CSCI 322
Principles of Concurrent Programming

Filip Jagodzinski
HW1 recap

Q1: Speedup ... why a pipeline of depth 6 will not produce a speedup of 6 for ANY number of instructions

(on the board explanation)
Q1: Speedup ... why a pipeline of depth 6 will not produce a speedup of 6 for ANY number of instructions

(on the board explanation)

Q2: Summation: Sum for 2 instructions their delay $4(k-1)$ where $k$ is instruction

\[
\begin{align*}
\sum_{k=0}^{1} 4k & \quad \sum_{k=0}^{2} 4k & \quad \sum_{k=0}^{2} 4(k-1) & \quad \sum_{k=0}^{1} 4(k-1) \\
\sum_{k=1}^{1} 4k & \quad \sum_{k=1}^{2} 4k & \quad \sum_{k=1}^{2} 4(k-1) & \quad \sum_{k=1}^{1} 4(k-1)
\end{align*}
\]
HW1 recap

Q1: Speedup ... why a pipeline of depth 6 will not produce a speedup of 6 for ANY number of instructions

(on the board explanation)

Q2: Summation: Sum for 2 instructions their delay $4(k-1)$ where $k$ is instruction

\[
\begin{align*}
\sum_{k=0}^{1} 4k & \quad \sum_{k=0}^{2} 4k & \quad \sum_{k=0}^{2} 4(k-1) & \quad \sum_{k=0}^{1} 4(k-1) \\
\sum_{k=1}^{1} 4k & \quad \sum_{k=1}^{2} 4k & \quad \sum_{k=1}^{2} 4(k-1) & \quad \sum_{k=1}^{1} 4(k-1)
\end{align*}
\]

Q3: Are the following two “equivalent”?

\[
\text{sum} = 0 \\
\text{for } x \text{ in range (0,2)}: \\
\text{sum} = \text{sum} + 4x \\
\sum_{k=0}^{1} 4k
\]
Q: How do we estimate the performance gain of using a computer with cache versus one without cache. Assume that the cache is a factor of \( \tau \) “faster” than memory?
Q: How do we estimate the performance gain of using a computer with cache versus one without cache. Assume that the cache is a factor of \( \tau \) "faster" than memory?

\[
\begin{align*}
\beta &= \\
T_m &= \\
T_c &= \\
T_{av} &= \\
G(\tau, \beta) &=
\end{align*}
\]
From last time ...

**Q:** How do we estimate the performance gain of using a computer with cache versus one without cache. Assume that the cache is a factor of \( \tau \) "faster" than memory?

\[ \beta = \text{cache reuse ratio; the fraction of loads or reads that can be reused from cache} \]

\[ T_m = \text{access time to main memory} \]

\[ T_c = T_m / \tau = \text{access time for cache} \]

\[ T_{av} = \beta T_c + (1 - \beta) T_m \]

\[ G(\tau, \beta) = T_m / T_{av} = \text{access performance gain} \]

\[ = \tau T_c / (\beta T_c + (1 - \beta) \tau T_c) \]

\[ = \tau / (\beta + \tau(1 - \beta)) \]
Q: How do we estimate the performance gain of using a computer with cache versus one without cache. Assume that the cache is a factor of $\tau$ “faster” than memory?

Using the above formula, a plot of different combinations of $\beta$ and $\tau$ produces the following performance gain curves.

Be able to interpret this plot and answer,
Q: For a specific $\beta$, $\tau$ combination, what change(s) must be made to $\beta$ or $\tau$ so that performance gain doubles?
From last time ...

Assume a square matrix \(( m = n )\)

```
int sumArray1(int a[m][n]){
    int i, j, sum=0;
    for (i=0; i<m, i++){
        for (j=0; j<n; j++){
            sum += a[i][j];
        }
    }
    return sum;
}
```

```
int sumArray2(int a[m][n]){
    int i, j, sum=0;
    for (j=0; j<m, j++){
        for (i=0; i<n; i++){
            sum += a[i][j];
        }
    }
    return sum;
}
```

Q: Do these two methods output the same result?

Q: Do these two methods consume the same amount of time?
From last time ...

Assume a square matrix \( m = n \)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>7</td>
<td>2</td>
</tr>
</tbody>
</table>

```java
int sumArray1(int a[m][n]){
    int i, j, sum=0;
    for (i=0; i<m, i++){
        for (j=0; j<n; j++){
            sum += a[i][j];
        }
    }
    return sum;
}
```

```java
int sumArray2(int a[m][n]){
    int i, j, sum=0;
    for (j=0; j<m, j++){
        for (i=0; i<n; i++){
            sum += a[i][j];
        }
    }
    return sum;
}
```
From last time ...

Assume a square matrix \( m = n \)

```
int sumArray1(int a[m][n]){
    int i, j, sum=0;
    for (i=0; i<m, i++){
        for (j=0; j<n; j++){
            sum += a[i][j];
        }
    }
    return sum;
}
```

```
int sumArray2(int a[m][n]){
    int i, j, sum=0;
    for (j=0; j<n, j++){
        for (i=0; i<m; i++){
            sum += a[i][j];
        }
    }
    return sum;
}
```

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>7</td>
<td>2</td>
</tr>
</tbody>
</table>

\( \text{sumArray1 : sum = ?} \)
\( \text{sumArray2 : sum = ?} \)
From last time … 

Assume a square matrix \((m = n)\)

```c
int sumArray1(int a[m][n]){
    int i, j, sum=0;
    for (i=0; i<m, i++){
        for (j=0; j<n; j++){
            sum += a[i][j];
        }
    }
    return sum;
}
```

```c
int sumArray2(int a[m][n]){
    int i, j, sum=0;
    for (j=0; j<m, j++){
        for (i=0; i<n; i++){
            sum += a[i][j];
        }
    }
    return sum;
}
```

\(\begin{array}{ccc}
  & 0 & 1 & 2 \\
\hline
0 & 1 & 3 & 6 \\
1 & 8 & 9 & 0 \\
2 & 6 & 7 & 2 \\
\end{array}\)

\[
\text{sumArray1}: \text{sum} = 1 + 3 + 6 + 8 + 9 + 0 + 6 + 7 + 2 = 42 \\
\text{sumArray2}: \text{sum} = 1 + 8 + 6 + 3 + 9 + 7 + 6 + 0 + 2 = 42 \\
\]

Same “sum”? : yes

Same “run time”? : for small \(m, n\), probably yes, for large \(m\) and \(n\), most likely not … **Why?**
From last time ...

Cache is loaded row order
If summation “order” is 1, 8, 6, -3, 3, 9, 7, -7, 6, 0, 2, and 10

Q: How many entries can be summed before an eviction needs to occur?
From last time ...

If summation “order” is 1, 3, 6, 8, 9, 0, 6, 7, 2, -3, -7, 10

Q: How many entries can be summed before an eviction needs to occur?
Andrews, Chapter 1

Multi cache, multi CPU architecture
Parallelism
Matrix Multiplication

If we can parallelize code (threads) should we?
Why or why not?
Task: Draw the components of a single CPU (with or without multiple pipelines (superscalar)), and discuss bottlenecks of moving around data
Already at this level of “simple” complexity, there are data and structural hazards that may slow down the execution of a program with multiple instruction threads.
Multi cache, multi CPU architecture

Already at this level of “simple” complexity, there are data and structural hazards that may slow down the execution of a program with multiple instruction threads.

Q: How many CPUs do today’s computers have?

If more than 1, then how are these architecture components connected?

Come up with 2 examples of component topologies with 2 CPUs, 2 cache, and 2 memories each

(on the board discussion)
Multi cache, multi CPU architecture

Topology 1
- Memory
  - Interconnection network
    - Cache
    - CPU

Topology 2
- Memory
  - Interconnection network
    - Cache
    - CPU
Multi cache, multi CPU architecture

Task: Be able to discuss the pros and cons of such topologies for the memory, cache, and CPU components

<table>
<thead>
<tr>
<th></th>
<th>Pro</th>
<th>Con</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topology 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Topology 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(on the board list)
Multi cache, multi CPU architecture

Task: Be able to discuss the pros and cons of such topologies for the memory, cache, and CPU components.

<table>
<thead>
<tr>
<th></th>
<th>Pro</th>
<th>Con</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topology 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Topology 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Memory (in)consistency
(on the board discussion)
Multi cache, multi CPU architecture

If it is so “risky” using memory data that is shared among multiple threads/processors, what sort of computation problems are “safe” for use among shared memory data?

Topology 1

Topology 2
Matrix Multiplication

For two matrices A, B, with dimensions m=n=2, what is the product matrix \( A \times B \)?

\[
\begin{array}{cc}
A & B \\
\begin{array}{cc}
2 & 3 \\
4 & 5 \\
\end{array} & \begin{array}{cc}
6 & 7 \\
8 & 9 \\
\end{array}
\end{array}
= \begin{array}{cc}
\hfil & \\
\hfil & \\
\hfil & \\
\hfil & \\
\end{array}
\]

How is the matrix product computed?
Matrix Multiplication

For two matrices A, B, with dimensions m=n=2, what is the product matrix A x B?

\[
\begin{array}{cc}
2 & 3 \\
4 & 5 \\
\end{array}
\times
\begin{array}{cc}
6 & 7 \\
8 & 9 \\
\end{array}
= \begin{array}{c}
? \\
? \\
\end{array}
\]

How is the matrix product computed?
Matrix Multiplication

For two matrices $A$, $B$, with dimensions $m=n=2$, what is the product matrix $A \times B$?

$$2 \times 6 + 3 \times 8 = 12 + 24 = 36$$
Matrix Multiplication

For two matrices A, B, with dimensions m=n=2, what is the product matrix A x B?

$$\begin{array}{cc}
A & B \\
2 & 3 & 6 & 7 \\
4 & 5 & 8 & 9 \\
\end{array}$$

$$\begin{array}{cc}
\times & = \\
& \\
\end{array}$$

$$\begin{array}{cc}
36 & ? \\
\end{array}$$
Matrix Multiplication

For two matrices $A$, $B$, with dimensions $m=n=2$, what is the product matrix $A \times B$?

\[
\begin{array}{cc}
2 & 3 \\
4 & 5 \\
\end{array}
\times
\begin{array}{cc}
6 & 7 \\
8 & 9 \\
\end{array}
= 
\begin{array}{cc}
36 & ? \\
\end{array}
\]

This is often referred to as the “pivot” which is the entry that is in “common” among both “input” row and column.

Use the provided worksheet to compute by-hand the product of these two matrices.
Matrix Multiplication

For two matrices $A$, $B$, with dimensions $m=n=2$, what is the product matrix $A \times B$?

$$
egin{array}{cc}
A & B \\
\begin{bmatrix}
2 & 3 \\
4 & 5 \\
\end{bmatrix} & \begin{bmatrix}
6 & 7 \\
8 & 9 \\
\end{bmatrix} \\
\end{array}
= \begin{bmatrix}
36 & 41 \\
64 & 73 \\
\end{bmatrix}
$$

Use the provided worksheet to write the pseudocode for calculating the product of two square matrices (do not use threads)
Matrix Multiplication

```c
double a[n,n], b[n,n], c[n,n];

for [i = 0 to n-1] {
    for [j = 0 to n-1] {
        # compute inner product of a[i,*] and b[*,j]
        c[i,j] = 0.0;
        for [k = 0 to n-1]
            c[i,j] = c[i,j] + a[i,k]*b[k,j];
    }
}

Q: Why is this approach embarrassingly parallelizable?
Q: How can we speed-up this program?

Observation/question : What feature of this program permits us to break up the problem into smaller manageable chunks?
Matrix Multiplication

```
co [i = 0 to n-1] {  # compute rows in parallel
  for [j = 0 to n-1] {
    c[i,j] = 0.0;
    for [k = 0 to n-1]
      c[i,j] = c[i,j] + a[i,k]*b[k,j];
  }
}
```

Q: What does the “co” refer to?
Q: Why is this approach “better”?
Q: Why is this approach possible?
Matrix Multiplication

Q: If we can parallelize the row computations, can we do the same for columns? Or for both?
Matrix Multiplication

Q: Can we run the inner-most loop (iterating from k=0 to k=n-1) concurrently? Why or why not?

```c
co [i = 0 to n-1, j = 0 to n-1] { # all rows and
c[i,j] = 0.0; # all columns
for [k = 0 to n-1]
    c[i,j] = c[i,j] + a[i,k]*b[k,j];
}
```
Matrix Multiplication

Q: Can we run the inner-most loop (iterating from k=0 to k=n-1) concurrently? Why or why not?

Q: What is the purpose of the $c[i,j] = 0.0;$ portion of the code?
Matrix Multiplication

```c
co [i = 0 to n-1, j = 0 to n-1] { # all rows and
c[i,j] = 0.0; # all columns
for [k = 0 to n-1]
   c[i,j] = c[i,j] + a[i,k]*b[k,j];
}
```

![Diagram of matrix multiplication process]

- **a, b, c**: Variables used in the matrix multiplication process.
- **Memory**: Stores intermediate results.
- **Interconnection network**: Facilitates data exchange between different parts of the system.
- **Cache**: Temporarily stores data for quick access.
- **CPU**: Performs calculations on the data stored in cache.
Matrix Multiplication

```c
co [i = 0 to n-1, j = 0 to n-1] { # all rows and
c[i,j] = 0.0; # all columns
    for [k = 0 to n-1]
        c[i,j] = c[i,j] + a[i,k]*b[k,j];
}
```

Another often-used terminology is to specify a process : runs in the background

```c
process row[i = 0 to n-1] { # rows in parallel
    for [j = 0 to n-1] {
        c[i,j] = 0.0;
        for [k = 0 to n-1]
            c[i,j] = c[ij] + a[i,k]*b[k,j];
    }
}
```
Shared Variable Programming