This paper appeared in the *Proceedings of the 16th IEEE International Conference on Computer Communications and Networks* (ICCCN), Honolulu, Hawaii, Aug. 2007.

An Energy-Efficient Scheduling Algorithm Using Dynamic Voltage Scaling for Parallel Applications on Clusters

Xiaojun Ruan[†], Xiao Qin[†*], Ziliang Zong[†], Kiranmai Bellam[†], Mais Nijim[‡]
Department of Computer Science and Software Engineering
Auburn University, Auburn, AL 36849[†]
{xruan, xqin, zzong, kbellam}@ eng.auburn.edu
Department of Computer Science
University of Southern Mississippi, Hattiesburg, MS 39406[‡]

Abstract

In the past decade cluster computing platforms have been widely applied to support a variety of scientific and commercial applications, many of which are parallel in nature. However, scheduling parallel applications on large scale clusters is technically challenging due to significant communication latencies and high energy consumption. As such, shortening schedule length and conserving energy consumption are two major concerns in designing economical and environmentally friendly clusters. In this paper, we propose an energy-efficient scheduling algorithm (TDVAS) using the dynamic voltage scaling technique to provide significant energy savings for clusters. The TDVAS algorithm aims at judiciously leveraging processor idle times to lower processor voltages (i.e., the dynamic voltage scaling technique or DVS), thereby reducing energy consumption experienced by parallel applications running on clusters. Reducing processor voltages, however, can inevitably lead to increased execution times of parallel task. The salient feature of the TDVAS algorithm is to tackle this problem by exploiting tasks precedence constraints. Thus, TDVAS applies the DVS technique to parallel tasks followed by idle processor times to conserve energy consumption without increasing schedule lengths of parallel applications. Experimental results clearly show that the TDVAS algorithm is conducive to reducing energy dissipation in large-scale clusters without adversely affecting system performance.

1. Introduction

With advancement of computers and networks, large-scale clusters have been widely applied to support scientific and commercial applications, many of which are parallel applications. One effective approach to saving energy consumption in clusters is to make use of cutting-edge energy-efficient processors. This is because processors in computing nodes of clusters are main components that consumers a whole lot of energy. Among an array of energy conservation techniques for processors, the dynamic voltage scaling scheme or DVS is one of the most attractive ways to provide significant energy savings [13]. The basic idea behind the DVS technique is to dynamically reduce processors' supply voltages while guaranteeing proper operations. The energy consumption rate *P* can be expressed as below:

$$P = c \cdot v^2 \cdot f, \qquad (1)$$

where *c* is the capacity of the circuit, *v* is the supply voltage, and *f* is the frequency. Note that the capacity *c* is a constant parameter when processor is working. Energy consumption *W* of a processor is written as a product of energy consumption rate *P* and active time interval *t*. Thus, we have

$$E = P \cdot t. \qquad (2)$$

The goal of this study is to design an energy-efficient scheduling algorithm or TDVAS using dynamic voltage scaling for parallel applications with precedence constraints on large-scale clusters. We also quantitatively evaluate energy savings provided by the TDVAS algorithm.

[*] Corresponding author. http://www.eng.auburn.edu/~xqin

The rest of the paper is organized as follows. We present a brief description of related work in section 2. Section 3 describes a way to measure energy dissipation in a cluster running parallel applications. Section 4 presents the TDVAS scheduling algorithm using the DVS technique. In Section 5, we evaluate the performance of the proposed scheduling algorithm by comparing it with a baseline scheduling algorithm where the DVS technique is not employed. Finally, Section 6 concludes that the paper with future research directions.

2. Related Work

In the last decade, energy saving techniques have made it possible to develop energy-efficient cluster computing platforms. The issue of energy conservation for parallel application running on large-scale clusters has attracted little attention. Recently, researchers started to pay attention to energy conservation techniques for clusters. For example, *Kim et al.* investigated the dynamic voltage scaling scheme (DVS) and developed a novel algorithm called dynamic link shutdown (DLS), which makes use of an appropriate adaptive routing algorithm to shut down links in a judicious way [7]. Kim *et al.* proposed an optimized buffer design to reduce energy consumption in cluster interconnects [8]. Juan *et al*. designed a protocol called cluster-based Energy–Saving Routing Algorithm (CERA), which allows mobile computing nodes to autonomously create clusters to minimize energy dissipation in mobile nodes based on the clusters in wired or wireless network [9].

Very recently, we developed an array of duplication-based scheduling algorithms to reduce energy dissipation in cluster interconnects. Our scheduling approaches are especially beneficial for communication-intensive parallel applications [5].

The communication-to-computation ratio or CCR plays an important role in achieving high performance of parallel applications on clusters. CCR shows the relationship between communication cost and computing time in a parallel application. Without loss of generality, in this study we use the highest processor frequency to determine the computing times of parallel tasks. Once the computing times are measured, we can quantify the CCR values of parallel applications.

3. Energy-Efficient TADVS Scheduling

Now we present the energy-efficient TADVS scheduling algorithm. Due to precedence constraints, parallel tasks are unable to start their execution unless their corresponding parents tasks have been completed. As such, in each computing node of a cluster, there is a strong likelihood to exist some idle time intervals among tasks allocated to the computing node. In other words, after having finished executing a task, a computing node may not immediately start the execution of the next task due to precedence constraints and; therefore, the computing node will have to be sitting idle for a period of time. The basic idea behind the TADVS scheduling algorithm is to exploit idle processor time intervals in each computing node of a cluster, thereby making it possible to leverage idle time intervals to dynamically reduce the supply voltage of the computing node to substantially conserve energy consumption. More specifically, computing nodes are not supposed to be running with the highest frequency when the nodes are sitting idle. Given a set of idle processor time intervals, it is unnecessary for the computing nodes to accomplish all allocated parallel task as soon as possible using the highest frequency. Rather, processor frequencies can be judiciously lowered down, provided that the overall schedule lengths of parallel applications are not adversely affected.

Let us first introduce three important parameters to be used in TADVS. The first parameter for each parallel task is the earliest start time or EST for short. EST of an entry task is 0. The EST of all the other tasks can be calculated in a top-down manner by recursively applying the second term on the right side of Eq. (3).

$$EST(v_i) = \begin{cases} 0, & \text{if } \forall 1 \le j \le n: (v_j, v_i) \notin E \\ \min\limits_{(v_j,v_i)\in E}\left(\max\limits_{(v_k,v_i)\in E, v_k \ne v_j} \left(ECT(v_j), ECT(v_k) + w_{ki} \right)\right), & \text{otherwise} \end{cases}$$
(3)

where *ECT(v$_i$)* is the second important parameter, which quantifies the earliest completion time of task *v$_i$*. *ECT(v$_i$)* can be derived from Eq. (4) as the summation of its earliest start time and execution time.

$$ECT(v_i) = EST(v_i) + t_i. \qquad (4)$$

The third parameter *LACT(v$_i$)* is the latest allowable completion times of all the other tasks. *LACT(v$_i$)* can be calculated in a top-down manner by recursively applying the second term on the right side of Eq. (5).

$$LACT(v_i) = \begin{cases} ECT(v_i), & \text{if } \forall 1 \le j \le n: (v_i, v_j) \notin E \\ \min\left(\min\limits_{(v_i,v_j)\in E, v_i \ne FP(v_j)} \left(LAST(v_j) - w_{ij} \right), \min\limits_{(v_i,v_j)\in E, v_i = FP(v_j)} \left(LAST(v_j) \right)\right), & \text{otherwise} \end{cases}$$
(5)

We can make use of these three parameters to dynamically choose the most appropriate processor frequency and supply voltage for each computing node. Note that the execution time of a task lies in its complexity and the corresponding processor frequency.

The processor frequency determined on the fly can be derived from the aforementioned three parameters. Given a task v_i, its shortest execution time $t_i^{\min}$ is the time interval between $ECT(v_i)$ and $EST(v_i)$. Task v_i will experience the shortest execution time if its computing node is processing the task with the highest frequency. In contrast, the execution time of task v_i can be increased up to $t_i^{\max} = LACT(v_i) – EST(v_i)$ if the processor frequency is dynamically reduced with respect to task v_i. Now our concern is how to choose the most appropriate processor frequency that can lead to the maximum execution time $t_i^{\max}$ without negatively affecting system performance. Choosing the best frequency is challenging because frequencies of processors are not continuous tunable. Task v_i must be accomplished before $LACT(v_i)$ and; therefore, the frequency f_i be higher than or equal to f_{best}. As such, the most appropriate frequency can be determined by Eq. (6).

$$f_{best} = \frac{f_{Highest}(ECT - EST)}{LACT - EST}. \qquad (6)$$

Eq. (7) below measures energy savings gained by TADVS for a given parallel application.

$$PowerRate = \frac{\sum_{i=1}^{n} cv_i^2 f_i (LACT_i - EST_i) \frac{f_{best\_i}}{f_i}}{\sum_{i=1}^{n} c(v_{Highest})^2 f_{Highest}(ECT_i - EST_i)}$$
(7).

Given processor power P_i at different supply voltages, we can substitute P_i and $P_{Highest}$ for $cv_i^2 f_i$ and $c(v_{Highest})^2 f_{Highest}$ in Eq. (7) to further simply Eq. (7). Thus, we have

$$PowerRate = \frac{\sum_{i=1}^{n} P_i (LACT_i - EST_i) \frac{f_{best\_i}}{f_i}}{\sum_{i=1}^{n} P_{Highest}(ECT_i - EST_i)}. \qquad (8)$$

Given the maximum power, the highest frequency and volatage of processors in computing nodes, we can readily derive the constant c (i.e., the capacity) using Eq. (1). Provided supply voltages and processor frequencies, we can use the constant c and Eq. (1) compute energy consumption rate P_i.

Now we calculate processing time of the computing nodes using Eq. (9) as follows.

$$\mathrm{Proc}_{TADVS} = \sum_{i=1}^{n} (LACT - EST) \frac{f_{best\_i}}{f_i} \qquad (9).$$

It is insufficient to demonstrate energy savings in processors. As such, we also consider energy dissipation in network interconnects. Eq. (10) below quantifies the overall energy savings achieved by our TADVS compared with a non-TADVS-based scheme.

$$\frac{\sum_{i=1}^{n} P_i (LACT_i - EST_i) \frac{f_{best\_i}}{f_i} + com + CPU_{idle} * (time_{all} - \mathrm{Proc}_{Non-TADVS})}{\sum_{i=1}^{n} P_{Highest}(ECT_i - EST_i) + com + CPU_{idle} * (time_{all} - \mathrm{Proc}_{TADVS})}$$
(10)

4. Performance Evaluation

To evaluate the performance of TADVS, we conduct extensive experiments using various parallel applications. Further, we compare TADVS with an existing algorithm - NDS [5]. The system parameters are shown in Table 1.

Table1. Characteristics of system Parameter

CPU	Pentium 4, 1.4GH
Idle power	5w-22w
Frequency	1400Hz,1200Hz,1000Hz, 800Hz, 600Hz
Voltage	1.484V,1.463V,1.308V,1.180V , 0.956V
Execution time for tree	{3, 3, 4, 2, 1, 10, 20, 7, 5, 8}
Execution time for Guassian tree	{5, 4, 1, 1, 1, 1, 10, 2, 3, 3, 3, 7, 8, 6, 6, 20, 30, 30}
Network busy	33.6w
Network Idle	5w

The performance metrics by which we evaluate system performance include: (1) Processor energy: Energy consumption incurred by computing nodes. (2) Total Energy: Energy caused by a set of parallel tasks.

Figs. 1 and 2 show the DFA and the Gaussian applications, which used to evaluate the performance of TADVS algorithm.

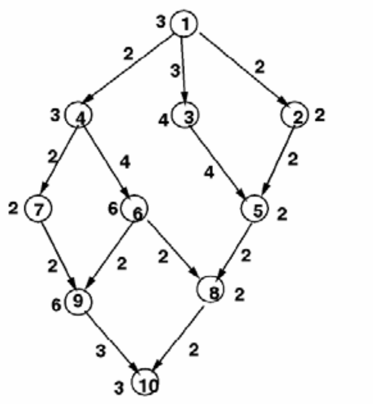

Fig. 1. The task graph of DFA

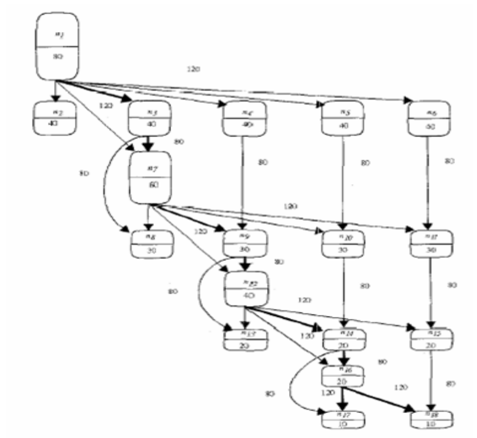

Fig. 2. The task graph of the Gaussian application.

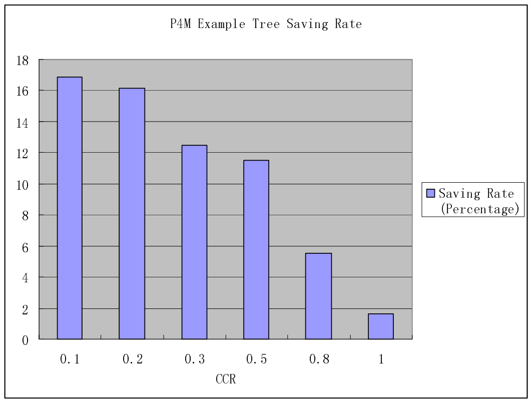

Fig. 3a. P4M, energy saving rates for DFA

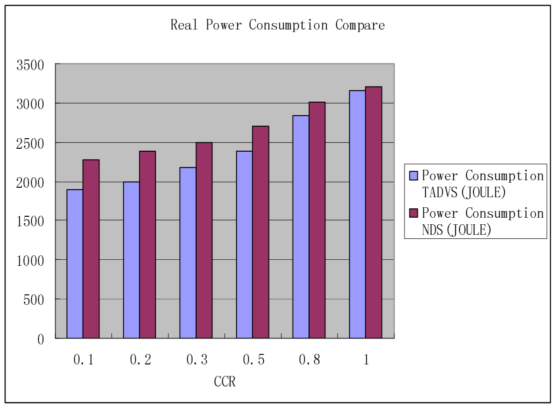

Fig. 3b. P4M, energy consumption for DFA

In this section, we varied CCR from 0.1 to 1 to examine the performance impacts of CCR on TADVS. Figs. 3a and 3b show that when CCR increases from 0.1 to 1, TADVS consumes less energy the least energy compared with NDS. As CCR increases, the power consumption gradually increases. This can be explained by the fact that a high CCR results in high communication cost, which in turn leads to an increased total energy consumption.

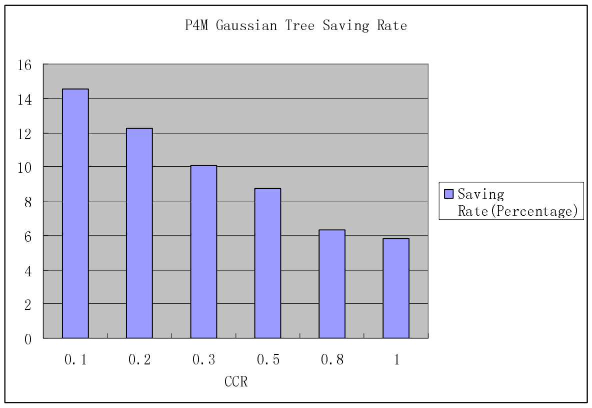

Fig. 3c. P4M, energy saving rates for Gaussian

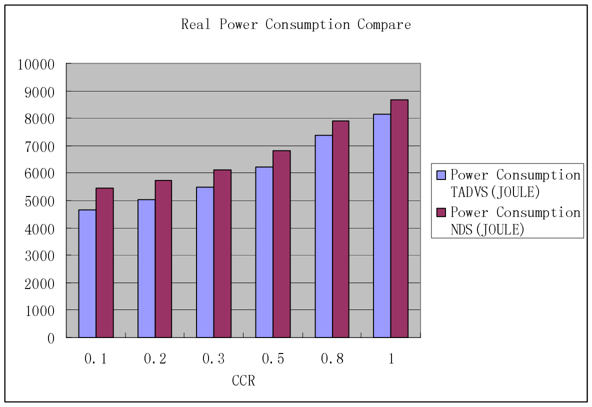

Fig. 3d. P4M, energy consumption for Gaussian

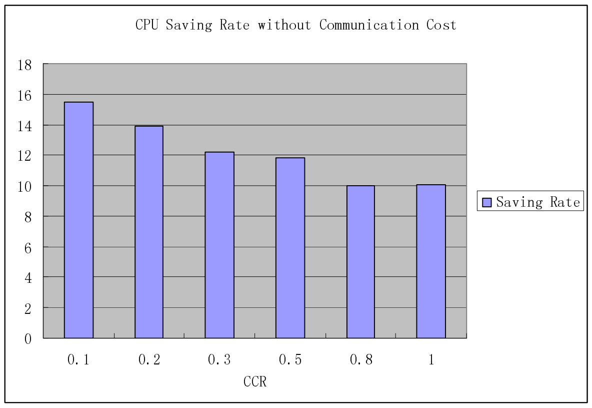

Fig. 3e. P4M, energy saving rates for Gaussian

Figs. 3c and 3d show performance of TADVS with respect to the Gaussian application (see Fig. 2). The experimental results reveal that TADVS can save energy consumption for the Gaussian application by up to 15%.

Figs 3f-3i plot experimental results for both the DFA and Gaussian applications running on a cluster, where XScale processors are used in each computing node. The goal of this set of experiment is to evaluate performance of TADVS using embedded processors. The empirical results show that TADVS is able to provide significant energy savings when it comes to embedded processors like Intel's XScale. Again, when the value of CCR increases, the energy saving rate decrease gradually. From Fig.3g and Fig.3i, the number of energy consuming is not so large, however, since the saving rate is significant, if TADVS is used widely, the power saving result will be great.

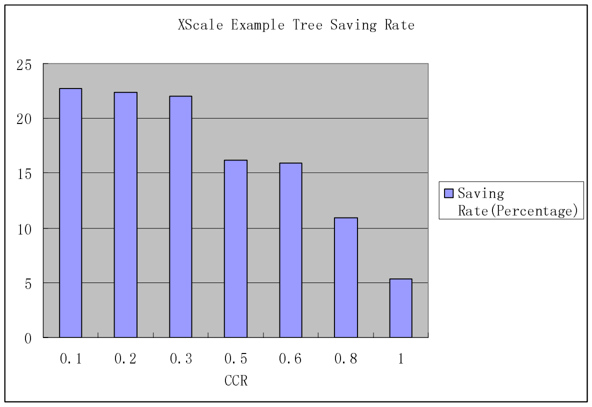

Fig. 3f. XScale, energy saving rates for DFA

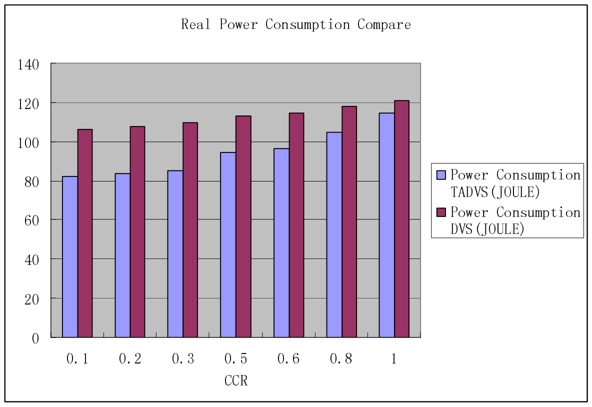

Fig. 3g. XScale, energy consumption for DFA

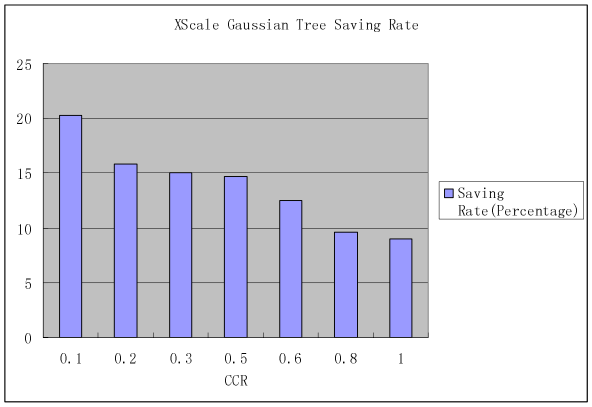

Fig. 3h. XScale, energy saving rates for Gaussian

In this set of experiments, we evaluate the processor frequency levels of each parallel task in the Gaussian application. For comparison purpose, Fig. 4 shows the results of NDS, where the highest frequency level is chosen to finish the tasks as soon as possible. We also show the optimal frequency levels in which energy saving rate can be maximized. Since processor frequencies are not continuous tunable, TADVS can not achieve the optimal energy savings. However, Fig. 4 demonstratively show that energy savings provided by TADVS is very close to the optimal solution. These results reveal that TADVS can result in sub-optimal energy savings for parallel applications running on both regular clusters and mobile clusters using embedded processors.

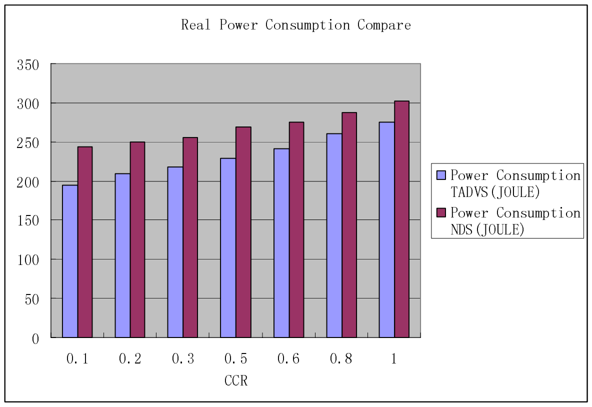

Fig. 3i. XScale, energy consumption for DFA

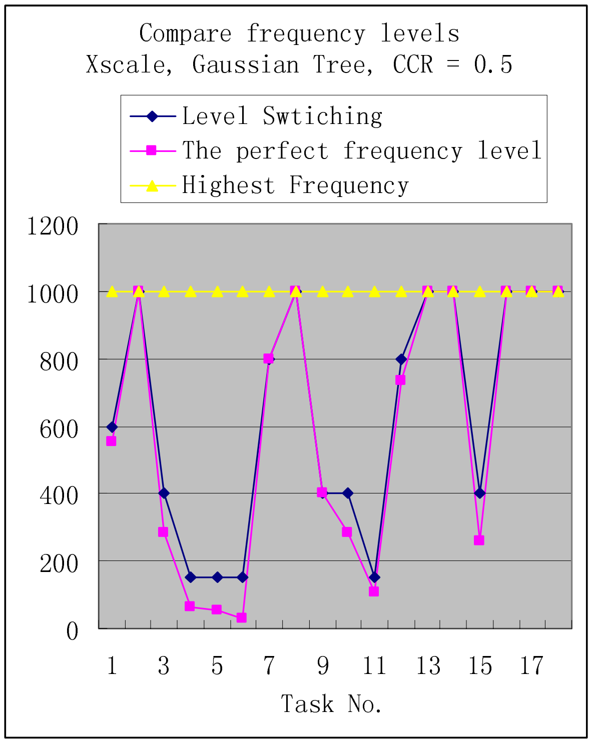

Fig. 4. Processor frequencies when CCR is set to 0.5

6. Conclusion

In the past decade clusters have been widely applied to support a variety of parallel applications. Scheduling parallel applications on clusters, especially on large-scale clusters, is challenging due to significant communication latencies and high energy consumption. Therefore, reducing schedule lengths and conserving energy consumption are two major concerns in the design of environmentally and economical friendly clusters. In this paper, we proposed an scheduling algorithm call TDVAS, which makes use of the dynamic voltage scaling technique (DVS) to provide significant energy savings for clusters. Our TDVAS scheme aims at judiciously leveraging processor idle times to lower processor voltages, thereby reducing energy consumption experienced by parallel applications running on clusters. However, decreasing supply voltages can inevitably lead to increased execution times of task in parallel applications. The salient feature of TDVAS is to tackle this problem by employing DVS to parallel tasks followed by idle processor times to conserve energy consumption without increasing schedule lengths of parallel applications. Experimental results demonstratively show that TDVAS is conducive to conserving energy consumption incurred by parallel application running on large-scale clusters without adversely affecting clusters' performance.

Acknowledgment

The work reported in this paper was supported by the US National Science Foundation under Grant No. CCF-0702781, Auburn University under a startup grant, New Mexico Institute of Mining and Technology under Grant No. 103295, the Intel Corporation under Grant No. 2005-04-070, and the Altera Corporation under an equipment grant.

References

[1] T.A. AlEnawy, H. Aydin, "On Energy-Constrained Real-Time Scheduling," *Proc. Euromicro Conf. Real-Time Systems*, pp. 165- 174, 2004.

[2] R. Ge, X. Feng, K.W. Cameron, "Performance-constrained Distributed Dvs Scheduling for Scientific applications on Power-aware Clusters," *Proc. ACM/IEEE Conf. Supercomputing*, 2005.

[3] Y.-K. Kwok and I. Ahmad, "Dynamic Critical-Path Scheduling: An Effective Technique for Allocating Task Graphs to Multiprocessors," *IEEE Trans. Parallel and Distributed Sys.*, vol. 7, no. 5, pp. 506-521, May 1996.

[4] Y.-H. Lee, Y. Doh, and C.M. Krishna, "EDF Scheduling Using Two-Mode Voltage-Clock-Scaling for Hard Real-Time System," *Proc. Int'l Conf. Compilers, Architecture, and Synthesis for Embedded Sys.*, pp. 221– 228, 2001

[5] Z. L. Zong, A. Manzanares, B. Stinar, and X. Qin, "Energy-Efficient Duplication Strategies for Scheduling Precedence Constrained Parallel Tasks on Clusters", *Proc. Int'l Conf. Cluster Computing*, 2006.

[6] Y. Huang, "Developing reliable applications on cluster systems," Proc. *Int'l Symposium on Reliable Distributed Systems*, 1996.

[7] Kim, E.J.; Link, G.M.; Yum, K.H.; Vijaykrishnan, N.; Kandemir, M.; Irwin, M.J.; Das, C.R., "A holistic approach to designing energy-efficient cluster interconnects," *IEEE Trans. Computers*, vol. 54, no. 6, 2005.

[8] Juan-Carlos Cano, Dongkyun Kim, Pietro Manzoni, "CERA: Cluster-Based Energy Saving Algorithm to Coordinate Routing in Short-Range Wireless Networks," *Proc. ICOIN*, pp. 306-315, 2003.

[9] S. Bansal, P. Kumar, and K. Singh, "An Improved Duplication Strategy for Scheduling Precedence Constrained Graph in Multiprocessor Systems," *IEEE Trans. Parallel and Distributed Sys.*, vol. 14, no. 6, pp. 533-544, 2003.

[10] S. Ranaweera, and D.P. Agrawal, "A Task Duplication Based Scheduling Algorithm for Heterogeneous Systems," *Proc. Int'l Symp. Parallel and Distributed Processing*, pp. 445-450, 2000.

[11] S.S. Pande, D.P. Agrawal and J. Mauney, "A Scalable Scheduling Method for Functional Parallelism on Distributed Memory Multiprocessors", *IEEE Trans. Parallel and Distributed Sys.*, vol. 6, no. 4, pp. 388-399, 1995.

[12] C.-H. Liu, C.-F. Li, K.-C. Lai, and C.-C. Wu; "A dynamic critical path duplication task scheduling algorithm for distributed heterogeneous computing systems," *Proc. Int'l Conf. Parallel and Distributed Systems*, pp. 365-374, 2006

[13] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A Dynamic Voltage Scaled Microprocessor System," *Proc. IEEE Int'l Solid-State Circuits Conf.*, pp. 294–295, 2000.

[14] AMD Inc., "AMD PowerNow Technology," 2000.

[15]Intel, Inc., "The Intel(R) XScale(TM) Microarchitecture Technical Summary," 2000.

[16] P. Pillai and K. G. Shin, "Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems," *Proc. ACM Symp. Operating Systems Principles*, 2001.

[17] H. Aydin, R. Melhem, D. Mosse, and P. M. Alvarez, "Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time Systems," *Proc. IEEE Real-Time Systems Symp.*, 2001.