Mathematical and Computational Applications

MDPI

Article

A Novel, Energy-Aware Task Duplication-Based Scheduling Algorithm of Parallel Tasks on Clusters

Aihua Liang [1,*] and Yu Pang [2]

[1] Beijing Key Laboratory of Information Service Engineering, Beijing Union University, No. 97 Beisihuan East Road, Chaoyang District, Beijing 100101, China

[2] Computer and Engineering School, Beihang University, No. 37 Xueyuanlu, Haidian District, Beijing 100191, China; pangyu57@126.com

* Correspondence: liangaihua@buu.edu.cn; Tel.: +86-10-64900229

Academic Editor: Fazal M. Mahomed

Received: 21 September 2016; Accepted: 19 December 2016; Published: 27 December 2016

Abstract: Increasing energy has become an important issue in high performance clusters. To balance the energy and performance, we proposed a novel, energy-aware duplication-based scheduling (NEADS). An existing energy-aware duplication-based algorithm replicates all qualified predecessor tasks in a bottom-up manner. Some tasks without direct relation may be replicated to the same processor, which cannot reduce the communication energy. Instead, the computation overhead may be increased. In contrast, the proposed algorithm only replicates the directly correlated predecessor tasks in the energy threshold range without lengthening the schedule length. The proposed algorithm is compared with the non-duplication algorithm and existing duplicated-based algorithm. Extensive experimental results show that the proposed algorithm can effectively reduce energy consumption in various applications. It has advantages over other algorithms on computation-intensive applications.

Keywords: task duplication; scheduling; energy-aware; clusters; Directed Acyclic Graph (DAG)

1. Introduction

In a high performance computing system, clusters have been widely applied to many fields of scientific and commercial applications as primary and cost-effective infrastructures. Performance improvement has long been the sole objective of different architectural and software studies [1]. However, with the increased system scale, high energy cost has become a prominent issue. For example, on November 2015, the number 1 of the top 500, Tianhe-2 (National University of Defense Technology, Guangzhou, China) consumed more than 17,000 Kilowatts each hour [2]. Therefore, designing energy-efficient clusters from hardware and software is highly urgent.

Increasing evidence shows that interconnection fabric consumes considerable amounts of energy in clusters. For example, interconnections consume 33% of the total energy in the Avici switch (Avici Systems Inc., North Billerica, MA, USA) [3]. Parallel task scheduling strategies can allocate the resources according to task priorities, which is one of the software methods for optimizing the system energy. Among them, task duplication has been proved to be an effective strategy to improve the performance of parallel task scheduling with precedence constraints [4,5]. For reducing the communication overheads, tasks are duplicated to several processors and executed in idle cycles of processors. Duplication-based scheduling algorithms replicate all possible tasks to shorten their length without considering energy consumption caused by duplicated task execution.

Therefore, some energy-aware duplication-based scheduling algorithms are proposed, which selectively replicate the tasks. However, these duplicate-based scheduling strategies replicate tasks only according to the energy difference between current task computation energy and communication energy of these two tasks. With this condition, no direct correlation tasks may

be replicated and executed in the same processor. It cannot reduce the communication overhead, but increase the computation overhead of the duplicated tasks. Thus, the overall energy consumption would increase.

To better balance the energy and performance, we propose a novel task duplication scheduling algorithm to replicate tasks according to direct precedence constraints and increased energy threshold. The proposed algorithm aims to reduce the communication energy without increasing the additional computation overhead.

Since our algorithm is based on the Directed Acyclic Graph (DAG), some important parameters of DAG are needed. The notation and description are listed in Table 1. Detailed descriptions are given respectively as follows. The detailed explanations of these parameters are presented in Section 4.

Table 1. Notation list.

Notation	Description
CCR	Communication-computation ratio
PRED(v_i)	The predecessor of v_i
SUCC(v_i)	The successor of v_i
$EST(v_i)$	The earliest start time of v_i
$ECT(v_i)$	The earliest complete time of v_i
$LAST(v_i)$	The latest allowed start time of v_i
$LACT(v_i)$	The latest allowed complete time of v_i
B-level(v_i)	The longest distance from v_i to exit task
T-level(v_i)	The longest distance from entry task to v_i
FP(v_i)	Favorite predecessor of v_i
SFP(v_i)	Second favorite predecessor of v_i

The rest of the paper is organized as follows: In Section 2, we present related work. Section 3 introduces computational models including a task model and an energy consumption model. In Section 4, the energy-aware duplication-based scheduling strategy is presented. Simulation results are demonstrated in Section 5. Finally, Section 6 provides the conclusion.

2. Related Work

Many studies have investigated power-aware techniques through scheduling algorithms. Dynamic power management (DPM) [6] and Dynamic voltage and frequency scaling (DVFS) techniques are widely adopted to reduce power consumption in clusters [7–10]. DPM is an effective energy saving method that attempts to adjust the power mode according to the workload variation [11,12]. In a broad sense, DVFS can be regarded as the CPU level DPM. This strategy can adjust the voltage or frequency of processors to achieve high energy efficiency for processors. So it has significant advantage in computation-intensive applications. However, for communication-intensive applications its benefits could not exist anymore.

A parallel application program can be divided into many tasks with precedence constraints. These tasks require data processing and data communication. Since task scheduling is an NP-hard problem, many heuristic scheduling strategies are proposed for performance improvements. Parallel task scheduling algorithms can be classified into three categories, called list scheduling, clustering-based scheduling, and duplication-based scheduling respectively. List scheduling assigns the priorities for tasks and maps tasks to processors based on assigned priorities [13]. Clustering-based scheduling clusters intercommunicating tasks in groups and allocate tasks of the same group to the same processor [14]. Duplication-based scheduling focuses on replicating some tasks when the scheduling length can be shortened [15,16]. Reducing communication overheads is the main goal of this scheduling strategy.

Baskiyar et al. [17] proposed a scheduling algorithm called EADAGS, which integrated dynamic voltage scaling and decisive path scheduling for a heterogeneous distributed processing

system. Dynamic Voltage Scaling (DVS) is applied to the slack time, and the real-time energy consumption can be calculated. Sinnen et al. focused on the communication contention problem [18]. The communication link competition in task communication is considered. They proposed an energy-aware list-based scheduling algorithm [18] and duplication-based scheduling algorithm [19]. The communication path is selected based on the routing algorithm. Task execution is related to the performance of routing algorithm. Choi et al. [20] presented a co-scheduling strategy, which can integrate the computation intensive tasks and communication intensive tasks by monitoring the processes. Ma'sko et al. [21] proposed the synthetic algorithm based upon list scheduling and clustering scheduling for Symmetric Multi-Processing (SMP) clusters. Shojafar et al. proposed the energy-efficient resource management for data centers [22,23]. A joint meta-heuristic approach to a cloud job scheduling algorithm is proposed in [24], which uses fuzzy theory and a genetic method.

Duplication-based scheduling strategies aim to reduce communication overheads. However, the computation overhead would increase because the duplicated tasks are executed in many processors. Accordingly, the overall energy consumption would augment. Zong et al. [25,26] focused on the energy optimization of task duplication-based scheduling strategies. They proposed two energy-aware duplication-based algorithms, called EAD (Energy-Aware Duplication) scheduling and PEBD (Performance-Energy Balanced Duplication) scheduling, which do not replicate all performance-oriented tasks. They consider both performance improvement and energy cost. The energy threshold and energy efficiency threshold are set to be the condition when duplicating tasks. The favorite predecessor task is defined. If the favorite predecessor satisfies the given condition, it would be replicated. However, if it is not qualified to replicate, its predecessor task is still judged under the same condition. Therefore, some tasks may be mapped to the same processor, but they are not directly correlated. If so, communication energy cannot be reduced, and more computation energy would be generated.

To address the above problem, a more energy-efficient duplication-based scheduling strategy is required to reduce the communication overhead and not increase the unnecessary computation energy consumption.

3. Computation Model

3.1. Task Model

A cluster consists of a set of computation processors, which can be represented as

$$P = \{P_1, P_2, \ldots, P_m\} \tag{1}$$

In the above equation, m is a positive integer and indicates the number of processors. P_i is one of the processor.

The tasks with precedence constraint can be represented as DAG, which includes the vertexes and directed edges with computation weight and communication weight. A vertex denotes a task and the directed edge denotes the communication relation of tasks. A task needs be allocated to a processor to run, which is indivisible and its execution cannot be interrupted. A task can be represented as the following vector.

$$G = (V, E, C, T) \tag{2}$$

V is the task set, which includes n parallel tasks. It can be represented as

$$V = \{v_1, v_2, \ldots, v_n\} \tag{3}$$

E is the communication edge set, which includes all directed edges of DAG. It denotes the precedence dependencies between tasks and can be represented as

$$E = \{e_{ij} = (v_i, v_j) | 1 \leq i, j \leq n\} \tag{4}$$

e_{ij} denotes the edge from task v_i to task v_j.

C is the set of communication overhead, which is the weight set of directed edges in DAG. c_{ij} means the communication time from task v_i to task v_j.

T is the set of computation overhead, which is the weight set of vertexes in DAG. t_i denotes the computation time of task v_i.

If there exists an edge e_{ij}, we call v_i is the predecessor of v_j and v_j is the successor of v_i. Only if the all predecessor tasks have finished can the current task start to run. The communication overhead is generated due to the tasks being executed in different processors. So if two related tasks are executed in the same processor, their communication overhead can be regarded as zero.

The schedule length or makespan of parallel tasks denotes the overall time from the first task (entry task of DAG) to the last task (exit task of DAG).

3.2. Energy Model

The overall energy includes the computation energy (EP) and communication energy (EC), which can be represented as the following equation.

$$E = EP + EC \tag{5}$$

The computation energy can be represented as the product of the processor speed and execution time. R_p is the speed of processor, which is assumed to be fixed. EP can be represented as

$$EP = \sum_{i=1}^{n} R_p t_i \tag{6}$$

The communication energy caused by edge e_{ij} can be denoted as el_{ij}, which can also be represented as the product of communication rate R_c and the communication time. Therefore, the el_{ij} can be represented as

$$el_{ij} = k \cdot R_c \cdot c_{ij} \tag{7}$$

In the above equation, k is the constant parameter. It denotes the energy consumption per Mbps. The overall communication energy of parallel tasks is the sum of communication energy of all edges. Then it can be represented as

$$EC = \sum_{i=1}^{n} \sum_{j=1}^{n} el_{ij} = \sum_{i=1}^{n} \sum_{j=1}^{n} k \cdot R_c \cdot c_{ij} = k \cdot R_c \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \tag{8}$$

4. Energy-Aware Task Duplication

4.1. Parameters Calculation

Communication computation ratio (CCR) is the ratio of sum of communication time to sum of computation time. It can be represented as follows.

$$CCR = \frac{\sum c_{ij}}{\sum t_i} \tag{9}$$

The earliest start time (EST) of a task can be calculated according to the following equation

$$EST = \begin{cases} 0 & i = 1 \\ \max\limits_{1 \leq j \leq n-1, e_{ji} \in E} EST(v_j) + c_{ji} & 1 < i \leq n \end{cases} \tag{10}$$

The earliest complete time (*ECT*) of a task can be calculated by its *EST* and execution time and represented as

$$ECT(v_i) = EST(v_i) + t_i \tag{11}$$

The latest allowable complete time (*LACT*) of a task needs to be calculated in bottom-up manner. The *LACT* of last task of DAG is its *ECT*. The *LACT* of other tasks can be calculated by the *LACT* of its successor task and the communication overhead between them. The calculation equation can be represented as

$$LACT(v_i) = \begin{cases} ECT(v_i) & i = n \\ \min\limits_{1 \le j \le n-1, e_{ji} \in E} LACT(v_j) - c_{ji} & 1 \le i \le n-1 \end{cases} \tag{12}$$

The latest allowable start time (*LAST*) of a task can be calculated by its *LACT* and computation time. It can be represented as

$$LAST(v_i) = LACT(v_i) - t_i \tag{13}$$

The *B-level* of a task can be calculated as

$$B - level = \begin{cases} 0 & if\ v_i\ has\ no\ successor \\ \max\{B - level(v_j) + c_{ij}\} & v_j \in successor(v_i) \end{cases} \tag{14}$$

Correspondingly, the *T-level* of a task can be represented as

$$T - level = \begin{cases} 0 & \text{if } v_i \text{ has no predecessor} \\ \max\{T - level(v_j) + c_{ji}\} & v_j \in \text{predecessor } (v_i) \end{cases} \tag{15}$$

The Favorite Predecessor (*FP*) of a task is defined as the task which has the maximum communication and computation overhead among all predecessor tasks. It can be represented as

$$FP(v_i) = v_j, \text{ where } \forall e_{ji} \in E,\ e_{ki} \in E,\ j \ne k | ECT(v_j) + c_{ji} \ge ECT(v_k) + c_{ki} \tag{16}$$

The Second Favorite Predecessor (*SFP*) of a task is defined as

$$SFP(v_i) = v_j, \text{ where } \forall e_{ji} \in E,\ e_{ki} \in E,\ v_j \ne FP(v_i),\ j \ne k | ECT(v_j) + c_{ji} \ge ECT(v_k) + c_{ki} \tag{17}$$

4.2. Task Duplication

The task scheduling process is shown in Figure 1. Precedence constraints of tasks of DAG must be guaranteed by executing the predecessor tasks before the successor task. So the task sequence needs to be ordered using the concept of *T-level*. Tasks are sorted in ascending order, and then the original task sequence is formed.

Based on the original task sequence, the tasks should be clustered and mapped to different processors. In the clustering process, the linear clustering method is adopted. The tasks with maximum communication are allocated to the same processor. Meanwhile, the task duplications are conducted according to the given condition.

The tasks are sorted in a bottom-up manner in the original task sequence. After the current task is mapped, the predecessor task would be investigated. Since the *FP* is the favorite predecessor task, which has the longest distance between *FP* and the current task, replicating the *FP* can reduce the communication overhead to the greatest extent. *SFP* is defined, which is the second favorite predecessor task. If the *FP* cannot meet the replication condition and the *SFP* meets the condition, the *SFP* would be replicated. Therefore, the communication overhead between *SFP* and the current task would be reduced to zero. The new current task would be checked until the task without the

predecessor task. Then, this round mapping finishes. The next round mapping would start from the first task of the current task sequence. The algorithm description is given in Figure 2.

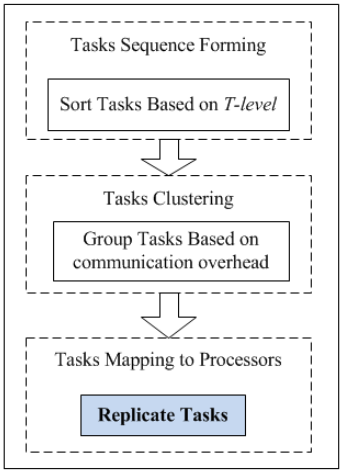


Figure 1. Scheduling flow chart of parallel tasks.

Algorithm Novel Energy Aware Duplication-based Scheduling

Input:
Task list
Output:
Scheduling list, Makespan, Energy consumption

```
1: v = first waiting task of scheduling queue;
2: i = 0;
3: assign v to p_i;
4: while ( Task list is not NULL)
5:    u = FP(v);
6:    z = SFP(v);
7:    if (u has already been assigned) then
8:       if (LAST(v) − LACT(u) < c_uv)&&(EP(u) − EC(e_uv) ≤ Threshold)) then
9:          replicate u to p_i;
10:         v = u;
11:      else
12:         if (z!=NULL && z has not been assigned) then
13:            assign z to p_i;
14:            v = z;
15:         else
16:               if(z!=NULL && z has already been assigned) then
17:                  if (LAST(v) − LACT(z) < c_vz&&(EP(z) − EC(e_vz) ≤ Threshold)) then
18:                     replicate z to p_i;
19:                     v = z ;
20:                  end if
21:               end if
22:         end if
23:      end if
24:   else
25:      assign u to p_i;
26:      v = u;
27:   end if
28:   if (v is the entry of DAG) then
29:      v = the next waiting task of scheduling queue;
30:      i = i + 1;
31:      assign v to p_i;
32:   end If
33: End while
34: Return Scheduling list, Makespan, Energy;
```

Figure 2. Algorithm description.

The threshold is defined as the difference between the computation energy of executing replicated tasks and the saved communication energy. Task duplication is conducted in task clustering and mapping process in a bottom-up manner. When the *FP* cannot meet the condition, *SFP* would be considered.

If the *FP* cannot meet the requirement of duplication, the favorite predecessor task of the *FP* would not be judged even if it can meet the condition. This is because there is no direct correlation between the favorite predecessor task of the *FP* and the current task. In this case, the task duplication only

increases the computation overhead. This is the essential difference between the proposed algorithm and EAD [15].

Task duplication condition includes two following items:

- $LAST(v_i) - LACT(v_j) < c_{ij}$
- $EP(v_i) - EC(e_{ij}) \leq$ threshold

The tasks are retrieved in bottom-up manner. The mapping list forms after the entire assignation has finished. Then, schedule length (makespan) and energy are also calculated based upon the models.

5. Energy Efficiency Evaluation

Simulation experiments are conducted to evaluate the performance and energy saving of the proposed algorithm. The standard task graph set is adopted [27], which includes random generated applications and real applications.

5.1. Test Set and Metrics

The three different parallel applications are used in our simulation. They are represented as a standard task graph, and are shown in Table 2. Among them, a parallel application is generated randomly. The other two are real parallel application benchmarks.

Table 2. Standard task Graph.

Parallel Application	Task Number	Edge Number
fpppp	334	1145
Random	50	164
Robot control	88	131

The evaluation metrics consist of makespan and energy consumption. The makespan is also called schedule length, which is the overall execution time of a parallel application. The makespan is the metric of efficiency. Energy consumption includes the computation energy consumed by processors and the communication energy of network. The task duplication-based algorithm belongs to energy-aware algorithms. To measure our algorithm more comprehensively, the two metrics are evaluated respectively. The parameters used in calculating energy are obtained from the Lenovo sure-fire servers R510 (Lenovo Inc., Beijing, China).

5.2. Baseline Algorithms

The baseline algorithms include Task Duplication Based Scheduling (TDS) [4] and Energy Aware Task Duplication Based Scheduling (EADS) [15].

- TDS: The goal of TDS is to improve performance, which attempts to generate the shortest schedule length through task duplication. All tasks that can shorten schedule length would be replicated. The same task may be executed on different processors.
- EADS: EADS is based on TDS and aims to reduce the energy consumption due to replicating tasks. The FP of the current task is judged based upon the replication condition in the bottom-up manner. Some indirect tasks may be duplicated in the same tasks.

5.3. Impact of CCR

CCR can measure the total time spent on computation and communication. To investigate the impact of *CCR*, we conduct the experiments using the random application, when *CCR* varies from 0.1 to 10. The computation energy and communication energy comparison results are shown respectively in Figure 3a,b.

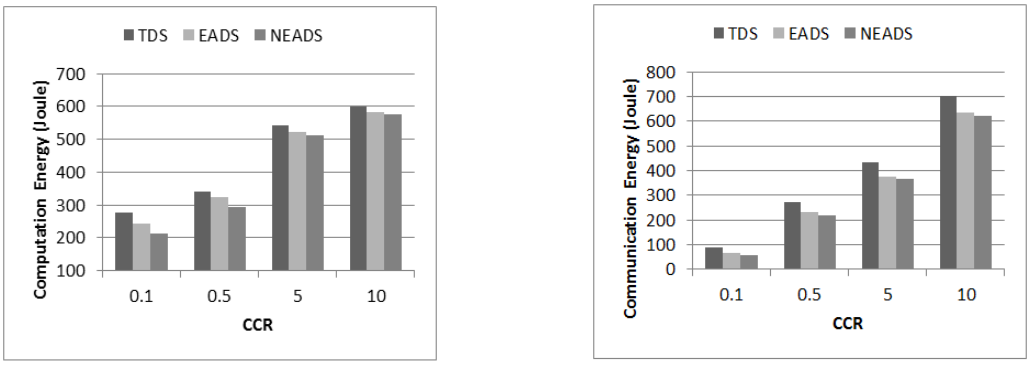


Figure 3. Energy comparison in different communication computation ratios (CCR). (**a**) Computational Energy comparison and (**b**) Communication Energy comparison.

The proposed algorithm NEADS can reduce the energy to some extent compared with two baseline algorithms. When *CCR* is less than 1, the computation energy has been significantly reduced. However, when *CCR* is greater than 1, the reduced computation energy is not obvious. Because NEADS focuses on diminishing the extra computation overheads of the indirect predecessor tasks, its advantage of computation energy reduction is more obvious than EADS. The maximum computation energy reduction over TDS and EADS are 22.5% and 12.7% when *CCR* is set to 0.1, respectively.

The communication energy has been markedly reduced compared with TDS. But NEADS has not great advantages when compared with EADS. Since both EADS and NEADS are based on the task duplication, both of them aim to reduce the communication energy. So NEADS can only achieve small communication energy reduction compared with EADS.

5.4. Impact of Applications

According to the ratio of computation and communication, the features of parallel applications can be classified into computation intensive, communication intensive and synthetic. In this section, the impact of application features on makespan and energy has been investigated. Three applications (i.e., fpppp, Robot control and random) have different features.

Figure 4 shows the comparison results in fpppp. We can see that the makespans in different algorithms are almost equal. However, the energy consumption has been obviously reduced in NEADS compared with other two algorithms. Fpppp belongs to computation intensive applications. Therefore, the proposed algorithm has advantage in reducing energy over other two algorithms for computation intensive applications without increasing the makespan.

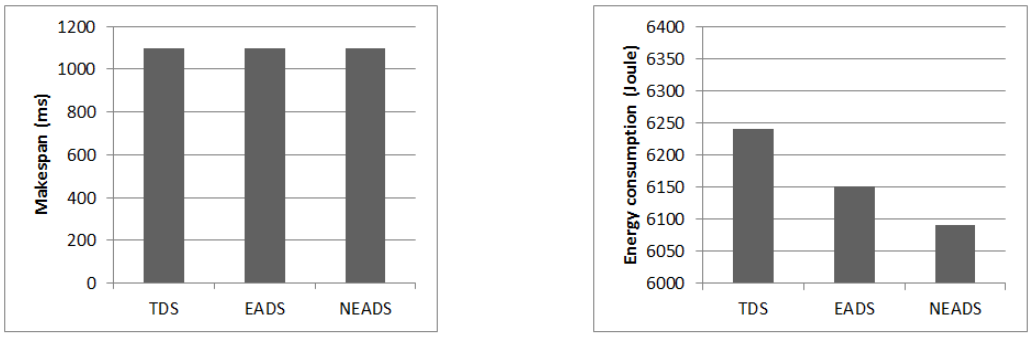
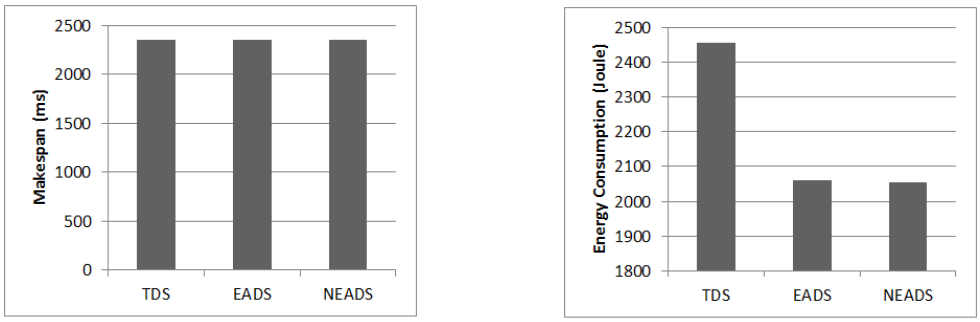


Figure 4. Makespan and energy comparison of fpppp. (**a**) Makespan comparison and (**b**) Energy consumption comparison.

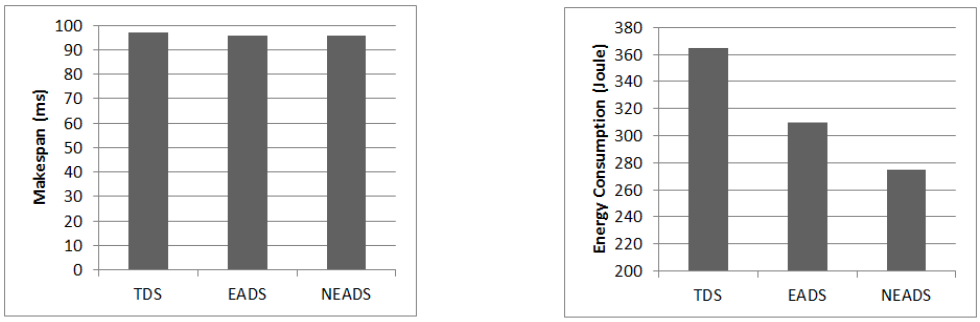
Figure 5 shows the comparison results in the robot control. Both makespan and energy have no obvious differences. Energy consumption has been slightly lowered. This shows that, for communication-intensive applications, the proposed algorithm has no distinct advantages.



(**a**) (**b**)

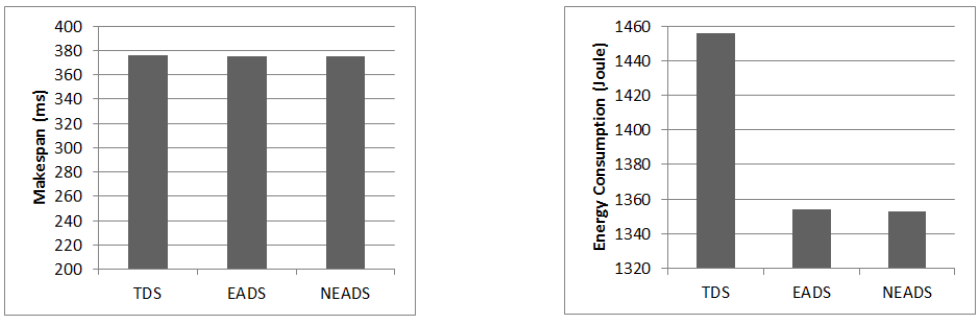
Figure 5. Makespan and energy comparison of Robot. (**a**) Makespan comparison and (**b**) Energy consumption comparison.

Random is a synthetic application. In order to indicate its feature, makespan and energy are calculated when *CCR* is defined as 0.1 and 5 respectively. The comparison results are shown in Figures 6 and 7. When *CCR* is 0.1, NEADS can dramatically reduce the energy consumption compared with TDS and EADS. However, when *CCR* is 5, NEADS has a close value in energy consumption with EADS. Since *CCR* can indicate its features, these results are consistent with those in different applications such as fpppp and robot control.



(**a**) (**b**)

Figure 6. Makespan and energy comparison of Random *CCR* = 0.1. (**a**) Makespan comparison and (**b**) Energy consumption comparison.



(**a**) (**b**)

Figure 7. Makespan and energy comparison of Random *CCR* = 5. **(a)** Makespan comparison and (**b**) Energy consumption comparison.

5.5. Overall Energy Efficiency

To illustrate overall energy efficiency of the proposed algorithm in synthetic applications, we conducted the comprehensive tests on multiple synthetic applications. Figures 8 and 9 show the reduced energy percent in 30 random applications compared with TDS and EADS. The overall energy has been reduced by 9.39% and 3.52% on average, respectively.

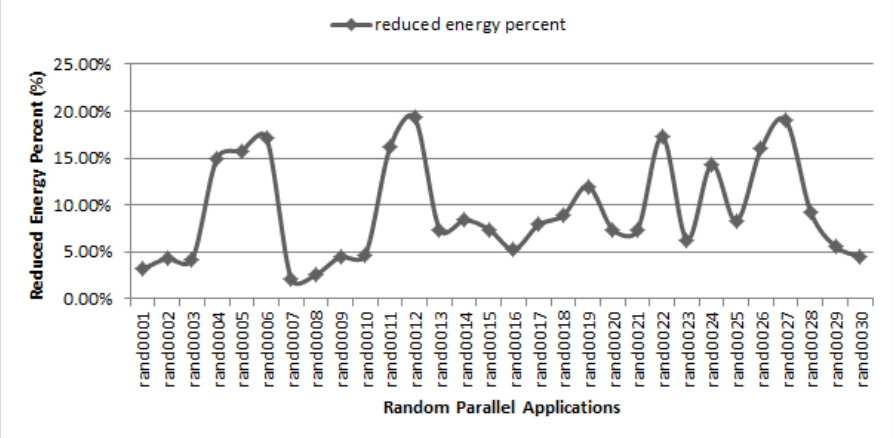


Figure 8. Reduced energy in different random applications compared with TDS.

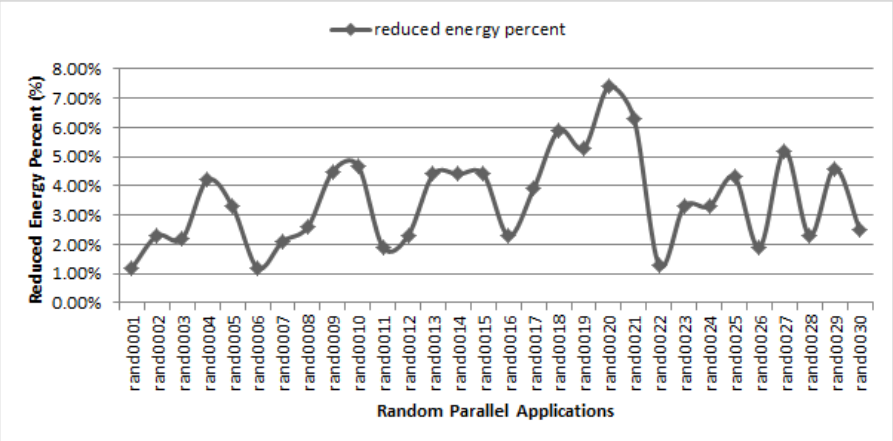


Figure 9. Reduced energy in different random applications compared with EADS.

6. Conclusions

In this paper, we proposed a novel duplication-based energy-aware scheduling algorithm. In the duplication process, both the favorite predecessor and second favorite predecessor are considered. When the favorite predecessor (*FP*) does not meet the duplication condition, the predecessor of *FP* would not be checked. Then, the second favorite predecessor would be judged. This avoids indirect tasks being replicated to the same processor. Correspondingly, extra communication and computation overhead can be reduced. To demonstrate the effectiveness of the proposed algorithm, we conducted extensive experiments under various applications. The experimental results show that, for synthetic applications, NEADS can effectively reduce the energy consumption.

Acknowledgments: The work was supported by Beijing Educational Committee Science and Technology Develop Project under Grant No. KM201511417003, the National Natural Science Foundation of China under Grant No. 61502036, Supporting Program of Sci & Tech Research of China (2014BAK08B02), The Importation and Development of High-Caliber Talents Project of Beijing Municipal Institutions under Grant No. CIT&TCD201404093.

Author Contributions: Aihua Liang and Yu Pang conceived and designed the study. Yu Pang performed the experiments. Aihua Liang wrote and revised the paper. All authors have given approval to the final version of the manuscript.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Chedid, W.; Yu, C. *Survey on Power Management Techniques for Energy Efficient Computer Systems*; Technical Report; Department of Electrical and Computer Engineering, Cleveland State University: Cleveland, OH, USA, 2002.
2. TOP500 Team: The TOP500 List. Available online: http://www.top500.org/lists/2015/11 (accessed on 15 January 2016).
3. Dally, W.J.; Carvey, P.P.; Dennison, L.R. The Avici Terabit Switch/Router. In Proceedings of the 6th Symposium on Hot Interconnects, Stanford, CA, USA, 13–15 August 1998; pp. 41–55.
4. Ranaweera, S.; Agrawal, D.P. A Task Duplication Based Scheduling Algorithm for Heterogeneous Systems. In Proceedings of the Parallel and Distributed Processing Symposium, Orlando, FL, USA, 29 May–2 June 2000; pp. 445–450.
5. Bansal, S.; Kumar, P.; Singh, K. An Improved Duplication Strategy for Scheduling Precedence Constrained Graphs in Multiprocessor Systems. *IEEE Trans. Parallel Distrib. Syst.* **2003**, *14*, 533–544. [CrossRef]
6. Liang, A.; Xiao, L.; Ruan, L. Adaptive workload driven dynamic power management for high performance computing clusters. *Comput. Electr. Eng.* **2013**, *39*, 2357–2368. [CrossRef]
7. Wang, L.; Laszewski, G.; Von Dayal, J. Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS. In Proceedings of the 10th IEEE/ACM International Conference Symposium on Cluster, Cloud and Grid Computing (CCGrid), Melbourne, Australia, 17–20 May 2010; pp. 368–377.
8. Saravanan, V.; Anpalagan, A.; Kothari, D.P. A comparative simulation study on the power performance of multi-core architecture. *J. Supercomput.* **2014**, *70*, 465–487. [CrossRef]
9. Mishraa, A.; Tripathib, A. Energy efficient voltage scheduling for multi-core processors with software controlled dynamic voltage scaling. *Appl. Math. Model.* **2014**, *38*, 3456–3466. [CrossRef]
10. Zhu, X.; He, C.; Li, K.; Qin, X. Adaptive energy-efficient scheduling for real-time tasks on DVS-enabled heterogeneous clusters. *J. Parallel Distrib. Comput.* **2012**, *72*, 751–763. [CrossRef]
11. Sridharan, R.; Gupta, N.; Mahapatra, R. Feedback-controlled reliability-aware power management for real-time embedded systems. In Proceedings of the 45th Annual Design Automation Conference, Anaheim, CA, USA, 08–13 June 2008; pp. 185–190.
12. Marzollaa, M.; Mirandolab, R. Dynamic power management for QoS-aware applications. *Sustain. Comput. Inform. Syst.* **2013**, *3*, 231–248. [CrossRef]
13. Falzon, G.; Li, M. Enhancing list scheduling heuristics for dependent job scheduling in grid computing environments. *J. Supercomput.* **2012**, *59*, 104–130. [CrossRef]
14. Liu, W.; Duan, Y.; Du, W. An energy efficient clustering-based scheduling algorithm for parallel tasks on homogeneous DVS-enabled clusters. In Proceedings of the 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Wuhan, China, 23–25 May 2012; pp. 575–582.
15. Tang, X.; Li, K.; Liao, G.; Li, R. List scheduling with duplication for heterogeneous computing systems. *J. Parallel Distrib.* **2010**, *70*, 323–329. [CrossRef]
16. Lai, K.-C.; Yang, C.-T. A dominant predecessor duplication scheduling algorithm for heterogeneous systems. *J. Supercomput.* **2007**, *44*, 126–145. [CrossRef]
17. Sanjeev, B.; Rabab, A. Energy aware DAG scheduling on heterogeneous systems. *Cluster Comput.* **2010**, *13*, 373–383.
18. Sinnen, O.; Sousa, L.A. Communication contention in task scheduling. *IEEE Trans. Parallel Distrib. Syst.* **2005**, *16*, 503–515. [CrossRef]
19. Sinnen, O.; To, A.; Kaur, M. Contention-aware scheduling with task duplication. *J. Parallel Distrib. Comput.* **2011**, *71*, 77–86. [CrossRef]
20. Choi, G.S.; Kim, J.; Ersoz, D.; Yoo, A.; Das, C. Coscheduling in clusters: Is it a viable alternative? In Proceedings of the ACM/IEEE Conference Supercomputing (SC '04), Pittsburgh, PA, USA, 6–12 November 2004.

21. Maosko, L.; Tudruj, M.; Mounie, G. Comparison of program task scheduling algorithms for dynamic SMP clusters with communication on the fly. In Proceedings of the Parallel Processing and Applied Mathematics, Wroclaw, Poland, 13–16 September 2009; pp. 31–41.
22. Shojafar, M.; Cordeschi, N.; Amendola, D.; Baccarelli, E. Energy-Saving Adaptive Computing and Traffic Engineering for Real-Time-Service Data Centers. In Proceedings of the 22nd IEEE ICC WORKSHOP, London, UK, 8–12 June 2015.
23. Shojafar, M.; Cordeschi, N.; Baccarelli, E. Energy-efficient Adaptive Resource Management for Real-time Vehicular Cloud Services. *IEEE Trans. Cloud Comput.* **2016**, *99*, 1–14. [CrossRef]
24. Shojafar, M.; Javanmardi, S.; Abolfazli, S.; Cordeschi, N. Erratum to: FUGE: A joint meta-heuristic approach to cloud jobs cheduling algorithm using fuzzy theory and a genetic method. *Cluster Comput.* **2015**, *18*, 845. [CrossRef]
25. Zong, Z.; Mais, N.; Adam, M. Energy efficient scheduling for parallel applications on mobile clusters. *Cluster Comput.* **2008**, *11*, 91–113. [CrossRef]
26. Zong, Z.; Adam, M.; Ruan, X.; Qin, X. EAD and PEBD: Two Energy-Aware Duplication Scheduling Algorithms for Parallel Tasks on Homogeneous Clusters. *IEEE Trans. Comput.* **2011**, *60*, 360–374. [CrossRef]
27. Standard Task Graph Set. Available online: http://www.kasahara.elec.waseda.ac.jp/schedule (accessed on 10 October 2015).