Migration Energy-Aware Workload Consolidation in Enterprise Clouds

Mohammad M. Hossain* Jen-Cheng Huang† Hsien-Hsin S. Lee†
*College of Computing †School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA 30332
{mhossain7, tommy24, leehs}@gatech.edu

***Abstract*—Consolidation through live VM migrations is a promising approach to improve server utilization. However, prior consolidation works have focused mostly on the performance impact of migration and neglected the associated energy overhead. Our research finds that energy impact of migration can offset over 12% of the energy saved through energy-conscious workload packing. To address this limitation of the current state of research, in this paper we devise new schemes to pack applications that targets a joint optimization of energy and performance overhead of VM migrations. Additionally, we develop a statistical workload modeling technique for simulating consolidation problem in enterprise cloud contexts. Our experiments with statistically generated synthetic trace and Google's production trace demonstrate that our schemes can improve energy savings up to 23% compared to state of the art power-aware workload consolidation strategy.**

***Keywords*-VM migration energy; energy-efficient consolidation; cloud resource management.**

I. Introduction

Energy related costs are predicted to be the single largest contributor of the operational cost in enterprise clouds. Inherently, IT resources consume power in *energy disproportional* manner. A key source of energy inefficiency comes from the idle energy consumption by underutilized server components. Low server utilization in production datacenters [1, 2] stems from over-provisioning the server farm's capacity to cope with infrequent spikes of request. To mitigate this energy wastage, adaptive resource provisioning through dynamic VM resizing and live VM migrations is used to consolidate applications on a small group of physical hosts, and the rest servers are decommissioned [3, 4].

Workloads' dynamism in enterprise clouds makes consolidation a very challenging task. Factors that constrain consolidation efficacy can be classified in direct and indirect overhead of VM migrations. The former component includes time taken for VM transfer, additional power incurred by migration operation, performance loss during the downtime phase of a VM migration, and so on. The later part reflects the penalty caused to co-located applications of the migrant workload that is hard to be modeled in an application agnostic manner, and thus falls beyond the scope of current work. The specific contributions of this paper are as follows:

- We address the energy impact of workload migration in consolidation perspective. Our findings reveal that instantaneous power amplifications at the source and the target servers during the period of migration mostly depend on the CPU-utilization levels of the associated hosts.
- We propose novel heuristics that leverage workloads' resource usages and servers' power profiles to make the consolidation schedule more energy-effective.
- For characterizing runtime task resource usages, we develop a statistical workload modeling technique for simulating different consolidation schemes. We validate our model by comprehensive experiments with Google's production trace [5] comprising over 12,500 servers.

The rest of this paper is structured as follows. Section II presents background and motivation, and Section III discusses relevant research works. Migration energy-cost aware heuristics are detailed in Sections IV. Section V describes our statistical task resource-usage modeling. Section VI analyzes the experimental results, and Section VII concludes our work.

II. Background and Motivation

A. Energy Overhead of VM migration

A precise analytical model for measuring energy cost of migration requires architectural details of the workload and the migration environment [6, 7]. In this paper, our goal is to demonstrate the necessity of incorporating migration energy overhead into consolidation context for large scale cloud settings. Towards that end, we raise the level of abstraction to model the applications, and characterize the workloads based on their resource usages. This raised abstraction of workload modeling facilitates simulating our proposed consolidation schemes and validating the results against production datacenter traces.

To analyze the migration energy overhead for different consolidation policies, we consider two key parameters: (a) *Migration Power* - the instantaneous power hike at the source and the destination servers during the migration period, and (b) *Migration Time* - time to transfer an application's memory footprint from the source server to the destination server. For each migration, migration power is multiplied with migration time to give an estimation of migration energy.

First, to measure migration power, we set up a testbed with two servers - one used as the source and the other used as the destination of the migration; both attached with WattsUp Pro meters to collect power consumption readings.

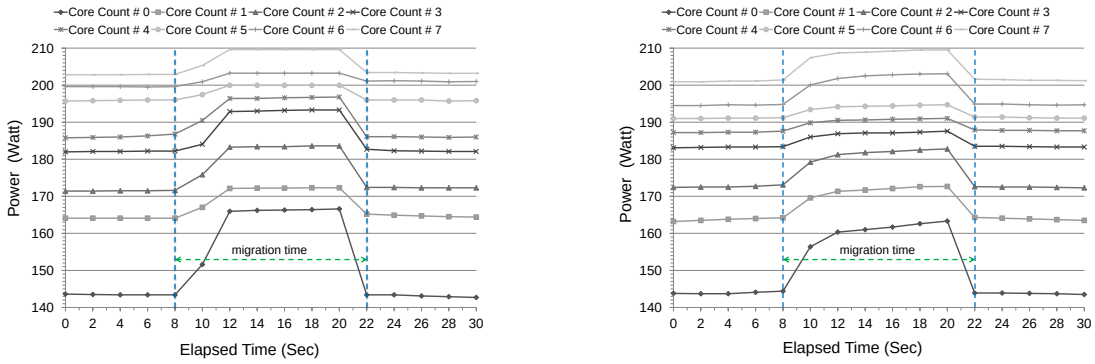

Figure 1. Migration Power Consumption Overhead at Destination and Source Servers

As the migration workload, we choose an idle guest image of Ubuntu OS configured with 2 VCPUs and 1.2GB RAM. As CPU usage dominantly impacts the system power consumption, we measure the migration power overhead at different resource consumption levels by tuning the CPU usages on the source and the destination servers. CPU utilization is tuned at the core granularity. As the server platforms in our experiment are eight-core machines, we tune the CPU usage level by launching increasing instances of CPU-intensive LAPACK[1] benchmarks (each instance being mapped to execute on an individual core).

Fig. 1(a) depicts the instantaneous power elevation at the destination server during the VM migration, when the destination's CPU is tuned by launching 0, 1, 2...up to 7 instances of LAPACK benchmark. A maximum peak of 23 watts is observed when the destination server remains idle, and migration is triggered by the hypervisor. The amplitude of power elevation due to migration gradually decreases as more cores are tuned on (for example, power increased by 7 watts when 7 cores are active), and hence the processor's architectural states are warmed up already. A similar trend is observed at the source end of the migration as shown in Fig. 1(b). This time the source server's CPU usage is modulated by launching 0, 1, 2...up to 7 instances of the same benchmark, while the VM migration is initiated by the hypervisor at the source end. Similar to the destination side, we see a maximum power elevation of 19 watts when the source server triggers the migration from complete idle state. Fig. 1(b) demonstrates that the amplitude of power hike at the source side is lower than that at the destination end. This is perhaps due to reason that migration operation is initiated at the source server, and so better optimization for data transfer and network traffic management is available on this end compared to the destination side.

Now, to measure VM migration time at different resource consumption levels, we observe that migration duration in all cases is almost constant as long as the VM image's size and underlying network interface's bandwidth remain unchanged. This is consistent to prior studies [8, 9, 3] which establish that migration time can be modeled as a linear function of the VM size. We follow a similar scheme for modeling the VM migration time. As live migration exerts additional load on the datacenter's network infrastructure, most virtualized deployments utilize an entirely different network and separate NIC per host to accommodate live migration traffic [10].

B. Performance Impact (Slowdown) of VM Migration

During VM Migration, performance is compromised while the application is transparently moving from the source to the destination. During the final phase of VM transfer, the application is suspended entirely, *i.e.* no CPU-cycles are granted to the migratory application. Suspension resumes after the dirty pages in penultimate phase are moved to the destination. While the migration period is usually tens of seconds, this suspension duration is in the order of tens to hundreds of milliseconds [6]. In this paper, we attribute this downtime duration times the application's CPU-usage as the performance penalty to the associated VM. This formulation takes the first order impact of migration on the throughput loss, while the second order performance overhead comes from cold start on the destination end after resumption. Modeling the second order impacts need knowledge about micro-architectural states of the application before and after the migration, and hence falls beyond the scope of this work.

C. Server Power Model

Server power modeling traditionally focused on developing a power profile that maps CPU utilization to a certain power usage. In this paper, we use the liner model validated by [1] and [11]:

$$P_{dyn} = P_{idle} + (P_{max} - P_{idle}) * CPU_{dyn} \quad (1)$$

[1]Linear Algebra PACKage is a software library for numerical linear algebra.

In our experiment, we measure $P_{idle} = 140W$ and $P_{max} = 210W$ by running LAPACK benchmarks to load 100% CPU stress.

D. Bin-Packing Heuristics for Consolidation Problem

Consolidation problem can be modeled as a variation of multidimensional bin-packing problem. Categorized as an NP-hard problem, different heuristics and linear programming based solutions are used for getting a near-optimal solution. Several proposed VM packing algorithms use a variant of First Fit Decreasing (FFD) heuristic. In FFD, servers on the system are sorted in descending order of their spare resource capacity, and then under-utilized servers are selected to unload their VMs to migrate to the other-end servers. Thus, VMs are consolidated on small number of active servers, each operating at elevated utilization levels. To keep the execution overhead very modest, our consolidation schemes are constrained to work with single dimension (*i.e.* CPU usage).

III. Related Work

In recent years, extensive researches are conducted to optimize either power or performance in datacenters. Perhaps the most relevant work is pMapper [3] that ranks the servers based on power-efficiency index, which is defined as the marginal increase in power for marginal increase in resource (*i.e.* CPU) capacity. As server power is modeled as a linear function of its CPU usage, the power-efficiency index in pMapper is equivalent to the ratio of the peak power to the CPU capacity of a server. For an application placement, pMapper first selects the server with the least power increase per unit increase in CPU usage. To ensure that servers are packed with minimal fragmentation, pMapper uses a variant of FFD algorithm. On several grounds our work varies from pMapper. pMapper does not consider energy overhead of migration; it translates the cost of migration to throughput-loss only. Additionally, pMapper's approach of ranking power-efficient servers falls short when the peak power and the peak capacity of all servers are same. In such scenario, pMapper turns to simple FFD scheme that performs non-optimally for energy-aware application packing.

Liu [7] *et al.* is the first to investigate the performance and energy overhead of live migrations combinedly. By exploiting the OS and hypervisor level architectural details, their result shows that a precise prediction can be made to quantify the impact of VM migrations within a modest range of modeling error. However, their work does not delve into the findings that migration power significantly vary due to the run-time power profile of the source and the destination servers. Besides, they do not address how to leverage the migration's energy impact to devise energy-efficient VM packing schemes. Moreover, as this analytical prediction model requires access to OS and hypervisor for collecting statistics, such as, page dirtying rate, this is not feasible to be validated against production datacenter traces.

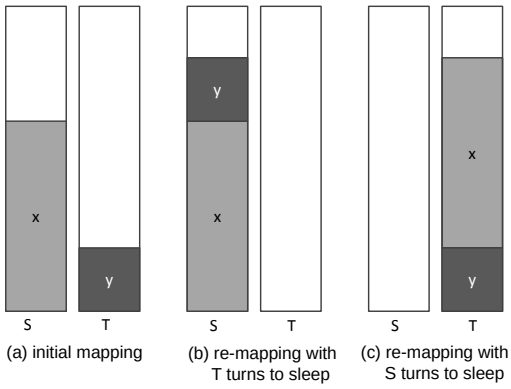

Figure 2. **Schematic Consolidation of Two Workloads**

IV. Migration Energy-Aware VM Packing

A. Design Parameters

Before presenting the details of the proposed migration energy-aware task packing schemes, we delineate the basic principle behind the development of these heuristics. For simplicity of illustration, we consider an initial mapping of Fig. 2(a), where two servers *S* and *T* each hosts one VM *x* and *y* respectively. After the consolidation, the two possible configurations are shown in Fig. 2(b) and Fig. 2(c). Now, to develop underlying design parameters for migration energy-aware application packing, we consider two scenarios below.

Case 1: power profiles of *S* and *T* are same, *i.e.* for a particular CPU utilization level, power usage value on *S*, *P*(*S*) and power usage value on *T*, *P*(*T*) are exactly same.

In this scenario, after the migration of workload *y* in Fig. 2(b) (or, workload *x* in Fig. 2(c)), one of the servers hosts both VMs and the other is switched off. As the aggregated CPU usage on server *S* in Fig 2(b) and server *T* in Fig 2(c) are exactly same and *P*(*S*) = *P*(*T*), both of the final configurations consume exactly same power. In such case, if migration energy of *x*, $E_m(x)$ is greater than migration energy of *y*, $E_m(y)$, the net energy preservation from the mapping of Fig 2(b) leads to higher savings.

Case 2: power profiles of *S* and *T* are not same. In such scenario, energy-efficiency of final configuration depends on migration energy of workloads *x* and *y*, and power profiles of *S* and *T*.

First, we consider the consolidation of Fig. 2(b). Let, E_s and E_t denote *S* and *T*'s energy usage in initial mapping respectively, and $E^{'}{}_s$ denotes *S*'s energy usage in consolidated mapping. Thus, total energy savings for final mapping in Fig. 2(b) = $(E_s + E_t)$ - $(E_m(y) + E^{'}{}_s) = E_t$ - $E_m(y)$ - $(E^{'}{}_s$ - $E_s)$. In a similar approach, total energy savings for final mapping of Fig. 2(c) can be estimated. The higher of these two measurements is preferred for the consolidation.

B. Consolidation Heuristics

Our proposed heuristics assign an energy saving score to a VM (or to a server) that indicates its effectiveness to save energy with less performance impact. Higher scored VMs (or servers) are selected first to schedule in consolidation, and successively they get packed with lower scored VMs (or servers). This strategy as a variant of FFD guarantees very modest resource fragmentation, while simultaneously considers the energy and performance overhead of migrations. The point to be noted here is that — the destination server's power profile is not included in the heuristics, because *at the commencement* of the schedule processing and *during the middle stage* of the schedule preparation the target's power profile may have been changed due to intermediate migrations mapped to the destination already.

1) Task Energy Saving Index (TESI) Heuristic: This heuristic assigns a score to *each VM*, with higher value indicates the VM's aptitude to save energy by getting migrated from its current host. TESI score is computed to reflect both maximizing energy savings and minimizing performance slowdown. For a workload *x* hosted on server *S*, TESI score is computed as in Eq.(2). The first component, E_S, in the numerator amounts energy savings from *S*, when it is decommissioned (after all its VMs are migrated). Eq.(1) is used to compute this value with CPU_{dyn} accounts for aggregated CPU usages of all VMs. The subtrahend part, $E_m(x)$, accounts the energy cost for migrating *x*, which is measured from the experimental findings of Fig. 1(a) and Fig. 1(b). Finally, the denominator, *Slowdown*(*x*), represents the performance impact on the application incurred by VM migration as detailed in Section II-B.

$$TESI(x) = \frac{E_S - E_m(x)}{Slowdown(x)} \tag{2}$$

2) Server Energy Saving Index (SESI) Heuristic: This heuristic assigns a score to *each server* that captures the server's capability to preserve higher energy. Server with the highest SESI score is selected first, and all its hosted VMs are scheduled for migration. Eq.(5) formulates the computation of SESI score for a server *S*. In Eq.(5), E_S denotes the same as in TESI. The second part in the numerator sums up the migration energy for all VMs mapped to *S*, and the denominator represents the combined performance impact.

$$E_m(S) = \sum_{x \in \{\text{VMs hosted on } S\}} E_m(x) \tag{3}$$

$$Slowdown(S) = \sum_{x \in \{\text{VMs hosted on } S\}} Slowdown(x) \tag{4}$$

$$SESI(S) = \frac{E_S - E_m(S)}{Slowdown(S)} \tag{5}$$

3) Hybrid Heuristic: From a VM's perspective, TESI heuristic schedules migrating higher TESI-scored VMs first. TESI offers modest slowdown, but leads to poor energy-efficiency. Contrarily, from a server's perspective, SESI heuristic schedules all VMs of a selected server at a time. SESI offers good energy effective packing, but incurs higher throughput loss. To exploit both benefits from TESI and SESI, in this hybrid approach, at first the TESI score is calculated for each VM. Similar to TESI, the highest TESI-scored VM is chosen for migration first. Now alike SESI, *all co-located VMs* with this selected VM are added to the consolidation schedule, before considering *the second-highest TESI-scored VM*.

V. Statistical Characterization of Task Resource Usage

In this section, our goal is to develop a resource-usage characterization model that is sufficiently accurate for producing synthetic workload trace. In this paper, we focus on characterizing run-time task resource usage for CPU and memory. To that direction, we first discuss on the underlying probability distribution used for generating the random samples. Later, we detail the process of parameter selection for the probability distributions that is used for synthetic trace generation. Finally, we validate proposed statistical characterization of task resource-usage against a historical Google trace [5].

In our implementation, we characterize each workload on basis of two major resource dimensions — CPU usage and memory usage. Normal distribution, which is characterized by two parameters — mean(μ) and variance(σ), is used to generate random samples along any resource dimension. The *empirical rule* of normal distribution, also known as the *68-95-99.7 rule*, states that when samples are drawn for a particular normal distribution, about 68%, 95% and 99.7% of the samples would lie within one, two and three standard deviations of the mean, respectively. So, for a given parameter of (μ, σ), the samples generated by the normal distribution lie in the range $[\mu - 3 * \sigma, \mu + 3 * \sigma]$, where $\mu - 3 * \sigma$ is the lower boundary and $\mu + 3 * \sigma$ is the upper boundary of the range.

The *68-95-99.7 rule* entails that if we use a single normal distribution for a resource dimension, such as, CPU usage, over two-thirds of the samples would be in close proximity around the mean. This would lead to limited variation in workloads' CPU-usage patterns that misrepresents the tasks' resource demands in production clouds. To alleviate this constraint imposed by using a single normal distribution, we employ multiple classes of distribution along any resource dimension. Therefore, we sub-divide each of the two resource dimensions into an arbitrary number of groups to model wide variability in the resource demand of all workloads. For example, we may divide CPU-usage dimension into four groups and memory-usage dimension

into six groups. For any particular resource dimension, each group represents a non-overlapped range for the samples fallen into that particular group. Hence, to formulate the probability distribution function of normal distribution for samples resided in a specific group, an appropriate mean and variance are set such that samples are constrained to fall into the range $[\mu-3*\sigma, \mu+3*\sigma]$. The number of groups along any resource dimension is a tunable parameter. By classifying applications in arbitrary number of groups, and choosing different mean and variance for the normal distribution of different groups, we are able to faithfully model the diverse nature of resource usages in modern datacenter workloads. Accordingly, our synthetic trace is generated based on:

1) mixture of different resource intensive workloads, and
2) parameters for normal probability distribution of — (a) CPU-usage value of each of the CPU groups, (b) memory-usage value of each of the memory groups.

Finally, to validate the statistical characterization of runtime task resource usage for a production Google trace [5], we divide the resource (CPU and memory) usage spectrum in twenty consecutive non-overlapped ranges (*i.e.* twenty groups for each resource dimension), R_1, R_2, $\ldots R_{20}$, such that each of the range contains five percent of the total samples. For each range R_i, let L_i and U_i be the lower and the upper boundary of R_i. Then, we compute the mean(μ_i) and the variance(σ_i) for each range R_i as below:

$$\mu_i = \frac{U_i + L_i}{2} \text{ and } \sigma_i = \frac{U_i - L_i}{6} \tag{6}$$

Next, for each range R_i, random variables are generated for five percent of the samples that reside in corresponding group; while the distribution of the samples are normal with parameters μ_i and σ_i as computed from Eq.(6). Now, for each task record of the Google trace, we first identify which group the usage-level for particular resource (either CPU or memory) a record maps to. Then, for each resource type we modify the trace by over-writing the actual task resource usage by a random sample drawn from respective group of generated samples. For example, if a task's CPU and memory usages are 0.20 and 0.05 respectively (both are normalized with respect to corresponding resource capacity), and if these usage-levels map to CPU group R_4 and memory group R_1 respectively; then, we replace the original resource usage value by CPU group R_4's and memory group R_1's normally-distributed sample. The other components of the workload, such as, arrival time and completion time are kept intact. In the final step of validation, we experiment different consolidation schemes on the historical trace and the modified trace, and compare the final consolidation state. We model the error as the discrepancy between the number of consolidated servers in two experiments. Our results show that the final consolidations with the statistically generated samples for all packing techniques, *i.e.* pMapper, TESI, SESI and Hybrid, very closely match (absolute error within 0.2 - 0.9%) with the outcomes of consolidation state using the historical trace from Google cluster.

VI. Evaluation

A. Experiments with Synthetic Workload Trace

To configure a datacenter's workloads' resources setting, the required parameters are enlisted in Table I and Table II. As explained in Section V, these parameters represent the classes of distribution for two resource dimensions — CPU usage and memory usage, along with representing a particular combination of different-resource intensive workloads. For example, Table I shows that out of all generated workload samples, consecutive CPU group 1-4 contains 60%, 20%, 10% and 10% of the samples, respectively. Each row in Table I represents parameters for one particular group of CPU-usage. For each group, corresponding range for the samples' value is listed alongside. We assume four CPU groups and four memory groups, and server's CPU capacity is set to 8 cores and memory capacity is set to 8GB.

CPU Groups	Fraction of Samples (%)	Mean	Variance	Range for Samples [CPU cores]
1	60	0.25	0.083	(0.0, 0.5)
2	20	1.00	0.167	(0.5, 1.5)
3	10	2.25	0.250	(1.5, 3.0)
4	10	4.00	0.334	(3.0, 5.0)

Table I
CPU Usage Distribution Parameters.

Memory Groups	Fraction of Samples (%)	Mean	Variance	Range for Samples [MB]
1	25	30	10	(0, 60)
2	25	90	10	(60, 120)
3	25	180	20	(120, 240)
4	25	300	20	(240, 360)

Table II
Memory Usage Distribution Parameters.

First, we configure three different datacenter settings by varying normal distribution's parameter (μ, σ) for memory-usage. In these three configurations, termed as C_1, C_2 and C_3, normal distribution parameters for CPU-usage are same as Table I. Now, for datacenter configuration C_1, memory-usage values are distributed as per Table II. For these assigned parameters to memory-usage distribution, the average memory-usage per task stands at 150 MB (computed as weighted mean of μ values in Table II). For, configurations C_2 and C_3, the μ and σ values are multiplied by 3 and 6 respectively. Hence, average memory-usage per task in C_2 and C_3 are 450 MB and 900 MB respectively.

Fig. 3(a) presents the server energy savings (with respect to default workload mapping) of pMapper, TESI, SESI and Hybrid schemes for above three configurations. Normalized to the savings of pMapper, the savings from SESI and Hybrid are translated to 19 - 23% and 9 - 11% gain, respectively. As shown, all four heuristics demonstrate declining energy savings with datacenter setting moved from C_1 to C_3. Higher VM memory footprint results in larger

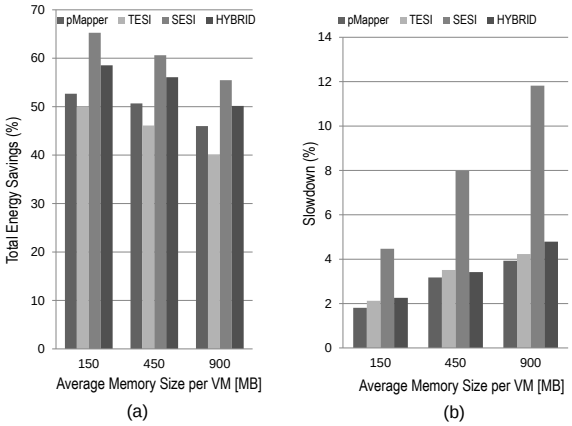

Figure 3. **Synthetic Trace Results with pMapper, TESI, SESI and Hybrid**

fragmentation, which entails lesser opportunity for compact consolidation. TESI being a task-grained migration policy is more sensitive to fragmentation issue. Next, Fig. 3(b) depicts performance slowdown impacted by VM migrations in different consolidation policies. SESI being a server-grained workload unloading scheme, disregard individual VM's parameters for preparing the consolidation schedule. This results in the highest slowdown in SESI. Thus, SESI is preferred due to highest energy-savings potentiality until average VM size becomes too large. And, for higher VM-size, Hybrid is preferred as it makes a good balance between higher energy preservation from SESI and more tolerable performance degradation from TESI.

B. Experiments with Google Cluster Trace

For better understanding of relative merits and limitations of pMapper, TESI, SESI and Hybrid consolidation, we experiment with recently released Google's utilization trace across a large compute cluster. For limited simulation time budget, we restrict the trace size upto 6 hours. During this period, a total of 1,647,757 tasks are submitted to the cluster. The task usage record keeps information of the submitted time, along with average CPU usage and memory usage. To conceal internals of the system, most resource measurements are normalized. The normalization is a scaling relative to the largest capacity of the resource on any server in the cluster. Results of different consolidation policies on Google trace are shown in Fig. 4. For the six hours long trace, the average normalized CPU-usage is 0.0264762 and normalized memory-usage is 0.0284707 across all the tasks. For a memory capacity of 16 GB, per task memory usage is translated to 466 MB. Fig. 4(a) depicts total energy savings for different schemes that translates to a relative energy savings of 22% and 13% for SESI and Hybrid policy (with respect to pMapper), respectively. On Google trace TESI performs as good as pMapper — this happens due to the existence of regular resource usage pattern among large

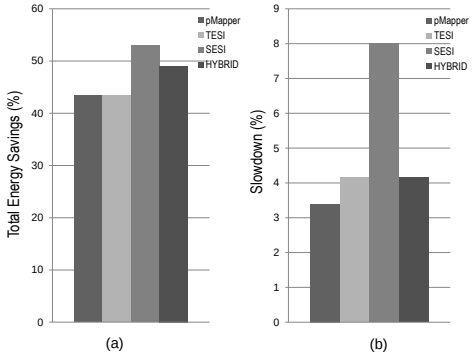

Figure 4. **Google Trace Results with pMapper, TESI, SESI and Hybrid**

number of tasks. Slowdown results are presented in Fig. 4(b) which closely match with the results of synthetic trace for datacenter setting with VM size of 450 MB in Fig. 3(b).

VII. Conclusion

This paper explores the migration energy overhead in the context of application consolidation within enterprise datacenters. We propose three novel application-packing heuristics. These consolidation schemes faithfully capture the energy overhead and performance impact caused by VM migration. Our extensive experiments using statistically generated synthetic trace guide to better understand the comparative analysis among proposed application-packing heuristics. Experimental results using Google trace reliably validate the stability, accuracy and practicability of the synthetic trace-driven consolidation methodology.

References

[1] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *ISCA-34*, 2007.

[2] D. Meisner, B. T. Gold, and T. F. Wenisch, "Powernap: eliminating server idle power," in *ASPLOS-14*, 2009.

[3] A. Verma, P. Ahuja, and A. Neogi, "pmapper: power and migration cost aware application placement in virtualized systems," in *Middleware*, 2008.

[4] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," in *HotPower*, 2008.

[5] http://code.google.com/p/googleclusterdata.

[6] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *NSDI*, 2005.

[7] H. Liu, C.-Z. Xu, H. Jin, J. Gong, and X. Liao, "Performance and energy modeling for live migration of virtual machines," in *HPDC-20*, 2011.

[8] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale: elastic resource scaling for multi-tenant cloud systems," in *ACM SOCC*, 2011.

[9] F. Hermenier, X. Lorca, J.-M. Menaud, G. Muller, and J. Lawall, "Entropy: a consolidation manager for clusters," in *ACM VEE*, 2009.

[10] V. Soundararajan and J. M. Anderson, "The impact of management operations on virtualized datacenter," in *ISCA-37*, 2010.

[11] S. Rivoire, P. Ranganathan, and C. Kozyrakis, "A comparison of high-level full-system power models," in *HotPower*, 2008.