CPR: Composable Performance Regression for Scalable Multiprocessor Models

Benjamin C. Lee

Computer Architecture Group
Microsoft Research

Jamison Collins, Hong Wang

Microarchitecture Research Lab
Intel Corporation

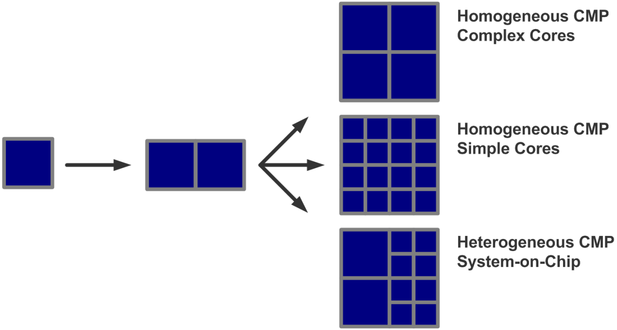
David Brooks

Engineering and Applied Sciences
Harvard University

International Symposium on Microarchitecture
11 November 2008

Technology Trends

- Moore's Law and increasing transistor densities
- Performance and power efficiency
- Transition to multi-core and parallelism

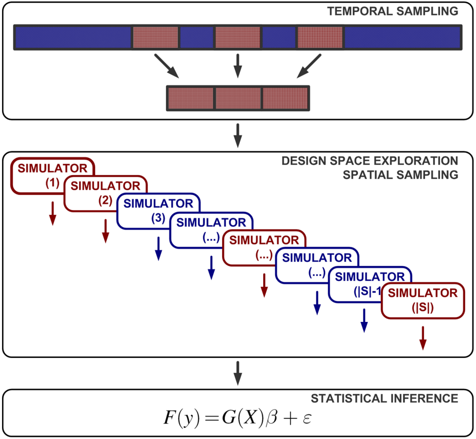


Simulation Challenges

- **Cycle-Accurate Simulation**
  - Accurately identifies trends in design space
  - Tracks instructions' progress through microprocessor
- **Simulation Costs**
  - Costs per simulation :: minutes, hours per design
  - Number of simulations :: scales exponentially (m^p)
    - p, m :: parameter count, resolution
  - Costs per simulation :: scales superlinearity (n^γ)
    - n cores, $\gamma > 1$

Statistical Inference

- Construct inferential models from samples [ASPLOS'06]
- Use models as efficient surrogates for simulator



Multiprocessor Inference

- **Expensive CMP Simulations**
  - Physical resource contention increases host cycles
  - Logical resource contention increases simulated cycles
  - Synchronization increases cost per simulated cycle
- **Composable Performance Regression**
  - Leverage core models to minimize CMP simulations
  - Core :: Uniprocessor performance
  - Contention :: Shared resource contention
  - Penalty :: Performance penalty from contention

Outline

Outline

- Motivation
  - Technology Trends
  - Simulation Challenges
  - Simulation Paradigm
- Uniprocessor
  - Regression Theory
  - Model Evaluation
  - Evolutionary Design
- Multiprocessor
  - CPR
  - Model Evaluation
  - Scalability

Regression Theory

- **Statistical Inference**
  - Models relationships between data
  - Requires initial data to train, formulate model
  - Leverages correlation from initial data for prediction
- **Regression Models**
  - Low training costs (sample 300 from 4.3B designs)
  - Accurate inference (1.5% median error)
  - Efficient computation (100's of predictions per second)

Formulation

- n simulated design samples, p design parameters
- Response :: $\mathbf{Y}$ design metrics (*e.g.*, performance)
- Predictor :: $\mathbf{X}$ design parameters (*e.g.*, ROB, cache)

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_{11} & \dots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{np} \end{bmatrix}$$

- Coefficients :: $\beta = [\beta_0, \dots, \beta_p]^T$
- Errors :: $\varepsilon = [\varepsilon_1, \dots, \varepsilon_n]^T$ where $\varepsilon_i \sim N(0, \sigma^2)$

$$\begin{aligned} \mathbf{Y} &= \mathbf{X}\beta + \varepsilon \\ F(\mathbf{Y}) &= G(\mathbf{X})\beta_G + \varepsilon \end{aligned}$$

Prediction

- **Requirements**
  - β known from least squares model training
  - $\mathbf{X}$ known for a given set of queries
- **Expected Response**
  - Response as weighted sum of predictor values
  - Computed efficiently as matrix-vector product

$$\begin{aligned} E[\mathbf{Y}] &= E[\mathbf{X}\beta + \varepsilon] \\ &= E[\mathbf{X}\beta] + E[\varepsilon] \\ &= \mathbf{X}\beta \end{aligned}$$

Experimental Methodology

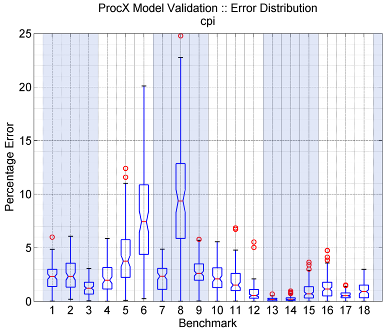
- **Intel Product Simulators**
  - Models consecutive generations of x86 μ-arch
  - Supports dual-, quad-core architectures.
- **Sampling Uniformly at Random (UAR)**
  - Parameter space includes predictors, ROB, caches
  - 15 parameters, 4.3B designs
  - Sample 300 designs for simulation
- **Statistical Framework**
  - R :: software environment for statistical computing
  - Hmisc and Design packages [Harrell]

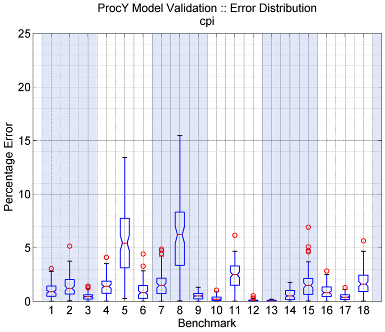
Benchmarks

Digital Home		
1	audio	audio conversion
2	video	video compression
3	photo	photoshop album
Games		
4	unreal	Unreal Tournament
5	halflife	Half-Life, modified Quake engine
6	homeworld	Homeworld, three-dimensional movement
Multimedia		
7	mentalray	rendering, ray tracing
8	painter	raster graphics package
9	tachyon	ray tracing
Office		
10	outlook	personal information manager
11	access	relational database management system
12	excel	spreadsheet application
Productivity		
13	md2	OpenSSL cryptographic hash function
14	encrypt	file encryption
15	flash	multimedia player
Server		
16	specweb	web server
17	tpcc	on-line transaction processing
18	specjapp	J2EE 1.3 application servers

Uniprocessor Model Accuracy

- Obtain 50 additional random samples for validation
- Core :: 1.5% median error



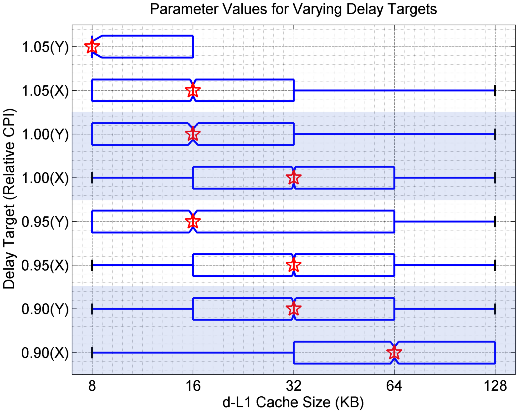


Evolutionary Design

- **Evolutionary Approach**
  - Optimize ProcX
  - Design ProcY, enhancing ProcX with μ-arch features
  - Re-construct models, accounting for μ-arch features
  - Optimize ProcY
- **Case Study**
  - Consecutive generations of x86 μ-arch
  - Improve FE (e.g., branch prediction)
  - Improve MEM (e.g., prefetching)
  - Improve OOO (e.g., memory disambiguation)

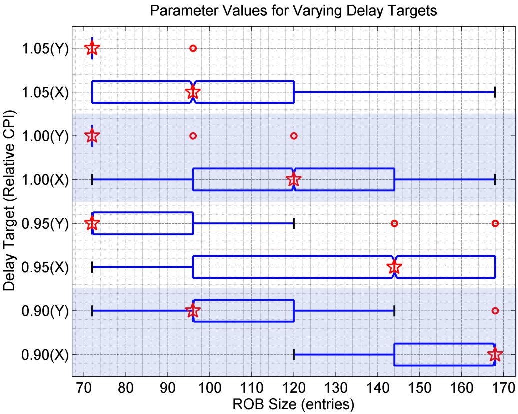
Evolving Caches

- Improve MEM: similar performance with smaller caches



Evolving ROB

- Improve FE: more instructions inflight, suggests larger ROB
- Improve MEM: fewer cache misses, suggests smaller ROB

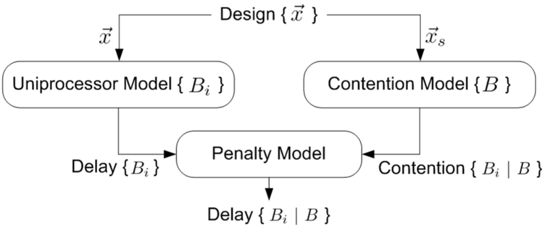


Outline

- Motivation
  - Technology Trends
  - Simulation Challenges
  - Simulation Paradigm
- Uniprocessor
  - Regression Theory
  - Model Evaluation
  - Evolutionary Design
- Multiprocessor
  - CPR
  - Model Evaluation
  - Scalability

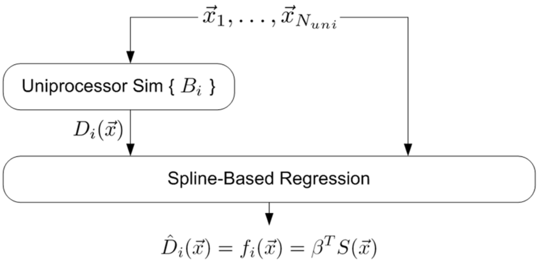
Composable Performance Regression

- CPR :: build separate core, contention, penalty models
- Requires simulations $N_{uni} > N_{con} \geq N_{pen}$
- Suppose core sims require T_1, multi-core sims require $T_1 n^{\gamma}$



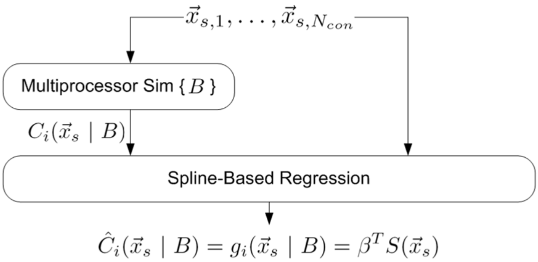
CPR: Core

- Train with uniprocessor sims from full parameter space
- Estimate per core delay from all design parameters



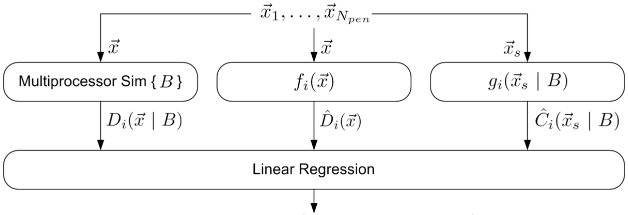
CPR: Contention

- Train with CMP, cache-only sims from reduced subspace
- Estimate cache hits/misses from shared cache parameters



CPR: Penalty

- Train with composed predictions, few CMP sims from full space
- Estimate CMP core delays from core, contention predictions



$$\begin{aligned}
\hat{D}_i(\vec{x} \mid B) &= h_i\Big(f_i(\vec{x}),\ g_i(\vec{x}_s \mid B) \mid B\Big) \\
&= \alpha f_i(\vec{x}) + \beta^T g_i(\vec{x}_s \mid B) \\
&= \alpha \hat{D}_i(\vec{x}) + \beta^T \hat{C}_i(\vec{x}_s \mid B)
\end{aligned}$$

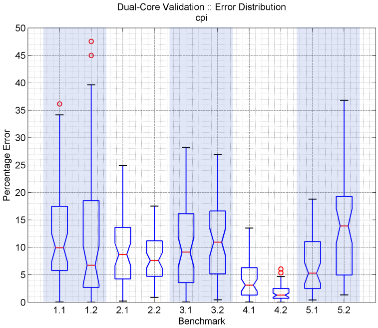
Benchmarks

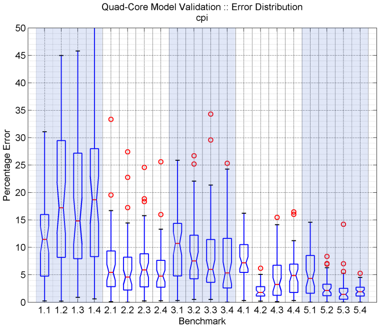
Dual-Core Benchmarks		
Set	.1	.2
1	painter	homeworld
2	access	mentalray
3	specjapp	specweb
4	homeworld	tachyon
5	dense	flash

Quad-Core Benchmarks				
Set	.1	.2	.3	.4
1	dense	excel	flash	md2
2	video	specjapp	specweb	tachyon
3	excel	homeworld	audio	unreal
4	outlook	encrypt	halflife	homeworld
5	painter	mentalray	outlook	encrypt

Multiprocessor Model Accuracy

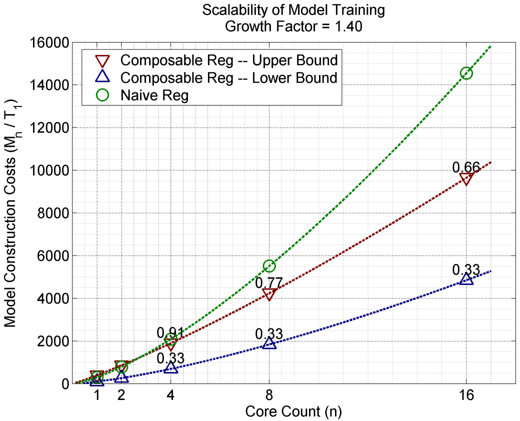
- Dual-core :: 6.6% median error
- Quad-core :: 4.8% median error





Scaling Trends

- Lower bound CPR costs 0.33x of naïve costs
- Approach lower bound as uniprocessor models built



Conclusion

- **Inference in Industry**
  - Effective inference for x86 μ-arch
  - 1.5% median errors relative to simulation
  - Evolutionary design for new features across generations
- **Composable Performance Regression**
  - Leverage core models to minimize CMP simulations
  - Construct separate core, contention, penalty models
  - 4.8 to 6.6% median errors for dual-, quad-core
  - 0.33x training costs of prior approaches

Future Directions

- **Efficient Multiprogramming Analysis**
  - Evaluate combinations without modeling every combination
- **Multi-Threaded Workloads**
  - Extend for homogeneous, heterogeneous threads.
  - Account for synchronization events
- **Many-Core Architectures**
  - Construct models without many-core simulators
  - Consider other shared resources (e.g., network)

CPR: Composable Performance Regression for Scalable Multiprocessor Models

Benjamin C. Lee

Computer Architecture Group
Microsoft Research

Jamison Collins, Hong Wang

Microarchitecture Research Lab
Intel Corporation

David Brooks

Engineering and Applied Sciences
Harvard University

International Symposium on Microarchitecture
11 November 2008