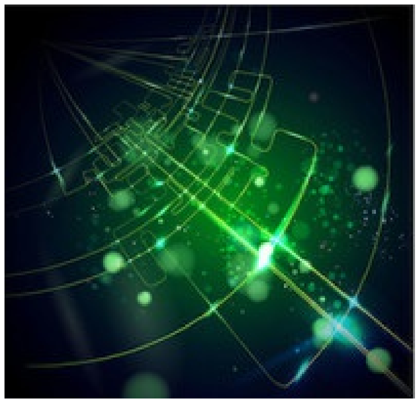
Charming Exascale Power Problems



Power, energy and reliability present major challenges to HPC researchers in their endeavor to build larger machines. As we approach the exascale era, both hardware and software designers need to account for these challenges while optimizing performance. The Parallel Programming Laboratory (PPL) at the University of Illinois at Urbana-Champaign (UIUC) has been actively working on meeting these challenges by leveraging the adaptive runtime system of the Charm++ programming model.

Current petascale machines have Mean Time Between Failures (MTBF) that can be anywhere from a few hours to days. Some reports predict exascale machines will have an MTBF in the range of 35-40 minutes. Intriguingly, past research describes a relation between a processor's temperature and its reliability: failure rates double with every 10C increase in temperature. Our work applies this relationship between processor temperature and reliability by restraining processor temperature, thereby reducing the frequency of faults and consequently improving application performance in fault-prone environments.

There are costs and benefits to improving reliability through temperature control driven by Dynamic Voltage and Frequency Scaling (DVFS). Improved reliability helps not only by directly decreasing failures; it also allows the code to checkpoint less frequently, decreasing overhead. However, it comes at a cost of slower processors and increased load imbalance.

By restraining processor temperatures, we can empower the runtime system to set the expected failure rate of the system, adjusting it within a feasible range. Our control strategy lets each processor work at its maximum frequency as long as its temperature is below a threshold parameter. If a processor's temperature crosses the maximum threshold, it is controlled by decreasing the voltage and frequency using DVFS. When the voltage and frequency are reduced, its power consumption will drop and hence the processor's temperature will fall.

When DVFS adjusts frequencies differently across the cores in a cluster, the workloads on those cores change relative to one another. This can significantly degrade performance of a tightly coupled parallel application, where processors synchronize after a time step before proceeding to the next time step. We mitigate the resultant timing penalty with a load balancing strategy that is conscious of the difference in speeds for different processors.

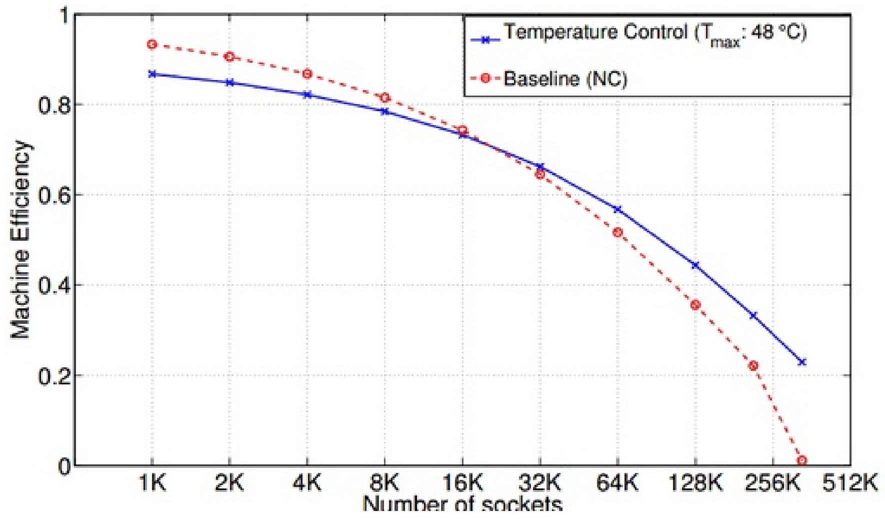
Our load balancing strategy, based on overdecomposition and object migration, uses the Charm++ adaptive runtime system to increase processor utilization. It analyzes the current load of each processor according to its new frequency and determines if it is overloaded or under-loaded. Once this decision is made, our scheme intelligently exchanges objects from overloaded (hot) processors to under-loaded (cold) processors to balance load. Temperature checking and corresponding load balancing can be invoked at user defined intervals.

One twist in this work is that different applications vary both in how hot they will make processors at a given frequency, and in how their performance is affected by different frequencies. Note that this implies different applications may actually experience a different MTBF on the same machine! Thus, we use three applications that present different conditions.

To gauge the effects of temperature control on MTBF and hence application performance, we formulate a model that relates total execution time of an application to reliability and the associated slowdown for temperature restraint. The model accounts for different execution speed at different frequencies, checkpointing overhead and recovery time depending on MTBF, and the additional overhead of experiencing and adapting to load imbalance. We validate the accuracy of our model for each application using a small experimental testbed.

We use our validated model to project the benefits of our scheme for larger machines. Our results point towards a tradeoff between improvement in reliability and the associated cost of applying temperature control. This tradeoff determines the optimal temperature threshold for a given application and machine size.

The following figure compares the machine efficiency (proportion of time spent doing useful work) for a 2D stencil application between a baseline run without temperature control and a constrained run with the temperature threshold set to 48C. Below 32K sockets, we get a lower efficiency than the baseline. However, above 32K sockets, our scheme starts outperforming the baseline case. For reference, the Blue Waters system at NCSA has nearly 50K sockets. For 256K sockets, our scheme is projected to operate the machine with an efficiency of 0.29 compared to 0.08 for the baseline. Finally, for 340K sockets, the baseline efficiency drops to 0.01, making the machine almost nonoperational, whereas our scheme can still operate the machine at an efficiency of 0.22.



These promising results encourage us to extend our work by investigating more detailed models, larger experimental systems, and more advanced fault tolerance protocols, such as message logging and parallel recovery.

This work is part of a research theme in our group: using adaptive runtime control to deal with the challenges presented by sophisticated applications and complexities of hardware. The Parallel Programming Laboratory has developed Charm++ for the past 20 years as a production quality parallel programming language, used in many CSE applications, including the Gordon Bell-winning biomolecular simulation program NAMD.

Author Biography:

Osman Sarood is a final year PhD student in the UIUC Computer Science department. His research is focused on performance optimization under thermal and power constraints.

Esteban Meneses is a Research Assistant Professor working in the Center for Simulation and Modeling at the University of Pittsburgh. His research is focused on load balancing and fault tolerance techniques for large-scale parallel applications. He holds a PhD degree in Computer Science from UIUC.

Laxmikant Kale received his PhD in computer science from State University of New York, Stony Brook, in 1985. He joined the Computer Science faculty of UIUC as an Assistant Professor in 1985, where he is currently employed as a full Professor. His research spans parallel computing, including parallel programming abstractions, scalability, automatic load balancing, communication optimizations, and fault tolerance. He has collaboratively developed several scalable CSE applications.

Tags: exascale