EAD and PEBD: Two Energy-Aware Duplication Scheduling Algorithms for Parallel Tasks on Homogeneous Clusters

Ziliang Zong, Adam Manzanares, Xiaojun Ruan, and Xiao Qin, *Senior Member*, *IEEE*

Abstract—High-performance clusters have been widely deployed to solve challenging and rigorous scientific and engineering tasks. On one hand, high performance is certainly an important consideration in designing clusters to run parallel applications. On the other hand, the ever increasing energy cost requires us to effectively conserve energy in clusters. To achieve the goal of optimizing both performance and energy efficiency in clusters, in this paper, we propose two energy-efficient duplication-based scheduling algorithms—Energy-Aware Duplication (EAD) scheduling and Performance-Energy Balanced Duplication (PEBD) scheduling. Existing duplication-based scheduling algorithms replicate all possible tasks to shorten schedule length without reducing energy consumption caused by duplication. Our algorithms, in contrast, strive to balance schedule lengths and energy savings by judiciously replicating predecessors of a task if the duplication can aid in performance without degrading energy efficiency. To illustrate the effectiveness of EAD and PEBD, we compare them with a nonduplication algorithm, a traditional duplication-based algorithm, and the dynamic voltage scaling (DVS) algorithm. Extensive experimental results using both synthetic benchmarks and real-world applications demonstrate that our algorithms can effectively save energy with marginal performance degradation.

Index Terms—Homogeneous clusters, energy-aware scheduling, duplication algorithms.

1 Introduction

With the advent of powerful microprocessors and high-speed interconnects, and the increasing demand of computing capability, high-performance clusters have served as primary and cost-effective infrastructures for complicated scientific and commercial applications. Parallel applications running on clusters are generally computation-intensive and data-intensive in nature. Accordingly, efficient parallel execution and prompt completion of massive parallel tasks are essential and desirable.

Due to the high-power consumption of microprocessors, networks, and storage disks, high-performance clusters consume significant amounts of energy. For example, the total power of a 360-Tflops high-performance cluster would exceed 10 megawatts, possibly approaching 20 megawatts. Ten megawatts is approximately equal to the amount of power used in 11,000 US households [1]. The Environment Protection Agency reported that, in 2006, the total energy consumption of servers and data centers of the United States was 61.4 billion KWh, which was almost equal to the total power cost of 5.8 million US households [2].

- *Z. Zong is with the Department of Mathematics and Computer Science, South Dakota School of Mines and Technology, 501 E. St. Joseph Street, Rapid City, SD 57701-8647. E-mail: Ziliang.Zong@sdsmt.edu.*
- *A. Manzanares, X. Ruan, and X. Qin are with the Department of Computer Science and Software Engineering, Shelby Center for Engineering Technology, Samuel Ginn College of Engineering, System Lab 2104, Auburn University, AL 36849-5347. E-mail: {acm0008, xzr0001, xqin}@auburn.edu.*

Manuscript received 17 Mar. 2008; revised 19 Sept. 2008; accepted 16 Dec. 2009; published online 20 Oct. 2010.
Recommended for acceptance by X. Zhang.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2008-03-0119.
Digital Object Identifier no. 10.1109/TC.2010.216.

It is obvious that high performance and high energy cost are two key features of modern clusters. Ignoring either of them is unreasonable and unpractical. Unfortunately, previous research on clusters has been primarily focused on performance improvement. Nowadays, high energy cost has become a salient constraint of clusters. Designing energy-efficient and environmental friendly clusters is highly desirable. In this paper, we design novel scheduling algorithms to achieve the goal of maximizing performance and energy efficiency in clusters. More specifically, we propose two energy-aware duplication scheduling algorithms—the Energy-Aware Duplication (EAD) and Performance-Energy Balanced Duplication (PEBD) scheduling algorithms.

Task duplication strategies have been proved to be an efficient strategy to improve the performance of scheduling parallel tasks with precedence constraints [3], [4], [5]. This is mainly because unnecessary communication delay among multiple processors can be eliminated through task duplications, thereby reducing overall communication overheads in clusters. However, most existing duplication-based scheduling algorithms replicate all possible tasks to shorten schedule length without considering energy consumption caused by making replicas. In other words, the negative impact of task duplications was ignored in the previous studies. In contrast, our algorithms strive to make trade-off between schedule lengths and energy savings by judiciously replicating predecessors of a task if the replicas can improve performance without noticeably increasing energy.

To save energy, clusters can be built using low-frequency, low-power processors with modest performance. In doing so, performance might be more efficiently enhanced through parallelism than through using higher power, higher frequency processors. Green Destiny at

0018-9340/11/$26.00 © 2011 IEEE Published by the IEEE Computer Society

Los Alamos National Laboratory makes use of this approach to consume three times less energy per unit than the Accelerated Strategic Computing Initiative (ASCI) Q machine [6]. The other successful commercial cluster using this idea is the IBM Blue Gene/L supercomputer [7]. However, this approach may sacrifice too much per-node performance to achieve their low-power goals. For example, because Green Destiny uses more power-efficient microprocessors, it is approximately 15 times slower per node compared with high-performance nodes [6]. Using low-power and low-frequency chips succeeds only if users can improve performance by scaling up to a large number of processors. Unfortunately, many commercial clusters are not large scale in terms of the number of computing nodes.

Alternatively, one can build clusters using power-hungry and high-performance processors coupled with smart power management mechanisms. Clusters designed using this approach are usually called power-scalable clusters. In power-scalable clusters, the power level will be scaled down when clusters are not fully utilized and scaled up when clusters are busy. Dynamic Voltage and Frequency Scaling or DVFS is one of the most effective strategies to reduce energy consumption in power-scalable clusters. For example, Intel developed the *SpeedStep* technology [8] and AMD *developed the PowerNow!* and *Cool'n'Quiet* technology [9]. While DVFS technologies have made important contributions in building energy-efficient clusters, most of them are only capable of saving energy in processors.

Increasing evidences have shown that in addition to processors, high-speed interconnections consume significant amounts of energy in clusters. For example, it is observed that interconnect consumes 33 percent of the total energy in an Avici switch [10], whereas routers and links consume 37 percent of the total power budget in a Mellanox server blade [12]. This situation is getting worse with the emergence of next generation high-speed interconnections like Gigabit Ethernet, Infiniband, Myrinet, and $\mathrm{QsNet}^{\mathrm{II}}$. For instance, measurements have shown that a 1 Gbps Ethernet consume about 4 W more energy than 100 Mbps Ethernet [13]. A 10 Gbps Ethernet may consume 10 to 20 W more energy in average [13]. Lack of energy conservation technology for cluster interconnects becomes a severe problem because, without such technology, reducing energy consumption caused by communication-intensive parallel applications is almost impossible.

In this paper, we investigate the possibility of saving energy through power-aware duplication-based scheduling for both processors and interconnects. Our algorithms leverage DVFS to save energy dissipation in processors. Rather than adjusting the voltage to the best fit to current workloads, our algorithms force processors to operate at the highest voltage and frequency levels as long as there is a task waiting in the processing queue and the input datum of this task is ready. Our idea of conserving energy is to immediately turn processors to the lowest voltage once no task is waiting or no task is ready for execution. This policy ensures that tasks can be executed as fast as possible. Meanwhile, tasks on the critical path will be duplicated under the condition that no significant energy overhead is introduced by the replicas. Duplications can avoid the performance degradation caused by waiting messages. The rationale behind our approach is twofold. First, energy overhead incurred by task replicas could be offset by energy savings in interconnects and by shortening schedule length. Second, the overall performance will be improved by the virtue of replicas.

The rest of the paper is organized as follows: In Section 2, we present related work. Next, Section 3 introduces mathematical models including a system model, a task model, and an energy consumption model. In Section 4, we present the energy-aware scheduling strategies. Experimental environment and simulation results are demonstrated in Section 5. Finally, Section 6 provides the concluding remarks and future research directions.

2 Related Work

Since many parallel applications running in clusters require intensive data processing and data communication, scheduling strategies deployed in clusters have a large impact on overall system performance. Basically, parallel scheduling strategies can be classified to three primary categories, called priority-based scheduling, cluster-based scheduling, and duplication-based scheduling, respectively. Priority-based scheduling involves the assignment of priorities to tasks and then maps those tasks to processors based upon assigned priorities [14]. Cluster-based scheduling algorithms cluster as many intercommunicating tasks as possible in a group and allocate it to the same processor, thereby eliminating communication overheads [15]. The basic idea of duplication-based scheduling is to replicate as many as predecessor tasks in the critical paths provided that the schedule length can be shortened. Duplication scheduling outperforms other scheduling algorithms in most cases, especially when communication time dominates the execution time of parallel applications. However, the performance improvement increases energy consumption because many tasks are duplicated and thus executed more than once by multiple processors. To address this problem, we propose two energy-aware duplication algorithms (EAD and PEBD) in this paper. Instead of duplicating all performance-oriented tasks, our algorithms replicate tasks with the consideration of both performance improvement and energy cost.

There has been a large body of previous studies investigating power-aware techniques to reduce energy consumption in processor and memory resources [16], [17], [18], [19] in the late 90's. Dynamic power management is a design methodology aiming to achieve specified performance with minimum number of active components or a minimum load on such components [20]. Dynamic power management consists of a collection of energy-efficient techniques that adaptively turn off cluster components or bring performance down when the components are idle or partially unexploited. For example, based on the observation of past idle and busy periods, predictive shutdown policies can make power management decisions before a new idle period starts [21].

Researchers have focused on energy-aware algorithms for power-scalable clusters. Among these algorithms, Dynamic Voltage Scaling (DVS) technology [22], [23], [24], [25], [26], [27] has been widely exploited to make processors energy-efficient in both portable and nonportable computing systems. Dynamic Frequency Scaling (also known as CPU throttling) is another similar technique in which a processor runs at a less-than-maximum frequency when it is not fully utilized in order to conserve power [23]. Very recently, many

studies have been reported in utilizing the Dynamic Voltage and Frequency Scaling (DVFS) technology to reduce power dissipations in clusters and high-performance computing platforms [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34]. Results have shown that these proposed schemes can achieve high energy efficiency for processors, indicating that DVFS is capable of saving significant amount of energy for computation-intensive applications. The benefits of DVFS may diminish when it comes to communication-intensive applications, because the energy consumed by interconnects dominates the total power consumption.

To address the above problem, researchers in Princeton University investigated the possibility of introducing DVS technology to interconnections [35]. Shang et al. proposed the architectural model for applying DVS to links. Soteriou and Peh investigated the potential of carrying DVS links to the extreme—dynamically turning links on/off in response to communication traffic variance [37]. Gunaratne et al. investigated Adaptive Link Rate (ALR) as a means of reducing the energy consumption of a typical Ethernet link by adaptively varying the link data rate in accordance with utilization [38]. Their simulation results demonstrated that applying DVS to interconnections can achieve noticeable energy savings. Their approaches heavily rely on hardware support, e.g., network equipments (Network Interface Cards and Switches) with various link rates. Unfortunately, interconnects with multiple link rate and link frequency are rarely used, especially for high-speed interconnections like Gigabit Ethernet, Infiniband, Myrinet, and $\mathrm{QsNet}^{\mathrm{II}}$. Furthermore, recent studies have shown that power consumption in clusters is independent of link utilization. For example, Gunaratne et al. have found that idle and fully utilized links consume about the same amount of power in Ethernet [38]. Zamani et al. also stated that the Myrinet-2000 equipment does not have power management technique and the network energy consumed by interconnects remains unchanged regardless of the network traffic [39].

One of the feasible approaches to conserving power consumption caused by interconnects is to make task duplication. Compared with existing energy-efficient techniques, duplication-based strategies have their unique advantages. First, task replicas can avoid communication overheads among tasks, thereby improving performance (see, for example, [3], [4], [5]). Second, as for communication-intensive applications, huge energy consumption in interconnects can be reduced. We will address this point in the following sections in detail. Third, the duplication strategies can be seamlessly integrated with the DVS technology to reduce energy dissipations in processors. Last but not the least, the duplication-based schemes can be used in combination with the Adaptive Link Rate technology if the required network devices are available on the market in the future.

3 Mathematical Models

In this section, we describe mathematical models used to represent clusters, precedence-constrained parallel tasks, and energy consumption in processors and interconnects.

3.1 Cluster Model

A cluster in this study is characterized by a set $\mathrm{P} = \{p_1, \mathrm{p}_2, \ldots, \mathrm{p}_\mathrm{m}\}$ of computational nodes (hereinafter referred to as nodes) connected by high-speed interconnects. It is assumed that the computational nodes are homogeneous in nature, meaning that all processors are identical in their capabilities. Similarly, the underlying interconnection is assumed to be homogeneous and, thus, communication overhead of a message with fixed data size between any pair of nodes is considered to be the same. Each node communicates with other nodes through message passing, and the communication time between two precedence-constrained tasks assigned to the same node is negligible. To simplify the cluster model without loss of generality, we assume that the cluster system is fault-free and the page fault service time of each task is integrated into its execution time. With respect to energy conservation, energy consumption rate of each node in the system is measured by Joule per unit time. Each interconnection link is characterized by its energy consumption rate that heavily relies on data size and the transmission rate of the link.

3.2 Task Model

Parallel applications with a set of precedence-constrained tasks can be represented in form of a Directed Acyclic Graph (DAG) [40]. In this paper, a parallel application running in clusters is modeled as a vector (V, E), where $\mathrm{V} = \{v_1, \mathrm{v}_2, \ldots, \mathrm{v}_\mathrm{n}\}$ represents a set of precedence-constrained parallel tasks, and E denotes a set of messages representing communications and precedence constraints among parallel tasks. It is assumed that all tasks in V are nonpreemptive and indivisible work units. For each task in V, t_i is defined as the required time to compute $v_i, 1 \leq i \leq n$. Similarly, $e_{ij} = (v_i, v_j) \in E$ is defined as a message transmitted from task v_i to v_j, and c_{ij} is the required time of passing the message $e_{ij} \in E$. Please note that e_{ij} is set to zero if v_i and v_j are assigned to the same computational node. We assume in this study that there is only one entry task and one exit task for parallel applications with a set of precedence-constrained tasks. The assumption is reasonable because in case of multiple entry or exit tasks exist, the multiple tasks can always be connected through a dummy task with zero computation cost and zero communication cost messages. A task allocation matrix (e.g., X) is an $n \times m$ binary matrix reflecting a mapping of n precedence-constrained parallel tasks to m computational nodes in a cluster. Element x_{ij} in X is "1" if task v_i is assigned to node p_j and is "0", otherwise.

3.3 Energy Consumption Model

We use a divide-and-conquer approach to derive the energy consumption models for processors and interconnections.

Let en_i be the energy consumption caused by task v_i running on a computational node, of which the energy consumption rate is PN_{high}. The energy dissipation of task v_i can be expressed as (1).

$$en_i = PN_{high} \times t_i. \tag{1}$$

Given a parallel application with a task set V and allocation matrix X, we can calculate the energy consumed by executing all the tasks in V using (2).

$$\begin{aligned} EN_{high} &= \sum_{i=1}^{|V|} en_i = \sum_{i=1}^{n} \left(PN_{high} \cdot t_i\right) \\ &= PN_{high} \sum_{i=1}^{n} t_i. \end{aligned} \tag{2}$$

Let PN_{low} be the power of a computational node when it is not executing a task, and f_i be the completion time of task t_i. The energy consumed by an inactive node is a product of the low-energy consumption rate PN_{low} and an idle period. Thus, we can use (3) to obtain the energy consumed by the jth computational node in a cluster when the node is sitting idle.

$$EN_{low}^{j} = PN_{low} \cdot \left(\max_{i=1}^{n}(f_i) - \sum_{i=1}^{n} \left(x_{ij} \cdot t_i\right) \right), \tag{3}$$

where $\max_{i=1}^{n}(f_i)$ is the schedule length, and $\max_{i=1}^{n}(f_i) - \sum_{i=1}^{n} x_{ij} \cdot t_i$ is the total idle time on the jth node. The total energy consumption of all the idle nodes is

$$\begin{aligned} EN_{low} &= \sum_{j=1}^{m} en_{low}^{j} = PN_{low} \cdot \sum_{j=1}^{m} \left(\max_{i=1}^{n}(f_i) - \sum_{i=1}^{n}(x_{ij} \cdot t_i) \right) \\ &= PN_{low} \cdot \left(m \cdot \max_{i=1}^{n}(f_i) - \sum_{j=1}^{m} \sum_{i=1}^{n}(x_{ij} \cdot t_i) \right). \end{aligned} \tag{4}$$

Consequently, the total energy consumption of the parallel application running on the cluster can be derived from (2) and (4) as

$$\begin{aligned} EN = EN_{high} + EN_{low} &= PN_{high} \sum_{i=1}^{n} t_i \\ &+ PN_{low} \cdot \left(m \cdot \max_{i=1}^{n}(f_i) - \sum_{j=1}^{m} \sum_{i=1}^{n} \left(x_{ij} \cdot t_i\right) \right). \end{aligned} \tag{5}$$

Please note that this energy consumption model is compatible with the DVFS technology. In DVFS, processors may have several voltage and frequency levels; scheduling algorithms may choose the best fit voltage to conserve energy. In that case, PN_{high} can be replaced with $PN_{best\text{-}fit}$ in the model.

To calculate the energy consumptions of interconnects, we denote el_{ij} as the energy consumed by the transmission of message $(t_i, t_j) \in E$. We can compute the energy consumption of the message as a product of its communication time and the power PL_{high} of the link when it is active:

$$el_{ij} = PL_{high} \times c_{ij}. \tag{6}$$

The cluster interconnect, in this study, is supposed to be homogeneous, which implies that all messages are transmitted over the network interconnects at the same transmission rate. The energy consumed by a network link between p_a and p_b is a cumulative energy consumption caused by all messages transmitted over the link. Therefore, the link's energy consumption is obtained by (7), where L_{ab} is a set of messages delivered on the link.

$$\begin{aligned} L_{ab} &= \left\{ \forall e_{ij} \in E, 1 \le a, b \le m | x_{ia} = 1 \wedge x_{jb} = 1 \right\}. \\ EL_{high}^{ab} &= \sum_{e_{ij} \in L_{ab}} el_{ij} = \sum_{e_{ij} \in L_{ab}} \left(PL_{high} \cdot c_{ij} \right) \\ &= \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} \left(x_{ia} \cdot x_{jb} \cdot PL_{high} \cdot c_{ij} \right). \end{aligned} \tag{7}$$

The energy consumption of the whole interconnection network is derived from (8) as the summation of all the links' energy consumption. Thus, we have

$$\begin{aligned} EL_{high} &= \sum_{a=1}^{m} \sum_{b=1, b \neq a}^{m} EL_{high}^{ab} \\ &= \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} \sum_{a=1}^{m} \sum_{b=1, b \neq a}^{m} \left(x_{ia} \cdot x_{jb} \cdot PL_{high} \cdot c_{ij} \right). \end{aligned} \tag{8}$$

Similarly, we can express energy consumed by a link when it is working in the low-power mode (e.g., idle mode) as a product of the low-power consumption rate and the period of the link working in this mode. Thus, we have

$$EL_{low}^{ab} = PL_{low} \cdot \left(\max_{i}^{n}(f_i) - \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} \left(x_{ia} \cdot x_{jb} \cdot c_{ij} \right) \right), \tag{9}$$

where PL_{low} is the power consumption rate of the link when it is in the low-power mode, and $\max_{i}^{n}(f_i) - \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} \left(x_{ia} \cdot x_{jb} \cdot c_{ij} \right)$ is the total time of the link stays in this mode.

Now, we can express energy incurred by the whole interconnection network during the low-power periods as

$$\begin{aligned} EL_{low} &= \sum_{a=1}^{m} \sum_{b=1, b \neq a}^{m} EL_{low}^{ab} \\ &= \sum_{a=1}^{m} \sum_{b=1, b \neq a}^{m} PL_{low} \left(\max_{i}^{n}(f_i) - \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} \left(x_{ia} \cdot x_{jb} \cdot c_{ij} \right) \right). \end{aligned} \tag{10}$$

Therefore, total energy consumption exhibited by the cluster interconnect is derived from (8) and (10) as

$$EL = EL_{high} + EL_{low}. \tag{11}$$

Note that in our experiments, PL_{high} and PL_{low} of some types of interconnects may be identical, i.e., $PL_{high} = PL_{low}$. That is because latest studies have shown that idle and fully utilized high-speed interconnects consume almost the same amount of energy in clusters [38], [39]. For example, there is no power management mechanism in Myrinet-2000; the network energy consumed by Myrinet-2000 switches remains unchanged regardless of network traffics [39]. We consider PL_{high} and PL_{low} in our model to make the model compatible with future interconnects coupled with the adaptive link rate and dynamic power management techniques.

Finally, the total energy consumption of the cluster executing the application can be derived from (5) and (11) as

$$E = EN + EL. \tag{12}$$

4 Energy-Aware Duplication Strategies

In this section, we present two energy-aware duplication strategies, called EAD and PEBD, for scheduling parallel applications with precedence constraints. The objective of the two scheduling strategies is to shorten schedule lengths while optimizing energy consumption of clusters. The scheduling problem studied in this paper has been proved to be NP-hard [41]. Therefore, the proposed two scheduling algorithms are

TABLE 1
Important Notations and Parameters

Notation	Definition
$EST(v_i)$	Earliest start time of task v_i
$ECT(v_i)$	Earliest completion time of task v_i
$FP(v_i)$	Favorite predecessor of task v_i
$LACT(v_i)$	Latest allowable completion time of task v_i
$LAST(v_i)$	Latest allowable start time of task v_i

heuristic in the sense that they only can produce suboptimal solutions. The EAD and PEBD algorithms consist of three major steps delineated in Sections 4.1-4.3.

4.1 Generate Original Task Scheduling Sequence

Precedence constraints of a set of parallel tasks have to be guaranteed by executing predecessor tasks before successor tasks. To achieve this goal, the first step in our algorithms is to generate an ordered task sequence using the concept of level. The level of each task is defined as the computation time from current task to the exit task. There are alternative ways to generate the task sequence for a DAG, we use a similar approach proposed in [4] to define the level $L(v_i)$ of task v_i as below:

$$L(v_i) = \begin{cases} t_i, & \text{if successor(i)} = \Phi, \\ \max\limits_{k \in succ(i)}(level(k)) + t_i, & otherwise. \end{cases} \quad (13)$$

The levels of other tasks can be calculated in a bottom-up fashion by recursively applying the second term on the right-hand side of (13). Once we obtain the levels, tasks will be sorted in ascending order of the levels and the sorted tasks form the original task-scheduling sequence.

4.2 Duplication Parameters Calculation

The second phase in the EAD and PEBD algorithms is to calculate important parameters, which the algorithms rely on to make duplication decision. The important notation and parameters are listed in Table 1. Note that similar notation was used by Ranaweera and Agrawal in [4].

The earliest start time of the entry task is 0 (see the first term on the right side of (14)). The earliest start times of all the other tasks can be calculated in a top-down manner by recursively applying the second term on the right side of (14).

$$EST(v_i) = \begin{cases} 0, & \text{if predecessor(i)} = \Phi, \\ \min\limits_{e_{ji} \in E}\left(\max\limits_{e_{ki} \in E, v_k \neq v_j}(ECT(v_j), ECT(v_k) + c_{ki})\right), & \text{otherwise.} \end{cases} \quad (14)$$

The earliest completion time of task v_i is expressed as the summation of its earliest start time and execution time. Thus, we have

$$ECT(v_i) = EST(v_i) + t_i. \quad (15)$$

Allocating task v_i and its favorite predecessor $FP(v_i)$ on the same computational node can lead to a shorter schedule length. As such, the favorite predecessor $FP(v_i)$ is defined as below:

$$FP(v_i) = v_j, \text{where } \forall e_{ji} \in E, e_{ki} \in E, j \neq k \big| ECT(v_j) + c_{ji} \geq ECT(v_k) + c_{ki}. \quad (16)$$

As shown by the first term on the right-hand side of (17), the latest allowable completion time of the exit task equals to its earliest completion time. The latest allowable completion times of all the other tasks are calculated in a top-down manner by recursively applying the second term on the right-hand side of (17).

$$LACT(v_i) = \begin{cases} ECT(v_i), & \text{if successor(i)} = \Phi, \\ \min\left(\min_{e_{ij} \in E, v_i \neq FP(v_j)}(LAST(v_j) - c_{ij}), \right. & \\ \left. \min\limits_{e_{ij} \in E, v_i = FP(v_j)}(LAST(v_j))\right), & \text{otherwise.} \end{cases} \quad (17)$$

The latest allowable start time of task v_i is derived from its latest allowable completion time and execution time. Hence, the $LAST(v_i)$ can be written as

$$LAST(v_i) = LACT(v_i) - t_i. \quad (18)$$

4.3 Energy-Aware Task Duplication and Allocation

4.3.1 The EAD Algorithm

Given a parallel application presented in form of a DAG, the EAD algorithm allocates each parallel task to a computational node in a way to aggressively shorten the schedule length of the DAG while conserving energy consumption. Fig. 1 shows the pseudocode of the EAD algorithm, which aims to provide the greatest energy savings when it reaches the point to duplicate a task. Most existing duplication-based scheduling schemes merely optimize schedule lengths without addressing the issue of energy conservation. As such, the existing duplication-based approaches tend to yield minimized schedule lengths at the cost of high energy consumption. To make trade-offs between energy savings and schedule lengths, we design the EAD algorithm in which task duplications are strictly forbidden if the duplications do not exhibit energy conservation (see steps 9-10). In other words, duplications are not allowed if they result in a significant increase in energy consumption (e.g., the increase exceeds a threshold). Consequently, the EAD algorithm ensures that performance is optimized through task duplication with little energy consumption.

Before this phase starts, phase 1 sorts all the tasks in a waiting queue, followed by phase 2 to calculate the important parameters. In phase 3, EAD strives to group communication-intensive parallel tasks together and have them allocated to the same computational node. Once multiple task groups are constructed, each group of tasks is assigned to a different node in the cluster. The process of grouping tasks is repeated from the first task in the queue by performing a depth-first style search, which traces the path from the first task to the entry task. Steps 5 and 6 choose a favorite predecessor if it has not been allocated a computational node. Otherwise, EAD may or may not replicate the favorite predecessor on the current node. For example, we assume that v_j is the favorite predecessor of the current task v_i, and v_j has been allocated to another node. If duplicating v_j on the current node to which v_i is

Phase 3 of EAD Algorithm

1. v = first waiting task of scheduling queue;
2. $i = 0$;
3. assign v to P_i;
4. **while** (not all tasks are allocated to computational nodes) **do**
5. $u = FP(v)$;
6. **if** (u has already been assigned to another processor) **then**
7. **if** ($LAST(v) - LACT(u) < c_{uv}$) **then** /* if duplicate u, we can shorten the schedule length */
8. moreenergy = $en_u - el_{uv}$; /*energy increase*/
9. **if** (moreenergy ≤ threshold h) **then** /* increased energy less than our threshold*/
10. assign u to P_i; /*duplicate u*/
11. **if** v has another predecessor $z \neq u$ has not yet been allocated to any node **then**
12. $u = z$;
13. **else**
14. **if** u is entry task **then**
15. u = the next task that has not yet been assigned to a node;
16. i++;
17. **else**
18. for another predecessor z of v, $z \neq u$,
19. **if** ($ECT(u) + cc_{uv} = ECT(z) + cc_{zv}$) and z hasn't been allocated) **then**
20. $u = z$; /* do not duplicate*/
21. **else**
22. for another predecessor z of v, z≠ u,
23. **if** ($ECT(u) + cc_{uv} = ECT(z) + cc_{zv}$) and z hasn't been allocated) **then**
24. $u = z$; /* do not duplicate*/
25. **else** allocate u to P_i;
26. v = u;
27. **if** v is entry task **then**
28. v = the next task that has not yet been allocated to a computational node;
29. i++;
30. assign v to P_i;
31. **return** schedule list, schedule length and energy consumption;

Phase 3 of PEBD Algorithm

1. v = first waiting task of scheduling queue;
2. $i = 0$;
3. assign v to P_i;
4. **while** (not all tasks are allocated to computational nodes) **do**
5. $u = FP(v)$;
6. **if** (u has already been assigned to another node) **then**
7. **if** ($LAST(v) - LACT(u) < c_{uv}$) **then** /* if duplicate u, we can shorten the execution time*/
8. moreenergy = $en_u - el_{uv}$; /*energy increase*/
9. lesstime = $LACT(u) + c_{uv} - LAST(v)$; /* schedule length is reduced */
10. cost ratio = moreenergy / lesstime; /*value of ratio: the smaller the better*/
11. **if** (ratio ≤ threshold h) **then** /* significantly shorten schedule length */
12. assign u to P_i; /*duplicate u*/
13. **if** v has another predecessor $v \neq u$ has not yet been assigned to any node **then**
14. $u = v$;
15. **else**
16. **if** u is entry task **then**
17. u = the next task that has not yet been allocated to a computational node;
18. i++;
19. **else**
20. for another predecessor z of v, $z \neq u$,
21. **if** ($ECT(u) + cc_{uv} = ECT(z) + cc_{zv}$) and z has not been allocated) **then**
22. $u = z$; /*do not duplicate*/
23. **else**
24. for another predecessor z of v, $z \neq u$,
25. **if** ($ECT(u) + cc_{uv} = ECT(z) + cc_{zv}$) and z has not been allocated) **then**
26. $u = z$; /*do not duplicate*/
27. **else** assign u to P_i;
28. $v = u$;
29. **if** v is entry task **then**
30. v = the next task that has not yet been allocated;
31. i++;
32. allocate v to P_i;
33. **return** schedule list, schedule length and energy consumption

Fig. 1. Pseudocode of phase 3 in the EAD and PEBD algorithms.

allocated can improve performance without sacrificing energy conservation, Step 12 makes a duplication of v_j.

The generation of a task group terminates once the path reaches the entry task. The next task group starts from the first unassigned task in the queue. If all the tasks are assigned to the computation nodes, then the algorithm terminates.

4.3.2 The PEBD Algorithm

The third phase of the PEBD algorithm is similar as that of EAD except that PEBD seamlessly integrate the approach to minimizing schedule lengths with the process of energy optimization (see Fig. 1). Unlike EAD, the development of PEBD is motivated by the needs of making the right trade-off between performance and energy conservation. Thus, the PEBD algorithm is geared to efficiently reduce schedule lengths while providing the greatest energy savings. Energy consumption incurred by duplicating a task involves judging whether the duplication is profitable or not. To facilitate the construction of PEBD, we introduce a concept of cost ratio of a duplication, which is defined as the ratio between the energy saving and schedule length reduction (see Step 10). While the energy saving of the duplication is obtained in Step 8, the reduction in schedule length is computed in Step 9. The PEBD algorithm is, of course, conducive to maintaining cost ratios at a low level, thereby efficiently shortening schedule lengths with low-energy consumption. This feature is accomplished by Steps 11-12, which duplicate a task in case the cost ratio of such duplication is smaller than a given threshold.

4.4 Time Complexity Analysis

In this section, we will analyze the time complexity of the EAD and PEBD algorithms.

Theorem 1. *Given a parallel application with multiple precedence-constrained tasks, the time complexity of EAD and PEBD to make scheduling decisions is* $O(2|E| + |V|(\lg|V| + 1) + h|V|)$, *where E is the number of messages, V is the number of parallel tasks, and h is the height of the DAG.*

Proof. The EAD and PEBD algorithms perform the three main phases, respectively, described in Sections 4.1-4.3. In the first phase, EAD and PEBD traverse all the tasks of the DAG to compute the levels of the tasks. The time complexity to calculate the levels is $O(|E|)$, where $|E|$ is

the number of messages. This is because all the messages have to be examined in the worst case. It takes $O(|V|\log|V|)$ time to sort the tasks in the nonincreasing order of the levels, where $|V|$ is the number of tasks. Therefore, the time complexity of phase 1 is $O(|E| + |V|\log|V|)$. □

The second phase is performed to obtain all the important parameters like *EST*, *ECT*, *FP*, *LACT*, and *LAST*. Phase 2 calculates these parameters by applying the depth-first search with the complexity of $O(|V| + |E|)$.

Recall that, in phase 3, the tasks are allocated to the computational nodes. First, all the tasks are checked and allocated to one or more nodes in the while loop based on duplication strategies. In the worst case, all the tasks in the critical path must be duplicated, meaning that the time complexity is $O(h|V|)$ time, where h is the height of the DAG.

Consequently, the overall time complexities of EAD and PEBD are $O(2|E| + |V|(\lg|V| + 1) + h|V|)$. Since parallel applications tend to have high parallelism, the time complexity of EAD and PEBD is approximate to $O(V\lg|V|)$.

5 Energy-Performance Evaluation

This section presents the comprehensive simulation results in terms of power-performance efficiency by comparing the proposed EAD and PEBD algorithms with the three existing approaches, namely the Modified Critical Path Scheduling (MCP) algorithm [42], the Task Duplication Scheduling (TDS) algorithm [4], and the DVS algorithm [27], [43]. We chose both synthetic DAGs and real-world applications to evaluate performance and energy efficiency of the five algorithms. MCP and TDS are two well-known performance-oriented algorithms; DVS is one of the most effective approaches to reducing energy consumption.

In this section, we first briefly introduce the three baseline algorithms. In Section 5.2, we discuss the hardware configurations used in our simulator. Next, we justify system parameters and explain the simulator in Section 5.3. Finally, in Sections 5.4-5.7, we investigate the impacts of processors, interconnects, applications and Communication-Computation-Ratio (CCR) on performance and energy efficiency of the algorithms.

5.1 Existing and Baseline Algorithms

Now we briefly describe the three baseline algorithms—MCP, TDS, and DVS. Note that the goal of MCP and TDS is to improve performance, whereas DVS aims at saving energy.

- **MCP** [42]. MCP was proposed to optimize the scheduling of parallel processes in a complicated multiprocessing environment. In this programming environment, all parallel processes have to exchange data with each other through message passing, which is very similar to communication mechanisms for parallel tasks running in clusters. The key step of MCP is to identify tasks with the most profound impacts on performance improvement and apply the as-soon-as-possible binding strategy to them. These tasks are marked as critical path tasks, which are given higher priority for execution.
- **TDS** [4]. TDS is another critical-path-based scheduling algorithm, which attempts to generate the shortest schedule length. The fundamental difference between TDS and MCP is that TDS duplicates tasks in critical path if the duplication can further improve performance. In MCP, the tasks in critical path have higher priority to obtain system resources for as-soon-as-possible execution, but all tasks are executed only once. Similarly, TDS allocates all tasks that are in a critical path to the same processor. However, if tasks have already been dispatched to other processors, TDS will duplicate the tasks to potentially shorten scheduling lengths. In other words, tasks in TDS may be executed more than once if task replicas can aid in performance improvement.
- **DVS** [27], [43]. DVS is an energy-aware scheduling algorithm for power-scalable clusters, where voltage of processors can be dynamically adjusted. DVS conserves energy by scaling down the processor voltages when processor is underutilized or idle. To avoid potential performance degradation, DVS has to exploit unbalanced workloads among processors so that the processor voltages can be scaled to the best-fit status according to workload conditions. Thus, increasing overall execution time can be prevented as tasks are executed "just in time." There are two conditions under which DVS achieves good energy-performance efficiency. First, processors must have slack times, which is time spent in waiting messages from other tasks. Second, the workload of processors is unbalanced. Therefore, the integration of DVS and MCP works better than the combination of DVS and TDS, because duplications not only eliminate most of the slack times but also result in balanced workloads.

5.2 Hardware Configuration Profiles

To quantitatively evaluate the energy efficiency and performance of our algorithms, we have experimented with five different types of processors and four different types of network interconnects. We outline the detailed hardware configuration profiles and summarize the characteristics of each type of processor and interconnect.

5.2.1 Processor Configuration Profiles

Five types of processors used in our studies include AMD Athlon 64 X2 4600+ with 85W TDP, AMD Athlon 64 X2 4600+ with 65W TDP, AMD Athlon 64 X2 3800+ with 35W TDP, Intel Core 2 Duo E6300 processor, and Intel Pentium M 1.4 GHz processor. Among these processors, three AMD processors and Intel Core 2 Duo E6300 are high-performance processors with different power consumption rates. Fig. 2 demonstrates the power consumption rate of each processor in idle and busy working mode [44].

The Intel Pentium M processor aims to conserve power with modest performance. It uses third-generation Speed-Step technology to lower their clock speeds and core voltages. Consequently, Pentium M is capable of delivering acceptable performance if it is necessary while consuming less energy. Pentium M can shut down internal components such as unused segments of L2 cache to draw even less power. Although Intel Pentium M is designed for embedded systems and mobile devices, it also can be used in dense

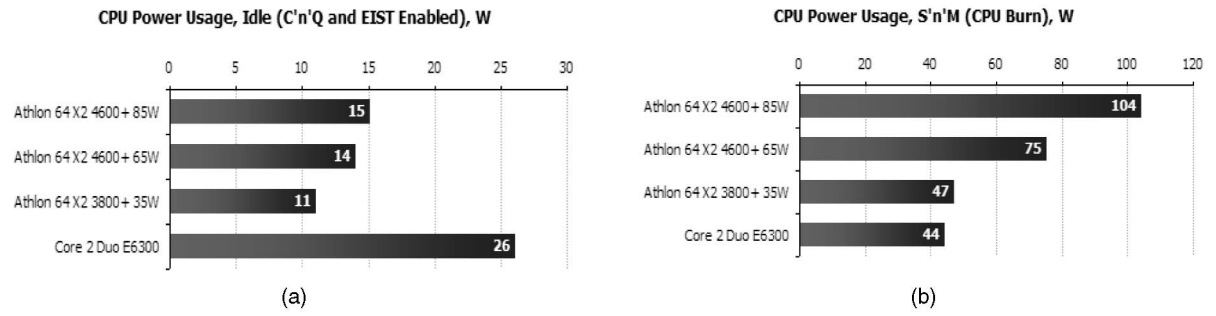


Fig. 2. Energy consumption parameters for processors in different working modes. (a) Power consumption rate in idle mode. (b) Power consumption rate in busy mode.

clustered server environments [45], [46]. Pentium M's clock speed scales between 600 MHz and 1.4 GHz; its voltage varies between 0.96V and 1.48V. This processor allows us to apply DVS to save energy. Table 2 summarizes the dynamic voltages and frequencies of Intel Pentium M processor.

5.2.2 Interconnection Configuration Profiles

To investigate the impacts of interconnects on power-performance efficiency, we consider four typical high-speed network interconnects: Gigabit Ethernet, Infiniband, Myrinet, and $\mathrm{QsNet}^{\mathrm{II}}$. These four types of interconnects with different power-performance profiles are widely used in real-world clusters. The features of the network interconnects are outlined as follows:

1. *Gigabit Ethernet* is a high-speed interconnect supporting full duplex links communication for computing nodes connected by switches. In our configuration profile for Gigabit Ethernet, we use Cisco Catalyst 2960G-24TC [47] as the switch and Intel PRO/1000 MT Dual Port Server Adapter [48] as the network interface card (NIC).
2. *Infiniband* is a switched fabric communications link primarily used in high-performance computing. For the infiniband configuration, the switch considered is Mellanox InfiniScaleTM III SDR [49] and NIC is Mellanox ConnectXTM IB Dual Copper Card [50].
3. *Myrinet* is a high-speed local area networking system designed by Myricom to be used as an interconnect between multiple machines to form computer clusters. The switch and NIC used for the Myrinet configuration are M3-4SW32-16Q Quad 32-Port Myrinet-2000 Switch Line Card [51] and Two-Port Myrinet-Fiber/PCI-X Network Interface Card [52].
4. $QsNet^{II}$ is a high-performance interconnect for supercomputer systems. The combination of high bandwidth, low latency, and scalability has made $\mathrm{QsNet}^{\mathrm{II}}$ the network choice for many of the world's fastest computer systems. For the configuration of $\mathrm{QsNet}^{\mathrm{II}}$, we choose E-Series Stand-alone switches QS8A [53] as the switch and QM509 (PCIe) [54] as the NIC.

Table 3 summarizes the configuration profiles for each type of interconnection. The power consumption rates (busy and idle) for switches in Gigabit Ethernet, Infiniband, and Myrinet are identical, because dynamic power management is not employed in these three network interconnects. However, power management for $\mathrm{QsNet}^{\mathrm{II}}$ switch is available (e.g., 42 W in busy mode and 36 W in idle mode). In terms of network speed, Myrinet serves as the standard network and we compare the message delay of other three interconnections over Myrinet. The detailed performance and energy efficiency of these interconnects can be found in [39]. The number of switches used in each interconnection may vary and it is decided by the processor number required and the ports number of switches. More specific, $\text{switch number} = \frac{\text{processor number}}{\text{port number}} + 1$. For example, given a parallel application requiring 30 processors, the number of switches served in Gigabit Ethernet, Infiniband, Myrinet, and $\mathrm{QsNet}^{\mathrm{II}}$ will be 2, 2, 1, and 4, respectively.

5.3 Simulator and Parameter Spaces

Schedule length and energy consumption are the two metrics used in our simulations to evaluate the performance and energy efficiency of the five algorithms. The schedule length indicates time spent in completing a parallel application. The energy consumption consists of two parts—processor energy consumption and network energy consumption.

A basic yet important rule applied in our simulations is Once Tuning One Parameter (OTOP). In each simulation experiment, we only change one parameter and keep the other parameters fixed. Tuning one parameter at a time allows us to clearly observe its impact on performance and energy efficiency of clusters.

The important parameters tuned in our simulations include processor type, interconnection type, and application type. Different processors and interconnects have various energy consumption profiles and latencies. We simulated two real-world parallel applications—the Robot Control application (with 88 tasks and 131 edges) and the fpppp application (with 334 tasks and 1,145 edges). The

TABLE 2
Dynamic Voltages and Frequencies of Intel Pentium M 1.4 GHz Processor

Frequency	Voltage
1.4GHz	1.484V
1.2GHz	1.436V
1.0GHz	1.308V
800MHz	1.180V
600MHz	0.956V

TABLE 3
Summaries of Network Configuration Profiles

Network	Switch Power (busy)	Switch Power (idle)	NIC Power	Port # per Switch	Message Delay over Myrinet
Gigabit Ethernet	75W	75 W	5 W	24	13
Infiniband	25 W	25 W	10.6 W	24	0.9
Myrinet	55.2 W	55.2 W	9.3 W	32	1
$\mathbf{QsNet^{II}}$	42 W	36 W	12 W	8	1.73

detailed information regarding these applications can be found at the Standard Task Graph website [55]. We also considered a large number of parallel applications generated by our synthetic parallel tasks generator.

Communication-Computation-Ratio (CCR) is an important parameter to represent the characteristic of a parallel application. CCR measures the ratio of communication time and computation time. A small CCR value means the application is computation-intensive; a large CCR value indicates that the application is communication-intensive. Generally speaking, an application running on a fixed number of processors and a certain type of interconnect has a specific CCR. However, CCR of the application may change when it is running on different processors and interconnects. Thus, we varied CCR in a reasonable range of 0.1 to 10.

5.4 Overall Performance-Energy Efficiency

Let us compare the overall performance-energy efficiency of the proposed EAD and PEBD algorithms against DVS, TDS, and MCP. Using $\mathrm{QsNet^{II}}$ network, we tested four applications, among which robot control and fpppp are real-world applications, random1 and random2 are synthetic parallel applications with 500 tasks.

We observe from Figs. 3a and 3b that TDS, EAD, and PEBD not only have the best performance, but also are the most energy-efficient algorithms for the robot control application. DVS has similar performance as MCP; DVS is more energy-efficient than MCP. EAD and PEBD are better than DVS in terms of both performance and energy savings. For example, EAD and PEBD improve the performance and energy efficiency of DVS by 10 and 6 percent, respectively. When it comes to MCP, the improvements are 10 percent in performance and 9 percent in energy efficiency, respectively.

Figs. 3c and 3d show that for the fpppp application, all five algorithms have similar performance. TDS is the least energy-efficient algorithm, whereas DVS is the most energy-efficient one. With respect to energy consumption, our algorithms are close to MCP. EAD and PEBD are slightly more energy-efficient than TDS, but they are two percent less energy-efficient than DVS.

The energy efficiency of the five algorithms are affected by the applications, because 1) robot control and fpppp have totally different DAGs and parallelism degrees and 2) the robot control is communication-intensive while fpppp is computation-intensive. For computation-intensive applications where CPU time dominates performance, task duplications simply pay extra energy overheads without boosting performance. In this experiment, our algorithms exhibit good capability of making a good balance between energy efficiency and performance.

Figs. 3e, 3f, 3g, and 3h show experimental results of two synthetic parallel applications. The results are consistent with those plotted in Figs. 3a, 3b, 3c, and 3d. Thus, the performance and energy efficiency of TDS are the best for communication-intensive applications, whereas DVS is likely to be the best choice for computation-intensive applications. Importantly, our EAD and PEBD are the only algorithms that maintain good performance and high energy efficiency for both computation-intensive and communication-intensive parallel applications.

5.5 Impact of Processors

Now we investigate the impacts of processors on the energy efficiency of cluster computing systems. To intuitively show energy consumption contributed by processors, we break down the total energy into two parts—CPU energy and network energy. In this experiment, we consider DVS-enabled processors as well as DVS-disabled processors. The DVS-disabled processors examined include AMD Athlon 64 X2 4600+ with 85W TDP, AMD Athlon 64 X2 4600+ with 65W TDP, AMD Athlon 64 X2 3800+ with 35W TDP, and Intel Core 2 Duo E6300 processor. Unlike the other processors, the Intel Pentium M processor is a DVS-enabled one, because it allows voltage and frequency to be adjusted on the fly based on dynamic workload conditions.

Figs. 4a and 4b show the total energy consumption and energy dissipation in the processors for the fpppp application. The first observation is that for all algorithms, Athlon 35W consumes the least energy while Core 2 duo consumes the most energy. The second observation is that the energy consumed by the processors dominates the total energy dissipation in the cluster, because the total energy consumption curves (see Fig. 4b) are very similar to the CPU energy curves (see Fig. 4a). This trend can be explained by taking a look at the power of these two CPUs. The power of Athlon 35W is 47 and 11 W when it is busy and idle; the busy and idle power rates of Core 2 duo are 44 and 26 W. Although they almost have the same power rate when CPU is busy, the gap between their idle powers is 15 W. Therefore, Athlon 35W can save a huge amount of energy provided that applications offer enough opportunities for it to transit into the idle mode.

Figs. 4c and 4d show the energy consumption trend for a synthetic application, which has a very similar DAG as that of fpppp but with higher CCR. In this case, Athlon 35W still consumes less energy than Core 2 duo processor. However, the total energy consumed by the Athlon 35W cluster exceeds the total energy consumed by the Core 2 duo cluster. This result indicates that the total energy in fpppp is dominated by the network interconnects, because the application is communication-intensive with the high CCR value.

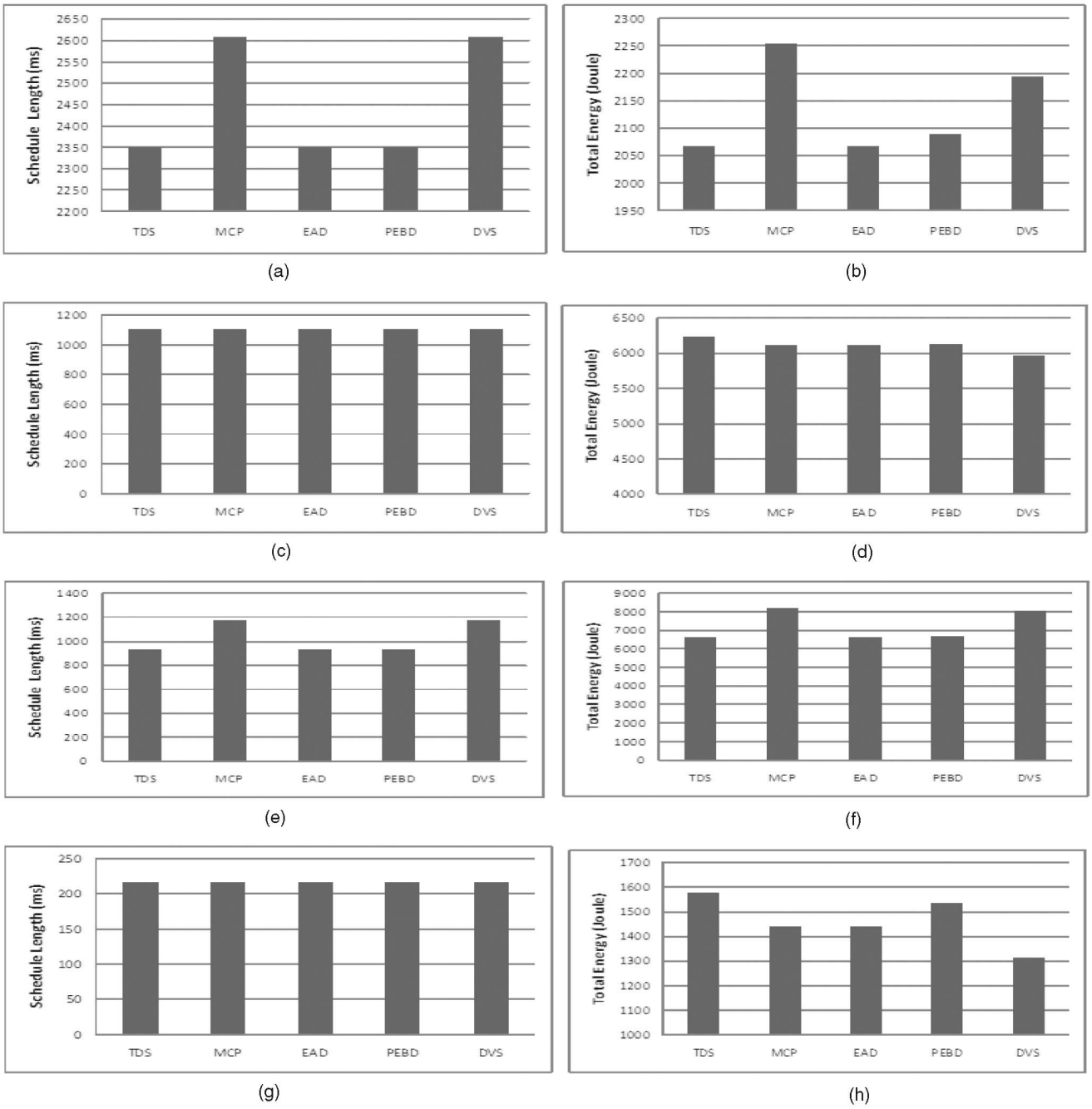


Fig. 3. Overall performance-energy efficiency comparisons. (a) Schedule length of robot control. (b) Energy consumption of robot control. (c) Schedule length of fpppp. (d) Energy consumption of fpppp. (e) Schedule length of random 1 ($\mathrm{ccr} = 5$). (f) Energy consumption of random 1 ($\mathrm{ccr} = 5$). (g) Schedule length of random 2 ($\mathrm{ccr} = 0.1$). (h) Energy consumption of random 2 ($\mathrm{ccr} = 0.1$).

Fig. 5 demonstrates the energy consumption incurred by two synthetic applications with 500 tasks on DVS-enabled processors. The first application is communication-intensive and the second one is computation-intensive. An obvious observation is that DVS is beneficial to save energy in processors for both computation-intensive and communication-intensive applications. The impact of DVS, however, may not be strong enough to dominate the total energy consumption. For example, Figs. 5a, 5b, and 5c show that DVS consumes much less energy in processors, but this advantage is leveraged in the network interconnects. Although our algorithms are not as good as DVS in terms of saving energy in processors, ours do conserve energy in networks by the virtue of task replicas, which lead to potential total energy savings for parallel applications with high CCR values. For applications with low CCRs, DVS is likely to be the best energy-efficient algorithm in most cases as CPU energy dominates total energy in clusters.

5.6 Impact of Interconnections

Processors and interconnections are two decisive factors making up of the energy-performance profiles of clusters. We have discussed the impact of processors in Section 5.5. In this section, we show the impact of interconnections on performance and energy efficiency. Our experimental results plotted in Fig. 6 are based on $\mathrm{QsNet}^{\mathrm{II}}$, Myrinet, Infiniband,

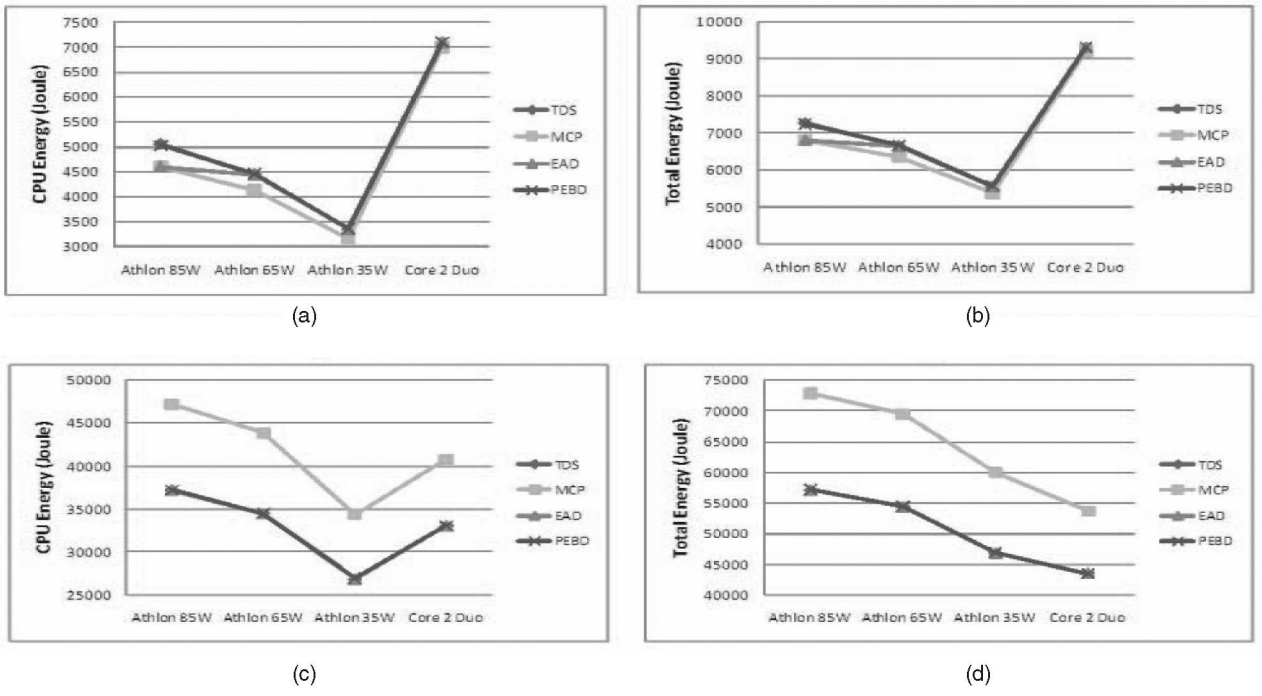


Fig. 4. Energy impact of four different DVS-disabled processors.

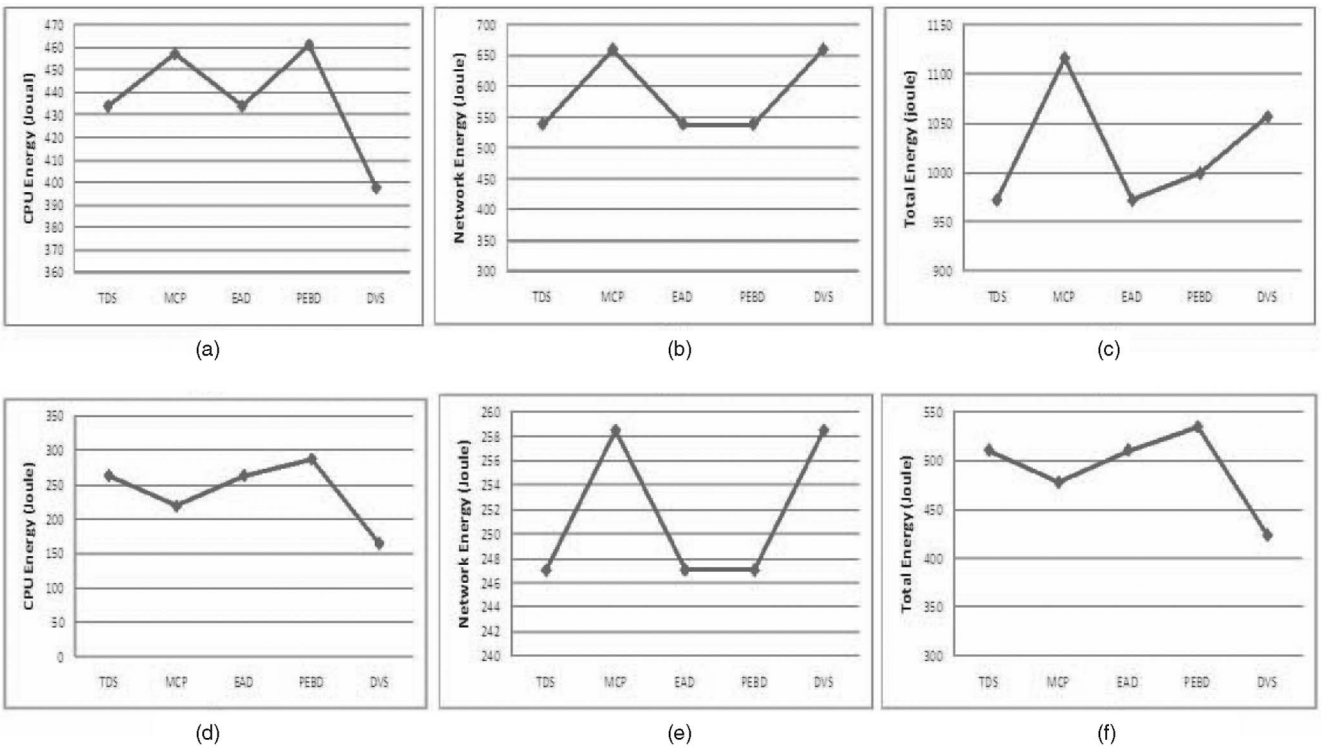


Fig. 5. Energy impact of DVS-enabled Intel Pentium M processor.

and Gigabit Ethernet, which are widely used interconnects in real-world clusters. Please refer to Section 5.2.2 for detailed information regarding these four types of interconnections.

We assume that the Intel Pentium M processors are used in the simulated clusters. We did not test any non-DVS-enabled processors because they do not support DVS. We use the robot control application rather than the fpppp application because robot control is communication-intensive. In doing so, we can highlight energy savings in network interconnects. To make fair comparison, we fixed all system parameters except for those of the four types of interconnections.

Figs. 6a and 6b reveal that the overall performance and energy efficiency of clusters equipped with the four interconnections. The result shows that regardless of the scheduling algorithms, the schedule lengths of the application running on a cluster with Ethernet is much longer than the

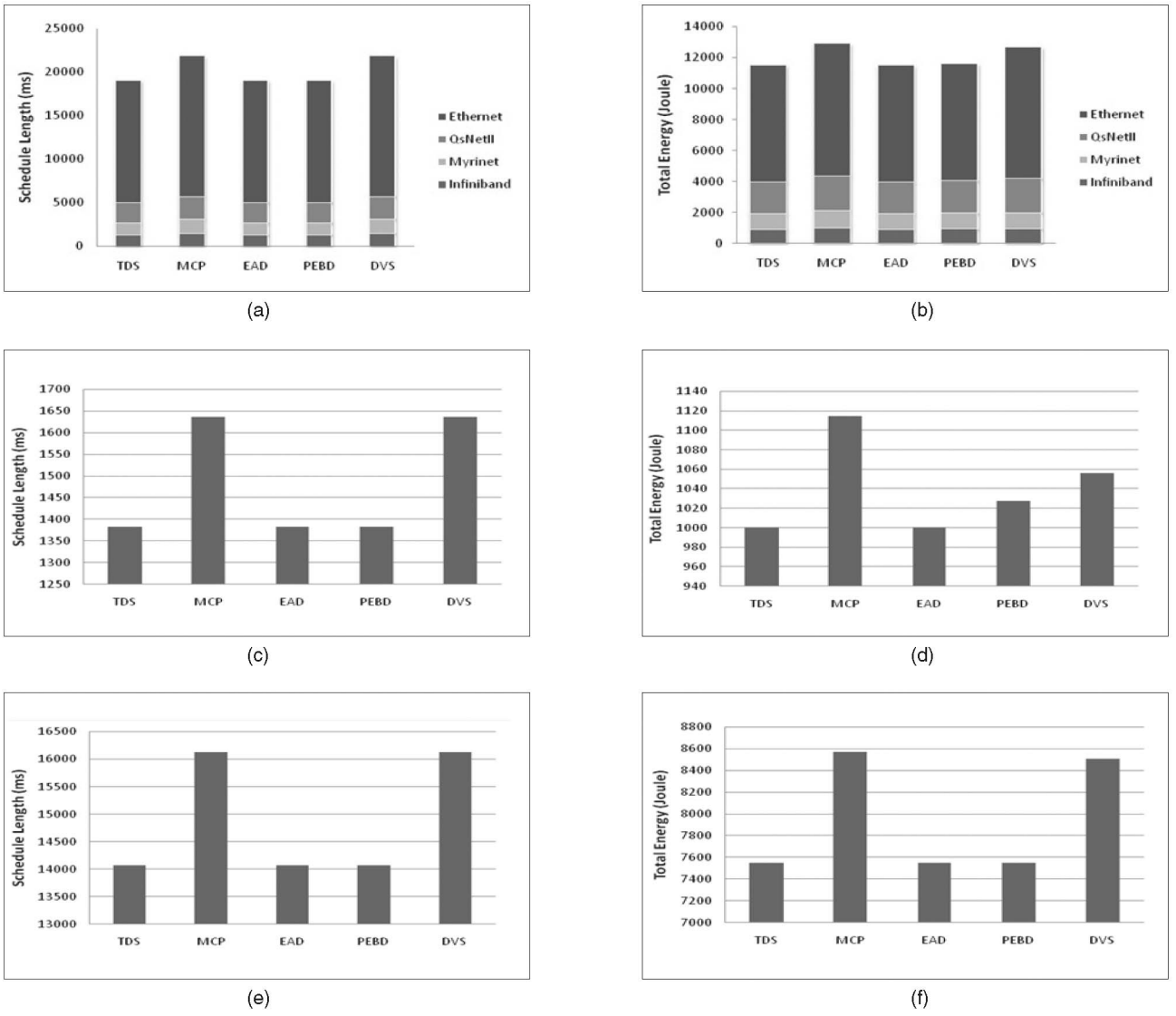


Fig. 6. Impact of interconnections on energy-performance efficiency. (a) Schedule length comparison. (b) Energy consumption comparison. (c) Schedule length comparison (Myrinet). (d) Energy consumption comparison (Myrinet). (e) Schedule length comparison (Ethernet). (f) Energy consumption comparison (Ethernet).

application on the same clusters with the other three types of interconnections. Accordingly, Ethernet consumes much more energy due to increased communication times.

A second observation drawn from Figs. 6a and 6b is that Myrinet and Infiniband have similar performance and energy efficiency. Myrinet and Infiniband are slightly better than QsNet in terms of schedule lengths and energy savings. The power differences among the four interconnects have marginal impact on energy efficiency. However, the differences among network latencies noticeably affect both performance and energy conservation. The latency of Ethernet is the largest among those of the four interconnections. Infiniband, on the other hand, has the shortest latency. The result implicates that scheduling algorithms can leverage interconnects with low network latencies to achieve high performance and energy efficiency. This implication is especially true for communication-intensive applications, because low network latencies can reduce communication times, which, in turn, lead to shortened schedule lengths and saved power.

Figs. 6c, 6d, 6e, and 6f show energy-performance comparison between Myrinet and Ethernet. We see from these figures that TDS and our algorithms are better than MCP and DVS with respect to both performance and energy savings. Although DVS has similar performance as that of MCP, DVS is more energy-efficient than MCP. The performance-energy improvements yielded by our algorithms vary as we deploy different interconnections to the cluster. For example, EAD improves performance by 18.36 percent and energy efficiency by 5.58 percent over DVS when Myrinet is employed. The performance and energy efficiency improvement of EAD over DVS become 14.68 and 12.74 percent if Ethernet is deployed in the cluster. Although performance and energy efficiency of EAD and PEBD are remarkably similar in many cases, they are distinct under some workload conditions (see, for example, Fig. 6d).

5.7 Impact of Communication-Computation-Ratio

Communication-Computation-Ratio measures the total time spent on computation and communication. In this set of

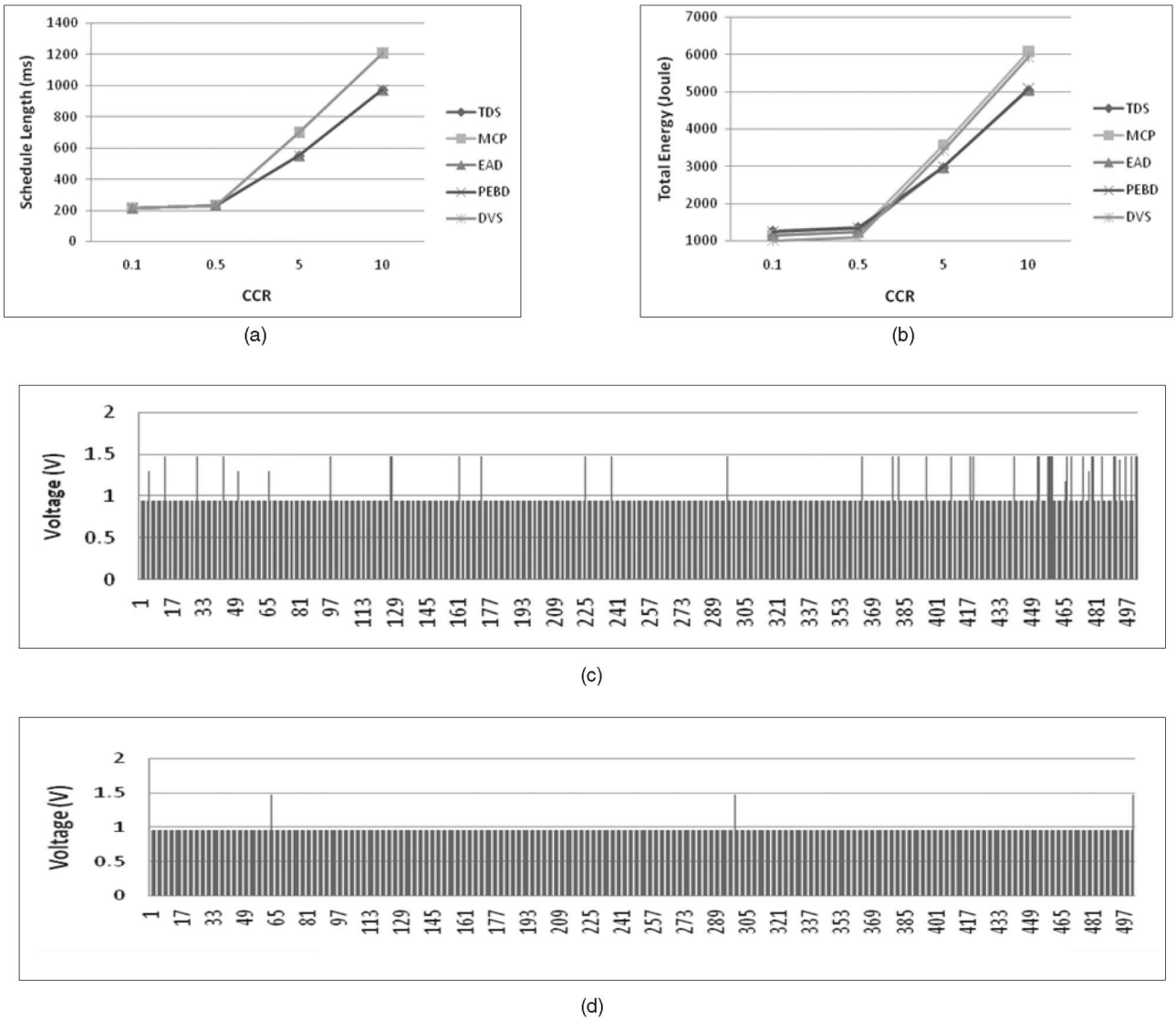


Fig. 7. Impact of CCR on energy-performance efficiency. (a) Schedule length comparison for different CCRs. (b) Energy consumption comparison for different CCRs. (c) Dynamic voltage scaling traces when $\mathrm{CCR} = 0.1$. (d) Dynamic voltage scaling traces when $\mathrm{CCR} = 5$.

experiments, we investigate the impact of CCR on performance and energy efficiency of parallel applications running on clusters. More generally speaking, CCR of an application is fixed for a given cluster computing platform. To analyze the impact of CCR on energy efficiency and performance, we vary the CCR of a synthetic application with 500 tasks.

Figs. 7a and 7b depict schedule lengths and energy consumption of the five scheduling algorithms. Two observations are evident from the analysis. First, when CCRs are small, DVS is slightly more energy-efficient than EAD and PEBD. The energy efficiency of our algorithms is noticeably better than those of MCP and TDS. Second, when CCR is increased to 10, EAD and PEBD are substantially better than MCP and DVS in terms of both schedule length and energy conservation. For example, the performance improvements over DVS and MCP are 27.36 and 24.12 percent when CCR is set to 5 and 10, respectively. EAD and PEBD improve energy efficiency by 20.1 percent over MCP and by 16 percent over DVS. These improvements are consistent with the improvements achieved by our algorithms when the robot control and fpppp applications are scheduled. The implication of this result is that EAD and PEBD are conducive to saving energy caused by communication-intensive parallel applications on clusters.

Figs. 7c and 7d record the voltage trace for the DVS-enabled processors when DVS is applied. Fig. 7c shows that the workload of the first application provides reasonable opportunities for DVS to conserve energy by dynamically adjusting voltage levels. The voltage varies in all levels between the lowest and the highest voltage levels. In contrast, Fig. 7d shows that the workload of the second application leaves no further room for DVS to save energy. The processor remains the lowest level of voltage in the majority of time, because in this case the schedule length depends on how fast messages can be passed rather than how fast tasks can be executed. The results plotted in Figs. 7c and 7d empirically validate our argument that DVS is an efficient algorithm to conserve energy of computation-intensive parallel applications on clusters.

6 Conclusions

In this paper, we have addressed the issue of scheduling and allocating parallel tasks running on homogeneous clusters with an objective of improving both performance and energy efficiency. To achieve this goal, we proposed two energy-aware duplication-based scheduling algorithms, namely the Energy-Aware Duplication (EAD) algorithm and the Performance-Energy Balanced Duplication (PEBD) algorithm.

In addition to presenting EAD and PEBD, we built mathematical models to describe a cluster computing framework, parallel applications with precedence constraints, and energy dissipations in clusters. To demonstrate the effectiveness and practicality of the proposed duplication-based scheduling algorithms, we conducted extensive experiments using both synthetic and real-world parallel applications running on a simulated cluster. The empirical results illustrate that EAD and PEBD are capable of substantially improving energy efficiency and performance of a cluster running communication-intensive parallel applications. Our novel scheduling algorithms can archive the overall performance-energy improvement over the existing solutions by up to 20 percent. The drawback of our approaches is that when it comes to computation-intensive applications, EAD and PEBD are slight less energy-efficient than the DVS technique. This shortcoming can be eliminated by using DVS to schedule computation-intensive parallel applications.

Future studies in this research can be performed in the following directions. First, we will extend our algorithms to multidimensional computing resources from which energy conservation can be achieved. In this study, we primarily considered the energy consumption of processors and interconnections. Memory access and I/O activities will be investigated in our future studies. Second, we will modify the EAD and PEBD algorithms to handle parallel applications on heterogeneous clusters, where computational nodes have different processing capabilities and network interconnects may have various performance.

7 Availability

The executable binaries and source code, along with the documentation for experimentation, will be freely available at http://www.mcs.sdsmt.edu/zzong/software/scheduling.html.

Acknowledgments

The authors sincerely appreciate the comments and feedbacks from the anonymous reviewers. Their valuable discussions and thoughts have tremendously helped in improving the quality of this paper. The work reported in this paper was supported by the US National Science Foundation under Grants No. CNS-0915762 (CSR), CCF-0845257 (CAREER), CNS-0757778 (CSR), CCF-0742187 (CPA), CNS-0917137 (CSR), CNS-0831502 (CyberTrust), CNS-0855251(CRI), OCI-0753305 (CI-TEAM), DUE-837341 (CCLI), and DUE-0830831 (SFS), as well as Auburn University under a start-up grant, a gift (Number 2005-04-070) from the Intel Corporation, and South Dakota School of Mines and Technology under the Nelson Research Grant.

References

[1] "Electrical Energy," *The New Book of Popular Science.* Grolier Inc., 2000.

[2] http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf, 2010.

[3] S. Darbha and D.P. Agrawal, "Optimal Scheduling Algorithm for Distributed-Memory Machines," *IEEE Trans. Parallel and Distributed Systems,* vol. 9, no. 1, pp. 87-95, Jan. 1998.

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

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

[6] M. Warren, E. Weigle, and W. Feng, "High-Density Computing: A 240-Node Beowulf in One Cubic Meter," *Proc. ACM/IEEE Supercoputing (SC '02),* Nov. 2002.

[7] A. Gara, M.A. Blumrich, D. Chen, G.L.-T. Chiu, P. Coteus, M.E. Giampapa, R.A. Haring, P. Heidelberger, D. Hoenicke, G.V. Kopcsay, T.A. Liebsch, M. Ohmacht, B.D. Steinmacher-Burow, T. Takken, and P. Vranas, "Overview of the Blue Gene/L System Architecture," *IBM J. Research and Development,* vol. 49, pp. 195-212, http://www.research.ibm.com/journal/rd49-23.html, 2005.

[8] "Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor," Intel white paper, ftp://download.intel.com/design/network/papers/30117401.pdf, 2010.

[9] "Cool'n'Quiet Technology Installation Guide for AMD Athlon 64 Processor Based Systems," http://www.amd.com/us-en/assets/content_type/DownloadableAssets/Cool_N_Quiet_Installation_Guide3.pdf, 2010.

[10] W. Dally, P. Carvey, and L. Dennison, "The Avici Terabit Switch/Rounter," *Proc. IEEE Hot Interconnects 6,* pp. 41-50, Aug. 1998.

[11] E.N.M. Elnozahy, M. Kistler, and R. Rajamony, "Energy-Efficient Server Clusters," *Proc. Int'l Workshop Power-Aware Computer Systems,* Feb. 2002.

[12] Mellanox Technologies Inc.,"Mellanox Performance, Price, Power, Volume Metric (PPPV)," http://www.mellanox.co/products/shared/PPPV.pdf, 2004.

[13] C. Gunaratne, K. Christensen, and B. Nordman, "Managing Energy Consumption Costs in Desktop PCs and LAN Switches with Proxying, Split TCP Connections, and Scaling of Link Speed," *Int'l J. Network Management,* vol. 15, no. 5, pp. 297-310, Sept./Oct. 2005.

[14] G.C. Sih and E.A. Lee, "A Compile Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processors Architectures," *IEEE Trans. Parallel and Distributed Systems,* vol. 4, no. 2, pp. 175-187, Feb. 1993.

[15] 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 Systems,* vol. 6, no. 4, pp. 388-399, Apr. 1995.

[16] L. Benini and G. De Micheli, *Dynamic Power Management: Design Techniques and CAD Tools.* Kluwer Academic Publishers, 1998.

[17] A.R. Chandrakasan and R.W. Brodersen, *Low Power Digital CMOS Design.* Kluwer Academic Publishers, 1995.

[18] *Lower Power Design Methodologies,* J. Rabaey and M. Pedram, eds., Kluwer Academic Publishers, 1998.

[19] A. Raghunathan, N.K. Jha, and S. Dey, *High-Level Power Analysis and Optimization.* Kluwer Academic Publishers, 1998.

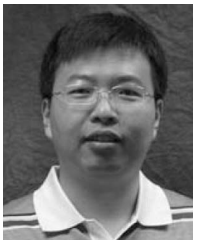
[20] L. Benini, A. Bogliolo, and G.D. Micheli, "A Survey of Design Techniques for System-Level Dynamic Power Management," *IEEE Trans. Very Large Scale Integration Systems,* vol. 8, no. 3, pp. 299-316, June 2000.

[21] M. Srivastava, A. Chandrakasan, and R. Brodersen, "Predictive System Shutdown and Other Architectural Techniques for Energy Efficient Programmable Computation," *IEEE Trans. Very Large Scale Integration Systems,* vol. 4, no. 1, pp. 42-55, Mar. 1996.

[22] K. Flautner, S.K. Reinhardt, and T.N. Mudge, "Automatic Performance Setting for Dynamic Voltage Scaling," *Proc. Seventh Conf. Mobile Computing and Networking,* pp. 260-271, 2001.

[23] D. Grunwald, P. Levis, K.I. Farkas, C.B. Morrey III, and M. Neufeld, "Policies for Dynamic Clock Scheduling," *Proc. Fourth Symp. Operating Systems Design and Implementation (OSDI),* pp. 73-86, 2000.

[24] J.R. Lorch and A.J. Smith, "Improving Dynamic Voltage Scaling Algorithms with PACE," *ACM SIGMETRICS Performance Evaluation Rev.*, vol. 29, pp. 50-61, 2001.
[25] T.L. Martin, "Balancing Batteries, Power, and Performance: System Issues in CPU Speed-Setting for Mobile Computing," PhD thesis, Carnegie Mellon Univ., 2001.
[26] A. Miyoshi, C. Lefurgy, E.C. Hensbergen, R. Rajamony, and R. Rajkumar, "Critical Power Slope: Understanding the Runtime Effects of Frequency Scaling," *Proc. 16th Int'l Conf. Supercomputing*, pp. 35-44, 2002.
[27] N. Kappiah, D.K. Lowenthal, and V.W. Freeh, "Just In Time Dynamic Voltage Scaling: Exploiting Inter-Node Slack to Save Energy in MPI Programs," *Proc. ACM/IEEE Supercomputing Conf. (SC '05)*, 2006.
[28] M. Annavaram, E. Grochowski, and J. Shen, "Mitigating Amdahl's Law through EPI throttling," *Proc. 32nd Ann. Int'l Symp. Computer Architecture (ISCA '05)*, pp. 298-309, June 2005.
[29] R. Bianchini and R. Rajamony, "Power and Energy Management for Server Systems," *Computer*, vol. 37, no. 11, pp. 68-76, Nov. 2004.
[30] M.Y. Lim, V.W. Freeh, and D.K. Lowenthal, "Adaptive, Transparent Frequency and Voltage Scaling of Communication Phases in MPI Programs," *Proc. ACM/IEEE Supercomputing (SC '06)*, 2006.
[31] R. Springer, D.K. Lowenthal, B. Rountree, and V.W. Freeh, "Minimizing Execution Time in MPI Programs on an Energy-Constrained, Power-Scalable Cluster," *Proc. 11th ACM SIGPLAN Symp. Principles and Practice of Parallel Programming (PPoPP '06)*, pp. 230-238, 2006.
[32] C.-H. Hsu and W.-C. Feng, "A Power-Aware Run-Time System for High-Performance Computing," *Proc. ACM/IEEE Supercomputing (SC '05)*, 2005.
[33] Y. Hotta, M. Sato, H. Kimura, S. Matsuoka, T. Boku, and D. Takahashi, "Profile-Based Optimization of Power Performance by Using Dynamic Voltage Scaling on a PC Cluster," *Proc. 20th IEEE Int'l Parallel and Distributed Processing Symp. (IPDPS '06)*, 2006.
[34] C.-H. Hsu and W.-C. Feng, "A Feasibility Analysis of Power Awareness in Commodity-Based High-Performance Clusters," *Proc. IEEE Int'l Conf. Cluster Computing (Cluster '05)*, 2005.
[35] L. Shang, L. Peh, and N.K. Jha, "Power-Efficient Interconnection Networks: Dynamic Voltage Scaling with Links," *Computer Architecture Letters*, vol. 1, no. 1, p. 6, Jan. 2002.
[36] L. Shang et al., "Dynamic Voltage Scaling with Links for Power Optimization of Interconnection Networks," *Proc. Ninth Int'l Symp. High-Performance Computer Architecture (HPCA-9)*, pp. 79-90, Feb. 2003.
[37] V. Soteriou and L.-S. Peh, "Dynamic Power Management for Power Optimization of Interconnection Networks Using On/Off Links," *Proc. 11th Symp. High Performance Interconnects (Hot Interconnects)*, Aug. 2003.
[38] C. Gunaratne, K. Christensen, B. Nordman, and S. Suen, "Reducing the Energy Consumption of Ethernet with Adaptive Link Rate (ALR)," *IEEE Trans. Computers*, vol.57, no. 4, pp. 448-461, Apr. 2008.
[39] R. Zamani, A. Afsahi, Y. Qian, and C. Hamacher, "A Feasibility Analysis of Power-Awareness and Energy Minimization in Modern Interconnects for High-Performance Computing," *Proc. Ninth IEEE Int'l Conf. Cluster Computing (Cluster '07)*, Sept. 2007.
[40] Y.-K. Kwok and I. Ahmad, "Efficient Scheduling of Arbitrary Task Graphs to Multiprocessors Using a Parallel Genetic Algorithm," *J. Parallel and Distributed Computing*, vol. 47, no.1, pp. 58-77, 1997.
[41] R.L. Graham, L.E. Lawler, J.K. Lenstra, and A.H. Kan, "Optimizing and Approximation in Deterministic Sequencing and Scheduling: A Survey," *Annals of Discrete Math*, vol. 5, pp. 287-326, 1979.
[42] M.Y. Wu and D.D. Gajski, "Hypertool: A Performance Aid for Message-Passing Systems," *IEEE Trans. Parallel and Distributed Systems*, vol. 1, no. 3, pp. 330-343, July 1990.
[43] R. Ge, X.Z. Feng, and K.W. Cameron, "Performance-Constrained Distributed DVS Scheduling for Scientific Applications on Power-Aware Clusters," *Proc. ACM/IEEE Supercomputing Conf. (SC '05)*, p. 34, Nov. 2005.
[44] http://www.xbitlabs.com/articles/cpu/display/amd-energy-efficient_6.html, 2010.
[45] http://www.intel.com/design/intarch/pentiumm/pentiumm.htm, 2010.
[46] http://techreport.com/articles.x/5454/1, 2010.
[47] http://www.euroone.hu/docs/WS2960_datasheet.pdf, 2010.
[48] http://www.intel.com/network/connectivity/resources/doc_library/data_sheets/pro1000mt_sa_dual.pdf, 2010.
[49] http://www.mellanox.com/pdf/products/silicon/InfiniScaleIII.pdf, 2010.
[50] http://www.mellanox.com/pdf/products/hca/ConnectX_IB_Card.pdf, 2010.
[51] http://www.myri.com/myrinet/14U_switches/M3-4SW32-16Q/, 2010.
[52] http://www.myri.com/myrinet/PCIX/m3f2-pcixe.html, 2010.
[53] http://www.quadrics.com/Quadrics/QuadricsHome.nsf/DisplayPages/3A912204F260613680256DD9005122C7, 2008.
[54] http://www.quadrics.com/Quadrics/QuadricsHome.nsf/DisplayPages/3A912204F260613680256DD9005122C7, 2008.
[55] Standard Task Graph Set web site, http://www.kasahara.elec.waseda.ac.jp, 2010.



Ziliang Zong received the BS and MS degrees in computer science from Shandong University of China in 2002 and 2005, respectively, and the PhD degree in computer science from Auburn University in 2008. Currently, he is an assistant professor in the Mathematics and Computer Science Department of South Dakota School of Mines and Technology. His research interests include multicore technologies, parallel programming, high-performance computing, and distributed storage systems. In 2009, he received the US National Science Foundation (NSF) Computer and Networked Systems (CNS) Award.



Adam Manzanares received the BS degree in computer science from the New Mexico Institute of Mining and Technology, in 2002. Currently, he is a PhD student in the Department of Computer Science and Software Engineering at Auburn University. During the summers of 2002-2007, he worked as a student intern at the Los Alamos National Laboratory. His research interests include energy-efficient computing, modeling and simulation, and high performance computing.



Xiaojun Ruan received the BS degree in computer science from Shandong University in 2005. Currently, he is a PhD student in the Department of Computer Science and Software Engineering at Auburn University. His research interests are in parallel and distributed systems, storage systems, real-time computing, performance evaluation, and fault tolerance. His research interests focus on high-performance parallel cluster computing, storage system, and distributed system.



Xiao Qin received the BS and MS degrees in computer science from Huazhong University of Science and Technology, China, in 1996 and 1999, respectively, and the PhD degree in computer science from the University of Nebraska-Lincoln in 2004. He is currently an associate professor of computer science at Auburn University. Before joining Auburn University, he was with New Mexico Institute of Mining and Technology. His research interests include parallel and distributed systems, real-time computing, storage systems, and performance evaluation. He won an NSF Career Award in 2009. He has served on the program committees of several conferences, including the IEEE Cluster, the IEEE IPCCC, and the ICPP. He is a senior member of the IEEE.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.