What are we going to learn?

- Creating 2-D arrays
- Thinking about “grain size”
- Introducing point-to-point communications
- Reading and printing 2-D matrices
- Analyzing performance when computations and communications overlap

All-pairs Shortest Path Problem

The adjacency matrix containing the resulting distances is shown below.
Floyd’s Algorithm

for $k \leftarrow 0$ to $n-1$
  for $i \leftarrow 0$ to $n-1$
    for $j \leftarrow 0$ to $n-1$
      $a[i,j] \leftarrow \min(a[i,j], a[i,k] + a[k,j])$
    endfor
  endfor
endfor

Why It Works

Shortest path from $i$ to $k$ through $0, 1, \ldots, k-1$

Shortest path from $i$ to $j$ through $0, 1, \ldots, k-1$

Shortest path from $k$ to $j$ through $0, 1, \ldots, k-1$
Dynamic 1-D Array Creation

Run-time Stack

Heap

Dynamic 2-D Array Creation

Run-time Stack

Heap
Designing Parallel Algorithm

- Partitioning
- Communication
- Agglomeration and Mapping

Partitioning

- Domain or functional decomposition?
- Consider the pseudocode
- Same assignment statement executed \( n^3 \) times
- No functional parallelism
- Domain decomposition: divide matrix \( A \) into its \( n^2 \) elements
  - Assign a task each of the \( n^2 \) elements
Communication

Primitive tasks

Iteration $k$: every task in row $k$ broadcasts its value w/in task column

Updating $a[3,4]$ when $k = 1$

Iteration $k$: every task in column $k$ broadcasts its value w/in task row

Agglomeration and Mapping

- Number of tasks: static
- Communication among tasks: structured
- Computation time per task: constant
- Strategy:
  - Agglomerate tasks to minimize communication
  - Create one task per MPI process
Mapping Strategy

Two Data Decompositions

Rowwise block striped

Columnwise block striped
Comparing Decompositions

- Columnwise block striped
  - Broadcast within columns eliminated
- Rowwise block striped
  - Broadcast within rows eliminated
  - Reading matrix from file simpler
- Choose rowwise block striped decomposition

File Input

File
Pop Quiz

Why don’t we input the entire file at once and then scatter its contents among the processes, allowing concurrent message passing?

Point-to-point Communication

- Involves a pair of processes
- One process sends a message
- Other process receives the message
Send/Receive Not Collective

Function MPI_Send

```c
int MPI_Send (    
    void       *message,    
    int        count,       
    MPI_Datatype datatype,  
    int        dest,        
    int        tag,         
    MPI_Comm   comm         
)  
```
**Function MPI_Recv**

```c
int MPI_Recv (
    void     *message,
    int       count,
    MPI_Datatype datatype,
    int       source,
    int       tag,
    MPI_Comm  comm,
    MPI_Status *status
)
```

**Coding Send/Receive**

```c
... if (ID == j) {
    ... Receive from I
    ...
} ...
...
... if (ID == i) {
    ... Send to j
    ...
} ...
... Receive is before Send.
Why does this work?
```
Inside MPI_Send and MPI_Recv

Sending Process

- Program Memory
- System Buffer

Receiving Process

- System Buffer
- Program Memory

MPI_Send

MPI_Recv

Return from MPI_Send

- Function blocks until message buffer free
- Message buffer is free when
  - Message copied to system buffer, or
  - Message transmitted
- Typical scenario
  - Message copied to system buffer
  - Transmission overlaps computation
Return from MPI_Recv

- Function blocks until message in buffer
- If message never arrives, function never returns

Deadlock

- Deadlock: process waiting for a condition that will never become true
- Easy to write send/receive code that deadlocks
  - Two processes: both receive before send
  - Send tag doesn’t match receive tag
  - Process sends message to wrong destination process
Computational Complexity

- Innermost loop has complexity $\Theta(n)$
- Middle loop executed at most $\left\lceil \frac{n}{p} \right\rceil$ times
- Outer loop executed $n$ times
- Overall complexity $\Theta(\frac{n^3}{p})$

Communication Complexity

- No communication in inner loop
- No communication in middle loop
- Broadcast in outer loop — complexity is $\Theta(n \log p)$
- Overall complexity $\Theta(n^2 \log p)$
Computation/communication Overlap

Predicted vs. Actual Performance

<table>
<thead>
<tr>
<th>Processes</th>
<th>Predicted</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25.54</td>
<td>25.54</td>
</tr>
<tr>
<td>2</td>
<td>13.02</td>
<td>13.89</td>
</tr>
<tr>
<td>3</td>
<td>9.01</td>
<td>9.60</td>
</tr>
<tr>
<td>4</td>
<td>6.89</td>
<td>7.29</td>
</tr>
<tr>
<td>5</td>
<td>5.86</td>
<td>5.99</td>
</tr>
<tr>
<td>6</td>
<td>5.01</td>
<td>5.16</td>
</tr>
<tr>
<td>7</td>
<td>4.40</td>
<td>4.50</td>
</tr>
<tr>
<td>8</td>
<td>3.94</td>
<td>3.98</td>
</tr>
</tbody>
</table>