Software Architecture for Dynamic Thermal Management in Datacenters

Tridib Mukherjee, Qinghui Tang, Corbett Ziesman, Sandeep K. S. Gupta
Arizona State University
Tempe, AZ 85287

Phil Cayton
Intel Corporation
Hillsboro, Oregon

***Abstract*— Minimizing the energy cost and improving thermal performance of power-limited datacenters, deploying large computing clusters, are the key issues towards optimizing their computing resources and maximally exploiting the computation capabilities. In this paper, we develop a unique merger between the physical infrastructure and resource management functions of a cluster management system to take a holistic view of datacenter management, and make global (at the level of a datacenter) thermal-aware job scheduling decisions. A software architecture is presented in this regard and implemented in a fully operational computational cluster in the ASU datacenter. The proposed architecture develops a feedback-control loop, by combining information from ambient and on-board sensors with the node allocation and job scheduling mechanisms, for managing the system load depending on the thermal distribution in the datacenter.**

I. Introduction

Computing clusters are increasingly deployed in current datacenters limited by power and thermal capacity. To achieve higher computation capability these datacenters are densely populated with computing servers. This results in high heat density in the datacenters producing potential hotspots. As a result the reliability and longevity of the computing servers is affected due to overheating, increasing possible downtime of the system. The current trend in datacenters to address this problem is to increase the cooling capacity of the datacenters taking into account the worst case scenarios. This, although solves the problem, gives rise to higher cooling cost which produces almost half of the total operational cost of the datacenter. Therefore, a dynamic thermal-aware control architecture is necessary for online thermal evaluation that can achieve a trade-off between these extremes. One of the major goals in this respect therefore is to answer the following question: "Given limited power and cooling capacities in a datacenter, how can a combination of ambient and on-board sensors and job scheduling manage the system load in such a way as to maximize throughput?"

To this effect, we present a software architecture which combines information from ambient and on-board sensors with resource management functions of a cluster management system to take a holistic view of datacenter management, and make global (at the level of a datacenter) thermal-aware job scheduling decisions. We implemented this thermal aware scheduling architecture in a fully operational computing cluster in the ASU datacenter [1] with a total of 4 racks, 5 chassis in each rack, and 10 Dell PowerEdge 1855 blade servers containing Intel 64-bit dual-processor Xeon EM64T CPUs in each chassis.

A typical datacenter is laid out with a hot-aisle and cold-aisle arrangement by installing the racks and perforated floor tiles in the raised floor. The air conditioners, normally referred to as Computer Room Air Conditioner (CRAC) or HVAC (Heating Ventilation Air Conditioner), deliver cold air under the elevated floor. In the sequel, this is referred to as *cooling air*. The cooling air enters the racks from their front side, pick up heat while flowing through these racks, and exits from the rear of the racks. The heated exit air forms hot aisles behind the racks, and is extracted back to the air conditioner intakes, which, in most cases, are positioned above the hot aisles. Each rack consists of several chassis, and each chassis accommodates several computational devices (servers or networking equipment).

Sensors are deployed in the front and back of the chassis to measure the inlet and outlet temperature of the chassis. This information is then collected by a central entity and incorporated in the resource management functions deployed. Specifically, the Dell PowerEdge 1855 blade used in our study is equipped with inlet temperature sensors, in-house sensors, and outlet sensors. These temperature sensors are a part of the chassis and the data is retrieved from them via SNMP. By measuring the temperature difference of the air inlet and the air outlets, we can determine how much heat is being generated.

We have used the Moab [2] cluster management software for this purpose and improved its operation by including thermal awareness. We chose Moab as it is widely used in managing computing clusters across the datacenters and has a detailed graphical interface for controlling the cluster. It further detaches the node-allocation and job scheduling decisions from the actual resource management software such as LSF, TORQUE/PBS, making it ideal to include thermal awareness due to reduced complexity.

The proposed software architecture

- Enables dynamic on-line thermal management during datacenter operations.
- Provides visualization of thermal distribution inside the datacenter.
- Aggregates environmental information which can be used by resource management and thermal-aware scheduling.
- Facilitates on-line upgradation of the scheduler with no requirement for system shutdown.

The rest of the paper is organized as follows. Section II

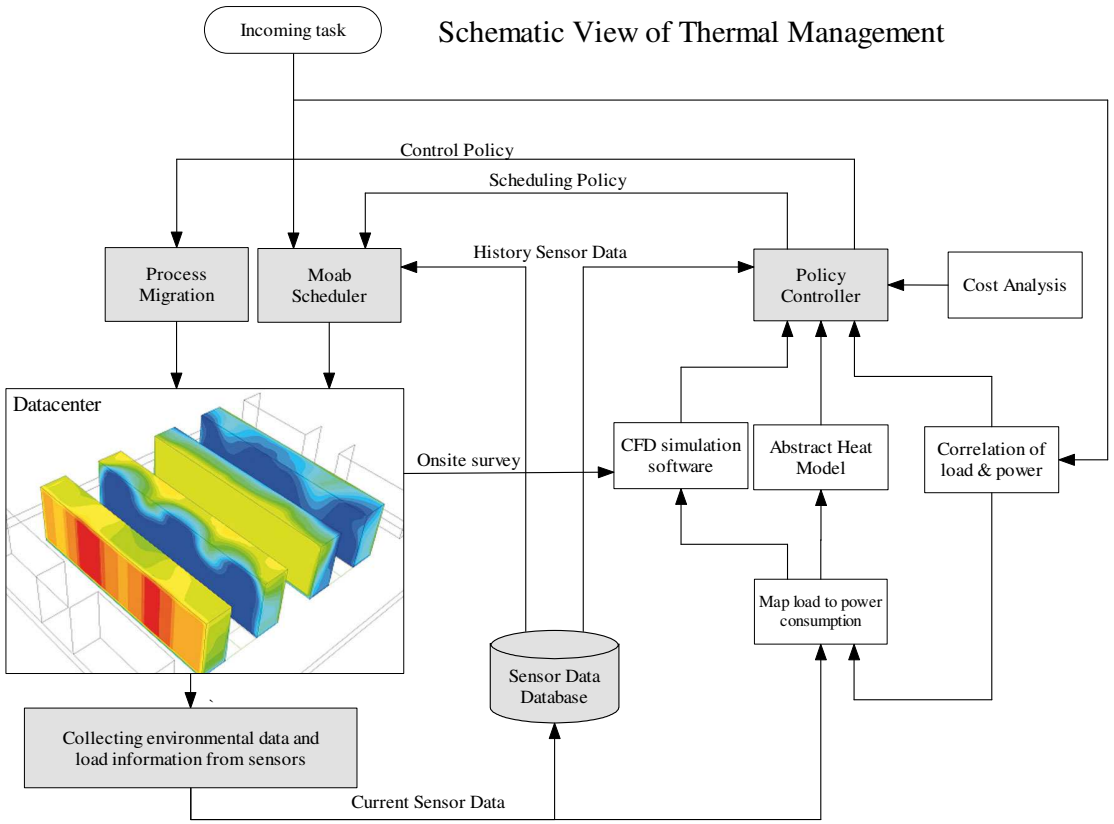

Fig. 1. Schematic view of thermal management of datacenter. The abstract heat model is based on CFD simulation and onsite sensor data measurements. The fast analysis will be provided to policy controller to generate high level thermal management based on cost analysis, thermal distribution and other requirements. The gray parts are modules accomplished by this paper.

provides the essential background information required for developing a thermal aware feedback-control loop for the datacenter along with its required components. Different thermal aware scheduling techniques are discussed in section III along with their comparative study. Section IV presents the details on the ASU datacenter on which we implemented thermal awareness followed by section V which puts everything together to present the software architecture being implemented. Section VI presents the key results followed by the related work and conclusions in sections VII and VIII respectively.

II. Background

A typical cost of operating a datacenter includes energy cost of cooling systems and computing devices, hardware cost for replacing old and dysfunctional computing devices, operation cost for infrastructure investment and labors. The specifications of computers (including networking devices) and cooling systems decide the energy cost for the normal operation of datacenter. The hardware failure model of equipment decides the operation cost such as the cost for replacing hardware, labor cost. Facility cost is one-time cost of purchasing and installing new devices, building infrastructures. In this work, however, we concentrate on the energy cost of cooling systems and computing devices, since other costs primarily depend on business management and policy scope.

The energy cost of computing devices and air conditioners is the source of heat dissipation. The assignment of computation task, the power consumption of different devices, the thermo-mechanic properties of various devices, the cooling capacity and performance of air conditioner, etc., will all directly or indirectly impact the thermal distribution inside a datacenter. The thermal distribution implicitly correlates with the operation cost of the datacenter. Thus, it is very challenging to observe, or understand how the three different properties of datacenter, namely, operation cost, thermal performance and capacity are interrelated with each other.

An integrated thermal-aware scheduling control for datacenters can optimize the total utility cost of running a datacenter. In our previous work [3], we formalized the total energy cost of a typical datacenter and showed that thermal-aware task scheduling at server level can be utilized to achieve better energy efficiency. In this paper we intend to develop the scheduling control architecture and validate it in an actual datacenter. This would involve the streaming of sensor data into a dynamic planning module which would, in realtime, use a thermal modeling framework to determine ”good” scheduling

and cooling policies.

Figure 1 shows a schematic view of thermal management of a datacenter and the requirement for a feedback-control loop for this purpose . In this paper, we intend to develop the highlighted portion (in gray color) of the schematic view[1]. For this purpose, we try to answer the following question: "Given an environmental sensor data gathering framework in a datacenter, how can we integrate it with the resource management functions and develop a software architecture for enabling on-line thermal-aware job scheduling without affecting the normal operations of a datacenter?".

A. Task Scheduling in Datacenters

Apparently, a different utilization rate of a server leads to a different amount of power consumption. From a whole datacenter point-of-view, different scheduling algorithms would result in different task assignments and utilization rates; consequently resulting in different power consumption distributions inside a datacenter. Moreover, the change in the power consumption distribution directly causes the change of temperature distribution. And, finally, different temperature distributions demand different cooling capacity and generate different total energy costs.

This can be better illustrated by Figure 2. Assume that we have to assign a computing task among a group of server nodes. Scheduling Algorithm A and B lead to different temperature distributions. Without cooling system, some of the inlet temperatures are above redline temperature. Cooling system is then introduced to drive the maximal inlet temperature below the redline temperature to reduce hardware fault rate. Obviously, the two algorithms require different efforts; more technically, different cooling capability to achieve this goal. Actually, even with the same total power dissipation for all the server nodes, different scheduling algorithms lead to different temperature distributions, and consequently, result in different total energy cost.

Note that thermal-aware task scheduling of datacenters will not have any impact on an application's performance; the only difference will be which subset of equally capable server nodes are used to perform the computing task. The computing task itself is not aware of such changes. Therefore, we can manipulate task scheduling to achieve the best temperature distribution, and consequently, minimize the total energy cost.

III. Thermal Aware Scheduling

The incorporation of thermal awareness in task scheduling depends on the thorough understanding of the correlation between task assignment and resulting thermal distribution in the datacenters.

A. Task Scheduling and Thermal Distribution Co-relation

A datacenter is composed of a set of computing nodes from 1 to n. Those physically separated nodes work individually or cooperatively to accomplish assigned tasks. These nodes are

[1]We have developed the other parts in our previous work [3] and is therefore excluded from this paper due to space constraints.

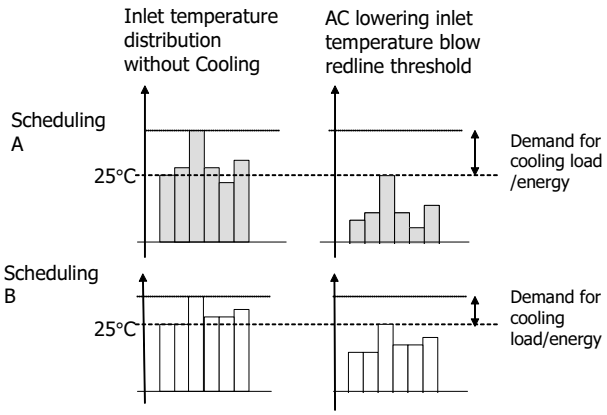

Fig. 2. This illustrates the reaction chain from scheduling algorithm to energy cost: different task assignments result in different power consumption distributions; different power consumption distributions results in different temperature distributions; different temperature distribution results in different total energy cost.

either heterogeneous or homogeneous. There is a scheduler to dispatch incoming tasks C_{Total} (a group of tasks) to individual distributed nodes depending on various scheduling policies, criteria and strategies. Each distributed node i consumes energy at the rate P_i during the execution of task set C_i (a subset of C_{Total}), and the power consumption rate depends on the hardware characteristics of distributed nodes and the task profiles (compute intensive, memory intensive or IO intensive):

$$P_i = G_i(C_i), \tag{1}$$

where G_i is a thermo-mechanic function which depends on the hardware specifications of distributed nodes. G_i can be obtained through experimental measurement or CFD simulation. We characterized the correlation between task and power consumption in our previous work [3].

According to the law of energy conservation and the fact that almost all power drawn by a computing device is dissipated as heat, the relationship between power consumption of a node and its inlet/outlet temperature can be presented as

$$P_i = \rho f_i C_p \left(T^i_{out} - T^i_{in}\right), \tag{2}$$

where C_p is the specific heat of air and ρ is the air density. In other words, the power consumption of node i will change the air temperature from the inlet air temperature T^i_{in} to the outlet temperature T^i_{out}. The generated hot air will also spread to other nodes. The temperature rise can be identified as **self-interference** (heating up air flowing through itself) and **cross interference** (heating up other nodes through recirculation).

B. Basic Thermal Awareness in Task Scheduling

We briefly review three naive thermal-aware scheduling algorithms presented in our previous work [3]. They are "naive" in the sense that they are based on observations and intuitions, and they did not take into account the heat recirculation phenomenon.

Uniform Outlet Profile (UOP): This scheme is similar to the OnePassAnalog algorithm presented in [4]. Based on the inlet temperature of each computing node, this algorithm will assign more tasks to nodes with low inlet temperatures, and fewer tasks to nodes with high inlet temperature. The objective is to achieve a uniform outlet temperature distribution.

Minimal Computing Energy (MCE): MCE minimizes the number of powered-on chassis and processors to concentrate computing energy costs on those active servers and processors and turn-offs all other idle processors or blades. Consequently, the resulting outlet temperatures of all computing nodes will not necessarily be equal. To reduce the thermal risk, the computing nodes with the lowest inlet temperature will be assigned tasks first.

Uniform Task (UT): With this scheme, all nodes are assigned the same amount of tasks.

Once we obtain a workload assignment, we map it into the power consumption, then we can use the sensor based temperature measurement from the servers/chassis to evaluate the thermal distribution for the given power consumption vector.

C. Characterizing Recirculation

Due to the complex nature of airflow inside a datacenter, some of the hot exhaust air from outlets of servers will recirculate into the inlet of other servers. Within a room environment, if the dimensions and locations of all major physical objects are fixed, no moving objects inside the room, the air flow pattern should be relatively stable and predictable. Our hypothesis is that the amount of air and heat recirculated from the outlet of one server to the inlet of another server is relatively stable. If we can characterize such heat flow then we can use it to accurately predict the temperature distribution given another different power consumption distribution.

The air recirculation inside a datacenter can be characterized as cross interference among server nodes. Figure 3 demonstrates our abstract heat flow model and the correlation between distributed nodes. Node N1 has inlet temperature T_{in}^i, which is a mixture of supplied cold air with temperature T_{sup} and recirculated exhaust warm air from other nodes. The outlet warm air of node N1 will partially return to the air conditioner, and partially recirculate into other nodes with constant rate. It also draws in exhaust hot air from the outlet of other nodes due to recirculation. The rate, or the percentage amount, of recirculated heat α_{ij} is defined as cross interference coefficients.

We assume the amount of recirculated heat from node i to node j is $\alpha_{ij}Q_{out}^i$, where Q_{out}^i is the energy of exhaust air from node i. The coefficient α_{ij} is the percentage of heat flow from node i to node j. Assuming cross-interference coefficients are constant, the matrix

$$\mathbf{A} = [\alpha_{ij}]_{n\times n} \tag{3}$$

defines the cross-interference among all the server nodes.

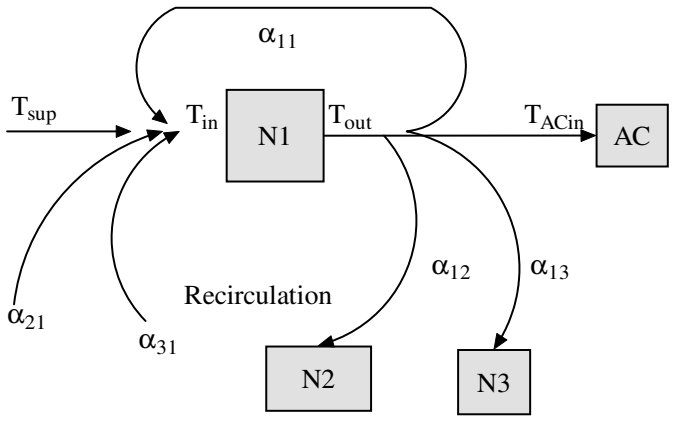

Fig. 3. This figure demonstrates the cross interference among distributed server nodes. Exhaust hot air from node 1 will partially return to AC and partially recirculate into other nodes' inlets. Meanwhile it 'inhales' exhaust hot air from other nodes. Those coefficients show the percentage amount of recirculated heat.

For a given node i, the total amount of heat exhausted in the out air flow, Q_{out}^i is given by:

$$Q_{out}^i = Q_{in}^i + P_i = \rho f_i C_p T_{out}^i, \tag{4}$$

where T_{out}^i is the outlet air temperature and Q_{in}^i is the input heat given by

$$Q_{in}^i = \rho f_i C_p T_{in}^i. \tag{5}$$

where T_{in}^i is the inlet air temperature. We assume the air density does not change (in practice, it changes from $1.205kg/m^3$ at $20°C$ to $1.067kg/m^3$ at $60°C$).

From the practical perspective, at the beginning we need to have a reference power distribution, measure the reference temperature distribution, then try another n different power distribution scenarios, and record all the outlet temperature distribution of these n scenarios. With these data, we can calculate the cross coefficients.

Once we obtained cross interference coefficients $\mathbf{A}$, we can calculate or predict temperature distribution without running CFD. The advantage of such an approach is that we can implement online **fast temperature evaluation** to optimize datacenter operation in real-time. Given a power distribution vector $\overrightarrow{\mathbf{P}}$, the outlet temperature distribution can be calculated using:

$$\overrightarrow{\mathbf{T}}_{out} = \overrightarrow{\mathbf{T}}_{\text{sup}} + (\mathbf{K} - \mathbf{A}'\mathbf{K})^{-1}\overrightarrow{\mathbf{P}}, \tag{6}$$

where K_i is the substitution of $\rho f_i C_p$ (in equations 4 and 5), and inlet temperature can be calculated through

$$\overrightarrow{\mathbf{T}}_{in} = \overrightarrow{\mathbf{T}}_{\text{sup}} + (\mathbf{K} - \mathbf{A}'\mathbf{K})^{-1}\overrightarrow{\mathbf{P}} - \mathbf{K}^{-1}\overrightarrow{\mathbf{P}}. \tag{7}$$

Please refer to our previous work [5] for simulation results of profiling cross interference and fast temperature evaluation. In short, we confirmed our hypothesis that heat recirculation could be characterized as cross interference quantitatively. In addition, we showed in [5] that cross interference based

fast thermal evaluation could be used in online thermal management to predict temperature distribution accurately and effectively.

D. Cross Interference Based Scheduling

To minimize the cooling energy cost of a datacenter, we need to minimize the heat recirculation, or reduce the maximal inlet temperature of all the server nodes. In this regard, we have developed **XInt**, Cross Interference based, recirculation minimized scheduling algorithm for minimizing maximum inlet temperature.

Based on the power consumption characterization of blade servers in [3], we assume that the correlation between power consumption and task assignment is linear: $P_i = a + bC_i$, where a and b are some constants obtained through power consumption characterization. We can rewrite Eq. 7 as:

$$\vec{\mathbf{T}}_{in} = \vec{\mathbf{M}} + \mathbf{N}\vec{\mathbf{C}}. \tag{8}$$

where

$$\vec{\mathbf{M}} = \vec{\mathbf{T}}_{\sup} + \left[(\mathbf{K} - \mathbf{A}' \cdot \mathbf{K})^{-1} - \mathbf{K}^{-1}\right] a \tag{9}$$

$$\mathbf{N} = b\left[(\mathbf{K} - \mathbf{A}' \cdot \mathbf{K})^{-1} - \mathbf{K}^{-1}\right]. \tag{10}$$

Note that $\vec{\mathbf{M}}$ and $\mathbf{N}$ are constants once we characterized recirculation as cross interference matrix $\mathbf{A}$. The optimization problem can be written as

$$\begin{aligned} &\text{minimize}(\max_i \vec{\mathbf{T}}^i_{in}) \\ st: &\sum_{j=1}^{n} C_j - C_{Total} = 0 \\ &: \vec{\mathbf{T}}_{in} = \vec{\mathbf{M}} + \mathbf{N}\vec{\mathbf{C}} \\ &: C_j \geqslant 0, j = 1...n, \end{aligned} \tag{11}$$

which can be transformed to a typical linear program after introducing a new variable v:

$$\begin{aligned} &\min(v) \\ st: v \geqslant \vec{\mathbf{T}}^i_{in} &= M_i + \sum_{j=1}^{n} N_{ij}C_j,\ i = 1...n \\ &: \sum_{j=1}^{n} C_j - C_{Total} = 0 \\ &: C_j \geqslant 0, j = 1...n. \end{aligned} \tag{12}$$

The physical meaning of the optimization problem is how to divide the total task C_{Total} into a task vector $\vec{\mathbf{C}} = \{C_1, C_2, ...C_N\}$ to achieve the minimal maximum inlet temperature. For analog energy model discussed in [3], the problem is just a typical Linear Program; and, it is straightforward to find the best task assignments. For discrete server energy models, such as **DNO** and **DO**, all C_j have to be integers, the problem changes to an Integer Linear Programming (ILP) problem and is relatively difficult to solve if the number of variables is large. However, we still could be able to find a near optimal solution through some heuristic optimization solution.

E. Comparative Analysis

Before presenting the software architecture for thermal aware scheduling, we compare different aspects of the aforementioned techniques. For this purpose, we simulated a small scale datacenter with physical dimensions $9.6m \times 8.4m \times 3.6m$ (see Figure 4) using Flovent [6], a CFD simulation software. The simulated datacenter has two rows of industry standard $42U$ racks arranged in a typical cold aisle and hot aisle layout. The cold air is supplied by one computer room air conditioner, with the flow rate $8m^3/s$. The cold air rises from raised floor plenum through vent tiles, and exhausted hot air returns to the air conditioner through ceiling vent tiles. There are 10 racks and each rack is equipped with 5 chassis (marked from bottom to top as A, B, C, D and E). The maximum computing capacity in the unit of number of processor is 1000. The total power consumption of the whole datacenter would be 232KWatts at full utilization rate. The total power consumption when all the processors are idle would be 86KWatts for DNO mode and 0KWatts for DO mode.

Figure 5 shows the inlet temperature distribution when all the servers are idle. Obviously, the chassis located at the lower part of the rack (A and B) obtain plenty of cold air from the floor vents and have a lower inlet temperature, where chassis located at the upper part (E) of the rack experience a highest inlet temperature due to the insufficient supply of cold air.

F. Comparing Cooling Cost

Since the computing energy cost of different scheduling algorithms are almost the same when the total datacenter utilization rate is above 20%, we only focus on cooling energy cost comparison

Figure 6 and Figure 7 show cooling cost comparison for the four aforementioned algorithms with IgnoreIdle policy. With IgnoreIdle policy, the naive algorithm MCE's total energy consumption has a very close performance to XInt, which could save 30% energy cost compared with UOP and UT when utilization rate is around 50%, and may save from 5% to 20% energy cost at other utilization rates. Figure 6 also shows the theoretical optimal lower bound for the cooling cost. The optimal scenario assumes no existence of heat recirculation, and the supplied cold air and all inlet temperatures are redline temperature $25°C$. The energy efficiency of MCE and XInt are very close to optimal performance at low datacenter utilization rates.

G. Comparing Task Assignment and Temperature

In this work, we only consider Discrete energy model since it is the practical way that servers are operated. First we would like to observe whether different algorithm have any impact on temperature distribution. As we discussed earlier, temperature distribution is an outcome of power consumption distribution.

Figure 8 and 9 show resulting temperature distributions with IgnoreIdle policy. Obviously, XInt has a very similar temperature distribution as MCE, but the peak inlet temperatures for XInt and MCE are $18.5°C$ and $20.5°C$, respectively. Therefore, even the total power consumption are the same as

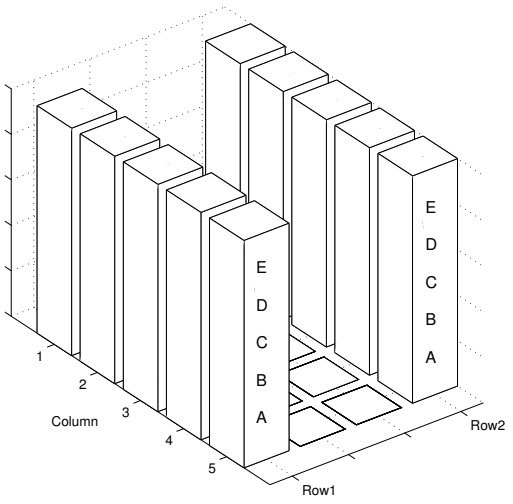

Fig. 4. Two Row datacenter used in our simulation study, each rack has 5 blade server chassis, marked from bottom to top as A, B, C , D and E.

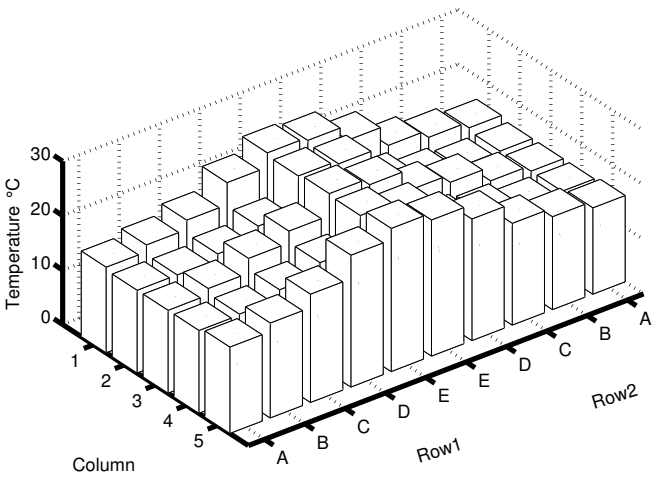

Fig. 5. Inlet temperature distribution: chassis locate at the lower part of the rack obtain plenty of cold air from floor vent and have a low inlet temperatures

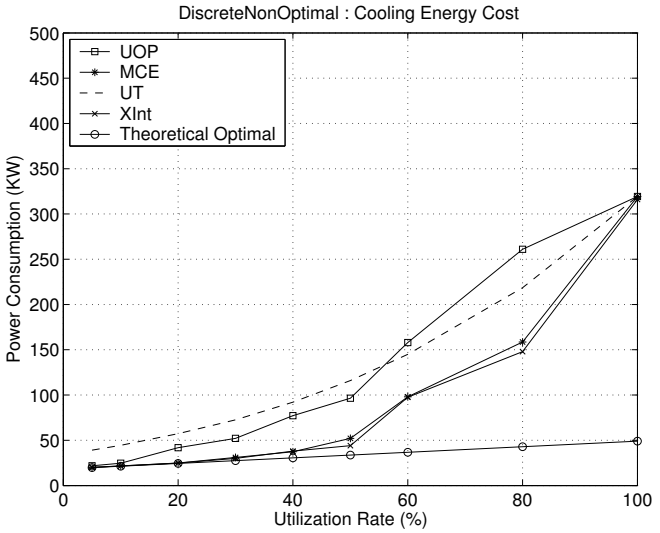

Fig. 6. XInt and MCE possess minimal cooling energy cost, the energy efficiency of MCE and XInt are very close to optimal performance at low datacenter utilization rate.

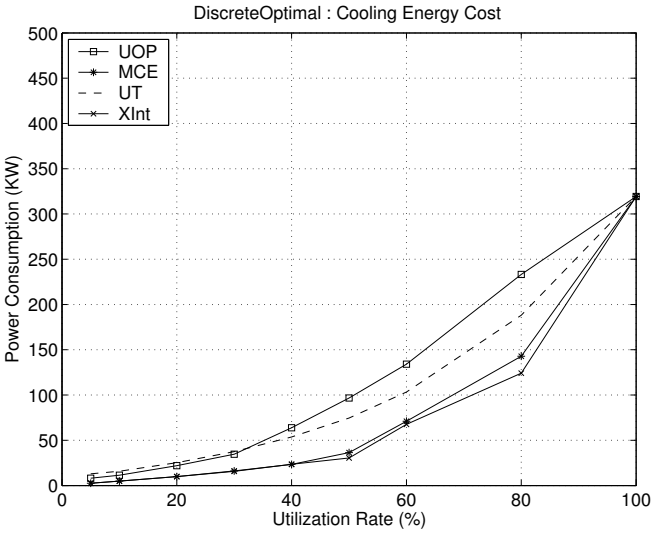

Fig. 7. XInt and MCE possess the minimal cooling energy cost with IgnoreIdle policy.

159.15KWatts for XInt and MCE, different task and power distributions could lead to different peak inlet temperature.

IV. System Model and Cluster Set-up

In the previous sections, we have discussed various techniques for thermal aware scheduling along with their relative comparison in terms of their thermal performance and energy-efficiency. This section provides the system model for implementing the thermal aware node allocation and job-scheduling in the datacenters. We used the central cluster set-up of the ASU datacenter for the implementation. The basic idea is to examine allocation of new jobs based on the temperature of the nodes in the cluster, change in thermal environment after the submission of these jobs, and migration of jobs when unacceptable thermal environment is reached.

Figure 10 presents the ASU datacenter set-up. There are three main components for implementing and visualizing the thermal aware cluster management in the datacenters:

1) Remote Client: This is a laptop or any remote machine which presents the results (in graphical form) of the thermal aware scheduling in the cluster.

2) Central Server: The central server controls the nodes in the cluster grid using Moab scheduler provided by Cluster Resources, Inc. [2].

3) Cluster grid: The cluster we used for our implementation is the centerpiece of the ASU Campus Grid, a 200 node,

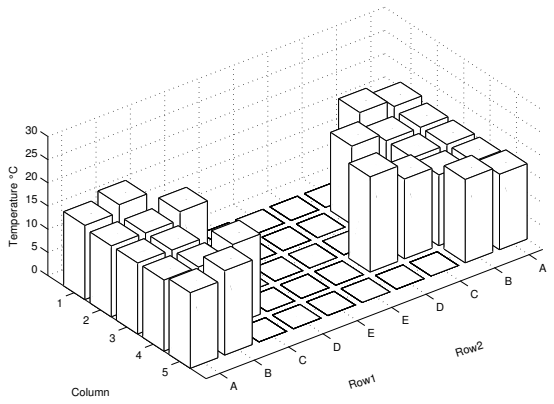

Fig. 8. Inlet temperature distribution of MCE, peak temperature is $18.5°C$.

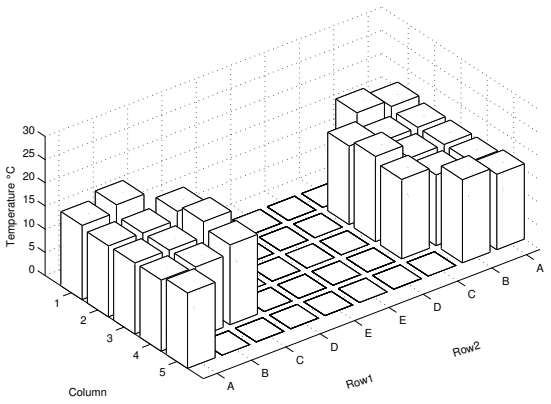

Fig. 9. Inlet temperature distribution of XInt, peak temperature is $20.5°C$.

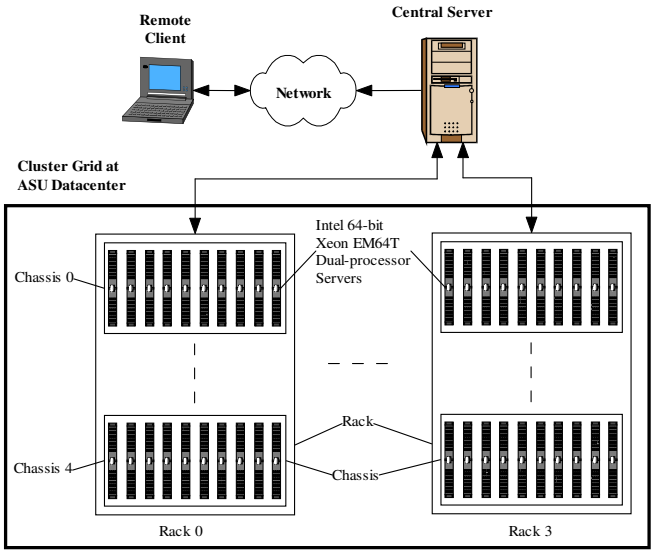

Fig. 10. Cluster Set-up at ASU datacenter.

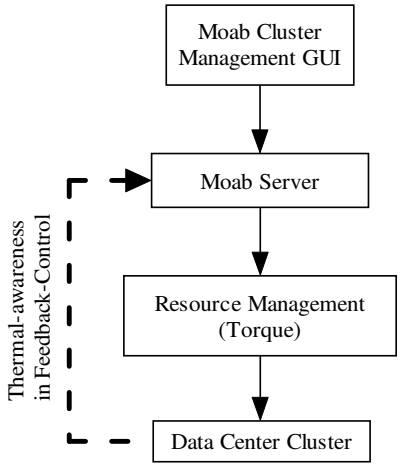

Fig. 11. Software structure of Moab. Developing the feedback-control loop is the goal of this paper.

400 processor system comprised of Intel 64-bit dual-processor Xeon EM64T CPUs manufactured by Dell, and interconnected via an Infiniband high speed interconnect. There are a total of 4 racks with 5 chassis in each rack. Each chassis has 10 nodes (and each node has 2 processors). Although we can check the status of the entire cluster, we have control over chassis 0 and 4 of the cluster (total of 20 nodes i.e.40 processors). This is because it is a fully operational cluster and there are always a large number of jobs running for different research projects spanning over various departments in ASU. Further, this restriction does not hinder our testing as chassis 0, at the bottom of the rack with low inlet temperature, and chassis 4, at the top of the rack with higher inlet temperature, provide discernible temperature differential. This is because the cold air flows from the bottom of the perforated floor making the inlet of the bottom chassis lower compared to that of the top chassis where it reaches after mixing with the ambient air in the room. Thus the effects of the thermal awareness in the scheduling can be easily verified with our set-up.

There are two basic aspects in implementing the thermal aware scheduling using the aforementioned components. Firstly to get the feedback from the on-board sensors about the thermal environment, we extract the temperature data from ASU data center log file. This file is generated by a SNMP query script which queries the on board sensors and logs their readings. Secondly, we integrated thermal awareness in the Moab scheduler for controlling the thermal environment. In the following sections we provide more details on these issues.

A. Cluster Management using Moab

We used the Moab scheduler to configure thermal awareness into it. There are two components of the Moab scheduler: 1) Moab server, and 2) Moab Cluster Manager (MCM). Moab Server runs in central server running Linux Fedora Core Operating System, and MCM acts as a client that can be run on any laptop/desktop. The MCM connects to the server and shows the status of the jobs and the nodes of the cluster controlled by the Moab server. For resource management

Moab server makes use of an underlying resource management software such as LSF, TORQUE/PBS. Figure 11 depicts the schematic view of the Moab software structure along with the feedback-control connection required for thermal-awareness in the node allocation and job scheduling mechanisms.

The cluster set-up we used uses the TORQUE resource manager. It is a freely available software which has two main components: 1) a central controller that maintains all the resource information along with submitting jobs to the servers, and 2) distributed agents executing at each individual servers that report to the central controller. Moab makes use of the resource information maintained by the central controller of TORQUE and schedules jobs accordingly. The resource and job information can be checked from command line on the machine where Moab server is installed or from remote laptop/desktop using the MCM software (as mentioned previously). Moab.cfg file is the configuration file (in the Moab server) that specifies the underlying resource manager type and the path where it is located. Moab also allows the use of a native resource manager which can report any generic metrics (not managed by the underlying resource manager) such as temperature information. A generic metrics file (in any format such as .html, .txt, .perl), containing the required information, is required to be maintained and updated in this regard. The path of this file is required to be specified in the moab.cfg file in order to enable it to be the native resource manager. This provision in the Moab cluster manager makes it ideal for the implementation of any thermal awareness into it.

The scheduling technique can also be configured using the moab.cfg file. This file (moab.cfg) is stored in the directory /opt/Moab on the host where Moab server is installed. In the next section we discuss the thermal aware scheduling techniques to be implemented using the Moab cluster manager discussed in this section.

V. Software Architecture

Figure 12 presents the overall software architecture putting together all the components described in section IV. The visualization tool extracts raw sensor data from the sensor history as updated by the SNMP query script in the central server and shows it in graphical form. Further, it updates the gemetric file for the Moab server in appropriate format so that Moab can be configured for thermal awareness. This tool can be executed in any local desktop[2] provided its location is properly fed into the Moab configuration.

A. Visualization and Aggregation of the sensor data

Figure 13 expands the visualization tool (of Figure 12) for monitoring the thermal distribution in the datacenter. It primarily consists of a component for Data Retrieval, Parsing, and File Generation that executes in an iteration. This component connects to the Central Server securely through SSH and extracts the most recent temperature readings for each sensor in each chassis from the Moab data log. The

[2]In our implementation we used the server 129.219.33.232. This can be replaced by any IP address where the visualization tool is located.

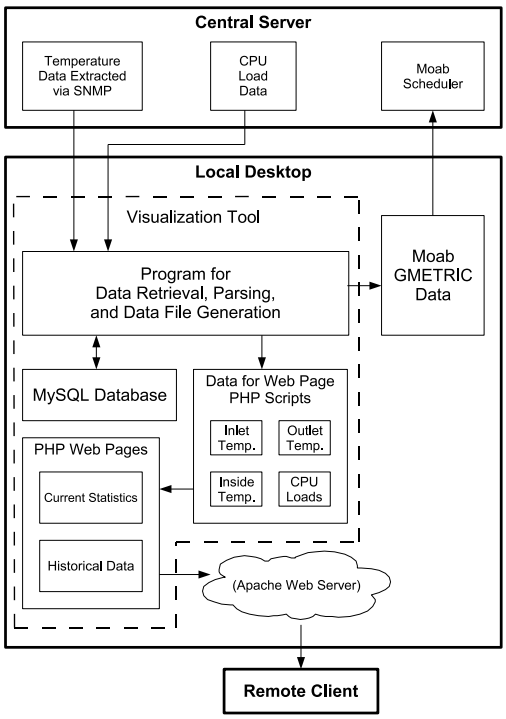

Fig. 13. Visualization tool for monitoring thermal distribution in the datacenter

temperature data is then stored in the MySQL database along with the time when the sensor measured the temperature. The program also executes a shell script on the Central Server which gathers the CPU performance data, and this data along with the current time is then also stored in the database. The database is a convenient way to store and retrieve the most recently collected sensor data when the last data collection attempt fails.

The program is also responsible for generating data for Moab and for the PHP Web Pages. The most recent temperature and CPU load data is retrieved from the MySQL database and a Moab generic metric file is made which can then be used by the Moab scheduler on the Central Server. Four files representing inlet temperatures, outlet temperatures, inside temperatures, and CPU loads are created for the PHP web pages that parse and display this information graphically when requested by a Remote Client. The PHP web pages are served by an Apache Web Server and display graphs that represent the current conditions of the servers in the datacenter, as well as historical trends for inlet temperature, outlet temperature, CPU loads, and estimated power consumption.

B. Implementation of the scheduling Algorithms in Moab

In this section we provide the basic changes to be performed in the Moab cluster manager. For simplicity we provide sample changes for the MCE algorithm (which assigns jobs to servers with lowest inlet temperature) discussed in section III. Overall, MCE performs close to the best as XInt does (Sections III-G and III-F). The implementation of the other algorithms includ-

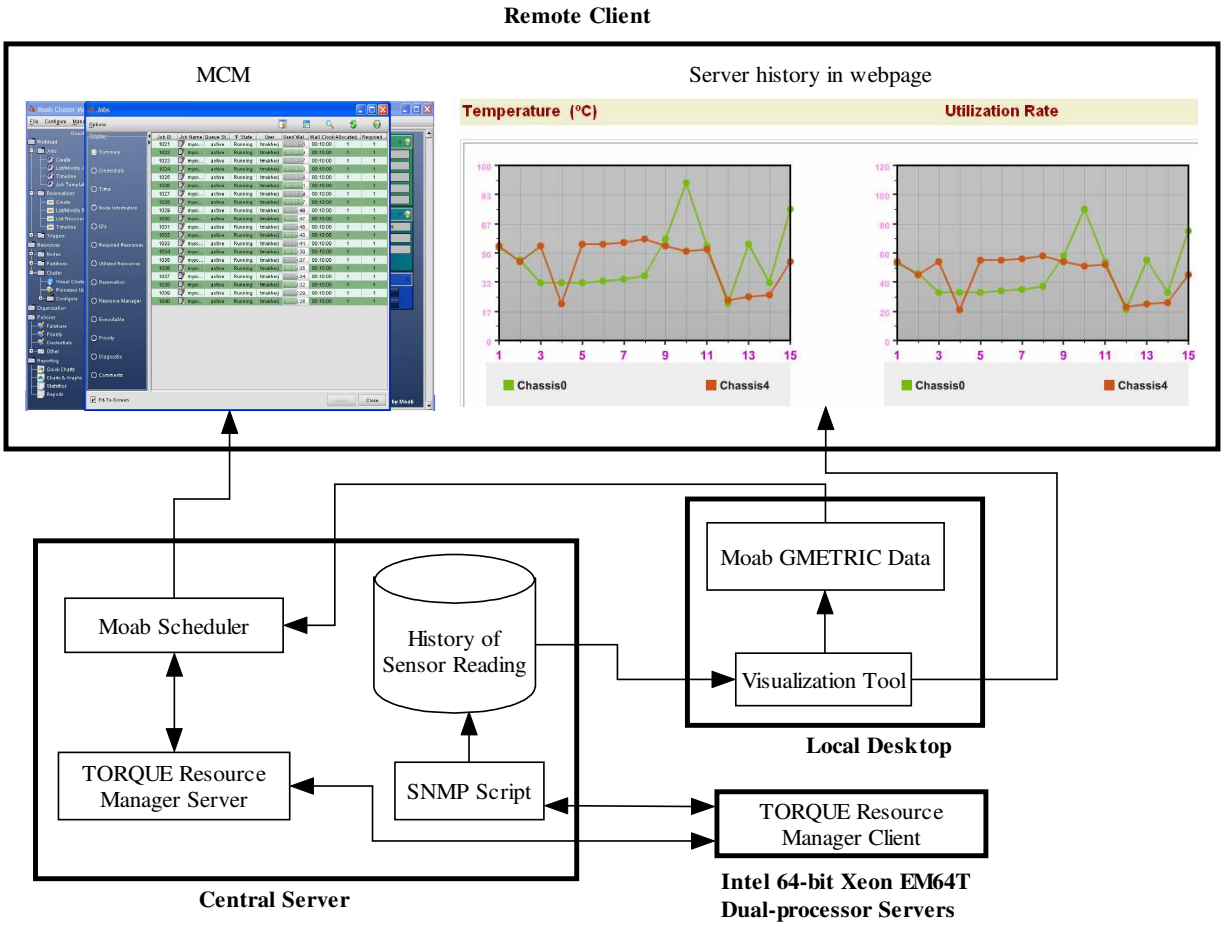

Fig. 12. Software Architecture for implementing thermal-aware scheduling.

ing XInt follows the same format with different configuration variables.

There are three main steps, presented below, in incorporating thermal awareness to the Moab cluster manager.

1) Integrating temperature data in the Moab: Temperature information of the nodes (in chassis 0 and 4) are stored as the generic metric in the following file

After this change, when the Moab is restarted, the command $checknode < nodeid >$ shows the three generic metrics for the nodes in chassis 0 and 4. This command is a native script that abstracts out the underlying resource management command. In our set up this script calls the TORQUE for checking the node status. The temperature information can now be used to define a function to be used as the priority of the individual nodes in the cluster.

2) Setting the Priority of the nodes: First we have to add the following configuration command in the *moab.cfg* file.

NODEALLOCATIONPOLICY PRIORITY

This configures the node allocation policy for the cluster to be driven by the priority of the nodes. The priority can be now defined as follows:

NODECFG[$< nodeid >$] PRIOF =
PRIORITY - 10 * GMETRIC[inlet]

where $PRIOF$ specifies the function to be used to determine the priority of the individual nodes, and $PRIORITY$ is any constant value big enough to ensure that $PRIOF$ always has a positive value.

In the above example the priority of the node is only dependent on the inlet temperature (maintained in the Moab generic metric file). Therefore, when the inlet temperature increases the priority decreases. After these changes to the *moab.cfg*, when we restart Moab, any newly submitted job would be allocated to the node based on its priority i.e. jobs would get submitted to the nodes with low inlet temperature.

3) Setting object triggers if temperature crosses a threshold: This is required to make sure that already running jobs are preempted when the temperature of the node (where it is currently allocated) goes beyond a specific value. The following configuration command ensures that jobs would get check-pointed during preemption so that it resumes execution from the same place where it got preempted.

PREEMPTPOLICY CHECKPOINT

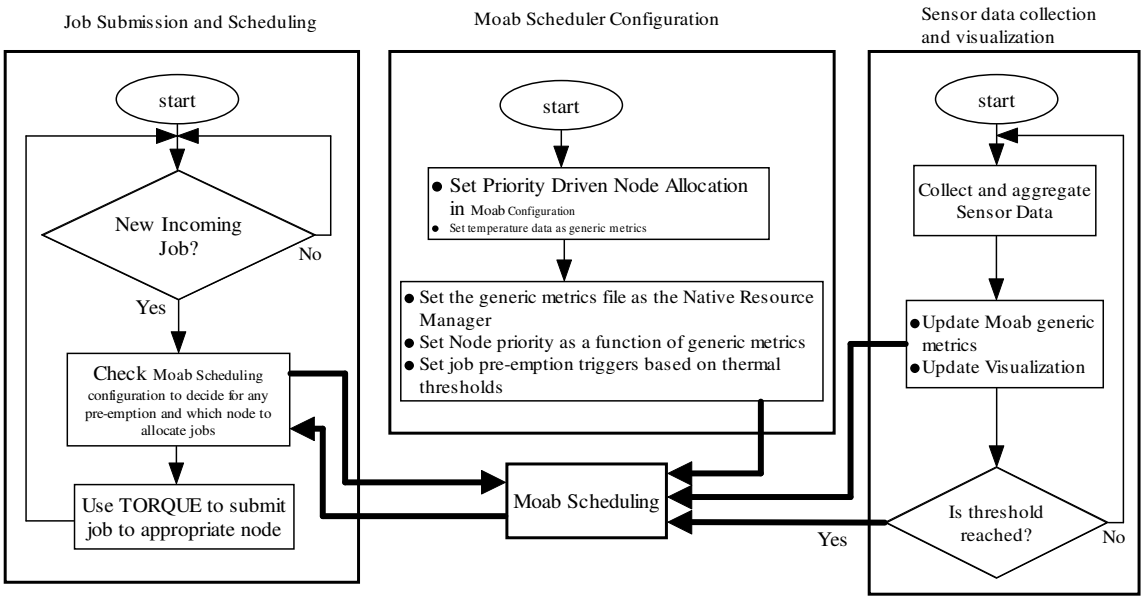

Fig. 14. Three principal modules for the thermal aware scheduler. The thick arrows are used to depict how different modules affect the Moab scheduling and vice versa.

The configuration variable *PREEMPTPOLICY* can also be set as REQUEUE to restart the job from the beginning after preemption. To set job-preemption as the desired action when the inlet temperature goes beyond 25 degrees centigrade we use the following configuration command in our *moab.cfg* file,

NODECFG[$< nodeid >$] TRIGGER =
atype = jobprempt, etype = threshold,
threshold = GMETRIC[inlet] > 25

where *atype* defines the type of action when trigger occurs, and *etype* specifies the type of event that causes the trigger. In this case, we specify *etype* as *threshold* to set the trigger when the inlet temperature goes beyond the threshold of 25 degrees centigrade. This ensures that jobs in node $< nodeid >$ are pre-empted and checkpointed for further execution in other nodes.

Any changes made in the moab.cfg file would take their effects after the execution of the script recycle from the command line. It has to be noted that the priority and triggers are only set for the nodes in chassis 0 and 4 of the saguaro cluster.

To summarize the software architecture we identify three basic modules for the thermal aware scheduler as depicted in Figure 14. The *Job Submission and Scheduling Module* is responsible for assigning newly submitted jobs to the appropriate nodes depending on the Moab scheduling configuration. It also pre-empts any jobs in nodes when certain thermal threshold is reached which activates triggers configured in the Moab scheduling decisions. The *Moab Scheduler Configuration* module is an one-time process that configures the Moab scheduling as described in section V-B. Finally, *Sensor data collection and visualization* module is responsible for online collection and aggregation of sensor data as described in section V-A.

VI. Results

The software architecture presented in the previous section was verified with actual prototype implementation and conforms to the results of our predictions. Our results show that the MCE algorithm (along with the XInt algorithm), in most cases, results in a minimal total energy costs - a conclusion that differs from the findings of previous research [4]. UOP performs better than UT at low datacenter utilization rates, whereas UT outperforms UOP at high utilization rates.

We also observed that the computing energy cost increases linearly with the increase of utilization rate, where cooling cost increases exponentially due to the nonlinearity of Coefficient of Performance of the cooling systems[3]. Normally, when the utilization is less than 60%, the dominant part of total energy costs is contributed by computing energy. Once the utilization rate exceeds 60%, the cooling cost replaces the computing cost as the most significant part. *The results confirmed that thermal-aware scheduling based on thermal performance evaluation improves energy efficiency of datacenter operation, and consequently increases the utilization rate and computation capability of datacenters.*

Even though MCE (and XInt) is the most energy efficient algorithm, it has some practical limitations. This is because under the MCE algorithm, the chassis at the lower part of the rack will be used excessively and will experience higher hardware failure rate due to unremitting long time operation. Our future work will consider hardware reliability models and hardware cost models to address this issue, we will balance the trade-off between energy cost, hardware failure cost, and resulting labor cost of replacing or repairing hardware.

VII. Related Work

Researchers at HP Labs and Duke University have published work [7] [8] on smart cooling techniques for datacenters. They have developed online measurement and control techniques to improve energy-efficiency of datacenters. They defined Supply Heat Index (SHI) and Return Heat Index (RHI) to characterize the energy efficiency of datacenter cooling systems.

Our work is similar to Splice, a datacenter measurement and monitoring infrastructure, proposed in [9]. But we integrated our proposed framework with available cluster management platform to makes as an extra plug-in module, thus make it more flexible and portable, easy to be deployed with datacenter environment.

VIII. Conclusions and Future Work

In this paper, we have developed a unique software architecture to develop thermal aware job scheduling for the datacenters. The proposed architecture enables dynamic on-line thermal management during datacenter operations, provides visualization for thermal distribution inside the datacenter, aggregates sensor data for feedback to the resource management functions, and facilitates on-line upgradation of the scheduler with no requirement for system shutdown. It is further implemented in a fully operational ASU datacenter and the results validate the theoretical foundations of the on-line thermal aware scheduling techniques.

Acknowledgment

This work is supported in part by a grant from Intel Corporation, and NSF grant #CNS-0649868. The authors are thankful to Dan Stanzione and Karl Lindekugel of Fulton High Performance Computing Initiative for providing ASU datacenter access and assisting in ASU cluster set-up for thermal aware scheduling. Further, the authors thank Cluster Resources Inc. for assistance in setting up Moab cluster management software.

References

[1] "Fulton high performance computing initiative." [Online]. Available: http://hpc.fulton.asu.edu/

[2] "Moab grid suite of cluster resources inc." [Online]. Available: http://www.clusterresources.com/

[3] Q. Tang, S. K. S. Gupta, D. Stanzione, and P. Cayton, "Thermal-aware task scheduling to minimize energy usage of blade server based datacenters," in *IEEE International Symposium on Dependable, Autonomic and Secure Computing (DASC06)*, October 2006.

[4] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, "Making scheduling "cool": Temperature-aware resource assignment in data centers," in *2005 Usenix Annual Technical Conference*, April 2005.

[5] Q. Tang, T. Mukherjee, S. K. S. Gupta, and P. Cayton, "Sensor-based fast thermal evaluation model for energy efcient high-performance datacenters," in *International Conf. on Intelligent Sensing Info. Proc. (ICISIP2006)*, December 2006.

[6] "Flovent CFD simulation software." [Online]. Available: http://www.flomerics.com/

[7] C. D. Patel, R. Sharma, C. E. Bash, and A. Beitelmal, "Thermal considerations in cooling large scale high compute density data centers," in *Proceedings of the Eight Inter-Society Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, San Diego, CA, June 2002, p. 767C776.

[8] M. H. Beitelmal and C. D. Patel, "Thermo-fluids provisioning of a high performance high density data center," Hewlett Packard Laboratories, Tech. Rep. HPL-2004-146, September 2004. [Online]. Available: http://www.hpl.hp.com/techreports/2004/HPL-2004-146.html

[9] J. Moore, J. Chase, K. Farkas, and P. Ranganathan, "Data center workload monitoring, analysis, and emulation," in *Eighth Workshop on Computer Architecture Evaluation using Commercial Workloads*, February 2005.