

CPS101
Computer Organization and Programming
Lecture 15: Cache Aware Software

Robert Wagner

cps 104 CacheSoftw.1 ©RW Fall 2000

Cache Performance

CPU time = $(\text{CPU_execution_clock_cycles} + \text{Memory_stall_clock_cycles}) \times \text{clock_cycle_time}$

Memory_stall_clock_cycles = Memory_accesses x Miss_rate x Miss_penalty

Example

- Assume every instruction takes 1 cycle
- Miss penalty = 20 cycles
- Miss rate = 10%
- 1000 total instructions, 300 memory accesses
- Memory stall cycles? CPU clocks?

cps 104 CacheSoftw.2 ©RW Fall 2000

Cache Performance

- Memory Stall cycles = $300 * 0.10 * 20 = 600$
- CPUclocks = $1000 + 600 = 1600$
- **60% slower because of cache misses!**

cps 104 CacheSoftw.3 ©RW Fall 2000

Improving Cache Performance

1. Reduce the miss rate,
2. Reduce the miss penalty, or
3. Reduce the time to hit in the cache.

Reducing Misses

◦ Classifying Misses: 3 Cs

- **Compulsory**—The first access to a block ALWAYS misses, since that block is not in the cache. Such misses are also called *cold start misses* or *first reference misses*.
(Misses in Infinite Cache)
- **Capacity**—If the cache cannot contain all the blocks needed during execution of a program, capacity misses will occur due to blocks being evicted from the cache and later retrieved.
(Misses in Size X Cache)
- **Conflict**—If the block-placement strategy is set associative or direct mapped, conflict misses (in addition to compulsory and capacity misses) will occur because a block can be discarded and later retrieved if too many blocks map to its set. These are also called *collision misses* or *interference misses*.
(Misses in N-way Associative, Size X Cache)

Cache Performance

- Your program and caches
- Can you affect performance?
- Think about 3Cs

Reducing Misses by Compiler Optimizations

- Instructions
 - Reorder procedures in memory so as to reduce misses
 - Profiling to look at conflicts
 - McFarling [1989] reduced caches misses by 75% on 8KB direct mapped cache with 4 byte blocks
- Data
 - **Merging Arrays:** improve spatial locality by single array of compound elements vs. 2 arrays
 - **Loop Interchange:** change nesting of loops to access data in order stored in memory
 - **Loop Fusion:** Combine a sequences of 2 loops that iterate the same number of times, and that use some variables in common
 - **Blocking:** Improve temporal locality by accessing "blocks" of data repeatedly vs. going down whole columns or rows

cps 104 CacheSoftw.7

©RW Fall 2000

Merging Arrays Example

```
/* Before */
int val[SZE];
int key[SZE];

/* After */
struct merge {
    int val;
    int key;
};
struct merge merged_array[SZE];
```

Reducing conflicts between val & key

cps 104 CacheSoftw.8

©RW Fall 2000

Loop Interchange Example

```
/* Before */
for (k = 0; k < 100; k = k+1)
    for (j = 0; j < 100; j = j+1)
        for (i = 0; i < 5000; i = i+1)
            x[i][j] = 2 * x[i][j];

/* After */
for (k = 0; k < 100; k = k+1)
    for (i = 0; i < 5000; i = i+1)
        for (j = 0; j < 100; j = j+1)
            x[i][j] = 2 * x[i][j];
```

Sequential accesses instead of striding through memory every 100 words

cps 104 CacheSoftw.9

©RW Fall 2000

Loop Fusion Example

```
/* Before */
for (i= 0; i < N; i= i+1)
  for (j= 0; j < N; j= j+1)
    a[i][j] = 1.b[i][j] * c[i][j];
for (i= 0; i < N; i= i+1)
  for (j= 0; j < N; j= j+1)
    d[i][j] = a[i][j] + c[i][j];
```

```
/* After */
for (i= 0; i < N; i= i+1)
  for (j= 0; j < N; j= j+1)
  {
    a[i][j] = 1.b[i][j] * c[i][j];
    d[i][j] = a[i][j] + c[i][j];
  }
```

2 misses per access to a & c vs. one miss per access

Blocking Example

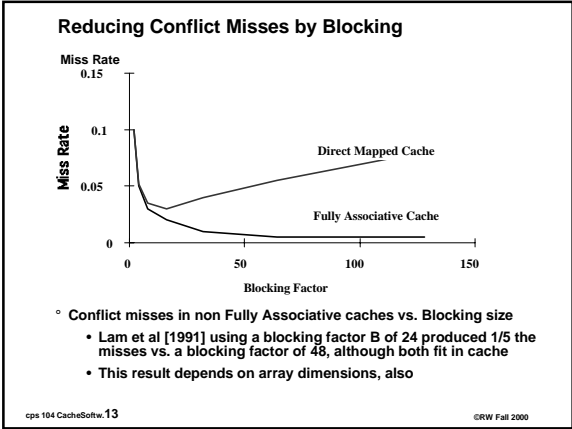
```
/* Before */
for (i= 0; i < N; i= i+1)
  for (j= 0; j < N; j= j+1)
  {
    x= 0;
    for (k= 0; k < N; k= k+1)
      x = x + y[i][k]*z[k][j];
    x[i][j] = x;
  };
```

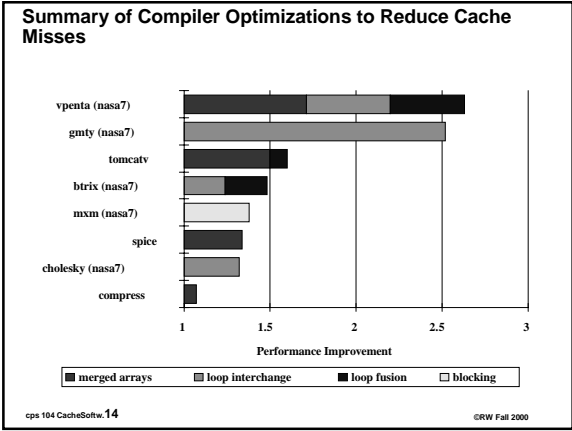
- Two Inner Loops:
 - Read all NxN elements of z[]
 - Read N elements of 1 row of y[] repeatedly
 - Write N elements of 1 row of x[]
- Capacity Misses a function of N & Cache Size:
 - 3 NxN => no capacity misses; otherwise ...
- Idea: compute on BxB submatrix that fits

Blocking Example

```
/* After */
for (jj= 0; jj < N; jj= jj+B)
  for (kk= 0; kk < N; kk= kk+B)
    for (i= 0; i < N; i= i+1)
      for (j= jj; j < min(jj+B-1,N); j= j+1)
      {
        x= 0;
        for (k= kk; k < min(kk+B-1,N); k= k+1) {
          x = x + y[i][k]*z[k][j];
        }
        x[i][j] = x;
      };
```

- Capacity Misses from $2N^3 + N^2$ to $2N^3/B + N^2$
- B called *Blocking Factor*
- Conflict Misses Too?





Summary

- Cost Effective Memory Hierarchy
- Split Instruction and Data Cache
- 4 Questions
- CPU cycles/time, Memory Stall Cycles
- Your programs and cache performance

Next Time

- Virtual Memory

cps 104 CacheSoftw.15 ©RW Fall 2000
