Software Optimization for Performance, Energy, and Thermal Distribution: Initial Case Studies

Md. Ashfaquzzaman Khan, Can Hankendi, Ayse Kivilcim Coskun, Martin C. Herbordt

{azkhan|hankendi|acoskun|herbordt}@bu.edu

Electrical and Computer Engineering

BOSTON UNIVERSITY

HPEC’11

09/21/2011

Funded by
Dean’s Catalyst Award, BU.
Energy consumption by data centers increases by **15% per year** [Koomey 08].

Motivation

Source: Pike Research

(BAU: Business as Usual)
Energy consumption by data centers increases by \textbf{15\% per year} [Koomey 08].

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{data_center_emissions.png}
\caption{Data Center Greenhouse Gas Emissions by Scenario, World Markets: 2009-2020}
\end{figure}

\textbf{Goal: Reduce energy consumption through temperature-aware software optimization}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{holyoke_dam.png}
\caption{Holyoke Dam}
\end{figure}

(BAU: Business as Usual) Source: Pike Research
Software to Reduce Power (basic)

- If cores are idle shut them off (Dynamic Power Management)
- If you have slack in the schedule, slow down (Dynamic Voltage/Frequency Scaling)
Temperature is also a concern

- Prohibitive cooling costs
- Performance problems
  - Increased circuit delay
  - Harder performance prediction at design
- Increased leakage power
  - Reaches 35-40% (e.g., 45nm process)
- Reliability degradation
  - Higher permanent fault rate
  - Hot spots
  - Thermal cycles
  - Spatial gradients

Figure: Intel

Figure: Santarini, EDN’05
Software to Reduce Temperature (basic)

- If the chip is too hot, back off (Dynamic Thermal Management)
A More Sophisticated Strategy*

- **Physical Goals**
  - Minimize total energy
  - Minimize max energy for each core
  - Per core -- reduce time spent above threshold temperature
  - Minimize spatial thermal gradients
  - Minimize temporal thermal gradients

- **Software Goals include**
  - Avoid hot spots by penalizing clustered jobs

*Coskun et al. TVLSI 2008*
Energy or performance-aware methods are not always effective for managing temperature. We need:

- Dynamic techniques specifically addressing temperature-induced problems
- Efficient framework for evaluating dynamic techniques
Opt for P ` Opt for E ` Opt for T

- P vs. E:
  - For E, less likely to work hard to gain marginal improvement in performance

- E vs. T:
  - For T, less likely to concentrate work temporally or spatially
Problems with these approaches

- Assume slack
- Assume knowledge of tasks
- Assume (known) deadlines rather than “fastest possible”
- Assume that critical temps are a problem
- Do not take advantage of intra-task optimization

Plan: Design software to be thermally optimized
Energy consumption by data centers increases by **15% per year** [Koomey 08].

High temperature:
- Higher cooling cost, degraded reliability

Software optimization for improving **Performance**, **Energy**, and **Temperature**:
- Potential for significantly better P, E, T profiles than HW-only optimization

**Jointly optimizing** P, E and T is necessary for high **energy-efficiency** while maintaining **high performance** and **reliability**.

*Source: Pike Research (BAU: Business as Usual)*
Contributions

- Demonstrating the need for optimizing PET instead of optimizing PE or PT only.
- Developing guidelines to design PET-aware software.
- Providing application-specific analysis to design metrics and tools to evaluate P, E and T.
- Two case studies for SW-based optimization:
  - **Software restructuring and tuning**
    - 36% reduction in system energy
    - 30% reduction in CPU energy
    - 56% reduction in temporal thermal variations.
  - **Investigating the effect of software on cooling energy**
    - 3°C increase in peak temperature translates into 12.7W increase in system power.
Outline

- Motivation/Goals
- Methodology
- Case studies
  - Software tuning to improve P,E and T
  - Effect of temperature optimization on system-energy
- Conclusion
- Questions
Measurement Setup

- System-under-test:
  - 12-core AMD Magny Cours processor, U1 server
Power and Temperature Estimation (ideal)

Power Measurements

Layout for Magny Cours

CPU0  L2 0  L2 1  CPU1  L2 2  CPU2

Northbridge Bus

CPU3  L2 3  L2 4  CPU4  L2 5  CPU5

Per-core power traces

HotSpot-5.0 Thermal Simulator [Skadron ISCA’03]

Temperature trace for each core

L2 caches
Per-core power and temperature measurements are often unavailable.

- Sparse power measurements
- Performance monitors

![Layout for Magny Cours](image)

- HotSpot-5.0 Thermal Simulator
  [Skadron ISCA’03]

- Temperature trace for each core

- Our Power Estimation Model

- Per-core power traces

- L2 caches
Motivation: Per-core power and temperature measurements are often not available.

We custom-designed six microbenchmarks to build the power estimation model.

**Power Estimation Methodology**

- **Performance counter data** + power measurements → **Linear Regression** → **Coefficients**

**Hardware events collected through the counters:**

- CPU cycles
- Retired micro-ops
- Retired MMX and FP instructions
- Retired SSE operations
- L2 cache misses
- Dispatch stalls
- Dispatched FPU instructions
- Retired branch instructions
Microbenchmarking
Microbenchmarking
Microbenchmarking
Microbenchmarking
Microbenchmarking

- In-cache matrix multiplication (double)
- In-cache matrix multiplication (short)
- Intensive memory access w/o sharing
- Intensive memory access w/ sharing
Microbenchmarking

- In-cache matrix multiplication (double)
- In-cache matrix multiplication (short)
- Intensive memory access w/o sharing
- Intensive memory access w/ sharing
- Intensive memory access w/ frequent synchronization
- In-cache matrix multiplication (short-simple)
**Power Model Validation**

*Power estimation for microbenchmarks*

- Projected
- Measured

*Average error for PARSEC benchmarks is less than 5%.*
Outline

- Motivation/Goals
- Methodology
- Case studies
  - Software tuning to improve P,E and T
  - Effect of temperature optimization on system-energy
- Conclusion
- Questions
Parallelization of dedup

- A kernel in PARSEC benchmark suite
- Implements a data compression method called “deduplication”
- Combines local and global compression
- “deduplication” is an emerging method for compressing:
  - storage footprints
  - communication data

Parallel Stages

1. Break up of data into coarse-grained chunks
2. Fine-grained segmentation
3. Hash-computation
4. Compression (Ziv-Lempel)
5. Hashes and compressed data output
Default dedup (Pipelined)

- OS schedules the parallel threads as data become available
- **Heavy data dependency** among threads
- **Increased** need for **synchronization**
- **Increased data movement** (less reuse) inside processing cores
- **Uneven computational load** leads to uneven power consumption
Task-decomposed dedup

- More data reuse
- Less synchronization
- Balanced computation load among cores
- Improved performance, energy and temperature behavior
- Parameter optimized for target architecture

Proposed version: Task-decomposed
Parameter Tuning

- Dedup threads takes specific number of tasks from the queue (default=20)

- **Number of tasks** between two synchronization points is critical for the application performance

- Tuning the number of tasks **balances the workload across threads**
  - Tuned value = 10

**Performance scaling of default and proposed version of dedup**
Power & Temperature Results

**Per-core temperature (°C)**

- **DEFAULT**
  - Chip power (W)
  - Per-core temperature

- **TASK-DECOMPOSED**
  - Chip power (W)
  - Per-core temperature

- **TASK-DECOMPOSED & PARAMETER TUNED**
  - Chip power (W)
  - Per-core temperature
Energy & Temperature Results

- Parameter-tuned task-based model improvements with respect to default parallelization model:
  - 30% reduction in CPU energy
  - 35% reduction in system energy
  - 56% reduction in per-core maximum temporal thermal variation
  - 41% reduction in spatial thermal variation
Motivation/Goals

Methodology

Case studies
- Software tuning to improve P,E and T
- Effect of temperature optimization on system-energy

Conclusion

Questions
Effect of SW Optimization on Temperature

- Optimizing temperature at µs granularity has substantial benefits.

- Quantifying effect of temperature optimization on system power
  - mPrime stress test
  - Microbenchmarks
mPrime Stress Test

- mPrime (Prime95 on Windows) is a commonly used stress benchmark
- 25°C ➔ +17W system power, +10W chip power

![Graph showing power and temperature trends over time]
Two benchmarks with different P and T profiles:

- **In-cache matrix multiplication (double) — MM**
  - **High power** due to stress on FPU units
- **Intensive memory access w/ frequent synchronization — Spinlock**
  - **Low power** due to memory and synchronization operations

**Effect of Temperature on System Power**

- $\Delta$ Peak Temp. = 3°C
- $\Delta$ Chip Power = 10.8W
- $\Delta$ System Power = 23.5W
We presented our initial results in application-level SW optimization for performance, energy and thermal distribution.

Our evaluation infrastructure includes: direct measurements and power/temperature modeling.

We presented 2 case studies:
- Effect of code restructuring on P, E, and T.
  - Software optimization reduces system energy and maximum thermal variance by %35 and 56%.
- Potential energy savings from temperature optimization:
  - 3°C reduction in peak temperature causes 12.7W system power savings.

Future work: Expanding the SW tuning strategies for parallel workloads, explicitly focusing on temperature.