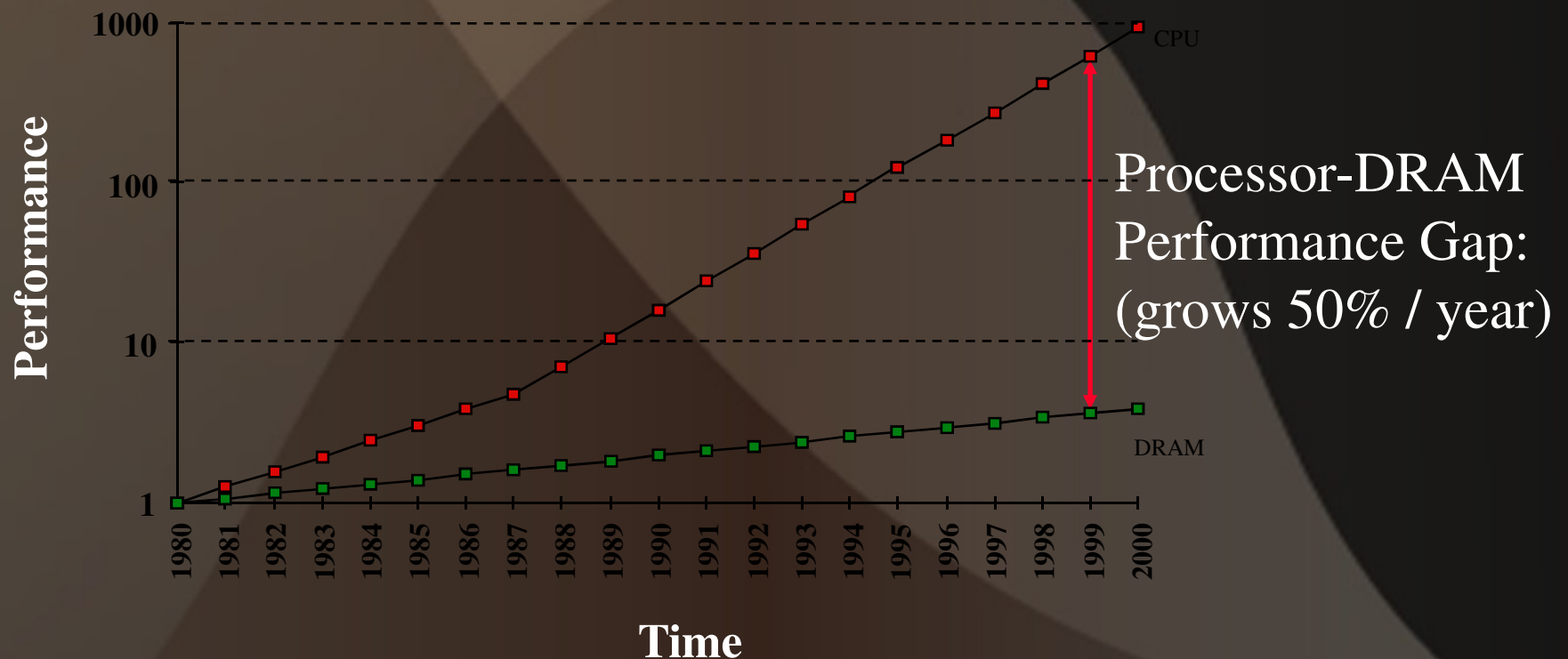


Lecture #35

An Overview of Cache Aware Compiler Optimization

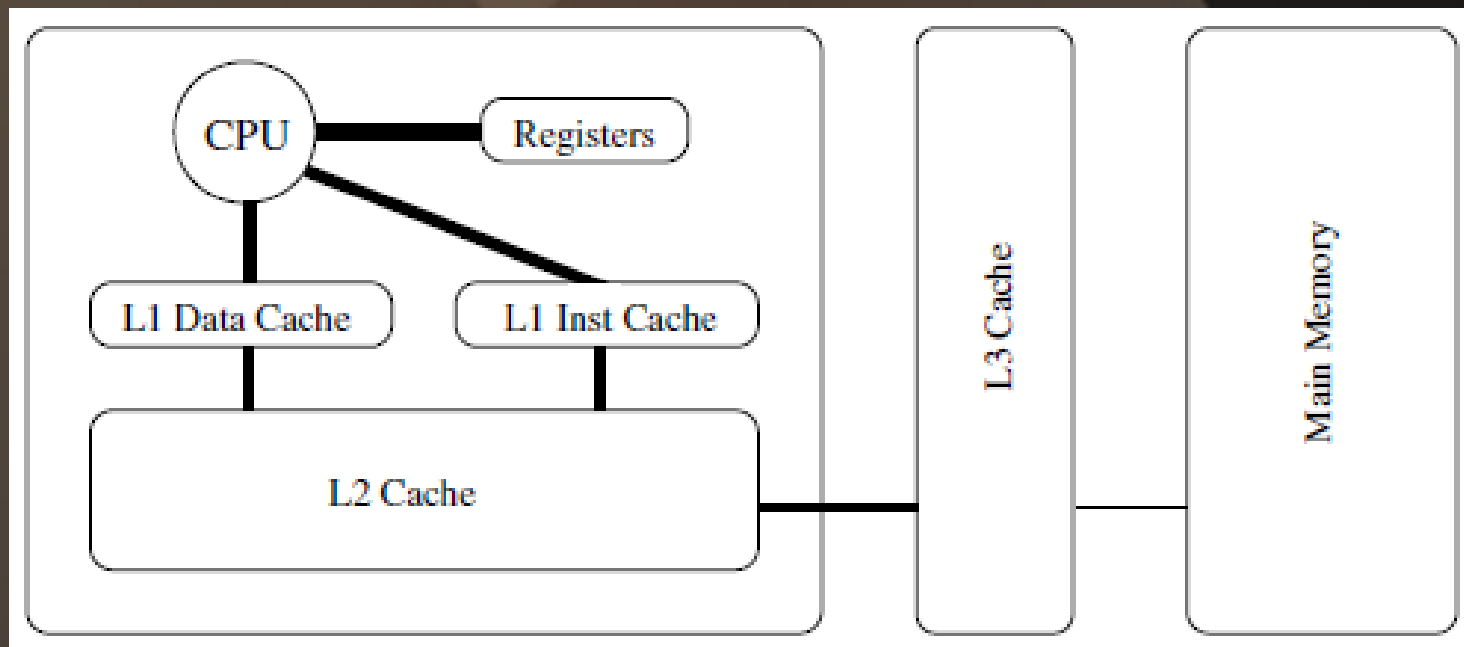
Memory Hierarchy

- Hierarchical memory Structures:
 - An attempt to mitigate the growing gap between CPU speed and memory performance
 - Low main memory bandwidth and high latency



Memory Hierarchy

- | | | |
|--------------------|---------------|----------------|
| • Registers | 1 cycle | 256-8000 bytes |
| • Cache (on-chip) | 3-10 cycles | 256k-1M |
| • Cache (off-chip) | 10-20 cycles | 1M – 16M |
| • Main memory | 20-100 cycles | 32M-4G |
| • Disk | 0.5-5M cycles | 4G-1T |



Locality of References

- Temporal Locality
 - The tendency of recently accessed data to be accessed again in the near future
- Spatial Locality
 - The tendency of data located close together in address space to be referenced close together in time

Aspects of Cache Architectures

- Cache Line
 - Holds the contents of a contiguous block of main memory
- Cache hits / misses
 - Data requested by processor is found / not found in cache line
- Memory block placement strategies:
 - Direct mapping
 - A memory block may be Placed in exactly one cache line
 - a -way set-associative mapping
 - Cache lines are grouped into sets of size a
 - Placement anywhere in corresponding set
 - Fully-associative mapping
 - A memory block can be placed into any cache line

Aspects of Cache Architectures

- Replacement strategies
 - Random
 - Chooses a random cache line for replacement
 - Least Recently Used (LRU)
 - Chooses block which has not been accessed for the longest time interval
 - Least Frequently Used (LFU)
 - First-In First-Out (FIFO)
- Measuring Cache Behaviour: Profiling
 - Hardware performance counters
 - Cache hits / misses, pipeline stalls, processor cycles, instruction issues, branch mis-predictions, etc.
 - Instrumentation
 - Insert calls to a monitoring library into the program to gather information – gprof does this

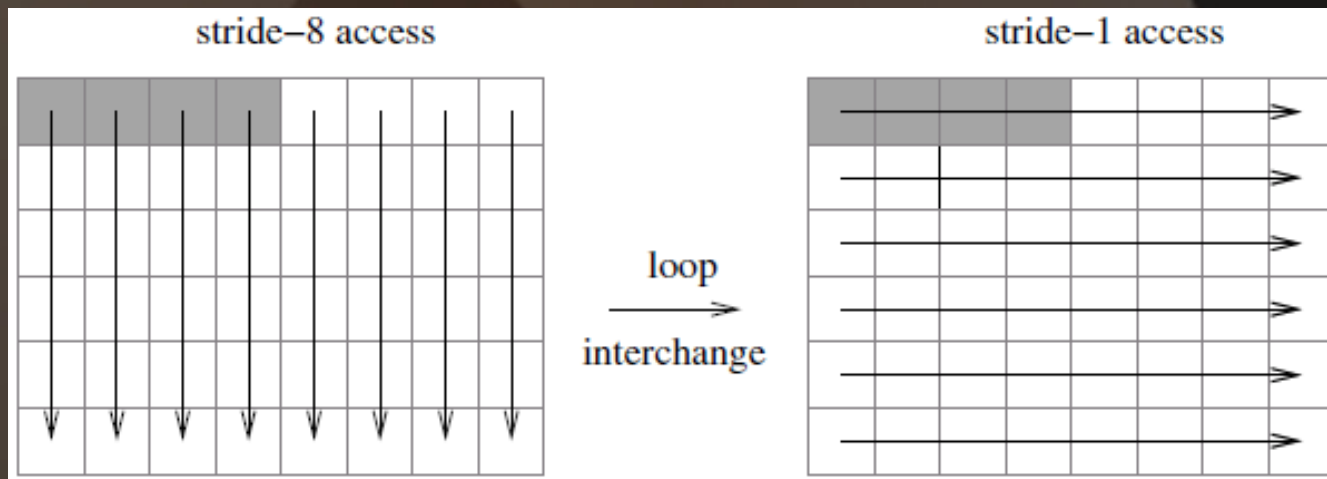
Optimizations

- Loop Interchange

Algorithm 10.3.1 Loop interchange

```
1: double sum;  
2: double a[n, n];  
3: // Original loop nest:  
4: for j = 1 to n do  
5:   for i = 1 to n do  
6:     sum += a[i, j];  
7:   end for  
8: end for
```

```
1: double sum;  
2: double a[n, n];  
3: // Interchanged loop nest:  
4: for i = 1 to n do  
5:   for j = 1 to n do  
6:     sum += a[i, j];  
7:   end for  
8: end for
```



Optimizations

- Loop Interchange: Example 2

DO J = 1, N

DO I = 1, M

D(I) = D(I) + B(I,J)

ENDDO

ENDDO

Optimizations

- Loop Fusion

Algorithm 10.3.2 Loop fusion

```
1: // Original code:  
2: for  $i = 1$  to  $n$  do  
3:    $b[i] = a[i] + 1.0$ ;  
4: end for  
5: for  $i = 1$  to  $n$  do  
6:    $c[i] = b[i] * 4.0$ ;  
7: end for
```

```
1: // After loop fusion:  
2: for  $i = 1$  to  $n$  do  
3:    $b[i] = a[i] + 1.0$ ;  
4:    $c[i] = b[i] * 4.0$ ;  
5: end for
```

Optimizations

- Loop Blocking (or Loop Tiling)

Algorithm 10.3.3 Loop blocking for matrix transposition

```
1: // Original code:  
2: for  $i = 1$  to  $n$  do  
3:   for  $j = 1$  to  $n$  do  
4:      $a[i, j] = b[j, i];$   
5:   end for  
6: end for
```

```
1: // Loop blocked code:  
2: for  $ii = 1$  to  $n$  by  $B$  do  
3:   for  $jj = 1$  to  $n$  by  $B$  do  
4:     for  $i = ii$  to  $\min(ii + B - 1, n)$  do  
5:       for  $j = jj$  to  $\min(jj + B - 1, n)$  do  
6:          $a[i, j] = b[j, i];$   
7:       end for  
8:     end for  
9:   end for  
10: end for
```

