Page Prefetching for Database Workloads

Rabin Sugumar and Partha Tirumalai
Sun Microsystems

Srikanth Srinivasan
Duke University

Motivation

- CPI on commercial workloads is dominated by data stream cache misses
  - 50% or more by some estimates
- There exists little published work on what code constructs cause misses.
- Insight on missing constructs in scientific workloads has led to new techniques
  - e.g. prefetching coupled with software pipelining

- Here,

  We identify a source of misses in database servers and propose a solution.

Overview

- Database pages
- Trace collection setup
- Database page access characteristics
- Page prefetch buffer
- Simulation results
- Conclusion

Database Pages

- Database tables and associated indexes are stored in units of pages.
  - These pages are typically 2KB to 8KB
  - Unrelated to virtual memory on the host system.
  - We call them *DB-pages* to avoid confusion.
- Database servers operate on DB-pages
  - For record lookup, update etc.
- Total data stored in DB-pages is huge
  - essentially the entire database, so ...

☞ *DB-page accesses have high miss rates*

A Toy Example

Root page

Page:0

Adam	1
Eve	2
John	3
Matt	4

Index pages

Page:1

Adam	6
Bill	5
Chris	8
Dave	7

Page:2

Eve	6
Fred	8
Hank	5
Jane	8

Page:3

John	6
Lisa	7
Luke	5
Mark	7

Page:4

Matt	7
Neal	8
Pat	5
Sue	6

Data pages

Page:5

Luke	*data*
Pat	*data*
Bill	*data*
Hank	*data*

Page:6

Adam	*data*
Sue	*data*
John	*data*
Eve	*data*

Page:7

Mark	*data*
Matt	*data*
Dave	*data*
Lisa	*data*

Page:8

Fred	*data*
Neal	*data*
Chris	*data*
Jane	*data*

☞ **To get record for Lisa pages accessed are:**
Page:0 Page:3 Page:7

DB-Page Processing Sequence

We call

- The sequence a *DB-page Processing Sequence (PPS)*
- and a load or store in a PPS a *DB-page access*.

Database Setup and Tracing

- Workload is a commercially available database running an order-entry application.
- Database size was about 12 GB.
- Collected a trace of 1 billion instructions.
  - Used Shade to collect the trace.
  - Did not trace kernel execution.
  - Activity from multiple threads interleaved in trace.
- Trace used for analyzing single CPU performance.

Sun microsystems

Trace Validation

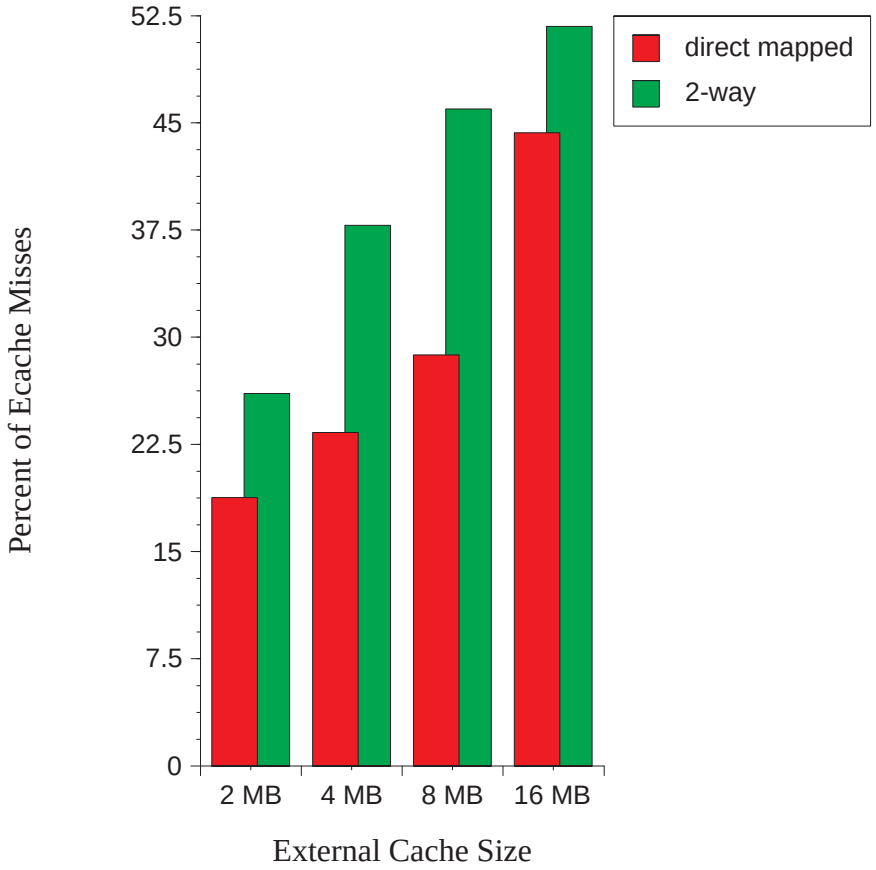
- Trace function profile matches gprof output.
- Traces collected at different execution points show similar behavior.
- Miss ratios from trace close to miss ratios measured on hardware.

- Kernel is 25% of execution -- significant.
- However, user misses subset of user+kernel misses.

DB-Page Access Characteristics

- DB-page accesses are 5% of all loads/stores.
- 20%-60% of E-cache misses on DB-page accesses.
- Per PPS there are on average
  - 70 accesses
  - 5 E-cache lines touched and
  - ~2000 instrs between first and last access.
- Few PPS's (1 to 2 typically) active at a time.
- Limited inter-PPS DB-page reuse.
  - More on root DB-pages.
  - Less on index/data DB-pages.

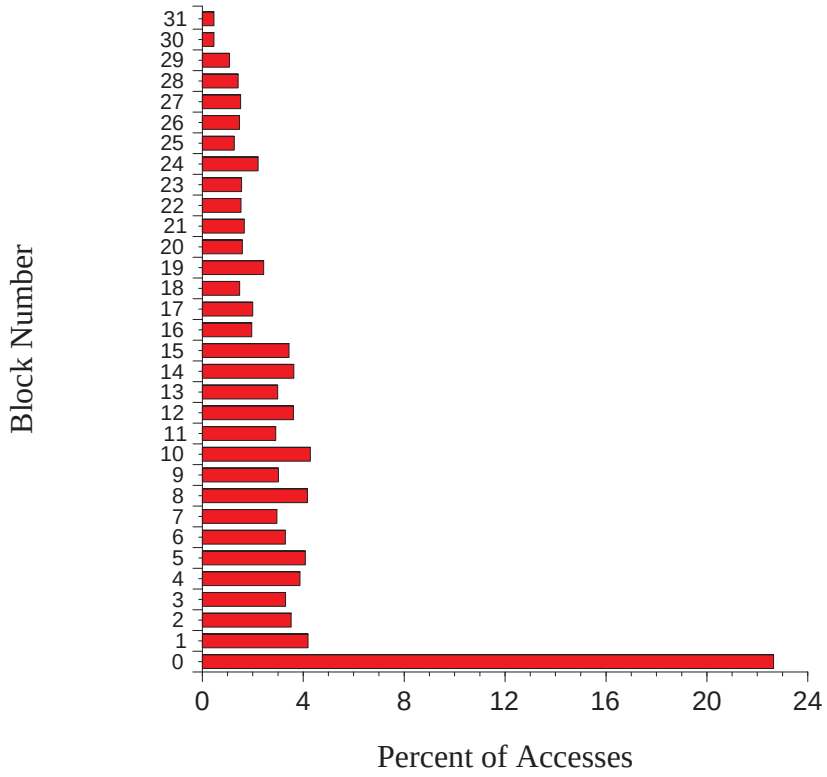
Misses on DB-Page Accesses



- As caches get bigger and more associative DB-page access misses dominate.

Note: Cache misses drop with increasing cache size. But percentage contribution of DB-page access misses increases

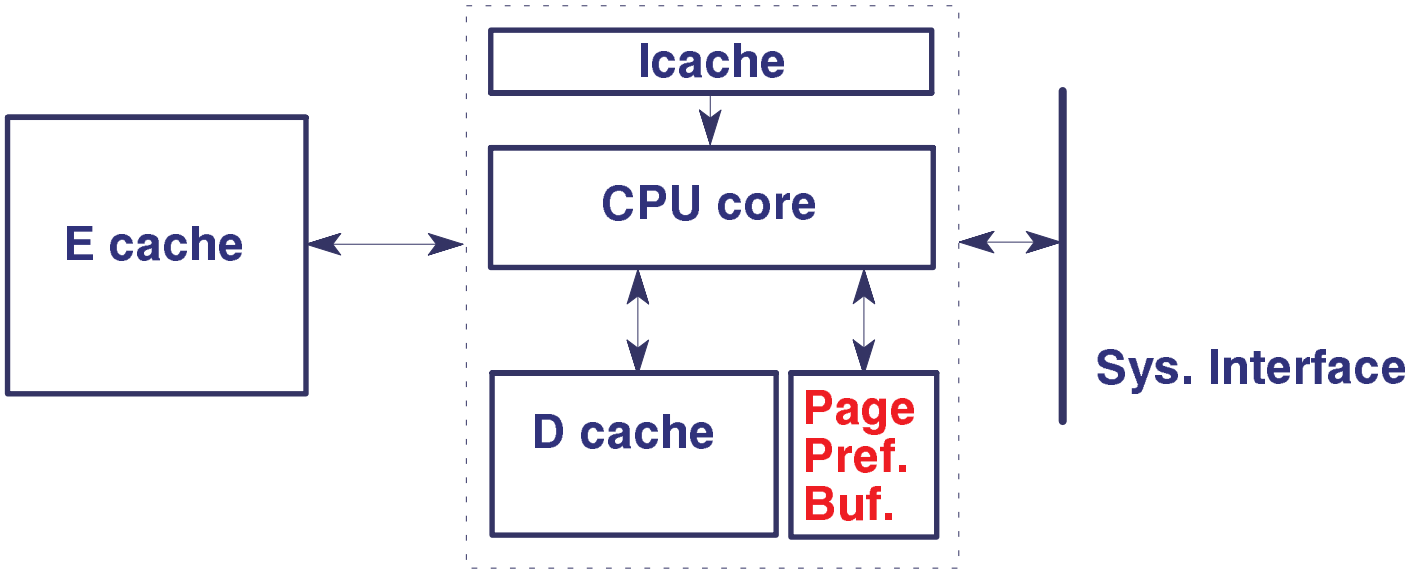
Distribution of Accesses within DB-Page



- Of the 32 64B blocks, Block 0 is hot.
- Accesses slightly skewed towards lower half.

DB-Page Prefetch Buffer

- Small fully associative buffer of 2 to 4 DB-pages.
- To prefetch and hold DB-pages during processing.
- Reduces DB-page access misses and cache pollution.



Related structures:

Stream buffers (Jouppi ’91), P-cache (US-III; Lauterbach ’97),
Assist Cache (HP PA7200; Chan et al ’96)

How it Works

- Define new flavor of prefetch instruction to move pages into the DB-page buffer.
- Insert prefetch at the time of DB-page fetch
  - Primary insertion points:
    - When page is located in memory buffer pool.
    - When page is fetched from disk.
- Insertion accomplished either
  - through pragmas in source or
  - through the compiler using profile feedback

\+ Cache misses reduced

– Higher memory traffic

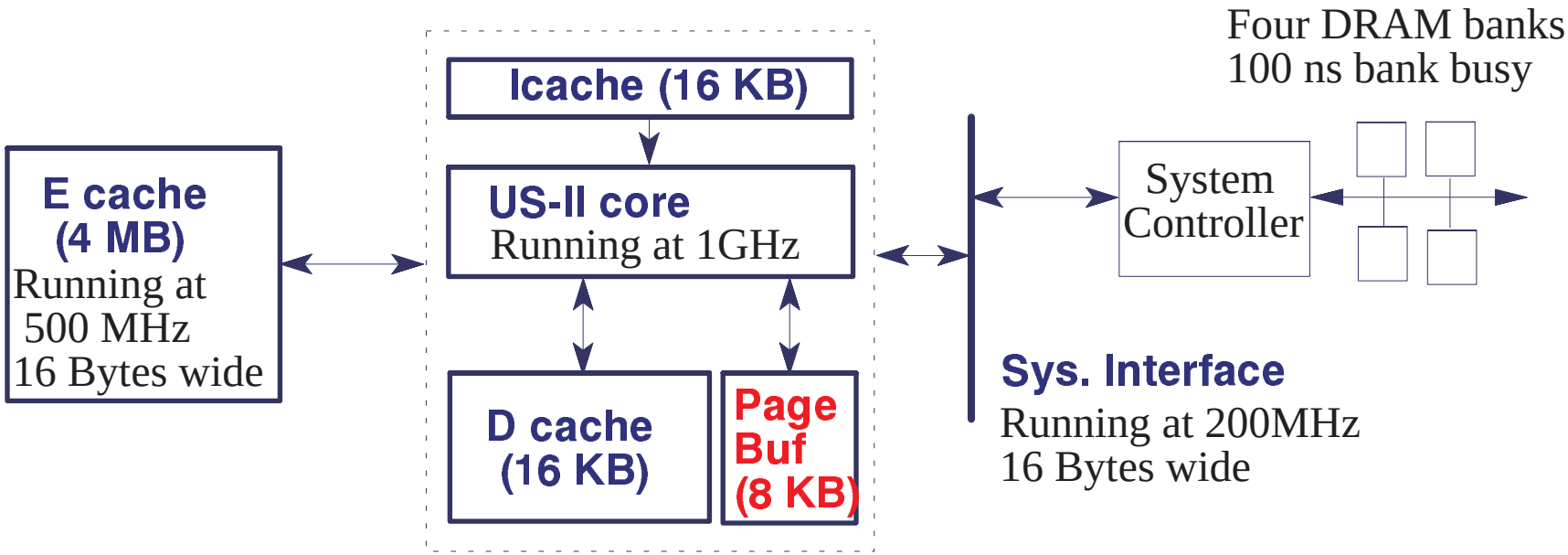
Simulation Infrastructure

- Trace preprocessed to identify prefetch points.
- Added DB-page buffer to our UltraSPARC-II model.
- CPU frequency and memory bandwidth scaled up to match future projections.
- Simulator inserts new prefetch instruction at identified points.
- Each prefetch sends out

  (DB-page size)/(E-cache line size)

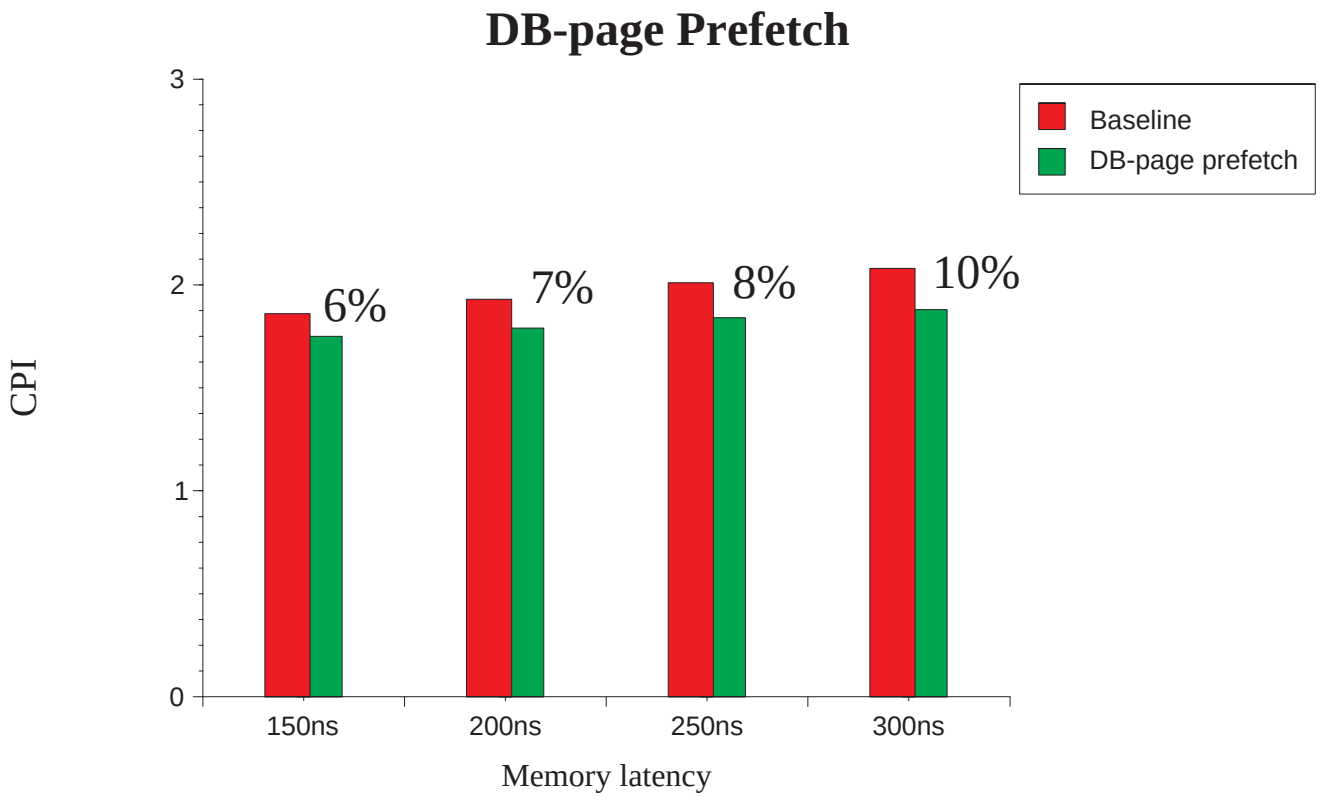
  individual requests to memory system.
- DB-page prefetches bypass D-cache and E-cache.
- Prefetch buffer is write-through (same as D-cache) and is snooped.

Simulator Configuration



- 8 KB page buffer that can hold four 2KB DB-pages.
- Peak memory bandwidth = 3.2 GB/s
- Sustained memory bandwidth on synthetic load benchmark = 2.2 GB/s

Simulation Results



- DB-page prefetch improves performance by 5% to 10%.
- Performance improvement increases with memory latency

Memory Bandwidth Impact

- Memory bandwidth utilization
  - Without prefetch: 4%
  - With prefetch: 12%
- 3X increase in bandwidth has little impact
  - Absolute utilization remains low.
- CPU memory bandwidth designed for scientific workloads
  - Not stressed on such commercial workloads.

Summary

- Database records are accessed as DB-pages.
- DB-page accesses cause many E-cache misses.
- Prefetching DB-pages could improve performance by 5% to 10%.

- Work to be done:
  - Investigate effect kernel has on prefetching.
  - Investigate impact of DB-page prefetching in MP systems where many misses are coherence related.