Multidiszciplináris tudományok, 12. kötet. (2022) 3 sz. pp. 256-263 **https://doi.org/10.35925/j.multi.2022.3.23**

AN INVESTIGATION ON IMPLEMENTING A SCENARIO ON DIFFERENT CLOUD SIMULATORS

Hasanein Rjeib
PhD student, University of Miskolc, Institute of Information Technology
3515 Miskolc, Miskolc-Egyetemváros, e-mail: rjeib@iit.uni-miskolc.hu

Gábor Kecskeméti
associate professor, University of Miskolc, Institute of Information Technology
3515 Miskolc, Miskolc-Egyetemváros, e-mail: kecskemeti@iit.uni-miskolc.hu

Abstract

Recently, many algorithms have been introduced to provide solutions for Virtual Machine Consolidation in Cloud Data Center. Such Algorithms are usually implemented using many Cloud Simulators, which may have different representation and behavior of cloud infrastructure entities like Data Centers, Physical Machines(PMs), and Virtual Machines(VMs). In this paper, we investigated the impact of using a simulator on the implementation of Cloud infrastructure that is necessary for the Consolidation process, we chose Cloudsim and DISSECTCF simulators in particular since they both support Cloud infrastructure entities. Our aim is to find common entities for cloud simulators that are necessary to build the same infrastructure with particular interest on monitoring the CPU utilization and energy consumption. We report our experience with the implementation on the two Simulators, in addition to the limitation and differences we found during the reproducing process.

Keywords: *Cloud Simulators, Cloud Infrastructure, Physical Machine, Virtual Machine, Energy Consumption*

1. Introduction

Recent research has shown the importance of energy efficient resource utilization of Cloud Data Center, in which Virtual Machines (VMs) can be consolidated into fewer Physical Machines(PMs) to save energy (Khan et al., 2018). This is done by using the virtualization technology, where single PM can host several VMs, providing better utilization of the available resources. In addition, VM live migration helps data center operators to comply with the changes in the workload, where VMs can be consolidated to a fewer PMs in case of the low load, whereas PMs can be turned on so that they can host new VMs and thus improving quality of service (QoS) in time of high load.

Cloud Simulators allow implementation and evaluation of several models in Cloud Computing infrastructure. They provide relatively easy setup for experimentation of many scenarios using several types of PMs and VMs that represent the Core entities of the cloud infrastructure. Every Cloud Simulator has different way of modeling PMs and VMs behavior despite having the same resources (CPU, Memory, Bandwidth, etc.), and also different way of handling power consumption models. Thus, implementing a specific VM consolidation scenario on two different simulators -having same specifications for PMs and VMs with different behavior- might not lead to same results in terms of energy consumption and resource utilization.

The aim of this research is to reproduce the same infrastructure for VM consolidation algorithm on different Simulators trying to answer the following questions: (1) How to ensure that we have a same exact PMs and VMs on two different simulators? (2) Are there any differences in the evaluation results (in particular, Energy consumption and resource utilization) using different cloud simulators?

To answer these questions, we started to reproduce the infrastructure related work done in (Wang and Tianfield, 2018) as they claimed that their Dynamic VM Consolidation plan have the best performance among other works. They chose Cloudsim simulator (Calheiros et al., 2011) - which is the most widely used Cloud Simulator- as their evaluation toolkit.

For our re-implementation, We selected DISSECT-CF simulator (Kecskeméti, 2015) which provide similar cloud infrastructure entities like PMs and VMs with very accurate results regarding the energy consumption. In addition, DISSECT-CF provides more efficiency and better scalability of simulation time compared to Cloudsim. The rest of this work is organized as follows. Section 2 discuss previous work. In section 3, we discuss our experimentation and re-implementation process regarding PM and VM creation including the changes we made in order to get identical result between the two simulators. In section 4, we present our evaluation process and the result we got regarding energy consumption and the host utilization. Finally, section 5 concludes the paper.

2. Related work

A number of research papers have been re-implementing works on different simulators trying to investigate the effect of a simulator on implementation process.

Mann (Mann, 2018) described their experience with porting a VM placement algorithm and its evaluation setup from one cloud simulator to another. they proposed a layer of abstraction for implementing VM allocation algorithm using Planetlab workload. However, they didn't consider having the same power model of PMs in both simulators. The authors in (Bahwaireth et al., 2016) compared several simulation tools capabilities by applying different scenarios, however, they didn't consider implementing the exact setup among the simulators. Most of the research done didn't consider having identical Cloud infrastructure between simulators.

A comparative analysis of many tools for cloud environments have been presented by (Bambrik, 2020). They compared the most used simulators in terms of the supported model, architecture, and high-level features. However, they didn't consider the internal behavior of cloud entities between simulators. The authors in (Mansouri et al., 2020) provided a detailed survey about the existing Cloud simulators, discussing the features and software architecture of several simulators. However, they didn't discuss the effect of different behavior of the simulators on algorithms implementation.

The authors in (Di and Cappello, 2015) tried to reproduce Google cloud environment with real experimental system setting and real-world large scale production trace. They have shown that simulation system could effectively reproduce the real checkpointing/restart events based on Google trace by leveraging Berkeley Lab Checkpoint/Restart tool (Hargrove and Duell, 2006). Nevertheless, they didn't compare their result with different simulator.

Some of the bin packing solutions to address VM placement problems have been implemented by Chowdhury et.al. (Chowdhury et al., 2015). In order to do so, they have followed exact similar procedures for VM allocation to detect both underloaded and overloaded hosts and VM selections tools for selecting VMs which are needed to be migrated from those hosts as discussed in (Beloglazov et al., 2012). However, they haven't try different simulation tools and sticked to the cloudsim for their re-implementation.

For our study, we investigated the differences of requirements needed to re-implement an identical infrastructure from one cloud simulator (Cloudsim) to another (DISSECT-CF). This includes cloud infrastructure entities such as PMs, VMs, and the Tasks/clouldlets, in addition to the power models and the running time for the cloud entities.

3. Experimentation

Our plan is to reproduce a cloud infrastructure (PMs and VMs) on both simulators with same specifications (CPU cores, Memory,and Bandwidth). In order to ensure that PMs are identical, we expect to have the same energy consumed by running identical task on VM. This VM consumes similar percentage of the PM resources on both simulators for specific period of time. We implemented a PM with same specification as the Hp ProLiant Ml 110 G4 which has two CPU cores, each with 1860 MIPS, 4 GB RAM, and 1 GB bandwidth. For the workload data, we have implemented new mechanism in DISSECT-CF simulator so that it can load the same PlanetLab (Beloglazov et al., 2012) (more details can be found in section 3.3) workload data in Cloudsim. Eventually, we expected to have the same amount of energy consumed for the PM considering we have launched identical task to run inside VM with same resource utilization percentage.

3.1. PM Creation

Many differences have been observed during the implementation of PMs on both simulators. In order to implement a Physical Machine in Cloudsim (called *Host*) while getting results regarding energy and utilization, we have used Cloudsim power package. We first created a Data center (*PowerDataCenter* class) object in which we could add PM (*PowerHostUtilizationHistory*) to it. For Host creation, we needed to specify the ID, RAM, Network bandwidth, storage, number of CPU, and power model. Fortunately, Cloudsim has power model called *PowerModelSpecPowerHpProLiantMl110G4Xeon3040* which reflects the energy consumption of the Server according to the CPU Utilization percentage. For DISSECT-CF, creating PM (*PhysicalMachine*) was more complicated than Cloudsim (DISSECT-CF tries to imitate real life Cloud infrastructure in more detail) as we needed to create a *Repository* object which represents the Disk, connected to network, and defining a power model for the power characteristics descriptions. Also, DISSECT-CF defines three consumption models (CPU, Memory, and network) inside the power model of a PM.

To reflect the same behavior of the *PowerModelSpecPowerHpProLiantMl110G4Xeon3040* model defined in Cloudsim, we have created a separate consumption models for each of the CPU, Memory, and network while making sure that the total amount of energy consumed on the server is identical to the HP Proliant server power model in Cloudsim where the power model is calculated with respect to the overall utilization of the Host. DISSECT-CF doesn't allow querying energy directly from the PM (Cloudsim does), instead, we have created a dedicated *PhysicalMachineEnergyMeter* and linked it to *PM* so we could have a reading of the PM's energy consumption. This meter can be started once the PM is turned ON and it could be stopped once a certain job finishes.

3.2. VM Creation

VM in Cloudsim has the following specifications: ID, Million Instructions Per Second (*MIPS*), Image size, bandwidth, number of cores, and task scheduler. Creating a VM in DISSECT-CF can be done by

invoking *requestVM* in the *PhysicalMachine* object. *requestVM* requires a *VirtualAppliance* representing the functional virtual machine images in the system, as well as a resource constraints representing the amount of resources the VM use compared to the hosting PM.

Cloudsim has separate classes for VM and the task to be run. To launch a task in the VM, one can create 2 separate objects for the VM (*Vm*) and for the task (*Cloudlet*) and then it is the responsibility for *Broker* to submit task to VM. In contrast, launching a task in DISSECT-CF can be done by calling the *newComputeTask* method in which it specifys the task length, processing limit, and *ConsumptionEventAdapter* providing basic functions to determine if a resource consumption has already been completed. Once a task is assigned to VM, it's not possible for the VM to change its utilization, to have a VM with varying utilization, we needed to make sure that the task finishes in specific period of time so we can launch another task with different utilization. This is done easily in Cloudsim as it can instruct the VM to change its utilization.

In order to have the same behavior for the VM in DISSECT-CF, we have created a VM and control the total number of instructions to be executed considering the processing capability (for instance, a task with 5 Million Instructions would take 5 seconds to finish if it is run on a VM with 1 Million instruction/ second power) so that VM can run for specific period of time with identical utilization percentage as if it was running in Cloudsim simulator.

The next step was to run several PMs and VMs to check the overall energy consumed by the simulation. In order to do that, we have implemented the same workload trace -existed in Cloudsim- in DISSECT-CF simulator so that PMs and VMs have the same power model and the same data for running. Section 3.3 discusses the procedure of the workload trace loading mechanism in DISSECT-CF.

3.3. Loading the workload trace

One of the advantages of Cloudsim that attracts many researcher is that it has a builtin workload traces (PlanetLab workload) (Park and Pai, 2006). It contains information from 10 days about CPU usage for around 1000 VMs, these information can be found in *examples/workload/planetlab* folder in Cloudsim. The CPU load data are stored as simple text files in which each file contains 288 values reflecting the CPU utilization of one VM for a day. Thus, each value in a file representing a CPU utilization taken every 5 minutes.

Beloglazov et al. (Beloglazov and Buyya, 2012) have made some arrangements so they could evaluate their algorithm with realistic data for testing. They have implemented *UtilizationModelPlanetLabInMemory* class for the cloudlet utilization model which reads the utilization values from a file. For their experimentation setup such as data center creation, cloudlet creation from file based utilization model, and setting the VM consolidation algorithm, and starting the simulation, They have made the *PlanetLabRunner* class with some helper classes (*PlanetLabHelper* and *PlanetLabConstants*) to provide parameters for their experiment. These parameters include the name of the folder corresponding to a specific date of the PlanetLab data in which the folder consists of many files contain the CPU values for a VM. Finally, they have created *helper* class to set up PMs and VMs based on the data in the *constants* class.

In order to use the PlanetLab data provided by Cloudsim in the DISSECT-CF simulator, we have created 2 new classes, *PlanetLabFolderReader* class in which it is responsible for choosing the experiment date (PlanetLab contains data from 10 days, the data of each day is saved in a folder). And then it is the responsibility of the class *PlanetLabFileReader* to open the files inside the folder and create a job for every value in the files. This class implements the *CreateJobFromLine* method in the

DISSECT-CF simulator. Figure 1 shows a brief description of the Trace loading mechanism. The *Job* (this class is responsible for creating the jobs/tasks from workload trace in DISSECT-CF) contains the following infromation:

- Start time of the job: Since we have many files, each with 288 values representing the cpu load for one VM for every 5 minutes for a complete day (24 hours), we have given each job different starting time with a 300 second (5 minutes) gab between any two succissive jobs (for example, if the first job start time is at the first second, the second job starts at 301 second, the next is 601 sec. and so on) so that each job can run on the same VM once the previous job finishes.
- Type of the job: the PlanetLabFileReader class will insure that all the 288 jobs to be created have the same executable (this parameter is part of the Job class in the DISSECT-CF simulator) value so that they could all run on a specific VM later (DISSECT-CF have VMSetPerKind map in which it bond a VM type to certain type of jobs).
- Other information: we have given every job a 300 second for execution time, and we assumed that all the data is running by a single user, and all the jobs are homogeneous (take the same amount of RAM, number of processors, etc.)

Trace loading mechanism

```
1: Open PlanetLab Directory
2: Choose specific day for loading data
3: For each file inside the folder
4:     While the file has Line:
5:         Construct a Job from the line
6:         Set up the submission time and the Executable value
7: Add the Job to a List of jobs
```

Figure 1. *Trace loading mechanism*

4. Evaluation

Time is measured in seconds in CLoudsim, while it is measured in Ticks in DISSECT-CF. We set up our simulations so that one Tick equals to 1 millisecond (users have free interpretation of the Tick). For the Simulation, we created *DataCenter* in Cloudsim having one *Host* with the same specification as HP Proliant G4 server (2 cores, 1860 MIPS, 4 GB memory). We have created a single *VM* consuming (50% and 100% respectively) of the whole resources of PM in two different scenarios, and we ran a task on the VM for 1, 10, and 30 minutes respectively. The scheduling interval was set to 300 ms. During the Simulation, we saved the result in terms of energy consumed by the PM in accordance with the resource utilization for later comparison with the DISSECT-CF. Next step was to re-implement the same setup on DISSECT-CF in order to compare the result gained. We have created the *PhysicalMachine* with exact specification of the CPU, Disk, network bandwidth, assigning the same power model as we discussed earlier in section 3.1.

For energy metering in Cloudsim, the power consumption can be queried by the *Host* object itself as it has a builtin method that provides the energy consumed (in real life, it is not possible to ask a physical machine how much energy did it consume!). While in DISSECT-CF, we have instructed the simulator to start a separate energy meter to record the consumption on the PM once it has its VM running (in real life, it is similar to attach a power meter to a physical machine's power supply). Thus, we can avoid the energy and time consumed for Switching PMs and VMs On/Off due to the fact that DISSECT-CF tries to imitate the real life behavior.

Also, we instructed the simulator to stop at a certain time specified for the simulation comparison (1, 10, and 30 minutes), this is done by using the *Tick* method in *Timed* class. The results obtained were identical on both simulator in terms of the energy consumption and resource utilization which proves that our implementation were identical on both Simulators. Table 1 shows the energy consumed by running a VM consuming 50% of the PM resources on both simulators, while Table 2 provide the energy consumption by running VM on 100% of PM resources.

Finally, we tested our new loading mechanism by running the experiment several times in order to insure that the jobs-to-VM mapping is done the same way as it is done in Cloudsim. We have observed VM's logs during simulation to check that all the data in each file of the PlanetLab workload goes to the same VM as it does in Cloudsim.

Table 1. *Energy consumed by running a PM at 50% utilization*

Execution (real) Time	**Energy Consumption**		**Simulation Time (ms)**	
	Cloudsim	**DISSECT-CF**	**Cloudsim**	**DISSECT-CF**
1 minute	6.09 e6	6.09 e6	58	50
10 minutes	6.12 e7	6.12 e7	145	132
30 minutes	2.09 e8	2.09 e8	361	287

Table 2. *Energy consumed by running a PM at 100% utilization*

Execution (real) Time	**Energy Consumption**		**Simulation Time (ms)**	
	Cloudsim	**DISSECT-CF**	**Cloudsim**	**DISSECT-CF**
1 minute	8.05 e6	8.05 e6	67	60
10 minutes	8.10 e7	8.10 e7	164	148
30 minutes	2.43 e8	2.43 e8	478	403

5. Conclusion

In this work, we described our experience with reproducing the same infrastructure in two cloud simulators. We highlighted the differences between two simulators in terms of infrastructure and the

behavior of cloud entities. And we have disscussed the changes to be made that are necessary to have the same exact setup regarding infrastructure considering the different behavior of cloud entities between cloud simulators. Finally, we have compared the performance of cloud systems in both simulators with regard to energy consumption and resource utilization. We will continue investigating the complete re-implementation on the whole VM Consolidation Scenario trying to find what's needed to be done to have identical results in both simulators. We will also extend our work done by implementing new VM placement and VM selection algorithms for a scalable cloud infrastructure. Finally, we will provide a complete study on the whole VM consolidation scenario implementation on different cloud simulators.

References

[1] Khan, M. A., Paplinski, A., Khan, A. M., Murshed, M., & Buyya, R. (2018). Dynamic virtual machine consolidation algorithms for energy-efficient cloud resource management: a review. *Sustainable cloud and energy services*, 135–165. **https://doi.org/10.1007/978-3-319-62238-5_6**

[2] Wang, H., & Tianfield, H. (2018). Energy-aware dynamic virtual machine consolidation for cloud datacenters. *IEEE Access*, 6, 15259–15273. **https://doi.org/10.1109/ACCESS.2018.2813541**

[3] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1), 23–50. **https://doi.org/10.1002/spe.995**

[4] Kecskemeti, G. (2015). DISSECT-CF: a simulator to foster energy-aware scheduling in infrastructure clouds. *Simulation Modelling Practice and Theory*, 58, 188–218. **https://doi.org/10.1016/j.simpat.2015.05.009**

[5] Mann, Z. Á. (2018). Cloud simulators in the implementation and evaluation of virtual machine placement algorithms. *Software: Practice and Experience*, 48(7), 1368–1389. **https://doi.org/10.1002/spe.2579**

[6] Bahwaireth, K., Tawalbeh, L. A., Benkhelifa, E., Jararweh, Y., & Tawalbeh, M. A. (2016). Experimental comparison of simulation tools for efficient cloud and mobile cloud computing applications. *EURASIP Journal on Information Security*, 2016(1), 1–14. **https://doi.org/10.1186/s13635-016-0039-y**

[7] Bambrik, I. (2020). A survey on cloud computing simulation and modeling. *SN Computer Science*, 1(5), 1–34. **https://doi.org/10.1007/s42979-020-00273-1**

[8] Mansouri, N., Ghafari, R., & Zade, B. M. H. (2020). Cloud computing simulators: A comprehensive review. *Simulation Modelling Practice and Theory*, 104, 102144. **https://doi.org/10.1016/j.simpat.2020.102144**

[9] Di, S., & Cappello, F. (2015). GloudSim: Google trace based cloud simulator with virtual machines. *Software: Practice and Experience*, 45(11), 1571–1590. **https://doi.org/10.1002/spe.2303**

[10] Chowdhury, M. R., Mahmud, M. R., & Rahman, R. M.: *Study and performance analysis of various VM placement strategies*, 2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD) (pp. 1-6). IEEE. **https://doi.org/10.1109/SNPD.2015.7176234**

[11] Beloglazov, A., Abawajy, J., & Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems*, 28(5), 755–768. **https://doi.org/10.1016/j.future.2011.04.017**

[12] Park, K., & Pai, V. S. (2006). CoMon: a mostly-scalable monitoring system for PlanetLab. *ACM SIGOPS Operating Systems Review*, 40(1), 65–74. **https://doi.org/10.1145/1113361.1113374**

[13] Beloglazov, A., & Buyya, R. (2012). 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*, 24(13), 1397–1420. **https://doi.org/10.1002/cpe.1867**

[14] Hargrove, P. H., & Duell, J. C. (2006). Berkeley lab checkpoint/restart (blcr) for linux clusters. *Journal of Physics: Conference Series*, 46(1), p. 067. IOP Publishing. **https://doi.org/10.1088/1742-6596/46/1/067**