J Supercomput (2012) 60:268–280
DOI 10.1007/s11227-010-0421-3

Energy efficient utilization of resources in cloud computing systems

Young Choon Lee · Albert Y. Zomaya

Published online: 19 March 2010

Abstract The energy consumption of under-utilized resources, particularly in a cloud environment, accounts for a substantial amount of the actual energy use. Inherently, a resource allocation strategy that takes into account resource utilization would lead to a better energy efficiency; this, in clouds, extends further with virtualization technologies in that tasks can be easily consolidated. Task consolidation is an effective method to increase resource utilization and in turn reduces energy consumption. Recent studies identified that server energy consumption scales linearly with (processor) resource utilization. This encouraging fact further highlights the significant contribution of task consolidation to the reduction in energy consumption. However, task consolidation can also lead to the freeing up of resources that can sit idling yet still drawing power. There have been some notable efforts to reduce idle power draw, typically by putting computer resources into some form of sleep/power-saving mode. In this paper, we present two energy-conscious task consolidation heuristics, which aim to maximize resource utilization and explicitly take into account both active and idle energy consumption. Our heuristics assign each task to the resource on which the energy consumption for executing the task is explicitly or implicitly minimized without the performance degradation of that task. Based on our experimental results, our heuristics demonstrate their promising energy-saving capability.

Keywords Cloud computing · Energy aware computing · Load balancing · Scheduling

Y.C. Lee · A.Y. Zomaya (✉)
Center for Distributed and High Performance Computing, School of Information Technologies, The University of Sydney, Sydney, NSW 2006, Australia
e-mail: zomaya@it.usyd.edu.au

Y.C. Lee
e-mail: yclee@it.usyd.edu.au

1 Introduction

Cloud computing has become a very promising paradigm for both consumers and providers in various fields of endeavor, such as science, engineering and business. A cloud typically consists of multiple resources possibly distributed and heterogeneous. Although the notion of a cloud existed in one form or another for some time now (its roots can be traced back to the mainframe era [1]), however, recent advances in virtualization technologies in particular have made it much more compelling compared to the time when it was first introduced. The adoption and deployment of clouds has many attractive benefits, such as scalability and reliability; however, clouds in essence aim to deliver more economical solutions to both parties (consumers and providers). By economical we mean that consumers only need to pay for what resources they need while providers can capitalize poorly utilized resources. From a provider's perspective, the maximization of the profit is a high priority. In this regard, the minimization of energy consumption plays a crucial role. Moreover, energy consumption can be much reduced by increasing resource utilization. Energy usage in large-scale computer systems like clouds also yields many other serious issues including carbon emissions and system reliability.

The recent advocacy of the so-called, *green* or *sustainable computing* (tightly coupled with energy consumption) has been getting a lot of attention. The scope of sustainable computing is not limited to main computing components (e.g., processors, storage devices and visualization facilities), but it can expand into a much larger range of resources associated with computing facilities including auxiliary equipments, water used for cooling and even physical/floor space that these resources occupy. Energy consumption in computing facilities raises various monetary, environmental and system performance concerns. A recent study on power consumption of server farms [2] shows that electricity use for servers worldwide—including their associated cooling and auxiliary equipment—in 2005 cost US$ 7.2 bn. The study also indicates that electricity consumption in that year had doubled as compared with consumption in 2000. Clearly, there are environmental issues with the generation of electricity. The number of transistors integrated into today's Intel Itanium 2 processor reaches to nearly 1 billion. If this rate continues, the heat (per square centimeter) produced by future Intel processors would exceed that of the surface of the sun [3], resulting in poor system performance.

Recent advances in hardware technologies have improved the energy consumption issue to a certain degree. However, it still remains a serious concern for sustainable computing because the amount of energy consumed by computing and auxiliary hardware resources is affected substantially by their usage patterns. In other words, resource under-utilization or over-loading incurs a higher volume of energy consumption when compared with efficiently utilized resources. This calls for the development of various software energy-saving techniques including scheduling and virtualization.

Energy consumption and resource utilization in clouds are highly coupled. Specifically, resources with a low utilization rate still consume an unacceptable amount of energy compared with their energy consumption when they are fully utilized or sufficiently loaded. According to recent studies in [4–7], average resource utilization in

most data centers can be as low as 20%; and the energy consumption of idle resources can be as much as 60% or peak power. In response to this poor resource utilization, task consolidation is an effective technique to increase resource utilization and in turn reduces energy consumption. This technique is greatly enabled by virtualization technologies that facilitate the running of several tasks on a single physical resource concurrently.

Recent studies identified that server energy consumption scales linearly with (processor) resource utilization [6, 8]. This encouraging fact further advocates the significant contribution of task consolidation to the reduction in energy consumption. However, task consolidation can also lead to the freeing up of resources that can sit idling yet still drawing power. There have been some noticeable efforts to reduce idle power draw, typically by putting computer resources into some form of sleep/power-saving mode [8, 9]; however, this mode switching is not possible when utilization is low. In this paper, we present two energy-conscious task consolidation heuristics (*ECTC* and *MaxUtil*), which aim to maximize resource utilization and explicitly take into account both active and idle energy consumption. Our heuristics assign each task to the resource on which the energy consumption for executing the task is minimized without any performance degradation. The energy consumption of a task in our study is computed based on an objective function derived from findings reported in the literature; that is, energy consumption can be significantly reduced when the task is consolidated with one or more tasks than when it is solely assigned to a resource. Based on our experimental results, our heuristics demonstrate their promising energy-saving capability.

The remainder of the paper is organized as follows. Section 2 describes the cloud, application and energy models, and the task consolidation problem used in this paper. Section 3 overviews the related work. The *ECTC* and *MaxUtil* heuristics are presented in Sect. 4 followed by performance evaluation results and conclusions in Sects. 5 and 6, respectively.

2 Models

In this section, we describe the cloud, application and energy models, and define the task consolidation problem targeted in this work. The details of the model presented in this section focus on resource management characteristics and issues from a cloud provider's perspective.

2.1 Cloud model

The target system used in this work consists of a set R of r resources/processors that are fully interconnected in the sense that a route exists between any two individual resources (Fig. 1). We assume that resources are homogeneous in terms of their computing capability and capacity; this can be justified by using virtualization technologies. Nowadays, as many-core processors and virtualization tools (e.g., Linux KVM, VMware Workstation & VMware Fusion, Xen, Parallels Desktop for Mac, VirtualBox) are commonplace, the number of concurrent tasks on a single physical

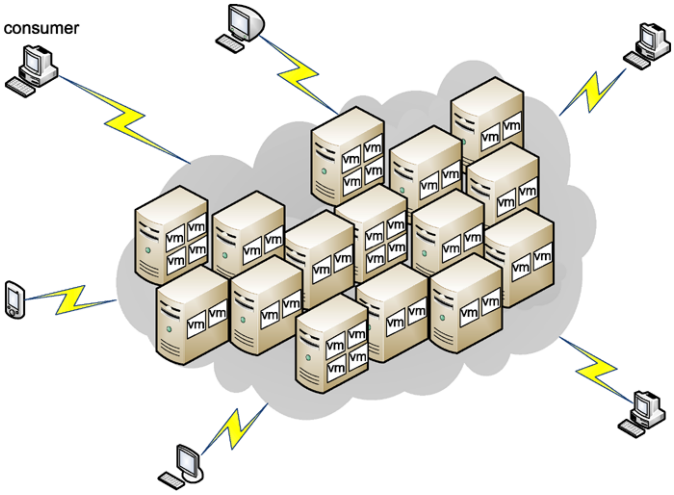

Fig. 1 Cloud model

resource is loosely bounded. Although a cloud can span across multiple geographical locations (i.e., distributed), the cloud model in our study is assumed to be confined to a particular physical location. The inter-processor communications are assumed to perform with the same speed on all links without substantial contentions. It is also assumed that a message can be transmitted from one resource to another while a task is being executed on the recipient resource, which is possible in many systems.

2.2 Application model

Services offered by cloud providers can be classified into software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS). Note that, when instances of these services are running, they can be regarded as computational tasks or simply tasks. While IaaS requests are typically tied with predetermined time frames (e.g., pay-per-hour), requests of SaaS and PaaS are often not strongly tied with a fixed amount of time (e.g., pay-per-use). However, it can be possible to have estimates for service requests for SaaS and PaaS based on historical data and/or consumer supplied service information. Service requests in our study arrive in a Poisson process and the requested processing time follows exponential distribution. We assume that the processor/CPU usage (utilization) of each service request can be identifiable. It is also assumed that disk and memory use correlates with processor utilization [6]. Hereafter, application, task and service are used interchangeably.

2.3 Energy model

Our energy model is devised on the basis that processor utilization has a linear relationship with energy consumption. In other words, for a particular task, the information on its processing time and processor utilization is sufficient to measure the

energy consumption for that task. For a resource r_i at any given time, the utilization U_i is defined as

$$U_i = \sum_{j=1}^{n} u_{i,j} \tag{1}$$

where n is the number of tasks running at that time and $u_{i,j}$ is the resource usage of a task t_j.

The energy consumption E_i of a resource r_i at any given time is defined as

$$E_i = (p_{\max} - p_{\min}) \times U_i + p_{\min} \tag{2}$$

where $p_{\max}$ is the power consumption at the peak load (or 100% utilization) and $p_{\min}$ is the minimum power consumption in the active mode (or as low as 1% utilization).

In this study, we assume that resources in the target system are incorporated with an effective power-saving mechanism (e.g., [8]) for idle time slots; this results in the difference in energy consumption of resources between active and idle states being significant. Specifically, the energy consumption of an idle resource at any given time is set to 10% of $p_{\min}$. Since the overhead to turn off and back on takes a non-negligible amount of time, this option for idle resources is not considered in our study.

2.4 Task consolidation problem

The task consolidation (also known as server/workload consolidation) problem in this study is the process of assigning a set N of n tasks (service requests or simply services) to a set R of r cloud resources—without violating time constraints—aiming to maximize resource utilization, ultimately to minimize energy consumption. Here, time constraints directly relate to resource usage associated with tasks; that is, the resource allocated to a particular task must sufficiently provide the resource usage of that task. For example, a task with its resource usage requirement of 60% cannot be assigned to a resource for which the resource utilization at the time of that task's arrival is 50%.

3 Related work

As cloud and green computing paradigms are closely related and they are gaining their momentum, the energy efficiency of clouds has become one of most crucial research issues. Advancements in hardware technologies [10], such as low–power CPUs, solid state drives, and energy–efficient computer monitors have helped relieve this energy issue to a certain degree. In the meantime, there also have been a considerable amount of research conducted using software approaches, such as scheduling and resource allocation [11–17] and task consolidation [18–21].

The scheduling and resource allocation approach is primarily enabled using *slack reclamation* with the support of dynamic voltage/frequency scaling (DVFS; more specifically 'processor undervolting') [22] incorporated into many recent commodity

processors. This technique temporarily decreases voltage supply level at the expense of lowering processing speed. Slack reclamation is made possible primarily by recent DVFS-enabled processors and the parallel nature of the deployed tasks. For example, when the execution of a task is dependent on two predecessor tasks and these two tasks have different completion times, the predecessor task with an earlier completion time can afford additional run–time (slack); this slack can then be exploited using undervolting for energy saving. Since most DVFS-based energy-aware scheduling and resource allocation techniques are static (offline) algorithms with an assumption of tight coupling between tasks and resources (i.e., local tasks and dedicated resources), their application to our cloud scenario is not apparent, if not possible. In [16] and [17], the main focus is rather on temperature and/or heat emissions of resources in data centers. Although thermal-aware load balancing or scheduling does not seem to be directly related to energy consumption, the cooling (air-conditioning) in these computing facilities consumes a considerable amount of energy [2, 17].

In the task consolidation approach, virtualization technologies play a key role. These technologies with the prevalence of many-core processors have greatly eased and boosted parallel processing; that is, a single many-core processor often runs multiple tasks (processes or threads) concurrently. This parallel processing practice at a glance seems to inherently increase performance and productivity. But the trade-off between this increase and the energy consumption should be carefully investigated. That is, load imbalance in those many-core processors in particular is a major source of energy drainage. This fact has much motivated many task consolidation studies [18–21].

Task consolidation in [18] is approached using the traditional bin-packing problem with two main characteristics, i.e., CPU and disk usage. The algorithm proposed in [18] attempts to consolidate tasks balancing energy consumption and performance on the basis of the Pareto frontier (optimal points). Two steps incorporated into the algorithms are: (1) the determination of optimal points from profiling data, and (2) energy-aware resource allocation using the Euclidean distance between the current allocation and the optimal point at each server.

In [19], a utility analytic model for Internet-oriented task consolidation is proposed. The model considers tasks being requested for services like e-books database or e-commerce Web services. The main performance goal is the maximization of resource utilization to reduce energy consumption with the same quality of service guarantee as in the use of dedicated servers. The model introduces the impact factor metric to reflect the performance degradation of consolidated tasks.

Task consolidation mechanisms developed in [20, 21] deal with energy reduction using different techniques, especially [20]. Unlike typical task consolidation strategies, the approach used in [20] adopts two interesting techniques, memory compression and request discrimination. The former enables the conversion of CPU power into extra memory capacity to allow more (memory intensive) tasks to be consolidated, whereas the latter blocks useless/unfavorable requests (coming from Web crawlers) to eliminate unnecessary resource usage. The VirtualPower approach proposed in [21] incorporates task consolidation into its power management combining ‘soft’ and ‘hard’ scaling methods. These two methods are based on power management facilities (e.g., resource usage control method and DVFS) equipped with virtual machines (VMs) and physical processors, respectively.

More recently, several noteworthy efforts on energy-aware scheduling (in large-scale distributed computing systems like grids) using game theoretic approaches have been reported (e.g., [23, 24]). In [23], a cooperative game model and the Nash Bargaining solution have been presented to address the grid load balancing problem. The main objective is to minimize energy consumption while maintaining a specified service quality, e.g., time and fairness. The work presented in [24] is similar to that in [23] in that they both deal with independent jobs with the (semi-)static scheduling mode. Moreover, they both leverage DVFS technique in their energy minimization.

4 Task consolidation algorithms

Task consolidation is an effective means to manage resources particularly in clouds both in the short and long terms. In the short term case, volume flux on incoming tasks can be "energy-efficiently" dealt with by reducing the number of active resources, and putting redundant resources into a power-saving mode or even turning off some idle resources systematically. In the long term case, cloud infrastructure providers can better model/provision power and resources; this alleviates the burden of excessive operational costs due to over provisioning. The focus in this paper on the short term case, even though task consolidation results our algorithms deliver can be used as an estimator in the long term provisioning case.

In this section, we present two energy-conscious task consolidation algorithms, *ECTC* and *MaxUtil*. They are in fact described side by side since they share several common features with the main difference being whether energy consumption is taken into account explicitly or implicitly. In other words, *MaxUtil* makes task consolidation decisions based on resource utilization, which is a key indicator for energy efficiency in our scenario.

4.1 Algorithm description

Both *ECTC* and *MaxUtil* follow similar steps (Fig. 2) with the main difference being their cost functions. In a nutshell, for a given task, two heuristics check every resource and identify the most energy-efficient resource for that task. The evaluation of the most energy-efficient resource is dependent on the used heuristic, or more specifically the cost function employed by the heuristic. The cost function of *ECTC* computes the actual energy consumption of the current task subtracting the minimum energy consumption ($p_{\min}$)—required to run a task—if there are other tasks running in parallel with that task. That is, the energy consumption of the overlapping time period among those tasks and the current task is explicitly taken into account. The cost function tends to discriminate the task being executed alone. The value $f_{i,j}$ of a task t_j on a resource r_i obtained using the cost function of *ECTC* is defined as:

$$f_{i,j} = \big((p_\Delta \times u_j + p_{\min}) \times \tau_0\big) - \big((p_\Delta \times u_j + p_{\min}) \times \tau_1 + p_\Delta \times u_j \times \tau_2\big) \quad (3)$$

where p_Δ is the difference between $p_{\max}$ and $p_{\min}$, u_j is the utilization rate of t_j, and τ_0, τ_1 and τ_2 are the total processing time of t_j, the time period t_j is running alone and that t_j is running in parallel with one or more tasks, respectively. The

Fig. 2 Algorithm description

Input: A task t_j and a set R of r cloud resources
Output: A task-resource match
1. Let $r^* = \emptyset$
2. **for** $\forall r_i \in R$ **do**
3. Compute the cost function value $f_{i,j}$ of t_j on r_i
4. **if** $f_{i,j} > f_{*,j}$ **then**
5. Let $r^* = r_i$
6 Let $f_{*,j} = f_{i,j}$
7. **end if**
8. **end for**
9. Assign t_j to r^*

rationale behind this function is that the energy consumption at the lowest utilization is far greater than that at idle and the additional energy consumption imposed by overlapping tasks contributes a relatively low increase.

The cost function of *MaxUtil* is devised with the average utilization—during the processing time of the current task—as its core component. This function aims to increase consolidation density; and its advantage is two-fold. The first and obvious advantage is energy consumption is reduced. And, the second benefit is that *MaxUtil*'s cost function implicitly decreases the number of active resources since it tends to intensify the utilization of a small number of resources compared with *ECTC*'s cost function. The value $f_{i,j}$ of a task t_j on a resource r_i using the cost function of *MaxUtil* is defined as:

$$f_{i,j} = \frac{\sum_{\tau=1}^{\tau_0} U_i}{\tau_0} \tag{4}$$

4.2 Performance analysis and discussion

As incorporated into our energy model, energy consumption is directly proportional to resource utilization. At a glimpse, for any two task-resource matches, the one with a higher utilization may be selected. However, since the determination of the right match is not entirely dependent on the current task, *ECTC* makes its decisions based rather on the (sole) energy consumption of that task. In Fig. 3a, task 3 (t_3) arrives at time 14 after tasks 0, 1 and 2, and it is assigned onto resource 1 (r_1) based on energy consumption (40 with $p_{\max}$ and $p_{\min}$ of 30 and 20, respectively), even though the utilization of resource 0 (r_0) is higher (64%, but energy consumption is 80) if t_3 is assigned on r_0. On the other hand, there are cases in which matches with higher utilization in fact (lead to) consume less energy (Fig. 4b). *MaxUtil* assigns t_3 onto r_0 and this leads to a better match for t_4 compared with *ECTC* (Fig. 3b). These contrary situations are often exhibited due to the dynamic nature of clouds—the decision for a given newly arrived task is made based on the current state of task-resource bindings, and thus the decision is only a local optimum. The superiority of performance of our heuristics can be hardly determined since the quality of their task consolidation decisions in any given period of time may differ with characteristics of subsequent tasks after that period.

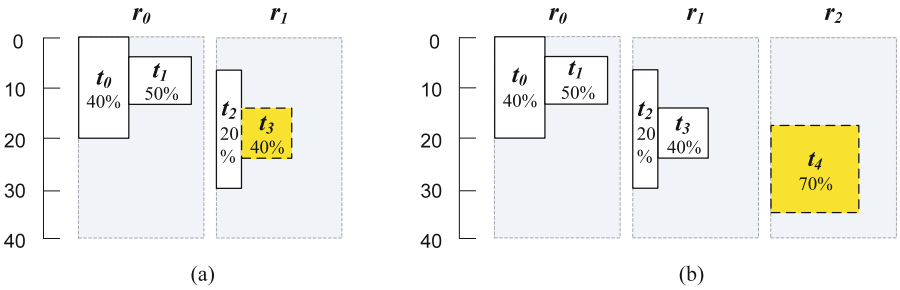

Fig. 3 Consolidation example for tasks in Table 1 using *ECTC*

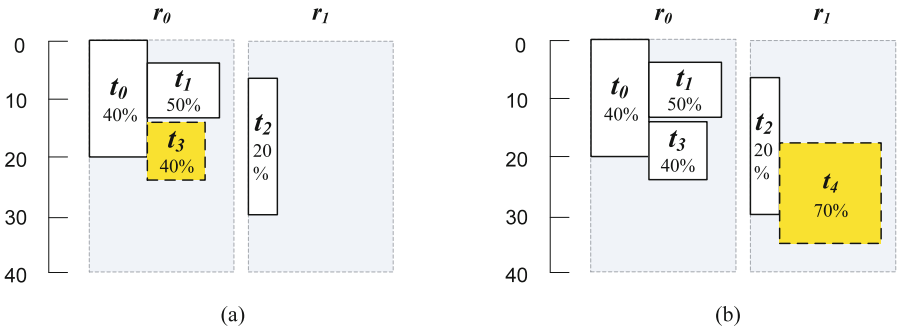

Fig. 4 Consolidation example for tasks in Table 1 using *MaxUtil*

Table 1 Task properties

Task	Arrival time	Processing time	Utilization
0	0	20	40%
1	3	8	50%
2	7	23	20%
3	14	10	40%
4	20	15	70%

5 Experimental evaluation

In this section, we describe experimental methods and settings including task characteristics and their generation. Experimental results are then presented based on energy consumption. While resource utilization might be a good performance measure, however, average utilization rates over all resources are not shown since they are already represented by energy consumption.

5.1 Experiments

The performance of *ECTC* and *MaxUtil* was thoroughly evaluated with a large number of experiments using a diverse set of tasks. In addition to task characteristics,

three algorithms (random, *ECTC* and *MaxUtil*) were used. Variants of these three algorithms were further implemented incorporating task migration.

The total number of experiments conducted is 1,500—50 different numbers of tasks (between 100 and 5,000 at intervals of 100), 10 mean inter-arrival times (between 10 and 100 with a random uniform distribution), and three kinds of resource usage patterns. Three different kinds of resource usage patterns are random, low and high. In the first case, resource usage of tasks generated is random and uniformly distributed between 10% and 100% (or 0.1 and 1.0). For tasks with low and high resource usage patterns, usage ranges are generated using a Gaussian random number generator with mean utilization rates of 30% and 70%, respectively. Task arrival times are modelled in a Poisson process and task processing times follow exponential distribution. We assume task processing times specified are hard deadlines; that is, performance degradation is not acceptable. We used $p_{\max}$ and $p_{\min}$ of 30 and 20, respectively. These values can be seen as rough estimates in actual resources and can be referenced as 300 watt and 200 watt, respectively.

Since (to the best of our knowledge) existing task consolidation algorithms are not directly comparable to our heuristics, our comparisons have been carried out between *ECTC*, *MaxUtil* and random. Specifically, the application of most DVFS-based energy-saving techniques is tied strongly with deadline-constrained and/or inter-dependent tasks; besides, they do not address the relationship between utilization and energy consumption, i.e., task consolidation is not considered. Those existing task consolidation techniques introduced in Sect. 3 exhibit substantial differences—from our heuristics—in energy and scheduling models.

During initial experiments with those three heuristics (*ECTC*, *MaxUtil* and random), we observed that in some circumstances the relocation of some running tasks can further reduce energy consumption. This observation motivated us to implement a variant for each of those three incorporating task migration and these variants are named *ECTC_m*, *MaxUtil_m* and random_m, respectively. Relocation is considered—for each running task—at any time resource utilization changes, i.e., task completion or task commencement.

5.2 Results

The entire results obtained from our extensive simulations are summarized in Table 2 and results for different resource usage patterns are presented in Fig. 5. Although the simulations were performed with 50 different numbers of tasks as stated in Sect. 5.1, only results obtained with 11 representative task volumes are presented in Fig. 5. Energy savings in Table 2 are relative rates to results obtained from experiments using random algorithms (random and random_m). These results in Table 2 and Fig. 5 clearly demonstrate the competent energy-saving capability of *ECTC* and *MaxUtil*. Overall, *ECTC* and *MaxUtil* outperformed random algorithms—regardless of the adoption of migration—by 18% and 13%, respectively. While energy savings with high and random resource usage patterns are still appealing, tasks with low resource usage are most suitable for task consolidation as shown in Fig. 5a. Interestingly, the benefit of using migration was not apparent (Table 2). This is mainly because migrated tasks tend to be with short remaining processing times; and these

Table 2 Relative energy savings

Usage pattern	Algorithm											
	MaxUtil						*ECTC*					
	low		high		random		low		high		random	
Migration	no	yes	no	yes	no	yes	no	yes	no	yes	no	yes
Energy savings	25%	23%	4%	5%	12%	11%	33%	32%	9%	9%	17%	16%
Average	13%						18%					

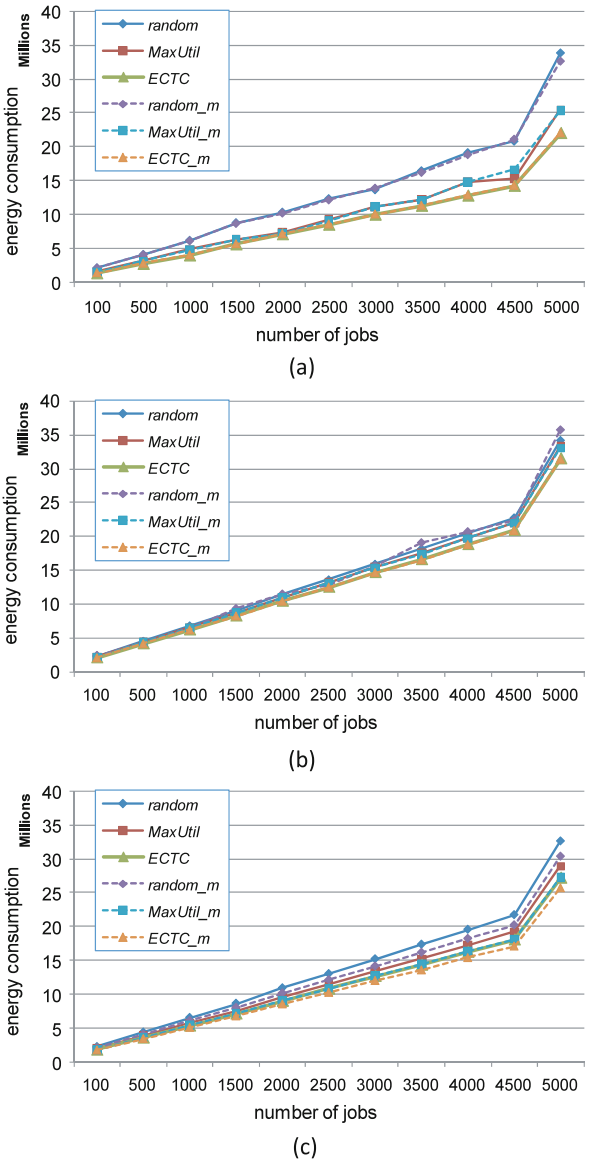

Fig. 5 Energy savings using different task consolidation techniques: (**a**) low resource usage, (**b**) high resource usage, (**c**) random resource usage

tasks likely to hinder the consolidation of new arriving tasks, resulting in more energy consumption compared with the case when migration is not considered.

6 Conclusion

Task consolidation particularly in clouds has become an important approach to streamline resource usage and in turn improve energy efficiency. Based on the fact that resource utilization directly relates to energy consumption, we have successfully modeled their relationship and developed two energy-conscious task consolidation heuristics. The cost functions incorporated into these heuristics effectively capture energy-saving possibilities and their capability has been verified by our evaluation study. The results in this study should not have only a direct impact on the reduction of electricity bills of cloud infrastructure providers, but also imply possible savings (with better resource provisioning) in other operational costs (e.g., rent for floor space). Of course, the reduction in the carbon footprint of clouds is another important spinoff.

Acknowledgements Professor Zomaya's work is supported by an Australian Research Grant DP1097110.

References

1. Parkhill D (1966) The challenge of the computer utility. Addison-Wesley Educational, Reading
2. Koomey JG (2007) Estimating total power consumption by servers in the U.S. and the world. Lawrence Berkeley National Laboratory, Stanford University
3. Koch G (2005) Discovering multi-core: Extending the benefits of Moore's law. Technology@Intel Magazine, (http://www.intel.com/technology/magazine/computing/multi-core-0705.pdf)
4. Barroso L, Holzle U (2007) The case for energy-proportional computing. IEEE Comput
5. Bohrer P, Elnozahy E, Keller T, Kistler M, Lefurgy C, Rajamony R (2002) The case for power management in web servers. Power Aware Comput 261–289
6. Fan X, Weber X-D, Barroso LA (2007) Power provisioning for a warehouse-sized computer. In: Proc 34th annual international symposium on computer architecture (ISCA '07), 2007, pp 13–23
7. Lefurgy C, Wang X, Ware M (2007) Server-level power control. In: Proc IEEE international conference on autonomic computing, Jan 2007
8. Meisner D, Gold BT, Wenisch TF (2009) PowerNap: eliminating server idle power. In: Proc 14th international conference on architectural support for programming languages and operating systems (ASPLOS '09), 2009, pp 205–216
9. Microsoft Inc (2009) Explore the features: performance. http://www.microsoft.com/windows/windows-vista/features/performance.aspx
10. Venkatachalam V, Franz M (2005) Power reduction techniques for microprocessor systems. ACM Comput Surv 37(3):195–237
11. Lee YC, Zomaya AY (2009) Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling. In: Proc the international symposium on cluster computing and the grid (CCGRID '09), 2009, pp 92–99
12. Kim KH, Buyya R, Kim J (2007) Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters. In: Proc seventh IEEE international symposium on cluster computing and the grid (CCGrid '07), 2007, pp 541–548
13. Zhu D, Melhem R, Childers BR (2003) Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems. IEEE Trans Parallel Distrib Syst 14(7):686–700

14. Ge R, Feng X, Cameron KW (2005) Performance-constrained distributed DVS scheduling for scientific applications on power-aware clusters. In: Proc the ACM/IEEE conference on supercomputing (SC '05), 2005, pp 34–44
15. Chen JJ, Kuo TW (2005) Multiprocessor energy-efficient scheduling for real-time tasks with different power characteristics. In: Proc international conference on parallel processing (ICPP '05), 2005, pp 13–20
16. Moore J, Chase J, Ranganathan P, Sharma R (2005) Making scheduling cool: temperature-aware workload placement in data centers. In: Proc USENIX annual technical conference
17. Tang Q, Gupta SK, Varsamopoulos G (2008) Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: a cyber-physical approach. IEEE Trans Parallel Distrib Syst 19(11):1458–1472
18. Srikantaiah S, Kansal A, Zhao F (2008) Energy aware consolidation for cloud computing. In: Proc USENIX workshop on power aware computing and systems in conjunction with OSDI, 2008, pp 1–5
19. Song Y, Zhang Y, Sun Y, Shi W (2009) Utility analysis for internet-oriented server consolidation in VM-based data centers. In: Proc IEEE international conference on cluster computing (Cluster '09), 2009
20. Torres J, Carrera D, Hogan K, Gavalda R, Beltran V, Poggi N (2008) Reducing wasted resources to help achieve green data centers. In: Proc 4th workshop on high-performance, power-aware computing (HPPAC '08), 2008
21. Nathuji R, Schwan K (2007) VirtualPower: coordinated power management in virtualized enterprise systems. In: Proc twenty-first ACM SIGOPS symposium on operating systems principles (SOSP '07), 2007, pp 265–278
22. Kuroda T, Suzuki K, Mita S, Fujita T, Yamane F, Sano F, Chiba A, Watanabe Y, Matsuda K, Maeda T, Sakurai T, Furuyama T (1998) Variable supply–voltage scheme for low–power high–speed CMOS digital design. IEEE J Solid-State Circuits 33(3):454–462
23. Subrata R, Zomaya AY, Landfeldt B (2010) Cooperative power-aware scheduling in grid computing environments. J Parallel Distrib Comput 70(2):84–91
24. Khan SU, Ahmad I (2009) A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids. IEEE Trans Parallel Distrib Syst 21(4):537–553