Virtual Machine Level Temperature Profiling and Prediction in Cloud Datacenters

Zhaohui Wu*, Xiang Li*, Peter Garraghan§, Xiaohong Jiang*, Kejiang Ye†, Albert Y. Zomaya††
* College of Computer Science, Zhejiang University, China
§ School of Computing, University of Leeds, UK
† Electrical and Computer Engineering, Carnegie Mellon University, USA
†† School of Information Technologies, University of Sydney, Australia
{wzh, lixiang2011, jiangxh}@zju.edu.cn; scspg@leeds.ac.uk; kejiangy@andrew.cmu.edu; albert.zomaya@sydney.edu.au

***Abstract*—Temperature prediction can enhance datacenter thermal management towards minimizing cooling power draw. Traditional approaches achieve this through analyzing task-temperature profiles or resistor-capacitor circuit models to predict CPU temperature. However, they are unable to capture task resource heterogeneity within multi-tenant environments and make predictions under dynamic scenarios such as virtual machine migration, which is one of the main characteristics of Cloud computing. This paper proposes virtual machine level temperature prediction in Cloud datacenters. Experiments show that the mean squared error of stable CPU temperature prediction is within 1.10, and dynamic CPU temperature prediction can achieve 1.60 in most scenarios.**

I. INTRODUCTION

The study within 2011 [1] shows that the energy used by datacenters increased by 36% in US and 56% worldwide from 2005 to 2010. It is expected to increase continuously during the coming years and the global annual datacenter construction size for 2020 is predicted to reach $78 billion [2]. Datacenter energy usage can be categorized as stemming from computing and cooling infrastructure, with the latter forming approximately half of the total consumption [3]. Thermal management is an approach to reduce energy usage within cooling infrastructure. This is achieved through minimizing temperature distribution disparity throughout the system to reduce the probability of hotspot occurrence within the system. Temperature prediction is a fundamental technique to conduct thermal management proactively and provides substantial value to decision making [4].

With the rapid development of Cloud computing, virtualization technology plays a key role for workload scheduling and resource allocation, and imposes a new set of challenges towards temperature prediction. Traditional approaches such as task-temperature [4] and Resistor-Capacitor (RC) thermal model [5] assume homogeneous workload characteristics within the system, and that only a single task is deployed at any point within a server or CPU core. For more complicated scenarios such as Virtual Machine (VM) migration, these approaches are unable to model CPU temperature. Virtualization allows VM creation with heterogeneous resource characteristics deployed within multi-tenant scenarios, imposing inevitable constraints to traditional temperature modeling approaches.

This paper proposes a method for VM level temperature prediction for Cloud datacenters. Our method comprises parameters cross-cutting datacenter infrastructure including server capacity, virtual machine characteristics and environmental conditions which are integrated into a supervised machine learning algorithm for modeling server temperature.

II. TEMPERATURE MODELING AND PREDICTION

Temperature prediction is typically used to discover the correlation between future temperature and current status. Numerous experiments were conducted under different scenarios in order to analyze the features of temperature profiles of VM executing within a server. When altering running conditions (e.g. number of VMs, environment temperature), we observed that temperate will first experience variation and subsequently stability after reaching a certain temperature value. Therefore, our method begins with *stable CPU temperature* (ψ_{stable}) prediction, calculated as the average CPU temperature after a certain period of time (t_{break}, set to 600s deduced from experiments).

$$\psi_{stable} = \left(\sum_{i=t_{break}}^{t_{exp}} \psi(i)\right) \Big/ \left(t_{exp} - t_{break}\right) \quad (1)$$

where t_{exp} is defined as the experiment duration measured in seconds. Formally, we use ψ and φ to denote predicted and actual measured temperature, respectively. To make *stable CPU temperature* prediction, we collect data from the Virtual Machine Manager (VMM) and temperature sensors. In addition, we collect data pertaining to the environment temperature reflecting the overall cooling capacity within a datacenter as environment temperature imposes a non-negligible impact on CPU temperature. A Support Vector Machine (SVM) model was trained from the collected data and deployed in real environment. Then the model received data collected online and output prediction values. The data collected used for training ($data_{train}$) and testing ($data_{test}$) follows the format shown below.

$$\begin{cases} data_{train} \text{ or } data_{test} = \{input, output\} \\ input = \{\theta_{cpu}, \theta_{memory}, \theta_{fan}, \xi_{VM}, \delta_{env}\}, output = \psi_{stable} \end{cases} \quad (2)$$

where the input comprises server CPU capacity (θ_{cpu}), memory size (θ_{memory}), fan status (θ_{fan}), VM status (ξ_{VM}, including VM configurations and deployed tasks), and environment temperature (δ_{env}). The output produces *stable CPU temperature* (ψ_{stable}), defined as a stable temperature value for a server shown in Equation (1). We select LIBSVM (Version 3.17) as the modeling tool using the Radial Basis Function (RBF) kernel. LIBSVM is an integrated software

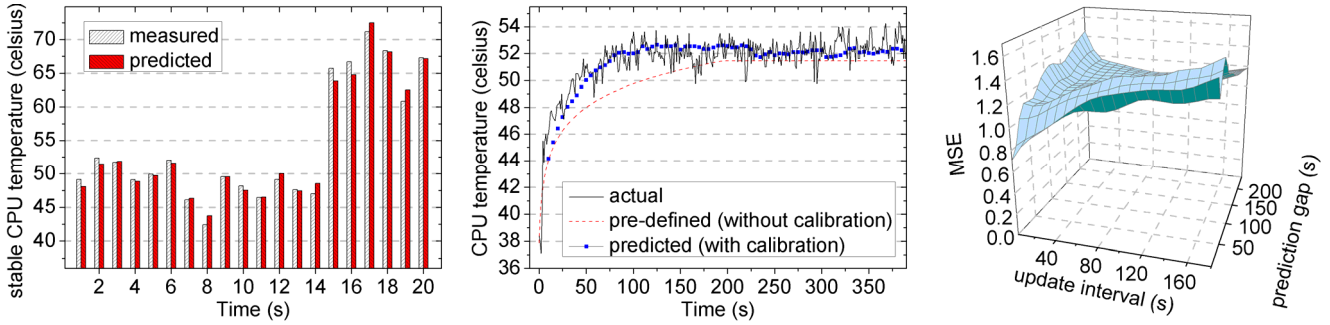

Figure 1(a): *stable CPU temperatur*e prediction results; (b): A case study of *dynamic CPU temperature* modeling against empirical data; (c): prediction accuracy with 4 server fans.

for support vector classification, regression and distribution estimation [6]. For each experiment, one record is produced as shown in Equation (2). Records are divided into training and test sets. Parameters for model training are selected using easygrid, a tool for grid parameter search, with 10-fold validation.

Based on the prediction of ψ_{stable}, we further consider *dynamic CPU temperature* prediction. We construct a pre-defined temperature curve $\psi^*(t)$ shown below.

$$\psi^*(t) = \begin{cases} \varphi(0) + \dfrac{(\varphi_{stable} - \varphi(0))\, ln(t+1)}{ln(t_{break}+1)}, & t \le \delta_{break} \\ \varphi_{stable}, & t > \delta_{break} \end{cases} \quad (3)$$

where $\varphi(0)$ is the temperature prior to experiment starts. The pre-defined model shown in Equation (3) provides coarse-grain prediction. It is necessary to calibrate the prediction based on $\psi^*(t)$ at run time. At the very beginning (t=0), the calibration γ is set to 0 (γ=0). For example, if the objective was to predict CPU temperature 60s after the time making prediction. The current time is 0, and the calibrated prediction based on $\psi^*(t)$ is:

$$\psi(60) = \psi^*(60) + \gamma = \psi^*(60) \quad (4)$$

However when $t \neq 0$ (e.g. 15s), we determine a difference (*dif*) between the actual measured value $\varphi(15)$ and the predicted value $\psi^*(15)$. The latter is the sum of pre-defined temperature and the latest calibration. Until now (t=15s), the calibration has not been updated and remains 0.

$$dif = \varphi(15) - (\psi^*(15) + \gamma) = \varphi(15) - \psi^*(15) \quad (5)$$

The difference between the prediction and measurement indicates the precision of our last prediction. The calibration is updated based on *dif* and a learning rate λ (defined as 0.8).

$$\gamma = \gamma + \lambda * dif = \lambda * (\varphi(15) - \psi^*(15)) \quad (6)$$

γ on left side is the newly updated calibration, while the other γ is the former calibration (equals 0 here). At t=15, if we were to predict temperature 60s later (i.e. 75s), it is

$$\psi(75) = \psi^*(75) + \gamma \quad (7)$$

where γ is derived from Equation (6). If *prediction gap* represents the time interval we want to predict (Δ_{gap}, 60s in this case), and *update interval* represents the time interval between two calibration updates (Δ_{update}, 15s in this case), the prediction can be formalized as follows.

$$\psi(t + \Delta_{gap}) = \psi^*(t + \Delta_{gap}) + \gamma \quad (8)$$

in which γ is updated on a regular interval of Δ_{update}. This represents a complete solution to predict CPU temperature dynamically in a virtualized environment. This is an important consideration because that Cloud computing characteristics result in input features such as server and VM configuration changing at run time.

III. Evaluation

The proposed temperature prediction model for stable and dynamic CPU temperature was evaluated through experiments to ascertain model accuracy compared to real system operation. Fig. 1(a) shows a comparison between empirical temperature readings and *stable CPU temperature* prediction for 20 randomized experiment cases with 2-12 VMs. It is observed that the model is capable of predicting *stable CPU temperature* with an average Mean Squared Error (MSE) value within 1.10. Fig. 1(b) shows a particular experiment case study of predicting temperature dynamically with/without calibration compared to empirical data. Results indicate that *dynamic CPU temperature* modeling with calibration at run time produces a lower MSE. Fig. 1(c) shows the MSE when varying *prediction gap* and *update interval* with 4 server fans. It is observable that the MSE varies from 0.70 to 1.50, indicating high prediction accuracy with different *prediction gaps* and *update intervals*.

References

[1] Jonathan K, Growth in data center electricity use 2005 to 2010. Oakland, CA: 2011. Analytics Press.

[2] Christian L. Projecting annual new datacenter construction market size. Technique report, Microsoft Corp., 2010.

[3] Lee Y C, et al. Energy efficient utilization of resources in cloud computing system, Journal of Supercomputing, 2012, pp. 268-280.

[4] Wang L, Khan S U, Dayal J. Thermal aware workload placement with task-temperature profiles in a data center, The Journal of Supercomputing, 2012, vol. 61(3), pp. 780-803.

[5] Zhang S, et al. Approximation algorithm for the temperature-aware scheduling problem, Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, 2007, pp. 281-288.

[6] LIBSVM - A Library for Support Vector Machines, http://www.csie.ntu.edu.tw/~cjlin/libsvm.