Cluster Comput (2010) 13: 113–126
DOI 10.1007/s10586-009-0110-2

Energy optimization schemes in cluster with virtual machines

Xiaofei Liao · Liting Hu · Hai Jin

Received: 15 May 2009 / Accepted: 5 November 2009 / Published online: 20 November 2009

Abstract Large scale clusters based on virtualization technologies have been widely used in many areas, including the data center and cloud computing environment. But how to save energy is a big challenge for building a "green cluster" recently. However, previous researches, including local approaches, which focus on saving the energy of the components in a single workstation without a global vision on the whole cluster, and cluster-wide energy saving techniques, which can only be applied to homogeneous workstations and specific applications, cannot solve the challenges. This paper describes the design and implementation of a novel scheme, called Magnet, that uses live migration of virtual machines to transfer load among the nodes on a multi-layer ring-based overlay. This scheme can reduce the power consumption greatly by regarding all the cluster nodes as a whole based on virtualization technologies. And, it can be applied to both the homogeneous and heterogeneous servers. Experimental measurements show that the new method can reduce the power consumption by 74.8% over base at most with certain adjustably acceptable overhead. The effectiveness and performance insights are also analytically verified.

Keywords Virtual machines · Cluster · Energy

X. Liao (✉) · L. Hu · H. Jin
Services Computing Technology and System Lab.,
Cluster and Grid Computing Lab., School of Computer Science
and Technology, Huazhong University of Science
and Technology, Wuhan, China
e-mail: xfliao@hust.edu.cn

H. Jin
e-mail: hjin@hust.edu.cn

1 Introduction

"Green computing" has been a good research topic in cluster computing for many years. Anecdotal evidence from data center operators (e.g. [2, 11]) indicates that a significant fraction of the operation cost of these centers is due to power consumption and cooling.

Great progresses have been made by applying "green computing" to clusters. However, the mainstream computer architecture imposes fundamental limitations on the maximum energy, which a data center can save.

First one is low utilization. Bohrer et al. have studied real webserver workloads from sports, e-commerce, financial, and internet proxy cluster and found that average server utilization varies between 11% and 50% [19]. Low utilization has two causes [3]: to guarantee good performance at periods of peak demands, processing capacity is over-provisioned. Another one is the traditional deployment pattern: one application per OS image and one OS image per PM (Physical Machine). This paradigm makes ad-hoc deployment of applications possible. But, only through consolidation at both the OS level and application level, the maximum power reduction can be achieved.

However, it is a pity that most of the previous local-wide techniques have focused on the improvement at the application level of single workstation without a global vision of the whole system. They save the energy by reducing the clock frequency, the supplied voltage or saving interconnect components including switches, network interface cards (NICs), and links [4, 17].

Second one is lacking the support for heterogeneous nodes. Most of the previous cluster-based techniques [5] have focused on redistributing the request to the smallest (in number of nodes) and then turning off the redundant nodes to save energy. But their work is based on the assumption that all cluster nodes are homogeneous. Actually, real-life server clusters are almost invariably heterogeneous in terms of performance, capacity and platforms [9]. A request served by Linux operation system may be refused if distributed to a PM on which Windows operation system runs.

The third problem follows from the second. It is the poor isolation between co-hosted applications within an OS image [3]. Diverse applications have diverse security requirements. As a given example, the OS patch required by one application to update may be incompatible with the other co-hosted applications.

Although significant progresses have been made on green computing techniques, including local techniques and cluster-wide techniques, it seems to reach the bottleneck already. Fortunately, the emergence of the virtual machine (VM) gives us a new horizon and thus we can look upon the problem at a different angle. A virtual machine is originally defined by Popek and Goldberg as an efficient, isolated duplicate of a real machine. The VM used in our work is System VM (sometimes called hardware VMs), which allows the multiplexing of the underlying physical machine between different VMs, each running its own operating system.

The virtual machine exhibits the unique advantages as follows: first, it allows the separation of hardware and software and thus addresses the problem caused by heterogeneous computing platforms; second, live migration [8, 12] of VMs allows the workload of a node to be transferred to another node.

That is not to say, however, that we can make virtual machines randomly migrate among all nodes. In fact, the potential overhead caused by live migrations of VMs can not be ignored, which may have serious negative effect on cluster utilization, throughput and QoS issues. Therefore the challenge is how to design a migration strategy to effectively implement "green computing" and meanwhile influence little on the performance of the cluster.

Our "green computing" algorithm tends to turn off the redundant nodes to save the energy, provided that the system performance is guaranteed by the left nodes. The reason is that each node in our cluster consumes approximately 160 watts when idle and 280 watts when all resources are stretched to the maximum. It means that: (1) there is a relatively small difference in power consumption between an idle node and a fully utilized node; (2) the penalty for keeping a node powered on is high even if it is idle. Thus, turning a node off always saves power, even if its load has to be transferred to one or more other nodes.

We proposed a policy, called Magnet, to implement "green computing" in the cluster with VMs. Magnet keeps track of all active nodes and organizes them into concentric, non-overlapping rings in terms of gradually decreasing workload, so it is easy to "squeeze" the existing running jobs which are widely distributed among lightweight nodes and then deliver them onto a subset of current active nodes and it is also easy to "release" the overweighted nodes, thereby (1) turning off the redundant nodes to save energy when the system is in non-intensive computing state to obtain energy savings; (2) transferring violating jobs or big jobs to the free nodes when the system is in intensive computing state to obtain performance gains.

By conducting simulations from generated application workload with different intensities in the cluster with VMs, we show that our method can effectively reduce the power consumption by 67.1%, 72.0%, 69.3%, 72.8% and 74.8% at most for five application workload groups (light, moderate, normal, moderately intensive and highly intensive job submission rates). Our method can increase the cluster utilization to 41.9%, 35.44%, 36.57%, 45.17% and 50.61% respectively. Our method can also increase the quality-of-service for the heavyweight workload group. Plus, we show that the total migration overhead is acceptable and adjustable. The effectiveness and performance insights are also demonstrated through a theoretical analysis.

The rest of this paper is organized as follows. Section 2 discusses the related work. The Magnet policy is described in Sect. 3. Section 4 demonstrates the effectiveness of Magnet through theoretical analysis. Section 5 describes our simulation methodology. Section 6 presents the performance evaluation. We conclude this work in Sect. 7.

2 Related works

We divide previous work into two groups: local and cluster-wide technique. Local techniques are implemented independently by each server, whereas cluster-wide techniques involve multiple servers.

2.1 Local techniques

Most of the local techniques aiming at reducing power consumption of a computing cluster focus on the improvement of the single node, by reducing the clock frequency, by reducing the supplied voltage or by saving interconnect components in computer, such as switches, network interface cards (NICs), and links. For instance, the DVS system dynamically reduces processors' supply voltages while guaranteeing proper operations. The DLS [16] project makes use of an appropriate adaptive routing algorithm to shut down links in a judicious way.

Elnozahy [6] proposed a new mechanism called request batching, in which the incoming requests are accumulated in memory by the network interface processor while the host processor of the server is kept in a low-power state. The host processor is awakened when an accumulated request has been pended for longer than a batching timeout.

However, such schemes do not achieve the maximal optimization as experimental results (SPECpower_ssj2008 [21]) confirm that the incremental energy savings from slowing down all CPUs (and scaling down their voltage) are far less than those from turning a machine off to reduce farm capacity by the same amount. Plus, request batching trades off system responsiveness to save energy, so it is not appropriate to trade or e-commerce server in that a very slow server will drive away customers.

2.2 Cluster-wide techniques

We divide cluster-wide techniques into two groups: the physical machine based (PM-based) techniques and virtual machine based (VM-based) techniques.

PM-based technique. Pinheiro et al. [9, 18], Chase et al. [5] and Heath et al. [10] proposed similar strategies for managing energy in the context of front-end server clusters. The basic idea of such approaches is to leverage the aggregate system load and then determine the minimal set of servers which could handle the load. Finally, energy savings can be obtained by adjusting the configuration and request distribution. Kephart et al. [15] coordinated multiple autonomic managers to implement both power and performance objectives. Chen et al. [7] presented the formalization to the dynamic optimization problem of server provisioning and DVS control for multiple applications including a response-time SLA.

VM-based technique. VMware VirtualCenter Distributed Resource Scheduler (DRS) [22] allows a collection of virtualized servers to be managed as a single pool of resources and automates dynamic migrations of VMs to balance load across hosts, enforcing resource allocation policies. Bobroff et al. [3] proposed a runtime VM placement strategy to reduce the amount of required physical capacity and the rate of SLA violations. The basic idea is to utilize the variability in workload to dynamically consolidate VMs.

PM-based techniques have drawbacks including that (1) they focus solely on homogeneous servers; (2) most of these approaches fail to deal with run-time reallocation of the load. VM-based techniques have drawbacks including that (1) they focus on load balance issue rather than power-aware placement issue; (2) they consider little about VM mapping problem under the constraint of multiple dimensions of resource consumption (memory, CPU, network bandwidth and disk bandwidth); (3) they do not explicitly consider the live migration overhead which results in undesirable behaviours such as system instability and thrashing.

3 MAGNET design

To achieve good performance in the cluster, Magnet faces the following challenges: (1) how to design a framework to run in the "green computing" way and meanwhile to influence little on the performance; (2) how to decrease the overhead caused by frequent live migrations of VMs; (3) how to maintain the service continuity and stability (decreasing the impact of interruption caused by the crash down of unknown node). In this section, we discuss the situations which lead to bad performance to the cluster computing first and then introduce the design of Magnet.

Generally the performance of the cluster, such as throughput, average job slowdown and QoS issues are likely to be influenced by two problems, called as inner job blocking problem and outer job blocking problem. The former is caused by certain violating jobs with seriously fluctuant resource requirements which lead to node thrashing. Previous studies (e.g. [24, 25]) focused on balancing the number of jobs/tasks among the workstations, but the CPU or memory requirement should be informed in advance. The latter happens due to the coming of a big job with remarkable working set requirement which can not be satisfied by the current active workstations, resulting in the blocked working flow of rest of the jobs. Towards outer job blocking problem, existing schemes like backfilling scheduling [20] and gang scheduling must consider the size of node needed by a job or estimated runtimes [23].

The above analysis indicates us that if only we could conserve the energy globally and meanwhile flexibly address the job blocking problems, we can obtain a win-win situation, saving energy and improving performance.

3.1 Overview

As illustrated in Fig. 1, the basic workflow of Magnet is as follows. First, the multilayer ring-based overlay is constructed and new jobs arrive continuously and are submitted to the service center. Second, a clearing module is employed to periodically collect the resource demand of each VM. After analyzing that historic record by a sliding window, the irregular VMs will be ruled out. Third, a detector is employed to supervise the "evil" state of the computing cluster

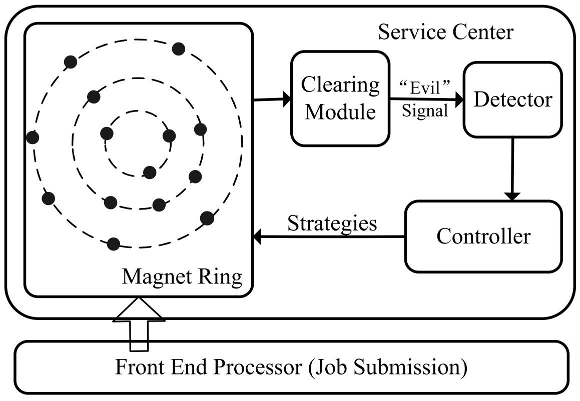


Fig. 1 The system diagram of Magnet

which can be categorized into four kinds: (1) saving energy state caused by the arrived lightweight working flow which persists for a long time; (2) inner job blocking state caused by node thrashing; (3) outer job blocking state caused by violating or big jobs; (4) fault tolerance state caused by certain nodes crashing down due to physical or software malfunctions. Finally, a controller is employed to choose corresponding strategies to response the "evil" states.

3.2 Clearing module

Not all workloads are suitable for dynamic placement, the objective of the clearing module is to rule out the VMs whose workloads are too irregular. There are several properties a regular VM should have: (1) its workload exhibits significant variability; (2) In order to make the remapping of VMs to PMs keep pace with the demand changes, the timescale over which the resource demand varies must exceed the reconfiguration interval τ; (3) resource demand needs to be predictable during the interval, meaning that the error distribution is significantly "narrower" than the distribution of demand.

Note that the workload on the VMs in data center exhibits different features, such as CPU intensive, memory intensive, I/O intensive. The classification of whether a VM is regular depends on whether its dominant feature exhibits regular behaviour or not (e.g. CPU utilization for a CPU intensive VM, memory utilization for a memory intensive VM).

Bobroff [3] proposed a formula to quickly decide whether a given VM is a good candidate, in which the relative gain from using the dynamic placement is quantified:

$$Gain(\tau) \approx 1 - \frac{E[U] + E_p(\tau)}{L_p} \tag{1}$$

Where $E[U]$, $E_p(\tau)$ are the mean of the demand distribution and *p-percentile* computed empirically from the demand history, L_p is the error distribution provided by well-studied forecasting algorithms [6, 13]. p is the maximum probability that the VM demands (exceeding capacity of the PM). We categorize VMs as regular or irregular based on the value of (1) (see more details in [3]).

3.3 Multilayer ring based overlay

Preliminaries. Let *degrad* be the maximum acceptable weight a node could suffer without performance degradation. "*degrad*" can be given by the user expectation, the QoS requirement of the application and the like. Under different situations, *degrad* is different and we can not set a constant value. Let $Max(vm)$ be the maximum number of VM containers a node could have. $Max(vm)$ is also uncertain as it changes in accordance with different configurations of the physical machine. Let *violate* be the weight a node could suffer which leads to unacceptable performance degradation. For example, let *violate* be 89%, then the node which consumes the resource ratio exceeding 89% will be regarded as violating node and should be released.

In the first operation of Magnet, each node boots up one VM and new jobs are submitted to these VMs. Then every VM, with a unique ID, consumes resources of the physical nodes at different rates (from 0% to 100%). Magnet keeps tracks of all active nodes, and organizes them into concentric, non-overlapping rings in accordance with decreasing resource consumed ratios. Magnet maintains 3-*layer* rings, the members of each ring suffer the workload that spans the range $[0, degrad/2)$, $[degrad/2, degrad)$, $[degrad, 100\%)$. Magnet deals with the members of the outer ring by squeezing them to the secondary ring. VMs on the secondary ring are not recommended to be merged together as the sum of their resource consumption rates exceed the performance upper bound (*degrad*).

In the second step, the leader takes the responsibility of maintaining a stable Magnet ring-based overlay for the reason that the memory and CPU demand of jobs may change dynamically and the execution time may not be known in advance. The leader is the node who suffers the maximum load among its ring members of the same layer.

Third, within each ring, the leader periodically updates its logical links with its members. At regular intervals, the leader checks whether the workload of its members is less than the threshold of its layer and if not, it removes the node which does not belong to its layer to the alternative appropriate layer.

Finally, the Magnet system addresses the resilience issue of Magnet ring-based overlay through the introduction of co-leaders. Each leader recruits the co-leader at the time being elected.

To detect unannounced departures, Magnet relies on heartbeats exchanged among leaders and their crews. Unreported nodes are given a fixed time interval before being considered to be dead. If the failure node happens to be the leader, the members of the leader's ring regards co-leader as the replacement leader. In that sense, co-leader improves the resilience of the Magnet ring-based overlay by avoiding dependencies on single node.

3.4 Squeeze measure and release measure

In this section, we analyze the overhead caused by the migrations of VMs and then introduce the "squeeze" and "release" migration measures which decrease the overhead to the maximum extent.

VM migration is the key in the energy saving method of aggregating and redistributing the system load in the cluster consisting of VMs. The most optimal effect can be achieved by (1) calculating the overall load of the system and being divided by the capacity of a single physical machine to

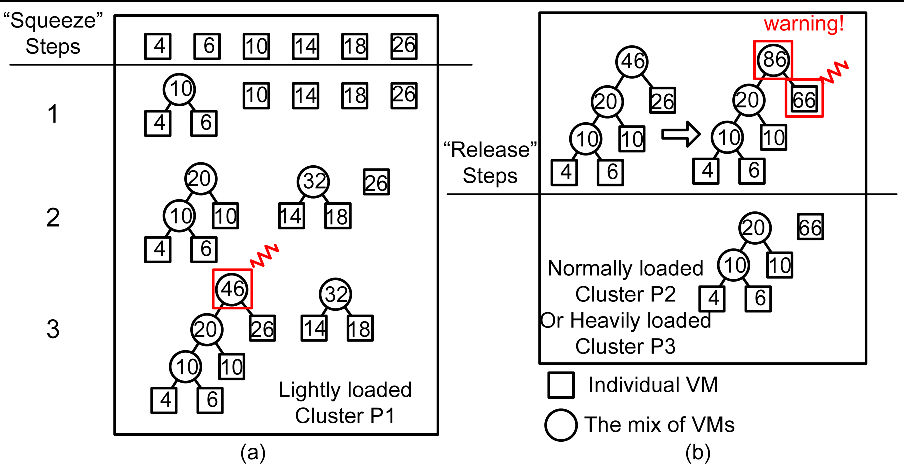

"Squeeze" Steps
1
2
3
Lightly loaded Cluster P1
(a)
"Release" Steps
warning!
Normally loaded Cluster P2
Or Heavily loaded Cluster P3
Individual VM
The mix of VMs
(b)

Fig. 2 (**a**) The "squeeze" steps in detail. (**b**) The "release" steps in detail

get the number of nodes which can handle the overall load; (2) transferring the scattered load to the calculated minimal set of servers by a sequence of live migrations. The challenge is how to obtain the optimal effect meanwhile decrease the overhead caused by VM migrations.

Overhead. Previous works [8, 26] have conducted a series of experiments to measure the overhead for migrating a number of running VMs from one physical host to another. The results show that, the overhead of VM migrations is reflected in two aspects, the time cost for all the migrations and the throughput loss of the competitive VMs on the target node.

First, let r be a fixed remote execution cost in second; let B be the bandwidth; let D be the amount of data in bits (OS image) to be transferred in the job migration; let N be the times of all VM migrations. The time cost for all the migrations can be given by

$$Overhead(Time(s)) = \left(r + \frac{D}{B}\right) \times N \qquad (2)$$

Second, the throughput loss is caused by the VMs' competition for the shared cache although the VMs have been already isolated in terms of CPU and memory. It has been shown that page faults frequently occur in some heavily loaded nodes but a few memory accesses or no memory accesses are requested on some lightly loaded nodes or idle nodes [1]. Therefore, in order to decrease the total throughput loss, it is recommended to merge lightly loaded VMs together on the physical host and avoid the situation of transferring one heavily loaded VM to the host inside which there is another heavily loaded VM running.

"*Squeeze*" *Measure*. The above analysis gives us the directions for minimizing the overhead: the less D, N, and probability of allocating heavily loaded VMs together, the less overhead will be achieved. While in *energy saving* state or *outer job blocking* state, Magnet will take "*squeeze*" measure to migrate a sequence of VMs on the outer layer ring, which is similar to the process of constructing an optimal tree. The difference lies in that the process will stop if the sum of load of VMs on a physical host exceeds the upper bound load of their layer ($degrad/2$). Figure 2(a) presents the detailed "*squeeze*" steps. Let the number inside the VM be the amount of load in terms of a percentage and 33% be the upper bound of the cluster P1 consisting of outer layer ring nodes ($degrad = 66\%$). The physical host which contains 4 VMs (4%, 6%, 10% and 26%) will be removed onto the secondary layer ring. Each step of "*squeeze*" measure merges two lightest loaded VMs together and thus guarantees the minimal D and minimal probability of the mergence of heavily loaded VMs. Moreover, the process of constructing the optimal tree leads a relatively smaller value of N.

"*Release*" *Measure*. While in *inner job blocking* state, Magnet will take "*release*" measure to the VM whose host machine consumes the resource ratio exceeding the upper bound (violate). "*Release*" measure is the inverse process of "*Squeeze*" measure and the disjointed violating VMs will be transferred to the free nodes on the lightly loaded nodes of the outer layer ring. Figure 2(b) presents the detailed "release" steps, in which the VM (26%) mutates to be a violating VM (66%) due to some unknown reasons, e.g. its jobs' fluctuant demand of a large memory space. Consequently, the physical machine of the VM (66%) becomes a *violating* node (suppose violate is 80%) and then it is released.

3.5 The rationale of our solutions

In this section, we separately discuss our solutions to the four common "*evil*" system states and the rationale behinds them.

In saving energy state, Magnet virtually reconfigures the cluster system to further utilize resources by taking "*squeeze*" measure on outer layer nodes (see Fig. 3). At

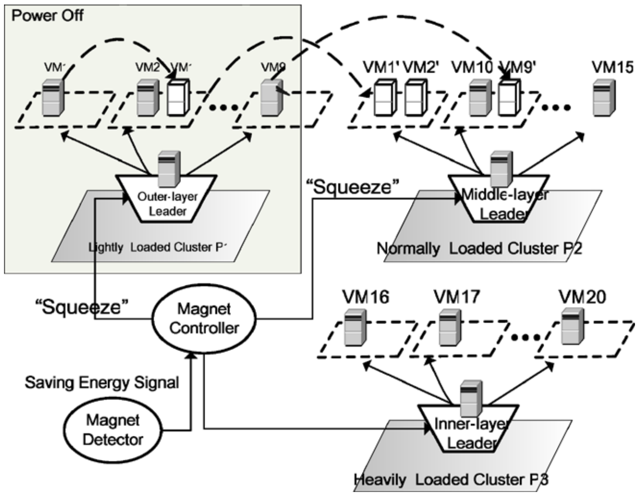


Fig. 3 Magnet reconfiguration in saving energy state

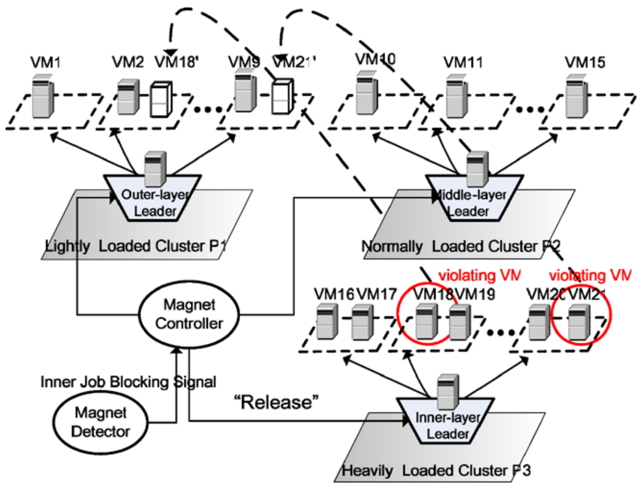


Fig. 4 Magnet reconfiguration in inner job blocking state

next step, Magnet switches the high-power state of the released nodes to the low-power state. There is one question that why we do not submit these lightweight jobs to certain nodes from the very beginning and thus save trouble and overhead of all these migrations. Actually, a situation is ignored, that jobs on cluster may change dynamically, heavily loaded nodes could become lightly loaded when its jobs are completed or terminated that we do not know in advance. Therefore, saving energy is not so much a predetermined strategy as a feedback-driven process.

In *inner job blocking* state, it is pressing to make the violating job migrate to the node which can provide it enough memory space or CPU resource. More importantly, experiments have shown that a large job is likely to be with long lifetime [6, 14]. In the sense, the candidate node should not be a heavyweight node as it is likely to be the same for a long time and thus causes another *inner job blocking* state.

The Magnet system takes "*release*" measure on the violating VMs. Magnet maintains a multilayer ring-based overlay among which the lightweight workload nodes are organized to be on the outer layer ring, so it is easy to find the candidate nodes by requesting the corresponding outer layer leader for the list of its members and then choosing one of them, as illustrated in Fig. 4.

In *outer job blocking* state, a new job is coming, which demands large memory space and CPU resource. Unfortunately, the available space of each individual node is not large enough to serve it and thus the following submissions to the workstation will be blocked. However, if the idle memory spaces of all individual nodes can be accumulated, then the sum may fit the large job. The Magnet system takes "*squeeze*" measure on outer layer nodes to release the resources.

More importantly, our policy can provide both the centralized and distributed job scheduling approaches in the cluster. Previous FEP (Front End Process) emphasizes a lot on the characterization of jobs by adjusting its scheduling strategies in accordance with job types, such as rigid jobs and moldable jobs. Undoubtedly, the centralized management has many advantages. However, when the scalability is restricted or high fault tolerance is required, distributed management is an alternative scheme of great importance. Our approach can transfer the responsibilities of the FEP to the workstations in cluster, making the workstations themselves decide which node to run certain task. In other words, the computing cluster acts like a "*black box*" which is transparent to the user and the FEP.

4 Analysis

4.1 Energy saving modeling

Preliminaries: Let V be the set of all workstations; let t be the given interval for Magnet reconfiguration; let $S_M(t) \subset V$ be the set of the active workstations during t; let ε be the average watt of electricity consumed by one workstation per second; let $\Delta E = (|V| - |SM|) \times \varepsilon \times t$, representing the energy saved during t.

Assuming that node N_i follows Poisson distribution to be required of K_i percentage of resource ($0 < K_i < 100$) with the mean rate λ_i (intensity of workload) [25], hence the probability density of resource consumption of N_i can be calculated as $P(X_i = k_i) = e^{-\lambda_i}\lambda_i^{k_i}/k_i!$.

As the sum of N independent Poisson distributions still follows Poisson distribution, at time t, the sum of workload on all the workstations follows Poisson distribution. Suppose that the *mean* workload of the outer layer ring, the secondary ring and the inner layer ring is $\overline{load_1} \in [0, k_1)$, $\overline{load_2} \in [k_1, k_2)$ and $\overline{load_3} \in [k_2, 100]$, respectively.

Assuming the percentage of workstations suffering $\overline{load_1}$, $\overline{load_2}$, $\overline{load_3}$ is P_1, P_2, and P_3 respectively, so there are $|V|P_1$, $|V|P_2$, $|V|P_3$ workstations on the outer ring, the secondary ring, and the inner ring, where

$$P_1 = \int_0^{k1} \frac{e^{-\lambda}\lambda^r}{r!} dr, \qquad P_2 = \int_{k1}^{k2} \frac{e^{-\lambda}\lambda^r}{r!} dr, \qquad P_3 = \int_{k2}^{100} \frac{e^{-\lambda}\lambda^r}{r!} dr \tag{3}$$

and $P_1 + P_2 + P_3 = 1$

Before reconfiguration: $|SM| = |V|$.

After reconfiguration:

The load on the outer ring and the secondary ring will be partially "squeezed" or "transferred". Magnet only deals with the members of the outer ring by squeezing them to the secondary ring, then

$$|SM| = |V|P1\frac{\overline{Load_1}}{\overline{Load_2}} + |V|P_2 + |V|P_3 \tag{4}$$

and then

$$\begin{aligned} |V| - |SM| &= |V| - |V|P_1\frac{\overline{Load_1}}{\overline{Load_2}} + |V|P_2 + |V|P_3 \\ &= |V|P_1\left(1 - \frac{\overline{Load_1}}{\overline{Load_2}}\right) \end{aligned} \tag{5}$$

Thus, the saved energy can be given by

$$\begin{aligned} \Delta E &= (|V| - |SM|) \times \varepsilon \times t \\ &= |V|P_1\left(1 - \frac{\overline{Load_1}}{\overline{Load_2}}\right) \times \varepsilon \times t \\ &= |V|\left(\int_0^{k1} \frac{e^{-\lambda}\lambda^r}{r!} dr\right)\left(1 - \frac{\overline{Load_1}}{\overline{Load_2}}\right) \times \varepsilon \times t \end{aligned} \tag{6}$$

We have $\overline{load_1} < \overline{load_2}$, so $\Delta E > 0$.

The above model gives conditions for the Magnet reconfiguration to reduce the total electricity. A key condition for performance gains is from k_1, λ and variance between $\overline{load_1}$ and $\overline{load_2}$. The more k_1, the less and the larger difference between $\overline{load_1}$ and $\overline{load_2}$, the more energy will be saved.

4.2 Quality of service modeling

The total execution time of job i in a workload for $i = 1, 2, \ldots, n$, $t_{exe}(i)$ is expressed as $t_{exe}(i) = t_{cpu}(i) + t_{page}(i) + t_{que}(i)$, where $t_{cpu}(i)$, $t_{page}(i)$, $t_{que}(i)$ are the CPU service time, the paging time for page faults, the queuing time waiting in a job queue.

$$\begin{aligned} T_{exe} &= \sum_{i=1}^{n} t_{cpu}(i) + \sum_{i=1}^{n} t_{page}(i) + \sum_{i=1}^{n} t_{que}(i) \\ T_{exe} &= T_{cpu} + T_{page} + T_{que} \end{aligned} \tag{7}$$

After Magnet reconfiguration, with (2), we have

$$\begin{aligned} \hat{T}_{exe} &= \hat{T}_{cpu} + \hat{T}_{page} + \hat{T}_{que} + \hat{T}_{mig} \\ \hat{T}_{exe} &= \hat{T}_{cpu} + \hat{T}_{page} + \hat{T}_{que} + \left(r + \frac{D}{B}\right) \times N \end{aligned} \tag{8}$$

The jobs demand identical CPU service on both cluster environments, so that $T_{cpu} = \hat{T}_{cpu}$.

For inner job blocking problem, the paging time reduction ($T_{page} - \hat{T}_{page}$) can be achieved by making jobs with large memory demands migrate to the nodes of the outer ring which has enough resources. The *inner job blocking* problem happens during the running process of the system, so that $\hat{T}_{que} = T_{que}$. In that sense,

$$\begin{aligned} \Delta T &= T_{exe} - \hat{T}_{exe} = (T_{page} - \hat{T}_{page}) - \hat{T}_{mig} \\ \Delta T &= (T_{page} - \hat{T}_{page}) - \left(r + \frac{D}{B}\right) \times N \end{aligned} \tag{9}$$

For *outer job blocking* problem, we can conclude that $\hat{T}_{que} > T_{que}$ as Magnet helps keep the workflow smooth. It is under the assumption that the total resource can satisfy the jobs, so $\hat{T}_{page} = T_{page}$. Then we have

$$\begin{aligned} \Delta T &= T_{exe} - \hat{T}_{exe} = (T_{que} - \hat{T}_{que}) - \hat{T}_{mig} \\ \Delta T &= (T_{que} - \hat{T}_{que}) - \left(r + \frac{D}{B}\right) \times N \end{aligned} \tag{10}$$

The above model gives conditions for the Magnet reconfiguration to reduce the total execution time of jobs. Equation (9) and (10) tell us that the more Magnet reconfigures the nodes to maintain the stable overlay, the more difference of T_{page} and $\hat{T}_{page}$ or T_{que} and $\hat{T}_{que}$ will be obtained. However, note that the migration times N will increase too, so ΔT is not always positive. Certainly, less D (data amount) and larger B (bandwidth) will lead to smaller time cost for migrations.

5 Experimental environment

5.1 A simulated cluster with VMs

We have simulated a cluster with 64 homogeneous hosts, each of which has an AMD Athlon 3500+ processor and 1 GB DDR RAM. Storage is accessed via iSCSI protocol from a NetApp F840 network attached storage server (NAS). Moreover, each host has an Intel Pro/1000 NIC to transfer the images of the VMs with 1000 Mbps network bandwidth. We used Xen 3.10 as the virtual machine monitor on each host in all cases and the host kernel for XenLinux is a modified version of Linux 2.6.18.

Table 1 Execution performance of the seven applications

Program	Data size	Lifetime (s)	CPU time (s)	Working set (MB)	I/O
bit-r	2^{23}	192.26	191.85	64.22	0.06
m-sort	2^{23}	82.76	0.84	64.27	0.27
m-m	$1{,}700^2$	4902.29	4901.12	66.37	0.0
t-sim	31,061	41.63	38.79	4.64	6.5
metis	1 MB∼4 MB	124.41	112.85	1.37 4.30	2.8
r-sphere	150,000	318.64	298.18	36.84 39.66	6.1
r-wing	500,000	72.28	28.98	19.53 23.39	58.38

In simulations of the cluster, the virtual machine is configured to use 512 MB of RAM, the memory page size is 4 Kbytes, page fault service time is 10 ms, and the context switch time is 0.1 ms. The remote submission/execution cost, r, is 0.01 second for 1000 Mbps network. Each host maintains a dynamically changed load index file which contains CPU, memory, and I/O load status information. The Magnet system periodically collects the load information among the workstations.

5.2 Application workload

In order to effectively conduct Magnet policy with unknown CPU or memory demands, in this section we need to select different benchmark programs which are representing different types of jobs and then we mix them together to generate the application workload at different submission rates.

The large scientific and system programs we use are from [6], and they are representative CPU-intensive, memory-intensive, and/or I/O-active jobs: bit-reversals (bit-r), merge-sort (m-sort), matrix multiplication (m-m), a trace-driven simulation (t-sim), partitioning meshes (metis), cell-projection volume rendering for a sphere (r-sphere), and cell-projection volume rendering for flow of an aircraft wing (r-wing). Chen [6] have measured the execution performances of each program and monitored their memory performance in a dedicated computing environment. Table 1 [6] presents the results of all the seven programs, where the "data size" is the number of entries of the input data, the "working set" gives a range of the memory space demand during the execution, the "lifetime" is the total execution time of each program.

SPECpower_ssj2008 [21] shows the relationship between the workload and the power consumptions (Table 2), which have been formalized by us to calculate the power consumption under certain target load. The results are applied as "input file".

The application workload consisting of different types of jobs is randomly submitted into the cluster. Each job has a header item recording the submission time, the job ID, and its lifetime measure in the dedicated environment. Following the header item, the execution activities of the jobs are recorded in a time interval of every 100 ms including CPU cycles, the memory allocation demand and the details of its VM migrations including its VM container ID, the source and destination node, the start and the end time. Thus, the power consumption can be calculated by the closely monitored CPU and memory utilization rates with Table 2. Meanwhile, the total execution time, the average slowdown and the time cost of migrations can be given by the logs.

Table 2 Benchmark result summary

Target Load	Actual Load	ssj_ops	Average Power
100%	99.2%	40,852	336
90%	89.1%	36,677	308
80%	80.7%	33,235	288
70%	69.0%	28,398	263
60%	58.7%	24,157	241
50%	49.8%	20,512	225
40%	39.5%	16,281	207
30%	30.0%	12,337	194
20%	20.0%	8,237	181
10%	10.1%	4,142	170
Active Idle	Active Idle	0	159

5.3 Job submission rate generation

In order to implement our policy across a broad range of workload intensities, we have conducted our experiment at different submission rates.

Similar to [8], we have also generated the job submission rates by the lognormal function:

$$R_{in}(t) = \begin{cases} \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(\ln t-\mu)^2}{2\sigma^2}} & t > 0 \\ 0 & t \leq 0 \end{cases} \quad (11)$$

Where $R_{in}(t)$ is the lognormal arrival rate function, t is the time duration for job submissions in a unit of seconds, and the values of μ and σ adjust the degree of the submission rate. The lognormal job submission rate has been observed in several practical studies (see [6, 11]). Five application workload groups with different arrival rates are

Table 3 Job submission rates of the application workload

Application Workload	σ	μ	Amount of Jobs	Submission Duration (s)
APP-1	3.9	3.9	318	5,499
APP-2	4.1	4.1	450	5,503
APP-3	4.3	4.3	565	5,497
APP-4	4.5	4.5	707	5,510
APP-5	4.6	4.6	993	5,498

illustrated in Table 3, where "APP-1", "APP-2", "APP-3", "APP-4", "APP-5" represents light, moderate, normal, moderately intensive, and highly intensive submission rate, respectively, "Submission Duration" is the time duration for job submissions in a unit of seconds.

5.4 Migration cost estimation

The approach we proposed is a continual optimization approach, where we dynamically make the VMs migrate from one physical server to another in order to minimize the total power consumption. The migration process between the two hosts involves the following stages: pre-migration, reservation, iterative pre-copy, stop-and-copy, commitment, and activation [8], which requires creation of a checkpoint on secondary storage and retrieval of the VM image on the target server, so applications can continue running during the migration. However, the performance of applications may be influenced in the transition because of cache misses (hardware caches are not migrated) and potential application quiescence. Thus, it is necessary to estimate the following cost: (1) time cost for one-time migration of the VM on which the benchmark program *r-wing* runs; (2) performance cost of *r-wing* in terms of running time for 30 operations while keeping the memory footprint of background *fma* low; (3) performance cost of *r-wing* in terms of running time for 30 operations while increasing the memory footprint of background *fma*.

Generally speaking, the time cost for the one-time VM migration contains the shutdown delay, the migration duration and the startup delay. As the VM is shut down after being migrated, the shutdown delay is irrelevant to the execution time of jobs running on the VM. As the booting of a new VM is informed in advance, the startup delay is irrelevant to execution time of jobs running on the VM neither. Therefore, we disregard the shutdown delay and startup delay of the 512M VM in our simulation.

We have measured the time cost and performance cost for the live migration of a VM with 512 MB of RAM, as illustrated in Fig. 5. The physical machine is a 3.6 GHz Pentium PC with 1 GB main memory and a swap space of 1 GB, running Linux version 2.6.9. The foreground load is *r-wing* as described in Table 2. The background load is a compute-intensive application namely *fma* from an HPC suite. We

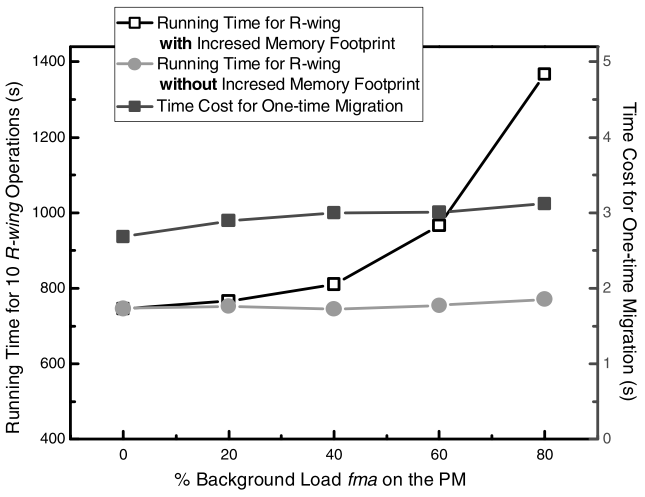


Fig. 5 Migration cost in second of a 512M VM at a time while under different background load

vary the intensity of the background load *fma* and measure the performance with and without an increased memory footprint.

We observe that migrations time is independent of the background load and depended only on the VM characteristics. However, it is based on the premise that the network is idle. Once the task execution environment is communication intensive, the bandwidth will be partially occupied and thus the result is not likely the same.

We also observe that the throughput of the foreground *r-wing* decreases with increasing of the background traffic (memory footprint). It means that although the applications are isolated in terms of CPU and memory, they still compete for the shared cache and thus the performance degrades because of many cache misses. However, the performance degradation can be limited to a certain range as long as the background traffic is not intensive.

6 Performance evaluations

6.1 Measured metrics and reconfiguration parameters

To better evaluate the performance of Magnet, we use the metrics as follows: (1) *Power savings* over base is defined as the ratio of the saved electricity to the total electricity during the entire lifetime of the application workload in percentage terms. (2) *Cluster utilization* is defined as the average ratio between the amounts of consumed memory volume to all memory space of active workstations (rule out the "shut down" nodes) during the entire lifetime of the application workload in percentage terms. (3) *Total execution time* is defined as the sum of the total CPU service time, the total paging time for page faults, the total queuing time waiting in a job queue and the total migration time. (4) *Job slowdown* is defined as the average ratio between its wall-clock execution time and its CPU execution time of all nodes. Plus, we also use the metric (5) MT/ET which is defined as the average ratio between the cumulative Magnet reconfiguration time and its total execution time to evaluate the *overhead* of Magnet.

For our experiment, we refer to the time interval between reconfigurations as the *elapse* parameter. Let $degrad = 80\%$ and $Max(vm) = 3$. The workload of each layer ranges in (0%∼40%), (40%∼80%), (80%∼100%), respectively, of the base resource. To guarantee the QoS of the tasks, the number of the active nodes should not be less than one third of the number of the "shut down" nodes and for our experiment, the threshold is eight. The threshold changes according to the different QoS requirements of the services.

Finally, towards each metric, we compare some of the following policies: (1) basic method without any virtual reconfiguration (Base); (2) current methods (Combined DVS and Batching, Disk Intense Batching and Disk Intense DVS [9]); (3) Magnet method with expected results deduced by mathematical analysis (Expected-Magnet); (4) Magnet method with practical results (Magnet).

The Expected-Magnet results can be calculated with the following formula:

$$\Delta E = |V|\left(\int_0^{k1} \frac{e^{-\lambda}\lambda^r}{r!} dr\right)\left(1 - \frac{\overline{Load_1}}{\overline{Load_2}}\right) \times \varepsilon \times t,$$

where $|V| = 64$, $\frac{\overline{Load_1}}{\overline{Load_2}} = (1 + 0.4)/(1 + 0.8)$. Since the selected application workload tends to be lightweight, $\int_0^{k1} \frac{e^{-\lambda}\lambda^r}{r!} dr$ is set to 1, and ε is set to 170 watts (average power from SPECpower_ssj2008).

6.2 Improving power consumption

Figure 6 presents the energy savings (in percentage) for the Magnet policy under five application workloads with an increasing *elapse* parameter, 250 seconds, 500 seconds and 1000 seconds. Compared to Combined (DVS+Batching), Disk-intensive-DVS and Disk-intensive-Batching methods, the Magnet policy exhibites much more power savings.

From Fig. 6(a), it can be seen that Magnet method, under the reconfiguration *elapse* of 250 seconds, significantly reduces power consumptions. The figure shows that power consumptions are reduced by 67.09%, 72.02%, 67.55%, 72.77% and 74.81% for light, moderate, normal, moderately intensive and highly intensive job submissions, respectively (APP-1,2,3,4,5).

From Fig. 6(b) and Fig. 6(c), regarding an increasingly Magnet reconfiguration *elapse* (500 seconds and 1000 seconds), the power consumptions are reduced by 67.36%, 69.53%, 69.28%, 70.79%, 70.67% and 61.81%, 63.69%, 63.68%, 69.90%, 68.28%. Note that when *elapse* is 1000 seconds, Magnet performs worse than that of 250 seconds and 500 seconds in the energy saving. It suggests that Magnet performs better while its ring-based overlay is maintained more frequently, for the reason that the less interval time between Magnet reconfigurations, the more redundant workstations can switch to "shut down" state. Meanwhile, from Fig. 6(a), Fig. 6(b) and Fig. 6(c), we can see that the less elapse is, the closer the practical Magnet results come to the theoretical Magnet results.

6.3 Improving cluster utilization

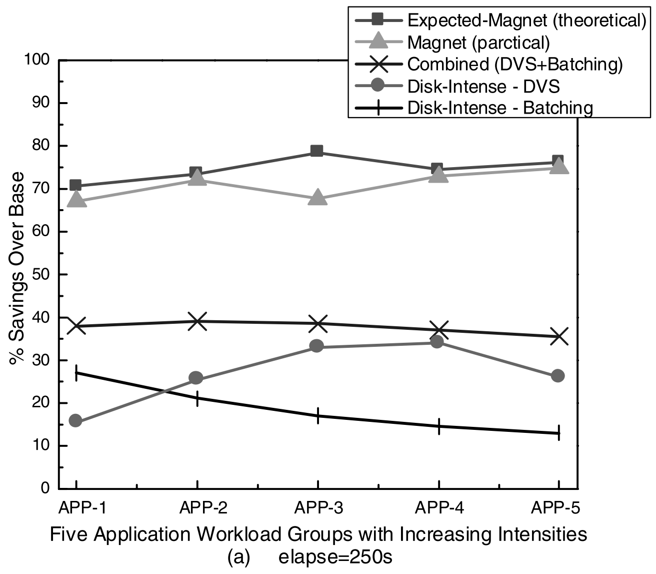
We have also observed the average total consumed memory volumes during the lifetime of job executions in each workload group. Figure 7 presents the comparative average cluster utilization during lifetimes of five workload groups using Magnet scheme and basic scheme.

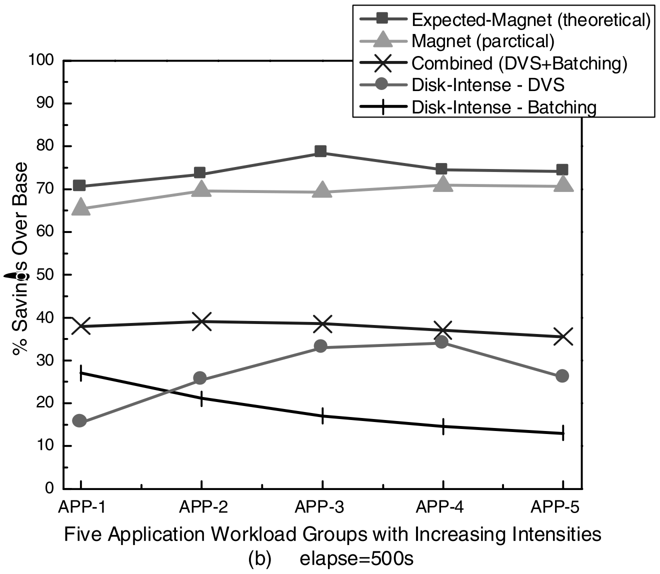
Compared to the original cluster utilization (10.24%, 14.73%, 18.75%, 20.20% and 21.20% for workload APP-1,2,3,4,5 respectively), our method can increase the average cluster utilization significantly. When *elapse* is 250 seconds, the average cluster utilization is increased by 41.89%, 32.04%, 36.57%, 45.17% and 50.61% respectively; when *elapse* is 500 seconds, it is increased by 33.72%, 35.44%, 32.36%, 34.40% and 39.68% respectively; when *elapse* is 1000 seconds, it is increased by 29.59%, 27.05%, 24.42%, 36.10% and 34.97%, respectively.

The increase of the average cluster utilization is caused mainly by the decreasing number of the idle nodes. By means of turning off the idle nodes, the overall workload can be "squeezed" onto a subset of active workstations and thus increase the throughput. Comparing bars of different elapse parameters (250 seconds, 500 seconds and 1000 seconds), it is clear that less value of elapse parameter leads to further utilization of active workstations.

6.4 Improving quality of service

Figure 8 and Fig. 9 present the comparative total execution time and job slowdown under five workloads using Magnet scheme and basic scheme with respect to an increasing value





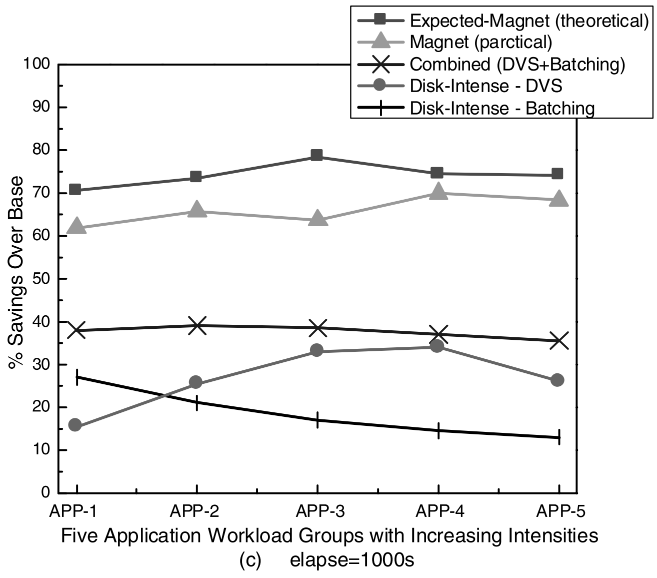


Fig. 6 Energy savings for Magnet, Expected-Magnet and current methods (**a**) *elapse* = 250 seconds, (**b**) *elapse* = 500 seconds, (**c**) *elapse* = 1000 seconds

of *elapse* parameter, from 250 seconds to 1000 seconds. From Fig. 8, it can be seen that when *elapse* is 250 seconds, the total execution time is reduced by −37.10%, −17.83%, −27.66%, 3.47% and 4.96% for workload APP-1,2,3,4,5, respectively; when *elapse* is 500 seconds, it is reduced by −40.72%, −21.65%, −5.62%, 0.30% and 5.10%; when *elapse* is 1000 seconds, it is reduced by −50.0%, −10.51%, 15.52%, 3.98% and −9.43%.

Note that the results are not positive for light job submissions, moderate job submissions and normal job submissions (APP-1,2,3). It is not surprising since job blocking problems happen under light workload and thus the time increased by live migrations exceeds the time reduced by addressing job blocking problems.

Figure 9 shows that Magnet generally decreases the average job slowdown under workload APP-1,2,3,4,5. When *elapse* is 250 seconds, we are able to reduce the average job slowdown by −43.82%, 17.61%, 15.35%, 22.85% and 38.21%; when *elapse* is 500 seconds, it is reduced by −6.74%, 17.39%, 46.93%, 31.84% and 31.79%; when *elapse* was 1000 seconds, it is reduced by 2.00%, 32.98%, 52.28%, 48.4% and 39.62%. Note that the results are not positive for light job submissions when elapse is 250 seconds and 500 seconds, the reason is that frequent migrations lead to longer waiting time addressed much less job blocking problems for lightweight working flow. Therefore, the time increased by live migrations exceeds the time reduced by addressing job blocking problems.

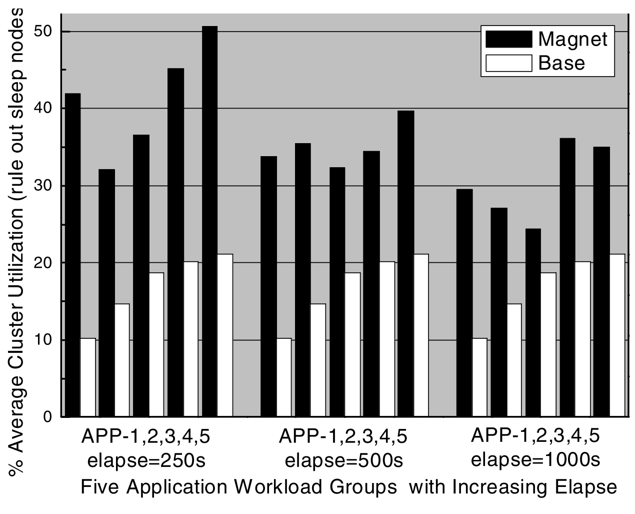


Fig. 7 The average cluster utilizations of the five application workload groups scheduled by Magnet scheme and the basic scheme (Base) with increasing elapses

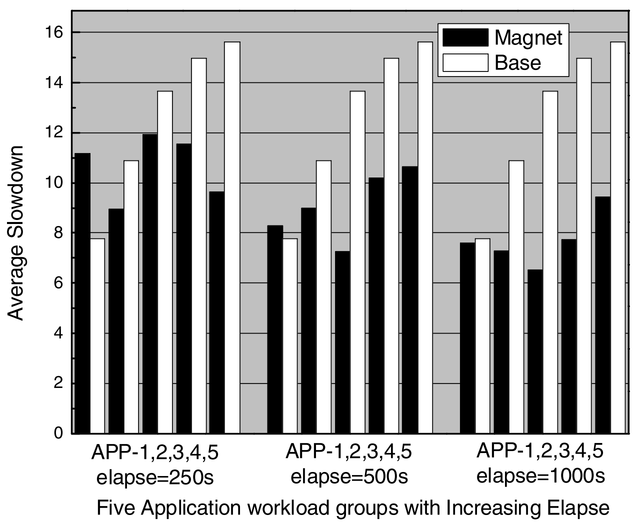


Fig. 9 The average slowdowns of the five application workload groups scheduled by Magnet scheme and the basic scheme (Base) with increasing elapses

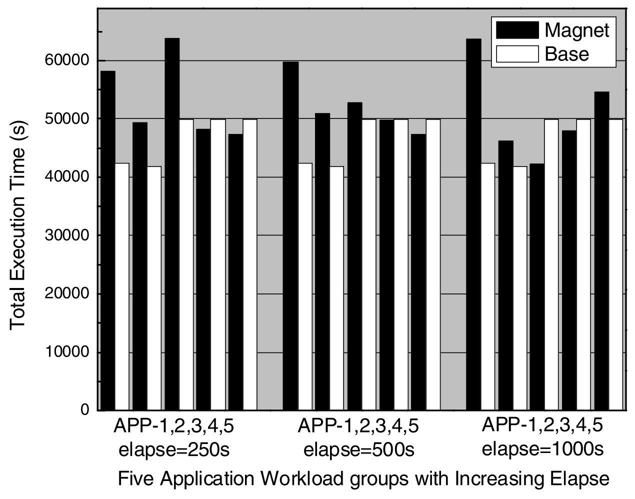


Fig. 8 The total execution times of the five application workload groups scheduled by Magnet scheme and the basic scheme (Base) with increasing elapses

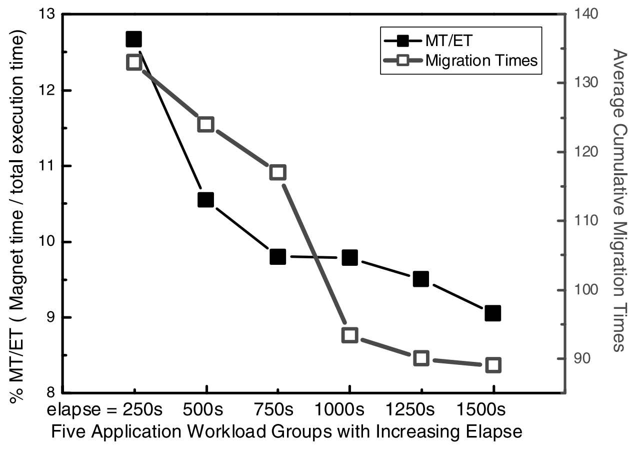


Fig. 10 MT/ET in percentage terms and the average cumulative migration times during the entire lifetimes for APP-1,2,3,4,5 with increasing elapses

6.5 Overhead analysis

As frequent Magnet reconfiguration will cause noticeable overhead, it is important to make sure that the QoS is not sacrificed excessively in favor of power and energy savings.

Finally, we test the MT/ET (Magnet reconfiguration time/Total execution time) and the average cumulative migration times during the entire lifetimes of the five different working flows (see Fig. 10) while increasing the interval between reconfigurations (*elapse*) gradually. MT can be estimated as the product of the migration cost in second and the total times of migrations. It shows that the increase of elapse leads to the decrease of MT/ET, indicating that although smaller value of elapse parameter achieves better performance on energy saving, it is at the expanse of more overhead on the total execution time. However, considering the benefits (more energy savings and cluster utilization) carried by high frequency (see Fig. 6, Fig. 7), it seems that the most optimal approach is a balanced one that an appropriate value of *elapse* parameter should be chosen.

In the Magnet scheme, we can collect the states of all nodes in the cluster by central manger or in distributed manner. In distributed manner, one responsible node in the cluster will be elected by voting algorithm dynamically. Then the voted node will play the same roles of the central manger. In our further research, we will focus on the distributed algorithm and give more detail experiments.

7 Conclusions

This paper aims at providing effective strategies to reduce the power consumption and meanwhile influence little on the performance. The contributions can be described as follows: (1) our scheme addresses the limitations caused by heterogeneous computing platforms; (2) an adaptive Magnet approach is proposed to obtain significant energy savings by taking the advantage of live migration of VMs; (3) through the theoretical analysis, we propose the "*squeeze*" and "*release*" measures to guide the live migrations aiming at the minimal overhead. Experimental results show that the method have positive impact on the average job slowdown and minor negative impact on the total execution time. Particularly, the overhead is adjustable by changing the parameter elapse.

In the future, we will try to optimize the power reduction effect by exploring more intelligent schemes according to the characteristics of jobs such as CPU intensive, memory intensive or I/O intensive. Also, we will analyze the strategies of the migration of multiple VMs, e.g. parallel migration and serial migration, to further reduce the impact of VM migration on the system performance.

Acknowledgements This work is supported by National 973 Basic Research Program of China under grant No. 2007CB310900, Program for New Century Excellent Talents in University under grant NCET-08-0218, China National Natural Science Fundation (NSFC) under grant No. 60703050,60973133, Wuhan Chenguang Research Plan Foundation with No. 200850731350 and the Ministry of Education-Intel Information Technology Special Research Fundation under grant No. MOE-INTEL-08-06.

References

1. Acharya, A., Setia, S.: Availability and utility of idle memory in workstation clusters. In: Proceedings of ACM SIGMETRICS Conference on Measuring and Modeling of Computer Systems, pp. 35–46 (1999)
2. APC-American Power Conversion. Determining Total Cost of Ownership for Data Center and Network Room Infrastructure. ftp://www.apcmedia.com/salestools/CMRP-5T9PQG_R2_EN.pdf (2003)
3. Bobroff, N., Kochut, A., Beaty, K.A.: Dynamic placement of virtual machines for managing SLA violations. In: Proceedings of 9th IFIP/IEEE International Symposium on Integrated Network Management, pp. 119–128. IEEE, New York (2007)
4. Burd, T., Pering, T., Stratakos, A., Brodersen, R.: A dynamic voltage scaled microprocessor system. In: Proceedings of IEEE International Conference on Solid-State Circuits, pp. 294–295 (2000)
5. Chase, J., Anderson, D., Thackar, P., Vahdat, A., Boyle, R.: Managing energy and server resources in hosting centers. In: Proceedings of the 18th Symposium on Operating Systems Principles (2001)
6. Chen, S., Xiao, L., Zhang, X.: Adaptive and virtual reconfigurations for effective dynamic job scheduling in cluster systems. In: Proceedings of the 22nd International Conference on Distributed Computing and Systems (ICDCS 2002) (2002)
7. Chen, Y., Das, A., Qin, W., Sivasubramaniam, A., Wang, Q., Gautam, N.: Managing server energy and operational costs in hosting centers. In: Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, Banff, Alberta, Canada (2005)
8. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I., Warfield, A.: Live migration of virtual machines. In: Proceedings of 2nd Symposium on Networked Systems Design and Implementation (NSDI 2005), USENIX (2005)
9. Elnozahy, E.N., Kistler, M., Rajamony, R.: Energy conservation policies for Web servers. In: Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (2003)
10. Heath, T., Diniz, B., Carrera, E.V., Meira, W. Jr., Bianchini, R.: Energy conservation in heterogeneous server clusters. In: Proceedings of the Tenth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP 2005), pp. 186–195. ACM, New York (2005)
11. Hopkins, M.: The onsite energy generation option. Data Center J. (2004). http://datacenterjournal.com/News/Article.asp?article_id=66
12. Huang, W., Gao, Q., Liu, J., Panda, D.K.: High performance virtual machine migration with RDMA over modern interconnects. In: Proceedings of IEEE International Conference on Cluster Computing (Cluster 2007), September (2007)
13. Jenkins, G., Reinsel, G., Box, G.: Time Series Analysis: Forecasting and Control. Prentice-Hall, New York (1994)
14. Jiang, S., Zhang, X.: TPF: a system thrashing protection facility in Linux. Softw. Pract. Exp. **32**(3), 295–318 (2002)
15. Kephart, J.O., Chan, H., Das, R., Levine, D.W., Tesauro, G., Rawson, F., Lefurgy, C.: Coordinating multiple autonomic managers to achieve specified power-performance tradeoffs. In: Proceedings of the Fourth International Conference on Autonomic Computing (ICAC-4), p. 24. IEEE Computer Society, Washington (2007)
16. Kim, E.J., Yum, K.H., Link, G.M., Vijaykrishnan, N., Kandemir, M., Irwin, M.J., Yousif, M., Das, C.R.: Energy optimization techniques in cluster interconnects. In: Proceedings of International Symposium on Low Power Electronics and Design (ISLPED 2003), pp. 459–464. ACM, New York (2003)
17. Kim, E.J., Yum, K.H., Link, G.M., Vijaykrishnan, N., Kandemir, M., Irwin, M.J., Yousif, M., Das, C.R.: A holistic approach to designing energy-efficient cluster interconnects. IEEE Trans. Comput. **54**(6) (2005)
18. Pinheiro, E., Bianchini, R., Carrera, E., Heath, T.: Dynamic cluster reconfiguration for power and performance. In: Benini, L., Kandemir, M., Ramanujam, J. (eds.) Compilers and Operating Systems for Low Power. Kluwer Academic, Dordrecht (2003)
19. Shafi, H., Bohrer, P.J., Phelan, J.: Design and validation of a performance and power simulator for PowerPC systems. IBM J. Res. Develop. **47**, 641–652 (2003)
20. Shmueli, E., Feitelson, D.G.: Backfilling with look ahead to optimize the performance of parallel job scheduling. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) Job Scheduling Strategies for Parallel Processing. Lecture Notes on Computer Science, vol. 2862, pp. 228–251. Springer, Berlin (2003)
21. Standard Performance Evaluation Corporation. [Online]. Available: www.spec.org (2008)
22. VMware Distributed Resource Scheduler. [Online]. Available: http://www.vmware.com/products/vi/vc/drs.html (2008)
23. Wiseman, Y., Feitelson, D.G.: Paired gang scheduling. IEEE Trans. Parallel Distrib. Syst. **14**(6), 581–592 (2003)
24. Xiao, L., Zhang, X., Kubricht, S.A.: Incorporating job migration and network RAM to share cluster memory resources. In:

Proceedings of the 9th IEEE International Symposium on High Performance Distributed Computing (HPDC 2000), pp. 71–78 (2000)
25. Zhang, X., Qu, Y., Xiao, L.: Improving distributed workload performance by sharing both CPU and memory resources. In: Proceedings of 20th International Conference on Distributed Computing Systems (ICDCS 2000), pp. 233–241 (2000)
26. Zhao, M., Figueiredo, R.J.: Experimental study of virtual machine migration in support of reservation of cluster resources. In: Proceedings of 2nd International Workshop on Virtualization Technologies in Distributed Computing (VTDC 2007) (2007)

Xiaofei Liao received a Ph.D. degree in computer science and engineering from Huazhong University of Science and Technology (HUST), China, in 2005. He is now an associate professor in School of Computer Science and Technology at HUST. His research interests are in the areas of computing system virtualization, peer-to-peer system, cluster computing and streaming services. He is a member of the IEEE and the IEEE Computer Society.

Liting Hu was Ph.D. candidate in computer engineering from Huazhong University of Science and Technology (HUST). Her research interests are in the areas of computing system virtualization and cluster computing.

Hai Jin received a B.Sc., a M.Sc. and a Ph.D. degree in computer engineering from Huazhong University of Science and Technology (HUST) in 1988, 1991 and 1994, respectively. Now he is a professor of computer science and engineering at HUST in China. He is now the Dean of School of Computer Science and Technology at HUST. He is the chief scientist of National 973 Basic Research Program, "Basic Theory and Methodology of Virtualization Technology for Computing System", and the largest grid computing project, ChinaGrid, in China. He is a senior member of IEEE and member of ACM. He is the member of Grid Forum Steering Group (GFSG). His research interests include virtualization technology, cluster computing and grid computing, peer-to-peer computing, network storage, network security, and high assurance computing.