Data Prefetching: Hiding Memory Latency

Ben Vandiver

6.911 Spring 2000

Key Ideas

- What is prefetched?
  When does the prefetch occur?
  - *Hardware driven*; hardware decides which memory addresses to prefetch based on past accesses or future instructions. (problems with lateness, inaccurate addresses, lengthening the critical path)
  - Software driven; compiler issues prefetch instructions. (problems with extra instruction overhead)

More Key Ideas

- Where does the result go?
  - Register (binding prefetch)
    problems with stale data
  - Cache (non-binding prefetch)
    problems with pollution
  - Other (FIFO queue)
    problems with critical path length, coherence

Hardware Driven Prefetch

- Caches
  - Prefetches data close to recent accesses
  - Multiprocessor environment needs coherence
    - Objective: Maintain low latency access with minimal network overhead.
    - Methods: Write-Update, Write-Invalidate, Snoopy-Reading, Random-Walk, etc..

Hardware Prefetch Engines

- Optimize Loops / Vector operations
- By knowing or guessing stride, predict upcoming accesses and prefetch.
- Trend is to have the compiler give parameters to the prefetcher hardware

Software Prefetch

- Use prefetch instruction
  - non-blocking, non-error-generating load
- DASH paper
  - Useful in NUMA environment; hides network latencies
  - 1998, paper on compiler that gets similar results
  - designed with coherency in mind

Decoupled Access/Execute

- Architecture of DAE machine
  - Access processor performs all accesses to memory and address calculation
  - Execute processor does all the “work”
  - Communicate via queues for data and branches
  - Processors run asynchronously

Decoupled Access/Execute

- Payoff
  - Access leads Execute, hiding memory latency
  - Claims speedup of 1.7 average, 2.5 max
  - Only stalls for RAW hazards and full queues
- DAE vs Caches
  - Original paper compared to a no-cache machine
  - DAE loses when latency high, can benefit from cache itself

DAE vs SS

- DAE vs SuperScalar machine:
  - DAE beats 3-way SS on 10/12 Lawrence Livermore loops.
- Why?
  - Register rename via queues
  - Out of order execution
  - Dynamic loop unrolling

Extensions to DAE

- Decouple control from data processing
- Control decoupling reduces Loss of Decoupling (LOD) events.
- More recent research (1993)
- Rumored existence of compiler for DAE

The Future of Prefetch

- Memory Latency isn’t going away
- Communicating access patterns to lower level architecture
- Intel includes data speculation in Itanium
  - Errors delivered on data use, not load
  - Schedule loads before stores (RAW avoidance)