International Journal of Grid Distribution Computing
Vol.7, No.6 (2014), pp.19-28
http://dx.doi.org/10.14257/ijgdc.2014.7.6.03

Incorporating Energy-Aware Mechanism into Workflow Scheduling Policy in Heterogeneous Distributed Systems

Hong He and Peng Xiao

Department of Computer and Communication, Hunan Institute of Engineering
hhong1970@126.com

Abstract

In the past decades, plenty scheduling policies have been proposed for improving the execution performance of workflow applications. However, few of them have addressed the issue of energy conservation. In this paper, a novel scheduling metric, namely Minimal Energy Consumption Path, is introduced to reducing the energy consumption when scheduling large-scale workflows. The proposed scheduling metric can be incorporated into current schedulers so as to enhance their capability of energy conservation as well as maintain their scheduling performance. A lot of experiments are performed by using different scheduler and experimental settings to evaluate the proposed scheduling metric and the results indicate it is effective for reducing the storage-related energy consumption, especially when the target workflows are data-intensive.

Keywords: *Energy Efficiency; Workflow; Heuristic Metric; Quality of Service*

1. Introduction

Traditionally, the primary performance goal of distributed systems has focused on reducing the execution time of applications with increasing complexity [1, 2]. This performance goal has been mostly achieved by the development of high-density datacenters. Recently, these systems provide very powerful processing capability and capacity. They often consist of tens or hundreds of thousands of processors and other resource-hungry devices. The energy consumption of these systems has become a major concern [3-5].

Meanwhile, the workflow model has become the most attractive paradigm for executing scientific applications in the past years [6, 7]. As it provides a structured means to representing and managing applications developed by scientific collaborations, scientific workflow has been embraced by domain scientists as a means of describing complex applications and managing their execution in distributed computing systems. Recently, workflows are being applied in a variety of scientific disciplines, such as astronomy, biology, climate model, earthquake sciences, physics, and many others.

In order to support workflows in distributed systems, plenty efforts are taken to develop integrated environments for the workflow programming, composition, scheduling, execution, performance monitoring, bottleneck analysis and *etc.*, [8-10]. As a result, it becomes an easy work for deploying some complex workflow applications on a distributed platform. On the other side, the execution performance of workflow applications still far from desirable when taking the user's QoS requirements into account. Among various kinds of QoS requirements, energy-efficiency has become the most mentioned one, due to the rapidly increasing of energy consumption in high-performance systems. In the past few years, energy conserving techniques have been widely studies and adopted in high-performance heterogeneous systems, such as DVFS [11], workload consolidation [12], green

ISSN: 2005-4262 IJGDC

network [13]. Unfortunately, few of efforts have been taken to optimize the efficiency consumption of deploying and executing large-scale workflows.

To address this problem, we present a novel heuristic called 'minimal energy consumption path', which defines a uniform energy consumption model for both computing and storage tasks. Based on this heuristic, we extend two classic workflow scheduling (HEFT and CPOP) and design two energy-efficiency scheduling algorithms for large-scale workflows. Although we only implement two algorithms, our heuristic can also be applied to other existing scheduling algorithms so as to incorporate energy-aware mechanism into these algorithms.

The rest of this paper is organized as following. In Section 2, we summarize the related work. In Section 3, we present the problem description. In Section 4, energy models are presented. In Section 5, extensive experiments are conducted and the results are carefully investigated and evaluated in terms of various metrics. Finally, Section 6 concludes the paper with a brief discussion of future work.

2. Related Work

Due to the NP-hard nature of the task scheduling problem in general cases, heuristics are the most popular scheduling model adopted by many researchers, and they deliver good solutions in less than polynomial time. For example, the HEFT algorithm [14] is a well-known heuristic scheduler with quite desirable scheduling performance. The DBUS algorithm [15] is a duplication-based scheduling heuristic that performs a CP-based listing for tasks and schedules them with task duplication and insertion. Even so, many studies have indicated that some special scheduling problems can be solved in polynomial time under certain conditions. For example, Benoit et al. proposed an algorithm with polynomial complexity for scheduling multiple pipeline workflows with constraint to energy consumption [16].

In addition, many researchers have designed various kinds of scheduling policies for improving the execution performance of workflow applications. For instance, Yuan et al. proposed a DET algorithm which tries to distribute the application's deadline among subtasks by assigning a time window to each subtask [17]. To achieve this goal, DET algorithm first applies dynamic programming technique to assign time windows to critical subtasks, and then iteratively search a suitable time window for those non-critical subtasks. In [18], the authors proposed two scheduling algorithms for different performance measurements: makespan and budget. In the first algorithm, they firstly schedule workflows with minimum makespan, and then refine the schedule until its budget constraint is meet. In the second one, they initially assign each task to its cheapest resource, and then refine the schedule to shorten the execution time under budget constraint. In [19], the authors proposed a Reinforcement Learning (RL) algorithm, which uses RL technique to help scheduler co-allocating multiple resource concurrently. In [20], the authors proposed a bi-criteria scheduling algorithm that follows a different approach to the optimization problem of two arbitrary independent criteria. In [21], the authors use Integer Programming technology to scheduling scientific workflows that have multiple optimal objectives. To transform the multi-objective problem to a single-objective one, this approach assigns a weight to each QoS parameter and the algorithm tries to optimize the weighted sum of the QoS parameters. In these years, game theory has been widely used for solving multi-objective optimization problems, such as workflow scheduling. For instance, in [22], the authors proposed two algorithms based on Game Theory for the scheduling of n independent workflows. The first one called Game-quick, tries to minimize the overall makespan of all workflows. The second algorithm called Game-cost, tries to minimize the overall cost of all workflows, while meeting each workflow's deadline. The main advantage of using

game theory is that it can quickly obtain equivalent solutions rather than searching the whole solution space.

3. Problem Formulation

Typically, a workflow application is represented by a directed acyclic graph (DAG), where the activities represent individual tasks, and the edges represent the precedence-dependencies between these tasks. When running a workflow, input data of an activity node is transferred from an independent storage node to the execution node, and output data is transferred back to the original storage node or others. For the convenience of representation in the following sections, we firstly give the related definitions in this section.

A workflow is noted as a directed acyclic graph $\boldsymbol{G}=<\boldsymbol{V},\boldsymbol{W}>$, where $\boldsymbol{V}$ is the set of activities representing the computing tasks of the workflow. Each activity V_i is represented as $<c_i, d_i^{in}, d_i^{out}>$, where c_i is the size of computing task, d_i^{in} is the size of input data that required by V_i, d_i^{out} is the size of output data that generated by V_i. A workflow scheduling scheme is noted as $\boldsymbol{M}:\boldsymbol{V}\times\boldsymbol{C}\times\boldsymbol{S}\rightarrow\{0,1\}$, where $\boldsymbol{C}$ is the set of physical computing nodes, $\boldsymbol{S}$ is the set of physical storage nodes. Element $M_{i,i',i''}$ indicates that activity V_i is scheduled on computing node $C_{i'}$ and its storage device comes from $S_{i''}$.

In this work, we mainly concentrate on data-intensive workflows, which have some significant differences from those computation-intensive workflows. For example, the intermediate date generated by a computation-intensive workflow is very small and therefore can be easily stored in the computing node's memory or local disk. While a data-intensive workflow will generate a large volume of intermediate data, which often requires being stored in independent storage nodes. An example of workflow scheduling scheme can be illustrated as Figure 1.

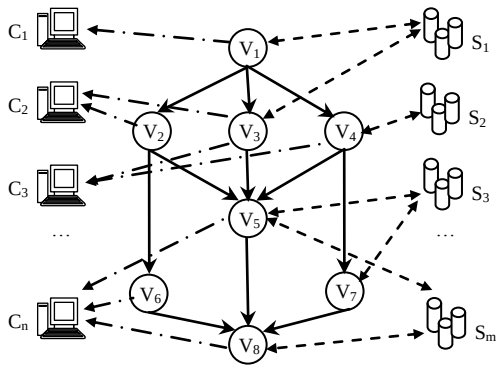


Figure 1. An Example of Scheduling Scheme

For a given scheduling scheme $\boldsymbol{M}$, the total energy consumption of completing the target workflow $\boldsymbol{G}$ is noted as $E(\boldsymbol{G}, \boldsymbol{M})$. Based on the above definitions, the problem of scheduling workflow for optimal energy consumption can be generally formulized as following:

$$\begin{aligned} &\min\ E(\boldsymbol{G},\boldsymbol{M}) \\ &s.t.\ \boldsymbol{M}:\boldsymbol{V}\times\boldsymbol{C}\times\boldsymbol{S}\rightarrow\{0,1\} \end{aligned} \tag{1}$$

It is clear that solving problem of $\min\{E(\boldsymbol{G},\boldsymbol{M})\}$ is a classic NP-complete, since the solution space of $\boldsymbol{M}$ is 2^{n+m+k}. Therefore, heuristic algorithm seems to be necessary for obtaining a suboptimal solution. Before proposing any heuristics, we firstly need to figure out the approach to modeling the energy consumption of a

data-intensive workflow under a given scheduling scheme, that is the formulation for calculating $E(\boldsymbol{G},\boldsymbol{M})$.

4. Energy-aware Heuristic Metric

4.1. Energy Consumption Model of Task Execution

The power consumption of machine consists of static part P_{static} and dynamic part P_{dynamic}. P_{static} is the fixed power consumption for keeping the machine in working state even there is no workload on it, while P_{dynamic} is related with the dynamic utilization of power-consuming components. Typically, the power model of a physical machine is formulated as

$$P(t) = P_{static} + \sum_{j \in \{cpu,mem,disk\}} P_j(t) \tag{2}$$

where $P_j(t)$ is the dynamic power consumption of component j.

Given a scheduling scheme $\boldsymbol{M}$, according to definitions in Section 3, $\boldsymbol{M}$ can be noted as $\{M_{i,i',i''} \mid i \in (1,\ldots,n), i' \in (1,\ldots,m), i'' \in (1,\ldots,k)\}$, where i is the index of an activity in the workflow, i' is the index of physical computing node, i'' is the index of physical storage node. Therefore, we can note the energy consumption of completing the activity V_i as $E(V_i, M_{i,i',i''})$.

As shown in Figure 1, part of $E(V_i, M_{i,i',i''})$ is spent on processor and memory when running the V_i's computing task, and the other part is the energy consumption of disk which is spend on data accessing. In the following, we note them as $E_c(V_i, M_{i,i',i''})$ and $E_d(V_i, M_{i,i',i''})$, respectively. When an activity node $V_i = <c_i, d_i^{in}, d_i^{out}>$ is assigned, its execution time of computing task can be noted as following

$$\mathrm{D}T_{exec} = \frac{c_i}{r_{cpu}^j \times f} \tag{3}$$

By the power model in Eq.(1), $E_c(V_i, M_{i,i',i''})$ can be measured as

$$E_c(V_i, M_{i,i',i''}) = P_j(t) \times \frac{c_i}{r_{cpu}^j \times f} \tag{4}$$

where f is the working frequency of the CPU. As the disk is come from external storage system, the disk component in Eq.(1) will be automatically ignored.

To calculate the energy consumption spent on data accessing, we must take into account the location of the storage node as well as the structure of the workflow. Based on the illustration in Figure 1, we can have the formulation of $E_d(V_i, M_{i,i',i''})$ as following.

$$E_d(V_i, M_{i,i',i''}) = \sum_{j \in Pred(v_i)} \left(P_{disk}^{j''}(t) \frac{d_j^{in}}{B_{i',j''}} \right) + \sum_{k \in Succ(v_i)} \left(P_{disk}^{i''}(t) \frac{d_i^{out}}{B_{i',i''}} \right) \tag{5}$$

where $P_{disk}^k(t)$ is the power model of storage node k, $B_{i,j}$ is the bandwidth between storage node i and computing node j, $d_{j \to i}^{in}$ is the input data from V_j to V_i, $d_{i \to k}^{out}$ is the output data from V_i to V_k, $Pred(V_i)$ and $Succ(V_i)$ are the predecessors and successors of V_i respectively.

It is clear that the first part of Eq.(5) is the energy consumption spent on obtaining input data before running V_i, and second part is the energy consumption spent on storing output data after finishing V_i. Combining Eq.(4) and Eq.(5), we can obtain the total energy consumption under a given scheduling scheme, which is shown as

$$E(\boldsymbol{G},\boldsymbol{M}) = \sum_{i=1}^{n} \left[E_c(V_i, M_{i,i',i''}) + E_d(V_i, M_{i,i',i''})\right] \quad (6)$$

4.2. Energy-aware Scheduling Policy

In this work we propose a novel heuristic, namely *Minimized Energy Consumption in Deployment and Scheduling* (MECDS), for data-intensive workflows in high-performance systems. The MECDS mainly consists of two phases: workflow deployment and workflow scheduling.

In the phase of workflow deployment, we select a storage node which can be allocated to a workflow instance with aiming to obtain minimal data accessing energy consumption for the current activities. To do this, we introduce a novel conception, called *Minimal Energy Consumption Path* (*MECP*), which is defined as the minimal total energy consumption from V_{init} to the current activity shown as following

$$MECP(V_i) = \begin{cases} E_d(V_i, M_{i,i',i''}), & if\ V_i = V_{init} \\ E_d(V_i, M_{i,i',i''}) + \min\limits_{V_j \in Pred(V_i)} \{MECP(V_j)\} \end{cases} \quad (7)$$

According to the definition of *MECP*, if a storage node $S_{i''}$ can satisfying $\min\{MECP(V_i)\}$ among all storage nodes, then the activity V_i should use it as the storage node. If a task uses $S_{i''}$ as the underlying storage node has already been created and deployed, then we can go on; otherwise a new workflow instance should be created and deployed, which uses $S_{i''}$ as underlying storage node. By repeating the above, we can complete the deployment.

In the phase of workflow scheduling, we can still uses the priorities that defined in existing algorithm such as b-level or t-level. However, we will incorporate the MECP heurist into them if energy-consumption has been taken into account. For example, the t-level or b-level can be redefined as following:

$$rank_t(V_i^*, S_k) = \frac{\max\limits_{V_j \in Prev(V_i^*)} \left\{ rank_t(V_i^*, S_j) + \frac{D_j^{out}}{B_{j,k}} + \frac{W_{j,i}}{B_{j,k}} \right\}}{MECP(V_i^*)} \quad (8)$$

$$rank_b(V_i^*, S_k) = \frac{\max\limits_{V_j \in Succ(V_i^*)} \left\{ rank_b(V_i^*, S_j) + \frac{D_j^{out}}{B_{j,k}} + \frac{W_{j,i}}{B_{j,k}} \right\}}{MECP(V_i^*)} \quad (9)$$

Based on the above scheduling models and definitions of b-level and t-level, we extend the classical HEFT and CPOP as HEFT-MECP and CPOP-MECP. The detailed implementations of the heuristic algorithms are shown as following.

HEFT-MECP Algorithm

Begin

1. Change $G = <V, W>$ into $G^* = <<V,V^*>, W>$ by insert $\{V_1^*, V_2^*, \ldots, V_{n-1}^*\}$ into the original DAG;
2. Compute $rank_b$ for all tasks by traversing G^* upward, starting from the exit task;
3. Sort the tasks in a scheduling list by non-increasing order of $rank_b$;
4. **while** there are unscheduled tasks in the list **do**
5. Select the first task V_i from the list for scheduling;
6. **if** $V_i \in \{V_1^*, V_2^*, \ldots, V_{n-1}^*\}$ **then**
7. **for each** S_j in $\{S_1, S_2, \ldots, S_M\}$ **do**

8. Compute $rank_t(V_i, S_j)$;
9. **end for**
10. Assign V_i to S_j that maximize $rank_b + rank_t$;
11. **else**
12. **for** each C_j in $\{C_1, C_2, \ldots, C_N\}$ **do**
13. Compute $EFT(M_{i,k,j})$;
14. **end for**
15. Assign task V_i to C_k that minimizes EFT of V_i by insert scheduling policy;
16. **end if**
17. **end while**

End.

CPOP-MECP Algorithm

Begin

1. Change $G = <V, W>$ into $G^* = <<V, V^*>, W>$ by insert $\{V_1^*, V_2^*, \ldots, V_{n-1}^*\}$ into the original DAG;
2. Compute $rank_b$ for all tasks by traversing G^* upward, starting from the exit task;
3. Compute $rank_t$ for all tasks by traversing G^* forward, starting from the init task;
4. Obtain the critical path of G^* and store into L_{cp};
5. Find C_k and S_j which minimize $\sum_{V_i \in L_{cp}} E(M_{i,k,j})$;
6. Assign all $V_i \in L_{cp}$ and $V_i \in \{V_1, \ldots, V_n\}$ to C_k;
7. Assign all $V_i \in L_{cp}$ and $V_i \in \{V_1^*, \ldots, V_{n-1}^*\}$ to S_k;
8. **while** there is unscheduled task in list **do**
9. Select V_i with $\max\{rank_b\}$.
10. **if** $V_i \in \{V_1^*, V_2^*, \ldots, V_{n-1}^*\}$ **then**
11. Assign V_i to S_k which maximize $rank_b + rank_t$;
12. **else**
13. Assign task V_i to C_k that minimizes the EFT by insert scheduling policy;
14. **end if**
15. Update the $rank_b$ of left tasks in unscheduled list;
16. **end while**

End.

5. Experiments and Performance Comparison

5.1. Experimental Settings

The experiments are conducted on the cluster platform that deployed in our HP high-performance Network Center. The platform consists of 20 computing nodes (CN_1~CN_{20}) and 7 storage nodes (SN_1~SN_7) as underlying physical resources. To take into account the heterogeneity, we adopts various kinds of equipments that made by different vendors. To minimize the interference when measuring energy consumption, we shutdown all the displays and set the fans and local disks of computing nodes in constant power mode. In experiments, we mainly focus on the effects of data-intensive characteristic on workflow energy consumption and execution performance (makespan). The flowchart of the target workflow in the simulations is shown in Figure 2, which has two features: firstly, it has massive paralleling subtasks which generate a large volume of intermediate data; secondly, its level-structure is quiet distinguishing, which enables us to exploit the potential of a cloud system as well as finding out its performance bottleneck. Such a workflow paradigm can be seen in many data-intensive applications, such as MapReduce applications and parameter sweep applications.

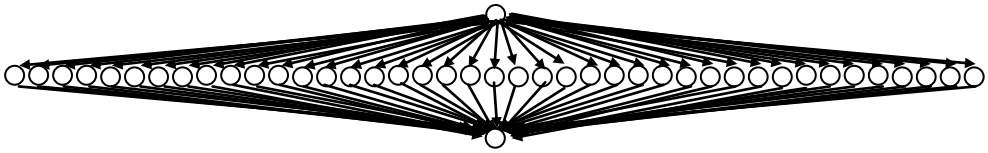
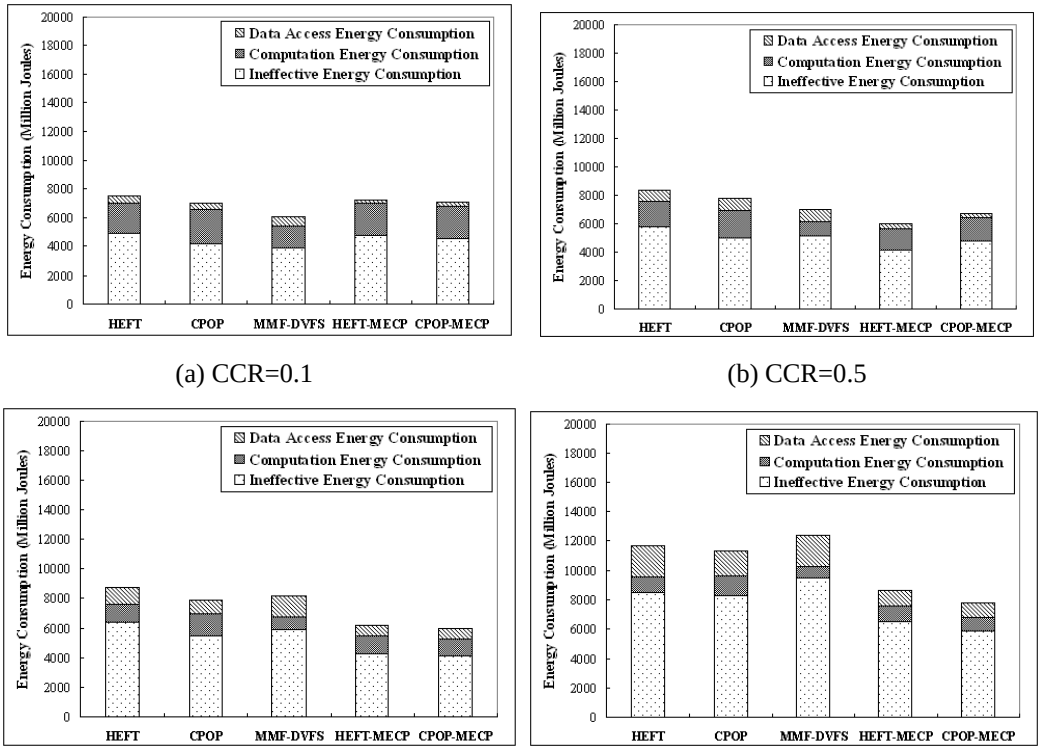


Figure 2. Flowchart of Simulative Workflow

5.2. Comparison of Performance and Analysis

Besides the HEFT and CPOP, we also use the MMF-DVFS algorithm [23] for performance comparison. The experimental results are shown in Figure 3 and Figure 4.



(a) CCR=0.1 (b) CCR=0.5

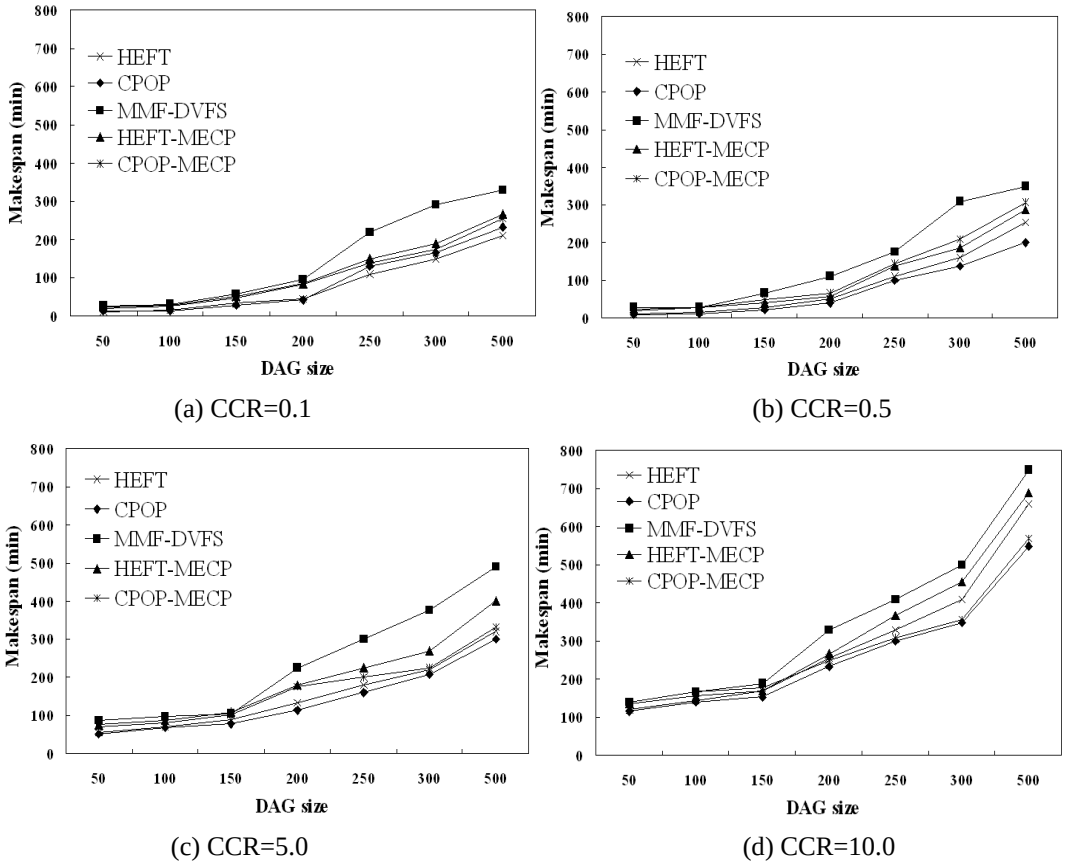
(c) CCR=5.0 (d) CCR=10.0

Figure 3. Comparison of Energy Consumption

As shown in Figure 3, when CCR is in low level, we notice that processors contribute most to the total energy consumption regardless of the used scheduling algorithm. In this case, HEFT performs worst among all the five algorithms, while MMF-DVFS obtains about 17% energy saving comparing with the former. The reason is that MMF-DVFS is specially designed for computation-intensive applications, which applies DVFS mechanism to reduce the processor energy consumption. The difference between MMF-DVFS and HEFT and CPOP is that MMF-DVFS seem more unstable than the other two, since the deviation of its CPU energy consumption is very high. The reason is that MMF-DVFS uses coarse-grained DVFS adjusting strategy which considers energy consumption minimization as the only objective.

When the CCR value is in moderate levels, the CPU energy consumptions of all algorithms are reduced at first and then increased. Taking CPOP algorithm as an example, its CPU energy consumption reduces about 33% when CCR increases from

0.1 to 1.0, and then increases about 24% when we further increases CCR to 2.0. To explain this, we show the mean makespan of all experiments in Figure 4.



(a) CCR=0.1 (b) CCR=0.5

(c) CCR=5.0 (d) CCR=10.0

Figure 4. Comparison of Makespan

It is clear that the mean makespan has the same trend just like energy consumption. When adjusting CCR in low level, we reduce the size of computing tasks so as to increase the CCR value. While the CCR is in moderate level, we increase the size of input/output data to increase the CCR value. Therefore, we can see that makespan is reduced at first and then increases, which in turn results in corresponding changing of energy consumption.

According to the results shown in Figure 3 and Figure 4, CCR=2.0 seems to be the turning point for both makespan and energy consumption. When CCR value is bigger than 2.0, the disk energy consumption dramatically increases. Therefore, we can know that the efficiency of storage nodes have become the performance bottleneck when CCR increases to a high level, not only for the energy consumption but also for the makespan. In this situation, MECP-base heuristics significantly outperforms other algorithms in terms of both metrics.

Although DVFS mechanism is effective for saving processor energy consumption, its energy savings still can not compensate the energy wastage caused by prolonged execution time. More specifically, many processors still consume lots of energy when they are waiting for the completion of massive data accessing, even their working frequency has been switched to lowest level. For example, the CPU energy consumption of MMF-DVFS increases about 34% when the CCR is increased from 2.0 to 10.0. We can easily guess that a great deal of CPU energy is wasted on I/O waiting. As to our MECP heuristic, we notice that its performance also depends on the CCR value. However, it is less sensitive than other heuristics. That is because

MECP has incorporated energy consumption into priorities of tasks. Therefore, when the energy consumption have been measured, it is effective to minimize the energy consumption as well makespan.

6. Conclusion

To address the issue of energy conservation in distributed environments, a novel scheduling metric is proposed in this paper. By incorporating this metric into current schedulers, we can obtain some energy-efficient schedulers for large-scale workflow applications. The main contribution of this study is that the proposed approach can be applied into many current schedulers. In this way, it is easy to re-design the existing scheduling policies with energy-efficiency enhanced feature. Currently, we only incorporate our metric into two typical heuristics. In the future, we are planning to use it into some other popular workflow schedulers and evaluate their performance as well as effectiveness.

References

[1] Z. C. Papazachos and H. D. Karatza, "Performance evaluation of bag of gangs scheduling in a heterogeneous distributed system", Journal of Systems and Software, vol. 83, no. 8, **(2010)**, pp. 1346-1354.
[2] B. F. Cornea and J. Bourgeois, "A framework for efficient performance prediction of distributed applications in heterogeneous systems", Journal of Supercomputing, vol. 62, no. 3, **(2012)**, pp. 1609-1634.
[3] J. Kim, W. Huang and S. Maddineni, "Energy landscape analysis for regulatory RNA finding using scalable distributed cyberinfrastructure", Concurrency and Computation-Practice & Experience, vol. 23, no. 17, **(2011)**, pp. 2292-2304.
[4] Y. C. Lee and A. Y. Zomaya, "Energy Conscious Scheduling for Distributed Computing Systems under Different Operating Conditions", IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 8, **(2011)**, pp. 1374-1381.
[5] S. U. Khan, P. Bouvry and T. Engel, "Energy-efficient high-performance parallel and distributed computing", Journal of Supercomputing, vol. 60, no. 2, **(2012)**, pp. 163-164.
[6] W. Dou, J. L. Zhao and S. Fan, "A collaborative scheduling approach for service-driven scientific workflow execution", Journal of Computer and System Sciences, vol. 76, no. 6, **(2010)**, pp. 416-427.
[7] S. Abrishami, M. Naghibzadeh and D. H. J. Epema, "Cost-Driven Scheduling of Grid Workflows Using Partial Critical Paths", IEEE TPDS, vol. 23, no. 8, **(2012)**, pp. 1400-1414.
[8] L. Wang, M. Kunze and J. Tao, "Performance evaluation of virtual machine-based Grid workflow system", Concurrency and Computation-Practice & Experience, vol. 20, no. 15, **(2008)**, pp. 1759-1771.
[9] Y. C. Lee, R. Subrata and A. Y. Zomaya, "On the Performance of a Dual-Objective Optimization Model for Workflow Applications on Grid Platforms", IEEE Transactions on Parallel and Distributed Systems, vol. 20, no. 9, **(2009)**, pp. 1273-1284.
[10] B. Wu, C.-H. Chi and Z. Chen, "Workflow-based resource allocation to optimize overall performance of composite services", Future Generation Computer Systems, vol. 25, no. 3, **(2009)**, pp. 199-212.
[11] S. Eyerman and L. Eeckhout, "Fine-Grained DVFS Using On-Chip Regulators", ACM Transactions on Architecture and Code Optimization, vol. 8, no. 1, **(2011)**.
[12] G. Kousiouris, T. Cucinotta and T. Varvarigou, "The effects of scheduling, workload type and consolidation scenarios on virtual machine performance and their prediction through optimized artificial neural networks," Journal of Systems and Software, vol. 84, no. 8, **(2011)**, pp. 1270-1291.
[13] B. Dougherty, J. White and D. C. Schmidt, "Model-driven auto-scaling of green cloud computing infrastructure", Future Generation Computer Systems, vol. 28, no. 2, **(2012)**, pp. 371-378.
[14] H. Topcuouglu, S. Hariri and M.-Y. Wu, "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing", IEEE TPDS, vol. 13, no. 3, **(2002)**, pp. 260-274.
[15] Z. Zong, A. Manzanares and X. Ruan, "EAD and PEBD: Two Energy-Aware Duplication Scheduling Algorithms for Parallel Tasks on Homogeneous Clusters", Ieee Transactions on Computers, vol. 60, no. 3, **(2011)**, pp. 360-374.
[16] A. Benoit, P.R Goud and Y. Robert, "Performance and energy optimization of concurrent pipelined applications", Proceedings of IEEE International Parallel and Distributed Processing Symposium, **(2010)**, pp. 1-12.
[17] Y. Yuan, X. Li, Q. Wang and X. Zhu, "Deadline Division-Based Heuristic for Cost Optimization in Workflow Scheduling", Information Sciences, vol. 179, no. 15, pp. 2562-2575.
[18] R. Sakellariou, H. Zhao, E. Tsiakkouri and M. D. Dikaiakos, "Scheduling Workflows with Budget Constraints", Proceedings o Integrated Research in GRID Computing, **(2007)**, pp. 189-202.

[19] J. Yao, C. K. Tham and K. Y. Ng, "Decentralized dynamic workflow scheduling for grid computing using reinforcement learning", Proceedings of International Conference on Networks, **(2006)**, pp. 1-6.
[20] R. Prodan and M. Wieczorek, "Negotiation-Based Scheduling of Scientific Grid Workflows Through Advance Reservations", Journal of Grid Computing, vol. 8, no. 4, **(2010)**, pp. 493-510.
[21] A. Hirales-Carbajal, A. Tchernykh and R. Yahyapour, "Multiple Workflow Scheduling Strategies with User Run Time Estimates on a Grid", Journal of Grid Computing, vol. 10, no. 2, **(2012)**, pp. 325-346.
[22] R. Duan, R. Prodan and T. Fahringer, "Performance and Cost Optimization for Multiple Large-Scale Grid Workflow Applications", Proceedings of ACM/IEEE Conference of Supercomputing, **(2007)**, pp. 1-12.
[23] R. Garg, S. W. Son and M. Kandemir, "Markov model based disk power management for data intensive workloads", Proceedings of IEEE/ACM International Symposium on Cluster Computing and the Grid, **(2009)**, pp. 76-83.

Authors

Hong He received his B.S. degree at Wuhan University of Technology in 1996, and M.S. degree at Xiangtan University in 2006. Currently, he works in Hunan Institute of Engineering as an associate professor. His research interesting is grid computing, cloud computing, distributed resource management.

Xiao Peng received the Ph.D degree in computer science in CSU at 2010. He is currently an associate professor in the Hunan Institute of Engineering. His research interests include cloud computing, distributed resource management. He is a member of ACM and IEEE.

