An Overview of the Scheduling Policies and Algorithms in Grid Computing

Article · January 2011

CITATIONS
29

READS
735

2 authors:

George Amalarethinam
Jamal Mohamed College
155 PUBLICATIONS **801** CITATIONS
SEE PROFILE

Muthulakshmi P
SRM Institute of Science and Technology
15 PUBLICATIONS **47** CITATIONS
SEE PROFILE

International Journal of Research and Reviews in Computer Science (IJRRCS) Vol. 2, No. 2, April 2011

An Overview of the Scheduling Policies and Algorithms in Grid Computing

Dr. D.I. George Amalarethinam[1], P. Muthulakshmi[2*]

1 Director-MCA & Associate Professor of Computer Science, Jamal Mohamed College (Autonomous), Tiruchirappalli, India
2 Lecturer, Department of Computer Science, SRM University, Kattankulathur, Chennai, India

Abstract: The advancement of computers and network technologies have taken us to the world of high performance computing called the Grid computing which is the evolution of parallel processing. Rather the improvement in performances is still expected. The orientation of grid computing is to integrate and aggregate the resources contributing themselves in a geographically distributed environment. The view on such environments remains incomplete if we fail to look into scheduling. Scheduling is the ultimate goal aiming process of mapping the jobs submitted to the grid and the appropriate resources available. This paper presents a package of reviews taking various factors that are having greater influence while scheduling the jobs. Here we consider the algorithms keyed with communication cost, execution time length, error factors, task duplications, data intensive, and heterogenic network behavior and so on. A sequence of classification and comparison of algorithms evolved from homogeneous scheduling to the state of art scheduling is listed. Though the earlier algorithms are the pioneers of scheduling, we have studied the perspectives, working principles, the grid domain in which the algorithms have been applied.

Key Terms: scheduling algorithms, scheduling policies, hybrid algorithms, dynamicity, heterogenic network behavior, grid computing, parallel processing

1. Introduction

Foster et al [1] defined Grid computing as a "coordinated resource sharing and problem solving dynamic, multi-institutional collaborations". The novel idea behind the technology is to identify the resources in the network of computers that are geographically spread and allow the computers to coordinate themselves in sharing the resources to solve the complex problems in short span of time and at economical rate. Parallel processing is a promising approach to meet the computational requirements of a large number of current and emerging applications, Hwang [2]. Parallel processing in the form of Grid Computing is a key to the computational requirements of extremely large number of today's and tomorrow's expectations and it has its evolutionary directions towards job scheduling and task scheduling. Scheduling refers to the way processes are assigned to run on the available processors. A job scheduling is a process of establishing queue to run a sequence of programs over a period of time.

A task scheduling is the mapping of tasks to a selected group of resources which may be distributed in multiple administration domains. But the problems lie in developing an algorithm to suit the various forms of applications, viz., process of partitioning the application into a heap of tasks, encouraging coordinated communication among tasks, monitoring the synchronization among tasks, and mapping (scheduling)the tasks to the machines. An appropriate scheduling of tasks to the resources is the greater potentiality that can enhance the functional capability of the Grid System. This paper presents the aspects of scheduling and the factors that influence the same. The core objective of scheduling is to minimize the completion time of a parallel application by properly and carefully allocating the tasks to the appropriate processors. Scheduling can be classified into two main root categories called static and dynamic. In static scheduling, there is no job failure and resources are assumed available all the time. But this is not so in dynamic scheduling. There are essentially two main aspects that determine the dynamics of the Grid scheduling, namely: (a) the dynamics of job execution, which refers to the situation when job execution could fail due to some resource failure or job execution could be stopped due to the arrival in the system of high priority jobs when the case of preemptive mode is considered; and (b) the dynamics of resources, in which resources can join or leave the Grid in an unpredictable way, their workload can significantly vary over time, the local policies on usage of resources could change over time, etc. These two factors decide the behavior of the Grid Scheduler (GS), ranging from static to highly dynamic.

The continuation of the paper is structured with the sections as follows; Section 2 shows scheduling in grid, Section 3 lists various types of scheduling policies, Section 4 shows various traditional scheduling algorithms, Section 5 is packed with the recent works on scheduling, Section 6 gives the comparison of various factors between some considered algorithms and finally the paper is concluded in Section7.

2. Scheduling in Grid

2.1. Taxonomy of Grid

The organization of the Grid infrastructure involves with the foundation/fabrication level, middleware, services, and applications as shown in Figure1. First comes the foundation/fabrication level, where the physical components are finding part, next sits middleware, which is actually the software, responsible for resource management, task execution, job scheduling, then is the vendors/users, who is going to use the efficiency of the grid solutions and finally the level with the services such as operational utilities and business solutions/tools, which will be made use by the users of the grid, that we can bind the services and the applications.

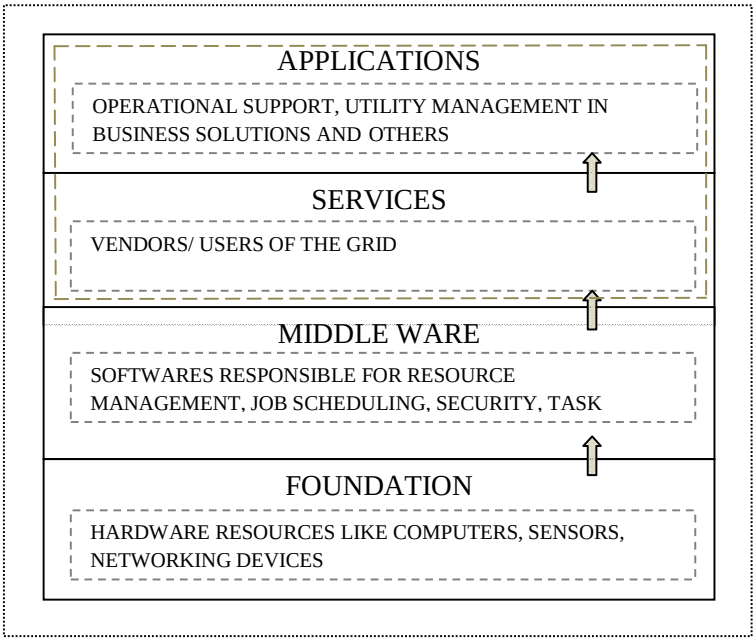

Figure1. Taxonomy of grid

2.2. Types of Grids

There are many types of grids based on the factors like, structure of the organization, the resource used in the grid , and services offered by the grid, namely, Departmental Grids , used to solve problems for a particular group, ex. Folding@Home, Enterprise Grids have been established by the enterprises consisting enterprise resources and offer services to customers, ex. SETI@home [3], Extraprise Grids are established between companies, partners, customers, ex. Amazon.com, Global Grids hold itself over the internet, ex. White Rose Grid, Compute Grids are solely for the use of providing access to computational resources, ex. Smart Grid, Data Grids are highly data intensive and are optimized for data oriented operations, Utility Grids are Grid Resource Provider, which provide access to resources, National Grids are nation run grids managed by the country to plan during disaster etc.,

2.3. Scheduling

The primary aim of grid is to provide transparent and efficient access to remote and geographically distributed resources. A scheduling service is needed to coordinate the access to the different resources. Due to the dynamicity and heterogeneity, scheduling jobs is a bit complicated matter. Jobs queuing and dispatching the jobs to the available nodes (processors) are the works of the scheduler. Grid schedulers dispatch jobs following the eminent algorithms like, First Come First Serve algorithm, Shortest Path First algorithm, Round Robin technique, Least Recently Serviced. Scheduling is done on taking the factors like job priority, reservation of resources, specific resource requirement of the job, etc., Keeping

check points and mentoring is a great mechanism whereby places at which the process stopped or malfunctioned can be identified and the process can restart there at right for further completion. The advantage of keeping check points is that, the stopped process can resume from the place at which it stopped rather starting from the initial stage of the process. Fault due to load imbalance and other cause results in the migration jobs to other node. As it is the decentralized system the jobs can be sent and received by the local schedulers, failure detection of any node may not affect the computation and a simultaneous rescheduling is needed. Figure2 shows some of the eminent factors influencing the job scheduling in grids and their various dimensions (facets). They are roughly said as, dependency among tasks, communication cost, execution cost, error factors like node failure, network failure, data storage, task duplications, execution time length, and heterogenic network behavior, fault tolerance, dynamicity, performance, cost efficiency, Quality of Service, time dependency, resource recovery, resource allocation and management, security so on.

FACTORS INFLUENCING THE WORK FLOW IN A GRID SYSTEM	**Object**	**Dimension**	**Segments**	
	Complexity of the problem	Low		
		Medium		
		High		
	Criteria to solve the problem	Cost	Communication cost	Uniform cost
				No cost
				Varying cost
			Computation cost	Uniform cost
				No cost
				Varying cost
		Time		
		Cost and time		
	Availability of the resource	Static		
		Dynamic	Load balancing	
			Task duplication	
			Job migration	
			Rescheduling	
	Nature of the resource	Homogeneous		
		Heterogeneous		
	Network behavior	Topology		
		Centralized		
		Decentralized		
		Bandwidth	Uniform	
			Varying	
	Nature of the task	Dependent		
		Independent		
	Orientation of the problem	Resource oriented		
		Application oriented	Data dependency	
	Mapping dependency	Priority based		
		Random allocation		
		Based on some relative algorithms		
	Data processing	Readily available		
		Data duplication	Memory availability	
		Data replication	Memory availability	
		Access from the source	Shortest path	
	Others	Error factors	Resource failure	Job migration
				Rescheduling
		Idle resources		

Figure2. Classification of factors and their attributes influencing the work flow in the Grid system

3. Scheduling policies

3.1. Immediate Mode [4]

In immediate mode scheduling, tasks are scheduled as soon as they enter the system. They will not be waiting for the next time interval. They will be scheduled as when the scheduler will get activated or the job arrival rate is small, having thus available resources to execute jobs immediately.

3.2. Batch Mode [4]

In batch mode scheduling, tasks are grouped into batches which are allocated to the resources by the scheduler. The results of processing are usually obtained at a later time. Batch scheduling could take better advantage of job and resource characteristics in deciding which job to map to which resource since they dispose of the time interval between two successive activations of the scheduler.

3.3. Preemptive Scheduling [4]

In the preemptive mode, preemption is allowed; that is, the current execution of the job can be interrupted and the job is migrated to another resource. Preemption can be useful if job priority is to be considered as one of the constraints.

3.4. Non Preemptive Scheduling [4]

In the non-preemptive mode, a task, job or application should entirely be completed in the resource (the resource cannot be taken away from the task, job or application).

3.5. Static Scheduling [5]

In static scheduling, the information regarding all the resources in the Grid as well as all the tasks in an application are assumed to be known in advance by the time the application is scheduled and further more a task is assigned once to a resource. As far when the scheduler's point of view, it is easier to adopt.

3.6. Dynamic Scheduling [5]

In dynamic scheduling, the task allocation is done on the go as the application executes, where it is not possible to find the execution time. The jobs are entering dynamically and the scheduler has to work hard in decision making to allocate resources. This is the place where the load balancing is a factor to be considered seriously. The advantage of the dynamic over the static scheduling is that the system need not possess the run time behavior of the application before it runs.

3.7. Independent Scheduling

In independent scheduling the applications are written to be able to be partitioned into almost independent parts (or loosely coupled), which can be scheduled independently.

3.8. Centralized Scheduling [5]

In dynamic scheduling scenarios, the responsibility for making global scheduling decisions may lie with one centralized scheduler, or may be shared by multiple distributed schedulers. The advantage of having centralized scheduling is the ease of implementation, but lacks scalability, fault tolerance and sometimes performance. In centralized scheduling, there is more control on resources: the scheduler has knowledge of the system by monitoring of the resource state, and therefore it is easier to obtain efficient schedulers. As this type of scheduling suffers from limited scalability and is not appropriate for large-scale Grids.

3.9. Co-operative Scheduling [5]

In cooperative scheduling, each grid scheduler has the responsibility to carry out its own portion of the scheduling task, but all schedulers are working towards common system-wide reach. A feasible schedule is computed through the cooperation of procedures, rules, and Grid users.

3.10. Non–cooperative Scheduling [5]

In non cooperative cases, individual schedulers act alone as autonomous entities and arrive at decisions regarding their own optimum objects independent of the effects of the decision on the rest of system.

3.11. Decentralized Scheduling [6]

In decentralized or distributed scheduling there is no central entity controlling the resources. The autonomous Grid sites make it more challenging to obtain efficient schedulers. In decentralized schedulers, the local schedulers play an important role. The scheduling requests, either by local users or other Grid schedulers, are sent to local schedulers, which manage and maintain the state of the job queue. This type of scheduling is more realistic for real Grid systems of large scale, although decentralized schedulers could be less efficient than centralized schedulers.

3.12. Adaptive Scheduling [7]

The changeability over time of the Grid computing environment requires adaptive scheduling techniques, which will take into account both the current status of the resources and predictions for their future status with the aim of detecting and avoiding performance deterioration. Rescheduling can also be seen as a form of adaptive scheduling in which running jobs are migrated to more suitable resources.

3.13. Hierarchical Scheduling [8]

In hierarchical scheduling, the computation is at three levels. The top level is called the meta-level, where we have a super manager to control group merging/partitioning. At this level, tasks are not scheduled directly, but reconfiguring the scheduler according to the characteristics of the input tasks. Such a process is called meta-scheduling.. The mediate level is called the group level, where the manager in each group collaborates with each other and allocates tasks for the workers in the group. The purpose of collaborating between managers is to improve the load balance across the groups. Specifically, the managers ensure that the workload in the local ready lists is roughly equal for all groups. The bottom level is called the within-group level, where the workers in each group perform self-scheduling. The hierarchical scheduler behaves between centralized and distributed schedulers, so that it can adapt to the input task graph.

3.14. List Scheduling [9]

A list scheduling heuristic prioritizes workflow tasks and schedules them based on their priorities. The basic idea of list scheduling is to make an ordered list of processes by assigning them some priorities, and then repeatedly execute the following two steps until a valid schedule is obtained :1) Select from the list, the process with the highest priority for scheduling. 2) Select a resource to accommodate this process. The Priorities are determined statically before scheduling process begins. The first step chooses the process with the highest priority; the second step selects the best possible resource. Some of the list scheduling strategies are Highest level first algorithm , Longest path algorithm, Longest processing time, Critical path method etc.

3.15. Master Slave Scheduling [10]

In Master Slave Scheduling model, each job has to be processed sequentially in three stages. In the first stage, the preprocessing task runs on a master machine; in the second stage, the slave task runs on a dedicated slave machine; and in the last stage, the post processing task again runs on a master machine, possibly different from the master machine in the first stage.

4. Traditional Scheduling Algorithms

4.1. First Come First Serve

U. Schwiegelshohn, R. Yahyapour [11] analyzed the First Come First Serve (FCFS) for parallel processing and is aiming at the resource with the smallest waiting queue time and is selected for the incoming task. This can be called as Opportunistic Load Balancing (OLB) [12] or myopic algorithm. It is found to be very simple, and can not be expected to be optimal. Opportunistic Load Balancing (OLB) assigns each task, in arbitrary order, to the next machine that is expected to be available, irrespective of the task's expected execution time on that machine [13] [14] [15]. It is noted that the intuition behind OLB is to keep all machines as busy as possible. Simplicity is the advantage of using OLB, as it is not considering expected task execution times, the mappings result in poor makespans.

4.2. Min_ Min

R. Armstrong et al [13] stated that the Min_Min heuristic begins with the set of all unmapped tasks having the set of minimum completion times. Then task with the overall minimum completion time from the set is selected and assigned to the corresponding machine .The newly mapped task is removed from the set, and the process repeats until all tasks are mapped (i.e., set is said to be empty). Min_Min is based on the Minimum Completion Time, MCT. However, Min_Min considers all unmapped tasks during each mapping decision and MCT only considers one task at a time. Min_Min maps the tasks in the order that changes the machine availability status by the least amount that any assignment could. The expectation is that a smaller makespan can be obtained if more tasks are assigned to the machines that complete them the earliest and also execute them the fastest.

4.3. Minimum Completion Time

R. F. Freund et al [14] said that the Minimum Completion Time (MCT) assigns each task, in arbitrary order, to the machine with the minimum expected completion time for that task [13]. This

causes some tasks to be assigned to machines that do not have the minimum execution time for them. The intuition behind MCT is to combine the benefits of OLB and MET, while avoiding the circumstances in which OLB and MET perform poorly.

4.4. Backfilling

Ahuva W. Mu'alem and Dror G. Feitelson [16] proposed the idea of back filling in distributed environments. Backfilling is a scheduling optimization that allows a scheduler to make better use of available resources by running jobs out of order. The scheduler prioritizes the jobs in the queue according to a number of factors and then orders the jobs into a highest priority first (or priority FIFO) sorted list. Backfilling allows small jobs to initiate before larger queued jobs as the larger queued jobs resources currently unavailable. It also requires that all jobs' service times be known. Stavrindis, et al [17] stated that the service time can be provided either as an estimation by users when the jobs are submitted, or predictions made by the system based on historical data. The utilization is dramatically improved when using this scheduling technique.

4.5. Round Robin Technique

Ruay-Shiung Chang et al [18] said that Round Robin (RR) algorithm focuses on the fairness. RR uses the ring as its queue to store jobs. Each job in a queue has the same execution time and it will be executed in turn. If a job can't be completed during its turn, it will be stored back to the queue waiting for the next turn. The advantage of RR algorithm is that each job will be executed in turn and they don't have to be waited for the previous one to get completed. But if the load is found to be heavy, RR will be taking a very long time to complete all the jobs. Each job will be given a priority value by the priority scheduling algorithm and the same is used to dispatch jobs. The priority value of each job will be depending on the job status such as the requirement of memory sizes, CPU time and so on. The main problem of this algorithm is that it may be causing an indefinite blocking or starvation if the requirement of a job is never being satisfied.

4.6. Earliest Deadline First

Earliest Deadline First (EDF) or Least Time to Go is a dynamic scheduling algorithm used in real-time operating systems. It places processes in a priority queue. Whenever a scheduling event occurs (task finishes, new task released, etc.) the queue will be searched for the process closest to its deadline, the found process will be the next to be scheduled for execution. EDF is an optimal scheduling algorithm on preemptive uni-processors.

4.7. Minimum Execution Time

R. F. Freund et al [14] investigated that in contrast to OLB, Minimum Execution Time (MET) assigns each task, in arbitrary order, to the machine with the best expected execution time for that task, regardless of that machine's availability. The motivation behind MET is to give each task to its best machine. This can cause a severe load imbalance across machines. In general, this heuristic is obviously not applicable to High Computing environments as said by Braun et al [19].

4.8. Max Min

O. H. Ibarra et al [20] given that the Max_Min heuristic is very similar to Min_min. The Max_min heuristic also begins with the set of all unmapped tasks. Then, the set of minimum completion times is found. Now, the task with the overall maximum completion time from the set is selected and assigned to the corresponding machine (hence said to be Max_Min). Now, the newly mapped task is removed from the set, and the process repeats until all tasks are mapped (i.e., set is found to be empty). Naturally, Max_Min attempts to minimize the penalty incur from performing tasks with longer execution times. It is assumed that the metatask being mapped has many tasks with very short execution times and one task with a very long execution time. Mapping the task with the longer execution time to its best machine first allows this task to be executed concurrently with the remaining tasks (with shorter execution times). This would be a better mapping than a Min_Min mapping, where all of the shorter tasks would execute first, and then the longer running task would execute while several machines sit idle. Hence it can be said that the Max_Min heuristic may give a mapping with a more balanced load across machines and a better makespan.

4.9. Ant Colony Optimization

A good schedule should accommodate its scheduling policy according to the dynamicity of the entire environment and the types (nature) of jobs. M. Dorigo et al [21] presented Ant Colony Optimization (ACO) is an appropriate algorithm for grids, which are dynamic in nature. ACO is a heuristic algorithm

with efficient local search for combinatorial problems. ACO imitates the behavior of real ant colonies in nature to search for food and to connect to each other by pheromone laid on paths traveled [22]. ACO has been used to solve NP-hard problems such as traveling salesman problem [23], graph coloring problem [24], vehicle routing problem [25], and so on.

4.10. Genetic Algorithm

Prodan and Fahringer [26] used Genetic Algorithm (GA) in grid for scheduling and GA was proposed by J. H. Holland in 1975. It is an evolutionary technique for large space search. Braun. R et al [19] given the general procedure of GA search and is as follows: 1) Population generation: A population is a set of chromosomes, representing a possible solution, can be a mapping sequence between tasks and machines. The initial population can be generated by other heuristic algorithms, such as Min_Min. 2) Chromosome evaluation: Each chromosome is associated with a fitness value, which is the makespan of the task-machine mapping this chromosome represents. The goal of GA search is to find the chromosome with optimal fitness value. 3) Crossover and Mutation operation: Crossover operation selects a random pair of chromosomes and chooses a random point in the first chromosome, from that point to the end of each chromosome, crossover exchanges machine assignments between corresponding tasks. Mutation randomly selects a chromosome, then randomly selects a task within the chromosome, and randomly reassigns it to a new machine.4) Evaluation of the modified chromosome: The chromosomes from this modified population are evaluated again. The above defined steps complete an iteration of the GA. The GA stops when a predefined number of evolutions have been reached or all chromosomes converge to the same mapping. GA randomly selects chromosomes. More precisely, Crossover is the process of swapping certain sub-sequences in the selected chromosomes. Mutation is the random process of replacing certain sub-sequences with some task-mapping choices that are new to the current population. After crossover and mutation, a new population is generated. Then this new population is evaluated, and the process starts over again until some stopping criteria are met. The stopping criteria can be, for example, 1) no improvement in recent evaluations; 2) all chromosomes converge to the same mapping; 3) a cost bound is met. GA is the most attracted and popular Nature's Law heuristic algorithm used in optimization problems.

4.11. Simulated Annealing

Y. Liu [27] investigated that Simulated Annealing (SA) is a search technique based on the actual process of annealing. It is the thermal process of obtaining low-energy crystalline states of a solid. Very first level of the processing starts with melting the solid, the temperature is increased to melt the solid. The temperature is slowly decreased; particles of the melted solid arrange themselves locally, in a stable "ground" state of a solid. SA theory states that if temperature is lowered sufficiently slowly, the solid will reach thermal equilibrium, which is an optimal state. The thermal equilibrium is an optimal task-machine mapping (optimization), the temperature is the total completion time of a mapping (cost function), and the change of temperature is the process of dynamicity of mapping. If the next temperature is higher, which means a worse mapping, the next state is accepted with certain probability. This is because the acceptance of some "worse" states provides a way to break out local optimality which occurs often in local search.

4.12. Particle Swarm Algorithm

Particle Swarm Optimization (PSO) is a proposed algorithm by James Kennedy and R. C. Eberhart in 1995 [28] motivated by social behavior of organisms such as bird flocking and fish schooling. PSO algorithm is not only a tool for optimization, but also a tool for representing socio cognition of human and artificial agents, based on principles of social psychology. PSO as an optimization tool provides a population-based search procedure in which individuals called particles change their position (state) with time. In a PSO system, particles fly around in a multidimensional search space. During it fly, each particle adjusts its position according to its own experience, and according to the experience of a neighboring particle, making use of the best position encountered by itself and its neighbor. PSO system combines local search methods with global search methods, attempting to balance exploration and exploitation.

4.13. Game Theory

The Grid scheduling problem is modeled using Game Theory (GT), L. Young et al [29], which is a technique commonly used to solve economic

problems. Each task is modeled as a player whose available strategies are the resources on which the task could run. The payoff for a player is defined to be the sum of the benefit value for running the task and all communication arriving at that task. GT, a job scheduling game is a game that models a scenario in which multiple selfish users wish to utilize multiple processing machines. Each user has a single job, and the user needs to choose a single machine to process it. The incentive of each user is to have his job run as fast as possible. Job scheduling games are the following set of problems: given a set of machines and a set of jobs. Each job is associated with a vector, corresponding to its size on each machine (i.e., is the processing time of job on a machine). Players correspond to jobs. The strategy set of each player is the set of machines. Given a strategy for each player, the total load on each machine is the sum of processing times of the jobs that chose that machine. Usually each player seeks to minimize the total load on its chosen machine. The standard objective function is minimizing the total load on the most-loaded machine (makespan minimization).

4.14. Tabu Search

R. Braun et al [19] implemented Tabu Search (TS) in their work about short hop procedure to find the nearest local minimum solution with in the solution space. TS is a solution space search that keeps track of the regions of the solution space which have already been searched so as not to repeat a search near these areas[30][31].TS is a meta-strategy for guiding known heuristics to overcome local optimality and has now become an established optimization approach that is rapidly spreading to many new fields. The method can be viewed as an iterative technique which explores a set of problem solutions, by repeatedly making moves from one solution *s* to another solution *s'* located in the neighborhoods. These moves are performed with the aim of efficiently reaching an optimal solution by minimizing some objective function.

4.15. Fuzzy Algorithms

Zhou et al [32] used Fuzzy Logic (FL) techniques to design an adaptive FL scheduler, which utilizes the FL control technology to select the most suitable computing node in the Grid environment. A fuzzy set is a set containing elements that have varying degrees of membership. There is fuzziness in all preemptive scheduling algorithms. A Fuzzy Neural Network was proposed by Yu et al [33], to develop a high-performance scheduling algorithm. The algorithms uses FL techniques to evaluate the Grid system load information, and adopt the Neural Networks (NN) to automatically tune the membership functions. Artificial Neural Network (ANN) is data-driven modeling tool that is able to capture and represent complex and non-linear input/output relationships. They are recognized as powerful and general technique for Machine Learning because of their nonlinear modeling abilities and robustness in handling noise ridden data. Hao et al [34] presented a Grid resource selection based on NN aiming to achieve QoS.

5. Recent Works on Scheduling

5.1. Modified OLB, Modified MET, Modified MCT, Modified Min_Min, Modified Min_Max

It is found by L. Y. Tseng et al [35] that most of the studies on scheduling algorithms related the mapping of task and computer by the Expected Time to Complete (ETC) matrix such that each row represented the ETC for each task to be completed on each computer and each column listed the ETC for each computer to execute each task. Further he stated that it is difficult to predict the completion time of each task but they utilized the ETC matrix for the presentation of heuristic feasibility. The Computer Availability Time (CAT) of each computer is accumulated for each task's expected execution times. They represented it through a matrix comprised of multiple tasks and computers, each row is composed of a task and computer represented i and j respectively, such that each task (i) is executed on computer (j) and got the minimum completion time. They highlighted three issues upon several observations, firstly they marked that assumption of most studies are based on high speed network communication environment, communication time of each task was not considered. Secondly, the importance of each task was not notable. And finally, the deadline for each task was not considered.

In their study they have considered factors such as: communication cost, weight and deadline for each task executions time. It is said in the paper that Pinedo and Chao in 1999 proposed Apparent Tardiness Cost Setups (ATCS) satisfied the requirements. But ATCS was to schedule all the jobs on one single machine. In this paper, they said that ATCS has to be modified to be applicable in a Grid environment. The ATCS rule

taken the following for its consideration: Weighted Shortest Processing Time Fist (WSPT), Minimum Slack first (MF), Shortest Setup Time first (SST), due date range, due date tightness, set up time severity. By integrating all the factors said above ATCS value is calculated and the job having maximum ATCS value shall be executed first. In their work they divided job into several tasks, since tasks can be dispatched to idle computers that could get better performance in Grid environment. Here every task is assigned with weight and deadline; provide they remained the same to any given computer. Deadline is generated randomly between ranges. Communication times had been simulated to the computers through the networks. The other settings of ATCS remained as it was and the communication time had been included. Furthermore, to avoid dispatching most tasks to few computers, if the computer had not the smallest completion time, the same had been integrated with MCT approach and allowed to dispatch the task to the computer featured the MCT. Hence the original ATCS had been revised as the Apparent Tardiness Cost Setups-Minimum Completion Time (ACTS-MCT). Followed the ACTS-MCT, they proposed OLB, MET, MCT, Min_Min, Max_Min with deadlines and compared with ACTS-MCT.

5.2. Modified FCFS that uses Backfilling

Sofia K. Dimitriadou et al [36] proposed Modified FCFS that uses Backfilling and the same is applied in problem dealt with resource allocation and therefore efficient job scheduling could be achieved. They stated that the grid jobs that enter the system are parallel jobs, and are gangs. Job scheduling is done at two levels, viz., grid level and local level. At both the levels the jobs are competing for the same resources. The goal is to provide services to all, provided the local jobs are treated as higher priority jobs and hence their waiting time is minimized. The local jobs are simple sequential jobs that require only a single processor for execution. A gang consists of a number of tasks that need to be processed simultaneously, thus each task must be allocated to a different processor. The Grid Scheduler (GS) has a queue where the jobs will be stored before they are dispatched. In order for a gang to start execution, all required processors must be available. As when FCFS policy is applied, the local jobs would get blocked by the gangs and would be delayed even if idle processors were available. To avoid such a kind of fragmentation, backfilling had been applied. Backfilling allows small jobs to initiate before larger queued jobs which require resources that are currently unavailable. Although this scheduling technique dramatically improves utilization, it also requires that all jobs' service times be known. Stavrinidis et al [17] stated in their work that predictions can be made with the help of earlier jobs done. Tchernykh. A et al [37] said that the poor estimates do not affect the overall performance.

The modified FCFS (FCFS and backfilling) would be allowing the small jobs to get into execution so as to avoid the idle processors time.

They proposed the scenarios at the local level when a local job enters a queue: the processor is idle, the queue is empty, and the job will be served immediately. Otherwise the processor is busy, and the job will wait at the end of the queue for its turn to come. If the queue is empty, the job will wait at the beginning of the queue and start execution once the processor is free. And finally, the processor is idle but the queue is not empty. The last scenario could occur when a gang is waiting for service, since a gang could not begin execution unless enough processors are available for all its tasks to be served simultaneously. A large gang may block local jobs behind it in the queue while waiting for sufficient resources to become available. If the First Come First Served (FCFS) policy is used, the system will suffer from severe fragmentation. Furthermore, the gangs would delay the service of local jobs while the locals are of higher importance. For those reasons, a modified FCFS policy that uses backfilling had been applied. This scheduling policy suggests that a local job can take over an idle processor under the condition that delays to the gang in queue are minimal.

5.3. Adaptive Hierarchical Scheduling Policy

J. H. Abawajy [38] proposed a hierarchical scheduling policy concentrated I/ O and service demands of parallel jobs in both homogeneous and heterogeneous systems. It is observed that the paper dealt all with enterprise grid computing environments. The paper focused on the ability to deploy commodity computational power, network, storage resources and how to share and manage. It is said by Thain et al [39] that the grid computing schedulers must bring jobs and data in close proximity in order to satisfy throughput, scalability and policy requirements.

The algorithm integrates the job assignment, affinity scheduling, self scheduling approaches to

assign resources to parallel jobs. The working of the algorithm spoke about two queues where the unscheduled jobs/tasks are maintained in the first queue(Queue(job)) and the second queue is to store the unsatisfied request for computation(RFC), (Queue(RFC)). A Boolean function to evaluate a particular node is not a root or leaf. When a non root node is in neutral state, it sends RFC to its parent RFC. If parent is in neutral state while receiving RFC, it generated its RFC and sends to its parent on the next level of the cluster tree. The process is recursively allowed to do until the RFC reaches either System Scheduler (SS) or a node with unassigned computation. If RFC reached SS, it backlogs the request provided there should not be any job to be scheduled. If SS or a node with no assigned computation is found, the scheduler follows the space-sharing (job/task assignment) policy. The assignment of job/task components allows a set of ideal jobs/tasks to transfer from parent node to a child node. Then the affinity scheduling is applied. If there is no computation with affinity condition then transfer rate is set 1(from parent to a child node). At last each job is partitioned into task and is assigned to processors on demand, where each processor is allowed to follow time sharing policy.

5.4. Player's Minimum Completion Time

Joanna Kolodziej et al [40] proposed the algorithm, which is devised for the computational grids concerned with high performance computations ensured with the request level of security in the data transfer. The proposed work is an improved version of GT model for secure scheduling presented by Joanna Kolodziej et al [41]. In their work they considered Independent Job Scheduling problem Maheswaran et al [12] in which tasks are processed in batch mode. They defined a non co-operative symmetric game model without additional synchronization mechanisms, it is said that the game scenario is simpler than the one presented by Kwok et al [42], where an additional mechanism is needed.

It is said that in this model, the players of the game are non co-operative, where the usage privileges are the same for all the users, in which each user tries to choose an optimal way of matching his task to machines to minimize the total cost of scheduling in the pool of tasks in which his task is found. The problem of minimization of the game cost consists of two co-operating functional hierarchical units (main unit and subordinate unit). Main unit solves the global level problem of the minimization of the function and the subordinate unit solves the local level problems of the minimization of the user's cost function.

They proposed modified MCT in the name of Player's MCT, which is incorporated with the concepts of GT for security assistances. The working of the said algorithm is proposed as; every task is assigned to the machine getting the Earliest Completion Time (ECT). The schedule form the main unit is scanned and the minimum completion time is found for the tasks of the individual user. The scenario of the algorithm stated that there can be a population of schedules and ready time of machines, which is in the main unit. There can be many users in the population and each user can have many tasks. Now minimum completion time algorithm is applied to the tasks of individual user and the same is computed for scheduler received from main unit until no schedule remain in the main unit. Then the minimum of all MCT is found and sent to the main unit.

5.5. Iterative Divisible Load Theory

The objective of the algorithm proposed by Monir Abdullah et al [43] is to design a load distribution model by taking the communication and computation time. It is said that the Divisible Load Theory (DLT) [44] is a powerful tool for modeling data intensive computational problems incorporating communication and computational issues [45]. They said that the load scheduling problem is to decompose the large data set into smaller datasets across virtual sites. The algorithm is all about finding minimum makespan to schedule a large volume loads within multiple sites.

The proposed algorithm started by dividing the large volume of divisible loads using either of the Adaptive DLT (ADLT) models [46] [47], then the makespan is calculated using the cost model defined in this paper. The current time will be taken as makespan if all the nodes finish computation at the same time. Otherwise the sum of the processing time of the entire node is calculated and an average is taken between the sum of the processing time and the number of virtual sites involved. Now the load is calculated and redistributed based on the average in an iterative manner, then the makespan is calculated in the same iterative manner until all the nodes get finished its execution. Finally they stated that all nodes will take the new load based on the new averages and for the last node the rest of the load is

given without considering the average. Finally, the last node will finish at the same time as the others.

5.6. GA based integrated job scheduling algorithm

Chao-Chin Wu et al [48] proposed a GA based integrated job scheduling strategy for grid systems that support four different fault tolerance mechanisms and they considered the following fault tolerance mechanisms;

i) Job Retry (JRT) mechanism. The JRT mechanism is the simplest fault-tolerance technique, which will re-execute the failed job from the beginning on the same computational node. ii) Job migration without checkpointing (JMG) mechanism. The JMG mechanism will move the failed job to another computational node and re-execute the job from the beginning on the latter computational node. iii) Job migration with checkpointing (JCP) mechanism. The JCP mechanism will record the state of the job periodically at rum time. If the job fails, it is moved to another computational node and resumed the execution from the last checkpoint. iv) Job Replication (JRP) mechanism. The JRP mechanism replicates a job to multiple computational nodes such that the job has higher success rate. If one of those replicas has already completed, then all other replicas should stop their execution to save the computing power. They said that each computational site in a grid system supports one of the three mechanism; JRT, JMG, JCP. In favor to JRP, the scheduler will allocate multiple computational sites to execute a certain job concurrently. Further, the scheduler can execute a certain job by any combination of these four different fault tolerance mechanisms.

They proposed a new chromosome encoding approach. A chromosome is a list of variable length integer sequences. Each integer sequence represents a gene, whose length depends on the type of adopted fault tolerance mechanism. The integer sequence represents the job identity (job id) and the node identity (node id). Based on the node ids the job will be moved to the next node for re-execution whenever a failure is encountered. They used Roulette Wheel Selection Method [49] to select chromosomes based on the fitness value; the highest fitness value will be the selected chromosome for the next generation. They employed cut and splice operator [50] to pick two points in two chromosomes and exchange chromosomes after the points. Since the length of the chromosome is variable in this proposed approach. The mutation operation randomly selects a gene in a chromosome and then mutates its value.

5.7. A Hybrid Scheduling Algorithm to minimize the cost with deadline constraint

Xin Liu et al [51] proposed the algorithm in which they tried to obtain minimum cost by perturbing the schedule of some tasks from minimum time solution. They proposed min-time algorithm to find the minimum completion time and the min-cost algorithm to find the minimum cost without considering the deadline constraint. The proposed algorithm is a hybrid scheduling algorithm to minimize some of the tasks lying to the first of the list follow min time and the remaining tasks in the list follow min cost algorithm. This is called as perturbation degree.

The proposed algorithm stated that the task from the list is allowed to evaluate the minimum completion time and if it is greater than the deadline, then there is no possibility of getting feasible solution, if the minimum completion time is less than the deadline then binary search is used recursively for largest perturbation degree, such that the current or the next perturbation degree is smaller than the deadline. Now the cost and perturbation degree is obtained and returned as schedule with minimum cost and finished before deadline.

5.8. Balanced Ant Colony Optimization Algorithm

Rua-Shiung Chang et al[52] stated that the Balanced Ant Colony Optimization Algorithm (BACO) inherits the basic ideas from Ant Colony Optimization (ACO) algorithm to decrease the computation time of jobs executing in Taiwan UniGrid [53] environment and it also considers about the loading of each resource. BACO changes the pheromone density according to the resources status by applying the local pheromone update and the global pheromone update functions. The purpose is to try to minimize the completion time for each job while balancing the system load. They related the ant system to the grid system, and the relationships are: An ant in the ant system is a job in the grid system and Pheromone value on a path in the ant system is a weight for a resource in the grid system. A resource with a larger weight value means that the resource has a better computing power. The scheduler collects data from Information Server and uses the data to calculate a weight value of a resource. The pheromone (weight) of each resource is stored in the scheduler and they made the scheduler to use it as the parameters for

BACO algorithm. Finally, the scheduler selects a resource by a scheduling algorithm and it sends jobs to the selected resource by the Application Programming Interface (API) of the Globus Toolkit [54].

5.9. Pro-active failure handling within scheduling algorithms

B. T. Benjamin Khoo et al [55] stated pro-active mechanisms are algorithms or heuristics where the failure consideration for the grid is made before the scheduling of a job. They introduced three pro-active failure handling strategies which allow existing scheduling algorithms to be modified to avoid job failures; the strategies are, Site availability based allocation (SAA), Node availability based allocation (NAA), and Node and Site based allocation (NSA).They compared pro-active failure mechanism with that of passive failure handling mechanism. They implemented the strategies in Backfill algorithm (BF), Replication Algorithm (REP) and Multi-Resource Scheduling Algorithm (MRS).

6. Comparison of algorithms with the factors influencing the Grid environment

A comparison of the factors influencing the grid environment has been taken for the consideration and a study of algorithms detailed in Section 5 of this paper with such factors is shown in Table1. In our survey of the algorithms, the characteristics like computation/communication cost (computation/communication time), rescheduling, task duplication, network behavior, data locality, memory availability to store data and other factors have been analyzed and tabulated. It is found that no algorithm is available in the literature that comprising all these characteristics. An attempt is being taken by us to propose a new algorithm involving almost all the characteristics to get better scheduling methodology.

Table1. Comparison of various factors in scheduling algorithms

COMPAREDD ALGORITHMS	Reference from Section 5 of this paper	Computation/ Communication cost	Computation/ Communication time	Rescheduling	Task Duplication	Heterogenic Network behavior	Grid Type	Data Locality	Memory to store data	Other Factors
	5.1	Y	Y	-	-	-	-	-	-	Deadline, Tardiness
	5.2	Considered message passing	negligible	Job Migration	-	-	-	-	-	Gang scheduling, Wait time
	5.3	-	-	Affinity scheduling, Self scheduling	-	Y	Enterprise Grid	Y	y	I/O requests, Load balancing
	5.4	Y	-	-	-	Y	Computational Grid	-	-	Security Assurance
	5.5	-	Y	-	-	-	Data Intensive	Y	Y	-

5.6	-	Y	Job Migration without check pointing	-	-	Computational Grid	-	-	Fault tolerance mechanism, job failure
5.7	Y	Y	-	-		Data Intensive	-	Y	Lack of wavelength
5.8	Y	-	-	-	Y	Computing grid	Not readily available	-	Bad resources, band width
5.9	-	Y	-	-	-	-	-	-	Job failure

7. Conclusion

The paper is proposed with packages of scheduling algorithms; scheduling policies, hybrid algorithms and a comparison of fewer algorithms with various factors influencing Grid system. An investigation on various factors that influence the scheduling in grid has been made and shown in this paper. This is an effort made to find the silver lining in the dark clouds which could paint an idea about the scheduling policies and the generated algorithms and how they are being multiplexed with other ideas to provide extensive work in the related area.

References

[1] Foster, C. Kesselman and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual organizations", International Journal of Supercomputer Applications, 15(2001).

[2] Hwang, K. 1993. "Advanced Computer Architecture: Parallelism, Scalability, and Programmability", Mc. Graw Hill, Inc., New York, NY.

[3] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, D.Werthimer, "Seti@home: An experiment in public-resource computing", Communications of the ACM 45 (11) (2002) 56–61.

[4] Fatos Xhafa, Ajith Abraham, 'Computational models and heuristic methods for Grid scheduling problems", Future Generation Computer Systems 26 (2010) 608_621.

[5] T. Casavant and J. Kuhl, "A Taxonomy of Scheduling in General Pupose Distributed Computing Systems", IEEE Trans. on Software Engineering Vol. 14, No. 2, PP 141-154, February 1988.

[6] M. Arora, S. K. Das, R. Biswas, "A Decentralized Scheduling and Load Balancing Algorithm for Heterogeneous Grid Environments", in Proc. Of International Conference on Parallel Processing Workshops(ICPPW'02), pp.:499-505, Vancouver, British Columbia Cananda, August 2002.

[7] L. Lee, C. Liang, H. Chang, "An adaptive task scheduling system for Grid Computing", Proceedings of the Sixth IEEE international Conference on Computer and information Technology, CIT'06, 2006, p. 57.

[8] Yinglong Xia, Viktor K. Prasanna and James Li, "Hierarchical Scheduling of DAG Structured Computations on Manycore Processors with Dynamic Thread Grouping, Job Scheduling strategies for parallel processing", Lecture notes in Computer Science,2010, Volume 6253/2010 , 154-174, DOI: 10.1007/978-3-642-16505-4_9.

[9] J. Yu, R. Buyya and K. Ramamohanrao, "Workflow Scheduling Algorithms for Grid Computing, Meta. for Sched. in Distri. Comp. Envi.", SCI 146, springerlink.com, pp. 173–214, 2008.

[10] S. Sahni. "Scheduling master-slave multiprocessor systems". IEEE Trans. on Computers, 45(10), pp. 1195-1199, 1996.

[11] U. Schwiegelshohn, R. Yahyapour, "Analysis of First-Come-First-Serve parallel job scheduling", in: Proceedings of the 9th SIAM Symposium on Discrete Algorithms, 1998, pp. 629-638.

[12] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen and R. F. Freund, "Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems", in J. of Parallel

and Distributed Computing, Vol. 59, No. 2, pp.107-131, November 1999.

[13] R. Armstrong, D. Hensgen, and T. Kidd, "The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions", in ``7th IEEE Heterogeneous Computing Workshop (HCW '98)," pp. 79_87, 1998.

[14] R. F. Freund, M. Gherrity, S. Ambrosius, M. Campbell, M. Halderman, D. Hensgen, E. Keith,T. Kidd, M. Kussow, J. D. Lima, F. Mirabile, L. Moore, B. Rust, and H. J. Siegel, "Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet", in "7th IEEE Heterogeneous Computing Workshop (HCW '98)," pp. 184_199, 1998.

[15] R. F. Freund and H. J. Siegel, "Heterogeneous processing", IEEE Comput. 26, 6 (June 1993),

[16]Ahuva W. Mu'alem and Dror G. Feitelson, "Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling". IEEE Transactions on Parallel and Distributed Systems, 12:(6), pp. 529-543, 2001.

[17] Stavrinidis, G. and H. D. Karatza , " Performance Evaluation of Gang Scheduling in Distributed Real-Time Systems with Possible Software Faults", Proceedings of the 2008 International Symposium on Performance Evaluation of Computer and Telecommunication Systems-SPECTS 2008, June 16-18, Edinburgh, Scotland, UK, 1–7.

[18] Ruay-Shiung Chang, Jih-Sheng Chang, Po-Sheng Lin, "An ant algorithm for balanced job scheduling in grids", Future Generation Computer Systems 25 (2009) 20–27.

[19] R. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther, J. Robertson, M. Theys, B. Yao, D. Hensgen and R. Freund, "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems", in J. of Parallel and Distributed Computing, vol.61, No. 6, pp.
810-837, 2001.

[20] O. H. Ibarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on non identical processors", J. Assoc. Comput. Mach. 24, 2 (Apr. 1977), 280_289.

[21] M. Dorigo, C. Blum, "Ant colony optimization theory: A survey ", Theoretical Computer Science 344 (2–3) (2005) 243–278.

[22] M. Dorigo, "Ant colony optimization", http://www.aco-metaheuristic.org.

[23] M. Dorigo, L.M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem", IEEE Transactions on Evolutionary Computation 1 (1) (1997) 53–66.

[24] E. Salari, K. Eshghi, "An ACO algorithm for graph coloring problem", in: Congress on Computational Intelligence Methods and Applications, 15–17 Dec. 2005, p. 5.

[25] Xiaoxia Zhang, Lixin Tang, "CT-ACO-hybridizing ant colony optimization with cycle transfer search for the vehicle routing problem", in: Congress on Computational Intelligence Methods and Applications,15–17Dec. 2005,p. 6.

[26] R. Prodan and T. Fahringer, "Dynamic Scheduling of Scientific Workflow Applications on the Grid using a Modular Optimisation Tool: A Case Study", The 20th Symposium of Applied Computing (SAC 2005), Santa Fe, New Mexico, USA, ACM Press, pp. 687- 694, March 2005.

[27] Y. Liu, "Survey on Grid Scheduling" (for Ph.D Qualifying Exam), Department of Computer Science, Universityof Iowa, http://www.cs.uiowa.edu/~yanliu/, April 2004.

[28] Kennedy, J.; Eberhart, R., "Particle swarm optimization", Proceedings: IEEE International Conference on Neural Networks, 1995. Nov/Dec 1995, vol.4 .

[29]L. Young, S. McGough, S. Newhouse, and J. Darlington, "Scheduling Architecture and Algorithms within the ICENI Grid Middleware", in Proc. of UK e-Science All Hands Meeting, pp. 5-12, Nottingham, UK, September 2003.

[30]I. De Falco, R. Del Balio, E. Tarantino, and R. Vaccaro, "Improving search by incorporating evolution principles in parallel Tabu search", in ``1994 IEEE Conference on Evolutionary Computation," Vol. 2, pp. 823_828, 1994.

[31]F. Glover and M. Laguna, ``Tabu Search," Kluwer Academic, Boston, MA, 1997.

[32] J. Zhou, K.M. Yu, Ch.H. Chou, L.A. Yang, Zh.J. Luo, "A dynamic resource broker and fuzzy logic based scheduling algorithm in Grid environment", ICANNGA 1 (2007) 604_613.

[33] K.M. Yu, Zh.J. Luo, Ch.H. Chou, Ch.K. Chen, J. Zhou, "A fuzzy neural network based scheduling algorithm for job assignment on Computational

Grids", in: NBiS 2007, in: LNCS, Springer, 2007, pp. 533_542.

[34] X. Hao, Y. Dai, B. Zhang, T. Chen, L. Yang, Yang, "QoS-driven Grid resource selection based on novel neural networks', GPC 2006 (2006) 456_465.

[35] Li-Ya Tseng, Yeh-Hao Chin, Shu-Ching Wang, 'A minimized makespan scheduler with multiple factors for Grid computing systems", Expert Systems with Applications 36(2009)11118-11130.

[36] Sofia K. Dimitriadou, Helen D. Karatza, "Multi-Site Allocation Policies on a Grid and Local Level", Electronic Notes in Theoretical Computer Science 261 (2010) 163–179.

[37] Tchernykh, A., J. M. Ramirez, A. Avetisyan, N. Kuzjurin, D. Grushin, S. Zhuk, "Two Level Job-Scheduling Strategies for a Computational Grid", In Parallel Processing and Applied Mathematics, Springer-Verlag, 2006, 774–781.

[38] J. H. Abawajy, "Adaptive Hierarchical Scheduling Policy for Enterprise Grid Computing Systems", Journal of Network and Computer Applications, 32(2009), P 770-779.

[39] Thain D, Bent J, Arpaci-Dusseau A, Arpaci-Dusseau R, Livny M. "Gathering at the well: creating communities for grid i/o.", In: Proceedings of supercomputing, Denver, Colorado, November 2001.

[40]Joanna Kolodziej, Fatos Xhafa, "Meeting Security and User Behavior Requirements in Grid Scheduling", Simulation Modeling Practices and Theory(2010).

[41]Kołodziej, J. and Xhafa, F.: "A Game-Theoretic and Hybrid Genetic meta-heuristic Model for Security-Assured Scheduling of Independent Jobs in Computational Grids", Proc. of CISIS 2010, Cracow, 15-18.02.2010, in L. Barolli, F. Xhafa and S. Venticinque eds., IEEE Press, USA, 2010, pp. 93–100.

[42]Kwok, Y.-K., Hwang, K., and Song, S.: "Selfish Grids: Game-Theoretic Modeling and NAS/PSA Benchmark Evaluation", IEEE Transactions on Parallel and Distributing Systems, 18(5), 2007, pp. 1–16.

[43]Monir Abdullah, Mohamed Othman, Hamidah Ibrahim, Shamala Subramaniam, "Optimal workload allocation model for scheduling divisible data grid applications", Future Generation Computer Systems 26 (2010) 971_978.

[44]Y.C. Cheng, T.G. Robertazzi, "Distributed computation with communication delays", IEEE Transactions on Aerospace and Electronic Systems 22 (1988) 60_79.

[45]S. Kim, J.B. Weissman, "A genetic algorithm based approach for scheduling decomposable data grid appklications", in: IEEE Proceeding of the International Conference on Parallel Processing, Washington, DC, USA 1, 2004, pp. 406_413.

[46]M. Othman, M. Abdullah, H. Ibrahim, S. Subramaniam, "Adaptive divisible load model for scheduling data-intensive grid applications, in: Computational Science", in: Lecture Notes in Computer Science, vol. 4487, Springer Verlag, 2007, pp. 446_453.

[47]M. Othman, M. Abdullah, H. Ibrahim, S. Subramaniam, "A2DLT: divisible load balancing model for scheduling communication-intensive grid applications", in: Computational Science, in: Lecture Notes in Computer Science, vol. 5101,Springer Verlag, 2008, pp. 498_507.

[48]Chao-Chin Wu, Ren-Yi Sun, "An integrated security-aware job scheduling strategy for large-scale computational grids", Future Generation Computer Systems 26 (2010) 198_206.

[49]M. Srinivas, L.M. Patnaik, "Genetic algorithms: A survey", IEEE Computer 27 (6) (1994) 17_26.

[50]S. Song, Y.-K. Kwok, K. Hwang, "Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling", IEEE Transactions on Computers 55 (6) (2006) 703_719.

[51]Xin Liu, Chunming Qiao, Wei Wei, Xiang Yu, Ting Wang, Weisheng Hu, Wei Guo, and Min-You Wu, "Task Scheduling and Lightpath Establishment in Optical Grids", Journal Of Light Wave Technology, 2009, p 1796-1805.

[52]Ruay-Shiung Chang_, Jih-Sheng Chang, Po-Sheng Lin, "An ant algorithm for balanced job scheduling in grids", Future Generation Computer Systems 25 (2009) 20–27.

[53]Taiwan unigrid project portal site, http://www.unigrid.org.tw.

[54]GlobusToolkit,v4,http://www.globus.org/toolkit/downloads/4.0.4/.

[55]B.T. Benjamin Khoo, Bharadwaj Veeravalli, "Pro-active failure handling mechanisms for scheduling in grid computing Environments", J. Parallel Distrib. Comput. 70 (2010) 189_200.