International Journal of Future Generation Communication and Networking
Vol.7, No.3 (2014), pp.37-48
http://dx.doi.org/10.14257/ijfgcn.2014.7.3.04

Optimizing Data-Accessing Energy Consumption for Workflow Applications in Clouds

Hong He[*] and Dongbo Liu

Department of Computer and Communication, Hunan Institute of Engineering
[*]hhong1970@126.com; liudongbo1974@gmail.com

Abstract

As more and more non-trivial applications have been deployed in cloud-based systems, the energy consumption of running these applications grow rapidly. Existing studies mainly focus on reducing the CPU-related energy consumption, while ignoring the data-accessing related energy costs. In the paper, we present a novel energy-efficient policy, which is aiming at reducing the data-accessing energy consumption of workflow applications that executed on cloud environments. The proposed policy uses a general energy model to describe the energy consumption of any given workflow. By this application-oriented energy model, a two-phase resource deploying algorithm is implemented, which is capable of generating task scheduling schemes with minimal data-accessing energy consumption. Massive experiments are conducted on a real-world cloud platform, and the results show that the proposed policy outperforms existing approaches in terms of energy efficiency and execution performance.

Keywords: *Cloud Computing; Workflow; Energy Efficiency; Task Scheduling; Virtual Machine*

1. Introduction

With the quickly development of high performance computing technology, energy consumed by computer servers has become a serious concern in the design of large-scale enterprise data centers [1, 2]. In addition to high electricity bills and negative environmental implications, increased power consumption may lead to system failures caused by power capacity overload or system overheating, as data centers increasingly deploy new high-density servers (e.g., blade servers), while their power distribution and cooling systems have already approached the peak capacities [3, 4, 5]. On the other side, to maintain desirable QoS, many datacenters tend to equip with more and more advanced IT devices and keep them available for users in 24 hours [6-7]. As a result, the energy consumption of large-scale datacenters grows rapidly, which significantly increases the operational costs of infrastructure providers [8-9]. Therefore, energy-efficient management and control has become a major concern in the design of modern datacenters.

As more and more workflows have been deployed on cloud platforms, energy-aware scheduling for workflow applications attracts plenty of attentions recently [6-8, 10-14]. Most workflow applications are data-intensive and consist of many computational tasks with constraint to data-flow dependencies. Cloud infrastructures have several advantages over traditional HPC systems for executing data-intensive workflows, such as configurable virtual execution environment and elastic service provision. However, these advantages also raise many challenging issues when implementing energy-aware scheduling for data-intensive workflows, which are briefly summarized as following: (1) Data-intensive workflow significantly increases the context-switch overhead and makes typical scheduling strategy inefficient and unpredictable [10-11-15]; (2) Improper intermediate data transferring and moving will result in high energy consumption [7-13-16]; (3) It is difficult to implement an unified scheduling scheme that can reduce both CPU and I/O related energy consumption at the same time [10-11-17]

To address these difficulties, in this paper we design a novel heuristic called Minimal Data-accessing Energy Path (MDEP) for scheduling data-intensive workflows in virtualized cloud platforms. The proposed scheduling algorithm consists of two distinguished phases: firstly, it uses MDEP heuristic for deploying and configuring VM instances with aiming to reduce the energy consumption spent on intermediate data accessing; secondly, it schedules workflow activities to the VM instances according to VM power model. In this way, both the execution performance and energy-efficiency are fully taken into consideration in the proposed algorithm.

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 for both VM instance and workflow scheduling 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

It is well-known that workflow scheduling is a nature NP-hard problem, which means that some heuristic algorithms should be used to obtain second-optimal results [18-20]. In the past yeas, several classical heuristic algorithm have been proposed for solving this problem. Unfortunately, early studies mainly concentrate on system performance, *i.e.*, throughput [21], load-balance [22], response time [16-19] and *etc.*, instead of energy efficiency. Recently, researchers began to take more efforts on energy saving when scheduling workflow applications. For example, in [23], several different scheduling algorithms using the concept of slack sharing among DVS-enabled processors were proposed. The rationale behind the algorithms is to utilize idle (slack) time slots of processors, lowering supply voltage (frequency/speed). This technique is known as slack reclamation. These slack time slots occur, due to earlier completion (than worst case execution time) and/or dependencies of tasks. In [24], two voltage scaling algorithms for periodic, sporadic, aperiodic tasks on a dynamic priority single-processor system are proposed. In [25], the authors implemented a system based on linear programming technology and slacking DVS. This system aims to deliver near-optimal schedules that tightly bound optimal solutions. It incorporates allowable time delays, communication slack, and memory pressure into its scheduling. The linear programming system mainly deals with energy reduction for a given pre-generated schedule with a makespan constraint as in most existing algorithms.

At the same time, the studies on conserving data-accessing related energy consumption mainly concentrate on storage architecture, such as I/O buffer frameworks. In [26], the authors proposed a buffer-based framework, in which a Buffer Controller component is responsible for accumulating several small-writing operations together and then sends these accumulated data to physical disks. In [27], the authors proposed a pre-fetching buffering technique, which uses the block accessing frequency as a heuristic to configure the corresponding buffer settings. The above studies are effective for saving the energy consumption of storage systems at the physical level. However they are not application-oriented, which means that they are only suitable for coarsegrained energy consumption management instead of fine-grained energy consumption optimization and control.

Recently, many researchers have taken their efforts into studying the co-relation between energy consumption and application's characteristics. For example, Cho et al. presented an excellent theoretical work on the interplay between the energy consumption and the application's structure [28]. On the other side, Kang et al. studied the performance and energy costs of scheduling MapReduce applications [29], which can be considered as a special kind of data-intensive workflow with only two levels (mapping level and reducing level). Comparing with the existing works, our work takes more efforts on storage-related energy conserving. To achieve this goal, we first present a general energy model for data-intensive workflow applications, which fully takes the energy consumption of data transferring into account. Secondly, our proposed algorithm uses a new heuristic, namely MDEP, which is very effective to measure the energy cost for any given scheduling scheme. Finally, unlike conventional DAG scheduling, we incorporate the VM deployment and task scheduling in the same framework, which is necessary for adopting it to current virtualized cloud platforms.

3. Problem Description and Formulation

Generally, a workflow is described as a directed acyclic graph (DAG), where the activities represent individual tasks, and the edges represent the precedence-dependencies between these tasks. 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. VM is noted as $<\mathrm{F_v},\mathrm{M_v},\mathrm{S_v}>$, where F_v is the frequency of virtual processor, M_v is the size of virtual memory, S_v is the size of virtual disk. In some real-world clouds (i.e., Amazon EC2), VM instances are pre-defined by the cloud provider and waiting for user's selection. In this work, we ignore some detailed parameters of VM instance such as OS type and price, since they are irrelevant with our topic.

Before scheduling a workflow onto a cloud platform, a set of VM instances should be created and deployed at first. This phase is often called VM provision or VM deployment. Generally speaking, a VM deployment scheme can be considered as a mapping from physical resources to virtual resources. So, we give a formal definition of VM deployment scheme as following. VM deployment scheme is noted as $\boldsymbol{D}:\boldsymbol{C}\times\boldsymbol{S}\rightarrow<F_v,M_v,S_v>$, where $\boldsymbol{C}$ is the set of physical computing nodes, $\boldsymbol{S}$ is the set of physical storage nodes. For a given VM deployment scheme, its output is a set of VM instances that will be deployed on the cloud platform for executing the target workflow. So, we can formalize the workflow scheduling scheme and its corresponding energy consumption as following. A workflow scheduling scheme is noted as $\mathbf{M}:\mathbf{V}\times\mathbf{D}\rightarrow\{0,1\}$ or $\boldsymbol{M}:\boldsymbol{V}\times\boldsymbol{C}\times\boldsymbol{S}\rightarrow\{0,1\}$, in which each element $M_{i,i',i''}$ indicates that activity V_i is scheduled on a VM instance whose virtual processor is allocated from computing node $C_{i'}$ and its virtual disk is from $S_{i''}$. For a given scheduling scheme $\boldsymbol{M}$, the total energy consumption of completing the target workflow $\boldsymbol{G}$ is noted as $E(\boldsymbol{G}, \boldsymbol{M})$.

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 $E(\boldsymbol{G},\boldsymbol{M})$. Given a scheduling scheme $\boldsymbol{M}$, it 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 $E(V_i, M_{i,i',i''})$. Part of $E(V_i, M_{i,i',i''})$ is spent on virtual processor and virtual memory when running the V_i's computing task, and the other part is the energy consumption of virtual 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 onto VM_j, $E_c(V_i, M_{i,i',i''})$ can be measured as

$$E_c(V_i, M_{i,i',i''}) = P_j^{vm}(t)\cdot\Delta T_{exec} \tag{1}$$

where $P_j^{vm}(t)$ is VM power model that can be formulized as

$$P_i^{vm}(t) = \frac{P_{static}}{M} + \sum_{j\in\{cpu,mem,disk\}} \left[r_j^i \cdot P_j(t)\right] \tag{2}$$

where P_{static} is the fixed power consumption for keeping the machine in working state even there is no workload on it, $P_j(t)$ is the dynamic power consumption of component *j*. 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, and the formulation of $E_d(V_i, M_{i,i',i''})$ as following.

$$E_d(V_i, M_{i,i',i''}) = \sum_{j\in Pred(v_i)} \left(P_{disk}^{j''}(t)\frac{d_{j\to i}^{in}}{B_{i',j''}}\right) + \sum_{k\in Succ(v_i)} \left(P_{disk}^{i''}(t)\frac{d_{i\to k}^{out}}{B_{i',i''}}\right) \tag{3}$$

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. Combining (1) and (3), we can obtain the total energy consumption under a given scheduling scheme, which is shown as

$$E(\boldsymbol{G}, \boldsymbol{M}) = \sum_{i=1}^{n} \left[E_c(V_i, M_{i,i',i''}) + E_d(V_i, M_{i,i',i''})\right] \tag{4}$$

In the next section, we will present a heuristic algorithm which is aiming to reduce $E_d(V_i, M_{i,i',i''})$ so as to saving the total energy consumption.

4. Scheduling Policy Implementation and Analysis

4.1. Scheduling Model

When scheduling precedent-constrained applications, *Earliest Start Time* (*EST*) is the key heuristic for many scheduling algorithms. It is defined as the earliest start time of activity V_i if it is scheduled on processor p_j, as shown by

$$EST(V_i) = \begin{cases} 0, & if\ V_i = V_{init} \\ \max\limits_{V_k\in pred(V_i)} \{EFT(V_k) + w_{k,i}\}, & otherwise \end{cases} \tag{5}$$

where $w_{k,i}$ is the costs of data transferring from V_k to V_i, $EFT(V_k)$ is the earliest finishing time of V_k.

As to cloud platform, the *EST* metric can still be used for workflow scheduling. However, some revision of (5) should be made if the data-intensive feature is taken into account, which is shown as following

$$EST(M_{i,i',i''}) = \begin{cases} 0, & if \ V_i = V_{init} \\ \max\limits_{V_j \in Pred(V_i)} \left\{ EFT(M_{j,j',j''}) + \dfrac{d^{out}_{j' \to j''}}{B_{j',j''}} + \dfrac{d^{in}_{j'' \to i'}}{B_{j'',i'}} \right\}, & otherwise \end{cases} \tag{6}$$

where $d^{out}_{j' \to j''}/B_{j',j''}$ is time of data transferring from the computing node of V_j (one of the predecessors of V_i) to its intermediate storage node, $d^{in}_{j'' \to i'}/B_{j'',i'}$ is the time of data transferring from V_j's intermediate storage node to the computing node that allocated to V_i.

The key difference between (5) and (6) is that the intermediate data generated by a data-intensive workflow should be accessed through independent storage nodes instead of being directly transferred between computing nodes. Although such a difference seems slightly, it will have significantly effects on the final performance of a scheduling scheme, including makespan and energy consumption. Motivated by the above observations, in this work we propose a novel heuristic, called *Minimized Energy Consumption in Deployment and Scheduling* (MECDS), for data-intensive workflows in virtualized cloud systems. The MECDS mainly consists of two phases: VM deployment and workflow scheduling. In the phase of VM deployment, we select a storage node with aiming to obtain minimal data accessing energy consumption for the current activities. To do this, we introduce a novel conception, called *Minimal Data-accessing Energy Path* (*MDEP*), which is defined as the minimal total energy consumption from V_{init} to the current activity shown as following

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

According to the definition of *MDEP*, if a storage node $S_{i''}$ can satisfying min{$MDEP(V_i)$} among all storage nodes, then the activity V_i should use it as the storage node. If a VM instance that uses $S_{i''}$ as the underlying storage node has already been created and deployed, then we can go on; otherwise a new VM instance should be created and deployed, which uses $S_{i''}$ as underlying storage node. By repeating the above, we can complete the deployment of VM instances. Therefore, The MDEP metric is effective to measure the communication-related (storage-related) energy consumption. More important, it has incorporated the conventional communication costs, therefore we can use it as a scheduling heuristic when both performance and energy-efficiency should be considered.

In the phase of workflow scheduling, we define the priorities of VM instances as following.

$$rank(VM_i) = MDEP(V_{exit}) + E_c(V_{exit}, VM_i) \tag{8}$$

So, the rank of VM_i indicates the minimal data-accessing energy consumption of V_{exit} if it is assigned to VM_i. Here, we take into account the computing energy consumption $E_c(V_{exit}, VM_i)$ in case two VM instances are configured with same storage node and different computing nodes. All the VM instances are sorted in ascendant order of

$rank(VM_i)$. As to the activity priority of activities, we directly use the classic b-level rank for the convenience of implementation, which is defined as

$$rank(V_i) = \max_{V_j \in Pred(V_i)} \{rank(V_j)\} + \frac{c_i}{r_{cpu}^{i'} F_v} \tag{9}$$

4.2. Implementation of Scheduling Algorithm

Based on the above scheduling model, the detailed implementation of the heuristic algorithm is shown as following

MECDS: *Minimized Energy Consumption in Deployment and Scheduling*

```
Begin
  1.    vset := {};
  2.    k := 0;
  3.    for each $V_i$ do
  4.      Find a $S_j$ that satisfying minimized $MDEP(V_i)$;
  5.      if existing a VM that using $S_i$ then
  6.        continue;
  7.      endif
  8.      k : = k+1;
  9.      Create $VM_k$ and add it to vset;
  10.   end for
  11.   Sort all VMs in vset in ascendant order of $rank(VM_i)$;
  12.   Compute $rank(V_i)$ for all activities by traversing $\mathbf{G}$ from $V_{exit}$ to $V_{init}$.;
  13.   Sort the activities in a scheduling list by non-increasing order of $rank(V_i)$;
  14.   while there are unscheduled activities do
  15.     Select the first task $V_i$ from the list for scheduling;
  16.     for each $VM_j$ in $\{VM_1,\ldots,VM_k\}$ do
  17.       computing $EST(M_{i,i',i''})+T_{exec}(V_i)$;
  18.     end for
  19.     Assign $V_i$ to $VM_{i''}$ that minimizes $EST(M_{i,i',i''})+T_{exec}(V_i)$ by insert scheduling;
  20.     if $S_v < d_i^{out}$ then
  21.       Reconfigure the virtual disk of $VM_{i''}$ to meet the requirement of $V_i$;
  22.     end if
  23.   end while
End.
```

There are two main phases in the implementation of MECDS. The first phase is VM configuration and deployment (as shown in Step 3 ~ Step 10). In this phase, MECDS traverses all activities in the target workflow. In each loop, a storage node S_i that satisfying $\min\{MDEP(V_i)\}$ should be selected out as candidate. It takes O(s) time to do this work in the worst case, and the average time-complexity is O((s-1)/2), where s is the number of storage nodes. Therefore, the total average time-complexity of the first phase is O(n×(s-1)/2), where n is the number of activities in the workflow. In the second phase, MECDS first sort all VM instances in ascendant order of $rank(VM_i)$ (step 11), which will take O(log|vset|) time to do this, where |vset| is the number of VM instances. In steps 12~13, the ranks of all activities should be calculated according to

(13) and sorted in non-increasing order, which will take O(n+n×log(n)) time to do this. In the following while-loop (step 14 ~ step 23), the algorithm traverses the unscheduled list and assign them to suitable VM instances. It takes O(n×log|vset|) time to do this.

Summarizing the above analysis, we can know that the total time-complexity of MECDS is O(n×s+ log|vset|+n+n×log(n)+n×log|vset|). As shown in MECDS, the |vset| will not exceed the number of storage nodes. So, we can know that 1≤|vset|≤s, which means that O(1)≤O(log|vset|)≤O(log|s|). Therefore, we can simply note the time complexity of MECDS as O(n×s+n×log(n)+n×log(s)).

5. Experiments and Performance Comparison

5.1. Experimental Settings

The real-world experiments are conducted on the cloud 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, which are virtualized by using XCP with version 1.1. 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. During the experiments, we use the Opofile to log the energy consumption related events. The target application we selected is the well-known INVMOD workflow, which is designed to study the effects of climatic changes on the water balance. The basic framework of INVMOD workflow is shown in Figure 1.

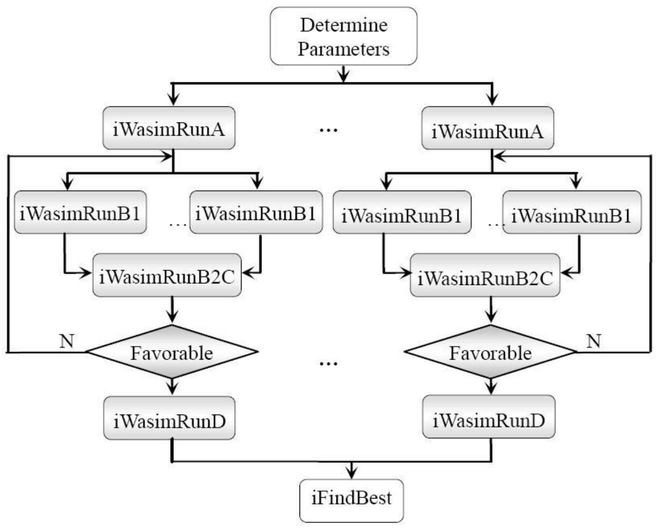
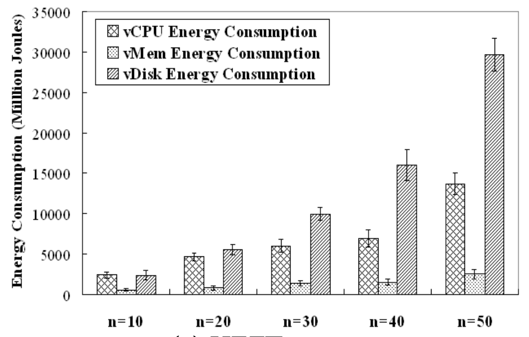


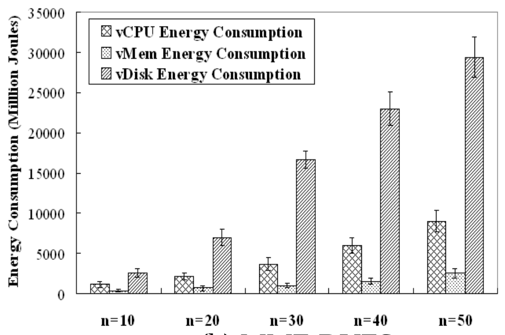
Figure 1. Framework of INVMOD Application

5.2. Comparison of Performance

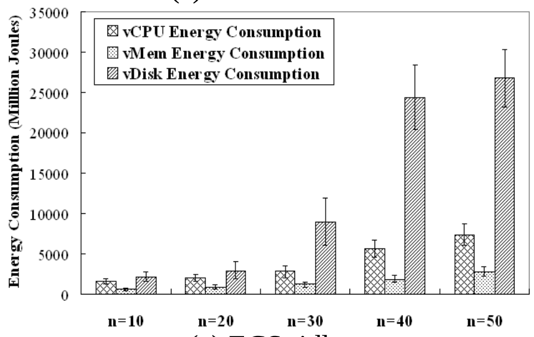
To compare the performance of our MECDS algorithm with other ones, we adapt four other existing scheduling algorithms including HEFT [30], MMF-DVFS [31], ECS+idle [32], and EADAGS [11]. Among the four algorithms, HEFT is the only one without energy-aware functionality, and we use it as the baseline for performance comparison. In this experiment, we use the pre-defined iteration counter (noted as n) to represent the size of the INVMOD, which is gradually increased from 10 to 50. The experimental results are shown in Figure 2 (a)~(e).



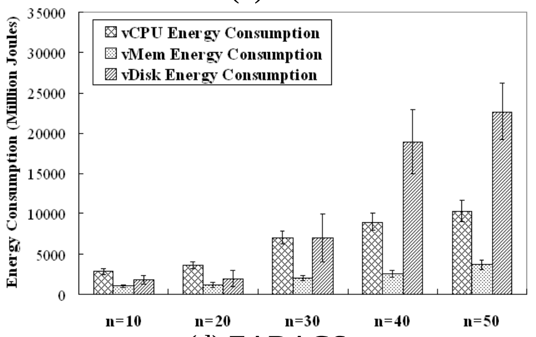
(a) HEFT



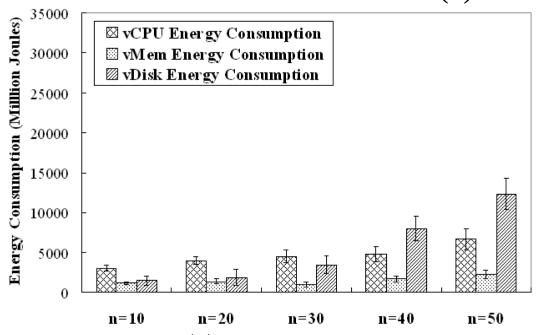
(b) MMF-DVFS



(c) ECS+idle



(d) EADAGS



(e) MECDS

Figure 2. Comparison of Energy Consumption and Distribution with Different Scheduling Algorithm

As we can see that vDisk energy consumption is significantly higher than other kinds of energy consumption. When *n*=10 and *n*=20, HEFT and MMF-DVFS seem perform worst in all the five algorithms in a term of total energy consumption. The difference between them is the energy consumption distribution. More specifically, HEFT costs more energy on vCPU, while MMF-DVFS costs more energy on vDisk. The reason is the same as mentioned in the simulative experiments. However, we found that the vDisk enery consumption of MMF-DVFS and ECS+idle increase more quickly than HEFT and EADAGS when *n* > 20. This is different from our simulative results.

To find out the reason, we investigate the logs of scheduling procedure during our experiments. The logs indicate that HEFT algorithm tends to assign tasks onto those VM instances that configured with higher computing capability (*i.e.*, CN_1~CN_8), while

MMF-DVFS and ECS+idles tend to map tasks onto the VMs that configured with more power-efficient computing nodes. In common sense, MMF-DVFS and ECS+idle should performs more energy-efficiently. However, the structural characteristic plays the more important role in this case. When n=10 it means that the loop will be executed 10 times at most. So, increasing n value will significantly prolong the execution time of solving sub-problems. More importantly, each loop has another parallel branch, in which all the tasks (iWasimRunB) require transferring data. Simply saying, the performance bottleneck of running INVMOD is the execution of this inside loop. When n is increasing, this bottleneck becomes more and more significant. As HEFT always tend to uses powerful computing resources, which enable it to reduce the overall makespan of INVMOD and in turn reduce the energy consumption. On the contrary, MMF-DVFS and ECS+idle do not notice this application-specific feature. Although they can reduce the vCPU energy consumption, it still can not compensate the energy wastage spent by idle storage nodes.

Like the HEFT, our MECDS also is effective to improve the execution efficiency of the inside loop. However, its strategy is to improve the data transferring efficiency of the parallel branch, which is quiet different from HEFT's strategy. More importantly, this strategy seems more robust than HEFT's. When n=50 HEFT performs as worst as MMF-DVFS in term of vDisk energy consumption. The overall energy consumption metric may be confusing, since we can not tell how much energy is effectively used and how much is wasted. To further investigate the energy-efficiency, we introduce three measurements here: *Effective Computing Energy Consumption* (ECEC), *Effective Data Accessing Energy Consumption* (EDAEC), *Ineffective Energy Consumption* (IEEC). We recorded all the three measurements in all cases. For the limitation of space, we only present the experiment results when n=50, since it is most representative for the data-intensive topic. The results are shown in Figure 3, and all the measurements are converted into percentage form for clear representation.

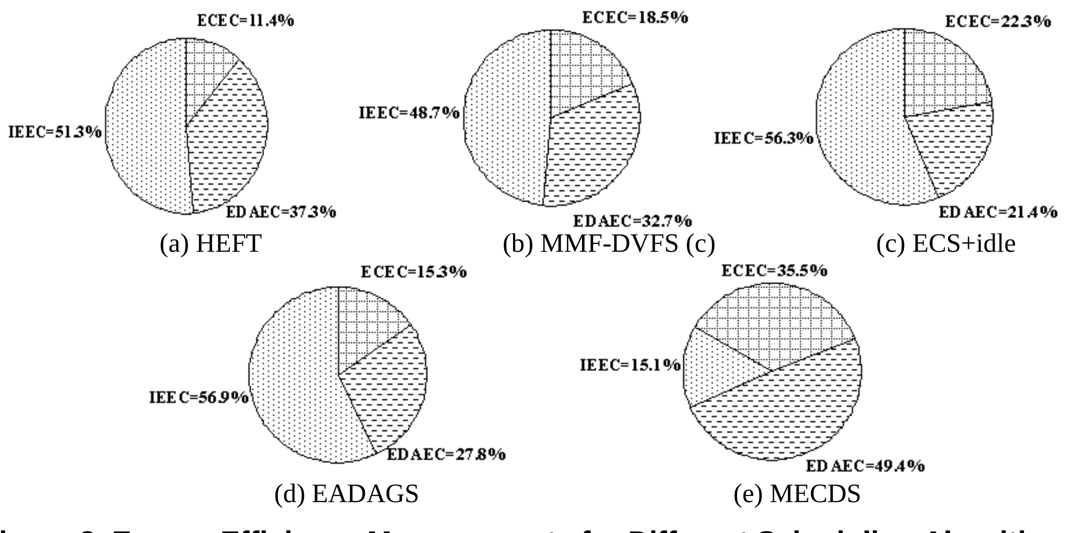


Figure 3. Energy-Efficiency Measurements for Different Scheduling Algorithm (*n*=50)

As shown in Figure 3, we can see that all the IEEC measurements of other four algorithms are more than 48%, which means that about half of the energy is wasted when running INVMOD application on our cloud platform. It is clear that MECDS outperforms other algorithms because of its high effective energy consumption for both computing nodes and storage nodes. It is noteworthy that MECDS is not aiming for optimizing the computing related energy consumption. So, its high ECEC measurement comes from the reduced execution time, which in turn reduced the total computing related energy consumption. Among the other four algorithms, HEFT has the lowest ECEC and the highest EDAEC. It is because the high-performance computing nodes tend to waste more energy if they are kept in low utilization for a long time. Relatively, its EDAEC measurement is increased as a result. Among the five algorithms, only EADAGS and MECDS are designed for data-intensive workflows. When n=10, 20 and 30, the performance difference between the two algorithms is not very distinguishing. When n is bigger than 30, the vDisk energy consumption of EADAGS increases significantly. EADAGS also uses DVFS mechanism for energy conservation. However, it does not directly aiming to reducing the processor related energy consumption. On the contrary, it uses CCR metric as its objective function which is dynamical adjusted through using DVFS mechanism. This strategy is difficult to be analyzed theoretically.

6. Conclusion

To address the issue of energy-aware scheduling for data-intensive workflow applications in virtualized platforms, this work presents a novel heuristic, namely *Minimal Data-accessing Energy Path* (MDEP), for VM deployment and task scheduling. By comparing the results with four existing heuristic algorithms, we are convinced that the MECDS is effective to conserve the energy consumption for data-intensive workflows. In addition, the experimental results also indicate that the MECDS algorithm is more robust than other DVFS-based algorithms, especially when the system is in presence of intensive data-accessing requests.In the future, we plan to incorporate some adaptive mechanisms into the MECDS algorithm, such as workload-aware and load-balance mechanism, configurable strategy for VM deployment, and energy-aware VM migration mechanism. Furthermore, we are planning to design a more energy-efficient VM scheduler in VM hypervisor level.

References

[1] M. D. Dikaiakos, G. Pallis and D. Katsaros. "Cloud computing distributed internet computing for IT and scientific research," IEEE Internet Computing, vol. 13, no. 5, **(2009)**, pp. 10-13.

[2] J. Ju, J. Wu and J. Fu, "A survey on cloud storage," Journal of Computers, vol. 6, no. 8, **(2011)**, pp. 1764-1771.

[3] S. K. Garg, C. S. Yeo and A. Anandasivam, "Environment-conscious scheduling of HPC applications on distributed Cloud-oriented data centers," Journal of Parallel and Distributed Computing, vol. 71, no. 6, **(2011)**, pp. 732-749.

[4] S. Son, G. Jung, and S. C. Jun, "An SLA-based cloud computing that facilitates resource allocation in the distributed data centers of a cloud provider," Journal of Supercomputing, vol. 64, no. 2, **(2013)**, pp. 606-637.

[5] X. Zhu, D. Young and B. J. Watson, "1000 islands: an integrated approach to resource management for virtualized data centers," Cluster Computing-the Journal of Networks Software Tools and Applications, vol. 12, no. 1, **(2009)**, pp. 45-57.

[6] A. Beloglazov, J. Abawajy and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing," Future Generation Computer Systems, vol. 28, no. 5, **(2012)**, pp. 755-768.

[7] D. Kliazovich, P. Bouvry and S. U. Khan, "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers," Journal of Supercomputing, vol. 62, no. 3, **(2012)** December, pp. 1263-1283.

[8] A. Beloglazov, and R. Buyya, "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints", IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 7, **(2013)**, pp. 1366-1379.

[9] J.-F. Pineau, Y. Robert and F. Vivien, "Energy-aware scheduling of bag-of-tasks applications on master-worker platforms", Concurrency and Computation-Practice & Experience, vol. 23, no. 2, **(2011)**, pp. 145-157.
[10] 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)** March, pp. 360-374.
[11] S. Baskiyar and R. A. Kader, "Energy aware DAG scheduling on heterogeneous systems", Cluster Computing-the Journal of Networks Software Tools and Applications, vol. 13, no. 4, **(2010)**, pp. 373-383.
[12] K. Li, "Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers", IEEE Transactions on Computers, vol. 61, no. 12, **(2012)**, pp. 1668-1681.
[13] X. Zhu, C. He and K. Li, "Adaptive energy-efficient scheduling for real-time tasks on DVS-enabled heterogeneous clusters", Journal of Parallel and Distributed Computing, vol. 72, no. 6, **(2012)**, pp. 751-763.
[14] W. Y. Lee, "Energy-efficient scheduling of periodic real-time tasks on lightly loaded multicore processors", IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 3, **(2012)**, pp. 530-537.
[15] M. Mezmaz, N. Melab and Y. Kessaci, "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems", Journal of Parallel and Distributed Computing, vol. 71, no. 11, **(2011)**, pp. 1497-1508.
[16] C. Yu and D. C. Marinescu, "Algorithms for divisible load scheduling of data-intensive applications", Journal of Grid Computing, vol. 8, no. 1, **(2010)**, pp. 133-155.
[17] F. Almeida, V. Blanco and A. Cabrera, "Modeling energy consumption for master-slave applications", Journal of Supercomputing, vol. 65, no. 3, **(2013)**, pp. 1137-1149.
[18] X. Wang, Y. Wang and H. Zhu, "Energy-efficient task scheduling model based on mapReduce for cloud computing using genetic algorithm", Journal of Computers, vol. 7, no. 12, **(2012)**, pp. 2962-2970.
[19] E. Aubanel, "Scheduling of tasks in the parareal algorithm", Parallel Computing, vol. 37, no. 3, **(2011)**, pp. 172-182.
[20] A. Z. S. Shahul and O. Sinnen, "Scheduling task graphs optimally with A", Journal of Supercomputing, vol. 51, no. 3, **(2010)** March, pp. 310-332.
[21] F. A. Omara and M. M. Arafa, "Genetic algorithms for task scheduling problem", Journal of Parallel and Distributed Computing, vol. 70, no. 1, **(2010)**, pp. 13-22.
[22] A. J. Page, T. M. Keane and T. J. Naughton, "Multi-heuristic dynamic task allocation using genetic algorithms in a heterogeneous distributed system", Journal of Parallel and Distributed Computing, vol. 70, no. 7, **(2010)** July, pp. 758-766.
[23] D. Zhu, R. Melhem and B.R. Childers, "Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems", IEEE Transactions on Parallel and Distributed Systems vol. 14, no. 7, **(2003)**, pp. 686-700.
[24] R. Ge, X. Feng and K.W. Cameron, "Performance-constrained distributed dvs scheduling for scientific applications on power-aware clusters", Proceedings of the ACM/IEEE Conference on Supercomputing, **(2005)** November, pp. 34-44.
[25] B. Rountree, D. K. Lowenthal and S. Funk, "Bounding energy consumption in large-scale MPI programs", Proceedings of the ACM/IEEE Conference on Supercomputing, **(2007)** November, pp. 1-9.
[26] Z. Zong, M. Briggs and N. Connor, "An energy-efficient framework for large-scale parallel storage systems", Proceedings of IEEE International Parallel and Distributed Processing Symposium, pp. 1-7, Long Beach CA, USA, **(2007)**.
[27] A. Manzanares, K. Bellam and X. Qin, "A prefetching scheme for energy conservation in parallel disk systems", Proc. of IEEE International Parallel and Distributed Processing Symposium, pp. 1-5, Miami Florida, USA, **(2008)**.
[28] S. Cho and R. G. Melhem, "On the interplay of parallelization, program performance, and energy consumption", IEEE Transactions on Parallel and Distributed Systems, vol. 21, no. 3, **(2010)**, pp. 342-353.
[29] H. Kang, Y. Chen and J. L. Wong, "Enhancement of xen's scheduler for mapreduce workloads", Proceedings of International Symposium on High Performance Distributed Computing, San Jose, California, USA, **(2011)**, pp. 251-262.
[30] H. Topcuoglu, S. Hariri and M.Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing", IEEE Transactions on Parallel and Distributed Systems, vol. 13, no. 2, **(2002)**, pp. 260-274.
[31] N. B. Rizvandi, J. Taheri, A. Y. Zomaya and Y. C. Lee, "Linear combinations of dvfs-enabled processor frequencies to modify the energy-aware scheduling algorithms", Proceedings of IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, Melbourne, Australia, **(2010)** May 17-20, pp. 388-397.
[32] 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.

Authors

Hong He, 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.

Dongbo Liu, he received his master degree in Hunan University in 2004. Now he works in Hunan Institute of Engineering and is a Ph.D candidate in Hunan University. His research interests include distributed intelligence, multi-agent systems, high-performance application. He is now a student member of CCF in China, and worked as Senior Engineer in HP High-performance Lab.