Homework Clarification

```
matrixProduct()
    // naive implementation
    // for, for, for loop
}

main()
    // create 2, 400x400 matrices
    // begin timer
    matrixProduct()
    matrixProduct()
    matrixProduct()
    // end timer
    // output time to run

```

Inputs
400
non-threaded
4 times

Output
12ns

```
matrixProduct()
    // naive implementation
    // for, for, for loop
}

main()
    // create 2, 800x800 matrices
    // begin timer
    // for i:0 to 7
    run matrixProduct in new thread
    // wait for completion
    // end timer
    // output time to run
```

Inputs
800
threaded
8 times

Output
33ns

```
matrixProduct()
    // naive implementation
    // for, for, for loop
}

main()
    // create 2, 800x800 matrices
    // begin timer
    // for i:0 to 15
    run matrixProduct in new thread
    // wait for completion
    // end timer
    // output time to run
```

Inputs
800
threaded
16 times

Output
45ns

```
matrixProduct()
    // naive implementation
    // for, for, for loop
}

main()
    // create 2, 400x400 matrices
    // begin timer
    // for i:0 to 7
    run matrixProduct in new thread
    // wait for completion
    // end timer
    // output time to run
```

Inputs
400
Non-threaded
2 times

Output
8ns
From last time ...
From last time ...

**Task:** Be able to discuss the pros and cons of such topologies

<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>

**Discuss:** Memory (in)consistency

If two or more CPUs/threads are accessing AND editing values at the same memory location, there is the high chance of data override conflicts.
From last time ...

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

\[
\begin{array}{cc}
AB \\
\begin{array}{cc}
36 & 41 \\
64 & 73 \\
\end{array} \\
\end{array}
\]
From last time ...

\[
\begin{array}{c|c}
A & B \\
\hline
2 & 3 \\
4 & 5 \\
\hline
\end{array}
\times
\begin{array}{c|c}
6 & 7 \\
8 & 9 \\
\hline
\end{array}
= \begin{array}{c|c}
36 & 41 \\
64 & 73 \\
\hline
\end{array}
\]

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];
        }
    }
}

\[(AB)_{i,j} = \sum_{k=1}^{m} A_{ik}B_{kj}\]
From last time ...

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

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];
    }
}

\[(AB)_{ij} = \sum_{k=1}^{m} A_{ik}B_{kj}\]
From last time ...

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

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];
        }
    }
}

\[(AB)_{ij} = \sum_{k=1}^{m} A_{ik}B_{kj}\]
Today

Andrews, Chapters 1 and 2

Parallelization
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?**

\[
A = \begin{bmatrix}
1 \\
5 \\
3
\end{bmatrix}
\]

\[
B = \begin{bmatrix}
4 \\
2 \\
9
\end{bmatrix}
\]

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 = \{1, 3, 5\}

B = \{2, 4, 9\}

M = \{3, 5, 6, 7\}

N = \{0, 1, 2, 4, 6, 8\}

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

Are sets M and N disjoint?
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?

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

A = \{1, 3, 5\}

B = \{2, 4, 9\}

M = \{3, 5, 6, 7\}

N = \{1, 2, 4, 6, 9\}

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.

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

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

**Task**: Be able to explain the reasoning behind these statements.

Separate processes, $P_a$, $P_b$, on separate CPUs and/or separate instructions streams.
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.

Assume that process $P_a$ reads from $a$, reads from $d$, and writes to $d$.
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.

Assume that process $P_b$ reads from $b$. 

```
  a  b  c  
  Memory
  Interconnection network
  Cache
  CPU
  $P_a$

  ...  
  .....
  ...   
  Cache
  CPU
  $P_b$
  d  e  f  
  Memory
```
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: Are $P_a$ and $P_b$ independent?
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: 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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Q: Are $P_a$ and $P_b$ independent?
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: Are $P_a$ and $P_b$ independent?
Two processes are **independent** if the write set of each is disjoint from both the read and write sets of the other.

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 \( P_a \)?
Two processes are independent if the write set of each is disjoint from both the read and write sets of the other.

Q: Are $P_a$ and $P_b$ independent?  
No

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 $P_a$?  
No
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: Are \( P_a \) and \( P_b \) independent?
No

Give an example (code, threads) that “show” why processes that are NOT independent may produce “wrong” results when run concurrently.
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.

\[
\begin{array}{cccccc}
\text{a} & \text{b} & \text{c} & \text{d} & \text{e} & \text{f} \\
3 & 4 \\
\end{array}
\]

\[
\begin{array}{ll}
b &= 3 \\
d &= 4 \\
\text{co} \\
i1 : b &= d \times 2 \\
i2 : d &= b \\
i3 : \text{print}(b) \\
\text{oc} \\
\end{array}
\]
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

```
<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

b = 3
d = 4
co
i1 : b = d * 2
i2 : d = b
i3 : print(b)
co

Q: What are the possible histories (assume an infinite number of 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.

```
<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
```

- \( b = 3 \)
- \( d = 4 \)
- \( \text{co} \)
- \( i_1 : b = d \times 2 \)
- \( i_2 : d = b \)
- \( i_3 : \text{print}(b) \)
- \( \text{oc} \)

\[
\begin{align*}
&i_1 < i_2 < i_3 & i_2 < i_3 < i_1 \\
&i_1 < i_3 < i_2 & i_3 < i_1 < i_2 \\
i_2 < i_1 < i_3 & i_3 < i_2 < i_1
\end{align*}
\]
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:** Are $P_a$ and $P_b$ independent?

**No**

**Q:** What are the possible histories (assume only 2 CPUs)?

**Q:** Which instructions are executed by $P_a$, and which by $P_b$?

- $b = 3$
- $d = 4$
- $co$
- $i1: b = d \times 2$
- $i2: d = b$
- $i3: \text{print}(b)$
- $oc$
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: Are $P_a$ and $P_b$ independent?

No

Instructions are executed sequentially in a thread / process / CPU, thus the only constraint is that $i_1 < i_2$.
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: Are $P_a$ and $P_b$ independent?

No
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: Are $P_a$ and $P_b$ independent?

No
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: Are $P_a$ and $P_b$ independent?

No

\[ b = 3 \]
\[ d = 4 \]

\[ i1 : b = d \times 2 \]
\[ i2 : d = b \]
\[ i3 : \text{print}(b) \]
Two processes are **independent** if the write set of each is disjoint from both the read and write sets of the other.

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

No
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: Are $P_a$ and $P_b$ independent?

No

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

No
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: Are $P_a$ and $P_b$ independent?**

**No**

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>

- $b = 3$
- $d = 4$
- `co`
- $i_1: b = d \times 2$
- $i_2: d = b$
- $i_3: \text{print}(b)$

<table>
<thead>
<tr>
<th>$P_a$</th>
<th>$i_1: b = d \times 2$</th>
<th>$i_2: d = b$</th>
<th>$i_3: \text{print}(b)$</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>$P_b$</th>
<th>$i_1 &lt; i_2 &lt; i_3$</th>
<th>$b = 8$</th>
<th>$d = 8$</th>
<th>$\text{print} : 8$</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>$P_a$</th>
<th>$i_1 &lt; i_3 &lt; i_2$</th>
<th>$b = 8$</th>
<th>$d = 8$</th>
<th>$\text{print} : 8$</th>
</tr>
</thead>
</table>

| $P_b$ | $i_3 < i_1 < i_2$ | $b = 8$ | $d = 8$ | $\text{print} : 8$ |
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:** Are $P_a$ and $P_b$ independent?

**No**
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:** Are \( P_a \) and \( P_b \) independent?

No
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: Are $P_a$ and $P_b$ independent?

No

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>$d$</th>
<th>$e$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$b = 3$
$d = 4$

$P_a$

- i1 : $b = d \times 2$
- i2 : $d = b$
- i3 : print($b$)

$P_b$

- i1 : $b = d = 8$
- i2 : $d = 8$
- i3 : print : 8

i1 < i2 < i3
i1 < i3 < i2
i3 < i1 < i2
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: Are \( P_a \) and \( P_b \) independent?

No
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: Are $P_a$ and $P_b$ independent?

No

- $P_a$:
  - $i_1: b = d \times 2$
  - $i_2: d = b$
  - $i_3: \text{print}(b)$

- $P_b$:
  - $i_1: b = 8$
  - $i_3: \text{print}(8)$

Let's analyze the operations:

- $i_1 < i_2 < i_3$
- $b = 8$
- $d = 8$
- $\text{print} : 8$

Since $i_3$ is executed after $i_1$, they are not independent.
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: Are $P_a$ and $P_b$ independent?

No

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>d</td>
<td></td>
<td></td>
</tr>
<tr>
<td></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>
</tbody>
</table>

$b = 3$

d = 4

$i1: b = d \times 2$

$i2: d = b$

$i3: \text{print}(b)$
When can code be parallelized

The code we’ve already seen...

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

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

```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: When the code is NOT parallelized, which CPU performs all of the calculations?

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

```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

```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];
    }
}
```

i = 0, 1, 2, ... n-1
j = 0, 1, 2, ... n-1
k = 0, 1, 2, ... n-1

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

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

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

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

(Be able to perform such “analysis” for the exams)
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

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

- \( i = 0, 1, 2, \ldots n-1 \)
- \( j = 0, 1, 2, \ldots n-1 \)
- \( k = 0, 1, 2, \ldots n-1 \)

Notice that there are \( n \) iterations for the \( i \) loop

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

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

```
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];
    }
}
```

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];
    }
}
```

- **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]

Notice that there are n iterations for the i loop

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

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 $k=[0, n-1]$

Task: Parallelize as "much" of this code as is possible
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.
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];
    }
}
```

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

```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];
    }
}
```

```java
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];
    }
}
```

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

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

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

```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];
    }
}
```

<table>
<thead>
<tr>
<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>m=4</td>
<td>0,4,8</td>
<td>1,5,9</td>
<td>2,6</td>
<td>3,7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m=5</td>
<td>0,5</td>
<td>1,6</td>
<td>2,7</td>
<td>3,8</td>
<td>4,9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m=6</td>
<td>0,6</td>
<td>1,7</td>
<td>2,8</td>
<td>3,9</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m=10</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

```

a,b,c
```

```
Memory           ...           Memory
Interconnection network
Cache           ...           Cache
CPU
```

0, 1, 2, 3, .. n-2, n-1
When can code be parallelized

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

```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

```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?

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
When can code be parallelized

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

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

\[ i = n-1, j = n-1 \]

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

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 < 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

Q: And if you have numCPU < n^2 processors, how do you dole out processes?
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

```c
for [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];
}
```

Assume we live in a perfect world, and we have a seemingly infinite number of processors ... then ...

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

On the board explanation
Up Next ...

Lab (Friday)