ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/329061959

CoolCloudSim: Integrating Cooling System Models in CloudSim

Conference Paper · September 2018

DOI: 10.1109/ICCP.2018.8516648

CITATIONS
5

READS
163

7 authors, including:

Marcel Antal
Universitatea Tehnica Cluj-Napoca
70 PUBLICATIONS **1,522** CITATIONS
SEE PROFILE

Claudia Daniela Antal
Universitatea Tehnica Cluj-Napoca
58 PUBLICATIONS **1,437** CITATIONS
SEE PROFILE

Tudor Cioara
Universitatea Tehnica Cluj-Napoca
143 PUBLICATIONS **2,080** CITATIONS
SEE PROFILE

Ionut Anghel
Universitatea Tehnica Cluj-Napoca
141 PUBLICATIONS **2,070** CITATIONS
SEE PROFILE

All content following this page was uploaded by Tudor Cioara on 05 February 2020.

The user has requested enhancement of the downloaded file.

CoolCloudSim: Integrating Cooling System Models in CloudSim

Abstract— **This paper addresses the problem of Data Centers (DC) energy efficiency from a thermal perspective by extending the CloudSim framework to allow simulation and testing scenarios of thermal aware resource allocation policies aiming to minimize the cooling system energy consumption. The proposed framework, CoolCloudSim, can be used to develop and test new thermal aware VM allocation strategies aiming to optimize the energy consumption of both cooling system and IT resources while meeting SLAs. The default CloudSim architecture is extended with mathematical models of the thermal processes within the server room. Furthermore, four new Virtual Machine (VM) allocation policies that consider cooling system energy consumption are developed based on the cooling system models. Finally, experiments are run to evaluate various metrics on a set of default CloudSim allocation algorithms vs. the proposed allocation algorithms. The results have shown that the proposed algorithms outperform the default CloudSim allocation strategy, Power Aware Best-Fit Decreasing (PABFD), in terms of overall energy consumption and the number of VM migrations and have on average better results than other existing allocation strategies.**

Keywords—Data Center; Energy Efficient; CloudSim; Thermal-Aware; Cooling System; Consolidation; Simulation;

I. INTRODUCTION

Datacenters (DCs) become of uttermost importance nowadays due to their huge computing power used to run services for almost every domain of the modern society. In this industry that continues to grow, studies predict that the energy consumption increase will be more than 400%, value recorded in the last decade [5], the IT equipment and the cooling system consume more than 80% of the total energy consumption. Furthermore, the server room density continues to rise [16] and the cooling system can consume up to 37% of the total energy consumption even in the best designed DCs [17]. Thus, a new trend in DC research and design is to actively manage the cooling system by intelligently deploying workload on servers to maximize the server room ambient temperature and minimize the cooling system energy consumption while meeting the clients QoS.

An important tool in DC development and management are the simulation tools that allow testing various algorithms, policies and strategies without impacting the delicate DC ecosystem. Such a tool is the CloudSim [1] simulator, widely used in the literature since its development in 2009 due to its complex and accurate modeling and simulation of tasks as well as energy consumption. A major drawback of the CloudSim simulator is the lack of simulation of the cooling system, a major energy consumption equipment of the DC that intelligently managed can improve the energy efficiency of the DC.

The paper proposes an extension of the CloudSim architecture by integrating cooling system models and thermal aware allocation policies to increase the energy efficiency of the DC. The main objectives of the paper are the following:

- Integration of a thermal model for cooling system management in the CloudSim architecture
- Develop VM policy allocations that consider cooling system power consumption and while maximizing the defined SLAs

The rest of the paper is structured as follows: Section II shows related work, Section III presents the extended CloudSim architecture, Section IV describes adaptive heuristics for thermal aware consolidation, Section V presents results obtained in a simulated environment, while Section VI concludes the paper

II. RELATED WORK

A key element in DC physical and algorithm design consists of special tools and simulators that help improve management algorithms as well as physical design, such as airflow and heat recirculation in order to develop highly efficient computing and cooling systems. Within the requirements of a DC simulator we identify the need to accurately model the complex environment within a DC and to keep track of important metrics and agree on physical and temporal scale of models.

CloudSim [1] offers the ability to simulate a virtualized data center cloud. It provides virtual machine provisioning features, system status monitoring, cloudlet execution executed on virtual machines, CPU, RAM, bandwidth, host-storage capability, VM, and Cloudlet. Users can create physical specifications for the applications that would run through the Cloudlet, and can define the physical characters for virtual machines and for hosts. Each host has defined power models currently used in DCs, according to real data obtained by experiments. Using CloudSim, cloud providers can test different policies for virtual machine allocation on servers, expanding the functionality they offer. They can also use and extend various quality assurance policies in the data center (QoS). Various physical infrastructure configurations can be created, combining them with different types of applications that would be run at different time intervals depending on

existing dispositions and using various virtual machine allocation / selection techniques applied to different levels.

Many other simulators are based on CloudSim [1], an event driven simulator with high extensibility due to its OOP design, built upon the core engine of grid simulator GridSim [2]. Some well-known extensions of CloudSim are: CloudAnalyst [3], NetworkCloudSim [7]. The simulator CloudSimSDN [9] allows simulation of policies for both computational and network resources allocation. It is built on top of CloudSim and aims to evaluate resource management policies. It simulates cloud data center, physical machines, switches, network links and virtual topologies to guarantee QoS, environment conservation and cost-reduction. The framework also does VM consolidation (best fit and worst fit policies) and traffic prioritization (priority traffic consumes more bandwidth than normal traffic). Because the CloudSim simulator does not have the feature for file processing, cost or time, another simulator was built on the CloudSim framework, MR-CloudSim [8]. This new simulator implements the MapReduce paradigm. None of the previously defined simulators define cooling system models. A complex simulator that defines such a model is proposed in [10] and is built upon the BlueSim CFD simulator and the Cyber Physical Simulation Engine (CPSE) that generates very fast predictive model of the DC based on resource management. The resulting tool has the ability to evaluate cyber performance (SLA violations, throughput), physical performance (energy usage, temperature), as well as sustainability metrics (PUE, energy reuse effectiveness). However, due to the Computational Fluid Dynamic component, it requires much more computational resources than CloudSim being unsuited for fast simulations.

Our paper builds upon the state-of-the art by integrating a cooling system model similar to the one developed in [11] and [12] in the CloudSim framework and developing allocation policies to improve the energy consumption by considering cooling system demand and thermal aware allocation strategies.

III. Extending the CloudSim architecture by integrating the Cooling System

This section presents the mathematical formulae of the cooling system model and the extension of the CloudSim architecture to integrate the proposed model.

A. Cooling System Model

The cooling system model was adapted from [11] and our previous work [12]. Thermal models are generally based on the first law of thermodynamics, the law on energy conservation, Newton's cooling law, and Fourier's law of conductivity. In general, the elements to be considered in a data center from a thermal point of view are the outlet / exhaust temperatures and the processor temperature of the servers. In [13] this generalized model is presented for both server exit temperature and processor-level temperature. Therefore, based on the duality between the thermal and electric phenomenon, the heat transferred by the thermal resistance of the processor is represented in equation (1), where C is the thermal capacity, $\frac{dT_{cpu}}{dt}$ represents the temperature raise of the CPU, T_{CPU} is the CPU temperature and T_{amb} is the room temperature, R is the thermal resistance of the CPU and P_{CPU} is the power consumption of the CPU.

$$P_{CPU} = C * \frac{dT_{cpu}}{dt} + \frac{T_{cpu} - T_{amb}}{R} \quad (1)$$

The *Lumped RC Thermal Model* [13] is developed from equation (1):

$$T_{CPU}(t) = P_{CPU} * R + T_{amb} + (P_{CPU} * R + T_{amb} - Tcpu(0))e^{\frac{-t}{R*C}} \quad (2)$$

The server internal temperature is correlated to the inlet/outlet temperature and the heat produced by executing the workload, heat equal to the power consumed:

$$T_{out} = \frac{P_{server}(\mu_s)}{K} + T_{in}, \quad (3)$$

where μ_s represents the percentage of server usage and *K*, the absorption capacity of the air is a constant defined as it follows:

$$K = \rho f C_p, \quad (4)$$

where ρ represents the air density, *f* the air flow, and C_p is the air specific heat.

The problem of determining T_{in} is complex and has been studied in [11]. In this paper we will summarize how to determine how energy is consumed by the cooling system. It is based on the Performance Coefficient of the system described here:

$$COP(T_{sup}) = a * T_{sup}^2 + b * T_{sup} + c \quad (5)$$

T_{sup} is the supply temperature used in the data center (supply temperature), a, b and c are experimentally determined constants. Featuring this metric, the energy consumed for cooling is defined as follows:

$$P_{cooling} = \frac{P_{IT}}{COP(T_{sup})} \quad (6)$$

B. Extended CloudSim Architecture

The first objective of this paper was to integrate the cooling system into the architecture of the CloudSim framework.

The cooling system model is based on the model described in [11] and an independent module "*CoolingSystemAbstract*" was introduced. Based on this and other existing classes within the framework, an extensible model has been developed and tested alongside with the existing power models.

To achieve this, the cooling system used in the ASU HPCI data center described in [14] was used. It shows real thermal influences between servers that have a classical placement on a Hot-Aisle / Cold-Aisle topology with servers located in racks of 10. At the same time, other specific elements needed for a complete modeling are presented, such as the IT infrastructure and the volume of air flow introduced into the room by the room's cooling system.

In the architecture from Fig 1, using the factory method from *PowerCoolingSystemFactory,* based on typed defined in the *PowerCoolingSystemType*, one of the two models of real cooling systems is instantiated. These contain specific features such as thermal influences, *COP*, *airflow*, *Tamb* (ambient temperature), infrastructure elements and topology. These are taken over and used by *PowerCoolingSystem* which provides support for calculating the cooling system power consumption.

This is calculated based on modeling performed in *CoolingSystemAbstract* class. *CoolingSystemAbstract* is responsible for the complex airflow modeling that takes place in the datacenter to determine the temperature at server input and output. We add the temperature modeling at the processor level based on the *"Lumped CPU Thermal Model"* [13].

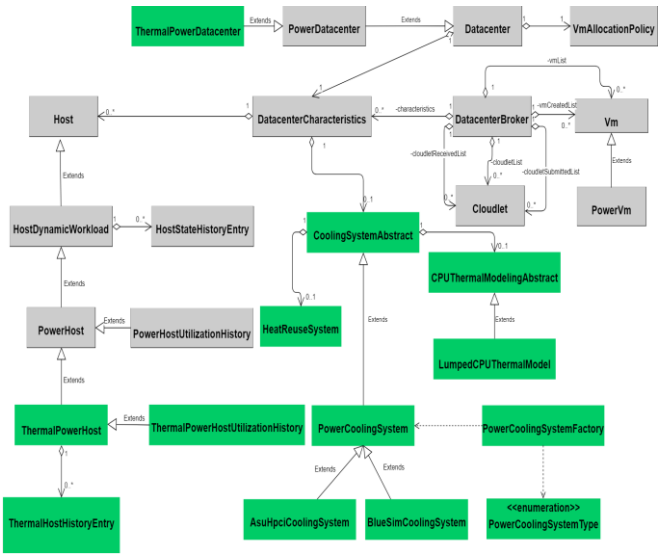

Fig.1 - CoolCloudSim architecture

For a flexible switch between a model based solely on IT electricity consumption and a model based on both IT consumption and electricity consumption needed for the cooling system, we decided at an architectural level to expand the specific entities in the *"power package"* of CloudSim. So we introduced the following entities: *ThermalPowerHost* - extends *PowerHost* and adds "T_{in}", inlet air temperature of the server "T_{out}", server evacuation temperature, "T_{CPU}", server-side temperature, "*thermalStateHistory*" it contains a list of historical thermal values collected during the simulation at the host level. The objects created in "*ThermalHostHistoryEntry*" class have the role of retaining specific time quantities of the host's thermal values, such as "T_{in}", "T_{out}", "T_{CPU}", the current time and an "*isActive*" variable that indicates whether or not the host is active at that time. Another added class is *ThermalPowerHostUtilizationHistory* that has the purpose of retaining historical usage at a *ThermalHost* based on VMs allocated along the simulation on the host. Finally, the *ThermalPowerDatacenter* class extends the datacenter power and adds the function needed to calculate the energy consumed by the cooling system and the total energy consumed by IT and cooling. Besides this, because Datacenter is the class that manages physical resources, at this level there is continuous updating of the thermal values at the server level, so they are current at every moment, and algorithms based on thermal values can be developed by modifying the policy allocation. Each Datacenter has an allocation policy that provides features such as finding a virtual machine host, optimizing allocation, finding a virtual machine host based on a scoreboard, detecting underutilized and overused hosts, keeping track of server usage, and so on. To develop new allocation policies, the *VmAllocationPolicy* class should be extended.

IV. Adaptive Heuristics for Dynamic Workload Consolidation

This section describes how the consolidation process works in CloudSim and illustrates several proposed allocation policies for optimizing the VM placement. The main objective of our algorithms is to minimize the power consumption and SLA violation of the data center, while being aware also about the servers' CPU thermal levels.

A. Default CloudSim Consolidation Model

In CloudSim, the consolidation process is divided into four steps. First, determine overused hosts. Then, from the overused hosts, select the virtual machines that must be migrated through a dedicated selection method. Step three identifies underused hosts and automatically adds all virtual machines on them, to the list of VMs that must be redistributed. After the VMs on the underused servers have migrated, these servers can be turned off. The fourth step and the most important is the allocation of virtual machines selected on other hosts. This is the part exploited in this chapter where we propose multiple VM placement solutions.

i. Detection of overloaded host

Initially, the authors of the CloudSim framework proposed a detection of overuse based on a static threshold. Because the workload is unpredictable and dynamic, this method did not work well and therefore the authors proposed [15] a self-adjusting upper bound, based on the analysis of the data used during the simulation. The developed techniques are based on CPU deviation [15]:

TABLE I. HOST OVERLOADING DETECTION POLICIES

Policy name	**Description**
Median Absolute Deviation (MAD)	Is a measure of dispersion, more robust than simple standard deviation because it is not heavily influenced by outliers.
Interquartile Range (IQR)	Is a metric of statistical dispersion and is calculated as the difference between the third and first quartile of data.
Local Regression (LR)	Is based on Loess method. The orginal data si approximated by clustering simple models in localized data subsets.
Robust Local Regression (LRR)	Presents the advantage, over the previous method, that is not so vulnerable to outliers. In order to make a robust evolution of the algorithm, the "bi-square" estimation method was added.

ii. VM selection policies

From the overused hosts we have to choose a virtual machine to be migrated. This procedure is repeated iteratively until the host is no longer overloaded. In CloudSim, there are three default policies for selecting a VM to migrate [15], as shown in Table II.

TABLE II. VM SELECTION POLICIES

Policy name	Description
Minimum Migration Time (MMT)	The VM that requires the minimum time to migrate will be moved on another host. This time is calculated based on the RAM used, divided by the bandwidth available of the host on which it is running.
Random Choice	A random VM running on the host is chosen.
Maximum Correlation	Is based on the idea that a large correlation between the resources used by the applications running on a server means a higher probability of overuse of a server. Consequently, there are selected to be migrated, those VMs that have the highest correlation of CPU usage with other VMs. Method "Multiple Correlation Coefficient" is used.

iii. CloudSim VM placement strategy

The problem of virtual machine allocation on servers is reduced to a bin packing problem where the objects to be packed are the VMs, and the bins are the servers. For a server to be considered a valid host for the VM, it must meet all the required constraints, from the point of view of the CPU, RAM, bandwidth and storage. In CloudSim, an adaption of the well-known Best-Fit Decreasing algorithm has been chosen as the VM allocation policy. The algorithm named Power Aware Best-Fit Drecreasing (PABFD) brings the improvement of sorting the VMs in descending order by the CPU usage, and assigning each VM to the host that has the least increase in power consumption after allocation. This technique harnesses the heterogeneity of servers, choosing the most energy efficient one first . The pseudocode [15] is illustrated in figure 2.

Cloudsim Algorithm: Power Aware Best Fit Decreasing (PABFD)

```
1 Input: hostList, vmList Output: allocation of VMs
2 vmList.sortDecreasingUtilization()
3 foreach vm in vmList do
4       minPower = Double.MAX_VALUE
5       allocatedHost  = NULL
6       foreach host in hostList do
7               if host has enough resources for vm then
8                       powerAfterAlloc  = getPowerAfterAllocation(host, vm)
9                       if powerAfterAlloc < minPower then
10                              allocatedHost = host
11                              minPower = powerAfterAlloc
12                      if allocatedHost != NULL then
13                              allocation.add(vm, allocatedHost)
14 return allocation
```

Fig. 2 – Power Aware Best-Fit Decreasing Algorithm

B. *Proposed allocation policies for optimizing the VM placement*

In this section we propose four different algorithms to improve the default CloudSim VM placement strategies.

The first algorithm that has been developed is Minimum Space Best-Fit Decreasing. This algorithm makes use of two key elements. The first element is to determine the host with the smallest CPU gap. In other words, it checks if the difference between the maximum power a server can consume and the power it would consume after allocating the VM to that server is minimum. The second key element is to verify if the SLA violation threshold is exceeded in terms of server operating temperature. At the end of the simulation of the virtual machine placement on the available hosts, if there are several hosts that do not violate SLA, then the one which has the minimum power difference is returned. If there is no such host, then the host having the minimum power consumption is returned. Otherwise, the initial setup of servers and VMs is similar to those of the default algorithm used by CloudSim (i.e. PABFD).

Minimum Space Best-Fit Decreasing (MSBFD)

```
1 Input: hostList, vmList Output: allocation of VMs
2 vmList.sortDecreasingUtilization()
3 foreach vm in vmList do
4       maxPower = Double.MIN_VALUE
5       allocatedHost  = NULL
6       CPUThrAllocatedHost = NULL
7       foreach host in hostList do
8               if host has enough resources for vm then
9                       powerAfterAlloc  = getPowerAfterAllocation(host, vm)
10                      tempAfterAlloc =
                        CoolingSystem.calculateServerCPUTemperature(powerAfterAlloc)
11                      powerDiff = host.getMaxPower() - powerAfterAllocation
12                      if powerDiff < minPowerSpace then
13                              allocatedHost = host
14                              minPowerSpace = powerDiff
15                              if tempAfterAlloc < CoolingSystem.CPUTempSLAViolationThreshold
16                                      allocation.add(vm, CPUThrAllocatedHost)
17                      allocation.add(vm, allocatedHost)
18 return allocation
```

Fig. 3 – Minimum Space Best-Fit Decreasing (MSBFD)

The second algorithm developed uses a different idea compared to the approaches experimented in the other allocation policies developed. Therefore, instead of choosing the least energy-intensive hosts, we chose to place virtual machines on servers that consume the most amount of energy after allocating a virtual machine. Using this logic, at the beginning, a maximum number of servers are turned on, but as the number of servers advances, the optimization attempts produce a much lower SLA violation and a lower server temperature.

Modified Worst-Fit Decreasing (MWFD)

```
1 Input: hostList, vmList Output: allocation of VMs
2 vmList.sortDecreasingUtilization()
3 foreach vm in vmList do
4       maxPower = Double.MIN_VALUE
5       MIGRATION_POWER_THR = computePowerThrBasedOnPowerModel()
6       allocatedHost  = NULL
7       currentPowerHost = vm.getHost()
8       foreach host in hostList do
9               if host has enough resources for vm then
10                      powerAfterAlloc  = getPowerAfterAllocation(host, vm)
11                      powerDiff = powerAfterAlloc – host.getPower()
12                      if powerDiff > maxPower then
13                              allocatedHost = host
14                              maxPower = powerAfterAlloc
15                      if currentPowerHost!=null && allocatedHost != NULL then
16                              if currentPowerHost.getPowerAfterAlloc(vm) +
                                MIGRATION_POWER_THR  > powerAfterAlloc then
17                                      allocation.add(vm, currentPowerHost)
18                      allocation.add(vm, allocatedHost)
19 return allocation
```

Fig. 4 – Modified Worst-Fit Decreasing (MWFD)

Until this point we presented the standard WFD algorithm. In our approach, we chose a reasoning opposite to the one presented in the previous algorithm. If the current host's energy consumption, plus the same threshold used in the previous algorithm, exceeds the energy consumption of the chosen server using the WFD algorithm, then we keep the current server, otherwise we return the host chosen by the WFD algorithm. In this manner we enforce that the most

utilized servers will be kept, condition which also includes the current host. Thus, a lot of the times the current server is not even changed anymore and because of that, the number of migrations is minimized. Because of that also the Performance degradation due to VM migration decreases dramatically while the power consumption is kept at an optimal level.

The third proposed algorithm uses some basic reasonings to optimize the allocation. First of all, it follows the principles of First-Fit Decreasing (FFD) sorting the VMs descending, according to their CPU usage. What brings new, is that for each VM allocation, the hosts are being reordered by their available MIPS capacity, in descending order. This principle is used because we are trying to keep a balanced placement without having too much underutilized or overused servers. Another idea that was used is not to place a virtual machine on a host if it would become overused as a result of this placement. Consequently, from the first two principles, less VMs will be migrated because the hosts become better consolidated. Last but not least, we want to maintain a reasonable temperature, therefore if the server exceeds a certain safety threshold, we will search for another host for that VM. The safety threshold value was chosen after multiple experimental simulations, and is a value that depends on the safety temperature threshold of the IT equipment.

First-Fit Decreasing using Decreasing Host Available MIPS (FFDDHAM)

```
1 Input: hostList, vmList  Output: allocation of VMs
2 vmList.sortByCpuUtilizationDecreasing()
3 foreach vm in vmList do
4     hostList.sortByAvailableMipsDecreasing()
5     allocatedHost=null
6     foreach host in hostList do
7         if host has enough resources for vm then
8             if isHostOverUtilizedAfterAllocation(host, vm)
                  || host.getTin() > SAFE_TEMPERATURE_LEVEL then
9                 continue
10        powerAfterAllocation = getPowerAfterAllocation(host, vm)
11        if powerAfterAllocation < host.getMaxPower() then
12            allocatedHost = host
13        if allocatedHost != null then
14            allocation.add(vm, allocatedHost)
15 return allocation
```

Fig. 5 – First-Fit Decreasing using Decreasing Host Available MIPS (FFDDHAM)

Finally, we introduce a temperature-sensitive algorithm that has been developed to improve the temperature at which servers operate, in order to provide increased performance to applications running on them for a high QoS. It is based on the best fit fitting logic and the reasoning that a low temperature causes low power consumption. The metric to be compared will be a weighted function based on the difference between 3 sets of values:

- The temperature at the output of a server and the temperature at its input
- Temperature at server exit and supply temperature
- Temperature at server input and supply temperature

The server chosen is not the lowest value of this metric, but the second best server reported for this metric. Based on this, a second-best-fit policy is being implemented.

Thermal Aware Second Best Fit Decreasing (TASBFD)

```
1 Input: hostList, vmList Output: allocation of VMs
2 vmList.sortDecreasingUtilization()
3 foreach vm in vmList do
4     minTempDiff = Double.MAX_VALUE
5     allocatedHost  = NULL
6     secondAllocatedHost = NULL
7     foreach host in hostList do
8         if host has enough resources for vm then
9             diffToutTin = host.getTout() – host.getTin()
10            diffToutTsup = host.getTout() – CoolingSystem.supplyTemperature
11            diffTinTsup = host.getTin() – CoolingSystem.supplyTemperature
12            tempWeightedValue=w1*diffToutTin+w2* diffToutTsup+w3*diffTinTsup
13            if tempWeightedValue < minTempDiff then
14                allocatedHost = host
15                minTempDiff = tempWeightedDiff
16                If( allocatedHost!=null)
17                    secondAllocatedHost = allocatedHost
18                    allocation.add(vm, secondAllocatedHost)
19            allocation.add(vm, host)
20 return allocation
```

Fig. 6 - TASBFD Algorithm

The algorithm starts by sorting the virtual machines in the decreasing order of their utilization. Hosts are iterated and for each host a check is made to detect if that host will be able to sustain the current VM. If it has the required resources the thermal weighted function is calculated. Based on this function, during the simulation the two hosts that obtain the minimum value of the weighted function are retained, this behavior follows the Best-Fit policy. In order to keep a better balance from the other metrics perspective we are always returning the second best host according to the weighted function, not the first one.

V. Experiments and Validation

This section presents a set of scenarios and experiments performed to assess and evaluate the CoolCloudSim framework.

A. Experimental Setup

To simulate a real Cloud environment we have used well-defined experimental tests from the CloudSim example package.

TABLE III. Server Characteristics

Server Host Type	Proliant_G4	Proliant_G5
Host Mips	1860	2660
Host Cores	2	2
Host RAM [MB]	2048	4096
HOST BW [Gbit/s]	1	1
Host Storage [TB]	1	1

Thus, simulations are executed on a DC having 50 hosts, 50 VMs and 50 cloudlets that require a single processing element to run and have equal lengths. The IT infrastructure features are presented in the tables III and IV.

TABLE IV. VM CHARACTERISTICS

VM Type	VM MIPS	VM RAM [MB]	VM Cores	VM Bandwidth [Mbit/s]	VM Size [GB]
1	500	613	1	100	2.5
2	1000	1740	1	100	2.5
3	2000	1740	1	100	2.5
4	2500	870	1	100	2.5

For modeling a real cooling system, data extracted from the existing ASU HPCI data center [14] was used. The Datacenter has a size of 9.6m, 8.4m, 3.6m and has two rows of 42U racks arranged under the hot-aisle / cold-aisle topology. The air is blown by a CRAC unit through the raised floor tiles.

TABLE V. COOLING SYSTEM CHARACTERISTICS

Thermal Parameter	Value	Measure Unit
Air Flow (**f**)	8	m^3/s
CRAC Air Temperature	18	˚C
Room Temperature ($\mathbf{T_{amb}}$)	21	˚C
Air Density ($\boldsymbol{\rho}$)	1.225	Kg/m^3
Specific Heat ($\mathbf{C_p}$)	1.005	J/˚C
Server Thermal Threshold	70	˚C
Thermal Capacity	40.1	Ws/ ˚C
Thermal Resistance	0.99	˚C/W

B. Experimental Results

In order to make the results as relevant as possible, in our experiments we did not choose a single selection or detection policy, but we did the most relevant six combinations of them and we ran each allocation algorithm with all six combinations, as shown in Table VII. As evaluation metric, we compared the total power consumption, SLA violation per Active Host (SLATAH), Perfomance degradation due to VM migration (PDM), SLA violation, VM migration number, CPUThermalSLA, ESV, defined in Table VI.

TABLE VI. METRICS DESCRIPTION

Metric name	Description
Total Power Consumption	IT POWER + COOLING POWER
VM Migration	The number of VM migrations that took place during the simulation.
SLATAH	The time for which a host could not allocate the total amount of MIPS requested.
PDM	How much the performance of the VM is affected because it needs to migrate. $\text{PDM} = \frac{1}{M}\sum_{j=1}^{M}\frac{\text{Cdj}}{\text{Crj}}$ (6.1) **M** – number of the VMs **$\mathbf{Cd_j}$** – The performance degradation estimation due to migrations. By default is set to 10% of the CPU utilization **$\mathbf{Cr_j}$** – total CPU capacity requested by VM j
SLA	SLA = SLATAH * PDM
CPUThermalSLA	The time for which a host runs at a CPU temperature higher than the threshold imposed by the hardware producer.
ESV	$\text{ESV} = \frac{1}{\text{TotalPower} * \text{SLA}}$

It is important to mention that the metrics presented above are all CloudSim defined metrics, except the CPUThermalSLA and ESV. Total Power Consumption has incorporated the Cooling power consumed. *CPUThermalSLA* is a thermal metric which has been integrated in CloudSim in order to evaluate algorithms from the thermal point of view.

ESV is a combination of the most relevant two metrics and provide a unique value to assess the quality of the developed allocation policies. The most important metrics from the cloud providers and respectively the cloud consumers are the power consumption and respectively SLA. ESV is the inverted product of these two metrics, we have used the inversion to compare in the increasing order, not decreasing like for the other metrics, hence the bigger the ESV value, the better the algorithm is.

All the algorithms presented in the Table VII have been developed by us, except for FFD which is an industry standard. In this paper we have only detailed the most relevant four of them, the main idea for the rest of them can be found in this table.

TABLE VII. ALGORITHM DESCRIPTION

Algorithm	Allocation Policy
PABFD – Power Aware Best-Fit Decreasing	Choose the server which causes minimum power increase after allocation.
WFD – Worst Fit Decreasing	Choose the server which causes maximum power increase after allocation.
MWFD – Modified Worst-Fit Decreasing	Keep the most overloaded host policy between WFD allocated host and current host
FFDDHAM – First-Fit Decreasing using Decreasing Host Available MIPS	Sorts hosts in decreasing order by the available CPU MIPS, at each step. Fills them one by one, but also makes sure that they will not become overused, and the temperature safeness threshold is not exceeded.
CUTPABFD – Correlated Utilization Thermal and Power Aware BFD	Choose the server with the minimum power and thermal weighted value (CPUtemp + weight * powerDiffAfterAllocation) from the servers with the most correlated utilization values.
MSBFD – Minimum space Best-Fit Decreasing	Choose the server which remains with minimum free mips after allocation.
TASBFD – Thermal Aware Best Fit Decreasing	Choose the server with the minimum server temperature based on weighted function.
WFDT – Worst-Fit Decreasing Thresholds	Choose the server which causes maximum power increase if the power consumed - current power < Threshold.
FFD – First-Fit Decreasing	Fill the first server until no more VM's can be allocated, then move to the next host.
BTDWPD – Best Temp Difference or Worst Power Difference	Choose the server with the minTemperature weighted function or the host with the worst powerDifference.

The main metric from the point of view of cloud service providers is Power Consumption. This is calculated taking into account both the IT energy consumed and the energy consumed by the cooling system. The lower the metric value, the better the algorithm is. It can be seen from Fig. 7 that the minimum values are recorded by the **WFD**, **MSBFD** and **MWFD** algorithms, and also, that all developed algorithms provide superior energy performance compared to the PABFD default algorithm. For instance, WFD reduces consumption on average with 16%, whilst the majority of the proposed solutions perform an improvement of at least 10%.

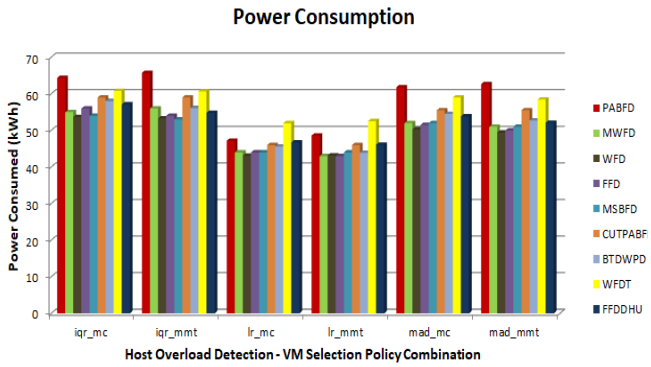

Fig.7 – Power Consumption Evaluation

Another useful metric for the cloud providers is the number of virtual machine migrations.

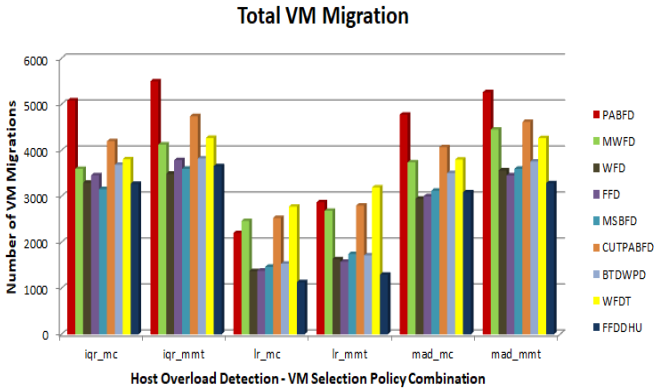

Fig. 8 – VM migration evaluation

The results obtained after the simulation show that, again all the algorithms have better performances than the default one. The best algorithm at this chapter is **FFDDHAM**, followed closely by **WFD**, **FFD**, **MSBFD**. FFDDHAM manages to migrate at least 35% less virtual machines compared to PABFD, in all cases, and in the best case scenario even 55% less.

SLA violation represents the most relevant control metric from the point of view of the cloud consumer. This is also a CloudSim defined metric and it represents the SLA Time Per Active Host Violation multiplied by the Performance degradation due to VM migration.

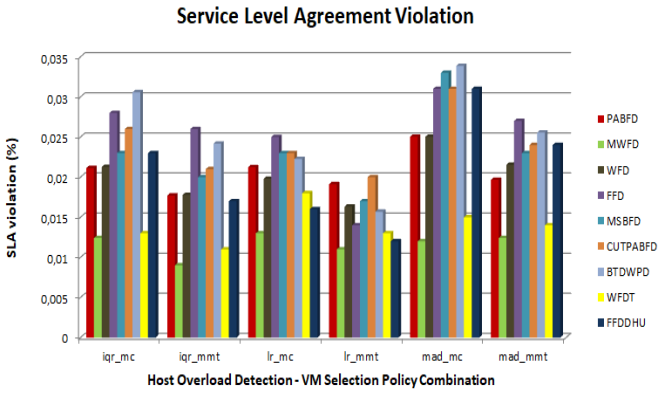

Fig. 9 - Service Level Agreement (SLA) violation evaluation

Best performance is obtained by the **MWFD** algorithm which alongside **WFDT** obtains much better values than the other algorithms. Regarding this metric, **FFDDHAM** and **WFD** obtain results close to **PABFD**. MWFD manages to obtain an overall 43% reduction of the SLA violation.

From thermal point of view, the policies that keep servers overheated least of the time, are **FFDDHAM** and **TASBFD.** FFDDHAM keeps the CPU overheated 6.5% of time, and TASBFD 12%, on average. PABFD allocation exceeds the CPU thermal threshold, 25.5% of the simulation time.

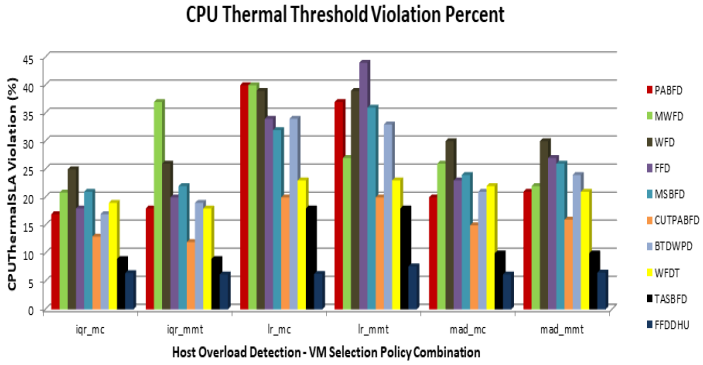

Fig. 10 – CPU Thermal Violation Evaluation

The most relevant metric is presented in the following lines. ESV evaluation is depicted in the chart below in the manner used previously.

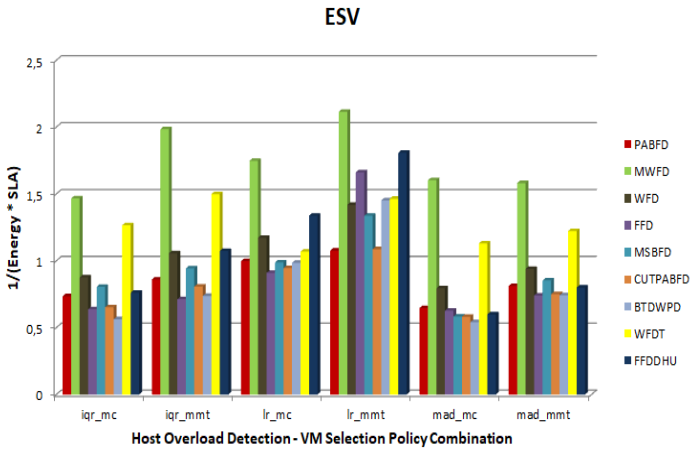

Fig. 11 - ESV evaluation

Besides this chart, we have also introduced a final chart which aggregates the ESV values for the six different combinations of host overload detection and VM selection policies for each allocation algorithm. Although the performance of the algorithms is a trade-off between different metrics, using ESV the most important two metrics are combined and hence using it with the cumulated value per allocation algorithm we can show a relevant score for each of the algorithms. Thus from both the chart above and the chart below it can be seen that **MWFD** obtains by far the best results, which is two times more efficient than the default CloudSim algorithm, PABFD.

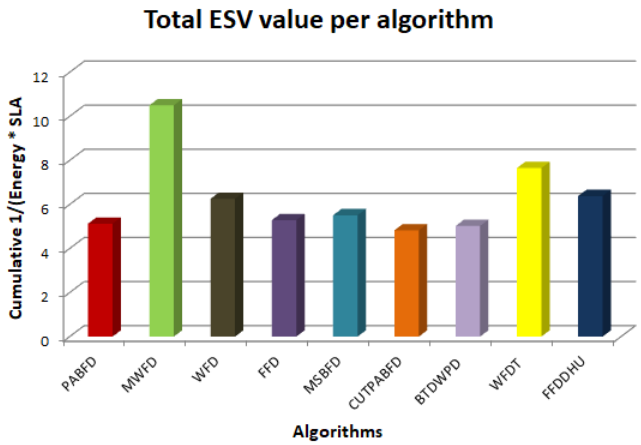

Fig. 12 - ESV cumulated value per algorithm

This algorithm is followed by the **WFDT**, **FFDDHAM** and **WFD** policies, and the rest of the algorithms present similar values to the default one.

VI. Conclusion

This paper presents a new framework, CoolCloudSim, built by adding cooling system models to the well-known CloudSim DC simulator architecture aiming to aid researchers from energy efficient DC area to develop and test thermal-aware allocation policies that minimize both IT and cooling system energy consumption. Besides adding mathematical models for the thermal processes within the server room, the paper also details four new thermal allocation strategies integrated in the new framework. Finally, a set of experiments evaluate the new algorithms by considering 6 evaluation metrics. The results show that the proposed thermal aware allocation algorithms outperform the default PABFD CloudSim allocation strategy, showing an overall energy consumption reduction of at least 10% while minimizing the number of VM migrations and maximizing the SLAs.

References

[1] R. N. Calheiros, R Ranjan, A Beloglazov1, C A. F. De Rose, R. Buyya,"CloudSim: a toolkit for modeling and simulation of cloud computingenvironments and evaluation of resource provisioning algorithms",Published online 24 August 2010 in Wiley Online Library.

[2] R. Buyya,M. Murshed, "GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing", Concurrency and Computation Practice and Experience 2002.

[3] B. Wickremasinghe, R. N. Calheiros2, R. Buyya, "CloudAnalyst: A CloudSim-based Visual Modeller for analysing Cloud Computing Environments and Applications", 24th IEEE International Conference on Advanced Information Networking and Applications, 2010.

[4] "Data center energy characterization study site report." [Online]. Available: http://hightech.lbl.gov/documents/data centers/dc benchmarking/ Data Center Facility1.pdf

[5] D. Filani, S. G. J. He, M. Rajappa, A. Kumar, P. Shah, and R. Nagappan,"Dynamic data center power management: Trends, issues, and solutions," Intel Technology Journal, vol. 12, no. 1, pp. 59–68, 2008.

[6] U.S. Environmental Protection Agency ENERGY STAR Program, "Report to congress on server and data center energy efficiency public law 109-431." 2007. [Online]. Available: http://hightech.lbl.gov/documents/data centers/epa-datacenters.pdf

[7] Garg, S. K., & Buyya, R. (2011, December). "NetworkCloudSim: modelling parallel applications in cloud simulations." In Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on (pp. 105-113). IEEE.

[8] J Jung,H Kim, "MR-CloudSim: Designing and implementing MapReduce computing model on CloudSim", International Conference on ICT Convergence (ICTC), 2012.

[9] Dabiah Ahmed Alboaneen, Bernardi Pranggono, Huaglory Tianfield, Energy-aware Virtual Machine Consolidation for Cloud Data Centers, IEEE/ACM 7th International Conference on Utility and Cloud Computing, 2014

[10] SANDEEP K. S. GUPTA, AYAN BANERJEE, ZAHRA ABBASI, GEORGIOS VARSAMOPOULOS, MICHAEL JONAS, and JOSHUA FERGUSON, GDCSim: A Simulator for Green Data Center Design and Analysis, ACM Transactions on Modeling and Computer Simulation, Vol. 24, No. 1, Article 3, Publication date: January 2014

[11] Q. Tang, T. Mukherjee, S. K. S. Gupta and P. Cayton, "Sensor-Based Fast Thermal Evaluation Model For Energy Efficient High-Performance Datacenters," Intelligent Sensing and Information Processing, 2006. ICISIP 2006. Fourth International Conference on, Bangalore, 2006, pp. 203-208.doi: 10.1109/ICISIP.2006.4286097

[12] A. Marcel *et al.*, "Thermal aware workload consolidation in cloud data centers," *2016 IEEE 12th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, 2016, pp.377-384. doi: 10.1109/ICCP.2016.7737177

[13] Piatek, Wojciech and Oleksiak, Ariel and Da Costa,Georges *Energy and thermal models for simulation of workload and resource management in computing systems.* (2015) Simulation Modelling Practice and Theory, vol. 58 (n° 1). pp. 40-54. ISSN 1569-190X

[14] ASU HPCI cooling system, http://impact.asu.edu/~mcn/publication/ICISIP2006-3.pdf

[15] Beloglazov A, Buyya R, Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers, 2012. Concurrency Computat. Pract Exper 24:1397–1420. doi:10.1002/cpe.1867

[16] R. L. Mitchell. 2010. Data Center Density Hits the Wall. Computerworld magazine.

[17] Five Strategies for Cutting Data Center Energy Costs through Enhanced Cooling Efficiency. Available online: http://www.emersonnetworkpower.com/documentation/en-us/brands/liebert/documents/white%20papers/data-center-energy-efficiency_151-47.pdf (accessed on 2 December 2017).

View publication stats