

Multicore Computer Architecture - Storage and Interconnects

Lecture 8

Advanced Cache Optimization Techniques-II



NPTEL Online Course : Multi-Core Computer Architecture and Interconnects Dr. John Jose, IIT Guwahati

Dr. John Jose

Assistant Professor

**Department of Computer Science & Engineering
Indian Institute of Technology Guwahati, Assam.**

Accessing Cache Memory



$$\text{Average memory access time} = \text{Hit time} + (\text{Miss rate} \times \text{Miss penalty})$$

- ❖ **Hit Time:** Time to find the block in the cache and return it to processor [*indexing, tag comparison, transfer*].
- ❖ **Miss Rate:** Fraction of cache access that result in a miss.
- ❖ **Miss Penalty:** Number of additional cycles required upon encountering a miss to fetch a block from the next level of memory hierarchy.

Merging Write Buffer

- ❖ **Write buffer merging to reduce miss penalty**
- ❖ Write buffer – Processor continue without waiting for writes to get over.
- ❖ When performing a store/write on a block that is already pending in the write buffer, update write buffer
- ❖ Reduces stalls due to full write buffer, improve buffer efficiency
- ❖ If buffer is full writes incur processor stall.

Write address	V		V		V		V	
100	1	Mem[100]	0		0		0	
108	1	Mem[108]	0		0		0	
116	1	Mem[116]	0		0		0	
124	1	Mem[124]	0		0		0	

•Each 64-bit word takes up one entry

No write buffering

Write address	V		V		V		V	
100	1	Mem[100]	1	Mem[108]	1	Mem[116]	1	Mem[124]
	0		0		0		0	
	0		0		0		0	
	0		0		0		0	

•Sequential addresses can be placed in one entry

Write buffering

Compiler Optimizations

❖ **Compiler optimization to reduce miss rate**

❖ **Loop Interchange**

- ❖ Swap nested loops to access memory in sequential order
- ❖ Maximize the use of data in cache before it is discarded

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

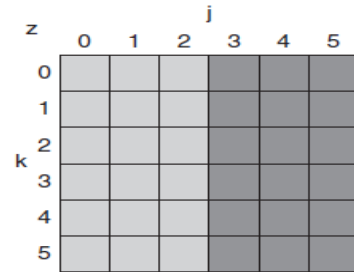
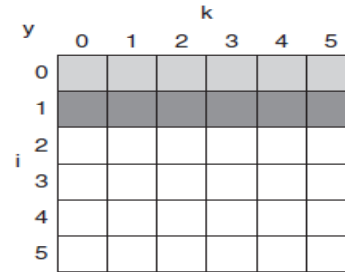
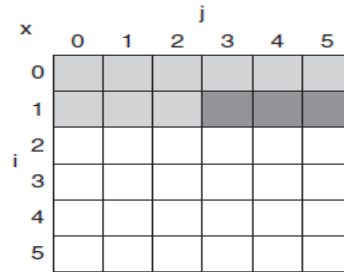
Compiler Optimizations

❖ Blocking

- ❖ Instead of accessing entire rows or columns, subdivide matrices into blocks
- ❖ Requires more accesses but improves locality of accesses

/* Before */

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



A snapshot of the three arrays **x**, **y**, and **z** when $N = 6$ and $i = 1$. The age of accesses to the array elements is indicated by shade: white means not yet touched, light means older accesses, and dark means newer accesses. Elements of **y** and **z** are read repeatedly to calculate new elements of **x**.

Compiler Optimizations

❖ Blocking

/* Before */

```
for (i = 0; i < N; i = i+1)
  for (j = 0; j < N; j = j+1)
    {r = 0;
```

```
      for (k = 0; k < N; k = k + 1)
        r = r + y[i][k]*z[k][j];
```

```
      x[i][j] = r;
```

		j					
x		0	1	2	3	4	5
	0						
	1						
i	2						
	3						
	4						
	5						

		k					
y		0	1	2	3	4	5
	0						
	1						
i	2						
	3						
	4						
	5						

		j					
z		0	1	2	3	4	5
	0						
	1						
k	2						
	3						
	4						
	5						

/* 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,N); j = j+1)
```

```
    {r = 0;
```

```
      for (k = kk; k < min(kk+B,N); k = k + 1)
```

```
        r = r + y[i][k]*z[k][j];
```

```
      x[i][j] = x[i][j] + r;
```

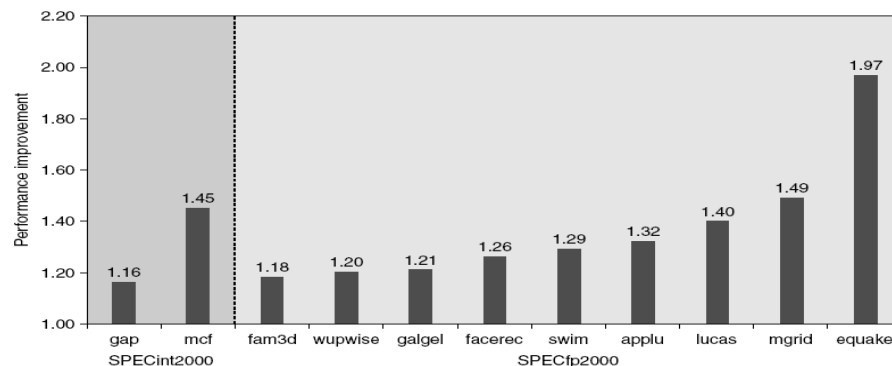
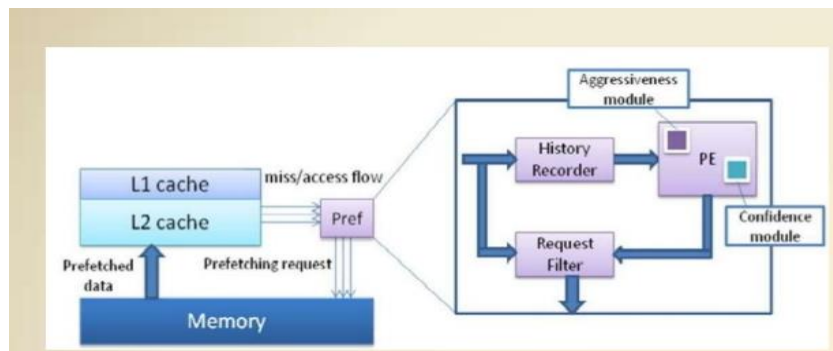
```
    };
```

NPTEL Online Course : Multi-Core Computer Architecture - Storage and Interconnects Dr. John Jose, IIT Guwahati

Hardware Prefetching

- ❖ **Pre-fetching to reduce miss rate and miss penalty.**
- ❖ Pre-fetch items before processor request them.
- ❖ Fetch more blocks on miss -include next sequential block
- ❖ Requested block is kept in I-cache and next in stream buffer.
- ❖ If a missed block is in stream buffer, cache miss is cancelled

NPTEL Online Course : Multi-Core Computer Architecture - Storage and Interconnects Dr. John Jose, IIT Guwahati



Compiler Controlled Pre-fetching

- ❖ **Pre-fetching to reduce miss rate and miss penalty.**
- ❖ Insert pre-fetch instructions before data is needed
- ❖ Pre-fetching will give performance only if processor reads from caches and executes while pre-fetching is in progress.
- ❖ Register pre-fetch
 - ❖ Loads data into register
- ❖ Cache pre-fetch
 - ❖ Loads data into cache
- ❖ Use loop unrolling and scheduling for pre-fetching data of adjacent iterations

**THANK
YOU!**

**FOR
YOUR
PRESENCE**

NPTEL Online Course : Multi-Core Computer Architecture - Storage and Interconnects Dr. John Jose, IIT Guwahati

johnjose@iitg.ac.in
<http://www.iitg.ac.in/johnjose/>