# Autonomic Management for Energy Efficient Data Centers

Forough Norouzi
Computer Science department, Western University
London, Canada
e-mail: fnorouz@uwo.ca

Michael Bauer
Computer Science department, Western University
London, Canada
e-mail: bauer@uwo.ca

***Abstract*—The complexity of today's data centers has led researchers to investigate ways in using autonomic methods for data center management. In this work, we consider using autonomic management techniques that can help reduce data center energy consumption. In particular, we consider policy-based, multi-level autonomic management for energy aware data centers. We advocate for a hierarchical model of managers with loosely coupled communication between them. We describe our manager topology, communications and manager operations. We implement our approach for high performance computing centers that may have one or more large high performance computing systems. A data center simulator has been implemented that calculates data center energy consumption. We evaluate different management policies and our approach using this simulator. Preliminary experiments show promising results in terms of minimizing energy consumption and overhead on service level expectations in high performance computing systems.**

***Keywords- autonomic computing; energy aware data center; self-management system; policy-based management.***

## I. INTRODUCTION

Today's data centers are large, complex and challenging to manage. One of the central challenges in data center management and operations is energy management. Data centers at the core of Internet-scale applications consume about 1.3% of the worldwide electricity supply, and this level is predicted to increase to 8% by 2020 [1]. Google alone, for example, consumed 2.26M MWh in 2010 [4]. Carbon emissions from data centers alone in November 2008 were 0.6% of the global total and predicted to be 2.6% by 2020 which is more than the total carbon emission of Germany [3]. Given these statistics, reducing the energy consumption of data centers and making them work in an energy-aware manner is a major topic of data center management research. Broadly, research into energy efficiency in data centers can be categorized into a number of areas. Server level energy management approaches take advantage of lower power states built into components e.g. CPU(Central Processor Unit) and memory. At the level of clusters, management models aim to use optimization and control theoretic approaches to optimize the number of required compute node for each running application. Virtualization looks at reducing the number of active physical servers by multiplexing them as virtual machines (VM) where having fewer physical servers means that other servers can be turned off or maintained in a low power state. Thermal aware scheduling considers energy consumption criteria for job scheduling and resource allocation. However, there are few approaches looking into overall holistic strategies and automated methods to support administrators. One strategy is to consider approaches based on autonomic management, particularly policy-based autonomic management, where part of the role of the administrator would be codifying management policy for data center operations. Autonomic Computing (AC) aims to embrace the notion of self-management in distributed and complex systems where administrator intervention in system management is reduced or minimized. Instead, administrators define the overall policy and strategy for system management according to system organizational objectives. Self-management based on use of policies is referred to as *policy-based management*; it is a promising approach for developing autonomic management in complex distributed systems.

We advocate for multiple autonomic managers rather than having a single centralized autonomic manager that could be a single point of failure and potential performance bottleneck. To the best of our knowledge, policy-based autonomic management utilizing multiple managers for energy aware data centers is only marginally addressed in previous research. The proposed management system focuses on multilateral interaction in a multi-agent autonomic computing environment where autonomic managers interact with each other in a hierarchical structure. Intuitively, a hierarchical arrangement of managers would seem to provide good scalability while keeping communication overhead low and some previous research has suggested the utility of hierarchical management [4][14]. A hierarchical approach also matches well the hierarchy of computational elements in the data center.

This paper organized as follows. Section 2 provides an overview of related work. Aspects of data center management and our proposed management system architecture will follow. The data center simulator and a number of implementation scenarios for a simple data center are in Section 5. We conclude with a discussion on management overhead and future plans.

## II. RELATED WORK

Autonomic Computing (AC) refers to the idea of a computing system or application being self-managing, that is, a system that can manage itself in such a way that it is adaptable to any changes in the system environment [6]. In the autonomic computing paradigm, a management module

 ISBN: 978-1-61208-388-9

which controls the behavior of a managed element (ME) is called an autonomic manager (AM). The managed element provides some sensors and actuators to the manager. The manager monitors available metrics through these sensors and analyses the monitored information. It can then plan for a series of actions that need to be taken, if any, and execute those actions through the provided actuators. This process is a feedback loop called the Monitor-Analyze-Plan-Execute (MAPE) loop [10]. In AC, different AMs control different resources in a distributed manner. This management could be done individually, i.e. each AM is responsible for its own MEs. More generally, in computing systems it is necessary that AMs interoperate. There may be heterogeneous types of AMs that may have different objectives. Research by Mukherjee [13] illustrates coordination between two independent AMs where the first AM deals with service level agreement (SLA) management and resource allocation, while the second AM deals with minimizing power consumption by turning off unused servers. Their work shows that the interaction between the managers is important in achieving the goals.

Khargharia et al. [5] introduced a three-level hierarchy for optimizing energy consumption and SLA violations. The hierarchy starts from the device level inside a server, proceeds to the server level and then encompasses the cluster level. Decisions are based on the power status of each managed element at each level. Their idea illustrates the value of a hierarchical approach, but needs some modification to be applicable for large scale data centers which may have many different types of applications and services. Anthony et al. [26] identify collaboration as a key aspect and suggest that AMs should be designed for collaboration and that the lack of collaboration between managers is a problem. Then, the authors attempt to tackle AM interoperability issues and define an interoperability service. The interoperability service keeps a database of registered AMs along with corresponding resources they manage and scope of their management operation. The interoperability service will detect potential conflicts and send messages to related AMs to, for example, suspend or stop their activities. Kusic et. al. [16] described an autonomic cluster management framework. They defined three different types of agents: general agents (implemented per node), optimization agents, and configuration agents (implemented per implementation of the management framework). The proposed management infrastructure is a hybrid of centralized and decentralized and communication between agents is done via message passing. Complexity of task distribution between agents makes it un-scalable for large scale environment. Kennedy [9] argues that the mechanism that defines interoperability between autonomic elements must be reusable and generic enough to prevent complexities. A standard means must be defined to exchange context between autonomic elements. This meta level needs to be context-aware. At this step, they have identified the main challenges for automated recovery in autonomic system. Thomas, et. al. [22] presented a management framework for the automated maintenance cycle in the computing cluster (part of the Data Grid project [23]). A number of management modules, e.g. job management, monitoring, fault recovery, and configuration management, have been defined where each produces information as an output which is used as input for others. Their system gets configuration states from an administrator. Each machine has a goal state which is stored in a configuration database and also has an actual state which comes from the monitoring agent. These states are compared within the fault detection and recovery system for any mismatch, which then applies any necessary actions to fix them. The fault detection system has its set of rules (policies) for each node where these rules are checked. A decentralized architecture, Unity, was introduced in [24]. Unity introduces a two level management model that tries to allocate optimized resources (servers) to different types of application environments running both batch type and interactive workload across the whole data center.

Policy Based Management (PBM) is a management paradigm that separates governing rules from the main functionality of the managed system. Bahati et al. [18] described an architecture for autonomic management and demonstrated how policies are defined and mapped to their corresponding elements. The authors of [13] propose a Model-driven Coordinated Management architecture to make dynamic management decisions based on energy benefits of different policies to handle events. They used a workload model, power model, and thermal model to predict the impact of different management policies. A central management unit monitors events, chooses the best policy and makes decisions.

## III. A MODEL FOR DATA CENTER MANAGEMENT SYSTEM

One can think of the AMs and their relationships as a kind of management overlay network on top of the elements of the data center. The actual position of management modules might be on single physical server, or even distributed over a number of servers. Number of essential questions need to be addressed before implementing the management system. For instance: what are managed objects in the data center? What metrics of an object should or could be monitored? And what are possible actions that the management system could take to control that specific object? To develop a management system, which contains a dynamic number of managers for a data center, several issues need to be addressed:

Topology of the AMs: AMs are more likely to have their own overlay network, with a specific protocol to communicate and exchange information e.g. SOAP (Simple Object Access protocol). The topology has implications for the coordination and communication among AMs and the decomposition of management tasks among AMs.

- Hierarchical management means that some AMs can monitor and influence or control the behavior of other AMs. In this case, lower AMs are considered as managed elements for the higher AM. AMs at different levels usually work at different time scales. In this topology the upper layer AM regulates and orchestrates the system by monitoring parameters of all of its lower level AMs (see Figure 1). The upper layer AM is privileged over lower

layer AMs, and has the authority to control or manipulate some parameters of the lower level AMs.

- A peer to peer topology entails AMs that can directly communicate with one another, exchange information and make decisions. In this paradigm, all AMs are often equally privileged.
- Indirect coordination between AMs involves an AM making changes in its MEs which are then sensed by other AMs causing them to perform actions. There is no direct communication between the AMs. Since the MEs (e.g. application, services, and virtual machine) may change over time (e.g. is finished or started), there should be a way such that the topologies of corresponding AMs can change on the go.

Collaboration Strategy: Depending on the topology, the next question is how AMs influence other AMs in the management system? How much information do they need to share? What kind of information? For example, one AM may be privileged over some set of other AMs because its management scope is wider than the others or it has more information about its surrounding environment. Alternatively, all AMs could be acting the same, e.g. as in a peer-to-peer topology. Finally, what is the nature of AMs interaction and coordination?

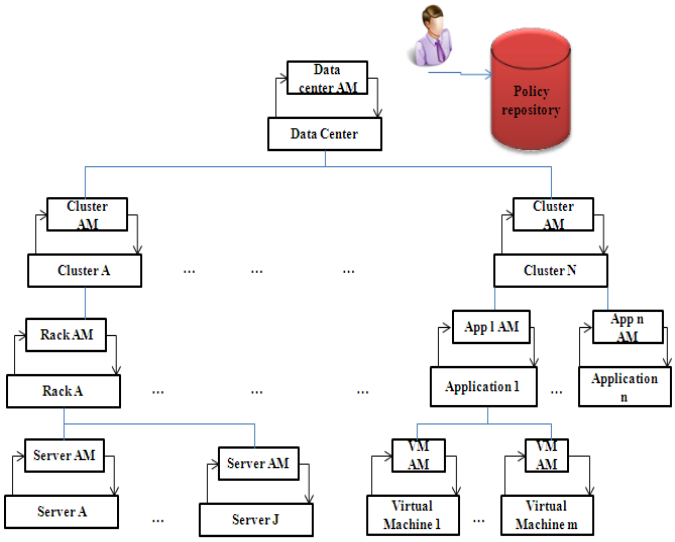


Figure 1. Hierarchical Policy-based Management. System.

Manager Life Cycle: An autonomic manager has its own life cycle which obviously corresponds to the life cycle of its associated MEs. For example, for a cloud user renting compute nodes and running an application for a period of a time, the corresponding AM is born and dies along with the application life cycle. One of the issues in multilevel management systems is that each level of the management model has to have the ability to create AMs based on the respective ME life cycle, then introduce it to the management system, and then destroy it at the end.

The overarching management approach assumes that the management system will associate an AM with the new arrival ME; this could be done automatically if the administrator specified a particular AM for that class of ME or could be just done manually by the administrator. We expect that in many cases, the MEs can be grouped into broad classes and, correspondingly, that AMs will be as well. In our approach, we assume classes of AMs that have similar requirements and characteristics will be defined and associated with classes of MEs. Table I. illustrates samples of MEs and corresponding classes of AMs. The management system refers to this table upon the initialization

TABLE I. CLASSES OF AMS AND MES

| ME class | Generic associated AM |
|---|---|
| VM | VM_AM.class |
| Cluster | clusterAM.class |
| Rack | rackAM.class |
| Application<br>-interactive<br>-Enterprise | <br>appIAM.class<br>appEAM.class |

or creation of a new ME to check which class of AM should be initiated for that particular ME. Part of the AM initialization is to identify its parent and to get its policies.

The proposed management system is policy-based which means that each AM has its own set of ECA (Event Condition Action) policies referred to as a policy profile. This set can be altered according to the system situation or even a direct change from the data center administrator. In our model, the parent AM can also make decisions regarding the policy profile of its children as part of its planning task. Policy repository holds the policy profiles associated with each of the managed element classes. Generally, though, we would expect that there would be much overlap, e.g. a policy for managing an application during a work day would be very similar to the policies for managing it at night or on a weekend.

### A. *Management System Configuration*

An administrator first needs to decide about the number of management levels in the management system and then the position of autonomic managers. For a given data center, an administrator may define the number of management levels and for each level the position of managers. Since different types of applications may come and go, there will be a dynamic number of MEs and, respectively, a dynamic number of AMs in each level. Upon arrival of any new application in the data center, the AM initiation module has to be invoked. During the AM initialization procedure, a unique ID is generated (for example, as a combination of an IP address of the host where the AM will run, the parentID or any local variables) for the AM. The AM also needs to have access to the policy repository. The policy repository server contains all policies for the AMs in the management system. The first time that an AM has access to the policy repository is at its bootstrapping phase to get initialized, although during its life cycle the AM may be asked by its parent to access the repository and get updated policies from there.

After AM initialization in which all environmental variables are initialized, the management loop starts to run

 ISBN: 978-1-61208-388-9 

(see Figure 2.). The AM management loop uses the monitoring heartbeat values of its managed elements and checks for incoming messages. Messages correspond to events and a timing event happens periodically. As events occur, policies are examined and the values of the parameters are used to evaluate conditions in policies. We assume that the upper layers AMs are privileged over their child and so their policies are affected by their parent's policy. For instance, the parent can change a child AM's policy profile to "green", which could mean that the AM should give higher priority to decreasing energy consumption of its MEs than to ensuring that the SLA violations are minimized.

## IV. DATA CENTER SIMULATOR

We have been developing a data center simulator [19] [28] in order to evaluate different configurations of autonomic managers and different policy sets. Our data center contains a set of systems (its definition follows) where each system runs its own kinds of applications. We can think of a data center abstractly as consisting of a number of racks $R=\{r_1, r_2, \ldots, r_R\}$ and coolers $\{c_1,c_2,\ldots,c_k\}$ laid out in some spatial configuration pattern with some network connections among them (multiple separate clusters are each collections of racks, with perhaps no communication between the racks in different clusters). Each rack is comprised of a number of chassis, and within each chassis there are numbers of servers (compute nodes).

```
Input parameters: ParentID, AMLevel, ProfilePolicy, ME, heartbeat,
        heartbeatValue, configVector, configVectorValue
1.  begin
2.    update heartbeat value
3.    While (!messageQue.isEmpty())
4.    begin
5.        msg=messageQue.dequeue
6.        if (msg.opcode== ReqForHeartbeat)
7.          send(UpdateHeartbeat, ParentID,AMID, heartbeatValue>
8.        if (msg.opcode== ChangeProfilePolicy)
9.          Update ProfilePolicy with received one in the message
10.       if (msg.opcode== PolicyChange)
11.         Update the policy received in the message with the one that
      already is in AM policy set
12.       if (msg.opcode== UpdateConfig)
13.         Update corr. param. in configVectorValue with parameter in
      the message
14.   end
15. //all triggered event are put in a queue
16.   while (!eventQu.isEmpty() )
17.   begin
18.       EV= eventQu.dequeue
19.       for (all PL∈ ProfilePolicy)
20.       begin
21.         invoke applied policy
22.       end
23.   end
24. end
```

Figure 2. Management Loop

On top of this physical infrastructure, we have defined a System; our terminology for a number of computes nodes inside a number of racks, which are capable of running the same type of jobs. We assume that systems are defined in terms of a set of racks. Compute nodes inside racks can be shared between different applications that run on that system. At any time, a node $n_i$ inside the system is either assigned to an application/user or ready to be assigned; also, individual nodes can be powered on or off (put to sleep).

Application behavior and workload are key elements in data center operations and have a direct impact on the energy consumed by a system and hence a data center. In our model, we consider three broad classes of applications:

Interactive: This system provides access to users across the Internet/intranet, such as web servers, transactional servers, etc. These applications process short requests (transactions) and fast response time is the main objective of these types of applications. We model this as an InteractiveSystem.

Enterprise: This system provides applications to different business units, where applications may require large amounts of secure, reliable data storage and high availability, running 24/7, e.g. a human resources 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.

High performance computing (HPC): This system runs scientific applications in batch mode and typically needs multiple CPUs to do high computation jobs.

With this definition in mind, we can think of a data center as a set of systems running different types of workloads, so our logical model of a data center is:

$DC=\{sys_1,sys_2,\ldots,sys_i\}$ where $sys_i$ is a system and a system, then, is defined as:

*<Name, RA, Sch, Rack-list, Node-list, App, AM>*

where:

*Name*: is a system id.

*RA*: is resource allocation algorithm assigned to the system. Assigning any compute node to the application is done by this algorithm. Anytime that management polices force an application to release/ allocate a compute node this algorithm will decide.

*Sch*: is a scheduling algorithm for all applications running in the system. It has just one output which is next job (any type) to be run.

*Rack-list*: a list of racks assigned to this system.

Node-list: a list of compute nodes that are assigned to this system. There may be situations which a rack is shared between a number of system. In this case list of each system compute node is important.

*App*: specifies the type of applications that run on the system; Enterprise (Ent) applications, Interactive Applications (Int) or HPC applications.

*AM*: autonomic manager attached to this system.

The applications that run on a particular system are described as follows:

- *Ent*: An Enterprise system has a number of applications, each application having an interactive type workload running on a list of servers and its own SLA violation description.
- *Int*: An Interactive system deals with a list of dynamic coming-going workload from users. This type of

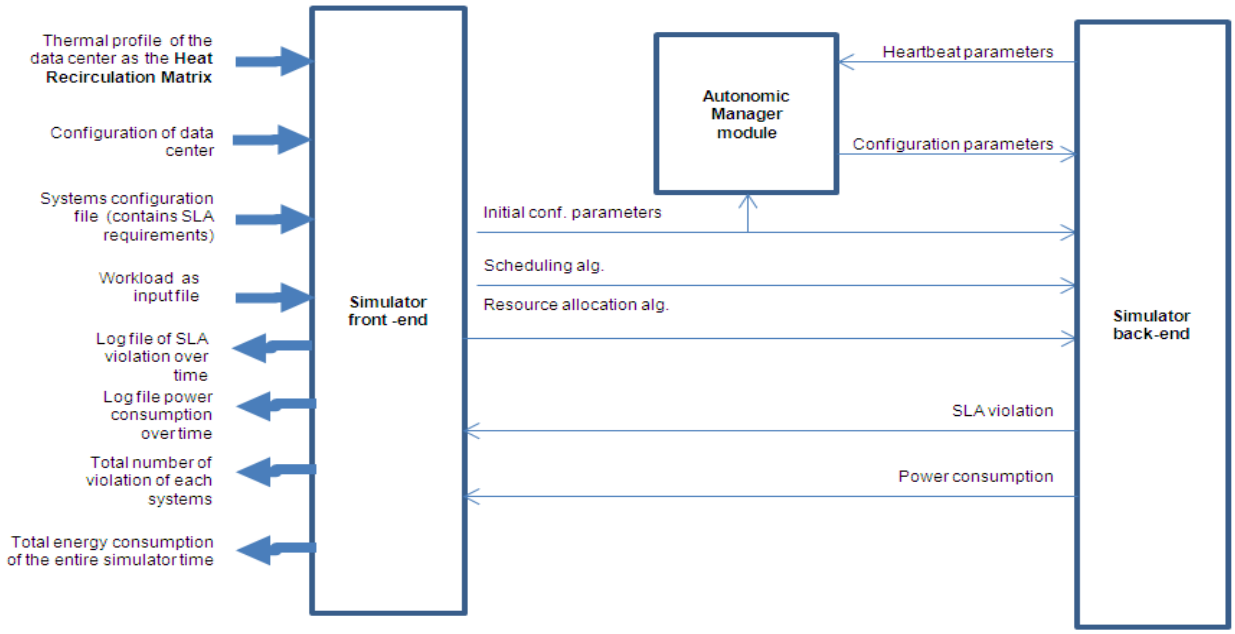


Figure 3. Overall Structure of Simulator

workload has arrival time, duration, and SLA violation definition. They are web based type applications.

- *HPC*: An HPC system just has HPC type jobs; each job has a duration, deadline, needed CPU utilization and number of nodes (for parallel processing jobs).

Putting this information together, a data center is then defined as:

<*RackList, Cooler, SysList, RedTemp, ThermalMap, AM*>

where:

*RackList*: is list of racks in the data center; information regarding chassis and blade servers inside the rack is part of the rack definition.

*Cooler*: is the cooling specification. The efficiency of the Computer Room Air Conditioner (CRAC) depends on air flow velocity and conductivity of materials which is quantified as the Coefficient of Performance (COP).

*SysList*: is the list of defined systems.

RedTemp: Red temperature: the maximum temperature that hardware in the data center can tolerate; this parameter will affect the cooling energy consumption.

*AM*: is the manager of the whole data center.

*thermalMap*: is used to calculate energy consumption of the data center; the thermal model used in this research was developed by Arizona State university [20]. Briefly, the computing and cooling power in data center are considered where the thermal model is a matrix, where an entry in the matrix specifies how much generated heat from each server will re-circulate to other servers. The overall structure of our simulator is presented in Figure 3. The illustrated autonomic management module is in charge of coordination and planning among different AMs across the data center. The simulator has been evaluated with different types of systems.

## V. EXPERIMENTS

To illustrate the impact of our proposed management

TABLE II. SLA PROFILE FOR AM ATTACHED TO HPC SYSTEM/WEBSERVER

| |
|---|
| PL0: |
| On Event: **Timer Triggered** |
| If (SLA is violated) |
| begin |
| (Increase freq. of all busy nodes → Activate all sleep nodes |
| End |

TABLE III. GREEN PROFILE POLICY FOR AM ATTACHED TO HPC SYSTEM/WEBSERVER

| |
|---|
| PL1: |
| On Event: **Timer Triggered** |
| If (SLA is violated) |
| begin |
| (Increase freq. of just fully utilized CPU node → Activate just half of sleep nodes) \| Activate just half of sleep nodes |
| end |
| PL2: |
| On Event: **Timer Triggered** |
| If (SLA is not violated) |
| begin |
| (Decrease freq. of all nodes → If node is ready and is not used make it sleep) \| If node is ready and is not used make it sleep |
| End |

TABLE IV. AM ATTACHED TO COOLER

| |
|---|
| PL3: |
| If (Max temperature is greater than Red temperature ) |
| begin |
| Send UpdateHeartbeat message to AM in DC level |
| End |

 ISBN: 978-1-61208-388-9 

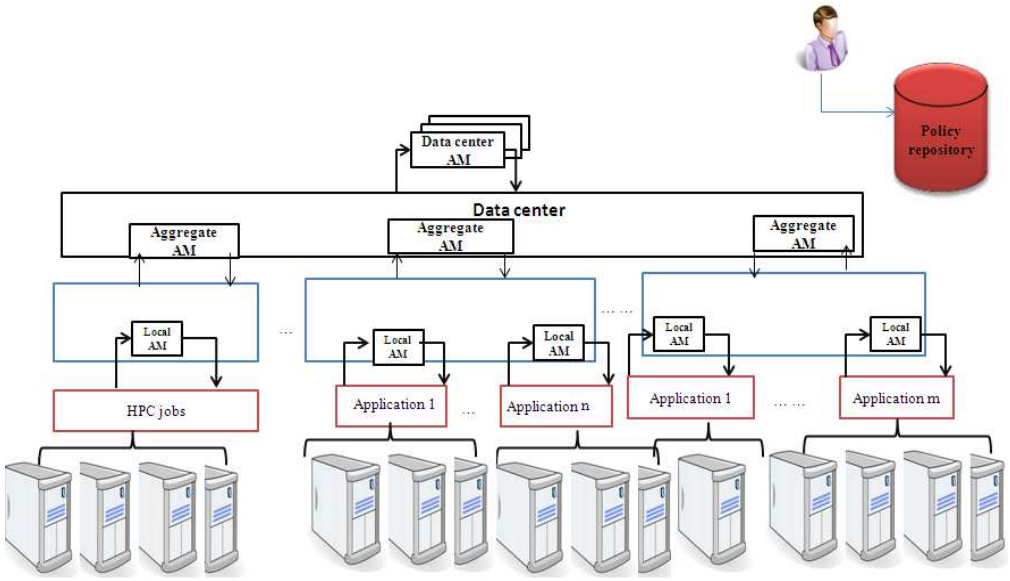


Figure 4. Prototype Management System

system, we evaluate different management scenarios for a hypothetical data center with and without management system. We describe two sets of experiments. In the first set we look at the impact of policies in managing the behavior of a hosted webserver in the simulated data center. In the second experiment, we present our prototype management system. We compare the effects of different policy profiles on energy consumption and SLA violations. For this experiment, we consider a data center with one or two HPC systems.

### A. *Experiment 1: Webserver Management*

Our simulated data center has 10 racks and in each rack there are 5 chasses, each has 5 servers (in total our data center has 250 compute nodes). The simulated servers are Proliant HP DL320. This server has a standby power consumption of 5Watts; when it is idle it consumes 100 Watts and, with a fully utilized CPU, it consumes 300 Watts [24][25]. The DL320 has an Intel® Xeon® E3-1200v2 processor which has 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 simulated data center has one HP cooler (refer to the thermal model in IV).

In this experiment, we host a webserver in the data center with 80 (minimum) to 90 (maximum) compute nodes allocated to the webserver. Compute nodes are allocated to the webserver. Workload is scaled up version of traffic from 1999 world cup web traffic. We have attached a manager to the webserver which monitors the SLA and, according to its active polices, does some actions. SLA is violated when the response time is more than two simulator cycles. Two distinct sets of policies are considered: a green profile and SLA profile (see Table II. and Table III.). These policies are trying to minimize energy consumption (Green policy profile) and minimize SLA violations (SLA policy profile). The SLA policy profile is a time-triggered policy (every 60 seconds). When triggered, the AM checks for any SLA violations in the system and tries to do dynamic CPU frequency scaling and activate sleep allocated compute nodes. If frequency scaling is not supported by compute nodes, this policy just activates sleeping nodes. The simulator counts violations during policy timer period. Green policy profile also tries to do dynamic frequency scaling and activation/deactivation of compute nodes if SLA violations happen. This profile tries to keep active compute nodes and CPUs at moderate frequency levels based on whether there are SLA violations or not. Results show (Table V.) that the Green policies result in less consumption of power than SLA based policies. In SLA based polices; the total energy consumption (cooling and computing) is not available since the inlet temperature is exceeded 475 times and this is not handled in the simulator. The main objective of this experiment is to show the scalability of the developed data center and also the impact of policy based management.

### B. *Prototype Management Environment*

We have modeled our prototype management system (illustrated in Figure 4.), using a three level hierarchy. At the bottom level, we have local AMs. Each local AM is attached to a number of compute nodes. The second level of AMs are called aggregate AMs; they logically aggregate management responsibility from the local level to the data center level. AMs at this level have the AMs at the first

TABLE V. COMPARISON BETWEEN GREEN AND SLA PROFILE POLICY IN SEMI-LARGE SIMULATED DATA CENTER

| Scenario | Green | SLA |
|---|---|---|
| Computing power of Webserver | 7.7 * 10^8 | 9.6*10^8 |
| total energy consumption (Watt × Simulation Time) | 1.9*10^9 | N/A |
| Mean power consumption (Watts) | 26982 | N/A |
| Number of times crossing red temperature | 0 | 475 |

level as their managed elements. At the top level, there is one (or more for replication) Data Center AM which is the coordinator among all aggregate AMs. AMs in our management system cooperate and so exchange information with the other AMs in the level above or below. This information is called heartbeat and configVector data that is the sensor and actuator information. Heartbeat information can be fetched by the parent periodically or upon the occurrence of any event, i.e. specified in the event part of AM policy set, e.g. on an SLA violation or power cap violation. Aggregate level or data center level AMs may inquire of their children for heartbeat updates to make better decisions. Any changes in configuration parameters or policies of the child are then sent from the parent AM as configuration parameters. We have simulated this prototype for just HPC type workload. We have attached an AM (local AM) to a HPC system and for number of HPC system (in our case we have two) we have a data center level AM which manages HPC systems behavior beside other AMs (the AM attached to the cooler in following it will be explained) in the data center.

### C. Experiment 2: HPC Data Center

In this experiment, we assume that each chassis has one blade server, so, in total, the data center has 50 physical servers configured into two separate HPC systems; one of 30 compute nodes and one of 20 compute nodes. Each of our HPC systems runs an HPC workload consisting of long and short batch jobs (their workloads are not the same). Each job in the workload has an arrival time, duration, needed CPU utilization (will be used for thermal model) and deadline (maximum waiting time in the system before dispatching to a compute node). An SLA violation occurs when a deadline is passed for a job in the workload. Two system workloads have 730 and 173 jobs respectively, which on average demand 3 compute nodes [27].

Ponder-like [21] notation has been used to describe our policies. For the simulation study, we assume, we have two policy profiles at the system level: a Green and SLA policy profiles as per our webserver experiment. We also have a data center AM which can change the system level AM's policy profile based on their SLA violation status. If there is any SLA violation on any of the HPC systems, the policy tries to change the policy profile of all systems based on whether they have an SLA violation (change it to SLA) or not (change to Green) (see policies PL4 and PL5 in Table VI). Upon any SLA violation in any system, they will send their heartbeat (SLA violation) to the data center AM. Data center AM will evaluate its policies and change any system with a violation to an SLA based policy and the rest of the systems will be set to Green. We expect that, by changing policy profile of system dynamically, we can get better results in terms of total energy consumption and still limit the number of SLA violations. Data center AM has policy to deal with the Cooler - if the AM in the cooler detects a red temperature then it sends message to the data center AM (see Table IV.). There is a policy for the data center AM to "block" an HPC system with lowest priority for a period of time (timer is set to 2 minutes) to relief data center load. What we have simulated for blocking HPC system is not running any jobs from the workload and in case of new arrival jobs just queuing them and not dispatching them to the compute nodes.

### D. Experimental Scenarios

Five different scenarios have been considered to evaluate the performance of having multiple autonomic managers with varying sets of policies.

***Scenario 1.*** No management: The data center has the two running HPC systems, one with 30 compute nodes and a workload of 730 jobs and another system with 20 compute nodes and a workload with 173 jobs.

***Scenario 2.*** There is a manager at the system level, which has a *SLA* policy profile (see Table II.SLA Profile For AM Attached to HPC System/Webserver ). This scenario runs for the small HPC system of 20 compute nodes (we assume that the large HPC system is not running

TABLE VI. GREEN POLICY PROFILE FOR DC LEVEL AM

| | |
|---|---|
| PL4:<br>On Event: (**SLAViolation$_1$ \| SLAViolation$_2$**)<br>if (SLA$_1$ is violated)<br>begin<br>Switch system$_1$ to SLA based<br>end<br>else<br>begin<br>Switch system$_1$ to Green<br>end | PL3:<br>On Event: **"block timer" trigger**<br>if (true)<br>begin<br>Unblock the blocked system<br>End<br><br>PL5:<br>On Event: (**SLAViolation$_1$ \| SLAViolation$_2$**)<br>if (SLA$_2$ is violated)<br>begin<br>Switch system$_2$ to SLA based<br>end<br>else<br>begin<br>Switch system$_2$ to Green<br>end |
| PL6:<br>On Event: **Receiving UpdateHeartbeat from Cooler**<br>if (true)<br>begin<br>(block HPC system with lowest priority ->start a timer: "block timer")\| Switch strategy of all others to Green<br>end | |

TABLE VII. COMPARISON BETWEEN DIFFERENT SCENARIOS

| Scenario | No management system (Scenario 1) | Single AM in system level (Scenario 2 and 3) | | Multiple AMs ( data center level and system level) (Scenario 4 and 5) | |
|---|---|---|---|---|---|
| Profile Policy | N/A | Green | SLA | DC AM Profile policy is Green | |
| Num. of HPC systems in DC | 2 | 1 | | 1 | 2 |
| Number of SLA Violation | 448 | 189 | 187 | 189 | 454 |
| total energy consumption (Watt × Simulation Time) | N/A | 8,000,000 | 9,800,000 | 8,318,000 | 23,171,143 |
| Mean power consumption (Watt) | N/A | 6,430 | 7,287 | 6,518 | 11,956 |
| Number of time crossing red temperature | 91 | 0 | 0 | 0 | 3 |
| Number of exchanged messages | 0 | 226 | 22 | 253 | 2,764 |

in the data center). The goal here is to evaluate the impact of the *SLA* policy profile on power and performance.

***Scenario 3.*** Exactly as scenario 2 except the *Green* policy profile for system level AM.

***Scenario 4.*** We have two managers: one AM at the system level (the small HPC system is running) and data center level. The data center AM's policy profile is *Green* (see Table VI.). We aim to evaluate the impact of changing policy profiles of the HPC systems dynamically on the power and performance. The main goal in Scenario 4 is to consider how the data center level AM impacts the behavior of its lower level AMs. While data center level AM is green means that it makes the system level AM to behave close to when it is Green itself.

***Scenario 5***. We have both HPC systems with their AMs running, an AM at the data center level, and the cooler has its own manager that just checks for its maximum inlet temperature. If the inlet temperature is greater than the red temperature of hardware in the data center (specified in the data center configuration), the cooler AM sends a message (UpdateHeartbeat message) to the data center AM asking it to do something (refer to Table IV.). In this scenario, we assume that the policy available to the AM in the data center indicates that a system should be "blocked" – essentially decrease processing by not executing additional jobs. Obviously, the blocked system will suffer from SLA violations but the gain is that this decision addresses exceeding the red temperature for the whole data center. System priority is defined with the HPC system configuration.

### *E. Experimental Results*

The result of running these scenarios is shown in Table VII. The first scenario does not have a management module and has two HPC systems. Running these systems under the workloads results in the inlet temperature of the cooler exceeding the red temperature 91 times; as a result, the simulator is not able to calculate the total energy consumed. As shown in Table VII. , Scenarios 2 and 3 involve a single HPC system with a manager. The *Green* policy profile consumes less energy and power than the same HPC system with the *SLA* policy profile while the number of SLA violations is about the same. This scenario shows how a small difference in policies can affect the overall behavior. In Scenario 4, we consider an AM at the data center level and its policy profile is *Green*. The data center AM with the *Green* policy profile is configured to dynamically change the policy profile of system level AMs in accordance with the system's SLA violations; if there are SLA violations at the system level, its policy profile is altered to be *SLA* based in order to put more priority on achieving SLAs than on energy conservation. The result shows that by having a data center level manager able to dynamically switch its corresponding system level AMs profile we can get the same results as when the system level has Green profile policy. In Scenario 4, it is data center level AM that controls the behavior of the system level AMs and by setting data center to Green we implicitly mak system level AM to behave close to Green profile policy.

### *F. Management Overhead*

The management system is responsible for configuring managing entities and making sure they have updated context information. All communications are based on message passing, which causes network traffic. Although hierarchical architecture is expected to have less communication overhead, we consider the number of exchanged messages between managing entities as management overhead. As shown in Table VII. , the last scenario which has an AM attached to the cooler and dual levels of AM is expected to have more messages. These messages are passed between four different zones: within the two HPC systems, between the data center AM and the HPC system AMs, and between the cooler AM and the AM in the data center. The other expected overhead is due to the actual computing resources consumed for management activities (from the initialization of the management system to running the MAPE loop in each manager). We do not actually execute managers within the simulation, so we cannot get an estimate from the simulator itself. However, to estimate computing resources consumed by a manager, we ran a "pseudo-manager" that executed and timed a MAPE loop with 10 policies with fairly CPU intensive actions as well as accessing a file to simulate the reading of policies – something that would not normally happen each time through the MAPE loop. This manager was run on a computer of roughly the same computational power as the HP DL320. The result showed that this MAPE loop consumes 0.00002% CPU utilization of the processor which is essentially negligible. Even if our measure is off by a

 ISBN: 978-1-61208-388-9 

factor of a thousand, the management overhead to evaluate policies and initiate actions is small. Of course, the resources consumed in executing those actions could be substantial, e.g. a virtual machine migration, but this depends on the specific actions and what makes sense in the contact of managing the system.

## VI. CONCLUSION AND FUTRUE WORK

This research aims to develop an autonomic management system that can help reduce data center energy consumption while still adhering to service level agreements and performance expectations. The results of combining autonomic computing and policy based management suggest a useful approach. We considered a hierarchical arrangement of autonomic managers that is based on the physical position of managed elements. A general approach for an autonomic management system has been introduced. The core principles that drive the management model are: a message passing approach and policy-driving autonomous managers. The approach has been evaluated on a hypothetical data center using a simulator. The simulator in this experiment has 50 nodes but has the ability to be extended by increasing the number of blade servers in each chassis. (Here we have one blade server in each chassis; webserver experiment addresses scalability). The results show that, first, by having simple policies (such as SLA and Green policies), we obtain an acceptable reduction in power consumption. The second lesson learned is the impact of upper layer AM profile on the behavior of its lower level AM. As shown in scenario 4 upper layer AM being Green causes the same behavior as while the system level AM is Green. Comparison of Scenario 1 and 5 shows that the proposed three-level management hierarchy with given policies controls the behavior of the data center in terms of minimizing power consumption with negligible management overhead and effect on SLA violation. The result of having different profile policy and different level of management shown in this work are workload agnostic. We have run experiments with web-based workload that shows promising horizon for our management system. Future work will look at the management algorithms and dealing with changes in the computing environment, e.g. the dynamic start of or termination of applications. It will also explore means for more general cooperation between managers and for different configurations for the management system, e.g. peer-to-peer, etc.

## REFERENCES


[1] J. Koomey, Growth in data center electricity use 2005 to 2010. Analytics Press, August 2011.

[2] B. Anton, J. Abawajy, and R. Buyya. "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing." Future Generation Computer Systems 28.5 (2012): 755-768.

[3] W. Forrest, J.M. Kaplan, and N. Kindler, “Data centers: How to cut carbon emissions and costs,” mckinseyquarterly.com, November 2008.

[4] Google green, http://www.google.com/green Last visited July 2014.

[5] Kh. Bithika, S.Hariri, and M.S. Yousif. "Autonomic power and performance management for computing systems." Cluster computing 11.2 (2008): pp.167-181.

[6] Computing, Autonomic. "An architectural blueprint for autonomic computing." IBM White Paper (2006).

[7] J. D. Moore, J.S. Chase, P. Ranganathan, and R.K. Sharma. "Making Scheduling" Cool": Temperature-Aware Workload Placement in Data Centers." In USENIX annual technical conference, General Track, pp. 61-75. 2005.

[8] http://institutes.lanl.gov/hec-fsio/ Last visited July 2014.

[9] K. Catriona. "Decentralised Metacognition in Context-Aware Autonomic Systems: Some Key Challenges." In Metacognition for Robust Social Systems. 2010.

[10] R. Boutaba, and A. Issam. "Policy-based management: A historical perspective." Journal of Network and Systems Management 15.4 (2007): pp.447-480.

[11] J.O.Kephart, and D. M. Chess. "The vision of autonomic computing." Computer 36.1 (2003):pp. 41-50.

[12] T. Mukherjee et al. "Spatio-temporal thermal-aware job scheduling to minimize energy consumption in virtualized heterogeneous data centers." Computer Networks 53, no. 17 (2009): pp.2888-2904.

[13] T. Mukherjee, et al. "Model-driven coordinated management of data centers." Computer Networks 54.16 (2010): pp.2869-2886.

[14] J.O.Kephart. "Coordinating Multiple Autonomic Managers to Achieve Specified Power-Performance Tradeoffs." Fourth International Conference on Autonomic Computing, IEEE Computer Society, 2007.pp. 24-33.

[15] Z.Abbasi. et al. "Thermal aware server provisioning and workload distribution for internet data centers." In Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing,ACM 2010, pp. 130-141.

[16] D. Kusic, J.O. Kephart, J.E. Hanson, N. Kandasamy, G. Jiang, “Power and performance management of virtualized computing environments via lookahead control”, Cluster Comput. 12 (1) (2009) pp. 1–15.

[17] J.D.Baldassari, et al. "Autonomic cluster management system (ACMS): A demonstration of autonomic principles at work." Engineering of Computer-Based Systems, 2005. ECBS'05. 12th IEEE International Conference and Workshops on the. IEEE, 2005. pp. 512-518.

[18] R.Bahatl, M.A. Bauer, Elvis M. Vieira, and O. K. Baek. "Using policies to drive autonomic management." In Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks, IEEE Computer Society, 2006, pp. 475-479.

[19] http://publish.uwo.ca/~fnorouz/publication/Simulator.pdf. Last visited July 2014.

[20] http://impact.asu.edu/BlueTool/wiki/index.php/BlueSim. Last visited July 2014.

[21] http://www.ponder2.net/. Last visited July 2014.

[22] R. Thomas, et al. "Autonomic management of large clusters and their integration into the grid." Journal of Grid computing 2.3 (2004): pp.247-260.

[23] http://eu-datagrid.web.cern.ch/eu-datagrid/ Last visited July 2014.

[24] T. Gerald, et al. "A multi-agent systems approach to autonomic computing." In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1, IEEE Computer Society, 2004. pp. 464-471.

[25] Standard Performance Evaluation Corporation. http://www.spec.org/power_ssj2008/results/power_ssj2008.html. Last visited July 2014.

[26] A. Richard, M. Pelc, and H. Shuaib. "The interoperability challenge for autonomic computing." EMERGING 2011, The Third International Conference on Emerging Network Intelligence. 2011.

[27] http://institutes.lanl.gov/hec-fsio/ Last visited July 2014.

[28] https://github.com/fnorouz/simulator/ Last visited January 2015.


 ISBN: 978-1-61208-388-9 