Class exercise: Understanding cache misses

COMP 40

October 3, 2011

Group

Keeper of the record:
Other group members:

Followup from Friday's lab

Last week we modeled the cache as follows:

> A block is not loaded into a cache line the until the CPU demands some address in that block.

I've graphed results from loading every byte in a 100MiB block with strides randing from 1 to 64. Look at the graphs and answer these questions:

1. Assume the line size is L. Given a stride of size k, what is the expected number of misses per load?

   Hint: Assume that address a is at the start of a cache line, and that there are L loads at the sequence of addresses $a, a+k, a+2k, \ldots a+(L-1)k$. How many lines are loaded? How many misses per load?

2. Which hardware conforms to the model?

3. Do older machines seem to have a different cache-line size from today's machines? How can you tell?

4. What is going on with the Core2 Quad Q6700?

Line splitting

For the questions below, assume that the cache-line size $L = 64$.

5. The processor requests a 32-bit word which starts at address $2^{16} + 62$. None of the relevant addresses are in the cache. How many cache lines are loaded and why?

6. A 64-bit pointer is loaded from a randomly chosen address not in the cache. On average, how many cache lines will have to be loaded?

7. A contiguous array of 80 64-bit pointers is stored at an unknown address. The whole array is in the cache, but the CPU can request at most one cache line per cycle.

   (a) Worst case, how many cache-line requests are needed to load *every* pointer in the array?

   (b) Best case, how many cache-line requests are needed to load *every* pointer in the array?

8. What, if anything, can the compiler and the standard C library do to ensure that every load and store uses the smallest possible number of cache lines?

Please return your work to the course staff.