Using Game Theory for Scheduling Tasks on Multi-Core Processors for Simultaneous Optimization of Performance and Energy

Ishfaq Ahmad[1]
Department of Computer Science and Engineering
University of Texas at Arlington
iahmad@cse.uta.edu

Sanjay Ranka
Department of Computer Science
University of Florida
ranka@cis.ufl.edu

Samee Ullah Khan[1]
Department of Electrical and Computer Engineering, Colorado State University
samee.khan@colostate.edu

Abstract

Multi-core processors are beginning to revolutionize the landscape of high-performance computing. In this paper, we address the problem of power-aware scheduling/mapping of tasks onto heterogeneous and homogeneous multi-core processor architectures. The objective of scheduling is to minimize the energy consumption as well as the makespan of computationally intensive problems. The multi-objective optimization problem is not properly handled by conventional approaches that try to maximize a single objective. Our proposed solution is based on game theory. We formulate the problem as a cooperate game. Although we can guarantee the existence of a Bargaining Point in this problem, the classical cooperative game theoretical techniques such as the Nash axiomatic technique cannot be used to identify the Bargaining Point due to low convergence rates and high complexity. Hence, we transform the problem to a max-max-min problem such that it can generate solutions with fast turnaround time.

1. Introduction

A major portion of electricity in the USA is now being consumed by computers [1] and this number is growing. A study by Dataquest [7] reported that the world-wide total power dissipation of processors in PCs was 160MW in 1992, and by 2001 it had grown to 9000MW [19]. It is now widely recognized that power-aware computing is no longer an issue confined to mobile and real-time computing environments, but is important for de[1]sktop and conventional computing as well. More recently, industry and researchers are eyeing multi-core processors, which can attain higher performance by running multiple threads in parallel [21]. By integrating multiple cores on a chip, designers hope to sustain performance growth while depending less on raw circuit speed and decreasing the power requirements per unit of performance.

In this paper, we address the problem of power-aware scheduling/mapping of tasks onto heterogeneous and homogeneous multi-core processor (HeMP) architectures. The objective is to minimize the energy consumption and the makespan of complex computationally intensive scientific problems, subject to the performance constraints, architectural requirements. Most energy minimization techniques are based on Dynamic Voltage Scaling (DVS). The DVS technique assigns differential voltages to each sub-task to minimize energy requirements of an application.

The rest of the paper is organized as follows. In Section 2, we present some methods used to address power issues in computing systems. In Section 3 we present related work on scheduling. Section 4 described the proposed scheduling methodology. In Section 5 we discuss experimental results and give some concluding remarks in Section 6.

2. Background

2.1. Multi-Core Processors

Most advanced processors will surpass one billion transistors on a single chip in 2007. Although, many of the architectures support shared access of global memory or caches, effective access by a core to a memory block limits concurrent access to the same block by other cores. Memory hierarchy plays a critical role at high clock rates. Locality of data reference within a core is necessary to achieve good per-processor performance. By optimally utilizing

[1] The work was supported by an NSF grant # CSR-0615047

978-1-4244-1694-3/08/$25.00 ©2008 IEEE

instruction and data prefetch queues and the functional units, these processors are capable of instruction cycle times in the range of a few nanoseconds. In the following, we briefly describe the key architectural features of multi-core machines. General purpose multi-core architectures allow multiple related tasks to be executed on different cores. Examples of such architectures include the IBM Cell Processor, AMD and Intel Multi-core processors [13]. The cores in

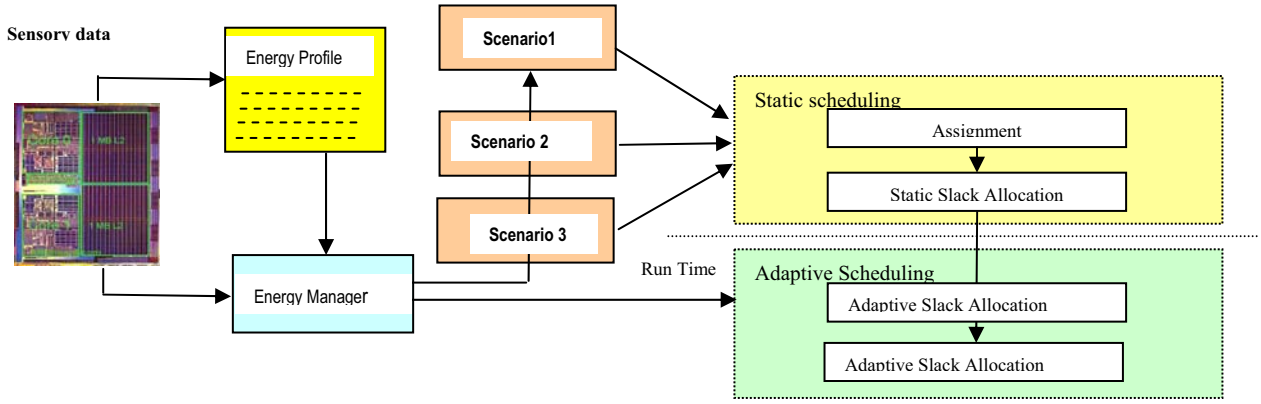

Figure 1: Energy-aware scheduling framework.

general will be heterogeneous in time and energy requirements. The scheduling algorithms have to be able to exploit this effectively.

2.2. Dynamic Voltage Scaling

Dynamic voltage scaling (DVS) technique reduces the energy dissipation by dynamically scaling the supply voltage and the clock frequency of processors. Mathematically, energy is simply a product of power and time. Power dissipation, P_d, is represented by $P_d = C_{ef} \cdot V_{dd}^2 \cdot f$, 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 $compTime_i = c_i / f_{\max}$.

3. Related Work

Effective energy consumption requires both monitoring and scheduling of resources. In the following, we briefly describe the key related wok in the context of the proposed research.

Compile time (static) techniques can be used to reduce the processor's activity. In [20], techniques for re-ordering the instructions of an application to reduce the switching activity between successive instructions are presented. This work focuses on reducing the switching activity of a data bus between the on-chip cache and main memory when instruction cache misses occur. A compilation service to reduce energy consumption of Java programs is proposed in [17]. It presents a detailed study on the relative merits and demerits of moving compilation to a server, with regards to energy.

A task partitioning approach using the MIN-CUT algorithm is presented in [13]. An energy cost model is developed along with the partitioning algorithm, but the time and space complexities associated with the algorithm are very high [14]. Most task-level scheduling algorithms use utilization bound for scheduling periodic tasks [2] to maintain the timeliness of processed jobs while conserving energy.

Some research has been presented in the literature for energy aware scheduling of tasks with precedence constraints [11], [19] and without precedence constrains [3] for parallel machines. We have also developed a preliminary version of energy-aware resource allocation in distributed systems for multiple tasks without precedence constraints. Our game theory based solution was shown to guarantee pareto-optimal solutions in mere $O(n\ m\log(m))$ time (where n is the number of tasks and m is the number of machines in the system).

4. Proposed Scheduling Approach

Since finding an optimal schedule is an NP-complete problem in general, researchers have resorted to devising a plethora of heuristics using a wide spectrum

of techniques, including branch-and-bound, integer-programming, searching, graph-theory, randomization, genetic algorithms, and evolutionary methods [12]. However, these methods are not applicable to energy-aware scheduling where the goal is to trade-off the completion time of a parallel application with the overall energy consumption.

The scheduling/mapping problem becomes a MOO problem in which the execution time is traded off with

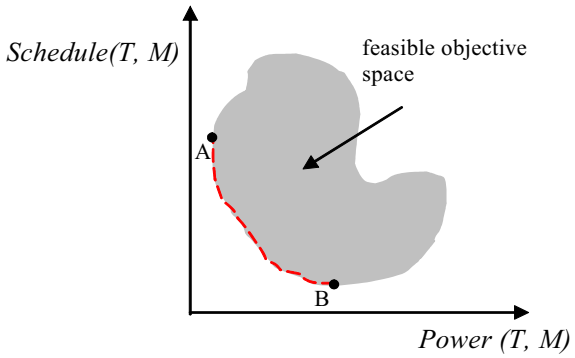

Figure 2: Illustration of MOO problem.

overall energy consumption. Our proposed solution is based on game theory that can solve this problem with fast turnaround time. It also strikes a balance between the two goals.

Figure 1 demonstrates the working of a hypothetical resource manager that utilizes multiple ways of achieving energy time trade-offs (These trade-off objectives are captured by multiple scenarios that are described in a later section), static and dynamic scheduling algorithms based on available resources and current state (for example temperature) of the system. We assume an energy manager invokes our scheduling algorithms, statically and then adaptively if needed. The manager, as part of the kernel, can work under an energy profile that is calculated and updated using the sensory data for power and temperature, etc. The energy profile drives the manager to choose one of the possible scenarios that enable a scheduling algorithm to work under different objective functions and dynamically when resource conflicts occur during run-time. In the rest of this section, we briefly describe the key issues and objective functions (scenarios) that must be considered to incorporate energy-time tradeoffs by a resource manager. We then describe our prior work for with and without precedent constrained applications in separate subsections. In these subsections, we also briefly describe proposed work that is of particularly relevance to each class.

4.1. Energy versus Performance Tradeoffs

Our assumption is that a resource manager can aim for the appropriate trade-off between energy and time requirements based on current temperature conditions, priorities of the applications and pricing issues. Our goal is to provide algorithms for a rich class of energy time tradeoffs that can be effectively utilized by the

Table 1: Three scenarios (and corresponding objective functions) for energy and execution time tradeoffs.

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 budget	For each parallel application, we determine its normal energy requirement and then reduce maximum allowable energy by some factor, say 80%. The goal is to find the minimum time 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.

resource manager. For such a multi-objectives optimization problem, there is no unique solution [8]. Figure 2 illustrates the concept of the MOO problem with conflicting objective functions. For a given application and multi-core processing machine, *M*, the curve *AB* is the pareto-optimal frontier, along which no further improvement can be done on energy consumption, *P,* or schedule length, *SL,* without sacrificing the other one. Note that points *B* and *A* are the operating points for the schedule length minimization problem and energy minimization problem, respectively. By formulating a MOO problem, we can choose an appropriate operating point along curve *AB* based on the current situation. There are several theoretical options to solve the problem: (a) Find a technique that yields pareto-optimal solutions in a reasonable amount of time; (b) Find various practical and useful scenarios, wherein the complex MOO

problem can be simplified by transforming it into simpler problems. Three scenarios for energy and execution time tradeoffs are described in Table 1.

A higher-level energy manager may choose from these scenarios for multiple applications or users. For example, some applications with stringent requirements may require better service. Since the manager cannot satisfy all the applications all the time it may choose a combination of the above scenarios over a period of time to minimize penalties and maximize the overall performance goals.

4.2. Scheduling for the First Scenario

We consider a HeMP processing unit, comprising a number of heterogeneous cores, each equipped with a DVS module. We call it the Energy-Aware Task Allocation (EATA) problem and elaborate it below.

HeMP = $\{core_1, core_2, \ldots, core_m\}$. Each core is characterized by:

1) The frequency of each core, f_j, given in cycles per unit time. With DVS, f_j can vary from f_j^{min} to f_j^{max}, where $0 < f_j^{min} < f_j^{max}$. From frequency it is easy to obtain the speed of the core, S_j, which is simply the inverse of the frequency.
2) The specific architecture of a core, $A(core_j)$, includes the type the core, its speed in GHz, I/O, local cache and/or memory in Bytes.

Tasks: Consider a parallel application, $T = \{t_1, t_2, \ldots, t_n\}$, where t_i is a task. Each task is characterized by:

1) The computational cycles, c_i, that it needs to complete. (The assumption here is that the c_i is known *a priori*.)
2) The specific core architecture type, $A(t_i)$, that it needs to complete its execution.
3) The deadline, d_i, before each task has to complete its execution.

The application, T, also has a deadline, D, which is met if and only if the deadlines of all its tasks are met. The notion of deadline here is different from real-time systems. Here, the deadline can be larger than the minimum execution time and represents the time that the user is willing to tolerate because of the performance-energy trade-offs. The "architectural type" of each task describes what core the task requires for its execution. We term this fulfillment as a feasible task-to-HeMP mapping. The mapping is only fulfilled when all of the architectural constraints are satisfied, otherwise not. A feasible task-to-HeMP mapping is achieved when:

1) Each task $t_i \in T$ can be mapped to at least one $core_j$ subject to the fulfillment of all the constraints associated with each task: Computational cycles, core architecture, and deadline.
2) The deadline constraint of T is also satisfied.

The number of computational cycles required by t_i to execute on $core_j$ is a finite positive number, denoted by c_{ij}. The execution time of t_i under a constant speed S_{ij}, given in cycles per second is $t_{ij} = c_{ij}/S_{ij}$. A task, t_i, when executed on $core_j$ draws, p_{ij} amount of power. Lowering the power will lower the core frequency and consequently will decrease the speed of the core, and hence will cause t_i to possibly miss its deadline. The objective of EATA is to determine the task-to-HeMP mapping, such that the total energy utilized by the HeMP is minimized and each task meets its deadline. Thus, this represents scenario 1. Formally, we can say:

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

to four constraints.

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

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

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

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

Constraint (1) is the mapping constraint, when x_{ij} =1, a task, t_i, is mapped to core, $core_j$. Constraint (2) elaborates on this mapping in conjunction to the architectural requirements. Constraint (3) relates to the fulfillment of the deadline of each task, and about the Boolean relationship between the deadline and the actual time of execution of the tasks. Constraint (4) relates to the deadline constraints of the application, which will hold if and only if the deadlines of all the tasks, d_i, i =1, 2, …n, are satisfied.

The objective here is to optimize **the cumulative performance** rather than to satisfy individual cores, we use cooperative game theory to solve the EATA problem. The two main branches of game theory are cooperative and non-cooperative games. Although a large portion of "Theory of Games and Economic Behavior" [16] (the first text on game theory) is devoted to cooperative game theory, it is non-cooperative game theory, pioneered by Nash [15] and epitomized by Nash equilibrium, that has held sway, largely due to the influence of economist and the strategic models that have developed in microeconomics and other fields. However, Nash equilibria have not proved completely satisfactory as a solution concept. Even though numerous refinements have been made in this concept, the most important being subgame perfection, consensus has not been reached on which refinements are most useful, and in what strategic situations. Other notions of equilibrium,

less myopic than Nash and grounded in theories that propose very different conceptions of a game and its rules of play, for instance, Backward Induction Equilibrium [5], True Equilibrium [10], have been useful but have not had widespread acceptance.

For a different scheduling problem, we showed that simple cooperation for joint resource allocation was significantly better than no cooperation [12]. Rewards and incentives are dealt as a collective entity in a HeMP because cores acting as players can benefit only if the overall HeMP can benefit from the execution of tasks. This collective benefit can be achieved very efficiently via the concept of NBS, which states that: "An NBS is a solution to a game in which players use bargaining interactions to demand a portion of some entity. The interactions continue till a resolve is met and all the players achieve their demands." The remarkable property of the NBS is that it guarantees pareto-optimality and fairness. Thus, NBS provides an excellent solution to our problem because of the system environment, including the objective (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 hence the load is balanced).

We convert the EATA problem into a cooperative game theory problem to minimize the energy consumption and the makespan simultaneously, while maintaining deadline constraints. A high complexity min-min-max optimization problem is converted using elegant cooperative game theoretical techniques into a low complexity max-max-min optimization problem. Besides lower complexity, the main benefit of this conversion is that we can always guarantee that the max-max-min optimization problem has a Bargaining Point and subsequently results in pareto-optimality and fairness. We derive an algorithm (called **NBS-EATA**) for obtaining NBS for the cooperative EATA game. Our NBS-EATA technique combines the classical game theoretical techniques with the Kuhn-Tucker conditions and the Lagrangian to derive a technique for identifying the Bargaining Point quickly.

5. Performance Evaluation

The proposed NBS-EATA technique was compared against a greedy heuristic and a LR technique. Figure 3 illustrates the relative performance of the techniques.

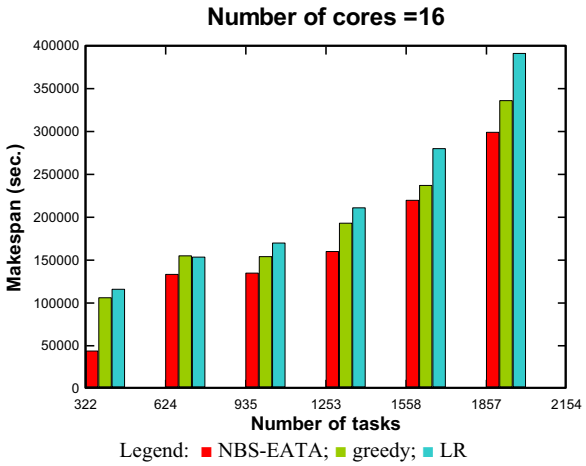

Figure 3: Makespan.

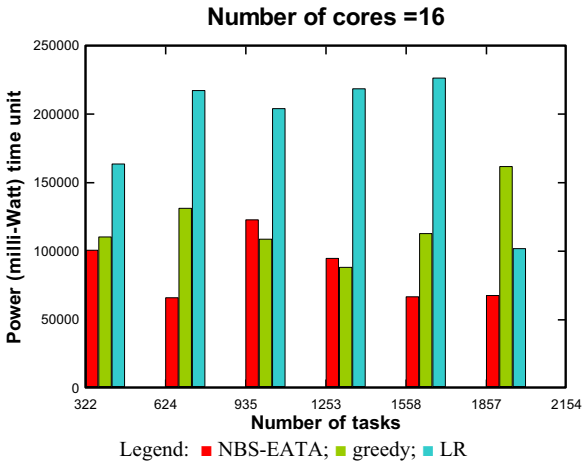

Figure 4: Power consumption.

The results indicate that NBS-EATA outperforms greedy and LR heuristics in identifying a smaller makespan. Next, we compare the power consumption of the NBS-EATA, greedy, and LR techniques (Figure 4). It can be seen that the NBS-EATA technique indentifies a makespan that is 22.75% smaller than greedy and 29.64% smaller than LR heuristic. Moreover, the NBS-EATA technique saves on average 24.32% more of energy consumption than the greedy and 50.61% more energy consumption than LR.

6. Conclusions

The proposed algorithm outperformed greedy and LR heuristics in identifying a smaller makespan and saving the energy. In our current work, we are investigating task graphs with precedence constraints. We are also investigating the second and third trade-off scenarios described in our problem formulation. With more complex multi-core architectures, resource constraints

need to be taken into account by the scheduler, statically or during time-time.

References

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

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

[4] F. 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] L. 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] B. 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," *Distributed and Network-based Processing*, 2008, to appear.

[12] S.U. Khan and I. Ahmad, "Non-cooperative, Semi-cooperative, and Cooperative Games-based 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 Mesh-connected 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.