DATA PREFETCH LAB

ADVANCED PROCESSOR ARCHITECTURE

Spring 2018

ANTONIO GONZÁLEZ
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Goal

- Design and evaluation of a data prefetcher
  - For L2 cache
- Evaluation based on its performance
  - On a private set of traces
  - Using four different processor configurations
- And its complexity/cost

Performance evaluation

- Each student will get four scores
  - Speedup for each of four processor configurations
    - Configurations provided in ANNEX 2
    - For each configuration, harmonic mean for a private set of traces
- The overall score is the sum of these four scores
- Students can evaluate their proposal
  - Using the provided traces
  - Or creating new ones

Deadlines

- Send me the code by May 1st
- Presentations around mid of May

Rules

- The prefetcher can use a maximum of 32KBytes of storage
- Functions to help students to build the prefetcher
  - l2_get_set: get the L2 set that a particular line occupies
  - l2_get_way: get the L2 way in its set (obtained by the above function) that a particular line occupies
- Cannot be used more than once for each invocation of the prefetching function

Framework http://docencia.ac.upc.edu/master/MIRI/APA

- Dcp2sim.tar file contains:
  - The simulation framework
    - Readme file explains how to compile and execute the simulator
  - Some examples of prefetchers
  - Some example of representative traces
  - A tool to create new traces
    - Readme file explains how to create new traces
  - Prefetcher.h contains the headers of provided functions

Code to be Developed

- Prefetching code should be added to skeleton.c file in example_prefetcher directory
  - Only the next functions should be modified
    - l2_prefetcher_initialize
    - l2_prefetcher_operate
      - It is called once per each L2 cache read request
      - The l2_prefetch_line function is called inside to generate prefetch requests
        - Each call prefetches a cache line
    - Prefetcher.h details the arguments of these functions
  - Any additional helper functions can be added

Dynamic HW Information to make Prefetching Decisions

- The prefetcher can use the following information to make prefetching decisions
  - The address of the current L2 read request
  - The address of the instruction that generated the current L2 read request
    - Called Instruction Pointer
  - Whether the current L2 read request was a hit or a miss in the L2
    - There is no helper function to know whether the request comes from a L1 prefetched line or not
  - The current occupancy of the L2 read queue
    - How many requests are waiting to lookup the L2: get_l2_read_queue_occupancy
  - The current occupancy of the L2 MSHRs
    - How many requests to L3 are currently outstanding: get_l2_mshr_occupancy

Other Considerations

- A prefetches fails (have no effect) if either
  - The L2 read queue is full, or
  - If the prefetch address is in a different 4KB physical page than the current L2 read request.
- Failed prefetches return a special value to indicate so

Other Considerations

- Prefetches are added to the tail of the L2 read queue, which is processed in FIFO order, and are not prioritized over demand requests
- The prefetcher is invoked ONLY on L2 demand requests, and is not invoked recursively on prefetch operations
- Prefetches may specify which level of the cache hierarchy they want the cache line to be prefetched into
  - L2 or L3
- Prefetches into the L3 have the advantage of not occupying an L2 MSHR

ANNEX 1: Description of Simulation Infrastructure

- Simple out-of-order core
- 256-entry instruction window with no scheduling restrictions (i.e., any instruction that is ready in the window can be scheduled out-of-order)
- Processor has a 6-wide pipeline. A maximum of two loads and a maximum of one store can be issued every cycle
- Perfect branch prediction (i.e., no front-end or fetch hazards).

ANNEX 1: Description of Simulation Infrastructure

- Instruction cache is not modelled
- Three-level data cache
  - L1 data cache is 16KB 8-way set-associative with LRU
  - L2 data cache is 128KB 8-way set-associative with LRU
  - L3 data cache is 16-way set-associative with LRU
    - Multiple configurations
- L2 data cache is inclusive of the L1, and the L3 data cache is inclusive of both the L1 and L2
- All instructions have one-cycle latency except for memory accesses. L1 cache hits have a 4 cycle latency. L2 cache hits have an additional 10 cycle latency. L3 cache hits have an additional 20 cycle latency

ANNEX 1: Description of Simulation Infrastructure

- Main memory is modeled in some detail (data bus contention, bank contention, write-to-read bus turnaround delays, etc.), and has a configurable amount of bandwidth
- All evaluated configurations use one 64-bit channel, which has 1 rank with 8 banks.
- Each cache has a request queue for storing pending requests to that cache. The L1 request queue is processed at a maximum rate of 2 requests/cycle. The L2 and L3 read queues are processed at a maximum rate of 1 read/cycle. There is no prioritization in these queues between prefetch and demand requests, and they are processed in FIFO order.
- The main memory read queue is processed out of order, according to a modified Open Row FR-FCFS policy. The priority is as follows (highest priority to lowest): demand row hit > prefetch row hit > demand row miss > prefetch row miss. Rows are left open after the last access to them has been processed

ANNEX 1: Description of Simulation Infrastructure

- DRAM access latency for row hits is approximately 13.5ns, and for row misses is approximately 40.5ns
- Prefetches are issued for whole cache blocks.
- The prefetchers can only see the stream of L1 cache misses / L2 cache reads (including those L2 reads caused by an L1 write miss)

ANNEX 1: Description of Simulation Infrastructure

- When a block comes into the cache, it replaces the LRU block in its set, whether it is a demand request or a prefetch request
- All cache fills from a lower level of the hierarchy/main memory must travel through a series of fill buffers between each level of the hierarchy, at a maximum rate of 1 fill/cycle. This means data will be available in the L1 at least 2 cycles after it is available in the L3.
- The L2 read queue is 32 entries. Both L1 misses and all L1 prefetch requests are inserted into this queue.

ANNEX 1: Description of Simulation Infrastructure

- There are limited numbers of request tracking registers in the L1 and L2 caches. The L1 is limited to 8 outstanding L2 requests, and the L2 is limited to 16 outstanding L3 requests
- If the request tracking registers are all full, all further misses at that level will be stalled until a request tracking register is freed
- The prefetcher is invoked on each L2 cache access, AFTER its 10-cycle L2 access latency, and after it has been determined to be an L2 hit or miss

ANNEX 2: Configurations Used for Evaluation

- Configuration 1: L3 size is 1 MB. Memory frequency is 1600 MT/s (12.8 GB/s).
- Configuration 2: L3 size is 256 KB. Memory frequency is 1600 MT/s (12.8 GB/s).
- Configuration 3: L3 size is 1 MB. Memory frequency is 400 MT/s (3.2 GB/s).
- Configuration 4: Same as Configuration 1, EXCEPT with additional randomness added to the L1 access stream.