- ***HGS-Sched (S)*** – multi-population hierarchical *HGS-Sched* scheduler working in the secure mode;
- ***IGA (R)*** – multi-population island GA scheduler working in the risky mode;
- ***IGA (S)*** – multi-population island GA scheduler working in the secure mode.

The main idea of the Hierarchic Genetic Scheduler (HGS-Sched) based on the concept of a concurrent search in grid environment by the execution of many dependent evolutionary processes. The strategy is modelled as a multi-level decision tree. The search process in *HGS-Sched* is initialized by activating a scheduler with low search accuracy. This scheduler is the main module of the entire strategy and is responsible for the "management" of the general structure of the tree[5] and exploration of new and unrecognized regions in the optimization domain. The accuracy of search in *HGS-Sched* branches is defined by the *degree* parameter, and depends on the mutation rate and in fact the size of the population, which can be different in the branches of different degrees.

Figure 4.4 depicts a simple graphical representation of 3-level structure of *HGS-Sched*.

New branches are sprouted by using the sprouting procedure SO after execution of α-generation evolutionary processes (α-*periodic metaepoch* Met_α). The extension of the tree is steered by using the *Branch Comparison* (BC). The detailed definition and interpretation of all *HGS-Sched* operators can be found in [29].

[5] The scheduler with the lowest accuracy of search is called *the core* of the HGS tree structure.

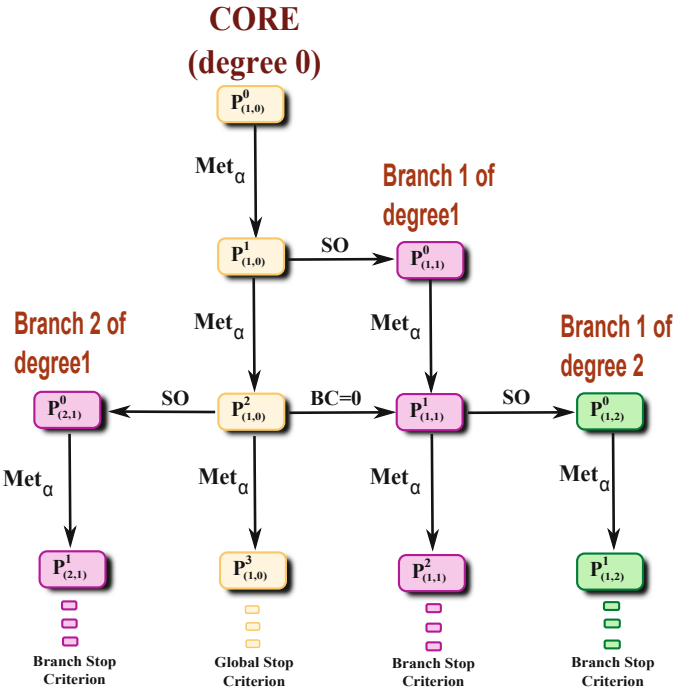


Fig. 4.4. 3 levels of *HGS-Sched* tree structure

The *Island Genetic Algorithm (IGA)* [48] is a well-known parallel GA technique. An initial (usually large) population is divided into several sub-populations, "islands" or "demes", for which single-population GAs with identical configurations of the parameters and operators are activated (one algorithm for each deme). After a fixed number of iterations, denoted as it_d, the migration procedure is activated. It enables a partial exchange (usually according to the standard ring topology) of the individuals among islands. The relative amount of the migrating individuals, represented by *mig*, is the algorithm global parameter commonly known as the *migration rate*, and calculated as:

$$mig = \frac{m_{deme}}{deme} \cdot 100\% \tag{4.45}$$

where *deme* is the size of the sub-population in *IGA* and m_{deme} is the number of migrating individuals in each deme. The procedure of migration is repeated after each execution of it_d iterations of genetic algorithm in each sub-population.

The $GA-CX-R-ST(R)$ and $GA-CX-R-ST(S)$ algorithms, selected as the most effective single-population schedulers in Sec. 4.6, have been used as the main genetic mechanisms in all considered implementation of *HGS-Sched* (in all types of branches) and *IGA*.

4.7.1 Empirical Analysis

In this section we present results of the empirical evaluation of two types of multi-population genetic schedulers and compare them with the results achieved by $GA-CX-R-ST(R)$ and $GA-CX-R-ST(S)$ algorithms. Similarly to the previous experiments, we have used the*Sim-G-Batch* grid simulator with the same configuration as specified in Sec. 4.6. The configurations of key parameters for both implementations of $GA-CX-R-ST(x)$ algorithms are the same as in the experiment provided for all types of single-population schedulers (see Table 4.4), the parameters for *IGA* and *Green-HGS-Sched* meta-heuristics are presented in Tables 4.13 and 4.14.

4.7.2 Results

Each experiment was repeated 30 times under the same configuration of operators and parameters. The box-plots of the statistical analysis of the values of the four scheduler performance measures averaged in 30 runs of the simulator are presented in Fig. 4.5–4.8.

In the case of $Makespan$, $Flowtime$ and $Fail_r$ the best results are achieved by the $HGS-Sched(S)$ algorithm. Especially in the dynamic grid the difference in the results generated by this scheduler and the the other meta-heuristics are significant. In the case of "Medium" grid the single-population $GA-CX-R-ST(S)$ algorithm is more effective in the $Makespan$ minimization that the secure implementation of the island model, which shows that in this case the best found solutions of the problem may be located very close to each other, and the island procedure do not improve the quality of search of the single-population scheduling mechanism implemented in each deme of *IGA* . In all of scheduling scenarios the secure implementations of the algorithms from the same class are more effective than their risky implementations for all three above mentioned scheduling criteria.

The situation is a bit different in the case of the minimization of the energy consumption. In this case the risky implementations of *HGS-Sched* and *IGA* algorithms

Table 4.13. *HGS-Sched* settings for static and dynamic benchmarks

Parameter	
period_of_metaepoch	$20*n$
nb_of_metaepochs	10
degrees of branches (t)	0 and 1
population size in the core	$3*(\lceil 4*(\log_2 n-1)/(11.8)\rceil)$
population size in the sprouted branches (b_pop_size)	$(\lceil (4*(\log_2 n-1)/(11.8)\rceil)$
intermediate pop. in the core	$abs((r\_pop\_size)/3)$
intermediate pop. in the sprouted branch	$abs((b\_pop\_size)/3)$
cross probab.	0.9
mutation probab. in core	0.4
mutation probab. in the sprouted branches	0.2
max_time_to_spend	40 secs (*static*) / 70 secs (*dynamic*)

Table 4.14. Configuration of *IGA* algorithm

Parameter	
it_d	$20 * n$
mig	5 %
number of islands (demes)	10
cross probab.	1.0
mutation probab.	0.2
max_time_to_spend	40 secs (*static*) / 70 secs (*dynamic*)

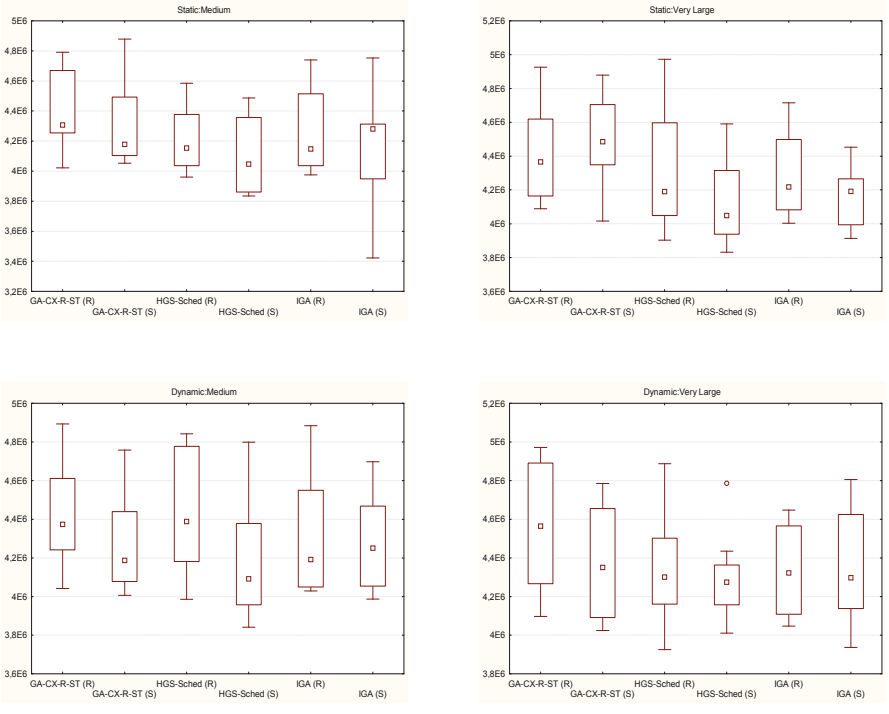


Fig. 4.5. The box-plot of the results for *Makespan* in static and dynamic scenarios

achieved the best (highest) values of the energy improvement rates in the "Medium" and "Very large" grids respectively. This may leads to the conclusion that the lowering the power supply of the system in those cases didn't increase the *Makespan* and *Flowtime* values, which are rather big for those two schedulers, but allowed to reduce significantly the energy utilization.

It can be observed that *HGS-Sched (S)* is the most stable scheduler in all considered scenarios. The differences between the minimal and maximal values, and the first and the third quantiles are the lowest for this meta-heuristic.

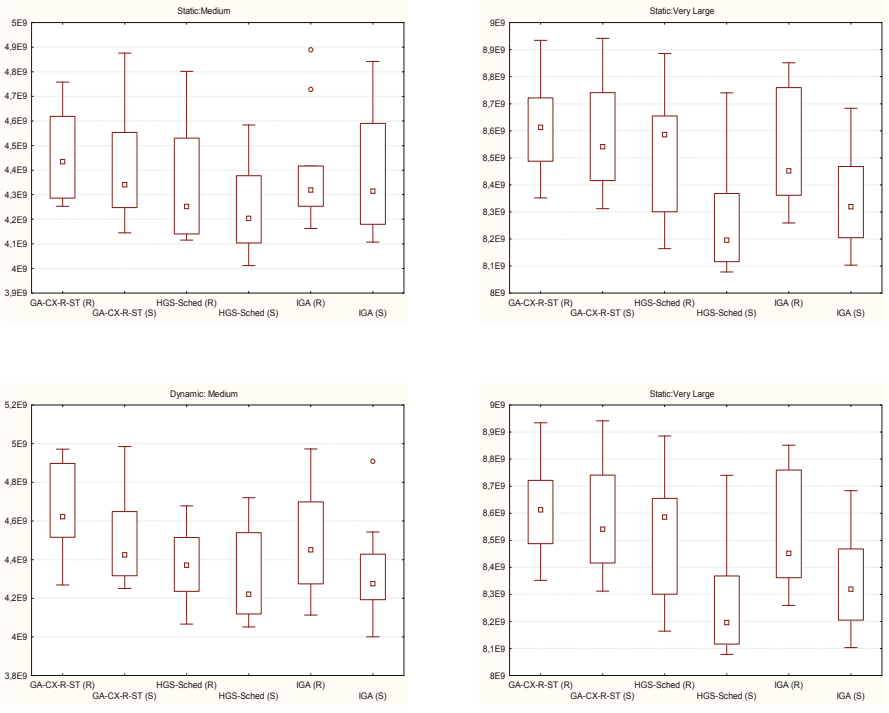


Fig. 4.6. The box-plot of the results for *Mean_Flowtime* in static and dynamic scenarios

4.8 Related Work

The security and resource reliability issues have been intensively studied in numerous publications over the last years [41]. However, many well-known scheduling approaches for grid computing largely ignore the security factor, with only a handful of exceptions.

A lot of efforts have been made on developing the intelligent management modules in the grid systems for controlling the accessing services through the authentication [20] in online scheduling, or for monitoring the execution of the grid applications and detection the resource failures due to the security restrictions [21]. In [1] developed a model, in which jobs are replicated at multiple grid sites to improve the probability of the satisfaction of the security requirements and successful job executions.

Although the security-aware scheduling in grids is well studied, there are not so many examples of successful application of the genetic-based meta-heuristics for solving this problem. In most of the publications in the domain security and task abortion mechanisms are usually implemented as the external procedures separated from the core of the scheduling system. In [47] the authors defines the security as additional scheduling criterion in online grid scheduling and aggregated this criterion

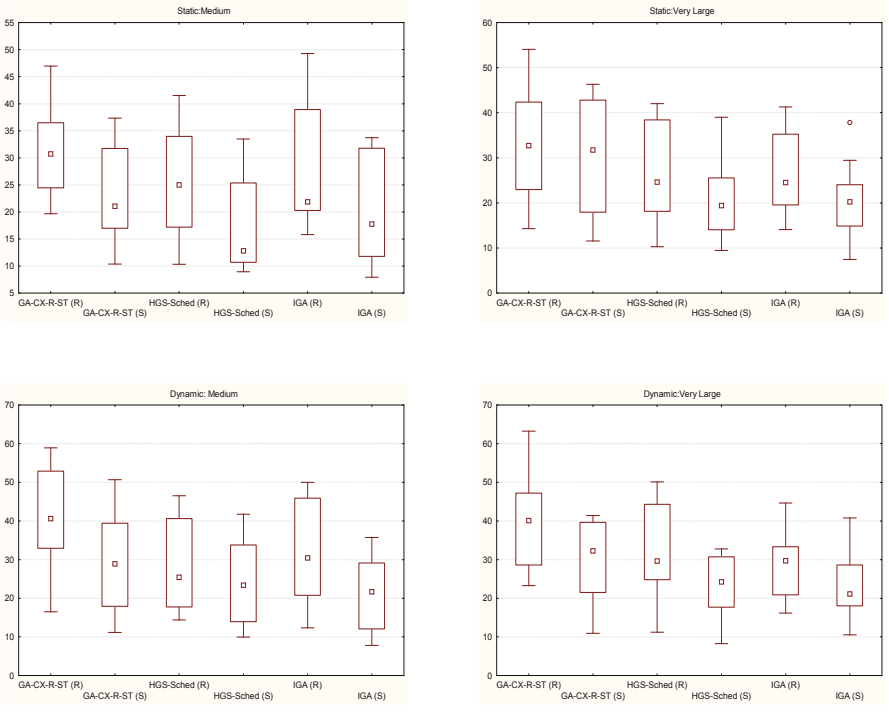


Fig. 4.7. The box-plot of the results for $Fail_r$ parameter in static and dynamic scenarios

with conventional scheduling objective functions such as *Makespan* and resource utilization.

One of the promising security-aware approaches in grid scheduling are based on the game-theoretical models. In [46] and [47] the authors considered the risky and insecure conditions in online scheduling in CGs caused by software vulnerability and distrusted security policy. They apply the game model introduced in [33] for simulating the resource owners selfish behavior. The results presented in [47] are extended by Wu et al. in [49]. The authors consider the heterogeneity of fault-tolerance mechanism in a security-assured Grid job scheduling and define four types of GA-based online schedulers for the simulation of fault-tolerance mechanisms. The other game-theoretical approaches are presented in [31] and [30]. In these papers the scheduling problem has been interpreted as a difficult decision problem for grid users working at different levels of the system. Users decisions and fundamental features arising in the users' behavior, such as cooperativeness, trustfulness and symmetric and asymmetric roles, were modelled based on the game theory paradigm. Another game-based model for security-aware grid scheduling is defined in [16]. The authors developed a node mobility prediction model for mobile grid systems and used the predetermined fair pricing strategies for secure access the mobile grid nodes. In all of the aforementioned models the final decisions on the

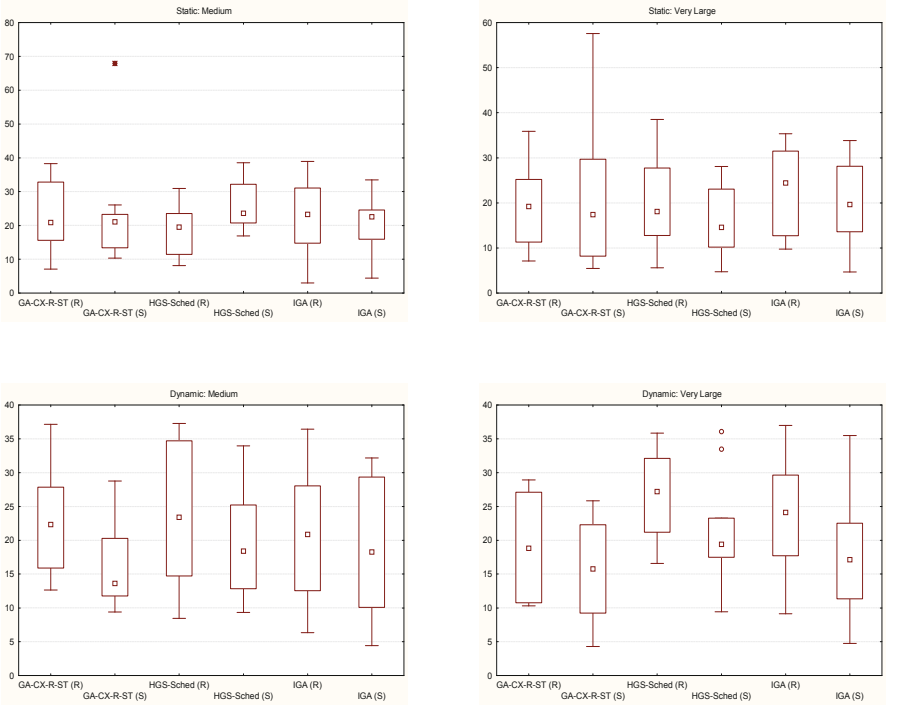


Fig. 4.8. The box-plot of the results for *Energy Improvement Rate* in static and dynamic scenarios

secure allocation of task to resources are made by the grid users who do not cooperate with each other. The costs of the risk-resilient tasks executions are interpreted as the users' cost functions, which are specified as the scheduling objectives and are minimized during the game. The main drawback of these approaches is their computational complexity. In many cases the games are provided on the different grid levels and to define an effective synchronization mechanism is a challenging task.

A significant volume of research has been done in the domain of energy aware resource management in modern large-scale distributed computational systems. Khan and Ahmad [25] have used the game theoretical methodologies to simultaneously optimize system performance and energy consumption in large data centers. Several research works have used similar models and approaches related to green computing in large-scale distributed systems, such as energy proportionality [19,24], memory-aware computations, data intensive computations [26,43], energy-efficient, and grid scheduling [35,53]. (The recent survey is presented in [8].)

There is still not so large family of energy-aware genetic-based schedulers in grid and cloud environments. In most of the DVFS approached the scheduling has been defined as classical or dynamic load balancing problem. In such cases linear, dynamic and goal programming are the major optimization techniques (see i.e. [35], [24], [23], [26]).

Shen et al. in [44] and [45] present a *shadow price* technique for solving the scheduling problems in cloud systems. The "shadow price" for a pair task-machine is defined as an average energy consumption per instruction for the processor that can operate at different voltage levels. Then the classical move and swap mutation operations are used for an optimal mapping of tasks to machines.

Kessaci et al. in [22] present two versions of multi-objective parallel Genetic Algorithm (MOPGA) hybridized with energy-conscious scheduling heuristics (ECS). The GA engine is based on the concepts of island GA and multi-start GA models. The voltage and frequencies of the processors are scaled up at 16 discrete levels and genes in GA chromosomes are defined by the task-processor labels and processor voltage. The objective function is composed of two criteria: privileged makespan and total energy consumption in the system.

The solution presented in [22] is dedicated to general computing and embedded systems. An application of such methodology in computational cloud is demonstrated by Mezmaz et al. in [38]. The energy conservation rate in cloud system is very similar to the results obtained in the general case.

Another hybrid GA approach is presented by Miao et al. in [39]. The authors propose a multi-objective genetic algorithm which is hybridized with simulated annealing for the improvement of the local solutions.

4.9 Conclusions

The main reason behind the complexity of the multi-criteria grid scheduling is that this problems consist of several interconnected components (criteria, sub-problems), that can make many standard approaches ineffective. Even if the exact and efficient algorithms for solving particular components or aspects of an overall problem are well-known, these algorithms only yield solutions to sub-problems, and it remains an open question how to integrate these partial solutions to achieve a global optimum. Meta-heuristics, due to their robustness and high scalability, are able to tackle the various and also sometimes conflicting scheduling attributes and criteria.

The comprehensive empirical analysis presented in this chapter shows the high effectiveness of the single- and multi-population genetic-based metaheuristics in optimization of the conventional grid scheduling objectives, namely *Makespan* and *Flowtime*, as well as the new criteria such as the cumulative energy consumed in the grid system and the security issues. These additional criteria are usually considered as the separate optimization problems. Here in our approach they are embedded in the proposed scheduling models. The simulated grid scenarios in such cases can better illustrate the realistic systems, in which large number of variables, numerous objectives, constraints, and business rules, all contributing in various ways must be analyzed.

All models presented in this chapter are in fact not restricted just to the grid systems. They may be easily adapted to cloud environments, where security awareness and intelligent power management are the hottest research issues.

References

1. Abawajy, J.: An efficient adaptive scheduling policy for high performance computing. Future Generation Computer Systems 25(3), 364–370 (2009)
2. Abraham, A., Buyya, R., Nath, B.: Nature's heuristics for scheduling jobs on computational grids. In: Proc. of the 8th IEEE International Conference on Advanced Computing and Communications, India, pp. 45–52 (2000)
3. Aguirre, H., Tanaka, K.: Working principles, behavior, and performance of moeas on mnk-landscapes. European Journal of Operational Research 181, 1670–1690 (2007)
4. Ali, S., Siegel, H., Maheswaran, M., Ali, S., Hensgen, D.: Task execution time modeling for heterogeneous computing systems. In: Proc. of the Workshop on Heterogeneous Computing, pp. 185–199 (2000)
5. Back, T., Fogel, D.B., Michalewicz, Z. (eds.): Evolutionary Computation, Part 1 and 2. IOP Publishing Ltd (2000)
6. Baker, R.J.: CMOS: circuit design, layout, and simulation, 2nd edn. Wiley (2008)
7. Bartschi Wall, M.:: A Genetic Algorithm for Resource-Constrained Scheduling. PhD Thesis, Massachusetts Institute of Technology, MA (1996)
8. Beloglazov, A., Buyya, R., Lee, Y., Zomaya, A.Y.: A taxonomy and survey of energy-efficient data centers and cloud computing systems. Advances in Computers 82, 47–111 (2011)
9. Brucker, P.: Scheduling Algorithms. Springer (2007)
10. Buyya, R., Murshed, M.: Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Concurrency and Computation: Practice and Experience 14(13-15), 1175–1220 (2002)
11. Carretero, J., Xhafa, F.: Using Genetic Algorithms for Scheduling Jobs in Large Scale Grid Applications. Journal of Technological and Economic Development –A Research Journal of Vilnius Gediminas Technical University 12(1), 11–17 (2006)
12. Davis, L. (ed.): Handbook of Genetic Algoriithms. Van Nostrand Reinhold, New York (1991)
13. Fibich, P., Matyska, L., Rudová, H.: Model of grid scheduling problem. In: Proc. of the AAAI 2005 Workshop on Exploring Planning and Scheduling for Web Services, Grid and Autonomic Computing (2005)
14. Garg, S., Buyya, R., Segel, H.: Scheduling parallel aplications on utility grids: Time and cost trade-off management. In: Mans, B. (ed.) Proc. of the 32nd ACSC, Wellington, Australia. CRPIT, vol. 91 (2009)
15. Goldberg, D.E., Lingle Jr., R.: Alleles, loci and the travelling salesman problem. In: Proc. of the ICA 1985, Pittsburgh (1985)
16. Gosh, P., Das, S.: Mobility-aware cost-efficient job scheduling for single-class grid jobs in a generic mobile grid architecture. Future Generation Computer Systems 26(8), 1356–1367 (2010)
17. Graham, R., Lawler, E., Lenstra, J., Rinnooy Kan, A.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics 5, 287–326 (1979)
18. Grueninger, T.: Multimodal optimization using genetic algorithms. Technical report, Department of Mechanical Engineering. MIT, Cambridge (1997)
19. Guzek, K., Pecero, J.E., Dorrosoro, B., Bouvry, P., Khan, S.U.: A cellular genetic algorithm for scheduling applications and energy-aware communication optimization. In: Proc. of the ACM/IEEE/IFIP Int. Conf. on High Performance Computing and Simulation (HPCS), Caen, France, pp. 241–248 (2010)
20. Humphrey, M., Thompson, M.: Security implications of typical grid computing usage scenarios. In: Proc. of the Conf. on High Performance Distributed Computing (2001)
21. Hwang, S., Kesselman, C.: A flexible framework for fault tolerance in the grid. J. of Grid Computing 1(3), 251–272 (2003)

22. Kessaci, Y., Mezmaz, M., Melab, N., Talbi, E.-G., Tuyttens, D.: Parallel Evolutionary Algorithms for Energy Aware Scheduling. In: Bouvry, P., González-Vélez, H., Kołodziej, J. (eds.) Intelligent Decision Systems in Large-Scale Distributed Environments. SCI, vol. 362, pp. 75–100. Springer, Heidelberg (2011)
23. Khan, S.: A goal programming approach for the joint optimization of energy consumption and response time in computational grids. In: Proc. of the 28th IEEE International Performance Computing and Communications Conference (IPCCC), Phoenix, AZ, USA, pp. 410–417 (2009a)
24. Khan, S.: A self-adaptive weighted sum technique for the joint optimization of performance and power consumption in data centers. In: Proc. of the 22nd International Conference on Parallel and Distributed Computing and Communication Systems (PDCCS), USA, pp. 13–18 (2009b)
25. Khan, S.U., Ahmad, I.: A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids. IEEE Tran. on Parallel and Distributed Systems 20(3), 346–360 (2009)
26. Kliazovich, D., Bouvry, P., Khan, S.U.: Dens: Data center energy-efficient network-aware scheduling. In: Proc. of ACM/IEEE International Conference on Green Computing and Communications (GreenCom), Hangzhou, China, pp. 69–75 (December 2010)
27. Klusaček, D., Rudová, H.: Efficient grid scheduling through the incremental schedule-based approach. Computational Intelligence 27(1), 4–22 (2011)
28. Kołodziej, J.: Evolutionary Hierarchical Multi-Criteria Metaheuristics for Scheduling in Large-Scale Grid Systems. SCI, vol. 419. Springer, Heidelberg (2012)
29. Kołodziej, J., Xhafa, F.: Enhancing the genetic-based scheduling in computational grids by a structured hierarchical population. Future Generation Computer Systems 27, 1035–1046 (2011)
30. Kołodziej, J., Xhafa, F.: Integration of task abortion and security requirements in ga-based meta-heuristics for independent batch grid scheduling. Computers and Mathematics with Applications 63, 350–364 (2011)
31. Kołodziej, J., Xhafa, F.: Meeting security and user behaviour requirements in grid scheduling. Simulation Modelling Practice and Theory 19(1), 213–226 (2011)
32. Kołodziej, J., Xhafa, F.: Modern approaches to modelling user requirements on resource and task allocation in hierarchical computational grids. Int. J. on Appled Mathematics and Computer Science 21(2), 243–257 (2011)
33. Kwok, Y.-K., Hwang, K., Song, S.: Selfish grids: Game-theoretic modeling and nas/psa benchmark evaluation. IEEE Tran. on Parallel and Distributing Systems 18(5), 1–16 (2007)
34. Lapin, L.: Probability and Statistics for Modern Engineering, 2nd edn. (1998)
35. Lee, Y.C., Zomaya, A.Y.: Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling. In: Proc. of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid CCGrid), Shanghai, China, pp. 92–99 (2009)
36. Mann, P.S.: Introductory Statistics, 7th edn. Wiley (2010)
37. Mejia-Alvarez, P., Levner, E., Mossé, D.: Adaptive scheduling server for power-aware real-time tasks. ACM Trans. Embed. Comput. Syst. 3(2), 284–306 (2004)
38. Mezmaz, M., Melab, N., Kessaci, Y., Lee, Y., Talbi, E.-G., Zomaya, A., Tuyttens, D.: A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. J. Parallel Distrib. Comput. (2011), doi:10.1016/j.jpdc.2011.04.007
39. Miao, L., Qi, Y., Hou, D., Dai, Y., Shi, Y.: A multi-objective hybrid genetic algorithm for energy saving task scheduling in cmp system. In: Proc. of IEEE Int. Conf. on Systems, Man and Cybernetics (ICSMC 2008), pp. 197–201 (2008)
40. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer (1992)
41. Muhammad, R., Hussain, F., Hussain, O.: Neural network-based approach for predicting trust values based on non-uniform input in mobile applications. The Computer Journal Online (2011)

42. Olivier, I., Smith, D., Holland, J.: A study of permutation crossover operators on the travelling salesman problem. In: Proc. of the ICGA 1987, Cambridge, MA, pp. 224–230 (1987)
43. Pinel, F., Pecero, J., Bouvry, P., Khan, S.U.: Memory-aware green scheduling on multicore processors. In: Proc. of the 39th IEEE International Conference on Parallel Processing (ICPP), pp. 485–488 (2010)
44. Shen, G., Zhang, Y.: A new evolutionary algorithm using shadow price guided operators. Applied Soft Computing 11(2), 1983–1992 (2011)
45. Shen, G., Zhang, Y.: A shadow price guided genetic algorithm for energy aware task scheduling on cloud computers. In: Proc. of the 3rd International Conference on Swarm Intelligence - (ICSI 2011), pp. 522–529 (2011)
46. Song, S., Hwang, K., Kwok, Y.: Trusted grid computing with security binding and trust integration. J. of Grid Computing 3(1-2), 53–73 (2005)
47. Song, S., Hwang, K., Kwok, Y.: Risk-resilient heuristics and genetic algorithms for security- assured grid job scheduling. IEEE Tran. on Computers 55(6), 703–719 (2006)
48. Whitley, D., Rana, S., Heckendorn, R.: The island model genetic algorithm: On separability, population size and convergence. Journal of Computing and Information Technology 7, 33–47 (1998)
49. Wu, C.-C., Sun, R.-Y.: An integrated security-aware job scheduling strategy for large-scale computational grids. Future Generation Computer Systems 26(2), 198–206 (2010)
50. Xhafa, F., Abraham, A.: Computational models and heuristic methods for grid scheduling problems. Future Generation Computer Systems 26, 608–621 (2010)
51. Xhafa, F., Carretero, J., Abraham, A.: Genetic algorithm based schedulers for grid computing systems. Int. J. of Innovative Computing, Information and Control 3(5), 1053–1071 (2007)
52. Xhafa, F., Carretero, J., Barolli, L., Durresi, A.: Requirements for an event-based simulation package for grid systems. Journal of Interconnection Networks 8(2), 163–178 (2007)
53. Zomaya, A.Y.: Energy-aware scheduling and resource allocation for large-scale distributed systems. In: 11th IEEE International Conference on High Performance Computing and Communications (HPCC), Seoul, Korea (2009)

Chapter 5
Power Consumption Constrained Task Scheduling Using Enhanced Genetic Algorithms

Gang Shen and Yanqing Zhang

Abstract. Two typical challenges in the energy aware tasking scheduling are (1) minimizing energy consumption with execution time constraint and (2) minimizing task execution time with energy consumption constraint. This chapter focuses on the later challenge. It is very important to efficiently schedule tasks in mobile devices and sensor networks that have limited power. The goal is to cooperatively complete a set of tasks among diverse computing resources with given energy. Traditional scheduling algorithms, such as list scheduling, are not very efficient for this scheduling problem. Linear programming optimization method does not fit well since it is a non-linear problem. An enhanced genetic algorithm can solve this problem effectively.

5.1 Introduction

Small, fast and portable computing devices, such as wireless laptop computers, cell phones, iPad, iCloud, are very popular. They all rely on limited on-device energy. It is important to make them more power efficient.

Network based cloud computing technology provides more sophisticated computing capability. They can be designed as a few computers loosely linked over the internet, or many computers hosted in a data center. A data center is more effective in providing complex computing services. It can efficiently provide various computing services requested by portable computing devices, complete complex business transactions, and conduct scientific calculations.

Gang Shen
Computer Science Department, Georgia State University, P.O. Box 3994, Atlanta, GA 30302-3994, USA
e-mail: gshen1@student.gsu.edu

Yanqing Zhang
Computer Science Department, Georgia State University, P.O. Box 3994, Atlanta, GA 30302-3994, USA
e-mail: yzhang@cs.gsu.edu

S.U. Khan et al. (Eds.): Evolutionary Based Solutions for Green Computing, SCI 432, pp. 139–159.
springerlink.com

Weather forecasting is a good example of how personal computers and a data center working together. Weather forecasting is a very complicated process. It needs lots of weather observations, satellite imaging processing, weather simulations, and meteorologists' explanations. It demands large amount of computing power to complete the work quickly. Large powerful computers in the data center are used for this kind of tasks. Results of the weather forecasting are broadcasted to personal computing devices that used by many consumers.

Computers consume energy in two common ways, direct and indirect computing related consumption. Energy consumed by supporting devices, such as air conditioning in the data center, is the indirect energy consumption. Energy used by computers is the direct energy consumption. Together, computing related energy consumption is roughly equivalent to the aviation industry's energy consumption. It accounts for 2% of anthropogenic CO2 from its share of energy consumption [4].

A computer center can host 10,000 or 150,000 servers [3]. These mega data centers can support many large companies' daily operations, conduct many e-commerce transactions, perform large scale scientific researches, and provide services to many other clients. These data centers use large amount of energy [14]; [29]. The energy used by the US servers and data centers is significant. It is estimated that they consumed about 61 billion kilowatt-hours (kWh) in 2006 (1.5 % of total U.S. electricity consumption) for a total electricity cost of about 4.5 billion USD. If the trend continues, this demand would rise to 12 gigaWatts (GW) by 2011. It would require an additional 10 power plants [28].

Green energy is the power generated from renewable sources, such as solar, wind, geothermal, biomass, and small hydro. They are renewable sources and more environmentally friendly than traditional electricity generation method. They emit little or no air pollution and leave behind no radioactive waste like nuclear power plant. Most importantly, they are naturally replenished by the earth and sun [33].

Brown energy is the power generated from environmental hostile technologies. The vast majority of electricity consumed in the United States comes from coal, nuclear, large hydro, and natural gas plants. This is the single greatest source of air pollution in the United States, contributing to both smog and acid rain. This is the greatest single contributor of global climate changing gases, including carbon dioxide and nitrogen oxide [33].

Majority of the power we consumed today is non-renewable environmental hostile energy. In 2006, green energy only accounts for 7% of total US energy supply. Petroleum, coal and natural gas burning generate 86% of the total energy supply [27]. Our appetite to more energy is growing more rapidly than ever [7].

Many research projects have conducted to improve data centers' energy efficiency, such as improving the design of the data center [6], improving equipment [1], and improving air conditioning [10]. They focused on reducing energy consumption and improving supporting devices' efficiency.

Improving energy usage efficiency and reducing energy usage are two major approaches of energy conservation and green computing. Efficient task scheduling in data center is one such research topic that can yield significant return. With optimized task scheduling, computers can complete tasks using less energy. It also

reduces energy consumptions from supporting devices. Combined energy savings from efficient task scheduling in a large data center can be significant.

From green computing perspective, efficient task scheduling can be defined as either minimizing energy consumption with schedule length constraint or minimizing schedule length with energy consumption constraint [15]. The objective of the first problem is to use the least amount of energy to complete all tasks within a given time frame. It is used mainly in real time processing environments. The second problem is to complete tasks as fast as possible under given energy limitation. Its objective is to use energy efficiently and has great usage in mobile computing, sensor network, etc.

There are many energy conserving task scheduling algorithms for multiple heterogeneous computers. Heuristic algorithms [16]; [18]; [32]; [29]; [30] can find good solutions among all possible ones. They tend to complete the searches very fast. They may not always locate the best solutions but guarantee local optimal solutions. These algorithms can usually find ones close to the optimal solutions.

Bio-inspired search algorithms [2]; [26], search the space by simulating nature process. The typical algorithms are Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), etc. They can find optimal or near optimal solutions. They are slower than heuristic algorithms on current computer architectures. Hybrid algorithms [19]; [11]; [17] try to combine better attributes from various algorithms.

We solved the first minimizing energy consumption problem using an enhanced GA [23]. We introduced Shadow Price concept into the *GA* and achieved much better result. In this chapter, we introduce another Shadow Price enhanced GA (*SPGA*) to solve the second problem - minimizing task execution time with energy constraint.

The rest of this chapter is organized as follows. Section 5.2 defines the energy constrained task scheduling problem. Section 5.3 introduces our new shadow price guided genetic task scheduling algorithm. Section 5.4 shows and analyzes experiment results. Section 5.5 concludes this chapter.

5.2 Power Consumption Constrained Task Scheduling Problem

In general, the amount of power an electrical device uses is the product of supplying voltage and the current it draws. The energy consumed is the product of power and time. In addition, computer processor's speed varies based on the voltage supplied. Within limits, a processor runs faster with higher voltage. Thus, the power consumption of a processor is directly linked to its running speed. Overclocking is one such technique that speeds up the processor by raising the voltage. The cost of this speed increase is more energy consumption. Tasks can be completed faster with higher speed. It's an optimization problem to achieve a balance between energy and time. Multiple heterogeneous computers and many tasks to complete further complicate the challenge.

If we quantify a task R as the number of instructions and execute it on a processor with speed S, the power consumption to complete this task is $E = C \cdot R \cdot S^{\alpha-1}$. C is a

constant that is greater than 1, and α is a constant greater than 3. Linear speed change results exponential change of energy consumption. Since speed is instruction count over time, energy consumption can be represented as $E = C \cdot R(R/T)^{\alpha-1}$ and T is the task execution time. The instruction count of a task has big impact on energy consumption as well.

The problem to be optimized can be defined as: using shortest time to finish m tasks on n heterogeneous computers and the total energy consumed is less than or equal to E.

$$MinimizeT = max_{i\in n}(T_i) \tag{5.1}$$

subject to

$$T_i = \sum_{j=1}^{k} \left(\frac{C_i \cdot \left(R_i^j\right)^{\alpha_i}}{E_i^j} \right)^{\frac{1}{\alpha_i - 1}} \tag{5.2}$$

$$E \geq \sum_{i=1}^{n} E_i \tag{5.3}$$

In the objective function formula (see Eq. (5.1)), T_i is the execution time of processor i. Eq. (5.2) is the execution time of k tasks assigned to processor i. R_i^j represents the instruction count of task j that assigned to processor i. E_i^j is the energy it consumes, C_i and α_i are constants for processor i, $\alpha_i = 1 + 2/\theta_i \geq 3$ and $0 < \theta_i \leq 1$. Since speed is commonly used in the specification of processor, Eq. (5.2) can be simplified into the following expressions:

$$T_i = \sum_{j=1}^{k} \frac{R_i^j}{S_i^j} \tag{5.4}$$

$$E_i = C_i \sum_{j=1}^{k} \left(R_i^j \left(S_i^j \right)^{\alpha_i - 1} \right) \tag{5.5}$$

$$X_i \leq S_i^j \leq Y_j \tag{5.6}$$

where S_i^j is the executing speed of task j on processor i, X_i and Y_i are the minimum and maximum running speed of processor i, R_i^j represents the instruction count of task j that assigned to processor i.

The objective is to find a schedule such that m tasks are completed in the shortest time and energy consumed is within the constraint E. There are multiple sub-optimization problems in the definition, energy, task, and speed. The first is to optimal distributing energy limitation E to each processor E_i. The second is to optimal assigning tasks $\{R\}$ to each processor. And the last one is to determine optimal running speed for each task assigned to a processor. All three sub problems are connected. Assigning higher energy to a processor enables it to process more tasks in

short period of time. Higher running speed demands more energy. Since the objective is to minimize the longest running time of a processor, all processors have to cooperate in energy and task assignment. It's a very complex combinational optimization problem.

It is proven that the schedule length is minimized when all tasks assigned to a processor execute with the same power [15]. To achieve the best result, tasks assigned to the same processor shall be executed at the same speed since power determines speed. Therefore, Eqs. (5.2), (5.4), and (5.5) can be simplified to the following formulas:

$$T_i = \left(\frac{C_i \cdot \left(\sum_{j=1}^{k} R_i^j \right)^{\alpha_i}}{E_i^j} \right)^{\frac{1}{\alpha_i - 1}} \tag{5.7}$$

$$T_i = \frac{\sum_{j=1}^{k} R_i^j}{S_i} \tag{5.8}$$

$$E_i = C_i \cdot \left(\sum_{j=1}^{k} R_i^j \right) \cdot (S_i)^{\alpha_i - 1} \tag{5.9}$$

Since all tasks running with the same speed on the same processor and the objective is to complete the tasks as fast as possible with assigned energy for the processor, we can use Eq. (5.7) to calculate the executing time, or Eq. (5.10) to calculate optimal speed. This solves the third sub optimization problem. What we need to solve now are the sub problems of distributing total allowed energy consumption to each processors and assigning tasks to them such that the executing time is minimal.

$$S_i = \left(\frac{E_i}{\left(C_i \cdot \left(\sum_{j=1}^{k} R_i^j \right) \right)} \right)^{\frac{1}{\alpha_i - 1}} \tag{5.10}$$

There can be two objective functions when optimizing execution time, minimizing either the concurrent running time on all available processors (the max of all processors' running time) or the accumulated execution time from all processors (summation of all processors' running time). Since tasks assigned to a processor shall be executing in the same speed, the later optimal problem becomes quite simple. Optimal can be achieved by selecting the most efficient processor and assign all tasks to it. We choose to optimize the difficult problem of optimizing concurrent running time (see Eq. (5.1)).

Instead of a minimal function, standard deviation on execution time can also be used as the objective function to optimize. It measures the distances from each processors' execution time to the average. The idea is to make all processors sharing the work load and enforce their execution time closing to the medium. This is a good objective function in general but may not work in a heterogeneous processors environment. In a very diverse setup, processors' energy and execution efficiency

can different significantly from one to the other. There can be optimal solutions that no work is assigned to less efficient processors. A simple minimal function is both efficient in calculation and flexible to cover most scenarios.

5.3 Enhanced Genetic Algorithm for the Green Task Scheduling Problem

5.3.1 Genetic Algorithm

Genetic Algorithm (*GA*) [8]; [9] is a reward based multi solution search algorithm. It is a branch of bio inspired evolution algorithm (*EA*). It has been applied to many applications successfully [12]. Comparing to other single solution search algorithms such as Linear Programming *LP*, *k*-opt, there are multiple feasible solutions concurrently evolving toward the best solution in the *GA* search process.

Instead from a single search point, *GA* randomly selects multiple starting points in the search space. These initial solutions are organized into a population. Then, the *GA* search process evolves all solutions in the population gradually towards the final optimal solutions. The evolutions are organized into generations. To evolve the generation, *GA* (see Fig. 5.1) search process applies crossover, mutation, random reseeding, and elite filtering operations to the solutions in the population. The crossover operator mimics parents producing child in nature. Based on problem domain, a breeding method is used to create the "child" solution from two parent solutions. The child solution inherits certain attributes from both parents. The general goal is to create a child solution that poses good characteristics from both parents and better than both parents.

To generate a new solution, the unary mutation operator modifies the state(-s) of one or a small number of components of an existing solution. Often times, the newly generated solution is much different than the original solution and may not even be a valid solution. It can bring search to an area that has not been visited before. It can be used to break the local optimal trap and quickly move the solution. Aside from mutation and crossover operators, new solutions are randomly generated in the evolution process in general as well. This is to further broad the search space and serves as an extra insurance of a global search.

Elite filtering is the reward mechanism in the *GA* search. It ensures better solutions surviving the evolution process and participating next generation's evolution. It guides search towards optimal or near optimal solutions.

Since *GA* imposes very few requirements for the optimization problem, it has been used widely in many fields and achieved very good results. But the solution quality and search speed are not always satisfactory. Many improvements have been made to enhance *GA*'s performance, such as adding local search [20]; [34], adding adaptive capability [5]; [35], and hybridizing with other algorithms [25]; [13]; [21]. We introduce another enhancement to the *GA* search that can achieve much better results.

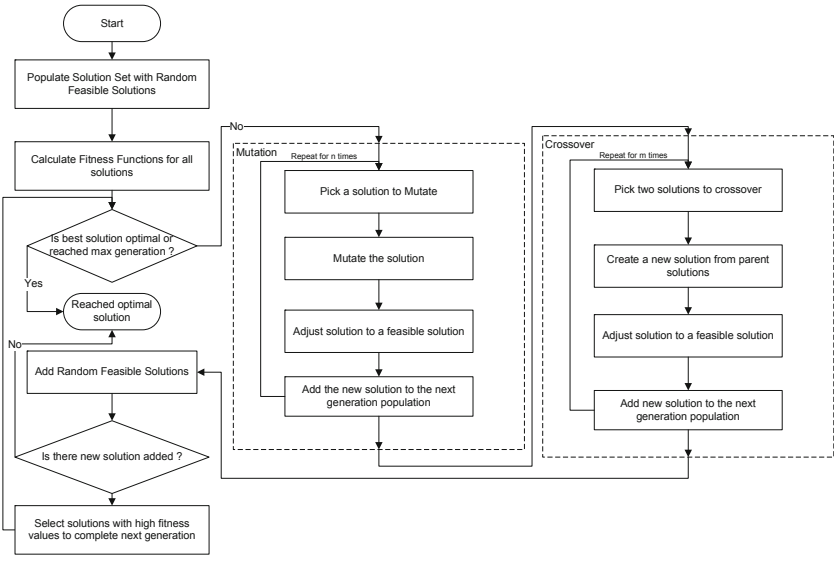


Fig. 5.1. Genetic Algorithm

5.3.2 Shadow Price Enhanced Genetic Algorithm

Randomness is the key to make *GA* a successful global search algorithm. But the randomness also introduces many unnecessary calculations in the search process. *GA* operations introduce many random, worse solutions. This waste of computing power leads to slow search speed and low quality results. We use shadow price to improve efficiency and still maintains the necessary randomness for the *GA* search.

Shadow price was introduced in the Linear Programming(*LP*) search algorithm initially. In a solution, each component that makes up the solution has a price associated with it. The sum of all components' prices is the cost of the solution. It's a concrete true value. In *LP*'s search process, shadow price is used as a measure of the component's potential contribution to the solution. It's a made up value that only useful in the search process, thus named shadow price. Shadow price is the contribution to the objective function that can be made by relaxing a constraint by one unit. Different constraints have different shadow prices, and every constraint has a shadow price. Each constraint's shadow price changes along with searching for an optimal solution.

In *GA*'s search process, fitness value is used to measure the quality of a solution. It is used to compare solutions and decides which solutions are the fit ones to survive the evolution process (the reward). In fact, it is the only measurement that used in the search process. Evolution operations are entirely based on randomness. Participating solutions are randomly selected, components are randomly selected, and operation parameters are randomly selected. There are no general directions to evolve the solutions and newly generated solutions are very diversified. It's not very efficient.

To cope with these shortcomings, we introduced shadow price concept into the *GA* search process [24]. We define the shadow price as the relative potential improvement to the solution fitness value with a change of a component. The shadow price is defined for a component. A solution's shadow price is the sum of all components' shadow prices. The shadow price we defined here is a pseudo value extracted from the component's current state. It is not directly linked to its true cost but a potential improvement from a change. The contribution may or may not be realizable since the solution has to be validated after a component change. The change can be a value change, a position change, or other applicable changes. High shadow priced component has the potential of making large contribution to the fitness value. We use shadow price to directly compare components, their relationships, and contributions toward a better solution.

Based on different problems, shadow prices can take on different meanings or values. In the traveling salesman problem, it can simply be the possible distance reduction from changing the next visiting city from the current one [24]. In manufacture, shadow price can be the cost of material, time, etc. [25].

With the introduction of shadow price, we can improve *GA* operator's performance significantly. Crossover operation intends to pass good components from parents to the children so that the child solutions are better than parents. Passing random selected components to children can produce many worse child solutions and waste many computation resources. To improve it, we can use shadow price to evaluate and compare components. Since shadow price quantifies contribution of components, we can selectively passing good components that make bigger contributions to the child. This can improve the probability of generating better children significantly. To avoid local optimal trap, it is necessary to build some randomness into the high shadow priced components' selecting process.

Mutation evolves the solution by change one or more components' states. It is a very effective operation that can dramatically change the solution. Often times, only mutation is used in *GA*. It is counterproductive and waste of computing resource to changing a component from a good state to a worse one. Selecting component and direction to evolve are the keys for efficiency. Since shadow price defines the potential contribution of component and enables components comparison, we can select high potential components to mutate and mutate them to low potential states. Instead of random evolution on random components, shadow price enabled mutation can produce many better solutions and improve computing resources efficiency. In order to avoid local optimal trap, randomness need to be built into components selection and direction setting.

In the new shadow price enhanced genetic algorithm (*SPGA*), we establish a two-measurement system: fitness value is used to evaluate overall solution goodness, and shadow price is used to evaluate components' goodness. We enhance the *GA* operations with shadow price. We use the components' shadow price to select components to operate and to set directions to evolve. We steer the search towards optimal solution and still keep the necessary randomness in the search process. The *SPGA* (see Fig. 5.2) is much more efficient in converging to the optimal solution and can provide better results.

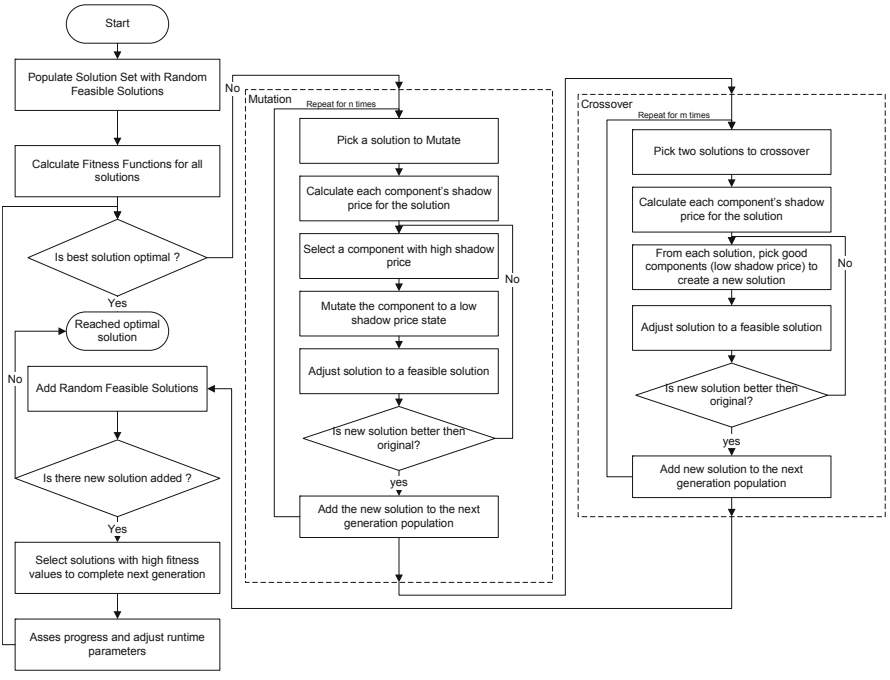


Fig. 5.2. Shadow Price Enhanced Genetic Algorithm

5.3.3 Green Task Scheduling Using S PGA

The main aim of our green task scheduling (see Eq. (5.1)) is to schedule m tasks on n computers such that the concurrent execution time is minimal and the total energy consumption is less than or equals to E.

It is proven that it is most efficient, for both energy minimization and execution time minimization, to schedule tasks on the same processor at the same speed [15]. The green task scheduling optimization problem breaks down to two sub problems, optimal distribute energy constraint E to n computers and optimal assign m tasks to n computers. The original third sub optimization problem, minimizing execution time for a processor i with m_i tasks and energy cap of E_i, can be solved directly using Eq. (5.7) and speed can be calculated using Eq. (5.10).

Encoding is very important in the *GA* search process since it directly impacts the performance. The schema in this problem is straightforward. Since the order of processors in the solution has no impact to the solution, a list of processors and tasks assigned to them can be used to represent a solution. The processor definition includes its attributes, such as is C_i , α_i, minimal speed, maximal speed, etc. Tasks associated with a processor can be stored in a list as well since the order is of no significance either.

There are two steps to construct a solution, distribute energy constraint and assign tasks. It does not impact the solution which task completes first. But both tasks have to be completed before the fitness value can be calculated for a solution.

To solve the scheduling problem, we can either treat it as a nested two optimization problems or an optimization problem with two sub tasks. In the nested optimization problem scenario, one sub problem will be selected as the parent problem and used to drive the other child problem. For example, if we select energy constraint distribution as the parent problem, the search process starts with creating various combinations of energy assignments to processors. Each energy constraint assignment will be treated as a separate optimization problem and solved individually. Various task assignments are evaluated and the assignment that with the least concurrent execution time is the fitness value for the energy assignment. The search process evolves the parent energy assignments and searches for best task combinations for each new assignment. The process repeats until the optimal solution is found.

We can also treat the two optimization tasks as two separate parameters in the same optimization problem and create a flat model. In the nested model, parent searches for the optimal energy assignment and the child searches for the best task assignments within the parent energy assignment. In the flat model, both parameters work together to optimize the same objective of minimizing solution execution time for all processors. Thus, we can ignore the relationship between these two parameters and only focus on the relationships from the two parameters to the solution. We can tune one parameter at a time and rotate. The process can be repeated until the optimal solution is found.

Nested optimization problems are difficult to solve and takes more time to converge [25]. In comparison, flat models are easier to solve since there is only one objective function. The complexity is that there are more parameters in the *GA* operations. Optimizing nested models use tree search and optimizing flat models use linear search with rotating parameters.

To work with flat model, we define two mutation operations, energy mutation and task mutation. There are two sub energy mutations, exchange energy between two processors and move some energy from one processor to the other. There are also two sub task mutations, exchange a pair of tasks between two processors and move one task from one processor to the other. The processors and tasks are randomly selected in the operations. There is no preference or direction to move the search process.

Our enhanced mutation operation only moves some energy from one processor to the other. Since the objective is to minimize concurrent execution time and more energy can improve speed, we want to move some energy from a short run time processor to a long run time processor. The long run time processor will benefit from added energy and shorten the run time. But the short run time processor may not have extra energy to give. There may be multiple reasons that cause processor use less time, such as the processor is very efficient and can run very fast with little energy, the processor is assigned with large amount of energy, or the processor is assigned with small tasks. So short run time cannot be used to select energy donor

processor. A combination of higher energy and less run time makes a good selection criterion.

In our definition, shadow price represents a component's potential. Here, shadow price is the combination of a processor's run time and energy assigned. Run time takes precedence over energy since we are selecting the energy donor processor. A processor's shadow price is high when it has a short run time and large energy. A processor's shadow price is low when it has a long run time or a short run time and smaller energy. We want to mutate a processor from high shadow priced state to a low shadow priced state. The mutation direction set by the shadow prices is to mutate a processor with below average run time to a longer run time or less energy state.

We previously used average energy consumption per instruction as the shadow price for each processor and successfully solved the minimizing energy consumption with time constraint green scheduling problem [23]. This definition, average energy consumption per instruction or average time spent per instruction, does not work for the task mutation here since each processor can be assigned with different energy and tasks. The average energy or time per instruction cannot be used to compare among processors. High average energy consumption per instruction can exist for processors with various energy or task assignments. Same fact holds true for time spent per instruction.

The goal of task mutation is to move task from a long running processor to a short time running processor. The task donor processor is easy to pick. It can simply be one of the long run time processors. The receiving processor shall be one of the short run time processors. We need to be very careful about selecting receiving processor since it can dramatically increase its run time. Since we are not rearranging energy in this task mutation, the ideal receiving processor is the one that its energy or execution time is not very sensitive to task increase. That is, task increase is not the most influential factor in a processor's executing time or energy calculation. Formula (5.7) shows execution time calculation with fixed energy and Eq. (5.9) shows energy calculation with known speed. Both are exponential functions. In an exponential function, exponent has far bigger impact to the result than the base. In both Eq. (5.7) and Eq. (5.9), task instruction count is in the base and α is in the exponent. Since α is a positive number and greater or equal to 3, it can generate bigger impact to the execute time and energy consumption. While comparing 2 processors with same tasks, the one with bigger α consumes more energy if speeds are the same or takes more execution time if energies are the same. So, it is preferred to add a task to a processor with smaller α since it may cause much small increase to the execution time. We define the shadow prices as the combination of execution time and α . We want to mutate the task from a long execution time processor to a processor with short execution time and a smaller α .

Our shadow price enhanced algorithm (see Alg. 2) follows standard *GA* algorithm framework. To avoid local optimal traps, we combine enhanced mutation with standard mutation operations.

Algorithm 2. Shadow price GA

Validate there is at least one feasible solution;
Build initial population;
while stop criteria has not met **do**
  Select a sub population to randomly apply one of the following operations;
    Energy move mutation operation;
      Randomly select two processors;
      Move some energy from one processor to the other processor;
      Validate the new solution;
    Energy exchange mutation operation;
      Randomly select two processors;
      Exchange energy assignments between them;
      Validate the new solution;
    Task move mutation operation;
      Randomly select two processors;
      Randomly select a task from one processor;
      Move the randomly selected task from one processor to the other processor;
      Validate the new solution;
    Task exchange mutation operation;
      Randomly select two processors;
      Randomly select a task from each processor;
      Exchange the selected tasks between the two processors;
      Validate the new solution;
    Shadow price enhanced energy mutation operation:
      Sort all processors based on execution time;
      Split processors into 2 sets, long run time processors and short run time processors;
      Create a subset from long run time processors to establish an energy receiving processor pool S_r;
      Random select one processor from S_r as the receiving processor P_r;
      Re-short the short run time processor set based on energy assignment;
      Create a subset from short run time processors to establish an high energy donating processor pool S_d;
      Random select one processor from S_d as the energy donating processor P_d;
      Move some energy from P_d to P_r;
      Validate the new solution;
    Shadow price enhanced task mutation operation:
      Sort all processors based on execution time;
      Split processors into 2 sets, long run time processors and short run time processors;
      Create a subset from long run time processors to establish an task donating processor pool S_d;
      Random select one processor from S_d as the donating processor P_d;
      Re-short the short run time processor set based on processor's α value;
      Create a subset from short run time processors to establish a small α value task receiving processor pool S_r;
      Random select one processor from S_r as the task receiving processor P_d;
      Randomly select one task from P_d and move to P_r;
      Validate the new solution;
    Add random solutions;
    Filter and build next generation;
end while
return Solution(-s)

Shadow price represents a state of a component relative to the current solution. It measures the potential contribution of a component. It is used to compare components within a solution in the search process. It can take on many different forms. It can be the true material waste in the stock cutting problem, a number relative to solution in the traveling salesman problem, or a function in the stock reduction problem. In this green scheduling problem, the shadow price is embedded in the mutation operations due to its complexity. It's a procedure. It measures the processor execution time, energy consumption, and processor's attribute α. It enables selecting of large potential component to mutate to a low potential state. The shadow price can eliminate many unnecessary calculations. It can greatly improve the search speed and solution quality. In our shadow price enhanced *GA* (see Alg. 2), we still use the fitness to filter the solution by using the command specified in Line 44. We implemented a two-measurement system, fitness to measure overall solution and shadow price to measure components, to improve *GA*'s speed and result.

5.4 Performance Analysis

We conducted a comprehensive comparative study between *GA* and our new shadow price enhanced *GA*. Both algorithms followed the standard *GA* framework and were identical except the mutation operations. *SPGA* was coded based on Alg. 2 and implemented all six mutation operations. Only the first four mutation operations were used in the standard *GA*.

By using command in Line 4 of Alg. 2 we check the existence of a valid solution. Formulas (5.5) and (5.9) show that energy consumption is at the lowest level when the speed is minimized for a given processor. To check if a processor can complete the tasks with limited energy, we only need to test it at its lowest speed. To check existence of at least one valid solution, we test all tasks for each processor at its lowest speed and compare energy consumptions. If there is one processor consumes less than or equal to energy constraint, there is at least one valid solution exist for the problem. Otherwise, there is no feasible solution for the problem and algorithm aborts.

We coded and tested both algorithms in Microsoft *C#*. All experiments were run on a Lenovo Thinkpad laptop T410 that equipped with Intel Core i5-M520 2.4 GHz CPU and 4 GB of memory running Windows 7.

To create test cases, we need processor definitions and tasks. We selected 20 latest commercial released CPUs [31] for our experiments (see Table 5.1). We used published speed as the processor's minimum speed. A random number between 5% and 25% was used as the speed improvement to define the processors' maximum speeds. Constant C was randomly generated between 1 and 100. Since α has big impact on the CPU energy consumption, it was randomly chosen within a tight range between 3 and 5. This was to avoid big disparity among processors. Instruction counts for tasks were also randomly generated between 500 and 100,000

instructions. To improve quality, we used a public available true random number generating services [22] instead of *C#* library.

Table 5.1. Processors

ID	Processor	Year	Min Speed (MIPS)	Speed up (%)	Max Speed (MIPS)	C	α
1	DEC Alpha 21064 EV4	1992	300	9	327	84	4.08
2	Intel Pentium III	1999	1354	15	1557	7	3.67
3	AMD Athlon	2000	3561	23	4380	8	4.51
4	AMD Athlon XP 2400+	2002	5935	14	6766	86	3.94
5	Pentium 4 Ex.Ed.	2003	9726	7	10407	74	3.50
6	AMD Athlon FX-57	2005	12000	9	13080	50	3.41
7	AMD Athlon 64 3800+ X2 (Dual Core)	2005	14564	13	16457	60	3.74
8	ARM Cortex A8	2005	2000	18	2360	52	4.57
9	Xbox360 IBM "Xenon" (Triple Core)	2005	19200	20	23040	55	4.92
10	AMD Athlon FX-60 (Dual Core)	2006	18938	17	22157	62	4.17
11	Intel Core 2 (Extreme X6800)	2006	27079	21	32766	19	4.03
12	Intel Core 2 (Extreme QX6700)	2006	49161	16	57027	22	4.33
13	PS3 Cell BE ((PPE only))	2006	10240	23	12595	45	3.13
14	P.A. Semi (PA6T-1682M)	2007	8800	25	11000	11	3.67
15	Intel Core 2 (Extreme QX9770)	2008	59455	17	69562	92	4.64
16	Intel Core i7 (Extreme 965EE)	2008	76383	18	90132	11	4.08
17	AMD Phenom II (X4 940 Black Ed.)	2009	42820	15	49243	42	4.57
18	AMD Phenom II (X6 1090T)	2010	68200	12	76384	77	3.20
19	Intel Core i7 (Extreme Ed. i980EE)	2010	147600	14	168264	29	3.82
20	IBM 5.2-GHz z196	2010	52286	15	60129	66	3.90

Experiment cases were created using various combinations of processors and tasks. Energy constraint for each experiment case was randomly created, validated and shortened to ensure that few processors idle in the optimal solutions.

We studied algorithms' performance in two aspects, result quality and convergence speed. For result quality, we test algorithm with various test cases under fixed energy constraint and fixed generation of evolutions.

Tables 5.2–5.6 show comparison test results between *GA* and *S PGA*. To make it easy to read, only the integer portion of data is displayed. The processor count ranges from 10 to 50. The task count R ranges from 500 to 5000. The *max* generation limits (G_{max}) are 500, 1000, 1500, 2000, 3000, and 5000. All combinations of task count R and generation limit G_{max} are tested for each processor count setup.

Each test case was run at least 10 times. Results were averaged and reported. The improvement percentages from *S PGA* optimal solution (T_{spga}) over *GA* optimal solutions (T_{ga}) are reported in the tables. Since the objective is to minimize concurrent execution time, a positive number shows *GA*'s best solution takes long time than *S PGA* and *S PGA*'s result is better than *GA*.

Tables 5.2–5.6 show all positive results. *S PGA* best solutions used less execution time than *GA* best solutions in all test cases. *S PGA* reached better solutions than *GA*.

Table 5.2. *S PGA* Time Improvement over *GA* $(T_{ga} - T_{spga}) \times \frac{100}{T_{ga}}$ for 10 Processors

G_{max}	$R = 500$	$R = 1000$	$R = 1500$	$R = 2000$	$R = 3000$	$R = 5000$
500	44	25	15	8	5	1
1000	85	56	30	22	12	2
1500	84	76	51	37	19	8
2000	74	78	71	50	27	13
3000	57	70	77	75	45	24
5000	23	58	68	72	76	50

Table 5.3. *S PGA* Time Improvement over *GA* $(T_{ga} - T_{spga}) \times \frac{100}{T_{ga}}$ for 20 Processors

G_{max}	$R = 500$	$R = 1000$	$R = 1500$	$R = 2000$	$R = 3000$	$R = 5000$
500	76	58	49	39	28	26
1000	69	79	70	60	55	41
1500	57	79	79	74	62	49
2000	49	78	81	80	73	56
3000	42	70	76	80	79	67
5000	31	56	69	75	78	77

To study algorithms' convergence speed, we reran all test cases with same energy constraints. Instead of limiting max generations, we set a target fitness value for the algorithms. The search only stops when the best solution's execution time meets the target value. The algorithm can take as much time or generations as needed to reach the target. Average execution times from above test cases were used as the target value.

Tests were run for each combination of processor count (P_c) and task count R. Each test was run at least ten times. Results were averaged and reported. Table 5.7 shows

Table 5.4. *S PGA* Time Improvement over *GA* $(T_{ga} - T_{spga}) \times \frac{100}{T_{ga}}$ for 30 Processors

G_{max}	$R = 500$	$R = 1000$	$R = 1500$	$R = 2000$	$R = 3000$	$R = 5000$
500	88	75	61	49	25	18
1000	90	86	79	72	54	36
1500	85	90	86	80	68	53
2000	81	87	90	86	82	68
3000	69	82	88	89	88	79
5000	46	76	82	86	90	86

Table 5.5. *S PGA* Time Improvement over *GA* $(T_{ga} - T_{spga}) \times \frac{100}{T_{ga}}$ for 40 Processors

G_{max}	$R = 500$	$R = 1000$	$R = 1500$	$R = 2000$	$R = 3000$	$R = 5000$
500	81	67	56	45	31	39
1000	81	83	74	68	60	53
1500	76	85	84	79	73	61
2000	70	83	87	85	80	66
3000	58	79	84	86	86	74
5000	43	71	79	82	85	85

Table 5.6. *S PGA* Time Improvement over *GA* $(T_{ga} - T_{spga}) \times \frac{100}{T_{ga}}$ for 50 Processors

G_{max}	$R = 500$	$R = 1000$	$R = 1500$	$R = 2000$	$R = 3000$	$R = 5000$
500	88	74	69	63	52	37
1000	90	85	81	76	72	61
1500	88	91	89	84	80	72
2000	85	89	91	88	84	76
3000	80	85	89	91	90	83
5000	61	80	85	88	91	89

S PGA's search time ($S\,T_{spga}$) savings over GA's search time ($S\,T_{ga}$), ($S\,T_{ga} - S\,T_{spga}$). A positive value shows *GA* takes longer time than *S PGA* to reach equivalent results. *S PGA* is faster with a positive value. Table 5.8 compares evolution generations used from *GA* (G_{ga}) over *S PGA* (G_{spga}), ($G_{ga} - G_{spga}$). A positive value states that *GA* took more generations of evolution than the *S PGA* to reach targeted solutions. Both tables show *S PGA* is faster than *GA*. Table 5.7 measures the speed in search time and Table 5.8 measures speed in evolution generations.

Table 5.7. *S PGA* Search Speed Improvement in Time(s), $ST_{ga} - ST_{spga}$

P_c	$R = 500$	$R = 1000$	$R = 1500$	$R = 2000$	$R = 3000$	$R = 5000$
10	3	3	10	18	33	60
20	3	8	12	15	36	52
30	5	7	10	13	21	42
40	6	8	12	17	31	43
50	8	10	12	16	20	38

Table 5.8. *S PGA* Search Speed Improvement in Generations, $G_{ga} - G_{spga}$

P_c	$R = 500$	$R = 1000$	$R = 1500$	$R = 2000$	$R = 3000$	$R = 5000$
10	791	538	783	845	782	407
20	945	1471	1424	1157	1456	1012
30	1154	1172	1273	1145	1115	1132
40	1327	1256	1454	1489	1714	1331
50	1479	1435	1365	1387	1193	1239

To put testing result into a global view, we charted *S PGA* and *GA*'s solution quality in Fig. 5.3. For each combination of CPU count and task count, we averaged final solutions' task execution times for *S PGA* and *GA*. The results are plotted in Fig. 5.3. The *X* axis in the chart represents the combination of CPU count P_c and task count R. It's in the format of (P_c, R). The *Y* axis represents the task execution time in seconds. Fig. 5.3 shows T_{ga} is greater than T_{spga} in all test cases. *S PGA* solutions use less execution time than *GA* solutions.

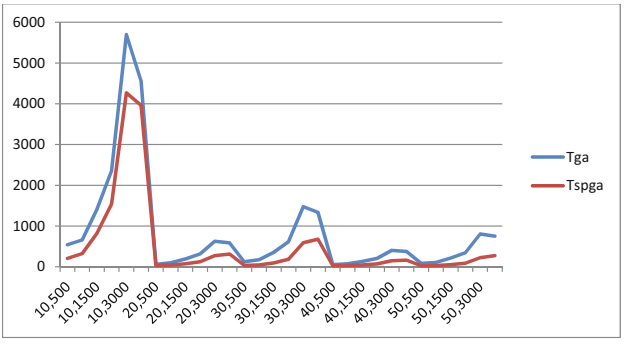


Fig. 5.3. Comparison of *S PGA*, *GA*'s Solution Quality

Figs. 5.4 and 5.5 compare *S PGA* and *GA*'s convergence speed. *X* axis is the same as Fig. 5.3 that represents the combinations of CPU count P_c and task count R in the form of (P_c, R). In Fig. 5.4, *Y* axis represents evolution generation counts used by algorithms. It shows *GA* used more generations than *S PGA* to reach equivalent solutions. *Y* axis shows algorithms' search time (seconds) in Fig. 5.5. In all cases, *GA* used more time than *S PGA* to reach equivalent solutions.

Fig. 5.4 shows *S PGA* used fewer generations of evolution than *GA*. Since *S PGA* used additional computing for shadow prices, each operation may take a little bit more time. Thus, fewer generations may not directly correlate to faster search speed. Fig. 5.5 show that *S PGA* used less search time than *GA*. It proves that spending a little extra time calculating shadow price can yield significant speed improvement. Figs. 5.4 and 5.5 together show that *S PGA* used less time and less evolution generations than *GA* to reach optimal solutions.

All test data and studies showed that final schedules from *S PGA* used less time to complete all tasks than final schedules from *GA*. *S PGA* achieved better solutions than *GA*. *S PGA* also used less time and fewer generations of evolution than *GA* to reach optimal solutions. Overall, *S PGA* find better results than *GA* and faster than *GA*.

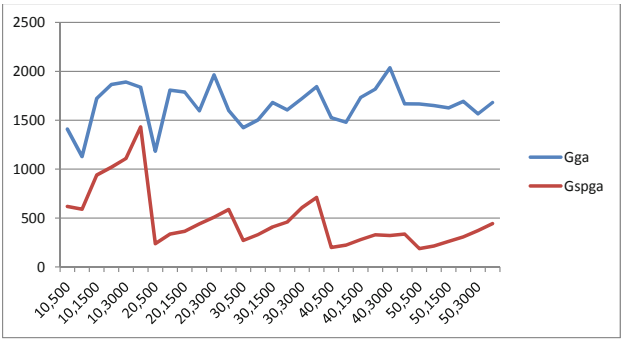


Fig. 5.4. Comparison of *S PGA*, *GA* Search Speed using Generation Count

5.5 Conclusions

GA is an effective population based global search algorithm that has solved many optimization problems. Many search algorithms have strict requirements. Linear programming can only optimize problems that can be represented in linear format. *GA* poses very little restrictions to the problem. It can optimize many problems that have clearly defined objective functions and operations.

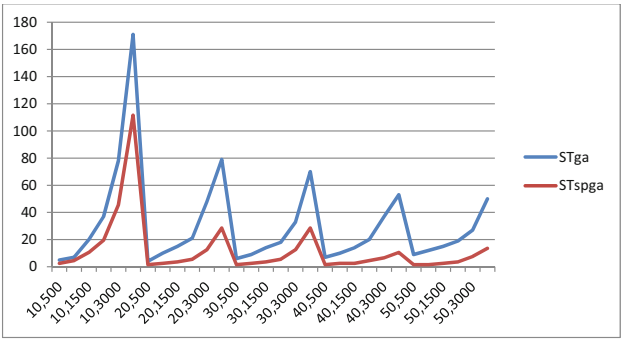


Fig. 5.5. Comparison of *S PGA*, *GA* Search Speed using Time(s)

The challenge of the versatile *GA* is its performance. For big complex problems, swam based *GA* needs large amount of computation, takes long time to converge, and may provide sub optimal solutions.

Evolution control is the key to *GA*'s performance. In the search process, *GA* operations generate many solutions randomly and elite filtering moves the search towards optimal solutions. Fitness value is used to filter candidate solutions. Generating and filtering many random solutions consume large amount of computing resource and search time. Thus, we introduce shadow price to generate better solutions.

We use shadow price to represent component's potential contribution to the solution. It is used to compare components and direct *GA* operations. The shadow price can be represented in many different forms, a true material price or a relative state. In this article, it is a procedure embedded in the operations.

The new *S PGA* implements a two-measurement system to enhance the algorithm. Fitness value evaluates overall solutions and evolves search towards optimal solutions. Shadow price evaluates components and helps generating better solutions. The *S PGA* for green scheduling delivered very good results. In all test cases and studies, the *S PGA* produced better solutions and converged faster.

References

1. Cabusao, G., Mochizuki, M., Mashiko, K., Kobayashi, T., Singh, R., Nguyen, T., Wu, P.: Data center energy conservation utilizing a heat pipe based ice storage system. In: 2010 IEEE CPMT Symposium Japan, pp. 1–4 (2010), doi:10.1109/CPMTSYMPJ.2010.5680287
2. Chang, P.C., Wu, I.W., Shann, J.J., Chung, C.P.: ETAHM: An energy-aware task allocation algorithm for heterogeneous multiprocessor. In: 45th ACM/IEEE Design Automation Conference, pp. 776–779 (2008)
3. Church, K., Greenberg, A., Hamilton, J.: On delivering embarrassingly distributed cloud services. In: ACM HotNets VII (2008), http://www.techrepublic.com/whitepapers/on-delivering-embarrassingly-distributed-cloud-services/2388125 (accessed May 2011)

4. Consortium for School Networking Initiative: Some Facts About Computer Energy Use (2010), http://www.cosn.org/Initiatives/GreenComputing/InterestingFacts/tabid/4639/Default.aspx (accessed July 2011)
5. Eriksson, R., Olsson, B.: On the performance of evolutionary algorithms with life-time adaptation in dynamic fitness landscapes, Congress. Evolutionary Computation 2, 1293–1300 (2004)
6. Hamann, H.F., López, V., Stepanchuk, A.: Thermal zones for more efficient data center energy management. In: 12th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm), pp. 1–6 (2010), doi:10.1109/ITHERM.2010.5501332
7. Hang, Y., Kuo, J., Ahmad, I.: Energy efficiency in data centers and cloud-based multimedia services: An overview and future directions. In: 2010 International Green Computing Conference, pp. 375–382 (2010), doi:10.1109/GREENCOMP.2010.5598292
8. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
9. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. The MIT Press (1992)
10. Iyengar, M., Schmidt, R., Caricari, J.: Reducing energy usage in data centers through control of Room Air Conditioning units. In: 12th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm), pp. 1–11 (2010), doi:10.1109/ITHERM.2010.5501418
11. Kliazovich, D., Bouvry, P., Khan, S.U.: DENS: Data Center Energy-Efficient Network-Aware Scheduling. In: 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom) Green Computing and Communications (GreenCom), pp. 69–75 (2010), doi:10.1109/GreenCom-CPSCom.2010.31
12. Koza, J., Keane, M., Streeter, M., Mydlowec, W., Yu, J., Lanza, G.: Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Springer (2005)
13. Koduru, P., Dong, Z., Das, S., Welch, S., Roe, J., Charbit, E.: A Multi objective Evolutionary-Simplex Hybrid Approach for the Optimization of Differential Equation Models of Gene Networks. IEEE Trans. Evolutionary Computation 12(5), 572–590 (2008)
14. Laszewski, G., Von Wang, L., Younge, A.J., He, X.: Power-aware scheduling of virtual machines in DVFS-enabled clusters. In: IEEE Intl. Conf. on Cluster Computing and Workshops, pp. 1–10 (2009), doi:10.1109/CLUSTR.2009.5289182
15. Li, K.: Performance Analysis of Power-Aware Task Scheduling Algorithms on Multiprocessor Computers with Dynamic Voltage and Speed. IEEE Trans. on Parallel and Distributed Systems 19(11), 1484–1497 (2008), doi:10.1109/TPDS.2008.122
16. Li, Y., Liu, Y., Qian, D.: A Heuristic Energy-aware Scheduling Algorithm for Heterogeneous Clusters. In: 2009 15th Intl. Conf. on Parallel and Distributed Systems (ICPADS), pp. 407–413 (2009), doi:10.1109/ICPADS.2009.33
17. Liu, Y., Yang, H., Luo, R., Wang, H.: Combining Genetic Algorithms Based Task Mapping and Optimal Voltage Selection for Energy-Efficient Distributed System Synthesis. In: 2006 Intl. Conf. on Communications, Circuits and System, vol. 3, pp. 2074–2078 (2006), doi:10.1109/ICCCAS.2006.285087
18. Miao, L., Qi, Y., Hou, D., Dai, Y.: Energy-Aware Scheduling Tasks on Chip Multiprocessor. In: Third International Conference on Natural Computation, vol. 4, pp. 319–323 (2007), doi:10.1109/ICNC.2007.356
19. Miao, L., Qi, Y., Hou, D., Dai, Y.H., Shi, Y.: "A multi-objective hybrid genetic algorithm for energy saving task scheduling in CMP system. In: IEEE Intl. Conf. on Systems, Man and Cybernetics, pp. 197–201 (2008), doi:10.1109/ICSMC.2008.4811274
20. Noman, N., Iba, H.: Accelerating Differential Evolution Using an Adaptive Local Search. IEEE Trans. Evolutionary Computation 12(1), 107–125 (2008)
21. Page, A., Naughton, J.: Dynamic Task Scheduling using Genetic Algorithms for Heterogeneous Distributed Computing. In: 19th IEEE Intl. Conf. on Parallel and Distributed Processing (2005), doi:10.1109/IPDPS.2005.184

22. Random.org (2011), http://www.random.org (accessed July 2011)
23. Shen, G., Zhang, Y.-Q.: A Shadow Price Guided Genetic Algorithm for Energy Aware Task Scheduling on Cloud Computers. In: Tan, Y., Shi, Y., Chai, Y., Wang, G. (eds.) ICSI 2011, Part I. LNCS, vol. 6728, pp. 522–529. Springer, Heidelberg (2011), doi:10.1007/978-3-642-21515-5-62
24. Shen, G., Zhang, Y.Q.: A New Evolutionary Algorithm Using Shadow Price Guided Operators. Applied Soft Computing 11(2), 1983–1992 (2011), doi:10.1016/j.asoc.2010.06.014
25. Shen, G., Zhang, Y.Q.: An Evolutionary Linear Programming Algorithm for Solving the Stock Reduction Problem. International Journal of Computer Applications in Technology (IJCAT) 44(6) (2012)
26. Tian, L., Arslan, T.: A genetic algorithm for energy efficient device scheduling in real-time systems. In: 2003 Congress on Evolutionary Computation, pp. 242–247 (2003), doi:10.1109/CEC.2003.1299581
27. U.S. Energy Information Administration: Independent Statistic and Analysis, Renewable Energy Consumption and Electricity Preliminary 2006 Statistics (2006), http://www.eia.doe.gov/cneaf/solar.renewables/page/prelim_trends/rea_prereport.html (accessed July 2011)
28. US Environmental Protection Agency: EPA Report on Server and Data Center Energy Efficiency (August 2007), http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf (accessed July 2011)
29. Wang, L., Von Laszewski, G., Dayal, J., He, X., Furlani, T.R.: Thermal Aware Workload Scheduling with Backfilling for Green Data Centers. In: the 28th IEEE Intl. Conf. on Performance Computing and Communications, pp. 289–296 (2009), doi:10.1109/PCCC.2009.5403821
30. Wang, L., Von Laszewski, G., Dayal, J., Wang, F.: Towards Energy Aware Scheduling for Precedence Constrained Parallel Tasks in a Cluster with DVFS. In: 2010 10th IEEE/ACM Intl. Conf. on Cluster, Cloud and Grid Computing (CCGrid), pp. 368–377 (2010), doi:10.1109/CCGRID.2010.19
31. Wikipedia: Instructions Per Second (2011), http://www.en.wikipedia.org/wiki/Instructions_per_second (accessed July 2011)
32. Xie, Y., Wang, Z., Wei, S.: An efficient algorithm for non-preemptive periodic task scheduling under energy constraints. In: 6th Intl. Conf. on ASIC, pp. 128–131 (2005), doi:10.1109/ICASIC.2005.1611282
33. Yahoo Green: Sustainable Energy 101 (2011), http://green.yahoo.com/global-warming/globalgreen-140/sustainable-energy-101.html (accessed July 2011)
34. Yang, K., Liu, X.: Improving the Performance of the Pareto Fitness Genetic Algorithm for Multi-Objective Discrete Optimization. In: Intl. Symposium Computational Intelligence and Design, vol. 2, pp. 394–397 (2008)
35. Yuen, S., Chow, C.: A Genetic Algorithm That Adaptively Mutates and Never Revisits. IEEE Trans. Evolutionary Computation 13(2), 454–472 (2009)

Chapter 6
Thermal Management in Many Core Systems

Dhireesha Kudithipudi, Qinru Qu, and Ayse K. Coskun

Abstract. High power densities and operating temperatures in multi-processor systems impose a number of undesirable effects, including performance degradation, high operational and cooling costs, and reliability deterioration leading to system failures. Many-core systems bring exciting opportunities in design and system management owing to the ample hardware parallelism, while introducing novel challenges due to their complexity and the highly variant workload that is expected to run on these systems. Efficient thermal monitoring and management, designing thermally-aware architectures, and multi-level parameter optimization can alleviate some of the undesirable thermal effects while maintaining the desired performance and energy levels. In particular, these techniques aid in the evolution of green computing systems. This chapter provides a qualitative discussion on thermal management techniques for many-core systems. We will elucidate the following questions in detail: What are the specific design challenges in monitoring the temperature of large-scale systems? How can we exploit the multi-level optimizations at runtime in response to the dynamic behavior of the processor's workload? How do emerging workloads affect the thermal distribution on many-core systems?

6.1 Introduction

Many-core systems offer the potential to deliver 10 to 100 times the processing capacity compared to the traditional single-core systems. It is observed in early

Dhireesha Kudithipudi
Computer Engineering Department, Rochester Institute of Technology
e-mail: dxkeec@rit.edu

Qinru Qu
Electrical and Computer Engineering Department, Syracuse University
e-mail: qiqiu@syr.edu

Ayse K. Coskun
Electrical and Computer Engineering Department, Boston University
e-mail: acoskun@bu.edu

S.U. Khan et al. (Eds.): Evolutionary Based Solutions for Green Computing, SCI 432, pp. 161–185.
springerlink.com

prototypes that by integrating a number of simpler cores on a single die, this many-core chip technology has several advantages including integration levels. Non-ideal power supply scaling and increased power densities (W/cm^2) are among the challenges that come with scaling.

High power densities give rise to large amounts of heat generation. For instance, even when a single core dissipates 1W of power, the overall power dissipated will be around 100W for a 100 core system. This fact, combined with the areal heterogeneity of power consumption in many-core integrated circuits as well as the relatively slow lateral diffusion of heat in silicon [1, 2] yields localized thermal hot spots. As a consequence, the strong correlation between temperature and various chip failure mechanisms severely degrades the reliability and life span of the chip. In fact, a small difference in the operating temperature (i.e., 10 °C - 15 °C) can result in a 2X difference in the lifespan of these devices [3, 4]. Borkar estimates that the thermal packaging needs increase the total cost per chip by $1/W, when the chip power is 35-40W [5]. Therefore, it is critical to monitor, manage/control on-chip temperatures in order to maximize device lifetimes and assure computational correctness.

In a many-core platform, chip hot spots are strongly workload-dependent. In order to simultaneously maximize performance and reliability in these devices, tasks and resources should be managed in a thermally-aware manner such that temperatures do not exceed critical threshold values and thermal gradients are minimized. Thus, in general, the goal of thermal management is to maximize system performance (or maintain the desired performance) while minimizing temperature hot spots and gradients. Thermal management in many-core systems can be organized into the following three phases: (1) Thermal monitoring, which handles the placement of thermal sensors and the dynamic monitoring of system thermal profile to provide feedback to a runtime thermal management unit; (2) temperature prediction, which handles modeling and forecasting of the runtime thermal behavior for efficient system design and management; and (3) runtime thermal management, which is a set of techniques to mitigate the harmful effects of temperature under energy and/or performance constraints. This chapter discusses each of these three steps in detail.

6.2 Thermal Monitoring

Technology scaling has enabled the integration of many-cores on a single die. This high power density trend has led to overall elevated on-chip temperatures and localized areas of significantly higher temperatures, referred to as hot spots. It has been reported in [6] that the thermal gradient across chips has reached as high as 50 °C, and this value rises with higher operating frequencies. High temperatures and large thermal variations across the chip introduce a set of reliability hazards that may cause both unexpected circuit functionality and weakened physical parameters of

the chip. To mitigate such thermal effects, various dynamic thermal management and optimization techniques have been proposed. For any dynamic thermal management to be efficient, it is extremely critical to have an accurate temperature sensing on the chip (e.g., for each core in a many-core system) and relay this information to the thermal management unit.

Several research groups [2, 7–11] have developed optimization algorithms that result in using a minimum number of sensors while maintaining adequate coverage. These optimization algorithms fall into two main categories: uniform and non-uniform sensor placement.

6.2.1 Uniform Sensor Placement

Uniform sensor placement optimization schemes are intended for use with chips that have an unknown typical thermal pattern. The sensors are placed in a uniform static grid throughout the entire chip with the intention of being able to detect all temperature violations, regardless of where they occur on the chip. As mentioned previously, only a finely-grained grid of sensors is capable of achieving near-perfect accuracy. Due to significant cost restrictions associated with sensor overheads, the granularity of the grid must be bound, limiting the accuracy of this model.

A straight-forward linear interpolation approach is proposed in [2] to account for this restriction and refine the temperature measurements. The interpolation scheme with a 4×4 grid of sensors was shown in [2] to improve upon a static uniform grid of the same size with no interpolation by an average of 1.59°C across the SPEC2000 benchmarks [12] in a single-core processor.

One advantage of implementing a uniform thermal sensor allocation technique is that it does not rely on thermal profiling data. No knowledge of hot spot locations and temperatures needs to be acquired prior to implementing a technique of this type. This characteristic, however, limits the accuracy of the uniform grid model because the distances between the sensor locations and the hot spots cannot be minimized. Without knowledge of common resulting thermal maps, sensors arranged in a uniform grid will not aways be able to detect hot spots as accurately as the same number of sensors located near common hot spots.

6.2.2 Non-uniform Sensor Placement

Non-uniform sensor placement optimization schemes are intended for use where thermal maps from typical chip execution across several applications are available for analysis. These types of techniques take advantage of the known hot spots on the chip to determine the most advantageous locations for sensors to be placed. A

naive approach is to place a sensor on each hot spot found through thermal profiling across several applications. Unfortunately, this approach is not practical because a high number of hot spots is very likely to occur, and using a large number of sensors is not practical. Ideally, a minimum number of sensors would be arranged on the chip such to provide coverage of all possible hot spots. It has been shown in [13] that hot spots will not always remain in the same locations on the chip during execution of a single program, and various applications running on the same chip will show hot spots in different regions. Hot spot locations and temperatures are application dependent, and it is unlikely that a solution optimized for a single application will be sufficient for other workloads. One sensor placement configuration must suffice for all hot spots that may arise during the execution of any program.

Several methods that detect thermal violations with a limited number of sensors have been developed based on hot spot locations and temperatures found via thermal profiling. Skadron et al [7] have proposed Equation 6.1 to describe the maximum radius R between a hot spot and a potential thermal sensor location, while capping the error to a degree ΔT. The value ΔT denotes the difference between the maximum and minimum temperature value in the chip. In this equation, K is used to represent the effects of the materials included in the chip. These parameters include the thickness of the processor package-die, heat spreader, and thermal interface material multiplied by thermal resistivity factors specific to each material.

$$R = 0.5 \cdot K \cdot \ln\left(\frac{T_{max}}{T_{max} - \Delta T}\right) \tag{6.1}$$

6.2.3 Quality-Threshold Clustering

The algorithm described in [10] incorporates Equation 6.1 with the quality threshold (QT) clustering algorithm commonly used in gene clustering [14]. Treating the hot spots as points that must be clustered, the hot spot groupings and corresponding sensor locations are determined based on the values of T_{max} for all of the hot spots in each respective cluster. QT clustering is an iterative technique that assigns hot spots to clusters based on their physical locations on the chip relative to the other hot spots. The sensor location for each cluster is refined after the addition of a candidate hot spot to be the centroid of the included hot spots, thus obtaining the best possible sensor location for the given set of hot spot data points. The newly added hot spot will be kept in this cluster only if every other hot spot in the cluster is located within the distance R from the cluster center.

The method in [10] resulted in placing 23 sensors with $T_{max} = 3°C$ using the QT clustering technique on an Alpha 21364 processor core and the hot spots produced by the SPEC2000 benchmarks. The 23 sensors sensed the complete thermal profile of the core with an average error of 0.2899°C.

Even though the clustering algorithm proves to be sufficient for monitoring thermal events, it does not incorporate the number of clusters or sensors that are available to use for a specific design. This could be detrimental for several reasons. The algorithm does not end execution until every hot spot is placed in a cluster, creating new clusters where necessary to include hot spots that are located far away from the others. The number of sensors required by the QT clustering algorithm may not be available for use in a practical design. To place fewer sensors using QT clustering, the allotted hot spot to sensor distance value must be increased, which may decrease the accuracy of the entire model's results.

6.2.4 K-Means Clustering

The basic k-means clustering algorithm requires the number of sensors to be placed as an input parameter, k. The hot spots are placed into k different clusters, with a temperature sensor placed at the centroid of each cluster. The cluster assignments are chosen such that the mean squared distance from each hot spot to the nearest cluster center is minimized [15]. First, the k cluster centers are chosen randomly from the set of known hot spot points. Each hot spot is then assigned to a cluster C_j such that Euclidean distance $E(O_j, h_i)$ between the hot spot h_i and this cluster's center O_j is minimized. The equation to determine the Euclidean distance between two points is shown in Equation 6.2. In this equation, (h_{ix}, h_{iy}) represents the location of a hot spot h_i and (O_{jx}, O_{jy}) represents the location of a cluster center O_j in the (x,y) plane.

$$E(O_j, h_i) = (O_{jx} - h_{ix})^2 + (O_{jy} - h_{iy})^2 \tag{6.2}$$

At the end of each iteration, each cluster center is updated to be the centroid of the locations of all hot spots assigned to that cluster. The Euclidean distances between the hot spots and the cluster centers are then recomputed. If a new minimum distance between a hot spot and a different cluster center is found, the hot spot is reassigned to the corresponding cluster. This process is repeated until no hot spot are reassigned to a different cluster or the total sum of all Euclidean distances does not have a significant increase.

A thermal gradient-aware version of the k-means clustering algorithm has been proposed in [8]. The main goal of this approach is to place the temperature sensors to hot spots that typically have higher temperatures. The clusters are formed in 3-D space using each hot spot's temperature, t, as the third dimension of calculating Euclidean distance, as shown in Equation 6.3.

$$E(O_j, h_i) = (O_{jx} - h_{ix})^2 + (O_{jy} - h_{iy})^2 + (O_{jt} - h_{it})^2 \tag{6.3}$$

The cluster centers are updated in each iteration with consideration of hot spot temperature. The hot spots are weighted in the centroid calculation relative to the

magnitude of their temperatures. As in the basic k-means clustering algorithm, the cluster centers and hot spot cluster assignments are iteratively refined until no hot spot has been reassigned to a different cluster or the total sum of all 3-D Euclidean distances does not have a significant increase.

As shown in [8], the thermal gradient-aware k-means clustering resulted in an average error at best of 2.10°C placing 16 sensors over a single core and an average error of 1.63°C using 36 sensors. Using the same two numbers of sensors with identical thermal profiling data, the basic k-means clustering algorithm resulted in an average error of 4.58°C and 3.05°C. This shows 2.48°C and 1.42°C improvements, respectively. All experiments were run using HotSpot configured for the Alpha 21364, and hot spot positions were determined from thermal patterns pertaining to the SPEC2000 benchmarks [12].

Although thermal gradient-aware k-means clustering works well under many conditions, this technique is not always optimal in complex hot spot distribution scenarios and may produce solutions worse than the basic k-means approach. In such cases, hot spots are often sorted into inappropriate clusters due to their common temperature and regardless of their physical locations on the chip. Figures 6.1(a) show k-means clustering results with eight sensors. Although the clustering of hot spots varies slightly for the two methods, the sensor placement locations are almost identical. To make a difference in sensor placement, the temperatures of the hot spots used in thermal gradient-aware k-means calculations can be scaled according to Equation 6.4, where a is a constant specifying the steepness of the temperature gradient.

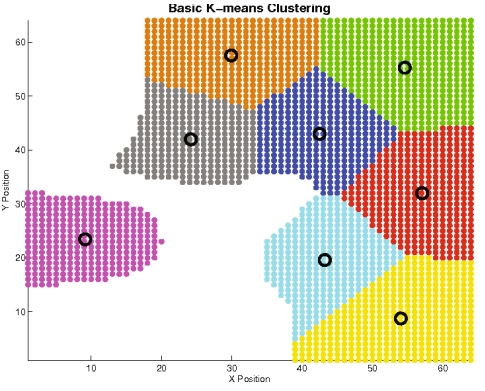
$$New\,Temperatures = a \cdot \frac{Original\,Temperatures}{Maximum\,Temperature} \tag{6.4}$$

Figures 6.1(b) and 6.1(c) show the clustering results of using Equation 6.4 with $a = 1000$ and $a = 2500$, respectively, on the same hot spot set used in Figures 6.1(a). In both situations, the sensors have been placed closer to the hot spots of higher temperature and further from the hot spots of lower temperature. Many of the hot spot cluster assignments, however, are not appropriate spatially. In Figure 6.1(c), for example, many of the red hot spots clustered with sensor S1 would be more appropriately grouped with the gray hot spots clustered with sensor S2, and vice versa.

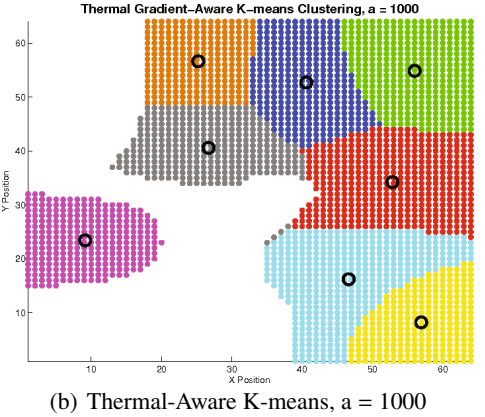
The work in [16] shows that while thermal-gradient aware k-means clustering is effective for single-core processors, it is not appropriate for multi-core processors with strong inter-core thermal interaction due to the unlikelihood of hot spots appearing in the same locations on every core.

6.2.5 *Determining Thermal Hot Spots to Aid Sensor Allocation*

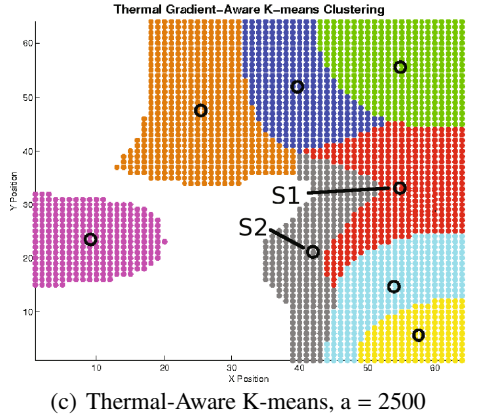
There are many properties to consider when determining which locations on the chip are considered as hot spots. Varying the determination rules has the potential to



(a) Basic K-means



(b) Thermal-Aware K-means, a = 1000



(c) Thermal-Aware K-means, a = 2500

Fig. 6.1. K-means clustering sensor placement variations.

significantly affect the resulting sensor placement locations. The trade-offs between full thermal map characterization and hot spot detection must be considered when identifying initial hot spot locations.

6.2.5.1 Local Hot Spots

One common determination rule is to assign a specified number of hot spots per functional block within the processing core, referred to as *local hot spots* [2, 10]. This technique encourages sensor placement across the entire die and is most appropriate for characterizing the full thermal map of the processor. The hot spot locations will be the points that reach the local maximum temperature for each functional block. For many functional blocks, the hottest points will be the near the edges of the block adjacent to a hotter component.

6.2.5.2 Global Hot Spots

A second method of hot spot determination is to record *global hot spots*, or any location on the die that reaches or surpasses a specified emergency temperature threshold, typically near 82°C or 355 *K* [2, 16]. Temperature sensors will be placed closer to the locations on the chip of significantly higher temperature, and thus will not likely be spread across the chip. This technique is best for quickly recognizing emergency temperatures as opposed to recovering the full thermal map of the entire processor. Furthermore, the number of hot spots determined by this technique is inversely correlated with the specified emergency temperature threshold. A higher threshold will result in fewer hot spots that could all be located within a single functional block in the processor. Alternatively, a low enough threshold will result in many hot spots, which could potentially cover more than 50% of the processor. A large number of hot spots would allow sensors to be spread through a larger area. As reported in [17], the integer register file is repeatedly the hottest component in the Alpha 21364 core across the SPEC2000 benchmarks. Choosing a high emergency temperature threshold could result in placing sensors only in the integer register file, while a lower threshold would allow sensors to be placed in adjacent functional blocks across the processor. The trade-offs between quick hot spot detection and full thermal map recovery must be considered.

6.2.6 Non-uniform Subsampling of Thermal Maps

In many-core architectures, there is a high likelihood of measuring a very large number of global hot spots. The number of hot spots may be so large that clustering

algorithms are not able to place a sufficient number of sensors near the hottest points. To reduce the number of points to be clustered while maintaining clear representation of thermal data, non-uniform subsampling algorithms can be used to obtain a subset of key thermal analysis locations on a chip. More samples are selected from regions of higher temperature and fewer points are selected from regions of lower temperature. Subsampling a thermal map will likely strike a balance between uniform temperature measurement and thermal emergency detection. After the thermal map has been subsampled, clustering algorithms can be used on the subsamples to determine sensor placement locations.

The two gradient based non-uniform subsampling algorithms for images proposed in [18] select sample pixels from a given image such that a constant gradient region will be represented by a number of samples linearly proportional to the gradient magnitude.

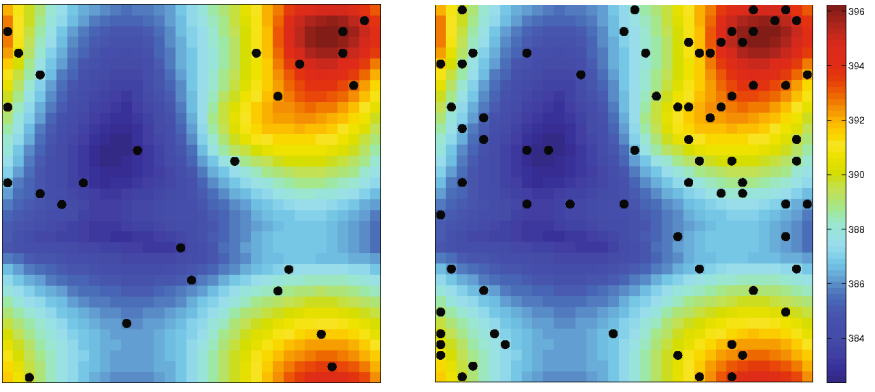
6.2.6.1 Deterministic Subsampling

The deterministic version of this algorithm states that in order to quantize the data set $\|\nabla I_1\|$ into Q levels, a list of pixel locations I_q with a quantized gradient norm of q must be built for each level q. After all pixels have been distributed into appropriate quantization levels,every s_q^{th} pixel in each list I_q will be selected, where $s_q = \lceil c/q \rceil$ for a constant c. This specification ensures that samples are selected more frequently in regions of high gradient. The constant value c is adjusted to yield a larger or smaller number of samples.

Applying subsampling algorithms to a full thermal map of a processor core while treating the temperature values as gradients results in more samples closer to the regions with more hot spots and fewer samples near cooler regions. Adjusting the value of c affects the number of samples taken from a thermal map. Before performing subsampling, the temperature should be normalized according to Equation 6.5 using the minimum temperature T_{min} and maximum temperature T_{max} in the thermal map used for analysis.

$$I_{norm} = \frac{\|\nabla I_1\| - T_{min}}{T_{max} - T_{min}} \tag{6.5}$$

Performing the deterministic non-uniform subsampling algorithm on a sample thermal map yielded the sampled results displayed in Figure 6.2(a) and 6.2(b). Both runs used 25 quantization levels. Figure 6.2(a) shows the results from setting the constant $c = 5$. This constant produced many fewer samples than setting the constant $c = 0.25$ as shown in Figure 6.2(b). Both plots reveal sampling locations spread throughout the entire thermal map. This algorithm is fairly conservative and similar to uniform sampling.



(a) Thermal deterministic subsamples for $c = 5$ (b) Thermal deterministic subsamples for $c = 0.25$

Fig. 6.2. Thermal deterministic non-uniform subsampling results with 25 quantization levels.

6.2.6.2 Stochastic Subsampling

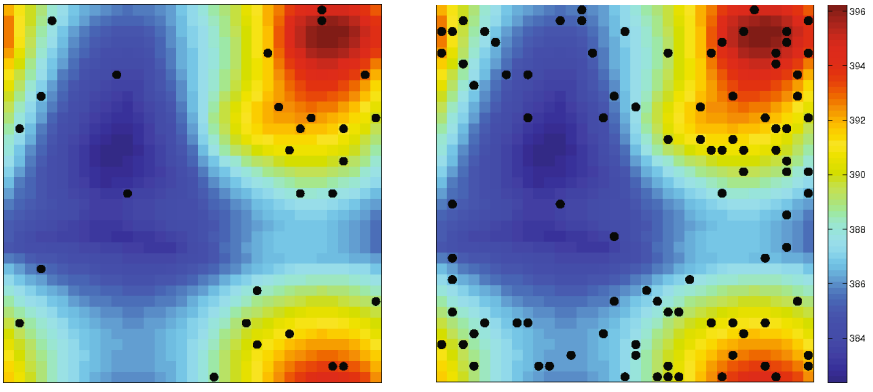
A stochastic version of non-uniform sampling looks at each individual pixel location (i, j) and decides whether to select this pixel as a sample or not. Pixels are selected with probability $p(i, j) = min(\alpha * \|\nabla I_1\|(i, j), 1)$. Adjusting the proportionality constant α yields fewer or additional samples.

Performing the stochastic non-uniform subsampling algorithm on a sample thermal map yielded the sampled results displayed in Figure 6.3. Figure 6.3(a) shows the results from setting the constant $\alpha = 1.5$, which produced fewer samples than setting the constant $\alpha = 5$ as shown in Figure 6.3(b). Both plots show many sampling points in the hottest regions, and only one or two samples in the coolest region. The samples taken in this stochastic subsampling algorithm accurately reflect the thermal gradient of the chip.

K-means clustering can be used to determine appropriate sensor locations when treating the samples as points to be clustered. Due to the sampled locations, the resulting thermal sensor placement will be able to maintain a balance between profiling the entire core and detecting thermal emergencies.

6.3 Temperature Modeling and Prediction Techniques

Temperature is a prominent factor in determining the performance, reliability, and leakage power consumption of modern processors. As a result, estimating temperature accurately is an essential step in successful thermal management. This



(a) Thermal stochastic subsamples for $\alpha = 1.5$

(b) Thermal stochastic subsamples for $\alpha = 5$

Fig. 6.3. Thermal stochastic non-uniform subsampling results.

section first discusses thermal modeling principles and tools. The goal of thermal modeling is to provide an accurate assessment of temperature to help system design and to enable evaluation of runtime scenarios. The second part of the section discusses thermal prediction, which relies on low cost techniques to estimate current or future temperature based on thermal sensors readings or power/performance profiles. Thermal predictions can be utilized as early warnings in predictive dynamic thermal management policies (see Section 6.4) or for providing more comprehensive thermal estimates for systems without a sufficient number sensors. In both cases, the main objective of thermal prediction is to improve decision making during thermal management.

6.3.1 Thermal Modeling

There are several temperature measurement methods at chip level. Gathering the temperature data by using point contact methods (thermocouples) is limited by the large number of points to be monitored and the small size of the components. Connecting tens or hundreds of thermocouples is very time consuming. Infrared (IR) thermal imaging (e.g., [19, 20]) is a new technique which addresses these issues by providing comprehensive two-dimensional (2-D) maps of thousands of temperatures in a matter of seconds. This is accomplished without the need to make contact with the components. This approach is, however, expensive and time-consuming and can only be applied post-design. Finally, on-chip thermal sensors provide temperature readings as well; however, the number of sensors that can be deployed on a chip is limited by chip design constraints such as area and cost (efficient sensor placement techniques are reviewed in Section 6.2). It is, therefore, important to have full-chip thermal models and simulation tools that can provide the temperature profile of the die.

On a chip, heat is generated in both the substrate and the interconnections. Additional power dissipation results from Joule heating (or self-heating) caused by the flow of current in the interconnect network [21]. Although interconnect Joule heating constitutes only a small fraction of the total power dissipation in the chip, the temperature rise in the interconnections due to Joule heating can be significant. This is because of the fact that interconnects are typically located away from the heat sink by several layers of insulating materials which have lower thermal conductivities than that of Silicon.

The major source of heat generation is the power dissipation of devices that are embedded in the substrate. The operating temperature of a VLSI chip can be calculated from the following linear equation:

$$T_{chip} = T_a + R_\theta \frac{P_{tot}}{A} \tag{6.6}$$

where T_{chip} is the average chip (Si junction) temperature, T_a is the ambient temperature, P_{tot} (in W) is the total power consumption, A (in cm^2) is the chip area, and R_θ is the equivalent thermal resistance of the substrate (Si) layer plus the package and heat sink ($cm^2 \deg C/W$). Thus, to compute temperature, we need to compute or measure the power consumption, construct the chip thermal model to calculate the thermal resistances, and have information on the environment (i.e., ambient temperature).

Power consumption of a VLSI chip consists of dynamic power, short-circuit power, and static (or leakage) power, out of which dynamic and leakage power are dominant in today's processors. Power modeling and measurement are closely connected areas to thermal modeling. Current power measurement practices involve using current sensors and power meters for chip or server level measurements (e.g., [22, 23]), or using voltage and current sensors on the chip, if available. Processor power modeling tools include architecture-level models such as Wattch [24] or McPat [25], which can be connected to instruction-accurate performance simulators, or circuit-level models that are available in commonly used circuit design/analysis tools.

While Equation 6.6 computes the temperature of an entire chip, calculating the temperature of individual blocks (e.g., microarchitectural units or cores) or of fine-grained grid cells requires constructing a thermal model to simulate the heat transfer among the units on the chip as well as the heat flow from the junction to air.

HotSpot [26] provides an automated thermal modeling tool to construct the thermal R-C network given the package properties, chip properties, and the chip layout. The tool is capable of providing both steady state and transient temperature response estimates for a given power consumption trace. Figure 6.4 demonstrates the R-C network for the chip layer. The heat spreader and heat sink are also modeled through a similar network.

To speed up performance and thermal simulation, one method is to emulate the core execution on FPGA platforms and run an automated thermal R-C network based model for computing temperature in conjunction with the performance emulation [28]. Recent extensions of automated thermal models include modeling capabilities for 3D stacked systems [29, 30] and for microchannel-based liquid cooling [31].

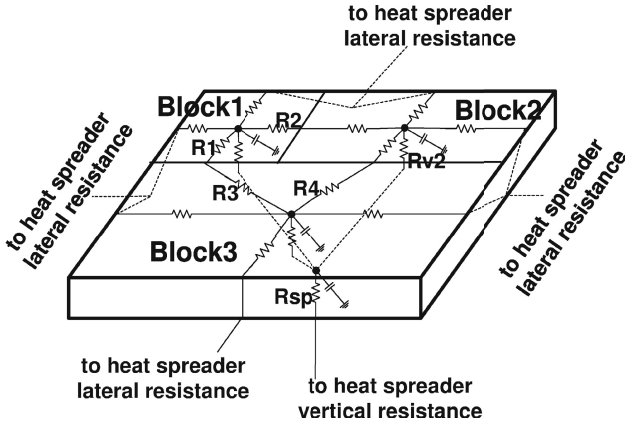


Fig. 6.4. Thermal R-C network for the chip [27]

In addition to these thermal simulation tools specifically targeting estimating chip temperature, general purpose finite element solvers can be utilized for temperature modeling as well.

The automated R-C network based thermal simulators provide high accuracy in modeling [32–34]. These models, however, have considerable runtime overhead, preventing them to be leveraged for runtime temperature estimates. For this reason, several low-cost temperature prediction (or forecasting) techniques have been proposed. Next, we discuss thermal prediction methods.

6.3.2 Temperature Prediction

Computing chip temperature using the linear equation (Eqn. 6.6) is one method for quick temperature estimation, and this method has been used in temperature-aware design optimization [35]. Another method of temperature prediction focuses on extrapolating the temperature values on the chip in detail based on temperature readings from a few sensors. This approach is especially useful for large many-core systems, considering a sufficient number of thermal sensors may not be available (see Section 6.2 for a detailed discussion on thermal sensor placement methods).

Sharifi et al. introduce a technique to accurately estimate the temperature at arbitrary locations on the die based on the noisy temperature readings from a limited number of sensors which are located further away from the locations of interest [36]. Their technique first constructs a R-C network of the chip offline. Model order reduction is used to generate a smaller yet accurate system for the thermal model. After the thermal model is constructed, the technique applies Kalman filtering to the reduced order model of the system. The calibration ends when the filter reaches its steady state. The resulting steady-state filter is used at runtime to perform the

temperature estimation. The Kalman filter estimates the temperature in a predict-correct manner based on a few temperature sensor values and power consumption estimates. At runtime, the predictor projects temperatures using the current temperatures. A "measurement update" stage following the projections incorporates the new measurements into the *a priori* estimate to obtain an improved *a posteriori* estimate of the temperature.

Another method for fine-grained temperature estimation using a limited number of thermal sensors is using Fourier analysis techniques [37]. Specifically, the authors use Nyquist-Shannon sampling theory to determine the fewest number of thermal sensors that can fully characterize the runtime thermal status of a processor. Given a limited number of thermal sensors, the technique applies signal reconstruction techniques that generate a full-resolution thermal characterization of a processor. Figure 6.5 demonstrates the flow of the runtime thermal characterization. The technique first takes the Fast Fourier Transform (FFT) of the temperature samples and the space-domain representation of the interpolation functions (sinc, nearest neighbor, cubic B-spline, linear function). It then multiplies the resultant spectral-domain representations to get the FFT of the reconstructed 2D thermal signal. Finally, inverse FFT (IFFT) is applied to get the full-resolution thermal characterization in the space-domain. The authors also propose methods that handle uniform and non-uniform sensor placements [37]; note that non-uniform placements may arise due to design and layout constraints.

Instead of using a few number of sensors for a detailed estimation of the on-chip temperature, some prediction techniques leverage the existing sensors and the thermal measurement history to predict the future temperatures. Detailed thermal estimates are useful in temperature management as the management policies can make more informed decisions using the fine-grained thermal projections. Temperature forecasting, on the other hand, focuses on learning the temperature behavior and estimating the near future. In this way, new policies can be designed that proactively make temperature-aware decisions, such as off-loading a core that is projected to get hotter within the next prediction interval [38] (see Section 6.4 for a detailed discussion of temperature management).

Yeo et al. estimate short-term (application-level) temperature changes using least square regression fit over the thermal sensor data, and compute core-level (slower) temperature changes based on steady-state temperature of the application [39]. The overall temperature estimate is then a weighted sum of short-term and long-term temperature estimates, where the weight factors are selected empirically. Prediction experiments on SPEC 2006 suite demonstrate high accuracy.

While application-based prediction through least squares fit provides good estimates when the predictor is trained with the application thermal characteristics, there are several challenges associated with the method. First, when the prediction distance (i.e., how many steps or seconds into the future are being predicted) is altered, the prediction accuracy may vary significantly [38]. The reason is that, as soon as we predict more than a few time steps into the future, the term with the biggest exponent in the least-squares fitting function dominates, and the prediction accuracy degrades from that point on. Another challenge is runtime adaptation: for dynamically

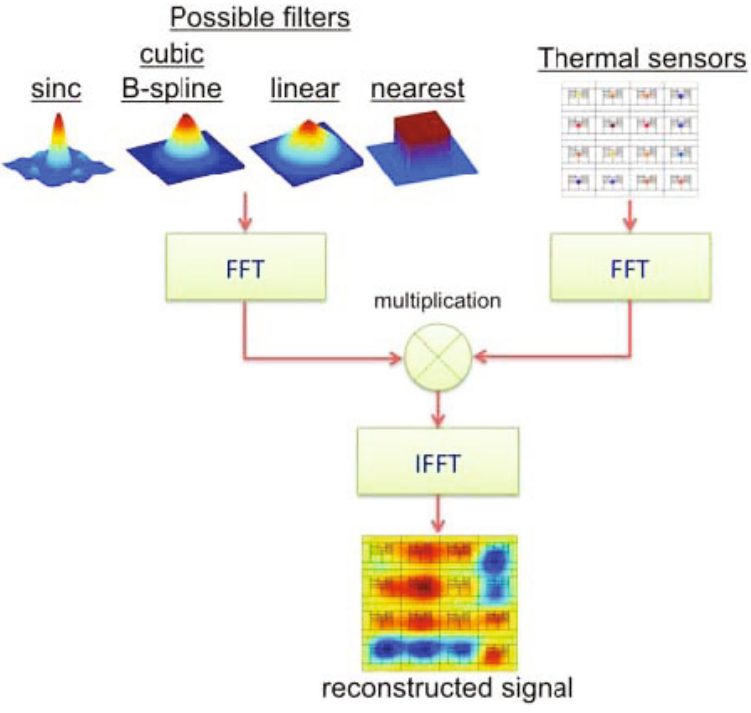


Fig. 6.5. Flow of the thermal characterization technique using FFT-based analysis [37]

changing workload, recursive least-squares continuously needs to update the coefficients of the model as new data arrive (otherwise, accuracy would drop).

Auto-regressive moving average (ARMA) modeling for forecasting temperature [38,40] addresses the challenges outlined above through construction a predictor using the following steps. *(1) Identification of the model degree and estimation of model parameters.* A well-fitting ARMA model has a low final prediction error [38]. *(2) Checking the model.* The error between measured and predicted values should be randomly distributed around a mean value of zero. *(3) Online adaptation using sequential probability ratio test (SPRT).* The probabilistic test rapidly determines if the existing model is not fitting the current workload temperature dynamics and if a new model should be constructed.

$$y_t + \sum_{i=1}^{p} (a_i \, y_{t-i}) = e_t + \sum_{i=1}^{q} (c_i \, e_{t-i}) \tag{6.7}$$

An ARMA(p,q) model is described by Equation 6.7. In the equation, y_t is the value of the series at time t (i.e., temperature at time t), a_i is the lag-i autoregressive coefficient, c_i is the moving average coefficient and e_t is called the noise, error or the residual. The residuals are assumed to be random in time (i.e., not autocorrelated),

and normally distributed. p and q represent the orders of the autoregressive (AR) and the moving average (MA) parts of the model, respectively.

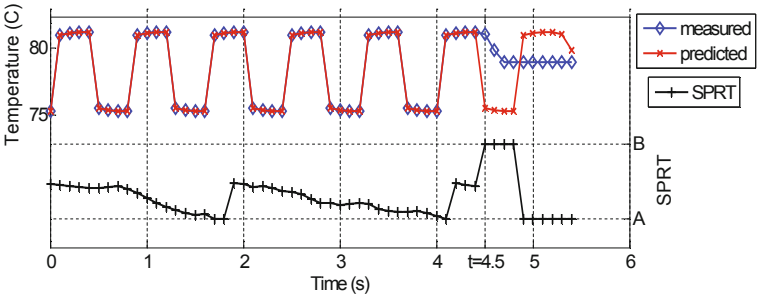


Fig. 6.6. ARMA-based temperature prediction and online detection of variations in thermal characteristics using a likelihood ratio test [38]

Figure 6.6 demonstrates the predicted and estimated temperature values using ARMA. Note that during this experiment, temperature dynamics change, and the SPRT detects this change immediately (see $t = 4.5s$ in the figure). A and B correspond to the SPRT decision making thresholds, computed based on user-defined false and missed alarm percentages.

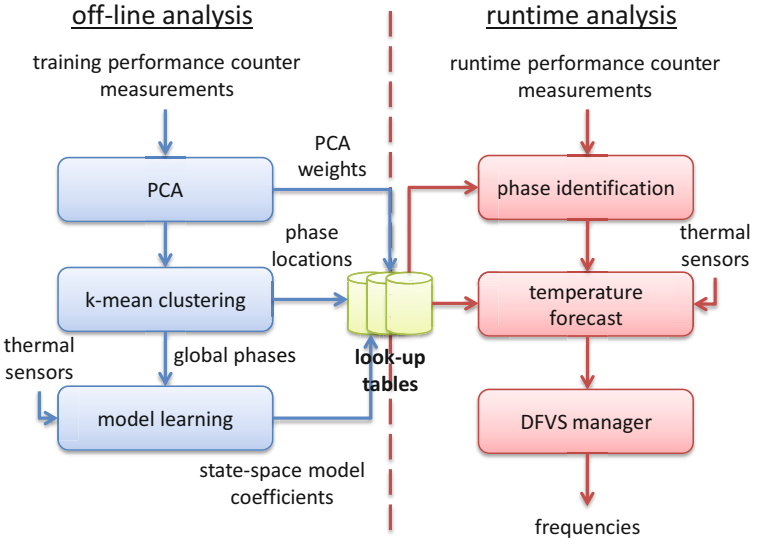


Fig. 6.7. Flow of the approach combining workload phase identification and thermal forecasting [41]

Recent work utilizes workload phase recognition in addition to temperature sensor information for low-cost, high-accuracy prediction [41]. Through principle component analysis (PCA), it is possible to determine the most relevant metrics to identify different workload phases. Then, through k-means clustering and model learning, the approach constructs a predictor offline. At runtime, the overhead is limited to identifying the phase through performance counter measurements and forecasting temperature. This forecast can then be used in thermal management decisions through tuning voltage-frequency settings. The flow of the approach is shown in Figure 6.7. Next, this chapter discusses runtime temperature measurement techniques that utilize various levels of temperature estimation and forecasting.

6.4 Runtime Thermal Management

State-of-the-art dynamic thermal management (DTM) techniques adapts to the changing workload and application environment in order to achieve the most robust performance. In general, adaptive thermal management techniques can be divided into model-based and model-free approaches. Model based thermal management learns the temperature model of the computing system. A survey on thermal modeling techniques are provided in Section 6.3. If the predicted temperature exceeds a threshold then actions will be taken to prevent hot spot. Instead of learning the temperature model, model-free adaptive thermal management adapts the control actions directly based on the collected feedback from the system. Both techniques monitor the system and extract information from recent system dynamics to develop the model either for temperature prediction or for decision making. Next, we discuss recent work in both model-based and model-free adaptive thermal management.

6.4.1 Model-Based Adaptive Thermal Management

The model-based thermal management can also be referred as predictive thermal management. These techniques make control decisions based on projected system dynamics. As long as the prediction is accurate, thermal emergencies can be avoided by taking appropriate actions in advance. Several predictive DTM policies have been proposed recently [42–45]. The main difference among these methods lies in the adopted temperature prediction models and the DTM actions. A detailed discussion of temperature prediction is provided in section 6.3. In this section we focus on the actuation based on the predictive models.

Two commonly used knobs by the DTM controllers are temperature-aware workload migration (or workload allocation) and dynamic voltage and frequency scaling (DVFS). The main goal of the DTM controller is to achieve an evenly distributed workload either spatially or temporally to limit temperature increase. We focus

on several recently proposed predictive workload migration/allocation and DVFS-based techniques in this chapter.

6.4.1.1 Predictive Temperature-Aware Task Migration

In a multi-core single-chip processor, the temperature of a core is not only determined by the power consumption of the software program running on it but also on its heat dissipation ability and the temperature of its neighbors. The basic concept of temperature-aware task migration is to dynamically move "hot" processes to cores with superior heat dissipation and/or cores surrounded by cooler neighbors while moving "cool" processes to cores with inferior heat dissipation and/or cores surrounded by hotter neighbors. Most predictive thermally-aware task migration methods assume that the power consumption of a task within a given workload can be obtained either by profiling or estimation. All of them require temperature prediction models which estimate either the steady state temperature [42, 44] or the temperature in future time instances [42, 43].

Based on predicted temperature, Yeo et al. [42] propose a thermally-aware task migration policy called Predictive Dynamic Thermal Management (PDTM). When a task running on a processor is projected to exceed the temperature threshold, it will be moved to a processor that is predicted to be the coolest in the future. The policy is evaluated on an Intel Quad-core processor. Processor temperature can be read from the digital thermal sensors embedded in the cores. The applications running on the system are libquantum, perlbench, bzip2, hmmer from SPEC 2006 Benchmarks. The prediction model achieves very high accuracy and the temperature prediction error is only 1.6%. Compared to existing thermal management schemes, the proposed task migration policy reduces the average temperature by 7% and peak temperature by 3 °C.

Coskun et al. [43] propose a Proactive Thermal Balancing (PTB) policy. Similar to the PDTM policy [42], this policy moves the tasks from a processor core predicted to be hotter to a core that is expected to be cooler. The difference is that PDTM moves the current running tasks while PTB gives priority to moving the tasks in the waiting queue. This enables the hot core to have a period with fewer number of threads and cool down after current task finishes, and hence avoids hot spots. As mechanism of migrating a waiting task in the ready queues has already been implemented in the OS scheduler for load balancing purposes, this technique does not introduce additional implementation overhead. Experimental results on an UltraSPARC T1 based processor model show that PTB reduces hot spot occurrences, spatial gradients, and thermal cycles by 60%, 80% and 75% respectively on average compared to reactive thermal management, where workload migration decisions are made after cores reach a given threshold. In addition, PTB only incurs a performance cost of less than 2% with respect to the default scheduling policy for load balancing, as measured on real-life processors.

Both PTB and PDTM are centralized approaches. This means that these techniques require a controller that monitors the temperature and workload distribution

of the entire chip and makes global decisions of resource allocation. Such centralized approaches may not scale well to a large number of cores. As the number of processing elements grows, the complexity of solving the resource management problem grows super-linearly. Furthermore, a centralized monitoring and commanding framework incurs a large overhead, as communication between central controller and cores will increase exponentially [46].

Ge et al. [44] propose a framework for distributed thermal management where a balanced thermal profile can be achieved by thermal throttling as well as thermally-aware task migrations among neighboring cores. The framework has a low-cost agent residing in each core. The agent observes the workload and temperature of local processor while communicating and exchanging tasks with its nearest neighbors. The goal of task migration is to distribute tasks to processors based on their heat dissipation capabilities and also ensure that each processor has a balanced mix of high power and low power tasks. The authors refer to the proposed technique as distributed thermal balancing migration (DTB-M) as it aims at balancing the workload and temperature of the processors simultaneously.

The DTB-M policy basically can be divided into 3 phases: temperature checking and prediction, information exchange, and task migration. Figure 6.8 shows the flowchart of the DTB-M execution in the i^{th} core. A DTB-M agent is initially neutral. It will enter the master mode if any of the three scenarios are true: (1)The local temperature reaches a threshold T_m (in this case, the DTB will first stall the processor to let it cool down before it enters the master mode), (2) the predicted future peak temperature exceeds the threshold, (3) the temperature difference of the local core and the neighbor core exceeds the thermal balancing threshold. Otherwise, it will enter the slave mode. A master DTB agent issues a task migration request to its nearest neighbors which are DTB slaves.

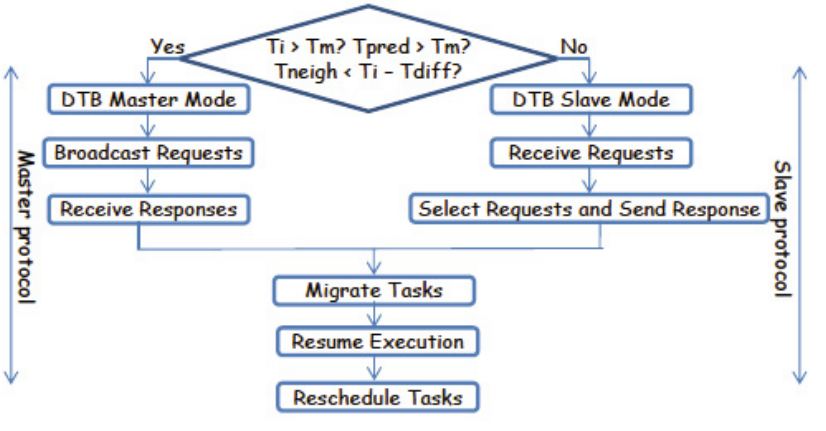


Fig. 6.8. Flow of the Distributed Thermal Balancing-Migration (DTB-M) technique

The DTB-M policy has two components. Both of them have quadratic complexity with respect to the number of tasks in the local task queue. The first component is a steady state temperature based migration policy (SSTM). It considers the long term thermal behavior of tasks, and distributes tasks to cores based on their different heat dissipation ability. The second component of DTB-M is a temperature prediction based migration policy (TPM), which predicts the peak temperatures of different task combinations when making migration decisions. It ensures that each core executes a balanced mixture of high power and low power tasks without having thermal emergencies.

These two components are complementary to each other as the SSTM considers long term average thermal effect and the TPM considers short term temporal variations. Experimental results show that the DTB-M has 66.79% lower frequency of hot spots and 40.21% higher performance compared to the existing techniques. Furthermore, DTB-M also has much lower migration overhead. The number of migrations is reduced by 33.84% while the overall migration distance is reduced by 70.7% due to the fact that processors only communicate and exchange tasks with their nearest neighbors.

6.4.1.2 Predictive DVFS for Thermal Management

Cochran et al. [41] utilize the program internal characteristics to predict future temperature. We know today that the execution of a program typically includes multiple distinct phases. A program phase is a stage of execution in which a workload exhibits near identical power, temperature or performance characteristics. Therefore, as long as we are able to identify the program phases during the run time and construct the relation between operating temperatures and program phases, we can predict the future temperature of an application.

The run time characteristics of a program can be characterized by architectural events such as instructions per cycle (IPC), memory access rate, cache miss rate and floating point (FP) instructions executed, etc. After selecting the most relevant events, the next step is to group similar program phases together as similar behavior will produce similar operating temperature. This process is carried out on a training data set with data points which are extracted from a set of representative workloads. The aim of the grouping process is to classify the n data points to k clusters such that the sum of the square distances between each training data point to its cluster center is minimized. Then a linear model is used to represent the relation between temperature and the workload phase. The model is trained using least square fitting. At every regular time interval, the event counters information is collected and the temperature model is evaluated for every available frequency setting. The highest frequency which will not cause thermal violation is selected to run for the next interval. Experimental results show that, compared to a reactive DVFS based thermal management policy, this algorithm cuts down the thermal violation considerably while reducing the runtime by 7.6%.

While all of the above mentioned techniques aim at reducing the temperature hot spots as the main objective, Bartolini et al. [45] propose to perform DVFS for both energy minimization and temperature management. A decoupled energy and thermal controller is introduced in this work. Based on the estimated clock cycles per instruction (CPI) count and performance constraints, the energy controller calculates the optimal frequency trajectory ($fEC(t)$) that minimizes energy while meeting the performance requirements. The thermal controller selects the clock frequency that has the least deviation from $fEC(t)$ while keeping the core temperature below the given threshold. The thermal controller predicts the future temperature using an ARX (AutoRegressive eXogenous) model, which is derived from a self-calibration routine.

6.4.1.3 Model-Free Adaptive Thermal Management

While model-based DTM selects action based on predicted temperature, model-free thermal management learns the actions directly. Ge et al. [47] propose to apply reinforcement learning (RL) to the DTM problem for multimedia applications. They model the DTM problem as a stochastic control process and adopt the RL algorithm to find the optimum policy during runtime. They model the processor's DTM controller as a learning agent and consider the rest of the system as the environment. After the learning agent takes an action, it observes the environment and estimates the reward or penalty induced by this action. The agent learns from this experience and tries to improve its future decisions to maximize the reward through the learning model. The proposed learning model has very small run time overhead. It only incurs a few table lookups and some simple arithmetic operations. Compared to a policy without thermal management, this learning policy reduces thermal violations by 34.27% while maintaining similar run times.

Coskun et al. [48] propose a switching experts based learning algorithm for thermal management. Their learning technique leverages a set of management policies, including dynamic power management to turn off idle cores (DPM), DVFS, thread migration and load balancing. These policies are referred to as experts. On top of the experts, there are specialists which are higher level policies that determines which expert to use in the next interval. Each specialist is assigned with a weight which is a function of performance and temperature. At each time interval, only specialists that select the active expert have their weight updated. The specialist that has the highest weight is selected and the corresponding expert policy is executed. Through updating weights for the specialists, the technique guarantees to converge to the policy that is best-fitting to current workload dynamics. Experimental results show that this learning technique is able to improve system thermal behavior, reduce energy, and increase performance significantly compared to the default scheduling policy in the OS.

6.4.1.4 Learning Based Thermal Management at Machine and Server Level

Learning-based DTM techniques are not restricted to microprocessors and could be applied at a higher level, such as a data center. For a data center, the major problem is to keep the temperature of the server chassis under a threshold while reducing the computing power and the power consumption of the cooling system, e.g., cooling fan or Computer Room Air Condition (CRAC), as much as possible. Pakbaznia et al. [49] propose a workload placement algorithm for large data centers. Their algorithm relies on an exponential workload requests predictor to forecast the future income requests to a data center. The rationale behind this prediction model is that the data center workloads typically show a repetitive pattern with a period in the order of hours, days, weeks and so forth. The accuracy of the prediction model is high: the predicted incoming requests are within 5% of the real incoming requests. The technique employs Integer Linear Programming (ILP) to allocate the workload and turn on or off specific servers to minimize the total data center energy. Experimental results show that their algorithm achieves consistent energy savings compared to greedy algorithms during a 24 hour simulation.

6.5 Conclusions

Thermal Management is an interesting and challenging design problem for the next generation processors. Thermal gradients are increasing with each new technology generation, forcing designers to think outside the traditional realm. Each of the design phases that were discussed in this chapter are interdependent and support the dynamic thermal management. In the future generation processors, the prediction and response mechanisms can further benefit from using evolutionary based algorithms for thermal management. For example, the run-time controller to manage the different thermal policies uses complex non-linear models and the time constants involved in arriving at accurate solutions is large. Designers can look at lean algorithms that provide faster and simpler numerical solutions. Such algorithms can benefit by abstracting design coefficients from different hierarchies, including thermal, electrical and system level. Another critical trend in the future generation processors is the use of 3D integration, with JEDEC [50] standards established recently. Thermal management in these systems requires an awareness of the physical behavior of the different stacks, associated cooling mechanisms, temperature control mechanisms that interface across the stacks, and algorithms that optimally distribute or allocate resources based on the thermal gradients.

Acknowledgements. The authors would like to thank Katherine E. Dellaquila from Hewlett Packard for her contributions on the thermal sensor placement mechanisms section.

References

1. Alley, R., Soto, M., Kwark, L., Crocco, P., Koester, D.: Modeling and validation of on-die cooling of dual-core cpu using embedded thermoelectric devices. In: Twenty-fourth Annual IEEE Semiconductor Thermal Measurement and Management Symposium, SemiTherm 2008, pp. 77–82 (March 2008)
2. Memik, S.O., Mukherjee, R., Ni, M., Long, J.: Optimizing Thermal Sensor Allocation for Microprocessors. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 27(3), 516–527 (2008), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4454017
3. Viswanath, R., Wakharkar, V., Watwe, A., Lebonheur, V.: Thermal performance challenges from silicon to systems. Intel Technology Journal Q3, 1–16 (2000)
4. Xiang, Y., Chantem, T., Dick, R.P., Hu, X.S., Shang, L.: System-level reliability modeling for mpsocs. In: Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, CODES/ISSS 2010, pp. 297–306 (2010), http://doi.acm.org/10.1145/1878961.1879013
5. Borkar, S.: Design challenges of technology scaling. IEEE Micro 19(4), 23–29 (1999)
6. Gronowski, P., Bowhill, W., Preston, R., Gowan, M., Allmon, R.: High-performance microprocessor design. IEEE Journal of Solid-State Circuits 33(5), 676–686 (1998)
7. Skadron, K., Huang, W.: Analytical model for sensor placement on microprocessors. In: 2005 International Conference on Computer Design, pp. 24–27 (2005), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1524125
8. Mukherjee, R., Memik, S.O.: Systematic temperature sensor allocation and placement for microprocessors. In: Proceedings of the 43rd Annual Design Automation Conference, DAC 2006, pp. 542–547. ACM, New York (2006)
9. Kwasinski, A., Kudithipudi, D.: Towards integrated circuit thermal profiling for reduced power consumption: Evaluation of distributed sensing techniques. In: Proceedings of the International Conference on Green Computing, GREENCOMP 2010, pp. 503–508. IEEE Computer Society, Washington, DC (2010)
10. Yun, X.: On-Chip Thermal Sensor Placement, Master's, University of Massachusetts Amherst (2008), http://scholarworks.umass.edu/cgi/viewcontent.cgi?article=1242&context=theses
11. Dellaquila, K.: Thermal Profiling of Homogeneous Multi-Core Processors Using Sensor Mini-Networks, Master's, Rochester institute of Technology (2010), https://ritdml.rit.edu/bitstream/.../KDellaquilaThesis8-2010.pdf?...1
12. SPEC-CPU 2000, Standard Performance Evaluation Council, Performance Evaluation in the New Millennium, Version 1.1 (2000)
13. Skadron, K., Lee, K.: Using Performance Counters for Runtime Temperature Sensing in High-Performance Processors. In: 19th IEEE International Parallel and Distributed Processing Symposium, pp. 232a–232a (2005), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1420152
14. Burger, D., Austin, T.: SimpleScalar Tutorial. In: 30th International Symposium on (1997), http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:SimpleScalar+Tutorial#2
15. Macqueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
16. Long, J., Memik, S., Memik, G., Mukherjee, R.: Thermal monitoring mechanisms for chip multiprocessors. ACM Transactions on Architecture and Code Optimization 5(2), 1–33 (2008), http://portal.acm.org/citation.cfm?doid=1400112.1400114
17. Skadron, K., Stan, M., Huang, W., Velusamy, S.: Temperature-aware microarchitecture: Extended discussion and results. University of Virginia, Department of Computer Science (2003), http://scholar.google.com/scholar?q=intitle:Temperature-Aware+Microarchitecture:+Extended+Discussion+and+Results#0

18. Sabuncu, M.R., Ramadge, P.J.: Gradient based nonuniform subsampling for information-theoretic alignment methods. In: 26th Annual International Conference of the IEEE on Engineering in Medicine and Biology Society (IEMBS), pp. 1683–1686 (2004)
19. Cochran, R., Nowroz, A.N., Reda, S.: Post-silicon power characterization using thermal infrared emissions. In: ISLPED, pp. 331–336 (2010)
20. Mesa-Martinez, F.J., Nayfach-Battilana, J., Renau, J.: Power model validation through thermal measurements. In: ISCA, pp. 302–311 (2007)
21. Pedram, M., Nazarian, S.: Thermal modeling, analysis, and management in vlsi circuits: Principles and methods. Proceedings of the IEEE 94(8), 1487–1501 (2006)
22. Isci, C., Martonosi, M.: Runtime power monitoring in high-end processors: Methodology and empirical data. In: Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 36, p. 93 (2003)
23. Khan, M.A., Hankendi, C., Coskun, A.K., Herbordt, M.C.: Software optimization for performance, energy, and thermal distribution: Initial case studies. In: IEEE Workshop on Thermal Modeling and Management: From Chips to Data Centers (in conj. with Green Computing Conference (IGCC)) (2011)
24. Brooks, D., Tiwari, V., Martonosi, M.: Wattch: a framework for architectural-level power analysis and optimizations. In: ISCA, pp. 83–94 (2000)
25. Li, S., Ahn, J.H., Strong, R.D., Brockman, J.B., Tullsen, D.M., Jouppi, N.P.: Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In: MICRO 42: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, pp. 469–480 (2009)
26. Skadron, K., Stan, M., Huang, W., Velusamy, S., Sankaranarayanan, K., Tarjan, D.: Temperature-Aware Microarchitecture. In: ISCA, pp. 2–13 (2003)
27. Skadron, K., Stan, M.R., Sankaranarayanan, K., Huang, W., Velusamy, S., Tarjan, D.: Temperature-aware microarchitecture: Modeling and implementation. In: TACO, vol. 1(1), pp. 94–125 (2004)
28. Atienza, D., Valle, P.D., Paci, G., Poletti, F., Benini, L., Micheli, G.D., Mendias, J.M.: A Fast HW/SW FPGA-Based Thermal Emulation Framework for Multi-Processor System-on-Chip. In: Design Automation Conference (DAC), pp. 618–623 (2006)
29. Coskun, A.K., Atienza, D., Rosing, T.S., Brunschwiler, T., Michel, B.: Energy-efficient variable-flow liquid cooling in 3d stacked architectures. In: DATE, pp. 111–116 (2010)
30. Link, G.M., Vijaykrishnan, N.: Thermal trends in emerging technologies. In: Proceedings of the 7th International Symposium on Quality Electronic Design, ISQED 2006, pp. 625–632. IEEE Computer Society, Washington, DC (2006)
31. Sridhar, A., Vincenzi, A., Ruggiero, M., Atienza, D., Brunschwiler, T.: 3d-ice: Fast compact transient thermal modeling for 3D-ICs with inter-tier liquid cooling. In: International Conference on Computer-Aided Design, ICCAD 2010 (2010)
32. Huang, W., Stan, M.R., Skadron, K., Sankaranarayanan, K., Ghosh, S., Velusam, S.: Compact thermal modeling for temperature-aware design. In: Proceedings of the 41st Annual Design Automation Conference, DAC 2004, pp. 878–883. ACM, New York (2004)
33. Velusamy, S., Huang, W., Lach, J., Stan, M.R., Skadron, K.: Monitoring temperature in fpga based socs. In: ICCD, pp. 634–640 (2005)
34. Sridhar, A., Vincenzi, A., Ruggiero, M., Brunschwiler, T., Atienza Alonso, D.: Compact transient thermal model for 3D ICs with liquid cooling via enhanced heat transfer cavity geometries. In: Proceedings of the 16th International Workshop on Thermal Investigations of ICs and Systems (THERMINIC 2010), vol. 1(1), pp. 105–110. IEEE Press, New York (2010)
35. Hung, W.-L., Link, G.M., Xie, Y., Vijaykrishnan, N., Irwin, M.J.: Interconnect and thermal-aware floorplanning for 3d microprocessors. In: ISQED, pp. 98–104 (2006)
36. Sharifi, S., Rosing, T.V.: Accurate direct and indirect on-chip temperature sensing for efficient dynamic thermal management. Trans. Comp.-Aided Des. Integ. Cir. Sys. 29, 1586–1599 (2010)

37. Cochran, R., Reda, S.: Spectral techniques for high-resolution thermal characterization with limited sensor data. In: DAC, pp. 478–483 (2009)
38. Coskun, A.K., Rosing, T.V., Gross, K.C.: Utilizing Predictors for Efficient Thermal Management in Multiprocessor SoCs. IEEE Transactions on CAD 28, 1503–1516 (2009)
39. Yeo, I., Liu, C.C., Kim, E.J.: Predictive dynamic thermal management for multicore systems. In: DAC, pp. 734–739 (June 2008)
40. Coskun, A.K., Rosing, T., Gross, K.: Proactive Temperature Balancing for Low-Cost Thermal Management in MPSoCs. In: International Conference on Computer-Aided Design (ICCAD), pp. 250–257 (2008)
41. Cochran, R., Reda, S.: Consistent runtime thermal prediction and control through workload phase detection. In: Design Automation Conference, DAC (2010)
42. Yeo, I., Liu, C.C., Kim, E.J.: Predictive dynamic thermal management for multicore systems. In: Proceedings of the 45th Annual Design Automation Conference, DAC 2008, pp. 734–739. ACM, New York (2008), http://doi.acm.org/10.1145/1391469.1391658
43. Coskun, A., Rosing, T., Gross, K.: Utilizing predictors for efficient thermal management in multiprocessor socs. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 28(10), 1503–1516 (2009)
44. Ge, Y., Malani, P., Qiu, Q.: Distributed task migration for thermal management in many-core systems. In: 2010 47th ACM/IEEE on Design Automation Conference (DAC), pp. 579–584 (June 2010)
45. Bartolini, A., Cacciari, M., Tilli, A., Benini, L.: A distributed and self-calibrating model-predictive controller for energy and thermal management of high-performance multicores. In: Design, Automation Test in Europe Conference Exhibition (DATE), pp. 1–6 (March 2011)
46. Ebi, T., Faruque, M., Henkel, J.: Tape: Thermal-aware agent-based power econom multi/many-core architectures. In: IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers, ICCAD 2009, pp. 302–309 (November 2009)
47. Ge, Y., Qiu, Q.: Dynamic thermal management for multimedia applications using machine learning. In: 2011 48th ACM/EDAC/IEEE on Design Automation Conference (DAC), pp. 95–100 (June 2011)
48. Coskun, A., Rosing, T., Gross, K.: Temperature management in multiprocessor socs using online learning. In: 45th ACM/IEEE on Design Automation Conference, DAC 2008, pp. 890–893 (June 2008)
49. Pakbaznia, E., Ghasemazar, M., Pedram, M.: Temperature-aware dynamic resource provisioning in a power-optimized datacenter. In: Design, Automation Test in Europe Conference Exhibition (DATE), pp. 124–129 (March 2010)
50. JEDEC, 3D IC Standards (2011), http://www.jedec.org/standards-documents/technology-focus-areas/3d-ics

Chapter 7
Sustainable and Reliable On-Chip Wireless Communication Infrastructure for Massive Multi-core Systems

Amlan Ganguly, Partha Pande, Benjamin Belzer, and Alireza Nojeh

Abstract. The Network-on-Chip paradigm has emerged as an enabling methodology to integrate high number of functional cores on a single die. However, the metal interconnect based multi-hop on-chip networks result in high latency and energy dissipation in data transfer. In order to alleviate these problems several emerging interconnect technologies have been proposed. Wireless NoC architectures are shown to outperform the wired counterparts by several orders of magnitude in energy dissipation while achieving higher data transfer rates. However, reliability of the wireless links along with the metal NoC interconnects are known to be a major concern in the future technology nodes. Powerful error control codes based on product codes can enhance the resilience of the wireless channels making the overall NoC more reliable. A unified error control mechanism to enhance the resilience to transient errors of the wireless links as well as the wireline links is presented. This chapter showcases the achievable performance benefits of the wireless NoC architectures while still maintaining acceptable robustness to transient errors.

Amlan Ganguly
Department of Computer Engineering Rochester Institute of Technology, USA
e-mail: amlan.ganguly@rit.edu

Partha Pande
School Of Electrical Engineering and Computer Science Washington State University, USA
e-mail: pande@eecs.wsu.edu

Benjamin Belzer
School Of Electrical Engineering and Computer Science Washington State University, USA
e-mail: pande@eecs.wsu.edu

Alireza Nojeh
Department of Electrical and Computer Engineering, University of British Columbia, Canada
e-mail: anojeh@ece.ubc.ca

S.U. Khan et al. (Eds.): Evolutionary Based Solutions for Green Computing, SCI 432, pp. 187–225.
springerlink.com

7.1 Introduction

Design of multi-core integrated systems beyond the current CMOS era will present unprecedented advantages and challenges, the former being related to very high device densities and the latter to soaring power dissipation issues. According to the International Technology Roadmap for Semiconductors (ITRS) in 2007, the contribution of interconnects to chip power dissipation is expected to increase from 51% in the $0.13\mu m$ technology generation to up to 80% in the next five year period. This clearly indicates the future design challenges associated with traditional scaling of conventional metal interconnects and material innovation. To enhance the performance of conventional metal interconnect-based multi-core chips, a few radically different interconnect technologies are being currently explored such as 3D integration [1], photonic interconnects [2] and multi-band RF [3] or wireless interconnects [4].

All these new technologies have been predicted to be capable of enabling multi-core designs, which improve the speed and power dissipation in data transfer significantly. Innovative architecture design coupled with these emerging interconnect technologies achieve high performance in on-chip data transfer while still dissipating orders of magnitude less energy compared to traditional metal interconnect based NoCs [5]. However, these alternative interconnect paradigms are in their formative stages and need to overcome significant challenges pertaining to reliability. Error Control Coding (ECC) has emerged as a viable solution to increase the reliability of on-chip communication links [6]. It has already been demonstrated that specially designed ECCs increase reliability as well as lower energy dissipation of traditional NoC links [7].

In this chapter we present a nature inspired methodology to design efficient NoC architectures with on-chip wireless links (WiNoC) and address reliability issues in such a hybrid wireless NoC [8] with a unified ECC framework with different schemes for wireline and wireless links. Wireless links in the hybrid NoC encounter higher error rates than regular wires and hence stronger ECC schemes are required to restore the reliability on such links. In this chapter we determine the typical error rates of the on-chip wireless links and present suitable ECC mechanisms proposed in [9] to improve their reliability. It is seen that using the proposed unified ECC scheme it is possible to achieve significantly low energy dissipation and high bandwidth in a wireless NoC while still maintaining similar reliability to that of a regular wireline counterpart.

7.2 Related Work

Conventional NoCs use multi-hop packet switched communication. To improve performance, the concept of express virtual channels is introduced in [10]. It is shown that by using virtual express lanes to connect distant cores in the network, it is

possible to avoid the router overhead at intermediate nodes, and thereby greatly improve NoC performance. NoCs have been shown to perform better by inserting long range wired links following principles of small world graphs [11].

The design principles of three-dimensional NoCs and photonic NoCs are elaborated in various recent publications [1,2]. It is estimated that 3D and photonic NoCs will dissipate significantly less power than their electronic counterpart. Another alternative for low power is NoCs with multi-band RF interconnects [3].

Recently, the design of a wireless NoC based on CMOS Ultra Wideband (UWB) technology was proposed [4]. In [12] the feasibility of designing on-chip wireless communication network with miniature antennas and simple transceivers that operate at the sub-THz range of 100-500 GHz has been demonstrated.

If the transmission frequencies can be increased to THz range then the corresponding antenna sizes decrease, occupying much less chip real estate. One possibility is to use nanoscale antennas based on CNTs operating in the THz frequency range [13]. Consequently building an on-chip wireless interconnection network using THz frequencies for inter-core communications becomes feasible. Design of a wireless NoC operating in the THz frequency range using CNT antennas is elaborated in [8]. All these emerging interconnect technologies are shown to improve the performance and power dissipation of NoCs. However due to the fact that these technologies are still in their formative stages their fabrication and integration with standard CMOS processes are unreliable. Moreover, system level techniques for sustaining gains in performance in presence of these inherently unreliable technologies have not received much attention from researchers. Hence, there is a need to explore NoC architectures that will sustain the advantages of such emerging technologies even in the presence of inherent unreliability.

In particular we present the design of a wireless NoC using THz wireless links with CNT based antennas which will enable high gains in performance and power dissipation despite high rates of failures of the wireless links.

Signal transmission in the current and future technology nodes presents unprecedented challenges due to several events. In the Ultra-Deep Submicron (UDSM) technologies crosstalk coupling between adjacent wires as well as several transient error mechanisms like ground bounce and alpha particles result in increased probability of multi-bit errors. ECC schemes have been explored in the context of NoC links to enhance the reliability of data transmission and reduce energy dissipation.

In [6] the authors presented an unified framework for applying ECC to SoC links, The authors of [14, 15] explore error recovery mechanisms suitable for NoC platforms. In [16–18], the authors proposed several joint crosstalk avoidance and single error correction codes based on simple parity and duplication mechanisms. The authors of [7] demonstrated the role of multiple error correction and crosstalk avoidance codes. The advantages of using simple [9] Hamming code based Product codes to improve the reliability of NoC links were explored in [19]. Even though several research groups have proposed wireless links to create high performance NoC

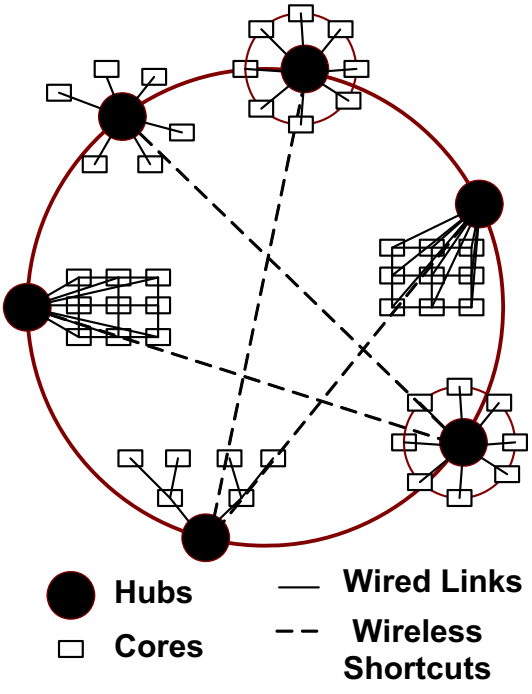


Fig. 7.1. A conceptual hybrid (wireless/wired) NoC architecture with heterogeneous subnets [9]

architectures [4, 8], the aspect of reliability of such wireless links has not been adequately addressed. An attempt to address reliability issues in on-chip wireless links was made in [9] using a unified ECC scheme for the hybrid.

In this chapter we will first present the methodologies that can be used to design wireless NoCs in section 7.3 and then how ECCs can be used to enhance the reliability of the WiNoCs in section 7.4.

7.3 Wireless NoC Architecture

In a generic wired NoC the constituent embedded cores communicate via multiple switches and wired links. This multi-hop communication results in data transfers with high energy dissipation and latency. To alleviate this problem we propose long-distance high bandwidth wireless links between distant cores in the chip. In the following subsections we will explain the design of a scalable architecture for WiNoCs of various system sizes.

7.3.1 Topology

Modern complex network theory [20] provides us with a powerful method to analyze network topologies and their properties. Between a regular, locally interconnected mesh network and a completely random Erdös-Rényi topology, there are other classes of graphs [20], such as small-world and scale-free graphs. Networks with the small-world property have a very short average path length, which is commonly measured as the number of hops between any pair of nodes. The average shortest path length of small-world graphs is bounded by a polynomial in $log(N)$, where N is the number of nodes, which makes them particularly interesting for efficient communication with minimal resources [21,22]. This feature of small-world graphs makes them particularly attractive for constructing scalable WiNoCs.

Most complex networks, such as social networks, the Internet, as well as certain parts of the brain exhibit the small-world property. A small-world topology can be constructed from a locally connected network by re-wiring connections randomly to any other node, which creates short cuts in the network [23]. These random long-range links between nodes can also be established following probability distributions depending on the distance separating the nodes [20]. It has been shown that such "shortcuts"in NoCs can significantly improve the performance compared to locally interconnected mesh-like networks [7,11] with fewer resources than a fully connected system.

Our goal here is to use the "small-world" approach to build a highly efficient NoC based on both wired and wireless links. Thus, for our purpose, we first divide the whole system into multiple small clusters of neighboring cores and call these smaller networks subnets. As subnets are smaller networks, intra-subnet communication will have a shorter average path length than a single NoC spanning the whole system. Figure 7.1 shows a WiNoC architecture with several subnets with mesh, star, ring and tree topologies. The subnets have NoC switches and links as in a standard NoC. The cores are connected to a centrally located hub through direct links and the hubs from all subnets are connected in a 2nd level network forming a hierarchical network. This upper level of the hierarchy is designed to have characteristics of small-world graphs.

Due to a limited number of possible wireless links, as discussed in later subsections, neighboring hubs are connected by traditional wired links forming a ring and a few wireless links are distributed between hubs separated by relatively long distances. Reducing long-distance multi-hop wired communication is essential in order to achieve the full benefit of on-chip wireless networks for multi-core systems. As the links are initially established probabilistically, the network performance might not be optimal. Hence, after the initial placement of the wireless links the network is further optimized for performance by using *Simulated Annealing (SA)* [24]. The particular probability distribution and the heuristics followed in establishing the network links are described in the next subsection. Key to our approach is establishing optimal overall network topology under given resource constraints, i.e., a limited number of wireless links. Figure 7.1 shows a possible interconnection topology. Instead of the ring used in this example, the hubs can be connected in any other

possible interconnect architecture. The size and number of subnets are chosen such that neither the subnets nor the upper level of the hierarchy become too large. This is because if either level of the hierarchy becomes too large then it causes a performance bottleneck by limiting the data throughput in that level. However, since the architecture of the two levels can be different causing their traffic characteristics to differ from each other, the exact hierarchical division can be obtained by performing system level simulations as shown in section 7.5.3.

We propose a hybrid wired/wireless NoC architecture. The hubs are interconnected via both wireless and wired links while the subnets are wired only. The hubs with wireless links are equipped with *wireless base stations (WBs)* that transmit and receive data packets over the wireless channels. When a packet needs to be sent to a core in a different subnet it travels from the source to its respective hub and reaches the hub of the destination subnet via the small-world network consisting of both wireless and wired links, where it is then routed to the final destination core. For inter-subnet and intra-subnet data transmission, wormhole routing is adopted. Data packets are broken down into smaller parts called flow control units or flits [25]. The header flit holds the routing and control information. It establishes a path, and subsequent payload or body flits follow that path. The routing protocol is described in section 7.3.4.

7.3.2 Wireless Link Insertion and Optimization

As mentioned above, the overall interconnect infrastructure of the WiNoC is formed by connecting the cores in the sub-nets with each other and to the central hub through traditional metal wires. The hubs are then connected by wires and wireless links such that the 2nd level of the network has the small-world property. The placement of the wireless links between a particular pair of source and destination hubs is important as this is responsible for establishing high-speed, low-energy interconnects on the network, which will eventually result in performance gains. Initially the links are placed probabilistically; i.e., between each pair of source and destination hubs, i and j respectively, the probability P_{ij} of having a wireless link is proportional to the distance measured in number of hops along the ring, h_{ij}, as shown in Equation (7.1).

$$P_{ij} = \frac{h_{ij}}{\sum_{i,j} h_{ij}}. \tag{7.1}$$

The probabilities are normalized such that their sum is equal to one. Such a distribution is chosen because in the presence of a wireless link, the distance between the pair becomes a single hop and hence it reduces the original distance between the communicating hubs through the ring. Depending on the number of available wireless links, they are inserted between randomly chosen pairs of hubs, which are chosen following the probability distribution mentioned above.

Once the network is initialized, an optimization by means of SA heuristics is performed. Since the subnet architectures are independent of the top level network, the optimization can be done only on the top level network of hubs and hence the sub-nets can be decoupled from this step. The optimization step is necessary as the random initialization might not produce the optimal network topology. SA offers a simple, well established and scalable approach for the optimization process as opposed to a brute force search.

If there are N hubs in the network and n wireless links to distribute, the size of the search space S is given by:

$$|S| = \left(\binom{N}{2} - N \right) \tag{7.2}$$
$$n$$

Thus, with increasing N, it becomes increasingly difficult to find the best solution by exhaustive search. In order to perform SA, a metric has been established, which is closely related to the connectivity of the network. The metric to be optimized is the average distance, measured in number of hops, between all source and destination hubs. To compute this metric the shortest distances between all hub pairs are computed following the routing strategy outlined in section 3.4. In each iteration of the SA process, a new network is created by randomly rewiring a wireless link in the current network. The metric for this new network is calculated and compared to the metric of the current network. The new network is always chosen as the current optimal solution if the metric is lower. However, even if the metric is higher we choose the new network probabilistically. This reduces the probability of getting stuck in a local optimum, which could happen if the SA process were to never choose a worse solution. The exponential probability shown in Equation (7.3) is used to determine whether or not a worse solution is chosen as the current optimal:

$$P(h,h',T) = exp[(h-h')/T]. \tag{7.3}$$

The optimization metrics for the current and new networks are h and h' respectively. T is a temperature parameter, which decreases with the number of optimization iterations according to an annealing schedule. Here we have used Cauchy scheduling, where the temperature varies inversely with the number of iterations [24]. The algorithm used to optimize the network is shown in Figure 7.2.

Here we assume a uniform spatial traffic distribution where a packet originating from any core is equally likely to have any other core on the die as its destination. However, with other kinds of spatial traffic distributions, where the network loads are localized in different clusters, the metric for optimization has to be changed accordingly to account for the non-uniform traffic patterns as discussed later in section 7.4.3.

An important component in the design of the WiNoCs is the on-chip antenna for the wireless links. In the next section we describe various alternative on-chip antenna choices and their pros and cons.

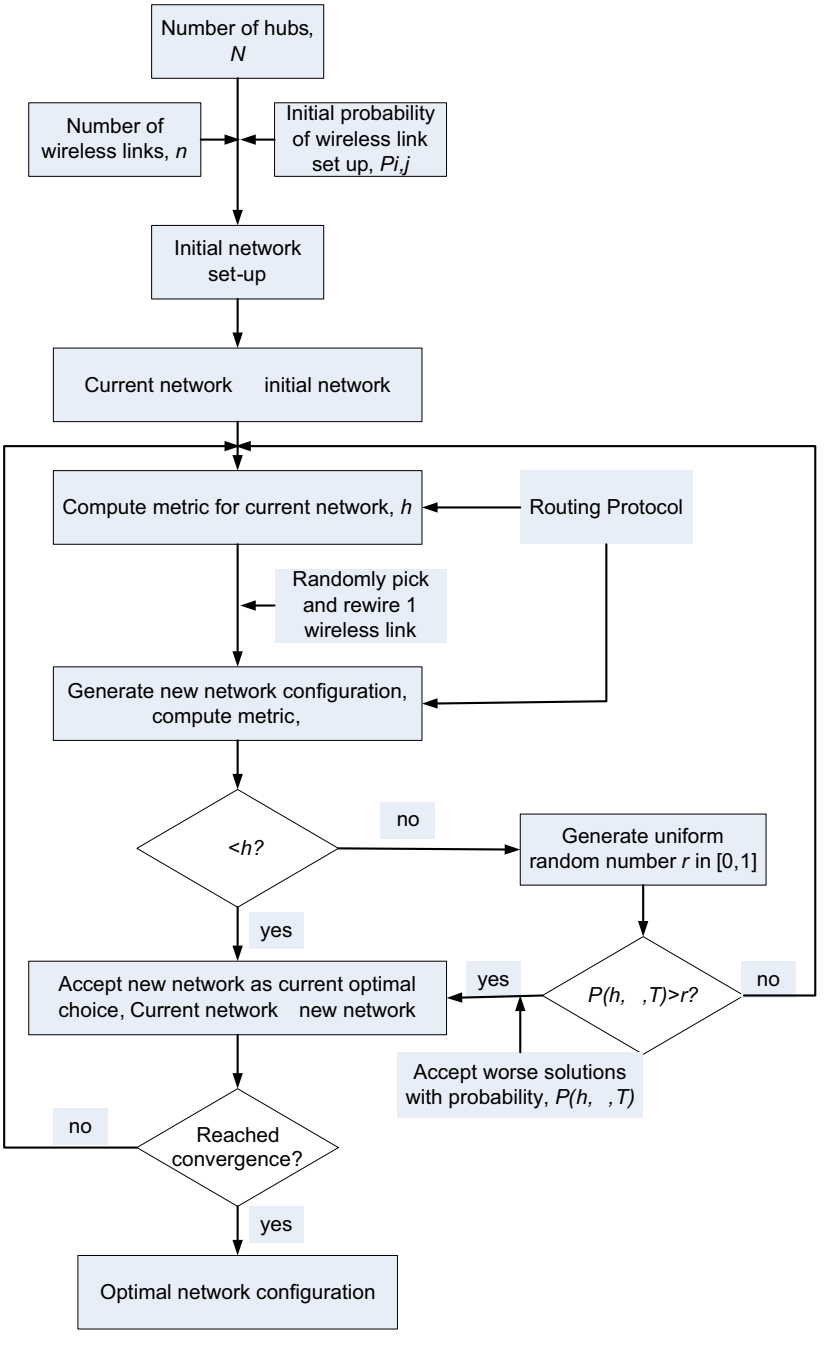


Fig. 7.2. Flow diagram for the simulated annealing based optimization of WiNoC architectures [8]

7.3.3 On-Chip Antennas

Suitable on-chip antennas are necessary to establish wireless links for WiNoCs. In [26] the authors demonstrated the performance of silicon integrated on-chip antennas for intra- and inter-chip communication. They have primarily used metal zig-zag antennas operating in the range of tens of GHz. Design of an *ultra wideband (UWB)* antenna for inter- and intra-chip communication is elaborated in [27]. This particular antenna was used in the design of a wireless NoC [4] mentioned earlier in section 7.2. The above mentioned antennas principally operate in the millimeter wave (tens of GHz) range and consequently their sizes are on the order of a few millimeters.

If the transmission frequencies can be increased to THz/optical range then the corresponding antenna sizes decrease, occupying much less chip real estate. Characteristics of metal antennas operating in the optical and near-infrared region of the spectrum of up to 750 THz have been studied [28].

Antenna characteristics of carbon nanotubes (CNTs) in the THz/optical frequency range have also been investigated both theoretically and experimentally [13, 29]. Bundles of CNTs are predicted to enhance performance of antenna modules by up to 40dB in radiation efficiency and provide excellent directional properties in far-field patterns [30]. Moreover these antennas can achieve a bandwidth of around 500 GHz, whereas the antennas operating in the millimeter wave range achieve bandwidths of tens of GHz [30]. Thus, antennas operating in the THz/optical frequency range can support much higher data rates. CNTs have numerous characteristics that make them suitable as on-chip antenna elements for optical frequencies. Given wavelengths of hundreds of nanometers to several micrometers, there is a need for virtually one-dimensional antenna structures for efficient transmission and reception. With diameters of a few nanometers and any length up to a few millimeters possible, CNTs are the perfect candidate. Such thin structures are almost impossible to achieve with traditional microfabrication techniques for metals. Virtually defect-free CNT structures do not suffer from power loss due to surface roughness and edge imperfections found in traditional metallic antennas. In CNTs, ballistic electron transport leads to quantum conductance, resulting in reduced resistive loss, which allows extremely high current densities in CNTs, namely 4-5 orders of magnitude higher than copper. This enables high transmitted powers from nanotube antennas, which is crucial for long-range communications. By shining an external laser source on the CNT, radiation characteristics of multi-walled carbon nanotube (MWCNT) antennas are observed to be in excellent quantitative agreement with traditional radio antenna theory [13], although at much higher frequencies of hundreds of THz. Using various lengths of the antenna elements corresponding to different multiples of the wavelengths of the external lasers, scattering and radiation patterns are shown to be improved. Such nanotube antennas are good candidates for establishing on-chip wireless communications links and are henceforth considered in this chapter.

Chemical vapor deposition (CVD) is the traditional method for growing nanotubes in specific locations by using lithographically patterned catalyst islands. The

application of an electric field during growth or the direction of gas flow during CVD can help align nanotubes. However, the high-temperature CVD could potentially damage some of the preexisting CMOS layers. To alleviate this, localized heaters in the CMOS fabrication process to enable localized CVD of nanotubes without exposing the entire chip to high temperatures are used [31].

As mentioned above, the NoC is divided into multiple subnets. Hence, the WBs in the subnets need to be equipped with transmitting and receiving antennas, which will be excited using external laser sources. As mentioned in [2], the laser sources can be located off-chip or bonded to the silicon die. Hence their power dissipation does not contribute to the chip power density. The requirements of using external sources to excite the antennas can be eliminated if the electroluminescence phenomenon from a CNT is utilized to design linearly polarized dipole radiation sources [32]. But further investigation is necessary to establish such devices as successful transceivers for on-chip wireless communication.

To achieve line of sight communication between WBs using CNT antennas at optical frequencies, the chip packaging material has to be elevated from the substrate surface to create a vacuum for transmission of the high frequency EM waves. Techniques for creating such vacuum packaging are already utilized for MEMS applications [33], and can be adopted to make creation of line of sight communication between CNT antennas viable. In classical antenna theory it is known that the received power degrades inversely with the 4th power of the separation between source and destination due to ground reflections beyond a certain distance. This threshold separation, r0 between source and destination antennas assuming a perfectly reflecting surface, is given by Equation (7.4:

$$r_0 = \frac{2 \cdot \Pi \cdot H^2}{\lambda}. \tag{7.4}$$

Here H is the height of the antenna above the reflecting surface and λ is the wavelength of the carrier. Thus, if the antenna elements are at a distance of H from the reflective surfaces like the packaging walls and the top of the die substrate, the received power degrades inversely with the square of the distance until it is r_0. Thus H can be adjusted to make the maximum possible separation smaller than the threshold separation r_0 for a particular frequency of radiation used. Considering the optical frequency ranges of CNT antennas, depending on the separation between the source and destination pairs in a single chip, the required elevation is a few tens of microns only.

7.3.4 Routing and Communication Protocols

In the proposed WiNoC, intra-subnet data routing depends on the topology of the subnets. For example, if the cores within a subnet are connected in a mesh architecture, then data routing within the subnet follows dimension order (e-cube) routing. Inter-subnet data is routed through the hubs, along the shortest path between the

source and destination subnets in terms of number of links traversed. The hubs in all the subnets are equipped with a pre-routing block to determine this path through a search across all potential paths between the hubs of the source and destination subnets. In the current work, paths involving only a single wireless link and none or any number of wired links on the ring are considered. All such paths as well as the completely wired path on the ring are compared and the one with the minimum number of link traversals is chosen for data transfer. For a data packet requiring inter-subnet routing, this computation is done only once for the header flit at the hub of the originating subnet. The header flit needs to have a field containing the address of the intermediate hub with a WB that will be used in the path. Only this information is sufficient as the header follows the default wireline path along the ring to that hub with the WB from its source, which is also the shortest path along the ring. Since each WB has a single, unique destination, the header reaches that destination and is then again routed via the wireline path to its final destination hub using normal ring routing. The rest of the flits follow the header as is usually the case with wormhole routing.

An alternative routing approach is to avoid the one-time evaluation of the shortest path at the original source hub and adopt a distributed routing mechanism. In this scenario, the path is determined at each node by checking for the existence of a wireless link at that node, which if taken will shorten the path length to the final destination. If this wireless link does not exist or shorten the path in comparison to the wireline path from that node, then the default routing mechanism along the ring is followed to the next node. This mechanism performs a check at every node by computing and comparing the path lengths by using the default wireline routing or the wireless link. The adopted centralized routing performs all the checks at the original source hub, which includes all the wireless links and the wireline path from the source to the destination.

By using multiband laser sources to excite CNT antennas, different frequency channels can be assigned to pairs of communicating subnets. This will require using antenna elements tuned to different frequencies for each pair, thus creating a form of *frequency division multiplexing (FDM)*. This is possible by using CNTs of different lengths, which are multiples of the wavelengths of the respective carrier frequencies. High directional gains of these antennas, demonstrated in [13, 30], aid in creating directed channels between source and destination pairs. In [34], 24 continuous wave laser sources of different frequencies are used. Thus, these 24 different frequencies can be assigned to multiple wireless links in the WiNoC in such a way that a single frequency channel is used only once to avoid signal interference on the same frequencies. This enables concurrent use of multi-band channels over the chip. The number of wireless links in the network can therefore vary from 24 links, each with a single frequency channel, to a single link with all 24 channels. Assigning multiple channels per link increases the link bandwidth. Currently, high-speed silicon integrated Mach-Zehnder optical modulators and demodulators, which convert electrical signals to optical signals and vice versa are commercially available [35]. The optical modulators can provide 10 Gbps data rate per channel on these links. At the receiver a low noise amplifier (LNA) can be used to boost the power of the received

electrical signal, which will then be routed into the destination subnet. As noted in [34], this data rate is expected to increase manifold with future technology scaling. The modulation scheme adopted is non-coherent *on-off keying (OOK)*, and therefore does not require complex clock recovery and synchronization circuits. Due to limitations in the number of distinct frequency channels that can be created through the CNT antennas, the flit width in NoCs is generally higher than the number of possible channels per link. Thus, to send a whole flit through the wireless link using a limited number of distinct frequencies, a proper channelization scheme needs to be adopted. Here we assume a flit width of 32 bits. Hence, to send the whole flit using the distinct frequency channels, time division multiplexing (TDM) is adopted. The various components of the wireless channel viz., the electro-optic modulators, the TDM modulator/demodulator, the LNA and the router for routing data on the network of hubs are implemented as a part of the WB. Figure 7.3 illustrates the adopted communication mechanism for the inter-subnet data transfer.

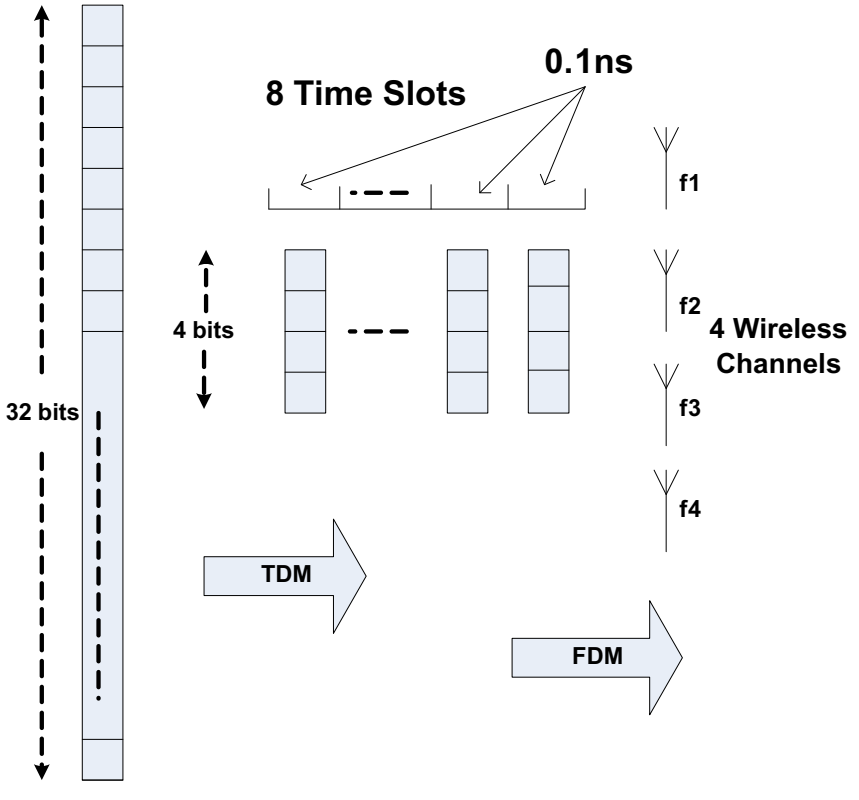


Fig. 7.3. Adopted communication protocol for the wireless channel [8]

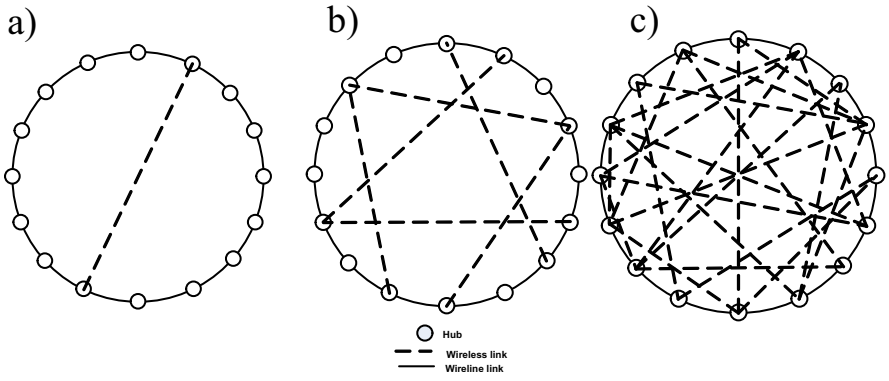


Fig. 7.4. The optimal wireless link arrangement for (a) 1, (b) 6, and (c) 24 wireless links among 16 hubs. Note that symmetrical solutions with the same performance are possible [8]

In this WiNoC example, we use a wireless link with 4 frequency channels. In this case, one flit is divided into 8 four bit nibbles, and each nibble is assigned a 0.1ns timeslot, corresponding to a bit rate of 10 Gbps. The bits in each nibble are transmitted simultaneously over four different carrier frequencies.

The routing mechanism discussed in this section is easily extendable to incorporate other addressing techniques like multicasting. Performance of traditional NoC architectures incorporating multicasting have been already investigated [36] and it can be similarly used to enhance the performance of the WiNoC developed. For example, let us consider a subnet in a 16-subnet system (as in Figure 7.4), which tries to send packets to 3 other subnets such that one of them is diagonally opposite to the source subnet and the other two are on either side of it. In absence of long-range wireless links, using multicasting the zero load latency for the delivery of a single flit is 9 cycles whereas without multicasting the same flit will need 11 cycles to be delivered to the respective destinations. Here the communication takes place only along the ring. However, if a wireless link exists along the diagonal from the source to the middle destination subnet then with multicasting the flit can be transferred in 5 cycles if there are 8 distinct channels in the link. Four cycles are needed to transfer a 32-bit flit to the diagonally opposite hub via the wireless links and one more hop along the ring to the final destinations on either side. The efficiency of using multicasting varies with number of channels in the link as it governs the bandwidth of the wireless link.

7.4 Performance Evaluations

In this section we analyze the characteristics of the proposed WiNoC architectures and study trends in their performance with scaling of system size. For our analysis, we have considered three different system sizes, namely 128, 256, and 512 cores

on a die of size $20mm \times 20mm$. We observe results of scaling up the system size by increasing both the number of subnets as well as the number of cores per subnet. Hence, in one scenario, we have considered a fixed number of cores per subnet to be 16 and varied the number of subnets between 8, 16, and 32. In the other case, we have kept the number of subnets fixed at 16 and varied the size of the subnets from 8 to 32 cores. These system configurations are chosen based on the experiments explained later in section 7.4.3. Establishment of wireless links using simulated annealing, however, depends only on the number of hubs on the 2nd level of the network.

7.4.1 Establishment of Wireless Links

Initially the hubs are connected in a ring through normal wires and the wireless links are established between randomly chosen hubs following the probability distribution given by Equation (7.1). We then use simulated annealing to achieve an optimal configuration by finding the positions of the wireless links which minimize the average distance between all source and destination pairs in the network. Figure 7.4 shows the location of 1, 6 and 24 wireless links with 24, 4 and 1 channels respectively in a network of 16 hubs. We followed the same optimization methodology for all the other networks. The corresponding average distances for the optimized networks with different system sizes are shown in Table 7.1.

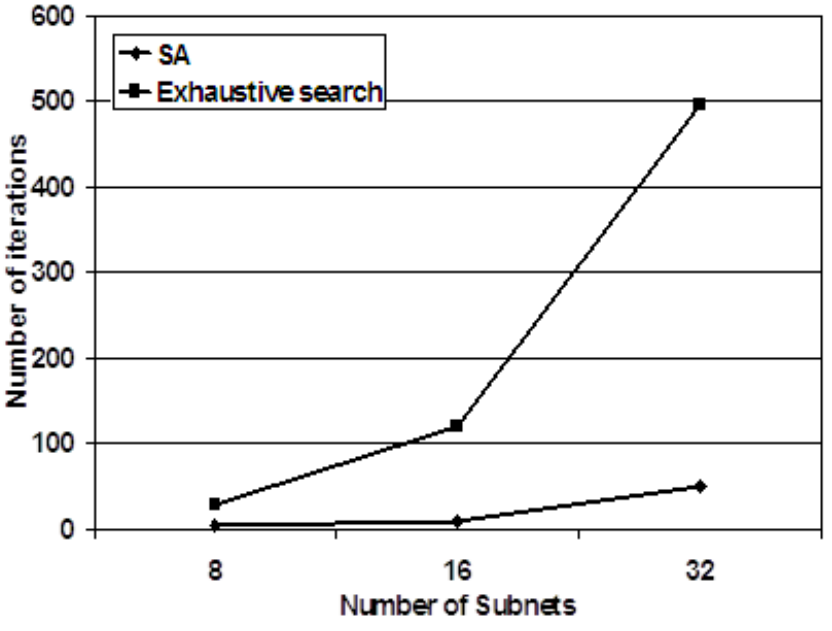
Table 7.1. Average distance for optimized WiNoCs [8]

No. of sub-nets (N)	Avg distance (hops)		
	1 Wireless Link	6 Wireless Links	24 Wireless Links
8	1.7188	1.3125	1.1250 (12 Links)*
16	3.2891	2.1875	1.5625
32	6.3301	3.8789	2.6309

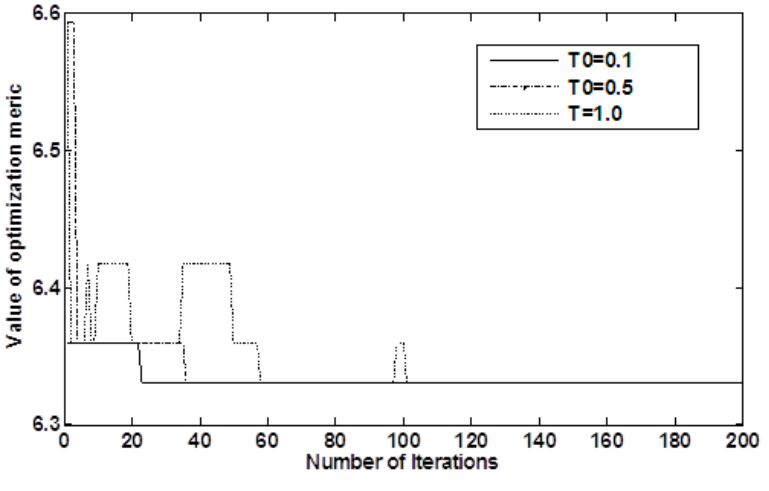
*In case of 8 subnets only 12 wireless links are used with 2 channels per link.

It should be noted that the particular placement of wireless links to obtain the optimal network configuration is not unique because of symmetric considerations in our setup, i.e., there are multiple configurations with the same optimal performance.

In order to establish the performance of the SA algorithm used, we compared the resultant optimization metric with the metric obtained through exhaustive search for the optimized network configuration for various system sizes. The SA algorithm produces network configurations with total average hop count exactly equal to that generated by the exhaustive search technique for the system configurations considered in this chapter. However, the obtained WiNoC configuration in terms of topology is non-unique as different configurations can have the same average hop count. Figure 7.5(a) shows the number of iterations required to arrive at the optimal solution with SA and exhaustive search algorithms. Clearly the SA algorithm converges



(a)



(b)

Fig. 7.5. (a) Number of iterations required to reach optimal solution by the SA and exhaustive search methods, (b) Convergence with different temperatures [8]

to the optimal configuration much faster than the exhaustive search technique. This advantage will increase for larger system sizes. Figure 7.5(b) shows the convergence of the metric for different values of the initial temperature to illustrate that the SA approach converges robustly to the optimal value of the average hopcount with numerical variation in the temperature. This simulation was performed for a system with 32 subnets with 1 wireless link. With higher values of the initial temperature it can take longer to converge. Naturally, for large enough values of the initial temperature, the metric does not converge. On the other hand, lower values of the initial temperature make the system converge faster but at the risk of getting stuck in a local optimum. Using the network configurations developed in this subsection, we will now evaluate the performance of the WiNoC based on well-established performance metrics.

7.4.2 Performance Metrics

To characterize the performance of the proposed WiNoC architectures, we consider three network parameters: latency, throughput, and energy dissipation. Latency refers to the number of clock cycles between the injection of a message header flit at the source node and the reception of the tail flit at the destination. Throughput is defined as the average number of flits successfully received per embedded core per clock cycle. Energy dissipation per packet is the average energy dissipated by a single packet when routed from the source to destination node; both the wired subnets and the wireless channels contribute to this. For the subnets, the sources of energy dissipation are the inter-switch wires and the switch blocks. For the wireless channels, the main contribution comes from the WBs, which include antennas, transceiver circuits and other communication modules like the TDM block and the LNA. Energy dissipation per packet, E_{pkt}, can be calculated according to Equation (7.5) below.

$$E_{pkt} = \frac{N_{intra-subnet} \cdot E_{subnet,hop} \cdot h_{subnet} + N_{inter-subnet} \cdot E_{sw} \cdot h_{sw}}{N_{intra-subnet} + N_{inter-subnet}} \tag{7.5}$$

In Equation (7.5), $N_{intrasubnet}$ and $N_{intersubnet}$ are the total number of packets routed within the subnet and between subnets respectively. $E_{subnet,hop}$ is the energy dissipated by a packet traversing a single hop on the wired subnet including a wired link and switch, and E_{sw} is the energy dissipated by a packet traversing a single hop on the 2nd network level of the WiNoC, which has the small-world properties. E_{sw} also includes the energy dissipation in the core to hub links. In Equation (7.5), h_{subnet} and h_{sw} are the average number of hops per packet in the subnet and the small-world network.

7.4.3 Performance Evaluation

The network architectures developed earlier in this section are simulated using a cycle accurate simulator which models the progress of data flits accurately per clock cycle accounting for flits that reach destination as well as those that are dropped. One hundred thousand iterations were performed to reach stable results in each case, eliminating the effect of transients in the first few thousand cycles.

The mesh subnet architecture is considered. The width of all wired links is considered to be same as the flit size, which is 32 here. The particular NoC switch architecture, adopted from [37] for the switches in the subnets, has three functional stages, namely, input arbitration, routing/switch traversal, and output arbitration. The input and output ports including the ones on the wireless links have four virtual channels per port, each having a buffer depth of 2 flits [37]. Each packet consists of 64 flits. Similar to the intra-subnet communication, we have adopted wormhole routing in the wireless channel too. Consequently, the hubs have similar architectures as the NoC switches in the subnets. Hence, each port of the hub has same input and output arbiters, and equal number of virtual channels with same buffer depths as the subnet switches. The number of ports in a hub depends on the number of links connected to it. The wireless ports of the WBs are assumed to be equipped with antennas, TDM modules, and electro-optic modulators and demodulators. The various components of a WB are shown in Figure 7.6.

A hub consisting of only ports to wired links is also highlighted in the figure to emphasize that a WB has additional components compared to a hub. The pipelined NoC switches and hubs, and the wired links are driven with a clock of frequency 2.5 GHz.

Figure 7.7 shows throughput and latency plots as a function of injection load for a system with 256 cores divided into 16 subnets, each with 16 cores. The delays incurred by the wired links from the cores to the hub for varying number of cores in the subnets for different system sizes are shown in Table 7.2.

Table 7.2. Delays on wired links in the WiNoCs [8]

System Size	No. of subnets	Subnet size	Core-hub link delay (ps)	Inter-hub link delay (ps)
128	8	16	96	181/86*
	16	8	60	86
256	16	16	60	86
	16	32	60	86
512	32	16	48	86/43*

*for 8 and 32 subnets the inter-subnet distances are different along the two planar directions.

The delays in the inter-hub wires for varying number of subnets are also shown. As can be seen these delays are all less than the clock period of 400ps and it may be noted that the lengths of both core-to-hub and inter-hub wireline links will reduce with increase in the number of subnets as then each subnet becomes smaller in area and the subnets also come closer to each other. The delays incurred by the

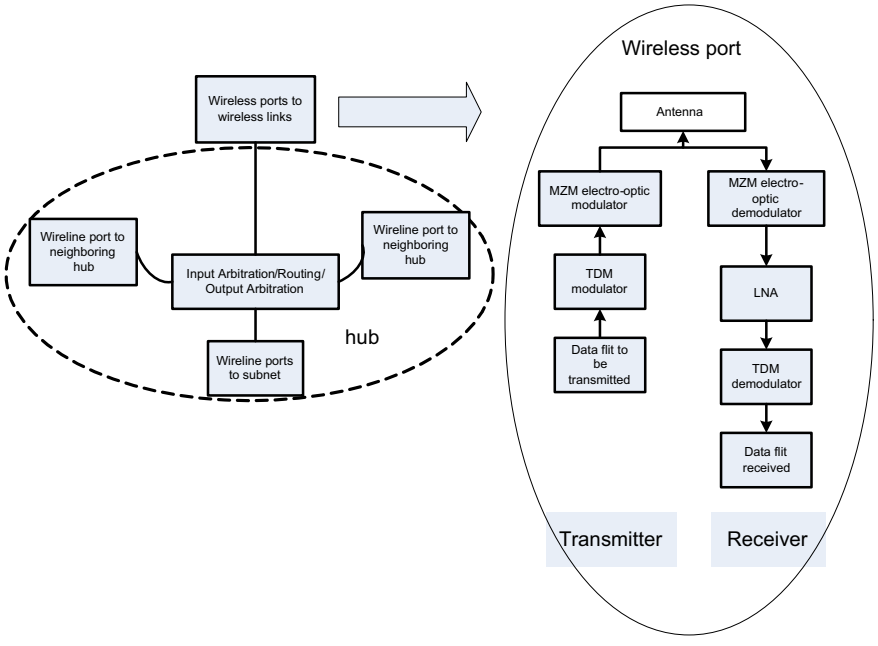
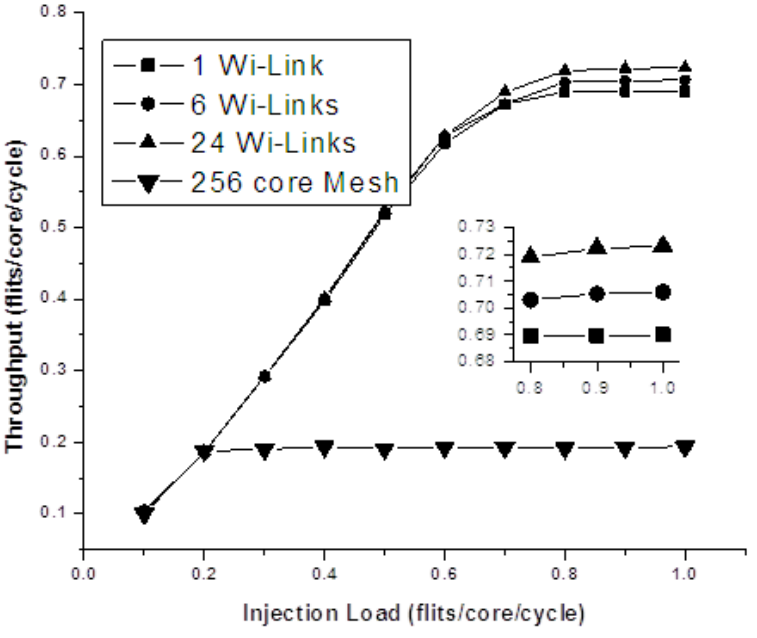


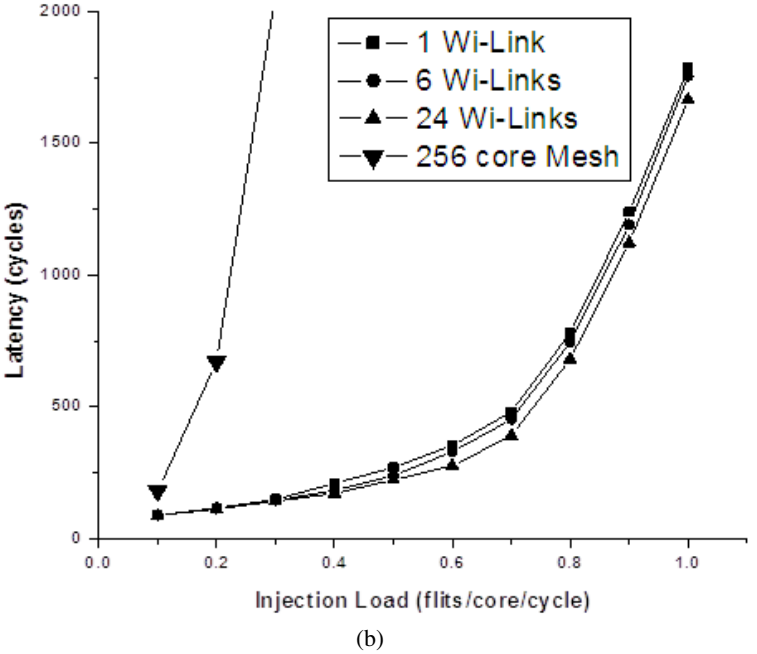
Fig. 7.6. The components of a WB with multiple wired and wireless ports at input and output [8]

electro-optic signal conversions with the MZM devices are 20ps. When computing the overall system latency and throughput of the WiNoCs the delays of these individual components are taken into account. This particular hierarchical topology was selected as it provided optimum system performance. Figure 7.8 shows the saturation throughputs for alternative ways of dividing the 256 core WiNoC into different numbers of subnets with a single wireless link. As can be seen from the plot all alternative configurations achieve worse saturation throughput. The same trend is observed if we vary the number of wireless links. Using the same method the suitable hierarchical division that achieves best performance is determined for all the other system sizes. For system sizes of 128 and 512, the hierachical divisions considered here achieved much better performance compared to the other possible divisions with either lower or higher number of subnets.

By varying the number of channels in the wireless links, various WiNoC configurations are created. We have considered WiNoCs with 1, 6, and 24 wireless links in our study. Since the total number of frequencies considered here is 24, the number of channels per link is 24, 4 and 1 respectively. As can be seen from Figure 7.8, the WiNoCs with different possible configurations outperform the single wired monolithic flat mesh architecture. It can also be observed that with increasing number of wireless links, throughput improves slightly.



(a)



(b)

Fig. 7.7. (a) Throughput and (b) latency of 256 core WiNoCs with different numbers of wireless links [8]

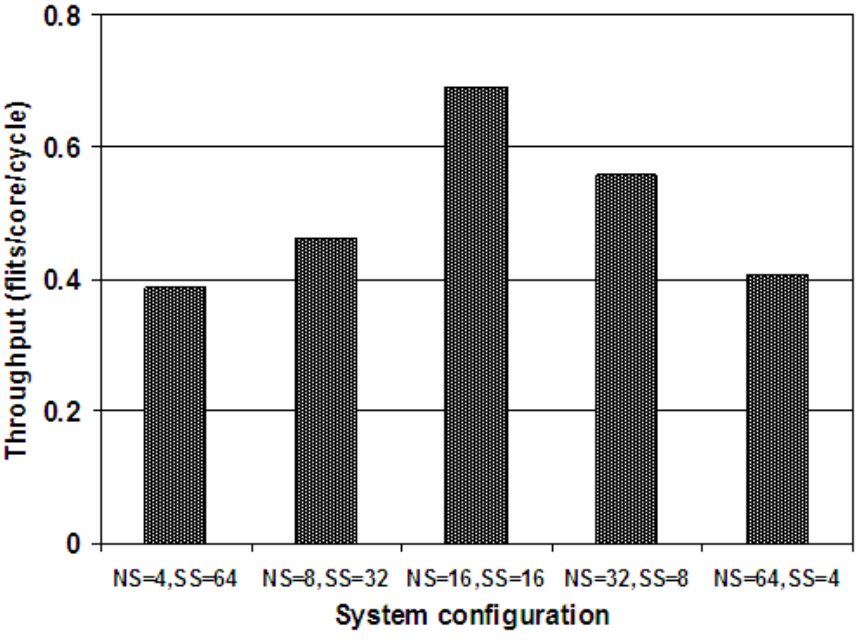
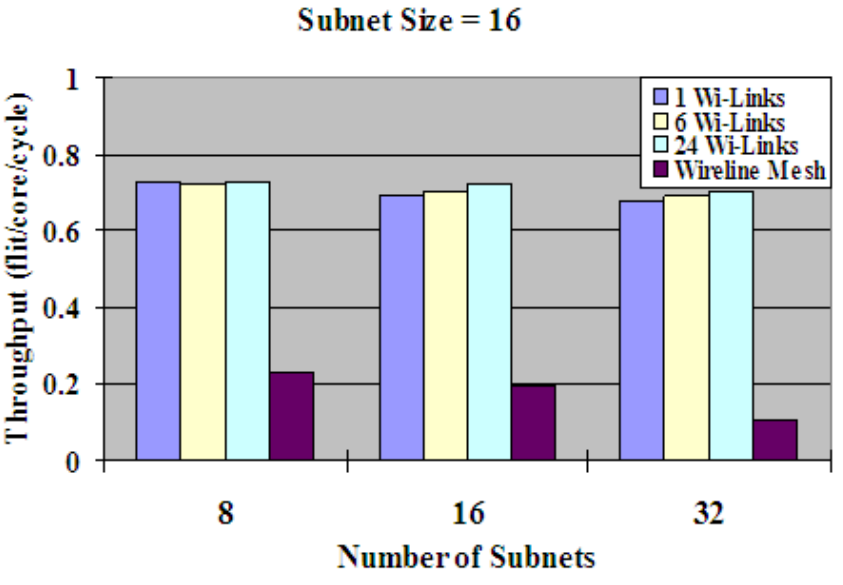


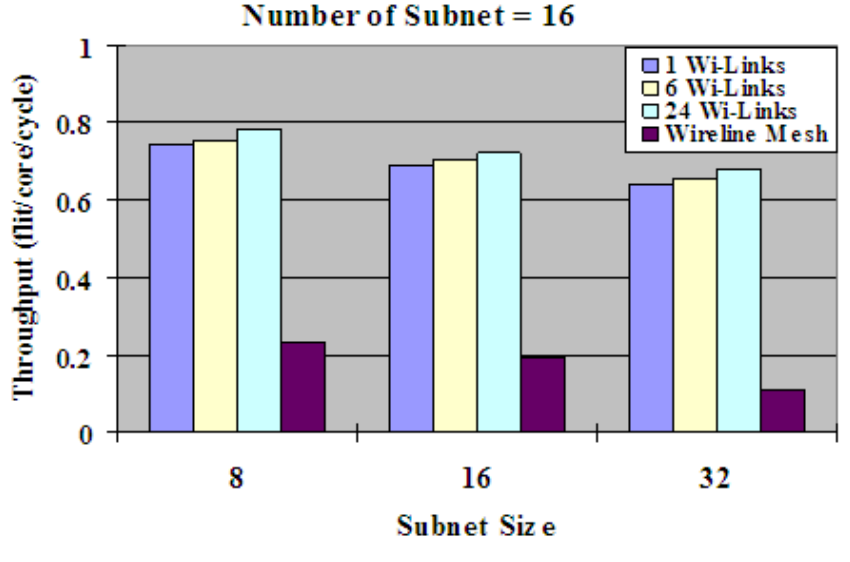
Fig. 7.8. Throughput of 256 core WiNoC for various hierarchical configurations [8]

It should be noted that even though increasing the number of links does increase the number of concurrent wireless communication links, the bandwidth on each link decreases as the total number of channels is fixed by the number of off-chip laser sources. This causes the total bandwidth over all the wireless channels to remain the same. The only difference is in the degree of distribution across the network. Consequently, network throughput increases only slightly with increasing number of wireless links. However, the hardware cost increases with increasing numbers of links as discussed in section 7.4.3. Thus, depending upon whether the demand on performance is critical the designer can choose to trade-off the area overhead of deploying the maximum number of wireless links possible. However, if the constraints on area overhead are really stringent then one can choose to employ only one wireless link and consequently provide more bandwidth per link and have only a little negative effect on performance.

In order to observe trends among various WiNoC configurations, we performed further analysis. Figure 9(a) shows the throughput at network saturation for various system sizes while keeping the subnet size fixed for different numbers of wireless links. Figure 9(b) shows the variation in throughput at saturation for different system sizes for a fixed number of subnets. For comparison, the throughput at network saturation for a single traditional wired mesh NoC of each system size is also shown in both of the plots. As in Figure 7.8 it may be noted from Figure 7.9 that for a WiNoC



(a)



(b)

Fig. 7.9. Saturation throughput with varying (a) number of subnets and (b) size of each subnet [8]

of any given size, number of subnets and subnet size, the throughput increases with increase in number of wireless links deployed.

As can be observed from the plots, the maximum achievable throughput in WiNoCs degrades with increasing system size for both cases. However, by scaling up the number of subnets, the degradation in throughput is smaller compared to when the subnet size is scaled up. By increasing the subnet size, we are increasing congestion in the wired subnets and load on the hubs and not fully using the capacity of the high speed wireless links in the upper level of the network. When the number of subnets scales up, traffic congestion in the subnets does not get worse and the optimal placement of the wireless links makes the top level network very efficient for data transfer. The effect on throughput with increasing system size is therefore marginal.

To determine the energy dissipation characteristics of the WiNoCs, we first estimated the energy dissipated by the antenna elements. As noted in [13], the directional gain of MWCNT antennas we propose to use is very high. The ratio of emitted power to incident power is around along the direction of maximum gain. Assuming an ideal line-of-sight channel over a few millimeters, transmitted power degrades with distance following the inverse square law. Therefore the received power P_R can be related to the transmitted power P_T as:

$$P_R = \frac{G_T \cdot A_R}{4 \cdot \Pi \cdot R^2} P_T. \tag{7.6}$$

In Equation (7.6), G_T is the transmitter antenna gain, which can be assumed to be -5dB [13]. A_R is the area of the receiving antenna and R is the distance between the transmitter and receiver. The energy dissipation of the transmitting antennas therefore depends on the range of communication. The area of the receiving antenna can be found by using the antenna configuration used in [13]. It uses a MWCNT of diameter 200 nm and length 7λ, where λ is the optical wavelength. The length 7λ was chosen as it was shown to produce the highest directional gain, G_T, at the transmitter. In one of the setups in [13], the wavelength of the laser used was 543.5 nm, and hence the length of the antenna is around 3.8 μm. Using these dimensions, the area of the receiving antenna, A_T can be calculated.

The noise floor of the LNA [38] is -101 dBm. Considering the MZM demodulators cause an additional loss of up to 3dB over the the operational bandwidth, the receiver sensitivity turns out to be in the worst case. The length of the longest possible wireless link considered among all WiNoC configurations is 23 mm. For this length and receiver sensitivity, a transmitted power of 1.3 mW is required. Considering the energy dissipation at the transmitting and receiving antennas, and the components of the transmitter and receiver circuitry such as the MZM, TDM block and the LNA, the energy dissipation of the longest possible wireless link on the chip is 0.33 pJ/bit. The energy dissipation of a wireless link, E_{Link} is given as

$$E_{Link} = \sum_{i=1}^{m} (E_{antenna,i} + E_{transceiver,i}), \tag{7.7}$$

where m is the number of frequency channels in the link and $E_{antenna,i}$ and $E_{transceiver,i}$ are the energy dissipations of the antenna element and transceiver circuits for the i-th frequency in the link.

The network switches and hubs are synthesized from a RTL level design using 65nm standard cell libraries from CMP [39], using Synopsys Design Vision and assuming a clock frequency of 2.5 GHz. A large set of data patterns were fed into the gate-level netlists of the network switches and hubs, and by running $Synopsys^{TM}$ Prime Power, their energy dissipation was obtained.

The energy dissipation of the wired links depends on their lengths. The lengths of the interswitch wires in the subnets can be found by using the formula

$$l_M = \frac{l_{edge}}{M-1}. \tag{7.8}$$

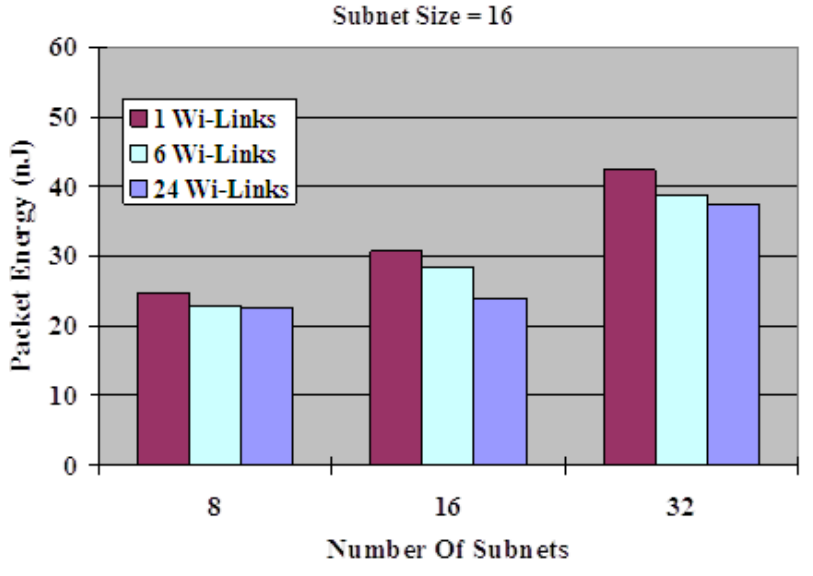
Here, M is number of cores along a particular edge of the subnet and ledge is the length of that edge. A $20mm \times 20mm$ die size is considered for all system sizes in our simulations. The inter-hub wire lengths are also computed similarly as these are assumed to be connected by wires parallel to the edges of the die in rectangular dimensions only. Hence, to compute inter-hub distances along the ring, parallel to a particular edge of the die, Equation (7.8) is modified by changing M to the number of hubs along that edge and ledge to the length of that particular edge. In each subnet the lengths of the links connecting the switches to the hub depend on the position of the switches in a $20mm \times 20mm$ die. The capacitances of each wired link, and subsequently their energy dissipation, were obtained through HSPICE simulations taking into account the specific layout for the subnets and the 2nd level of the ring network.

Figures 10(a) and 10(b) show the packet energy dissipation for each of the network configurations considered here. The packet energy for the flat wired mesh architecture is not shown as it is higher than that of the WiNoCs by orders of magnitude, and hence cannot be shown on the same scale. The comparison with the wired case is shown in Table 7.3 in the next subsection along with another hierarchical wired architecture.

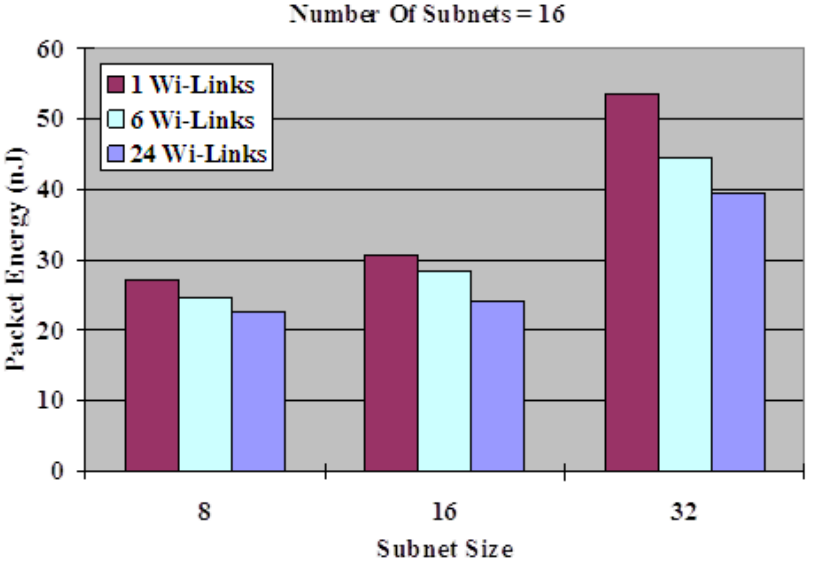
Table 7.3. Packet energy dissipation for flat wired mesh, WiNoC and hierarchical G-line NoC architectures [8]

System Size	Subnet Size	No. of Subnets	Flat Mesh(nJ)	WiNoC (nJ)	NoC with G-Line (nJ)
128	16	8	1319	22.57	490.30
256	16	16	2936	24.02	734.50
512	16	32	4992	37.48	1012.81

From the plots it is clear that the packet energy dissipation increases with increasing system size. However, scaling up the number of subnets has a lower impact on the average packet energy. The reason for this is that the throughput does not degrade much and the average latency per packet also does not change significantly. Hence, the data packets occupy the network resources for less duration, causing



(a)



(b)

Fig. 7.10. Packet energy dissipation with varying (a) number of subnets and (b) size of each subnet [8]

only a small increase in packet energy. However, with an increase in subnet size, the throughput degrades noticeably, and so does latency. In this case, the packet energy increases significantly as each packet occupies network resources for a longer period of time. With an increase in the number of wireless links while keeping the number of subnets and subnet size constant, the packet energy decreases. This is because higher connectivity of the network results in higher throughput (or lower latency), which means that packets get routed faster, occupy network resources for less time, and consume less energy during the transmission. Since the wireline subnets pose the major bottleneck as is made evident by the trends in the plots of Figures 7.9 and 7.10 their size should be optimized. In other words, smaller subnets imply better performance and lower packet energies. Hence, as a designer one should target to limit the size of the subnets as long as the size of the upper level of the network does not impact the performance of the overall system negatively. The exact optimal solution also depends on the architecture of the upper level of the network, which need not be restricted to the ring topology chosen in this chapter as an example.

The adopted centralized routing strategy is compared with the distributed routing discussed in section 7.3.4 for a WiNoC of size 256 cores split into 16 subnets with 16 cores in each. Twenty four wireless links were deployed in the exact same topology for both cases. With distributed routing the throughput was 0.67 flits/core/cycle whereas with centralized routing it was 0.72 flits/core/cycle as already noted in Figure 7.7, which is 7.5% higher. Centralized routing results in a better throughput as it finds the shortest path whereas the distributed routing uses non-optimal paths in some cases. Hence, the distributed routing has lower throughput. The distributed routing dissipates a packet energy of 31.2 nJ compared to 30.8 nJ with centralized routing. This is because on an average the number of path length computation with the distributed routing is more per packet, as this computation occurs at every intermediate WB. However, with centralized routing each hub has additional hardware overhead to compute the shortest path by comparing all the paths using the wireless links.

7.5 Reliability in WiNoCs

Aggressive scaling in the nanometer technology nodes result in inherently unreliable or defect prone devices. The performance of the wireless links in the WiNoC depends on the CNT antennas. Like any other nanodevices, CNT antennas are expected to have higher manufacturing defect rates, operational uncertainties and process variability. Error Control Coding (ECC) has been proposed for mitigating the inherent reliability issues in on-chip communication [14]. Defect-prone CNTs can lead to relatively high random error rates and cause multi-bit errors or burst errors. Therefore the coding scheme should be robust against both random and burst errors. As a first step, in the next subsection we present a model of the on-chip wireless channel and evaluate the corresponding bit error rates. In latter subsections we

propose and evaluate ECC schemes for the on-chip wireless links and for the wired links, in order to improve the reliability of a WiNoC.

7.5.1 Wireless Channel Model

By elevating the chip packaging material from the substrate to create a vacuum for transmission of the high frequency EM waves, LOS communication between WBs using CNT antennas at optical frequencies can be achieved. Techniques for creating such vacuum packaging are already utilized for MEMS applications, and can be adopted to make creation of LOS communication between CNT antennas viable. However, reflection from the surfaces of the substrates and the packaging material interfere with the LOS transmitted power. In the channel model we account for the multipath reflection from all 6 surfaces of the packaging as well as the thermal noise coupled to the received signal from the LOS transmission.

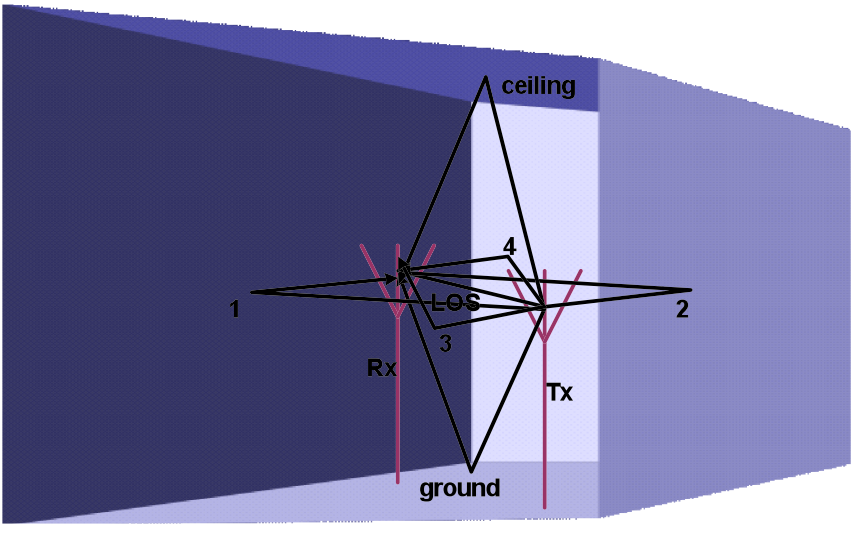


Fig. 7.11. Multi-path channel model for on-chip wireless links [9]

Figure 7.11 shows the multipath transmission of the signal from the transmitter to the receiver with all the possible reflected rays from four walls, ceiling and ground. The total received power is given by

$$P_R = \frac{A_R}{4\Pi} \cdot P_T \cdot \left[\begin{array}{c} G_{T,LOS} \frac{e^{-j\frac{2\Pi}{\lambda}R_{LOS}}}{R_{LOS}} + \Gamma_1 G_{T1} \frac{e^{-j\frac{2\Pi}{\lambda}R_1}}{R_1} + \Gamma_2 G_{T2} \frac{e^{-j\frac{2\Pi}{\lambda}R_21}}{R_2} + \\ + \Gamma_3 G_{T3} \frac{e^{-j\frac{2\Pi}{\lambda}R_3}}{R_3} + \Gamma_4 G_{T1} \frac{e^{-j\frac{2\Pi}{\lambda}R_4}}{R_1} + \Gamma_{ceiling} G_{T,ceiling} \frac{e^{-j\frac{2\Pi}{\lambda}R_{ceiling}}}{R_{ceiling}} + \\ + \Gamma_{ground} G_{T,ground} \frac{e^{-j\frac{2\Pi}{\lambda}R_{ground}}}{R_{ground}} \end{array} \right]^2, \quad (7.9)$$

where $G_{T,LOS}$ is the transmitter antenna gain along the LOS, which is shown to be -5dB [13]. A_R is the area of the receiving antenna and R_{LOS} is the LOS distance between the transmitter and receiver. R_1, R_2, R_3, R_4, $R_{ceiling}$ and R_{ground} are the distances along the different reflected paths. Γ_1, Γ_2, Γ_3, Γ_4, $\Gamma_{ceiling}$ and Γ_{ground} are the coefficients of reflections on the substrate and packaging surfaces obtained from [40]. G_{T1}, G_{T2}, G_{T3}, G_{T4}, $G_{T,ceiling}$ and $G_{T,ground}$ are the antenna gains along the directions of the reflected paths which are all less than -10dB [13]. Due to low coefficients of reflection of the packaging materials and high directional gains of the antennas only primary reflections are considered in this model. Subsequent reflections will diminish the reflected power further and may be neglected in this case. The thermal noise power is given by

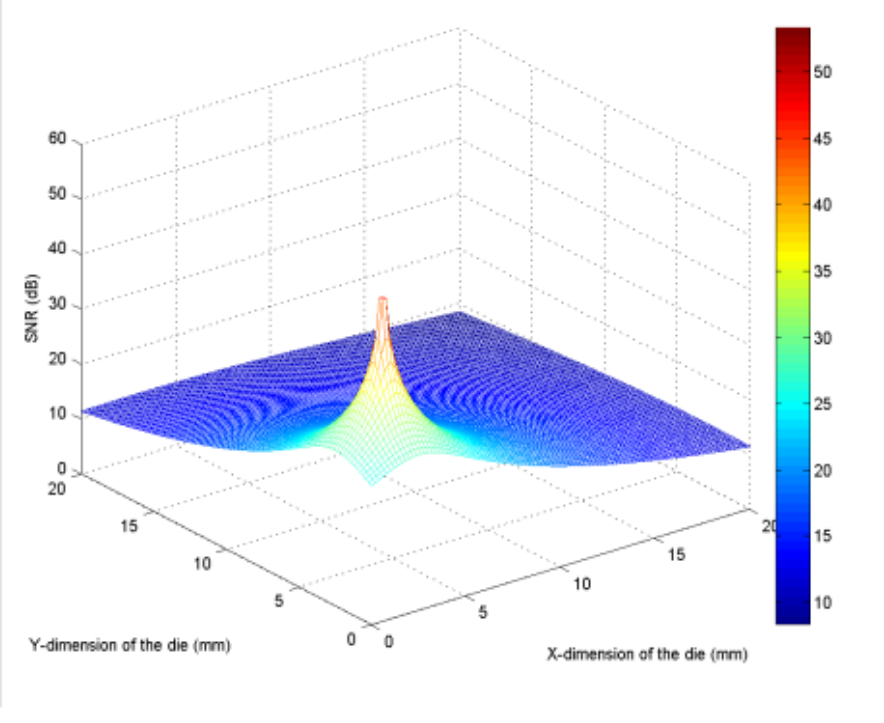


Fig. 7.12. SNR over the die area due to multipath radiation from a transmitter placed in the first subnet at its center (X=2.5 mm, Y=2.5 mm) [9]

$$N_0 = k \cdot T_0 \cdot F = k \cdot T_0 \cdot \left(\frac{T_{antenna}}{T_0} + F_r \right) \tag{7.10}$$

where k is the Boltzmann constant, T_0 is the room temperature taken as 290K, $T_{antenna}$ is the antenna temperature assumed to be 330K, and F_r is the receiver noise figure of 4dB [38]. The coupling of the chip switching noise to the wireless channels is negligible as the wireless channels are in very high frequency bands of a few THz. Hence, the *Signal-to-Noise Ratio (SNR)* is given by

$$SNR = \frac{P_R}{N_0} \tag{7.11}$$

Figure 7.12 shows the variation of the SNR over the $20mm \times 20mm$ die area for a fixed position of the transmitter. The transmitter in this case is placed at the center of the subnet at the near corner in figure 12. The received SNR is the maximum near the transmitter and gradually diminishes with distance. Even though there is multipath reflection from the substrate and packaging walls, the reflected power is negligible as signified by the quadratic variation of the received SNR with distance in Figure 7.12. This is due to low coefficient of reflection of the reflecting surfaces and high directivity of the antennas which attenuate the radiations in the directions other than the LOS. The SNR vs. bit-error-rate (BER) characteristics for the wireless links correspond to the adopted modulation scheme, which is non-coherent OOK.

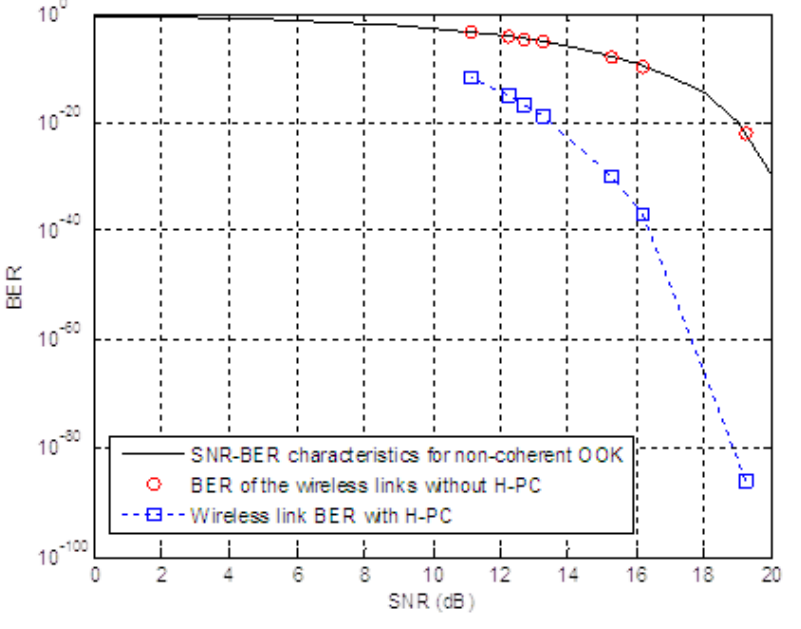


Fig. 7.13. SNR vs. BER plot of the wireless channel with and without coding [9]

Figure 7.13 shows the variation of BER with SNR for a non-coherent OOK receiver. The particular configuration of the wireless links established for optimal network performance by simulated annealing places the 24 links between specific subnets in the WiNoC. The SNR and hence the BER corresponding to those links are also marked with circles in the plot. The SNR and hence the BER on those 24 links varies as each link is of a different length resulting in different path loss and reflected radiation patterns. However, some of the 24 links have the same SNR and hence appear as the same point on the plot. As can be seen the highest BER for the wireless links on the chip is around $4 times 10^{-4}$. Since this is the highest BER among all the wireless links it can be referred to as the effective BER of the WiNoC. This effective BER of the WiNoC is much higher than the BER of wireline links, which is typically around 10-15 or less [6]. Hence, we propose using powerful multiple/burst error correction codes for the wireless channels. In the next subsection we describe a product code based ECC to enhance the reliability of the wireless links.

7.5.2 *Proposed Product Code for the Wireless Links*

In order to achieve simultaneous random and burst-error correction on the wireless links simple Hamming code based product codes (H-PC) are proposed in [9]. The authors of [19] have already shown that product codes designed from simple single error correcting Hamming codes in 2 dimensions can perform better than multiple error correction codes like *Bose-Chaudhuri-Hocquenghem (BCH)* codes or *Reed-Solomon (RS)* codes in terms of trade-offs between overall performance and overhead. Here we show that a simple product code can be used to achieve multiple error correction as well as burst error correction of data transmitted through the wireless links.

Figure 7.14 14 schematically shows the product code structure. Let us assume a flit size of k_1 bits and a block of k_2 such flits. (n_1, k_1) Hamming encoding is performed in the spatial dimension on the flits. Each of the (n_1, k_1) Hamming encoded flits are then encoded using a (n_2, k_2) Hamming encoding is done in the time dimension to give a $(n_1 \times n_2, k_1 \times k_2)$ product code. In this chapter, we chose a $(38, 32)$ shortened Hamming code in the spatial dimension to encode a whole 32 bit flit at a time. In the time dimension a $(7, 4)$ Hamming code is chosen to minimize the latency and buffering overheads of storing a block of bigger size. Any bigger code would cause higher buffering requirements at the wireless nodes in the WiNoC. The product code decoder utilizes a row-column decoding technique where first the $(38, 32)$ Hamming decoder operates on the received columns of the block and then the $(7, 4)$ Hamming decoders decode the rows to give back the 4 received 32 bit flits. This decoding technique is referred to as column-first decoding. In order to mask the latency penalty of the decoding, the decoder is designed such that the $(38, 32)$ Hamming decoder operates on the received flits as they arrive and 32 parallel $(7, 4)$

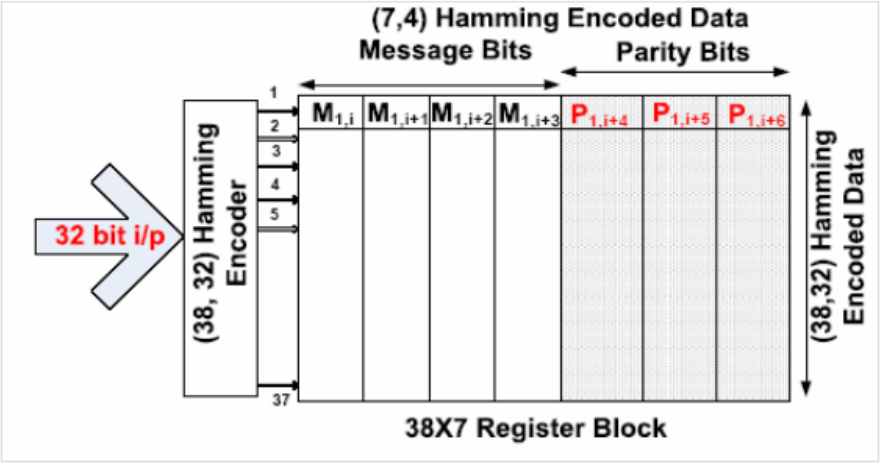


Fig. 7.14. Schematic structure of the proposed H-PC encoder [9]

Hamming decoders then operate in parallel on the received 32 bit flits. This minimizes the latency overhead of the H-PC decoder.

7.5.3 Residual BER of the Wireless Channel with H-PC

In order to estimate the effectiveness of the proposed coding scheme we perform a residual BER analysis after implementing the ECC. This analysis is based on the smallest weight error events that are uncorrectable by the product code, since these events will dominate the BER at high SNR.

Figure 7.15 shows the various correctable and uncorrectable error patterns in a block of size $n_1 \times n_2$ for the row-column decoding technique. The scenario shown in Figure 7.15(a) has a spatial burst along a particular flit represented by the shaded

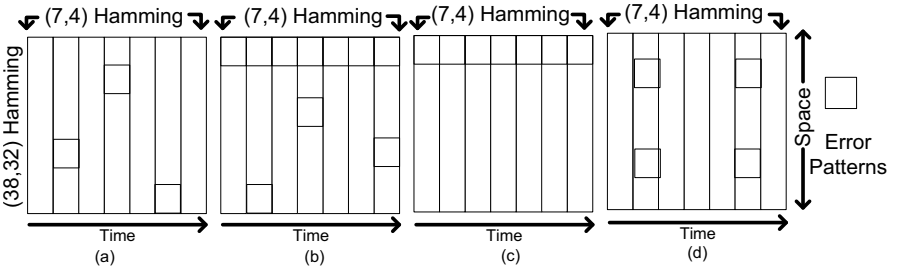


Fig. 7.15. (a)-(c): Different correctable error patterns. (d) Uncorrectable pattern [9]

column, as well as single bit random errors in other flits. This can be corrected if column decoding is done first followed by row decoding, as in the adopted column-first decoding. The column decoding will correct all the random errors but not the burst, which, however, will be corrected by row decoding. For the scenario in Figure 7.15(b) with a single burst in time and some random errors, the column-first decoding scheme will correct all errors if the uncorrectable error patterns on the columns produce correctable error patterns on the rows after column decoding. However this case is not likely to occur because each antenna is responsible for transmission of multiple bits in the same flit before transmitting bits of another flit. The case shown in Figure 7.15(c) with a burst in each direction can be corrected completely, as column decoding will correct the burst in time except the top left bit. The resulting pattern is only a burst in space, which can be corrected by row decoding. Rectangular error patterns as shown in Figure 7.15(d), cannot be corrected by row-column decoding because double errors occur in both rows and columns. This is the highest probability event as only 4 erroneous bits in a whole block lead to uncorrectable error patterns. The probability of this event is given by

$$P_{rectangle} = N_{rectangle} \cdot \epsilon^4 \left(1 - \epsilon^{n_1 \times n_2 - 4}\right), \tag{7.12}$$

where $N_{rectangle}$ is the number of rectangular patterns possible in a block of size $n_1 \times n_2$ and ϵ is the BER without coding. $N_{rectangle}$ is computed as

$$N_{rectangle} = \sum_{l=2}^{n_1} \sum_{b=2}^{n_2} (n_1 - l + 1) \cdot (N - 2 - b + 1). \tag{7.13}$$

At high SNR where the highest probability events dominate the BER, the residual BER with the product code is given by

$$\epsilon_{PC} = \left(\frac{1}{n_1 n_2}\right) P_{rectangle}. \tag{7.14}$$

The new SNR vs. residual BER on the particular wireless links are shown in Figure 7.13 after implementing the product code with squares on a dotted line. The effect of the product code is to significantly lower residual BER. The worst case BER on the link with the lowest SNR is 1.99×10^{-12}. This reduction in BER gives a higher level of reliability compared to the uncoded system. In addition, the worst case BER of the wireless channel becomes comparable with the BER in the wireline links by using the H-PC scheme.

7.5.4 Error Control Coding for the Wireline Links

In the subnets, the data transmission takes place through the wireline links. It is well known that with shrinking geometry, wireline links will be increasingly exposed to different sources of transient noise affecting signal integrity and system reliability. Data-dependent crosstalk between adjacent wires is also a major source of such

transient noise. Due to shrinking feature size in future technologies the transient error rate is predicted to increase by several orders of magnitude. As these errors are not necessarily correlated, higher rate of failures can cause uncorrelated multiple bit errors in data blocks. In [7]] a family of joint crosstalk avoidance and multiple error correction codes have been proposed. These codes avoid worst-case crosstalk between adjacent wires as well as correct up to three random errors in a flit and can detect four errors. The underlying idea behind this coding scheme is that the Hamming distance between two Single Error Correcting and Double Error Detecting (SEC-DED) Hsiao code [41] words is 4. Duplication of these encoded bits result in doubling the minimum Hamming distance between the code words to 8. Thus it is possible to correct 3 errors in the code word and simultaneously detect any 4-error events. Duplicating the bits and interleaving them beside the original bit lines also serves the purpose of avoiding worst case crosstalk by eliminating opposite transitions on either side of any particular wire. This coding scheme is referred to as Joint Crosstalk Avoidance Triple Error Correction and Simultaneous Quadruple Error Detection Code (JTEC-SQED) [7].

7.5.4.1 Residual BER with JTEC-SQED

To compute the residual word error probability for the JTEC-SQED scheme let us assume that the total number of bits in the flit is $2n+2$, where there are 2 copies of a $(n+1,k)$ SEC-DED codeword. Since JTEC-SQED can either correct or detect up to four errors, the lower bound on the probability of correct decoding can be obtained as

$$P_{correct} \geq P(2n+2,0) + \ldots + P(2n+2,4), \tag{7.15}$$

where $P(i,j)$ is the probability of j random errors in i bits given by

$$P(i,j) = \binom{i}{j} \epsilon^j (1-\epsilon)^{i-j} \tag{7.16}$$

The set of correctly decoded words is complementary to the set of residual word errors. Hence the residual word error probability can be computed using

$$P_{JTEC-SQED} = 1 - P_{orrect}, \tag{7.17}$$

where $P_{JTEC-SQED}$ is the residual word error probability in presence of coding and P_{orrect} is the probability of correct decoding. Using Equation (7.9) and (7.11) the residual word error probability of the JTEC-SQED scheme for small values of ϵ can be approximated as:

$$P_{JTEC-SQED} = \binom{2n+2}{5} \cdot \epsilon^5 \tag{7.18}$$

Incorporation of error control coding enhances the reliability of the communication channel as it becomes robust against transient malfunctions. Increase in reliability by incorporating coding can be translated into a reduction in voltage swing on the

interconnect wires as they can tolerate lower noise margins. This results in a savings in energy dissipation as it depends quadratically on the voltage swing. In this section we quantify these gains by modeling the voltage swing reduction as a function of increased error correction capability. The cumulative effect of all transient UDSM noise sources can be modeled as an additive Gaussian noise voltage V_N with variance σ_N^2 [6]. Using this model, the BER ϵ depends on the voltage swing V_dd according to

$$\epsilon = Q \cdot \left(\frac{V_{dd}}{2\sigma} \right) \tag{7.19}$$

where the Q-function is given by

$$Q(x) = \frac{1}{\sqrt{2\Pi} \int_x^\infty e^{\frac{y^2}{2}} dy}. \tag{7.20}$$

The word error probability is a function of the channel BER ϵ. If $P_{UNC}(\epsilon)$ is the residual probability of word error in the uncoded case and $P_{ECC}(\epsilon)$ is the residual probability of word error with error control coding, then it is desirable that $P_{ECC}(\epsilon) \leq P_{UNC}(\epsilon)$. Using Equation (7.13), we can reduce the supply voltage in presence of coding to $\widehat{V}_{dd}$, given by [6]

$$\widehat{V}_{dd} = V_{dd} \frac{Q^{-1}(\widehat{\epsilon})}{Q^{-1}(\epsilon)}. \tag{7.21}$$

In Equation (7.21), V_{dd} is the nominal supply voltage in the absence of any coding, $\widehat{V}_{dd}$ is the reduced voltage swing with coding and $\widehat{\epsilon}$ is the BER such that

$$P_{ECC}(\widehat{\epsilon}) = P_{UNC}(\epsilon). \tag{7.22}$$

Use of lower voltage swing makes the probability of multi-bit error patterns higher, necessitating the use of multiple error correcting codes in order to maintain the same word error probability as the uncoded case. As the JTEQ-SQED scheme can reduce the voltage swing the most and correct the highest number of errors in a single flit it was chosen for the wireline links of the WiNoC.

By using JTEC-SQED on the wireline links of the NoC it is possible to maintain the same level of reliability as without any ECC but with lower energy dissipation due to reduced voltage swing and crosstalk capacitance on the wires. For the wireless links as shown in Figure 7.13 the BER without any ECC is quite high. However, with H-PC it is possible to achieve the BER comparable to that of the wireline NoC without ECC. We investigated the performance of a WiNoC with a unified coding scheme using JTEC-SQED for the wireline links and H-PC for the wireless links to achieve the same BER as a completely wireline NoC without any ECC. In the following section we present the performance and energy dissipation characteristics of a WiNoC with the unified coding scheme and compare it with a wireline NoC.

7.6 Experimental Results

In order to characterize the performance of the proposed coding schemes in a WiNoC, we consider a system consisting of 256 cores. Following the design principles for highest throughput in [8] the WiNoC is divided into 16 subnets each with 16 cores, and the 24 wireless links are distributed among the subnets. The size of the upper and lower level of the hierarchical WiNoC are chosen to be the same so as not to make either one larger than the other resulting in that level becoming the bottleneck. The cores within each subnet are connected with wireline links following a mesh topology. We assume a die size of $20mm \times 20mm$. The switch and hub architectures are adopted from [8].

The network switches, hubs and codecs for the ECCs are synthesized from a RTL level design using 65 nm standard cell libraries from CMP (www.cmp.imag.fr), using Synopsys Design Vision and assuming a clock frequency of 2.5 GHz. The energy dissipation on the wires was obtained from CADENCE Spectre. The energy dissipation on the wireless links was obtained from Equation (7.3). The WiNoC is simulated using a cycle accurate simulator which models the progress of data flits accurately per clock cycle accounting for flits that reach destination as well as those that are dropped.

Figure 7.16 shows the packet energy dissipation of the WiNoC with and without the proposed unified ECC scheme. Packet energy dissipation is the energy dissipated in transferring one packet from source to destination. For comparison, the packet energy dissipation in a completely wireline mesh NoC with and without ECC is also

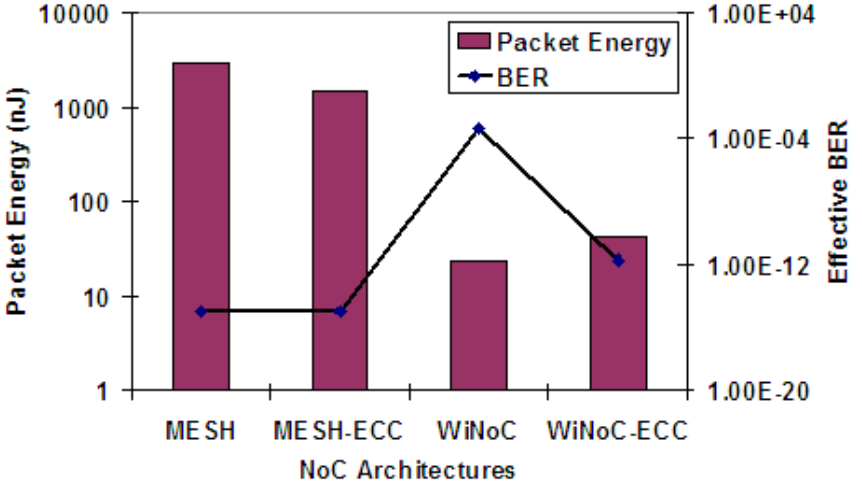


Fig. 7.16. Packet energy dissipation and worst case channel BER for WiNoC and mesh architectures with and without ECC [9]

shown. The ECC used in the wireline links of the subnets is JTEC-SQED as it is the most energy efficient coding scheme [7]. The effective BER on the communication links in the corresponding cases are also shown. For the mesh with JTEC-SQED, the BER is the same as in the mesh without any ECC as the voltage swing was reduced on the links in accordance with Equation (7.15). For the WiNoCs the BER on the wireless channels is much higher than that of the wireline links and hence this represents the effective BER on the communication links. As shown in [7] the packet energy of the wireline mesh is reduced due to JTEC-SQED, as the voltage swing on the wireline links can be reduced significantly according to Equation (7.15). In addition, the crosstalk coupling capacitances on the wires are reduced. The energy savings due to these two factors are more than the overheads due to the codecs and redundant links [7]. The overall reliability on the wires is the same in both cases as the reduction in energy dissipation is projected for the same BER on the wires. For the WiNoC without ECC the worst case BER is much higher due to low SNR. But with the powerful HPC coding on the wireless channels the BER of the wireless channels are reduced to around 10–12.

Hence, the overall reliability of the WiNoC is improved with the coding schemes. The overheads of the H-PC scheme and the JTEC-SQED are considered and we find that there is an increase in the packet energy due to the codec overheads of the ECCs compared to the WiNoC without any coding. However, at the cost of this slight increase in packet energy we are able to achieve a comparable overall BER on the WiNoC as that of the completely wireline mesh NoC. The packet energy dissipation of the WiNoC with ECC however still remains several orders of magnitude lower than that of a complete wireline counterpart due to the low-power long range wireless links.

We also estimate the timing characteristics of the WiNoC in comparison to the wireline mesh. The end-to-end message latency of the wireline mesh with JTEC-SQED is higher than that of the mesh without ECC. This is because the ECC codecs add overheads to the critical paths in the NoC switches. The WiNoC without any ECC achieves a much lower latency compared to the wireline network due to the advantages of the long-range wireless shortcuts introduced into the network as well as the hierarchical division of the NoC. Due to the wireless shortcuts in the WiNoC, the average hop-count between cores is much less compared to that of a mesh of the same size. Hence, as shown in [8] the performance of the WiNoC is much better compared to the wireline mesh NoC. However, similar to the JTEC-SQED scheme, the H-PC codec also adds timing overheads to the wireless links. The encoder design as outlined earlier requires a $(38,32)$ encoding on each 32 bit flit which then are stored until 4 flits are received. All 4 flits are then encoded with 38 parallel $(7,4)$ Hamming encoders. Hence, the overhead of the H-PC encoder is equal to the delay of one $(38,32)$ Hamming encoder and one $(7,4)$ Hamming encoder as all the $(7,4)$ encoders operate in parallel. The delays of the various stages of the H-PC and JTEC-SQED schemes are shown in Table 7.4.

The latency penalty due to the code-rate of $(32\times4)/(38\times7) = 0.48$ is also taken into account. In the wireline links however, extra cycles were not required due to the

Table 7.4. Delay for Each Coding Scheme [9]

Coding Scheme	Codec Delay (ps)	
	Encoder	Decoder
H-PC	150	354
JTEC-SQED	133	315

code-rate as the redundant bits could be transferred in the same cycle with additional wires.

Figure 7.17 shows the overall latency characteristics of the WiNoC and the wireline mesh with and without coding. Due to the low code rate of the H-PC as well as codec overheads the overall latency of the WiNoC with coding is higher than the WiNoC without coding. However, even with coding the latency of the WiNoC is much less compared to that of a wireline mesh NoC without any coding.

The codecs introduce additional hardware components and hence also require silicon area overheads. Table 7.5 summarizes the area overheads of each of the coding schemes. The reported area is the area of the codec required per port of the switches or hubs.

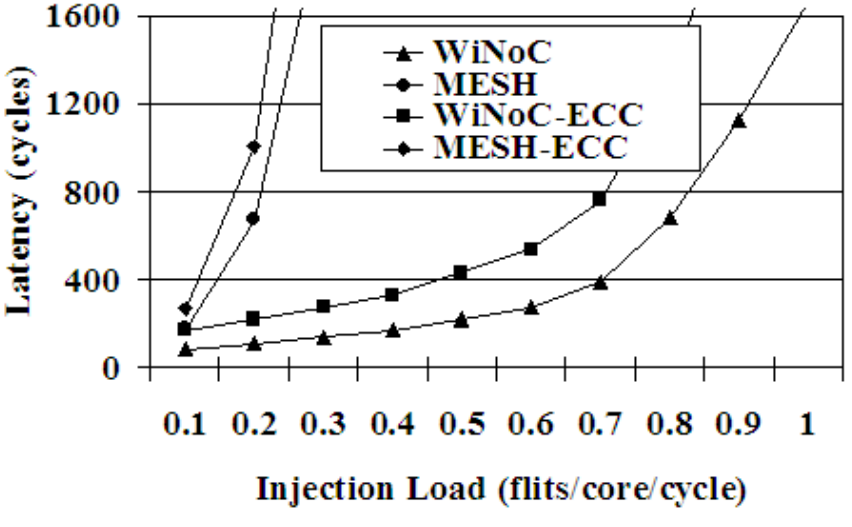


Fig. 7.17. Latency characteristics of mesh and WiNoC architectures with and without ECC [9]

Table 7.5. Area Overhead of the Codec for Each Coding Scheme [9]

Coding Scheme	Area (μm^2)
H-PC	29710
JTEC-SQED	11055

7.7 Conclusion

Wireless Network-on-Chip has emerged as one of the many alternative interconnect technologies for future multi-core chips. However, the reliability of on-chip wireless links is much less compared to the wired interconnects. Here we present a hierarchical hybrid nature inspired WiNoC architecture along with a unified ECC mechanism by which we can restore the reliability of the wireless links to that of the wired interconnects and also reduce energy dissipation on the wired interconnects. It is demonstrated that with such ECC enhanced wireless links on the chip it is possible to achieve lower energy dissipation, lower latency and almost equally reliable on-chip data communication compared to traditional multi-hop wireline NoCs. Thus leading towards a sustainable and reliable multi-core computing paradigm.

References

1. Pavlidis, V.F., Friedman, E.G.: 3-D Topologies for Networks-on-Chip. IEEE Transactions on Very Large Scale Integration (VLSI) 5(10), 1081–1090 (2007)
2. Shacham, A., et al.: Photonic Network-on-Chip for Future Genera-tions of Chip Multi-Processors. IEEE Transactions on Computers 57(9), 1246–1260 (2008)
3. Chang, M.F., et al.: CMP Network-on-Chip Overlaid With Multi-Band RF-Interconnect. In: Proc. of IEEE International Symposium on High-Performance Computer Architecture (HPCA), February 16-20, pp. 191–202 (2008)
4. Zhao, D., Wang, Y.: SD-MAC: Design and Synthesis of A Hardware-Efficient Collision-Free QoS-Aware MAC Protocol for Wireless Network-on-Chip. IEEE Transactions on Computers 57(9), 1230–1245 (2008)
5. Benini, L., Micheli, G.D.: Networks on Chips: A New SoC Paradigm. IEEE Computer 35(1), 70–78 (2002)
6. Sridhara, S.R., Shanbhag, N.R.: Coding for System-on-Chip Networks: A Unified Framework. IEEE Transactions on Very Large Scale Integration (TVLSI) Systems 13(6), 655–667 (2005)
7. Ganguly, A., et al.: Crosstalk-Aware Channel Coding Schemes for Energy Efficient and Reliable NoC Interconnects. IEEE Transactions on VLSI (TVLSI) 17(11), 1626–1639 (2009)
8. Ganguly, A., et al.: Scalable Hybrid Wireless Network-on-Chip Architectures for Multi-Core Systems. IEEE Transactions on Computers (TC) 60(10), 1485–1502 (2011)
9. Ganguly, A., et al.: A Unified Error Control Coding Scheme to Enhance the Reliability of a Hybrid Wireless Network-on-Chip. In: Proc. of the IEEE Defect and Fault Tolerance Symposium (DFTS), Vancouver, Canada (October 2011)
10. Kumar, A., et al.: Toward Ideal On-Chip Communication Using Express Virtual Channels. IEEE Micro 28(1), 80–90 (2008)
11. Ogras, U.Y., Marculescu, R.: "It's a Small World After All": NoC Performance Optimization Via Long-Range Link Insertion. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 14(7), 693–706 (2006)
12. Chang, M.F., et al.: RF Interconnects for Communications On-Chip. In: Proc. of International Symposium on Physical Design, April 13-16, pp. 78–83 (2008)
13. Kempa, K., et al.: Carbon Nanotubes as Optical Antennae. Advanced Materials 19, 421–426 (2007)
14. Bertozzi, D., Benini, L., De Micheli, G.: Error control schemes for on-chip communication links: The energy-reliability trade off. IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 24(6), 818–831 (2005)

15. Murali, S., De Micheli, G., Benini, L., Theocharides, T., Vijaykrishnan, N., Irwin, M.: Analysis of error recovery schemes for networks on chips. IEEE Design Test Comput. 22(5), 434–442 (2005)
16. Rossi, D., Metra, C., Nieuwland, A.K., Katoch, A.: Exploiting ECC redundancy to minimize crosstalk impact. IEEE Design Test Comput. 22(1), 59–70 (2005)
17. Patel, K.N., Markov, I.L.: Error-correction and crosstalk avoidance in DSM busses. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 12(10), 1076–1080 (2004)
18. Rossi, D., Metra, C., Nieuwland, A.K., Katoch, A.: New ECC for crosstalk effect minimization. IEEE Design Test Comput. 22(4), 340–348 (2005)
19. Fu, B., Ampadu, P.: Error control combining Hamming and product codes for energy efficient nanoscale on-chip interconnects. IET Computers & Digital Techniques 4(3), 251–261 (2010)
20. Watts, D.J.: Small Worlds: The Dynamics of Networks between Order and Randomness pp. xvi–262. Princeton University Press (1999)
21. Albert, R., Barabasi, A.-L.: Statistical mechanics of complex networks. Reviews of Modern Physics 74, 47–97 (2002)
22. Buchanan, M.: Nexus: Small Worlds and the Groundbreaking Theory of Networks. Norton, W. W. & Company, Inc. (2003)
23. Teuscher, C.: Nature-Inspired Interconnects for Self-Assembled Large-Scale Network-on-Chip Designs. Chaos 17(2), 26–106 (2007)
24. Kirkpatrick, S., et al.: Optimization by Simulated Annealing. Science. New Series 220(45978), 671–680
25. Duato, J., et al.: Interconnection Networks - An Engineering Approach. Morgan Kaufmann (2002)
26. Floyd, B.A., et al.: Intra-Chip Wireless Interconnect for Clock Distribution Implemented With Integrated Antennas, Receivers, and Transmitters. IEEE Journal of Solid-State Circuits 37(5), 543–552 (2002)
27. Fukuda, M., et al.: A 0.18 m CMOS Impulse Radio Based UWB Transmitter for Global Wireless Interconnections of 3D Stacked-Chip System. In: Proc. of International Conference Solid State Devices and Materials, pp. 72–73 (September 2006)
28. Hanson, G.W.: On the Applicability of the Surface Imped-ance Integral Equation for Optical and Near Infrared Copper Dipole Antennas. IEEE Transactions on Antennas and Propagation 54(12), 3677–3685 (2006)
29. Burke, P.J., et al.: Quantitative Theory of Nanowire and Nanotube Antenna Performance. IEEE Transactions on Nanotechnology 5(4), 314–334 (2006)
30. Huang, Y., et al.: Performance Prediction of Carbon Nano-tube Bundle Dipole Antennas. IEEE Transactions on Nanotechnology 7(3), 331–337 (2008)
31. Zhou, Y., et al.: Design and Fabrication of Microheaters for Localized Carbon Nanotube Growth. In: Proc. of IEEE Conference on Nanotechnology, pp. 452–455 (2008)
32. Freitag, M., et al.: Hot carrier electroluminescence from a single carbon nanotube. Nano Letters 4(6), 1063–1066 (2004)
33. Marinis, T.S., et al.: Wafer level vacuum packaging of MEMS sensors. In: Proc. of Electronic Components and Technology Conference, May 31-June 3, vol. 2, pp. 1081–1088 (2005)
34. Lee, B.G., et al.: Ultrahigh-Bandwidth Silicon Photonic Nan-owire Waveguides for On-Chip Networks. IEEE Photonics Technology Letters 20(6), 398–400 (2008)
35. Green, W.M.J., et al.: Ultra-compact, low RF power, 10Gb/s silicon Mach-Zehnder modulator. Optics Express 15(25), 17106–17113
36. Lu, Z., et al.: Connection-oriented Multicasting in Wormhole-switched Networks on Chip. In: Proc. of IEEE Computer Society Annual Symposium on VLSI, pp. 205–210 (2006)
37. Pande, P.P., et al.: Performance Evaluation and Design Trade-offs for Network-on-chip Interconnect Architectures. IEEE Transactions on Computers 54(8), 1025–1040 (2005)

38. Ismail, A., Abidi, A.: A 3 to 10GHz LNA Using a Wide-band LC-ladder Matching Network. In: Proc. of IEEE International Solid-State Circuits Conference, February 15-19, pp. 384–534 (2004)
39. Circuits Multi-Projects, http://www.cmp.imag.fr
40. Lojek, B.: Reflectivity of the Silicon Semiconductor Substrate and its Dependence on the Doping Concentration and Intensity of the Irradiation. In: Proc. of the 11th IEEE International Conference on Advanced Thermal Processing of Semiconductors, pp. 215–220 (2003)
41. Hsiao, M.Y.: A class of optimal minimum odd-weight-column SEC-DED codes. IBM J. Res. Dev. 14(4), 395–401 (1970)

Chapter 8
Exploiting Multi–Objective Evolutionary Algorithms for Designing Energy–Efficient Solutions to Data Compression and Node Localization in Wireless Sensor Networks

Francesco Marcelloni and Massimo Vecchio

Abstract. Wireless sensor network (WSN) technology promises to have a high potential to tackle environmental challenges and to monitor and reduce energy and greenhouse gas emissions. Indeed, WSNs have already been successfully employed in applications such as intelligent buildings, smart grids and energy control systems, transportation and logistics, and precision agriculture. All these applications generally require the exchange of a large amount of data and the localization of the sensor nodes. Both these two tasks can be particularly energy–hungry. Since sensor nodes are typically powered by small batteries, appropriate energy saving strategies have to be employed so as to prolong the lifetime of the WSNs and to make their use attractive and effective. To this aim, the study of data compression algorithms suitable for the reduced storage and computational resources of a sensor node, and the exploration of node localization techniques aimed at estimating the positions of all sensor nodes of a WSN from the knowledge of the exact locations of a restricted number of these nodes, have attracted a large interest in the last years. In this chapter, we discuss how multi–objective evolutionary algorithms can successfully be exploited to generate energy–aware data compressors and to solve the node localization problem. Simulation results show that, in both the tasks, the solutions produced by the evolutionary processes outperform the most interesting approaches recently proposed in the literature.

Francesco Marcelloni
Dipartimento di Ingegneria dell'Informazione, University of Pisa, Largo Lucio Lazzarino 1, 56122 Pisa - Italy
e-mail: f.marcelloni@iet.unipi.it

Massimo Vecchio
Departamento de Teoría de la Señal y las Comunicaciones, University of Vigo, C/ Maxwell s/n, 36310 Vigo - Spain
e-mail: massimo@gts.uvigo.es

S.U. Khan et al. (Eds.): Evolutionary Based Solutions for Green Computing, SCI 432, pp. 227–255.
springerlink.com

8.1 Introduction

A Wireless Sensor Network (WSN) is a collection of nodes organized into a cooperative network. Each node is a tiny device composed of three basic units: a processing unit with limited memory and computational power, a sensing unit for data acquisition from the surrounding environment and a communication unit, usually a radio transceiver, to transmit data to a central collection point, denoted sink node or base station. Typically, nodes are powered by small batteries which cannot be generally changed or recharged [1, 10].

WSNs represent nowadays a mature technology impacting on several real world applications. Just as an example, let us consider how WSNs are transforming classical agriculture. The availability of cheap battery–powered sensor nodes, which can be deployed over large areas, is enabling farmers, for instance, to employ electronic guidance aids to direct equipment movements more accurately and to provide precise positioning for all equipment actions and chemical applications. Further, the large amount of different sensors allow collecting measures of agronomic and climatic parameters in real–time, thus making possible to detect undesired conditions and promptly counteract their effects. Generally, once randomly deployed over the area to be monitored, sensor nodes are able to autonomously build ad–hoc network topologies. The ad–hoc configuration represents a key feature in WSNs since it enables easy and fast deployments and provides a certain degree of reconfigurability, reliability and fault–tolerance.

In typical environmental applications of WSN, each sensor node monitors environmental parameters and produces a stream of measures which has to be transmitted from the sensor node itself to the base station, through multi–hop routing communication. It follows that nodes acting as routers have to manage the measures collected by the sensors on board the nodes themselves, and to store and to forward towards the sink both these measures and data coming from other nodes. Since radio communication is typically considered as the main cause of power consumption, the lifetimes of these nodes and consequently of the overall WSN tend to be very short if appropriate strategies aimed at limiting transmission/reception of data as much as possible are not undertaken.

Data compression appears a very appealing and effective tool to achieve this objective [25]. Data compression algorithms may be broadly classified into two classes: lossless and lossy algorithms. Lossless algorithms guarantee the integrity of data during the compression/decompression process. On the contrary, lossy algorithms may generate a loss of information, but generally ensure a higher compression ratio. Unfortunately, the tiny engineering design of the commercially available sensor nodes prevents the deployment of several classical data compression schemes which require an amount of memory and a computational power not available in these nodes [3]. Further, since sensors on board nodes are typically quite cheap, measures of environmental parameters collected by these sensors are often affected by noise. This noise increases the information entropy and therefore hinders lossless compression algorithms to achieve considerable compression ratios. Thus, just to transmit noise, the use of lossless compression algorithms does not allow

increasing the lifetime of the sensor nodes so much. The ideal solution would be to adopt on the sensor node a lossy compression algorithm in which the lost information was just the noise. In this case, we might achieve high compression ratios without losing relevant information. To this aim, we exploit the observation that data typically collected by WSNs are strongly correlated. Thus, differences between consecutive samples are generally quite small. If this does not occur, it is likely that samples are affected by noise. To de–noise and simultaneously compress the samples, we quantize the differences between consecutive samples. Further, to reduce the number of bits required to code these differences, we adopt a Differential Pulse Code Modulation (DPCM) scheme [11]. Of course, different combinations of the quantization process parameters determine different trade–offs between compression performance and information loss. To generate a set of optimal combinations of these parameters, we employ a multi–objective optimization approach.

Multi–objective optimization is the process of simultaneously optimizing two or more objectives subject to certain constraints. For nontrivial multi–objective problems, a unique solution that simultaneously optimizes each objective cannot be determined. Indeed, when searching for solutions, one arrives at a stage in which the further optimization of an objective implies that the other objectives suffer as a result. A tentative solution is called non–dominated or Pareto optimal if it cannot be replaced by another solution which improves an objective without worsening another. More formally, a solution x associated with a performance vector $\mathbf{u}$ dominates a solution y associated with a performance vector $\mathbf{v}$ if and only if, $\forall i \in \{1,...,I\}$, with I the number of objectives, u_i performs better than, or equal to, v_i and $\exists i \in \{1,...,I\}$ such that u_i performs better than v_i, where u_i and v_i are the i–th elements of vectors $\mathbf{u}$ and $\mathbf{v}$, respectively. The set of Pareto–optimal solutions is denoted as Pareto front. Thus, the aim of any multi–objective algorithm is to discover a family of solutions that are a good approximation of the Pareto front [26]. The Multi–Objective Evolutionary Algorithms (MOEAs), employed in this work, achieve this goal by simulating the process of natural evolution. The first purpose of this chapter is to present that the use of an MOEA is an effective approach to the generation of compressors for WSNs by comparing a representative solution of the approximated Pareto front with one of the most effective compressors for WSNs recently proposed in the literature.

Once the compressed information achieves the sink, it is elaborated for the specific considered application. Often, especially in environmental domain, the elaboration process requires to know the locations of the nodes where the single measures have been collected. Although location awareness can be enabled in principle by the use of a Global Positioning System (GPS) on board the sensor node, this solution is not always viable in practice, as the cost and power consumption of GPS receivers are not negligible. In addition, GPS is not well suited to indoor and underground deployments, and the presence of obstacles like dense foliage or tall buildings may impair the outdoor communication with satellites. These limitations have motivated alternative approaches to the problem [23, 30, 31], among which *fine–grained* localization schemes may be considered as the most suitable. In these schemes, only a few nodes of the network (the reference or *anchor* nodes) are endowed with their

exact positions through GPS or manual placement, while all nodes are able to estimate their distances to nearby nodes by using appropriate measurement techniques, such as Received Signal Strength (RSS), Time of Arrival (ToA), Time Difference of Arrival (TDoA) (reviews of these techniques can be found in [23, 30]). Thus, assuming that the coordinates of anchor nodes are known, and exploiting pairwise distance measurements among the nodes, the fine–grained localization schemes aims to determine the positions of all non–anchor nodes by minimizing a cost function (CF) computed as the squared error between the estimated and the corresponding measured inter–node distances. This task is a multivariable nonconvex optimization problem which has been proved to be NP–hard [2] and is therefore rather difficult to solve by using traditional techniques. More, the pairwise distance measurements are invariably corrupted by noise. Finally, even if the distance measurements were accurate, a sufficient condition for the topology to be uniquely localizable is not easily identifiable [6]. We recall that a network is said to be localizable if there exists only one possible geometry compatible with the available data.

In the last years, several efforts have been made to reduce the computational complexity of the nonconvex optimization problem by relaxing it into a convex optimization problem. Loosely speaking, any convex relaxation re–formulation aims to trade off computation time for some accuracy in the final estimate. Thus, relaxed methods result to be well suited to solve the localization problem in large–scale and mobile networks. On the other hand, there are practical applications of WSNs which do not require highly scalable real–time solutions. Consider, for instance, the precision agriculture application we mentioned above: it is obviously preferable to spend some additional time for obtaining an accurate estimate of the node coordinates rather than, for instance, to administer a fertilizer or a pesticide to a wrong zone of the monitored field. Moreover, current existent sensor network testbeds are rarely composed by more than one hundred nodes and their mobility is mainly enabled for security and surveillance applications. Thus, we are interested in localization schemes yielding accurate estimates, although we are well conscious that such schemes may not be suited to applications that demand high scalability and require real–time operation.

The second purpose of this chapter is to show how the nonconvex optimation problem can be tackled by adopting an MOEA that takes concurrently into account during the evolutionary process both the CF and certain topological constraints induced by connectivity considerations. These constraints are especially useful in order to alleviate localizability issues. Indeed, if the network is not localizable, then multiple minima of CF will be present, with only one of them corresponding to the actual geometry of the deployment. Thus, in settings which are close to not being localizable, any localization algorithm will become extremely sensitive to these false minima, resulting in very large localization errors [18, 37]. We present that the use of an MOEA is an effective approach to solve the localization problem by comparing a representative solution of the approximated Pareto front with two of the most performing approaches recently proposed in the literature.

In conclusion, the main aim of this chapter is therefore to discuss how MOEAs can successfully be exploited to tackle two very relevant issues in WSNs: to

generate energy–aware lossy data compressors and to solve the node localization problem. The chapter is organized as follows. Section 8.2 discusses the related works, while Sections 8.3 and 8.4 show the proposed MOEA–based approaches to generate effective data compressors and to solve the localization problem, respectively. In Section 8.5, we describe the MOEAs used in our experiments. Sections 8.6 and 8.7 show some experiments performed by using the proposed approaches. In particular, we highlight how the solutions obtained by the application of the MOEAs outperform some state–of–the–art approaches recently proposed in the literature. Finally, Section 8.8 draws some conclusion.

8.2 Related Works

In this section we briefly review the works related to data compression and node localization in WSN.

8.2.1 Data Compression in WSN

Due to the limited resources available in sensor nodes, data compression in WSNs requires specifically designed algorithms. Two approaches have been adopted in the literature: to distribute the computational cost on the overall network and to enable compression acting at single node independently of the others (see [26] and the references therein). The first approach is natural in cooperative and dense WSNs, where the nodes can collaborate with each other so as to carry out tasks they could not execute alone. Moreover, thanks to the particular topology of these WSNs, data measured by neighboring nodes are correlated: in this scenario it may make sense for the sensor nodes to jointly estimate a largely correlated phenomenon, simply by exchanging messages. The Distributed Source Coding (DSC) scheme proposed in [32, 44] can be considered an exception with respect to the schemes proposed for the first approach. Indeed, unlike the previous methods it does not require inter–node message exchanges. In DSC the sensor nodes simply send their compressed outputs to a central point which is in charge of jointly decoding the encoded streams. Unfortunately, the strongest assumption in all the distributed schemes is that sensor nodes are densely deployed so that their readings are highly correlated with those of their neighbors.

Unlike the distributed approach, the second approach to data compression does not exploit the correlation of the measurements collected from neighboring nodes: each node exploits only its local information for compressing data. This could make the achievement of satisfactory compression ratios more difficult. Further, it could require the execution of a larger number of instructions. In fact, power saving can be achieved only if the execution of the compression algorithm does not require an amount of energy larger than the one saved in reducing communication. Indeed,

after analyzing several families of classic compression algorithms, Barr and Asanović conclude that compression prior to transmission in wireless battery–powered devices may actually cause an overall increase of power consumption, if no energy awareness is introduced [3]. On the other hand, standard compression algorithms are aimed at saving storage and not energy. Thus, appropriate strategies have to be adopted.

Examples of compression techniques applied to the single node adapt some existing dictionary–based compression algorithms to the constraints imposed by the limited resources available on the sensor nodes. For instance, the lossless compression algorithms proposed in [22, 24, 33] are purposely adapted versions of LZ77, Exponential–Golomb code and LZW, respectively. The Lightweight Temporal Compression (LTC) algorithm proposed in [35] is an efficient and simple lossy compression technique for the context of habitat monitoring. LTC introduces a small amount of error into each reading bounded by a control knob: the larger the bound on this error, the greater the saving by compression. Basically LTC is similar to Run Length Encoding (RLE) in the sense that it attempts to represent a long sequence of similar data with a single symbol [34]. The difference with RLE is that while the latter searches for strings of a repeated symbol, LTC searches for linear trends. We will use LTC as comparison since, to the best of our knowledge, LTC is the unique lossy compression algorithm specifically designed to lossy compress data on a single sensor node of a WSN. Further, MOEAs have not been previously applied to generate lossy compressors for WSN.

8.2.2 Node Localization in WSN

The fine–grained localization problem was proved to be an NP–hard problem [2]. In the literature three different approaches can be found to tackle the problem, namely, stochastic optimization, multidimensional scaling, and convex relaxation. The first approach attempts to avoid local minima by resorting to global optimization methods, such as *e.g.* simulated annealing [19]. Multidimensional scaling [9, 17] is a *connectivity–based* technique that, in addition to distance measurements, exploits knowledge about the topology of the network; this information imposes additional constraints on the problem, since nodes within communication range of each other cannot be arbitrarily far apart. The third approach relax the original nonconvex formulation in order to obtain a Semi–Definite Programming (SDP) or a Second–Order Cone Programming (SOCP) problems. Global solutions to these relaxed, convex problems can be then obtained with moderate computational effort [7, 40] and constitute approximate solutions to the original nonconvex problem. In [40] it was shown that the solutions obtained by SOCP relaxation are less accurate than those obtained by SDP relaxation. Moreover, in order to improve the computational efficiency of the original SDP approach proposed in [7], further relaxations have been recently proposed. For instance, in [20, 43] examples of such relaxations were presented, which are able to handle much larger–sized problems, at the expense of an

increase in localization error with respect to the original SDP approach. Basically, all these methods aim to relax the original problem at the modeling level, so as to maintain the sparsity of the graph by limiting the amount of selected edges connected to each sensor or anchor node. Just the original SDP and one of its variants, which exploits a regularization term [7], have been adopted as comparison in the experimental results. To the best of our knowledge, no MOEA–based approach has been proposed in the literature to solve the fine–grained localization problem.

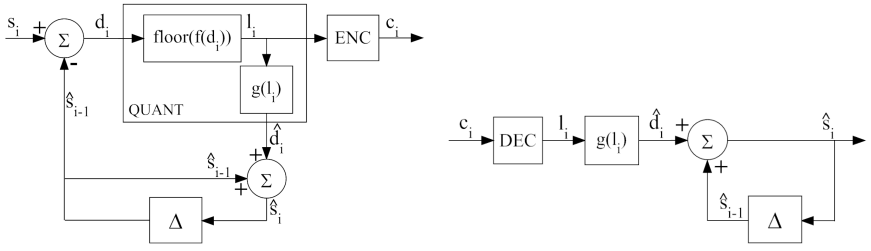
8.3 Data Compression in WSN: An MOEA–Based Solution

In this section we describe our MOEA–based approach to generate energy–aware lossy data compressors in WSNs. In particular, Section 8.3.1 introduces the problem statement by recalling some basic quantization principles. In Sections 8.3.2 we show an overview of our approach. Finally, Section 8.3.3 describes the chromosome representation and the mating operators.

8.3.1 Problem Statement

Environmental data are typically characterized by a high correlation between neighboring samples. For this type of signals, differential compression methods have proved to be very effective [34]. Thus, in order to compress data in WSN we adopt a purposely adapted version of the DPCM scheme, often used for digital audio signal compression. DPCM is a member of the family of differential compression methods.

Figs. 8.1(a) and 8.1(b) show the block diagrams of our compressor and uncompressor, respectively. As regards the compressor, the generic difference d_i is calculated by subtracting the most recent reconstructed value $\hat{s}_{i-1}$ from current sample s_i. To use $\hat{s}_{i-1}$ rather than the original value s_{i-1} avoids the well–known *accumulation of errors* problem [34]. Difference d_i is therefore input to the quantization block *QUANT*. Let $S = \{S_0, \ldots, S_{L-1}\}$ be a set of *cells* S_l, with $l \in [0, L-1]$, which form a disjoint and exhaustive partition of the input domain D (difference domain in our case). Let $C = \{y_0, \ldots, y_{L-1}\}$ be a set of levels $y_l \in S_l$. The $floor(f(\cdot))$ block returns the index l_i of the cell S_{l_i} which d_i belongs to. The index l_i is input to the $g(\cdot)$ block, which computes the quantized difference $\hat{d}_i$, and to the entropy encoder *ENC*, which generates the binary codeword c_i [29]. Since environmental signals are quite smooth and therefore small differences are more probable than large ones, an entropy encoder results to be particularly performing. Indeed, these encoders encode more probable indexes with lower number of bits. In the uncompressor, the codeword c_i is analyzed by the decoding block *DEC* which outputs the index l_i. This index is elaborated by the block $g(\cdot)$ to produce $\hat{d}_i$, which is added to $\hat{s}_{i-1}$ for generating output $\hat{s}_i$.



(a) compressor (b) uncompressor

Fig. 8.1. Block diagrams of (a) the compressor and (b) the uncompressor

The quantization process depends on the number and size of the cells S_l and on the position of the levels $y_l \in S_l$. A small number of wide cells allows achieving a high data reduction, but also generates a high reconstruction error at the decoder. On the other hand, if the number of cells is high, the reconstruction error decreases but the compression ratio also decreases. Thus, a right trade–off between compression and reconstruction error has to be found.

8.3.2 Overview of Our Approach

With the aim of determining a set of optimal quantization parameters, we adopt an MOEA which concurrently optimises two objectives, namely, the entropy H and the mean square error MSE between the quantized and the ideal de–noised samples (we will denote such a measure as MSE^*).

Information entropy H provides an indirect measure of the possible obtainable compression ratios and is defined as:

$$H = -\sum_{l=1}^{L} p_l \cdot \log_2(p_l) \tag{8.1}$$

where p_l is the probability mass function of quantization index l.

MSE quantifies the distortion between two signals and is defined as

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(s_i^{(1)} - s_i^{(2)}\right)^2 \tag{8.2}$$

where N is the number of samples, and $s_i^{(1)}$ and $s_i^{(2)}$ are the two signals to be compared. In particular, in order to measure the loss of information of the reconstructed signal with respect to the ideal (not affected by noise) signal, in MSE^*, $s^{(1)}$ and $s^{(2)}$ are represented by the quantized and the original de–noised samples.

The MOEA is applied to a small set (*training set*) of samples which are representative of the overall signal. First, the samples are de–noised using some standard technique. Then, the MOEA is executed by using the original and the de–noised samples. At the end of the optimization process, we obtain a family of non–dominated solutions with respect to the two objectives. The encoder encodes the indexes which identify the cells by exploiting a dictionary generated by using the Huffman's algorithm [16]. This algorithm provides a systematic method of designing binary codes with the smallest possible average length for a given set of symbol occurrence frequencies. To determine an estimate of these frequencies, we exploit again the training set. For the specific quantizer, we compute the probability with which each quantization index occurs when quantizing the differences between consecutive samples of the training set. Once the binary codeword representation of each quantization index has been computed and stored in the sensor node, the encoding phase reduces to a look–up table consultation.

Finally, to assess the performances of a compression algorithm using a specific quantizer we use the compression ratio CR defined as:

$$CR = \left(1 - \frac{comSize}{origSize}\right) \times 100\%, \tag{8.3}$$

where *comSize* and *origSize* represent the sizes of the compressed and original bitstreams, respectively.

8.3.3 Chromosome Coding and Mating Operators

Each chromosome is composed of $2 \times L_{MAX} + 1$ integer genes. The first gene (N) establishes the number of cells to be considered in the positive domain (we decided to consider quantizers with symmetric characteristic with respect to the origin of the axes). Gene N can assume values in $[1, L_{MAX}]$, where L_{MAX} is set by the user and identify the maximum possible number of cells. We introduced L_{MAX} to limit the search space. On the other side, since the quantization process is applied to the differences between consecutive samples of environmental signals and these differences are not typically very high, to fix a bound on the number of cells speeds up the execution of the MOEA without affecting the goodness of the final results. The second gene (DZ) represents the distance between the level 0 and the upper bound a_0 of the zero–cell. When a good rate–distortion performance is requested to a quantizer, the zero–cell width is usually treated individually. Since each input within the zero–cell is quantized to 0, this cell is often called *dead zone*. The subsequent $L_{MAX} - 1$ pairs C_l of genes codify, for each cell S_l, the distance between the level y_l and the lower threshold $a_{l-1} + 1$ of the cell[1], and the distance between the level y_l and the upper threshold a_l of the cell. The last gene SR represents the distance between

[1] Since we consider samples generated by an analog–to–digital converter, they are represented as integer values. Thus, the generic cell $S_l = [a_{l-1} + 1, a_l]$.

the level $y_{L_{MAX}}$ and the lower threshold $a_{L_{MAX}-1}+1$ of the cell $S_{L_{MAX}}$. Every input value larger than the threshold $a_{L_{MAX}-1}+1$ of the last cell is quantized to the last level (saturation region). It follows that, except for the first gene, the remaining genes describe the cells S_l by deciding the position of the level and the position of the lower and upper thresholds $[a_{l-1}+1, a_l]$. All the genes, except for gene N, assume values in $[0, D/4]$, where D is the difference domain.

As mating operators, we have applied the classical one–point crossover operator and a gene mutation operator [28]. The common point of the one–point crossover is chosen by extracting randomly a number in $(1, 2 \times L_{MAX}+1)$. The crossover operator is applied with probability P_X; the mutation operator is applied whenever the crossover is not applied. After applying any mating operator, we perform the following check: we count the number NA of cells which are actually activated by the training set. If $N > NA$, then we set N to NA. Thus, we make the chromosome consistent with the computation of the entropy (indeed, cells, which are activated by no sample, have entropy equal to 0 and therefore do not influence the entropy).

8.4 Node Localization in WSN: An MOEA–Based Solution

In this section we describe our MOEA–based approach to solve the fine–grained localization problem in WSNs. In particular, Section 8.4.1 introduces the problem statement. In Section 8.4.2 we show an overview of our approach by discussing the topology–based constraints and the objectives used to evaluate the goodness of the topology estimations. Finally, Section 8.4.3 describes the chromosome representation and the mating operators.

8.4.1 Problem Statement

Let us suppose that the WSN has n nodes deployed in $T = [0,1] \times [0,1] \subset \mathbb{R}^2$ and that nodes 1 through m, with $m < n$, are anchor nodes whose coordinates $\mathbf{p}_i = (x_i, y_i) \in T$, $i = 1, \ldots, m$, are known. Further, we assume that two nodes, say i and j, can communicate with each other if and only if the $r_{ij} \leq R$, where $r_{ij} = \left\| \mathbf{p}_i - \mathbf{p}_j \right\|$ is the actual distance between nodes i and j ($\|\cdot\|$ denotes the Euclidean norm) and R is the connectivity radius (this assumption, denoted as disk model, is often made in the literature, though it results to be a coarse approximation of the reality). Finally, we suppose that if nodes i and j are within the connectivity range of each other, then their inter–node distance d_{ij} can be estimated by using some measurement technique (see Section 8.1) and can be modeled as

$$d_{ij} = r_{ij} + e_{ij} \tag{8.4}$$

Similar to [5–7, 19], we assume that measurement errors e_{ij} follow a zero–mean Gaussian distribution with variance σ^2, and that the random variables e_{ij} and e_{kl} are statistically independent for $(i, j) \neq (k, l)$.

We refer to nodes j such that $r_{ij} \leq R$ as *first–level neighbors* of node i. Let

$$N_i = \left\{ j \in 1 \ldots n, j \neq i : r_{ij} \leq R \right\} \tag{8.5}$$

$$\overline{N}_i = \left\{ j \in 1 \ldots n, j \neq i : r_{ij} > R \right\} \tag{8.6}$$

be the set of the first–level neighbors of node i and its complement, respectively. We assume that sets N_i and $\overline{N}_i$ are known for all $i = 1, \ldots, n$. This is a reasonable assumption, since each node can easily determine which other nodes it can communicate with. Given the positions of the m anchor nodes and the estimated inter–node distances d_{ij}, we aim to determine the positions of the non–anchor nodes by exploiting an MOEA–based approach.

8.4.2 Overview of Our Approach

Based on the position of the anchor nodes and the connectivity ranges, we can classify a non–anchor node i into three different classes [42]:

- Class 1: at least an anchor node is a first–level neighbor of i
- Class 2: i does not belong to Class 1 and at least a non–anchor node, which has an anchor node as first–level neighbor, is a first–level neighbor of i
- Class 3: i belongs to neither Class 1 nor Class 2.

The membership of a non–anchor node to one of the three classes allows restricting the space where the node can be located. This information can be exploited both in the generation of the initial population and in the execution of the mating operators so as to speed up the execution of the MOEA by avoiding the generation of solutions which certainly cannot be optimal.

Each estimation is associated with a vector of two elements, which represent the values of the two objective functions CF and CV. Let $\hat{\mathbf{p}}_i = (\hat{x}_i, \hat{y}_i), i = m+1, \ldots, n$ be the estimated positions of the non–anchor nodes i.

CF is defined as

$$CF = \sum_{i=m+1}^{n} \left(\sum_{j \in N_i} \left(\hat{d}_{ij} - d_{ij} \right)^2 \right), \tag{8.7}$$

where d_{ij} and $\hat{d}_{ij}$ represent the measured and the estimated distances between nodes i and j.

CV counts the number of connectivity constraints which are not satisfied by the candidate geometry, and is defined as

$$CV = \sum_{i=1}^{n} \left(\sum_{j \in N_i} \delta_{ij} + \sum_{j \in \overline{N}_i} \left(1 - \delta_{ij} \right) \right), \tag{8.8}$$

where $\delta_{ij} = 1$ if $\hat{d}_{ij} > R$, and 0 otherwise.

In order to evaluate the accuracy of the estimates, we consider the *normalized localization error* (*NLE*), defined as

$$NLE = \frac{1}{R}\sqrt{\frac{1}{(n-m)}\sum_{i=m+1}^{n}\left(\|\mathbf{p}_i - \hat{\mathbf{p}}_i\|^2\right)} \times 100\%. \tag{8.9}$$

Thus, assuming that the estimate is unbiased, *NLE* can be interpreted as the ratio of the standard deviation to the connectivity radius.

8.4.3 Chromosome Coding and Mating Operators

Each chromosome encodes the positions of all non–anchor nodes in the network. Thus, each chromosome consists of $n-m$ pairs of real numbers, where each pair represents the coordinates $\hat{x}$ and $\hat{y}$ of a non–anchor node. The variation range of each coordinate is bounded by the geometrical constraints previously described. We enforce compliance with these constraints in the initial population. Further, whenever mutations are applied during the evolutionary process, only mutated individuals satisfying these constraints are generated.

The first mutation operator, denoted *node mutation* operator, performs a uniform–like mutation: the position of each non–anchor sensor node is mutated with probability $P_U = 1/(n-m)$. Positions are randomly generated within the geometrical constraints imposed on the specific node location. The second mutation operator, denoted *neighborhood mutation* operator, is applied when the first operator is not selected. The neighborhood mutation operator mutates, with probability P_U, the position of each non–anchor sensor node within the geometrical constraints determined for the specific node, but unlike the first operator, it applies the same rigid translation, which has brought the mutated node i from the pre–mutation position to the post–mutation position, to the neighbors of i with a certain probability (*rigid translation probability*). As we have already discussed in [42], this operator results to be particularly suitable for dealing with specific topological configurations.

8.5 Multi–Objective Evolutionary Algorithms

Multi–objective evolutionary optimization has been investigated by several authors in recent years [45] and several different algorithms have been proposed. Some of the most popular among these algorithms are the Strength Pareto Evolutionary Algorithm (SPEA) [47] and its evolution (SPEA2) [46], the Niched Pareto Genetic Algorithm (NPGA) [15], the different versions of the Pareto Archived Evolution Strategy (PAES) [21], and the Non–dominated Sorting Genetic Algorithm (NSGA) [39] and its evolution (NSGA–II) [12]. In the following, we briefly introduce NSGA–II and

PAES, which have proved to be particularly effective in generating energy–aware lossy data compressors and solving the node localization problem, respectively.

8.5.1 NSGA–II

NSGA–II is a population–based multi–objective genetic algorithm. Each individual of the population is associated with a rank equal to its non–dominance level (1 for the best level, 2 for the next–best level, and so on). To determine the non–dominance level (and consequently the rank), for each individual p, the number n_p of individuals that dominate p and the set S_p of individuals dominated by p are computed. All individuals with $n_p = 0$ belong to the best non–dominance level associated with rank 1. To determine the individuals associated with rank 2, for each solution p with rank 1, each member q of the set S_p is visited and n_q is decreased by one. If n_q becomes zero, then q belongs to the non–dominance level associated with rank 2. The procedure is repeated for each solution with rank 2, rank 3 and so on until all fronts are identified. NSGA–II starts from an initial random population P_0 of N_{pop} individuals sorted based on the non–dominance. At each iteration t, an offspring population Q_t of size N_{pop} is generated by selecting mating individuals through the binary tournament selection, and by applying the crossover and mutation operators. Parent population P_t and offspring population Q_t are combined so as to generate a new population $P_t^{ext} = P_t \cup Q_t$. Then, a rank is assigned to each individual in P_t^{ext}. Finally, P_t^{ext} is split into different non–dominated fronts, one for each different rank. Within each front, a specific crowding measure, which represents the sum of the distances to the closest individual along each objective, is used to define an ordering among individuals: in order to cover the overall objective space, individuals with large crowding distance are preferred to individuals with small crowding distance. The new parent population P_{t+1} is generated by selecting the best N_{pop} individuals (considering first the ordering among the fronts and then among the individuals) from P_t^{ext}.

8.5.2 PAES

PAES is an archive–based MOEA. It maintains a single current solution c, and, at each iteration, produces a new solution m from c, by using only mutation operators. Then, m is compared with c. Three different cases can arise:

I. c dominates m: m is discarded;
II. m dominates c: m is inserted into the archive and possible solutions in the archive dominated by m are removed; m replaces c in the role of current solution;
III. neither condition is satisfied: m is added to the archive only if it is dominated by no solution contained in the archive; m replaces c in the role of current solution

only if m belongs to a region with a crowding degree smaller than, or equal to, the region of c.

The crowding degree is computed by firstly dividing the space where the solutions of the archive lie into a number (*numReg*) of equally sized regions and then by counting the solutions that belong to the regions. The number of these solutions determines the crowding degree of a region. This approach tends to prefer solutions which belong to poorly crowded regions, so as to guarantee a uniform distribution of the solutions along the Pareto front.

PAES terminates after a given number *maxEvals* of evaluations. The candidate solution acceptance strategy generates an archive which contains only non–dominated solutions. On PAES termination, the archive (at most *archSize*) includes the set of solutions which are an approximation of the Pareto front. At the beginning, the archive is empty and the first current solution is randomly generated.

8.6 Experimental Results for the Data Compression Approach

In this section, first we introduce the experimental setup used in the experiments. Then, we show the results obtained by applying different MOEAs for generating energy–aware lossy data compressors with different trade–offs between MSE^* and entropy H on a given training set. By statistically analyze the distribution of the hypervolumes obtained by the different MOEAs, we show that NSGA–II outperforms the other MOEAs. Thus, we select NSGA–II as MOEA for our approach and apply it to the other datasets different from the training set. Finally, we compare the results obtained by NSGA–II with the ones achieved by a state–of–the–art algorithm, namely LTC, recently proposed in the literature. We will show that our approach can achieve significant compression ratios despite negligible reconstruction errors and outperforms LTC in terms of compression rate and complexity.

8.6.1 Experimental Setup

For the sake of brevity, we will discuss the application of our approach to only samples collected by temperature sensors. We employed the public data of the SensorScope deployments [4]. We chose to adopt public domain datasets rather than to generate data by ourselves to make the assessment as fair as possible. The WSNs adopted in the deployments employ a TinyNode node type [14], which uses a TI MSP430 microcontroller, a Xemics XE1205 radio and a Sensirion SHT75 sensor module [36]. This module is a single chip which includes a bandgap temperature sensor, coupled to a 14–bit ADC and a serial interface circuit. The Sensirion SHT75 can sense air temperature in the $[-20^oC, +60^oC]$ range. Each ADC output raw_t is represented with resolution of 14 bits and normally converted into a measure t in Celsius degrees (^oC) as described in [36]. The datasets collected in the deployment

contain measures t. On the other hand, our algorithm works on raw_t data. Thus, before applying the algorithm, we extracted raw_t from t, by using the inverted versions of the conversion functions in [36].

Since our approach requires a training set for the execution of the MOEAs, we chose a node, namely node 101, of the Fish–Net deployment (one of the SensorScope deployments), and used $N = 5040$ temperature sample data collected from August 9 2007 to August 16 2007 (we will refer to this dataset with the symbolic name FN). Since the samples are collected at a frequency of 1 sample each 2 minutes, the training set corresponds to the monitoring of a week. The extracted portion of the original signal is first de–noised, using the wavelet shrinkage and thresholding method proposed in [13]. In particular, we have used the Symmlet 8 wavelet, a level of decomposition equal to 5 and the soft universal thresholding rule for thresholding the detail coefficients at each level. The de–noising process has been performed by using standard Matlab built–in functions.

8.6.2 Selecting an MOEA for the Specific Problem

As a preliminary step, in order to choose the best MOEA to tackle the specific problem, we have executed 50 independent runs of NSGA–II, SPEA2 and PAES and have compared the three MOEAs by using the hypervolume, whose features and properties have been discussed in detail in [48].

We have fixed $L_{MAX} = 16$ and the maximum number T_{MAX} of evaluations to 10^6. Further, when the mutation is applied, gene N is mutated with probability 0.5. If gene N is not mutated, then one of the remaining genes is randomly selected. For NSGA–II and SPEA2 we adopted a population size $N_{pop} = 100$ and a crossover probability $P_X = 0.8$. For PAES, we used a number of regions $numReg = 5$ and an archive size $archSize = 100$.

In two–objective optimization problems, the hypervolume measures the area of the portion of the objective space that is weakly dominated by a Pareto front approximation. To compare the three MOEAs we have used the hypervolume indicator (HV) computed in the performance assessment package provided in the PISA toolkit [8]. Here, the lower the value of HV is, the higher the quality of the corresponding algorithm is.

Fig 8.2 shows the box plots of the HV indicator for the three MOEAs. From the analysis of the three box–plots we can deduce that NSGA–II outperforms the other two MOEAs in the specific problem.

In order to evaluate the stability of NSGA–II in the specific problem, in [27] we have also performed an analysis of the distribution of the fronts generated in 50 trials on the $MSE^* - H$ plane. We have observed that the fronts generated in the different trials are quite close to each other, thus confirming that the algorithm is quite stable. In Fig. 8.3 we show the final Pareto front obtained in one of the trials. We can observe that the front is quite wide and the solutions are characterized by a good trade–off between H and MSE^*.

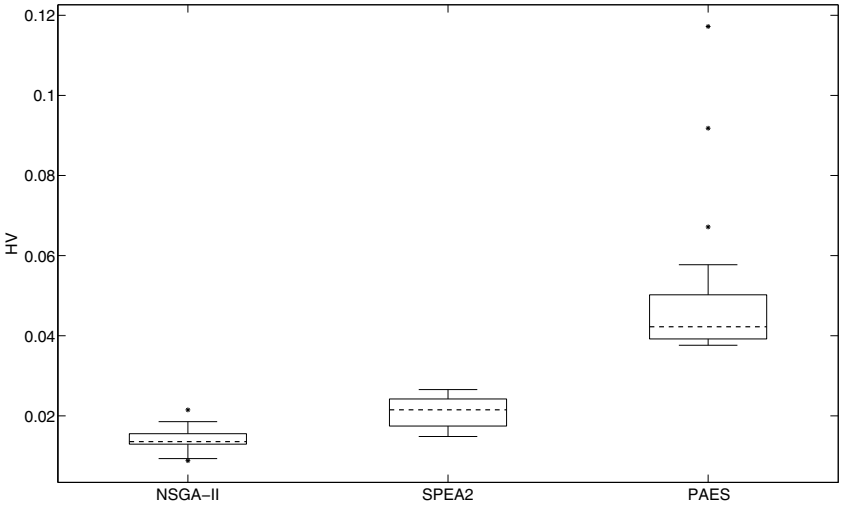


Fig. 8.2. Box plots of the hypervolume indicator HV for NSGA–II, SPEA2 and PAES

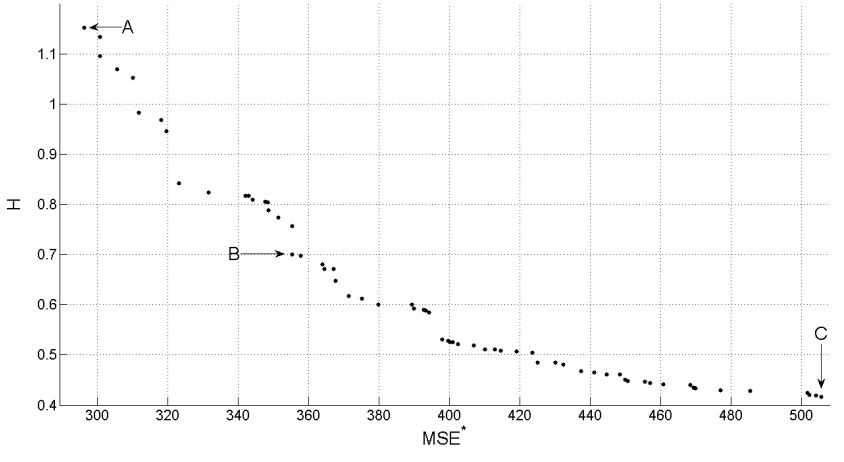


Fig. 8.3. An example of Pareto front approximation

8.6.3 Experimental Results

We have tested our approach with NSGA–II by using three temperature datasets from two SensorScope deployments [4], namely Grand–St–Bernard and Le Gènèpi deployments. Table 8.1 summarizes the main characteristics of the datasets (for completeness, we have shown in the table also the characteristics of the training set). In the following we will refer to the mentioned datasets by using their symbolic names. In particular, we have computed the temperature range ([*min*,*max*]),

the information entropy H of the samples and the MSE_{REF}, which represents the MSE between the original dataset and its de–noised version. By analyzing the temperature ranges of the different sensors, we realize that sensors measure both very cold and warm temperatures. We expressly chose these datasets to show that our approach does not depend on the specific temperature values measured by the sensors.

Table 8.1. Main characteristics of the datasets

Symbolic name	Deployment name	Node ID	num. of samples	Time interval (mm/dd/yyyy) From	To	$[min, max]$	H	MSE_{REF}
FN	Fish–Net	101	5040	08/09/2007	08/16/2007	[7.7, 26.51]	10.01	349.51
$GSB1$	Gr.St Bernard	31	2160	10/04/2007	10/06/2007	[4.16, 14.32]	8.95	1417.83
$GSB2$	Gr.St Bernard	31	2160	10/20/2007	10/22/2007	[−11, 0.75]	9.15	1615.99
LG	Le Gènèpi	2	4320	09/25/2007	10/30/2007	[−4.66, 8.89]	9.42	1251.04

To perform an accurate analysis of some solution, we selected from the front in Fig. 8.3 three significant quantizers: solutions (A) and (C) characterized by, respectively, the highest H and MSE^*, and solution (B) characterized by a good trade–off between H and MSE^*. Solutions (A), (B) and (C) correspond to the quantization rules represented in Table 8.2. Here, $\max_D$ denotes the maximum possible value of the differences d_i in domain D.

Table 8.2. Quantization rules for solutions (A), (B) and (C)

l	cell	solution (A)	solution (B)	solution (C)
0	S_0	$[0, 11]$	$[0, 29]$	$[0, 32]$
	y_0	0	0	0
1	S_1	$[12, \max_D]$	$[30, 62]$	$[33, 80]$
	y_1	12	32	49
2	S_2	–	$[63, \max_D]$	$[81, 87]$
	y_2	–	71	85
3	S_3	–	–	$[88, \max_D]$
	y_3	–	–	120

Table 8.3 shows the values of CR, MSE^*, MSE_N and MSE_D obtained by the three selected quantizers on FN, $GSB1$, $GSB2$ and LG datasets. MSE_N and MSE_D are, respectively, the MSE computed between the noise reconstructed and noise original signals, and between the de–noised reconstructed and de–noised original signals.

We can observe that all solutions achieve good trade–offs between compression ratios and MSEs. Further, there do not exist considerable differences between the results obtained in the training set (FN) and the ones achieved in the other three

Table 8.3. Results obtained by solutions (A), (B) and (C) on the four datasets

Dataset	Solution	CR	MSE^*	MSE_N	MSE_D
FN	A	91.94	296.20	141.52	49.19
	B	92.90	397.89	193.37	81.00
	C	93.12	505.69	232.92	63.07
$GSB1$	A	90.69	685.38	799.64	106.33
	B	91.41	1286.82	252.69	30.56
	C	91.76	1467.10	269.77	36.94
$GSB2$	A	89.78	863.21	917.18	112.40
	B	90.05	1350.20	252.70	20.89
	C	90.25	1685.92	253.25	19.16
LG	A	90.87	473.70	909.27	99.36
	B	91.46	1027.95	292.08	32.52
	C	91.50	1320.84	255.50	36.38

datasets. This result could be considered enough surprising. Indeed, we highlight that both the optimization and the Huffman's algorithm were executed using a signal collected by a node in a different deployment (*i.e.*, FN). Thus, the $GSB1$, $GSB2$ and LG datasets are completely unknown to the compression scheme. We are conscious that the procedure adopted for the training set could have been exhaustively applied also to the other datasets, in order to find ad–hoc solutions for the particular deployment. On the other hand, the three temperature datasets were collected, though in different places and times, by the same sensor nodes with the same type of temperature sensor and the same sampling frequency.

In order to assess the effectiveness of the solutions generated by NSGA–II with respect to purposely–defined compressors recently proposed in the literature, we select solution (B), which represents a good trade–off between compression ratios and MSE^*, and compare its performances against the LTC algorithm introduced in [35].

8.6.4 Comparison with LTC

LTC generates a set of line segments which form a piecewise continuous function. This function approximates the original dataset in such a way that no original sample is farther than a fixed error e from the closest line segment. Thus, before executing the LTC algorithm, we have to set error e. We choose e as a percentage of the Sensor Manufactured Error (SME). From the Sensirion SHT75 sensor data sheet [36], we have $SME = \pm 0.3^oC$ for temperature. To analyze the trend of the CRs with respect to increasing values of e, we varied e from 0% to 230% of the SME with step 10%.

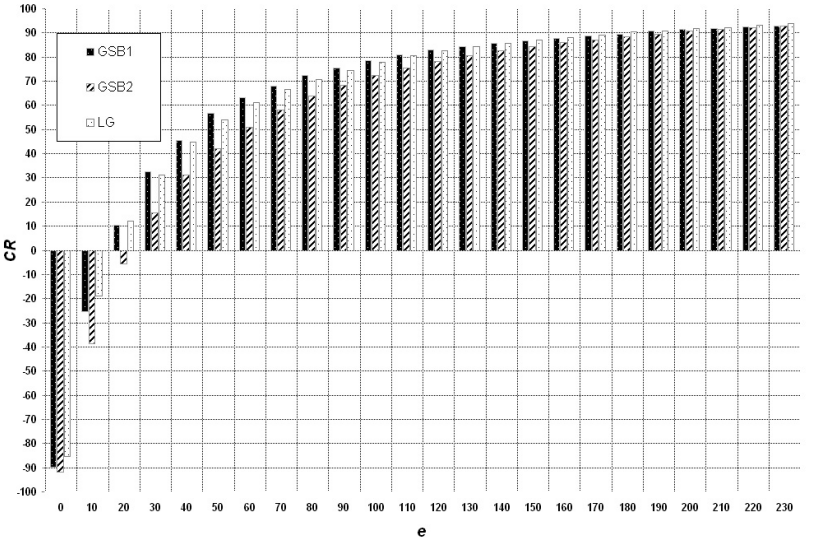


Fig. 8.4. Compression ratios obtained by the LTC algorithm for different error values on the three datasets

Figs. 8.4–8.7 show, for all the three datasets, the trends of CR, MSE^*, MSE_N and MSE_D, respectively, obtained by the LTC algorithm for different error values. Due to the particular approach based on linear approximation used in LTC, we observe that the lossless version of LTC (corresponding to $e = 0$) generates a negative CR, that is, the size of the compressed data is larger than the original data. Further, we note that interesting CRs are obtained only despite relevant compression errors. By comparing Table 8.3, where we have shown the CRs obtained by our algorithm, with Figs. 8.4–8.7, we can observe that the LTC algorithm can achieve the same CRs as our algorithm, but despite quite high MSEs. For instance, our algorithm achieves a CR equal to 91.41 for $GSB1$. From Fig. 8.4, we can observe that, in order to achieve similar CRs, the LTC algorithm should be executed with a value of e between 200% and 210% of the SME. From Fig. 8.5, we can deduce that these values of e lead to have an MSE^* between 1386.46 and 1335.80 against an $MSE^* = 1286.81$ for our algorithm. Similar considerations can be made on MSE_N and MSE_D and Fig. 8.6 and Fig. 8.7. Table 8.4 shows the correspondences between the CR achieved by our algorithm, and the CR, MSE^*, MSE_N and MSE_D obtained by LTC on the three datasets. By comparing Table 8.4 with Table 8.3, we can observe that our algorithm obtains lower MSEs than LTC in correspondence to equal CRs (except for $GSB2$ and MSE^*). For example, for the $GSB1$ and CR equal to 91.41, MSE_N (MSE_D) is between 1015.92 (318.83) and 1076.30 (348.43) for LTC, whereas $MSE_N = 252.69$ ($MSE_D = 30.56$) for our algorithm.

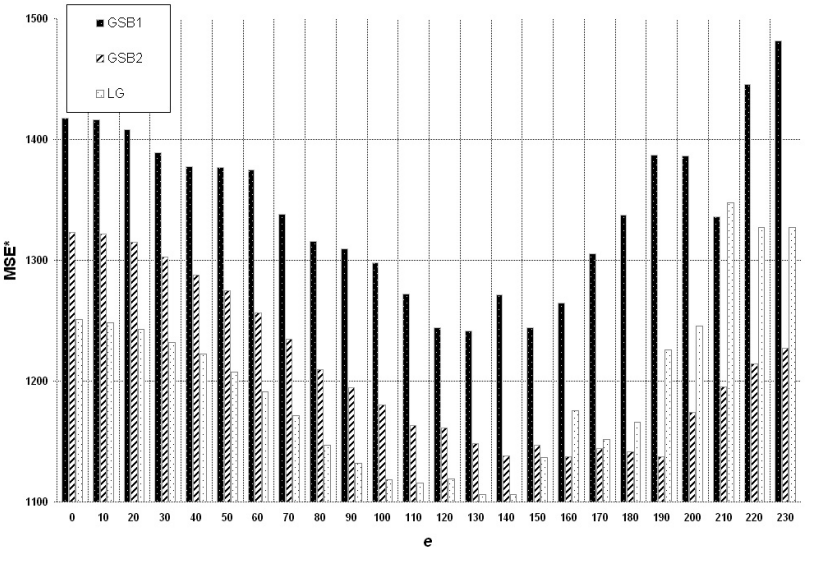


Fig. 8.5. MSE^*s obtained by the LTC algorithm for different error values on the three datasets

Table 8.4. Correspondences between CRs achieved by our algorithm, and CRs and MSEs achieved by LTC on the three datasets

Dataset	CR	e $[min,max]$	CR $[min,max]$	MSE^* $[min,max]$	MSE_N $[min,max]$	MSE_D $[min,max]$
GSB1	91.41	[200,210]	[91.39,91.94]	[1386.46,1335.80]	[1015.92,1076.30]	[318.83,348.43]
GSB2	90.05	[190,200]	[89.47,90.68]	[1137.23,1173.90]	[899.90,1005.65]	[203.62,246.59]
LG	91.46	[190,200]	[90.88,91.81]	[1225.95,1245.53]	[926.28,976.78]	[370.01,384.54]

Since the compressors have been deployed on a node with reduced computational and memory capabilities, the comparison cannot only take compression ratio and MSE into consideration, but has also to consider complexity. To this aim, we have performed a comparative analysis on the number of instructions required by our compressor and by LTC exploiting the Sim–It Arm simulator [38]. Sim–It Arm is an instruction–set simulator that runs both system–level and user–level ARM programs. For LTC, we have set e to the left extremes of the e intervals in Table 8.4 (we recall that the left extremes are the most favorable cases for the LTC algorithm). We have measured that, on average, our algorithm executes 6.06 instructions for each saved bit against 40.99 executed by LTC. Thus, we can conclude that, though our algorithm achieves lower MSEs at the same bitrate as LTC, it requires a lower number of instructions.

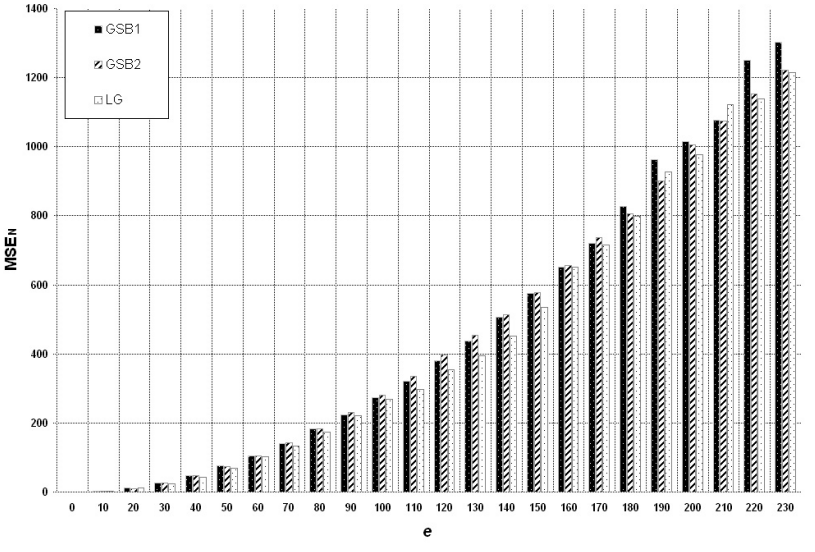


Fig. 8.6. MSE_Ns obtained by the LTC algorithm for different error values on the three datasets

8.7 Experimental Results for the Node Localization Approach

In this section, first we introduce the experimental setup used in the experiments. Then, we show the results obtained by applying PAES for solving the node localization problem and compare these results with the ones achieved by two state–of–the–art algorithms recently proposed in the literature. We highlight that, in all the experiments, our approach achieves considerable accuracies, thus manifesting its effectiveness and stability, and outperforms on average the other algorithms. PAES was chosen as MOEA after a long experimentation in which we compared PAES with other four multi–objective evolutionary approaches, including NSGA–II, on the same type of problem. The comparison has been shown in one of our recent papers [41] and therefore we will not further discuss it here.

8.7.1 Experimental Setup

We have built different network topologies by randomly placing 200 nodes with a uniform distribution in T. We have varied the percentage of anchor nodes to 8%, 10% and 12% (thus each topology consists of 16, 20 and 24 anchor nodes and 184, 180 and 176 non–anchor nodes, respectively). Further, we have varied the connectivity radius R in the interval $[0.11, 0.16]$ with step 0.01. The distance measurements

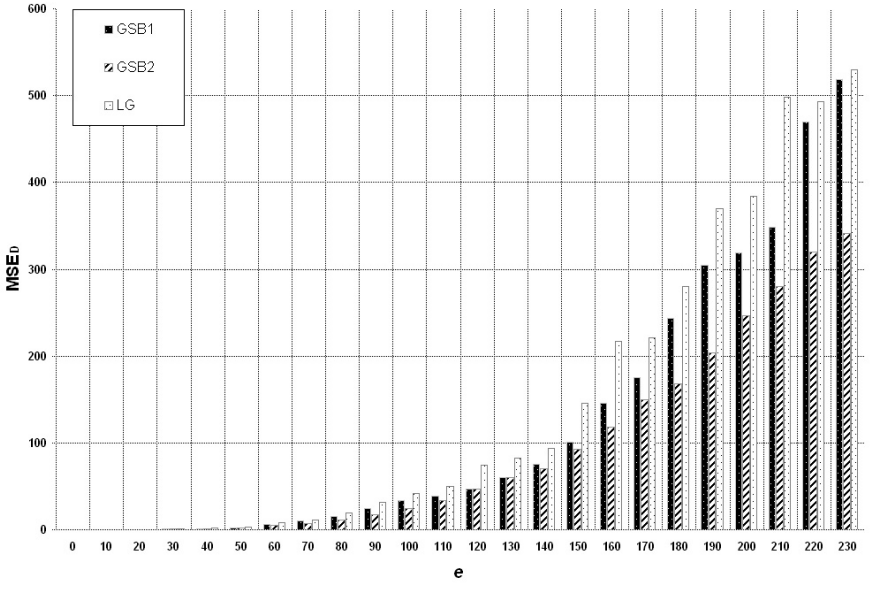


Fig. 8.7. MSE_Ds obtained by the LTC algorithm for different error values on the three datasets

between neighboring nodes were generated according to the model (8.4). We assume that these distance estimates are derived from RSS measurements, which are commonly affected by log–normal shadowing with standard deviation of the errors proportional to the actual range r_{ij} [30]. Thus, the variance of e_{ij} is given by $\sigma^2 = \alpha^2 r_{ij}^2$. A value of $\alpha = 0.1$ was used in the simulations.

For each value of R, 10 random network topologies were generated. After this step, different scenarios were built by varying the percentage of anchor nodes. Thus, we were able to obtain a better control on the effects of both the different connectivity radii and the different percentage of anchor nodes on the normalized localization error. Table 8.5 shows the average values of some network indicators, namely the number of first–level anchor and non–anchor neighbor nodes, the percentage of anchor nodes, the percentage of non–anchor nodes classified in Class 1 and Class 2 (the percentage of nodes in Class 3 can be easily deduced from the first two), the percentage of non–anchor nodes with no anchor node in their neighborhoods and the percentage of non–anchor nodes having at least 3 anchor nodes in their neighborhoods. The analysis of Table 8.5 reveals that, when the connectivity radius and the percentage of anchor nodes are low, the localization problem becomes very complex: indeed, with a connectivity radius $R = 0.11$ and 8% of anchor–nodes, only a small fraction of non–anchor nodes can rely on 3 or more anchor neighbors (2.28%), while more than the half of them (57.72%) are in communication with no anchor node. Moreover it is worth noting that, even when the connectivity radius is increased to 0.16 and the percentage of anchor nodes to 12%, the average percentage

Table 8.5. Average values of several main network indicators for different connectivity radii and percentages of anchor nodes

R	number of first–level neighbors	anchor (%)	Class 1 (%)	Class 2 (%)	0 anchor (%)	3(or more) anchors (%)
0.11	6.86	8	42.28	31.68	57.72	2.28
		10	50.44	32.72	49.56	3.56
		12	56.02	30.68	43.98	5.97
0.12	8.09	8	51.03	31.85	48.97	2.12
		10	59.67	29.22	40.33	4.50
		12	66.48	25.80	33.52	5.85
0.13	9.41	8	58.04	32.72	41.96	2.17
		10	65.28	29.72	34.72	5.33
		12	69.72	26.14	30.28	9.43
0.14	10.84	8	56.52	30.43	43.48	5.98
		10	67.28	25.94	32.72	8.94
		12	75.06	20.68	24.94	14.20
0.15	12.28	8	62.01	29.35	37.99	7.66
		10	70.94	24.11	29.06	12.78
		12	75.74	21.42	24.26	17.16
0.16	14.16	8	67.83	26.20	32.17	11.09
		10	74.17	22.72	25.83	17.22
		12	81.36	16.76	18.64	24.55

of non–anchor nodes with no anchor neighbor is not negligible (18.64%), while the average percentage of non–anchor nodes with 3 or more anchor neighbors is still significantly low (24.55%).

8.7.2 Experimental Results and Comparisons

For each scenario, 15 trials of PAES were executed. We set $archSize = 20$, $numReg = 5$, $maxEvals = 4 \times 10^5$. The first mutation operator is applied with probability 0.9 and the rigid translation probability is set to 0.3. In [42], we have already verified that the fronts are close to each other, thus confirming the stability of the approach. Further, we have shown by using a Wilcoxon test that no statistical difference exists in terms of NLE among the solutions in the final Pareto front approximations. Thus, we can select any solution in order to perform a comparison with other localization techniques. For the sake of brevity, we consider a unique solution, namely the solution characterized by the lowest value of CF, and compare the performances obtained by such solutions in the different topologies against the most accurate convex relaxation approach, *i.e.* the original full–SDP formulation proposed in [7] (we will refer to this method as FSDP). It should be mentioned that FSDP may still incur

significant estimation errors, and that a regularized version (referred to as FSDPr) and a gradient–descent refinement technique have been proposed in [6] and [5], respectively, in order to improve its performance.

FSDPr adds a regularization term to the objective function in order to reduce the tendency of SDP solutions to have points crowded together, which occurs after the last step of projecting the high–rank SDP solution back onto the two–dimensional plane. Thus, the regularization term suggested in [6] penalizes small node separations. The main issue in FSDPr is the choice of the regularization term (λ). To this aim, we have adopted the following heuristic strategy. Given a scenario, we first solve the non–regularized problem and then exploit the non–regularized solution to compute the upper bound of λ (λ^*), as suggested in [6]. λ^* is used as starting value in the main tuning loop, where the regularized problem is solved; if the current regularized solution is not feasible, then the current value of λ is divided by two and the new regularized problem is solved again, until a feasible solution is obtained or a maximum number of tries (MAX_TRIES) is reached. In the latter case, the regularized solution coincides with the non–regularized one (*i.e.* $\lambda = 0$). In our experiments we have fixed $MAX\_TRIES = 5$.

Finally, the goal of gradient–based refinements is to improve the final estimation given by a localization algorithm. Since gradient–based methods generally do not deliver a global optimal solution when the problem is not convex, this technique can be applied as a fine–tuning phase once an approximation to the global solution has been found [5, 6]. Thus, the technique can be applied to any localization method.

In Figs. 8.8–8.10, we have plotted as solid lines the average *NLE*s obtained by the FSDP, FSDPr and PAES algorithms versus the six values of radius R used in the experiments. Further, we have shown as dotted lines the average *NLE*s obtained by applying the gradient refinement described in [5, 6] to the final solutions computed by the three algorithms.

The analysis of the figures highlights that, as expected, FSDPr slightly outperforms FSDP. Further, we observe that the gradient refinement is able to improve the estimation only when it is already sufficiently accurate. Indeed, if we consider the solutions generated by FSDP and FSDPr for the lowest connectivity radii ($R = 0.11$ and $R = 0.12$ for FSDP, and $R = 0.11$ for FSDPr), we realize that the gradient method is unable to perturb the estimation out of the reached local minimum. On the contrary, when the solutions are characterised by a low *NLE*, the gradient method is able to improve them. In particular, the almost constant gap between the solid lines and the dotted lines for PAES suggests a stable improvement introduced by the refinement phase.

The *NLE*s obtained by PAES are comparable to those obtained by FSDPr+REF when the connectivity radii are sufficiently high ($R \geq 0.14$), while they are slightly lower when $R < 0.14$. Further, in all the experiments, PAES+REF considerably outperforms FSDPr+REF. For example, when the percentage of anchor nodes is 8% and the connectivity radius is 0.11, from Fig. 8.8 we can derive that PAES+REF provides estimations which are on average 36.57% more accurate than FSDPr+REF. This percentage increases to 45.75% when the connectivity radius is equal to 0.16. Similar conclusions can be deduced by analyzing Figs. 8.9–8.10, which show the

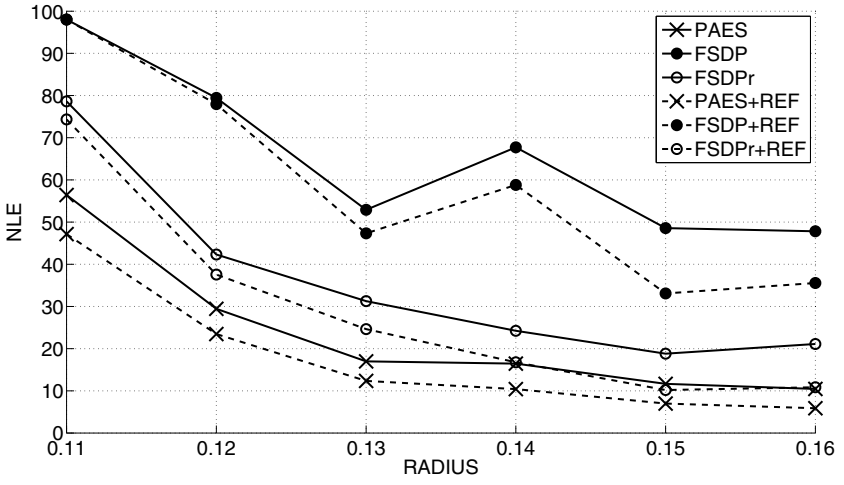


Fig. 8.8. Comparison among PAES, FSDP and FSDPr without (solid line) and with (dotted line) gradient refinement (*REF*) using 8% of the nodes as anchor nodes

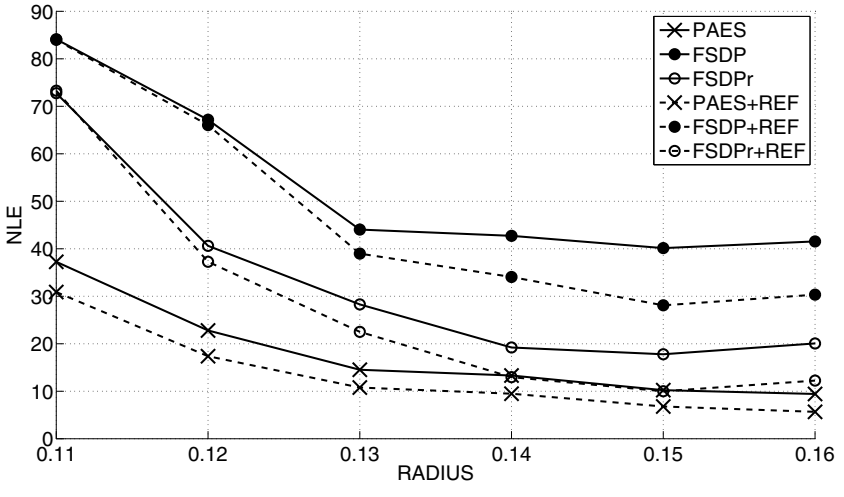


Fig. 8.9. Comparison among PAES, FSDP and FSDPr without (solid line) and with (dotted line) gradient refinement (*REF*) using 10% of the nodes as anchor nodes

results obtained by using the 10% and 12% of anchor nodes: PAES+REF generate solutions which are 57.84% and 53.72% (61.79% and 48.87%) more accurate when the percentage of anchor nodes is equal to 10% (12%) and the connectivity radius is 0.11 and 0.16.

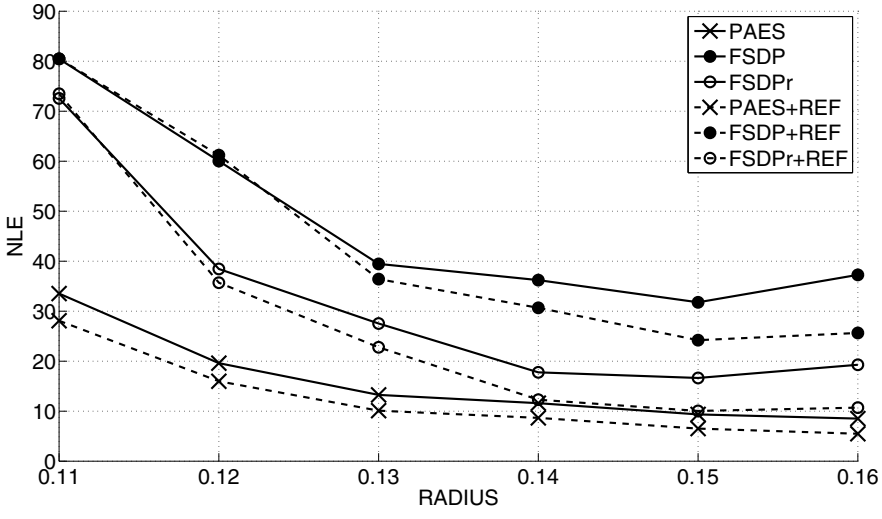


Fig. 8.10. Comparison among PAES, FSDP and FSDPr without (solid line) and with (dotted line) gradient refinement (*REF*) using 12% of the nodes as anchor nodes

8.8 Conclusions

Saving energy is a very critical issue in wireless sensor networks since sensor nodes are typically powered by batteries with a limited capacity. Since the radio is the main cause of power consumption in a sensor node, transmission/reception of data can be limited by means of data compression. In this framework, we have proposed a lossy compression algorithm purposely–designed for the limited resources available on board sensor nodes and based on a differential pulse code modulation scheme where the differences between consecutive samples are quantized. Since the quantization process affects both the compression ratio and the information loss, we applied NSGA–II to generate different combinations of the quantization process parameters corresponding to different optimal trade–offs between compression performance and information loss. The user can therefore choose the combination with the most suitable trade–off for the specific application. We tested our lossy compression approach on three datasets collected by real WSNs. We have obtained compression ratios up to 93.48% with very low reconstruction errors. Moreover, we have shown how our approach outperforms LTC, a lossy compression algorithm purposely designed to be embedded in sensor nodes, in terms of both compression ratio and complexity.

Once the compressed information achieves the sink, it is essential to know the locations of the nodes where the single measures have been collected. To enable location awareness we have proposed to tackle the problem by exploiting a two–objective evolutionary algorithm and relying on a better exploitation of the connectivity graph so as to define topological constraints. Such constraints define zones

of the space where each sensor can or cannot be located, thus reducing the search space of the evolutionary algorithm and contextually the chance of ambiguously flipping node locations. Moreover we have discussed the possibility of using a standard gradient–based technique able to refine the final estimation produced by the evolutionary algorithm. We have shown that the proposed approach is able to solve the localization problem with high accuracy for a number of different topologies, connectivity radii and percentages of anchor nodes. Further, we have discussed how our approach outperforms the standard SDP–based technique and its regularized version.

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Computer Networks 38, 393–422 (2002)
2. Aspnes, J., Goldenberg, D., Yang, Y.R.: On the Computational Complexity of Sensor Network Localization. In: Nikoletseas, S.E., Rolim, J.D.P. (eds.) ALGOSENSORS 2004. LNCS, vol. 3121, pp. 32–44. Springer, Heidelberg (2004)
3. Barr, K.C., Asanović, K.: Energy-aware lossless data compression. ACM Trans. Comput. Syst. 24(3), 250–291 (2006)
4. Barrenetxea, G., Ingelrest, F., Schaefer, G., Vetterli, M., Couach, O., Parlange, M.: SensorScope: Out-of-the-Box Environmental Monitoring. In: Proc. of the 7th Int. Conf. on Information Processing in Sensor Networks, pp. 332–343 (2008)
5. Biswas, P., Lian, T.C., Wang, T.C., Ye, Y.: Semidefinite programming based algorithms for sensor network localization. ACM Trans. Sen. Netw. 2, 188–220 (2006)
6. Biswas, P., Liang, T.C., Toh, K.C., Ye, Y., Wang, T.C.: Semidefinite programming approaches for sensor network localization with noisy distance measurements. IEEE Trans. Autom. Sci. Eng. 3(4), 360–371 (2006)
7. Biswas, P., Ye, Y.: Semidefinite programming for ad hoc wireless sensor network localization. In: Proc. of the 3rd Int. Conf. on Information Processing in Sensor Networks, pp. 46–54 (2004)
8. Bleuler, S., Laumanns, M., Thiele, L., Zitzler, E.: PISA – A Platform and Programming Language Independent Interface for Search Algorithms. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 494–508. Springer, Heidelberg (2003)
9. Costa, J.A., Patwari, N., Hero III, A.O.: Distributed weighted–multidimensional scaling for node localization in sensor networks. ACM Trans. on Sensor Networks 2(1), 39–64 (2006)
10. Croce, S., Marcelloni, F., Vecchio, M.: Reducing power consumption in wireless sensor networks using a novel approach to data aggregation. The Computer Journal 51(2), 227–239 (2008)
11. Cutler, C.C.: Differential quantization of communication signals. Patent 2 605 361 (1952)
12. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6(2), 182–197 (2002)
13. Donoho, D.L., Johnstone, I.M.: Ideal spatial adaptation by wavelet shrinkage. Biometrika 81(3), 425–455 (1994)
14. Ferrière, H.D., Fabre, L., Meier, R., Metrailler, P.: TinyNode: a comprehensive platform for wireless sensor network applications. In: IPSN 2006: Proc. of the 5th Int. Conf. on Information Processing in Sensor Networks, pp. 358–365 (2006)

15. Horn, J., Nafpliotis, N., Goldberg, D.E.: A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In: Proc. of the 1st IEEE World Congress on Evolutionary Computation, vol. 1, pp. 82–87 (1994)
16. Huffman, D.: A method for the construction of minimum-redundancy codes. In: Proc. of the IRE, vol. 40(9), pp. 1098–1101 (1952)
17. Ji, X., Zha, H.: Sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling. In: INFOCOM 2004: Proc. of the 23rd Annual Joint Conf. of the IEEE Computer and Communications Societies, vol. 4, pp. 2652–2661 (2004)
18. Kannan, A.A., Fidan, B., Mao, G.: Analysis of flip ambiguities for robust sensor network localization. IEEE Trans. Veh. Technol. 59(4), 2057–2070 (2010)
19. Kannan, A.A., Mao, G., Vucetic, B.: Simulated annealing based wireless sensor network localization with flip ambiguity mitigation. In: VTC 2006: Proc. of the 63rd IEEE Vehicular Technology Conf., pp. 1022–1026 (2006)
20. Kim, S., Kojima, M., Waki, H.: Exploiting sparsity in SDP relaxation for sensor network localization. SIAM J. Optim. 20, 192–215 (2009)
21. Knowles, J.D., Corne, D.W.: Approximating the nondominated front using the Pareto Archived Evolution Strategy. IEEE Trans. Evol. Comput. 8(2), 149–172 (2000)
22. LZO homepage (2011), `http://www.oberhumer.com/opensource/lzo/`
23. Mao, G., Fidan, B., Anderson, B.D.O.: Wireless sensor network localization techniques. Computer Networks 51(10), 2529–2553 (2007)
24. Marcelloni, F., Vecchio, M.: A simple algorithm for data compression in wireless sensor networks. IEEE Commun. Lett. 12(6), 411–413 (2008)
25. Marcelloni, F., Vecchio, M.: An efficient lossless compression algorithm for tiny nodes of monitoring wireless sensor networks. The Computer Journal 52(8), 969–987 (2009)
26. Marcelloni, F., Vecchio, M.: Enabling energy-efficient and lossy-aware data compression in wireless sensor networks by multi-objective evolutionary optimization. Information Sciences 180(10), 1924–1941 (2010)
27. Marcelloni, F., Vecchio, M.: A two-objective evolutionary approach to design lossy compression algorithms for tiny nodes of wireless sensor networks. Evolutionary Intelligence 3, 137–153 (2010)
28. Michalewicz, Z.: Genetic algorithms + data structures = evolution programs, 2nd edn. Springer, New York (1994)
29. O'Neal, J.: Differential pulse-code modulation (PCM) with entropy coding. IEEE Trans. Inf. Theory 22(2), 169–174 (1976)
30. Patwari, N., Ash, J.N., Kyperountas, S., Hero III, A.O., Moses, R.L., Correal, N.S.: Locating the nodes: cooperative localization in wireless sensor networks. IEEE Signal Process. Mag. 22(4), 54–69 (2005)
31. Peralta, L.M.R.: Collaborative localization in wireless sensor networks. In: SENSORCOMM 2007: Proc. of the 2007 Int. Conf. on Sensor Technologies and Applications, pp. 94–100 (2007)
32. Pradhan, S., Kusuma, J., Ramchandran, K.: Distributed compression in a dense microsensor network. IEEE Signal Process. Mag. 19(2), 51–60 (2002)
33. Sadler, C.M., Martonosi, M.: Data compression algorithms for energy-constrained devices in delay tolerant networks. In: SenSys 2006: Proc. of the 4th Int. Conf. on Embedded Networked Sensor Systems, pp. 265–278 (2006)
34. Salomon, D.: Data Compression: The Complete Reference, 4th edn. Springer, London (2007)
35. Schoellhammer, T., Greenstein, B., Osterweil, E., Wimbrow, M., Estrin, D.: Lightweight Temporal Compression of microclimate datasets. In: LCN 2004: Proc. of the 29th Annual IEEE Int. Conf. on Local Computer Networks, pp. 516–524 (2004)
36. Sensirion homepage (2011), `www.sensirion.com`
37. Severi, S., Abreu, G., Destino, G., Dardari, D.: Understanding and solving flip-ambiguity in network localization via semidefinite programming. In: GLOBECOM 2009: Proc. of the 28th IEEE Conf. on Global Telecommunications, pp. 3910–3915 (2009)

38. SimIt-ARM homepage (2011), http://simit-arm.sourceforge.net/
39. Srinivas, N., Deb, K.: Multiobjective optimization using Nondominated Sorting in Genetic Algorithms. IEEE Trans. Evol. Comput. 2(3), 221–248 (1994)
40. Tseng, P.: Second–order cone programming relaxation of sensor network localization. SIAM J. Optim. 18(1), 156–185 (2007)
41. Vecchio, M., López-Valcarce, R., Marcelloni, F.: A study on the application of different two-objective evolutionary algorithms to the node localization problem in wireless sensor networks. In: ISDA 2011: Proc. of the 11th IEEE Int. Conf. on Intelligent Systems Design and Applications, pp. 1008–1013 (2011)
42. Vecchio, M., López-Valcarce, R., Marcelloni, F.: A two-objective evolutionary approach based on topological constraints for node localization in wireless sensor networks. Applied Soft Computing 12(7), 1891–1901 (2012), doi:10.1016/j.asoc.2011.03.012
43. Wang, Z., Zheng, S., Ye, Y., Boyd, S.: Further relaxations of the semidefinite programming approach to sensor network localization. SIAM J. Optim. 19(2), 655–673 (2008)
44. Xiong, Z., Liveris, A., Cheng, S.: Distributed source coding for sensor networks. IEEE Signal Process. Mag. 21(5), 80–94 (2004)
45. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. IEEE Trans. Evol. Comput. 8(2), 173–195 (2000)
46. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou, K., et al. (eds.) Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001), pp. 95–100. Int. Center for Numerical Methods in Engineering (CIMNE), Barcelona (2002)
47. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans. Evol. Comput. 3(4), 257–271 (1999)
48. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, G.V.: Performance assessment of multiobjective optimizers: An analysis and review. IEEE Trans. Evol. Comput. 7, 117–132 (2002)