International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:3, No:8, 2009

Energy Efficient Resource Allocation in Distributed Computing Systems

Samee Ullah Khan and Cemal Ardil

***Abstract*—**The problem of mapping tasks onto a computational grid with the aim to minimize the power consumption and the makespan subject to the constraints of deadlines and architectural requirements is considered in this paper. To solve this problem, we propose a solution from cooperative game theory based on the concept of Nash Bargaining Solution. The proposed game theoretical technique is compared against several traditional techniques. The experimental results show that when the deadline constraints are tight, the proposed technique achieves superior performance and reports competitive performance relative to the optimal solution.

***Keywords*—**Energy efficient algorithms, resource allocation, resource management, cooperative game theory.

I. Introduction

POWER management in various computing systems is widely recognized to be an important research problem. Power consumption of Internet and Web servers accounts for 8% of the US electricity consumption. By reducing the demand for energy, the amount of CO_2 produced each year by electricity generators can also be mitigated. For example, generating electricity for the next generation large-scale data centers would release about 25M tons of additional CO_2 each year [19]. Power consumption is also a critical and crucial problem in large distributed computing systems, such as, computational grids because they consume massive amounts of power and have high cooling costs. These systems must be designed to meet functional and timing requirements while being energy-efficient. The quality of service delivered by such systems depends not only on the accuracy of computations, but on their timeliness [36].

A computational grid (or a large distributed computing system) is composed of a set of heterogeneous machines, which may be geographically distributed heterogeneous multiprocessors that exploit task-level parallelism in applications. Resource allocation in computational grids is already a challenging problem due to the need to address deadline constraints and system heterogeneity. The problem becomes more challenging when power management is an additional design objective because power consumption of the system must be carefully balanced against other performance measures. Power management can be achieved by two methods. The Dynamic Power Management (DPM) [27] approach brings a processor into a power-down mode, where only certain parts of the computer system (e.g., clock generation and time circuits) are kept running, while the processor is in an idle state. The Dynamic Voltage Scaling (DVS) [34] approach exploits the convex relation between the CPU supply voltage and power consumption. The rationale behind DVS technique is to stretch out task execution time through CPU frequency and voltage reduction.

Power-aware resource allocation using DVS can be classified as static and dynamic techniques. Static techniques are applied at design time by allocating and scheduling resources using off-line approaches, while dynamic techniques control the runtime behavior of the systems to reduce power consumption.

The traditional resource allocation and scheduling theory deals with fixed CPU speed, and hence cannot be directly applied to this situation. In this paper, we study the problem of power-aware task allocation (PATA) for assigning a set of tasks onto a computational grid each equipped with DVS feature. The PATA problem is formulated as multi-constrained multi-objective extension of the Generalized Assignment Problem (GAP). PATA is then solved using a novel solution from cooperative game theory based on the celebrated Nash Bargaining Solution (NBS) [22]; we shall acronym this solution concept as NBS-PATA.

The rest of the paper is organized as follows: A brief discussion of related work is presented in Section II. The PATA problem formulation and background information are discussed in Section III. In Section IV, we model a cooperative game played among the machines for task allocation with the objective to minimize power consumption and makespan, simultaneously. Experimental results and concluding remarks are provided in Sections V and VI, respectively.

II. Related Work

Most DPM techniques utilize power management features supported by hardware. For example, in [4], the authors extend the operating system's power manager by an adaptive power manager (APM) that uses the processor's DVS capabilities to reduce or increase the CPU frequency thereby minimizing the overall energy consumption [6]. The DVS technique at the processor-level together with a turn on/off

S. U. Khan is with the Department of Electrical and Computer Engineering, North Dakota State University, Fargo, ND 58102, USA (phone: 701-231-7615; fax: 701-231-8677; e-mail: samee.khan@ ndsu.edu).

C. Ardil is with the National Academy of Aviation, Baku, Azerbaijan, (e-mail: cemalardil@gmail.com).

technique at the cluster-level to achieve high power savings while maintaining the response time is proposed in [11]. In [26] the authors introduce a scheme to concentrate the workload on a limited number of servers in a cluster such that the rest of the servers can remain switched-off for a longer time. Other techniques use a utilization bound for scheduling a-periodic tasks [1], [2] to maintain the timeliness of processed jobs while conserving power.

While the closest techniques to combining device power models to build a whole system has been presented in [13], our approach aims at building a general framework for autonomic power and performance management where we bring together and exploit existing device power management techniques from a whole system's perspective. Furthermore, while most power management techniques are either heuristic-based approaches [15], [16], [19], [30], or stochastic optimization techniques [10], [29], [34], we use game theory to seek radically fast and efficient solutions compared with the traditional approaches, e.g., heuristics, genetic algorithms, linear and dynamic programming, branch-and-bound etc. With game theoretical techniques the solution may not be globally optimal in the traditional sense, but would be optimal under given circumstances [17].

Another advantage of using game theory is that our overall management strategy allows to use lower level information to dynamically tune the high-level management policies freeing the need to execute complex algorithms [18]. The proposed generic management framework not only enables us to experiment with different types of power management techniques ranging from heuristic approaches but also provides a mechanism to consolidate power management with other autonomic management objectives pertinent to computational grids, such as fault-tolerance and security.

III. Problem Formulation

A. Background Information

The power consumption in CMOS circuits is captured by the following:

$$P = V^2 \times f \times C_{EFF}, \quad (1)$$

where V, f, and C_{EFF} are the supply voltage, clock frequency, and effective switched capacitance of the circuits. It is to be understood that time to finish an operation is inversely proportional to the clock frequency. This relationship can be extended to gather an insight on the energy consumption of the processor, by simply recalling that energy is power times time. Therefore, the energy per operation, E_{op}, is proportional to V^2. This implies that lowering the supply voltage will reduce the energy consumption of the system in a quadratic fashion. However, lowering the supply voltage also decreases the maximum achievable clock speed. More specifically, f is (approximately) linearly proportional to V [5]. Therefore, we have:

$$P \propto f^3, \text{ and } E_{op} \propto f^2. \quad (2)$$

A computing device's power consumption can be significantly reduced by running the device's CPU at a slower frequency. This is the key idea behind the DVS technology. In conventional system design with fixed supply voltage and clock frequency, clock cycles, and hence energy, are wasted when the CPU workload is light and the processor becomes idle. Reducing the supply voltage in conjunction with the clock frequency eliminates the idle cycles and saves the energy significantly. We have:

$$P \propto t^{-3}, \text{ and } E_{op} \propto t^{-2} \quad (3)$$

since $f \propto t^{-1}$, where t is the time to complete an operation. Thus, the reduction of the supply voltage would reduce the energy dissemination, it would substantially slow down the time to complete an operation – a balance is needed.

B. The System Model

We consider the system as a collection of machines that comprise the computational grid and the collection of tasks.

Machines: Consider a computational grid comprising of a set of machines, $M = \{m_1, m_2, \ldots, m_m\}$. Assume that each machine is equipped with a DVS module. Each machine is characterized by:

1. The frequency of the CPU, f_j, given in cycles per unit time. With the help of a 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 CPU, S_j, which is simply the inverse of the frequency.
2. The specific machine architecture, $A(m_j)$. The architecture would include the type of CPU (Intel, AMD, RISC), bus types and speeds in GHz, I/O, and memory in Bytes.

Tasks: Consider a metatask, $T = \{t_1, t_2, \ldots, t_n\}$, 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 machine architecture, $A(t_i)$, that it needs to complete its execution.

The deadline, d_i, before it has to complete its execution.

It is obvious that the metatask, T, also has a deadline, D, which is met if and only if the deadlines of all its tasks are met.

C. Preliminaries

Suppose we are given a computational grid and a metatask, T, and we are required to map T on the computational grid such that all the characteristics of the tasks and the deadline constraint of T are fulfilled. We term this fulfillment as a feasible task to machine mapping. A feasible task to machine mapping happens when:

1. Each task $t_i \in T$ can be mapped to at least one m_j subject to the fulfillment of all the constraints associated with each task: Computational cycles, architecture, and deadline.
2. The deadline constraint of T is also satisfied.
3. The number of computational cycles required by t_i to execute on m_j is assumed to be 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 machine m_j draws, p_{ij} amount of power. Lowering the power, will lower the CPU frequency and consequently will decrease the speed of the CPU, and hence cause t_i to possibly miss its deadline. For simplicity assume that the overhead of switching the CPU frequency is minimal and hence ignored.

The architectural requirements of each task are recorded as a tuple with each element bearing a specific requirement. We assume that the mapping of architectural requirements is a boolean operation. That is, the architectural mapping is only fulfilled when all of the architectural constraints are satisfied, otherwise not.

D.Problem Statement

Given is a computation grid and a metatask T. Find the task to machine mapping, where:

"The total power utilized by the computational grid is minimized such that the makespan of the metatask, T, is minimized."

Mathematically, we can say:

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

$$x_{ij} \in \{0,1\}, i = 1,2,...,n; j = 1,2,...m. \quad (4)$$

$$\text{if } t_i \to m_j, \forall i, \forall j, \text{ such that } A(t_i) = A(m_j), \text{ then } x_{ij} = 1 \quad (5)$$

$$t_{ij}x_{ij} \le d_i, \forall i, \forall j, x_{ij} = 1 \quad (6)$$

$$\left(t_{ij}x_{ij} \le d_i\right) \in \{0,1\} \quad (7)$$

$$\prod_{i=1}^{n} (t_{ij}x_{ij} \le d_i) = 1, \forall i, \forall j, x_{ij} = 1 \quad (8)$$

Constraint (4) is the mapping constraint, when x_{ij} =1, a task, t_i, is mapped to machine, m_j. Constraint (5) elaborates on this mapping in conjunction to the architectural requirements and it states that a mapping can only exists if the architecture is mapped. Constraint (6) relates to the fulfillment of the deadline of each task, and constraint (7) tells us about the boolean relationship between the deadline and the actual time of execution of the tasks. Constraint (8) relates to the deadline constraints of the metatask, which will hold if and only if all the deadlines of the tasks, d_i, i =1, 2, …n, are satisfied.

This formulation is in the same form as that of a GAP except for constraints (6), (7), and (8). The major difference between PATA and GAP is that the capacity of resources in PATA, in terms of the utilization of power, are defined in groups, whereas in case of GAP, they are defined individually.

IV. Proposed Game Theoretical Technique

We consider the system model described in Section 3. The cooperative game presented here considers each machine in the computational grid as a player. The goal of the players is to execute task in a manner that reduces the overall makespan of the metatask, while keeping the aggregate power consumption to its minimum. If p_{ij} is the power consumed and t_{ij} is the time taken by machine j to execute task i, then the objective of the cooperative game is to minimize both p_{ij} and t_{ij}. Hypothetically, we can express this cooperative game (**CG$_1$**) as:

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

subject to (4), (5), (6), (7), (8) and

$$\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \le P \quad (9)$$

$$p_{ij} \ge 0, i = 1,2,...,n; j = 1,2,...m \quad (10)$$

The conservation condition (9) states that the total power allocated is bounded. Clearly, the power consumption has to be a positive number (10). These constraints make the PATA problem convex. Otherwise, the NBS set of points can grow exponentially, making the extraction of an agreement point an NP-hard problem.

In transforming the problem, the above cooperative game (**CG$_1$**) is equivalent to the following cooperative game (**CG$_2$**):

$$\max \sum_{i=1}^{n}\sum_{j=i}^{m} \left(-p_{ij}x_{ij}\right) \text{ such that } \max \min_{1\le j\le m} \sum_{i=1}^{n} \left(-t_{ij}x_{ij}\right)$$

subject to (4), (5), (9)

$$-\left(t_{ij}x_{ij}\right) \ge -d_i, \forall i, \forall j, x_{ij} = 1 \quad (11)$$

$$\left(-\left(t_{ij}x_{ij}\right) \ge -d_i\right) \in \{0,1\} \quad (12)$$

$$\prod_{i=1}^{n} \left(-\left(t_{ij}x_{ij}\right) \ge -d_i\right) = 1, \forall i, \forall j, x_{ij} = 1 \quad (13)$$

$$-p_{ij} \le 0, i = 1,2,...,n; j = 1,2,...m. \quad (14)$$

Based on the above, we can formulate a cooperative game as follows. The machines in the computational grid are the players, each having $f_j(x) = - p_{ij}$ as an objective function. In a cooperative game, all players need to optimize their objective functions simultaneously.

Assume that all players are able to achieve performance strictly superior to the initial performance, that is the set $J = M$. The initial performance of player j is given by γ_j^0. This corresponds to the peak power consumption of the machine j. This will always be an agreeable point because this is the minimum acceptable performance, but another NBS with greater performance is desired by reducing the power as much

Input: Machines each initialized to its maximum power using the DVS $=\{dvs_1, dvs_2, \ldots dvs_l\}$.
Output: A task to machine mapping consumes the minimum power and that has the minimum possible makespan.

1. For $i = 1$ to n
2. Sort the machines in decreasing order of their current power consumption: $p_1 \geq p_2 \geq \ldots \geq p_m$.
3. $p_{av} = (\sum p_j)/m$
4. while $(p_{av} > p_m)$ do
   a. $m = m - 1$
   b. $p_{av} = (p_{av} - (p_{m+1}/m+1))\ (m+1/m)$
5. If the architectural constraint (5) is met then goto Step **5a** else goto Step **5c**.
   a. If the smallest dvs_l that satisfies the deadline constraint (6) is found then goto Step **5b** else goto Step **6**.
   b. Assign task t_i to machine m_m with dvs_l, update p_m and goto Step **2**.
   c. If $m > 1$ then $m = m - 1$ and goto Step **4** else goto Step **6**.
6. Initialize all machines to maximum power and goto Step **2**.

Fig. 1 The pseudo-code for the proposed technique

TABLE I
VARIOUS PARAMETRIC VALUES FOR THE COEFFICIENT OF VARIANCE METHOD

Variable	Values
V_{task}	$\{0.10, 0.50\}$
μ_{task}	2×10^{12}
α_{task}	$\{2, 10\}$
β_{task}	$\{2\times10^{11}, 1\times10^{12}\}$
V_{mach}	$\{0.10, 0.50\}$
μ_{mach}	$[1.315\times10^{10}, 6.5214\times10^{14}]$
α_{mach}	$\{2, 10\}$
β_{task}	$\{[1.32\times10^{9}, 6.58\times10^{9}], [6.52\times10^{13}, 3.26\times10^{14}]\}$

as possible.

Theorem 1 (NBS-PATA): *The NBS for the cooperative PATA game is determined by solving the following optimization problem.*

$$\max \sum_{i=1}^{n}\sum_{j=i}^{m}\left(\gamma_j^0 x_{ij} - p_{ij}x_{ij}\right) \text{ such that } \max \min_{1\leq j\leq m} \sum_{i=1}^{n} -t_{ij}x_{ij}$$

subject to (4), (5), (9), (10) (11), (12), (13).

Proof: Consider $f_j(\gamma_j) = -p_{ij}$ which is concave and bounded above, and hence guarantees a solution but with higher complexity. The set of solutions determined by the constraint is convex and compact, and hence is not always guaranteed but has a lower complexity. Using the fact that $f_j(\gamma_j) = -p_{ij}$ are 1-1 functions of p_{ij} the results follows. ■

Now, assume that the machines in the computational grid are sorted in the decreasing order of their current power consumption. Given such a list, we assign a task to the machine that is currently running on a power just above the weighted average power consumption. The assignment warrants adjusting the power consumption to the appropriate DVS level DVS = $\{dvs_1, dvs_2, \ldots dvs_l\}$ as to which the machine can guarantee the deadline associated with the task. This procedure is applied until a feasible solution is found.

Based on Theorem 1, we derive an algorithm (called **NBS-PATA**) for obtaining NBS for the cooperative PATA game. The pseudo-code for NBS-PATA is depicted in Fig. 1.

Theorem 2 (Correctness of the NBS-PATA): The power adjustments, p_{ij}s, computed by the NBS-PATA technique solve the optimization problem in Theorem 4.

Proof: The while loop in Step 4a finds the minimum index m for which

$$\gamma_m < \left(\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}\right) - P \Big/ m \tag{15}$$

If $|J| = |M|$ then it means that all γ_js are positive. Applying Theorem 6 and the proof follows.
If $|J| < |M|$ (which may be the most probable case) then we get

$$\gamma_{|J|} < \left(\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}\right) - P \Big/ |J| \tag{16}$$

when γ_js are sorted as $\gamma_1 \geq \gamma_2 \geq \ldots \geq \gamma_{|J|}$.

Now, we know that $\gamma_{|J|} \neq 0$. Also, we know that NBS-PATA will not allocate any tasks to $\gamma_{|J|}$ in the while loop. The only argument that we need to be concerned with is the integrity of the while loop. That can be answered by showing that at any given instance, the while loop always produces a task to machine assignment that is covered by Theorem 1. By definition, the while loop if stooped at an instance where the first k machines were dealt with, then those k machines will correspond to the k fastest machines that bare the power consumption of $\gamma_1, \gamma_2, \ldots, \gamma_k$. But we know from Theorem 1

that a machine j's best strategy is:

$$\gamma_j = \left(\left(\sum_{i=1}^{n} \sum_{\forall k \in M, k \neq j} p_{ik} \right) - P \right) \Big/ m \quad (17)$$

and in terms of power is given as:

$$p_{ij} = \left(\left(\sum_{i=1}^{n} \sum_{\forall k \in M, k \neq j} p_{ik} \right) - P \right) \Big/ m \quad (18)$$

This completes the proof. ■

The execution of NBS-PATA technique is in $O(n\, m \log(m))$. The complexity is determined by observing that Step 2 in Fig. 1 takes $O(m\log(m))$. Step 2 enclosed in Step 1 which takes $O(n)$. This polynomial time complexity is possible because the objective functions are convex; however, in general determining an NBS is an NP-hard problem [25].

V. Experiments and Discussions

In this section, we demonstrate the quality of solutions produced by our NBS-PATA technique. The simulation study assumed synthetic task sets (explained in the subsequent text). We set forth two major goals for our experimental study:

1. To measure and compare the performance of NBS-PATA against 1) the optimal solution and 2) the min-min heuristic [8].
2. To measure the impact of system parameter variations, such as, the tightness of the task related constraints.

Based on the size of the problems, the experiments were divided in two parts. For small size problems, we used an Integer Linear Programming tool called LINDO [28]. LINDO is useful to obtain optimal solutions, provided the problem size is relatively small. Hence for small problem sizes, the performance of the NBS-PATA is compared against 1) the optimal solution using LINDO and 2) the min-min heuristic [8]. The LINDO implementation and the min-min heuristic do not consider power as an optimization constraint; however, they are very effective for the optimization of the makespan. Thus, the comparison provides us with a wide range of results. On one extreme we have the optimal algorithm, on the other a technique which scales well with the corresponding increase in the problem size. For large size problems, it becomes impractical to compute the optimal solution by LINDO. Hence, we only consider comparisons against the min-min heuristic.

The system heterogeneity is captured by the distribution of the number of CPU cycles, c_{ij}, on different m_js. Let C denote the matrix composed by c_{ij}, where $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$. The C matrix was generated using the coefficient of variation method described in [3]. The method requires various inputs which are reported in Table I. d_i, the deadline of task t_i was generated using the method described in [36]. Let w_i be the largest value among the elements in the i-th row of C and let w_is corresponding machine be denoted by m_0. Let $Z = n/m$, where n is the number of tasks and m is the number of machines. d_i is calculated as $K \times (w_i/S_{m0}) \times Z$, where K is a pre-specified positive value for adjusting the relative deadlines of tasks and S_{m0} is the speed of machine m_0 running at DVS level of 100%. For this study, we keep the architectural affinity requirements confined to memory. (Adding other requirements such as, I/O, processor type, etc. will bear no affect on our experimental setup or theoretical results.) Each machine is assigned a memory on random from within the range [500-5000] GB, while each task is associated a corresponding memory requirement on random from within the range [20-50] MB.

For small size problems, the number of machines was fixed at 5, while the number of tasks varied from 20 to 40. The number of DVS levels per machine was set to 4. The frequencies of the machines were randomly mapped from within the range [200MHz-2000MHz]. We assumed that the potential difference of 1mV across a CMOS circuit generates a frequency of 1MHz. For large size problems, the number of machines was fixed at 16, while the number of tasks varied from 1000 to 5000. The number of DVS levels per m_j was set to 8. Other parameters were the same as those for small size problems.

Min-min: The Min-min heuristic begins with the set U of all unmapped tasks. Then, the set of minimum completion times, $CT = \{ct_i \mid ct_i = \min_j t_{ij}, \text{ for each } i \in U\}$. Next, the task with the overall minimum completion time from CT is selected and assigned to the corresponding machine. Lastly, the newly mapped task is removed from U, and the process repeats until all tasks are mapped.

Comparative results: The experimental results for small size problems with K equal to 1.5 and 1.0 are reported in Figs. 2 and 3. These figures show the ratio of the makespan obtained from the two techniques and the optimal. The plots clearly show that the NBS-PATA technique performs extremely well and achieves a performance level of 10%-15% of the optimal when K was set at a very tight bound 1.0.

For large problem instances, first, we compare the makespan identified by the min-min and the NBS-PATA technique. Since the min-min heuristic does not optimize power consumption, we compared the min-min with a version of NBS-PATA that ran on full power and also compared it with the (original) version that optimized power. Figs. 4 and 5 show the relative performance of the techniques with various values of K, V_{task}, and V_{mach}. The results indicate that NBS-PATA outperforms the min-min technique in identifying a smaller makespan when power is not considered as an optimization criteria. The performance of NBS-PATA is notably superior to the min- min technique when the deadline constraints are relatively loose. It can also be observed that NBS-PATA, when considering power as an optimization resource, identifies a task to machine mapping that produces a makespan that is within 5%-10% of the min-min technique. It was noticed that the relative performance of the min-min technique was much better for large size problems, compared with small size problems, because with the increase in the

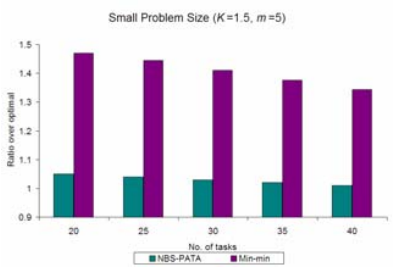

Fig. 2 Makespan ratio of min-min and NBS-PATA over optimal

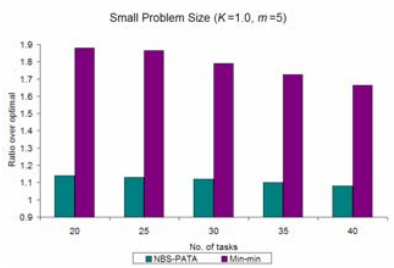

Fig. 3 Makespan ratio of min-min and NBS-PATA over optimal

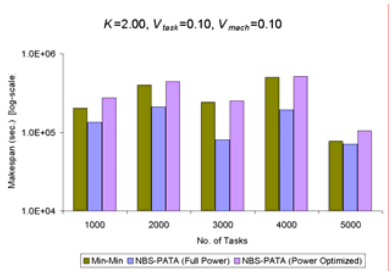

Fig. 4 Makespan

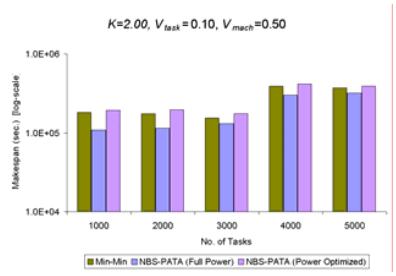

Fig. 5 Makespan

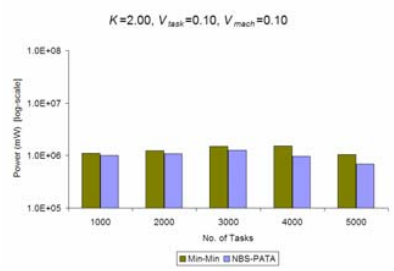

Fig. 6 Power consumption

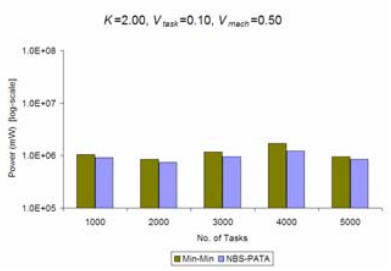

Fig. 7 Power consumption

TABLE II
AVERAGE EXECUTION TIME (SEC.) OF THE THREE TECHNIQUES

Problem size	$K = 1.5, m = 5$					$K = 1.0, m = 5$				
No. of tasks	*20*	*20*	*20*	*20*	*20*	*20*	*25*	*30*	*35*	*40*
Optimal	0.2732	0.2934	0.2934	0.2934	0.2934	0.2934	0.3470	0.4183	0.4938	0.5290
Min-Min	0.0032	0.0031	0.0031	0.0031	0.0031	0.0031	0.0042	0.0045	0.0052	0.0057
NBS-PATA	0.0673	0.0793	0.0793	0.0793	0.0793	0.0793	0.0872	0.0971	0.1004	0.1159

TABLE III
AVERAGE EXECUTION TIME (SEC.) OF THE THREE TECHNIQUES

Problem size	$K = 0.50, m = 16$					$K = 0.25, m = 16$				
No. of tasks	*1000*	*2000*	*3000*	*4000*	*5000*	*1000*	*2000*	*3000*	*4000*	*5000*
Min-Min	0.1395	0.1470	0.1537	0.1682	0.1753	0.1534	0.2985	0.3408	0.4485	0.5230
NBS-PATA	5.8401	8.3245	4.8314	6.4984	8.7219	5.7817	6.4629	7.3901	8.0293	9.6038

size of the C matrix, the probability of obtaining larger values of w_is also increases. Moreover, the relative performance of NBS-PATA was also much better for large size problems, compared with small size problems, because the DVS levels for the large problem size are twice more than the DVS levels for the small problem size.

Next, we compare the power consumption of both the techniques. Figs. 6 and 7 reveal that on average the NBS-technique utilizes 60%-65% less power as compared to the min-min technique. That is a significant amount of savings considering that the makespan identified by NBS-PATA is within 5%-10% of the makespan identified by the min-min technique.

Lastly, we analyze the running time for both small and large problem sizes. For completion, the running time of the optimal for small problem size is also presented for comparisons. It can be seen that when the number of tasks increase, the ratio of running time of NBS-PATA to that of the min-min heuristic decreases in the case of small problem size from 22 to 17 and in the case of large problem size from 56 to 18. The results are depicted in Tables II and III.

VI. CONCLUSIONS

This paper presented a power-aware resource allocation strategy in computational grids for multiple tasks. The problem was formulated as an extension of the Generalized Assignment Problem. A solution from cooperative game theory based on the concept of Nash Bargaining Solution (NBS-PATA) was proposed for this problem. We proved through rigorous mathematical proofs that the proposed NBS-PATA can 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). The solution quality of the NBS-PATA was experimentally compared against the optimal (for small problems sizes) and the min-min heuristic (for large problem sizes). The experimental results confirm superior performance of the proposed scheme in terms of reduction in power consumption, makespan compared to the min-min heuristic, and comparative solution compared to the optimal.

We plan to study power-aware resource allocation problems which fulfill application specific demands. We also plat to investigate power-aware resource allocation techniques that cater for dynamic environments and which allocate resource on run-time. Moreover, modeling, measuring, and optimizing the communication, the messages, and the energy costs offer new challenges, especially in sensor networks, primarily because the communication is carried out in an ad hoc environment. We plan on extending our current study to dynamic, real-time, and ad hoc environments.

REFERENCES

[1] T. Abdelzaher and V. Sharma, "A Synthetic Utilization Bound for Aperiodic Tasks with Resource Requirements," in *Euromicro Conference on Real Time Systems*, 2003.

[2] T. Abdelzaher and C. Lu, "Schedulability Analysis and Utilization Bounds for Highly Scalable Real-time Services," in *IEEE Real-Time Technology and Applications Symposium*, 2001.

[3] S. Ali, H. J. Siegel, M. Maheswaran, D. Hensgen and S. Ali, "Task Execution Time Modeling for Heterogeneous Computing Systems," in *9th Heterogeneous Computing Workshop*, May 2000, pp. 185-199.

[4] Application Specific and Automatic Power Management based on Whole Program Analysis, available at: http://cslab.snu.ac.kr/~egger/apm/final-report.pdf, 2004.

[5] H. Aydin, R. Melhem, D. Mosse and P. Mejya-Alvarez, "Dynamic and Aggressive Scheduling Techniques for Power-aware Real-time Systems," in *IEEE Real-Time Systems Symposium*, Dec. 2001, pp. 95-105.

[6] R. Bianchini and R. Rajamony, "Power and Energy Management for Server Systems," *IEEE Computer*, vol. 37, no. 11, pp. 68-74, 2004.

[7] S. J. Brams, *Theory of Moves*, Cambridge University Press, New York, USA, 1994.

[8] T. D. Braun, S. Ali, H. J. Siegel and A. A. Maciejewski, "Using the Min-Min Heuristic to Map tasks onto Heterogeneous High-performance Computing Systems," in *2nd Symposium of the Los Alamos Computer Science Institute*, Oct. 2001.

[9] A. P. Chandrakasan, S. Sheng and R. W. Brodersen, "Low Power CMOS Digital Design, *IEEE Journal of Solid-State Circuits*, 27(4):473-484, 1992.

[10] E. Chung, L. Benini, A. Bogliolo and G. Micheli, "Dynamic Power Management for non-stationary service requests", *IEEE Transactions on Computers*, vol. 51, no. 11, pp. 1345-1361, 2002.

[11] E. N. Elnozahy, M. Kistler, R. Rajamony, "Energy-Efficient Server Clusters," in *2nd Workshop on Power-Aware Computing Systems*, 2002.

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

[13] S. Gurumurthi, A. Sivasubramaniam, M.J. Irwin, N. Vijaykrishnan, M. Kandemir, T. Li and L.K. John, "Using Complete Machine Simulation for Software Power Estimation: The SoftWatt Approach," in *International Symposium on High Performance Computer Architecture*, 2000, pp. 141-150.

[14] I. Hong, G. Qu, M. Potkonajak and M. B. Srivastava, "Synthesis Techniques for Low-power Hard Real-time Systems on Variable Voltage Processors," in *IEEE Real-time Systems Symposium*, 1998, pp. 178-187.

[15] C. Hsu and U. Kremer, "The Design, Implementation, and Evaluation of a Compiler Algorithm for CPU Energy Reduction," in *International Conference on Programming Language Design and Implementation*, 2003.

[16] C.-H. Hwang and A. Wu, "A Predictive System Shutdown Method for Energy Saving of Event-driven Computation," in *International Conference on Computer-Aided Design*, 1997, pp. 28-32.

[17] S. U. Khan and I. Ahmad, "A Powerful Direct Mechanism for Optimal WWW Content Replication," in *19th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2005.

[18] S.U. Khan and I. Ahmad, "Non-cooperative, Semi-cooperative, and Cooperative Games-based Grid Resource Allocation," in *20th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2006.

[19] B. Khargharia, S. Hariri, F. Szidarovszky, M. Houri, H. El-Rewini, S. U. Khan, I. Ahmad, and M. S. Yousif, "Autonomic Power & Performance Management for Large-Scale Data Centers," NSF Next Generation Software Program Workshop, in *21th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2007.

[20] D. G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley, 1984.

[21] C. Montet and D. Serra, *Game Theory and Economics*, Palgrave, 2003.

[22] J. Nash, "The Bargaining Problem," *Econometrica*, vol. 18, pp. 155-162, 1950.

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

[24] M. J. Osborne and A. Rubistein, *A Course in Game Theory*, MIT Press, Massachusetts, U.S.A., 1994.

[25] C. H. Papadimitriou and M. Yannakakis, "On the Approximability of Trade-offs and Optimal Access of Web Sources," in *41st IEEE Symposium on Foundations of Computer Science*, 2000, pp. 86-92.

[26] E. Pinheiro, R. Bianchini, E. V. Carrera and T. Heath, "Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems," in *Workshop on Compilers and Operating Systems for Low Power*, 2001.

[27] Q. Qiu and M. Pedram, "Dynamic Power Management Continuous-time Markov Decision Processes," in *36th Design Automation Conference*, 1999, pp. 555-561.

[28] L. Schrage, *Linear, Integer, and Quadratic Programming with LINDO*, The Scientific Press, USA, 1986.

[29] T. Simunic, "Dynamic Management of Power Consumption", in *Power Aware Computing*, pp. 101-125, 2002.

[30] M. Srivastava, A. Chandrakasan and R. Brodersen, "Predictive System Shutdown and other Architectural Techniques for Energy Efficient Programmable Computation," *IEEE Transactions on VLSI Systems*, vol. 4, pp. 42-55, 1996.

[31] A. Stefanescu and M. W. Stefanescu, "The Arbitrated Solution for Multiobjective Convex Programming," *Rev. Roum. Math. Pure Applicat.*, vol. 29, pp. 593-598, 1984.

[32] T. E. Truman, T. Pering, R. Doering and R. W. Brodersen, "The InfoPad Multimedia Terminal: A Portable Device for Wireless Information Access," *IEEE Trans. Computers*, 47(10):1073-1087, 1998.

[33] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*, 3ed, Princeton University Press, 1953.

[34] M. Weiser, B. Welch, A. J. Demers and S. Shenker, "Scheduling for reduced CPU Energy," in *Symposium on Operating Systems Design and Implementation*, Nov. 1994, pp. 13-23.

[35] H. Yaiche, R. R. Mazumdar and C. Rosenberg, "A Game Theoretic Framework for Bandwidth Allocation and Pricing in Broadband Networks," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 667-678, 2000.

[36] Y. Yu and V. K. Prasanna, "Power-Aware Resource Allocation for Independent Tasks in Heterogeneous Real-Time Systems," in *9th IEEE International Conference on Parallel and Distributed Systems*, 2002.