Cache Performance of Operating System and Multiprogramming Workloads

Anant Agarwal, John Hennessy and Mark Horowitz

Presented by: Zhiheng Wang

03/06/01
Outline

• Introduction
• Methodology (Trace-Driven Simulation)
• Transient Behavior and Trace Sampling
• Performance of System References
• Impact of Multiprogramming
• Conclusion
I. Introduction

- Faster CPUs need better cache performance
- Cache Performance
  
  \[ T_c + mT_m \]

- Differences from the former studies
  - cache size
  - number of processes
  - examination for OS workload
II. Methodology

- Trace-Driven Simulation (TDS)
- ATUM: Address Tracing Using Microcode
- Parameter Space
II. Methodology

Trace-Driven Simulation

- Hardware measurement
  - Limited to existing cache organizations
- Analytical modeling
  - Estimation, not for fine-tuned results
- (Previous) Trace-Driven Simulation
  - No multiprogramming and OS workload
  - Limited trace length
II. Methodology

ATUM: Address Tracing Using Microcode

• Microcode Modification
  – can capture all processes’ mem. activities

• Traces Used
  – VAX8200 (Ultrix and VMS)
  – 30 traces were used
  – about 400,000 references each (0.5 sec)
II. Methodology

Parameter Space

• Cache parameters
  cache size, # of sets, set size, block size, replacement algo. (LRU)

• Performance metric: cache miss rate ($m$)
  – important to performance
  – independent of processor and system implementation
  – facilitate the comparison to other studies
III. Transient Behavior and Trace Sampling

• Transient Behavior Analysis
  – Motivation
  – Definition of cold-start and warm-start
  – Analysis of start-up effect

• Trace Stitching
  – Why need to stitch?
  – Why can be stitched?
Transient Behavior Analysis

• Motivation
  – Start-up distortion
  – Simulation goal
    obtaining a “steady state” miss rate
  – Trace sampling can be affected by start-up distortion
III. Transient Behavior and Trace Sampling -- Transient Behavior Analysis

start-up distortion

Fig. 1. Miss rate versus time for benchmark IVEX. Set size is 1; block size is 4.
Definition of cold-start and warm-start

- Previous definition
  - cold-start: start with an empty cache
  - warm-start: first allow the cache to fill up
  
  **Not suitable for large cache size!**

- New definition
  - warn-start: first allow the cache to fill up (cache saturation), **or the initial work set of the program or workload to be cached (trace saturation)**
  - cold miss
warm-region detection: cold miss rate

Fig. 2. Cumulative cold misses (or misses to empty cache block frames) for IVEX for a 16K-byte cache and block size 4 bytes.
Analysis of start-up effect

• Goal
  To determine the length of the trace to get to “steady state”.

• Start-up effect in Single process traces
• Start-up effect in multiprocess traces
• Summary
III. Transient Behavior and Trace Sampling-- Analysis of start-up effect

Start-up Effect in Single Process Traces

• Conclusions:
  – entering steady state after 25K ref.
  – trace size issue are only relevant to large caches
  – workload characteristics is important --> multiprogramming
III. Transient Behavior and Trace Sampling—Analysis of start-up effect

**Start-up Effect in Multiprocess Traces**

- Conclusions (cache size: 16KB)
  - With low multiprogramming level, the interference is not increased.
  - With high multiprogramming level, start-up period reduce due to interference.

![Graphs showing cumulative cold misses with different multiprogramming levels](image-url)

*Fig. 4. Compare cumulative cold misses for uniprogramming and multiprogramming. Cache size is 16K bytes, block size is 16 bytes, and set size is 1.*
Conclusions

- Cache saturation dominate in cache up to 64 KB.
- In larger caches, trace saturation causes the cold-start period for multiprogramming to be bounded above by 600K ref.
Summary

- Uniprogramming traces enter steady state within 100K ref.
- Multiprogramming traces show a much longer start-up phase, often over 600K ref.
- High multiprogramming level does not increase the cold start period too much due to interference
III. Transient Behavior and Trace Sampling

Trace Stitching

- Why need to stitch?
  - For multiprogramming, to get the miss rate in steady state we need 600K ref.
  - Each of the traces only contain about 400K ref.
  - Basic idea: concatenate the short segments into a longer segment
Trace Stitching (cont.)

• Why can be stitched?
  – co-similarity and auto-similarity
    e.g. A=1,2,3,4,5,3,4,5,6,7, B=6,7,8,3,4,5,3,4,5,6,7,8,3,4
    as(A)=3   as(B)=6   cs(AB)=5
  – Similarity ratio $\rho$
    similarity ratio $= \frac{5}{(3+6)/2} = 1.1$
  – If the similarity ratio between two trace samples is no less than 1, the two traces can be stitched to form a longer trace
IV. Performance of System References

• Data filtering
  – Traces with low levels of multiprogramming was chosen to exclude multitask effect
  – Result for both VMS and Ultrix will be discussed.
  – 20 traces were used
  – Average system references percentage 20%(VMS) and 50%(Ultrix)
Motivation

- System reference degrades the cache performance
IV. Performance of System References

Miss Rate Components

- User component drops fast
- For large cache, OS-User component is small

Fig. 8. Components of system and user miss rate. Set size is 1, block size is 16 bytes.
IV. Performance of System References

**Analysis of system miss rate**

- Compare system references to user references

---

**Fig. 9.** Comparing inherent system and user miss rate for VMS and Ultrix traces.
IV. Performance of System References

Analysis of system miss rate (cont.)

- System references have a much higher miss rate than user reference.

- Explanations
  - system code and data structures are bigger
  - system loops have fewer iterations
IV. Performance of System References

**Associativity**

- Aim at reducing the OS-User interference

The figures show diminishing returns of associativity.
IV. Performance of System References

Block size

• Aim at reducing cold-miss

The effect is almost the same to OS and User

• Block-size effects are more significant in large caches

Fig. 12. Effect of block size on user and system miss rates (a) VMS (b) Unix.
IV. Performance of System References

**Split Caches**

- Aim at reducing OS-User interference
- Split OS-User cache
  - no better than unified caches
  - much worse in small caches
- Split Instruction-Data cache
  - miss rate are about the same all the time
V. Impact of Multiprogramming

• Data filtering
  – MUL3, MUL6, MUL10
  – 3 traces for each level
  – Stitching traces in the same level to form a longer trace
  – pre-load the cache with 600K references
V. Impact of Multiprogramming

Multitasking Cache Techniques

- Two techniques: flush and PID
- Relative performance

![Graphs showing cache miss rates](image)
Impact of Shared System Code

- Purging only user references is better than purging the whole cache
- When shared system code included, PID scheme can renders better performance than uniprocess!
V. Impact of Multiprogramming

Process Switch Statistics

• How does the process switch affect the cache performance?

Conclusion:
Lack of strong correlation between the multiprogramming level and the average time quantum
V. Impact of Multiprogramming

Associativity

- In small cache, more impact to the uniprogramming workload
- In large cache, more impact to the multiprogramming workload
V. Impact of Multiprogramming

Block Size

• Block size has a strong impact on multiprogramming performance
  – reduce cache miss when switching back
VI. Conclusion

• System references degrade cache performance due to their large working set and less repetitive

• The combined working set of multiple processes degrades cache performance, although less significant in large cache or in systems sharing data across processes.