Journal of Computer Sciences

Original Research Paper

Towards Solving the Problem of Virtual Machine Placement in Cloud Computing: A Job Classification Approach

[1]Auday Al-Dulaimy, [2]Ahmed Zekri, [3]Wassim Itani and [4]Rached Zantout

[1]*Department of Mathematics and Computer Science, Beirut Arab University, Beirut, Lebanon*
[2]*Department of Mathematics and Computer Science, Beirut Arab University, Beirut, Lebanon,*
On leave from the Department of Mathematics and Computer Science, Alexandria University, Alexandria, Egypt
[3]*Department of Electrical and Computer Engineering, Beirut Arab University, Beirut, Lebanon*
[4]*Department of Electrical and Computer Engineering, Rafic Hariri University, Beirut, Lebanon*

Article history
Received: 02-12-2015
Revised: 23-03-2016
Accepted: 02-04-2016

Corresponding Author:
Auday Al-Dulaimy
Department of Mathematics and Computer Science, Beirut Arab University, Beirut, Lebanon
Email:
auday.aldulaimy@gmail.com
a.aldulaimy@student.bau.edu.lb

Abstract: Cloud Computing is a paradigm that delivers services by providing an access to wide range of shared resources which are hosted in cloud data centers. One of the recent challenges in this paradigm is to enhance the energy efficiency in these data centers. In this study, a model that identifies common patterns for the jobs submitted to the cloud is proposed. This model is able to predict the type of the job submitted and accordingly, the set of users' jobs is classified into four subsets. Each subset contains jobs that have similar requirements. In addition to the jobs' common pattern and requirements, the users' history is considered in the jobs' type prediction model. The goal of job classification is to find a way to propose useful strategy that helps to improve power efficiency. Based on the process of jobs' classification, the best fit virtual machine is allocated to each job. Then, the virtual machines are placed on the physical machines according to a novel strategy, called Mixed Type Placement strategy. The core idea of the proposed strategy is to place virtual machines of the jobs of different types in the same physical machine whenever possible. The placement process is based on Multi Choice Knapsack Problem which is a generalization of the classical Knapsack Problem. This is because different types of jobs do not intensively use the same compute or storage resources in the physical machine. This strategy minimizes the number of active physical machines which, in turn, leads to major reduction in the total energy consumption in the data center. The total execution time and the cost of executing the jobs submitted are considered in the placement process. To evaluate the performance of the proposed strategy, the CloudSim simulator is used with a real workload trace to simulate the cloud computing environment. The results show that the proposed strategy outperform both Genetic Algorithm and Round Robin from energy efficiency perspective.

Keywords: Cloud Computing, Data Center, Virtualization Management, Energy Efficiency

Introduction

With the rapid growth in the cloud computing model, the reduction in the total energy consumed by the cloud data centers has become essential. This reduction can be achieved by observing how the power is delivered to computing resources and how these resources are utilized to serve the jobs which request these services.

The continuous growth in the cloud computing model poses a challenge for existing works to enhance the energy efficiency in cloud data centers. Hence, the need for improving the existing resource allocation and management algorithms in cloud data centers and proposing new ones is highly recommended. In this study, a new model is suggested to predict the job type in the workload based on common behaviors and patterns of the job. If there is a history for the user who submits the job to the cloud, it will also be considered in predicting the job type. If not, the job requirements are the parameters which are considered in the classification process. According to the job type, the best fit Virtual Machine (VM) is allocated to job. Then, VMs are placed to the Physical Machines (PMs) considering that no VMs of the same type of jobs will consolidate on the same PM relying on the Knapsack Problem (KP). High Performance Computing (HPC) jobs can be effectively combined with Data Intensive (DI) jobs on the same PM. HPC jobs mostly request compute resources, whereas DI jobs request storage and network bandwidth resources. In other words, HPC jobs and DI jobs do not intensively use the same resources. This can potentially lead to more efficient resource provisioning and reduction in the number of switched-on PMs and, consequently, higher energy efficiency. If there is no available PM for combining VMs of different types of jobs, then the KP algorithm to place the VMs is used. To note, Knapsack is not the only possible way for solving the VM placement problem, there are other methods, for example the bin packing algorithm used in (Beloglazov, 2013). KP is used in this study because it has different variants; one of them is called the Multi Choice Knapsack Problem (MCKP). It is used when the items should be chosen from different sets to be combined in one knapsack (Pisinger, 1995). This suits the proposed model of combining two VMs from two different sets into the same PM.

Thus, reducing energy consumption can be achieved through two situations addressed in this study.

First situation: Minimizing the number of switched-on PMs by combining the VMs of different kind of users' jobs and placing them on the same PM. *Second*: Minimizing the CPU frequency as much as possible for the VMs of the DI jobs using the Dynamic Voltage Frequency Scaling (DVFS) technology. To note, minimizing frequency is a very crucial issue. Minimizing frequency is done only when the VM of the DI job works in space-shared policy to prevent any effect on the other VMs hosted on the same PM. In addition, minimizing frequency must not violate the DI job's Quality of Service (QoS).

This work assumes that: Upcoming jobs are organized as Bag-of-Tasks (BoT) where jobs are independent and they can be executed in any order and one VM is allocated to each job.

This work tackles the problem of high consumed power in cloud data centers by proposing the following key contributions:

- An online job classification model, which divides the workload into subsets. Each subset utilizes different kind of resources. This can potentially lead to more efficient resource provisioning and reduce the number of switched-on PMs which, in turn, results in higher energy efficiency
- A Near-optimal energy-efficient scheduling algorithm in heterogeneous virtualized cloud data centers

The rest of this paper is organized as follows. The next section presents the related work. Section 2 is a background to the concept of virtualization. Section 3 describes the components of the system model proposed to submit users' jobs to the cloud. It explains how these jobs are classified. In section 4, the details of the proposed model and the idea of the new proposed Mixed Type Placement (MTP) strategy are described. Experimental results are discussed and analyzed in section 5. Finally, conclusions listed in section 6.

Related Work

Many works tried to improve the power efficiency in data centers. The authors in (Horvath *et al.*, 2007; Wang and Lu, 2008; Wang *et al.*, 2009; Li *et al.*, 2011) used DVFS techniques to save energy. Some works like the one presented in (Lawson and Smirni, 2005) dynamically adjust the number of CPUs in a cluster to operate in ''sleep'' mode when the utilization is low, others like in (Lang and Patel, 2010; Heo *et al.*, 2011; Chakravarty and Sinha, 2013) switch the PMs from on to off. Bradley *et al.* (2003; Guenter *et al.*, 2011; Bobroff *et al.*, 2007), the authors statistically analyze the workload data and examine how to minimize the power consumption using workload history. The studies in (Garg *et al.*, 2011; Tesauro *et al.*, 2007; Zhang *et al.*, 2014a) optimized multiple aspects of data center behavior such as consumed power, service cost and the overall performance. The authors in (Chase *et al.*, 2001) proposed a cost approach to manage shared server resources such that the service requests for these resources are represented as a function taking into account the energy consumption. The authors in (Burge *et al.*, 2007) scheduled tasks to heterogeneous machines according to the energy costs of each one to maximize the profit. In (Deore and Patil, 2013), the research argued energy-efficient job scheduling and allocation scheme to minimize the number of switched-on PMs, so the amount of consumed power can be reduced. The thesis in (Feller, 2013) focused on the IaaS cloud service model whose goal is to offer a computing infrastructure by provisioning VMs on-demand. The thesis presented in (Beloglazov, 2013), presented some algorithms for distributed dynamic consolidation of VMs in virtualized cloud data centers. The VM placement strategy used in this thesis is based on bin packing algorithm.

Various optimization methods such as, Ant Colony Optimization (ACO) (Gao *et al.*, 2013), Particle Swarm Optimization (PSO) (Zhang *et al.*, 2014b) and Genetic Algorithms (GA) (Portaluri *et al.*, 2014; Dong *et al.*, 2014) were proposed to do the process of VM placement aiming to reduce energy consumption in data centers.

However, to the best of our knowledge, no work from the literature investigate the effects of combining different types of jobs on the same PM on energy efficiency.

Background

This section presents a background on the virtualization concept.

Virtualization

Virtualization is creating a virtual version of something (e.g., operating system, CPU, storage device, or network device). A single PM, which is the real hardware, can host one or more VM. A VM is a piece of software running on PM that emulates the properties of a separated PM.

Supplementary, the concept of virtualization breaks the traditional model of the PM that host a single Operating System (OS). It creates several VMs which are hosted on one PM, each VM may have its own OS. This concept is organized using hypervisor technology. A hypervisor (Josyula *et al*., 2011), also called Virtual Machine Manager (VMM), is a software that controls all PM resources, allocate resources needed by each operating system, monitors the utilization of the resources in turn and makes sure that the guest operating systems of the VMs cannot disrupt each other. Figure 1 illustrates the virtualization architecture.

VM Management

VM Management (Josyula *et al*., 2011) is the process of coordinated provisioning of the virtualized resources, as well as the runtime of such provisioning. This feature includes the mapping of the virtual resources to the physical ones and also overall management capabilities such as capacity, billing and Service Level Agreement (SLA) contract.

An important issue in VM management is the VM migration process, which is the process of transferring a VM from source to destination PM. Basically there are two kinds of VM migrations: Live and offline.

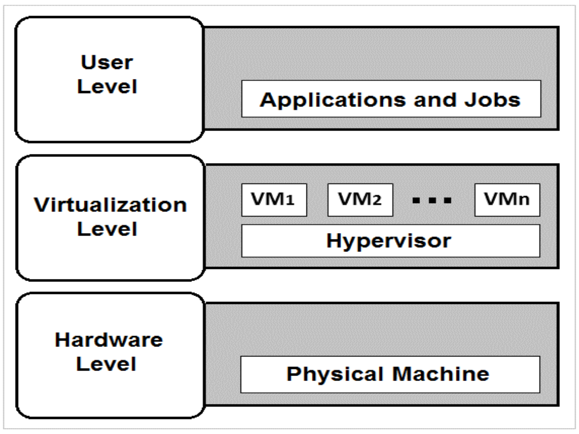

Fig. 1. Virtualization architecture

System Components

The proposed system is based on the Cloud computing environment paradigm, whereby Cloud users are able to request the services offered by the Cloud providers to execute their jobs. Indeed, this proposed system consists of two main components: A cloud user and a cloud provider, as shown in Fig. 2.

The cloud provider has an essential node called global scheduler. This node acts as an interface between users and the cloud infrastructure. It profiles and analyzes the service requirements of the submitted jobs and decides whether to accept or reject them based on the availability of resources. It selects the data center that execute the jobs such that the power consumption can be reduced, while the QoS requirements of the submitted jobs are met. As data centers are located in different geographical regions, they have different energy costs depending on regional constraints. Each data center is responsible for updating this information to the global scheduler in order to achieve an energy-efficient scheduling.

The two components, Cloud user and Cloud provider, are discussed in details below.

Cloud User

The cloud user submits the job (or set of jobs) to the cloud provider to be executed. Job can be defined as a task that is executed by the resources of the cloud data center, typically with an implied QoS requirement. A Job can be considered as a finer abstraction of an application service being hosted in the VM. This work proposes a job submission model as shown below:

$$job_i\,(QoS)$$

QoS includes the deadline and budget for the user jobs:

- DL_i: The deadline of job_i
- B_i: The maximum budget of job_i

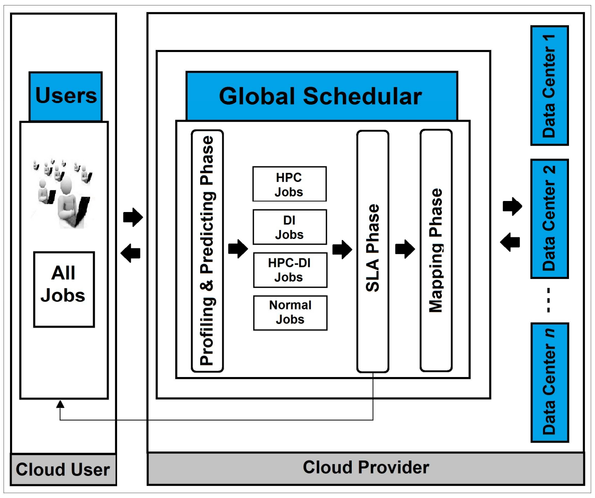

Fig. 2. System architectural components

Cloud users can submit their jobs (code and data) to be executed and manipulated in a specific cloud data centers. The job submission process is done via cloud provider.

Cloud Provider

Cloud provider component includes two models: global scheduler and data center (s). The function of this component is to receive the users' jobs, execute them on a specific data center and then send back the result of the jobs execution to their users.

Global Scheduler Model

The cloud provider has an essential node called the global scheduler. This node receives the users' jobs and profiles them. The profile process contains sufficient information about the jobs. This information is very useful in executing these jobs. Global scheduler model operates in three main phases: Prediction phase, SLA phase and mapping phase.

Profiling and Prediction Phase

This phase is in charge of profiling users' jobs and deciding their types before the VM allocation process. Four types of jobs result from this phase: High Performance Computing (HPC), Data Intensive (DI), High Performance Computing and Data Intensive (HPC-DI) and Normal.

Firstly, Job profiling results in the following infrastructure required to execute the jobs within their QoS requirements:

$$\left(peNumber_{user}, Time_{user}, Storage_{user}, RAM_{user}, BW_{user}\right)$$

Where:

$PeNember_{User}$	=	Number of Process Elements (PEs) needed by job_i. Also known as cores
$Time_{User}$	=	Required execution time for job_i when using $PeNember_{User}$
$Storage_{User}$	=	Storage size needed by job_i
RAM_{User}	=	RAM size needed by job_i
BW_{User}	=	Bandwidth needed by job_i

However, in reality, accurate job requirements such as execution time are not readily available in cloud environments. It is possible to examine and anticipate such requirements by executing part of the code. Many studies assumes that execution time with specific processing elements is a known quantity or estimated by the cloud user (Garg *et al*., 2011; Lee, 2012). To estimate the job execution time, four major solutions were proposed:

- Code analysis (Nudd *et al*., 2000; Reistad and Gifford, 1994)
- Analytic benchmarking/code profiling (Yang *et al*., 1993)
- Historical/Statistical prediction (Sanjay and Vadhiyar, 2008; Iverson *et al*., 1996)
- Empirical analysis (Berman *et al*., 2005)

Using these solutions, or by executing part of the code, other information related to user jobs can be obtained. In addition, there exists many studies which proposed models to estimate specific parameters (e.g., Cycle Per Instruction (CPI) (Chen and John, 2011; Intel Corporation, 2008), Memory Access Per Instruction (MPI) (Zhang and Chang, 2014) and estimated bandwidth (Zhu *et al*., 2012).

After profiling, this phase utilizes a mix of statistical information and some input features for the jobs to predict the job type. This is done in two stages:

User Id Checker (UIC): The users in cloud environments, most probably, submit the same type of jobs multiple times to be executed or served. Therefore, information of such users and their jobs can be collected by the provider they are dealing with and saved in log files. This work suggests creation of a Log File (LF) to help and support the decision of specifying the types of cloud users and their frequent jobs. So, the LF of the executed jobs contains statistical information about the jobs and their users. LF could be described as a list of records, one record for each job. The record is of the form:

$$Job\ (JobID,\ JobType,\ Date,\ UserID)$$

When users submit their jobs to the cloud provider, UIC checks if there is a match between the upcoming jobs and an existing record in LF. If there is a match, the type of the job will be predicted in this stage according to the history of the user who submits this job. This work supposes that even when there is a match, it is up to the provider to specify the job type depending on this match and to the user history, or by considering some features that are related to the new job submitted. This is because the user may change the requirements associated with the job at the subsequent submission, which may lead to a wrong type prediction. If there is no match for the job in this stage, the next one will perform the job type prediction.

Job Features Checker (JFC): If there is no available record in LF for the submitted job, the provider cannot depend on the job history when specifying the job type. Instead, the provider relies on some information associated with each job (which is obtained after profiling the job or given by the user) in the process of job type prediction. The features and or parameters associated with jobs and examined to help in job type prediction are: Total execution time of the job, Cycle Per Instruction (CPI), Memory Access Per Instruction (MPI), Size of the job and Bandwidth. JFC benefits from these values (which are listed in Table 1) to guess the type of the job.

Table 1. The features associated with each job which are used to predict the job type

Parameter	Abbreviation
Total execution time	ExT
Cycle per instruction	CPI
Memory access per instruction	MPI
Size of the job	Size
Bandwidth	BW

This work proposes four algorithms to predict the users' types considering the history of such users. The prediction depends on the type of the jobs submitted (either all jobs or last n jobs submitted by a specific user), the type of jobs submitted by the user within a past period of time, or depending on both. The algorithms are:

i. All Submitted Jobs (allSJ) Algorithm: This algorithm predicts the user type depending on the type of all the jobs submitted by this user.

Algorithm 1: Type of All Submitted Jobs (allSJ)

```
Algorithm: allSJ
Input:    LF
Output:   User type
1         for each (Job.UserID) in LF do
2             switch (LF.JobType)
3                 Case HPC: CounterHPC ++
4                 Case DI: CounterDI ++
5                 Case HPC-DI: CounterHPC-DI ++
6                 Case Normal: CounterNormal ++
7             end switch
8         return (Type of the max counter)
```

ii. Last n Submitted Jobs (lastnSJ) Algorithm: This algorithm predicts the user type depending on the type of the last n jobs submitted by this user.

Algorithm 2: Type of Last n Submitted Jobs (lastnSJ)

```
Algorithm: lastnJS
Input:    LF, nJOB
Output:   User type
1     JobCounter = 0;
2     Start checking with the last job submitted by
      a specific user
3     while (JobCounter < nJOB) or (LF has more
      jobs)
4             switch (Job.JobType)
5                 Case HPC: CounterHPC ++
6                 Case DI: CounterDI ++
7                 Case HPC-DI: CounterHPC-DI ++
8                 Case Normal: CounterNormal ++
9             end switch
10            JobCounter ++
11            Go to the previous job submitted by
              the same user
12    end while
13    return (Type of the max counter)
```

iii. Submitted Jobs within a Period of Time (SJPT) Algorithm: This algorithm predicts the user type depending on the type of jobs submitted by the user within last n days.

Algorithm 3: Submitted Jobs within a Period of Time (SJPT)

```
Algorithm: SJPT
Input:    LF, mDAY
Output:   User type
1         StartDate = Current Date
2         EndDate = StartDate – mDAY
3         for each job belongs to the same user in LF do
4             If (Job.Date <= StartDate) and
              (Job.Date >= EndDate)
5                 switch (Job.JobType)
6                     Case HPC: CounterHPC ++
7                     Case DI: CounterDI ++
8                     Case HPC-DI: CounterHPC-DI
                      ++
9                     Case Normal: CounterNormal
                      ++
10                end switch
11            end if
12        return (Type of the max counter)
```

iv. Last n Submitted Jobs within a Period of Time (lastnSJPT) Algorithm: This algorithm predicts the user type based on both the type of the last n jobs submitted and the type of jobs submitted by the user within a past period of time. It is a hybrid from the lastnJS and the SJPT algorithms.

Algorithm 4: Last n Submitted Jobs within Period of Time

```
Algorithm: lastnSJPT
Input:    LF, nJOB, mDAY
Output:   User type
1         JobCounter = 0;
2         StartDate = Current Date
3         EndDate = StartDate – mDAY
4         Start checking with the last job submitted by
          a specific user while (JobCounter < nJOB) or
          (LF has more jobs)
6             If (Job.Date <= StartDate and Job.Date
              >= EndDate)
7                 switch (Job.JobType)
8                     Case HPC: CounterHPC ++
9                     Case DI: CounterDI ++
10                    Case HPC-DI: CounterHPC-DI
                      ++
11                    Case Normal: CounterNormal
                      ++
12                end switch
13                JobCounter ++
14                Go to the previous job submitted by
                  the same user
15            end if
```

```
16    end while
17    return (Type of the max counter)
```

To make accurate prediction of the user type, the cloud provider must perform a statistical analysis to choose the values of nJOB (which represents n jobs submitted by a specific user) and mDAY (which represents m days of jobs' submission by a specific user) to be used in the four algorithms illustrated above. These values may vary over time depending on the users who are dealing with the provider and how they deal.

In case of the absence of user's history, the provider can predict the user type based on the requirements associated with the jobs. Equations 1 to 4 are proposed to calculate the total requirements for all jobs of a user as below:

$$\mathrm{Re}qTime_{user} = \sum_{i=1}^{NoOfJobs} Time_i \tag{1}$$

$$ReqMemory_{user} = \sum_{i=1}^{NoOfJobs} RAM_i \tag{2}$$

$$ReqStorage_{user} = \sum_{i=1}^{NoOfJobs} Storage_i \tag{3}$$

$$ReqBW_{user} = \sum_{i=1}^{NoOfJobs} BW_i \tag{4}$$

Equations 1 to 4 calculate the total values requested for time execution, memory, storage and bandwidth respectively, which are required to execute jobs of a specific user. The resulting values specify the total requirements for every user.

By comparing these requirements with previous requirements (which stored and considered as benchmarks by provider), the user type can be predicted.

In job type prediction process, this work assigns weights to the proposed checkers in the prediction phase. Equations 5 to 8 are proposed to be used in this study:

$$\begin{aligned} HPC &= \left(LF_{HPC} * w_1\right) + \\ &\left(\left(\left(job_i^{ExT} \geq ExecTh\right) or \left(job_i^{CPI} \geq CpiTh\right) or\right.\right. \\ &\left.\left.\left(job_i^{MPI} \geq MpiTh\right)\right) * w_2\right) + \\ &\left(\left(\left(job_i^{Size} < SizeTh\right) or \left(job_i^{BW} < BwTh\right)\right) * w_3\right) \end{aligned} \tag{5}$$

$$\begin{aligned} DI &= \left(LF_{DI} * w_1\right) + \\ &\left(\left(\left(job_i^{ExT} < ExecTh\right) or \left(job_i^{CPI} < CpiTh\right) or\right.\right. \\ &\left.\left.\left(job_i^{MPI} \geq MpiTh\right)\right) * w_2\right) + \\ &\left(\left(\left(job_i^{Size} \geq SizeTh\right) or \left(job_i^{BW} \geq BwTh\right)\right) * w_3\right) \end{aligned} \tag{6}$$

$$\begin{aligned} HPC\text{-}DI &= \left(LF_{BOTH} * w_1\right) + \\ &\left(\left(\left(job_i^{ExT} \geq ExecTh\right) or \left(job_i^{CPI} \geq CpiTh\right) or\right.\right. \\ &\left.\left.\left(job_i^{MPI} \geq MpiTh\right)\right) * w_2\right) + \\ &\left(\left(\left(job_i^{Size} \geq SizeTh\right) or \left(job_i^{BW} \geq BwTh\right)\right) * w_3\right) \end{aligned} \tag{7}$$

$$\begin{aligned} NORMAL &= \left(LF_{NORMAL} * w_1\right) + \\ &\left(\left(\left(job_i^{ExT} < ExecTh\right) or \left(job_i^{CPI} < CpiTh\right) or\right.\right. \\ &\left.\left.\left(job_i^{MPI} < MpiTh\right)\right) * w_2\right) + \\ &\left(\left(\left(job_i^{Size} < SizeTh\right) or \left(job_i^{BW} < BwTh\right)\right) * w_3\right) \end{aligned} \tag{8}$$

Where:
w1, w2 and w3 are weight values, such that:
w1+w2+w3 = 1

This work suggests that it is up to the cloud provider to assign the weight values. w1 is assigned to the history of the user who submits the job (if any), w2 is assigned to the values related to HPC jobs, which represents the total execution time, CPI and MPI. Usually, these values are high in HPC jobs. Finally, w3is assigned to the values related to DI jobs, which are job size and bandwidth. In general, these values are high in DI jobs.

And:

- $LF_{HPC} = \begin{cases} 1 \; \textit{if the history of the user in LF is HPC} \\ 0 \;\; \textit{otherwise} \end{cases}$
- $LF_{DI} = \begin{cases} 1 \; \textit{if the history of the user in LF is DI} \\ 0 \;\; \textit{otherwise} \end{cases}$
- $LF_{BOTH} = \begin{cases} 1 \; \textit{if the history of the user in LF is HPC - DI} \\ 0 \;\; \textit{otherwise} \end{cases}$
- $LF_{NORMAL} = \begin{cases} 1 \text{ if } LF_{HPC} = LF_{DI} = 0 \\ 0 \;\; \textit{otherwise} \end{cases}$
- *ExecTh:* Thershold for job execution time
- *CpiTh:* Threshold for CPI of the job
- *MpiTh:* Threshokd for MPI of the job
- *SizeTh:* Threshold for job size
- *BwTh:* Threshold for the required bandwidth for the job

These thresholds are chosen by examining real workload trace. A histogram is sketched for each one of the above five parameters to select the proper thresholds. Based on these histograms, two methods are used: The average and median. Figure 3 is an example of the histogram sketched for selecting CPI threshold for 100 jobs.

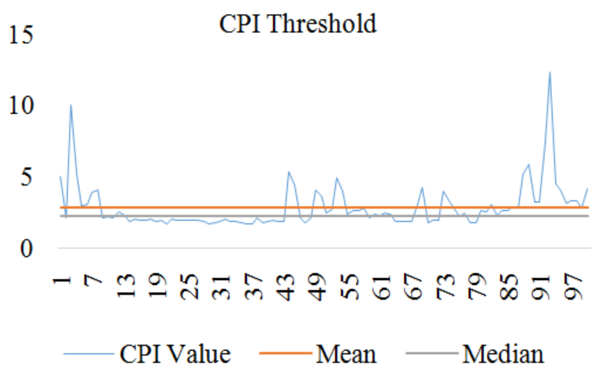

Fig. 3. Selecting CPI threshold for set of jobs

The job type is selected according to the maximum value from Equation 1 to 4. If two or more values are equal (ambiguity state), the type in the LF is the dominant. In case of no information about the user in the LF, then there is no weight to the history of this job. The weight of the user's history is considered zero and Equation 5 to 8 rewritten as in the Equation 9 to 12:

$$HPC = \Big(\Big(\big(job_i^{ExT} \geq ExecTh\big) or \big(job_i^{CPI} \geq CpiTh\big) or \big(job_i^{MPI} \geq MpiTh\big)\Big)*w_2\Big) + \Big(\Big(\big(job_i^{Size} < SizeTh\big) or \big(job_i^{BW} < BwTh\big)\Big)*w_3\Big) \quad (9)$$

$$DI = \Big(\Big(\big(job_i^{ExT} \geq ExecTh\big) or \big(job_i^{CPI} < CpiTh\big) or \big(job_i^{MPI} < MpiTh\big)\Big)*w_2\Big) + \Big(\Big(\big(job_i^{Size} \geq SizeTh\big) or \big(job_i^{BW} \geq BwTh\big)\Big)*w_3\Big) \quad (10)$$

$$HPC - DI = \Big(\Big(\big(job_i^{ExT} \geq ExecTh\big) or \big(job_i^{CPI} \geq CpiTh\big) or \big(job_i^{MPI} \geq MpiTh\big)\Big)*w_2\Big) + \Big(\Big(\big(job_i^{Size} \geq SizeTh\big) or \big(job_i^{BW} \geq BwTh\big)\Big)*w_3\Big) \quad (11)$$

$$NORMAL = \Big(\Big(\big(job_i^{ExT} < ExecTh\big) or \big(job_i^{CPI} < CpiTh\big) or \big(job_i^{MPI} < MpiTh\big)\Big)*w_2\Big) + \Big(\Big(\big(job_i^{Size} < SizeTh\big) or \big(job_i^{BW} < BwTh\big)\Big)*w_3\Big) \quad (12)$$

If two or more values are equal and there is history for the user that submit the job, the requirement associated with the job can help in job type prediction (as in Equation 1 to 4).

The SLA Phase

In this phase, an agreement about the offered services between the user and the cloud provider is achieved. SLA is a legal contract between participants (user and the cloud provider) to ensure that the QoS requirements of the users are met. If any party violates the SLA terms, the defaulter has to pay a penalty or do legal actions according to the clauses defined in this SLA (Wu and Buyya, 2011; Moustafa, 2015). In this study, the SLA phase decides whether the provider can execute the user's job or not depending on the job requirements and the available resources. If the provider is not able to execute the job with the required QoS (deadline and budget), user will be informed in this phase. Otherwise, the job will pass to the mapping phase. The form for the SLA contract between the user and the provider is presented in Table 2.

Mapping Phase

After specifying the types of jobs for all users and the SLA between the users and the providers is confirmed, this phase is concerned with selecting the best data center to serve the jobs (mapping jobs to a data center). The selected data center is the one that consumes the minimum amount of power when executing the users' jobs. The global scheduler interacts with the local schedulers of each data center to execute the jobs submitted to it. The CPU availability at particular times in the future and all information about the free time slots are available to the global scheduler based on the information provided by the local schedulers. The global scheduler receives the detailed information periodically from each cloud data center. These information includes the available resources and some other values like the amount of the consumed power resulted when executing the jobs on the available resources of the data center.

In the mapping phase, the provider maps the jobs to a specific cloud site based on the information provided by the local schedulers. The mapping process must consider the SLA constraints. At the selected cloud site, the local scheduler will perform the jobs scheduling based on the proposed MTP strategy.

The Data Center Model

In addition to the global scheduler. The cloud provider has a number of data centers. Each data center consists of *n* heterogeneous PM_s refer to Fig. 4. Each PM_i is equipped with multicore processors and characterized by the configuration shown below:

$$PM_i\left(PE_{PM}, Speed_{MP}, Storage_{PM}, RAM_{PM}, BW_{PM}\right)$$

Where:

PE_{PM}	=	Number of PEs in PM_i
$Speed_{PM}$	=	Speed of each PE in PM_i
$Storage_{PM}$	=	Size of the storage in PM_i
RAM_{PM}	=	RAM size in PM_i
BW_{PM}	=	Bandwidth in PM_i

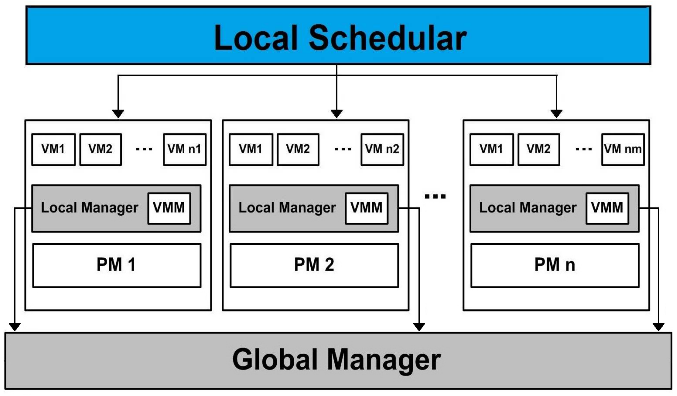

Fig. 4. The data center model

Table 2. The main items in a SLA contract

Item	Details
Job execution time	Specify the estimation of the total execution time.
Job execution fees	Specify the total fees the user should pay to the provider as a result for executing the user's job.
Violation	Typically a reduction in the fees that the user has to pay to the provider, plus some additional compensation and a corrective action plan. Usually, such fees reduction and compensation vary depending on the Number of Violations (NoV).

Each *PM* consists of one or more *VMs*. The *VM* configuration is shown below:

$$VM_i\left(peNumber_{VM}, Speed_{VP}, Storage_{VM}, RAM_{VM}, BW_{VM}\right)$$

Where:

$peNumber_{VM}$	=	Number of PEs in VM_i
$Speed_{VM}$	=	Speed of each PE in VM_i
$Storage_{VM}$	=	Size of the storage in VM_i
RAM_{VM}	=	RAM size in VM_i
BW_{VM}	=	Bandwidth in VM_i

Based on the model presented in (Beloglazov, 2013), each data center has a local scheduler, which performs the scheduling process for all jobs received by the data center. In addition, the data center has a local manager (resides on each *PM*) and a global manager (maintains the overall system's resource utilization of a set of *PMs* by interacting with their local managers).

The Proposed Model

In cloud computing service models, different users submit their jobs to the cloud provider to be executed by some heterogeneous *VMs*. Each job will be mapped to a *VM* which satisfies its requirements. The mapping process requires an efficient strategy for allocating resources (heterogeneous CPU, storage, memory and network bandwidth) needed to execute a huge number of jobs. The resource allocation problem is NP-hard (Zhang *et al*., 2014a; Duan *et al*., 2007).

The problem even becomes more challenging when trying to effectively schedule many jobs in distributed, heterogeneous and virtualized cloud systems. So, this work proposes a model presented in the next sections to solve this problem.

Model Representation

The proposed system, *S*, can be modeled as four-tuple (*D*, *PM*, *VM*, *J*), such that:

- *D* is a set of data centers, each element $D_d \in D$ represents a single data center in the system
- *PM* is a set of physical machines in the data center; $PM_{pm,d} \in PM$ each element represents a single PM_{pm} in D_d
- *VM* is a set of virtual machines associated with physical machines in the data center; each element $VM_{vm,pmd} \in VM$ represents a single VM_{vm} on single in data center D_d
- *J* is a set of jobs; each element $job_j \in J$ represents a single job

Multi Objective Function

This paper proposes a multi-objective optimization scheduling algorithm, which aims to minimize the energy consumption, while maintaining the SLA specifications.

The power consumed by PMs in data centers usually determined by the CPU, disk storage, memory, network interfaces (Beloglazov *et al*., 2012). Among these components, the CPU consumes the most amount of energy. So, in this study, only the energy consumed by the CPU is considered. However, the consumption of the other components becomes very costly as a result of establishing numerous data centers round the world. Each data center contains thousands of PMs composed of these components. Equation 13 is used to find the total power of each *PM*:

$$PM_i^P = P_{CPU} + P_{Memory} + P_{Storage} \quad (13)$$

Where:

P_{CPU}	=	The power consumed by the CPU
P_{Memory}	=	The power consumed by the memory
$P_{Storage}$	=	The power consumed by the storage disk

The power consumption model of the CPU, which is generally composed of CMOS circuits, is the sum of both the CPU static power ($P_{CPU\text{-}static}$) and the CPU dynamic power ($P_{CPU\_dynamic}$). Equation (14) is used to compute the power consumed by the CPU (Brooks *et al*., 2000; Elnozahy *et al*., 2003; Chaudhry *et al*., 2015):

$$P_{CPU} = P_{CPU-Dynamic} + P_{CPU-Static} \quad (14)$$

where, ($P_{CPU\text{-}Static}$) is a constan *t*, say ω and (*PCPU-Dynamic*) as in (15):

$$P_{CPU-Dynamic} = ACV^2 f \quad (15)$$

Where:

A	=	An activity factor that accounts how frequency gates switch
C	=	The total capacitance at the gate outputs
V	=	The voltage of the CPU and f is the operating frequency

Voltage V can be expressed as a linear function in frequency, *V* = *af*, such that *a* is constant. All constants (ω, *A* and *C*) can be combined together in a constant, say*β*. So, Equation 14 that compute power consumption model of CPU can be written in (16):

$$P_{CPU} = \beta f^3 \quad (16)$$

Frequency *f* is the only variable value in (16). Thus, DVFS technique is considered in this study, which usually results in a linear power-to-frequency relationship for a data center. DVFS modulates a processor's clock frequency and supply voltage in lockstep as programs execute. The premise is that a processor's workloads vary according to job requirements. When the processor has less work, it can be slowed down without affecting performance adversely. The architecture of the CPU supports different frequencies, so when the type of job is DI, this work minimizes the CPU frequency to a minimum level.

The PMs on/off switching technique is also considered in this study. In general, idle PM consumes approximately 70% of the power consumed when this PM running at the full CPU speed (Beloglazov, 2013). This fact justifies the technique of switching idle servers off to reduce the total power consumption.

Let n represents the total number of jobs, the total power consumption for executing these jobs (T_{energy}) can be measured using Equation 17:

$$T_{energy} = \sum\nolimits_{i=1}^{n} x_i * PM_i^p \quad (17)$$

Where:

x_i	=	Equal to 0 if the machine
PM_i	=	Off and equal to 1 if it is on.

So, the main objective of the proposed system is to minimize the value of T_{energy}.

In addition to the total consumed energy, an important value to be measured is the total execution time of the set of all jobs (T_{time}). The execution time for any job is the sum of its data stage in and stage out, adding to the actual time needed to execute the job.

The execution time $ExT_i^{vm,pm,d}$ can be calculated using Equation 18:

$$ExT_i^{vm,pm,d} = \left(\frac{InputSize_i}{BW_{vm}}\right) + \left(\frac{job_i^l}{f_i}\right) + \left(\frac{OutputSize_i}{BW_{vm}}\right) \quad (18)$$

Where:

job_i^l	=	Length of job_i in term of millions of instructions
$InputSize_i$	=	Input size of job_i
$OutputSize_i$	=	Output size of job_i
f_i	=	The frequency of the core when executing job_i

So, the time needed to execute all jobs is calculated using (19):

$$Ttime = \sum\nolimits_{i=1}^{n} ExT^{vm,pm,d} \quad (19)$$

In fact, T_{time} is not the total time of executing all jobs because jobs are executing in parallel. The maximum finish time among all jobs in any workload (makespan) is the actual finish time of jobs' execution. It is an important value to be measured. This value reflects the accuracy of the jobs to VMs allocation process from the execution time point of view. Usually, minimizing the makespan value leads to minimizing T_{time}. Makespan can be measured as in (20):

$$Makespan = max\left\{FT_i^{vm,pm,d} \mid \forall i \in J\right\} \quad (20)$$

where, $FT_i^{vm,pm,d}$ is the finish time when executing job_i by VM_{vm}.

Another important value to be calculated is the total cost for executing the set of jobs (T_{cost}), which can be measured using (21):

$$T_{cost} = \sum\nolimits_{i=1}^{n} C_i^{vm,pm,d} * ExT_i^{vm,pm,d} \quad (21)$$

Where:

$C_i^{vm,pm,d}$ = The cost of executing job_i per unit time using the resources of VM_v hosted in PM_p in D_d

The objectives of the proposed system are to minimize the values of consumed power, time and cost which is expressed in (17), (20) and (21) respectively.

The new allocation strategy (suggested in this study) to allocate jobs to *VMs* is discussed in the next section.

VM Allocation and VM Placement Strategy

In cloud computing environments, cloud users are served by providing *VMs* to execute their jobs. Cloud providers provide such services.

Cloud services can be delivered in the following sequence:

- A cloud user submits the job (or set of jobs) to a cloud provider
- A cloud provider analyzes the submitted jobs and predicts their types and requirements
- According to the job requirements, the provider performs the VM provisioning to user's job (i.e., VM allocation) using best fit algorithm. Two main policies for the VMs to jobs allocation in cloud computing environments (Calheiros *et al*., 2011) can be used
- Space-shared policy: The result is a *VM* with one or more cores
- Time-shared policy: The result is a core that holds two or more *VMs*

Every job is allocated to a VM with specific frequency. Since the CPU architecture supports different frequencies for the same CPU, it is possible to scale the frequency of the CPU cores up or down using DVFS technology. In the proposed work, when DI jobs are allocated to VMs in a space shared policy, the core frequency is minimized to a minimum frequency level if this will not affect the job's QoS requirements. This leads to better energy efficiency due to the cubic relation existing between energy and frequency as illustrated in (16). Usually, DI jobs do not need to utilize the maximum frequency of the compute power resources. Instead, they utilize storage and bandwidth resources.

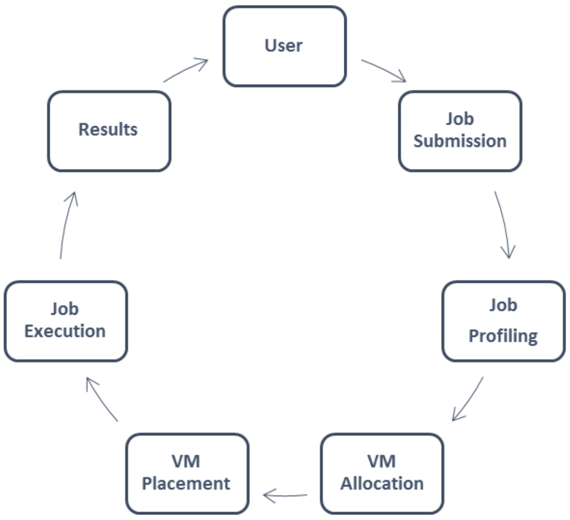

Fig. 5. User's job life cycle

After VM provisioning, the provider performs the VM placement, which is the process of placing the VM on the proper PM. The VM placement process is implemented using the MCKP model (Pisinger, 1995). Knapsacks in this case represent the PMs of the data center. The aim is hosting the requested VMs on the minimum possible number of knapsacks. To note, the Job type is the essential factor in this placement. In this study, two VMs which belong to the same type of jobs are not placed on the same PM. This leads to further minimizing the number of switched-on PMs which is guaranteed by KP. However, when there is no available PM for combining VMs of different types of jobs, the classical KP algorithm is used to place the VMs.

The above sequence represents the job life cycle as illustrated in Fig. 5.

Performance Analysis

This section presents the evaluation of the proposed model. The CloudSim simulator, which is an extensible simulation toolkit that enables modeling and simulation of cloud computing systems and application provisioning environments, is used. The CloudSim supports both system and behavior modeling of cloud system components such as data centers, PMs, VMs and resource provisioning policies (Calheiros *et al*., 2011). A real workload trace to simulate the cloud computing environment is used. The jobs information is based on real data provided by (Google, 2015). The Google workload traces are collected from large cloud systems (about 12,500 compute nodes over 29 days). The traces consist of different types of jobs. Real workload traces can provide a very high level of realism when used directly in performance evaluation experiments. More details about this data are available in (Reiss and Wilkes, 2013).

Table 3. VM instance types in M3 family offered by Amazon

VM type	CPU	Clock	vCPU	Memory	BW
M3.meduim	Intel Xeon E5-2670 v2 processors	2500	1	3750	Moderate
M3.large	Intel Xeon E5-2670 v2 processors	2500	2	7500	Moderate
M3.xlarge	Intel Xeon E5-2670 v2 processors	2500	3	15000	High
M3.2xlarge	Intel Xeon E5-2670 v2 processors	2500	4	30000	High

Three parameters are applied in the experiments: Number of PMs, number of VMs, number of jobs in the workload. The PMs and VMs configurations are as those provided by Amazon cloud data centers (Amazon, 2015). Table 3 illustrates the VMs instance types, called M3, offered by Amazon and used in the experiments in this study.

Results

Extensive experiments are carried out and repeated ten times for different number of jobs. Then, the results are compared with some strategies from the literature, e.g., Round Robin (RR) and Genetic Algorithm (GA). The results show that the proposed strategy outperforms both RR and GA as detailed in the next sections.

Energy Consumption

The evaluation in this section is based on the total consumed energy as a measurement. The experiments are done and repeated 10 times for different numbers of PMs, VMs and jobs.

In this study, the sets of jobs which are tested consist of 50, 200, 400, 600, 800 and 1000 jobs. To evaluate the proposed MTP strategy, a comparison with many classical job scheduling and placement strategies (e.g., Round Robin (RR) and Genetic Algorithm (GA)) are done. GA is a general purpose optimization technique inspired by the biological evolution. The initial population is produced randomly in the experiment. Then, it evolves to better approximate solutions from generation to generation iteratively (The number of iterations is set to 100) based on a specific fitness function (the consumed power in the experiment). The individual VMs cross by the genetic operators and combine in PMs. Each new population represents a new solution of VMs to PMs placement. Similar work can be found in (Dong *et al*., 2014).

In RR, VMs are placed and distributed to PMs in the data center sequentially in a circular manner, like the one in Eucalyptus, which is an open source software for building private and hybrid clouds (Eucalyptus, 2015).

The comparison of the proposed algorithm with the GA and RR approaches in terms of the total consumed energy resulting from executing different number of jobs is presented in Fig. 6.

The energy consumption of each set of jobs increased as the number of jobs increased. This is because the total execution time of a job will increase as more jobs are submitted. Within the same set of jobs and from energy efficiency perspective, MTP outperforms both GA and RR as illustrated in Fig. 6. MTP is better in enhancing energy efficiency because it involves minimum number of PMs in executing the set of jobs (as shown in Fig. 10). Minimizing the number of involved PMs is the main factor in enhancing the energy efficiency in cloud data centers, because even the completely idle PMs consume about 70% of their peak power. Another reason is that the resources of the involved PMs using MTP are better utilized than when using RR and GA (as shown in Fig. 7).

Total Execution Time

The total time required to execute different numbers of jobs using RR, GA and MTP is listed in Fig. 8.

In most scenarios, the total time required to execute users' jobs using MTP is more than the time of using RR and GA. This is because the VM to PM placement process in MTP spends time searching the best PM to place VM on it. The best PM is the PM that is already switched on and also hosts a VM allocated for a job of different type from the job served by the VM to be placed on it. Forcing jobs to be executed in minimum number of PMs will increase the turnaround time of these jobs. Consequently, the jobs' total execution time (makespan) will increase.

In GA, VMs are crossed and combined on PMs according to genetic operators. So, no time to spend in searching for specific PMs before the process of VM placement as in MTP. In RR, VMs are placed on PMs sequentially in a circular manner. No time to spend neither in cross and combine operations as in GA, nor in searching for specific PMs before the process of VM placement as in MTP. So, the total time for jobs execution is better in RR.

Cost

The experiments related to cost depend on real cost values offered by Amazon (2015). The cost per hour values are illustrated in Table 4.

As illustrated in Fig. 9, the tests show that the proposed MTP outperforms RR and GA from the cost perspective. The reason is that MTP utilizes the minimum resources that meet the QoS requirements and execute users' jobs.

MTP do not involve any VM with resources that exceed the actual jobs' requirements which, in turn, avoids users from paying any extra costs.

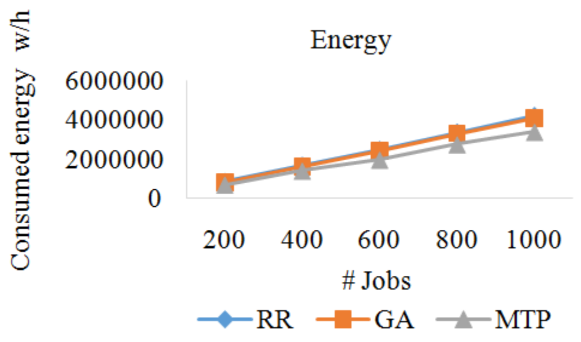

Fig. 6. Consumed power when executing different number of jobs

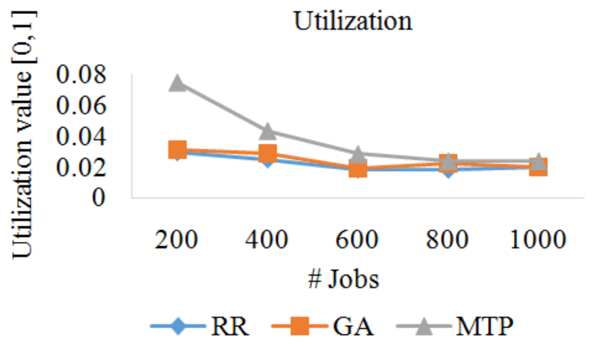

Fig. 7. Total utilization when executing different number of jobs

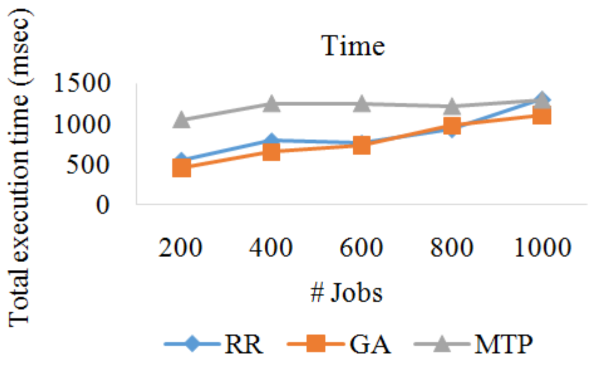

Fig. 8. Total execution time resulting from executing different number of jobs

Impact of Using MTP

The proposed MTP has some impacts which are detailed in the next sections.

Impact on the Number of Active PMs

The proposed MTP strategy results in the minimum possible number of active PMs that are able to execute the total jobs compared to RR and GA. This outcome is due to the combination of high compute intensive jobs with data intensive jobs in the same PM, as the high compute intensive job mostly relies on the CPU performance, whereas the data intensive job utilizes disk storage.

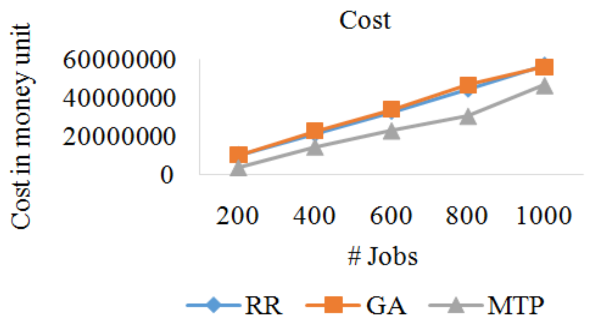

Fig. 9. Costs of executing different number of jobs

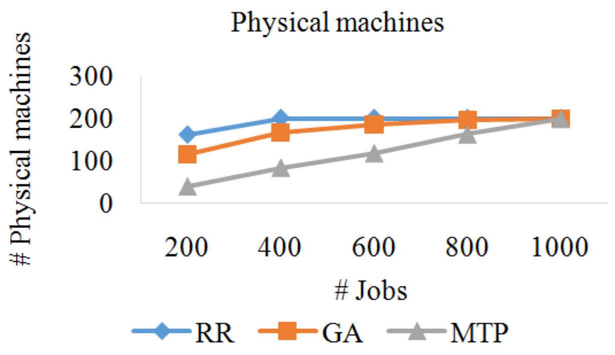

Fig. 10. Number of PMs needed when executing different sets of jobs using MTP, GA and RR

Table 4. The cost prices offered by Amazon for VM types in M3 family

VM type	Cost/h Linux
M3.meduim	$ 0.07
M3.large	$ 0.14
M3.xlarge	$ 0.28
M3.2xlarge	$ 0.56

The MTP strategy forces the scheduler to combine the VMs allocated to different types of jobs on the same PM. There is no exception unless if there is an SLA violation, or there is no available resources in an active PM to serve the VM.

Figure 10 illustrates the differences in the number of active PMs using RR, GA and MTP for different number of jobs. The reason is that the compute and storage resources of each PM is utilized in an optimal way. Such utilization prevent the idle state for the resources (which is not guaranteed in GA and RR). In turn, the total number of switches on PMs needed to execute each set of jobs is minimized.

Impact with DVFS on the Type of Job

Since the CPU architecture supports different frequencies, the proposed algorithm minimizes the CPU frequency to a minimum level unconditionally when the type of job is data intensive.

Table 5. The resulted energy consumption when executing different number of jobs using MTP strategy with and without frequency scaling

No of jobs	MTP without DVFS Energy W/h	MTP with DVFS Energy W/h
200	820205	783161
400	1622788	1468543
600	2218623	2174261
800	3291324	2976292
1000	4080028	3927554

This does not affect the system performance and at the same time reduces the power consumption by an acceptable percentage. The percentage value differ from workload to another depending on the number of data intensive jobs it contains. Table 5 shows the reduction in power consumption when applying the frequency scale down to MTP strategy in executing different number of jobs.

Conclusion

In this study, a model that identifies common patterns for the jobs submitted to the cloud is presented. This model predicts the type of the job submitted; and accordingly, the set of users' jobs is classified into four subsets. Each subset contains jobs that have similar requirements. In addition to the jobs' common pattern and requirements, the users' history is considered in the jobs' type prediction model. The goal of job classification is to find a way to propose useful strategy that helps to improve power efficiency. Following the process of jobs' classification, a new VM placement strategy, called Mixed Types Placement (MTP), is proposed. Its core idea is to place the VMs of the jobs of different types in the same PM. The evaluation of MTP shows promising results in reducing the total energy consumed in the cloud data centers. The reduction in the consumed energy using MTP results in minimizing the number of PMs involved in executing users' jobs and the resources of the involved PMs are utilized optimally.

As a future work, we are working to subdivide the HPC and DI subsets based on new features and rely on the new sub-subsets in the process of VM placement. The HPC subset is divided into two main categories: CPU intensive and Memory intensive. Similarly, the DI subset is divided into two main categories: Hard disk intensive and Bandwidth intensive. Moreover, the VM management approaches, such as VM migration and consolidation, will be applied to the proposed model to make it more integral to work in cloud computing paradigm.

Acknowledgement

The authors wish to thank the anonymous reviewers for offering useful comments about this manuscript.

Funding Information

No funding information concerning to the work presented in this manuscript.

Author's Contributions

Auday Al-Dulaimy: Proposed the main idea of the article, explained the methodology, analyzed the results, and drafted the manuscript.

Ahmed Zekri, Wassim Itani and Rached Zantout: Discussed, provided notes and revised the manuscript. All authors read and approved the final manuscript.

Ethics

There are no ethical issues involved in publishing this manuscript.

References

Amazon, 2015. http://aws.amazon.com

Beloglazov, A., 2013. Energy-efficient management of virtual machines in data centers for cloud computing. PhD Thesis, Department of Computing and Information Systems, The University of Melbourne.

Beloglazov, A., J. Abawajyb and R. Buyya, 2012. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. Future Generat. Comput. Syst., 28: 755-768. DOI: 10.1016/j.future.2011.04.017

Berman, F., H. Casanova, A. Chien, K. Cooper and H. Dail *et al.*, 2005. New grid scheduling and rescheduling methods in the grads project. Int. J. Parallel Programm., 33: 209-229. DOI: 10.1007/s10766-005-3584-4

Bobroff, N., A. Kochut and K. Beaty, 2007. Dynamic placement of virtual machines for managing SLA violations. Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management, May 21-25, IEEE Xplore Press, Munich, pp: 119-128. DOI: 10.1109/INM.2007.374776

Bradley, D., R. Harper and S. Hunter, 2003. Workload-based power management for Parallel computer systems. IBM J. Res. Develop., 47: 703-718. DOI: 10.1147/rd.475.0703

Brooks, D., V. Tiwari and M. Martonosi, 2000. Wattch: A framework for architectural-level power analysis and optimizations. Proceedings of the 27th Annual International Symposium on Computer Architecture, Jun. 10-14, IEEE Xplore Press, Vancouver, BC, Canada, pp: 83-94. DOI: 10.1145/339647.339657

Burge, J., P. Ranganathan and J.L. Wiener, 2007. Cost-aware scheduling for heterogeneous enterprise machines. Proceedings of the International Conference on Cluster Computing, Sept. 17-20, IEEE Xplore Press, Austin, TX, pp: 481-487. DOI: 10.1109/CLUSTR.2007.4629273

Calheiros, R.N., R. Ranjan, A. Beloglazov, C.A. Rose and R. Buyya, 2011. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Pract, Experience J., 41: 23-50. DOI: 10.1002/spe.995

Chakravarty, P. and R. Sinha, 2013. Power aware approach of live migration for resource management in cloud data centre. Int. J. Comput. Sci. Manage. Res., 2: 1402-1407.

Reiss, C. and J. Charles, 2013. Google cluster-usage traces: Format and schema. Google Inc.

Chase, J.S., D.C. Anderson, P.N. Thakar, A.M. Vahdat and R.P. Doyle, 2001. Managing energy and server resources in hosting centers. Proceedings of the 18th ACM Symposium on Operating Systems Principles, Oct. 21-24, Banff, Canada, pp: 103-116. DOI: 10.1145/502034.502045

Chaudhry, M.T., T.C. Ling, A. Manzoor, S.A. Hussain and J. Kim, 2015. Thermal-aware scheduling in green data centers. ACM Comput. Surveys. DOI: 10.1145/2678278

Chen, J. and L.K. John, 2011. Predictive coordination of multiple on-chip resources for chip multiprocessors. Proceedings of the International Conference on Supercomputing, May 31-Jun. 04, Tucson, AZ, USA, pp: 192-201. DOI: 10.1145/1995896.1995927

Deore, S.S. and A.N. Patil, 2013. Energy-efficient job scheduling and allocation scheme for virtual machines in private clouds. Int. J. Applied Inform. Syst., 5: 56-60. DOI: 10.5120/ijais12-450842

Dong, Y.S., G.C. Xu and X.D. Fu, 2014. A distributed parallel genetic algorithm of placement strategy for virtual machines deployment on cloud platform. Sci. World J. DOI: 10.1155/2014/259139

Duan, R., R. Prodan and T. Fahringer, 2007. Performance and cost optimization for multiple large-scale grid workflow applications. Proceedings of the IEEE/ACM International Conference on Super Computing, Nov. 10-16, Reno, NV, USA, pp: 1-12. DOI: 10.1145/1362622.1362639

Elnozahy, M., M. Kistler and R. Rajamony, 2003. Energy-efficient server clusters. Proceedings of the 2nd International Conference on Power-Aware Computer Systems, (PCS' 03), Cambridge, MA, USA, pp: 179-197. DOI: 10.1007/3-540-36612-1_12

Eucalyptus, 2015. https://www.eucalyptus.com/

Feller, E., 2013. Autonomic and energy-efficient management of large-scale virtualized data centers. PhD Thesis, University of Rennes.

Gao, Y., H. Guan, Z. Qi, Y. Hou and L. Liu, 2013. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. J. Comput. Syst. Sci., 79: 1230-1242. DOI: 10.1016/j.jcss.2013.02.004

Garg, S.K., C.S. Yeob, A. Anandasivamc and R. Buyya, 2011. Environment-conscious scheduling of HPC applications on distributed Cloud-oriented data centers. J. Parallel Distrib. Comput., 71: 732-749. DOI: 10.1016/j.jpdc.2010.04.004

Google, 2015. https://cloud.google.com/storage/docs/overview

Guenter, B., N. Jain and C. Williams, 2011. Managing cost, performance and reliability tradeoffs for energy-aware server provisioning. Proceedings of the IEEE INFOCOM, Apr. 10-15, IEEE Xplore Press, Shanghai, pp: 1332-1340. DOI: 10.1109/INFCOM.2011.5934917

Heo, J., P. Jayachandran, I. Shin, D. Wang and T. Abdelzaher *et al*., 2011. OptiTuner: On performance composition and server farm energy minimization application. IEEE Trans. Parallel Distrib. Syst., 22: 1871-1878. DOI: 10.1109/TPDS.2011.52

Horvath, T., T. Abdelzaher, K. Skadron and X. Liu, 2007. Dynamic voltage scaling in multitier web servers with end-to-end delay control. IEEE Trans. Comput., 56: 444-458. DOI: 10.1109/TC.2007.1003

Intel Corporation, 2008. Intel-64 and IA-32 architectures software developers manual. 3B System Programming Guide, Part 2.

Iverson, M.A., F. Ozguner and G.J. Follen, 1996. Run-time statistical estimation of task execution times for heterogeneous distributed computing. Proceedings of the 5th IEEE International Symposium on High Performance Distributed Computing, Aug. 6-9, IEEE Xplore Press, Syracuse, NY, USA, pp: 263-270. DOI: 10.1109/HPDC.1996.546196

Josyula, V., M. Orr and G. Page, 2011. Cloud Computing: Automating the Virtualized Data Center. 1st Edn., Cisco Press, Indianapolis, IN, ISBN-10: 1587204347, pp: 371.

Lang, W. and J.M. Patel, 2010. Energy management for map-reduce clusters. Proceedings of the 36th International Conference on Very Large Data Bases, (LDB' 10), pp: 129-139.

Lawson, B. and E. Smirni, 2005. Power-aware resource allocation in high-end systems via online simulation. Proceedings of the 19th Annual International Conference on Supercomputing, Jun. 18-21, Cambridge, MA, USA, pp: 229-238. DOI: 10.1145/1088149.1088179

Lee, G., 2012. Resource allocation and scheduling in heterogeneous cloud environments. PhD Thesis, College of Engineering, University of California, Berkeley.

Li, S., T. Abdelzaher and M. Yuan, 2011. TAPA: Temperature aware power allocation in data center with Map-Reduce. Proceedings of the International Green Computing Conference and Workshops, Jul. 25-28, IEEE Xplore Press, Orlando, FL, pp: 1-8. DOI: 10.1109/IGCC.2011.6008602

Moustafa, S.B., 2015. SLA monitoring for federated cloud services. MSc Thesis, School of Computing, Queen's University.

Nudd, G., D. Kerbyson, E. Papaefstathiou, S. Perry and J. Harper *et al*., 2000. PACE: A toolset for the performance prediction of parallel and distributed systems. In. J. High Perform. Comput. Applic., 14: 228-251. DOI: 10.1177/109434200001400306

Pisinger, D., 1995. Algorithms for knapsack problems. PhD Thesis, University of Copenhagen.

Portaluri, G., S. Giordano, D. Kliazovich and B. Dorronsoro, 2014. A power efficient genetic algorithm for resource allocation in cloud computing data centers. Proceedings of the 3rd IEEE International Conference on Cloud Networking, Oct. 8-10, IEEE Xplore Press, Luxembourg, pp: 58-63. DOI: 10.1109/CloudNet.2014.6968969

Reistad, B. and D. Gifford, 1994. Static dependent costs for estimating execution time. Proceedings of the ACM Conference on LISP and Functional Programming, Jun. 27-29, Orlando, FL, USA, pp: 65-78. DOI: 10.1145/182409.182439

Sanjay, H. and S. Vadhiyar, 2008. Performance modeling of parallel applications for grid scheduling. J. Parallel Distrib. Comput., 68: 1135-1145. DOI: 10.1016/j.jpdc.2008.02.006

Tesauro, G., R. Das, H. Chan, J.O. Kephart and C. Lefurgy *et al*., 2007. Managing power consumption and performance of computing systems using reinforcement learning. Proceedings of the 21st Annual Conference on Neural Information Processing Systems, (IPS' 07), Vancouver, Canada, pp: 1497-1504.

Wang, L. and Y. Lu, 2008. Efficient power management of heterogeneous soft real-time clusters. Proceedings of the Real-Time Systems Symposium, Nov. 30-Dec. 3, IEEE Xplore Press, Barcelona, pp: 323-332. DOI: 10.1109/RTSS.2008.31

Wang, X., X. Fu, X. Liu and Z. Gu, 2009. Power-aware CPU utilization control for distributed real-time systems. Proceedings of the 15th IEEE Real-Time and Embedded Technology and Applications Symposium, Apr. 13-16, IEEE Xplore Press, San Francisco, CA, pp: 233-242. DOI: 10.1109/RTAS.2009.12

Wu, L. and R. Buyya, 2011. Service Level Agreement (SLA) in Utility Computing Systems. In: Performance and Dependability in Service Computing: Concepts, Techniques and Research Directions, Cardellini, V. (Ed.), Information Science Reference, Hershey, PA, ISBN-10: 1609607953, pp: 1-25.

Yang, J., I. Ahmad and A. Ghafoor, 1993. Estimation of execution times on heterogeneous supercomputer architectures. Proceedings of the International Conference on Parallel Processing, Aug. 16-20, IEEE Xplore Press, Syracuse, NY, USA, pp: 219-226. DOI: 10.1109/ICPP.1993.80

Zhang, F., J. Cao, K. Li, S.U. Khan and K. Hwang, 2014a. Multi-objective scheduling of many tasks in cloud platforms. Future Generat. Comput. Syst., 37: 309-320. DOI: 10.1016/j.future.2013.09.006

Zhang, W., H. Xie, B. Cao and A.M. Cheng, 2014b. Energy-aware real-time task scheduling for heterogeneous multiprocessors with particle swarm optimization algorithm. Mathem. Prob. Eng. DOI: 10.1155/2014/287475

Zhang, Z. and J.M. Chang, 2014. A cool scheduler for multi-core systems exploiting program phases. IEEE Trans. Comput., 63: 1061-1073. DOI: 10.1109/TC.2012.283

Zhu, J., D. Li, J. Wu, H. Liu and Y. Zhangy *et al*., 2012. Towards bandwidth guarantee in multi-tenancy cloud computing networks. Proceedings of the 20th IEEE International Conference on Network Protocols, IEEE Xplore Press, Austin, TX, pp: 1-10. DOI: 10.1109/ICNP.2012.6459986