Session 2: Tracing and Characterization

Optimizing UNIX for OLTP on CC-NUMA
Darrell Suggs
Data General Corporation

Tracing and Characterization of NT-based System Workloads
Jason Casmira, David Kaeli - Northeastern University
David Hunter - DEC Software Partners Engineering Group

Analysis of Commercial and Technical Workloads on AlphaServer Platforms
Zarka Cvetanovic
Digital Equipment Corporation

Characterizing TPC-D on a MIPS R10K Architecture
Qiang Cao, Pedro Trancoso, and Josep Torrellas
University of Illinois at Urbana Champaign

Optimizing UNIX for OLTP on CC-NUMA

Darrell Suggs, PhD
Performance Architect
Data General Corp.

Overview

- In late ‘96 our challenge was ...
- Tune software today for “future” architecture
  - no reasonable prototypes available
  - software development lead time is significant
  - design issues very complex, exceed our intuitive abilities
  -
- Specific target
  - Architecture: 16-32 Intel Pentium Pro, CC-NUMA
  - Operating System: DG/UX, commercial/enterprise UNIX
  - Application: Oracle RDBMS
  - Workload: TPC-C

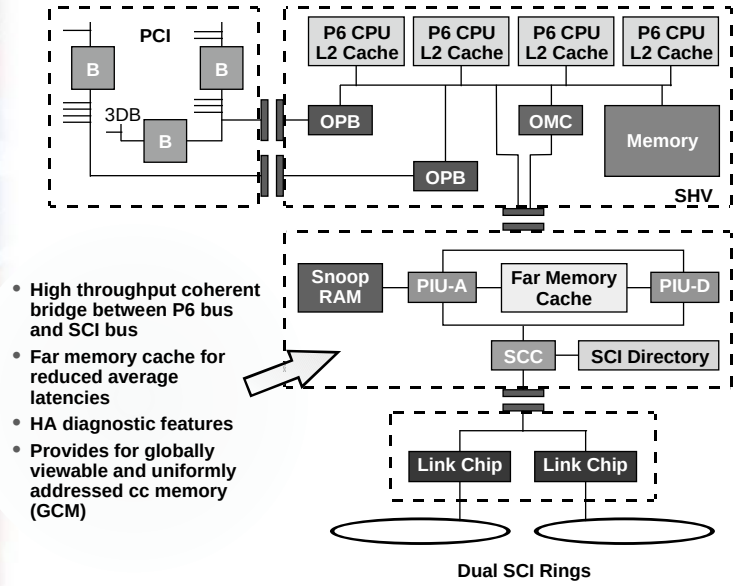
Product Status

- AV20000 Product Shipped for Revenue in ‘97
- Demonstrated industry leading performance
- First in a line of CC-NUMA products

Basic Approach to SW Scaling

- Construct advanced analysis environment
  - Obtain architecture independent traces
  - Construct detailed cache simulation of target platform
  - Simulate with model/traces looking for SW scaling issues
- Use analysis environment to:
  - Prototype changes in OS and App
  - Re-trace prototype software, verify increased scaling
  - Work with OS/APP developers to implement changes
- Repeat until no more high leverage scaling issues found

NUMA Building Block Architecture



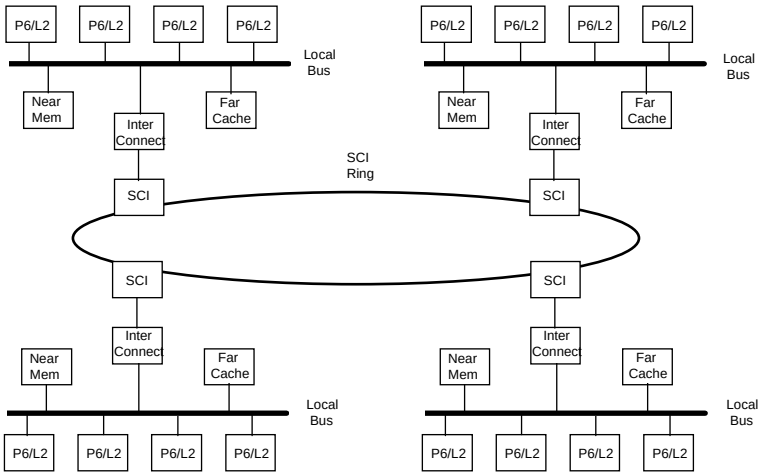
- High throughput coherent bridge between P6 bus and SCI bus
- Far memory cache for reduced average latencies
- HA diagnostic features
- Provides for globally viewable and uniformly addressed cc memory (GCM)

PGS970925-6

CC-NUMA Architecture

- Platform Characteristics
  - 16 Intel/Ppro with 1MB L2 Cache
  - Full service local memory controller (OMC)
  - Far Memory Cache controller, 128MB Direct mapped (OMC like)
  - Distributed coherent memory -- single image
  - SCI directory based cache coherency at interconnect
  - Local access latency: ~300ns
  - Remote access latency: ~3 to 5 microsecs
  -
- Key scaling issue
  - Number of interconnect operations per unit of work
  - Interconnect operation demand per second

CC-NUMA Architecture



CC-NUMA Architecture Simulation

- Construct detailed discrete event simulation
  - Models all system cache contents and protocols
  - Models all busses/interconnects and associated protocols
    - E.g. full simulation of SCI protocol
  - Specifically,
    - 16 L2's, 4 system busses/far memory caches, 4 SCI directories7
- Model driven by physical, pre-L2, address traces
  - flexibility to change all cache geometries (except L1)
  - can examine impact of various protocol optimizations
- Simulation Tool - SES Workbench
  - Scientific and Engineering Software
  - Mature and flexible tool for commercial grade simulation

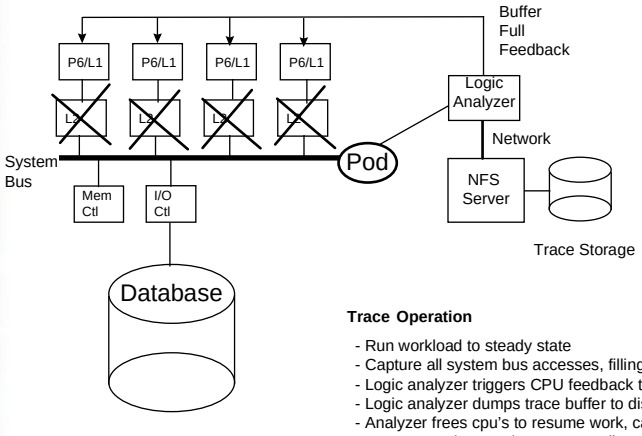
Architecture Independent Traces

- Objective: Capture traces on existing HW to be
  - extensible to different architectures (diff L2's, bus structs, cpu counts, etc.)
  - physical addresses for both user and kernel
  - long, contiguous traces for large cache simulation & continuity
  - representative, but manageable, sample (30 to 60 secs)
    - pre-L2, post-L1

Architecture Independent Traces

- Technique Overview
  - Use largest available SMP (quad P6)
  - Start with well balanced OLTP configuration (TPC-C)
  - Trace all processes executing on SMP
    - annotate traces to identify individual PID's
  - Process traces to identify independent process address streams
  - Simulate HW by assigning processes to simulated CPU's

Trace Environment



Trace Operation

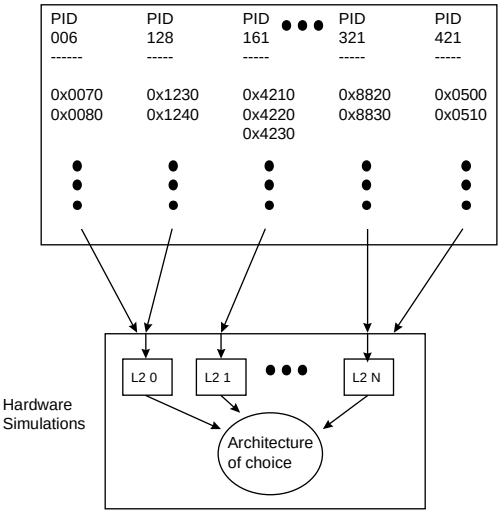
- Run workload to steady state
- Capture all system bus accesses, filling analyzer buffer
- Logic analyzer triggers CPU feedback to "halt" cpu
- Logic analyzer dumps trace buffer to disk, no cpu activity occurs
- Analyzer frees cpu's to resume work, captures addresses til full
- Repeat start/capture/stop repeatedly
- Results in long, contiguous traces. 30 to 60 system secs.
  Hundreds of millions of accesses captured.

Process Simulation

Traced Data

CPU	PID	Address
0	128	0x1230
0	128	0x1240
1	321	0x8820
3	161	0x4210
3	161	0x4220
2	421	0x0500
1	321	0x8830
1	006	0x0070
2	421	0x0510
3	161	0x4230
1	006	0x0080

Post Processed Data



Architecture Independent Traces

- Issues with trace technique
  - Post-L1 data is filtered
    - pre-L1 data is too dense to handle 30 second sample (100's of GB)
    - compensate for L1 filter by flushing L1's on context switch
    - capture all addresses accessed, not every access to each address
  - Increased process count for number of CPU's
    - overloaded scheduler has high context switch rate
    - compensate by configuring "run to block" scheduling
  - I/O Service times skewed due to start/stop
  - Start/stop perturbs environment
    - minimal impact on sequence of physical addresses per process

CC-NUMA Software Scaling Issues

- Motivating Issues
  - Major HW issue: high interconnect latency
  - Major SW issue: long access time for shared data
  - Key scaling leverage: ** Interconnect operations **
  - Basic NUMA optimizations were already applied
- Classes of shared data
  - True sharing: locks, write shared data
  - False sharing: write shared data on cache line with read-only data
  - "Partner data": data should be on same cache line
    - e.g. a lock structure and the data that it guards
- Approach
  - Find & fix all high frequency false sharing/partner data
  - Develop algorithmic changes to minimize true sharing

Interconnect Operation Trends

- Intial SW interconnect ops
  - 15,000/TPC-C (new order)
- Reduced (via simulation/analysis/prototype) to
  - 6,700/TPC-C (as measured via simulation)
- Actual system measurement
  - 6,600/TPC-C (with prototype changes productized)
  -
- System performance improvement
  - 35% increase in TPM
-
- Areas where performance problems persisted:
  - I/O device drivers, controllers, etc
  - The main area ignored in simulation

Additional Benefits of Techniques

- Simulation/analysis feedback to HW design
  - cache geometries
  - protocol optimizations
  - HW buffers and other low-level resource tuning
  -
- Framework for studying advanced architecture design
  - Supporting coarse grain block-diagram tradeoffs
  - Early positioning of product performance
  - Understanding other OS/SW issues with CC-NUMA and high processor count SMP

Tracing and Characterization of NT-based System Workloads

J. Casmira, D. Kaeli Northeastern University

D. Hunter Digital Equipment Corp.

Outline

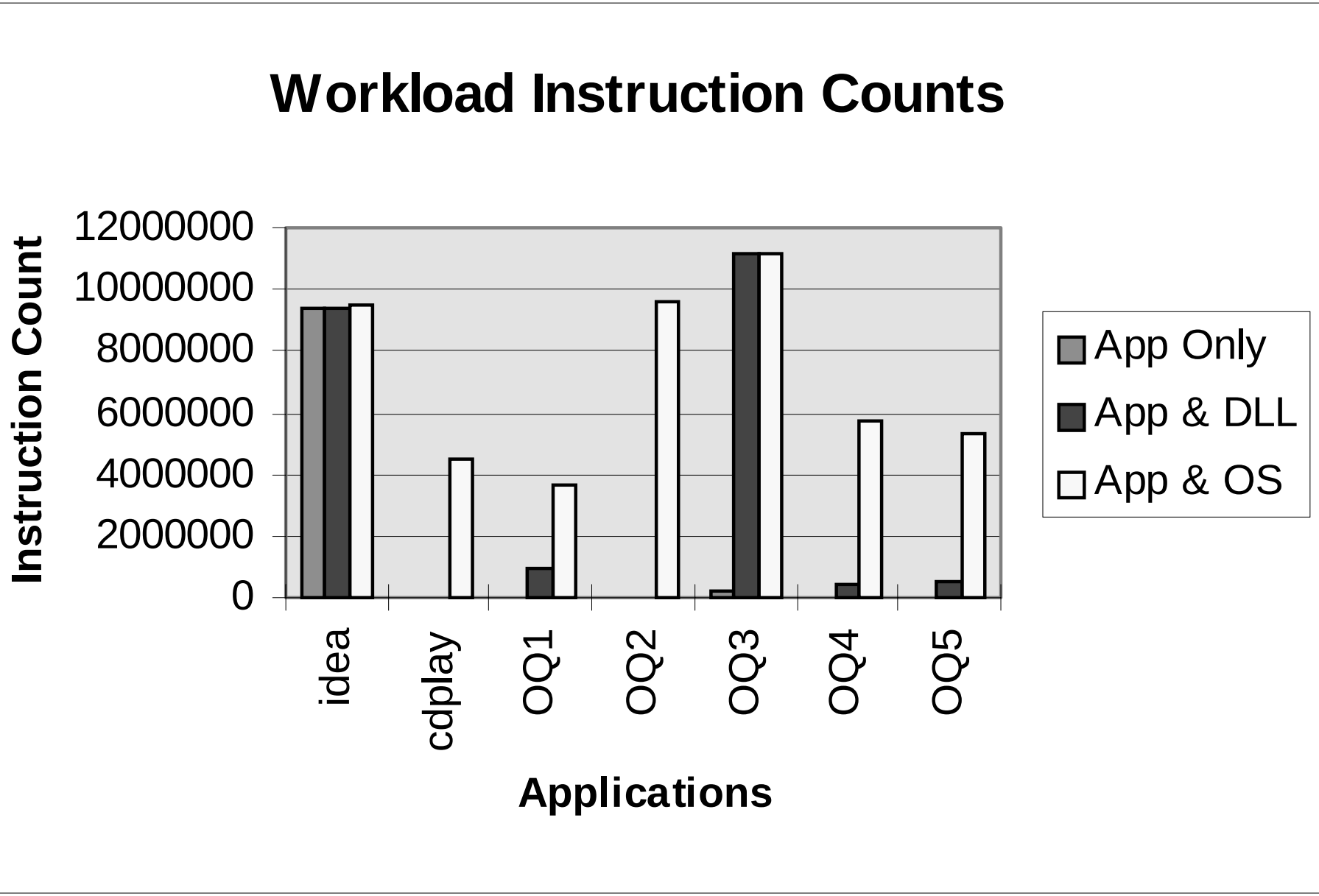
- Overview
- Workloads
- Results
- Conclusions

Overview

- Issues with trace-driven simulation
  - results only as good as input trace (GIGO)
  - typically only capture application behavior
- Existing trace tools
  - Shade
  - ATOM
  - SimOS

Current Technology

- Trace driven studies using OS-rich traces
  - ISCA96 27% ; ISCA97 16%
  - HPCA96 0% ; HPCA97 8%


Workload Instruction Counts
Instruction Count
12000000
10000000
8000000
6000000
4000000
2000000
0
idea
cdplay
OQ1
OQ2
OQ3
OQ4
OQ5
Applications
App Only
App & DLL
App & OS

What is PatchWrx?

- Dynamic Execution Tracing Tool Suite
  - system instrumentation
  - trace capture
  - stream reconstruct
- DEC Alpha 21064 Windows NT platforms
- Low overhead with minimum slowdown
  - 2X when instrumented; 4X while tracing

How Does PWX Work?

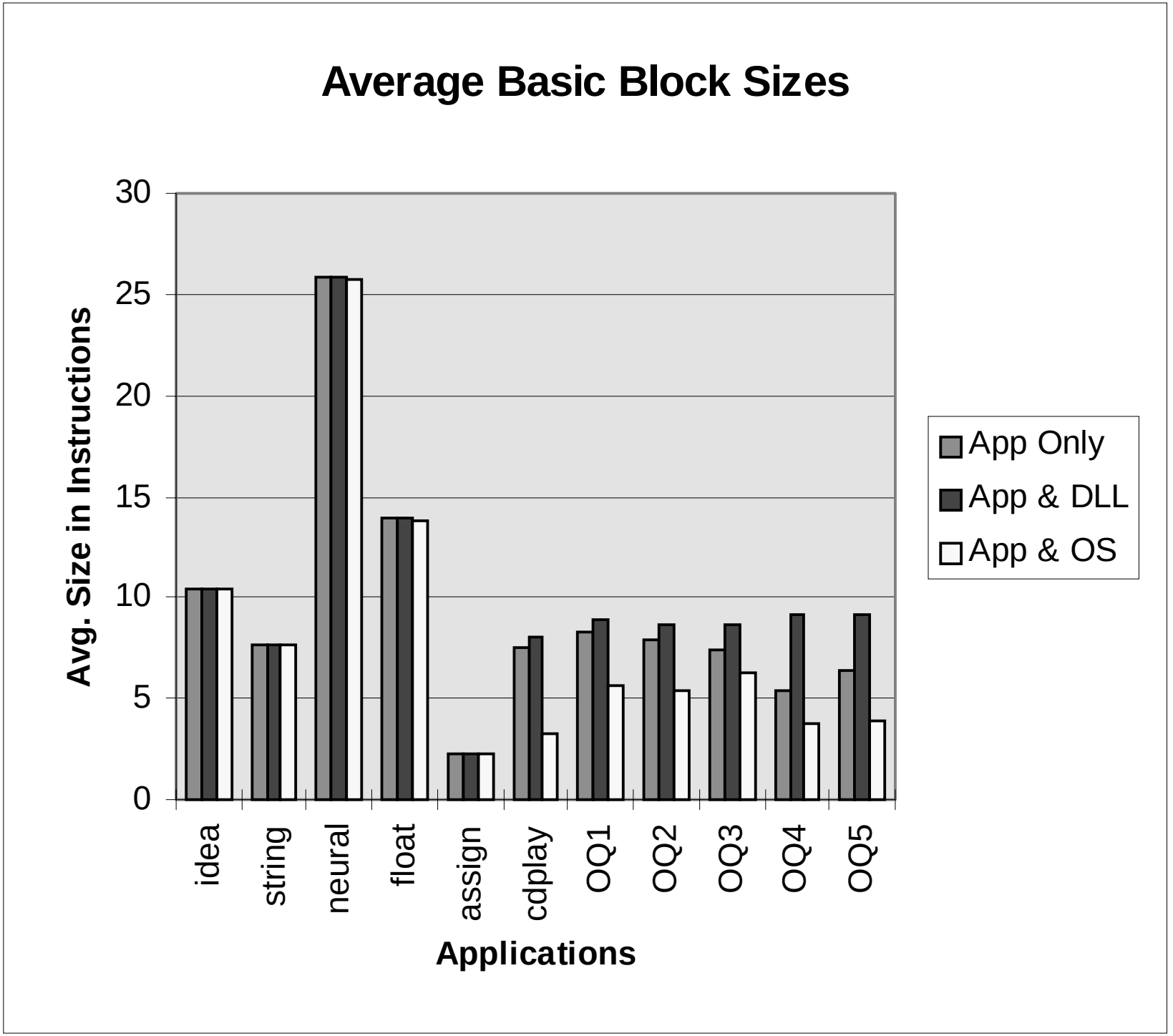
- Instrument NT binary images
- Using DEC Alpha PALcalls
  - reserve trace buffer at boot time
  - log branch instruction trace entries
- Using instrumented images & trace log, reconstruct original stream

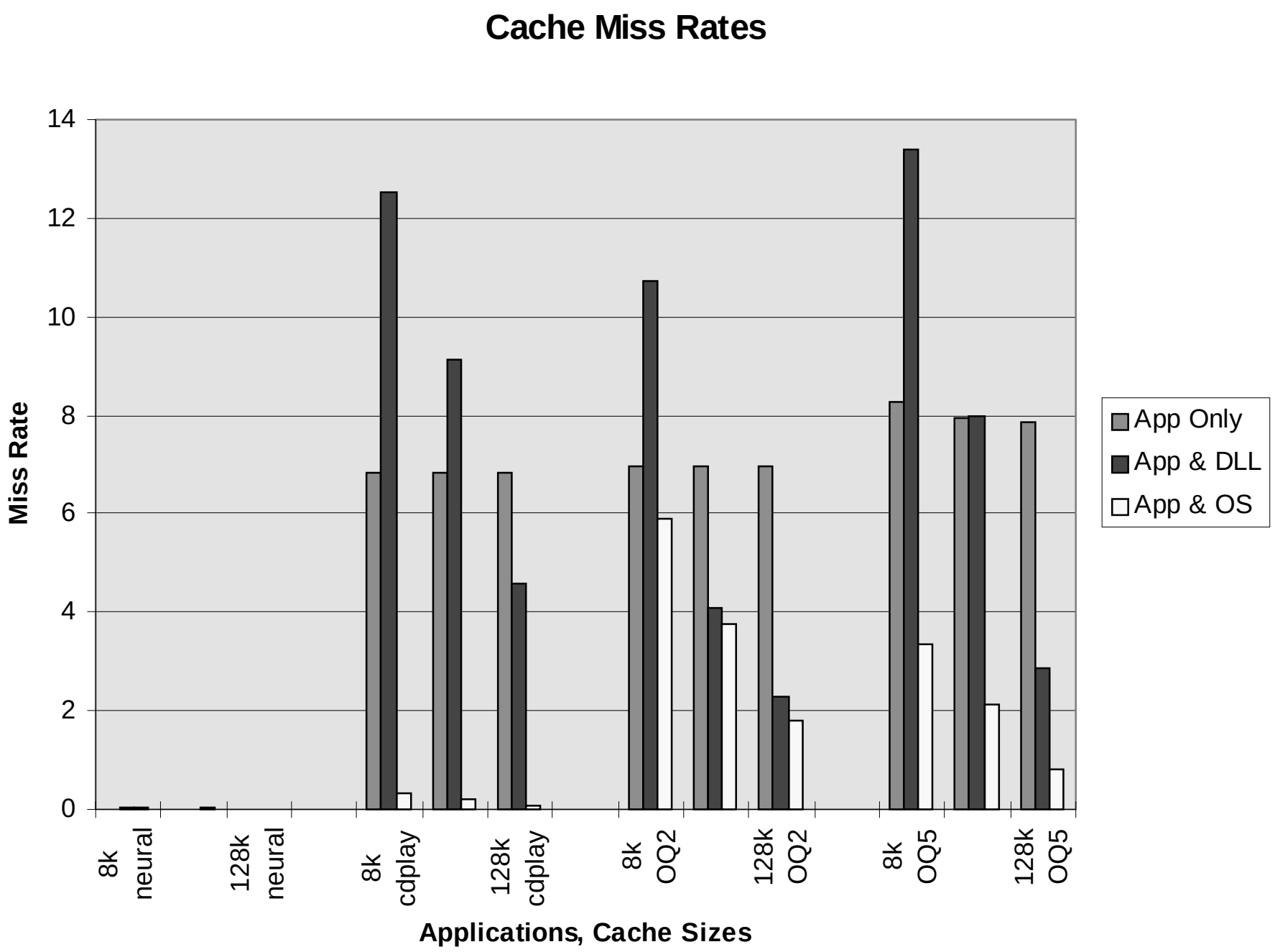
Workloads

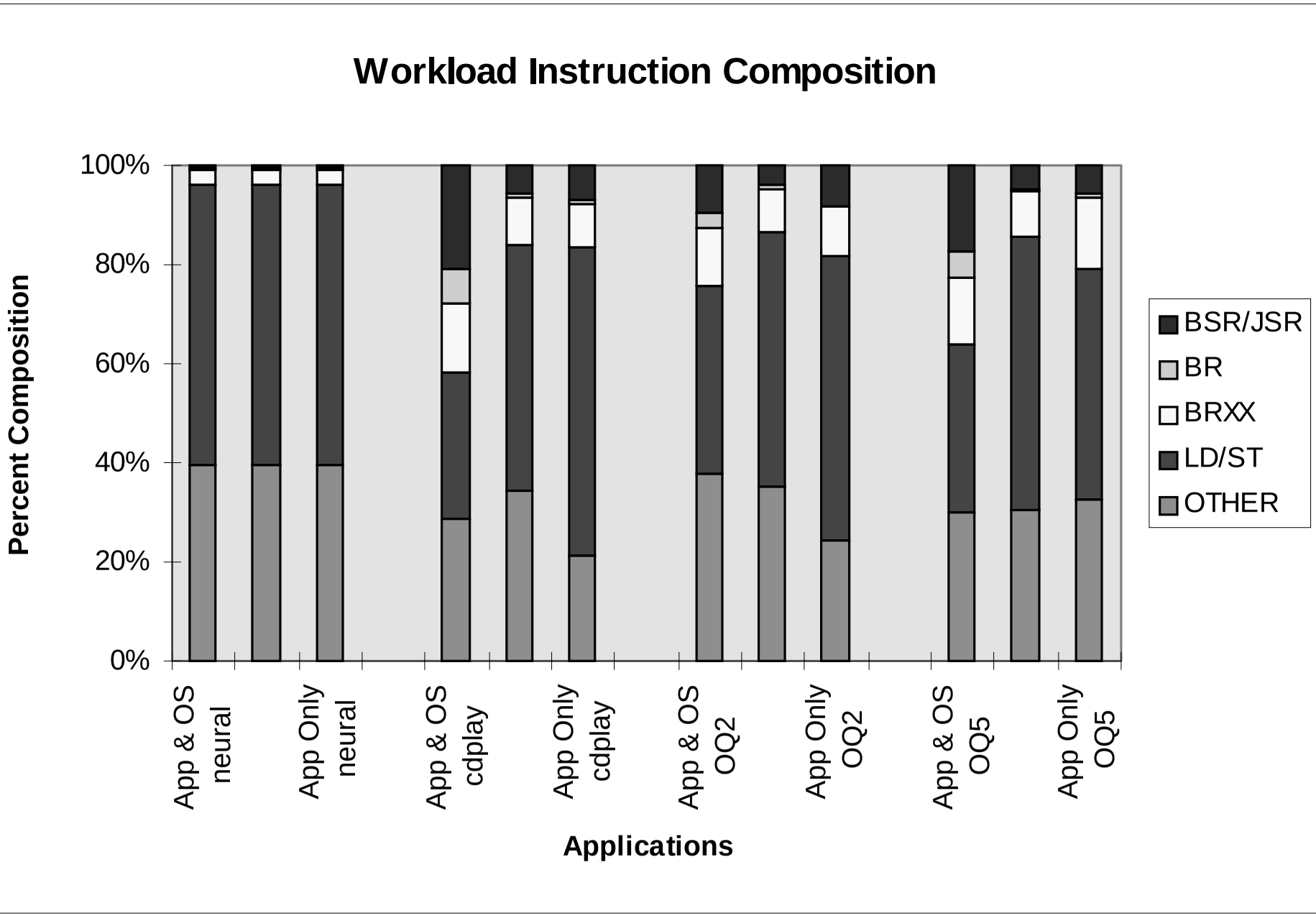
- BYTEmark benchmarks
  - typical “industry standard” benchmark
- MS Internet Explorer
  - web-browser application
- MS CD Player
  - NT packaged utility/application
- Oracle 7.3
  - 3rd party NT database

Characteristics

- Instruction Counts and Basic Block Sizes
- Instruction cache performance
- Instruction mix
- Application only
- Application and DLLs
- Application, DLLs, and OS


Average Basic Block Sizes
Avg. Size in Instructions
30
25
20
15
10
5
0
idea
string
neural
float
assign
cdplay
OQ1
OQ2
OQ3
OQ4
OQ5
Applications
App Only
App & DLL
App & OS


Cache Miss Rates
Miss Rate
14
12
10
8
6
4
2
0
8k neural
128k neural
8k cdplay
128k cdplay
8k OQ2
128k OQ2
8k OQ5
128k OQ5
Applications, Cache Sizes
App Only
App & DLL
App & OS

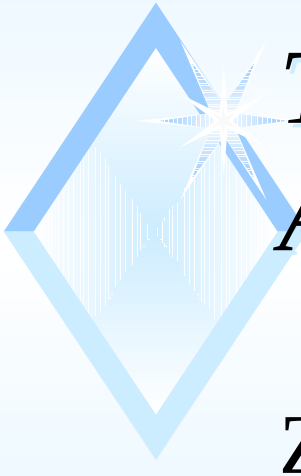

Workload Instruction Composition
Percent Composition
100%
80%
60%
40%
20%
0%
App & OS neural
App Only neural
App & OS cdplay
App Only cdplay
App & OS OQ2
App Only OQ2
App & OS OQ5
App Only OQ5
Applications
BSR/JSR
BR
BRXX
LD/ST
OTHER

Summary

- OS can dominate execution in commercial applications
- OS reduces the average basic block length
- OS can dramatically change the cache behavior
- OS can significantly alter the instruction mix
- OS must be included in trace-driven simulations to provide an accurate picture of application execution

Future Work

- Full D-Stream Reconstruction
- FX!32
- Multiprocessor Traces
- Microsoft Windows NT 5.0
- DEC Alpha 21164



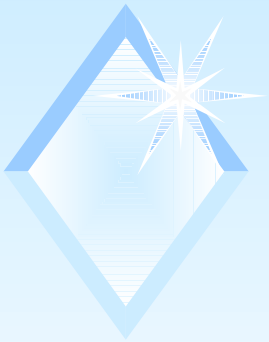
Analysis of Commercial and Technical Workloads on AlphaServer Platforms

Zarka Cvetanovic

Digital Equipment Corporation

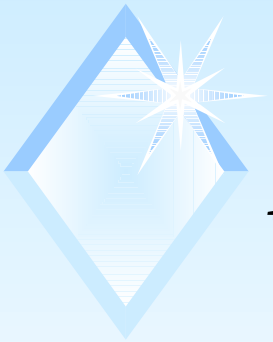
February 1, 1998

digital



Goals

- highlight differences between commercial and technical workloads on AlphaServers
- identify architectural components that are important for commercial performance

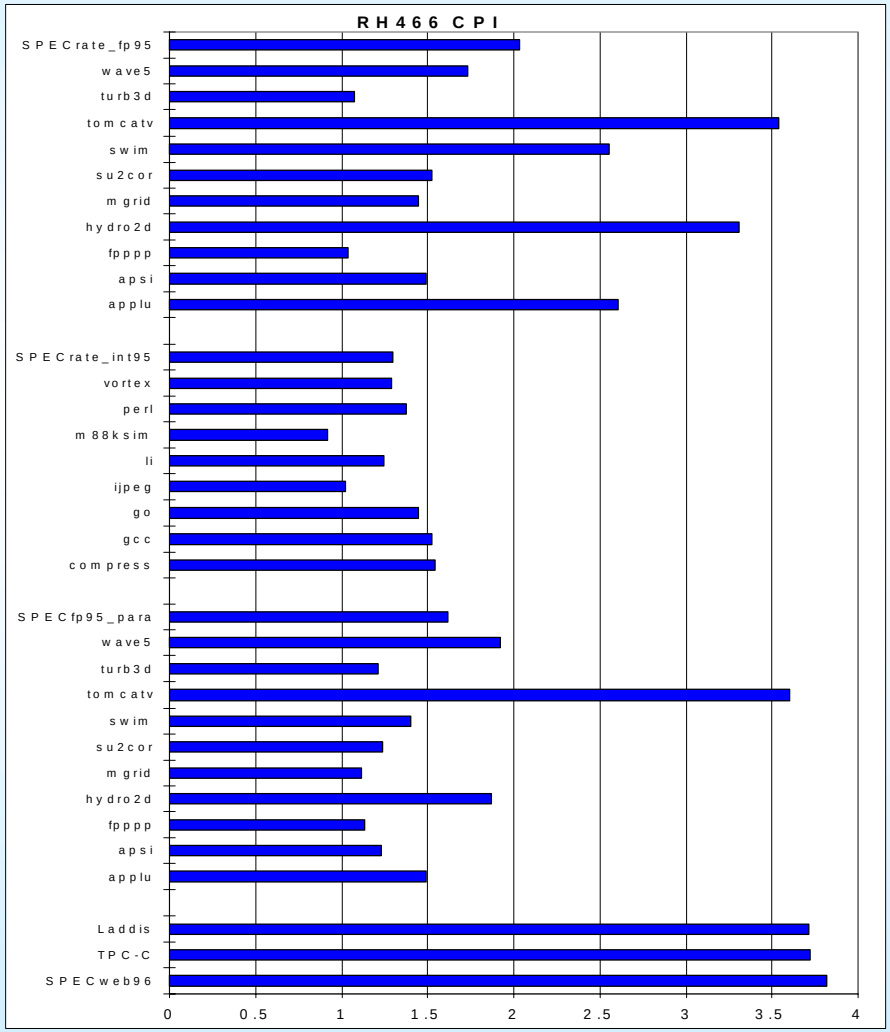


Introduction

- **systems:** AlphaServer 4100, 8400
- **tools:** CPU/platform performance counters
- **workloads:**
  - ***commercial:*** TPC-C, SPECweb96, Laddis
  - ***technical:*** SPEC95 (rates, parallel), NAS Parallel, Streams

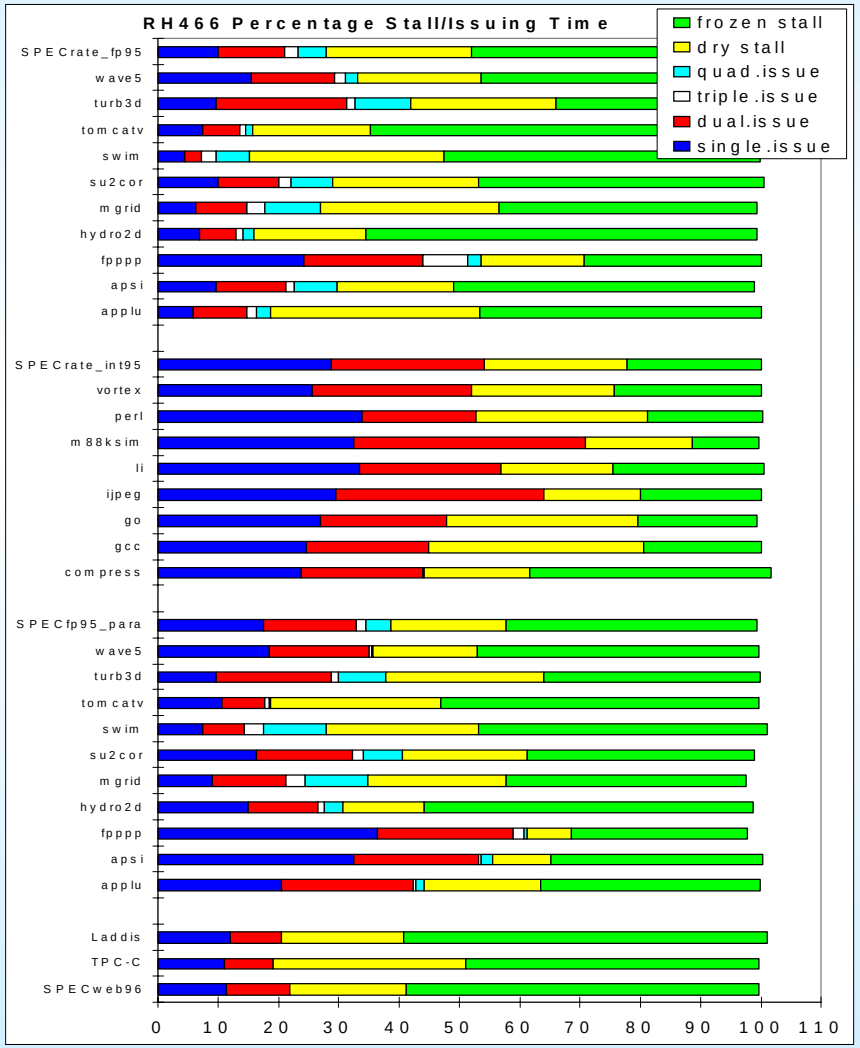
Cycles Per Instruction (CPI)

- CPI higher in commercial than the majority of technical
- several technical (tomcatv, hydro2d) have as high CPI as commercial



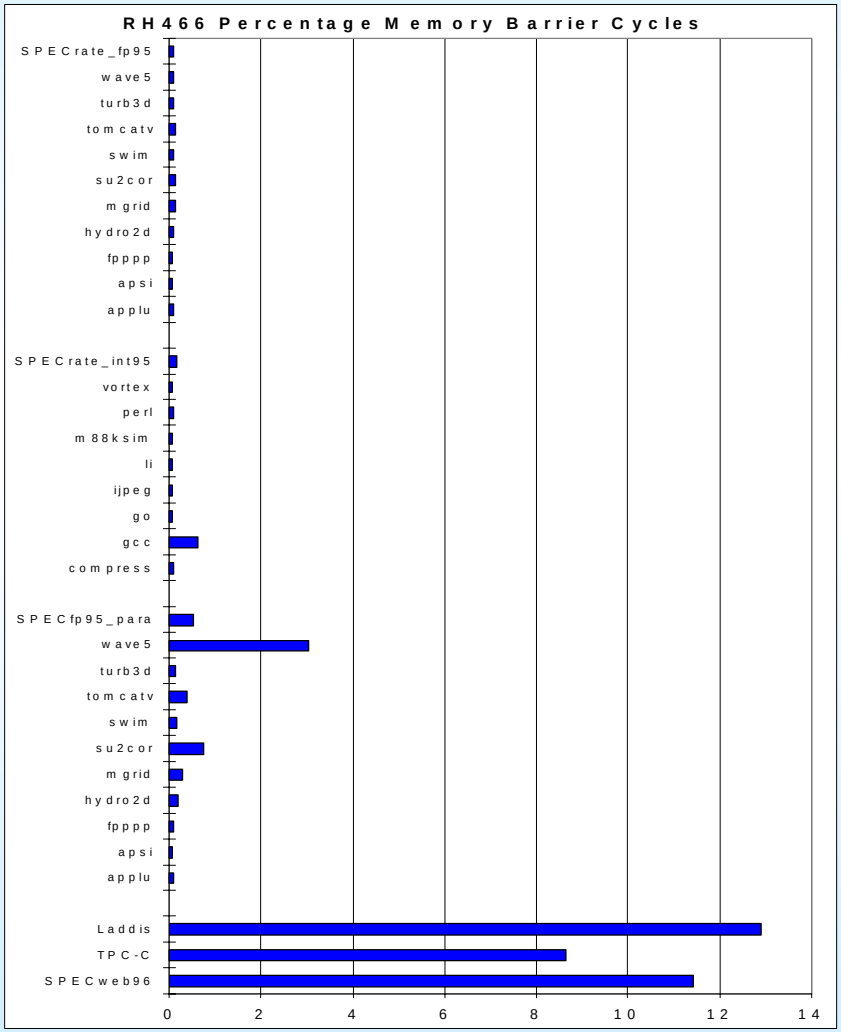
Issuing and Stall Time

- issuing time
  - comparable single and dual issuing time
  - no triple/quad issuing in commercial (no fp)
- stall time
  - higher in commercial than SPECint95
  - SPECfp95: comparable
  - frozen stalls (Dstream) higher than dry (Istream)



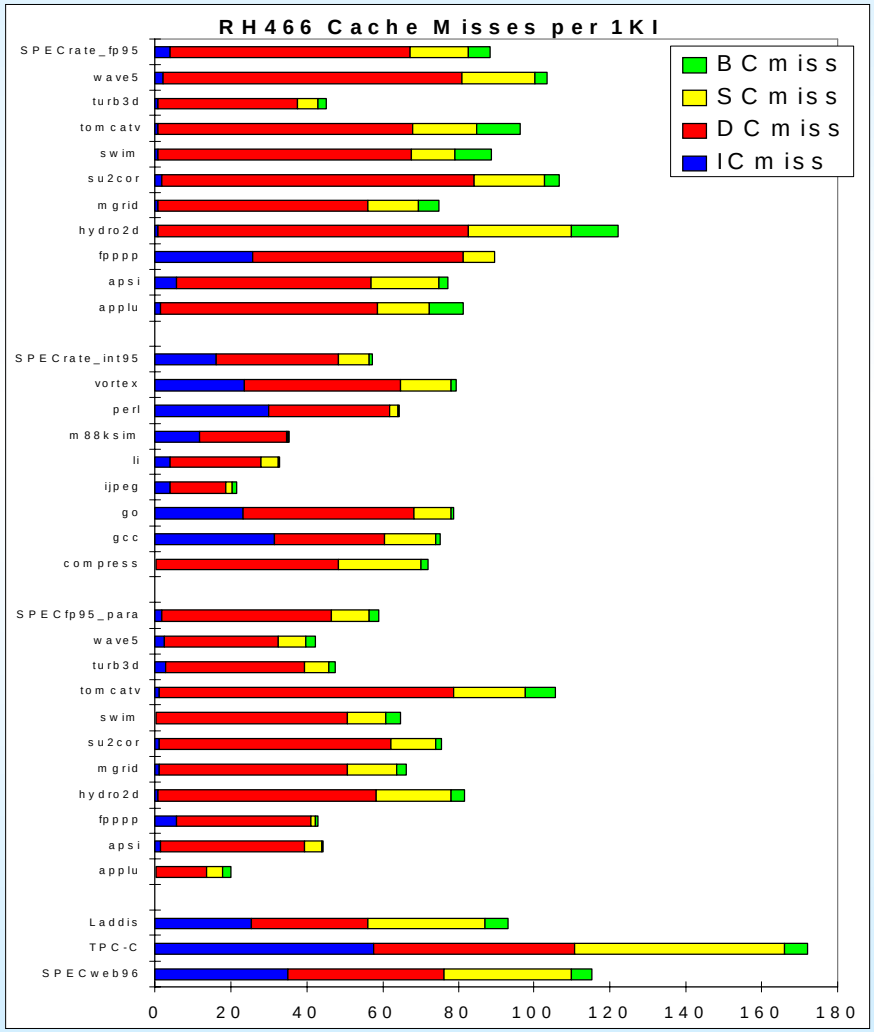
Memory Barrier Time

- MB time high in commercial
- MBs have little effect on SPEC95
  - except parallel: still lower than commercial



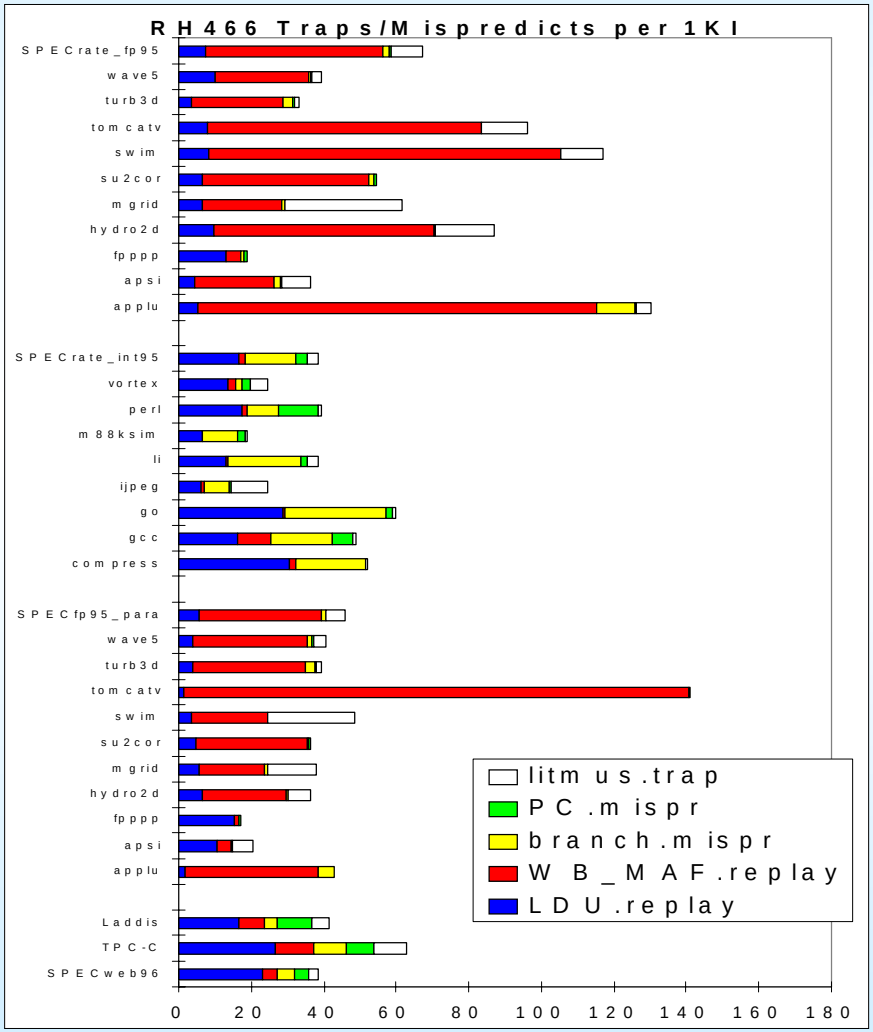
Cache Misses

- high SC misses in commercial **(Bcache bandwidth important)**
- other caches:
  - IC misses higher in commercial (and int95)
  - DC misses higher in SPECfp95 than commercial
  - BC misses higher in SPECfp95 than commercial



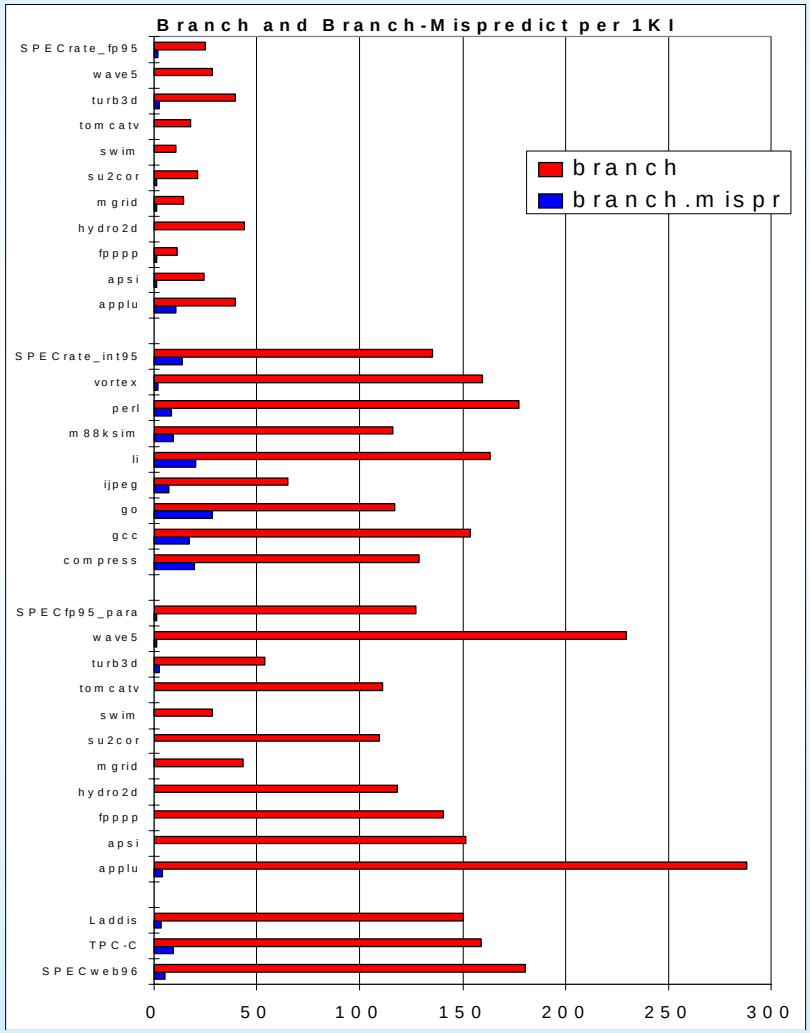
Replay Traps and Mispredicts

- Replays:
  - LDU replays high in commercial (and SPECint95)
  - WB_MAF_FULL replays higher in SPECfp95 than commercial
- branch/PC mispredicts
  - higher in SPECint95 than commercial



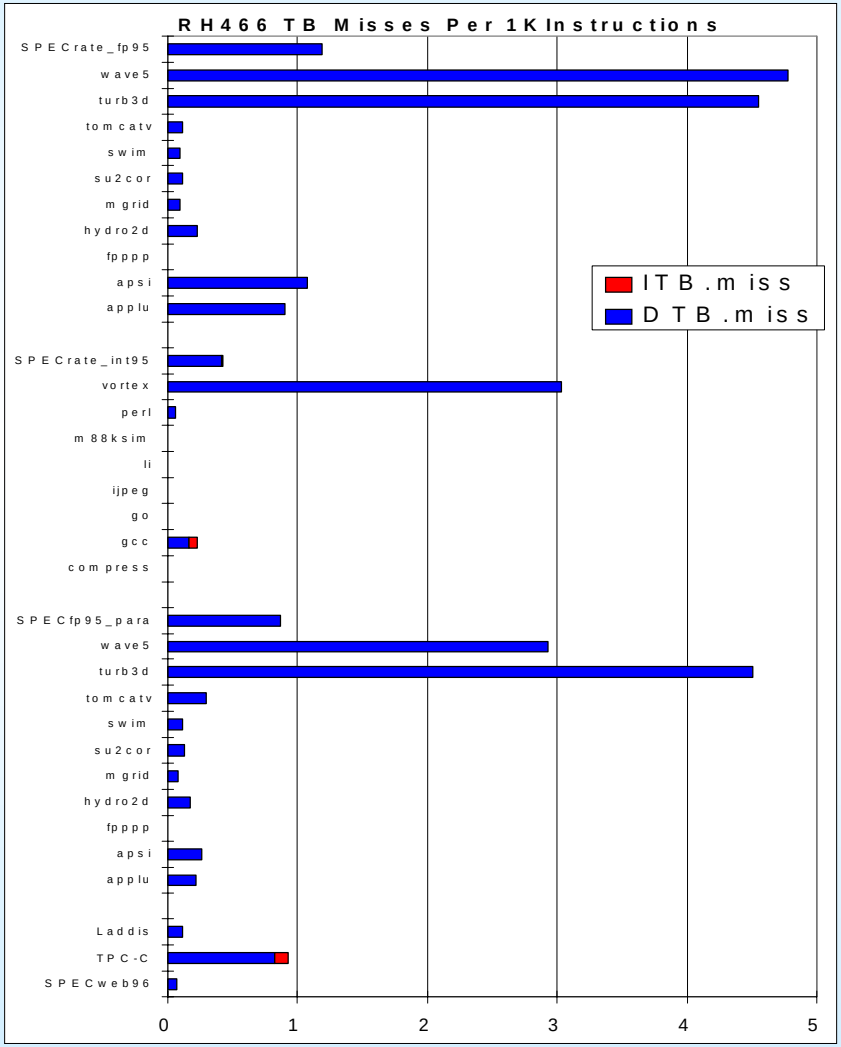
Branch Mispredicts

- branch mispredicts not crucial for commercial performance:
  - number of branches and mispredicts in commercial is comparable to SPECint95



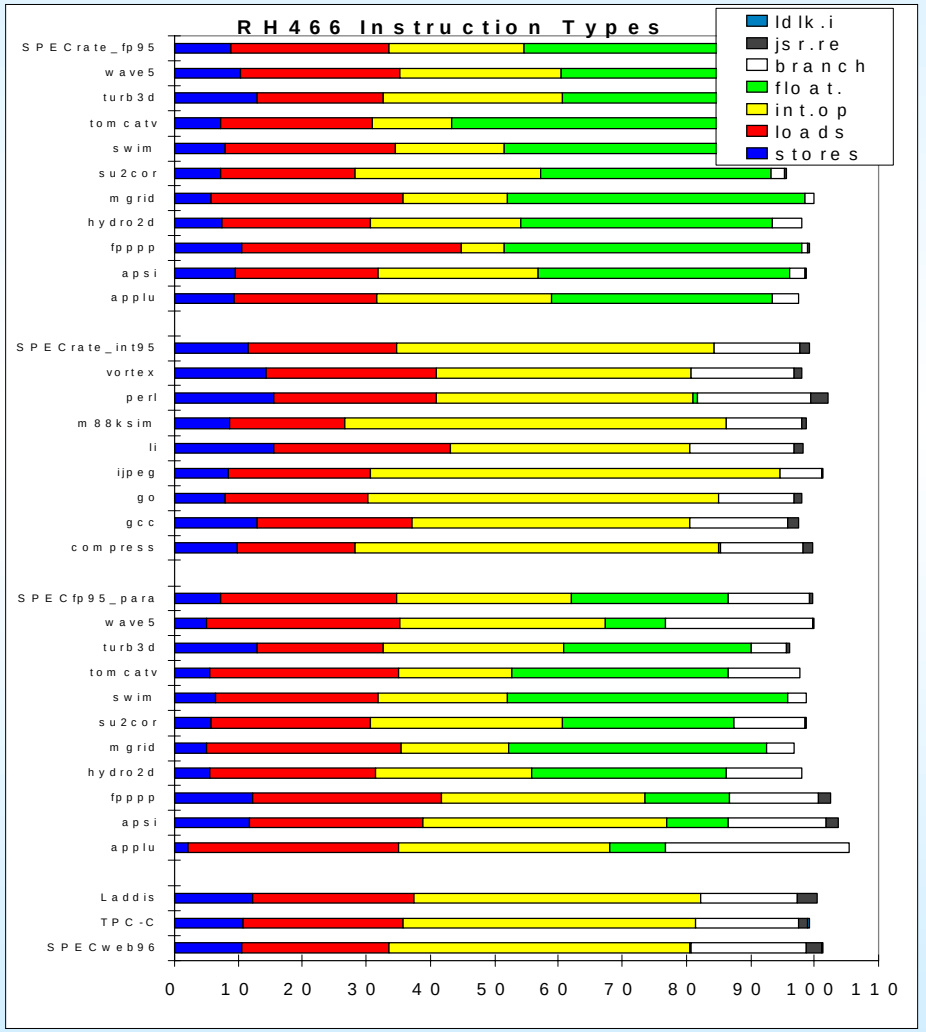
TB Misses

- TB misses not crucial for commercial performance:
  - several technical workloads have higher DTB misses than commercial
  - ITB misses low



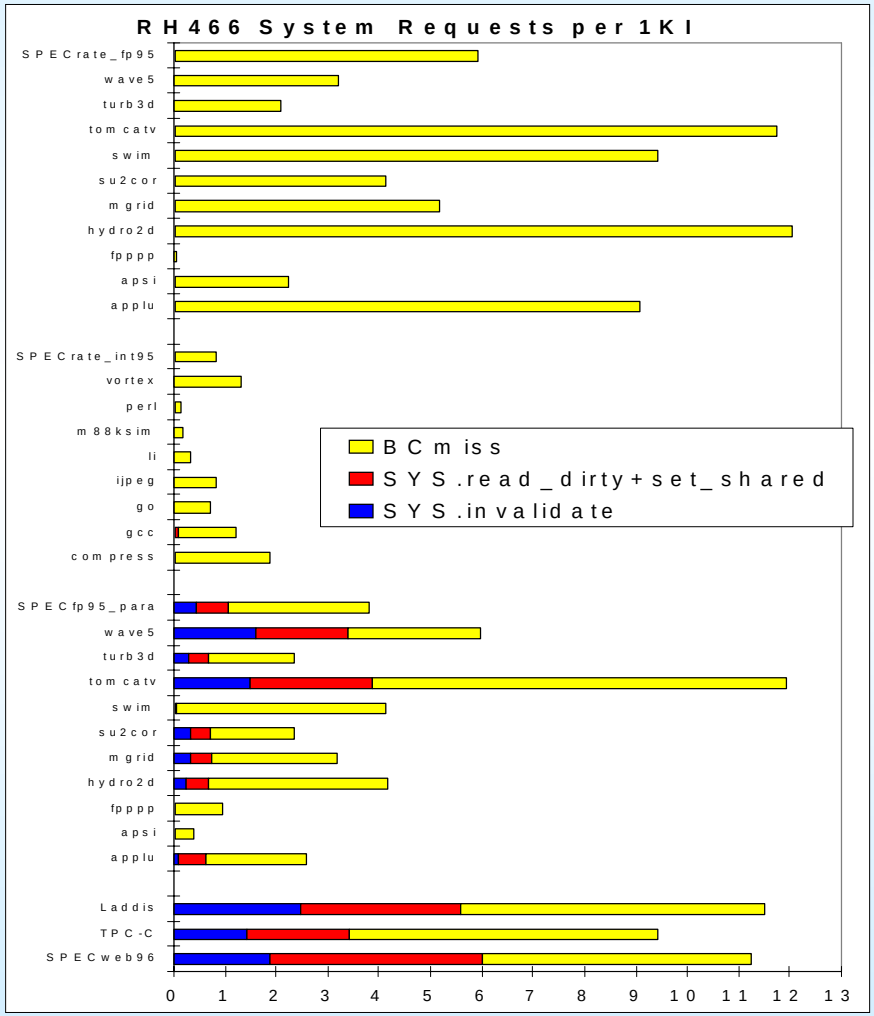
Instruction profiles

- commercial profiles comparable to SPECint95:
  - no fp instructions
  - ~25% loads
  - ~10% stores
  - ~50% integer
  - ~15% branches



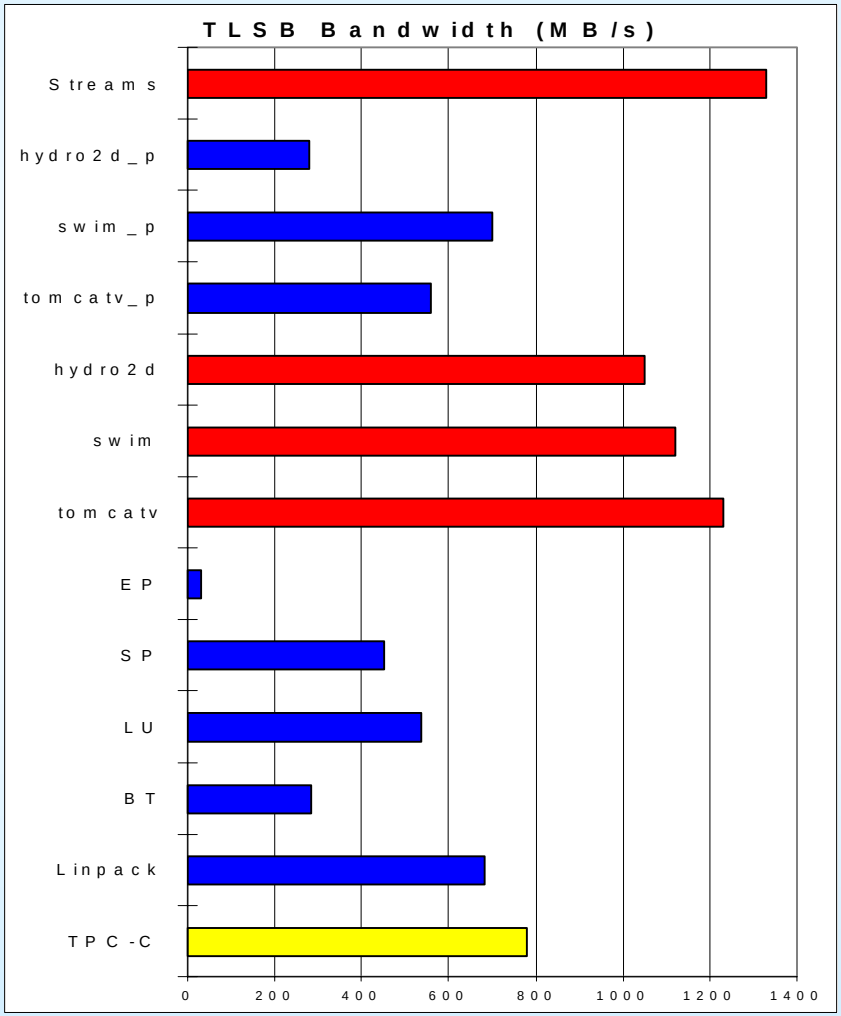
System Requests

- **commercial:** high sharing (ReadDirty and Invalidate)
- **parallel:** high sharing in several workloads
- **rates:**
  - no sharing
  - high bus bandwidth requirements



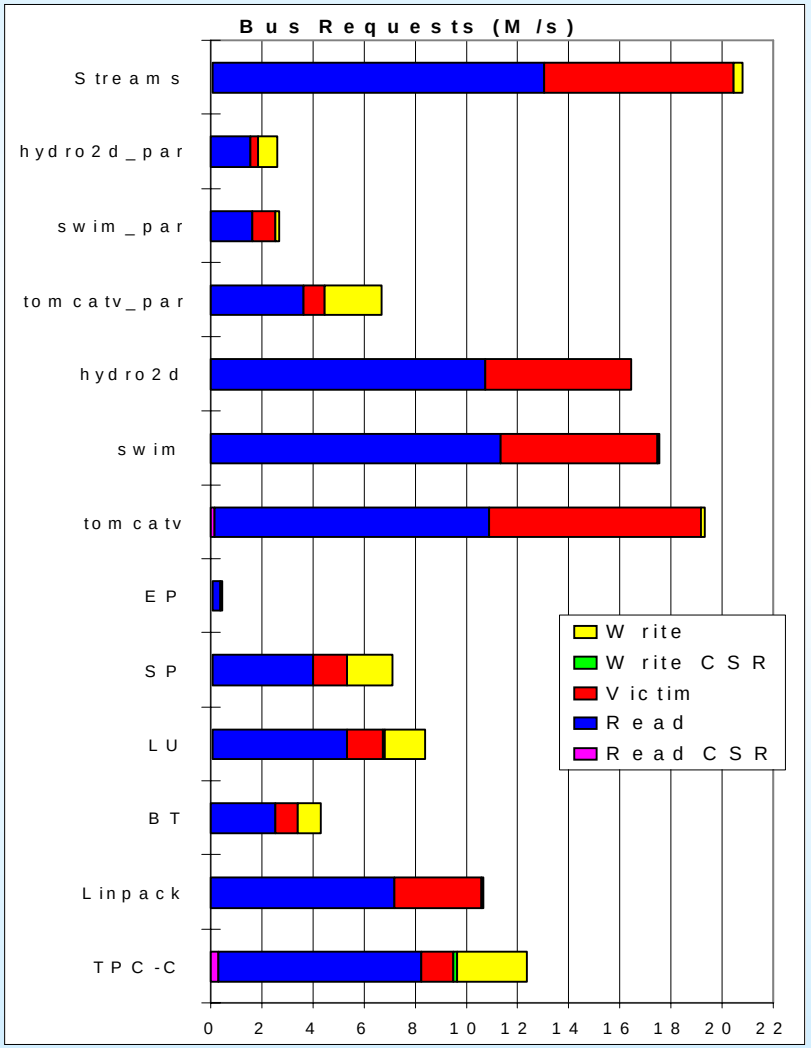
Memory Bus Bandwidth

- AlphaServer 8400:
  - 12 CPUs
- commercial:
  - lower bus traffic than technical multistream
  - not affected significantly by bank conflicts (technical affected profoundly)



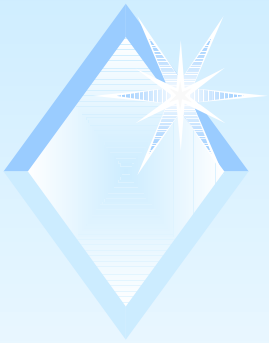
Bus Requests

- commercial:
  - high shared traffic on the bus (Shared Writes)
  - Read/Victim traffic lower than technical
- technical
  - parallel: high shared traffic
  - multistream: high bandwidth (no sharing)



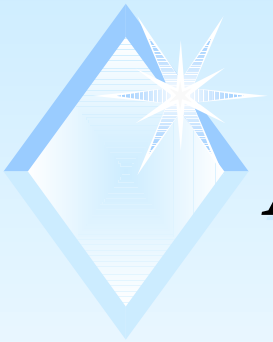
Time-Allocation Model

- model derived from measured events
- high stall components in commercial:
  - S-to-Bcache
  - B-to-memory
  - MB



Summary/Conclusions

- Key factors for commercial performance:
  - high Bcache latency/bandwidth
    - 96KB cache not sufficient
  - low latency data sharing
    - (ReadDirty/Invalidate) on the bus
  - efficient Memory Barriers
    - efficient locks implementation
  - low CPU time per I/O



Acknowledgments

- Thanks to John Shakshober, Huy Phan, Dave Wilson, Paula Smith, Judy Piantedosi for help with profiling data collection

Characterizing TPC-D on a MIPS R10K Architecture

Qiang Cao, Pedro Trancoso,
Josep Lluis Larriba-Pey*, Josep Torrellas

Department of Computer Science
University of Illinois at Urbana-Champaign

***Departament d'Arquitectura de Computadors,**
Universitat Politecnica de Catalunya

Topics Covered

- TPC-D Benchmark & R10K processor
- Query cache misses
- Scaling
- Operation Cost
- Indexing

TPC-D Benchmark

- Decision support benchmark
- 19 queries, including two update queries
- Complex queries:
  - multi-table joins
  - extensive sorting, grouping and aggregation
  - sequential scans
- Running on Postgres95

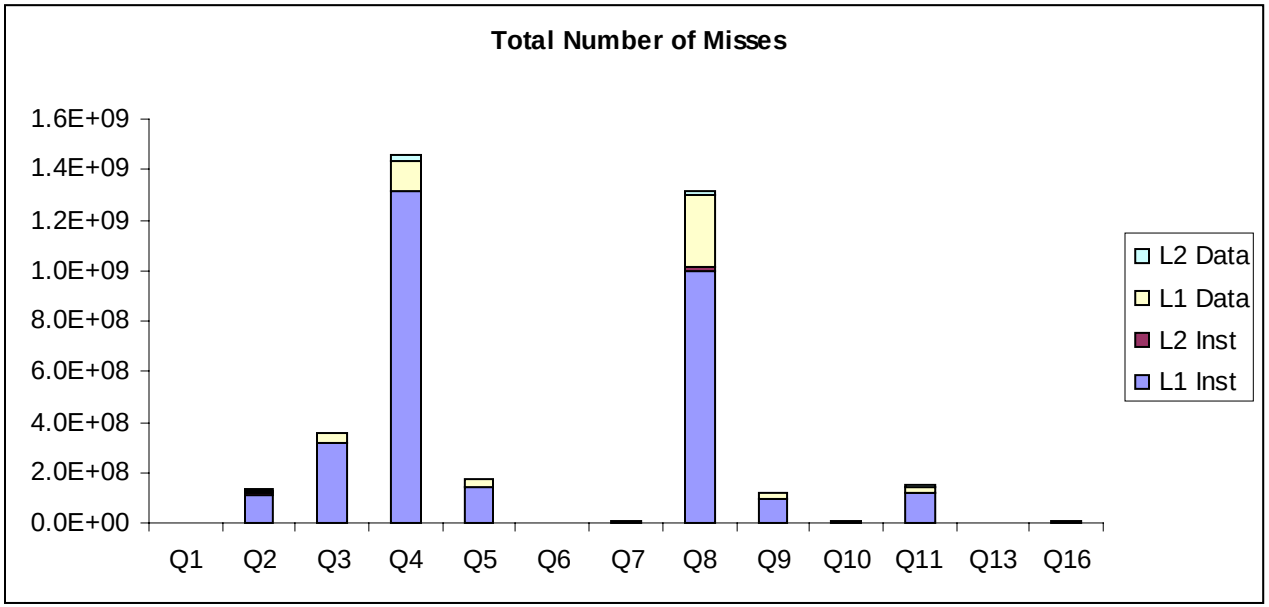
R10K processor

- Four issue superscalar processor
- Two performance counters measure up to 32 events: cycles, L1/L2 Instruction/Data cache misses, etc.
- Events are measured per process
- Save time over simulation

SGI Origin 200

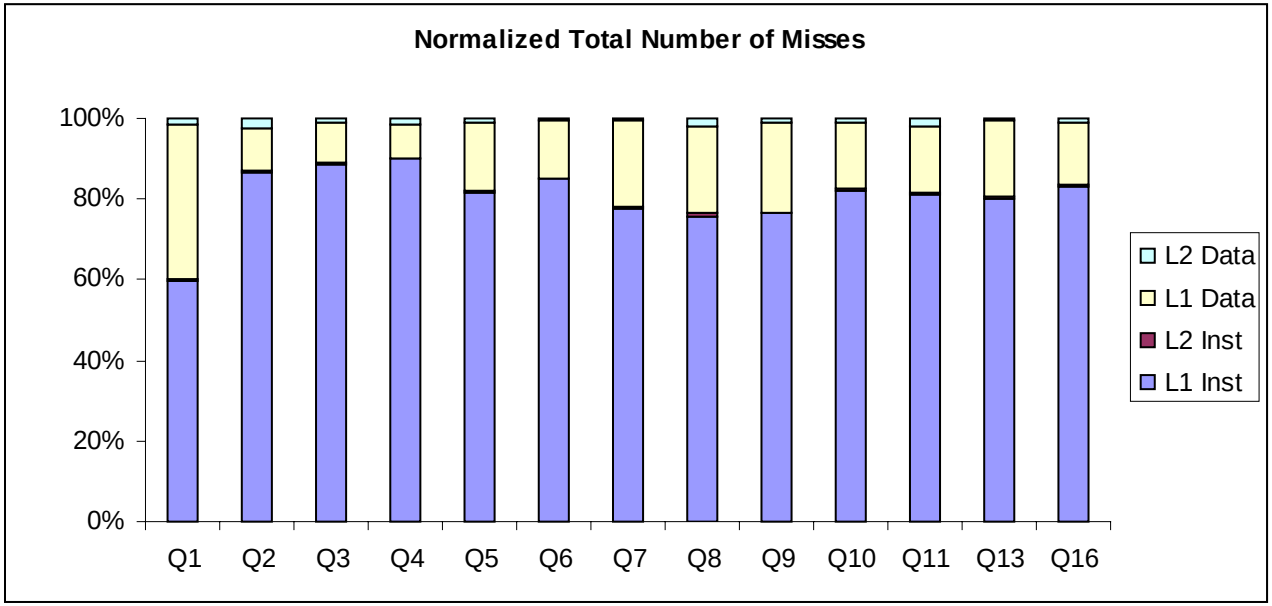
- Scalable Shared-memory
- 4 processors
- 128 MB main memory
- 32 KB L1 instruction cache and 32KB L1 data cache
- 1 MB unified L2 instruction/data cache

Query Cache Misses



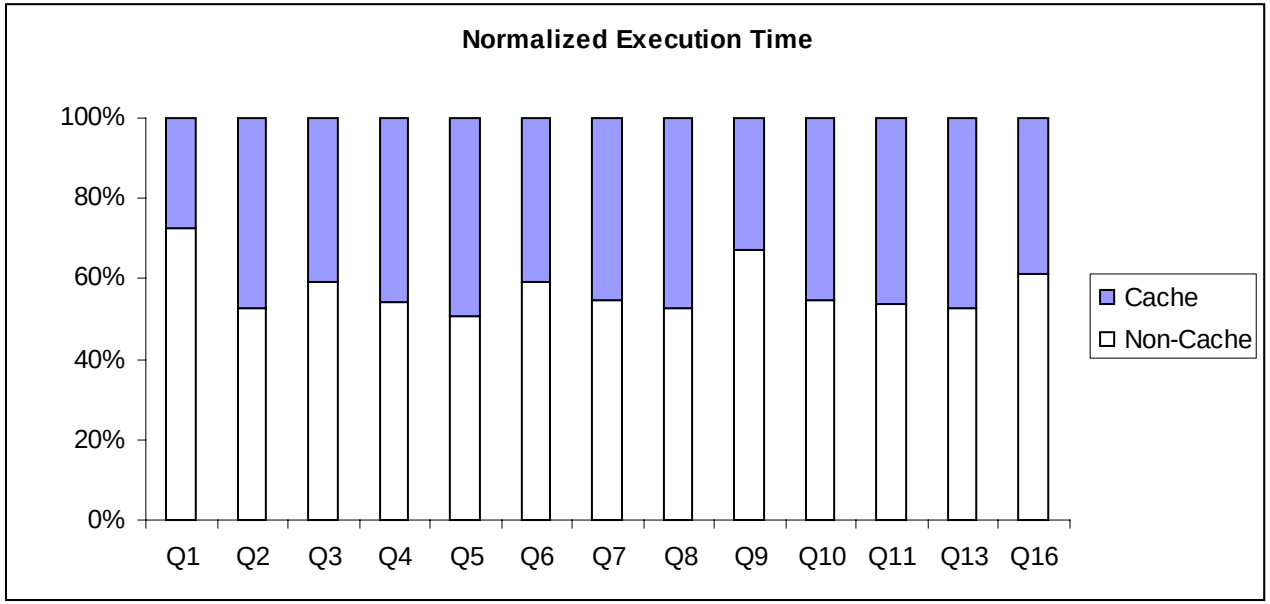
- Some queries (Q4, Q8) have more misses

Query Cache Misses



- L1 instruction misses dominate by far

Query Cache Misses



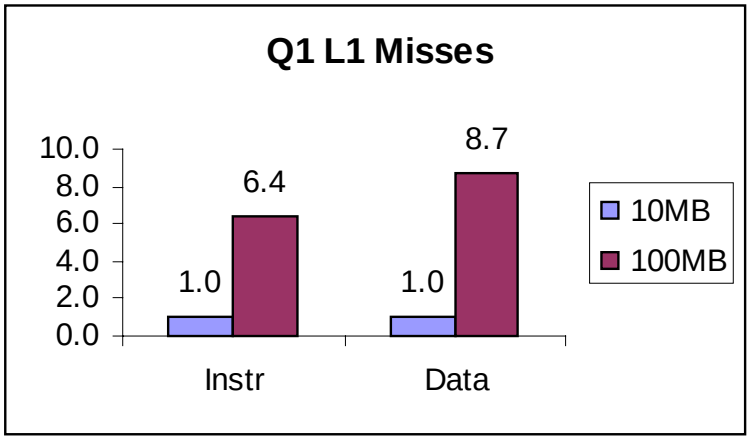
- Cache penalty has significant effect on total execution time

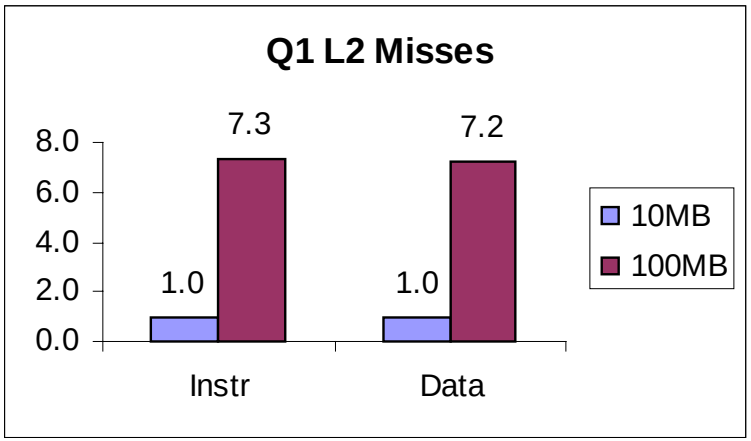
Scaling

- TPC-D specifies a scale factor of 1 to 1000 (1GB to 1TB database)
- Demanding space and time requirement for each run
- Most research studies use scale factor < 1

Scaling

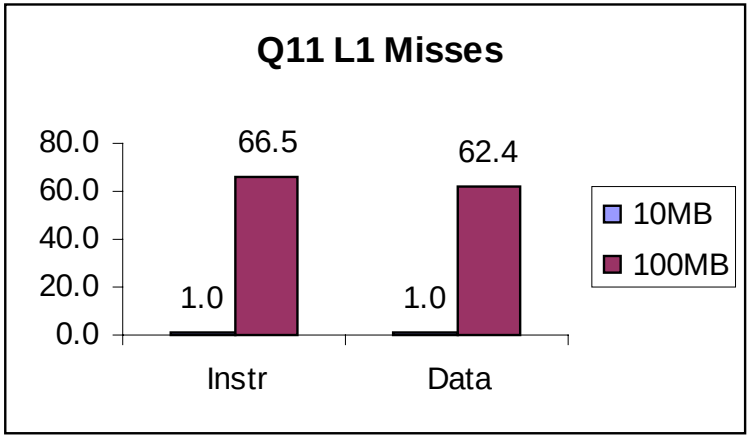
- Cache misses of some queries increase proportionally with the scale factor. Example: Q1

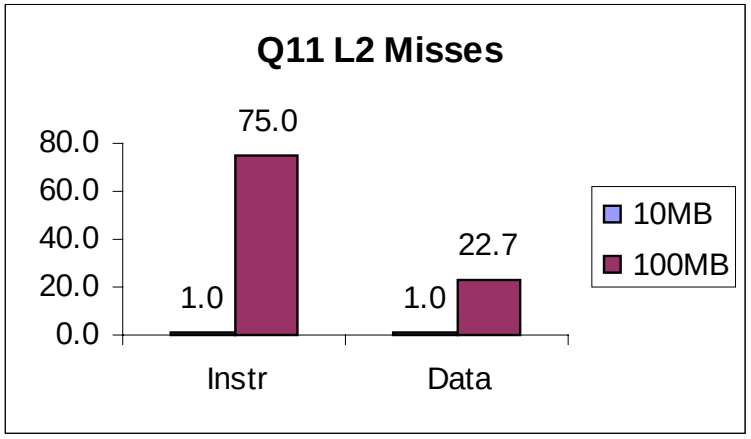




Scaling

- Other queries demonstrate much higher misses than the scale factor. Example: Q11

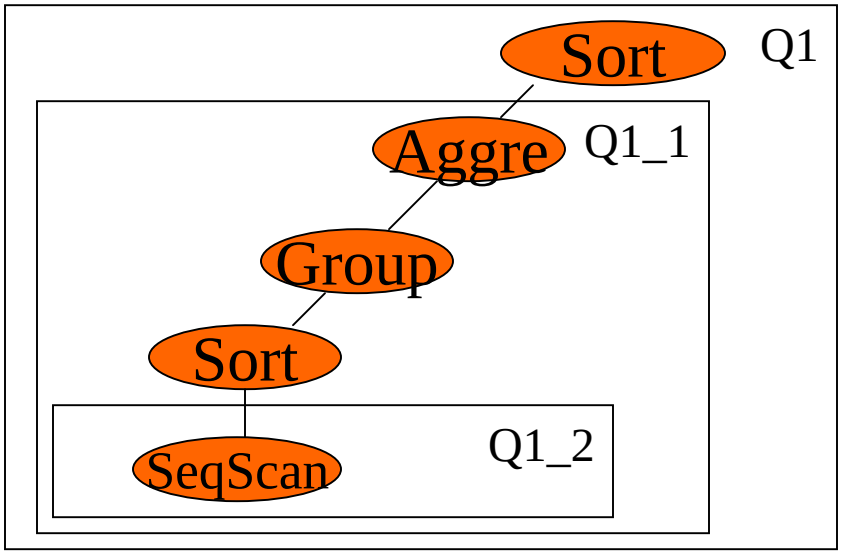


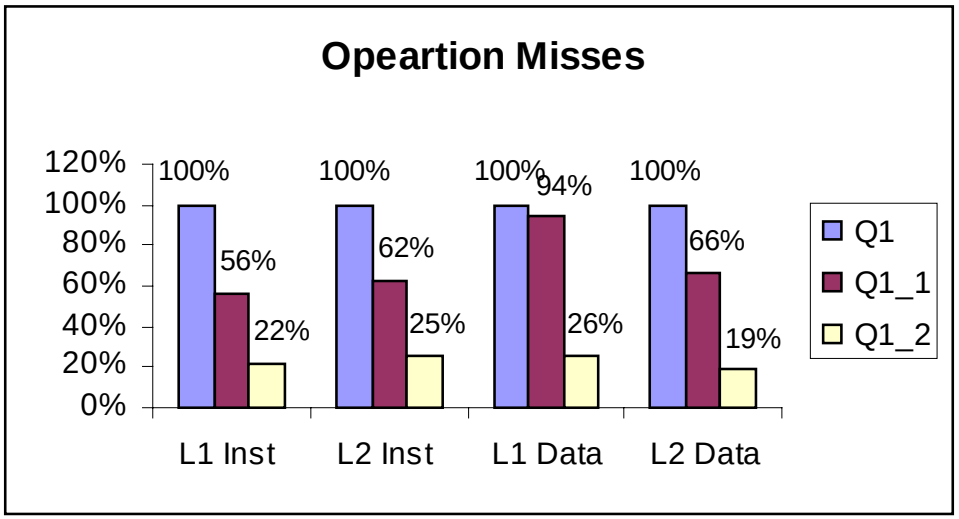


- Queries behave differently with the data size change: Hard to scale down accurately

Operation Cost

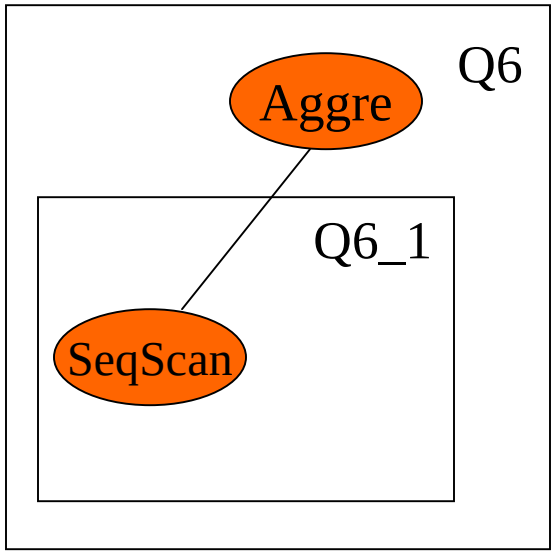
- In some queries, the cost of scan is small

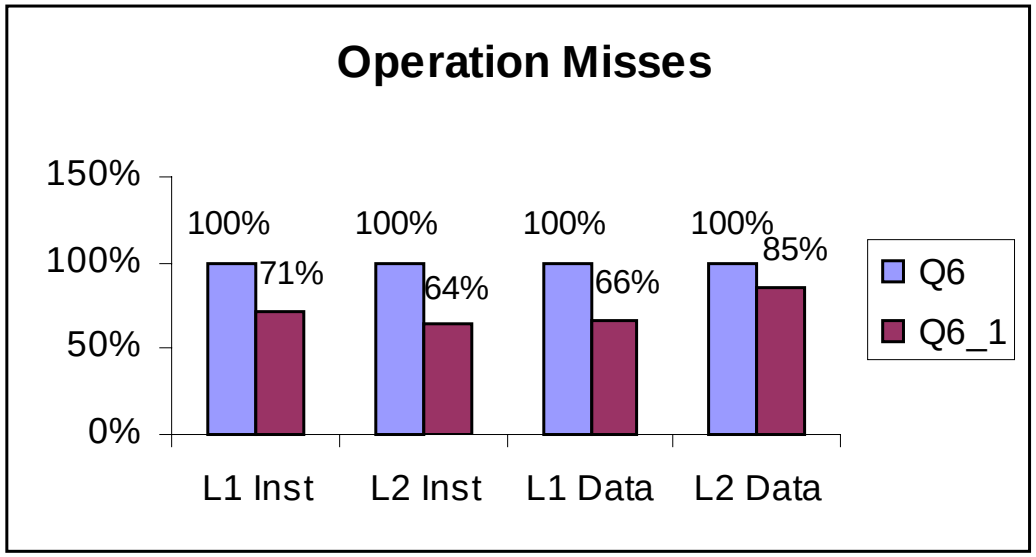




Operation Cost

- In some queries, the cost of scan dominates

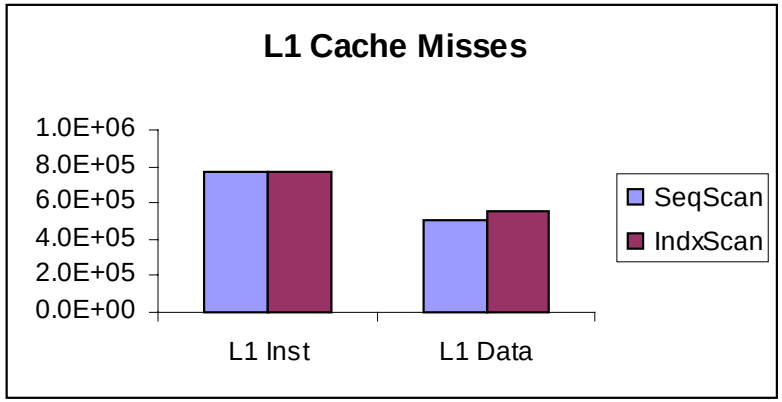


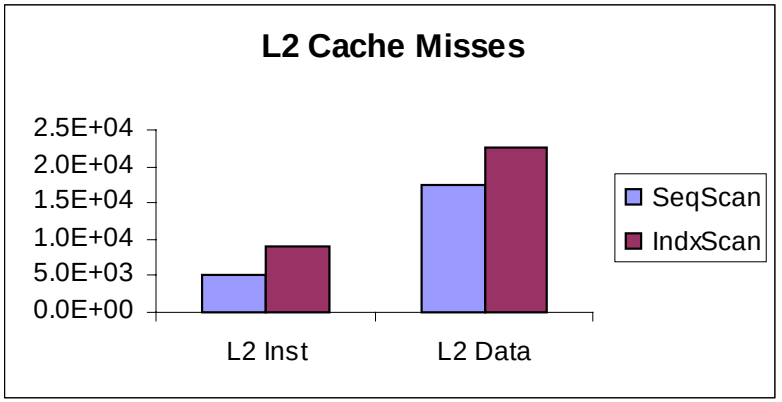


- Conclusion: Need to simulate whole query tree

Indexing

- How does the index structure affect the cache misses ?

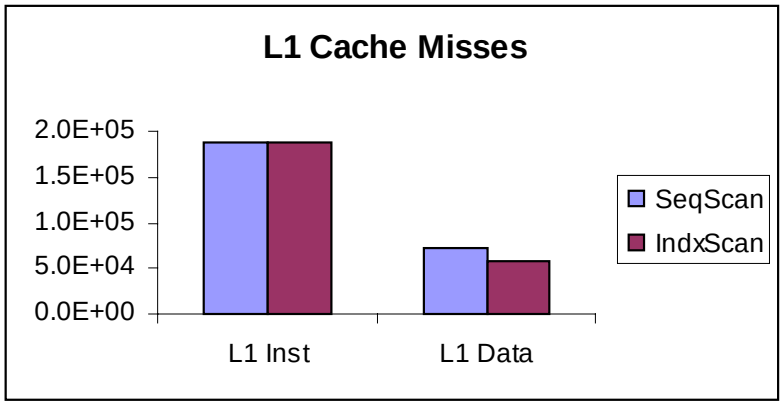


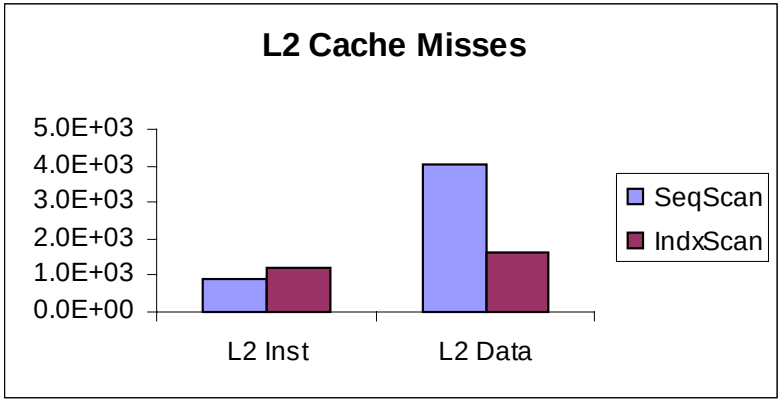


- Complicated indexing structure cause the index scan to suffer more cache misses than the sequential scan

Indexing

- A Modified Q1 with higher selectivity shows fewer data cache misses for index scan





- Optimizer needs to use selectivity factor to choose optimal access method for cache misses

Conclusions

- Instruction misses (L1 especially) dominate
  - Instruction misses should not be neglected in simulation
- Different queries have different scaling behavior
  - Hard to scale down the data accurately
- Operation other than scan can cause many misses
  - Simulation of the whole tree is necessary
- Index Scan can increase cache misses
  - Selectivity factor should be used to choose optimal scan method

Future Work

- More experiments on larger data size
- The effect of initial data allocation
- Interaction of multiple queries