

High-performance, power-aware distributed computing framework

Kirk W. Cameron Rong Ge Xizhou Feng Drew Varner Chris Jones

Scalable Performance Laboratory
Department of Computer Science and Engineering
University of South Carolina

Problem Statement:

Left unchecked, the fundamental drive to increase peak performance using tens of thousands of power hungry components will lead to intolerable operating costs and failure rates.

Earth Simulator requires 18 megawatts of power. Petaflop systems may require 100 megawatts of power, nearly the output of a small power plant (300 megawatts). This would supply 1.6 million 60-watt light bulbs; lighting requirements for a small city. At \$100 per megawatt (\$.10 per kilowatt), peak operation of this petaflop machine is \$10,000 per hour. Pessimistically, annual operational costs surpass \$85 million! Conservative estimates of 20% peak power consumption imply a lower, but not necessarily manageable \$17 million annual operational cost (\$2,000 per hour). These estimates ignore the additional cost (~40%) of dedicated cooling.

Commodity components fail at an annual rate of 2-3%. A petaflop system of about 12,000 nodes (CPU, DRAM, NIC, disk) will sustain hardware failures once every twenty-four hours. Component life expectancy decreases 50% for every 10° C (18° F) temperature increase. Reducing a component's operating temperature the same amount (consuming less energy) doubles the life expectancy.

Our approach

High-performance, power-aware distributed computing reduces power and energy consumption of distributed applications and systems without sacrificing performance. Recent work has shown application characteristics of single-processor, memory-bound non-interactive codes (Hsu and Kremer) and distributed, interactive web services (Bianchini et al) can be exploited to conserve power and energy with minimal performance impact. Our novel approach is to exploit parallel performance inefficiencies characteristic of non-interactive, distributed scientific applications, conserving energy without impacting time-to-solution (TTS) significantly, reducing cost and improving reliability.

Results

We present a framework to profile/control, analyze and optimize distributed power-performance implemented on a 32-node PIII-based and a 16-node Centrino-based cluster.

1) Power-performance profile/control. We created a tool suite (PowerPack + PA-MPICH) enabling power and energy profiling and control of parallel systems. PowerPack consists of tools for multiple multi-meter data management and application-level libraries to synchronize multi-meter measurements to source code. PA-MPICH is a power-aware version of MPICH that integrates dynamic processor voltage scaling (DVS) in MPI calls. Overheads for mode transitions vary (10-26 microseconds) with transition type, direction (up or down), and locality (time between transitions). Collective transitions are more costly (~450 microseconds on 16-nodes, 1.4 GHz to 600 MHz).

2) Power-performance analysis. To the best of our knowledge, this is the first study of its kind. We use direct measurements to show distributed scientific application power profiles are often regular (coinciding with communication and computation), yet patterns vary by node, application (e.g. NAS PB), component (CPU, DRAM, disk, NIC), and workload. For example, parallel NAS FT consumes ~60 watts during computation and ~40 watts during communication. Strong scaling improves performance, however total energy consumption is non-linear with the number of nodes, varying with workload and application. NAS code EP consumes ~7500 Joules overall for 1, 2, 4, 8 and 16 processors while TTS is halved respectively; 32 processors use ~8000 Joules (TTS halves again). NAS code FT consumes ~3200 Joules for 1 processor increasing dramatically up to ~13600 Joules on 8 processors (TTS improves steadily); 16 processors improves TTS while energy consumption changes little. Component analysis reveals CPU and memory typically dominate power consumption and their use correlates to these energy variances.

3) Power-performance optimization. We provide optimization results for two DVS strategies: static and dynamic. Application characteristics determine the effectiveness of each approach. In static mode, transitions to DVS states are fixed for the duration of an application's run-time. For example, energy savings of 28% are obtained for parallel transpose (5000 x 5000 matrix, 4 nodes) with only 1% performance loss at 800 MHz (versus 1.4 GHz). Similarly, NAS FT showed 16% energy savings on 8 nodes with 4.53% performance loss after powering down CPUs. In contrast, scaling NAS EP from 1 to 8 nodes conserves overall energy 25% by shortening TTS (57%) running in faster 1.4 GHz mode (versus 600 MHz). We are experimenting with dynamic transitions (triggered using PA-MPICH) by code blocks in parallel applications. Initial results show energy savings of 6.45% (performance loss of .26%) for single MPI_Send of 128 Kbytes after powering down CPUs dynamically from 1.4 GHz down to 1 GHz. We will provide further results in the poster.

Published Abstract

The fundamental drive to increase peak performance using tens of thousands of power hungry components will lead to intolerable operating costs and failure rates.

Petaflop systems may require 100 megawatts of power, nearly the output of a small power plant (300 megawatts). This would supply 1.6 million 60-watt light bulbs; lighting for a small city. At \$100 per megawatt, peak operation of this petaflop machine costs \$85 million annually (\$10,000/hour). Conservative estimates of 20% peak power consumption imply a lower, but not necessarily manageable \$17 million annually (\$2,000/hour). These estimates ignore the cost (+40%) of dedicated cooling.

Our novel approach is to exploit parallel performance inefficiencies characteristic of non-interactive, distributed scientific applications, conserving energy without impacting time-to-solution (TTS) significantly (<5%) and reducing cost (e.g. 25% overall energy savings).

We present a framework to profile/control, analyze and optimize distributed power-performance implemented on a 32-node PIII-based and a 16-node Centrino-based cluster.