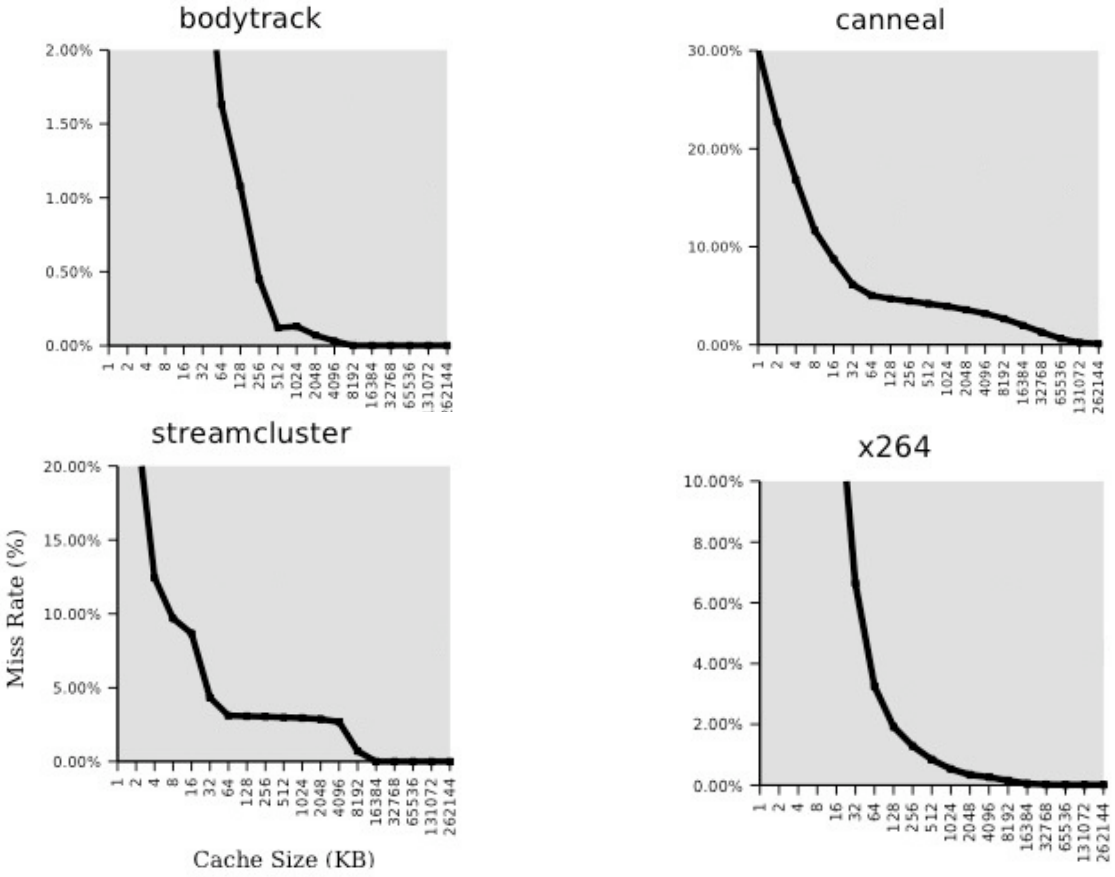
Section 5: Memory Hierarchies

TA: Scott Beamer

CS 152
Spring 2009

Problem 1: Working Set Size

Below are some plots of cache miss rates from four applications in the PARSEC benchmark suite. The cache modeled has 64B lines, is 4-way set associative, and its capacity is varied.



a) For each of the plots above, determine the application's working set size.

bodytrack - 512 KB; canneal - 64KB; streamcluster - 64KB, 16MB; x264 64KB-4MB

Things to notice:

- streamcluster displays two plateaus implying there is even a hierarchy of working sets
- canneal has a long tail, implying that for many of its accesses there is poor locality
- x264 has a gradual slope, indicating that the algorithm has been tuned to perform well on any cache size

b) Why might bodytrack's miss rate have gone up with a bigger cache?

Long running, real world applications can sometimes exhibit strange behavior. For any cache, one can devise a pathologically bad memory access stream. In this case somehow the 512KB cache ended up with the right blocks slightly more of the time than the 1024KB cache.

Problem 2: Short Answer

a) Cache A has a 90% hit rate and cache B has a 95% hit rate. Numerically, how much better is cache B?

A small 5% increase in hit rate halves the miss rate for cache B, so its nearly twice as fast when you consider the AMAT.

b) Can you have a no-write allocate, write-back cache?

Yes. On a write miss the write will go straight to memory and will not be stored in the cache.

c) If you are concerned about memory bandwidth (to DRAM), should you have a write allocate cache?

No. With write-allocate, on a write miss you will be bringing in the rest of cache block using up more bandwidth, so you will probably not allocate to save bandwidth.

Problem 3: Write Buffers

For this problem we are considering adding a write buffer to a single issue in-order CPU with one level of cache. The cache is write back, no write allocate, and for this problem you don't have to worry about lines ever getting evicted. By adding a write buffer, if a write misses in the cache, it can go directly into the write buffer rather than stalling the system for the miss.

	Quantity	Time (cycles)
read	Hit time	2
	Miss penalty	200
	Hit (after adding WB)	$3 + 2\log_2$(WB depth)
write	Hit time	4
	Miss penalty	240
	Hit/miss time (after adding WB)	6
	Cache read hit rate	98%
	Cache write hit rate	70%

a) What is the average memory access time (AMAT) for reads before and after a write buffer of depth 4 is added?

before: (hit time) + (miss rate)(miss penalty) = 2 + (1-0.98)(200) = 2 + (.02)(200) = 6 cycles

after: $(3 + 2\log_2(4)) + (.02)(200) = 11$ cycles

b) What is the AMAT for writes before adding the write buffer?

4 + (1 - 0.7)(240) = 76 cycles

c) Assume a program averages a write every 24 cycles and that they are well spread out. What is the worst average write miss rate the program can have and expect a finite sized write buffer to not overflow?

(Problem changed) For the WB, it needs (rate in) = (rate out) for it to not overflow. A write to memory takes 240 cycles, so a write should miss the cache no more than that. Thus:

$$\text{write miss interval} = \frac{\text{write interval}}{\text{write miss rate}} \quad \text{so} \quad \text{write miss rate} = \frac{\text{write interval}}{\text{write miss interval}} = 10\%$$