Analyzing the Processor Bottlenecks in SPEC CPU 2000

Joshua Yi (Freescale Semiconductor Inc.)
Ajay Joshi (Univ. of Texas)
Resit Sendag (Univ. of Rhode Island)
Lieven Eeckhout (Ghent Univ.)
David Lilja (Univ. of Minnesota)

SPEC Benchmarking Workshop

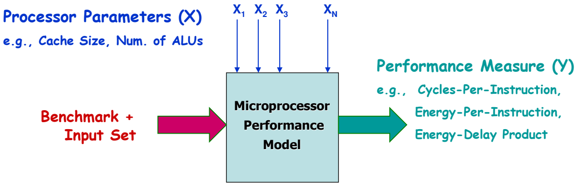
January 23, 2006

Presentation Overview

- **Bottleneck Characterization**
- **Plackett & Burman Design**
- **Performance / Power Bottlenecks**
- **Benchmark Classification**
- **Summary**

Bottleneck Characterization

- Rank processor parameters (X) based on their effect on Y



- Statistical Techniques for Ranking Parameters
  - ANOVA - Captures All Interactions - But 2^N Test Cases
  - One-at-a-Time - N Test Cases - But Only Single Parameter Effects

Plackett & Burman (P&B) Design

- Efficient screening design to quantify significance of parameters
- Vary values of X parameters simultaneously over 2N test cases (N is next multiple of 4 greater than X)
- Possible values of parameters

  +1 : Higher than normal value (e.g., Num. of ALUs = 8)

  -1 : Lower than normal value (e.g., Num of ALUs = 1)
- Amount of Information
  - All single parameter effects (X_1, X_2 … X_N)
  - Two parameter interactions (X_1X_2, X_1X_3 ….)

Plackett & Burman Mechanics

+1: High Value for Parameter
e.g., Number of Integer ALUs = 8

1st Row From PB Paper

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	Execution Time
1	+1	+1	+1	-1	+1	-1	-1	9
2	-1	+1	+1	+1	-1	+1	-1	11
3	-1	-1	+1	+1	+1	-1	+1	20
4	+1	-1	-1	+1	+1	+1	-1	1
5	-1	+1	-1	-1	+1	+1	+1	1
6	+1	-1	+1	-1	-1	+1	+1	9
7	+1	+1	-1	+1	-1	-1	+1	19
8	-1	-1	-1	-1	-1	-1	-1	74
Effect	-68	-64	-46	-42	-82	-100	-46	

-1: Low Value for Parameter
e.g., Number of Integer ALUs = 1

-1*9+1*11-1*20+1*1+1*1 …-1*74= -100
Most Significant Parameter

Finding Significant Bottlenecks

1. Execute Plackett and Burman Design
   - Run Simulations
   - Calculate Effect of All Parameters
2. For Each Benchmark
   - Sort Parameters in Descending Order
   - Rank the Parameters (1=Most Important)
3. Across Benchmarks, Average the Ranks
4. Lowest Ranked Parameters are the Most Significant

X_1 = 100 → 5
X_2 = 200 → 1
X_3 = 150 → 3
X_4 = 120 → 4
X_5 = 175 → 2

X_1	5	4	5	4.7
X_2	1	3	2	2.0
X_3	3	2	1	2.0
X_4	4	1	2	2.3
X_5	2	5	4	3.7

Experiment Framework

Plackett and Burman Design

- 43 parameters (processor core and memory core) of a superscalar microprocessor
- 88 (very) different processor configurations

Simulation Environment

- SimpleScalar Simulator
- sim-outorder performance model

Benchmarks

- SPEC CPU2000 benchmark 46 program-input pairs (ref)
- Alpha Binaries compiled at -O3

P&B High/Low Values – Processor Core

Parameter	Low Value	High Value
Fetch Queue Entries	4	32
Branch Predictor	2-Level	Perfect
Branch MPred Penalty	10 Cycles	2 Cycles
RAS Entries	4	64
BTB Entries	16	512
BTB Assoc	2-Way	Fully-Assoc
Spec Branch Update	In Commit	In Decode
Decode/Issue Width	4-Way	
ROB Entries	8	64
LSQ Entries	0.25 * ROB	1.0 * ROB
Memory Ports	1	4

P&B High/Low Values – Functional Units

Parameter	Low Value	High Value
Int ALUs	1	4
Int ALU Latency	2 Cycles	1 Cycle
Int ALU Throughput	1	
FP ALUs	1	4
FP ALU Latency	5 Cycles	1 Cycle
FP ALU Throughputs	1	
Int Mult/Div Units	1	4
Int Mult Latency	15 Cycles	2 Cycles
Int Div Latency	80 Cycles	10 Cycles
Int Mult Throughput	1	
Int Div Throughput	Equal to Int Div Latency	
FP Mult/Div Units	1	4
FP Mult Latency	5 Cycles	2 Cycles
FP Div Latency	35 Cycles	10 Cycles
FP Sqrt Latency	35 Cycles	15 Cycles
FP Mult Throughput	Equal to FP Mult Latency	
FP Div Throughput	Equal to FP Div Latency	
FP Sqrt Throughput	Equal to FP Sqrt Latency	

P&B High/Low Values – Memory System (1)

Parameter	Low Value	High Value
L1 I-Cache Size	4 KB	128 KB
L1 I-Cache Assoc	1-Way	8-Way
L1 I-Cache Block Size	16 Bytes	64 Bytes
L1 I-Cache Repl Policy	Least Recently Used	
L1 I-Cache Latency	4 Cycles	1 Cycle
L1 D-Cache Size	4 KB	128 KB
L1 D-Cache Assoc	1-Way	8-Way
L1 D-Cache Block Size	16 Bytes	64 Bytes
L1 D-Cache Repl Policy	Least Recently Used	
L1 D-Cache Latency	4 Cycles	1 Cycle
L2 Cache Size	256 KB	8192 KB
L2 Cache Assoc	1-Way	8-Way
L2 Cache Block Size	64 Bytes	256 Bytes

P&B High/Low Values – Memory System (2)

Parameter	Low Value	High Value
L2 Cache Repl Policy	Least Recently Used	
L2 Cache Latency	20 Cycles	5 Cycles
Mem Latency, First	200 Cycles	50 Cycles
Mem Latency, Next	0.02 * Mem Latency, First	
Mem Bandwidth	4 Bytes	32 Bytes
I-TLB Size	32 Entries	256 Entries
I-TLB Page Size	4 KB	4096 KB
I-TLB Assoc	2-Way	Fully Assoc
I-TLB Latency	80 Cycles	30 Cycles
D-TLB Size	32 Entries	256 Entries
D-TLB Page Size	Same as I-TLB Page Size	
D-TLB Assoc	2-Way	Fully-Assoc
D-TLB Latency	Same as I-TLB Latency	
Memory Ports	1	4

Most Significant Performance Bottlenecks

Rank	Parameter	*gzip (graphic)*	*gcc (200)*	*mcf*	*equake*
1	ROB Entries	1	2	3	2
2	L2 Cache Size	11	1	1	8
3	Memory Latency First	13	3	2	1
4	L2 Cache Latency	7	4	5	5
5	Branch Predictor Accuracy	2	5	8	11
6	L1 I-Cache Size	17	8	16	42
7	Number of Integer ALUs	3	6	9	37
8	Load Store Queue Entries	5	13	7	6
9	L1 D-Cache Latency	4	7	22	12
10	L1 I-Cache Block Size	29	10	34	34
11	Memory Bandwidth	23	11	4	4
12	L1 D-Cache Size	12	35	33	14

Most Significant Power Bottlenecks

Rank	Parameter	*gzip (graphic)*	*gcc (200)*	*mcf*	*equake*
1	**BTB Associativity**	3	1	3	2
2	**BTB Entries**	2	2	4	3
3	**Branch Predictor Accuracy**	1	3	11	12
4	**Memory Latency First**	28	6	1	1
5	**L2 Cache Latency**	13	4	6	11
6	**L1 I-Cache Size**	4	8	10	10
7	**L2 Cache Size**	5	39	2	8
8	**ROB Entries**	16	19	7	4
9	**L1 D-Cache Size**	7	5	8	6
10	**L1 D-Cache Block Size**	23	7	19	9
11	**Memory Bandwidth**	25	12	5	7
12	**Number of Integer ALUs**	6	13	29	21

Similarity Between Benchmarks

P&B Bottleneck Characterization for each benchmark-input set

e.g., Vector of 43 ranks for each Benchmark
< 1, 22, 41, 5, 3 ……… >

↓

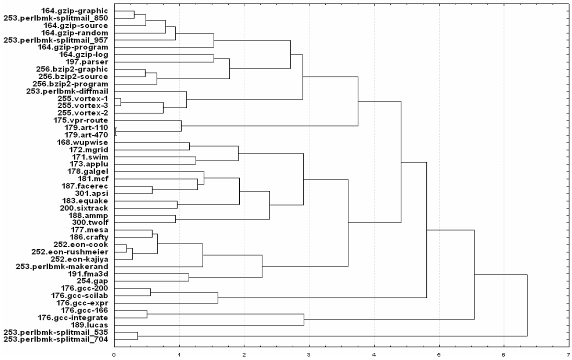
Remove Correlation & Reduce Dimensions using Principal Component Analysis

↓

Apply Clustering Algorithm
(e.g., K-means, Hierarchical) to group programs

Classification Intuition:
Similar Effect → Similar Significant Parameters → Similar Bottlenecks

Classification Across All Bottlenecks



Processor Core Bottlenecks

Cluster	Benchmarks
1	gcc-expr, gcc-200, **gcc-scilab**
2	gzip-graphic, gzip-program, gzip-random, **gzip-source**
3	**eon-cook**, eon-kajiya, eon-rushmeier, crafty
4	galgel, equake, facerec, fma3d, sixtrack perlbmk-makerand, perlbmk-splitmail_850, **perlbmk-splitmail_957**, gap, bzip2-graphic, bzip2-program, bzip2-source, twolf, apsi
5	**wupwise**
6	mcf, ammp, perlbmk-splitmail_535, perlbmk-splitmail_704, **vortex-1**, vortex-3,
7	**gcc-166**, gcc-integrate
8	**lucas**
9	swim, **mgrid**, applu
10	gzip-log, **parser**
11	**vpr-route**, mesa, art-110, art-470, perlbmk-diffmail, vortex-2

Data Memory Bottlenecks

Cluster	Benchmarks
1	**gcc-166**, gcc-integrate, lucas
2	vpr-route, galgel, **facerec**, equake, parser, bzip2-graphic, bzip2-program, bzip2-source, apsi
3	art-110, art-470, **mcf**, ammp, twolf
4	wupwise, swim, mgrid, **applu**
5	mesa, crafty, fma3d, eon-cook, eon-kajiya, eon-rushmeier, perlbmk-diffmail, perlbmk-makerand, gap, vortex-1, **vortex-2**, vortex-3
6	gcc-200, gcc-expr, **gcc-scilab**
7	gzip-graphic, gzip-log, gzip-program, gzip-random, **gzip-source**, sixtrack, perlbmk-splitmail_850, perlbmk-splitmail_957
8	**perlbmk-splitmail_535**, perlbmk-splitmail_704

Instruction Memory Bottlenecks

Cluster	Benchmarks
1	gzip-graphic, gzip-log, gzip-random, **gzip-source**, art-110, art-470, facerec, ammp, parser, bzip2-graphic, bzip2-program, bzip2-source
2	mesa, crafty, fma3d, eon-cook, eon-kajiya, eon-rushmeier, **perlbmk-makerand**,
3	**vpr-route**, galgel, perlbmk-splitmail_535, perlbmk-splitmail_704
4	applu, **gcc-166**, gcc-integrate, lucas
5	perlbmk-diffmail, **vortex-1**, vortex-2, vortex-3
6	**wupwise**, swim, mgrid, gcc-200, gcc-expr, gcc-scilab
7	gzip-program, **perlbmk-splitmail_850**, mcf, equake, sixtrack, perlbmk-splitmail_957, twolf, apsi

Control Flow Bottlenecks

Cluster	Benchmarks
1	gzip-log, **parser**
2	gzip-graphic, gzip-program, **gzip-random**, gzip-source, perlbmk-splitmail_535, perlbmk-splitmail_704, perlbmk-splitmail_850, perlbmk-splitmail_957, gap, vortex-1, vortex-2, vortex-3, bzip2-graphic, bzip2-program, bzip2-source
3	mesa, equake, **crafty**, facerec, sixtrack, eon-cook, eon-kajiya, eon-rushmeier, perlbmk-makerand
4	swim, galgel, art-110, **art-470**, mcf, ammp, fma3d, apsi
5	wupwise, **vpr-route**, twolf
6	mgrid, applu, **gcc-166**, gcc-integrate
7	gcc-200, gcc-expr, **gcc-scilab**
8	**lucas**

Classification Across All Bottlenecks

Cluster	Benchmarks
1	mesa, crafty, eon-cook, **eon-kajiya**, eon-rushmeier, perlbmk-makerand
2	**perlbmk-splitmail_535**, perlbmk-splitmail_704
3	perlbmk-diffmail, **vortex-1**, vortex-2, vortex-3
4	**wupwise**, swim, mgrid, equake, fma3d, sixtrack, gap
5	applu, **gcc-166**, gcc-integrate
6	gzip-graphic, gzip-program, gzip-random, **gzip-source**, perlbmk-splitmail_850, perlbmk-splitmail_957
7	gcc-200, gcc-expr, **gcc-scilab**
8	gzip-log, parser, bzip2-graphic, **bzip2-program**, bzip2-source
9	mcf, facerec, **ammp**, twolf, apsi
10	vpr-route, galgel, art-110, **art-470**
11	**lucas**

Summary

- **Plackett & Burman bottleneck characterization**
  - Computer Architect – Understand Bottlenecks
  - Benchmark Designer – Similarity & Diversity
- **Bottleneck Characterization of SPEC CPU2000**
  - ROB entries, L2 cache size, and L1 I-cache size, Memory Latency are key bottlenecks
  - Overall power and performance bottlenecks are similar (Except BTB entries)
  - Bottlenecks for gzip, gcc, and perlbmk depend on input-set
  - lucas has most unique bottleneck characteristics