Energy-Aware Application-Centric VM Allocation for HPC Workloads

H. Viswanathan, E.K. Lee, I. Rodero, D. Pompili, M. Parashar
NSF Center for Autonomic Computing
Rutgers University, New Brunswick, NJ
{hari_viswanathan, eunkyung_lee, irodero, pompili, parashar}@cac.rutgers.edu

M. Gamell
Open University of Catalonia
Barcelona, Spain
marcgamell@uoc.edu

***Abstract*—Virtualized datacenters and clouds are being increasingly considered for traditional High-Performance Computing (HPC) workloads that have typically targeted Grids and conventional HPC platforms. However, maximizing energy efficiency, cost-effectiveness, and utilization of datacenter resources while ensuring performance and other Quality of Service (QoS) guarantees for HPC applications requires careful consideration of important and extremely challenging tradeoffs.**

An innovative application-centric energy-aware strategy for Virtual Machine (VM) allocation is presented. The proposed strategy ensures high resource utilization and energy efficiency through VM consolidation while satisfying application QoS. While existing VM allocation solutions are aimed at satisfying only the resource utilization requirements of applications along only one dimension (CPU utilization), the proposed approach is more generic as it employs knowledge obtained through application profiling along multiple dimensions. The results of our evaluation show that the proposed VM allocation strategy enables significant reduction either in energy consumption or in execution time, depending on the optimization goals.

I. Introduction

Virtualized datacenters and clouds provide the abstraction of nearly-unlimited computing resources through the elastic use of consolidated resource pools. These platforms are being increasingly considered for traditional High-Performance Computing (HPC) workloads that have typically targeted Grids and conventional HPC platforms. The scale and overall complexity of modern datacenters are growing at an alarming rate (current datacenters contain tens to hundreds of thousands of computing and storage devices running complex applications) and, hence, energy consumption, heat generation, and cooling requirements have become critical concerns both in terms of the growing operating costs as well as their environmental and societal impacts [1]. Addressing these concerns while balancing multiple requirements, including performance, Quality of Service (QoS), and reliability, is thus an important task as it can help service providers increase profitability by reducing operational costs and environmental impact without increasing the number of Service Level Agreement (SLA) violations.

However, maximizing energy efficiency, cost effectiveness, and utilization of datacenter resources while ensuring performance and other QoS guarantees for HPC applications requires careful consideration of important and extremely challenging tradeoffs. These include, for example, the tradeoff between the need to proactively provision and allocate Virtual Machines (VMs) on datacenter resources (requiring prior knowledge of resource requirements) and the need to accommodate the heterogeneous and dynamic resource demands and runtimes of these applications. In virtualized HPC datacenters, one or more VMs are created for every workload or job request and each VM is provisioned with resources that sufficiently satisfy the workload QoS requirements, which are based on SLAs. Once VMs are provisioned, they have to be allocated to servers.

The problem of VM allocation arises in the following two situations: initial mapping of VMs to physical servers and migration of VMs from one physical server to another either in anticipation of (proactive) [2] or in response to (reactive) undesired thermal behavior (e.g., equipment overheating) and/or SLA violations. In our previous work [3], we studied different reactive thermal management techniques (including VM migration) for virtualized datacenters from the energy perspective. We concluded that an application-centric energy-aware allocation model for VMs can help reduce the number of SLA violations, avoid undesired thermal behavior, and minimize the energy costs by improving resource utilization and by avoiding costly VM migrations. In this paper, we present an innovative application-centric energy-aware strategy for VM allocation that ensures high resource utilization and energy efficiency through VM consolidation while satisfying application QoS. Minimizing the number of servers that are in operation (by increasing the utilization of active computing resources) through VM consolidation will help reduce the energy consumption for computation and, hence, the total energy consumption of a datacenter.

Along with the growing operational costs associated to energy consumption for computing as well as cooling, over-provisioning of VM resources based on peak application demand is another major issue for cloud service providers as it makes VM consolidation more difficult and reduces efficiency of resource utilization [4]. Static VM allocation techniques based on bin-packing heuristics [5] as well as dynamic techniques (for handling variations in VM's utilization requirements) based on live VM migrations [6], [7], [8], [2], [3] (both proactive and reactive) and dynamic server resource provisioning [9], [10], are aimed at satisfying only the resource utilization requirement of an application along only one dimension (CPU utilization). In contrast, our approach to VM consolidation is application centric as in [11], [12], [13] and, in addition, considers the application's resource utilization requirements along multiple dimensions, i.e., CPU, memory, disk I/O, and network subsystems. Application awareness, which in our solution is acquired through HPC benchmarks profiling, enables the VM allocation algorithm to co-locate compatible VMs on servers during consolidation, thus minimizing the usual adverse effects of resource contention and virtualization overhead on application performance while retaining the benefits of server consolidation.

Our application-centric proactive VM allocation based on an empirical model can significantly contribute to energy efficiency (average 12% reduction in energy consumption compared to the traditional first-fit approach with and without VM multiplexing) and/or optimization of the system performance (18% shorter execution times compared to the traditional first-fit approach due to fewer contentions for resources) depending on the optimization goals. The optimization goal can be minimization of energy consumption, minimization of the execution time, or a weighted combination of both. The results we obtained encourage us to extend our work in future i) to consider thermal efficiency in VM allocation, and ii) to support heterogeneous server hardware.

The main contributions of this paper are summarized as follows: we (i) create a VM allocation model empirically from data obtained by running HPC workloads extensively on a system with a general-purpose rack server configuration, (ii) develop a proactive application-centric VM allocation algorithm that uses this empirical model, (iii) validate our approach through extensive simulations and quantify performance gains in terms of execution time and energy consumption.

The rest of the paper is organized as follows. In Sect. II, we discuss background and related work. In Sect. III, we describe our empirical VM allocation model and present our VM allocation algorithm that uses this model. In Sect. IV, we discuss our evaluation methodology as well as the results we obtained. Finally, in Sect. V, we conclude the paper and outline directions for future work.

II. Related Work

Typically, VM consolidation techniques involve filling up physical servers with VMs (using heuristics like first fit, best fit, etc.) until high server subsystem (CPU, memory, disk storage, network interface) utilization is achieved while still ensuring that the individual VM's subsystem utilization requirements are met. Apparao et al. [14] present a study on the impact of consolidating several applications on a single server running Xen. Server resource provisioning or VM allocation can be static or dynamic. It is static when a VM is being allocated physical resources for the first time and the problem of VM allocation reduces to a deterministic or statistical bin/vector-packing problem [5] depending on how the VM utilization requirement is characterized. In the dynamic case, VMs are first consolidated using any simple bin-packing heuristic and the variations in VM's utilization requirements are handled through live VM migrations [6], [15], [7], [8], [2] or through dynamic server resource provisioning [9], [10] whenever necessary.

As mentioned earlier, VM migrations are performed either reactively [3] or proactively [2] in such a way as to avoid equipment overheating and/or SLA violations. Kochut et al. [16] provide an estimate of the expected improvement in response time due to a migration decision and determines which VMs are best candidates to be placed together. In [8], the authors determine the order in which the VM migrations should occur in addition to which VMs to migrate so to minimize the impact on application performance in terms of execution time. Stoess et al. [6] developed a multi-tiered infrastructure that enables intra-node virtual CPU (vCPU) migration and inter-node live VM migration for workload consolidation and thermal balancing. To enable power-aware VM migration, Verma et al. [15], [7] investigated job allocation for HPC with the focus on CPU and memory subsystems. Voorsluys et al. [17] present a performance evaluation on the effects of live migration of virtual machines on the performance of applications running inside Xen VMs.

On-demand server resource provisioning techniques monitor the workloads on a set of VMs and adjust the instantaneous resources availed by VMs. Song et al. [18] propose an adaptive and dynamic scheme for adjusting resources (specifically, CPU and memory) among virtual machines on a single server to share the physical resources efficiently. Menasce et al. [9] proposed an autonomic controller and showed how it can be used to dynamically allocate CPUs in virtualized environments with varying workload levels by optimizing a global utility function. Nathuji et al. [10] consider the heterogeneity of the underlying platforms to efficiently map the workloads to the best fitting platforms. In particular, they consider different combinations of processor architecture and memory subsystem. Recently, Meng et al. [19] exploited statistical multiplexing of VMs to enable joint VM provisioning and consolidation based on aggregated capacity needs. However, all the aforementioned VM allocation techniques are aimed at satisfying the resource utilization level guarantees and do not consider the application-level performance (execution time). In contrast, as mentioned earlier, we follow an application-centric and energy-aware approach to VM allocation.

Recently, researchers have started to focus on application/workload-aware VM consolidation that not only achieves all the objectives as its traditional resource-utilization- and energy-aware counterparts but also ensures minimum degradation to application performance due to resource multiplexing and virtualization overhead [11], [12], [13]. In other words, application-centric VM allocation is not only aimed at energy-efficient VM consolidation but also at co-locating VMs that are "compatible" so that further gains can be achieved in terms of energy savings and overhead for virtualization. An application running on a VM and, hence, a VM itself can be labeled CPU-, memory-, storage-, or network/communication-intensive or a combination of those. For example, when only CPU-intensive VMs that do not utilize the disk storage and network interface much are consolidated on a server, additional energy savings can be obtained by shutting down or operating the under-utilized subsystems in low power states as we have shown in [20]. In [11], [12], the authors propose to consolidate VMs with similar memory content on the same hosts for higher memory sharing and Govindan et al. [13] propose to consolidate based on inter-process communication patterns.

In contrast, we follow a generic approach to smart co-location by considering the compatibility among all types of applications or VMs (CPU-, memory-, disk I/O-, and/or network-intensive). We achieve this by running HPC benchmark workloads exhaustively (all possible allocations based on number and type of VMs) on a system with a general-purpose rack server configuration and deriving an empirical model from the raw data for average application performance (execution time), energy consumption, and their tradeoffs. We store the model in a database and exploit it for proactive application-aware VM consolidation. In the following sections, we explain our VM allocation model and discuss the results of our experiments.

III. VM Allocation Model and Implementation

As discussed in the previous section, in this paper, we study how to increase the resource utilization (and, hence, the energy efficiency), i.e., how to maximize the system throughput by allocating the maximum possible number of VMs per node, without penalizing the applications' performance. In other words, the objective of this work is to find out the trade-off between the applications' performance and the overall datacenter energy consumption when different number and combinations of a variety of VMs are allocated to a physical server. The performance of an application is measured in terms of its *average execution time*, which is defined as the ratio of the maximum application execution time when a number of VMs are running simultaneously to the number of VMs. This metric gives an insight into the gains obtained by multiplexing VMs (i.e., running them in parallel) over running them sequentially one after the other. It is important to note that by considering the average execution time of VMs we do not focus on minimizing the execution time of each application individually but we strive to improve the QoS by minimizing the number of SLA violations.

As the best number of VMs per node may be different for different application types (based on their usage of different subsystems), we consider the applications' profiles. Specifically, we focus on finding the best partition and allocation of VMs when the different VMs run applications of different types. In this paper, we assume that the applications' profiles are known in advance (e.g., specified by the user in the job definition). To find the allocation for a set of VMs that best matches energy efficiency/performance goals while ensuring QoS guarantees, we rely on a model based on empirical data from experiments. We have developed a methodology composed of the following steps:

1) Profile a comprehensive set of applications (standard HPC benchmark workloads);
2) Run benchmarks exhaustively (all possible allocations based on number of VMs and application type) and collect data;
3) Create a model (database) with all the data collected during the benchmarking process, including execution time and energy consumed;
4) Implement an algorithm that, given the i) model, ii) an optimization goal (either minimize energy consumption or minimize execution time), iii) a set of servers with their current allocations, and iv) a set of VMs along with their characteristics, returns a set of partitions and allocations of the VMs in the servers.

A. Application profiling

The methodology involves profiling an application's behavior as *I/O-intensive, memory-intensive,* and/or *CPU-intensive* based on its usage of different subsystems. Most of the standard profiling utilities are designed for comparing computation efficiency of the applications on systems on which they are running and, therefore, their outputs are not very useful from the subsystem usage point of view. We profiled standard HPC benchmarks with respect to their behaviors and subsystem usage on individual servers. To collect run-time OS-level metrics for CPU utilization, hard disk I/O, and network I/O we used different mechanisms such as "mpstat", "iostat", "netstat" or "PowerTOP" from Intel. We also patched the Linux kernel 2.6.18 with "perfctr" so that we can read hardware performance counters on-line with relatively small overhead. We instrumented the applications with PAPI and, as the server architecture does not support total memory LD/ST counter, we counted the number of L2 cache misses, which indicates (approximately) the activity of memory.

We chose a comprehensive set of HPC benchmark workloads. Each workload stresses *one or more* of the following subsystems - CPU, memory, disk (storage), and network interface. They can be classified as:

- **CPU intensive,** e.g., `HPL Linpack`, which solves a (random) dense linear system in double precision arithmetic, and `FFTW`, which computes the discrete Fourier transform.
- **Memory intensive,** e.g., `sysbench`, which is a multi-threaded benchmark developed originally to evaluate systems running a database under intensive load.
- **I/O intensive,** e.g., `b_eff_io`, which is a MPI-I/O application, and `bonnie++`, which focuses on hard-drive and file-system performance.

An application usually demands the services of a given subsystem in discrete time windows. However, if the average demand for a subsystem X is significant, we consider the application to be X-intensive. Figure 1(left) shows different subsystem utilizations of a CPU-intensive workload. Note that an application can also be deemed to be intensive along multiple dimensions if the demand for resources from multiple subsystems are significant. Figure 1(right) shows different subsystem utilizations of a network- and CPU- intensive workload. The utilization of a particular subsystem by two different

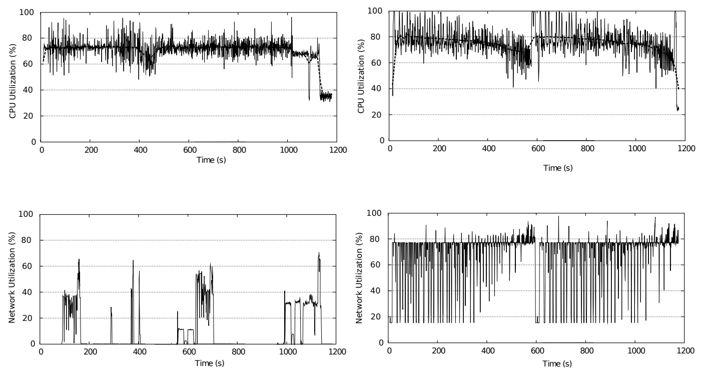


Fig. 1: Sub-system utilization over time for a CPU-intensive workload (left) and a CPU- cum network-intensive workload (right)

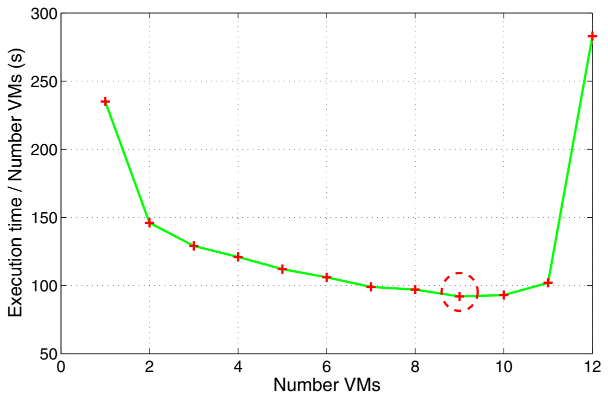


Fig. 2: Execution times of the FFTW benchmark

VMs running applications may either overlap (resulting in contention) or not overlap (be contention free).

B. Benchmarking

The benchmarking was conducted using Dell servers, each with a Intel quad-core Xeon X3220 processors, 4GB of memory, two hard disks, and two 1Gb Ethernet interfaces. This is intended to represent a general-purpose rack server configuration, widely used in virtualized datacenters. The servers run CentOS operating system based on a patched Linux kernel (2.6.18) running Xen hypervisor version 3.1. To empirically measure the instantaneous power consumption of the servers we used a Watts Up? .NET power meter. This power meter has an accuracy of 1.5% of the measured power with sampling rate of 1Hz. The meter was mounted between the wall power outlet and the server. We estimate the consumed energy by integrating the actual power measures over time. In addition to using a single server type, we made some additional assumptions, such as a single process per VM, to reduce the complexity. To run multiple processes (e.g., MPI applications) multiple VMs are required.

In order to acquire sufficient data to create a VM allocation model, firstly, we conducted a set of base tests that consolidate different VM instances running applications of the same type in a single server. This allowed us to find out, on the one hand, the optimal scenarios to either maximize performance or minimize energy consumption and, on the other hand, the maximum number of VMs that can be consolidated in a single server without adversely impacting energy consumption and the applications' performance. We ran the base experiments with different number of VMs (up to 16) running the same application type for each of the application's profiles.

For example, Figure 2 shows the average execution time of the FFTW benchmark (single thread, with long initialization phase) when the number of VMs in a single physical server is increased. In this case, the shortest average execution time (the optimal scenario) is obtained with 9 VMs running on a single server. With more than 11 VMs the average execution time increases significantly. This means that when more than 11 VMs running this FFTW benchmark are co-located in a physical server, the average execution time of a VM is comparable to the average execution time of a VM when a set of benchmarks are executed sequentially one after the other.

From the base tests, we obtained a set of optimal scenarios (`OS`), i.e., optimal number of VMs for the shortest average execution times (`OSP`) and for minimum energy consumption (`OSE`). Table I summarizes the parameters that will be most useful for the experiments combining different workload types. For example, `OSPC` denotes the optimal number of CPU-intensive VMs that can be run simultaneously on a single physical server so that the average execution time of VMs is the shortest. `TC`, `TM` and `TI` are the execution times for a single CPU-, memory, and I/O-intensive workload, respectively.

TABLE I: Summary of parameters obtained in base tests

	Test		
	CPU	Memory	I/O
#VMs that optimize performance	OSPC	OSPM	OSPI
#VMs that optimize energy	OSEC	OSEM	OSEI
Run time of single test on 1 VM	TC	TM	TI

In order to reduce the number of combinations, we define the following parameters based on the number of VMs used in the optimal scenarios: `OSC=max(OSPC, OSEC)`, `OSM=max(OSPM, OSEM)` and `OSI=max(OSPI, OSEI)`.

The second part of the benchmarking consists of running all the possible combinations of workload types with different number of VMs. Considering the limitations introduced previously, the following number of experiments were required: `(OSC+1)·(OSM+1)·(OSI+1)-(1+OSC+OSM+OSI)`. The combinations excluded are those that do not require any VM of each workload type and the base tests. The experiments took several days to be completed and they were conducted using a platform that we developed to automatically run the benchmarks and process the data.

C. Database

In order to make our model available for proactive VM consolidation, the information collected from the benchmarking (base and combined tests) was stored in a database. As the amount of information was manageable using text files, we used a plain-text file with comma-separated values (CSV) instead of an actual database management system. Table II summarizes the information contained in the database.

TABLE II: Summary of the information stored in the database

Field	Description
`Ncpu`	#VMs running a CPU-intensive benchmark
`Nmem`	#VMs running a Memory-intensive benchmark
`Nio`	#VMs running an I/O-intensive benchmark
`Time`	Total execution time of the outcome (seconds)
`avgTimeVM` (for each VM)	Average execution time for each VM (`avgTimeVM` = `Time` / (`Ncpu`+`Nmem`+`Nio`))
`Energy`	Energy consumed to run the outcome (Joules)
`MaxPower`	Maximum power dissipation measured (Watts)
`EDP`	Energy Delay Product (Joules$\times$seconds)

In addition to the information listed in Table II, we store other relevant information from the base experiments such as the number of VMs of optimal scenarios (e.g., `OSC, OSM, OSI`) and reference execution times (e.g., `TC, TM, TI`), in an auxiliary file. We do not include any information from the system because in this work we are focused on a single platform. However, if multiple server configurations are used, we should include system characteristics such as number of CPUs, amount of memory, reference performance index, etc. As the registers of the database are accessed using binary search, the searching cost is $\mathcal{O}(log(num\_tests))$. Therefore, we sorted (in the ascending order) the registers of the database by a searching key, which is composed of the parameters that indicate the number of VMs of each workload type (`Ncpu`, `Nmem`, `Nio`).

D. VM allocation algorithm

Our VM allocation algorithm takes advantage of the model (in the form of a database) that is described above. It has two main objectives, (i) minimize energy consumption and (ii) maximize the performance (i.e., minimize workload execution time). As these are two conflicting objectives, we use a parameter α to adjust the possible trade-off between energy efficiency and performance; α is defined as follows: $\alpha \in \mathcal{R} \cap \alpha \in (0, \ldots, 1)$ and emphasizes the energy efficiency goal while 1-α emphasizes performance. For example, if α=0.7 the algorithm will try to minimize the energy consumption first (70% of preference) and then the performance but with less intensity (30% of preference).

The allocation algorithm focuses on the objectives discussed above and does not consider specific policies such as those based on priorities. The input parameters of the algorithm are: (i) the database with the allocation model, (ii) values from the base experiments such as `OSC/OSM/OSI` (can be extracted from the auxiliary file), (iii) a set of VMs and the application's profile and maximum execution time (QoS guarantees) for each of them, and (iv) the optimization goal (α). The algorithm returns the allocation of VMs that best matches the input optimization goal while satisfying the QoS constraints. The algorithm can be relaxed by disregarding the QoS guarantees but it might be not acceptable for production system.

To find the best partitions of the input set of VMs for allocation in individual servers, we used a brute-force search algorithm over the servers with their current VM allocations. Specifically, it computes the estimated execution time and energy consumption for each partition of the initial set using the allocation algorithm described above. As the number of partitions of a set might be large, we used the search algorithm discussed in [21], which is efficient in terms of complexity. If two partitions have the same rank in different servers, we select the first server of the list. Figure 3 shows the main components and control flow of our VM allocation algorithm.

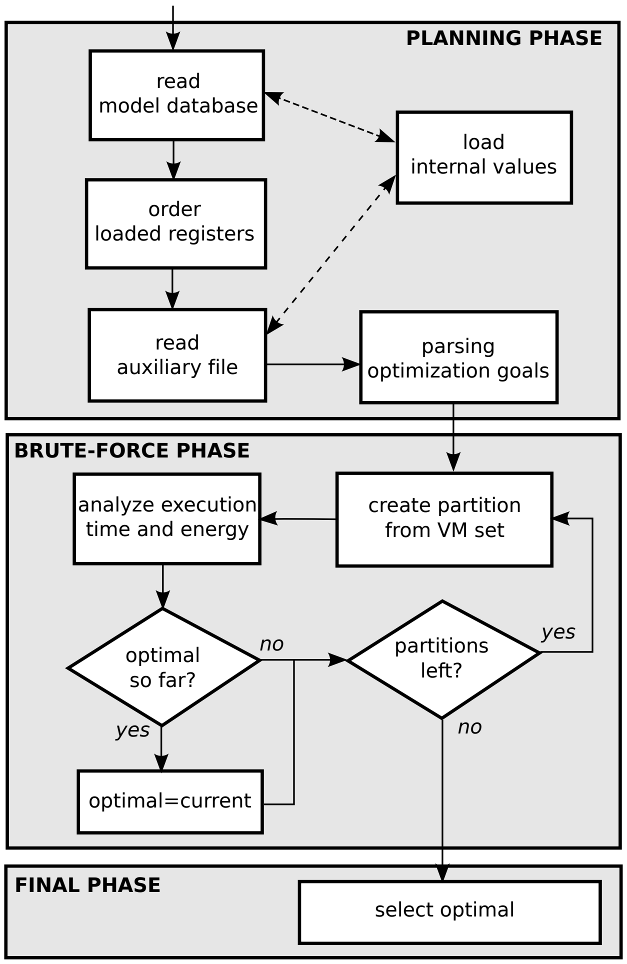
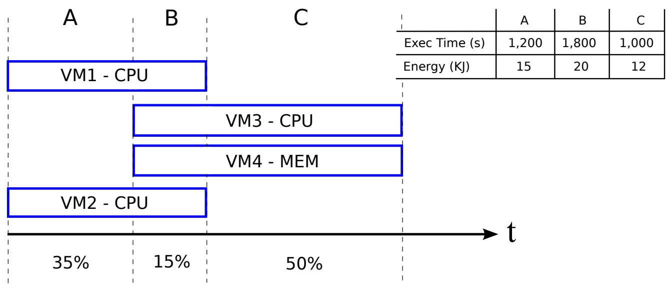


Fig. 3: VM allocation algorithm

IV. Evaluation

In this section, we evaluate the possible energy savings and performance tradeoffs that can be achieved at the datacenter level using our proactive VM allocation. We conducted simulations with traces of parallel workloads along with the algorithm described in the previous section.



	A	B	C
Exec Time (s)	1,200	1,800	1,000
Energy (KJ)	15	20	12

Fig. 4: Possible VM allocation outcome over time

A. Methodology

We used workload traces from real HPC production systems to evaluate the performance and energy efficiency of the proposed approach. As not all of the required information cannot be obtained from these traces, some data manipulation was needed as explained in the following subsection. The simulations are based on the empirical data obtained from real experiments to create the model described in the previous section. Therefore, in our simulations we used a system model composed of several servers with the same characteristics of our real testbed. To compute the estimated execution times and energy consumption we used the information of our allocation model. Given a specific partition with a subset of VMs running their associated applications types, we lookup in our model database and use the matching values proportionally. As VM allocations may vary over time, we compute the estimated execution time and energy consumption with the weighted average of the values associated to each interval of time. We also assume a fixed power dissipation of 125 W when a server is idle.

Figure 4 illustrates a possible VM allocation outcome over time for a server. The application type associated with each VM is also shown. Different time intervals (A, B, C) have different VM allocations and, therefore, the estimated execution time of the applications and energy consumption for each interval will be different. For example, the execution time of VM1 will be computed considering the relative weight of each allocation (70% of allocation A and 30% of allocation B) as follows: $ExecTime_{VM1} = 0.7 \cdot 1200s + 0.3 \cdot 1800s = 1380s$ and the energy consumption for the whole outcome will be: $Energy = 0.35 \cdot 15KJ + 0.15 \cdot 20KJ + 0.5 \cdot 12KJ = 14.25KJ$. As we focus on studying the impact of VM allocation on the performance/energy efficiency we do not consider the overhead for scheduling and resource provisioning.

B. Workloads

As mentioned earlier, we used production workload traces from the Grid Observatory [22], which collects, publishes, and analyzes logs on the behavior of the EGEE Grid [23]. As the traces are in different formats and include data that are not useful for our purpose, they were pre-processed before being input to the simulations. First, we converted the input traces to the Standard Workload Format (SWF) [24]. As they are usually composed of multiple files we combined them into a single file. Then, we cleaned the trace, now in SWF format, in order to eliminate failed jobs, cancelled jobs and anomalies. As the traces found from different systems did not provide all the information needed for our analysis, we needed to complete them using a model based on the benchmarking of HPC applications (see Sect. III). We randomly assigned one of the possible benchmark profiles to each request in the input trace, following a uniform distribution by bursts. The bursts of job requests were sized (randomly) from 1 to 5 job requests. These traces are intended to illustrate the submission of scientific HPC workflows, which are composed of sets of jobs with the same resource requirements.

As the EGEE Grid is a large-scale heterogeneous distributed system composed of a large number of nodes, we scaled and adapted the job requests to the characteristics of our system model and evaluation methodology. Specifically, we assigned 1 to 4 VMs per job request rather than the original CPU demand and we defined the QoS requirements (maximum in response time) per application type and not for each specific request.

C. Metrics

We evaluate the impact of our approach in terms of the following metrics: makespan (workload execution time in seconds, which is the difference between the earliest time of submission of any of the workload tasks, and the latest time of completion of any of its tasks), energy consumption (in Joules), and percentage of SLA violations. The number of SLA violations were calculated by summing the number of missed deadlines of all applications. The deadline here refers to the maximum response time as specified by the QoS requirements.

D. Strategies

We have conducted our simulations using different allocation strategies that have different goals. Specifically, we have evaluated the following allocation strategies:

- FIRST-FIT (FF), in which job requests are allocated following the first-fit policy based on CPU slots. It means that an incoming job request is allocated to the first available server until the number of allocated VMs is equal to the number of CPUs (VM multiplexing on CPUs is not allowed). FIRST-FIT-2 (FF-2) and FIRST-FIT-3 (FF-3) are two variants of FIRST-FIT that allow multiplexing up to 2 and 3 VMs on each CPU, respectively.
- PROACTIVE, in which job requests are allocated to servers following the algorithm described in Sect. III. We consider the following variations:
  - α=1 (PA-1): the goal is minimizing the energy consumed;
  - α=0 (PA-0): the goal is minimizing the execution time;
  - α=0.5 (PA-0.5): the goal is finding the best tradeoff between execution time and energy consumption.

E. Results

Figures 5, 6 and 7 show the results obtained using the VM allocation strategies and workloads traces described previously. Furthermore, in order to control the pressure of the

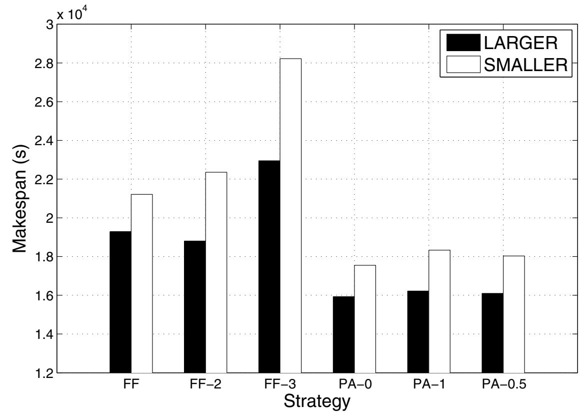


Fig. 5: Makespan (s)

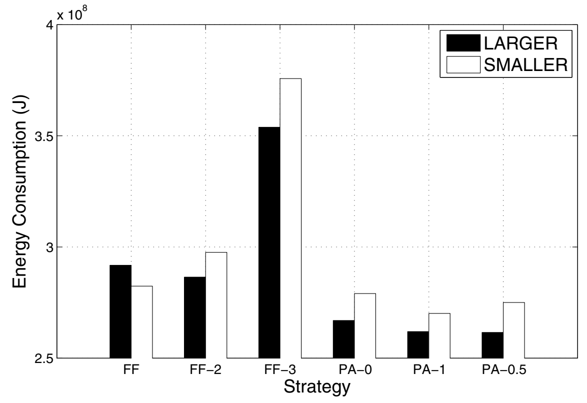


Fig. 6: Energy consumption (J)

system load, we modeled two different Clouds of different sizes rather than using different input traces with different arrival rates. The SMALLER Cloud system is the reference one and the LARGER Cloud system is over-dimensioned (15% approximately), which means that the former one is expected to be more loaded than the latter. The input trace used in the simulations requests a total of 10,000 VMs.

As we can observe in Figure 5, the PROACTIVE strategy can provide up to 18% shorter execution times. This is explained due to the fact that the application awareness results in fewer contentions for resources as only the most compatible VMs are consolidated. With the FIRST-FIT strategy the execution times are longer due to resource contention, especially when multiplexing 3 VMs on the same CPU. Furthermore, Figure 5 shows that the PROACTIVE strategy with the performance optimization goal reduces the execution times by more than 3% in comparison to the same strategy with the energy optimization goal. We can also appreciate that the execution times in the SMALLER system are higher than the execution times in the LARGER system due to higher load pressure. This is specially relevant with the FIRST-FIT strategy (multiplexing 3 VMs) due to possible additional resource contention.

Figure 6 shows that the PROACTIVE strategy saves around 12% of energy consumption on average with respect to first-fit

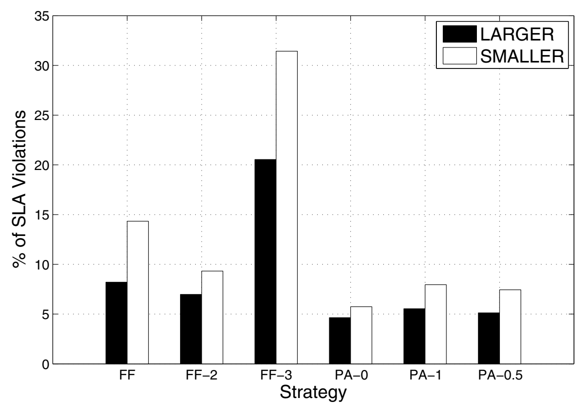


Fig. 7: Percentage of SLA violations

(with and without VM multiplexing). In fact, makespan and energy consumption follow a similar pattern in the LARGER system. Furthermore, Figure 6 shows that the PROACTIVE strategy with the energy optimization goal saves almost 3% more energy than the same strategy with the performance optimization goal, and with the goal of finding the best tradeoff it provides intermediate results (but the variations are not very significant, i.e., $<2\%$). Although the makespan in the SMALLER system is higher than the makespan in the LARGER system, the energy consumption in the SMALLER system is lower than the energy consumption in the LARGER system as in the SMALLER system there are fewer servers consuming energy and there are more opportunities for consolidation. However, with the FIRST-FIT strategy the resource contention penalizes the energy efficiency significantly when multiplexing of 2 or 3 VMs is allowed on the same CPU.

Figure 7 shows that the percentage of SLA violations with the PROACTIVE strategies are also less compared to the traditional schemes. It means that the PROACTIVE strategy can maintain or even provide better QoS guarantees. Furthermore, we can observe in Figure 7 a correlation between execution time and SLA violations, the higher the makespan higher the percentage of SLA violations. We also can appreciate that the strategies evaluated present similar behaviors under higher load conditions. We do not show in this paper the results obtained with other possible configurations of the PROACTIVE strategy (e.g., α=0.75) since the variation in the results was not significant enough.

V. Conclusions and Future Work

In this paper, we presented and evaluated a novel application-centric energy-aware strategy for VM allocation that aims at maximizing the resource utilization and energy efficiency through VM consolidation while satisfying QoS guarantees. To do this, we developed an empirical model for the average energy consumption and execution time based on measurements from extensive execution of standard HPC workload benchmarks (all possible allocations based on number and type of VMs), and designed an algorithm to determine the best VM allocation that achieves an optimization goal such as minimization of energy consumption and/or execution time. The results obtained from simulations using real production HPC workload traces show that proactive VM allocation can significantly contribute to energy efficiency and/or optimization of the application performance depending on optimization goals. Specifically, the experimental results showed that our proactive VM allocation algorithm can save up to 12% in energy consumption and/or up to 18% in execution time compared to the traditional first-fit approach. However, the impact of the algorithm's optimization goals on the performance and energy consumption is moderate. We conclude that profiling and modeling HPC applications is an effective strategy to efficiently manage cloud data centers.

We chose to use a brute-force search algorithm for selecting the best possible VM allocation and a traditional first-fit approach for comparison in order to demonstrate and study the potential of application-centric proactive VM allocation. Our current research efforts are geared towards i) using machine learning techniques to extract on-the-fly a model out of the sub-system utilization data collected from offline experiments using benchmarks as well as from real applications running on VMs and ii) compare our proposed solution against some of the state of the art discussed in Sect. II by implementing them. Our planned future research efforts include, i) extending the solution to be aware of and support heterogeneous server hardware, which is required for evaluation on a real testbed and ii) integrating the proposed solution with schemes for autonomic thermal management in instrumented datacenters.

Acknowledgments

This research was conducted within the National Science Foundation (NSF) Center for Autonomic Computing and is supported in part by The Extreme Scale Systems Center at ORNL, by the Department of Defense, and by an IBM Faculty Award. This material was based on work supported by the NSF, while co-author M. Parashar was working at the Foundation. Any opinion, finding, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the NSF. The authors would like to thank the Grid Observatory, which is part of the EGEE-III EU project INFSO-RI-222667

References

[1] "Report to congress on server and data center energy efficiency," U.S. Environmental Protection Agency, Tech. Rep., August 2007.

[2] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic Placement of Virtual Machines for Managing SLA Violations," in *Proc. of IFIP/IEEE Symp. on Integrated Network Management*, Munich, Germany, May 2007, pp. 119–128.

[3] I. Rodero, E. K. Lee, D. Pompili, M. Parashar, M. Gamell, and R. J. Figueiredo, "Exploiting VM Technologies for Reactive Thermal Management in Instrumented Datacenters," in *Workshop on Energy Efficient Grids, Clouds, and Clusters in conjunction with IEEE Grid*, Brussels, Belguim, Oct. 2010, pp. 321–328.

[4] A. Turner, A. Sangpetch, and H. S. Kim, "How to Tame Your VMs: An Automated Control System for Virtualized Services," in *Proc. of Large Installation System Administration Conf.*, San Jose, CA, Nov. 2010, pp. 179–188.

[5] Y. Ajiro and A. Tanaka, "Improving Packing Algorithms for Server Consolidation," in *Proc. of Computer Measurement Group Conf.*, San Diego, CA, Dec. 2007, pp. 399–406.

[6] J. Stoess, C. Lang, and F. Bellosa, "Energy Management for Hypervisor-based Virtual Machines," in *Proc. of USENIX Annual Technical Conf.*, Santa Clara, CA, Jun. 2007, pp. 1–14.

[7] A. Verma, P. Ahuja, and A. Neogi, "Power-aware Dynamic Placement of HPC Applications," in *Proc. of Intl. Conf. on Supercomputing*, Island of Kos, Greece, Jun. 2008, pp. 175–184.

[8] F. Hermenier, X. Lorca, J.-M. Menaud, G. Muller, and J. Lawall, "Entropy: A Consolidation Manager for Clusters," in *Proc. of the ACM SIGPLAN/SIGOPS Conf. on Virtual Execution Environments*, Washington, DC, Mar. 2009, pp. 41–50.

[9] D. A. Menasce and M. N. Bennani, "Autonomic Virtualized Environments," in *Proc. of Intl. Conf. on Autonomic and Autonomous Systems*, Silicon Valley, CA, Jul. 2006, p. 28.

[10] R. Nathuji, C. Isci, and E. Gorbatov, "Exploiting Platform Heterogeneity for Power Efficient Data Centers," in *Proc. of Intl. Conf. on Autonomic Computing*, Jacksonville, FL, Jun. 2007, p. 5.

[11] T. Wood, G. Tarasuk-Levin, P. Shenoy, P. Desnoyers, E. Cecchet, and M. D. Corner, "Memory Buddies: Exploiting Page Sharing for Smart Colocation in Virtualized Data Centers," in *Proc. of the ACM SIGPLAN/SIGOPS Conf. on Virtual Execution Environments*, Washington, DC, Mar. 2009, pp. 31–40.

[12] D. Gupta, S. Lee, M. Vrable, S. Savage, A. C. Snoeren, G. Varghese, G. M. Voelker, and A. Vahdat, "Difference Engine: Harnessing Memory Redundancy in Virtual Machines," *Communications of the ACM*, vol. 53, pp. 85–93, Oct. 2010.

[13] S. Govindan, A. R. Nath, A. Das, B. Urgaonkar, and A. Sivasubramaniam, "Xen and co.: Communication-aware CPU Scheduling for Consolidated Xen-based Hosting Platforms," in *Proc. of the Intl. Conf. on Virtual Execution Environments*, San Diego, CA, Jun. 2007, pp. 126–136.

[14] P. Apparao, R. Iyer, X. Zhang, D. Newell, and T. Adelmeyer, "Characterization & Analysis of a Server Consolidation Benchmark," in *Proc. of ACM SIGPLAN/SIGOPS Conf. on Virtual Execution Environments*, Seattle, WA, Mar. 2008, pp. 21–30.

[15] A. Verma, P. Ahuja, and A. Neogi, "pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems," in *Proc. of ACM/IFIP/USENIX Intl. Conf. on Middleware*, Leuven, Belgium, Dec. 2008, pp. 243–264.

[16] A. Kochut and K. Beaty, "On Strategies for Dynamic Resource Management in Virtualized Server Environments," in *Proc. of the Intl. Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, Istanbul, Turkey, Oct. 2007, pp. 193–200.

[17] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation," in *Proc. of Intl. Conf. on Cloud Computing*, Beijing, China, Dec. 2009, pp. 254–265.

[18] Y. Song, Y. Sun, H. Wang, and X. Song, "An Adaptive Resource Flowing Scheme amongst VMs in a VM-Based Utility Computing," in *Proc. of IEEE Intl. Conf. on Computer and Information Technology*, Fukushima, Japan, Oct. 2007, pp. 1053–1058.

[19] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient Resource Provisioning in Compute Clouds via VM Multiplexing," in *Proc. of Intl. Conf. on Autonomic Computing*, Washington, DC, Jun. 2010, pp. 11–20.

[20] I. Rodero, S. Chandra, M. Parashar, R. Muralidhar, H. Seshadri, and S. Poole, "Investigating the Potential of Application-Centric Aggressive Power Management for HPC Workloads," in *Proc. of the IEEE Intl. Conf. on High Performance Computing (HiPC)*, Goa, India, Dec. 2010, pp. 1–10.

[21] M. Orlov, "Efficient Generation of Set Partitions," Engineering and Computer Sciences, University of Ulm, Tech. Rep., 2002.

[22] "Grid Observatory," 2010. [Online]. Available: http://www.grid-observatory.org/

[23] "Enabling Grid for E-sciencE," 2010. [Online]. Available: http://www.eu-egee.org/

[24] D. Feitelson, "Parallel Workload Archive," 2010. [Online]. Available: http://www.cs.huji.ac.il/labs/parallel/workload/