

CMU 18-447 INTRODUCTION TO COMPUTER ARCHITECTURE, SPRING 2013
HW 7: COHERENCE, PREFETCHING, PARALLELISM AND TOPOLOGIES

Instructor: Prof. Onur Mutlu
TAs: Justin Meza, Yoongu Kim, Jason Lin

Assigned: Wed., 4/24, 2013

This homework is OPTIONAL and does not need to be turned in.

1 Prefetching

An architect is designing the prefetch engine for his machine. He first runs two applications A and B on the machine, with a stride prefetcher.

Application A:

```
uint8_t a[1000];
sum = 0;
for (i = 0; i < 1000; i += 4)
{
    sum += a[i];
}
```

Application B:

```
uint8_t a[1000];
sum = 0;
for (i = 1; i < 1000; i *= 4)
{
    sum += a[i];
}
```

i and sum are in registers, while the array a is in memory. A cache block is 4 bytes in size.

- What is the prefetch accuracy and coverage for applications A and B using a stride prefetcher. This stride prefetcher detects the stride between two consecutive memory accesses and prefetches the cache block at this stride distance from the currently accessed block.
- Suggest a prefetcher that would provide better accuracy and coverage for
 - application A?
 - application B?
- Would you suggest using runahead execution for
 - application A. Why or why not?
 - application B. Why or why not?

2 More Prefetching

You and your colleague are tasked with designing the prefetcher of a machine your company is designing. The machine has a single core, L1 and L2 caches and a DRAM memory system.

We will examine different prefetcher designs and analyze the trade-offs involved.

- For all parts of this question, we want to compute prefetch accuracy, coverage and bandwidth overhead after the prefetcher is trained and is in steady state. Therefore, **exclude the first six requests from all computations.**
- If there is a request already outstanding to a cache block, a new request for the same cache block will not be generated. The new request will be merged with the already outstanding request in the MSHRs.

- (a) You first design a stride prefetcher that observes the last three cache block requests. If there is a constant stride between the last three requests, it prefetches the next cache block using that stride.

You run an application that has the following access pattern to memory (these are cache block addresses):

A A+1 A+2 A+7 A+8 A+9 A+14 A+15 A+16 A+21 A+22 A+23 A+28 A+29 A+30...

Assume this pattern continues for a long time.

- (i) Compute the coverage of your stride prefetcher for this application.
 - (ii) Compute the accuracy of your stride prefetcher for this application.
- (b) Your colleague designs a new prefetcher that, on a cache block access, prefetches the next N cache blocks.
- (i) The coverage and accuracy of this prefetcher are 66.67% and 50% respectively for the above application. What is the value of N?
 - (ii) We define the bandwidth overhead of a prefetcher as

$$\frac{\text{Total number of cache block requests with the prefetcher}}{\text{Total number of cache block requests without the prefetcher}} \quad (1)$$

What is the bandwidth overhead of this next-N-block prefetcher for the above application?

- (c) Your colleague wants to improve the coverage of her next-N-block prefetcher further for the above application, but is willing to tolerate a bandwidth overhead of at most 2x. Is this possible? Why or why not?
- (d) What is the minimum value of N required to achieve a coverage of 100% for the above application? Remember that you should exclude the first six requests from your computations. What is the bandwidth overhead at this value of N?
- (e) You are not happy with the large bandwidth overhead required to achieve a prefetch coverage of 100% with a next-N-block prefetcher. You aim to design a prefetcher that achieves a coverage of 100% with a 1x bandwidth overhead. Propose a prefetcher design that accomplishes this goal. Be concrete and clear.

3 Markov Prefetchers vs. Runahead Execution

- (a) Provide two advantages of runahead execution over markov prefetchers.
- (b) Provide two advantages of markov prefetchers over runahead execution.
- (c) Describe one memory access pattern in which runahead execution performs better than markov prefetchers. Show pseudo-code.
- (d) Describe one memory access pattern in which runahead execution performs worse than markov prefetchers. Show pseudo-code.

4 Pre-Execution

A machine's designer is trying to use thread-based pre-execution to speed up an application A's performance. The machine has two cores. Each core has its own private L1 and L2 cache. The cores share an L3 cache and the memory. The designer tries to improve application A's performance by i) running a pruned speculative thread on a separate core and ii) running a pruned speculative thread on a separate thread context on the same core.

- (a) Give one reason why executing the speculative thread on a separate core could provide better performance than executing the speculative thread on a separate thread context on the same core.
- (b) Give one reason why executing the speculative thread on a separate thread context on the same core could provide better performance than executing the speculative thread on a separate core.

- (c) The designer finds that executing the speculative thread on a separate core provides better performance for application A. Now, the core executing the speculative thread is hit by gamma rays and a bit in its register file flips. How does this affect the correctness of application A's execution?
- (d) The designer finds a way to parallelize application A and splits its work into two threads. He expects that running the two threads on the two cores would provide better performance than running a speculative thread on one of the cores and the single-threaded version of the program on the other core, as in (c). However, he is surprised to see that the opposite is true. Why do you think this is the case?
- (e) When the two threads of application A are executing on the two cores, the second core is hit by gamma rays and a bit in its register file flips. How does this affect the correctness of application A's execution?

5 Cache Coherence

- (a) What is the advantage of the MESI protocol over the MSI cache coherence protocol?
- (b) A designer has built a directory-based cache coherent system using the MESI invalidation protocol. Under certain workloads, however, the system performs very poorly, and after a simulation run and closer examination, the designer finds that there is a constant stream of invalidation requests between four of the nodes for one particular cache block. Why does this happen?
- (c) Where and how is the problem best solved?

6 Coherence Protocols

Suppose we have a multiprocessor system with 512 processors. Each processor has a 1 Megabyte private writeback cache with 64-byte cache blocks. The main memory size is 1 Gigabyte.

- (a) If we implement a snoopy bus based MESI cache coherence protocol, how many bits of state do we need in the entire system for the coherence protocol implementation? Where do these bits reside?
- (b) If we instead implement a directory based cache coherence protocol *as we discussed in class*, how many bits of state do we need in the entire system for the coherence protocol implementation? Where do these bits reside?
- (c) Which of the above two protocols would you choose for this system? Why?

7 Parallel Speedup

You are a programmer at a large corporation, and you have been asked to parallelize an old program so that it runs faster on modern multicore processors.

- (a) You parallelize the program and discover that its speedup over the single-threaded version of the same program is significantly less than the number of processors. You find that many cache invalidations are occurring in each core's data cache. What program behavior could be causing these invalidations (in 20 words or less)?
- (b) You modify the program to fix this first performance issue. However, now you find that the program is slowed down by a global state update that must happen in only a single thread after every parallel computation. In particular, your program performs 90% of its work (measured as processor-seconds) in the parallel portion and 10% of its work in this serial portion. The parallel portion is perfectly parallelizable. What is the maximum speedup of the program if the multicore processor had an infinite number of cores?
- (c) How many processors would be required to attain a speedup of 4?
- (d) In order to execute your program with parallel and serial portions more efficiently, your corporation decides to design a custom heterogeneous processor.

- This processor will have one large core (which executes code more quickly but also takes greater die area on-chip) and multiple small cores (which execute code more slowly but also consume less area), all sharing one processor die.
 - When your program is in its parallel portion, all of its threads execute **only** on small cores.
 - When your program is in its serial portion, the one active thread executes on the large core.
 - Performance (execution speed) of a core is proportional to the square root of its area.
 - Assume that there are 16 units of die area available. A small core must take 1 unit of die area. The large core may take any number of units of die area n^2 , where n is a positive integer.
 - Assume that any area not used by the large core will be filled with small cores.
- (i) How large would you make the large core for the fastest possible execution of your program?
- (ii) What would the same program's speedup be if all 16 units of die area were used to build a homogeneous system with 16 small cores, the serial portion ran on one of the small cores, and the parallel portion ran on all 16 small cores?
- (iii) Does it make sense to use a heterogeneous system for this program which has 10% of its work in serial sections?
Why or why not?
- (e) Now you optimize the serial portion of your program and it becomes only 4% of total work (the parallel portion is the remaining 96%).
- (i) What is the best choice for the size of the large core in this case?
- (ii) What is the program's speedup for this choice of large core size?
- (iii) What would the same program's speedup be for this 4%/96% serial/parallel split if all 16 units of die area were used to build a homogeneous system with 16 small cores, the serial portion ran on one of the small cores, and the parallel portion ran on all 16 small cores?
- (iv) Does it make sense to use a heterogeneous system for this program which has 4% of its work in serial sections?
Why or why not?

8 Topologies

Suppose you would like to connect 625 processors, and you are considering three different topologies: bus, point-to-point network, and mesh.

Describe one disadvantage of each:

- (i) A Single Bus.
- (ii) A Point-to-Point Network.
- (iii) A 25x25 Mesh.

Which one would you choose? Why?