Quality of Service-Aware, Scalable Cache Tuning Algorithm in Consumer-based Embedded Devices

M. Hammam Alsafrjalani and Ann Gordon-Ross

Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA

INTRODUCTION

Background & Motivation

- Consumer-based Embedded Devices (CEDs) are ubiquitous

- High consumer-defined quality of service (QoS) expectations (consumer goals) and stringent energy constraints, often contending

- **Goal:** innovate a CEDs design approach that enables CEDs to adhere to consumer goals and energy requirements, independently of applications/deployment
- **Challenges**
  - Consumer goals must be known during design time
    - *Implausible given ubiquity nature of systems (diverse consumers)*
  - Software applications must be known during design time[1]
    - *Implausible, given rapid growth of unknown third party applications (~1.5 million+ apps for Android)*

Methodology

Using configurable hardware and associated tuning algorithms

- **Configurable hardware**
  - Contains modifiable/tunable parameters
    - *Clock, voltage, memory, etc.*
  - Parameter values change to application's hardware requirements
  - Parameter values dictated by a tuning algorithm
- **Tuning algorithm**
  - Monitors application execution, evaluate energy consumption and quality of service
  - Explores the design space (available parameter values)
  - Adjusts parameter values such that CED
    - *Meets QoS expectation and consume lowest energy*
- **Target effective component**
  - Cache memory
    - *Has high impact on energy and performance*

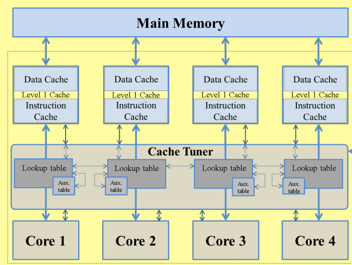
Contribution

Application-scalable hardware and runtime tuning algorithm

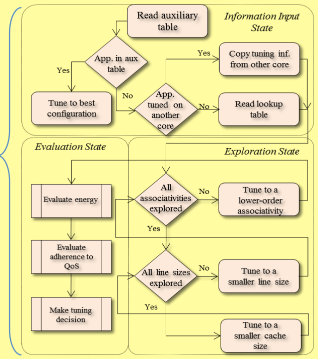
- **Scalable hardware**
  - Compressed tuning information into auxiliary tables
  - Employed LRU policy to enable scalability of application-tuning
    - *Only 4% area (hardware) overhead*
- **Dynamic, general purpose CED cache-tuning algorithm**
  - Requires no a priori knowledge of applications
  - Flexible: Conservative and Moderate modes
    - *For disparate QoS expectations*
    - *Trades off energy savings for higher QoS*

Design

Configurable Hardware



Dynamic Tuning Algorithm



Hardware Specifications

- **Quad-core system**
- **Configurable, private level 1 cache memory**
  - Possible configurations: 2KB 1-way; 4KB 1- or 2-way; and 8KB 1-, 2-, or 4-way
- **Hardware cache tuner**
  - Global, monitors all cores
  - Tunes the cores based on the tuning algorithm
  - Stores tuning information in lookup and auxiliary tables

Algorithm Stages

- **Information Input:** Reads tuning information from lookup and auxiliary tables; tunes to best configuration, or resumes exploration
- **Exploration:** Determines the configuration to tune the hardware to, based on the tuning information and tuning mode; updates the tables
- **Evaluation:** Evaluates if the configuration saves energy and/or degrades QoS; decides if tuning is done

Energy Model

E(total) *= E(sta) + E(dyn)*
E(dyn) *= cache_hits * E(hit) + cache_misses * E(miss)*
E(miss) *= E(off_chip_access) + miss_cycles * E(CPU_stall) + E(cache_fill)*
Miss Cycles *= cache_misses * miss_latency + (cache_misses * (line_size/16)) * memory_band_width)*
E(sta) *= total_cycles * E(static_per_cycle)*
E(static_per_cycle) *= E(per_Kbyte) * cache_size_in_Kbytes*
E(per_Kbyte) *= (E(dyn_of_base_cache) * 10%) / (base_cache_size_in_Kbytes)*

Quality of Service Logic

QoS Expectation *= performance > threshold*
Threshold *= minimum acceptable performance*
If *performance < threshold, QoS_degradation = true*
Else *QoS_degradation = false*

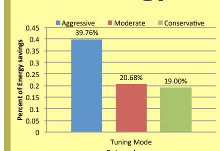
Evaluation Methodology

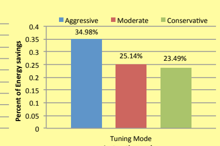
Energy savings and QoS: our approach vs. base system and prior work

- **Energy savings:** Measured application's energy consumption of best configuration determined by the tuning algorithm, calculated energy percent decrease with respect to the base system, and averaged the energy savings for all applications (34 total)
- **Quality of Service:** Calculated the number of QoS degradation occurrences while tuning each application, compared the result to a perfect (no QoS degradation) system, and averaged the QoS degradation for all applications (34 total)
- **Comparison to prior work:** Incorporated a tuning mode which represents prior work approach; an aggressive mode which determined the lowest-energy configuration without considering QoS

Experimental Results

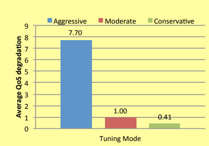
Energy Savings Results

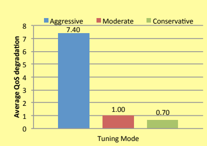




- Aggressive(prior work) achieved highest savings
  - *However* required priori knowledge of application
- Moderate and conservative modes results in comparable energy savings

Quality of Service Impact Results





Otherwise →

- Aggressive(prior work) imposed QoS degradation as high as **7X**, on average
- Moderate mode imposed QoS degradation, at most **1**
- Conservative mode average QoS degradation < **1.00**

Conclusion

Designing approach for CED's

- Saving energy does not require a priori knowledge of applications
  - Enables scalability to an arbitrary number of unknown (future), third-party applications
  - Meets consumer goals
- Tuning modes provide flexible adherence to QoS expectations
  - Can incorporate tuning modes for disparate conditions/situations, such as

Environmental | **Experience/Personal**

Comparison to prior work

Prior work[1][2] saved more energy, however

- (-) Incurred as much as 109.2% and 132.5% more tuning overhead while tuning
- (-) Requires long applications execution to amortize the tuning overhead

Prior work degraded QoS up to 7X more, compared to our approach

- (-) Can be completely avoided only with a priori knowledge of applications[1]
  - Not possible for CEDs

Future expansion

Interdisciplinary collaboration

- Psychology behind consumer expectations and its impact on design constraints
  - *Quantification of consumer feelings, moods, and thoughts*
  - *Other factors such as time of use (day, night, etc.), place of operation (office, construction site, on-route, etc.), ...*

Architecture innovations

- Performance & energy expectations vs. privacy & security
- Adaptability to Internet of Things (IoT) devices
  - Tuning through IoT

References

[1] Wang, W., Mishra, P., Gordon-Ross, A. "SACR: Scheduling-Aware Cache Reconfiguration for Real-Time Embedded Systems," *Int. Con. on VLSI Design*, 2009.

[2] Zhang, C., Vahid, F. "Cache configuration exploration on prototyping platforms," *IEEE International Workshop on Rapid Systems Prototyping*, 2003