Summer 2014

Task Scheduling for Efficient Resource Utilization in Cloud

A Project Report for course COEN 241

Under the guidance of,
Dr.Ming-Hwa Wang

Submitted by :
Najuka Sankhe
Nikitha Karkala
Nimisha Sara Mathews

ACKNOWLEDGEMENT

We would like to express our special appreciation and gratitude to our Professor, Dr.Ming-Hwa Wang for giving us this opportunity and encouraging our research.

We would also like to thank the Santa Clara University library for providing us with group discussion rooms in the college, which helped us in concentrating on the task.

Lastly, we would like to thank our family and friends who supported us during our work.

Table of Contents

1. Abstract

The rapid growth in demand for computational power driven by modern service applications combined with the shift to the Cloud computing model have led to the establishment of large-scale virtualized data centers. Such data centers consume enormous amounts of electrical energy resulting in high operating costs and carbon dioxide emissions. Dynamic consolidation of virtual machines (VMs) using live migration and switching idle nodes to the sleep mode allow Cloud providers to optimize resource usage and reduce energy consumption. However, the obligation of providing high quality of service to customers leads to the necessity in dealing with the energy-performance trade-off, as aggressive consolidation may lead to performance degradation. Due to the variability of workloads experienced by modern applications, the VM placement should be optimized continuously in an online manner. Efficient task scheduling mechanism can meet users' requirements, and improve the resource utilization, thereby enhancing the overall performance of the cloud computing environment. **A scheduling algorithm which takes into account maintaining quality of service availed to the users, as well as, optimally reducing energy consumption and increasing efficient resource utilization is proposed and experimented with, on a simulation environment**.

2. Introduction

2.1. Objective

Cloud computing leverages virtualization of computing resources allowing customers to provision resources on-demand on a pay-as-you-go basis. Instead of incurring high upfront costs in purchasing IT infrastructure and dealing with the maintenance and upgrades of both software and hardware, organizations can outsource their computational needs to the Cloud. The proliferation of Cloud computing has resulted in the establishment of large-scale data centers containing thousands of computing nodes and consuming enormous amounts of electrical energy. It has been estimated that by 2014 infrastructure and energy costs would contribute about 75%, whereas IT would contribute just 25% to the overall cost of operating a data center. The reason for this extremely high energy consumption is not just the quantity of computing resources and the power inefficiency of hardware, but rather lies in the inefficient usage of these resources.

Data collected from more than 5000 production servers over a six-month period have shown that although servers usually are not idle, the utilization rarely approaches 100%. Most of the time servers operate at 10-50% of their full capacity, leading to extra expenses on over-provisioning. One of the ways to address the energy inefficiency is to leverage the capabilities of the virtualization technology. The virtualization technology allows Cloud providers to create multiple Virtual Machine (VMs) instances on a single physical server, thus improving the utilization of resources. The reduction in energy consumption can be achieved by switching idle nodes to low-power modes (i.e., sleep, hibernation), thus eliminating the idle power consumption. Moreover, by using live migration the VMs can be dynamically consolidated to the minimal number of physical nodes according to their current resource requirements. Through this project we aim at proposing an a scheduling algorithm that maximizes energy efficiency of the data center, while not compromising other services offered.

2.2 What is the problem

Efficient resource management in Clouds is not trivial, as modern service applications often experience highly variable workloads causing dynamic resource usage patterns. **Aggressive consolidation of VMs can lead to performance degradation when an application encounters an increasing demand resulting in an unexpected rise of the resource usage**. If the resource requirements of an application are not fulfilled, the application can face increased response times, time-outs or failures. Ensuring reliable Quality of Service (QoS) defined via Service Level Agreements (SLAs) established between Cloud providers and their customers is essential for Cloud computing environments; therefore, Cloud providers have to deal with an energy-performance trade-off – the minimization of energy consumption, while meeting the SLAs. The focus of this work is on energy and performance efficient task scheduling and resource management strategies that can be applied in a virtualized data center by a Cloud provider (e.g. Amazon EC2).

For task scheduling, unlike traditional algorithms, cloud computing has an extra level of virtualization which comes with an advantage of being scalable but has a downside of requiring an additional step in scheduling.
i.e There are two decisions that need to be made for scheduling resources in Cloud.

1. Mapping between virtual machines and hosts. Which VM should be run on which host?
2. Mapping between user's task and VM. i.e Which task runs on which VM?

Also once the virtual machines are running on the host, it is essential to watch out for overloading of hosts or underloading of hosts. **Overloading should be detected to avoid violation of SLAs and provide sufficient computing power and QoS to each task as per user requirements. Underloading of hosts should be detected so that underloaded hosts can be unloaded, by migrating all VMs from it, and then leaving it on low-power mode, as to reduce power consumption.**

2.3 Why this is a project related the class

A cloud environment consists of multiple customers requesting for the available resources. Cloud vendors who offer Infrastructure as a Service should enable efficient management of the available resources. Proper scheduling in cloud enables the selection of best suitable resources for task execution. Additionally, timely detection of imbalance of host loads, can amend SLA violations and wastage of power. Due to the complexities and dynamics in the cloud environment, task scheduling is a highly researched problem in Cloud Computing.

2.4 Why other approach is no good

Most of the approaches for task scheduling in Cloud considers only mapping user's tasks to VM's. These task scheduling algorithms do not consider whether the host is overloaded or underloaded and hence result in an imbalance and low efficiency in resource utilization.

2.5 Why you think your approach is better

Our approach involves a **task scheduler that is energy aware** . It not only dynamically maps tasks to Virtual Machines but also considers the amount of resources available on the host system. Additionally, it is a more ecologically friendly approach as it aims at reducing energy consumption, without reducing service standards. Hence along with meeting user's requirements it results in high resource utilization unlike other approaches.

2.6 Statement of the problem

Design a task scheduling algorithm that results in high resource utilization efficiency while meeting the user's requirements. The current task scheduling algorithms in Cloud simply map tasks to VM's without considering the load balancing of the host systems. This may be to avoid the overhead involved since meeting user's requirements is of highest priority without violating the SLA than resource utilization. But inefficient resource utilization can incur significant wastage of power resources and loss to the cloud service providers.

Hence our solution is to design a task scheduling algorithm that also considers the load of the hosts, avoiding overload so as to meet the user's requirements on time (adhering to SLA and improving QoS), and detect underloaded hosts to avoid unnecessary consumption of energy.

2.7 Area or scope of investigation

- ➔ Task scheduling : Map user's tasks to VM's
- ➔ Load balancing : At each step check if the host is under or over utilized, and take steps to avoid over/under loading.
- ➔ Simulate on CloudSim.
- ➔ Comparison with Existing solutions
- ➔ Try implementing same algorithm on OpenStack.

3. Theoretical Bases and Literature Review

3.1 Definition of the problem

In Cloud, virtualization technology is used for efficient management of resources. This projects aims at solving the problem of dynamic and efficient task scheduling and resource utilization in a virtualized data center by mapping user's requests (tasks) to virtual machines and mapping from virtual machines to hosts, followed by addressing imbalance in loads on hosts for efficient utilization of resources.

3.1 Theoretical background of the problem

Cloud computing has added the extra level of virtualization in the task allocation process. This gives an advantage of scalability, but has the downside of requiring an additional step in the scheduling. Therefore, two scheduling decisions are required in Cloud. The first level scheduling is from the users' requests to the virtual machine, and the second is from the virtual machine to host resources.

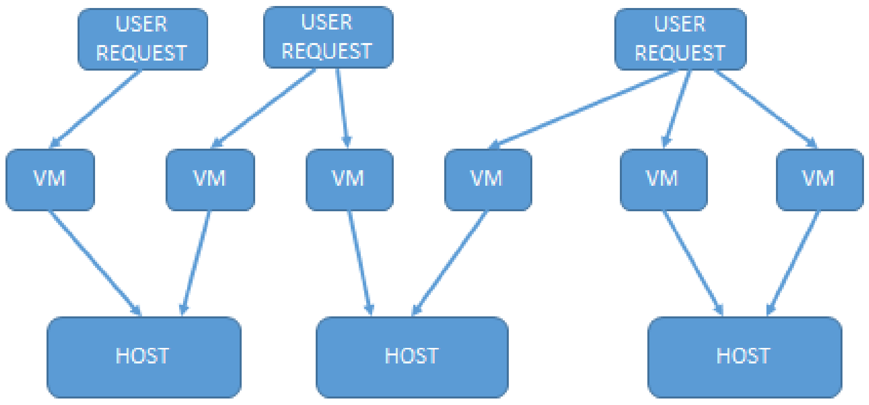


Fig. Two levels of scheduling model in cloud computing

Many algorithms like FCFS, Round Robin, Active-VM monitoring and Throttled are used for executing clients request with a minimum response time and also assigning the requests to the virtual machines. But the constraints such as high communication delays, underutilization of the resources are not addressed clearly and efficiently, which leads to many of the resources does not participate in executing the requests and hence leads to imbalance of cloud system.

3.2 Related research to solve the problem

3.2.1 Online Multi-Resource Scheduling for Minimum Task Completion Time in Cloud Server

In this paper, an algorithm for online scheduling of resources is proposed.There are two phases involved. The first phase is triggered when a task arrives for execution and second phase is triggered after the completion of the task. In the first phase , best fit is used to choose the server on which the task will run. In the second phase , all the tasks are sorted based on execution times , and a task that can run on the free VM is chosen . This method performed better than all traditional methods such as best-fit,worst-fit etc in terms of the CPU and memory utilization.

3.2.2 Scheduling Using Improved Genetic Algorithm in Cloud Computing for Independent Tasks

In this paper , a modification to a genetic algorithm is proposed for scheduling tasks to Virtual Machines. The algorithm was compared against the traditional genetic algorithm and performed better in terms of the makespan for resource utilization.

3.2.3 Optimal Load Balancing in Cloud Computing By Efficient Utilization of Virtual Machines

In this paper, a novel VM-assign algorithm is presented which allocates incoming jobs to available virtual machines. Here the virtual machine assigned depending on its load i.e. VM with least request is found and then new request is allotted. With this algorithm underutilization of the virtual machine is improved significantly and later it is compared with existing Active-VM algorithm.

3.2.4 Cost-Optimized Resource Provisioning in Cloud

In this paper, the virtualization technology is used to provide resources to the cloud consumers through the cloud broker.This paper proposes cost-optimized resource provisioning in cloud using Bin-packing VM placement algorithm which comprised of job placement and VM placements modules. Consumer demands are placed into the VMs using Best-fit strategy.

3.2.5 An Empirical Investigation on the Simulation of Priority and Shortest Job First Scheduling for Cloud-based Software Systems

In this paper, a scheduling algorithm integrated with task grouping, priority-aware and SJF (shortest-job-first) to reduce the waiting time and makespan, as well as to maximize resource utilization is proposed. It is seen that in comparison with existing task grouping algorithms, results show that the proposed algorithm waiting time and processing time decreased significantly (over 30%).

3.2.6 Cloud Task Scheduling Based on Ant Colony Optimization

In this paper, cloud task scheduling is viewed as an NP-complete problem and an ant colony optimization is used for scheduling tasks to virtual machines. The main goal of the algorithms is minimizing the makespan of a given tasks set. Experimental results showed that the ant colony optimization outperformed FCFS and round-robin algorithms.

3.3 Advantage/disadvantage of those research

3.3.1 Online Multi-Resource Scheduling for Minimum Task Completion Time in Cloud Servers

- **Advantages**
  - High resource utilization.
  - Minimum queuing delay**.**

- **Disadvantages**
  - Does not consider virtualization into consideration.

3.3.2 Scheduling Using Improved Genetic Algorithm in Cloud Computing for Independent Tasks

- **Advantages**
  - Reduces makespan

- **Disadvantages**
  - Does not consider the mapping between VM's and hosts. i.e no load balancing is done.

3.3.3. Optimal Load Balancing in Cloud Computing By Efficient Utilization of Virtual Machines

- **Advantages**
  - It solves the problem of inefficient utilization of the VMs / resources compared to existing algorithm.
  - Management of the dynamic resources in cloud platform can be efficiently given by virtualization technology. It provides a new way to improve the power efficiency of the data centers.

- **Disadvantages**
  - While making comparison the author has considered only load distribution aspect. Other aspects like response time, energy consumption is not analyzed.
  - Existing Active-VM algorithm is used for comparison is also proposed by same author.
  - How algorithm will respond if we mix both static and dynamic loads is not specified in the paper.

3.3.4 Cost-Optimized Resource Provisioning in Cloud

- **Advantages**
    - In the proposed system, the concept of Bin-Packing approach is used, that make use of the concept of server virtualization to minimize the power consumption.
    - With this approach, it is possible to minimize the cost of a running data center.
    - Proposed work reduces the number of physical machine required to execute the demanded jobs.

- **Disadvantages**
    - In this paper, underutilization of the resources are not addressed clearly and efficiently, which leads to imbalance of cloud system.

3.3.5 An Empirical Investigation on the Simulation of Priority and Shortest-Job-First Scheduling for Cloud-based Software Systems

- **Advantages**
    - Reduces waiting time
    - Minimized turnaround time of tasks
    - Reduced influence on the bottleneck of bandwidth usage

- **Disadvantages**
    - Load balancing is not being considered.
    - Longer jobs always tend to get pushed back, as shorter jobs get priority.

3.3.6 Cloud Task Scheduling Based on Ant Colony Optimization

- **Advantages**
    - Considerable reduction in task completion time.
    - Dynamic task allocation is made possible.

- **Disadvantages**
    - The same resource and VM would be over utilized, since ants would converge to first available position.

3.4 Your solution to solve this problem

Our proposed solution is to use a scheduling model, where scheduling occurs from tasks to VM, and then VMs are placed on suitable resources, i.e, there are two levels of scheduling. The scheduling mechanism take into account the dynamic requirements of users and the load balancing in cloud environment.

In this two level scheduling model, the first scheduling phase creates the description of a virtual machine, including the set of computing resources, network resources, storage resources, and other configuration information, according to the demands of a task. Then the second scheduling phase finds appropriate resources for the virtual machine in the host resources based on the virtual machine description provided from the first phase.

Load balancing is achieved with dynamic migration operation. If the virtual machine scheduled to a host experiences an increase in computational amount, leading to a heavy load on the virtual machine, resulting in load imbalance, then we use a dynamic migration operation to maintain load balance in current environment. And if a host experiences an underload, all VMs from that host are transferred to other active hosts, and the current host is switched to low power mode to save power.

3.5 Where your solution different from others

Most task scheduling algorithms that we have encountered, deals with mapping user's task to a virtual machine. Very few task scheduling algorithms consider mapping from virtual machine to host resources. But, both mappings ie. user's task to virtual machine and virtual machine to host resources are rarely addressed in single approach. This is one of the main areas where the solution we are proposing differs from other methods.We also aim at minimizing energy by switching off underutilized host.

There are many algorithms for executing clients request with a minimum response time and also assigning the requests to the virtual machines. However, very few address issues like underutilization of resources and overutilization of resources. The two level task scheduling along with efficient utilization of resources is not addressed by any of the above referenced papers.

3.6 Why your solution is better

Our proposed solution is based on scheduling model with load balancing approach. Scheduling model is divided into two levels. One is the mapping from task to a virtual machine, another is mapping from the virtual machine to host resources.Dynamic migration strategy is used for efficient utilization of resources ie.load balancing.Our proposed solution considers both decisions required for scheduling resources in Cloud. At the same time it also deals with efficient utilization of resources. Hence, under/over utilization situation will not arise.

In simple words, two level task scheduling i.e. mapping of resources from the user's to the virtual machine and from the virtual machines to host will meet the dynamic task requirement of the users, and reduce response time whereas load Balancing will improve utilization of resources. This method of scheduling would be profitable for both the users and providers of the computational resource.

4. Methodology

As cloud computing is a very new and emerging field, there are not many sources of standard inputs or benchmarks to perform a comparative analysis with. Additionally, to perform benchmarking we need to perform experiments on a repeatable, dependable, and scalable environment, which is not possible in the real-world cloud, because of the large array of different cloud service providers, and the differences in their scheduling policies, and environments.

So, to obtain a holistic software framework for modelling cloud computing environments and perform testing, we use a simulator called CloudSim for modelling the cloud. The primary function of CloudSim is to provide a generalized, and extensible simulation framework that enables seamless modeling, simulation, and experimentation of emerging Cloud computing infrastructures and application services. By using CloudSim, researchers and developers can focus on specific system design issues that they want to investigate, without getting concerned about the low level details related to Cloud-based infrastructures and services.

Using CloudSim, we propose a comparison of the execution time and resource efficiency with other scheduling algorithms, using a common input.

4.1 Input

To make a simulation-based evaluation applicable, it is important to conduct experiments using workload traces from a real system. In CloudSim we use data provided as a part of the CoMon project, a monitoring infrastructure for PlanetLab.

The input will consist of the data on the CPU utilization by more than a thousand VMs from servers located at more than 500 places around the world. The interval of utilization measurements is 5 minutes and each traced file have 288 lines, therefore, each one represent a VMs CPU utilization about 24 hours.

4.2 Output

The energy utilization and SLA violation for tasks ranging from sets of 10 - 1000 is observed. Graphs are plotted, comparing the results obtained running the proposed method, and other scheduling algorithms, to ascertain the real statistical difference in performance between the methods.

4.3 Algorithm Design Description

The algorithm follows a two-level scheme of scheduling. In order to describe that, we define a set of tasks $T = \{t_0, t_1, t_2, \dots t_{n-1}\}$ and the number of tasks is $n = | T |$, and a set of hosts $H = \{h_0, h_1, h_2, .$

. . h_{m-1}} and the number of hosts is m = | H |. The load on a host machine is defined as the average load of virtual machines that run on it.

We describe the algorithm below:

Step 1 : Establish the host resource set, H = {h_0, h_1, h_2, . . . h_{m-1}}, and sort in ascending order of their processing power.

Step 2 : Establish the set of tasks, T = {t_0, t_1, t_2, . . . t_{n-1}}. Then according to the properties of each task, the first level of the scheduler establishes the description of the virtual machine required, and thus providing configuration information for allocation of resources and creation of the virtual machine.

Step 3 : According to the virtual machine description of Task $t_i \in T$, select a host resource h_j that can meet the required resources and the load is lightest. If the host exists, create the virtual machine and allocate the required resource for it then update the available resources Host h_j, otherwise take the Task t_i to the tail of the task queue and wait for the next scheduling.

Step 4 : If the resource requirements of the Task t_i increase, find whether the host whose virtual machine of Task t_i run on can meet the additional required resources, if it exists, allocate the additional required resources for it, reconfigure the virtual machine, and then update the host's available resources. Otherwise, the virtual machine is migrated to the host with lightest load and the additional required resources to execute continuously.

Step 5 : If the resource requirements of the Task t_i reduce, release the excess resources that the virtual machine occupied, and update the available resources held by the host.

Step 6 : If Task t_i has been completed, then destroy the virtual machine of Task t_i and release the occupied resources for the other unfinished tasks.

Step 7 : Calculate the load on each host, and find the standard deviation, if the load on a particular host is much higher than the rest, select a virtual machine with the lightest load from that host and migrate it to a host with the lightest load. If the load on a particular hosts is very light, migrate all VMs from it, to other hosts.

Step 8 : Repeat Steps 3 to 7 until all tasks are completed.

In the above algorithm, the virtual machine is scheduled to the host with lightest load each time. The advantage is to avoid overloading for the host. If a particular virtual machine is scheduled to a host and the computational amount increases, leading to a higher load on the virtual machine, resulting in load imbalance, then take the dynamic migration operation (described in Step 7), maintaining load balance in current environment.

Language Used: Java

Tools Used: Eclipse , CloudSim.
Benchmark: PlanetLab virtual machine workload traces.

The algorithm will be implemented on the java version of CloudSim v3.0.0, using Eclipse IDE.

5. Implementation:

5.1 Code

The code consists of the source code to implement cloudSim. The following files were modified to run and test the algorithm on this tool.

1. RunnerAbstract.java

The following files are added to implement our proposed algorithm:

1. PlanetLabRun.java
2. StDev.java
3. MyVmAllocationPolicy.java
4. MyVmSelectionPolicy.java

Modules Pseudocode :

I. VM to Host Mapping : This module chooses the host with the lightest load for allocating VM

```
1. Input: ListofTargetHosts,, Output: Host to allocate the VM
2. minCapacity ←MAX // In order to calculate the capacity we are using the RAM of the host.
3. allocatedHost←NULL
4. For each host in ListofTargetHosts do
5.      If target host has enough resources for VM
6.      capacity←getCapacityofHost();
7. If capacity < minCapacity
        minCapacity=Capacity
8.      allocated host←host
9. If allocatedHost≠NULL then
10.     Allocation.add(vm,allocatedHost)
11. Return allocation
```

II. VM allocation policies : Decide if a host is underutilized or overutilized

```
1.Input : ListOfHosts , Output  : Return true if a host is overutilized
2. upperThreshold = 0
3. set upperThreshold = calculateStdDev(historyOfUtizations)
4. totalMipsRequested=0
5.For each Vm in Host do
6.        totalRequestedMips += vm.getCurrentRequestedTotalMips()
7. utilization = totalRequestedMips / host.getTotalMips();
8.Return utilization > upperThreshold
```

III. Vm allocation policies 2 : Return host that is underutilized

```
1.Input : ListOfHosts , Output : Return true if a host is underutilized
2.minUtilization=1
3.underUtlizedHost=NULL
4.For each Host from ListOfHosts
5.        utilization = host.getUtilizationOfCpu()
6.        if (utilization > 0 && utilization < minUtilization
                          && !areAllVmsMigratingOutOrAnyVmMigratingIn(host)) {
            minUtilization = utilization
            underUtilizedHost=host }
7.if underUtilizedHost!=NULL
8. return underUtilized
```

IV. VM selection policy : Choose which VM to migrate if host is overloaded

```
1.Input : Overutilized host, Output : Vm to migrate
2.vmToMigrate = NULL
3.minMetric=MAX
4.For each VM in Host
5.        metric=vm.getRAM()
6.       if(metric < minMetric)
                  minMetric = metric;
                  vmToMigrate = vm;
7.return vmToMigrate;
```

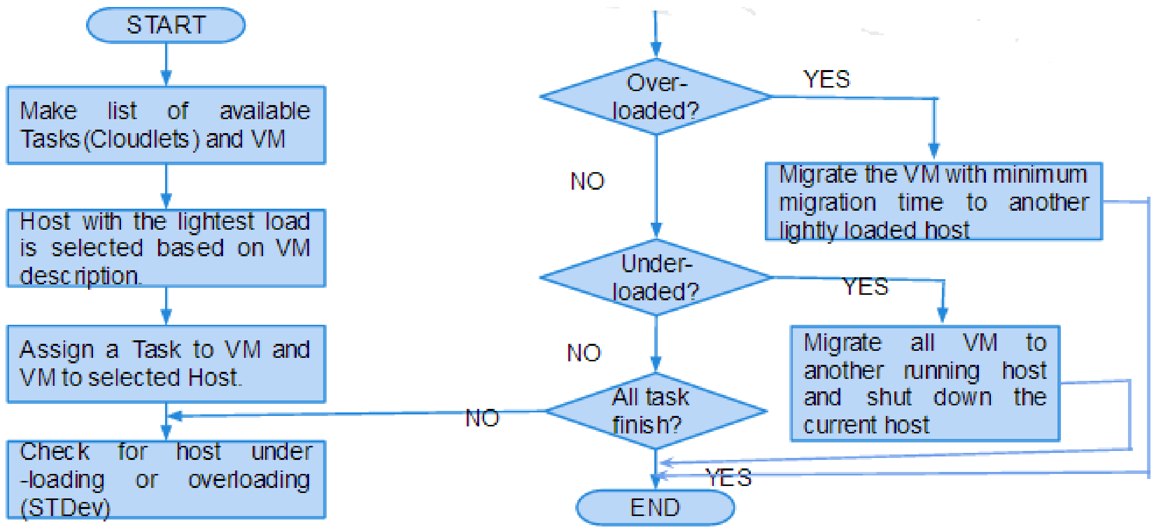
5.2 Design and flowchart

Entire flow:

1. Make a list of available tasks(cloudlets) and hosts.

2. According to the properties of each task(cloudlets), define the properties of the VM that it can be run on.
3. According to the VM description, a host is selected from the list of hosts, which can accommodate the VM, and whose load is lightest.
4. Assign the task to the VM, and the VM to the selected host.
5. Check hosts for overloading or underloading (StDev).
    a. If host is overloaded, migrate the VM with the minimum migration time (VM selection) to another host which is lightly loaded.
    b. If host is underloaded, migrate all the VMs from that host to another host which is running, and shut down the current host
6. Step 5 is repeated till all tasks finish execution

Flowchart :

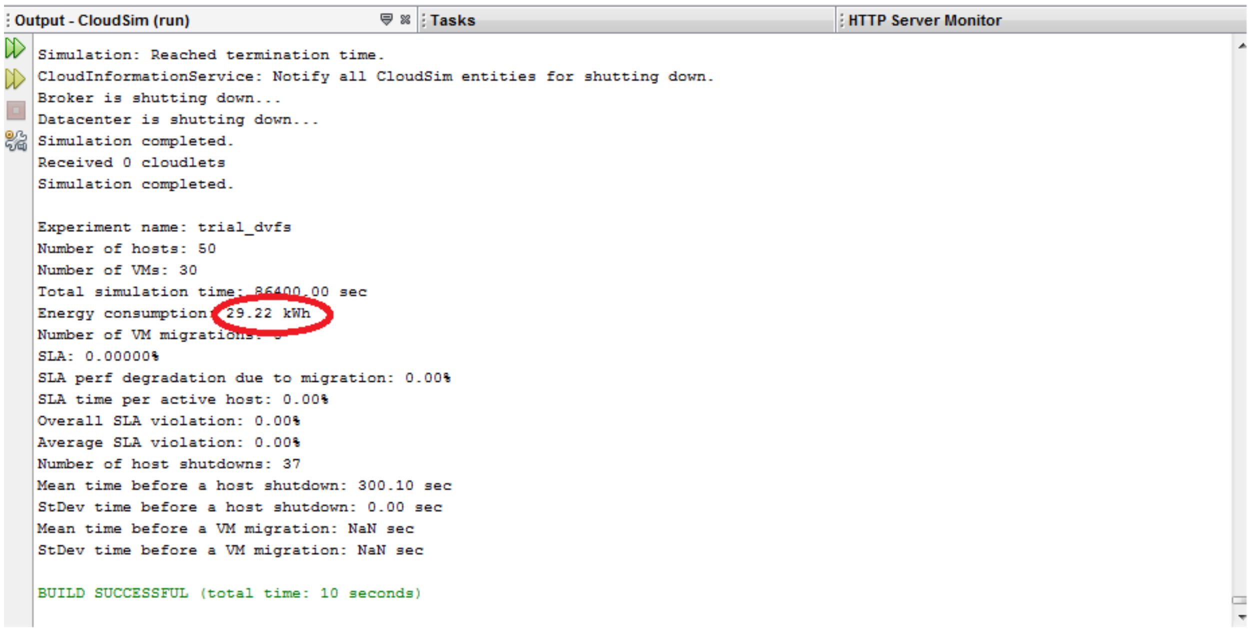

START
Make list of available Tasks (Cloudlets) and VM
Host with the lightest load is selected based on VM description.
Assign a Task to VM and VM to selected Host.
Check for host under -loading or overloading (STDev)
Over-loaded?
YES
NO
Migrate the VM with minimum migration time to another lightly loaded host
Under-loaded?
YES
NO
Migrate all VM to another running host and shut down the current host
All task finish?
NO
YES
END

6. Data Analysis and Discussion:

6.1 Output generation

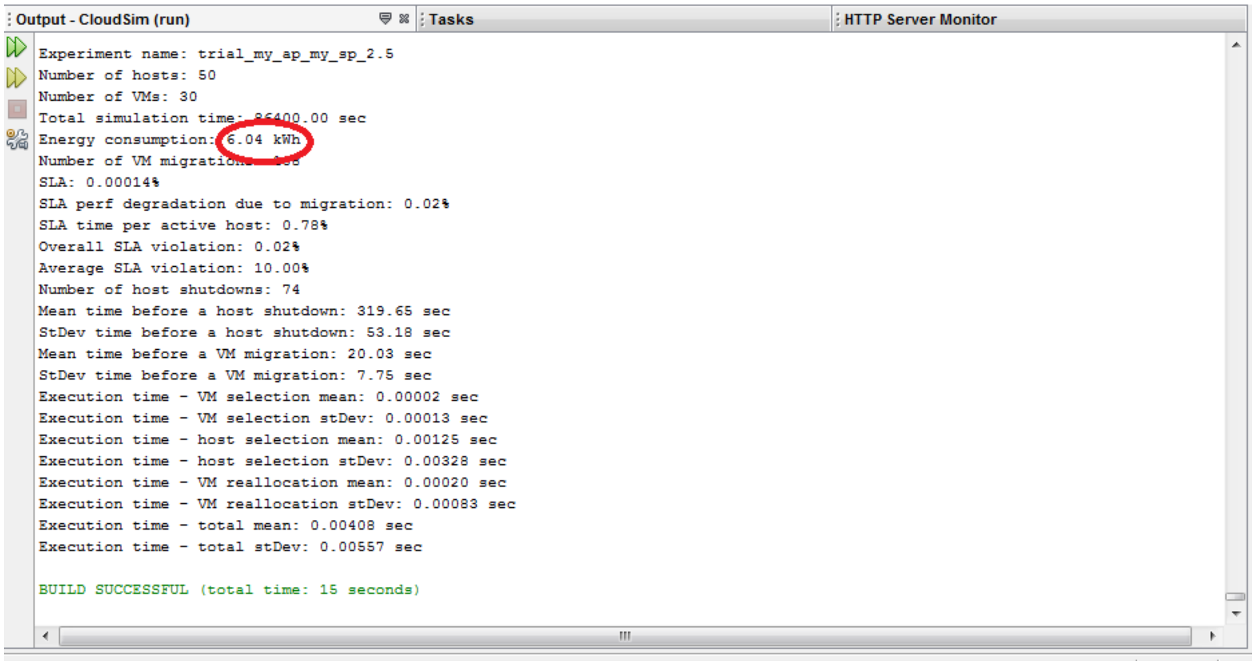
In order to generate the output we ran the implemented algorithm along with existing algorithm such as DVFS. We have used the total energy consumption metric to compare DVFS and our method. Further in order to look into the effects of migration in CLoud we record the SLA violation metric.

We have used traces of **PlanetLab as benchmark** for running all the algorithms.
Below are the screen shots after running of DVFS and our proposed algorithm with the same workload.



```
Simulation: Reached termination time.
CloudInformationService: Notify all CloudSim entities for shutting down.
Broker is shutting down...
Datacenter is shutting down...
Simulation completed.
Received 0 cloudlets
Simulation completed.

Experiment name: trial_dvfs
Number of hosts: 50
Number of VMs: 30
Total simulation time: 86400.00 sec
Energy consumption: 29.22 kWh
Number of VM migrations: 0
SLA: 0.00000%
SLA perf degradation due to migration: 0.00%
SLA time per active host: 0.00%
Overall SLA violation: 0.00%
Average SLA violation: 0.00%
Number of host shutdowns: 37
Mean time before a host shutdown: 300.10 sec
StDev time before a host shutdown: 0.00 sec
Mean time before a VM migration: NaN sec
StDev time before a VM migration: NaN sec

BUILD SUCCESSFUL (total time: 10 seconds)
```

Figure : DVFS Snapshot



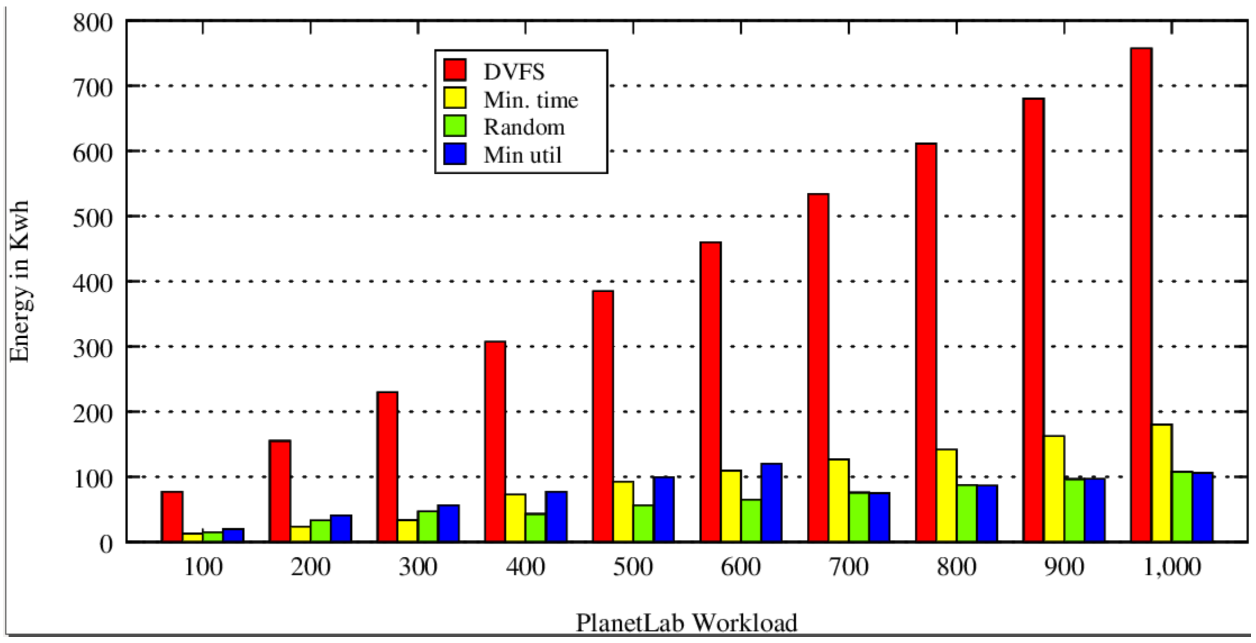
```
Experiment name: trial_my_ap_my_sp_2.5
Number of hosts: 50
Number of VMs: 30
Total simulation time: 86400.00 sec
Energy consumption: 6.04 kWh
Number of VM migrations: [illegible]
SLA: 0.00014%
SLA perf degradation due to migration: 0.02%
SLA time per active host: 0.78%
Overall SLA violation: 0.02%
Average SLA violation: 10.00%
Number of host shutdowns: 74
Mean time before a host shutdown: 319.65 sec
StDev time before a host shutdown: 53.18 sec
Mean time before a VM migration: 20.03 sec
StDev time before a VM migration: 7.75 sec
Execution time - VM selection mean: 0.00002 sec
Execution time - VM selection stDev: 0.00013 sec
Execution time - host selection mean: 0.00125 sec
Execution time - host selection stDev: 0.00328 sec
Execution time - VM reallocation mean: 0.00020 sec
Execution time - VM reallocation stDev: 0.00083 sec
Execution time - total mean: 0.00408 sec
Execution time - total stDev: 0.00557 sec

BUILD SUCCESSFUL (total time: 15 seconds)
```

Figure: Our working algorithm snapshot

6.2 Output analysis

First we compare DVFS with our task scheduling method to look into the amount of energy consumed. Below graph shows the statistics for the following:

1. DVFS algorithm
2. Our solution using - min time as a VM selection during migration
3. Our solution using - random VM selection during migration
4. Our solution using - Min utilization metric for VM selection during migration



From the above graph we can see that our approach saves much more energy than DVFS.

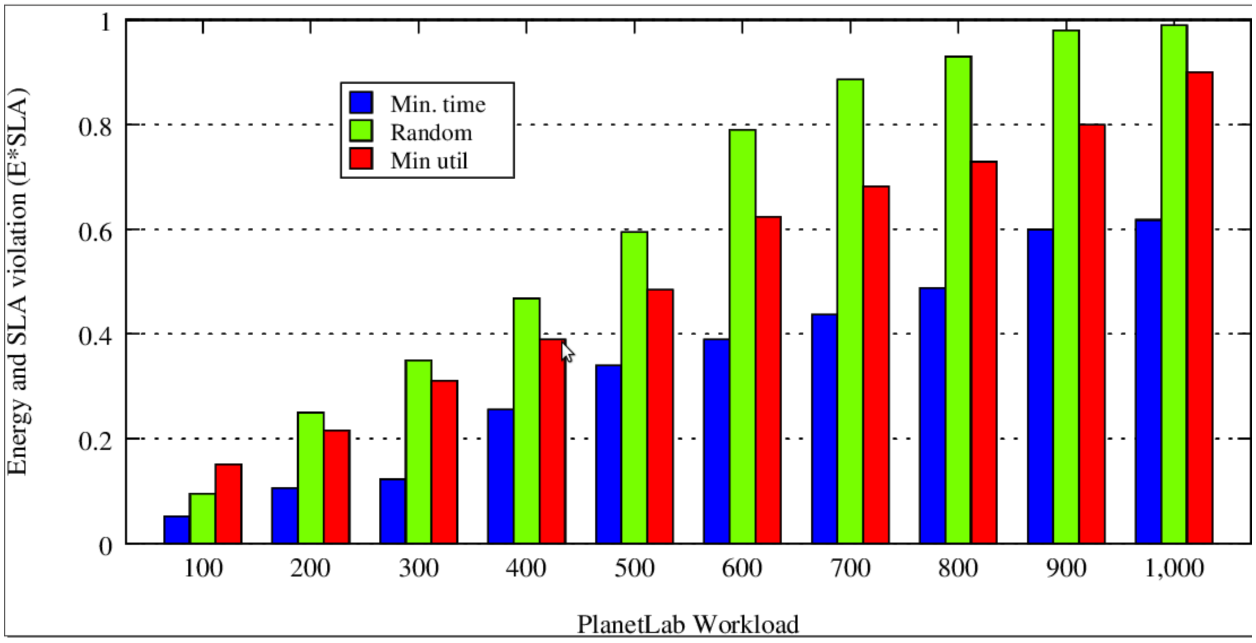
Energy reduction compared to DVFS is around 70-80%. i.e 80% of the power consumption can be reduced.

But it is not clear which among the 3 approaches to choose VM to migrate is doing good. Moreover we need to look at the SLA violations between them.

For this we used the following metric to assess the best method. It is a combined metric that takes both energy consumption and SLA violation into consideration.

Energy-SLA = (Energy consumed * SLA violations)

Any method that can gives us the least value of Energy-SLA can be considered as the best.



After plotting energy consumption and SLA violation , we can see that the min migration time method works the best.

7. Conclusion

7.1 Summary

We have evaluated the proposed two-level task scheduling algorithm through extensive simulations on a large-scale experiment setup using workload traces from more than a thousand PlanetLab VMs. **The results of the experiments have shown that the proposed two-level scheduling policy that does dynamic consolidation of virtual machines uses significantly less energy compared to other task scheduling algorithms that try to conserve energy like DVFS.We were able to achieve an energy reduction of 70-80%**.

Further in order to look into the cost of migrations , we recorded the SLA violations. We conclude that by **migrating VM's with minimum migration time we achieve the least number of SLA violations** . Such algorithms can be used by cloud IaaS providers to reduce power consumption, minimizing SLA violations

7.2 Recommendations for future studies

In order to evaluate the proposed system in a real Cloud infrastructure, we plan to implement it by extending a real-world Cloud platform, such as OpenStack. Another direction for future research is the investigation of more complex workload models, e.g. models based on Markov chains, and development of algorithms that will leverage these workload models.

Besides the reduction in infrastructure and on-going operating costs, this work also has social significance as it decreases carbon dioxide footprints and energy consumption by modern IT infrastructures

8. Bibliography

[1] NoroozOliaee M.; Hamdaoui, B.; Guizani, M.; Ben Ghorbel, M., "*Online multi-resource scheduling for minimum task completion time in cloud servers*," 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp.375-379, April 27 2014-May 2 2014.

[2] Hao Yin, Huilin Wu, Jiliu Zhou, "*An Improved Genetic Algorithm with Limited Iteration for Grid Scheduling*", Sixth International Conference on Grid and Cooperative Computing, pp.221-227, 16-18 Aug. 2007.

[3] Domanal, S.G.; Reddy, G.R.M., "*Optimal load balancing in cloud computing by efficient utilization of virtual machines*", 2014 Sixth International Conference on Communication Systems and Networks (COMSNETS), pp.1-4, 6-10 Jan. 2014

[4] Varalakshmi, P.; Maheshwari, K., "*Cost-optimized resource provisioning in cloud*", International Conference on Recent Trends in Information Technology (ICRTIT), pp.108-112, 25-27 July 2013.

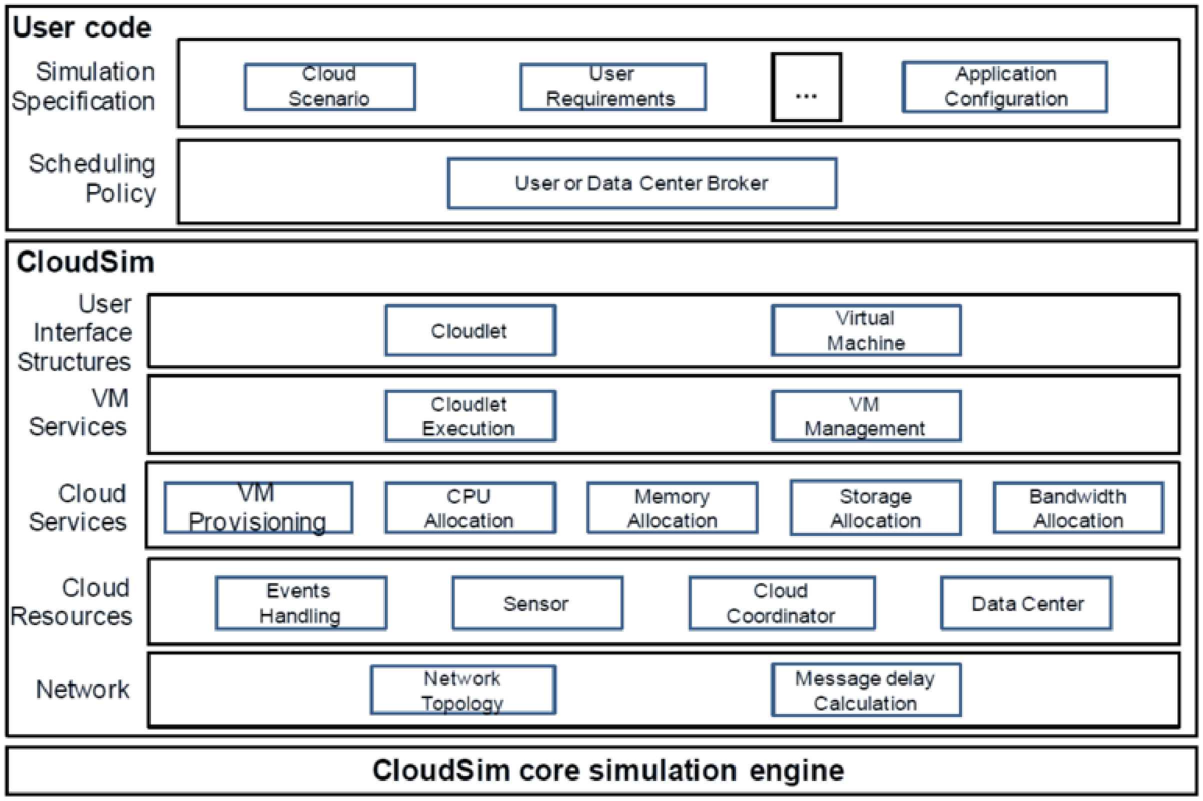
[5] Jia Ru; Keung, J., "*An Empirical Investigation on the Simulation of Priority and Shortest Job First Scheduling for Cloud-Based Software Systems*", 22nd Australian Software Engineering Conference (ASWEC), pp.78-87, 4-7 June 2013.

[6] Tawfeek, M.A; El-Sisi, A; Keshk, AE.; Torkey, F.A, "*Cloud task scheduling based on ant colony optimization*", 8th International Conference on Computer Engineering & Systems (ICCES), pp.64-69, 26-28 Nov. 2013.

9. Appendix:

A. CloudSim Architecture

CloudSim is a simulating program from CLOUDS lab in University of Melbourne for cloud computing. It is developed in java platform including the pre-developed modules such as SimJava and GridSim. The following is the architecture for CloudSim.



The infrastructure-level services (IaaS) related to the clouds can be simulated by extending the Datacenter entity of CloudSim. The datacenter entity manages a number of host entities. The hosts are assigned to one or more VMs based on a VM allocation policy that should be defined by the cloud service provider. A Datacenter can manage several hosts that in turn manage VMs during their life cycles. Host is a CloudSim component that represents a physical computing server in a Cloud: it is assigned a pre-configured processing capability (expressed in millions of instructions per seconds – MIPS), memory, storage, and a provisioning policy for allocating processing cores to virtual machines.

VM allocation (provisioning) is the process of creating VM instances on hosts that match the critical characteristics (storage, memory), configurations (software environment), and requirements (availability zone) of the SaaS provider. By default, VmAllocationPolicy implements a straightforward policy that allocates VMs to the Host in First-Come-First-Serve (FCFS) basis. To implement your own VM Allocation policy, just extends the class VmAllocationPolicy. Power consumption by computing nodes in data centers is mostly determined by the CPU, memory, disk storage, power supplies and cooling systems. Recent studies have shown that the power consumption by servers can be accurately described by a linear relationship between the power consumption and CPU utilization.