Automating CPU Dynamic Thermal Control for High Performance Computing

Presenter: Ghazanfar Ali, Texas Tech University

Authors: Ghazanfar Ali\textsuperscript{\dagger}, Lowell Wofford\textsuperscript{\ddagger}, Christopher Turner\textsuperscript{\dagger}, and Yong Chen\textsuperscript{\dagger}

\textsuperscript{\dagger}Texas Tech University, Lubbock, TX, USA
\textsuperscript{\ddagger}Los Alamos National Laboratory, Los Alamos, NM, USA
Outline

- Introduction
- Background and Motivation
- Design
- Evaluation and Experiences
- Conclusions and Future Work
Introduction

- Numerous factors, including workload compute intensity, cooling infrastructure failure, and the use of economized cooling can substantially increase the CPU temperature.
- Studies have shown that slight variances in the operational temperature can significantly impact the lifetime, durability, and performance of a CPU.
- Therefore, it is critical to monitor and control the operating temperature of the CPU.
- Designed an automated and continuous CPU thermal monitoring and control methodology to maintain a healthy CPU thermal state using Kraken and Redfish.
- Evaluated our methodology using a cluster of 150 Raspberry Pi3 nodes.
- Based on our experiments, 70°C (158°F) was the best temperature to trigger thermal control.
Compute-Intensive workloads can cause frequent increase in CPU temperature.

CPU temperature analysis of an overheated CPU over a month at TTU’s HPCC.

This CPU ultimately got malfunctioned, which could have been avoided by maintaining a healthy CPU temperature.
A cooling infrastructure failure can cause a sudden increase in CPU temperature.

At the time of failure, CPUs running compute-intensive workloads can get overheated quickly.

A reliable thermal control mechanism can increase ride-through time for fixing the failure or graceful shutdown.
Background & Motivation – Usage of Economizers

- Due to low energy cost, many HPC data centers are employing economizers for cooling.
- Economizers can cause fluctuation in ambient temperature between 18°C (64.4°F) and 27°C (80.6°F) from the desired ambient operating temperature (ASHRAE 2016).
- These fluctuations in temperature can indirectly increase the temperature of a CPU.
- A dynamic CPU thermal controller is desirable in this scenario.
CPU Temperature, Power, and MTTF

\[ Power \cdot \theta_{ja} = T_j - T_a, \]  

(1)

Arrhenius Eq. (impact of temperature on the failure rate of semiconductor):

\[ \lambda = \lambda_{ref} \cdot \exp \left(-\frac{E_{dev}}{k \cdot T}\right), \]  

(2)

\[ MTTF = \frac{1}{\lambda}. \]  

(3)
DVFS: CPU Power Management

User Space Programs
- HostFrequencyScaling
- Policy
- Attributes

Kernel Space Frequency Scaling Governors
- performance
- powersave
- schedutil
- ondemand
- conservative
- userspace
- CPUFreq driver management interface

Platform Specific Drivers
- cpufreq-cpu0
- intel_pstate
- acpi-cpufreq
- pcc-cpufreq
- Exynos-cpufreq
- Omapi-cpufreq
Kraken:

- Kraken is a distributed state discovery & control engine that can maintain states across a large set of computers.
- Works in Parent and Child manner
- It is designed to provide full-lifecycle maintenance of HPC compute clusters.

Key Terminology:

- **Configuration State**: This is the configuration data which describes the desired state of the node.
- **Discovered State**: This is the actual state of the node as discovered by the Kraken.
- **Mutation**: is referred to action or set of actions which are performed to achieve configuration state if discovered state is not same as configuration state.
CPU Thermal State Discovery Mechanism:

- In-band CPU Thermal State Discovery
- Out-of-band CPU Thermal State Discovery

In-band CPU Thermal State Discovery:
- Accesses CPU temperature via Operating System
- CPU temperature is reported to a Parent Kraken node via a Child Kraken node
Design: In-band CPU Thermal State Discovery

CPU Thermal Characterization
- CPU temperature is characterized as normal, high, or critical

Overall Procedure:
- HostThermalDiscovery module reads CPU temperature (step 1)
- CPU temperature characterization (step 2)
- CPU thermal state is reported to Child Kraken
Design: Out-of-band CPU Thermal State Discovery

- **Redfish**
  - A standard that widely adopted in many server products
  - Expose an interface to access Baseboard Management Controller (BMC) through **Redfish API**

- **Overall Procedure:**
  - RFThermalDiscovery module on Parent Kraken sends CPU temperature request to RFAggregator
  - RFAggregator fans-out CPU temperature request to each node
  - RFAggregator forwards CPU temperature to Parent Kraken

- **Benefits:**
  - Agent-less
  - Uses BMC capability and saves CPU cycles
CPU Frequency Scaling Procedure

- Frequency scaling is performed only via in-band mechanism

- **Child Kraken** checks for Frequency Scaling mutations (step 1)

- Kraken analyzes mutation for possible action

- **HostFrequencyScaling** performs mutations to change frequency configs (step 3)
CPU Thermal Mutations

CPU Thermal State Mutation allows Kraken to control CPU thermal state.

When the CPU thermal state is CPU_TEMP_HIGH or CPU_TEMP_CRITICAL, CPU thermal state mutations provide a mechanism to revert to the desired CPU thermal state (CPU_TEMP_NORMAL).

CPU Frequency Scaling Mutations

CPU frequency scaling mutations are governed by real-time CPU thermal states.

The Child Kraken mutates CPU frequency scaling states locally using the HostFrequencyScaling module.
Implementation

- **Hardware Setup:** The hardware infrastructure includes a cluster of **150 Raspberry Pi3 nodes**, **one master node**, and a fan.

- **Software Stack:**
  - **pipxe:** Provides PXE-boot capabilities for Raspberry Pi3 nodes.
  - **rfpipower:** Emulates Redfish power control capabilities. Powers on and off the Raspberry Pi3 nodes remotely.
  - **cpuburn:** Induces load and stresses the Raspberry Pi3 nodes.
  - **RFAggregator:** Emulates and handles the Redfish group requests.
  - **RFThermalDiscovery:** Emulates the Redfish API to monitor CPU temperature of the Raspberry Pi3 nodes using RFAggregator.
  - **HostThermalDiscovery:** Performs CPU temperature monitoring of the Raspberry Pi3 nodes through the in-band mechanism.
  - **HostFrequencyScaling:** Makes use of DVFS to scale up or scale down CPU frequency to control CPU temperature and CPU performance.
Overheated CPU

CPU with "normal" temp.
Evaluation - Maximum CPU temperature attained

Max temperature attained with the highest frequency (performance):

- CPU attained high temperature in ~10 minutes
- CPU was running with maximum load

Max temperature with the lowest frequency (power_save):

- CPU attained high temperature in ~20 minutes
- The attained temperature is comparatively lower than the temperature attained with the highest frequency.
(2) Thermal control at 55 °C
- Maximum temperature was observed ~70 °C

(1) Thermal control at 50 °C
- Maximum temperature was observed ~70 °C
Evaluation - Impact of CPU Temperature Thresholds on Temperature and Frequency (2)

(3) Thermal control at 60 °C
- Maximum temperature was observed ~70 °C

(4) Thermal control at 65 °C
- Maximum temperature was observed > 70 °C
(6) Thermal control at 75 °C

- Maximum temperature was observed ~ 75 °C
- There were lot of back-and-forth switches between the lowest and highest frequencies

(5) Thermal control at 70 °C

- Maximum temperature was observed > 70 °C
- There were lot of back-and-forth switches between the lowest and highest frequencies
Evaluation - Impact of CPU Temperature Thresholds on Temperature and Frequency (4)

(8) Thermal control at 85 °C
- Maximum temperature was observed > 85 °C
- There were few switches between the highest and lowest frequency configs

(7) Thermal control at 80 °C
- Maximum temperature was observed > 80 °C
Discussion and Key Takeaways

- We studied impact of applying thermal control at eight temperature configurations
- Applying thermal control at lower temperature configurations restricted temperature to ~70°C
- Applying thermal control at upper temperature configurations resulted in attaining higher temperature ~85°C
- There is a quite back and forth between PERFORMANCE and POWER_SAVE modes for some temperature configurations

Two Takeaways:
- Applying thermal control at lower temperature configurations is better to slow down the overall temperature progression
- Applying thermal control for a certain time interval is better than just a temperature threshold
Conclusions & Future Work

- Modern CPUs are prone to overheating in **unfavorable thermal conditions**
- Successfully shown the automated healing of an **overheated** CPU temperature of a node
- Our method allows **scaling up the CPU frequency** when the CPU thermal state is **back to normal**
- Implementation leverages state-of-the-art **Redfish** and **Kraken**, which is an HPC automation and control framework

More investigation is needed to:

- Scale down the CPU frequency **coherently to heal a node involved in a group of nodes executing a message passing interface (MPI) workload**
- Perform a **graceful powering-off** of a node when the CPU thermal state reaches an **irreversible critical level**
Thank you!

For more details, please read our paper.
ghazanfar.ali@ttu.edu