Fast Design Space Subsetting

University of Florida
Electrical and Computer Engineering Department
Embedded Systems Lab

Motivation & Greater Impact

Energy & Data Centers

- Estimated[1] energy by servers data centers *only* for 2011
  - More than 100 billion kWh
  - Apprx $7.4 billion annual cost in energy
  - Require about 10 power plants
    - At $1.5 billion dollars each

[1] "Report to Congress on Server and Data Center Energy Efficiency", U.S. Environmental Protection Agency, August 2, 2007

System Design Challenges

- Goal of system design
  - *Reducing energy*, increasing performance, etc.
- Applications have specific system requirements
  - *Example*
- System configurations affect application energy/performance
  - Examples: fast clock and I/O bound, cache size and data size
- Myriad of options to consider = large design space
  - Voltage, clock frequency, bus width, cache parameters, etc

Example

- Instruction cache
  - Size 128B to 64kB = 10 options
  - Line size 4, 8, 16, 32 or 64B = 5 options
  - Associativity: 1, 2, 4, 8, 16 = 5 options
  - Total is 250
- Data cache: same value, 250
- uProcessor to instruciton cache buses
  - bit width: 2, 4, 8, 16, or 32
  - bus encoding: binary, bus-invert, grey coding
  - Total: 5*3 = 15 options
- uProcessor to instruciton cache buses to data cache bus:
  - same options, 15
- Same parameters for address bus
  - 15 and 15 for instruction and data caches
- Cache to main memory bus
  - Two busses, same parameters
- Voltages:
  - 1.0 to 4.1, in 0.1 increments: 32 options
- Total options = 250*250*15*15*15*15*15*15*32 = 22 trillion options!!!

Other Challenges

- Options exponentially grow in number for multi- and many-core systems
- Determining best configuration becomes harder
- What about target applications?
  - Redesign for each set?

A Good Candidate for Research

- Cache memory consumes about 40% of core energy
- Cache memory is very well studied
- Configurable caches
  – Enable parameters to be specialized to application requirements
  – Provide plethora of options ➔ very large design space ➔ finding best cache configuration is has challenging
- Cache tuning
  – Process of changing cache parameters to obtain different cache configuration
- Good configuration
  – A configuration that adheres closely to the design goal; e.g., consumes nearly as much energy as a best configuration

Tradeoffs for Cache Tuning

- Large design space
  - Finer optimization vs. tuning overhead
  - Finer optimization = larger design space
  - Even more with multi-/many-core and shared memory systems
- Goal of study
  - Reduce design space
  - Keep good configurations

Reducing Design Space

- Reduce design space into smaller subsets
  - Subsetting: trimming the design space to a very small subset of good configurations
- Challenges
  - How to select which configuration to remove?
    - What is the energy impact if a configuration is removed
      - Is this configuration the best configuration for most or some application?
  - This set must contains most promising configurations
  - Set must be large enough to contain good candidates and small enough to reduce design space efforts
- Target
  - smallest set meets the threshold
  - Threshold is maximum energy consumption increase
- What configurations left in the subset?
  - Good luck

Design Space Subsetting

- Brute Force method
  - Evaluate all possible subsets size from 1 to *m*
  - Decide a threshold
- Use clustering and data mining algorithms
  - Hierarchical clustering
  - Other data mining techniques

Brute Force

- Example: configurable cache
  - 3 sizes, 2 associativities, 3 line sizes
  - Subset size $|S|$

	16B	32B	64B
2K_1W	c_1	c_7	c_{13}
4K_1W	c_2	c_8	c_{14}
4K_2W	c_3	c_9	c_{15}
8K_1W	c_4	c_{10}	c_{16}
8K_2W	c_5	c_{11}	c_{17}
8K_4W	c_6	c_{12}	c_{18}

- Brute Force evaluation
  - Selecting a configuration given by $N(|S|) = \dfrac{m!}{|S|!\,(m-|S|)!}$
  - $\sum_{|S|=1}^{18} N\binom{18}{|S|} = 262{,}143$
  - For $|S| = 4$, ave energy within 5% of complete design space!

Alternative Approach

- Using data mining algorithms
  - If configurations can be viewed as data points
    - Give best data points that resemble the entire design space
  - This is analogous to data mining problems
    - E.g., time series segmentation, graphic decimation
    - Remove nuances of colors from a picture, reserve essential nuances: i.e., remove nuances that won't distort picture colors
    - Nuances → configurations
    - Color distortion → energy consumption
  - What is a good time series segmentation algorithm
- SWAB
  - SWAB is a combination of **S**liding **W**indow **A**nd **B**ottom-Up data mining algorithms

SWAB & Design Space Exploration

- Evaluates adjacent pairs of configurations
  - Compares energy of all applications using these configurations
  - Keeps the configuration resulted in lower energy

```
Begin S = C;                        // start with complete configuration space
While (|S|>x)                       // repeat until x configurations remain
    For all adjacent pairs (cj,ck)
        For all applications ai
            find_μΔ;                // evaluate average energy change
        End
    End
    Min_pair = find_Pair_min_μΔ     // return pair with minimum μΔ
   merge_pair(Min_pair);
   S = S-ci;                        // remove merged configuration
End
Return (S);
end.
```

$$e_{\Delta}(c_j, c_k, a_i) = \frac{e(c_j, a_i) - e(c_k, a_i)}{e(c_k, a_i)}$$

$$\mu_{\Delta}(c_j, c_k) = \frac{1}{n}\sum_{i=1}^{n} e_{\Delta}(c_j, c_k, a_i)$$

SWAB Requirements

- Based on previous algorithm, what do you think a designer needs to know?

SWAB Requirements

- Applications, configurations, energy to run applications for each configurations, etc
- Can we eliminate any of these requirements?
  - Or at least reduce it?

Less Applications?

- Run SWAB on a small set of applications
  - Called training applications
- Challenge
  - which applications?
- Proposed research
  - Select applications randomly and analyze results
  - Select applications based on cache miss rate

Random Application Selections

- Randomly select a set of applications from all anticipated applications
  - These applications are the *training set applications*
  - The remaining, unselected applications, are the *testing set applications*
- Apply SWAB to training set and complete cache configuration design space
  - Determines design space subsets for all subset sizes
- Evaluate the subsets' qualities:

  (A) With respect to best configuration in complete design space

  1. Run testing set with best configuration from subsetted design space, and measure energy consumption
  2. Run testing set with best configuration from complete design space, and measure energy consumption
  3. Compare these energy consumptions

  - *Less energy increase means higher quality subset*

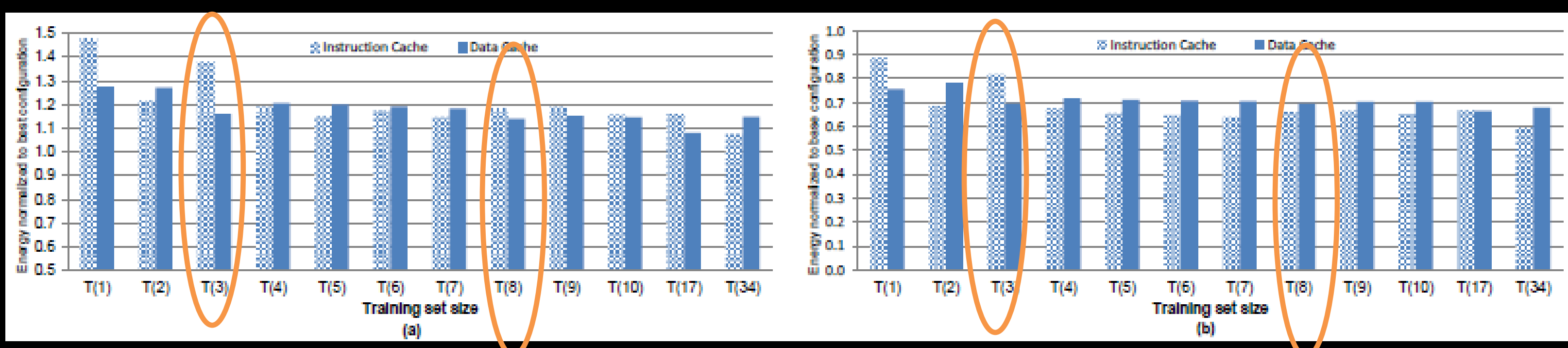
  (B) With respect to base configuration

  1. Run testing set with best configuration from subsetted design space, and measure energy consumption
  2. Run testing set with base configuration from complete design space, and measure energy consumption
  3. Compare these energy consumptions

  - *More energy savings means higher quality subset*
- Evaluate for all combinations sizes of randomly-selected training set applications

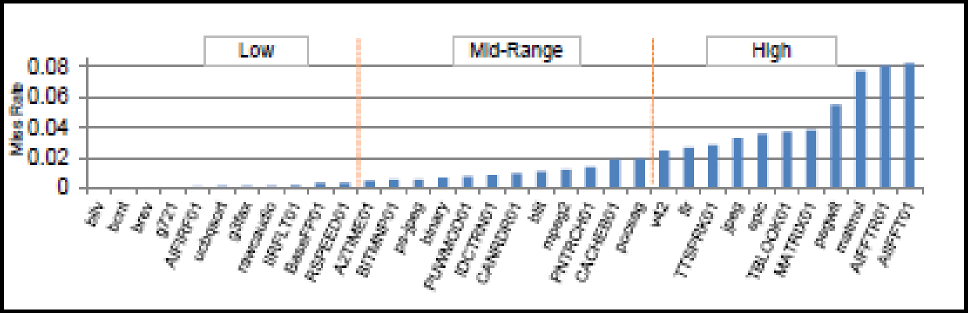
Random application selection Results

- Selected application group size 1➔ all
- Compare energy w.r.t to full design space
- $T_{(n)}$; n = number of applications



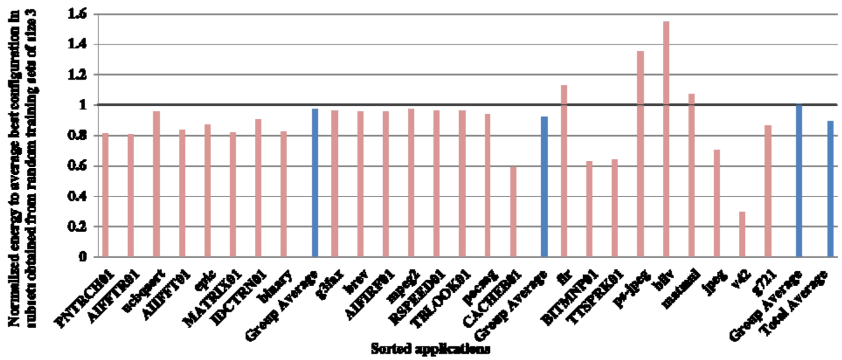
Cache Miss Rate Application Selection

- Given a set of anticipated applications, sort applications into domain-specific groups based on cache hierarchy requirements—miss rate using base configuration

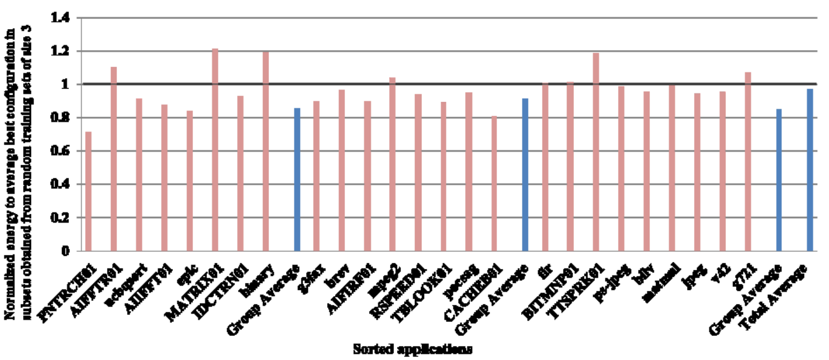


- Select three training applications from each group
  - Called <u>*domain specific training set applications*</u>
  - Unselected applications called <u>*testing set applications*</u>
- Apply SWAB to training set and complete cache configuration design space
  - Determines design space subsets for all subset sizes for each group
- Evaluate the subsets' qualities:
  1. Run testing set with best configuration from domain-specific subsetted design space, and measure energy consumption
  2. Run testing set with best configurations from subsetted design space, obtained with random training sets, and measure average energy consumption
  3. Compare energy consumptions
  - <u>*Lower energy*</u> values using domain-specific subsets means <u>*higher quality subset*</u>

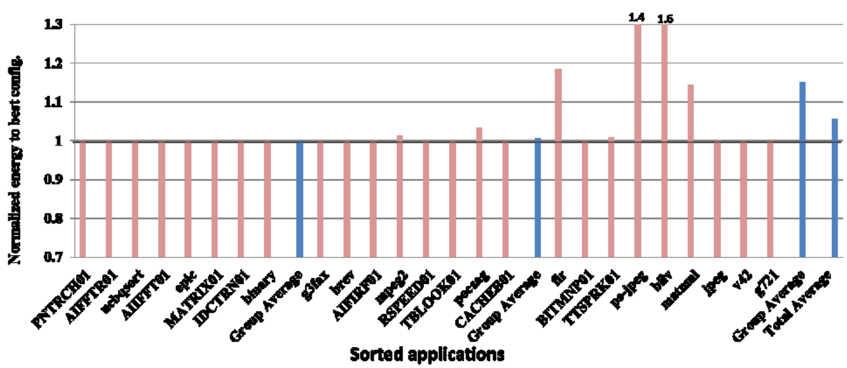
Cache Miss Rate Application Selection Results



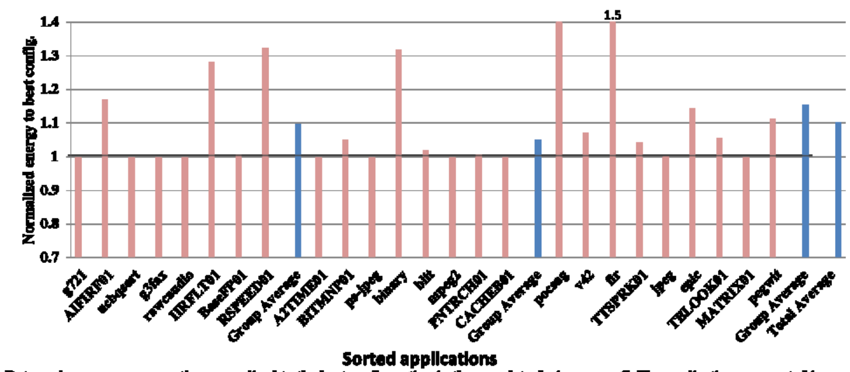
Instruction cache energy consumptions normalized to the average best configuration of each application in *T3*. The applications are sorted in ascending cache miss rate order from left to right with demarcations showing the low, mid-range, and high miss-rate groups.



Data cache energy consumptions normalized to the average best configuration of each application in *T3*. The applications are sorted in ascending cache miss rate order from left to right with demarcations showing the low, mid-range, and high miss-rate groups.



Instruction cache energy consumption normalized to the best configuration in the complete design space, C. The applications are sorted in ascending cache miss rate order from left to right with demarcations showing the low, mid-range, and high cache miss rate groups



Data cache energy consumption normalized to the best configuration in the complete design space, C. The applications are sorted in ascending cache miss rate order from left to right with demarcations showing the low, mid-range, and high cache miss rate groups

Random Training Set Subset Quality Analysis

- Instruction and data cache exhibit similar energy consumption trends as random training set size increases
- Larger training set sizes do not necessarily result in higher quality subsets
- Training sets always provided energy savings, compared to base configuration

Criticality of a Priori Application Knowledge Using Domain-Specific Training Sets Analysis

- Domain-specific training applications of size three as compared to subsets created using random training sets of size three
- Increased subset energy savings by 10% and 3% for instruction and data caches, respectively
- Running applications using subsets targeted for different domains
- Increased energy consumption (degraded energy savings) by as much as 290% and 640% for instruction and data caches, respectively

Reduction in Design-Time Effort Using Domain-Specific Training Sets

- Exploring the design space using domain-specific training applications of size three is 4X faster , compared to using all anticipated applications
- Domain-specific training sets can significantly enhance subset quality and reduce design-time efforts, with only general knowledge of anticipated applications

Conclusion

- Configurable caches provide excellent solution to various domain-specific application requirements
  - However, highly configurable caches require prohibitive design space exploration time
- Reducing design space exploration time
  - Used training set applications to evaluate design space subsetting, and evaluated the subsets' energy savings using disjoint testing applications
- Subset quality
  - Random training set applications provided quality configuration subsets, and domain-specific training application increased subset quality
- 4X reduction in design space exploration time using domain-specific training applications as compared to using all anticipated applications
- Our training set methods enable designers to harness configurable cache energy savings with minimal design effort

Questions?