SCITECH
RESEARCH ORGANISATION|

Volume 6, Issue 1
July 07, 2016

Journal of Information Sciences and Computing Technologies
www.scitecresearch.com/journals

# Hybrid Approach for Resource Provisioning in Cloud Computing

Moh'd Z. Freaj[1], Azzam Sleit[2]
[1]University of Jordan
King Abdullah II School for IT
Department of Computer Science
Amman, Jordan
moe.freij@hotmail.com
[2]University of Jordan
King Abdullah II School for IT
Department of Computer Science
Amman, Jordan
asleit@ju.edu.jo

## Abstract

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. Elasticity of resources is considered as a key characteristic of cloud computing using this key characteristic; internet services are allocated the only-needed resources. This allocation of resources however should not be at the expense of the services' performance. Allocation of resources without degrading performance is called resource provisioning. Resource provisioning does not only support the elasticity of resources, but also enhances cost efficiency and sustainability.

The goal of this work is to investigate resource provisioning to increase the percentage of resources utilization without degrading the performance so that the power consumption of the cloud data centers is reduced. To achieve this goal, a hybrid-approach for resource provisioning is developed. In this approach, a list of virtual machines is requested, passed to a selection algorithm, sorting the machines according to their load, compute the threshold of the machines' load, and combining the high load with low load from two different virtual machines on one super virtual machine. The approach was implemented in a simulator called CloudSim. It was used to run two sets of experiments. The first is to measure the power consumption of the data center as whole and hosts as well. And the second is concerned with the processing times and memory usage.

The results have shown that this approach outperforms traditional counterparts in resource provisioning. The results showed that the hybrid approach achieved reduction of (5.85 MW/s) in power consumption compared with the traditional counterparts for the whole data center, as well as reduction of (2.48 MW/s) in power consumption for the hosts.

**Keywords:** Cloud Computing; Resource Provisioning; Energy Utilization; Resource Utilization; Power Aware; Cost Efficient; Sustainability; Elasticity of Resources.

## 1. Introduction

Cloud computing is a promising computer paradigm that makes the resources available when needed at reduced costs. It is also used to refer to a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction [8]. This paradigm has three delivery models, as well as four deployment models. Delivery models of Cloud computing are: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Deployment Models are: Private, Hybrid, Community, and Public [2].

Cloud computing is attractive because of a set of characteristics such as elasticity of resources, scalability, location transparency, reliability, cost efficiency, and sustainability [4]. The key characteristic is elasticity of resources in which internet services are provided with the only-needed resources [21]. Scalability is the ability of a cloud to be enlarged, or to handle the growth of resources in a capable manner [1]. Location transparency means that process location is independent of both the user's location and the resource location [5]. The probability of failure can define in terms of Reliability [9]. Cloud computing is cost efficient since it enabled the concept of pay-as-you-go, which minimizes the number of wasted resources [21]. Sustainability can be supported by maximizing the energy efficiency, i.e. reducing the power consumption. According to sustainability the cloud computing can be categorized as green computing, or green IT [21].

Although cloud computing is promising in terms of flexibility, it faces challenges such as resources provisioning, and security.

Resource provisioning problem is concerned with assigning resources to virtual machines while striving to maximize resource utilization. This problem is an NP-Complete problem [22]. Resource provisioning has two aspects under-provisioning and over-provisioning. Both situations are bad, the best situation for the resources provisioning is to provision only needed resources to a Virtual Machine (VM).

In this paper, we designed and implemented a Hybrid Approach for Resource provisioning in cloud computing (HARP). This approach tackles the problem of resource provisioning. Since resource provisioning affects the main advantages of cloud computing which are the illusion of having an infinite computing resources, energy utilization, and pay-as-you-go policy. These advantages can be mapped to the main characteristics of Cloud Computing: Elasticity of Resources, Sustainability, and Cost Efficiency. As a result, if we managed to have a good provisioning approach, it will support all of the mentioned characteristics. In such settings, the need of having a robust auto provisioning technique is becoming more essential. Therefore, HARP has been developed. It was built over thresholding, statistical median, and VM multiplexing concepts. In this approach, a set of virtual machines is passed to a selection algorithm, the selection algorithm first computes the threshold value by getting the maximum needed resources and multiplies it by the resistance ratio value, then it sorts the Virtual Machines (VMs) according to their workloads, and combining the high load with low load from two different VMs on one Super Virtual Machine (SVM). The approach was implemented in a simulator called CloudSim. It was used to run two sets of experiments. The first is to measure the power consumption of the data center as whole and hosts as well. And the second is concerned with the processing times and memory analysis.

The results showed that HARP outperforms its traditional counterparts. Since HARP achieved a reduction of (5.85 MW/s) in power consumption compared with the traditional counterparts for the whole data center, as well as a reduction of (2.48 MW/s) in power consumption for the hosts. Moreover, it achieved the same processing times with its traditional counterpart on a space shared environment but with extra (0.3%) overhead in the total running time of the simulation for the hybrid approach. Also, it achieved a reduction in the memory usage of (0.125%) and achieved the same processing times with its traditional counterpart on a time shared environment but with extra (0.02%) overhead in the total running time of the simulation.

The rest of this paper is organized as follows. Chapter 2 investigates the background and the related work of resource provisioning in cloud computing. Chapter 3 describes the Hybrid Approach for Resource Provisioning (HARP), and provides a theoretical example of how HARP works. Chapter 4 introduces the design of the experiments and provides an assessment of the experimental results of HARP. Chapter 5 concludes this paper and discusses the possibilities of future work.

## 2. Literature Review

In this work, the resource provisioning problem will be investigated. As we believe that enhancing the resource provisioning process, increases the percentage of resources utilization and reduction of power consumption will support the elasticity of resources, the sustainability, and the cost efficiency characteristics of Cloud Computing. In this chapter we will provide an overview of the resource provisioning problem and discuss the related work.

## 2.1 Resource Provisioning Background

Resource Provisioning or Resource Allocation is the problem of assigning resources to VMs by having as much high percentage of resource utilization as possible without degrading the application performance. Since resource allocation problem is difficult to solve, it has been addressed as an NP-Complete problem [22].

Resource provisioning problem has two aspects (1) under-provisioning and (2) over-provisioning. Resources over-provisioning means that the resources allocated to services are low utilized; i.e. waste of resources. However, resources under-provisioning means that resources are not enough for the VMs; i.e. services performance would be significantly degraded. See figure 1. Both situations are bad, the best situation for the resources provisioning is to provision only needed resources to a VM. And this should be equivalent to having the resources utilized. The most common techniques that were introduced to avoid the over and under provisioning problems were by defining an Service Level Agreement (SLA) as a performance constraint to demonstrate the needed resources for a specific VM.

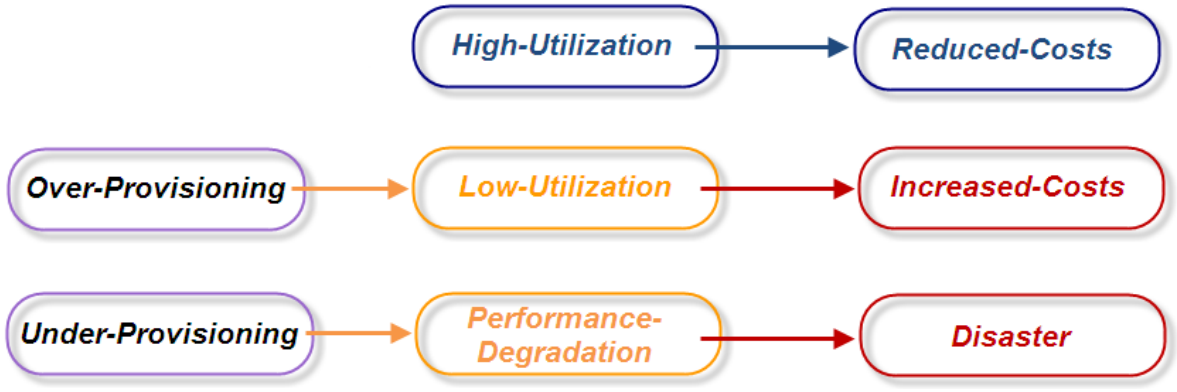


**Figure 1: Resource provisioning aspects**

## 2.2 Related Work

The resource provisioning in cloud computing is the main challenge as well as the main advantage of cloud computing. Researchers in this field were interested either in finding an auto provisioning algorithm, or to find a prediction algorithm to predict resources needed by a VM for ahead of time resource allocation [15], [13], [25]. See figure 2.

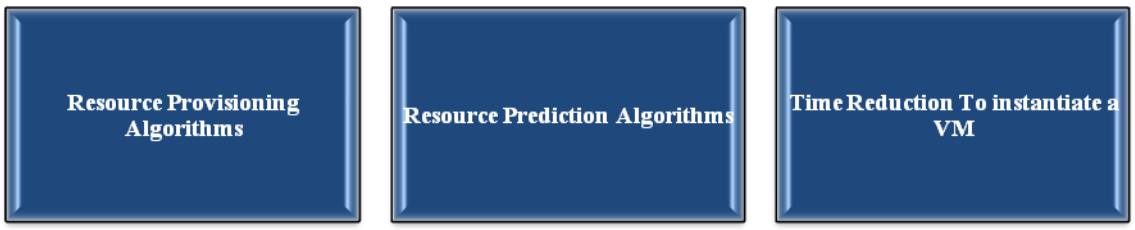


**Figure 2: The related studies categories**

### 2.2.1 Resource Provisioning Algorithms

In [15] Meng, et al. proposed a joint-VM provisioning approach in which multiple VMs are consolidated and provisioned together, based on an estimate of their aggregate capacity needs. Their approach exploits statistical multiplexing among the workload patterns of multiple VMs, i.e., the peaks and valleys in one workload pattern do not necessarily coincide with the others. Thus, the unused resources of a low utilized VM can be borrowed by the other co-located VMs with high utilization. Compared to individual-VM based provisioning, joint-VM provisioning could lead to much higher resource utilization.

In [14] Lim, et al. addressed the challenge of building an effective controller as a customer add-on outside of the cloud utility service itself. Such external controllers must function within the constraints of the utility service APIs.

In [23] Xiong & Suh presented an approach for resource provisioning, to minimize the total cost of cluster computing resources used by an application service provider for an e-business application, which often requires parallel computation for high service performance, availability, and reliability while satisfying an SLA. Simulation experiments demonstrate the applicability of the approach. Moreover, their approach minimized the total cost of computing resources allocated to a customer so that a given set of SLAs including percentile of the response time and cluster utilization is satisfied. They have further formulated the resource provisioning problem as an optimization problem subject to SLA constraints for a typical SLA-based cluster computing system, and developed an efficient approach to solving the problem. Finally, they have demonstrated how to use their proposed approach to finding the minimum values of computing resources required for the customer SLA guarantee by conducting numerical experiments.

In [18] Rogers, et al. discussed the problem of resource provisioning for database management systems operating on top of an Infrastructure-As-A-Service (IaaS) cloud. To solve this problem, they developed an extensible framework that, given a target query workload, continually optimizes the system's operational cost, estimated based on the IaaS provider's pricing model, while satisfying QoS expectations. Specifically, they described two different approaches, a "white-box" approach that uses a fine-grained estimation of the expected resource consumption for a workload, and a "black-box" approach that relies on coarse-grained profiling to characterize the workload's end-to-end performance across various cloud resources. They formalized both approaches as a constraint programming problem and using a generic constraint solver to efficiently tackle them. They presented preliminary experimental numbers, obtained by running TPC-H queries with PostsgreSQL on Amazon's EC2, that provide evidence of the feasibility and utility of their approaches.

In [24] Zhang, et al. applied a regression-based approximation of the CPU demand of client transactions on a given hardware. Then they used this approximation in an analytic model of a simple network of queues, each queue representing a tier, and show the approximation's effectiveness for modeling diverse workloads with a changing transaction mix over time. Using the TPCW benchmark and its three different transaction mixes they investigated factors that impact the efficiency and accuracy of the proposed performance prediction models. Experimental results showed that this regression-based approach provides a simple and powerful solution for efficient capacity planning and resource provisioning of multi-tier applications under changing workload conditions.

In [11] Govindan, et al. explored a combination of statistical multiplexing techniques to improve the utilization of the power hierarchy within a data center. At the highest level of the power hierarchy, they employed controlled under-provisioning and over-booking of hosted workloads' power needs. At the lower levels, they introduced the novel notion of soft fuses to flexibly distribute provisioned power among hosted workloads based on their needs. Their techniques were built upon a measurement-driven profiling and prediction framework to characterize key statistical properties of the power needs of hosted workloads and their aggregates. They characterized the gains in terms of the amount of computational work (CPU cycles) per provisioned unit of power – Computation per Provisioned Watt (CPW). Their technique is able to double the CPWoffered by a Power Distribution Unit (PDU) running the e-commerce benchmark TPC-W compared to conventional provisioning practices. Over-booking the PDU by 10% based on tails of power profiles yields a further improvement of 20%. Reactive techniques implemented on their Xen VMM-based servers dynamically modulate CPU DVFS states to ensure power draw below the limits imposed by soft fuses. Finally, information captured in our profiles also provides ways of controlling application performance degradation despite overbooking.

In [16] Padala, et al. developed an adaptive resource control system that dynamically adjusts the resource shares to individual tiers in order to meet application-level QoS goals while achieving high resource utilization in the data center. Their control system is developed using classical control theory, and they used a black-box system modeling approach to overcome the absence of first principle models for complex enterprise applications and systems. To evaluate their controllers, they built a testbed simulating a virtual data center using Xen virtual machines. they experimented with two multi-tier applications in this virtual data center: a two-tier implementation of RUBiS, an online auction site, and a two-tier Java implementation of TPC-W. Their results indicate that the proposed control system is able to maintain high resource utilization and meets QoS goals in spite of varying resource demands from the applications. In other words, they built a testbed for a data center hosting multiple multi-tier applications using virtualization. They have developed a two-layered controller using classical control theory. The controller algorithms were designed based on input-output models inferred from empirical data using a black-box approach.

### 2.2.2 Resource Prediction Algorithms

In [13] Islam, et al. they developed prediction-based resource measurement and provisioning strategies using Neural Network and Linear Regression to satisfy upcoming resource demands. Experimental results demonstrate that the

proposed technique offers more adaptive resource management for applications hosted in cloud environment, an important mechanism to achieve on-demand resource allocation in the cloud. They provided an evolutionary approach to constructing an effective prediction model for adaptive resource provisioning in the cloud in order to facilitate dynamic and proactive resource management, scheduling and capacity planning for interactive e-commerce applications where immediacy and responsiveness are vitally important. Throughout their study, they have evaluated several major machine learning algorithms, in particular varying sliding window size with a view to providing accurate forecasting ahead of time. They exemplified their proposed prediction techniques in the context of the dataset obtained by using TPC-W, a benchmark which is well-established for e-commerce applications.

In [7] Caron, et al. proposed an approach to the problem of workload prediction based on identifying similar past occurrences to the current short-term workload history. They presented in detail the auto-scaling algorithm that uses the above approach as well as experimental results by using real-world data and an overall evaluation of this approach, its potential and usefulness. As we know, one of the most important benefits of Cloud Computing is the ability of a Cloud Client to be dynamically scalable based on its use. This has great implications on cost saving as resources are not paid for when they are not used. Dynamic scalability is achieved through virtualization. The downside of virtualization is that they have a non-zero setup time that has to be taken into consideration for an efficient use of the platform. It follows that a prediction method would greatly aid a Cloud Client in making its auto-scaling decisions. In this approach a new resource usage prediction algorithm is presented. It uses a set of historic data to identify similar usage patterns to a current window of records that occurred in the past. The algorithm then predicts the system usage by interpolating what follows after the identified patterns from the historical data. Experiments have shown that the algorithm has good results when presented with relevant input data and, more importantly, that its results can improve by increasing the historic data size. This makes the evaluation of the algorithm be context dependent.

In [10] Gmach, et al. proposed and evaluated aspects of a capacity management process for automating the efficient use of such pools when hosting large numbers of services. They used a trace based approach to capacity management that relies on; a definition for required capacity, the characterization of workload demand patterns, the generation of synthetic workloads that predict future demands based on the patterns, and a workload placement recommendation service.

### 2.2.3 Time Reduction to Instantiate a VM

In [25] Zhu, et al. they studied the time reduction to instantiate a VM problem: how can the VMs and the applications inside are brought up as quickly as possible? This problem has not been solved satisfactorily in existing cloud services. They develop a fast start technique for cloud applications by restoring previously created VM snapshots of fully initialized application. They proposed a set of optimizations, including working set estimation, demand prediction, and free page avoidance, that allow an application to start running with only partially loaded memory, yet without noticeable performance penalty during its subsequent execution. They implemented their system, called Twinkle, in the Xen hypervisor and employ the two-dimensional page walks supported by the latest virtualization technology. They used the RUBiS and TPC-W benchmarks to evaluate its performance under flash crowd and failure over scenarios. The results indicate that Twinkle can provision VMs and restore the QoS significantly faster than the current approaches. In other words, they presented Twinkle, a fast resource provisioning mechanism for the Internet services which facilitate the feature of auto scaling in the cloud. By starting a virtual machine from partial snapshot and other techniques, this would reduce the time to provision a virtual machine to a few seconds without noticeable performance overhead. With Twinkle, the Internet services can keep closer to capacity requirement and can maintain application level performance in the cases of flash crowds and failures.

In [20] Silva, et al. presented a heuristic to optimize the number of machines that should be allocated to process tasks so that for a given budget the speedups are maximal. We have simulated the proposed heuristics against real and theoretical workloads and evaluated the ratios between number of allocated hosts, charged times, speedups and processing times. With the proposed heuristics, it is possible to obtain speedups in line with the number of allocated computers, while being charged approximately the same predefined budget. In other words, they presented a heuristic that efficiently defines the number of hosts to allocate on a utility computing infrastructure in order to solve bag-of-tasks problems. In the target environment, hosts may be allocated on demand, the user will later be charged for the time each host was used and jobs are composed of tasks whose execution times are not known before their execution. The results show that our heuristic determines the number of necessary hosts to guarantee that the charged time is close to desired value. The number of allocated hosts is close to the optimal value that would be found if task duration were previously known. The speedups accomplished are close to the number of allocated hosts. The presented heuristic can provide both a conservative as well as a more aggressive behavior. Varying both the creationRatio and the increaseRatio it is possible to lower the charged time (with higher timespan) or lower the job timespan with an increase in payment. So, if the user has a guess on the tasks processing time, this

information can be used to initially launch several computers. The number of computers to launch should be corrected with the creationRatio, in order to avoid the allocation of too much machines.

## 3. Proposed Work

This paper introduces a new provisioning approach to allocate resources in cloud computing. The goal of the new approach is to maximize both resources and energy utilization. The approach is expected to enhance performance by suggesting a threshold concept. This threshold incorporates the concepts of resistance ratio and statistical median.

The new approach works as follows. First, a list of virtual machines is requested and passed to a selection algorithm. The virtual machines are sorted according to their load and the statistical median of the virtual machine's load is computed. Finally, the high load and low load from two different VMs are combined on one SVM. The approach was implemented in a simulator, CloudSim, to run two sets of experiments. The first is to measure the power consumption of the data center as whole and its constituent hosts, and the second is concerned with the processing times and memory analysis. The new approach is called Hybrid Approach for Resource Provisioning or HARP for short.

### 3.1 Selection Algorithm

The HARP has utilized the concepts of VM multiplexing and thresholding depending on the resistance ratio concepts. This is achieved by inferring the multiplexing technique into the resource provisioning module. Thus, before assigning the requested resources to the VMs, a set of virtual machines is passed to a selection algorithm illustrated in figure 4. In the Figure, the set of virtual machines is represented by the variable VMList.

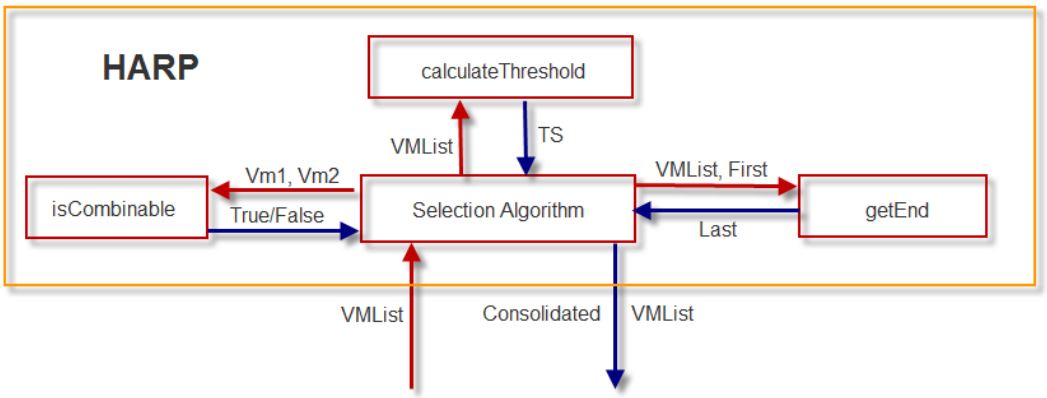


**Figure 3: HARP design in terms of input, output and process**

Figure 3 illustrates HARP in terms of Input, output, and process. The selection algorithm receives as an input a list of VMs and produces a modified List of VMs as an output. The algorithm, first, computes the threshold value. Then the VMs in the VMList are sorted according to their workloads. After that the variables: First and Last are being assigned, which represents the beginning and the end of the VMList are defined. Finally, the VMs consolidation decision is taken, i.e., either consolidate the VMs or not.

The selection n algorithm computes the threshold by invoking the calculate Threshold function, represented in figure 5, and the last, which defines the end of the VMList, is assigned by invoking the get End function, represented in figure 6.

```
Algorithm: Selection Algorithm
Input: List of VMs
Output: modified List of VMs
Start of the Algorithm

1.  TS = calculateThreshold(VMList VMs);
2.  Sort (VMs); //Sort VMs according to their Usage
3.  First = 0;
4.  Last = getEnd (VMList VMs, int Start);
5.
6.  For n = First to Last
7.  {
8.      If (isCominable (VMList{n}, VMList{Last}))
9.      {
10.         Combine (VMList{n}, VMList{Last});
11.     }
12.     Last = Last – 1;
13. }

End of the Algorithm
```

**Figure 4: Selection Algorithm**

The selection algorithm represented in figure 4. Its first line is an invocation to the calculate Threshold function by passing the VMList. The function returns the Threshold value which will be stored in the TS variable. The sorting will take place at the 2nd line of the selection algorithm and will be according to the workload usage. Merge sort on linked lists will be used due to its performance in terms of time and space. The third line will assign a value of (0) to the First variable. getEnd function will be invoked and will receive the VMList and First variables as its parameters. It will return a numeric value that is stored in the Last variable. We called the lines from 6 to 13 the consolidation part. Since the consolidation process is done in this part. The consolidation part starts with the loop. This loop applies the concept of statistical median as it loops over pairs of VMs, such as ([0, n], [1, n-1] …). These pairs are sorted according to their workload the least loaded with the most loaded. The loop has two counters; n counter which is incremental, and Last counter which is decremental. In the body of this loop, two VMs are passed to the is Combinable function. If the sum of the two VMs were less than or equals to the threshold, they will be consolidated. If not, they will be provisioned separately.

By consolidation, the VMs that are consolidated together are applicable to be provisioned in a utilized way. The remaining VMs are provisioned as normal, i.e., one-by-one. The threshold is the most vital part of HARP. Since the consolidation between two VMs is constrained by having a summation of workloads less than or equal to the threshold. In order to find the threshold value the calculate Threshold function was designed and implemented. In figure 5, the calculate Threshold function is represented. This function's input is a List of VMs and it returns a numeric value as its output.

```
Algorithm: calculateThreshold function
Input: List of VMs
Output: numeric value
Start of the Algorithm

1.  calculateThreshold(VMList VMs)
2.  {
3.      For each VM in VMList
4.      {
5.          If ( Max < VM -> NR)
6.          {
7.              Max = VM -> NR;
8.          }
9.      }
10.     return (Max) * RR;
11. }

End of the Algorithm
```

**Figure 5: Calculate Threshold function**

The calculate Threshold function, in the first 9 lines, loops through all of the VMs in the List of VMs, and gets the maximum needed resources. In the 10th line the calculate Threshold function multiplies the maximum needed resources by the resistance ratio value. The result of this multiplication is returned as the threshold value.

The getEnd function is presented in figure 6. This function returns a numeric value as the end of the VMList and takes as its input List of VMs and numeric value. This numeric value represents the end of the List of VMs.

```
Algorithm: getEnd function
Input: List of VMs, Start (numeric value)
Output: numeric value
Start of the Algorithm

1.  getEnd(VMList VMs, int Start)
2.  {
3.      End = VMs -> Count;
4.      If ( (End – Start) Mod 2 <> 0)
5.      {
6.          End = End + 1;
7.      }
8.      return End;
9.  }

End of the Algorithm
```

**Figure 6: getEnd function**

The getEnd function in the 3rd line assigns the End variable with the value of the count of VMs. In the lines (4 – 7), the function makes sure that the different between the beginning and the end of the VMList is even. If it is odd, then we increment the End variable by one; In order to make it an even number. Finally, getEnd function will return the End variable at the 8th line. This function was important to ensure the smoothness of the application of the statistical median concept in the selection algorithm.

The consolidation process is done depending on the computed threshold. This condition was implemented in the isCombinable function, presented in figure 7. This function will be returning a boolean value as its output, and takes two VMs as an input.

```
Algorithm: isCombinable function
Input: vm1, vm2
Output: Boolean value
Start of the Algorithm

1.  isCombinable(VM vm1,VM vm2)
2.  {
3.      Cap = vm1 ->NR + vm2 ->NR;
4.      If (Cap > TS)
5.      {
6.          Return False;
7.      }
8.      Return True;
9.  }

End of the Algorithm
```

**Figure 7: isCombinable function**

The isCombinable function in the 3rd line sum the resources needed by the two VMs that has been passed to it as an input and store it in the Cap variable. Then, in the line (4 – 9), it will compare the Cap and TS variables if the Cap was greater than the TS it will return false. Otherwise, it will return true.

The isCombinable function returns a decision to the selection algorithm whether to proceed to consolidation or not, of course not for any reason. Since HARP only supports resources conditions. I.e. if the RAM needed by VM1 is

(512MB), and for VM2 is (1024MB) and the threshold value is only (1024MB), then those VMs will not be consolidated to be provisioned together.

The idea behind VM multiplexing is to group VMs regarding to their workloads into sets of VMs. We would call a group as a super VM. Grouping VMs is done via the selection algorithm, which selects VMs which workload pattern differs, i.e. selecting VMs with complementary workloads, to reduce the number of VMs, and to get semi-unified sized VMs; which helps achieving the maximum percentage of the utilization, and speeds up the process of provisioning the resources. And as mentioned before that the selection algorithm comprises of three functions, that we demonstrate the responsibilities of each one of them.

In spite of, provisioning VMs in a VM-by-VM basis, VM multiplexing is to provision the grouped VMs unit. i.e. super VMs. The selection algorithm first finds the threshold depending on the number of resources needed by the most needy VMs and multiply it by the resistance ratio [12]. Then it finds the most compatible VMs regarding to the statistical median to group them into a super VM. To assure achieving highly resource savings, highly complementary should be grouped together. The selection algorithm is somehow simple, but efficient; because the demands of VMs vary and we guarantee to combine the VMs that totally complement each other.

The complexity of the selection algorithm is O (n) in terms of time and O (log n) in terms of space. Since, the calculateThreshold function takes O (n) as its complexity in terms of time and O (1) in terms of space. The sorting operation takes as O (n log n) as its complexity in terms of time and O (log n) in terms of space. Since we used merge sort on linked lists, which require O (log n) only pointers to be changed out of place [19]. O (1) is considered to be the complexity of the process of assigning the first and last variables, in terms of both time and space. And regarding the loop, which was called the consolidation part; it takes O (n/2) in terms of time and O (1) in terms of space.

HARP may have a set of limitations regarding security, because we may raise Multi-tenancy and/or Data Remanence threats [26].

## Experimental Design and Assessment of Results

This chapter introduces the design of the experiments and provides assessment of the results. The experiments were conducted using simulation via CloudSim Toolkit. CloudSim was used to conduct two sets of experiments. The first set measured the power consumption of a data center and its constituent hosts. The second set however measured the processing performance of the HARP in terms of processing times, run time, and memory usage. Initially, the Chapter introduces the experimental environment and then it presents the assessment of the two simulation experiments.

### 4.1. Experimental Environment

It is challenging to conduct repeatable, large-scale experiments on real cloud infrastructure. As a result, simulation using CloudSim Toolkit is chosen as a method to implement and evaluate the performance of the HARP. Simulation also reduces the cost and facilitates the repeatability of experiments.

#### *4.1.1. CloudSim Toolkit*

The CloudSim toolkit [6] is chosen as a simulation platform. It provides modeling and simulation environment for cloud computing infrastructure and services. In contrast to alternative simulation toolkits such as SimGrid and GangSim, it allows the modeling of virtual environments and supports and manages on demand resource provisioning. CloudSim is recently extended with crucial features such as enabling energy aware simulation and the ability to simulate service applications with dynamic workloads.

In this study, HARP will be implemented in CloudSim and simulation experiments will be conducted. The experiments aim at evaluating the performance of the HARP algorithm and compare it with other traditional resource provisioning techniques.

#### *4.1.2. Simulation Setup*

This study simulates a data center that comprises 800 heterogeneous physical nodes divided evenly into two categories of servers: HP ProLiant ML110 G4 and HP ProLiant ML110 G5. The specifications of the server are briefly presented in table 1.

**Table 1: Specifications of the simulated servers**

| | HP ProLiant ML110 G4 | HP ProLiant ML110 G5 |
|---|---|---|
| MIPS | 3720 | 37280 |
| Network Bandwidth | 1 Gb/s | 1 Gb/s |
| RAM | 4 GB | 8 GB |
| Storage | 1 TB | 1 TB |

#### *4.1.3. Performance Metrics*

Several performance metrics are used to measure the efficiency of HARP. The metrics include the following: (1) the power consumption of the data center; (2) the power consumption of the hosts; (3) the starting time of the cloudlets; (4) the waiting time of the cloudlets; (5) the response time of the cloudlets; (6) the execution time of the cloudlets; (7) the finishing time of the cloudlets; (8) the run time of the simulation; and (9) the memory usage in the simulation.

In the first set of experiments, the power consumption of the whole data center and its constituent hosts will be measured. In the second set of experiments, the following metrics will be measured: the starting time, the waiting time, the response time, the execution time, and the finishing time of the cloudlets, and the run time of the simulation and the memory usage in the simulation.

#### *4.1.4. Workload Data*

The experiments conducted in this study will use a set of data from the CoMon project, a monitoring infrastructure for PlanetLab [17]. The data was collected from servers located at more than 500 global locations. We have randomly chosen the workload data trace collected over 10 days between March and April 2011. While conducting the experiments, each VM is randomly assigned a workload trace from one of 10 days.

### 4.2. Assessment of Results

To measure the effectiveness of the HARP approach, the HARP algorithm was implemented in CloudSim and two sets experiments were run. The first set measured the power consumption of a data center and its constituent hosts. The second set measured the processing performance of the HARP in terms of processing times, run time, and memory usage.

#### *4.2.1. First Set of Experiments*

This set of experiments measures the power consumption of the data center as whole and its constituent hosts as individuals. Two power aware policies were implemented in CloudSim [3], Local Regression Virtual Machine allocation and Minimum Migration Time Virtual Machine selection. The goal is to measure the power consumption resulted from the implementation of the two policies before and after implementing HARP. The simulation of the policies before applying HARP are referred to as non-HARP policies while the policies after applying HARP are referred to as HARP policies.

##### *4.2.1.1. Power Consumption of Data Center*

The results of part 1 of the first set of experiments are presented in figure 8. In this Figure, the y-axis represents the power consumption in Watt per second and the x-axis represents the run time intervals of the simulation. We can see the simulation run time intervals are represented as double numbers. This is because CloudSim uses an incremental counter of type double to represent the time slices of the simulation. The simulations results of the non-HARP policies are represented by a blue color and the HARP policies are represented by Red color.

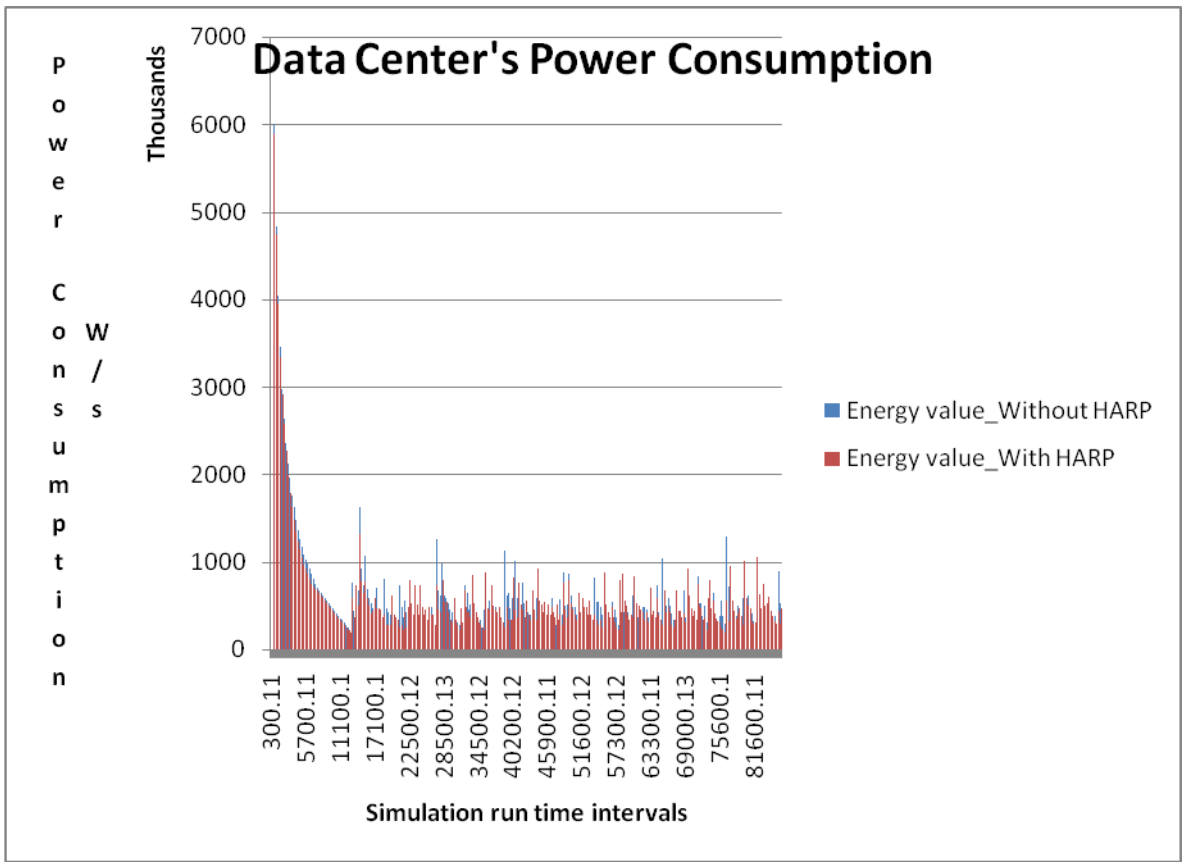


**Figure 8: Data Center power consumption**

The values in figure 8 show convergence in power consumption for the non-HARP and HARP policies. However, the results of the non-HARP policies show more leaps than their HARP counterparts.

The statistical information of the power consumption of data centers during the simulation run time is presented in table 2. The table presents the total and the average values of power consumption during all time intervals for both; HARP and non-HARP policies.

**Table 2: Data Center power consumption**

| | non-HARP policies | HARP policies |
|---|---|---|
| Total power consumption | 199,501,777.4 | 193,645,800 |
| Average power consumption | 260,817.6256 | 255,261.6699 |

The values in table 2 show that the results of the simulation of the HARP policies achieved less power consumption than non-HARP. For instance, the HARP policies achieved (5,855,977.35 W/s) reduction in power consumption of the data center as a whole. This reduction was caused by reducing the number of provisioned VMs. This maximizes the resources utilization and as a result minimizes the power consumption. Moreover, the results also shows that HARP policies achieved (5,555.96) reduction in the averages value of power consumption compared to non-HARP policies. This result was also caused by the reduction in the number of the provisioned VMs. This reduction provides an indicator that all the hosts and devices are reaching a higher percentage of both resource and energy utilization.

#### *4.2.1.2. Power Consumption of The Hosts*

Part 2 of the first set of experiments measures the power consumption of HARP and non-HARP policies for the individual hosts.

In figure 9, the x-axis represents the hosts while the y-axis represents the corresponding power consumption in Watt per second. The simulations results of the non-HARP policies are represented by a blue color and the HARP policies are represented by Red color.

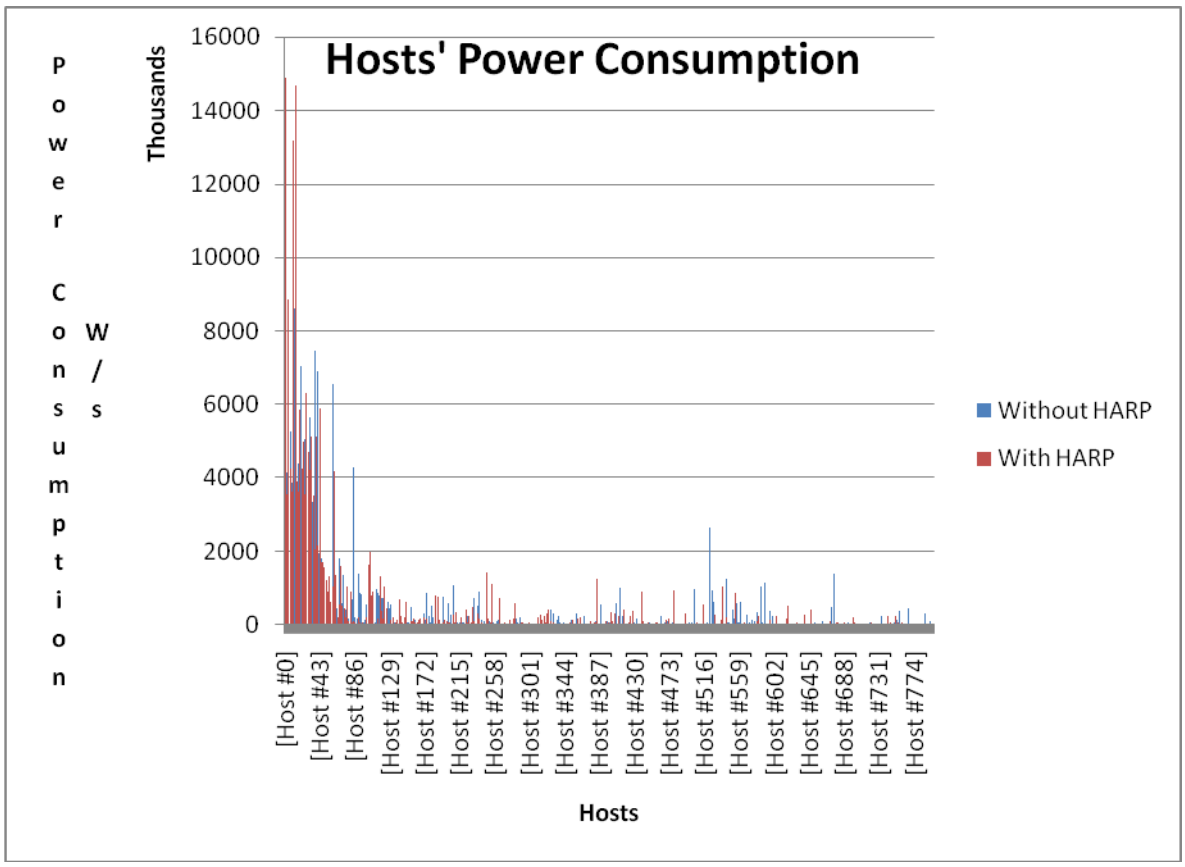


**Figure 9: Hosts' power consumption**

The values in figure 9 show that the non-HARP policies hosts consume power than their HAPR counterparts. It's noteworthy that the non-HARP policies increase in discrete, big leaps while their HARP counterparts increase slowly. To have a better vision of hosts' power consumption, statistical information regarding the overall power consumption must be considered. Table 3 introduces statistical information the power consumption of the hosts.

**Table 3: Statistics of Hosts' power Consumption**

| | non-HARP policies | HARP policies |
|---|---|---|
| Total power consumption | 189,576,188.8 | 187,092,637 |
| Average power consumption | 236,970.236 | 233,865.7963 |

The values in the table show that power consumption of the HARP policies is less than that of their non-HARP counterparts. For instance, the HARP policies achieved (2,483,551.72 W/s) reduction in power consumption of the data center as a whole. The reduction was caused by reducing the number of provisioned VMs and as a result maximizes the resource utilization. Moreover, the results also shows that HARP policies achieved (3,104.45) reduction in the averages value of power consumption compared to non-HARP policies. This result was also caused by the reduction in the number of the provisioned VMs. This reduction provides an indicator that all the hosts and devices are reaching a higher percentage of both resource and energy utilization.

In all, the results of the first set of experiments showed that the HARP policies outperformed their non-HARP policies in terms of power consumption at either center or its constituent hosts. For instance, the HARP policies achieved reduction of power consumption for the whole data center of (5.85 MW/s) compared to its non-HARP counterparts. Furthermore, the HARP policies achieved reduction in power consumption on hosts level of (2.48 MW/s) compared to its non-HARP counterparts.

### *4.2.2. Second Set of Experiments*

This set of experiments measures the performance of HARP in terms of metrics such as starting time, waiting time, response time, execution time, and finishing time of the cloudlets. The HARP approach is closely related to these metrics since the HARP executes before executing the cloudlets, and it consolidates VMS, which affect the

parameters. Other interesting metrics are the run time and memory usage. These extra metrics indicate the impact of the HARP on memory usage or on introducing any extra time overhead.

CloudSim originally has two policies, time sharing and space sharing scheduling policies. These two policies were used to emulate a time sharing and space sharing environments, respectively. The goal of this study is to measure the performance of the two policies in terms of all of the above mentioned metrics before and after implementing HARP. Again, the simulation of the policies before and after applying HARP is referred to as non-HARP policies and HARP polices, respectively.

#### *4.2.2.1. Time Sharing Environment*

Part 1 of the second set of experiments measures the starting time, the waiting time, the response time, the execution time, and the finishing time of cloudlets. The run time and total memory use are also measured.

The results of measuring the starting time, the waiting time, the response time, the execution time, and the finishing time of cloudlets by applying the HARP policies versus the non-HARP policies on a time sharing environment showed that although the HARP policies achieved in the first experiments of reduction in power consumption, this reduction did not affect the starting time, the waiting time, the response time, the execution time, and the finishing time of cloudlets. This means that HARP execution did not cause any extra overhead regarding the starting time, the waiting time, the response time, the execution time, and the finishing time of cloudlets. This is because the consolidation of the VMs was achieved according to a threshold value. The threshold value depends on the resources usage value of the neediest VM.

The results of measuring the run time and memory usage by applying the HARP policies versus the non-HARP policies on a time sharing environment are presented in table 4. This table compares the run time and memory usage caused by HARP Policies, represented in the first column, and non-HARP policies, represented in the second column.

**Table 4: Run time and memory usage comparison on time sharing environment**

| | non-HARP policies | HARP policies |
|---|---|---|
| **Run time** | 9.516s | 9.688s |
| **Memory usage** | 8M | 8M |

The values in table 4 shows that memory use in the simulation run times were the same for both policies. Introducing the HARP policies was expected to incur more memory use overhead due to the computational tasks within the policies. However, they did not since the computations were performed on variables passed by reference (not by value). From the table, once can notice an extra run time overhead of (0.02%) caused by the HARP policies compared to their non-HARP counterparts.

#### *4.2.2.2. Space Sharing Environment*

Part 2 of the second set of experiments measures the starting time, the waiting time, the response time, the execution time, and the finishing time of cloudlets. The run time and memory usage are also measured on a space shared environment.

The results of measuring the starting time, the waiting time, the response time, the execution time, and the finishing time of cloudlets by applying the HARP policies versus the non-HARP policies on a space sharing environment showed that although the HARP policies achieved reduction in power consumption as shown in the first experiments, they did not affect the starting time of the cloudlets. This means that the HARP execution did not cause any extra overhead regarding the starting time of the cloudlets. This is because the consolidation of the VMs was achieved according to a threshold value. The threshold value depends on the resources usage value of the neediest VM. The results of measuring the run time and memory usage by applying the HARP versus the non-HARP policies on a space sharing environment are presented in table 5. The table compares the run time and total memory of the HARP Policies, represented in the first column, with those of the non-HARP policies, represented in the second column.

**Table 5: Run time and memory use comparison on space sharing environment**

| | non-HARP policies | HARP policies |
|---|---|---|
| **Run time** | 9.501s | 12.323s |
| **Memory usage** | 8M | 7M |

The values in table shows that memory usage in the simulation run times were reduced by (0.125%) by the application of HARP policies on space sharing environment. This reduction was due to passing variables by reference (not by value) and doing the needed computations. The table also shows that the application of HARP policies introduced an extra overhead of (0.3%) on run time compared to non-HARP policies.

In all, the results of the second set of experiments showed that the HARP policies achieved a minimal overhead compared to their non-HARP policies in terms of processing times and memory usage at either time sharing or space sharing environments. For instance, HARP policies achieved the same starting time, waiting time, response time, execution time, finishing time, and memory usage as with its traditional counterparts on time sharing environment. Furthermore, the HARP caused extra (0.02%) overhead in the run time of the simulation. Whereas, on a space sharing environment, HARP achieved the same starting time, waiting time, response time, execution time, and finishing time as with its traditional counterpart. But HARP caused an extra (2.822s) which can be expressed as (0.3%) overhead in the run time of the simulation on a space sharing environment. HARP reduced the memory usage by (1M) which can be expressed as (0.125%) of the memory usage in the simulation.

## 5. Conclusion and Future Work

The goal of this work is to investigate resource provisioning in order to increase the percentage of both resources and energy utilization without degrading the performance. To achieve this goal, the HARP was developed. In this approach, a list of virtual machines is requested, passed to a selection algorithm, sorting the VMs according to their workload, compute the statistical median of the VM's workload, and combining the high load with low load from two different VMs on one SVM. The approach was implemented in a simulator called CloudSim. It was used to run two sets of experiments. The first is to measure the power consumption of the data center as whole and hosts as well. The second is concerned with the processing times and memory analysis.

The results have shown that this approach outperforms traditional counterparts in resource provisioning. The results of the first experiment showed that the hybrid approach achieved reduction of (5,855,977.35 W/s) which can be expressed as (0.04%) in power consumption compared with the traditional counterparts for the whole data center, as well as a reduction of (2,483,551.72 W/s) which can be expressed as (0.013%) in power consumption for the hosts.

For the second experiment on time sharing environment, it showed that hybrid approach achieved the same processing times with the traditional counterparts. Furthermore, the hybrid approach caused extra (0.172s) which can be expressed as (0.02%) overhead in the total running time of the simulation compared to its traditional counterparts. Also the results of the second experiment on a space shared environment achieved a reduction of memory usage of (1M) which can be expressed as (0.125%). It also achieved the same processing times for the hybrid approach and its traditional counterpart but with extra (2.822s) which can be expressed as (0.3%) overhead in the total running time of the simulation for the hybrid approach against its traditional counterparts.

Our next moves will be towards the concepts of forecasting the resources needed by a specific VM ahead of time. We are looking forward to develop a resource prediction algorithm for HARP that applies the concepts of forecasting the resources needed ahead of time, depending on the history of resources usage for each VM. This algorithm will be estimating the resources that a VM will need after a period of time; we believe that by implementing the prediction algorithm for HARP, it will give much better results than what we have for now.

## References

[1] D. Agrawal, A. El Abbadi, S. Das, and A. J. Elmore, Database scalability, elasticity, and autonomy in the cloud, in Proceedings of the 16th Intl. conference on Database systems for advanced applications-Volume Part I, ser. DASFAA'11, 2011, pp. 2 - 15.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stocia, and M. Zahira, Above the clouds: A Berkeley view of cloud computing, U.C. Berkeley, EECS Department, 2009.

[3] A. Beloglazov and R. Buyya, Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers, Concurrency and Computation: Practice and Experience (CCPE), pp. 1397-1420, 2012.

[4] M. A. Bhat, R. M. Shah, B. Ahmad, and I. R. Bhat, Cloud Computing: A Solution to Information Support System (ISS), International Journal of Computer Applications, 2010.

[5] S. J. Biggs and S. Vidalis, Cloud Computing and The Impact On Digital Forensic Investigations, International Conference for Internet Technology and Secured Transactions ICITST, 2009.

[6] R. N. Calheiros, R. Ranjan, A. F. Beloglazov, C. A. De Rose, and R. Buyya, CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms.

[7] E. Caron, F. Desprez, and A. Muresan, Forecasting for Cloud computing on-demand resources based on pattern matching, INRIA, 2010.

[8] M. Carroll, P. Kotzé, and A. van der Merwe, Securing Virtual and Cloud Environments. In I. Ivanov et al. Cloud Computing and Services Science, Service Science: Research and Innovations in the Service Economy, 2012.

[9] T. Dillon, C. Wu, and E. Chang, Cloud Computing: Issues and Challenges, IEEE Computer Society, 2010, pp. 27-33.

[10] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, Capacity management and demand prediction for next generation data centers, In IEEE Intl. Conference on Web Services, 2007, pp. 43–50.

[11] S. Govindan, J. Choi, B. Urgaonkar, A. Sivasubramaniam, and A. Baldini, Statistical profiling-based techniques for effective power provisioning in data centers, In EuroSys '09: Proceedings of the 4th ACM European conference on Computer systems, ACM, 2009, pp. 317–330.

[12] Gupta and Nikhil, Fibonacci Retracements and Self-Fulfilling Prophecy, Honors Projects, Paper 41, http://digitalcommons.macalester.edu/economics_honors_projects/41, 2011.

[13] S. Islam, J. Keung, K. Lee, and A. Liu, Empirical prediction models for adaptive resource provisioning in the cloud, Future Generation Computer Systems, 28(1), 2012, pp. 155-162.

[14] H. C. Lim, S. Babu, J. S. Chase, and S. S. Parekh, Automated control in cloud computing: challenges and opportunities, ACDC '09: Proceedings of the 1st workshop on Automated control for datacenters and clouds, New York, NY, USA: ACM, 2009, pp. 13-18.

[15] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, Efficient resource provisioning in compute clouds via VM multiplexing, Proceeding of the 7th international conference on Autonomic computing, Washington, DC, USA, 2010.

[16] P. Padala, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, and K. Salem, Adaptive control of virtualized resources in utility computing environments, In ACM SIGOPS/EuroSys European Conference on Computer Systems, ACM, 2007.

[17] KS. Park and VS. Pai, CoMon: a mostly-scalable monitoring system for PlanetLab. ACM SIGOPS Operating Systems Review 2006, 2006, 40(1):74.

[18] J. Rogers, O. Papaemmanouil, and U. Cetintemel, A generic auto-provisioning framework for cloud databases, Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on 1-6 March 2010, Long Beach, CA, 2010, pp. 63-68.

[19] Robert Sedgewick, Algorithms in Java, 3rd Edition, Pearson Education, Inc, 2003.

[20] J. a. Silva, L. Veiga, and P. Ferreira, Heuristic for resources allocation on utility computing infrastructures, MGC '08 Proceedings of the 6th international workshop on Middleware for grid computing, New York, NY, USA: ACM, 2008, pp. 1-6.

[21] C. Weinhardt, A. Anandasivam, B. Blau, N. Borissov, T. Meinl, W. Michalk, and J. Stößer, Cloud Computing – A Classification, Business Models, and Research Directions. Business and Information Systems Engineering, 1 (5), 2009, pp. 391-399.

[22] J. Weinman, Cloud Computing is NP-Complete. Retrieved March 21, 2013, from Joe Weinman: http://www.joeweinman.com/Resources/Joe_Weinman_Cloud_Computing_Is_NP-Complete.pdf, 2011.

[23] K. Xiong and S. Suh, Resource provisioning in SLA-based cluster computing, Proceedings of the 15th international conference on Job scheduling strategies for parallel processing (JSSPP'10). Heidelberg: Springer-Verlag Berlin, 2010.

[24] Q. Zhang, L. Cherkasova, and E. Smirni, A Regression-Based Analytic Model for Dynamic Resource Provisioning of Multi-Tier Applications, Fourth International Conference on Autonomic Computing (ICAC'07), 2007, p. 27.

[25] J. Zhu, Z. Jiang, and Z. Xiao, Twinkle: A fast resource provisioning mechanism for internet services, Proceedings of INFOCOM 2011, Shanghai, 2011, pp. 802-810.

[26] D. Zissis, and D. Lekkas, Addressing cloud computing security issues, Future Generation Computer Systems 28, pp. 583–592, 2012.

[27] A. Sleit, M. Al-Akhras, I. Juma, and M. Alian, "Applying ordinal association rules for cleansing data with missing values," Journal of American Science, vol. 5, no. 3, pp. 52–62, 2009.

[28] Al-Hasan, H., Qatawneh, M., Sleit, A., and Almobaideen, W. (2011) EAPHRN: Energy-Aware PEGASISBased Hierarchal Routing Protocol for Wireless Sensor Networks, Journal of American Science, 7(8), pp. 753-758.

[29] Wesam Almobaideen, Dimah Al-Khateeb, Azzam Sleit, Mohammad Qatawneh, Khadejeh Qadadeh, Rasha Al-Khdour, Hadeel Abu Hafeeza, "Improved Stability Based Partially Disjoint AOMDV", International Journal of Communications, Network and System Sciences, vol.6, pp.244-250, 2013.

[30] Sleit, A., Qatawneh, M., Al-Sharief, M., Al-Jabaly, R., & Karajeh, O. (2011). Image Clustering using Color, Texture and Shape Features. KSII Transactions on Internet & Information Systems, 5(1).