

# Homework Set 1

**Due: Wednesday, 10/15/2008, 5pm**

**Please work in groups of 3 students**

**Instructions:** Submit to the box outside Gates 310 by the due date above. Show your work, state your assumptions, and justify your answers to receive full credit; we prefer concise, correct explanations. Check the Bulletin Board for corrections and updates.

**Homework Policies:** Collaboration on homework assignments is encouraged subject to the Honor Code and the following guidelines:

No more than three (3) people may collaborate on a homework solution.

- A group should submit a single solution.
- Any assistance received should be acknowledged in writing on the assignment.
- All students should work on all problems.

SCPD students will receive graded homeworks back through SCPD. There will be no extensions or exceptions to the submission deadline. Solutions to the HW will be posted on the class webpage a few hours after the submission deadline.

## Problem 1 (5 points)—Amdahl's Law

Your company has just bought a new system with a dual-core Pentium processor, and you have been tasked with optimizing your software for this system. You will run two applications on this dual-core Pentium, but the resource requirements are not equal. The first application needs 75% of the resources, and the other only 25% of the resources.

- a) Given that 60% of the first application is parallelizable, how much speedup would you achieve with that application if run in isolation?
- b) Given that 95% of the second application is parallelizable, how much speedup would this application observe if run in isolation?
- c) Given that 60% of the first application is parallelizable, how much *overall system speedup* would you observe if you parallelized it, but not the second application?
- d) How much overall system speedup would you achieve if you parallelized both applications, given the information in parts (a) and (b)?

## Problem 2 (5 points) – Benchmarks

Your company is trying to decide between a single-core system and a dual-core system. The dual-core system is more expensive but may lead to better performance. The figure below gives the performance on two sets of benchmarks—a memory benchmark and a processor benchmark for a range of systems with either a single-core or a dual-core chip. Note that higher numbers mean better performance. You know that your application will spend 30% of its time on memory-bound computations, and 70% of its time on processor-bound computations.

| Chip               | # of cores | Clock Frequency (MHz) | Memory Benchmark Performance | Processor Benchmark Performance |
|--------------------|------------|-----------------------|------------------------------|---------------------------------|
| Athlon 64 X2 4800+ | 2          | 2,400                 | 3,423                        | 20,718                          |
| Pentium EE 840     | 2          | 2,200                 | 3,228                        | 18,893                          |
| Pentium D 820      | 2          | 3,000                 | 3,000                        | 15,220                          |
| Athlon 64 X2 3800+ | 2          | 3,200                 | 2,941                        | 17,129                          |
| Pentium 4          | 1          | 2,800                 | 2,731                        | 7,621                           |
| Athlon 64 3000+    | 1          | 1,800                 | 2,953                        | 7,628                           |
| Pentium 4 570      | 1          | 2,800                 | 3,501                        | 11,210                          |
| Processor X        | 1          | 3,000                 | 7,000                        | 5,000                           |

- Calculate the weighted performance of the benchmarks for the Pentium 4 and Athlon 64 X2 3800+.
- How much speedup do you anticipate getting if you move from using a Pentium 4 to an Athlon 64 X2 3800+ on a memory intensive application suite?
- You are using a dual-core Athlon processor, and you are choosing between two ways to implement the same algorithm. The first is to create a large lookup table to store 4K words of data. When you need the result, you look up the answer. The second method would be to calculate the result in a very tight computational loop. What are the advantages and disadvantages of each implementation?

### Problem 3 (10 points) —Weighted Averages

You are trying to benchmark a new system and are required to produce a score: the geometric mean of normalized runtimes from a collection of three particular programs. These runtimes in seconds are shown in the following table.

Table 1: New runtimes

| Program A | Program B | Program C |
|-----------|-----------|-----------|
| 20        | 10        | 1080      |

Table 2: Old runtimes

| Program A | Program B | Program C |
|-----------|-----------|-----------|
| 24        | 16        | 972       |

Unfortunately, the fancy reference machine was destroyed and its performance numbers were lost. You recall, however, that you already calculated this benchmark on a different system. The score for the old system was 36.

- Prove that the geometric mean of ratios is equal to the ratio of geometric means for  $n \geq 1$ . You should assume strictly positive series elements and show your steps
- What is the benchmark score for your new machine?
- With the reference machine permanently broken, you wonder if it might be better to use the arithmetic mean with all weights set to the same value. What are the scores for your old and new system? Did you discover an improvement similar to what the geometric mean suggested?
- Gene Amdahl appears to you in a dream and convinces you that a benchmark should reflect his intuition about systems: optimize the common case. Rationalizing that you expect all three programs to get an equal share of the system time, you set the weights such that  $w_i t_i$ , the product of the weight and runtimes, should be equal  $V_i$ . That is, if  $t_i = 1$  and  $t_j = 10$  then the weights would be set to  $w_i = 10$  and  $w_j = 1$ . What are the smallest, integer weights you might select for your new system? Using those weights for both systems, what scores do they get? If you could reduce the runtime of a single program by one second, which would you choose?
- One of Christos's graduate students, Jacob, points out that your weighted arithmetic mean is misleading and arbitrary. Persuaded once again, you decide to recalculate the weights. Suggest a method for selecting appropriate weights, and argue in favor of it. Consider that a couple of solid reasons are more convincing than a dozen mediocre ones.

**Problem 4 (20 points)—Basic Cache Review**

1) You are given an  $N$ -way set-associative cache containing a total of  $T$  bytes of data, with a block size of  $B$  words. Assume that  $N$ ,  $T$  and  $B$  are integer powers of 2; that is  $N=2^n$ ,  $T=2^t$  and  $B=2^b$  for some non-negative integers  $n$ ,  $t$  and  $b$ . Assume 32-bit addresses, and assume that a valid bit is used (which is the only bookkeeping bit used). What is the total number of storage bits necessary to implement the cache?

2) You are building a system around a simple processor with in-order execution that runs at 2.8 GHz and has a CPI of 0.8, excluding memory stalls. The only instructions that read or write data from memory are loads (25% of all instructions) and stores (4% of all instructions). The CPI of 0.8 includes the time that load and store instructions spend in the processor pipeline (in the decode stage, etc.) but not the extra time to fetch data from the memory subsystem. This system has a split L1 caches (separate I-cache and D-cache), and a unified L2 cache.

**L1 cache:** The L1 caches have a hit time of zero; that is, a hit is serviced in the same cycle and this was already incorporated into the CPI above. The I-cache and D-cache are each direct-mapped and 32 KB. The I-cache has a 3% miss rate and 32-byte blocks. The D-cache is write-through, no write-allocate, with a 10% miss rate and 16-byte blocks. The D-cache has a write buffer that eliminates stalls for 94% of all writes

**L2 cache:** The L2 cache is 512 KB and write-back, with 64-byte blocks. Its access latency is 15 ns. It is connected to the L1 cache by a 128-bit data bus that runs at 266 MHz and can transfer one 128-bit word per bus cycle. Of all memory references sent to the L2 cache in this system, 82% are satisfied without going to main memory, 47% of all blocks replaced are dirty and need to be written back

**Main memory:** The 128-bit wide main memory has an access latency of 70 ns, after which any number of bus words may be transferred at the rate of one per bus cycle on the 128-bit, 133 MHz main memory bus.

Assume there is bus contention, miss penalties are cumulative, delivery is in address order, and the stated bus latency is the time *before* the access can be made.

- a) Calculate the average memory access time for instruction accesses.
- b) Calculate the average memory access time for data reads.
- c) Calculate the average memory access time for data writes.
- d) Calculate the overall CPI, including memory accesses.
- e) If you upgrade the CPU to a new model with a 3 GHz processor that is otherwise identical, how much faster would the new system be?

### Problem 5 (20 points) — Advanced Cache Techniques

The transpose of a matrix interchanges its rows and columns and is illustrated below:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix} \rightarrow \begin{bmatrix} A_{11} & A_{21} & A_{31} & A_{41} \\ A_{12} & A_{22} & A_{32} & A_{42} \\ A_{13} & A_{23} & A_{33} & A_{43} \\ A_{14} & A_{24} & A_{34} & A_{44} \end{bmatrix}$$

Here is a simple C loop to show the transpose for an NxN matrix:

```
for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
        output[j][i] = input[i][j];
    }
}
```

Assume both the input and output matrices are stored in the row major order (row major order means row index changes fastest). Assume you are executing a 256 x 256 double-precision transpose on a processor with a 16 KB fully associative (so you don't have to worry about cache conflicts) LRU replacement level 1 data cache with 64-byte blocks. Assume level 1 cache misses or prefetches require 16 cycles, always hit in the level 2 cache, and the level 2 cache can process a request every 2 processor cycles. Assume that each iteration of the inner loop above requires 4 cycles if the data is present in the level 1 cache. Assume the cache has a write-allocate fetch-on-write policy for write misses. Unrealistically assume writing back dirty cache blocks requires 0 cycles.

For the simple implementation given above, this access order would be non-ideal for the input matrix. However, applying a loop interchange optimization would create a non-ideal access order for the output matrix. Because loop interchange is not sufficient to improve its performance, it must be blocked instead.

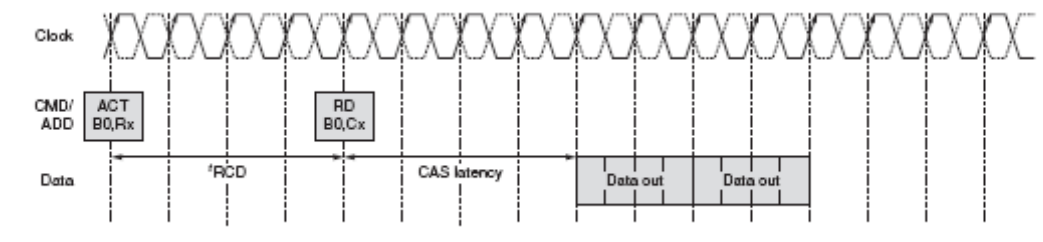
- What block size should be used to completely fill the data cache with one input and output block if the level 1 cache is fully associative 64 KB?
- What is the minimum associativity required of the level 1 caches for consistent performance independent of both arrays' position in memory?

Assume you are designing a hardware prefetcher for the unblocked matrix transposition code above. The simplest type of hardware prefetcher only prefetches sequential cache blocks after a miss. More complicated “nonunit stride” hardware prefetchers can analyze a miss reference stream, and detect and prefetch nonunit strides. In contrast, software prefetching can determine nonunit strides as easily as it can determine unit strides. Assume prefetches write directly into the cache and no *pollution* (overwriting data that needs to be used before the data that is prefetched).

c) For best performance given a nonunit stride prefetcher, in the steady state of the inner loop, how many prefetches need to be outstanding at a given time?

### Problem 6 (20 points) — Main Memory Organization

Using figure 5.14 in the textbook, you will consider the design of a variety of main memory systems. Assume a multi-core chip with eight 3 GHz cores and directly attached memory controllers (i.e. integrated northbridge) as in the case of the AMD Opteron chips. The multi-core chip contains a single shared level 2 cache, with misses from that level going straight to main memory (i.e. no level 3 cache). A sample timing diagram for DDR2 SDRAM chips appears below.  $t_{RCD}$  is the time required to activate a row in a bank, while the CAS latency (CL) is the number of cycles required to read out a column in a row. Assume that the main memory is constructed using a standard DDR2 DIMM with ECC that has 72 data lines connecting it to the northbridge. Also assume burst lengths of 8 which read out 8 bits per data line, or a total of 32B from the DIMM. Assume the DRAMs have a 1KB page size, 8 banks,  $t_{RCD} = CL \times$  Clock\_frequency, and Clock\_frequency = Transfers\_per\_second/2. The on-chip latency on a cache miss through levels 1 and 2 and back not including the DRAM access is 20 ns. Assume a DDR2-667 1 GB DIMM with CL = 5 is available for \$130 and a DDR2-533 1 GB DIMM with CL = 4 is available for \$100. (See <http://download.micron.com/pdf/technotes/ddr2/TN4702.pdf> for more details on DDR2 memory organization and timing.)



Assume the system is your desktop PC and only one core on the CMP is active. Assume there is only one memory channel.

- a) How many DRAM chips are on the DIMM if 1 Gbit DRAMs are used, and how many data I/Os must each DRAM have if only one DRAM connects to each DIMM data pin?

- b) What burst length is required to support 32B level-2 cache blocks?
- c) What is the peak bandwidth ratio between the two types of DIMMs for reads from an active page?
- d) How much time is required from the presentation of the activate command until the last requested bit of data from the DRAM transitions from valid to invalid for the DDR2-667 1 GB CL=5 DIMM?
- e) What is the relative latency when using the DDR2-667 DIMM of a read requiring a bank activate versus one to an already open page, including the time required to process the miss inside the processor?

Assume two DIMMs are used in a system, and the rest of the system costs \$800. Consider the performance of the system using the DDR2-667 and DDR2-533 DIMMs on a workload with 3.33 level 2 misses per 1K instructions, and assume all DRAM reads require an activate. Assume all 8 cores are active with the same workload.

- f) What is the cost divided by performance of the whole system when using the different DIMMs assuming only one level 2 miss is outstanding at a time and an in-order core with a CPI of 2.0 non including level 2 cache miss memory access time?

You are provisioning a server based on the system above. All 8 cores on the CMP will be busy with an overall CPI of 2.0 (assuming level 2 cache miss refills are not delayed).

- g) Given the DDR2-667 DIMMs above, how many independent memory channels should be provided so that the system is not limited by memory bandwidth if the bandwidth required is sometimes 2X the average?

**Problem 7 (20 points) — Memory Hierarchy Detective**

Do problem 5.18 from the text, using the attached graph instead of figure 5.33. Explain your answers fully. When calculating the L1 miss penalty in part C, assume an L2 hit.

**Memory access latencies for arrays of various sizes**

