Data Center Simulator

University of Western Ontario

Contents

1. Introduction 3
2. Overview 5
   2.1 General features 5
   2.2 Assumptions 5
   2.3 Architecture 6
3. Physical Components 7
   - Compute node 9
   - Chassis 10
   - Rack 11
   - Data Center Physical Description 11
4. Logical Components 15
   - System 16
5. Computing energy consumption 23
   - Interactive and enterprise applications 23
   - HPC 24
6. The Simulator Operation 25
7. Management module 26
8. Evolving the simulator for new data center 27
   - Extracting thermal map 28
9. Examples 29
10. References 37

1. Introduction

IT energy consumption has become a significant issue and has stimulated a great deal of research into approaches and strategies to mitigate energy consumption of data centers and computing infrastructure. Most of the research in this field tries to apply algorithms which work for various levels of data center and cloud environments. To the best of our knowledge, there is little work, if any, considering a holistic approach looking at minimizing energy consumption while trying to minimize user SLA violations across the entire data center.

In our research, we have been concerned with policy based multi-level autonomic management frameworks for energy aware data centers. A management framework aims to minimize energy consumption of a data center and also ensure compliance to QoS requirements. A first step in exploring these problems is to have an environment which can be used to evaluate the performance of different management scenarios. The simulator described in this report provides a platform for exploring management of data centers by simulating different types of workloads running on computing equipment. The simulator reports on CPU utilization and SLA violations. CPU utilization of nodes is used to drive a thermal model of a data center which is used to compute power consumption (cooling and computing). Autonomic manager modules can be included within the simulator in order to examine the impact of management strategies. These managers are policy-based, can operate at different levels in a data center, and can make use of metrics about its domain e.g. SLA violations, CPU utilization, etc.

Simulation versus real experiments. Running a data center (with compute nodes and cooling equipment together) which is capable of supporting different configurations of applications and different types of hardware is the preliminary step. Simulation of such environment, rather than working with a small, toy data center, is the best solution for our purpose for several reasons. First, running a real data center, even a small one, is costly because providing a thermal insulation environment with chillers and several servers in it is expensive. Second, evaluation of different management ideas need to be evaluated in different types of computing loads on several hundreds of compute nodes, possibly heterogeneous, which may be hard to attain in a real data center. Third, examining a specific policy for different data center layouts (different placement of chillers and compute nodes or different cold/hot aisle plan) is more feasible with simulation. A simulator opens up the possibility of evaluating different scenarios; this is most important at this stage.

Other simulators. Several other data center simulators have been proposed and exist. CLOUDSIM tries to simulate the dynamic behavior of applications on virtual machines running on a cloud [3]. This simulator has powerful modules for evaluating virtualization policies. An energy management module for CLOUDSIM has been developed [3] and they now take into account the computing power of the host running the virtual machine workload; they do not take into consideration the cooling power consumption which obviously gets affected by the physical layouts of the data center.

GDCSIM [6] is another simulator which provides simulation of the thermal behavior of a data center with a given physical layout. GDCSIM does not provide a framework that can support different types of applications/users running jobs nor does it provide a mechanism for considering different SLA.

Key Features. The first objective behind the design of the simulator was the development of a platform for evaluation of different collaboration models of autonomic computing agents across the abstract model of data center and to explore the influence of the collaboration models on minimizing energy consumption of a data center with compliance to SLAs. To the best of our knowledge, this simulator is the only simulator which tries to simulate the management of energy consumption of a data center inspired by autonomic computing. The thermal models used in the simulator are validated model based on the BlueSim thermal data center simulator [7]. Another key feature of the simulator is that it supports different types of computing systems (enterprise, interactive and high performance computing) and different types of workloads with various SLA parameter definitions. Likewise, its hierarchical abstraction model aims to accommodate monitoring and management modules.

Key challenges. CPU utilization is assumed to be the primary parameter in determinig energy consumption of a server. This is a general assumption in the literature and so our focus has been on the timely calculation of server CPU utilization while jobs are running. Dependency of this simulator on BlueSim (to get thermal model) can also be considered a challenge since such a model is needed, although without having thermal model of the data centre computing power calculations and other features can still be done.

2. Overview

The initial objective in designing this simulator was to develop a platform to evaluate different collaboration models between different autonomic computing agents and to explore the influence of different collaboration architectures on energy consumption of a data center while considering compliance with SLAs. The simulator is basically an infrastructure for this research. The novelty of this approach lies in considering the datacenter as an entity managed by autonomic agents. In our case, we assume that these agents make use of defined policies, often defined at various levels, in order to manage a variety of aspects of a data center, from allocation of jobs to management or workload.

2.1 General features

Given our objectives, the simulator needed to support a number of features and capabilities:

- Support heterogeneous servers: the ability to support different types of servers with different levels of energy consumption and computing power.
- Support different workload models: the ability to define several types of workloads, e.g., compute intensive or interactive jobs.
- Support dynamic resource allocation for dynamically allocating server to different meet application requirements under changing workloads.
- Support different workload scheduling algorithms.
- Support different types of SLAs: the ability to define different kinds of SLA for each type of workload and also keep tracking of SLA violations.
- The ability to record energy consumption of the data center (computing and cooling).
- The ability to change CPU working frequency of servers.

2.2 Assumptions

The main assumption in our simulator is that the *thermal model* of the data center is given. The thermal model utilizes the CPU utilization of servers and then computes the total energy consumption (computing and cooling). Research shows that the power consumption of a compute node is mainly affected by the CPU utilization [5] and also that there is a linear relationship between power consumption and CPU utilization of a computer node [6]. Figure 1 shows the power consumption model of a server.

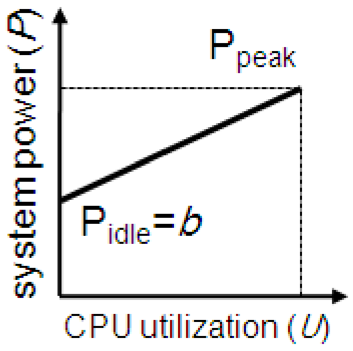


Figure 1. Power consumption model.

We assume that power consumed is computed by the relationship:

$$Power = a * cpu_{utilization} + b$$

Where:

1. The difference between peak power and idle power is called the *dynamic power consumption*.
2. The idle power consumption which is called the *static power consumption*.

2.3 Architecture

Figure 2 shows the overall structure of the simulator. The simulator gets the thermal profile of the data center and simulator inputs, e.g. configuration of systems and computing loads. During the simulator run, the results are output to a logfile which includes data on the thermal behavior of the system and SLA violations.

The front-end configures the simulator and records outputs. The back-end runs the workload and simulates the system. Dynamic changes in the system configuration based on policies is done in the autonomic manager module, which may contain several modules, e.g. representing several layers autonomic mangers The thermal profile of the data center is extracted from the BlueSim package [7]. The basic objective of BlueSim is to generate a Heat Recirculation Matrix (HRM) for a given topology of data center.

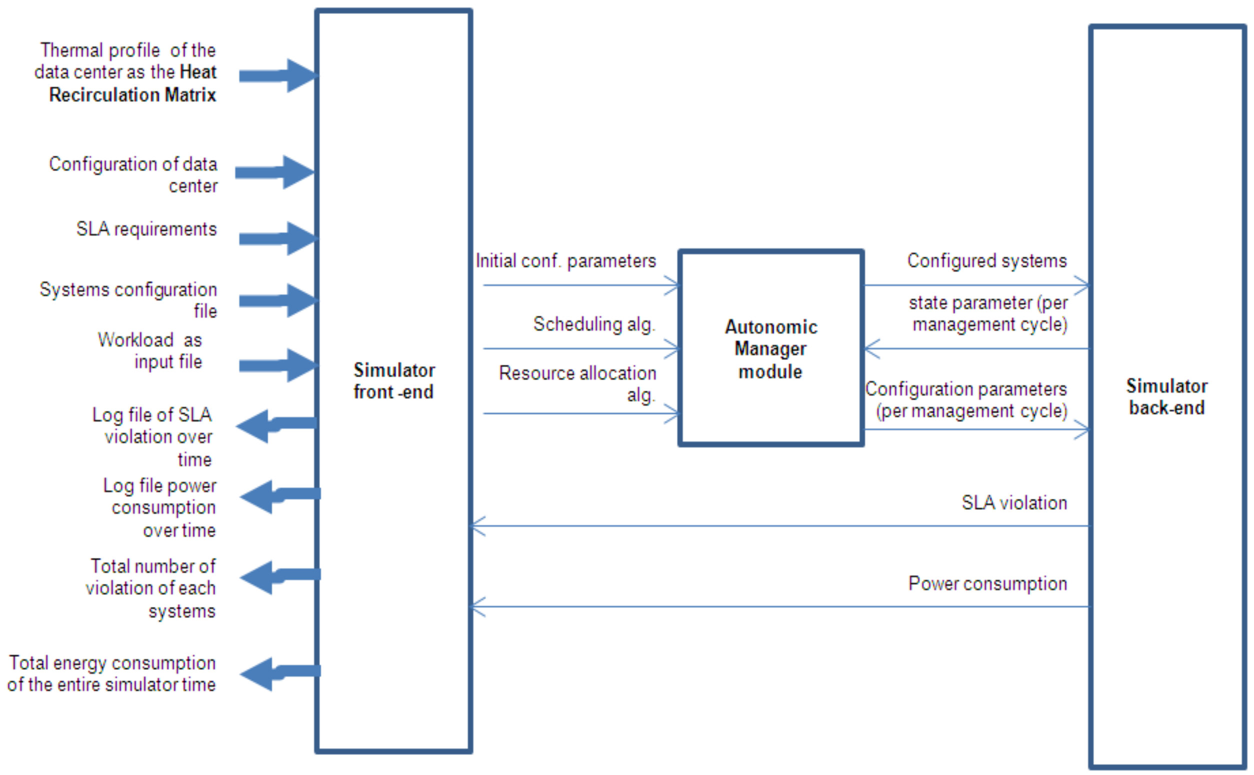

Thermal profile of the data center as the Heat Recirculation Matrix
Configuration of data center
SLA requirements
Systems configuration file
Workload as input file
Log file of SLA violation over time
Log file power consumption over time
Total number of violation of each systems
Total energy consumption of the entire simulator time
Simulator front -end
Initial conf. parameters
Scheduling alg.
Resource allocation alg.
Autonomic Manager module
Configured systems
state parameter (per management cycle)
Configuration parameters (per management cycle)
SLA violation
Power consumption
Simulator back-end

Figure 2. Overall structure of the simulator.

3. Physical Components

This section articulates the physical component perspective of the simulator and its architecture. The simulator looks at each type of application/user as encapsulated in a package named as a **system**, i.e., each system executes the same type of applications. The simulator supports three types of systems and corresponding workloads: HPC/computing, Enterprise, and Interactive. A datacenter can have any of number of different systems running applications and each system can have different types of applications/jobs, computing nodes, schedulers, and resource allocation algorithms.

Figure 3 shows the abstract model of the simulator. As shown, a data center contains a number of systems where each can contain a number of user/application modules where each of those has been allocated some number of compute nodes. The number of computing nodes for each application/user is determined by parameters specifying a minimum and maximum number of requested compute nodes. The number can vary dynamically based on decisions by managers based, in turn, on workload volume at a specific time and system situation; i.e. the number of compute nodes may vary during its life. Ultimately, jobs are performed over a data center physical layout which contains compute nodes and cooling equipment.

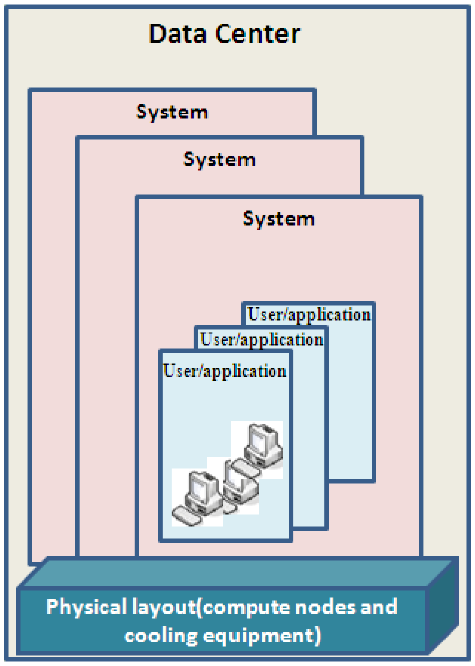


Figure 3. Abstraction model of simulator.

Datacenters have different type of workloads - from batch type computing jobs to interactive processing, such as a web based workload. Our current taxonomy of data center applications is as follows:

1. Interactive: provide access to users across the Internet/intranet; these applications process short requests (transactions) and fast response time is their main objective e.g. web transactions.
2. Enterprise applications: provide applications to different business units where applications may require large amounts of secure, reliable data transfer and high availability e.g. a Human Resource system. Workloads in these kinds of application vary – from short requests/jobs to much longer activities, e.g. report generation. The key characteristic is that these systems typically run for long periods.
3. HPC/computing applications: scientific applications which mostly need multiple CPUs or are enterprise jobs that perform batch processing.

In the following, we provide more details on each of the components and their definitions.

- Compute node

Each server, also called a **compute node**, is a typical blade server or rack mounted server located in a chassis which has its own specification. These configuration parameters are stored in the simulator configuration file. XSD in has been used to describe the syntax for the configuration files used in the simulator. Table 1 shows the XSD definition and provides an example of an XML version for describing a component. Configuration parameters specified include: the normalized MIPS levels, power consumption (idle state and fully utilized) levles, and standby power consumption of server.

Today's processors often have different working levels which provide different performance states (P-state), e.g.:

P0: *Minimum* p-state, highest power consumption and computation power.
P1, P2, P3….: P1 > P2 > P3 and so on, in terms of power consumption.
Pn : *Maximum* p-state with the lowest power consumption.

P-state power management can be seen in modern CPUs and GPUs. Each P-state is actually an operation point of frequency and voltage which has its own energy consumption and computing power. It allows a CPU/GPU to control their active power according to the load at any particular moment. The number of P-states is device-specific. We have included this feature in our simulator in order to investigate its use to dynamically change the state of processor based on the data center management policies. In order to get these parameters for a given server, we require data on the server that provides the standby power consumption and different frequency scaling level designed for it and for each of these levels we need to evaluate minimum power consumption when CPU utilization is zero and also is 100%. This information is then specified in the "MIPS", "FullyLoadedPwr", and "IdlePwer" elements of the definition.

In Table 1, the sample XML description is for an "HP Proliant DL3xx" server. As shown, its standby power consumption is 5Watt (defined by the "StandbyPwr" element), when it is idle it consumes 100 watts and with a fully utilized CPU consumes 300 watts [4,5]. In order to determine the power consumption of this server at different working frequency levels, we use the results of experiments in [3] that calculate the power consumption of most well known CPUs in their different working frequencies in with zero and 100% utilization. Since, we know that for instance "HP Proliant DL320e" has an Intel® Xeon® E3-1200v2 (Intel® Core™ i3 product family) processor, we use power consumption of Intel-Xeon processor for their different working frequencies.

Another configuration parameter is MIPS (the "MIPS" element in the XSD). This is defined as the ratio of the different frequency levels, e.g., as those that have been used in [3]. For the "HP Proliant DL3xx", we use "Intel® Core™ i3" frequency scaling levels which are 3.07, 3.2,and 4.2 GHz, which when normalized to the "base level" are 1, 1.07 and 1.37. The corresponding fully loaded and idle power (in watts) are then: 300, 336, 448 (defined in the "FullyLoadedPwr" element) and 100, 100, 128 (defined in the "IdlePwr" element), respectively

Table 1. Description of compute node.

Compute Node	
XSD description	XML description

`<xsd:complexType name="Server">`
`  <xsd:sequence>`
`    <xsd:element name="Type" type="xsd:string"/>`
`    <xsd:element name="MIPS" type="doubleList"/>`
`    <xsd:element name="FullyLoadedPwr" type="doubleList"/>`
`    <xsd:element name="IdlePwr" type="doubleList"/>`
`    <xsd:element name="StandbyPwr" type="xsd:integer"/>`
`  </xsd:sequence>`
` </xsd:complexType>`
	`<Server>`
	`  <Type> HP Proliant DL3xx </Type>`
	`  <MIPS> 1 1.04 1.37 </MIPS>`
	`  <FullyLoadedPwr> 300 336 448 </FullyLoadedPwr>`
	`   <IdlePwr>   100 100 128 </IdlePwr>`
	`   <StandbyPwr> 5 </StandbyPwr>`
	`</Server>`
	
	`<Server>`
	`  <Type> IBM Systems x3650 M2 </Type>`
	`  <MIPS> 1  </MIPS>`
	`  <FullyLoadedPwr> 300 </FullyLoadedPwr>`
	`   <IdlePwr>  100  </IdlePwr>`
	`   <StandbyPwr> 5  </StandbyPwr>`
	`</Server>`

- Chassis

A number of compute nodes are integrated in a chassis located in a rack. A chassis depends on the type of server inside the chassis and different types of chassis can be defined; see the XSD in Table 2. To define a chassis, a "ChassisType" is specified (a name), the number of servers in the chassis ("numberOfServer") and their type (the "ServerType" element of the definition) are specified. An example of an XML definition of a chassis ("basic_1") is defined in Table 2.

Table 2. Description of Chassis.

Chassis

XSD description	XML description

`<xsd:complexType name="Chassis">`
`  <xsd:sequence>`
`    <xsd:element name="ChassisType" type="xsd:string"/>`
`    <xsd:element name="numberOfServer" type="IntList"/>`
`    <xsd:element name="ServerType" type="StrList"/>`
`  </xsd:sequence>`
` </xsd:complexType>`
	`<Chassis>`
	`    <ChassisType>  basic_1   </ChassisType>`
	`    <numberOfServer> 14 </numberOfServer>`
	`    <ServerType>  HP Proliant DL3xx </ServerType>`
	` </Chassis>`

- Rack

To define a rack, we just need to specify the number of chassis in the rack ("NumberOfChassis") and their type ("ChassisType"). Each rack has its own "Rack_ID" which will be used for data center definition.

Table 3. Description of a Rack.

Rack	
XSD description	XML description

`<xsd:complexType name="Rack">`
`  <xsd:sequence>`
`    <xsd:element name="Rack_ID" type="xsd:integer"/>`
`    <xsd:element name="NumberOfChassis" type="IntList"/>`
`    <xsd:element name="ChassisType" type="StrList"/>`
`  </xsd:sequence>`
` </xsd:complexType>`
	`<Rack>`
	`    <Rack_ID> #R0 </Rack_ID>`
	`    <NumberOfChassis> 5 </NumberOfChassis>`
	`    <ChassisType> basic_1 </ChassisType>`
	`  </Rack>`

- Data Center Physical Description

All of the previously mentioned components are used to specify the physical elements data center physical layout. The actual description of the physical layout of the datacenter specifies two things: the racks in the datacenter and the thermal model of the data center. Note that the physical location (e.g. in a machine room model) is not done (this could be a useful extension in the future), rather, the thermal model implicitly implies the relative proximity and location of racks, chillers, hot and cold aisles in the computer room. The thermal model of the datacenter is defined by a Heat Recirculation Matrix (HRM) (described in more detail in Section 8) and describes the thermal affect of each compute node on other nodes in the datacenter. As indicated, there is no explicit description of the arrangement of the servers and racks as it is essentially hidden in the HRM matrix.

Red temperature of a computing device is the maximum external temperature that is considered "safe" for the device, i.e., is the maximum temperature before the device may fail to operate correctly. This is another datacenter configuration parameter which depends on the type of hardware. In the simulator, we use a single red temperature, which we take as the minimum red temperature of all hardware in the data center.

Since CPU is the main power consuming component of a compute node, our simulator is designed in such a way that it calculates the CPU utilization of all compute nodes in the data center. The simulator expects that the given thermal model calculates cooling and computing power of the whole data center given the CPU utilization of all the compute nodes.

Figure 4 shows the black box thermal model which the simulator is designed to deal with. Thermal specifications of coolers, the physical specification of servers, and the data center are inputs. Physical specification of servers and data center come from configuration files. The thermal specification of cooler is more complicated and explained in the following.

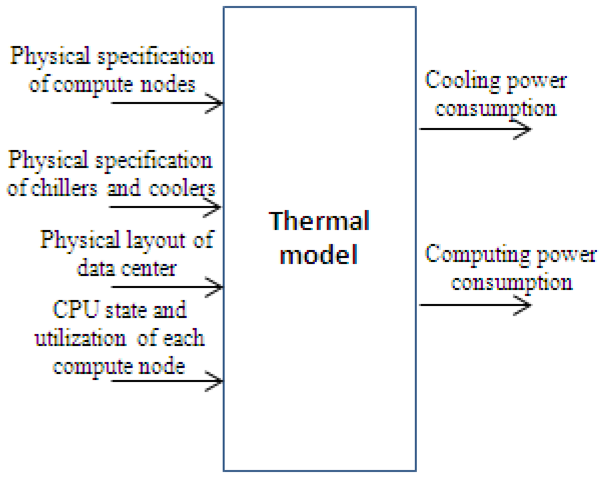

Physical specification of compute nodes
Physical specification of chillers and coolers
Physical layout of data center
CPU state and utilization of each compute node
Thermal model
Cooling power consumption
Computing power consumption

Figure 4. Abstract thermal model.

The efficiency of a Computer Room Air Conditioner (CRAC) depends on air flow velocity and conductivity of materials which is quantified as the Coefficient of Performance (COP). In the other words, the COP measures how much heat is removed by consuming a unit amount of heat (heat as energy). Therefore, the higher the COP value is, the more efficient the cooling system is [2]. The COP for a system is not constant and will increase with increasing the air temperature. Figure 5 shows how the COP changes over temperature as published by Hewlett-Packard (HP) laboratories [2]. This graph is well known in the simulation of energy consumption.

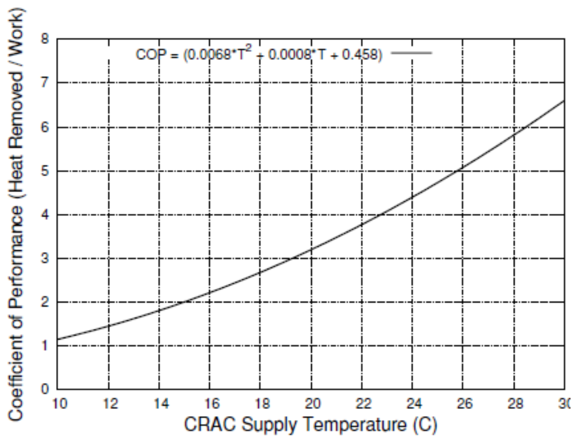


Figure 5. COP over temperature of CRAC units at the HP Lab

For COP changes over temperature, we have used the HP water-chilled CRAC unit change rate which is:

$$COP = 0.0068*T^2+0.0008*T+0.458.$$

Based on this formula, it is better to maintain the temperature of the air coming into the CRAC as high as possible. *But how much high?* For an upper limit on the CRAC output temperature, the nominal temperature at which devices work reliably is a suitable bound. (Also called the redline temperature). The redline nominal value is between 25C-35C.

But how is the COP and Thermal Model used to calculate cooling and computing power consumption of defined data center? We elaborate on this based on the work in [7].

In the CFD modeling of a data center [1, 3], heat distribution between computing nodes is modeled as a Heat Recirculation Matrix (HRM) $D=\{d_{ij}\}_{N}\times_{N}$. Each entry in the matrix shows the coefficient of distributed heat from node i to node j. Figure 6 shows an HRM with the coeeficient of re-circulated heat for each i and j server. Note that large values are observed along diagonal and that there is strong recirculation among neighboring servers.

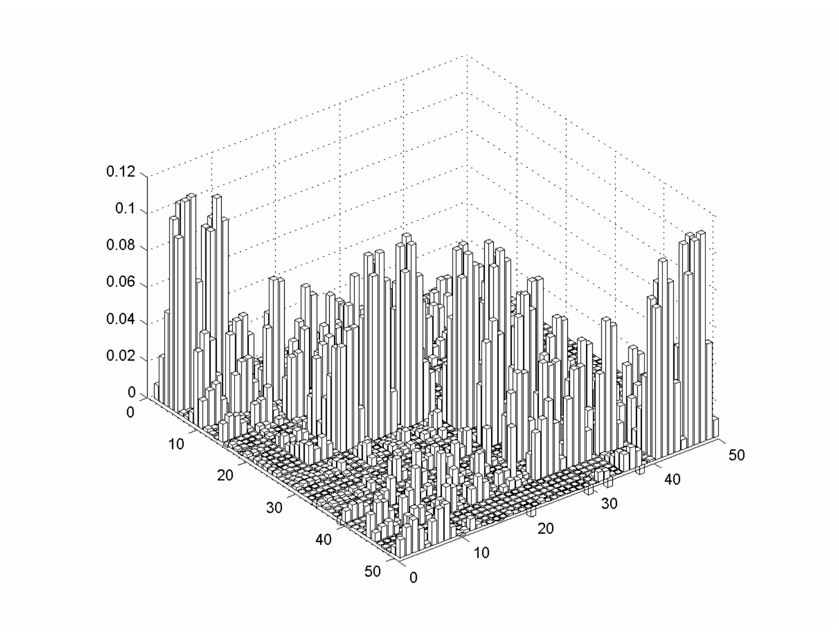
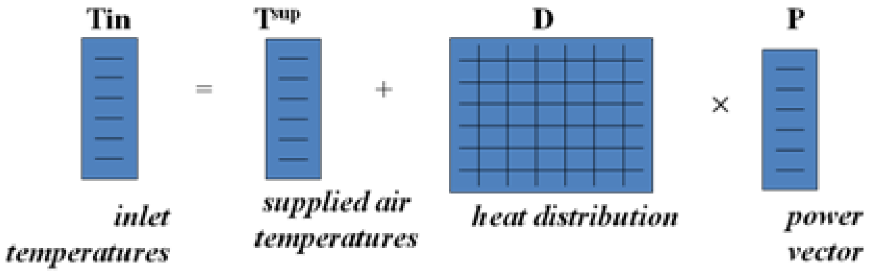


Figure 6. Strong recirculation among neighboring servers [7]

Each row shows the heat effect of a node on the others and each column are heat recirculation coefficients to that node. Next figure shows how the temperature of a node can be calculated:



For the total system:

$$P^{Computing} = \sum_{i \text{ all nodes}} P_i \qquad (3)$$

$$P^{Cooling} = \frac{P^{Computing}}{COP} \qquad (4)$$

For calculating the COP, maximum temperature of computing nodes and red line temperature are needed, here they are computed as follows [7]:

$\{T_i^{in}\} < T^{red}$ for all chassis's

Or $\max\{T^{in}\} < T^{red} \rightarrow T^{sup} + \max\{DP_i\} < T^{red}$

$\rightarrow T^{sup} < T^{red} - \max\{DP_i\}$

$$P^{Cooling} = \frac{P^{computing}}{COP(T^{red} - \max\{DP_i\})}$$

$$P^{Total} = P^{computing}\left(1 + \frac{1}{COP(T^{red} - \max\{DP_i\})}\right)$$

For each server, the computing power is the sum of the power consumption of all elements (memory, CPU). As mentioned, one of our basic assumptions is that CPU utilization is the main factor in determining the power consumption of a compute node. To compute the energy consumption of the data center, we just consider the summation of power consumption during the time.

Table 4 defines the XSD for specifying a data center layout.

Table 4. Data Center physical layout.

Data Center physical layout	
XSD description	XML description

`<xsd:complexType name="layout">`
  `<xsd:sequence>`
    `<xsd:element name="Rack" type="Rack" maxOccurs="unbounded" minOccurs="1"/>`
    `<xsd:element name="Chassis" type="Chassis" maxOccurs="unbounded" minOccurs="1"/>`
    `<xsd:element name="Server" type="Server" maxOccurs="unbounded" minOccurs="1"/>`
    `<xsd:element name="ThermalModel" type="xsd:string" maxOccurs="1" minOccurs="1"/>`
    `<xsd:element name="RedTemperature" type="xsd:integer" maxOccurs="1" minOccurs="1"/>`
  `</xsd:sequence>`
 `</xsd:complexType>`
	`<layout>`
	  `<Rack>`
	    `<Rack_ID>0</Rack_ID>`
	
	`<NumberOfChassis>5</NumberOfChassis>`
	    `<ChassisType>basic_1</ChassisType>`
	  `</Rack>`
	  `<Rack>`
	    `<Rack_ID>1</Rack_ID>`
	
	`<NumberOfChassis>5</NumberOfChassis>`
	    `<ChassisType>basic_1</ChassisType>`
	  `</Rack>`
	  `<Rack>`
	…..
	   //Definition of Chassis and Servers as came in pervious sections
	 `<ThermalModel>D15.txt</ThermalModel>`
	 `<RedTemperature>30</RedTemperature>`
	`</layout>`

4. Logical Components

After defining physical layout and thermal model of a data center, it is necessary to define the models of the applications to be run on the data center; we refer to the physical nodes running a set of applications as a *system* (definition comes later). Different type of applications defines different type of systems.

The main configuration file of the simulator defines the complete data center with the data center layout, thermal model and all systems running on the computers in the data center. Table 5 shows the XSD and XML logical description of data center. The elements of the configuration file for the simulator are the description of the data center physical layout (XML file), defined in the “layout” tag in an XML file for a “DataCenter” and then there is a definition of each of the defined systems in the data center.

Table 5. Data center logical description

Data center logical description

XSD description	XML description

`<xsd:complexType name="DataCenter">`
`  <xsd:sequence>`
`    <xsd:element name="layout" type="layout"/>`
`    <xsd:element name="system" minOccurs="1" type ="system"/>`
`  </xsd:sequence>`
` </xsd:complexType>`
` <!--      system configuration      -->`
` <xsd:complexType name="system">`
`  <xsd:sequence>`
`    <xsd:element name="name" type="xsd:string"/>`
`    <xsd:element name="type" type="xsd:string"/>`
`    <xsd:element name="sysDef" maxOccurs="1" minOccurs="1" type="sysDef"/>`
`  </xsd:sequence>`
` </xsd:complexType>`
` <!—Different system definition choice -->`
` <xsd:complexType name="sysDef">`
`  <xsd:choice minOccurs="0" maxOccurs="1">`
`    <xsd:element name="Interactive" type="Interactive"/>`
`    <xsd:element name="Enterprise" type="Enterprise"/>`
`    <xsd:element name="HPC" type="HPC"/>`
`  </xsd:choice>`
` </xsd:complexType>`
	`<DataCenter>`
	`    <layout> layoutFile.xml   </layout>`
	`     <System>`
	`        …`
	`      </System>`
	`     <System>`
	`        …`
	`      </System>`
	`</DataCenter>`

- System

A *system* represents a *number of compute nodes running the same type of workload for specific applications or a user*. The simulator defines different type of systems based on general categorization of workloads. Three different types of systems are defined: **enterprise systems**, **interactive systems**, and **computing systems (HPC)**.

At any time during the simulation of a data center, we have sets of running jobs which belong to a number of users or applications. For instance, in an enterprise system, each application has a number of compute nodes and those compute nodes can be shared between all running applications under the supervision of a resource allocation algorithm. In an interactive system, we have a dynamic set of compute nodes running workload which belong to various users; one can think of this as a cloud operating to support Infrastructure as a Service (IaaS).

A system configuration is specified by the following parameters:

1. A number of compute nodes: this is the number of compute nodes to be assigned to the system from the nodes in the physical data center. In XSD configuration file is defined as ***ComputeNode*** tag.
2. List of system's designated rack: the administrator may specify list of racks which are designated to the system. All servers in these racks can be shared between user/application running in the system. In XSD configuration file is defined as ***RackList***.
3. Resource allocation algorithms: allocates compute nodes to each application/user inside the system. Different algorithms can be implemented based on several parameters e.g. the SLA violation, amount of workload, or even total energy consumption level of system. For now, "minimum heat recirculation effect node first" has been implemented, which chooses the node with minimum heat effect first. Implementing a new algorithm needs to change the simulator code. In XSD configuration file is defined as ***ResourceAllocationAlg*** tag.
4. Scheduling algorithms does scheduling for jobs to be run. In XSD configuration file is defined as ***Scheduler*** tag.
5. Workload configuration parameters: these depend on the particular type of system and are used to specify attributes of the workload of particular system type. For example, for an enterprise system, all enterprise applications are defined at the beginning of simulation, however, for an interactive workload, the configuration of each application is specified in the input workload file and during run time the applications are initiated dynamically based on the description of the interactive application, including the minimum number of required node and SLA definition parameters. In XSD configuration file is defined as ***Enterprise Application WorkLoad*** tag.

What is running inside each system are batch, interactive, and enterprise type workloads. The simulator supports these three different types of workloads and each needs different parameters to define. Enterprise and interactive workloads are inherently the same, i.e. transaction based; however, their application models and the SLAs are different. In the following, we describe each type of system and its parameters.

4.1.1 Enterprise System

An Enterprise System represents a system which houses a number (1 or more) "enterprise applications", such as a financial or human resources system. Such systems are typically characterized by applications which run for long periods of time, are often transaction based, can run some number of batch jobs (e.g., reports), etc. An enterprise application workload, then, consists of requests in which the arrival time and number of transactions are specified.

In Internet data centers, the SLA statistically bounds the delay, i.e., the delay of q percent of requests should not go beyond the reference service delay. We borrowed the performance model provided by [8, 9]. So, the SLA definition is *for X percentage of transactions expected response time should be less than expected threshold (ET) time*. Here, *X* and *ET* are configuration parameters specified in system/application configuration file. In order to detect SLA violation, in each simulator time, we calculate response time for all run jobs and then calculate percentage of them which have response time more than threshold. The SLA violations are reported in a logfile at each simulator time period.

As indicated, an enterprise system can run multiple "enterprise applications". Each such application can have different workload parameters. The following are particular parameters for specifying an enterprise application:

- MaxNumberOfRequest & NumberofBasicNode: these two are used for performance modeling, briefly means: number of requests that fully utilize ***NumberofBasicNode*** of basic compute node. (More detail in section 5 4.1.1)
- minProcessor: Minimum number of required compute nodes that can be dedicated to this application.
- maxProcessor: Maximum number of required compute nodes that can be dedicated to this application.
- SLA definition: as explained earlier in this section, relaxed SLA definition has two parameters timeTreshold and Percentage, which overall says: SLA violation occurs if more than this Percentage of workload during last simulation time has response time more than timeTreshold.
- WLfile: a text file which contains workload for hosted application. Workload is transaction based and each record consists of: Arrival time and Number of transactions.

Putting all these parameters together we can define enterprise system and its applications. Table 6 shows the XSD description for an Enterprise System.

Table 6. Description of an Enterprise System

XSD description	XML description

<xsd:complexType name="Enterprise">
  <xsd:sequence>
    <xsd:element name="ComputeNode" type="xsd:integer"/>
    <xsd:element name="Rack">
      <xsd:simpleType>
        <xsd:list itemType="myRackId"/>
      </xsd:simpleType>
    </xsd:element>
    <xsd:element name="Scheduler" type="xsd:string"/>
    <xsd:element name="ResourceAllocation" type="xsd:string"/>
    <xsd:element name="Enterprise Application" minOccurs="1"
      type =" Enterprise Application "/>
    </xsd:sequence>
  </xsd:complexType>
 	
<!--   Application configuration   -->
<xsd:complexType name="Enterprise Application">
    <xsd:sequence>
      <xsd:element name="id" type="xsd:integer"/>
      <xsd:element name="MaxNumberOfRequest" type="xsd:integer"
/>
      <xsd:element name="NumberofBasicNode" type="xsd:integer" />
      <xsd:element name="minProcessor" type="xsd:integer"/>
      <xsd:element name="maxProcessor" type="xsd:integer" />
      <xsd:element name="Enterprise Application WorkLoad"
minOccurs="1" maxOccurs="unbounded" type ="AppJob" />
      <xsd:element name="timeTreshold" type="xsd:integer" />
      <xsd:element name="Percentage" type="xsd:integer" />
    </xsd:sequence>
</xsd:complexType>
 	
<!--   Rack List   -->
  <xsd:simpleType name="myRackId">
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="0"/>
    </xsd:restriction>
  </xsd:simpleType>
 	
  <!--WorkLoad file contains AppJob -->
  <xsd:complexType name="AppJob">
    <xsd:sequence>
      <xsd:element name="ArrivalTime" type="xsd:integer"/>
      <xsd:element name="NumofTransactionDuration"
type="xsd:integer"/>
    </xsd:sequence>
  </xsd:complexType>
	<Enterprise>
	    <ComputeNode> 5 </ComputeNode>
	  <RackList> R0,R1,R2 </RackList>
	  <ResourceAllocationAlg> Minimum-Heat-Recirculation
	</ResourceAllocationAlg>
	  <Scheduler> FCFS </Scheduler>
	  <Enterprise Application>
	    <id> N </id>
	    <MaxNumberOfRequest> M </MaxNumberOfRequest>
	    <NumberofBasicNode> K </NumberofBasicNode>
	    <minProcessor>J</minProcessor>
	    <maxProcessor> P</maxProcessor>
	    <Enterprise Application WorkLoad> Log.txt</Enterprise
	Application WorkLoad>
	    <timeTreshold> T </timeTreshold>
	    <Percentage> Pr </Percentage>
	  </Enterprise Application>
	 </Enterprise>

4.1.2 Interactive System

An Interactive System represents a system where users ask for a number of compute nodes for a period of time to run their applications, such as a web server. These systems can also be thought of as cloud systems where multiple virtual machines are run on a number of hosts by users for their particular applications. Once the application terminates, the compute nodes are released and they can be reused.

Unlike an Enterprise System where the applications to run are "defined", an Interactive System has a workload which describes the arrival time, duration and characteristics of each of the applications that are to be executed on the system. In turn, each of these applications has its own workload. This workload for the system is defined as follows:

Each record in workload file has the following fields:

- *Arrival time*: user starting time[1].
- *Duration*: roughly estimation of how long the application would stay in the system. This parameter would be used by scheduling algorithm e.g. choosing user or workload that has minimum duration first.
- *maxNumofNode:* Maximum number of required compute nodes that can be dedicated to this user.
- *minNumofNode*: Minimum number of required compute nodes that can be dedicated to this user.
- *WorkLoad*: a text file that contains the workload for this user hosted in the interactive system; we assume that each user application is transaction based and the workload of each would consist of an arrival time and numbers of transactions i.e. interpreted as these numbers of transaction (or one transaction) have come in this time.
- SLA definition: The SLA for each user is *maximum expected response time (ERT)* and defined as one of configuration parameters specified in description of an interactive application. The ERT is interpreted as: if the response time for a job exceeds the specified ERT, then an SLA violation occurs.
- *MaxNumberOfRequest & NumberofBasicNode*: these two are used for performance modeling, briefly means: number of requests that fully utilize ***NumberofBasicNode*** of basic compute node. (More detail in section 5 4.1.1)

[1] The only consideration for time in the simulator is that they all should be in same scale, e.g. if arrival time is based on second then duration of jobs should as well be in second.

Putting all mentioned parameter together, we have description of user in Interactive system. Table 7 shows the XSD description for an Interactive System.

Table 7. Description of an Interactive System

XSD description	XML description
<xsd:element name="Interactive">	<Interactive>
<xsd:complexType>	<ComputeNode> number </ComputeNode>
<xsd:sequence>	<RackList>R1,R2,… </RackList>
<xsd:element name="ComputeNode" type="xsd:integer"/>	<ResourceAllocationAlg> Minimum-Heat-
<xsd:element name="Rack">	Recirculation </ResourceAllocationAlg>
<xsd:simpleType>	<Scheduler> FCFS</Scheduler>
<xsd:list itemType="myRackId"/>	<WorkLoad>Log.txt</WorkLoad>
</xsd:simpleType>	</Interactive>
</xsd:element>	
<xsd:element name="ResourceAllocation" type="xsd:string"/>	
<xsd:element name="Scheduler" type="xsd:string"/>	
<xsd:element name="WorkLoad" minOccurs="1" maxOccurs="unbounded" type ="UsrJob" />	
</xsd:sequence>	
</xsd:complexType>	
</xsd:element>	
<!--WorkLoad file contains User Jobs-->	
<xsd:complexType name="UsrJob">	
<xsd:sequence>	
<xsd:element name="ArrivalTime" type="xsd:integer"/>	
<xsd:element name="Duration" type="xsd:integer"/>	
<xsd:element name="minNumofNode" type="xsd:integer"/>	
<xsd:element name="maxNumofNode" type="xsd:integer"/>	
<xsd:element name="WorkLoad" minOccurs="1" maxOccurs="unbounded" type ="transJob" />	
<xsd:element name="MaxNumberOfRequest" type="xsd:integer" />	
<xsd:element name="NumberofBasicNode" type="xsd:integer" />	
<xsd:element name="SLA" type="xsd:integer"/>	
</xsd:sequence>	
</xsd:complexType>	
<!--WorkLoad file contains Transaction Jobs-->	
<xsd:complexType name="transJob">	
<xsd:sequence>	
<xsd:element name="ArrivalTime" type="xsd:integer"/>	
<xsd:element name="NumofTransactionDuration" type="xsd:integer"/>	
</xsd:sequence>	
</xsd:complexType>	

4.1.3 HPC (Compute) System

A HPC System models a system devoted to batch processing of CPU intensive jobs where each job is assumed to have its own estimation of average CPU utilization. Basically, each HPC job has the following parameters as its descriptors:

i. Arrival time: arrival time of HPC job in the simulator, this job may be run any time later on according to availability of computing resource.
ii. Duration.
iii. CPU utilization: is average CPU utilization (maximum to 100) of this specific job on all specified nodes during its execution time.
iv. Number of needed nodes: HPC type jobs need to have multiple CPUs to get done. Here we have a relaxed definition for number of processor which is actual number of physical nodes, in other word how many basic compute nodes are needed to run this job?
v. Deadline: maximum acceptable waiting time of a job before it is dispatched to all requested compute nodes/node. An SLA violation will be recorded if the job got its needed compute node in longer than deadline waiting time.

Jobs are allocated the number of requested nodes. Whenever a job is considered as a finished job, i.e., duration has been reached, we assume that all parts of the job on other nodes are finished as well. Further, we assume that all the computing nodes for a job are being used, that is, we cannot turn off or idle any one node since they may be dependent on other computing nodes.

For an HPC system, whenever the deadline for running a job is passed, simulator will record DEADLINEPASSED violation. For each HPC job beside arrival time there is a deadline refers to maximum time between arrival of a job and putting job on running queue of a compute node. Another SLA violation flag is NODESHORTAGE which refers to situations in which there are not enough ready nodes for running the job. Of course in this situation jobs will be held in system waiting list till a new compute node becomes available. (Another job gets finished or new compute node is allocated to this system)

Putting all description together, an HPC type system has just general definition of a system (which explained before). Each HPC job definition record has ArrivalTime, Duration, CPU utilization, numberOfNodes, and deadline fields. Table 8 shows the XSD description for an HPC system, along workload definition required tags

Table 8. Description of an HPC System

XSD description	XML description
<xsd:complexType name="HPC"> <xsd:sequence> <xsd:element name="ComputeNode" type="xsd:integer"/> <xsd:element name="Rack"> <xsd:simpleType> <xsd:list itemType="myRackId"/> </xsd:simpleType> </xsd:element> <xsd:element name="ResourceAllocation" type="xsd:string"/> <xsd:element name="Scheduler" type="xsd:string"/> <xsd:element name="WorkLoad" minOccurs="1" maxOccurs="unbounded" type ="HPCJob" /> </xsd:sequence> </xsd:complexType> <!--WorkLoad file contains HPCJobs--> <xsd:complexType name="HPCJob"> <xsd:sequence> <xsd:element name="ArrivalTime" type="xsd:integer"/> <xsd:element name="Duration" type="xsd:integer"/> <xsd:element name="CPUutilization" type="xsd:double"/> <xsd:element name="numberOfNodes" type="xsd:integer"/> <xsd:element name="deadline" type="xsd:integer"/> </xsd:sequence> </xsd:complexType>	<HPC> <ComputeNode> N</ComputeNode> <RackList>R1,R2,… </RackList> <ResourceAllocationAlg> Minimum-Heat-Recirculation </ResourceAllocationAlg> <Scheduler> FCFS</Scheduler> <Workload>Log.txt</Workload> </HPC>

5. Computing energy consumption

The simulator gets the configuration file sets up the physical datacenter (initializing array list of chassis's and servers) and requested systems with defined users/applications in them, afterward starts to read workload files and runs them. Since CPU utilization is the primary parameter in energy consumption of a server, here our focus is timely calculation of CPU utilization while jobs are running. This section describes how the simulator runs different types of jobs and calculates CPU utilization.

- Interactive and enterprise applications.

For interactive and enterprise based applications, which have a similar type of workload, i.e. transaction based, each record of the workload input file contains the arrival time of requests and the number of arriving requests.

The utilization of a server depends on its workload and its physical characteristics. Workload of a compute node hosting an application is a function of its online number of requests. Therefore, we assume the following linear model for utilization of a compute node:

$$u_t = c\, n_t,$$

where:

$u_{;t}$ denotes the average utilization that the application impose to compute node during epoch t,

$c_{:}$ is the average utilization that one online request of the application imposes on the compute node during one second which depends on the system hardware characteristics and type of the application,

n_{t} :is the total number of requests that are assigned to the compute node at epoch t.

This model is frequently used in existing literature and experimental results show its sufficiency [7, 8].

In definition of each application we need to somehow configure **c** parameter. The value of **c** is determined at configuration time for each application of an enterprise system and for application of an interactive system. To do that, we use two parameters in configuration file, namely:

```
<MaxNumberOfRequest> 1000</MaxNumberOfRequest>
<NumberofBasicNode> 1 </NumberofBasicNode>
```

Simulator interprets these parameters as follows:

An application will consume one compute node, i.e., fully utilize, to execute "MaxNumberOfRequest" requests at its primary frequency level or MIPS value.

To calculate CPU utilization of a node running an application, which has variable levels of MIPS, beside current number of request we need to consider current compute node MIPS level. CPU utilization value is calculated proportional to compute node current MIPS value.

- **HPC**

For HPC jobs, the simulator expects to have its CPU utilization i.e. specified in workload definition. At each simulation cycle, based on the number of jobs running in compute node, the simulator divides 100% CPU utilization equally between jobs, then for each job if it's requested CPU is less than the share, then just the requested CPU is used for the job and remain amount would be used for other jobs running on the compute node.

Remain execution time of a job decreased by one if its assigned compute node has exact amount of job requested CPU utilization, otherwise remain execution time is decrease proportionally (definitely is less than one).

6. The Simulator Operation

The main processes in the simulator after the configuration step, are running the workload, resource management, and timely calculation of energy consumption. Energy consumption is calculated using the thermal model which is defined in the configuration module (configurePysicalLayout). Overall pseudo code of the simulator is shown in Table 9.

Table 9. The simulator lifecycle pseudo code

Data center life cycle procedure

```
procedure DClifeCycle()
begin
    dataCenterInitialization();
    while(jobs are not finished)
      begin
        for each systems  i in the data center
                SysList.get(i).RunACycle();
        end for
        Compute power consumption();

        if( managerIsAttached)
            runManagementCycle();
        end if

    end while
end procedure

procedure dataCenterInitialization ()
begin
    configurePysicalLayout();
    for  all number of requested system
    begin
        Read confing file();
        initialize systems();
        //insert current system in data center system list
        SysList.add(sys) ;
    end for
end procedure
```

```
procedure System ::RunACycle()
begin
        for each application app in the system
                app.runAcycle();
        end for

         if( managerIsAttached)
            runManagementCycle();
        end if

end procedure

procedure application::RunACycle()
begin
      readJob();
      dispatchJobToComputeNode();
      for each computeNode node in the application
            node.calculateCPUutilization();
      end for
      check&SetViolation();
      if managerIsAttached()
            runManagementCycle();
      end if
end procedure
```

For each system there is an output file which records each time a SLA violation occurs and the information recorded depends on type of system:

For interactive and enterprise applications, each time an SLA is violated, based on the SLA definition for an application, the violation is recorded in output file of corresponding system.

Finally, another output file will record power consumption of whole data center during simulation time. Each record of this file includes: *computing power*, *cooling power*, and *time*.

Class diagram of what we have implemented is shown in next figure.

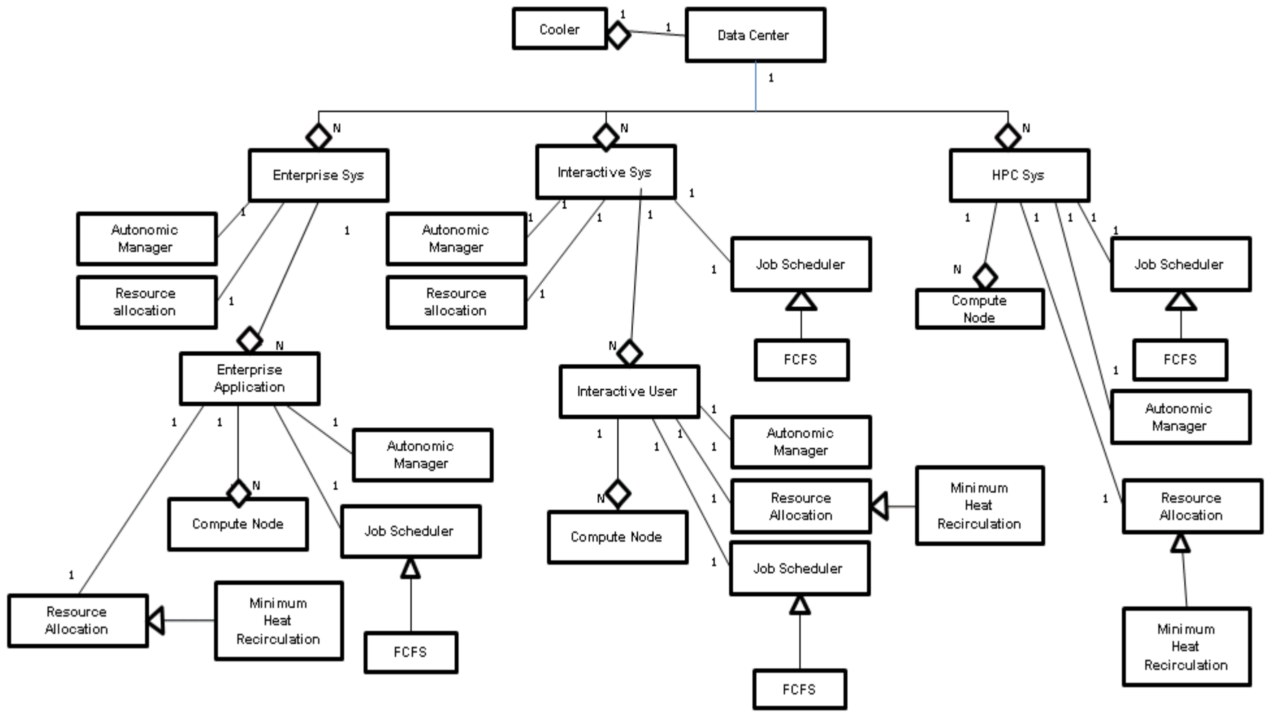


Figure 5 Simulator class diagram

The simulator generates output file for each system recording what type of SLA violation occurs and when. Example on output file has come in ???

7. Management module

The main purpose of developing this simulator is providing an infrastructure for simulating IaaS data center behavior. As shown in Figure 2 simulator back-end runs jobs according to simulator configuration done by user. On top of this platform we may have distributed *autonomic managers*, each dedicated to an element. Each manager tries to manage its *managed element* based on its first monitored environmental variables and management strategy of managed element. Managed element could be any defined element in simulator e.g. a system, rack or even a compute node.

As shown in Figure 2 logical separation between *simulator back-end* and management module makes code development for management module much easy.

We have defined an *AM class* for this purpose; in its constructor it needs to get all public variables of the *attached managed element*. AM class main function is the Analyze method which based on whether SLA violation occurs or not and based on its *defined hardcoded policies* makes decision whether to apply *any configuration change* to managed element or not.

In this research we advocate for hierarchical management framework. There are distributed *autonomic managers* over the data center. Next figure shows the arrangement of autonomic manager in our simulated data center. As shown manager are assigned to each system, user (in interactive system) or application (in enterprise system), there is one manager for whole data center that does coordination between managers.

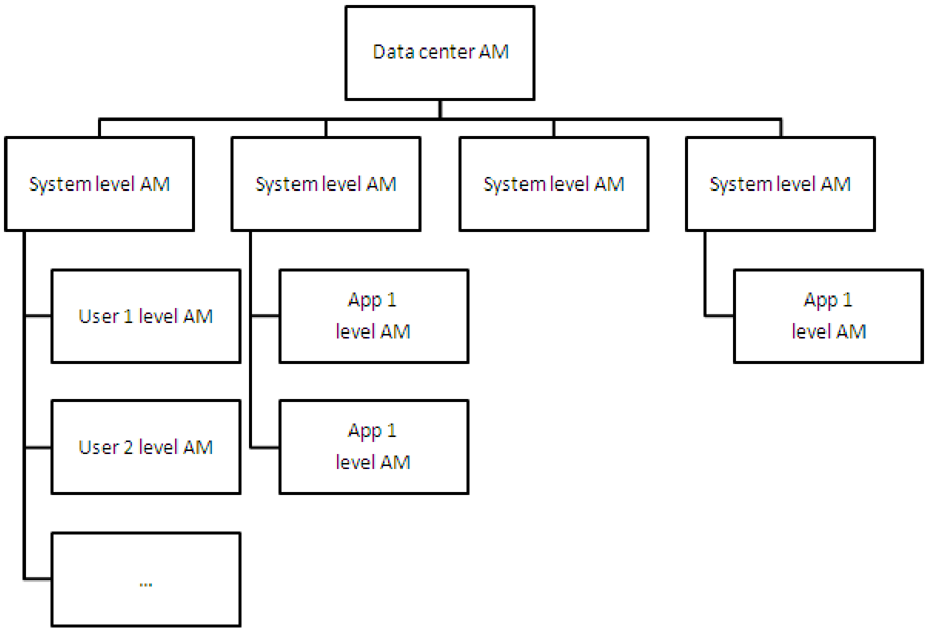


Figure 6 Proposed management framework

8. Evolving the simulator for new data center

Now, the question is that what would be the action plan for using the simulator for a new data center?

- Define the physical configuration of data center as depicted in **Data Center Physical Description** section.
- Get a thermal map for given data center using BlueSim as described in the section on the thermal map.
- Define the systems, applications, SLAs, and workloads.

- Defining the position of autonomic managers and management scenarios: to do so, programming is needed.
- Developing management policies and scenarios.

- Extracting thermal map

The Arizona State datacenter team introduced a package -BlueSim- which aims to generate HRM for any given description of datacenter [2]. BlueSim is a simulation package, which integrates various software for geometry generation, CFD simulation, and post processing. The main idea behind BlueSim is to generate an Heat Recirculation Matrix (HRM) for different configurations of the data centers. There are three main subcomponents in BlueSim: [2]

- **Pre-processing**: The input to BlueSim is a high level XML-based data center description in Computer Infrastructure Engineering Language (CIELA). It has a range of elements that capture the generic layout of a data center: (i) equipment configuration, i.e. how is the arrangement of servers into chassis and chassis into racks, likewise the arrangement of these racks into rows (ii) physical datacenter layout, i.e. presence of raised floors, lowered ceilings. A parser parses the given XML specification to create a geometry file which depicts the layout of datacenter. The layout is then converted to a mesh file using GMSH, a standalone open source meshing software. *Note that for getting thermal model of a new data center, just defining the physical description of data center in CIELA is needed, the rest is done by BlueSim.*

- **Processing**: This component runs a series of offline CFD simulation of the specified data center to enable determination of HRM. The simulation scenarios aim to extract the effect of each chassis on others by fully utilizing the node and getting thermal information of other nodes from the CFD model.

- **Post-processing**: This component generates an array of different HRMs with different active server sets. Having number of HRMs based on different situations gives better thermal model precision. The current version generates only a single HRM with all the servers active. The claim they have plan for future releases of BlueSim that will generate an array of HRMs for different active server sets.

9. Examples

This section illustrates configuration of the simulator by going through an example. This is a homogenous data center with a number of systems running: two enterprise systems, one interactive system, and one HPC system. The logical configuration of this data center is defined in Table 10 and each of the systems in the data center are defined in Table 11.

Table 10. Logical definition of example data center.

Logical description of Datacenter

<DataCenter>
  <layout>DC.xml</layout>
  <System>
      <name> Enterprise_01 </name>
      <type> Enterprise </type>
      <Enterprise> ES.xml </ Enterprise >
   </System>
   <System>
      <name> Enterprise_02 </name>
      <type> Enterprise </type>
      < Enterprise > ES2.xml </ Enterprise >
   </System>
  <System>
      <name> Inter_01 </name>
      <type> Interactive </type>
      <Interactive> IS.xml </ Interactive>
   </System>
   <System>
      <name> HPC_01 </name>
      <type> HPC </type>
      <HPC> CS.xml </HPC>
   </System>
</DataCenter>

Table 11. Configuration of example systems.

Enterprise system	Interactive System	HPC system

< Enterprise >
  <ComputeNode> 5</ComputeNode>
  <Rack>2 </Rack>
  <ResourceAllocationAlg> Minimum-Heat-Recirculation </ResourceAllocationAlg>
  <Scheduler> FCFS </Scheduler>
  <Enterprise Application>
    <id> 1 </id>
    <MaxNumberOfRequest> 1000</MaxNumberOfRequest>
    <NumberofBasicNode> 1 </NumberofBasicNode>
    <minProcessor>3</minProcessor>
    <maxProcessor> 5</maxProcessor>
    <Enterprise Application WorkLoad> 18HourLog.txt</ Enterprise Application
	<Interactive>
	  <ComputeNode> 5 </ComputeNode>
	  <Rack>6 </Rack>
	  <ResourceAllocationAlg> Minimum-Heat-Recirculation </ResourceAllocationAlg>
	  <Scheduler> FCFS</Scheduler>
	<WorkLoad>test_inter.txt</WorkLoad>
	</Interactive>
		<HPC>
		  <ComputeNode> 30</ComputeNode>
		  <Rack>3,4,5,7,8,9 </Rack>
		  <ResourceAllocationAlg> Minimum-Heat-Recirculation </ResourceAllocationAlg>
		  <Scheduler> FCFS</Scheduler>
		<Workload>100in40.txt</Workload>
		</HPC>

```
WorkLoad >
        <timeTreshold> 2 </timeTreshold>
        <Percentage> 90  </Percentage>
    </ Enterprise Application >
 < /Enterprise >
```

```
<Enterprise>
    <ComputeNode> 8</ComputeNode>
    <Rack>0,1 </Rack>
    <ResourceAllocationAlg> Minimum-
Heat-Recirculation
</ResourceAllocationAlg>
    <Scheduler> FCFS </Scheduler>
    < Enterprise Application >
        <id> 1 </id>
        <MaxNumberOfRequest>
500</MaxNumberOfRequest>
        <NumberofBasicNode> 1
</NumberofBasicNode>
        <minProcessor>2</minProcessor>
        <maxProcessor> 8</maxProcessor>
        < Enterprise Application WorkLoad
> 18HourLog.txt</ Enterprise
Application WorkLoad >
        <timeTreshold> 2 </timeTreshold>
        <Percentage> 90  </Percentage>
    </ Enterprise Application >
  </Enterprise>
```

Next tables show sample of input files for each of systems.

HPC type sample input file

ArvTime	Duration	CPU	#Node	DeadLine
1	1	41.07	2	1
1	1	51.19	2	1
1	1	48.33	1	1
1	8	45.16	1	1
1	9	52.78	1	1
1	7	57.89	3	1
1	13	48.89	3	1
1	6	48	3	1
1	7	38.46	3	1
1	8	40.63	3	1
2	370	27.72	2	1
4	8	58.46	2	1
4	17	37.5	2	1
4	1	49.23	2	1
4	1	50	2	1
4	8	53.08	3	1
4	7	40.74	1	1
4	411	10.9	1	1
5	10	35.8	1	1

Figure 7 HPC sample workload

ArvTime	Duration	max#node	min#node	wrkld	ERT	MaxNumber OfRequest	NumberofBasicNode

1	40	50	10	Day82by15.txt	2	500	1

Figure 8 Interactive type sample input file

ArvTime	# of Request
1	225
2	272
3	241
4	215
5	249
6	240
7	266
8	265
9	269
10	262
11	241
12	288
13	246
14	267
15	304

Figure 9 Sample input file for enterprise application and interactive user.

These systems are configured on a homogenous data center. The next table shows the data center configuration file of the data center which has 10 racks (a 42U rack), each of which has 5 chassis in it and inside each chassis, there is one server.

Table 12. Sample physical layout configuration file

Physical data center layout description

```
<layout>
     <Rack>
        <Rack_ID> 0 </Rack_ID>
        <NumberOfChassis> 5 </NumberOfChassis>
        <ChassisType> basic_1 </ChassisType>
    </Rack>
     <Rack>
        <Rack_ID> 1 </Rack_ID>
        <NumberOfChassis> 5 </NumberOfChassis>
        <ChassisType> basic_1 </ChassisType>
    </Rack>
    <Rack>
        <Rack_ID> 2 </Rack_ID>
        <NumberOfChassis> 5 </NumberOfChassis>
        <ChassisType> basic_1 </ChassisType>
     </Rack>
     <Rack>
        <Rack_ID> 3 </Rack_ID>
        <NumberOfChassis> 5 </NumberOfChassis>
        <ChassisType> basic_1 </ChassisType>
    </Rack>
    <Rack>
        <Rack_ID> 4 </Rack_ID>
        <NumberOfChassis> 5 </NumberOfChassis>
        <ChassisType> basic_1 </ChassisType>
     </Rack>
     <Rack>
        <Rack_ID> 5 </Rack_ID>
        <NumberOfChassis> 5 </NumberOfChassis>
        <ChassisType> basic_1 </ChassisType>
    </Rack>
    <Rack>
```

```
            <Rack_ID> 6 </Rack_ID>
            <NumberOfChassis> 5 </NumberOfChassis>
            <ChassisType> basic_1 </ChassisType>
         </Rack>
         <Rack>
            <Rack_ID> 7 </Rack_ID>
            <NumberOfChassis> 5 </NumberOfChassis>
            <ChassisType> basic_1 </ChassisType>
        </Rack>
        <Rack>
            <Rack_ID> 8 </Rack_ID>
            <NumberOfChassis> 5 </NumberOfChassis>
            <ChassisType> basic_1 </ChassisType>
         </Rack>
         <Rack>
            <Rack_ID> 9 </Rack_ID>
            <NumberOfChassis> 5 </NumberOfChassis>
            <ChassisType> basic_1 </ChassisType>
        </Rack>
          <Chassis>
            <ChassisType>  basic_1   </ChassisType>
            <numberOfServer> 1 </numberOfServer>
            <ServerType>  HP Proliant DL3xx </ServerType>
         </Chassis>
<Server>
  <ServerType> HP Proliant DL3xx </ServerType>
  <MIPS> 1 1.04 1.4 </MIPS>
  <FullyLoadedPwr> 300 336 448 </FullyLoadedPwr>
    <IdlePwr>   100 100 128 </IdlePwr>
    <StandbyPwr> 5 </StandbyPwr>
</Server>
          <ThermalModel> D15.txt </ThermalModel>
        <RedTemperature> 26 </RedTemperature>
</layout>
```

The thermal model used in this example is the ASU thermal model [7], which is extracted from their data center which has 50 chassis's which each of them has 5 servers inside. The specification of the server in each chassis in terms of number of cores and its CPU parameters is configurable. The physical layout of the ASU HPC datacenter is shown in next Figure 7. The ASU datacenter's physical dimensions are 9.6*m* × 8.4*m* × 3.6*m*, with two rows of industry standard 42U racks. There are 10 racks; each rack is equipped with 5 chassis.

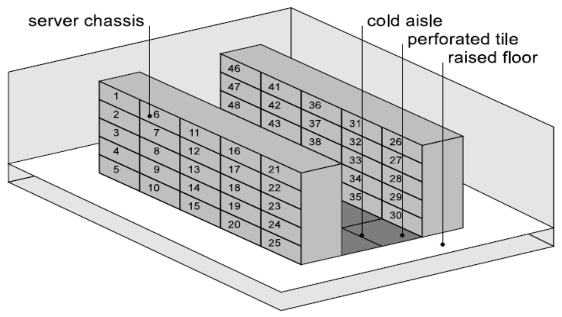


Figure 7. Physical layout of ASU datacenter []

After running the simulator, we get the results. First output is the resource allocation for each system (chassis number and server id) and then each system starst to run it assigned workload. The results include the simulation time, number of SLA violation and total power consumption. Beside these parameters, simulator generates number of log files: one for each system type and one which shows timely energy consumption (computing and cooling) for the whole data center.

Table 13. Sample output from the simulation.

Result of running this simulator for given systems

```
-----------------------------------------
Initialization of Enterprise System Name=Enterprise_01
Enterprise System: ChassisID=14  & Server id = 0
Enterprise System: ChassisID=13  & Server id = 0
Enterprise System: ChassisID=12  & Server id = 0
Enterprise System: ChassisID=10  & Server id = 0
Enterprise System: ChassisID=11  & Server id = 0
-----------------------------------------
Initialization of Enterprise System Name=Enterprise_02
Enterprise System: ChassisID=4  & Server id = 0
Enterprise System: ChassisID=3  & Server id = 0
Enterprise System: ChassisID=9  & Server id = 0
Enterprise System: ChassisID=8  & Server id = 0
Enterprise System: ChassisID=0  & Server id = 0
Enterprise System: ChassisID=1  & Server id = 0
Enterprise System: ChassisID=2  & Server id = 0
Enterprise System: ChassisID=5  & Server id = 0
-----------------------------------------
Initialization of Interactive System Name=Inter_01
Interactive system: ChassisID= 34 & Server= 0
Interactive system: ChassisID= 33 & Server= 0
Interactive system: ChassisID= 32 & Server= 0
Interactive system: ChassisID= 30 & Server= 0
Interactive system: ChassisID= 31 & Server= 0
-----------------------------------------
Initialization of HPC System Name=HPC_01
HPC System: ChassisID=39  & Server id = 0
HPC System: ChassisID=19  & Server id = 0
HPC System: ChassisID=49  & Server id = 0
HPC System: ChassisID=24  & Server id = 0
HPC System: ChassisID=44  & Server id = 0
HPC System: ChassisID=18  & Server id = 0
HPC System: ChassisID=38  & Server id = 0
HPC System: ChassisID=48  & Server id = 0
HPC System: ChassisID=43  & Server id = 0
HPC System: ChassisID=28  & Server id = 0
HPC System: ChassisID=23  & Server id = 0
HPC System: ChassisID=29  & Server id = 0
HPC System: ChassisID=45  & Server id = 0
HPC System: ChassisID=46  & Server id = 0
HPC System: ChassisID=47  & Server id = 0
HPC System: ChassisID=40  & Server id = 0
HPC System: ChassisID=22  & Server id = 0
HPC System: ChassisID=41  & Server id = 0
HPC System: ChassisID=37  & Server id = 0
HPC System: ChassisID=42  & Server id = 0
HPC System: ChassisID=27  & Server id = 0
HPC System: ChassisID=25  & Server id = 0
HPC System: ChassisID=26  & Server id = 0
HPC System: ChassisID=17  & Server id = 0
HPC System: ChassisID=35  & Server id = 0
HPC System: ChassisID=21  & Server id = 0
HPC System: ChassisID=20  & Server id = 0
HPC System: ChassisID=36  & Server id = 0
HPC System: ChassisID=16  & Server id = 0
HPC System: ChassisID=15  & Server id = 0
------------------------------------------
Systems start running
------------------------------------------
finishing Time Interactive sys:  Inter_01 at time: 29
Interactive sys: Number of violation: 25
Computing Power Consumed by  Inter_01 is: 36288.6
--------------------------------------
finishing Time HPC_Sys:  HPC_01 at time: 1182
Total Response Time= 17884.0
Number of violation HPC : 187
Computing Power Consumed by  HPC_01 is: 6930046
--------------------------------------
Number of violation in 1th application=  0
finishing Time EnterSys: Enterprise_01 at time: 65581
Computing Power Consumed by  Enterprise_01 is: 4.006354939999967E7
--------------------------------------
Number of violation in 1th application=  35824
finishing Time EnterSys: Enterprise_02 at time: 73934
Computing Power Consumed by Enterprise_02 is:
4.9452025600000024E7
------------------------------------------
Total energy Consumption= 5.3123816702378535E8
LocalTime= 73934
Mean Power Consumption= 7185.30
```

Number of violation in output listing for each system is number of time that system encounter SLA violation based on definition has come in their configure file.

Output file related to SLA violation of HPC system

System Name	Time	SLA violation type
HPC_01	5	ComputeNodeShortage
HPC_01	6	ComputeNodeShortage
HPC_01	7	ComputeNodeShortage
HPC_01	8	ComputeNodeShortage
HPC_01	9	DEADLINEPASSED
HPC_01	9	ComputeNodeShortage
HPC_01	10	DEADLINEPASSED
HPC_01	10	DEADLINEPASSED
HPC_01	10	ComputeNodeShortage
HPC_01	11	DEADLINEPASSED
HPC_01	11	DEADLINEPASSED

HPC_01	11	ComputeNodeShortage
HPC_01	12	DEADLINEPASSED
HPC_01	12	DEADLINEPASSED
HPC_01	12	ComputeNodeShortage
HPC_01	13	ComputeNodeShortage
HPC_01	14	DEADLINEPASSED
HPC_01	14	DEADLINEPASSED
HPC_01	14	DEADLINEPASSED
HPC_01	14	ComputeNodeShortage

Output file related to SLA violation of Interactive system. SLA violation here is total number of all workload belong to all user in the system where have more response time than ERT defined in user configuration file.

System Name	Time	SLA Violation of all users
Inter_01	3	1
Inter_01	4	1
Inter_01	5	1
Inter_01	6	2
Inter_01	7	1
Inter_01	8	1
Inter_01	9	2
Inter_01	10	1
Inter_01	11	1
Inter_01	12	2
Inter_01	13	1
Inter_01	14	1

Output file related to SLA violation of enterprise system, last column is summation of all SLA violation percentage for all applications inside the system determined in application definition.

System Name	Time	SLA violation of all application
Enterprise_02	37978	3
Enterprise_02	37979	9
Enterprise_02	37980	12
Enterprise_02	37981	11

Enterprise_02	37982	12
Enterprise_02	37983	6
Enterprise_02	38116	2
Enterprise_02	38117	8
Enterprise_02	38118	14
Enterprise_02	38119	22
Enterprise_02	38120	24
Enterprise_02	38121	22
Enterprise_02	38122	29
Enterprise_02	38123	33
Enterprise_02	38124	38
Enterprise_02	38125	40

References

[1] http://www.techarp.com/showarticle.aspx?artno=420&pgno=7

[2] HP Proliant DL3XX power specification:
http://h50146.www5.hp.com/products/software/oe/linux/mainstream/support/whitepaper/pdfs/c03231802.pdf

[3] Intel CPU overclocking vs. power consumption:
http://www.xbitlabs.com/articles/cpu/display/power-consumption-overclocking.html

[4] Standard Performance Evaluation Corporation. http://www.spec.org/power_ssj2008/results/power_ssj2008.html.

[5] Joint optimization of idle and cooling power in data centers while maintaining response time - Faraz, Vijaykumar – 2010

[6] Gupta, Sandeep KS, et al. "GDCSim: A tool for analyzing Green Data Center design and resource management techniques." *Green Computing Conference and Workshops (IGCC), 2011 International*. IEEE, 2011.

[7] http://impact.asu.edu/BlueTool/wiki/index.php/BlueWiki

[8] J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle, “Managing energy and server resources in hosting centers,” in Proc. of the eighteenth ACM symposium on Operating systems principles (SOSP 01), 2001, pp. 103–116.

[9] Z. Abbasi, G. Varsamopoulos, and S. K. S. Gupta, “Thermal aware server provisioning and workload distribution for Internet data centers,” in ACM International Symposium on High Performance Distributed Computing (HPDC10), 2010, pp. 130–141.