Multithreaded Architectures

Metha Jeeradit and Wajahat Qadeer

Outline

- Introduction
- Papers
- Discussion

Introduction: Motivations

- Long Memory latency
  - Speed gap between memory system and processor is increasing
  - Especially in multiprocessors
- Two approaches
  - Avoid it (e.g., cache)
  - Tolerate it (e.g., dynamic scheduling)
- Multithreading is a latency tolerant scheme
  - Any latency and not just memory

Introduction: Requirements

- Way to store multiple contexts
  - Register states and PCs for each context
- Non-blocking cache
- Bandwidth (Memory and Network)
- Low context-switching cost

Introduction: Existing Architectures

- 3 categories:
  - Fine-grained
    - Perform work from different threads in each cycle
    - Zero-cycle switching overhead
    - Only allow one instruction from each context to be active in the pipeline (no interlock)
    - No data cache
  - Blocked (Fast-context switching)
    - Condition switch when long latency event is encountered
      - e.g., on every branch misprediction or on every cache miss
    - Significant Switching cost
  - Interleaved
    - Like Fine-grained but has data cache and interlock
    - Substantial complexity

5

Paper 1: Analysis of Multithreaded Architecture for Parallel Computing

(R. H. Saavedra-Barrera, D. E. Culler, T. V. Eicken)

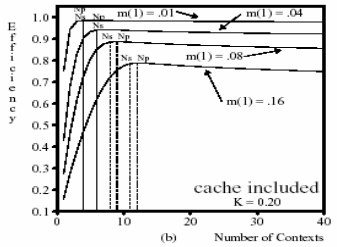
- Summary
  - Characterize the behavior of a multithreaded architecture with an analytical model based on 4 parameters: Latency (L), number of threads (N), switching cost (C), and run-length interval (R)
    - First 3 parameters are architectural dependent
    - R is both architectural and application dependent

6

Paper 1 (cont.): Analysis of Multithreaded Architecture for Parallel Computing

(R. H. Saavedra-Barrera, D. E. Culler, T. V. Eicken)

- Summary (cont.)
  - 2 operating regions: Linear vs Saturation



  - C/R is the most important parameter
    - Ratio dictates peak utilization value

7

Paper 1 (cont.): Analysis of Multithreaded Architecture for Parallel Computing

(R. H. Saavedra-Barrera, D. E. Culler, T. V. Eicken)

- Strengths
  - A deterministic model that can be used to gauge the performance of your designed architecture
- Weaknesses
  - Debatable assumptions:
    - Sufficient parallelism available
    - Ignore synchronization issues
    - Constant latency values
- Future Work
  - Improve the model by incorporating synchronization issues and limited parallelism effects

8

Paper 2: Interleaving: A Multithreading Technique Targeting Multiprocessors and Workstations

(James Laudon, Anoop Gupta, Mark Horowitz)

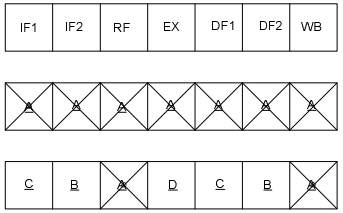
- Summary

  Architectural changes in commodity microprocessors benefiting workstations and multiprocessors with little hardware complexity

Paper 2: Interleaving: A Multithreading Technique Targeting Multiprocessors and Workstations

(James Laudon, Anoop Gupta, Mark Horowitz)

- Interleaved Multiple Context
  - Extension of fine-grained multiple-context model
  - Addition of caching and pipeline interlocks
  - Efficient support for single as well as multiple contexts



Paper 2: Interleaving: A Multithreading Technique Targeting Multiprocessors and Workstations

(James Laudon, Anoop Gupta, Mark Horowitz)

- Results

  Considerable improvement over blocked scheme in both workstation and multiprocessor environments due to low switching cost and ability to hide small latencies

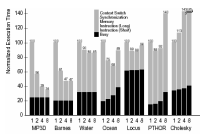


Figure 8: Application execution time breakdown for the blocked scheme.

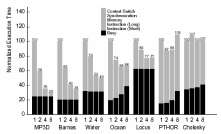


Figure 9: Application execution time breakdown for the interleaved scheme.

Paper 2: Interleaving: A Multithreading Technique Targeting Multiprocessors and Workstations

(James Laudon, Anoop Gupta, Mark Horowitz)

- Strengths
  - Performance improvement for both workstation and multiprocessor environments with the same microprocessor
- Weaknesses
  - Significant hardware complexity for caches and PCU for RISC based machines
  - Hardware complexity for Superscalars dramatically higher

Paper 2: Interleaving: A Multithreading Technique Targeting Multiprocessors and Workstations

(James Laudon, Anoop Gupta, Mark Horowitz)

- Future Work

  Extension of interleaved multiple context scheme to dynamic super-scalar processors.

Paper 3: Comparative Evaluation of Latency Reducing and Tolerating Techniques

(Anoop Gupta, John Henessy, Kourosh Gharachorloo, Todd Mowry and Wolf-Dietrich Weber)

- Summary

  Provides a consistent framework for the evaluation of the following techniques for multi-processor architectures

  - Coherent Caches
  - Memory consistency models
  - Software controlled pre-fetching
  - Multiple contexts

Paper 3: Comparative Evaluation of Latency Reducing and Tolerating Techniques

(Anoop Gupta, John Henessy, Kourosh Gharachorloo, Todd Mowry and Wolf-Dietrich Weber)

- Results Summary

  - Coherent Caches offer a substantial gain in performance
  - Relaxed Memory Consistency Model offers potential performance gains
  - Software controlled prefetching though application dependent offers gains in reads and writes
  - Multiple Context Processors also application dependent, offer little gain when combined with pre-fetching

Paper 3: Comparative Evaluation of Latency Reducing and Tolerating Techniques

(Anoop Gupta, John Henessy, Kourosh Gharachorloo, Todd Mowry and Wolf-Dietrich Weber)

- Strengths

  Various performance enhancement techniques in multi-processors considered in a systematic and consistent manner with sufficient overlapping.

- Weaknesses

  - Not enough applications to substantiate results
  - Sophisticated software control needed for pre-fetching
  - Pre-fetching and multiple contexts not considered appropriately
  - Lock-up free caches not exploited for read misses

Discussion Issues

- Usefulness
  - What are the negative impacts of multithreading? How big must a thread be and when do you want to switch threads?
  - How useful is multithreading compared to other latency tolerance techniques?
- Synchronization
  - How would we implement synchronization in multiprocessors?
  - How do we prioritize threads?

Discussion Issues (cont.)

- Relevance to CMP
  - What changes are needed to use multithreading in CMP?
  - How many threads do you need to keep a CMP with x cores busy?
- Other issues
  - Can we live without coherent caches or just L2 coherency?
  - How do we implement coherent caches?
  - What about memory consistency model?