Vasudevan, Meera, Tian, Glen, Tang, Maolin, & Kozan, Erhan
(2017)
Profile-based application assignment for greener and more energy-efficient data centers.
Future Generation Computer Systems, *67*, pp. 94-108.

https://doi.org/10.1016/j.future.2016.06.037

Future Generation Computer Systems, in press, 2016.
DOI: 10.1016/j.future.2016.06.037

Profile-based Application Assignment for Greener and More Energy-Efficient Data Centers

Meera Vasudevan[a], Yu-Chu Tian[a,b,*], Maolin Tang[a], Erhan Kozan[c]

[a] *School of Electrical Engineering and Computer Science, Queensland University of Technology, GPO Box 2434, Brisbane QLD 4001.*
[b] *College of Information Engineering, Taiyuan University of Technology, Taiyuan, Shanxi 030024, China*
[c] *School of Mathematical Sciences, Queensland University of Technology, GPO Box 2434, Brisbane QLD 4001.*

Abstract

The cloud computing era has brought significant challenges in energy and operational costs of data centers. As a result, green initiatives with regard to energy-efficient management of data center infrastructure for cloud computing have become essential. Addressing a big class of widely deployed data centers with relatively consistent workload and applications, this paper presents a new profile-based application assignment approach for greener and more energy-efficient data centers. It builds realistic profiles from the raw data measured from data centers and then establishes a theoretical framework for profile-based application assignment. A penalty-based profile matching algorithm (PPMA) is further developed to obtain an assignment solution, which gives near-optimal allocations whilst satisfying energy-efficiency, resource utilization efficiency and application completion time constraints. Through experimental studies, the profiling approach is demonstrated to be feasible, scalable and energy-efficient when compared to the commonly used general and workload history based application management approaches.

Keywords: Data center, application assignment, profile, resource management, energy efficiency

*Corresponding Author
Email address: `y.tian@qut.edu.au` (Yu-Chu Tian)

1. Introduction

In today's economy, data centers and cloud computing are increasingly used everyday by the sky-rocketing number of Internet users. This is predictably escalating the energy and costs to power and maintain these systems at an alarming pace. Overall, data centers consume 1.1% to 1.5% of the world's total electricity consumption [1]. They are responsible for 14% of the Information and Communication Technology (ICT) carbon footprint according to the Smart2020 analysis [2]. More than 35% of the current data center operational expenses are accounted for by energy consumption. This figure is projected to double in a few years. According to a report by the Natural Resources Defence Council (NRDC), data centers consumed 91 billion kWh of electrical energy in 2013. This statistics is projected to increase by 53% by year 2020 [3].

With different purposes, various data centers contribute to the energy consumption and carbon footprint differently. Large-scale data centres are mainly used to host public clouds with dynamic workload. Typical hyper-scale large data centers are those from giant IT corporations like Microsoft, Google, Apple, Amazon, and Facebook. In comparison, medium- and small-scale data centers are typically run by business companies, universities and government agencies. They typically provide services via private clouds or clusters/grids with virtualized management. Therefore, they have relatively consistent workload. The NRDC reports that there is a distinct gap in energy-efficient initiatives when comparing well-managed hyper-scale large data centers and the numerous less-efficient small- to medium-scale data centers. The hyper-scale large data centers only share 5% of the global data center energy usage, while the remaining 95% is made up of small- to medium-scale data centers [3]. Therefore, energy management for small- to medium-scale data centers with relatively consistent workload is globally more significant than that for hyper-scale large data centers with very dynamic workload. This paper targets the widely deployed small- to medium-scale data centers.

The necessity for green and energy-efficient measures to reduce carbon footprint and the exorbitant energy costs has become very real and emerging. Energy and cost distribution studies, e.g., Le *et al.* [4], have confirmed that deploying green initiatives at data centers reduces the carbon footprint by 35% at only a 3% cost increase. However, energy-aware measures with simultaneous maximum performance efficiency and minimum energy consumption [5] are not easy to achieve. In most cases, deploying an energy-efficient solution inevitably degrades the performance efficiency of the data centers.

To tackle this challenging issue, our preliminary work [6] introduced the concept of profiling for application assignment to Virtual Machines (VMs). It formulated the application assignment as a linear optimization with utilization of fully synthetic application and VM profiles. It also developed a simple profile matching algorithm to solve the optimization problem. The aim of the preliminary work was to introduce the profiling concept as a feasible and scalable application assignment method.

Extending our preliminary work significantly for improved solutions, this paper aims to develop a new profile-based application assignment framework for greener and more energy-efficient data centers. The new framework uses realistic profiles and also fulfils energy, resource and performance constraints or requirements. In comparison with our preliminary work [6], distinct contributions of this paper include the following four aspects:

- Physical Machine (PM) profiles: In addition to application and VM profiles, PM profiles are integrated into the profile-based application assignment, enabling derivation of actual energy savings of the servers from the application assignment;
- Profile building: Different from synthetic profiles, realistic application, VM and PM profiles are built from raw data of a real-world data center through systematic methods, allowing more realistic application assignment based on profiles;
- Optimization framework: a penalty-based linear optimization framework is formulated for profile-based application assignment with consideration of memory constrains in addition to CPU resources; and
- Solution algorithm: Refined from a simple profile matching algorithm, a penalty-based profile matching algorithm (PPMA) that uses some heuristics is presented to solve the new penalty-based optimization problem with considerations of memory, CPU and performance constraints.

Moreover, new and comprehensive case studies are carried out in this paper to demonstrate the effectiveness of the Profiling approach. The experimental results are compared with those from the commonly used general approach and workload history based application management strategy.

The energy management of a virturalized data center can be implemented at three layers: application, VM and PM layer, as shown in Figure 1. The application management at the top layer assigns applications to VMs. The VM management layer is responsible for VM placement to PMs, VM sizing and VM migration. The PM management layer at the bottom layer is in charge of ON/OFF operations of PMs, sleep cycles, cooling and DVFS. While each of the three layers contributes to the overall data center energy savings, this paper limits its scope to the application management layer. Applications requested by cloud consumers or data center users are assigned to VMs, thereby allowing access to data center resources such as CPU and memory. The application assignment strategies typically consider application runtime, server workload, resource requirements or availability, energy consumption and performance efficiency. Thus, one of the key objectives of our research is to create such an energy-efficient application management strategy whilst maintaining the data center performance efficiency. Our investigation into the application assignment to VMs complements current research on the problem of VM placement to PMs.

Among various data centers, a big class of widely deployed data centers with nearly consistent workload and applications is investigated in this paper. These data centers are generally managed by universities, government agencies,

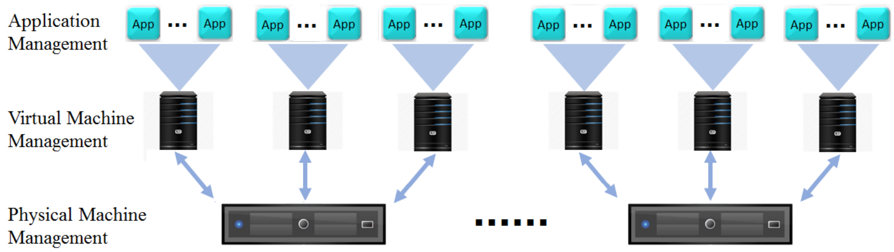

Figure 1: Energy management architecture for data centers.

and small corporate businesses. According to our investigations into a real-world data center, such data centers typically have a well-defined workload characterized by an almost constant number of VMs. The number of VMs hosted in PMs is typically reviewed every three to six months during which no adjustment is made. From the raw data collected from the real data center and using a workload model, this paper generates load synthetically for profile-based application assignment to VMs.

The paper is organized as follows. Section 2 reviews related work and motivates the research. Section 3 discusses the concept of profiles and the methodology of building profiles. A profile-based energy-efficient framework is presented in Section 4 with framework formulation and algorithmic solution. Experimental studies are conducted in Section 5. Finally, Section 6 concludes the paper.

2. Related Work and Motivations

Energy-efficiency of data centers has been a focus of many studies from various perspectives. In the work [7], energy consumption was minimized for fattree data center networks. The issue of colocation demand response was investigated by Ren and Islam [8]. Yoon *et al.* [9] presented techniques of efficient data mapping and buffering for multilevel cell phase-change memories. The work by Kumar *et al.* [10] studied cloud data management through workload-aware data placement and replica strategies. Energy-aware management of data centers for cloud computing was also investigated through heuristic resource allocation [11]. Leon and Navarro [12] built a quantitative model to describe the problem of minimizing energy consumption for resource allocation in data centers. All those studies used different methods to achieve energy savings, but none of them used the profiling concept on which our work in this paper is based.

Other investigations were also reported on energy-efficient data center management. The work in [13] focused on the topic of energy proportional data center networks. Liu and He [14] discussed the fairness of replica resource sharing in IaaS clouds. Combs *et al.* [15] investigated the energy management with high-performance computing workloads. In the work by Yeo *et al.* [16], an ambient temperature-aware capping approach was developed for power savings in data centers. Autonomous virtual resource management in data centers is discussed by Chen, Shen and Sapra [17] through a Markov decision process. A fractal framework is presented by Ghorbani *et al.* for effective management of burst cloud workloads. Our work presented in this paper is different from those recent reports in the sense that the concept of profiles is utilized for efficient energy management of application assignment to VMs in data centers.

The static profiling technique was discussed in [18, 19] to predict performance degradation in relation to multiple application assignment to a single machine. The method of Bubble-Up and Bubble-Flux [20] was developed to accurately predict the performance degradation incurred on allocating multiple workloads to servers for maximum utilization. Bubble-Up maintains a trade-off between machine utilization and Quality-of-Service (QoS) degradation by setting a degradation threshold for each application. However, it has limitations such as the requirement of workload knowledge, prediction inflexibility in terms of load changes, and the incapability of predicting more than two co-running applications. Those limitations are overcome with the Bubble-Flux management strategy. The Bubble-Flux accurately manages the QoS to provide maximum utilization of servers. Servers are monitored to observe shared resource fluctuations in real-time to predict the effect on the QoS of latency-sensitive applications.

Energy-Efficient Workload Aware (EEWA) task scheduler [21] has been developed to use online profiling to collect workload information of tasks for CPU-bound parallel applications. A workload-aware frequency adjuster tunes the core frequencies using this information. The tasks are allocated to the cores by the preference-based scheduler. EEWA task scheduler maintains a trade-off between energy consumption and performance for CPU-bound applications in multi-core architectures.

Nguyen *et al.* [22] have proposed an elastic distributed resource scaling framework called AGILE. AGILE is capable of handling dynamic workloads with minimum penalty incurred. It uses online profiling to model the violation rate and carry out wavelet-based resource demand prediction. It further employs this prediction to handle variations in workloads. In comparison with online profiling, offline profiling is used in an overdriver framework [23] to analyse the memory overload probability of VMs.

Behavioural and performance profiles have been used in some existing approaches for resource allocation. Vu Do *et al.* [24] have investigated the relationship between resource demands and application performance metrics. An application profiling technique is proposed using a Canonical Correlation Analysis (CCA) method. The CCA analyzes and builds the performance efficiency profile of the applications in terms of their resource usage. Then, the resulting performance profiles are used to build a performance prediction model. Although profiling has been previously discussed as a means of evaluation in terms of performance and behaviour analysis, the designing of profiles in the decision making process of initial and continued application assignment has not been discussed. The present paper will implement an energy-efficient application assignment strategy based on profiling.

Most recently, Ye *et al.* [25] have proposed an energy-efficient server consolidation framework. It reduces the number of active physical servers and VM migrations in data centers whilst maintaining workload performance. Profiles are used as a key concept in the framework. They consist of performance losses of workloads during colocation and migration of VMs.

Shi *et al.* [26] have presented an application placement framework (EAPAC). The objective is to assign a certain number of mixed data-intensive applications to physical nodes and to resolve resource conflicts which arise due to the increase in the application processing time. The framework overcomes this issue by ensuring that a mixture of applications with different resource requests are assigned to individual servers. The EAPAC consists of an application level load balancer and an application server manager. The load balancer assigns applications to server hosts while the server manager monitors the resource provisioning amongst servers. The EAPAC is claimed to be able to improve the task response time by 4 times as compared to Tang's method presented in [27] for dynamic application placement in data centers. However, the EAPAC is intended for deployment in non-virtualized environments.

After study of the numerous energy-efficient measures for application assignment, the following technological gaps are identified, which motivate the research of this paper:

- Matching application to VMs based on the number of cores/memory specified at submission time does not implement any energy-efficient measures. Implementation of such measures requires application processing before assignment. For data centers with relatively consistent workload and applications, the presented profile-based assignment strategy collects and reuses data such as resource demands to reduce the application processing time.

- Profiling has been previously considered for resource consumption pattern identification, behavioural and performance analysis. This paper presents a novel approach of using profiles in the decision making stage of mapping applications to VMs for data centers with consistent workloads.

These technical gaps motivate the research of this paper.

In our previously presented conference paper [6], an energy-efficient application management approach was introduced by using the concept of Profiles. The classic assignment problem was described by employing a commonly accepted linear programming model. It was then solved using the standard Hungarian algorithm and a new profile matching algorithm. While giving optimal solutions, the Hungarian algorithm was severely limited in terms of scalability. Thus, the profile matching algorithm showed its advantage in sub-optimal solution and good scalability for large-scale problems of modern data centers.

However, this preliminary work had limited scopes in profiles, problem formulation, and problem solving. 1) On profiles, PM profiles were not considered at all. The application and VM profiles used in the approach were not built from real data of data centers. They were fully synthetic with consideration of

CPU resources as the only resource parameter. 2) On problem formulation, the simple linear programming model did not capture all performance and resource requirements. 3) On problem solving, the profile matching algorithm only considered stnthetic application and VM profiles to make application assignment decisions. Nevertheless, the preliminary work derived theoretical VM energy savings and well demonstrated the feasibility and scalability of the Profiling concept in application assignment.

The work of the present paper extends our preliminary work significantly through the following five distinct features. 1) PM profiles are integrated into our profile-based application assignment problem; 2) Application, VM and PM profiles are built from the raw data logs of a real-world data center; and application profiles are built from a well-established workload model; 3) a penalty-based optimization is formulated for profile-based application assignment with consideration of energy, resource and performance constrains or requirements; 4) the new optimization problem is solved using a new penalty-based profile matching algorithm; and 5) new and more comprehensive experimental studies are undertaken to demonstrate the new profile-based application assignment approach. The first four have been claimed as new contributions in Section 1. The use of realistic profiles built from the raw data logs facilitates derivation of near-optimal application assignment solutions without unduly increasing the computational effort. The theory of utilising application, VM and PM profiles in terms of energy-efficient application management is novel and as yet has remained unexplored.

The following sections will describe the concept of profiles and the methodology to build realistic application VM and PM profiles from the raw data center logs. Then, a profile-based energy-efficient application assignment framework is presented. The assignment solution is derived in the form of a profile-based matching algorithm with constraints of resource utilization efficiency and application completion time.

3. Profiles and Profile Building

This section first expands the concept of profiling that was initially introduced in our preliminary work [6]. Then, it develops a methodology to build off-line profiles for applications, VMs and PMs. In previous work, a completely synthetic workload has been used to build all those three types of profiles. However, realistic VM and PM profiles are built in this paper directly from the workload trace of a real data center. The application profiles are still built synthetically by using a commonly used workload model.

3.1. The Concept of Profiles

Profiling has been previously considered for behavioural and performance analysis. However, to the best of our knowledge, applying profiles in the decision making stage of applications assignment has not been investigated, and thus is a novel concept. Considering a big class of widely deployed data centers with relatively consistent workload and applications, this paper describes the relevance and effectiveness of Profiling for a deterministic application assignment problem. This will produce off-line optimization solutions.

A nearly consistent workload trace from a real data center is collected over a period of 14 days to build the VM and PM profiles offline. It is further observed that the pre-set VM and PM parameters like CPU, VCPU, and memory are reviewed every 6-12 months and seldom changed in small to medium density data centers. Therefore, the VM and PM profiles are stable for application allocation. Applications are habitually processed over time with varying instructions per cycle and memory. This is incorporated by the profiles on regular update, thereby validating the application profiles for allocation. From our continuous monitoring of a real data center over 14 days, only a very small number of new applications have been observed, for which the profiles built offline have not captured. In other words, data centers managed by universities, government agencies and corporate businesses have relatively consistent applications with varying parameters.

In our study of the workload, applications are categorized as web requests, data analysis, media streaming, e-commerce, social network and others. Some applications are executed in a single task whereas others like data analysis with MapReduce may consist of multiple tasks. Each of the single-task applications has a single profile. For applications with multiple tasks, each of the tasks in an application is treated as a sub-application with a profile. All profiles of the sub-applications in the application share the same application ID.

Occasional new applications whose profiles have not been captured previously will be handled differently. When such an application arrives, it will be allocated randomly. Then, its profile is recorded and appended to existing application profiles. If an application is the same as a previous one but has a different dataset for all the parameters, it is considered as a new application in the profiling approach of this paper. Conversely, if only some of the parameters are different, then the profiles are updated immediately and the allocation proceeds. To maintain the performance efficiency of the application assignment, the profiles are updated regularly.

Profiles are a set of well-organized information about specific data center components and their impact on energy consumptions. In this paper, a profile is created initially for each of the applications, VMs and PMs of the data center. Application profiles include those data related to CPU, memory requirements, actual arrival and execution times of individual applications. VM profiles include the data related to CPU processing and memory availability of each node corresponding to interval hours. PM profiles represent the workload and energy consumption of the data center. Other performance metrics can also be easily integrated into the profiles. After these profiles are created, they are used to create an energy cost matrix to identify the best possible application to VM assignment.

Typically, an extensive amount of data is readily available from the raw data logs of a data center to build these profiles off-line. Once the profiles are built, regular updates take considerably less processing time. As a result the overhead of creating profiles is insubstantial. Application and VM profiles enhance the functions of the allocation manager through 1) retrieval of resource requirements and availability information, and 2) prediction of application arrival and VM workload. This enhancement helps make prompt decisions of application assignment. Applications with profiles are mapped to VMs incurring the least possible energy cost whilst maintaining a trade-off with CPU utilization efficiency, memory and application completion time requirements.

The initial step of building profiles involves the accumulation of a large amount of specific data such as energy, CPU, memory, execution times and frequency, standard deviation and interval time. The following subsections will discuss the process of building profiles for PMs, VMs and Applications.

3.2. Building PM Profiles

PM profiles are directly derived from the raw data collected from a data center. In industrial practice, every data center keeps logs of their usage and performance measures for various purposes. This paper has used the raw data of servers over a period of 24 hours for 14 days (the 5th to 19th of May, 2014) from a real data center. The name of the data center is omitted here due to the commercial confidentiality.

Some of the raw data that have been collected include:

1. CPU utilization (%) every 60 minutes from multiple measurements during this time duration;
2. Memory used (%) every 60 minutes from multiple measurements during this time duration; and
3. Energy consumption every 5 minutes.

We conducted an analysis of server behaviour for the PMs in the data center. For a randomly chosen physical server (server ID: PH015), Figure 2 displays the behaviour pattern with respect to CPU utilization over a 24-hour period for four days. The standard deviation of CPU utilization over 24 hours for PH015 is determined as low as 2.36. Similar analysis is carried out for all servers with respect to minimum, maximum and average CPU utilizations per hour interval. The results demonstrate low variance, therefore justifying the assumption of a near consistent workload to build and utilise the PM and VM profiles for a reasonably realistic allocation strategy. Appendix A shows an example of CPU utilization attributes in physical server profiles over a period of 24 hours.

3.3. Building VM Profiles

VM Profiles basically encapsulate the workload history of each of the VMs. In this paper, the CPU and memory statistics of virtualized physical servers are collected from a real data center over a period of 14 days. For test purposes, it is assumed that each PM is capable of hosting up to 10 VMs. The VMs have varying sizes in terms of the CPU and memory allocated to them. The number of VMs per server and their sizes are pre-set during configuration.

A VM profile consists of the following parameters:

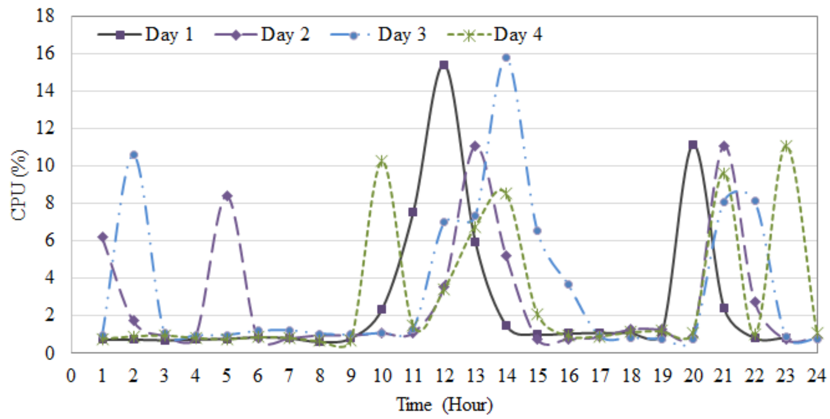

Figure 2: The behaviour pattern of physical server PH015.

1. VM ID;
2. Physical Host;
3. Total CPU capacity;
4. Interval;
5. Used CPU (%); and
6. Used Memory (%).

These six parameters of the VM profiles are explained as follows. Each VM has a unique identifier and the ID of the server hosting the VM is given by the PM host. The host determines the total resource capacity available to the VM. These parameters can be modified during configuration. The interval represents the time period under consideration. Each VM has a CPU and memory utilization associated with the corresponding time interval. The values of these parameters are derived directly from the real data center logs and the PM profiles. Figure 3 presents the profile data structure of 5 random VMs during the interval of 10.00 to 11.00. The pointer directs to a linked list consisting of all the applications allocated to the VM under consideration. A 24-hour profile of a VM (VM ID: 23) residing in server PH031 is displayed in Appendix A.

V_{id}	PH #	Total CPU (MIPS)	Total Mem (Bytes)	CPU (%)	Mem (%)	pointer
23	31	1000	1000	26.07	19.95	
40	34	1500	3000	8.96	3.00	
63	47	1000	2000	11.64	5.85	
72	65	2000	5000	24.04	21.89	
97	88	1500	4000	6.29	2.12	

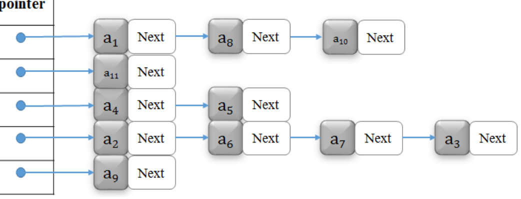

Figure 3: Profile data structure of randomly chosen five VMs in interval 10.00-11.00.

3.4. Building Application Profiles

A data center hosts hundreds of thousands of applications. Each application consists of a configuration file, which specifies the CPU, memory and disk space requirements for task execution. In this research, the generated application profiles consider CPU, memory, actual arrival time and run-time parameters. While the PM and VM profiles are generated directly from the data logs of the data center, the data logs from the data center do not include all information for building application profiles directly. Therefore, a commonly used synthetic workload model designed by Lublin and Feitelson [28] is adopted in this paper to build application profiles through some distributions. This is particularly for creating application parameters such as arrival time, run-time and resource requirements. For example, the workload generation model uses gamma distribution to generate wait times.

The application profile parameters are generated as follows. Initially, the number of applications is calculated for every hour using a cumulative distribution function. The arrival time, which is a random variable, is modelled with gamma distribution for each application (Algorithm 1) and is an input variable for our simulation experiments. The approximate CPU percentage and memory required to run the application is calculated using a two-stage uniform distribution. The application run-time is calculated using a hyper-gamma distribution [28] (Algorithm 2).

Algorithm 1: Application Arrival Time

Calculate number of applications per hour using Cumulative Distribution Function;
for *Each Application* **do**
    Generate random variable from gamma distribution;
Set arrival time to generated random variable;

Algorithm 2: Application Run-Time

Define parameters for gamma distributions 1 and 2, respectively;
Define relation probability between the two gamma distributions;
Generate a uniformly distributed random number between the range of 0.0 to 1.0;
if *(Generated a random number $\leq$ Relation probability)* **then**
    Gamma distribution 1 is active;
else
    Gamma distribution 2 is active;
Generate a random variable from the active gamma distribution;
Set run-time to the generated random variable;

It is worth mentioning that the run-time of workloads can be measured and it has been actually measured in our paper after an application is completed on a VM. But allocating an application to different VMs leads to different completion times $[T_{11}, T_{12}, \cdots, T_{1M}]$. However, before the run-time can be actually measured, the application must be allocated to one of the VMs in terms of some criteria determined by a number of parameters including an estimated run-time to maximize the optimization function. Therefore, an initial run-time of workloads is derived using a distribution and is included in the application profiles initially. In general, the VM where the completion time is closest to the initially generated run-time θ_1 is preferred.

Nearly 50,000 application profiles are generated using the workload model in C programming language. Figure 4 shows the data structure of randomly chosen five application profiles with the following five parameters:

1. Application ID;
2. Arrival Time (s);
3. Run-Time (s);
4. Requested CPU (%); and
5. Requested Memory (Bytes)

Arrival Time (S)	a_{ID}	pointer
16	9199	
61	1310	
76	24183	
296	45276	
17197	45299	

Run-time (S)	CPU (%)	Mem (Bytes)
193	74	122281
211	15	57688
96	55	97573
88	9	7200
1108	24	36516

Figure 4: Profile data structure of randomly chosen five applications.

The parameters of the application profiles are explained below. The application ID is a unique identifier associated with each application. In our studies for a real data center, the identifiers range from 0 to 49,999. The arrival time represents the time instant in seconds at which the application arrives at the data center. During application allocation, this time instant is compared with the interval time to select the VM hosts. The run-time represents the time duration (in seconds) in which the application is active. The requested CPU and memory represent the resource requirements to successfully execute the application.

After the application profiles are generated, a parser code is written in C++ programming language to process and incorporate the Profile data into our heuristic algorithm for application assignment. This will be discussed later in Section 4.

4. Profile-based Application Assignment Framework

With various profiles built in the last section, this section presents a profile-based and energy-efficient framework for application assignment in data centers.

Preliminary studies on profile-based application assignment model and algorithm have been recently presented at a conference [6]. They are substantially extended with the use of more realistic profiles, addition of memory constraints and a penalty-based assignment optimization model. In addition to energy saving, other objectives of the framework include effective performance levels in terms of execution time and CPU utilization efficiency. In essence, the framework aims to minimise the CPU energy of the physical node, which hosts the VMs for the timely and successful execution of applications.

For model development, some notations are defined below. Let us denote:

- $I \triangleq \{1, ..., N\}$ is a set of Applications
- $J \triangleq \{1, ..., M\}$ is a set of VMs; and
- $K \triangleq \{1, ..., L\}$ is a set of PMs.

A binary decision variable $x_{ij}, i \in I, j \in J$ represents the assignment of an application a_i, $i \in I$, onto a VM V_j, $j \in J$:

$$x_{ij} = \begin{cases} 1 & \text{if } a_i \text{ is allocated to } V_j;\ i \in I, j \in J, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Furthermore, for an application $a_i, i \in I$, CPU and memory requirements are denoted by μ_i and ω_i, respectively. If the application $a_i, i \in I$, is hosted by the VM $V_j, j \in J$, the actual memory allocated from the V_j to the a_i is represented by ω_{ij}. The CPU capacity of $V_j, j \in J$, is denoted by μ_{vj}.

A profile-based linear programming model has been designed to identify and carry out near-optimal placement of applications on VMs. The objectives include resource utilization efficiency, application completion time within its deadline and minimised energy cost. This will be discussed in more detail below.

4.1. CPU Utilization Efficiency

The CPU utilization efficiency of a VM $V_j, j \in J$, is a ratio of the total CPU percentage in use by the applications to the total CPU capacity of the VM. It is represented by $\eta_{cpu}(j) \in [0, 1]$ and derived at a time instance before application assignment as follows:

$$\eta_{cpu(j)} = \frac{\sum_{i=0}^{N} \mu_i x_{ij}}{\mu_{vj}};\ i \in I, j \in J \tag{2}$$

where μ_i represents the CPU requirement of the application $a_i, i \in I$; and μ_{vj} is the total CPU capacity of VM $V_j, j \in J$, as defined previously.

A penalty function is introduced to encourage applications to be packed onto active VMs such that the maximum CPU capacity is utilized. A higher the CPU utilization is given a lower the penalty. If the CPU utilization efficiency falls below 0.5, then a penalty equal to the capacity of the VM μ_{vj} is applied. When the utilization efficiency increases, the penalty decreases by half $\mu_{vj}/2$.

The maximum CPU utilization efficiency incurs 0 penalty. The CPU utilization efficiency constraint restricts overloading the VMs by considering any solution with $\eta_{cpu(j)} > 1$ as infeasible by assigning a high penalty of ∞. Therefore, the penalty $p_{cpu(j)}$ for the different values of $\eta_{cpu(j)}$ is set as follows:

$$p_{cpu(j)} = \begin{cases} \mu_{vj}, & \eta_{cpu(j)} \leqslant 0.5 \\ \mu_{vj}/2, & 0.5 < \eta_{cpu(j)} < 1 \\ 0, & \eta_{cpu(j)} = 1 \\ \infty, & \eta_{cpu(j)} > 1 \end{cases} \tag{3}$$

4.2. Memory Allocation

When application $a_i, i \in I$, with memory requirement ω_i is hosted by VM $V_j, j \in J$, the memory assigned from V_j to a_i is given by ω_{ij}, as notationally defined previously. The memory allocation efficiency $\eta_{mem(j)}$ is the ratio of ω_{ij} to ω_i as per the application profiles:

$$\eta_{mem(j)} \quad = \quad \omega_{ij}/\omega_i, \ i \in I, j \in J \tag{4}$$

Because $\omega_{ij} \geqslant \omega_i$, it follows from Equation (4) that $\eta_{mem(j)} \geq 1$. The memory allocation constraint ensures that the application has the required memory to successfully execute.

4.3. Application Completion Time

The application profiles include approximate average run-time θ_i for application $a_i, \forall i \in I$. In order to ensure application assignment efficiency, the actual completion time T_{ij} taken by the individual VM $V_j, j \in J$, to successfully execute the application $a_i, i \in I$, must fall within a threshold value set at $\alpha \cdot \theta_i$, i.e.,

$$T_{ij} \leqslant \alpha \cdot \theta_i; \quad i \in I, j \in J. \tag{5}$$

If the scope of T_{ij} is expected to fall within 50% more than θ_i, we set $\alpha = 1.5$. This constraint is put in place to ensure that the execution efficiency of the application is not compromised when producing energy-efficient assignment solutions. Every application has discrete completion times corresponding to different VM hosts. The completion times depend on the CPU availability, speed and memory available to a VM.

For example, an application allocated to VM V_1 may have the smallest energy cost and a long completion time. However, the same application executed in VM V_2 results in a slightly higher energy cost but shorter completion time. The latter provides a better solution in terms of computing performance efficiency.

4.4. Energy Cost

Energy efficiency of the presented profile-based application assignment approach for data centers is the main objective of this paper. It is modelled by minimizing the total energy cost of the application assignment. Energy cost is directly proportional to the approximate power required to carry out an application in a VM. Approximate power consumed by a physical node is calculated from the power model defined by Blackburn [29]. From this linear model, the Energy Cost C_{ij} of executing application $a_i, i \in I$, on VM $V_j, j \in J$, is calculated as the product of the CPU requirement μ_i of the application a_i and a coefficient β_{ij}:

$$C_{ij} = \beta_{ij} \cdot \mu_i \tag{6}$$

where the coefficient β_{ij} characterizes how energy-efficient the VM V_j is to host the application a_i, and it is the difference in power between the maximum and idle utilizations.

4.5. Profile-based Application Assignment Model

This subsection formally presents our profile-based application assignment model for data centers under consideration. The research problem of near-optimal allocation of applications to VMs is formulated using a penalty-based linear programming approach. The profile-based application assignment model seeks to make the best possible use of the available resources for greener and more energy-efficient assignment solutions.

The Profile-based Energy-Efficient Application Assignment Model is mathematically defined as follows:

$$\begin{aligned}
\min z \quad = \quad & \sum_{j=1}^{M}\sum_{i=1}^{N} C_{ij}x_{ij} + \sum_{j=1}^{M} p_{cpu(j)} \qquad (7)\\
\text{s.t.} \quad & \eta_{mem(j)} \geqslant 1,\ \forall j \in J;\\
& T_{ij} \leq \alpha \cdot \theta_i,\ \forall i \in I, j \in J;\\
& \sum_{i=1}^{N} \mu_i x_{ij} \leq \mu_{vj},\ \forall j \in J;\\
& \sum_{j=1}^{M} x_{ij} = 1,\ \forall i \in I;\\
& x_{ij} = 0 \text{ or } 1,\ \forall i \in I, j \in J.
\end{aligned}$$

Apart from the CPU utilization efficiency, memory allocation efficiency, application completion time and binary constraints, the model also ensures that each application must be assigned to one and only one VM. This will avoid redundancy in the form of multiple VMs attempting to execute the same application. The resources assigned to the applications hosted on a VM should not exceed the total resource capacity of the VM. This ensures that the VM is not overloaded and thus can continue to perform efficiently.

Figure 5 gives a flowchart for the working of the profile-based application assignment framework. The model is solved with the help of the proposed assignment solution in the form of a Profile-based Matching Algorithm discussed in the following subsection.

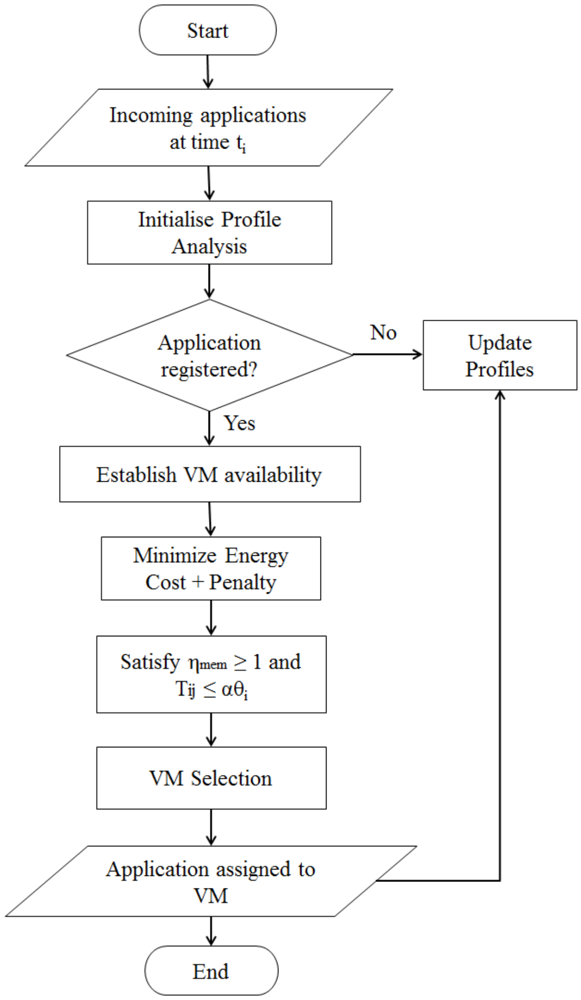

Figure 5: Profile-based linear programming model.

4.6. Penalty-based Profile Matching Algorithm

The Penalty-based Profile Matching Algorithm (PPMA) is designed to solve the profile-based application assignment model defined in Equation (7). The primary objective of PPMA is to improve energy efficiency with respect to application assignment problem in data centers. The side constraints include CPU utilization efficiency, memory and application completion time as discussed in the previous section.

To address the issues of low computing efficiency and scalability in deriving optimal solutions, PPMA aims to obtain near-optimal assignment solutions with high computing efficiency and scalability. Therefore, PPMA makes use of some heuristics to derive solutions. Developing heuristics rather than employing conventional solution techniques simplifies the problem-solving process, thus improve the scalability of the problem-solving. This is advantageous to conventional assignment algorithms such as Hungarian Algorithm, which obtains optimal solutions but has low scalability [30].

The initial algorithm (PMA) [6] is significantly improved in this paper as a Penalty-based Profile Matching Algorithm (PPMA). The new problem solution considers all three components of a data center; application, VM and PM. Energy consumption of servers were derived to demonstrate actual energy savings. Assignment of an application to a VM effectively considers the physical host of the VM. Each PM has varying values for power consumption at maximum and idle utilizations. This in turn affects the cost of assigning applications to VMs. The PPMA imposes a penalty to ensure CPU utilization efficiency as the aim of this research is to maintain a trade-off between energy consumption and resource utilization.

Being self-explained, Algorithm 3 presents the pseudo-code for PPMA. The

---

Algorithm 3: Penalty-based Profile Matching Algorithm (PPMA)

Read energy cost $[C_{ij}]_{N \times M}$ data from profiles;
Read application CPU and memory requirements from profiles;
Set scope to number of applications to be allocated;
while *Within Scope* **do**
    Initialise $[x_{ij}]_{N \times M}$ and $[Temp[i][j]]_{N \times M}$ as null matrices;
    Copy matrix E_{ij} to a temporary matrix $Temp[i][j]$;
    for *Every Application* **do**
        Set $Temp[i][1]$ as the minimum value;
        for *Every VM* **do**
            if $Temp[i][j]$ *is minimum* **then**
                Update $Temp[i][j]$ as the minimum value;
        Subtract minimum value from each value;
    for *Each matrix Temp value* **do**
        if *Zero* **then**
            Calculate penalty;
            Check memory allocation constraint;
            Check application completion time constraint;
        if *Constraints are satisfied* **then**
            Confirm allocation as $x_{ij} = 1$;
            break;
        else
            Set value $Temp[i][j]$ to a large number;
            goto step 7;

---

initial and most crucial element of the algorithm is the deciphering of the Profiles. Once the necessary data have been retrieved, the energy cost matrix $[C_{ij}]_{N\times M}$ is built. As the profiles are updated periodically, the energy cost of allocation is also updated regularly. This allows real-time events to be taken into consideration, thus improving the efficiency of the allocation manager.

The assignment solution is verified in the algorithm by determining the CPU utilization, memory efficiency and application completion time achieved. If all the conditions are satisfied, the algorithm moves on to the next assignment. In case of assignment unsuitability, the next best assignment is considered and the same process follows until a suitable assignment is achieved and the matrix $[E_{ij}]_{N\times M}$ and penalty functions are modified accordingly.

4.7. Dealing with Varied Workload

The focus of this paper is on profile-based application assignment with relatively consistent workload, which is a reasonable assumption for a large class of data centers. However, there are occasions of varied workload. The approach presented in this paper are not directly applicable in these occasions without extension and further development. Nevertheless, the concepts and principles presented in this paper are useful for future development of a dynamic version of the profile-based application assignment to deal with varied workload.

There are typical scenarios of varied workload. One example is the same application with different parameters and/or resource requirements. Another example is a new and periodic application. A further example is a new and sporadic application. For any such an application coming to the system, it undergoes profiling before its assignment to a VM. Then, it is assigned to a VM in a way that minimizes the energy consumption while meeting the resource constraints and performance requirements. How to design dynamic strategies to assignment applications to VMs for varied workload is beyond the scope of this paper, and will be investigated in our future work.

5. Experimental Studies

This section conducts experimental studies to demonstrate the profile-based application assignment approach presented in this paper. The effectiveness of the approach is evaluated from the following four aspects: feasibility, scalability, CPU utilization, and energy efficiency. The section begins with experimental setups followed by detailed experimental studies.

5.1. Experimental Setup

The experimental studies are conducted using two different test setups: Test Setup 1 and Test Setup 2. Originally investigated in our preliminary work [6], Test Setup 1 is used to determine the feasibility and scalability of our approach. Test Setup 2 is used to determine the efficiency of the profiling application management approach over general and workload history approaches. Shown in Table 1, the two setups are described below:

Table 1: Two test setups with different scenarios.

Test Setup 1 (100 PMs)						
Scenario	**1**	**2**	**3**	**4**	**5**	
VMs	400	400	800	800	1000	
Applications	500	1500	2000	2500	4000	
Test Setup 2 (150 PMs)						
Scenario	**6**	**7**	**8**	**9**	**10**	**11**
VMs	100	400	800	1200	1600	2000
Applications	500	1000	2000	3000	4000	5000

- **Test Setup 1:** A data center consisting of 100 PMs with an average of four to ten VMs each is considered. The total number of VMs ranges from 400 to 1000. The total number of applications varies from 500 to 4000. The scenarios of Test Setup 1 are presented in the first half of Table 1.

- **Test Setup 2:** A data center consisting of 150 PMs is considered. Each server is capable of hosting up to 15 VMs. For our evaluation, six different scenarios are considered where the number of applications ranges from 500 to 5000 with corresponding number of VMs as shown in the second half of Table 1.

In our experiments, the logs from a real data center are used to create realistic VM and PM profiles. The application profiles are synthetically generated as in our preliminary work [6]. The application and VM profiles generated from Test Setup 2 are enclosed in Appendix A. All evaluations are carried out on a Windows platform of Intel(R) Core(TM) i7-2640M CPU at 2.80GHz using C and Python programming.

The results derived from PPMA will be compared with those obtained from Hungarian Algorithm. A high-level description of Hungarian Algorithm is depicted in Algorithm 4, which is self-explained. For more detailed information about Hungarian Algorithm, please refer to references [6] and [30].

5.2. Feasibility

Test Setup 1 is used to validate the feasibility of the profile-based application assignment framework. The application assignment results for Scenario 2 are presented in Figure 6. Analysing the results shows that an average number of 15 applications are hosted by each server through VMs, with a maximum of 32 applications hosted by a server. More than 25 applications are hosted by 15% of the servers individually. 11% of the total servers are idle and can be switched off by the allocation manager. The proposed approach successfully solves the penalty-based linear optimization model (Equation 7) and satisfies the resource constraints, thereby supporting the feasibility of the presented Profile-based Assignment Model.

Algorithm 4: Hungarian Algorithm (HA).

```
1  Convert [C_ij]_{N×M} into square energy cost matrix using dummy values ;
2  for Each Row do
3      Identify and subtract minimum value from all elements;
4  for Each Column do
5      Identify and subtract minimum value from all elements;
6  while Solution matrix not complete do
7      if Column contains more than one '0' element then
8          Repeat step 2 forall columns ;
9      for Each Column do
10         Identify columns with negative elements ;
11         Select minimum value and add to each element ;
12     Flag rows and columns with '0' elements ;
13     Identify and subtract minimum value from unflagged elements ;
14     Add minimum value from unflagged elements to twice flagged
         elements ;
```

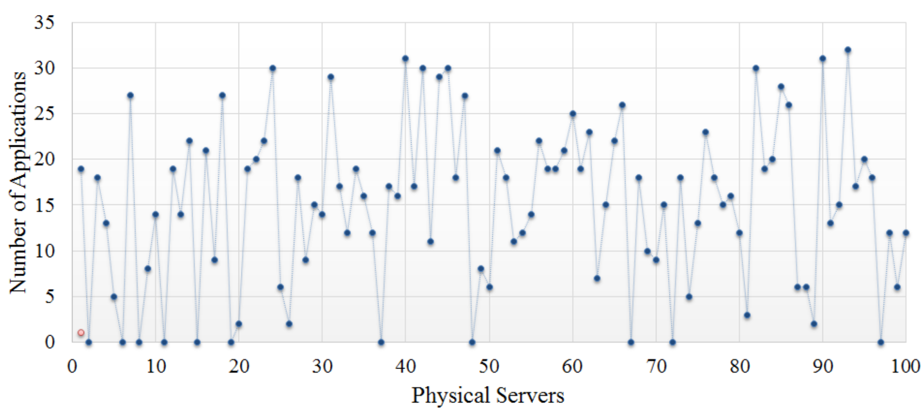

Figure 6: Application assignment for Scenario 2 of Test Setup 1.

5.3. Scalability

The scalability of the Penalty-based Profile Matching Algorithm (PPMA) is compared with that of the Hungarian Algorithm (HA) described in Algorithm 4. Both algorithms are applied to the five scenarios of Test Setup 1 (Table 1). The solution time in seconds for each of the two algorithms is obtained and tabulated in Table 2. The results demonstrate that as the numbers of applications and VMs increase, the PPMA is capable of finding near-optimal solutions in much lesser time than the HA. The Hungarian Algorithm gives optimal assignment solutions but compromises heavily on the time taken to obtain the solution due to the large problem size. This demonstrates that the presented PPMA scales well.

Table 2: Comparisons of solution time (sec) from the two algorithms for Test Setup 1.

Scenario	**1**	**2**	**3**	**4**	**5**
The Hungarian algorithm	4	27	41	72	248
PPMA of this work	5	22	26	31	52

5.4. CPU Utilization Efficiency

This subsection aims to demonstrate that the presented profile-based assignment framework makes the best possible use of available resources such as the CPU of the server nodes. The scenarios from Test Setup 1 (Table 1) are considered in this case study. There is a high variance in the results from the PPMA of this work and the Hungarian Algorithm for problems of smaller sizes. This is demonstrated in Figure 7 for the CPU utilization efficiency variation graph over a 24 hour period for Scenario 1. However, as the ratio of applications assigned to VMs increases with the problem size, the CPU utilization efficiency also increases. The average CPU utilization efficiency of the PPMA of this work and the Hungarian Algorithmfor all scenarios of Test Setup 1 (in the first half of Table 1) is compared in Table 3. The PPMA of this work achieves results that are close in utilization efficiency to the Hungarian Algorithm with the increase in the problem size as evidenced by a decrease in variation from 19% to 1.1%.

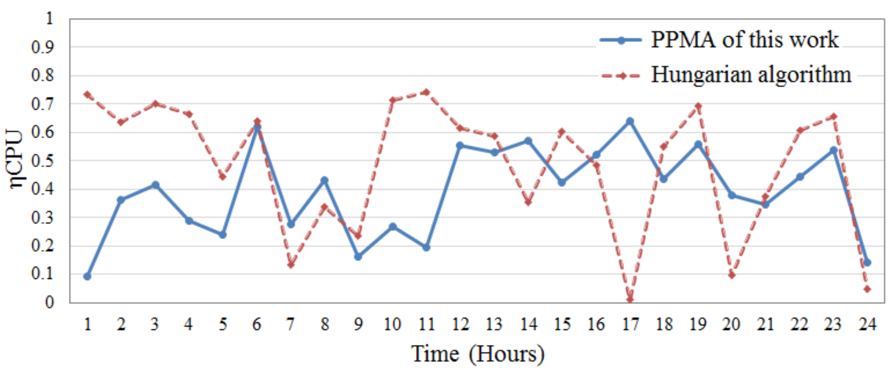

Figure 7: Average CPU utilization efficiency derived from the PPMA of this work (solid line) and the Hungarian algorithm (dashed line) over 24 hours for Scenario 1 of Test Setup 1.

Table 3: Average CPU utilization efficiency (η_{cpu}) for Test Setup 1.

Scenario	**1**	**2**	**3**	**4**	**5**
The Hungarian algorithm	0.486	0.531	0.545	0.578	0.702
PPMA of this work	0.394	0.499	0.526	0.569	0.694

Although the Hungarian Algorithm is more efficient than PPMA, it compromises on the consistency and scalability of the assignment solutions. The PPMA maintains consistent CPU utilization efficiency with the increase in the scale of the assignment problems.

5.5. Energy Efficiency

This subsection demonstrates the energy efficiency of the profile-based application assignment approach for Test Setups 1 and 2 implementations. Test Setup 1 (discussed in our previous work [6]) is used to compare the energy results derived by the PPMA of this work and the optimal Hungarian algorithm. Test Setup 2 is used to validate the energy-efficiency of the Profiling application management approach when compared with the commonly used General and Workload application management approach.

5.5.1. Energy Efficiency in Test Setup 1

The Hungarian algorithm provides a high quality of solution at the cost of a high solution time and poor scalability [31]. Therefore, the energy-efficient solutions provided by the PPMA of this work is compared with that of the optimal results provided by the Hungarian algorithm. In order to evaluate the energy-efficiency, the average CPU utilization is deduced after the application assignment using both algorithms. The energy consumption E for the Test Setup 1 scenarios (shown in the first half of Table 1) is then calculated using the following equations [29].

$$P_k = \frac{(P_k^{max} - P_k^{idle}) * \eta_{cpu(k)}}{100} + P_k^{idle}, \tag{8}$$

$$E = \int_{t_0}^{t_1} P_k(t)dt \tag{9}$$

Power consumed for machine k at maximum utilization and idle state is given by P_k^{max} and P_k^{idle}, respectively. For calculation purposes, it is assumed that $P_k^{max} = 350$W and $P_k^{idle} = 200$W. Total CPU utilization of the server is represented by $\eta_{cpu(k)}$.

Figure 8 demonstrates the energy consumption graph of a server in a 24 hour period for both the PPMA and Hungarian Algorithm. The average energy consumptions for the PPMA and Hungarian Algorithm are 286.5 Wh and 269.75 Wh, respectively. The results confirm that the PPMA is only 5.85% worse in energy-efficiency than the ideal Hungarian Algorithm. In order to demonstrate the decrease in variation of energy consumption results as the problem size increases, a bar graph displaying the total energy consumption for all scenarios of Test Setup 1 is presented in Figure 9. The PPMA results show a 11.8% to 0.4% variation from the Hungarian Algorithm solutions. This proves that the proposed PPMA is sufficiently energy-efficient.

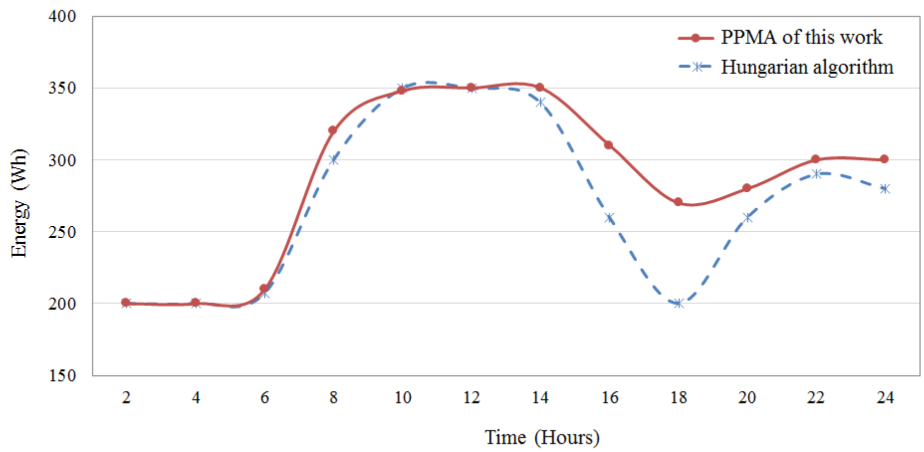

Figure 8: Energy consumption of a server using the PPMA of this work (solid line) and the Hungarian algorithm (dashed line).

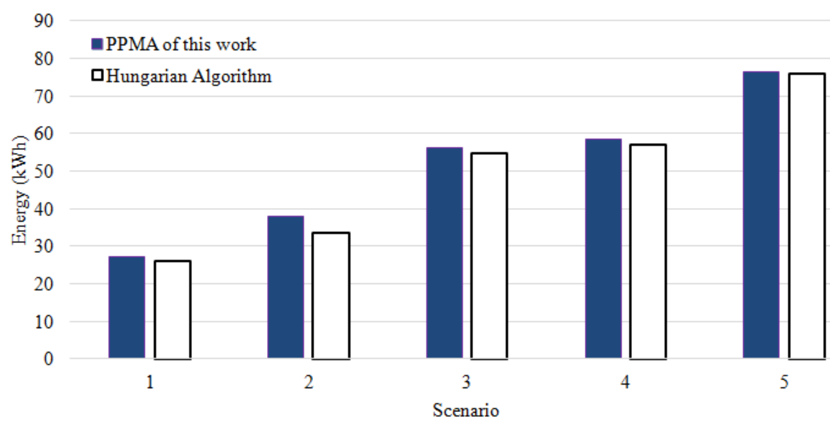

Figure 9: Total energy consumption derived from the PPMA of this work (filled bars) and the Hungarian algorithm (unfilled bars) for Test Setup 1 scenarios.

5.5.2. Energy-Efficiency in Test Setup 2

The effectiveness of the Profile-based application assignment approach in terms of execution time and energy-efficiency is evaluated with comparison with other assignment approaches. Test Setup 2, as seen in the second half of Table 1, is used to compare the results obtained by the following three approaches:

- General Application Assignment;
- Workload History based Application Assignment; and
- Profile-based Application Assignment

The general application assignment is the simplest form of allocation and does not implement any efficiency strategy. The applications are allocated at the time of their arrival to the first available VM that fits the execution requirements in CPU, memory and run-time.

Workload History based application assignment utilizes the recorded logs of CPU cycles with corresponding time of the VMs to make allocation decisions. This approach functions on the assumption that workload behaviour of a data center varies little during day-to-day operations. However, it only considers the VM information, whereas our Profiles are built for applications, VMs and PMs. Moreover, workload history approach does not have the option of updating data unlike the profiling approach.

All three approaches: General, Workload History and Profiling are implemented with a simple First-Fit Decreasing (FFD) assignment algorithm in the three-layer energy management (Figure 1). Our assignment problem resembles a bin-packing problem:

1. **General Assignment** - The applications arrive at the data center. The CPU requirement is determined. Applications are arranged in terms of decreasing CPU requirement. The FFD is invoked and the application assigned to the first VM that can accommodate the requirements. Algorithm 5 gives the process of this approach.
2. **Workload History Assignment** - During time interval $T-1$, the VMs are arranged in decreasing order of CPU availability as per the workload logs. Applications arriving at the data center during time interval T are assigned to the first suitable VM with the help of FFD algorithm. Algorithm 6 represents the process of this approach.

Algorithm 5: General Assignment Approach

```
for Each Application do
    Determine CPU and memory requirement;
    if Requirement satisfied then
        Invoke FFD Algorithm (Allocate to first available VM);
```

Algorithm 6: Workload History Approach

```
for Interval Time T-1 do
    Workload Logs: VM arranged in decreasing order of CPU
     availability at Time T;
for Interval Time T do
    for Each Application do
        if Requirement satisfied then
            Invoke FFD Algorithm (Allocate to first available VM);
```

3. **Profile-based Assignment** - During time interval $T-1$, the energy cost of allocation of each predicted application to a VM is retrieved. A VM yielding lowest cost is selected. At interval time T, the FFD allocates the application to the pre-selected VM with the minimum energy cost incurred. Algorithm 7 describes the process of this approach.

The results of energy-efficiency and execution time for the six different test scenarios in Test Setup 2 are presented in Table 4 for the general, workload history and profiling approaches.

Consider the execution time behaviour as seen in Figure 10. The General allocation initially has the lowest execution time upto 1500 applications. However, as the number of applications increases, there is a corresponding increase in the execution time. Both workload history and profiling approaches have a steady, consistent, and linear increase with the number of applications. On examination, the profiling approach presented in this paper is 5% more efficient than the workload history approach in execution time.

Figure 11 shows the total energy consumption of the data center with re-

Algorithm 7: Profiling Approach of This Work

for *Interval Time T-1* **do**
- **Profiles:** Determine applications arriving at Time T;
- **Profiles:** Retrieve associated energy cost of allocation of each application;
- **Profiles:** Select best possible VM hosts $Selected_{VM} = \{VM_1, VM_2, ...\}$;

for *Interval Time T* **do**
- **for** *Each Application* **do**
  - **if** *Requirement satisfied* **then**
    - Invoke FFD Algorithm (Allocate to first available VM);

Table 4: Energy efficiency and execution time performance from Test Setup 2 for General, Workload History and our Profiling approaches.

Scenario	Our Profiling		Workload History		General	
	Energy (Wh)	Time (s)	Energy (Wh)	Time (s)	Energy (Wh)	Time (s)
6	25623	1.2	26186	1	27015	1
7	25948	1.9	26748	2.2	28975	1.5
8	26782	4.4	27493	4.8	32675	4.1
9	27835	5.7	30759	7.1	33402	8.3
10	28940	6.9	32472	7.9	37238	12.4
11	32752	8.6	35871	9.2	39478	14.7

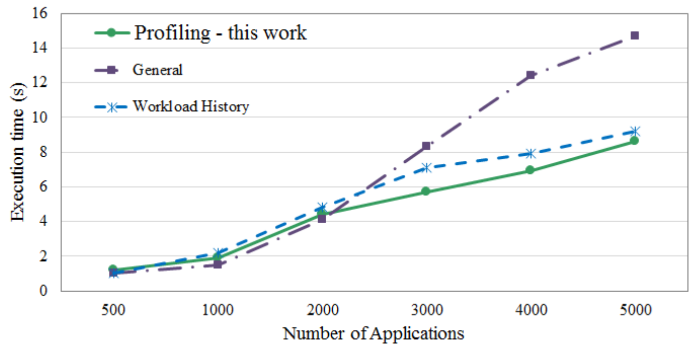

Figure 10: Comparisons of execution time for our Profiling approach of this work (solid line), Workload History approach (dashed line) and General approach (dash-dotted line).

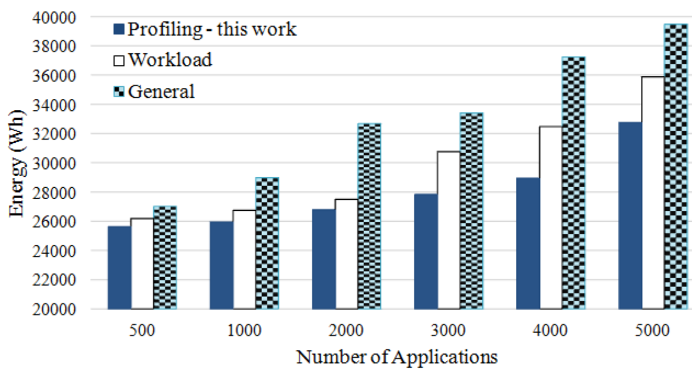

Figure 11: Comparisons of energy-efficiency for our Profiling approach of this work (filled bars), Workload History approach (unfilled bars) and General approach (patterned bars), respectively.

spect to the increasing number of applications for all three approaches. The General approach consumes the most energy as application-VM allocations are not optimal due to the absence of energy cost constraints. The Workload History approach is efficient upto 2000 applications, however increases significantly with the increase in the number of applications. The Profiling approach presented in this paper outperforms the other two approaches in energy consumption due to energy cost based allocations derived from the profiles.

Table 5 provides an overview of the three different approaches considered in the experimental evaluations. The standard deviations in terms of energy and execution time demonstrate that our Profiling approach is consistent and more efficient than the other approaches.

5.6. Summary of Experimental Studies

Experimental studies have been conducted on the feasibility, scalability, effectiveness, CPU utilization efficiency and energy-efficiency of the proposed

Table 5: Overview of the three approaches ($\surd$: known; $\times$: unknown).

Data/Strategy	**General approach**	**Workload History approach**	**Profiling (this work)**
Virtual Machine	$\times$	$\surd$	$\surd$
Application	$\times$	$\times$	$\surd$
Std Deviation: Energy	$4,735.38$	$3,808.49$	$2,639.53$
Std Deviation: Exec Time	5.74	3.27	2.87

Profile-based Application Assignment approach. The experimental results are summarized as follows:

- The PPMA is feasible and scalable within the tested range of 100 to 2000 VM nodes;
- There is a trade-off between scalability and CPU utilization efficiency for increasing problem sizes;
- The profile-based application assignment approach is more energy-efficient with steady execution times in comparison with commonly used General and Workload History assignment approaches; and
- The energy efficiency achieved is close to that of the optimal Hungarian Algorithm solution.

It is worth mentioning that the overhead of the profile-based application assignment is minimal for data centers with relatively consistent workloads considered in this paper. The profiles of the data centers can be established offline. The un-profiled workload that requires online processing is insubstantial. With the established profiles, static assignment of applications to virtual machines can be scheduled in advance. Efficient dynamic scheduling of application assignment for data centers with uncertain and variable workloads is beyond of the scope of this paper and will be investigated in our future work.

The case studies presented in this paper have been carried out by using the raw data collected from a real-world data center. The data sets are not available to the public. One may asks for verifiability and reproducibility of our results if the data are not available to the public. Keep in mind that the main theme of the paper is the profile-based approach, which includes the concepts of profiles, profile building, formulation of the application assignment problem as a penalty-based optimization subject to a number of constraints, and a penalty-based profile matching algorithm to solve the optimization problem. To verify the approach, any data sets collected from a similar type of data center are fine as long as they are used by following our approach presented in the paper. For example, one may collect data from a data center of his/her own institution. In this sense, our work presented in this paper does not have the problem of the lack of verifiability and reproducibility.

6. Conclusion

One of the significant research problems concerning data centers and cloud computing is how to reduce the energy consumption whilst maintaining high performance efficiency. A novel concept of energy-efficient application assignment using Profiles has been presented in this paper. From this concept, realistic Application, VM and PM Profiles have been built from the raw data center logs. A profile-based application assignment framework has also been established, and an assignment solution has further been derived in the from of a Penalty-based Profile Matching Algorithm. Experimental studies have shown that the profile-based application assignment approach is feasible, scalable and effective in comparison with other existing approaches, implying greener and more energy-efficient assignment solutions with acceptable CPU utilization efficiency and execution times within their deadlines.

Our future work will consider varied workload. This requires dynamic strategies for profile-based application assignment. The development of a dynamic version of the approach presented in this paper will enable implementation of the profile-based application assignment in a wider class of data centers.

Contributions of the authors: M. Vasudevan conducted detailed research and experiments, and wrote the paper. Y.-C. Tian designed the project and supervised the research and manuscript writing. M. Tang provided guidance on profile building and helped polishing up the manuscript. E. Kozan supervised formulation of constrained optimization problems.

Acknowledgement

Authors Y.-C. Tian and E. Kozan would like to acknowledge the Australian Research Council (ARC) for its support under the Linkage Projects Scheme (Grant No. LP140100394). This work is also supported in part by the Science and Technology Department of Shanxi Provincial Government of China under the International Collaboration Projects Scheme (Grant No. 2015081007), and the same Chinese government department under the Talent Projects Grant Scheme in 2016, both to Author Y.-C. Tian.

References

[1] J. Koomey, Growth in data center electricity use 2005 to 2010, Tech. rep., Analytics Press, Oakland, California, USA (1 Aug 2011).

[2] M. Webb, Smart 2020: Enabling the low carbon economy in the information age, Tech. rep., The Climate Group and the Global e-Sustainability Initiative, London, UK (2008).

[3] J. Whitney, P. Delforge, Scaling up energy efficiency across the data center industry: evaluating key drivers and barriers (Issue Paper), Natural Resources Defense Council (NRDC), August 2014.

[4] K. Le, R. Bianchini, M. Martonosi, T. Nguyen, Cost- and energy-aware load distribution across data centers, in: Proceedings of HotPower, Montana, USA, 2009, pp. 1–5.

[5] A. Greenberg, J. Hamilton, D. A. Maltz, P. Patel, The cost of a cloud: research problems in data center networks, SIGCOMM Computer Communication Review 39 (1) (2008) 68–73.

[6] M. Vasudevan, Y.-C. Tian, M. Tang, E. Kozan, Profiling: an application assignment approach for green data centers, in: Proceedings of the IEEE 40th Annual Conference of the Industrial Electronics Society, Dallas, TX, USA, 29 Oct - 1 Nov 2014, pp. 5400–5406.

[7] Q. Yi, S. Singh, Minimizing energy consumption of fattree data center networks, SIGMETRICS Perform. Eval. Rev. 42 (3) (2014) 67–72.

[8] S. Ren, M. A. Islam, A first look at colocation demand response, SIGMETRICS Perform. Eval. Rev. 42 (3) (2014) 73–75.

[9] H. Yoon, J. Meza, N. Muralimanohar, N. P. Jouppi, O. Mutlu, Efficient data mapping and buffering techniques for multilevel cell phase-change memories, ACM Trans. Archit. Code Optim. 11 (4) (2014) 40:1–40:25.

[10] K. A. Kumar, A. Quamar, A. Deshpande, S. Khuller, Sword: Workload-aware data placement and replica selection for cloud data management systems, The VLDB Journal 23 (6) (2014) 845–870.

[11] A. Beloglazov, J. Abawajyb, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future Generation Computer Systems 28 (2012) 755–768.

[12] X. León, L. Navarro, A stackelberg game to derive the limits of energy savings for the allocation of data center resources, Future Generation Computer Systems 29 (2013) 74–83.

[13] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, H. Liu, Energy proportional datacenter networks, in: Proceedings of the 37th Annual International Symposium on Computer Architecture, ISCA '10, ACM, New York, NY, USA, 2010, pp. 338–347.

[14] H. Liu, B. He, Reciprocal resource fairness: Towards cooperative multiple-resource fair sharing in IaaS clouds, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '14, IEEE Press, Piscataway, NJ, USA, 2014, pp. 970–981.

[15] J. Combs, J. Nazor, R. Thysell, F. Santiago, M. Hardwick, L. Olson, S. Rivoire, C.-H. Hsu, S. W. Poole, Power signatures of high-performance computing workloads, in: Proceedings of the 2Nd International Workshop on Energy Efficient Supercomputing, E2SC '14, IEEE Press, Piscataway, NJ, USA, 2014, pp. 70–78.

[16] S. Yeo, M. M. Hossain, J.-C. Huang, H.-H. S. Lee, ATAC: Ambient temperature-aware capping for power efficient datacenters, in: Proceedings of the ACM Symposium on Cloud Computing, SOCC '14, ACM, New York, NY, USA, 2014, pp. 17:1–17:14.

[17] L. Chen, H. Shen, K. Sapra, Distributed autonomous virtual resource management in datacenters using finite-markov decision process, in: Proceedings of the ACM Symposium on Cloud Computing, SOCC '14, ACM, New York, NY, USA, 2014, pp. 24:1–24:13.

[18] J. Mars, L. Tang, K. Skadron, M. Soffa, R. Hundt, Increasing utilization in modern warehouse-scale computers using bubble-up, IEEE Micro 32 (3) (2012) 88–99.

[19] J. Mars, L. Tang, R. Hundt, K. Skadron, M. L. Soffa, Bubble-up: Increasing utilization in modern warehouse scale computers via sensible co-locations, in: Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-44, New York, NY, USA, 2011, pp. 248–259.

[20] H. Yang, A. Breslow, J. Mars, L. Tang, Bubble-flux: Precise online QoS management for increased utilization in warehouse scale computers, SIGARCH Compututer Architecture News 41 (3) (2013) 607–618.

[21] Q. Chen, L. Zheng, M. Guo, Z. Huang, Eewa: Energy-efficient workload-aware task scheduling in multi-core architectures, in: IEEE International Parallel Distributed Processing Symposium Workshops (IPDPSW), 2014, pp. 642–651.

[22] H. Nguyen, Z. Shen, X. Gu, S. Subbiah, J. Wilkes, Agile: Elastic distributed resource scaling for infrastructure-as-a-service, in: Proceedings of the 10th International Conference on Autonomic Computing (ICAC 13), USENIX, San Jose, CA, 2013, pp. 69–82.

[23] D. Williams, H. Jamjoom, Y.-H. Liu, H. Weatherspoon, Overdriver: Handling memory overload in an oversubscribed cloud, in: Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, VEE'2011, New York, NY, USA, 2011, pp. 205–216.

[24] A. V. Do, J. Chen, C. Wang, Y. C. Lee, A. Zomaya, B. B. Zhou, Profiling applications for virtual machine placement in clouds, in: Proceedings of the IEEE 4th International Conference on Cloud Computing, Washington, DC, USA, 2011, pp. 660–667.

[25] K. Ye, Z. Wu, C. Wang, B. B. Zhou, W. Si, X. Jiang, A. Zomaya, Profiling-based workload consolidation and migration in virtualized data centers, IEEE Transactions on Parallel and Distributed Systems 26 (3) (2015) 878–890.

[26] X. Shi, H. Jiang, L. He, H. Jin, C. Wang, B. Yu, F. Wang, Eapac: An enhanced application placement framework for data centers, in: Proceedings of the IEEE 14th International Conference on Computational Science and Engineering, Dalian, China, 2011, pp. 34–43.

[27] C. Tang, M. Steinder, M. Spreitzer, G. Pacifici, A scalable application placement controller for enterprise data centers, in: Proceedings of the 16th International Conference on World Wide Web, Alberta, Canada, 2007, pp. 331–340.

[28] U. Lublin, D. G. Feitelson, The workload on parallel supercomputers: Modeling the characteristics of rigid jobs, Journal of Parallel and Distributed Computing 63 (2001) 2003.

[29] M. Blackburn, Five ways to reduce data center power consumption (white paper), The Green Grid, 2008.

[30] J. A. Winter, D. H. Albonesi, The scalability of scheduling algorithms for unpredictably heterogeneous CMP architectures, in: Proceedings of the 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Anchorage, Alaska, USA, 2008, pp. 42–51.

[31] Y. Chaobo, Z. Qianchuan, Advances in assignment problem and comparison of algorithms, in: Proceedings of the 27th Chinese Control Conference, Kunming, Yunnan, China, 2008, pp. 607–611.

Appendix A. Test Setups in Experimental Studies

Setups of VMs and applications for the experimental studies carried out in this paper are summarized in this appendix. Fig. A.12 shows the test setup for VM profiles. Setup for applications in depicted in Fig. A.13.

Table A.6: Profile for a physical machine (PH015) over 24 hours.

Interval	**Min CPU (%)**	**Max CPU (%)**	**Avg CPU (%)**
0.00 - 1.00	9.575	10.24	9.908
1.00 - 2.00	9.649	10.24	9.943
2.00 - 3.00	8.298	8.298	8.298
3.00 - 4.00	8.559	8.559	8.559
4.00 - 5.00	12.61	12.61	12.61
5.00 - 6.00	12.04	12.04	12.04
6.00 - 7.00	10.61	23.66	16.01
7.00 - 8.00	9.863	11.5	10.68
8.00 - 9.00	9.863	10.21	10.04
9.00 - 10.00	10.21	23.04	15.55
10.00 - 11.00	10.48	10.48	10.48
11.00 - 12.00	9.625	9.625	9.625
12.00 - 13.00	10.48	10.48	10.48
13.00 - 14.00	10.65	10.65	10.65
14.00 - 15.00	9.674	9.674	9.674
15.00 - 16.00	9.467	29.14	17.52
16.00 - 17.00	10.38	10.64	10.51
17.00 - 18.00	10.64	10.94	10.79
18.00 - 19.00	9.949	10.94	10.45
19.00 - 20.00	9.052	9.949	9.5
20.00 - 21.00	9.052	9.429	9.24
21.00 - 22.00	9.429	10.7	10.27
22.00 - 23.00	11.66	18.23	13.85
23.00 - 24.00	10.1	10.26	10.21
		Mean	11.12
		Standard Deviation	2.36

Table A.7: Profile of a VM (VM ID: 23) over 24 hours.

Interval	**Used CPU (%)**	**Used Mem (%)**
0.00 - 1.00	13.15	21.18
1.00 - 2.00	10.61	16.6
2.00 - 3.00	13.4	15.06
3.00 - 4.00	10.18	16.42
4.00 - 5.00	11.93	16.93
5.00 - 6.00	10.2	17.18
6.00 - 7.00	9.345	17.37
7.00 - 8.00	7.501	18.25
8.00 - 9.00	13.27	19.07
9.00 - 10.00	20.38	19.94
10.00 - 11.00	26.07	19.95
11.00 - 12.00	11.76	20.41
12.00 -13.00	18.97	20.62
13.00 - 14.00	24.58	21.1
14.00 - 15.00	16.11	20.95
15.00 - 16.00	15.37	21.06
16.00 - 17.00	22	21.15
17.00 - 18.00	15.36	21.22
18.00 - 19.00	9.096	21.27
19.00 - 20.00	10.67	21.4
20.00 - 21.00	10.16	21.69
21.00 - 22.00	9.254	21.47
22.00 - 23.00	8.65	21.41
23.00 - 24.00	10.35	21.51
Mean	13.68	19.72
Standard Deviation	5.20	2.04

ID	Avg_CPU	ID	Avg_CPU	ID	Avg_CPU
VM0	13.58	VM34	14.706	VM68	14.03
VM1	9.818	VM35	19.436	VM69	12.605
VM2	13.585	VM36	8.9825	VM70	14.385
VM3	12.975	VM37	10.6465	VM71	15.4445
VM4	12.44	VM38	10.1465	VM72	24.035
VM5	12.345	VM39	6.7105	VM73	14.2805
VM6	12.11	VM40	8.964	VM74	9.2155
VM7	13.125	VM41	6.0935	VM75	8.932
VM8	13.635	VM42	6.119	VM76	9.087
VM9	18.25	VM43	11.9095	VM77	8.9485
VM10	23.75	VM44	8.6385	VM78	9.2475
VM11	16.595	VM45	8.8425	VM79	9.1835
VM12	15.81	VM46	12.047	VM80	12.295
VM13	15.075	VM47	13.52	VM81	19.0805
VM14	15.555	VM48	13.315	VM82	18.3775
VM15	17.575	VM49	8.0615	VM83	16.535
VM16	13.475	VM50	5.774	VM84	12.593
VM17	13.56	VM51	14.2275	VM85	7.819
VM18	12.905	VM52	6.1685	VM86	10.189
VM19	12.805	VM53	6.0255	VM87	6.6105
VM20	14.175	VM54	5.918	VM88	9.1925
VM21	17.06	VM55	21.3255	VM89	4.738
VM22	24.815	VM56	7.19	VM90	7.8355
VM23	15.83	VM57	5.9205	VM91	7.2445
VM24	16.21	VM58	22.8915	VM92	16.501
VM25	15.84	VM59	24.0145	VM93	11.165
VM26	15.015	VM60	14.538	VM94	11.2195
VM27	24.28	VM61	18.057	VM95	15.855
VM28	15.955	VM62	11.942	VM96	13.234
VM29	15.495	VM63	11.639	VM97	6.29
VM30	15.755	VM64	6.6875	VM98	6.186
VM31	17.135	VM65	9.385	VM99	6.337
VM32	24.04	VM66	9.2295		
VM33	28.801	VM67	24.95		

Figure A.12: Test setup: VM profiles

ID	Arr_time	run	CPU	S#
1	26	1	28	0
2	46	4	8	0
3	50	5	4	0
4	201	51	2	0
5	26	5	8	0
6	1044	11266	1	1
7	17	30646	1	1
8	590	24701	16	1
9	161	20669	16	1
10	1924	2700	16	1
11	193	55	4	1
12	8370	7	1	0
13	461	23	4	0
14	78	113	4	0
15	771	10	6	0
16	10331	34	2	0
17	153	9	8	0
18	426	6	4	0
19	781	2	16	0
20	228	9	8	0
21	538	71	16	0
22	338	2	8	0
23	31	5	1	0
24	36	92	41	0
25	1221	2	4	0
26	1248	7	2	0
27	1065	6	4	0
28	7748	158	4	0
29	43	30	1	0
30	27	7	6	0
31	15	3	4	0
32	17	3089	24	0
33	204	26	4	0
34	126	4873	8	0
35	82	10	4	0
36	40	2	8	0
37	56	26	2	0
38	67	18	1	0
39	85	21	5	0
40	205	16	8	0
41	17	9	8	0
42	11	7	16	0
43	12	2	4	0
44	6	6	8	0
45	53	8	16	0
46	9	249	8	0
47	46	115	8	0
48	7	1	4	0
49	22	88	24	0
50	25	2	2	0
51	72	59	8	0
52	373	24	1	0
53	33	84	21	0
54	25	10	1	0
55	188	1	1	0
446	63	15	1	0
447	5	112	1	0
448	19	76	32	0
449	105	4	1	0
450	7	7	4	0
451	62	6	32	0
452	21	595	32	0
453	36	48	16	0
454	4	26	3	0
455	54	48	4	0
456	3	4	4	0
457	30	14	16	0
458	6	10	4	0
459	3	96	4	0
460	32	7	32	0
461	29	1	20	0
462	239	25	4	0
463	67	14	8	0
464	256	109	16	0
465	8	66	10	0
466	35	6	2	0
467	2	13	1	0
468	78	125	8	0
469	102	4	32	0
470	51	11	4	0
471	2	3	5	0
472	19	5	8	0
473	14	11	4	0
474	10	31	8	0
475	8	48	32	0
476	372	11	8	0
477	308	4	4	0
478	85	5	8	0
479	31	16	16	0
480	11	6	37	0
481	155	216	2	0
482	30	3	4	0
483	22	74	1	0
484	99	6	32	0
485	404	54	8	0
486	274	9	2	0
487	47	2	4	0
488	37	16	32	0
489	11	5	8	0
490	7	2	12	0
491	1	14	2	0
492	4	1	8	0
493	61	7	4	0
494	529	2	8	0
495	90	81	4	0
496	11	181	28	0
497	212	21	40	0
498	4024	7	4	0
499	2	23	16	0
500	16	19	8	0

Figure A.13: Test setup: application profiles