2009 15th International Conference on Parallel and Distributed Systems

An Energy-aware Heuristic Scheduling Algorithm for Heterogeneous Clusters

Yu Li Yi Liu Depei Qian
Sino-Germen Joint Software Institute, Beihang University, BeiJing, China
yu.li@jsi.buaa.edu.cn yi.liu@jsi.buaa.edu.cn depeiq@buaa.edu.cn

Abstract

With the rapid development of supercomputers, the power consumption by large scale computer systems has become a big concern. How to reduce the power consumption is now a critical issue in designing high performance computers. Energy-aware scheduling for large scale clusters, especially the high performance heterogeneous ones, is one of the strategies for energy saving. Proposed in this paper is a novel energy-aware task scheduling algorithm (EAMM) for heterogeneous clusters, which is based on the general adaptive scheduling heuristics Min-Min algorithm. The algorithm is evaluated on a simulated heterogeneous cluster. The experiment results show that the new energy-aware algorithm can achieve a good time-energy trade-off and outperform the original Min-Min algorithm under various conditions.

1. Introduction

Supercomputers have been widely used to provide amazing computing capability for both commercial and scientific applications. However, huge power consumption has prevented the further application of the large-scale computer systems. According to the latest statistics [4, 10], computing center servers accounted for 1.2% of the electricity consumption in the United States in 2005, doubled since 2000. By 2008, 50% of the existing computing centers will have insufficient power supply, and power will be the second-highest operating cost in 70% of all the computing centers. How to reduce the power consumption is therefore a critical issue in designing high performance computers. Basically, processors, networks, disks and cooling system are the four key components of a cluster computer system which consume majority of the energy. The focus of our study is on reducing the processor energy consumption by putting the idle processors into different power-saving states while the scheduling procedures are being performed. Energy-aware scheduling for large scale clusters, especially the high performance heterogeneous ones, is one of the strategies to achieve this goal.

It is technically challenging to design an energy-aware scheduling algorithm for heterogeneous clusters because there are multiple designing parameters to take into account, including performance, energy efficiency and heterogeneity of the system. In this paper, an online dynamic power management strategy with multiple power-saving states is firstly introduced into cluster systems. Then we propose a novel energy-aware scheduling algorithm called Energy-aware Min-Min (EAMM for short), which is based on the classic scheduling algorithm Min-Min [2]. Empirical results show that on average the EAMM algorithm achieves up to 34% energy saving, while the execution time increases by no more than 1% under different task loads. Moreover, the algorithm exhibits excellent scalability as the cluster scales up from 16 to 256 nodes.

The rest of the paper is organized as follows. In section 2, we review the related work. We put our emphasis in Section 3 on defining mathematical models including an online power-saving state control model and an energy consumption model for heterogeneous clusters. In Section 4 we present the EAMM algorithm in detail. EAMM and the conventional Min-Min algorithms are compared in section 5 by using the extensive simulation results. Finally, we conclude our work in section 6.

2. Related Works

Several strategies for energy saving in heterogeneous clusters have been proposed and studied. In [6], Chase et al. illustrated a method of determining the aggregate system load and the minimal set of servers that can process the load. A similar idea leverages work in cluster load balancing to determine when to turn machines on or off to handle a given load [5, 24]. A critical problem for these ideas is that in order to turn lightly loaded machines off or to assign workload to newly turned-on machines, the task need to be transferred from one machine to another. But almost all the operating systems used in the real clusters, e.g. Windows, Unix and Linux, cannot support such kind of operations. So in their research specific OS features have to be developed and applied, which in turn limits the practicability of their approaches. In [9], Feng et al. proposed a method of finding the best match of the number of cluster nodes and their uniform frequency (called “energy gear” in their research). But they did not consider much about the effect of scheduling algorithms. There are indeed a few high-performance computers designed with energy-saving in mind, such as

1521-9097/09 $26.00 © 2009 IEEE
DOI 10.1109/ICPADS.2009.33

BlueGene/L [1], which uses a "system on a chip" to reduce energy consumption, and Green Destiny [3], which uses low-power Transmeta nodes. But their concern on energy saving is only confined to the design of hardware, with nothing to do with the strategies for power control at run-time, which also plays an important role.

There is also a large effort in saving energy for desktop and mobile systems. In fact, most of the early researches in energy-aware computing were on these systems. At the system level, there has been work in trying to make the OS energy-aware by making energy the first class resource [8, 12, 16]. On device-specific energy saving, studies have been conducted on saving the energy consumed by CPU [11, 13], by disk [17], and by memory and network [19, 20]. A number of good methods and ideas in these studies could be introduced to the energy saving schemes in cluster systems.

In this paper, we design a novel energy-aware scheduling algorithm specifically for heterogeneous cluster computer systems. The algorithm, called EAMM, introduces an online power-saving strategy into cluster systems, which can be implemented in the common operating systems (Linux/Windows) and make good tradeoffs between performance and energy savings.

3. Online Power-saving State Control Strategy and Energy Consumption Model

In this section, we first illustrate the online power-saving state control strategy to be employed into cluster systems in our EAMM algorithm which will be described later. Then, based on the strategy, we propose the energy consumption model.

3.1 Online Power-saving State Control Strategy

Most of the current desktop and server systems support power-saving mode. The energy saving control can be at the level of the system board, the processor, the memory, the disk, etc. Standards related to multi-power state management, such as APM and ACPI [12], have been accepted by the manufacturers. Bear in mind the fact that even in the idle condition, the computer consumes up to 70 percent of the full-load energy [5]. This means that making power-mode-related decisions based on information available at runtime plays an important role in energy saving. Among a handful of previous studies for desktop and mobile systems, we introduce an online strategy called OPSCS [14] into cluster systems. The strategy can be briefly described as follows.

Suppose that each node in the cluster can be in one of the *k+1* power states denoted by *{state 0, …, state k}*. The power consumption for state *i* is denoted by p_i. The states are ordered so that $p_i > p_j$ if and only if $i < j$. Thus, state 0 is the active state with the highest power consumption. The strategy for state switching during individual idle periods can be described by a sequence of thresholds, each of which is associated with a power consumption state. As soon as an idle period is beyond a given threshold, the node switches to the associated power consumption state. To simplify the model, we only consider the time and the energy necessary to power up the system. And once the idle state is broken, the node will only switch to the active state. In other words, we need only the time and the total energy consumed in switching from each state *i* to the active state (state 0). The total energy used in such a switching is denoted by β_i and the switching time is denoted by w_i.

To get the optimal cost, we use lines $e = p_i t + \beta_i$ to represent the energy cost of spending the entire interval in state *i,* which is a function of *t*, the length of the interval. Take the lower envelope of all these lines. Let *LE(t)* denote the optimal cost for an interval of length *t*. We have $LE(t) = \min_i (p_i t + \beta_i)$. The online algorithm follows the function *LE(t)*, meaning it will remain in state *j* as long as $p_j t + \beta_j = \min_i (p_i t + \beta_i)$, for the current time *t*. For $j = 1, \ldots, k$, let t_j be the time needed for the cluster node to switch from state $j-1$ to state *j*, and $t_1 < t_2 < \cdots < t_{k-1} < t_k$. Then t_j is the solution to the equation $p_j t + \beta_j = p_{j-1} t + \beta_{j-1}$. The online strategy can be illustrated in Figure 1.

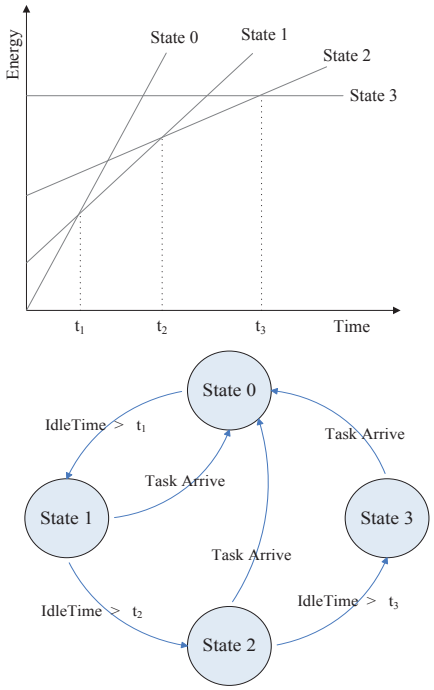

Figure 1. An illustration of online power control strategy

3.2 Energy Consumption Model

Based on the above node power control strategy, if we do not consider the influence of disk, network and cooling system, the energy consumption model of a heterogeneous cluster can be expressed as:

$$E = \sum_{i=1}^{n} en_i = \sum_{i=1}^{n} (\sum_{x=0}^{k} p_{ix} \cdot t_{ix} + \sum_{j=0}^{l} \beta_{ij}),$$

where n is the number of the nodes, en_i the energy consumption caused by node i, k the number of power states of node i, p_{ix} the power consumption rate of node i in state x, t_{ix} the total time that node i stays in state x, l the total state-switching times of node i, and β_{ij} the transition energy of node i in the j_{th} state-switching.

4. EAMM Algorithm

In this section, we present the EAMM algorithm in detail. Basically, given parallel applications, the objective of our scheduling algorithm is to allocate and schedule the parallel tasks to some computing nodes in a way of reducing energy consumption while limiting the increase of its execution time. By mapping it to a scheduling problem which has already been proved to be NP-complete [7], the scheduling algorithm studied here is easily proved to be NP-hard. Therefore, the proposed scheduling algorithm is heuristic in the sense that it can produce suboptimal solutions in polynomial-time.

4.1 Terminology

The terminology to be used hereafter is as follows [2, 22].

(1) Task Set T: $T = \{t_1, t_2 \ldots t_k\}$, where t_i represents task i

(2) Node Set N: $N = \{n_1, n_2 \ldots n_k\}$, where n_j represents computing node j.

(3) The expected execution time ET_{ij}: the amount of time taken by node n_j to execute task t_i given that n_j has no load when t_i is assigned.

(4) The expected completion time CT_{ij}: the wall-clock time at which n_j completes t_i (after having finished all previously assigned tasks).

(5) Let b_{ij} be the beginning time of t_i if node j executes t_j. From (3) and (4) we have $CT_{ij} = ET_{ij} + b_{ij}$.

4.2 Algorithm Description

The existing heuristic scheduling can be classified into two categories: online mode and batch mode. Online mode scheduling maps a task to some computing nodes as soon as it arrives at the scheduler. Batch mode scheduling implies that the task is not mapped to the nodes as it arrives. Instead, the tasks are collected in a set that is examined for mapping at a prescheduled time called a mapping event. Our algorithm refers to the batch mode scheduling.

Several simple heuristics for scheduling independent tasks are proposed in [2, 15], including Min-Min, Max-Min, Sufferage and XSufferage. The general structure of the algorithm [18] is shown as follows.

Algorithm 1: General Adaptive Scheduling Algorithm

Input: Task Set T
Cluster Node Set N
Output: Task-Node mapping

```
1   while there are tasks to schedule
2     for each t_i ∈ T
3       for each n_j ∈ N
4         Compute CT_ij = ET_ij + b_ij
5       end for
6       Compute metric_i = f_1(CT_i1, CT_i2, ...)
7     end for
8     Select best metric match
      (t, n) = f_2(metric_1, metric_2, ...)
9     Compute minimum CT_t,n
10    Schedule task t on n
11  end while
```

The scheduling algorithm iteratively assigns tasks to processors by computing their expected Minimum Completion Time (MCT). For each task, this is done by at first tentatively scheduling it to each node (line3) and then estimating the task's completion time on each node (line 4). Also for each task, a metric function "f_1" is computed over all expected completion times (line 6). Then the task/node pair with the best metric match *(t,n)* is selected by using selection function "f_2" (line 8). After that, the MCT of this task/node pair is computed (line 9) and the task t is assigned to the node n (line 10). The process is repeated until all the tasks have been scheduled (line 1 and line 11).

The difference among the four heuristics, Min-Min, Max-Min, Sufferage, and Xsufferage, lies in the different definitions of the evaluation function "f_1" and that of the best metric match selection function "f_2". For example, both Min-Min and Max-Min heuristics define "f_1" as the minimum completion time. That is, for task *i,* they select the match with the minimum completion time over all the nodes. However, for "f_2", Min-Min selects the minimum completion time over all tasks (the minimum among $metric_i$), whereas Max-Min selects the maximum completion time over all tasks (the maximum among $metric_i$).

From the description given above we see that the previous adaptive scheduling algorithms do not take energy consumption into consideration. We propose a

novel energy-aware scheduling algorithm EAMM based on the general adaptive algorithm. It is shown as follows.

Algorithm 2: Energy-aware Min-Min Algorithm

Input: Task Set T
Cluster Node Set N
Output: Task-Node mapping

```
1   while there are tasks to schedule
2     for each t_i ∈ T
3       for each n_j ∈ N
4         Get power-state S_j of node n_j
5         if S_j ≠ 0
6           Modify the beginning time of t_i:
      b_ij' = b_ij + w_j
7         end if
8         Compute CT_ij = ET_ij + b_ij'
9       end for
10      Compute metric_i = min{CT_i1, CT_i2, ...}
11    end for
12    Select best metric match
      (t, m) = min{metric_1, metric_2, ...}
13    Compute minimum CT_m,n
14    Schedule task m on n
15    Modify b_ij = b_ij + ET_m,n, j ≠ n for each task i
      not scheduled yet. And set each node to the proper
      power state according to the online strategy
      OPSCS
16  end while
```

In this energy-aware Min-Min heuristic scheduling algorithm, we use the strategy OPSCS mentioned in Section 3.1 to dynamically control the power state of each node in the cluster. This control will influence the task beginning time. So we need to modify the beginning time b_{ij} (line 6) according to the node's power state before computing the task completing time CT_{ij} (line 8). For the same reason we need to modify the nodes' power state after finishing every schedule iteration (line 15). Besides, we also need to modify the beginning time of each unscheduled task on node n which has just been assigned the new task m before the next schedule iteration (line 15).

5. Performance Evaluation

In this section, we evaluate the performance of EAMM. The experiments were carried out by simulation. The EAMM was compared with the general adaptive algorithm Min-Min in both the energy consumption and the execution time.

5.1 Overview

It is difficult to precisely evaluate the performance of a scheduling algorithm for heterogeneous cluster computing because it involves a number of parameters such as task execution time and heterogeneous hardware of the nodes. Performing tests on a specific cluster are not universally meaningful because the results are only valid for that cluster. Simulation allows us to demonstrate the effectiveness of our scheduling algorithm across a broad range of conditions, though there is no generally accepted set of benchmarks.

In our simulation studies, all tasks to be scheduled are initially known (batch processing) and independent, and the network traffic is not simulated in detail. By these assumptions, the need to accurately simulate specific features of the cluster is minimized. In this particular case, we also assume single-programming, i.e., each machine executes a single task at a time. It is a reasonable simplification for isolating the effects of EAMM and Min-Min.

5.2 Simulation Set-up

The simulation set-up is composed of a task generator, a scheduler and a nodes simulator, as illustrated in Figure 2.

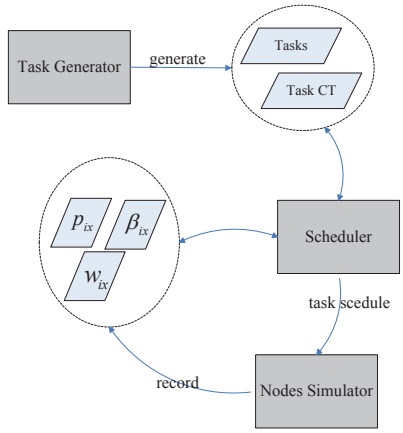

Figure 2. Simulation Setup

The task generator generates independent tasks and the expected task completion time (CT). The execution time of the tasks follows an exponential distribution and the tasks arrival follows Poisson distribution in a typical academic environment [23]. So in our simulation experiment, task execution time is randomly generated by using the exponential distribution with a mean of 1000 ($\mu = 1000$), just as what the previous research did [21], and the task generation rate follows the Poisson distribution with a mean of 10 ($\lambda = 10$).

Both EAMM and Min-Min are called by the scheduler to allocate tasks to the nodes according to the node status information including the power consumption rate p_{ix}, the

transition energy β_{ix}, the transition time w_{ix}, and the expected task completion time (CT).

The nodes simulator simulates the execution of tasks, records the change of node status and computes the total execution time (makespan) and the total energy consumption. All data about a node's multiple power states comes from real machines [9, 14].

We firstly compare the performance of EAMM and Min-Min with fixed cluster scale and scheduling interval but varied task workloads. Then experiments are carried out with fixed scheduling interval and task workload but varied cluster scales. Finally, we fix the workload and the cluster scale, and allow the scheduling interval to change. The purpose of using a combination of different experiments is to show that EAMM is generally applicable.

5.3 Experiment Results

In this section we give the experiment results and the performance evaluation of EAMM. In the following tables, the data in the column of "*Comparison*" are obtained by calculating $\frac{X_{EAMM} - X_{MinMin}}{X_{MinMin}}$, where X is the item to be compared.

5.3.1 Performance under different task workloads

In this experiment we set the cluster scale to fixed 128 nodes, the schedule interval to fixed 15s, the number of tasks (loads) ranging from 1000 to 10000. The task execution time follows the exponential distribution with $\mu = 1000$ and the task arrival rate follows the Poisson distribution with $\lambda = 10$. All the experiment scenarios are numbered from 1 to 5. The results are shown in Table 1 and Figure 3.

Table 1. Experiment results under different task workloads

Load	Makespan(s)		Energy Consumption(J)		Comparison	
	Min-Min	EAMM	Min-Min	EAMM	Make Span	Energy Consume
1000	167.76	168.02	1156196.32	602600.81	0.15%	-47.88%
3000	335.32	337.74	2605263.14	1713991.01	0.72%	-34.21%
5000	526.85	533.88	4206501.10	2834340.80	1.33%	-32.62%
8000	798.79	806.69	6555012.65	4746660.82	0.99%	-27.58%
10000	956.95	969.25	7879747.35	5633417.54	1.29%	-28.51%

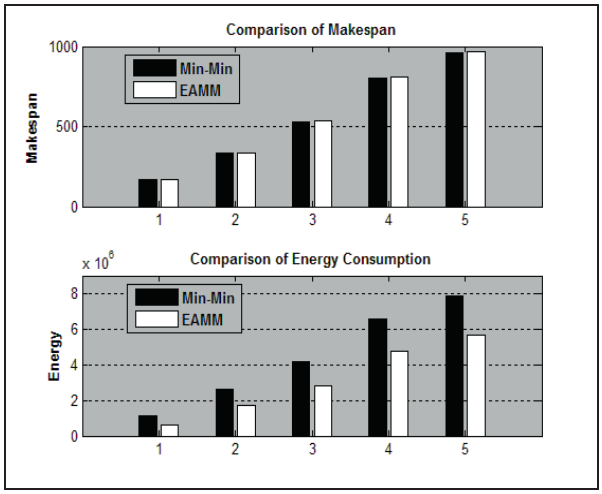

Figure 3. Experiment results under different task workloads

As shown in Figure 3, for all five scenarios the EAMM outperforms the conventional Min-Min algorithm with regard to energy consumption. In our test scenarios, the energy consumption by EAMM can be up to 48% lower than that by the conventional Min-Min, and the smallest energy reduction is 27%. We can also see from the results that as the number of tasks i.e. workloads increases, the energy saved by EAMM declines. This is because the higher the workload, the lower the probability of a node entering the low-power state, thus the less the power energy saving.

With regard to makespan, EAMM's performance is almost identical to the conventional Min-Min, with the largest increment to makespan being only 1.5%. This is because that the sleep-wakeup mechanism used in the online power state control strategy partially resolves the workload imbalance caused by the local optimum of the conventional Min-Min so as to reduce the overall execution time. And this time-reduction compensates the prolongation of makespan caused by state transition.

We can see that EAMM outperforms the conventional Min-Min heuristics under different task workloads.

5.3.2 Performance under different cluster scales

In this experiment we fix the number of tasks at 1000 and the schedule interval at 15s. The cluster scale varies from 16 to 256. Again the task execution time follows the exponential distribution with $\mu = 1000$ and the task arrival rate follows the Poisson distribution with $\lambda = 10$. All the experiment scenarios are numbered from 1 to 5. The results are shown in Table 2 and Figure 4.

Table 2. Experiment results under different cluster scales

Cluster Scale	Makespan(s)		Energy Consumption(J)		Comparison	
	Min-Min	EAMM	Min-Min	EAMM	Make Span	Energy Consume
16	4391.59	4506.52	4801454.75	4618731.76	2.62%	-3.81%
32	1339.74	1344.93	2757507.52	2381782.34	0.39%	-13.63%
64	425.79	411.13	1656898.59	1301948.88	-3.44%	-21.42%
128	167.76	168.02	1156196.32	602600.81	0.15%	-47.88%
256	114.05	117.85	1330800.69	137322.64	3.33%	-89.68%

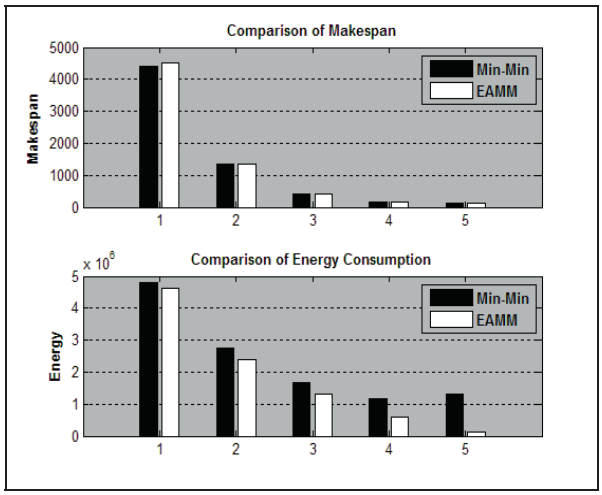

Figure 4. Experiment results under different cluster scales

It can be seen from the results that as the scale of the cluster increases, the difference in energy consumption between EAMM and Min-Min becomes larger, and the reduction increases from 3.81% to 89.68%. This shows that EAMM is well scalable under different cluster scales.

Meanwhile, the results also indicate that the energy reduction is not so significant when the cluster scale is small. But this is not a problem in the real world as most of the clusters are of large scale.

5.3.3 **Performance under different schedule intervals**

The scheduling interval has a significant impact to the efficiency of the scheduling algorithms. If the interval between scheduling events is too small, one can get only the local optimum. If the interval is too large, some nodes will become idle before the next scheduling event so that the resource cannot be fully utilized. With energy-saving parameter being introduced, we need to study the impact of schedule interval on both algorithms.

In this experiment we set the number of tasks to 1000 and the cluster scale to 128. The scheduling interval changes from 5s to 40s. The task execution time follows the exponential distribution with $\mu = 1000$ and task arrival rate follows the Poisson distribution with $\lambda = 10$. The results are shown in Table 3 and Figure 5.

Table 3. Experiment results with different schedule intervals

Schedule Interval(s)	Makespan(s)		Energy Consumption(J)	
	Min-Min	EAMM	Min-Min	EAMM
5	986.02	990.42	8183052.92	7042013.09
10	985.38	988.40	8079430.39	6313237.80
15	987.97	998.42	8035652.62	5683256.33
20	998.25	1008.08	8019005.17	5031789.43
25	1003.61	1005.94	7996590.82	4492209.38
30	1003.61	1008.95	7950829.73	4042520.00
35	1006.23	1016.01	7915487.10	3685702.04
40	1018.25	1025.9	7936075.52	3255458.65

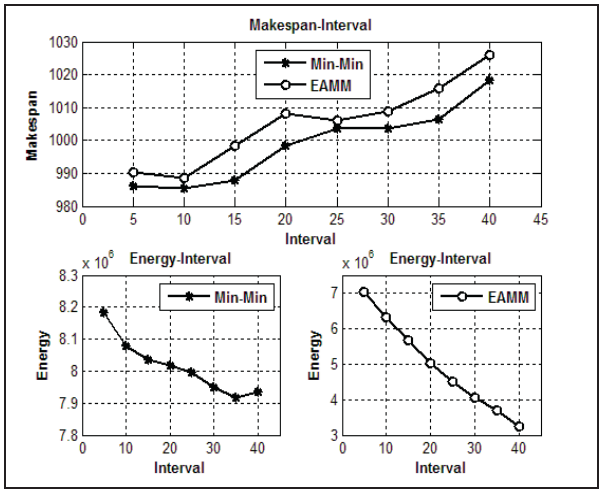

Figure 5. Experiment results with different schedule intervals

We can learn from Figure 5 the influence of the scheduling intervals on the performance of both algorithms. With regard to makespan, there is no obvious deviation in the trend between the two curves, and both algorithms achieve the optimum makespans when the scheduling internal is 10s. As for the energy consumption, because the longer the scheduling interval, the larger the probability for the nodes to enter the low power state with EAMM, we see that the energy-interval curve of EAMM monotonically declines and the energy consumption becomes smaller, whereas the curve of Min-Min has a turning point at the scheduling interval of 35s, after which the energy consumption increases.

6. Conclusion

In this paper, we propose an energy-aware scheduling algorithm EAMM to reduce energy consumption in heterogeneous clusters. Evolved from the conventional Min-Min algorithm, EAMM is implemented with extra

consideration to the energy consumption parameter. The performance of EAMM in energy saving is evaluated by simulation experiments and compared with that of conventional Min-Min algorithm. The empirical results show that EAMM can significantly reduce energy consumption of large heterogeneous cluster systems with only a marginal degradation in makespan performance. Furthermore, EAMM also has good scalability under different scales of the cluster systems.

Acknowledgment

The work reported in this paper is supported by the National Hi-tech Research and Development Program (863 Project) of China under grant No. 2009AA01Z107 and 2007AA01A127.

References

[1] N. Adiga et al. An overview of the BlueGene/L supercomputer. In Supercomputing 2002, Baltimore, Maryland, Nov. 2002.

[2] M.Maheswaran, S.Ali, H.J.Siegel et al. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems[C]. In the 8th IEEE Heterogeneous Computing Workshop (HCW'99), San Juan, Puerto Rico, Apr. 1999, pp. 30-44.

[3] M. Warren, E. Weigle, and W. Feng. High-density computing: A 240-node beowulf in one cubic meter. In Supercomputing 2002, Baltimore, Maryland, Nov. 2002.

[4] Gartner Inc. Gartner Says 50 Percent of Data Centers Will Have Insufficient Power and Cooling Capacity by 2008. Press Release, Nov. 2006.

[5] 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, Barcelona, Spain, Sept. 2001.

[6] J. S. Chase, D. C. Anderson, P. N. Thakar, A. Vahdat, and R. P. Doyle. Managing energy and server resources in hosting centers. In Symposium on Operating Systems Principles, Banff, Canada, Oct. 2001, pp. 103–116.

[7] R.L. Graham, L.E. Lawler, J.K. Lenstra, and A.H. Kan, Optimizing and Approximation in Deterministic Sequencing and Scheduling: A Survey, In Annals of Disc. Math, 1979, pp.287-326.

[8] C. Ellis. The case for higher-level power management. In Proceedings of the 7th Workshop on Hot Topics in Operating Systems, Rio Rico, Arizona, Mar. 1999.

[9] Feng Pan, Vincent W. Freeh, Daniel M. Smith: Exploring the Energy-Time Tradeoff in High-Performance Computing. In International Parallel and Distribution Processing Symposium, Denver, Colorado, Apr. 2005.

[10] J. G. Koomey. Estimating total power consumption by servers in the U.S. and the world. http://enterprise.amd.com/Downloads/svrpwrusecompletefinal.pdf, 2007.

[11] K. Flautner, S. Reinhardt, and T. Mudge. Automatic performance-setting for dynamic voltage scaling. In Proceedings of the 7th Conference on Mobile Computing and Networking, Rome, Italy, Jul. 2001.

[12] Compaq Computer Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd., and Toshiba Corporation, "Advanced Configuration and Power Interface Specification (ACPI) Revision 2.0b."http://www.acpi.info/DOWNLOADS/ACPIspec-2-0b.pdf, Oct. 2002.

[13] D. Grunwald, P. Levis, K. Farkas, C. Morrey, and M. Neufeld. Policies for dynamic clock scheduling. In Proceedings of 4th Symposium on Operating System Design and Implementation, San Diego, California, Oct. 2000.

[14] Sandy Irani, Sandeep K. Shukla, Rajesh K. Gupta: Online strategies for dynamic power management in systems with multiple power-saving states. ACM Transactions in Embedded Computing Systems, Feb. 2003, Vol.2, No. 3, pp.325-346

[15] Henri Casanova, Arnaud Legrand, Dmitrii Zagorodnov and Francine Berman, Heuristics for
scheduling parameter sweep applications in Grid environments. In Proc. of the 9th Heterogeneous
Computing Workshop, Cancun, Mexico, May 2000, pp.349-363.

[16] A. Vahdat, A. Lebeck, and C. Ellis. Every joule is precious: The case for revisiting operating system design for energy efficiency. SIGOPS European Workshop, Kolding, Denmark, Sept. 2000.

[17] D. P. Helmbold, D. D. E. Long, and B. Sherrod. A dynamic disk spin-down technique for mobile computing. In Proceedings of the 2nd Conference on Mobile Computing and Networking, Rye, New York, Nov. 1996, pp. 130–142.

[18] Henri Casanova, Graziano Obertelli, Francine Berman and Rich Wolski. The AppLeS parameter sweep template: User-level middleware for the Grid. In Proc. the Super Computing Conference, Dallas, Texas, Nov. 2000.

[19] R. Krashinsky and H. Balakrishnan. Minimizing energy for wireless web access with bounded slowdown. In Proceedings of the 8th Conference on Mobile Computing and Networking, Atlanta, Georgia, Sept. 2002.

[20] A. R. Lebeck, X. Fan, H. Zeng, and C. S. Ellis. Power aware page allocation. In Architectural Support for Programming Languages and Operating Systems, Cambridge, Massachusetts, Nov. 2000, pp. 105–116.

[21] Freund R F, Gherrity M, Ambrosius S, et al, Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet. In Proc. The 7th IEEE Heterogeneous Computing Workshop, Orlando, Florida, Mar. 1998, pp.184–199.

[22] Pinedo M. Scheduling: Theory, Algorithms, and Systems. Prentice Hall, Englewood Cliffs, New Jersey,1995.

[23] M.Harchol-Bulter and A.B.Downey. Exploiting process lifetime distributions for dynamic load balancing. Technical Report UCB/CSD-95-887, University of California, Berkely, Nov. 1995.

[24] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath. Dynamic cluster reconfiguration for power and performance. In Compilers and Operating Systems for Low Power, Sept. 2001.