Performance and Energy Modeling for Live Migration of Virtual Machines

Haikun Liu[†‡], Cheng-Zhong Xu[‡], Hai Jin[†], Jiayu Gong[‡], Xiaofei Liao[†]

[†]School of Computer Science and Technology
Huazhong University of Science and Technology
Wuhan, 430074, China
{hjin, xfliao}@hust.edu.cn

[‡]Department of Electrical and Computer Engineering
Wayne State University
Detroit, MI, 48202, USA
{hkliu, czxu, jygong}@wayne.edu

ABSTRACT

Live migration of virtual machine (VM) provides a significant benefit for virtual server mobility without disrupting service. It is widely used for system management in virtualized data centers. However, migration costs may vary significantly for different workloads due to the variety of VM configurations and workload characteristics. To take into account the migration overhead in migration decision-making, we investigate design methodologies to quantitatively predict the migration performance and energy cost. We thoroughly analyze the key parameters that affect the migration cost from theory to practice. We construct two application-oblivious models for the cost prediction by using learned knowledge about the workloads at the hypervisor (also called VMM) level. This should be the first kind of work to estimate VM live migration cost in terms of both performance and energy in a quantitative approach. We evaluate the models using five representative workloads on a Xen virtualized environment. Experimental results show that the refined model yields higher than 90% prediction accuracy in comparison with measured cost. Model-guided decisions can significantly reduce the migration cost by more than 72.9% at an energy saving of 73.6%.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Modeling techniques; D.4.8 [**Operating Systems**]: Performance – *Modeling and prediction*

General Terms

Measurement, Performance, Design, Experimentation.

Keywords

Virtual Machine, Live Migration, Performance Model, Energy.

1. INTRODUCTION

Virtualization [8, 23] is a rapidly evolving technology that provides a range of benefits to computing systems, such as improved resource utilization and management, application isolation and portability, and system reliability. Among these features, live migration is a core function to replace running VMs seamlessly across distinct physical hosts [10, 25]. It has become an extremely powerful tool for system management in a variety of key scenarios, such as VM load balancing [33], fault tolerance [24], power management [26] and other applications[20, 29].

VM live migration technology has attracted considerable interest for data center management and cluster computing in recent years [10, 18, 22, 25]. Representative works include XenMotion [10] and Vmotion [25] which were implemented as build-in tools in their virtualization platforms. There were also many other studies on the migration strategy for a variety of application cases, concerning the issues of where and when a VM should be migrated [14, 28, 31, 33]. However, few studies are available on the issue of which VM should be the best candidate for cost-efficient migration. For example, for the purpose of load balancing in a virtual cluster, all the VMs hosted by an overloaded physical node would be potential candidates for migration. However, different migration choices may lead to significant differences in performance and energy consumption. Considering migration downtime, previous studies demonstrated that it could vary significantly between different workloads, ranging from 60 milliseconds for a Quake 3 game server to 3.5 seconds in the case of a diabolical workload MMuncher [10]. This is mostly due to the diversity of VM configurations and workload characteristics. For instance, the initial memory size of a VM and applications' memory access pattern are critical factors that have a decisive effect on the migration latency, *i.e.* the total time a VM is undergoing performance penalty and high power state.

Power management is another increasingly important case for live VM migration. A rationale behind is that light loaded VMs can be consolidated into fewer physical hosts so that the offloaded physical hosts can be decommissioned to save energy. Previous studies demonstrated the effectiveness of this policy to achieve significant power saving [13, 17]. However, the previous works mostly focused on the schemes of VM placement with performance constrains [21, 31]. There was little work concerning about both the performance and energy costs during the consolidation actions.

In this paper, we model the VM migration performance based on theoretical analysis and empirical studies on Xen platform. In wired network environments, we experimentally verify that the migration energy consumption is proportional to the data volume of network traffic due to VM migration. We design a detailed analytical model to estimate the migration performance by scrutinizing Xen's live migration algorithm. We obtain some semantic knowledge about the workloads at the VMM level and train the model parameters using linear regression technology. The model reads the statistical data of memory updating and other performance parameters collected on

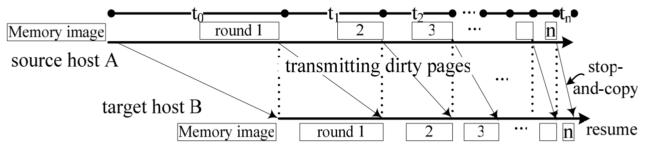


Figure 1. Live migration algorithm performs pre-copying in iterative rounds.

real workloads, and simulates the migration process to predict the migration cost. The experimental results demonstrate that our models are able to predict the migration performance and energy with an accuracy of more than 90% on five representative workloads. The contributions of this paper are summarized as follows:

1) This is the first kind of work to quantitatively model VM migration costs in terms of both performance and energy. The models would facilitate the design of optimal migration strategies.
2) We analyze the challenges of modeling the energy consumption of a VM migration and construct an application-oblivious high-level energy model with high prediction accuracy.
3) We validate the models by conducting a large set of experiments. The comprehensive evaluation results demonstrate the effectiveness of model-guided live migration in both performance and energy costs.

The remainder of the paper is organized as follows. Section 2 gives a brief introduction to our groundwork. Section 3 presents the design of our two models for prediction of migration performance and energy consumption. Section 4 describes the implementation details of our prototype for model validation. Section 5 presents the evaluation methodologies and experimental results. Section 6 discusses about the related work. Finally, we conclude our work in Section 7.

2. LIVE VM MIGRATION

Live VM migration technologies have proven to be a very effective tool to enable data center management in a non-disruptive manner. Both Xen and VMware adopts pre-copying algorithm for VM live migration in a memory-to-memory approach [10, 25], as shown in Figure 2. In the approach, physical memory image is pushed across network to the new destination while the source VM continues running. Pages dirtied during the migration must be iteratively re-sent to ensure memory consistency. By iterative it means that pre-copying occurs in several rounds and the data to be transmitted during a round are the dirty pages generated in the previous round. The pre-copying phase terminates 1) if the memory dirtying rate exceeds the memory transmission rate; or 2) if the remaining dirty memory becomes smaller than a pre-defined threshold value; or 3) if the number of iterations exceeds a given value; or 4) the network traffic exceeds a multiple of the VM memory size. After several rounds of synchronization, a very short stop-and-copy phase is performed to transmit the remaining dirty pages. As the data transferred is relatively small, this mechanism results in a nearly negligible best-case migration downtime.

We note that the performance of live VM migration is affected by many factors. First of all, the size of VM memory has a main effect on the total migration time and network traffic. Secondly, the memory dirtying rate, which reflects the memory access pattern of different applications, impacts the number of iteration rounds and data transferred in each pre-copying round, and hence indirectly affects the migration time and network traffic. Thirdly, the network transmission rate together with the configuration of migration algorithm is also crucial to migration performance. However, migration performance varies significantly depending on the workload characterization even when other conditions remain the same. For instance, migration of a VM running memory-intensive applications would lead to more performance degradation in terms of network traffic, migration downtime and latency.

Table 1. Parameters for performance and energy models of VM live migration.

V_{mem}	Current size of VM memory during migration.
V_{mig}	Total network traffic during migration.
T_{mig}	Migration latency, i.e., total migration time.
T_{down}	Application downtime during migration.
R	Memory transmission rate during migration.
D	Memory dirtying rate during migration.
V_{thd}	Threshold value of the remaining dirty memory that should be transferred at the last iteration.
W	Writable Working Set, it defined as a certain set of "hot" pages that will be written frequently.

3. PERFORMANCE AND ENERGY MODELS

A primary goal of our models is to determine which VM should be migrated within a server farm with minimum migration cost. This is complementary to other issues like minimizing physical resource consumption without compromising service level agreements (SLA) [21, 31], and the issue of when a VM should be migrated and to which host [33]. A key point for cost-aware migration decision is how to accurately predict the performance and energy consumption of each VM migration in a server farm.

3.1 Base Model of Migration Performance

Modeling the performance of migration involves several factors: the size of VM memory, the workload characteristic (denotes the memory dirtying rate), network transmission rate, and the migration algorithm (different configurations of migration algorithm means great variations of migration performance). The most challenge is characterizing the memory access pattern of each running workloads correctly.

In Table 1, we define a number of key parameters and their notations for our performance and energy models. Live VM migration achieves negligible application downtime by iteratively pre-copying the pages dirtied at previous round of transmission. Assume the pre-copying algorithm proceeds in *n* rounds. We denote the data volume transmitted at each round as V_i ($0 \le i \le n$), and the elapsed time at each transferring round as T_i ($0 \le i \le n$). V_0 is equivalent to the VM memory size V_{mem}. T_0 represents the time consumed to transfer the data of VM memory image and T_i is the time to transfer the dirty memory generated during previous rounds. The data transmitted in round *i* can be calculated as:

$$V_i = \begin{cases} V_{mem}, & \text{if } i = 0; \\ D \cdot T_{i-1}, & \text{otherwise.} \end{cases} \quad (1)$$

The elapsed time at each round can be calculated as:

$$T_i = \frac{D \cdot T_{i-1}}{R} = \frac{V_{mem} \cdot D^i}{R^{i+1}}. \tag{2}$$

At first, we consider the scenario that the memory dirtying rate is smaller than the memory transmission rate on average. Let λ denote the ratio of D to R:

$$\lambda = D / R. \tag{3}$$

Combining equations (1), (2) and (3), we have the network traffic during the round i:

$$V_i = D \cdot \frac{V_{mem}}{R} \lambda^{i-1} = V_{mem} \lambda^i. \tag{4}$$

Then the total network traffic during the migration can be summed up as:

$$V_{mig} = \sum_{i=0}^{n} V_i = V_{mem} \cdot \frac{1-\lambda^{n+1}}{1-\lambda}. \tag{5}$$

Combining equations (2) and (3), we have the migration latency:

$$T_{mig} = \sum_{i=0}^{n} T_i = \frac{V_{mem}}{R} \cdot \frac{1-\lambda^{n+1}}{1-\lambda}. \tag{6}$$

Because T_{mig} is the duration of migration that has negative effect on the performance of running applications, it is a key performance metric for migration decision.

Now, we analyze the migration downtime caused in the whole migration process. It is composed of two parts: the time the remaining dirty pages are transferred during the stop-and-copy phase, denoted as T_n; and the other time that is spent on VM resuming at the destination host, denoted as T_{resume}. The second part has little variation and can be represented as a constant; we set it to 20ms in our experiments. Then the migration downtime can be represented as:

$$T_{down} = T_n + T_{resume}. \tag{7}$$

To evaluate the convergence rate of VM migration algorithm, we can calculate the total rounds of the iteration by the inequality $V_n \le V_{thd}$. It is the condition when the iterative pre-copying should be terminated. From equation (4), it follows $V_{mem}\lambda^n \le V_{thd}$. As a result, the number of pre-copying iteration becomes:

$$n = \left\lceil \log_\lambda \frac{V_{thd}}{V_{mem}} \right\rceil. \tag{8}$$

For a given VM, the memory size V_{mem} and V_{thd} (determined by migration algorithms) are fixed. Consequently, the iterative pre-copying would converge faster if λ is smaller. We therefore refer to λ as the convergence coefficient of live VM migration.

From the above analyses, we conclude that a VM with smaller size of memory image and smaller λ would generate less network traffic and lead to shorter migration latency, hence should be a better candidate for migration.

Second, we consider the scenario that the average memory dirtying rate is even larger than the memory transmission rate. According to equation (4), the data transferred in each pre-copying round V_i would even exceed the VM memory size V_{mem}. This is not applicable in our model. However, the migration algorithm implemented in Xen is able to solve this issue by harnessing the statistical information of dirty memory. It is known that every workload exhibits a set of 'hot' pages which would be updated extremely frequently, so called *Writable Working Set* (*WWS*) [10]. These pages are often used for the stack and local variables of the running processes as well as pages allocated for network and disk buffer. The hottest pages would be dirtied as fast as the migration daemon can transfer them, and hence should be skipped during the iterative pre-copying. Their transfer is postponed till the final stop-and-copy phase. In practice, the migration algorithm periodically peeks the current round's dirty bitmap and retains the pages that are dirtied in two consecutive peeking periods. For most of workloads, we observed that the size of WWS is approximately proportional to the pages dirtied in each pre-copying round. That is

$$W_i = \gamma \cdot V_i, \tag{9}$$

where γ is the ratio correlating with the memory dirtying rate and the duration of each iteration. We characterize this relationship in a linear equation model:

$$\gamma = \omega \cdot T_{i-1} + \xi \cdot D + \psi, \tag{10}$$

where ω, ξ and ψ are the model coefficients to be learned. Taking multiple observations of the equation allows estimating the model parameters using learning techniques such as linear regression. We estimate the model coefficients by running DaCapo [9] benchmark, which consists of a suit of Java applications. Some applications generate non-trivial memory loads. We trained our model by executing the migration of each application one by one. The trained model with these workloads should be able to embody the memory access patterns of a large variety of applications.

During each migration, we track the amount of dirtied pages and skipped pages in each round of pre-copying to calculate the value of γ. With the elapsed time and memory dirtying rate of each pre-copying round, we can build a set of independent equations based on the experimental results. Finally we derive the value of each parameter, $\gamma = -0.0463T_{i-1} - 0.0001D + 0.3586$, where T_{i-1} is normalized by second and D is normalized by MB/sec. There are a number of approaches to evaluate how well the model fits the practice. Amongst them, the coefficient of determination is a widely used term, denoted as R-square statistic. The model is better as the R^2 is closer to 1. The R^2 value of this model is 0.843.

Based on the above analysis, our migration performance model can be shown in Algorithm 1. All input variables can be measured from practical workloads. The size of VM memory V_{mem} can be obtained from *xenstore*, which is an information storage space shared between VMs and somewhat similar in spirit to *procfs*. The V_{thd} and *max_round* are determined by the configuration of migration algorithm. The memory dirtying rate D can be measured before the VM migration decision, and the value of memory transmission rate R can be configured according to the observed value of D. The detailed measurement of D and R is described in Section 4.

3.2 Refined Model of Migration Performance

The base model discussed in the preceding section achieves good estimates when the observed memory dirtying rate holds steady. However, in a data center, most of workloads are service-oriented and heterogeneous. The memory access pattern of each application may vary in response to the change of the service requests. In such cases, cost estimation may cause more deviation if the model only uses on-line sampled parameter values.

When a VM is in migration, the migration daemon will continuously track the dirty pages to direct the memory pre-copying. Moreover, it also logs other statistical information such as data transmission rate, memory dirtying rate in each round of iteration. The data represents a long-term application characteristic in the whole execution of migration. This means that we can employ the historical data

Algorithm 1: **Performance Model of VM Migration**

```
1.  Input: $V_{mem}$, $V_{thd}$, $D$, $R$        Output: $V_{mig}$, $T_{mig}$, $T_{down}$
2.  let $V_0 \leftarrow V_{mem}$      /*data transferred at first round*/
3.    for $i = 0$ to $max\_round$ do
4.        $T_i \leftarrow V_i / R$
5.        $\gamma \leftarrow a\, T_i + b\, D + c$
6.        $W_{i+1} \leftarrow \gamma\, T_i\, D$
7.        $V_{i+1} \leftarrow T_i\, D - W_{i+1}$
8.        if $V_{i+1} <= V_{thd}$ or $V_{i+1} > V_i$ then
9.            $V_{i+1} \leftarrow T_i\, D$   /*data transferred at last round*/
10.           $T_{i+1} \leftarrow V_{i+1} / R$
11.           $T_{down} \leftarrow T_{i+1} + T_{resume}$
12.           break
13.       end if
14.   end for
15.       $V_{mig} \leftarrow \sum_{i=0}^{max\_round} V_i$
16.       $T_{mig} \leftarrow \sum_{i=0}^{max\_round} T_i$
```

to refine our model when the VM needs to migrate again. We can represent the average memory dirtying rate during a migration with the weighted arithmetic mean of D in each round of pre-copying:

$$\overline{D} = \frac{\sum D_i \cdot T_i}{\sum T_i} . \tag{11}$$

Considering the scenario that a VM may migrate many times, we use exponential weighted moving average (EWMA) to describe the current memory dirtying rate. This is a popular method to estimate a non-deterministic variable (such as RTT estimation in computer networks [11]). Combining the statistical data counted from the most recent migration and historical records, it becomes:

$$\hat{D}_m = \mu \overline{D}_m + (1-\mu)\, \hat{D}_{m-1} , \tag{12}$$

where m represents the number of migration times in a VM's lifecycle, and μ denotes the weight of observation value from most recent migration, its value should be larger than 0.5 so that the synthesized value should always represent the current workload characteristic. Combining the historical logs with on-line sampled data, we can calculate the refined value of D as following:

$$\tilde{D}_m = \varphi S_m + (1-\varphi)\, \hat{D}_{m-1} , \tag{13}$$

where $0<\varphi<1$, the default value is set as 0.5 and it can be self-tuned according to the feedback of model errors. S_m denotes the current sampled value of memory dirtying rate.

3.3 Energy Model of VM Migration

Energy has become a key concern in large scale systems such as grid and cloud computing data centers due to their ever increasing total operational cost and energy consumption [27]. In a data center with hundreds and thousands of VMs, VM migration is a commonplace and the energy consumption due to migration should not be overlooked. The following describes the model for an energy-efficient migration decision.

3.3.1 Modeling Challenges

It is known that the power drawn by a physical server consists of a static portion and a dynamic portion [15]. The static portion is the stable power consumption even if the server is completely idle. The dynamic portion is defined as the additional power consumed by the physical resources when working on behalf of some VM or applications [19]. The power drawn by the migration procedure is a dynamic portion that involves several hardware components of the server, such as CPU, memory and network interface card. In practice, it is not applicable to measure the power consumption of each component separately but only the full-system power.

Early works on power modeling were mostly under the assumption of homogeneous applications. Power modeling methodologies can be designed by profiling the resource utilization at hardware layer [12]. In practice, applications residing in a VM are usually heterogeneous and may show a large variety of physical resource usage at different execution phases. In addition, VM migration is a short lived procedure that would act on all kinds of workloads. Its cost estimation should be modeled in an application-agnostic manner. On the other hand, modeling power in an application-agnostic manner tends to introduce significant errors for heterogeneous applications [16, 31]. Moreover, power model based on low-level resource utilization greatly depends on the power characteristic of hardware. For example, different CPU design technologies and frequencies would draw different amount of power and pose requirements for different power models. Creating a model for different applications on different hardware platforms using an offline training method is practically infeasible, if not impossible. Another key issue is that because our model is designed for online decision of VM migration, energy cost of VM migration should be estimated prior to migration decisions. These reasons pose significant challenges to model the energy consumption of VM migration.

3.3.2 Model Construction

The power drawn by VM migration is mainly determined by the data transmission rate. As shown in Figure 2, the power consumption due to VM migration at the source host increases as the data transmission rate is increased. On the other hand, the migration latency becomes shorter when the data transmission rate is higher. As energy is defined as product of power consumption and time duration, how does the energy consumption vary with different data transmission rates? Given a specific VM, Figure 3 shows that the average energy consumption at different data transmission rates leads to less than 6 percentile of variation. This observation implies that the energy consumption due to migration is largely independent of the data transmission rate in a wired network.

We note that VM migration is an I/O-intensive application and the energy is mainly consumed by data transferring and receiving over the networks. We conjecture that the energy cost due to VM migration is only determined by the data volume of network traffic. Based on this we design a high-level model to estimate the energy cost of VM migration.

VM migration involves source host, network switch, and destination hosts. Because switching fabric may be very complex thus the energy is hard to quantify, our model only considers the energy drawn by each migration side. The data transmitted on source host and the data received on destination host are equal, and we find very little difference of energy consumption for data transmission and receiving in a homogeneous platform. Hence in the following model, the energy consumption is assumed to increase linearly with the network traffic of VM migration:

$$E_{mig} = E_{sour} + E_{dest} = (\alpha_s + \alpha_d) V_{mig} + (\beta_s + \beta_d) , \tag{14}$$

where α_s, α_d, β_s, β_d are model parameters to be trained. We note that for heterogeneous physical hosts, although equation (14) is still applicable, the model parameters should be re-trained for each of two different platforms. As current virtualization platforms in-

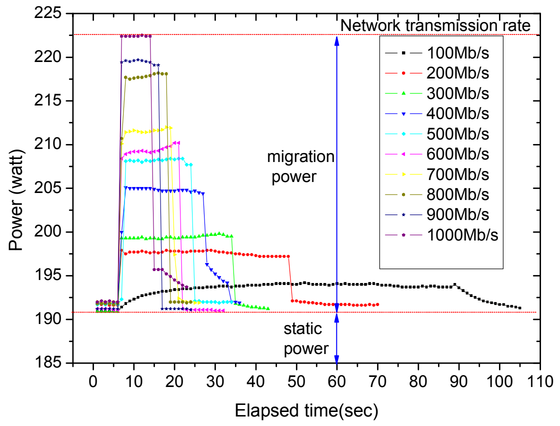


Figure 2. When an idle VM with 1GB memory is migrated at incremental data transmission rate, the power consumption progressively increases while the migration latency decreases.

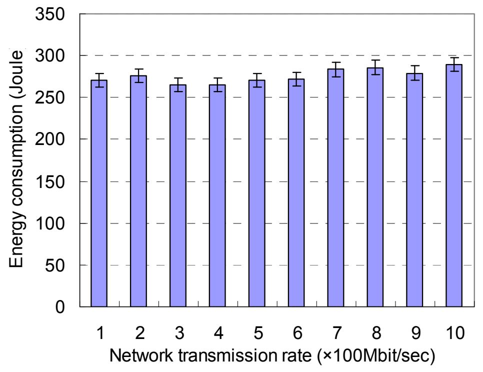
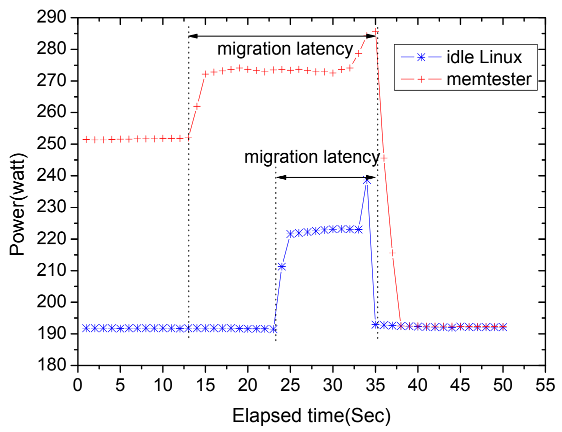
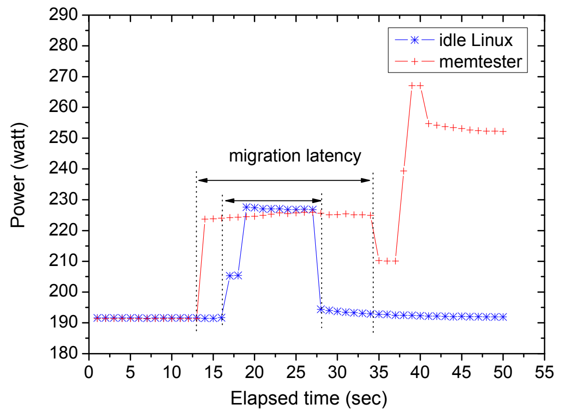


Figure 3. Additional energy consumption due to migration of an idle VM with 1GB memory (at the source host) under different data transmission rates.



(a) Power drawn at source host



(b) Power drawn at destination host

Figure 4. Power consumption at the source and target host while migrating a VM with 1GB RAM.

cluding Xen and VMware only support VMs migration between homogeneous hosts, to simplify the problem, we train the energy model in a homogeneous environment. Consequently, equation (14) is reduced to:

$$E_{mig} = E_{sour} + E_{dest} = \alpha V_{mig} + \beta \, , \tag{15}$$

where network traffic V_{mig} is measured in Megabytes and energy E_{mig} is measured in Joules.

3.3.3 Model Parameters Training

We use linear regression with ordinary least squares estimation to train the model parameters α and β in equation (15). To generate a sufficient number of samples resulting in linearly independent equations and spanning a large range of data traffic values, we conduct a large number of experiments to migrate a VM with different sizes of memory.

V_{mig} can be easily obtained from practical migrations in our training experiments. To calculate the additional energy consumption (denoted by E_{mig}), we should obtain the increase of power due to VM live migration. We first measure the static power of physical hosts when the workloads residing in the VM reach a stable state. To improve the accuracy of measurements, we need to avoid the fluctuations when we meter the static power. The VM being migrated is the only one running on the source host and the applications are homogeneous. As shown in Figure 4, an idle Linux and "memtester" [1] application show stable power consumption during their normal execution. The measured values of static power are with less than 1 Watt difference and hence can be represented as a constant P_0. Then we measure the power drawn by the host when the VM is been migrating, denoted by P_t. As the measured P_t is a discrete variable and the sampling rate of our power meter is 1Hz, the dynamic power due to VM migration can be calculated as follows:

$$E_{mig} = \int_0^{T_{mig}} (P_t - P_0)dt = \sum_{t=1}^{\lceil T_{mig} \rceil} (P_t - P_0) \, . \tag{16}$$

For each experiment, we record the data volume transmitted during migration and calculate the energy consumption using equation (16). In Figure 5, we plot these (E_{mig}, V_{mig}) pairs and the equation of our energy model learned by linear regression technique. We derive the parameters α =0.512, β=20.165, and the R^2 is as high as 0.993. Then our energy model for VM live migration in (15) becomes:

$$E_{mig} = 0.512 V_{mig} + 20.165 \, . \tag{17}$$

3.4 Model-Guided VM Migration Decision

In a server farm, there are massive VMs hosting various applications. The VMs may exhibit different sizes and memory access patterns. Their migration would incur a large difference of migration over-

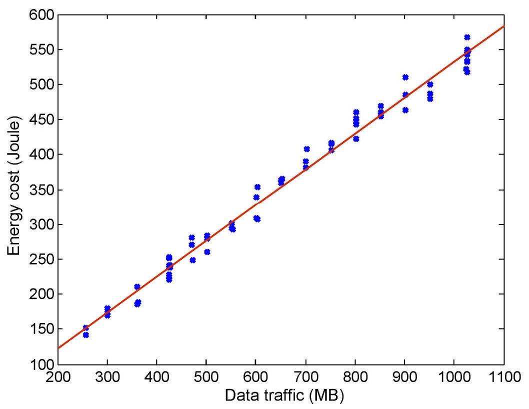


Figure 5. Training the energy model of VM live migration using linear regression.

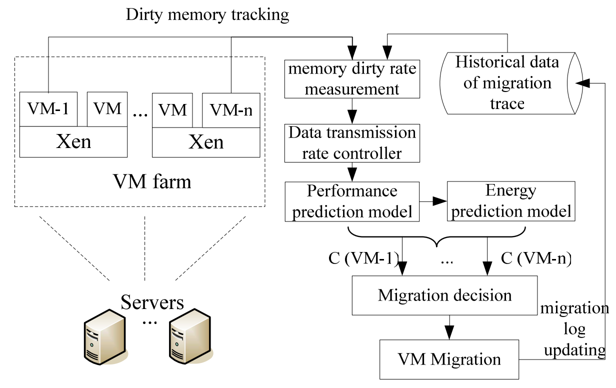


Figure 6. Block diagram of system framework.

head. In order to minimize the migration cost, we should make the best migration decision. We propose a synthesized formula that integrates all the migration cost metrics to direct the migration choice.

$$C(VM_i) = a\,T_{down} + b\,T_{mig} + c\,V_{mig} + d\,E_{mig}. \qquad (18)$$

where $a+b+c+d=1$, a, b, c and d are the weights of cost metrics respectively. For numerical stability, it is preferable to normalize each variable T_{down}, T_{mig}, V_{mig}, E_{mig} with respect to their maximum values observed during migration. It means that we should scale up or down the observed values to similar magnitudes.

4. IMPLEMENTATION

To demonstrate the validity of our models, we have designed and implemented a prototype based on Xen 3.4 platform. The models are also applicable for other virtualization platforms that use similar pre-copying algorithm for live migration.

4.1 System Framework

The system framework of our models is shown in Figure 6. It also includes a flow diagram of using these models for migration decision. When some VMs need to be migrated for the reason of load balancing or server consolidation, we should choose the best candidates to minimize the migration cost. For this purpose, we first take a peek at current memory access pattern of each running VM. Combining with the historical data, we can estimate the memory dirtying rate of each workload, and then the memory transmission data can be configured to adapt to the memory dirtying rate. After all the input variables in our model are ready, we can use the performance and energy prediction models to estimate the migration cost. Finally, we can choose the best candidate for migration by sorting the synthetic migration cost. After the VM migration is finished, the on-line profile data during migration are counted to update the historical statistical data for next time of migration.

4.2 Memory Dirtying Rate Measurement

Xen provides the ability to track memory accesses of guest domains (*i.e.*, VMs) using a mechanism called shadow page tables. The shadow page tables are maintained by the hypervisor and translated from guest page tables on demand. In memory log-dirty model, all VM memory pages are transparently marked as read only. Guest pages are protected to trap the guest updating to their internal versions of page tables, allowing the hypervisor to track the updates and propagate them to the external shadow versions as appropriate. In this way, the hypervisor is able to trap all memory updates within a VM, and maintains a bitmap to mark the dirty pages.

To track the memory access pattern of different workloads, we employed Xen's shadow paging mode to track dirtying statistics on all pages used by a running VM. As VM live migration also works in shadow paging model, we can measure a VM's memory dirtying rate incidentally before the pre-migration phase. As shown in Figure 7, when shadow paging model is enabled, all guest page table entries mapping in shadow page tables are write-access protected, and any page updating would trigger a page fault and is trapped by Xen. If write access is permitted by the relevant guest page table entry (PTE), this permission is extended to the shadow PTE. At the same time, the appropriate bit in the dirty page bitmap is set to 1. In a very short observation window, we peek at the dirty page bitmap and count the dirty pages to calculate memory dirtying rate. The performance cost is mainly due to two hypercalls (interfaces provided by the hypervisor) with respect to cleaning and peeking at the dirty page bitmap. The processing is transparent to the application's execution and the overhead is in the order of milliseconds.

4.3 Adaptive Data Transmission Rate

In general, we should ensure that the migration would not disrupt other online services that reside in the same host due to resource contention (*e.g.*, CPU and network bandwidth). To mitigate the performance degradation, we should limit the network bandwidth reserved for the migration daemon.

It is known that the VM memory is transferred in batches of fixed pages. This facilitates the control of migration throughput based on a budget constrained controller. Given a specific data transmission rate, an adequate time slice is pre-allocated for each batch of data sending. The duration is calculated according to an empirically pre-defined converter between network transmission rate and time. A higher data transmission rate corresponds to a shorter time slice. The budget, representing the number of pages need to send in each batch, is a specific value. During the period of a burst of data sending, the budget is continuously consumed until it is used up, and then the migration daemon will do nothing in the remaining time slice. In this way, the observed network throughput can be limited to the target with less than 5% deviation. As most of applications exhibit a dynamic memory dirtying rate, the memory transmission rate should be dynamically adjusted according to the observed

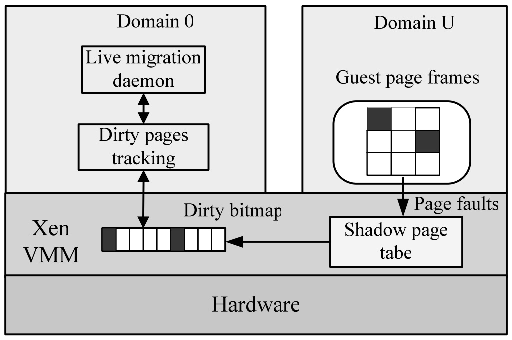


Figure 7. Dirty memory pages tracking of Xen VMM.

memory dirtying rate in each round of pre-copying. Among a predefined minimum and maximum bandwidth limit, the data transmission rate for each round is determined by adding a constant increment to the previous round's memory dirtying rate. That means

$$R = D + \delta\,, \tag{18}$$

where δ is a constant variable and its default value is empirically set as 100Mb/s.

5. EVALUATIONS

In this section, we first introduce the experimental environment. Then we validate the models by comparing the model estimated values with measured ones in terms of the following metrics:

1) Migration latency: the duration from the time VM migration is initiated to the time the migrated VM gets a consistent state with the original one.
2) Migration downtime: the duration when the services residing in the migrating VM are entirely unavailable, neither at the source side nor at the target side.
3) Total network traffic: the total data volume that need to transmit during the migration processing.
4) Energy consumption for migration: the additional energy cost due to the VM live migration.

5.1 Experiment Setup

We conducted all experiments on Dell PowerEdge1950 servers with two Intel quad-core Xeon E5450 3GHz processors, 8GB memory, one 250GB SATA hard disk, and 1Gbit Ethernet interface. We used WattsUp Pro [2] to measure the power consumption. The power meter has an accuracy of ±1.5% of the measured RMS (root-mean-square) power with sampling rate of 1Hz. The host machines were running Red Hat 4.1.2 distribution and the hypervisor was Xen 3.4.1 with Linux 2.6.18.8-Xen kernel. The guest OSes were also running Red Hat 4.1.2 with Linux 2.6.18.8 kernel. For migration, all the VM image files could be accessed with NFS. In order to simplify the problem, all VMs were configured to use 4 VCPUs and 1GB RAM. In this way we do not need to consider the parameter of VM memory size V_{mem}. To reserve bandwidth for other applications, we limited the network transmission rate for VM migration within a range from 400Mbit/sec to 800Mbit/sec.

In each experiment a VM was migrated for ten times between two physical hosts. The results reported are the average of the ten trials. The experiments used five workloads, representative of typical server applications in today's data centers:

1) Linux idle: an idle Linux OS for daily use. This workload is used as a frame of reference for comparison.

2) TPC-C [3]: it is an on-line transaction processing (OLTP) benchmark. TPC-C simulates a complete environment where a population of terminal operators executes transactions against a database. We configure 1000 terminals threads and 500 database connections in our experiments.

3) Dbench [4]: it is an open source benchmark emulating the file system load. This benchmark can simulate a variety of real file servers (such as proxy-server, mail-server, Web-server) by executing create/write/read/delete operations on a large number of directories and files with different sizes. We configure 10 simultaneous connections to generate reasonable disk load.

4) LINPACK [5]: we use the Intel optimized MP LINPACK benchmark to perform massive vector and matrix operations, which produces severe CPU and memory pressure.

5) SPECweb2005 [6]: it is a complex application-level benchmark for evaluating web servers. The workload is a complex mix of dynamic page requests. We use two client machines to generate the load for the web server. Each client is configured with 800 concurrent connections.

5.2 Dirty Memory Measurement

The shadow model of Xen allows us to track the set of pages to be dirty within any time windows. We conducted a set of experiments to measure the memory dirtying rate for a variety of workloads within 60 seconds. In each case we started the relevant application in a virtual machine, read the dirty bitmap and then cleaned it every half second. We chose a 0.5 second window to measure the memory dirtying rate because most of workloads consume approximate half second during each pre-copying rounds. Figure 8 plots the number of dirty pages generated by each of the representative workloads in a VM with 1GB RAM. The x-axis shows elapsed time and the y-axis measures the number of 4KB pages of memory dirtied within the corresponding 0.5 second interval. From these curves we can see that the observed memory dirtying rate varies significantly between different applications. For workloads such as TPC-C and idle Linux, the memory dirtying rate retains at relative low level and hence are the excellent candidate for migration. In contrast, Linpack and SPECweb2005 have a consistently high dirtying rate and would be problematic to migrate.

The memory dirtying rate can be measured from another aspect as well. In this case, we started the relevant application in a VM and peeked the dirty bitmap every 50ms in a 60 seconds window. Unlike the experiments aforementioned, we did not clean the dirty bitmap after we had a peek at it. This allows us to observe the writable working set in a relative long time window (60s) but estimate it at a finer granularity. Figure 9 plots the number of dirty pages generated by the representative workloads. The slope of each line joining the origin and the point on each curve represents the memory dirtying rate. We can see that most of the applications exhibited a higher memory dirtying rate at the beginning of measurement. With the increase of the observation window size, the memory dirtying rate drops in all applications. This is because all the applications have a set of 'hot' pages that tend to be written many times, while the relevant dirty bitmap does not record the number of updating for each page. However, it implies that we can measure the WWS when the slope of a curve at a particular point does not change. For example, the WWS of workloads Dbench and SPECweb become stable only after 20~30 times sampling, while TPC-C shows a relatively long-term of increasing WWS before the 400th sampling. This means most of dirty pages of OLTP benchmark are generated by allocating new memory pages for upcoming connections and only a few pages would be repetitively updated. For Dbench, although a set of pages

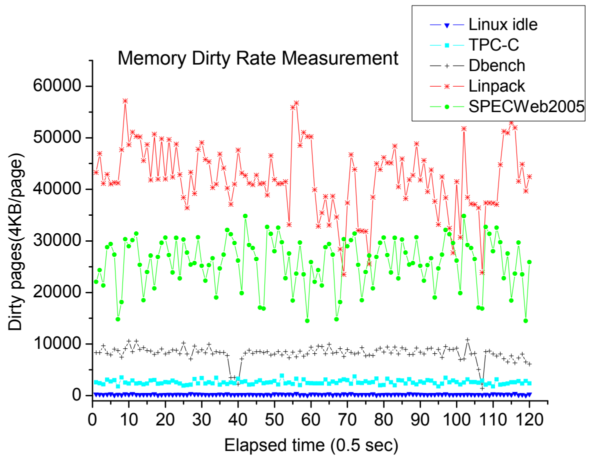


Figure 8. Rate of pages dirtied in each half second for different workloads in a VM with 1GB RAM.

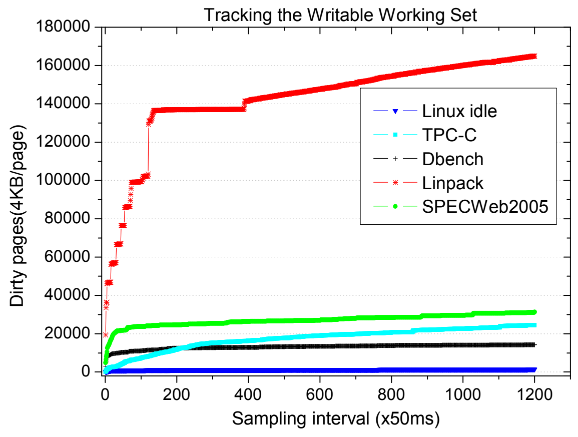


Figure 9. Writeable working set measurement for different workloads in a VM with 1GB RAM.

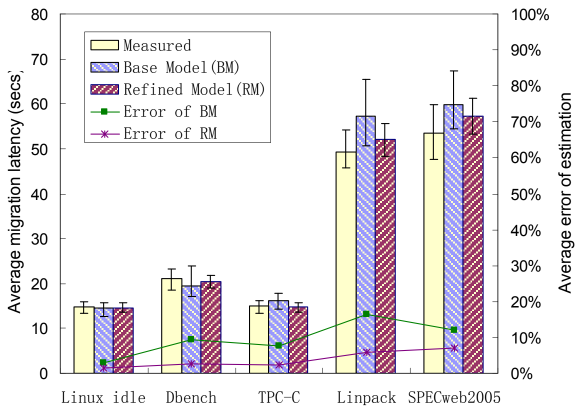


Figure 10. Model errors on migration latency estimation.

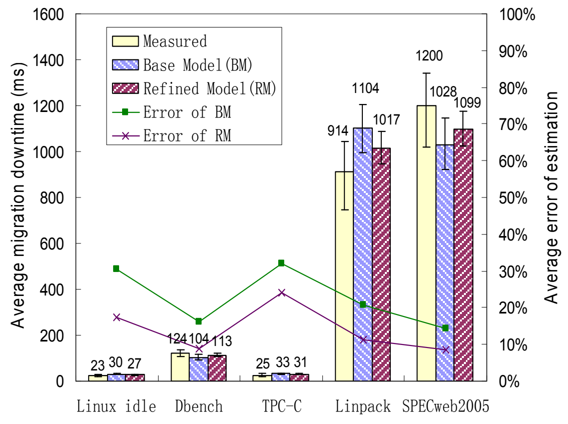


Figure 11. Model errors on migration downtime estimation.

is intensively updated, it maintains a relative smaller dirty memory footprint than TPC-C. As Linpack is a both CPU and memory intensive workload, it shows a quite large WWS and very high memory dirtying rate, thus should be evicted from the migration candidates.

5.3 Model Accuracy Validation

We conducted a set of experiments to validate the effectiveness of our models. For each metrics, we compared the experimentally measured values with the estimates using the base model (BM) and the refined model (RM), and showed the prediction errors of BM and RM respectively.

Figure 10 shows migration latency of the representative workloads and the prediction error of our model. The leftmost bar of each set of data displays the measured value of migration latency, the other two bars show the migration latency estimated by our base model and refined model. The line plots illustrate the estimation errors of the two models respectively. Our refined model only has less than 7% estimation errors. This means our model can achieve satisfying accuracy for practical use. From the experimental results, we can find that the migration latency of Linpack and SPECweb show a higher standard deviation than an idle Linux, Dbench and TPC-C, because the observed memory dirtying rate of the two workloads exhibits more widespread dispersion of data from its mean, as shown in Figure 8. It is the same reason why the estimation errors for Linpack and SPECweb are higher than the other workloads. The refined model have taken into account the historical statistical data of migration trace, the synthetic value of memory dirtying rate is expected to be more accurate than the online observed value, so the refined model shows much less estimation errors than the base model.

Migration downtime is a key performance metric that would be the most concern in some scenarios such as high availability applications. Our model also provides a quantitative estimation on this metric. Migration downtime is mostly correlated with the data volume of dirty memory that should be transferred in the stop-and-copy phase. As shown in Figure 11, Linpack and SPECweb exhibit much longer migration downtime than the other workloads because of their high memory dirtying rate. An idle Linux and TPC-C displays even more estimation errors than Linpack and SPECweb because even small absolute deviation can lead to a large relative error. However, the errors of these applications that show very short migration downtime would not interfere our migration decision. They are always the best candidates for cost-efficient migration.

Figure 12 shows the network traffic of VM migration for the representative workloads and the corresponding estimation errors. For an idle Linux and TPC-C, most of the network traffic is the 1GB VM memory image, and the remaining portion refers to the small

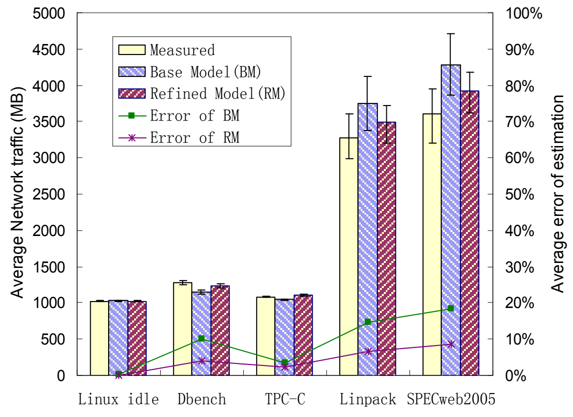


Figure 12. Model errors on network traffic estimation.

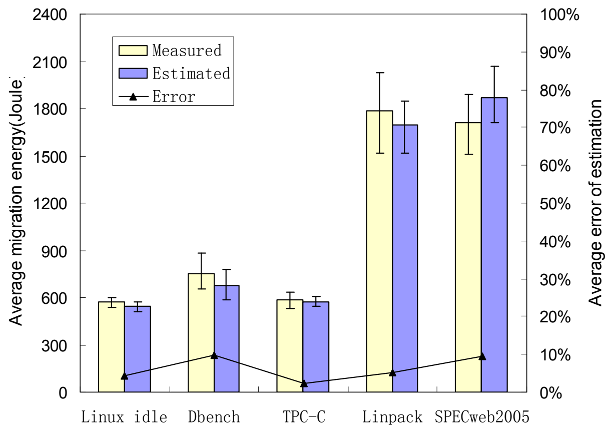


Figure 13. Model errors on migration energy estimation.

amount of dirty pages that are iteratively transferred during the pre-copying. That is why these workloads with slow memory dirtying rate show relative low estimation errors. Although TPC-C finally exhibits an even larger WWS than Dbench after a continuous increase (as shown in Figure 9), its slower memory dirtying rate leads to smaller total network traffic than Dbench. In contract, Linpack shows both larger WWS and higher memory dirtying rate than SPECweb, however the total network traffic of Linpack is even smaller than SPECweb. It seems to be incorrect by intuition. We track all the iterative pre-copying rounds of VM migration for the two workloads, and find that Linpack exhibits a much larger set of 'hot' pages than SPECweb. The migration algorithm does not transfer these frequently updated pages, so that Linpack has even less data transferred during each round of pre-copying than SPECweb. Our model can correctly simulate such condition and the below 10% estimation errors verified our model's effectiveness.

We verify the energy model in Figure 13. We use the data of network traffic estimated by our refined model as the input of energy model. The right column of each set of data shows the estimated energy cost of each VM migration. Compared to experimental measured energy, the model errors is as low as 10%. From Figure 12 and Figure 13, we can find that the energy cost is approximately linear with the network traffic. This validates the correctness of our linear energy model.

5.4 Model-guided Migration Cost

We now show how our models are useful for migration decision making. In this experiment, we run 8 VMs on the source host, the benchmark Dbench, TPC-C, Linpack, and SPECweb2005 are running on two VMs respectively. Each VM is configured to use 1 VCPU and 800MB RAM. The destination host is initially set to be idle. We compare the decision making scheme using our models with random selecting method. Each scheme is tested for 20 times, and then the total cost is summed up for each migration metric. We show the effect of using our models for migration decision in Figure 14. We can find that the migration cost can be significantly reduced with the guide of our models. The migration latency, migration downtime, network traffic and energy consumption are reduced by 72.9%, 93.5%, 74.5% and 73.6% respectively. The ground truth is that for random selecting of VMs, we perform the migration of a VM running Dbench, TPC-C, Linpack, and SPECweb2005 for 4, 5, 5 and 6 times respectively. However, guided by our model, we always choose a VM running TPC-C for migration, which is the best candidate with minimum migration cost.

6. RELATED WORKS

The performance of live VM migration had been studied on a variety of workloads in [10]. This technique had proven to be a very effective tool to enable service relocation in a non-disruptive manner. William *et al.* [32] evaluated the performance degradation of Web2.0 applications running inside VMs while they were being migrated. The main objective they concerned is the service level agreement (SLA) violation rather than the migration performance. None of these works provided a methodology to estimate the VM migration performance.

pMapper is a framework that investigate the VM placement algorithms in a virtualized sever cluster [31]. The expected performance benefit as well as migration cost were considered while determining a VM's placement in physical servers. However, the migration cost was quantified by estimating the decrease of application throughput or SLA violation. This black-box model can only be used in an offline manner and can't predict the migration cost for a workload-agnostic VM.

Sandpiper proposed both gray-box and black-box approaches to automatically mitigate hotspots in a virtualized cluster by employing the facility of VM live migration [33]. Sandpiper implemented a hotspot detection algorithm that determines when to migrate VMs, and a hotspot mitigation scheme that uses a greedy algorithm to determine what and where to migrate and how much resource to allocate after the migration. Tarighi *et al.* presented a scheduling algorithm for VM migration based on fuzzy decision making [30]. They employed TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) algorithm to find the most loaded servers and make more effective migration decision when the decision making condition is vague or uncertain. All the two approaches concentrated on VM placement strategies, the migration cost isn't their concern. Our work provides another criterion for migration decision from the prospective of migration performance and energy cost.

PADD presented a scheme of VM consolidation to save energy in a virtualized environment [21]. The scheme saved power by dynamically migrating virtual machines and packing them onto fewer physical machines. It focused on the VM placement algorithm to minimize total energy while avoiding SLA violation. However, the performance penalty and energy consumption during migration actions were not considered. Our models can be a beneficial complement for the migration decision making.

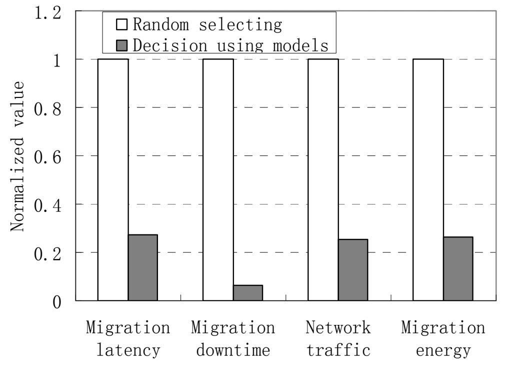


Figure 14. Migration cost saving by using our models

Sherif *et al.* designed a simulator based on Xen's migration algorithm to characterize the downtime and total migration time [7]. However, their simulation model relies on dynamic information like hotspot sizes collected in pre-copying iterations, it is hard to use for a *prior* migration decision before the migration begins. In contrast, our model makes no such assumptions and would facilitate online migration decisions. Moreover, our models provide estimates of more than downtime and migration latency, traffic volume and energy consumption due to migration were considered as well. We also exploit the statistical data from historical migration trace and adaptive network bandwidth limitation to enhance the practicability and effectiveness of our models.

7. CONCLUSION

In this paper, we designed two models to estimate VM migration performance. Our theoretical analysis and experimental results showed that the parameters such as VM memory size, network speed and memory dirtying rate are the major factors impacting migration performance in terms of migration downtime, migration latency and network traffic. Based on the performance model, we design a high-level linear model to estimate the migration energy. We validated the models by comparing the estimates with experimentally measured results. The experimental results showed that the prediction accuracy is higher than 90% in terms of both performance and energy metrics. The case study verified that our models could significantly reduce the migration cost by more than 72.9%.

We note that our performance and energy models for live migration would facilitate data center administrators to explore myriad of choices for optimal migration decision making. In addition, the models should also be able to introspectively guide the design of migration algorithm for different tradeoffs amongst the performance metrics. It will be investigated in our future work.

8. ACKNOWLEDGMENTS

This work is supported by National 973 Basic Research Program of China under grant No. 2007CB310900, the MoE-Intel Information Technology Special Research Foundation under grant No. MOE-INTEL-10-05, and U.S. NSF under grants CRI-0708232, CNS-0702488, CNS-0914330, CCF-1016966.

REFERENCES

[1] http://pyropus.ca/software/memtester/

[2] Electronic educational devices inc., "watts up pro power meter", http://www.wattsupmeters.com

[3] http://www.tpc.org/tpcc.

[4] http://samba.org/ftp/tridge/dbench.

[5] http://www.netlib.org/linpack.

[6] http://www.spec.org/web2005.

[7] S. Akoush, R. Sohan, A. Rice, A. W. Moore, A. Hopper. Predicting the Performance of Virtual Machine Migration. In *The 18th Annual IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'10)*, Miami, Florida, USA, August 17-19, 2010, pp.37-46.

[8] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *Proceedings of the nineteenth ACM symposium on Operating Systems Principles (SOSP'03)*, Lake George, New York, USA, October 19-22, 2003, pp.164-177.

[9] S. M. Blackburn, R. Garner, C. Hoffman, A. M. Khan, K. S. McKinley, R. Bentzur, A. Diwan, D. Feinberg, D. Frampton, S. Z. Guyer, M. Hirzel, A. Hosking, M. Jump, H. Lee, J. E. B. Moss, A. Phansalkar, D. Stefanovi´c, T. VanDrunen, D. von Dincklage, and B. Wiedermann. The DaCapo Benchmarks: Java Benchmarking Development and Analysis. In *Proceedings of the 21st annual ACM SIGPLAN conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'06)*, Portland, OR, USA, October 22-26, 2006

[10] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live Migration of Virtual Machines. In *Proceedings of Second Symposium Networked Systems Design and Implementation (NSDI'05)*, May 2-4, 2005, pp. 273-286

[11] D. Comer, Internetworking with TCP/IP. Page 226. Upper Saddle River, N.J.: Prentice Hall, 2000. Print.

[12] J. Choi, S. Govindan, B. Urgaonkar and A. Sivasubramaniam. Profiling, Prediction, and Capping of Power Consumption in Consolidated Environments. In *Proceedings of IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'08)*, Baltimore, MD. USA, Sept 8-10, 2008, pp.1-10.

[13] M. Cardosa, M. R. Korupolu and A. Singh. Shares and Utilities Based Power Consolidation in Virtualized Server Environments. In *Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management (IM'09)*, New York, NY, USA, June 1-5, 2009, pp. 327-334.

[14] S. Fu and C-Z. Xu, "Stochastic Modeling and Analysis of Hybrid Mobility in Reconfigurable Distributed Virtual Machines", *Journal of Parallel and Distributed Computing*, Vol. 66, No. 11, November 2006, pp.1442-1454.

[15] J. Gong and C-Z. Xu. A Gray-box Feedback Control Approach for System-Level Peak Power Management. In *Proceedings of 39th International Conference on Parallel Processing (ICPP'10)*, San Diego, CA, USA, September 13-16, 2010, pp.555-564.

[16] J. Gong and C-Z. Xu. vPnP: Automated Coordination of Power and Performance in Virtualized Datacenters. In *The IEEE International Workshop on Quality of Service (IWQoS'10)*, Beijing, China, June 16-18, 2010, pp.1-9.

[17] L. Hu, H. Jin, X. Liao, X. Xiong and H. Liu. Magnet: A Novel Scheduling Policy for Power Reduction in Cluster with Virtual

Machines. In *Proceeding of 2008 IEEE International Conference on Cluster Computing (Cluster'08)*, September 29- October 1 2008, Tsukuba, Japan, pp.13-22.

[18] M. Hines and K. Gopalan. Post-Copy Based Live Virtual Machine Migration Using Adaptive Pre-Paging And Dynamic Self-Ballooning. In *International Conference on Virtual Execution Environments (VEE'09)*, Washington DC, USA, March 11-13, 2009, pp.51-60.

[19] B. Krishnan, H. Amur, A. Gavrilovska, K. Schwan. VM Power Metering: Feasibility and Challenges*. In The Second Green Metrics Workshop, in conjunction with SIGMETRICS'10,* New York, NY, USA, June 14, 2010.

[20] A. Kangarlou, P. Eugster, D. Xu. VNsnap: Taking Snapshots of Virtual Networked Environments with Minimal Downtime. In *Proceedings of IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'09)*, Estoril, Portugal, June 29- July 2, 2009, pp.524-533.

[21] M. Y. Lim, F. Rawson, T. Bletsch, and V. W. Freeh. PADD: Power Aware Domain Distribution. *In Proceedings of the 29th IEEE International Conference on Distributed Computing Systems (ICDCS'09)*, Montreal, Quebec, Canada, June 22-26, 2009, pp.239-247.

[22] H. Liu, H. Jin, X. Liao, L. Hu and C. Yu. Live Migration of Virtual Machine Based on Full System Trace and Replay. *In Proceedings of the 18th International Symposium on High Performance Distributed Computing (HPDC'09)*, June 11-13, 2009, Munich, Germany, pp.101-110.

[23] J. Lange, K. Pedretti, T. Hudson, P. Dinda, Z. Cui, L. Xia, P. Bridges, A. Gocke, S. Jaconette, M. Levenhagen and R. Brightwell. Palacios and Kitten: New High Performance Operating Systems for Scalable Virtualized and Native Supercomputing. In *Proceedings of the 24th IEEE International Parallel and Distributed Processing Symposium (IPDPS'10)*, Atlanta, Georgia, USA, April 19-23, 2010, pp.1-12.

[24] A. B. Nagarajan, F. Mueller, C. Engelmann, and S. L. Scott. Proactive Fault Tolerance for HPC with Xen Virtualization. In *Proceedings of ACM Annual International Conference on Supercomputing (ICS'07)*, Seattle, Washington, USA, June 17-21, 2007, pp. 23-32.

[25] M. Nelson, B. H. Lim, and G. Hutchins. Fast Transparent Migration for Virtual Machines. In *Proceedings of USENIX Annual Technical Conference (USENIX'05)*, Anaheim, California, USA, April 10-15, 2005, pp.391-394.

[26] R. Nathuji and K. Schwan. Virtual Power: Coordinated Power Management in Virtualized Enterprise Systems. In *Proceedings of ACM Symposium on Operating Systems Principles (SOSP'07)*, Stevenson, WA, USA, October 14-17, 2007.

[27] I. Rodero, J. Jaramillo, A. Quiroz, M. Parashar and F. Guim. Towards Energy-aware Autonomic Provisioning for Virtualized Environments. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC'10)*, Chicago, Illinois, USA, June 20-25, 2010, pp. 320-323.

[28] K. Sato, H. Sato, S. Matsuoka. Model-based Optimization for Data-intensive Application on Virtual Cluster. In *The 9th IEEE/ACM International Conference on Grid Computing (Grid'08),* Tsukuba, Japan, pp.367-368

[29] B. Sotomayor, K. Keahey and I. Foster. Combining Batch Execution and Leasing Using Virtual Machines. In *Proceedings of the Eighteenth International Symposium on High Performance Distributed Computing (HPDC'08)*, June 23–27, 2008, Boston, MA, USA, pp.87-96.

[30] M. Tarighi, S. A. Motamedi and S. Sharifian. "A New Model for Virtual Machine Migration in Virtualized Cluster Server Based on Fuzzy Decision Making", *Journal of Telecommunications*, Vol.1, No.1, February 2010, pp.40-51.

[31] A. Verma, P. Ahuja, and A. Neogi. pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems. *In Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware (Middleware'08*), Springer-Verlag, Leuven, Belgium, December 1-5, 2008, pp.243-264.

[32] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya. Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation. In *Proceedings of the 1st International Conference on Cloud Computing*, Lecture Notes In Computer Science, Beijing, China, December, 2009, pp.254-265.

[33] T. Wood, P. Shenoy, A. Venkataramani and M. Yousif. Black-box and Gray-box Strategies for Virtual Machine Migration. In *Proceedings of 4th USENIX Symposium on Networked Systems Design and Implementation (NSDI'07),* Cambridge, MA, USA, April 11-13, 2007, pp. 229-242.