Announcements - Lab

Last week’s lab

- C, threads
- No “coordination” among threads

```
Start
  ▼
  ▼
thread_1 → thread_2 → thread_3 → End
```
Announcements - Lab

Last week’s lab

• C, threads
• No “coordination” among threads

This week’s lab

• C++, asynchronous execution
• “collecting” the outputs of multiple threads
Announcements - Lab

Last week’s lab

- C, threads
- No “coordination” among threads

This week’s lab

- C++, asynchronous execution
- “collecting” the outputs of multiple threads
- Python, semaphores
Motivation: How might you use threads to speed up summing the entries of a large array?
Motivation: How might you use threads to speed up summing the entries of a large array?
Motivation: How might you use threads to speed up summing the entries of a large array?
Motivation: How might you use threads to speed up summing the entries of a large array?

```cpp
// array dimension
int dim = 1000;

// create a 3d array up of pointers pointer pointers to doubles
double ***my3DArray = new double***[dim];
for (int i=0; i<dim; i++){
    my3DArray[i] = new double*[dim];
    for (int j=0; j<dim; j++){
        my3DArray[i][j] = new double[dim];
        for (int k=0; k<dim; k++){
            my3DArray[i][j][k] = 3.6;
        }
    }
}

// A function, myFunction, that receives two arguments,
// one of which is a pointer to a 3D array, the other
// an int, and which returns a long double
long double myFunction(double*** a3DArray, int anInt){
    long double aValue;
    // calculations magic done here
    return aValue;
}
```
Announcements - Lab

Motivation: How might you use threads to speed up summing the entries of a large array?

1. Specify the dimension for your 3D array.
2. Specify how many asynchronous threads you issued and explain WHY.
3. Provide a screen shot of the output of your program (as for example above).
4. Discuss/explain why having two threads does not necessarily reduce the execution time (summation) by half. If the speedup of using threads is not as expected, speculate why that may be the case.
Task 1: Use semaphores to restrict access to the shared variable count so that only one Thread may be accessing that variable.

Thread A

\[
\text{count} = \text{count} - 1
\]

Thread B

\[
\text{count} = \text{count} + 1
\]

Thread C

\[
\text{count} = 43
\]
From last time ...

Task 1: Use semaphores to restrict access to the shared variable `count` so that only one Thread may be accessing that variable.

<table>
<thead>
<tr>
<th>Thread A</th>
<th>Thread B</th>
<th>Thread C</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mutex.wait()</code></td>
<td><code>mutex.wait()</code></td>
<td><code>mutex.wait()</code></td>
</tr>
<tr>
<td><code>count = count - 1</code></td>
<td><code>count = count + 1</code></td>
<td><code>count = 43</code></td>
</tr>
<tr>
<td><code>mutex.signal()</code></td>
<td><code>mutex.signal()</code></td>
<td><code>mutex.signal()</code></td>
</tr>
</tbody>
</table>

`mutex = Semaphore(1)`

In this scenario, at most one Thread will proceed past its `wait()` method.
Task 2: Use semaphores to restrict access to the shared variable count so that at most two Threads may be accessing that variable

Thread A: count = count - 1
Thread B: count = count + 1
Thread C: count = 43
Task 2: Use semaphores to restrict access to the shared variable count so that at most two Threads may be accessing that variable.

Thread A
- `multi.wait()`
- `count = count - 1`
- `multi.signal()`

Thread B
- `multi.wait()`
- `count = count + 1`
- `multi.signal()`

Thread C
- `multi.wait()`
- `count = 43`
- `multi.signal()`

In this scenario, at most two Threads will proceed past the `wait()` method.
From last time ...

Q: If there is more than one “correct” instruction execution history, does it matter which one we use?
From last time ...

Q: If there is more than one “correct” instruction execution history, does it matter which one we use?

Q: What specs about the computer and/or program are needed?

Q: Is cache that is twice as fast result in a system that is twice as good? (the salesman pitch)
From last time ...

Q: How do we estimate the performance gain of a cache that is a factor of $\tau$ “faster” than memory?
From last time ...

Q: How do we estimate the performance gain of a cache that is a factor of $\tau$ “faster” than memory?

$\beta$ = cache reuse ratio; the fraction of loads or reads that can be reused from cache

$T_m$ = access time to main memory

$T_c$ = $T_m / \tau$ = access time for cache

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

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

= $\tau T_c / (\beta T_c + (1-\beta)\tau T_c)$

= $\tau / (\beta + \tau (1-\beta))$
Cache Friendly Code
Multi-cache, Multi-CPU architectures
Matrix Multiplication
Parallelizable Code
Modern Processors - Cache

\[ G(\tau, \beta) = \frac{\tau}{\beta + \tau(1-\beta)} \]

\( \beta = \text{cache reuse ratio} \)

\( \tau = \frac{T_m}{T_c} \)

Assume a program with cache reuse ratio of 0.6, and a computer with 60ns cache access time and 120ns access time to main memory.

Q: What performance improvement can you expect if you upgrade your cache so that it has a 30ns access time?

In class exercise
Q: When \( \tau \) is increased from 5 to 10, how is performance improved?
Q: Is there a scenario when cache does not have a positive effect on performance?
Modern Processors - Cache

Q: Is there a scenario when cache does not have a positive effect on performance?

Cache improves performance ONLY if data access has locality of reference: data items are used from cache before they are evicted.

Q: What are the 2 main types of locality of reference?
Modern Processors - Cache

Q: Is there a scenario when cache does not have a positive effect on performance?

Cache improves performance ONLY if data access has **locality of reference**: data items are used from cache before they are evicted.

**Temporal locality:**

**Spatial locality:**
Q: Is there a scenario when cache does not have a positive effect on performance?

Cache improves performance ONLY if data access has **locality of reference**: data items are used from cache before they are evicted.

**Temporal locality**: reuse of cache memory location; the concept that a resource that is referenced at time $x$ will be referenced soon in the near future.

**Spatial locality**: if a cache memory location is used, then it is likely that a nearby memory location will be referenced in the near future.
Q: Is there a scenario when cache does not have a positive effect on performance?

Cache improves performance ONLY if data access has **locality of reference**: data items are used from cache before they are evicted.

**Temporal locality**: reuse of cache memory location; the concept that a resource that is referenced at time $x$ will be referenced soon in the near future.

**Spatial locality**: if a cache memory location is used, then it is likely that a nearby memory location will be referenced in the near future.

**Spatial locality** is concerned with how data is stored in cache.

Q: Is data always stored “the same way” in cache? Is it OS specific? Language specific?
Example: Matrix summation

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>...</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Q: What is the sum of the entries of the matrix? Assume $M >> N$

Task: What is possible pseudocode?
Example: Matrix summation

<table>
<thead>
<tr>
<th>row</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>...</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Matrix:

Cache:
Modern Processors - Cache

Example: Matrix summation

Matrix stored by rows

Cache:
Example: Matrix summation

Matrix stored by rows

Cache:
Modern Processors - Cache

Example: Matrix summation

Matrix stored by rows

Cache:
Modern Processors - Cache

Example: Matrix summation

<table>
<thead>
<tr>
<th></th>
<th>column</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

Matrix stored by rows

<table>
<thead>
<tr>
<th></th>
<th>Block 0</th>
<th>Block 1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cache:

<table>
<thead>
<tr>
<th></th>
<th>Block 0</th>
<th>Block 1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Example: Matrix summation**

<table>
<thead>
<tr>
<th></th>
<th>column</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>row</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Block 0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
</tr>
</tbody>
</table>

**Matrix stored by rows**

<table>
<thead>
<tr>
<th></th>
<th>Block 0</th>
<th>Block 1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
Example: Matrix summation

Matrix stored by rows

Cache:
Example: Matrix summation

Matrix data items are placed into cache in rows, and some Block data items from the Matrix span multiple rows in the cache.

All columns of some rows, but not all columns of all rows, fit into cache.

Q: What are the consequence of these facts?
Modern Processors - Cache

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;
}
```

Q: What is different between these two pieces of code?

(on the board discussion)
For each row, sum the columns

For each column, sum the rows

Q: Do these two methods output the same result?
Q: Do these two methods consume the same amount of time?
Modern Processors - Cache

Assume a square matrix (m = n)

For each row, sum the columns

For each column, sum the rows

Q: Do these two methods output the same result?

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

```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;
}
```
Modern Processors - Cache

Assume a square matrix (m = n)

For each row, sum the columns

For each column, sum the rows

Using sumArray1, you sum the rows of the array, each of which is “small enough” to fit into cache, so for each a[i], the [j] component is in cache, and does not need to be fetched from memory. For the completion of the inner-most for loop, a cache miss is NOT guaranteed.
For each row, sum the columns

For each column, sum the rows

Using `sumArray2`, you sum the **columns** of each array, but ALL of the columns for each row are NOT stored in cache, because cache stores arrays “by row,” and is not big enough to hold a large array. For the completion of the inner-most for loop, there is a GUARANTEED cache miss.
Multi cache, multi CPU architecture

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

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)

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

Topology 1

Topology 2
Multi cache, multi CPU architecture

Task: Discuss the pros and cons of such topologies for the memory, cache, and CPU components

<table>
<thead>
<tr>
<th>Topology</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: Discuss the pros and cons of such topologies for the memory, cache, and CPU components

<table>
<thead>
<tr>
<th>Topology</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)
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?
For two matrices $A$, $B$, with dimensions $m=n=2$, what is the product matrix $A \times B$?

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

How is the matrix product computed?
Matrix Multiplication

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

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

How is the matrix product computed?
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 $A \times B$?

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

\[
\begin{array}{cc}
2 & 3 \\
4 & 5
\end{array}
\times
\begin{array}{cc}
6 & 7 \\
8 & 9
\end{array}
= 
\begin{array}{cc}
36 & 41 \\
64 & 73
\end{array}
\]

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

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?
Q: If we can parallelize the row computations, can we concurrently “compute” the summation that involves the columns?
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];
}
```
Matrix Multiplication

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

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

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

```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[i,j] + a[i,k]*b[k,j];
    }
}
```
When can code be parallelized

```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: Which portions of the code can be parallelized (for the time being think of this as “separate threads”) and which portion(s) of it cannot?
When can code be parallelized

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];
    }
}
When can code be parallelized

Q: When the code is NOT parallelized, which CPU performs all of the calculations?
When can code be parallelized

```java
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];
    }
}
```

When the code is NOT parallelized, the entire calculation is performed by a single CPU

Q: Where do the a, b, and c arrays reside, and what is their lifespan?
When can code be parallelized

```
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];
    }
}
```

The a and b arrays reside in memory, along with the c array. But the c array, unlike the a and b arrays, is updated during each iteration of the j loop.

Q: Are the values of a and b ever updated?
When can code be parallelized

```java
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[*],
        c[i,j] = 0.0;
        for [k = 0 to n-1]
            c[i,j] = c[i,j] + a[i,k]*b[k,j];
    }
}
```

When the code is NOT parallelized, the entire calculation is performed by a single CPU

Claim: Two operations can be executed in parallel if they are independent (be sure you can discuss/reason about WHY this is true)
When can code be parallelized

```java
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];
    }
}
```

Task: Parallelize as “much” of this code as is possible

- \( i = 0, 1, 2, \ldots n-1 \)
- \( j = 0, 1, 2, \ldots n-1 \)
- \( k = 0, 1, 2, \ldots n-1 \)
When can code be parallelized

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

for [i = 0 to n-1] {
    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];
    }
}
```

Task: Parallelize as “much” of this code as is possible

Q: If you assume that you have n processors/CPUs, then how can this code be parallelized?
When can code be parallelized

```
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];
    }
}
```

Task: Parallelize as “much” of this code as is possible

Notice that there are $n$ iterations for the $i$ loop

$i = 0, 1, 2, \ldots n-1$

$j = 0, 1, 2, \ldots n-1$

$k = 0, 1, 2, \ldots n-1$
When can code be parallelized

```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];
    }
}
```

Task: Parallelize as “much” of this code as is possible

Notice that there are n iterations for the i loop
- Iteration $i=0$ has inner loops $j=[0, n-1]$ and $k=[0, n-1]
When can code be parallelized

```
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];
    }
}
```

Task: Parallelize as “much” of this code as is possible

Notice that there are n iterations for the i loop
- Iteration i=0 has inner loops j=[0, n-1] and k=[0,n-1]
- Iteration i=1 has inner loops j=[0, n-1] and k=[0,n-1]
When can code be parallelized

Task: Parallelize as “much” of this code as is possible

Notice that there are n iterations for the i loop
- Iteration \( i=0 \) has inner loops \( j=[0, n-1] \) and \( k=[0, n-1] \)
- Iteration \( i=1 \) has inner loops \( j=[0, n-1] \) and \( k=[0, n-1] \)
- Iteration \( i=n-1 \) has inner loops \( j=[0, n-1] \) and \( k=[0, n-1] \)
When can code be parallelized

The non parallelized version works because \( c[i, j] \) is “reset” (has scope) during the execution of each \( j \) loop.

```
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];
    }
}
```

Task: Parallelize as “much” of this code as is possible.
When can code be parallelized

The first “parallelization” approach is to give each of the $n$ processors a separate iteration of the outermost $i$ loop

Q: Why/how is this possible?
When can code be parallelized

```java
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];
    }
}
```

Task: Parallelize as “much” of this code as is possible

Each thread is “responsible” for a unique i.

There is no way that thread i can be overriding an entry of c that is needed by another thread
When can code be parallelized

```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];
    }
}
```

Task: Parallelize as “much” of this code as is possible

```
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];
    }
}
```
When can code be parallelized

```java
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: If numCPUs < n, can parallelization still be performed? If so, then how?

(in class exercise, worksheet)
When can code be parallelized

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];
    }
}

---

### Example Table

<table>
<thead>
<tr>
<th>m</th>
<th>P0</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>P9</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0,4,8</td>
<td>1,5,9</td>
<td>2,6,3,7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0,5</td>
<td>1,6</td>
<td>2,7,3,8,4,9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0,6</td>
<td>1,7</td>
<td>2,8,3,9,4,5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1,2</td>
<td>3,4,5,6,7,8,9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
When can code be parallelized

```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: Are there other ways to dole out the iterations of i to multiple processors?  
If so, is any one better than another?
When can code be parallelized

Here's the code snippet:

```c
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: Can we parallelize this code further?

If yes, do we need fewer, more, or the same number of processors?
When can code be parallelized

Q: Can we parallelize this code further?

Assume you have $n^2$ processors

Q: Which portions (iterations of loops) should each processor receive?

```c
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];
    }
}
```
When can code be parallelized

Q: Can we parallelize this code further?

Assume you have $n^2$ processors
When can code be parallelized

Q: Can we parallelize this code further?

Assume you have $n^2$ processors.
When can code be parallelized

Q: Can we parallelize this code further?

Assume you have $n^2$ processors

The second “parallelization” approach is to give each of the $n^2$ processors a separate iteration of the outermost $i$ and second $j$ loop combinations.

Q: Why/how is this possible?
When can code be parallelized

Q: Can we parallelize this code further?

Assume you have $n^2$ processors

Each thread is “responsible” for a unique $i$, $j$ pair.

There is no way that a thread can be overriding a value of $c$ that is needed by another thread.
When can code be parallelized

Q: Can we parallelize this code further?

Assume you have $n^2$ processors
When can code be parallelized

Q: Can we parallelize this code further?

Q: And if you have numCPU < n^2 processors, how do you dole out processes?
When can code be parallelized

Q: And if you have numCPU < n² processors, how do you dole out processes?
When can code be parallelized

Q: And if you have numCPU < n^2 processors, how do you dole out processes?
When can code be parallelized

<table>
<thead>
<tr>
<th>Thread_0</th>
<th>CPU_0</th>
<th>CPU_1</th>
<th>CPU_2</th>
<th>…</th>
<th>CPU_9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread_0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Thread_9</td>
</tr>
<tr>
<td>Thread_10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Thread_19</td>
</tr>
<tr>
<td>Thread_20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Q: What factors affect how many processes are assigned to a CPU?

Q: What is the max or min number of processes each CPU receives, assuming an even distribution of threads among the CPUs?

Q: And if you have numCPU < n² processors, how do you dole out processes?
When can code be parallelized

Assume we live in a perfect world, and we have a seemingly infinite number of processors ... in this case we have $n \times n \times n$ processors .... then ...

Q: Can we parallelize this code further?
Q: Why or why not?

On the board explanation
When can code be parallelized

Goal: we want a systematic algorithmic approach to determine which portions of our code can be threaded, and which cannot
When can code be parallelized

Read set :
Write set :
When can code be parallelized

**Read set**: set of variables read by a process

**Write set**: set of variables written to by a process
When can code be parallelized

**Read set**: set of variables read by a process

**Write set**: set of variables written to by a process

Variables “in here.”

This does not mean that a process/thread cannot have its own (private) variables that have scope only to that thread

Separate processes, $P_a, P_b$, on separate CPUs
When can code be parallelized

**Read set**: set of variables read by a process

**Write set**: set of variables written to by a process

Separate processes, $P_a, P_b$, on separate CPUs
When can code be parallelized

**Read set**: set of variables read by a process

**Write set**: set of variables written to by a process

Separate processes, \( P_a, P_b \), on separate CPUs
When can code be parallelized

Two processes are independent if the write set of each is disjoint from both the read and write sets of the other.

Q: What does disjoint sets mean?
When can code be parallelized

Two processes are **independent** if the write set of each is **disjoint** from both the read and write sets of the other.

**Q:** What does disjoint sets mean?

```plaintext
A =  
  1 3  
  5

B =  
  4 9  
  2
```

Sets A and B are disjoint, because they have no elements in common.
When can code be parallelized

Two processes are **independent** if the write set of each is **disjoint** from both the read and write sets of the other.

**Q:** What does disjoint sets mean?

**A:**

Sets A and B are disjoint, because they have no elements in common.

Are sets M and N disjoint?
Two processes are **independent** if the write set of each is **disjoint** from both the read and write sets of the other.

**Q:** What does disjoint sets mean?

**Sets A and B are disjoint, because they have no elements in common.**

**Are sets M and N disjoint?**

No because they have the element 6 in common.
When can code be parallelized

Two processes are **independent** if the write set of each is **disjoint** from both the read and write sets of the other.

Task: Explain in your own words (intuitively) what it means for two processes to be independent.
Two processes are independent if the write set of each is disjoint from both the read and write sets of the other.

Task: Explain in your own words (intuitively) what it means for two processes to be independent.

It is safe for two or more processes to read variables that do not change. However, it is unsafe for two processes to write into the same variable, or for one process to read a variable that the other writes into.
When can code be parallelized

Scenario 1: Process $P_a$ reads from $a$, reads from $d$, and writes to $d$, and process $P_b$ reads from $b$

Q: Can we safely run both processes concurrently?
When can code be parallelized

Scenario 1: Process $P_a$ reads from $a$, reads from $d$, and writes to $d$, and process $P_b$ reads from $b$

Q: Can we safely run both processes concurrently?

First, start with an architecture understanding of processes, memory locations, cache, etc.
When can code be parallelized

Scenario 1: Process $P_a$ reads from $a$, reads from $d$, and writes to $d$, and process $P_b$ reads from $b$.

Q: Can we safely run both processes concurrently?

Task: Draw arrows to specify how $P_a$ interacts with items in main memory.
When can code be parallelized

Scenario 1: Process $P_a$ reads from $a$, reads from $d$, and writes to $d$, and process $P_b$ reads from $b$

Q: Can we safely run both processes concurrently?

Task: Draw arrows to specify how $P_a$ interacts with items in main memory.
When can code be parallelized

Scenario 1: Process $P_a$ reads from $a$, reads from $d$, and writes to $d$, and process $P_b$ reads from $b$

Q: Can we safely run both processes concurrently?

Task: Draw arrows to specify how $P_b$ interacts with items in main memory
Scenario 1: Process $P_a$ reads from $a$, reads from $d$, and writes to $d$, and process $P_b$ reads from $b$.

Q: Can we safely run both processes concurrently?

Task: Draw arrows to specify how $P_b$ interacts with items in main memory.
When can code be parallelized

Scenario 1: Process $P_a$ reads from $a$, reads from $d$, and writes to $d$, and process $P_b$ reads from $b$

Q: Can we safely run both processes concurrently?

The combined read and write diagram for processes $a$ and $b$
Scenario 1: Process $P_a$ reads from $a$, reads from $d$, and writes to $d$, and process $P_b$ reads from $b$

Q: Can we safely run both processes concurrently?

Two processes are independent if the write set of each is disjoint from both the read and write sets of the other.
When can code be parallelized

Scenario 1: Process $P_a$ reads from $a$, reads from $d$, and writes to $d$, and process $P_b$ reads from $b$

Q: Can we safely run both processes concurrently?

Q: Are $P_a$ and $P_b$ independent?
When can code be parallelized

**Scenario 1**: Process $P_a$ reads from $a$, reads from $d$, and writes to $d$, and process $P_b$ reads from $b$.

Q: Are $P_a$ and $P_b$ independent?

Q: What are the read and write sets of $P_a$ and $P_b$?
When can code be parallelized

Scenario 1: Process $P_a$ reads from $a$, reads from $d$, and writes to $d$, and process $P_b$ reads from $b$

Q: Are $P_a$ and $P_b$ independent?

Q: Is $P_a$’s write set disjoint from the read and write sets of $P_b$?
When can code be parallelized

Scenario 1: Process \( P_a \) reads from \( a \), reads from \( d \), and writes to \( d \), and process \( P_b \) reads from \( b \)

Q: Are \( P_a \) and \( P_b \) independent?

Q: Is \( P_a \)'s write set disjoint from the read and write sets of \( P_b \)?
When can code be parallelized

Scenario 1: Process $P_a$ reads from $a$, reads from $d$, and writes to $d$, and process $P_b$ reads from $b$

Q: Are $P_a$ and $P_b$ independent?

Q: Is $P_a$’s write set disjoint from the read and write sets of $P_b$?

Q: Is $P_b$’s write set disjoint from the read and write sets of $P_a$?

Yes
When can code be parallelized

Scenario 1: Process $P_a$ reads from $a$, reads from $d$, and writes to $d$, and process $P_b$ reads from $b$

Q: Are $P_a$ and $P_b$ independent?

Q: Is $P_a$’s write set disjoint from the read and write sets of $P_b$?
Yes

Q: Is $P_b$’s write set disjoint from the read and write sets of $P_a$?
When can code be parallelized

Scenario 1: Process $P_a$ reads from $a$, reads from $d$, and writes to $d$, and process $P_b$ reads from $b$

Q: Are $P_a$ and $P_b$ independent?  
Yes

Q: Is $P_a$’s write set disjoint from the read and write sets of $P_b$?  
Yes

Q: Is $P_b$’s write set disjoint from the read and write sets of $P_a$?  
Yes
When can code be parallelized

Scenario 1: Process $P_a$ reads from $a$, reads from $d$, and writes to $d$, and process $P_b$ reads from $b$

Q: Are $P_a$ and $P_b$ independent?
Yes

Q: Intuitively, why does $P_a$ reading from and writing to $d$ NOT make $P_a$ and $P_b$ dependent?
When can code be parallelized

Scenario 1: Process $P_a$ reads from $a$, reads from $d$, and writes to $d$, and process $P_b$ reads from $b$

Q: Are $P_a$ and $P_b$ independent?  
Yes

Q: Intuitively, why does $P_a$ reading from and writing to $d$ NOT make $P_a$ and $P_b$ dependent?

Instructions executed by a single process (thread, $P_a$ in this case), are executed sequentially, so although $P_a$ might experience a stall due to a structural hazard in its pipeline, there is no chance that $d$’s value at any time will be “erroneous”
When can code be parallelized

Scenario 2: Process $P_a$ reads from $d$, and writes to $d$ and $b$, and process $P_b$ reads from $b$

Q: Can we safely run both processes concurrently?
Up Next

Lab (Thursday)