Automatic Management of TurboMode

David Lo
Christos Kozyrakis

Stanford University
http://mast.stanford.edu
Executive Summary

- **TurboMode** overclocks cores to exhaust thermal budget
  - An important performance feature of multi-core x86 servers

- **Challenge:** turbomode does not always benefit workloads
  - Naively turning TurboMode on often leads to high energy waste

- **Solution:** predictive model to manage TurboMode (on/off)
  - Using machine learning on performance counter data
  - Eliminates negative cases, boosts ED and ED² by 47% and 68%
What is TurboMode (TM)?

- Dynamic overclocking of cores to exhaust thermal budget
  - Matches actual power consumption to max design TDP
  - Big performance gains: up to 60% frequency boost
  - Found on all modern x86 multi-cores

- TurboMode control
  - Black-box HW control decides when and how much to overclock
  - SW has limited control: can only turn TurboMode on/off
Characterizing TurboMode

- Evaluate the effects of TM across the board
  - Efficiency metrics: EDP, ED$^2$P, throughput/W, throughput/$\$, ...
  - Many hardware platforms: Intel/AMD, server/notebook
  - Many workloads: SpecCPU, SpecPower, websearch, ...

- Characterization
  - Run with TurboMode on and TM off
  - Compare impact on all of efficiency metrics
Efficiency Metrics

Guidelines
- We all care about performance and energy consumption
- Capture both latency and throughput workloads

Metric recap
- ED: latency & energy
- ED²: latency & energy, more weighted towards latency (think servers)
- Throughput/W: throughput & energy
- Throughput/\$: throughput & cost efficiency (think datacenter TCO)
Evaluation Hardware

- Intel Sandy Bridge server [SBServer]: 19% max boost
- Intel Sandy Bridge mobile [SBMobile]: 44% max boost
- AMD Interlagos [ILServer]: 59% max boost
- Intel Ivy Bridge server [IBServer]: 12% max boost
- Intel Haswell server [Hserver]: 13% max boost
Evaluation Workloads

- Representative of multiple domains
- CPU, memory, and IO workloads

- Single-threaded SpecCPU benchmarks
- Multi-programmed SpecCPU mixes
- Multi-threaded PARSEC
- Enterprise SPECpower_ssj2008
- Websearch

>100 configs
Observation: No Optimal On/Off Setting

Sandy Bridge Server

Interlagos Server 127%

Sandy Bridge Mobile

% improvement over TurboMode off

Mix 1 Mix 2 Websearch

Mix 1 Mix 2 Websearch

Mix 1 Mix 2 Websearch

Workload

ED Workload

ED² QPS/W QPS/$
Observation: TM leads to High Variance on Efficiency

50% mixes benefit from TM
50% mixes suffer due to TM
Characterization Analysis

- TurboMode mostly benefits CPU bound workloads
  - Boost in performance and efficiency from higher frequency
  - SpecCPU mixes of CPU-intensive workloads, SpecPower, websearch, ...
- TurboMode ineffective when memory/IO bound
  - Interference on memory/IO really aggravates this
  - Small/no performance gain, high energy waste with higher frequency
  - SpecCPU mixes of memory-intensive workloads, canneal, streamcluster, ...
- Applications have multiple phases
  - CPU bound vs. memory/IO bound
  - SpecCPU mixes
TurboMode Control

- Naïve TM control
  - Always off: miss boost on CPU bound applications
  - Always on: suffer inefficiency on interference-bound applications

- Need dynamic TM control
  - Understands applications running and metric of interest
  - Predicts optimal setting (on/off), adjust dynamically to phases
  - No a priori knowledge of applications, no new hardware needed
Predictive Model for TurboMode

- **Idea:** use runtime info to dynamically predict TM benefits

- Focus primarily on detecting memory interference
  - Build predictive model based on performance counters
  - Use performance counters & model to predict interference severity
  - If too severe, turn off TurboMode
Autoturbo: Predictive Control for TurboMode

Sample perf counters per core

App properties per core

Enable/disable TurboMode

Training data

App N

TM on/off

Metric

Core 1  Core 2  Core N-1  Core N

Sample perf counters per core

App properties per core

Enable/disable TurboMode

Metric
Training the Predictive Model

**Raw training data**
- Single SpecCPU, TurboMode on
- Single SpecCPU, TurboMode off
- Single SpecCPU +stream, TurboMode on
- Single SpecCPU +stream, TurboMode off

**Feature selection**

**Model selection**
- Naïve Bayes: 85%
- Logistic Regression: 81%
- Nearest Neighbors: 73%
- Decision Tree: 75%
Model Validation

- **Model accuracy**: ~90% on cross-validation

- Best counters: those that indicate memory-bound workload
  - **SBServer/SBMobile**: % cycles with outstanding memory requests, ...
  - **ILServer**: L2 MPKI, # requests to memory/instruction, ...

- CPU/thermal intensity counters don’t correlate strongly!
  - E.g., floating-point intensity counters
Autoturbo Evaluation

- Used autoturbo in conjunction with workloads
  - Evaluation workloads are apps other than single-thread SpecCPU
- Measure efficiency metrics

- Compare against
  - **Baseline**: TurboMode is always off
  - **Naïve TM**: TurboMode is always on
  - **Static oracle**: TurboMode on if leads to benefit for the overall run
Autoturbo results

Sandy Bridge Mobile QPS/$

-5% 0% 5% 10%
QPS/$ improvement

Naïve Auto Static Oracle

Sandy Bridge Server ED²

-20% 0% 20% 40%
ED² improvement

Naïve Auto Static Oracle

Gains over never using TurboMode
Gains over always using TurboMode

HPCA-20 February 19, 2014
Autoturbo Analysis

- **Autoturbo** gets best of both worlds
  - Reduces cases where TM causes efficiency degradation
  - Keeps cases where TM leads to benefits

- **Autoturbo** often disables TM even though it is beneficial
  - **Cause**: the interference predictor assumes worst case interference

- **Autoturbo** beats the static oracle
  - **Cause**: **Autoturbo** can take advantage of dynamism during the run
Conclusions

- TurboMode is useful but must be managed dynamically

- This work: dynamic TurboMode control
  - Predictive model for memory interference
  - Dynamic control with no hand-tuning needed
  - Eliminates efficiency drops, maintains efficiency gains of TurboMode

- Future work
  - Apply similar approach to manage advanced power settings
autoturbo dealing with a phase change

autoturbo dynamic adjustment on Sandy Bridge Mobile

Memory interference occurs mid-workload