Game Theoritic Scheduling

Izuchukwu George Enekwa
dept. electronics engineering
Hochschule Hamm-Lippstadt
Germany
izuchukwu-george.enekwa@stud.hshl.de

***Abstract*—Task scheduling has emerged as a significant area for research due to the rising popularity of cloud computing technologies. The cloud computing system's task scheduling issue is more complicated than that of a conventional distributed system. We developed a simplified model for the job scheduling system in cloud computing based on the examination of cloud computing in relevant literature. The simplified model in this study is built on game theory as a mathematical tool, in contrast to earlier studies on cloud computing task scheduling algorithm. A proposed task scheduling technique that takes into account the dependability of the balanced work is based on game theory. The job scheduling model for computer nodes is proposed and is based on the balanced scheduling method. The work of calculating the rate allocation strategy on the node is carried out using game strategy in the cooperative game model. The proposed approach provides a better optimization effect, as demonstrated through analysis of the experimental findings.**

I. Introduction

Computers today use a significant amount of electricity, and this percentage is rising. According to a research by Dataquest [7], the total power dissipated globally by PC processors increased from 160 MW in 1992 to 9 000 MW in 2001 [19]. It is now commonly acknowledged that power-aware computing is a problem that affects desktop and traditional computing contexts as well. It is no longer just a problem for mobile and real-time computing environments.Researchers and business are more recently focusing on multi-core CPUs since they can achieve higher performance by simultaneously running many threads [21] By placing numerous cores on a single chip, designers want to maintain performance improvement while relying focusing less on raw circuit speed and cutting back on power requirements for each performance unit. We discuss the issue of power-conscious task scheduling and mapping onto heterogeneous and homogeneous multi-core processor (HeMP) architectures in this study. Subject to performance limits and architectural considerations, the goal is to reduce the amount of energy used and the time needed to solve challenging computationally intensive scientific issues. Dynamic Voltage Scaling is the foundation of the majority of energy conservation methods (DVS). To reduce an application's energy requirements, the DVS approach allocates differential voltages to each sub-task. The remainder of the paper is structured as follows. For the section 1, we explain what Game Theory is, We outline a few approaches in Section 2 for dealing with power problems in computer systems. We discuss similar work on scheduling in Section 3. The suggested scheduling approach was discussed in Section 4. We analyze the findings of the experiment in Section 5 and offer some closing thoughts in Section 6.

II. Game Theory

The ideas in game theory offer a common language for describing, organizing, analyzing, and ultimately comprehending various strategical scenarios. Game theory often examines conflict situations, agent interactions, and agent decisions. A game in the game theory sense is comprised of a set (usually finite) number of players that interact in accordance with predetermined rules. These participants could be people, teams, organizations, businesses, associations, and so forth. They are interdependent since their interactions will have an effect on both the individual players and the entire group of players.To be more specific, a game is defined by a group of participants and their potential to play the game in accordance with the rules, or more specifically, by their collection of strategies.
A broad definition of game theory can be obtained from this description:

- The subject of game theory are situations, where the result for a player does not only depend on his own decisions, but also on the behaviour of the other players

Game theory is a subfield of mathematics, as we have shown in the preceding section. These games can be described using a common language provided by mathematics. Additionally, von Neumann's use of game theory in economics has been demonstrated. Game theory can be applied when there is rivalry for a resource to explain current behavior or to enhance tactics.

III. Multicore Systems

In 2007, the most sophisticated computers will have more than one billion transistors on a single chip. The effective access by a core to a memory block restricts concurrent access to the same block by other cores, even if many architectures provide shared access to global memory or caches. At high clock rates, memory hierarchy is crucial. To get good per-processor performance, data references within a core must be local. By making the most of

These processors have functional units, instruction and data prefetch queues, and instruction cycle speeds that are in the range of a few nanoseconds. The main architectural components of multi-core devices are succinctly described in the sections that follow. general intent Multiple related tasks can run simultaneously on various cores thanks to multi-core designs. instances of such

AMD, the IBM Cell Processor, and other designs

and multi-core Intel processors [13].The time and energy requirements of the cores will generally vary. Effectively utilizing this must be possible for the scheduling methods.

IV. Dynamic scaling

By dynamically scaling the processors' supply voltage and clock frequency, the dynamic voltage scaling (DVS) technology lowers energy loss. Energy is only the mathematical result of power and time.

$P_d = C_{ef} \cdot V_{dd}^2 \cdot f$, where C_{ef} is the switched capacitance, V_{dd} is the supply voltage, and f is the operating frequency [4]. The relationship between the supply voltage and the frequency is represented by $f = k \cdot (V_{dd} - V_t)^2 / V_{dd}$, where k is a constant of the circuit and V_t is the threshold voltage. The energy consumed to execute task T_i, E_i, is expressed by $E_i = C_{ef} \cdot V_{dd}^2 \cdot c_i$, where c_i is the number of cycles to execute the task. The supply voltage can be reduced by decreasing the processor speed. It also reduces energy consumption of task. A task's normal execution time at the maximum supply voltage is given as $compTime_i = c_i / f_{\max}$.

V. Research work

Resource scheduling and monitoring are both necessary for efficient energy utilization. The important connected works in the context of the suggested research are briefly described in the sections that follow. Techniques employed at compile time (static) can be utilized to lower CPU activity. [20] presents methods for rearranging an application's instructions in order to minimize switching activity between instructions.The goal of this work is to decrease the switching activity of a data link between the main memory and on-chip cache when instruction cache misses happen. [17] suggests a compilation service to lower Java programs' energy consumption. It gives a thorough analysis of the relative benefits and drawbacks, in terms of energy use, of shifting compilation to a server.[13] presents the algorithm. The partitioning technique is created in tandem with an energy cost model, although the time and spatial complexity of the approach is very high [14]. Utilization bounds are used by the majority of task-level scheduling algorithms [2] to schedule recurring tasks in a timely manner while using less energy. For tasks with precedence constraints [11], [19] and without precedence constraints [3] for parallel machines, some research has been published in the literature. Additionally, we have created a preliminary version of energy-aware resource allocation in distributed systems for multiple tasks without precedence constraints.

VI. The Proposed Algorithm

Since determining an ideal schedule is generally an NP-complete problem, academics have turned to

using a variety of methods, such as branch-and-bound, integer programming, searching, graph theory, and randomization, to create a large number of heuristics,

evolutionary techniques and genetic algorithms [12]. The purpose of energy-aware scheduling, which uses these strategies, is to balance the energy usage overall with the time it takes for a parallel application to finish.The scheduling/mapping issue transforms into a MOO issue where the execution time is balanced against total energy usage. Based on game theory, our suggested method can address this issue with

quick turnaround. In addition, it achieves both objectives in an even manner. demonstrates the operation of a hypothetical resource manager that makes use of static and dynamic scheduling algorithms based on available resources and the system's current state (for example, temperature), as well as multiple methods of achieving energy time trade-offs (These trade-off objectives are captured by a variety of scenarios that are described in a later section).We presum that an energy manager will use both static and adaptive scheduling algorithms when necessary. The manager can operate under an energy profile that is generated and updated using the sensory data for power, temperature, etc. The manager is a component of the kernel.The manager is compelled by the energy profile to pick one of the potential outcomes that enables a scheduling algorithm to operate under various objective functions and dynamically when resource conflicts arise in real-time. The main concerns and objective functions (scenarios) that a resource manager must take into account in order to incorporate energy-time tradeoffs are briefly discussed in the remaining portions of this section. Then, in different subsections, we explain our earlier work for applications with and without precedent constraints. We also include a brief description of any proposed work that is especially pertinent to each class in these subsections.

VII. Tradeoffs Between Energy and Performance

We assume that a resource manager can aim for the right trade-off between energy and time needs based on the environment's temperature, the importance of the applications, and pricing considerations. Our objective is to provide algorithms for a wide range of energy-time tradeoffs that the resource manager can use to good advantage. There is no one solution to such a multi-objective optimization problem [8]. The curve AB represents the pareto-optimal frontier for a particular application and multi-core processing machine, M, and along it, neither the energy consumption (P) nor the schedule length (SL) can be improved without degrading the other. The operating points for the problems of schedule length minimization and energy minimization, respectively, are noted as being at points B and A. We can select a suitable operating point along curve AB based on the current circumstance by constructing a MOO issue.

The issue can be resolved theoretically in a number of ways:

- Discover a method that produces pareto-optimal results quickly
- Recognize numerous real-world applications where the difficult MOO B a workable objective area Schedule for Power (T, M) (T, M)

These options may be selected by a higher-level energy manager for various applications or users. For instance, certain applications with strict requirements might call for greater service. The management may select a combination of the aforementioned circumstances over time to reduce fines and maximize overall performance goals because they cannot always meet all applications.

Scenario —Objective —Description		
1	Reduce energy consumption with allowable reduction in schedule length	Assume we know the schedule of a parallel application that minimizes the execution time on a HeMP. Determine (with or without success) a new schedule that tries to minimize the energy consumption on the HeMP assuming an additional fractional slack over the execution time. An important special case is when the additional slack is zero
2	Minimize time requirements given a energy budge	For each parallel application, we determine its normal energy requirement and then reduce maximum allowable energy by some factor, say 80time requirements for the given energy constraints
3	Meet task and energy constraints so that the overall penalty function is minimized	We are given a budget for energy and execution requirements and there are penalties to be incurred if either or both the requirements are not met. The goal is to minimize the overall penalty. Special cases correspond to cases when the penalties are large (infinite) only for energy or time constraints.

VIII. Scheduling

We take into consideration a HeMP processing unit, which consists of several heterogeneous cores, each of which has a DVS module. Below, we explain the Energy-Aware Task Allocation (EATA) problem.

HeMP = core1, core2, ..., corem. Each core is characterized by:

- The frequency of each core, fj, given in cycles per unit time. From frequency it is easy to obtain the speed of the core, Sj, which is simply the inverse of the frequency.
- The specific architecture of a core, A(corej), includes the type the core, its speed in GHz, I/O, local cache and/or memory in Bytes. Tasks: Consider a parallel application, T = t1, t2, ..., tn, where ti is a task. Each task is characterized by:

  The computational cycles, ci, that it needs to complete.

  (The assumption here is that the ci is known a priori.)

  The specific core architecture type, A(ti), that it needs to

complete its execution. The deadline, di, before each task has to complete its execution.

The application, T, has a deadline, D, which is satisfied if and only if all of its tasks' deadlines are satisfied. In contrast to real-time systems, this concept of a deadline is distinct. Due to the performance-energy trade-offs, the deadline in this case can be longer than the minimum execution time and represents the amount of time that the user is willing to tolerate.Each task's "architectural kind" identifies the essential elements needed to complete it. This fulfillment is what we refer to as a workable task-to-HeMP mapping. Only when all architectural requirements are met is the mapping complete; otherwise, it is not. When this occurs, a workable task-to-HeMP mapping is made:

1) If all the requirements for each job—computational cycles, core architecture, and deadline—are met, each task ti T can be mapped to at least one core j.
2) The deadline constraint of T is also satisfied.

A finite positive number, indicated by cij, represents the quantity of computational cycles needed by ti to run on corej. Tij = cij/Sij is the cycle rate at which ti will execute at a constant speed Sij. When a task, ti, is carried out on corej, pij quantity of electricity is consumed. Lowering the power could result in it missing its deadline because doing so will lower the core frequency and, in turn, core speed. EATA's goal is to identify the task-to-HeMP mapping so that the total energy used by the HeMP is reduced and each task is completed by its due date. This is therefore scenario 1. Officially, we can state:

$$\min \sum_{i=1}^{n}\sum_{j=i}^{m} p_{ij}x_{ij} \text{ such that } \min \max_{1\le j\le m} \sum_{i=1}^{n} t_{ij}x_{ij} \text{ subject}$$

to four constraints.

$$(1):\ x_{ij} \in \{0,1\}, i = 1,2,...,n;\ j = 1,2,...m.;$$

$$(2)\ t_i \to core_j, \forall i, \forall j, such\ that\ A(t_i) = A(core_j), then\ x_{ij} = 1;$$

$$(3):\ t_{ij}x_{ij} \le d_i, \forall i, \forall j, x_{ij} = 1,\ \left(t_{ij}x_{ij} \le d_i\right) \in \{0,1\};\ \text{and}$$

$$(4):\ \prod_{i=1}^{n}(t_{ij}x_{ij} \le d_i) = 1, \forall i, \forall j, x_{ij} = 1$$

The mapping constraint, constraint (1), states that when xij = 1, a task, ti, maps to a core, corej. This mapping is described in further detail in Constraint (2) in relation to the architectural requirements. The third constraint deals with meeting each task's deadline and the Boolean link between the deadline and the actual time the work was completed. Constraint (4) refers to the application's deadline restrictions, which are only valid if all of the tasks, di, I = 1, 2,... n, have valid deadlines.Instead than focusing on individual cores here, the goal is to maximize overall performance. We address the EATA problem using cooperative game theory.

issue. Game theory has two primary subfields:

Games can be cooperative or competitive. Despite a

part of "Theory of Games and Economic

The earliest book on game theory, Behavior" [16], is

It is noncooperative game theory, which was developed by Nash [15] and is devoted to cooperative game theory.

Nash equilibrium, which has prevailed, serves as an example.

primarily as a result of economic and political

types of strategy that have emerged in

microeconomics and related disciplines. Nash, however

As a result, equilibrium hasn't always been totally satisfactory.

idea for a solution. Although this notion has undergone various improvements—the most significant of which is subgame perfection—there is still no general agreement on which improvements are most advantageous and in which tactical circumstances. Other theories of equilibrium, less myopic than Nash and based on concepts of a game and its rules that are fundamentally different from Nash's, such as Backward Induction Equilibrium [5] and True Equilibrium [10], have proven beneficial but have not gained broad adoption. We demonstrated that simple cooperation for joint resource allocation was noticeably superior to no cooperation for a different scheduling problem [12].In a HeMP, rewards and incentives are distributed as a group since cores acting as players can only profit if the HeMP as a whole benefits from job completion. The idea of NBS, which argues that: "An NBS is a solution to a game in which participants utilize bargaining interactions to demand a portion of some entity, can be used to attain this collective benefit very effectively. The interactions go on until a solution is reached and every player gets what they want.The NBS's outstanding characteristic is its assurance of fairness and pareto-optimality. The system environment, which includes the objective (to collectively minimize makespan and energy consumption), the preference (pareto-optimality in terms of balancing the two objectives), and the additional requirements (allocation is fair on all the cores in the HeMP, and thus the load is balanced), makes NBS an excellent solution to our problem.To simultaneously reduce energy usage and makespan while upholding deadline limitations, we transform the EATA dilemma into a cooperative game theory problem. Elegant cooperative game theoretical strategies are used to transform a high complexity min-min-max optimization issue into a low complexity max-max-min optimization problem. The key advantage of this conversion, aside from lesser complexity, is that we can always ensure that the max-max-min optimization problem has a Bargaining Point, which leads to pareto-optimality and fairness. For the cooperative EATA game, we derive an algorithm (named NBS-EATA) for getting NBS. Our NBS-EATA method creates a method for swiftly determining the Bargaining Point by fusing traditional game theory methods with Kuhn-Tucker conditions and the Lagrangian.

IX. Conclusion

High-performance computing is beginning to be revolutionized by multicore processors. This paper addresses the problem of power-aware scheduling/mapping of tasks onto heterogeneous and homogeneous multi-core processor architectures. The goal of scheduling is to minimize the energy consumption and the computation time of computationally intensive problems. Conventional approaches that focus on maximizing a single objective do not adequately address the multiobjective optimization problem. We propose a game-theory approach as a solution. We formulate the problem as a cooperative game. Due to low convergence rates and high complexity, the classical cooperative game theoretical techniques such as the Nash axiomatic technique cannot be used to identify the Bargaining Point in this problem, even though it is guaranteed that there is a Bargaining Point. We then transform the problem into a max-max-min problem that can generate solutions in a short time.

X. Affidavit

I ENEKWA IZUCHUKWU GEORGE herewith declare that I have composed the present paper and work myself and without use of any other than the cited sources and aids. Sentences or parts of sentence quoted literally are marked as such: other references with regard to the statement and scope are indicated by full details of the publications concerned. The paper and work in the same or similar form has not been submitted to any examination body and has not been published. This paper was not yet, even in part, used in another examination or as a course performance.

References

[1] eA (formerly American Electronics Association) Report Cybernation, www.aeanet.org.

[2] . Abdelzaher and V. Sharma, "A Synthetic Utilization Bound for Aperiodic Tasks with Resource Requirements," Euromicro Conf. on Real Time Systems, 2003 pp. 67-75.

[3] . Aydin, R. Melhem, D. Moss , and P. MejaAlvarez, "Power-Aware Scheduling for Periodic Real-Time Tasks," IEEE Trans. on Computers, 53(5), May 2004, pp. 584-600.

[4] . Berman, A. Chien, K. Cooper, J. Dongarra, I. Foster, D. Gannon, L. Johnsson, K. Kennedy, C. Kesselman, J. Mellor-Crummey, D. Reed, L. Torczon, and R. Wolski, "The GrADS Project: Software Support for High-Level Grid Application Development," International Journal of High Performance Computing Applications, 15(4):327–344, Winter 2001.

[5] . S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley, ScaLAPACK Users' Guide, SIAM Publications, Philadelphia, 1997.

[6] . Brook and K. Rajamani, "Dynamic Power Management for Embedded Systems," IEEE Int'l SOC Conference, 2003, pp. 104-111.

[7] Dataquest, Electronically available at: http://data1.cde.ca.gov/dataquest/

[8] K. Deb, Multi-Objective Optimization using Evolutionary Algorithms, Wiley, 2001.

[9] Environment Protection Agency, Electronically available at: http://www.epa.gov/.

[10] J. Greenberg, The Theory of Social Situations: An Alternative Game-Theoretic Approach, Cambridge University Press, Cambridge, UK, 1990

[11] J. Kang and S. Ranka, "Dynamic Algorithms for Energy Minimization on Parallel Machines, Euromicro Intl. Conf. on Parallel," Distributedand Network-based Processing, 2008, to appear.

[12] S.U. Khan and I. Ahmad, "'Non-cooperative, Semi-cooperative, and Cooperative Gamesbased Grid Resource Allocation," Int'l Parallel and Distributed Processing Symposium, 2006.

[13] M. Lee, Y.S. Ryu, S. Hong, and C. Lee, "Performance Impact of Resource Conflicts On Chip Multi-Processor Servers," Applied Parallel Computing, State of the Art in Scientific Computing, Lecture Notes in Computer Science, Springer, Vol. 4699, 2007, pp. 67-75.

[14] Y.-H. Lu, T. Simunic, and G. De Micheli, "Software Controlled Power Management," in 7th Int'l Workshop on Hardware/Software Codesign, 1999, pp. 157-161.

[15] J. Nash, "Non-Cooperative Games," Annals of Mathematics, vol. 54, pp. 286-295, 1951.

[16] J. von Neumann and O. Morgenstern, Theory of Games and Economic Behavior, Princeton University Press, 1953.

[17] Rudenko, P. Reiher, G.J. Popek, and G.H. Kuenning, "The Remote Processing Framework for Portable Computer Power Saving," ACM Symposium on Applied Computing, 1999, pp. 31-42

[18] E. J. Rudkin G. L. and Loughnan, "Vortec – The Marine Energy Solution," Marine Renewable Energy Conference, 2001, pp. 67- 74.

[19] M. T. Schmitz and B. M. Al-Hashimi, "Considering Power Variations of DVS Processing Elements for Energy Minimisation in Distributed Systems," Int'l Symposium on System Synthesis, 2001, pp. 250-255.

[20] Q. F. Stout, "Minimizing Peak Energy on Meshconnected Systems," ACM Symposium on Parallelism in Algorithms and Architectures, 2006, pp. 331-331.

[21] S. Williams, L. Oliker, R. Vuduc, K. Yelick, J. Demmel, and J. Shalf, "Optimization of Sparse Matrix-vector Multiplication on Emerging Multicore Platforms," Int'l Conference on Supercomputing, 2007, pp. 37-46.