Designing Parallel Programs

Abhinav Bhavele, Department of Computer Science
Announcements

• Zaratan accounts have been created for everyone
• When emailing me, please cc the TAs also
  • Emails are on the class website: https://www.cs.umd.edu/class/fall2022/cmse416/index.shtml
  • Prefix [CMSC416] or [CMSC818X] to your email subject
• Assignment 0 will be posted on Sept 19 and will be due on Sept 26
  • Not graded, 0 points
Getting started with zaratan

- Over 380 nodes with AMD Milan processors (128 cores/node)
- 20 nodes with four NVIDIA A100 GPUs

ssh username@login.zaratan.umd.edu
Writing parallel programs

• Decide the serial algorithm first

• Data: how to distribute data among threads/processes?
  • Data locality: assignment of data to specific processes to minimize data movement

• Computation: how to divide work among threads/processes?

• Figure out how often communication will be needed
Two-dimensional stencil computation

- Commonly found kernel in computational codes
- Heat diffusion, Jacobi method, Gauss-Seidel method

Serial code
for(int t=0; t<num_steps; t++) {
    ...
    for(i ...) {
        for(j ...) {
        }
    }
    // copy contents of A_new into A
    ...
}
2D stencil computation in parallel
2D stencil computation in parallel

- 1D decomposition
  - Divide rows (or columns) among processes
2D stencil computation in parallel

- 1D decomposition
  - Divide rows (or columns) among processes
2D stencil computation in parallel

- **1D decomposition**
  - Divide rows (or columns) among processes

- **2D decomposition**
  - Divide both rows and columns (2d blocks) among processes
2D stencil computation in parallel

• **1D decomposition**
  - Divide rows (or columns) among processes

• **2D decomposition**
  - Divide both rows and columns (2d blocks) among processes
Prefix sum

• Calculate partial sums of elements in array

• Also called a “scan” sometimes

\[ pSum[0] = A[0] \]

\[
\text{for}(i=1; i<N; i++) \{
    pSum[i] = pSum[i-1] + A[i]
\}
\]
Parallel prefix sum

| 2 | 8 | 3 | 5 | 7 | 4 | 1 | 6 |
Parallel prefix sum

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>8</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>
## Parallel prefix sum

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>8</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>10</td>
<td>11</td>
<td>8</td>
<td>12</td>
<td>11</td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>
Parallel prefix sum

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>8</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>4</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>10</td>
<td>11</td>
<td>8</td>
<td>12</td>
<td>11</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>10</td>
<td>13</td>
<td>18</td>
<td>23</td>
<td>19</td>
<td>17</td>
<td>18</td>
</tr>
</tbody>
</table>
Parallel prefix sum
In practice
In practice

- You have $N$ numbers and $P$ processes, $N \gg P$
In practice

• You have $N$ numbers and $P$ processes, $N \gg P$
• Assign a $N/P$ block to each process
  • Do calculation for the blocks on each process locally
In practice

- You have \( N \) numbers and \( P \) processes, \( N \gg P \)
- Assign a \( N/P \) block to each process
  - Do calculation for the blocks on each process locally
- Then do parallel algorithm with partial prefix sums
The \( n \)-body problem

- Simulate the motion of celestial objects interacting with one another due to gravitational forces

- Naive algorithm: \( O(n^2) \)
  - Every body calculates forces pair-wise with every other body (particle)

Data distribution in $n$-body problems

- Naive approach: Assign $n/p$ particles to each process
- Other approaches?
Data distribution in $n$-body problems

- Naive approach: Assign $n/p$ particles to each process
- Other approaches?

http://datagenetics.com/blog/march22013/
https://en.wikipedia.org/wiki/Z-order_curve
Data distribution in \( n \)-body problems

- Naive approach: Assign \( n/p \) particles to each process
- Other approaches?

Space-filling curves

http://datagenetics.com/blog/march22013/
https://en.wikipedia.org/wiki/Z-order_curve
Data distribution in \( n \)-body problems

- Naive approach: Assign \( n/p \) particles to each process
- Other approaches?

[Image of space-filling curves and a tree diagram]

http://datagenetics.com/blog/march2013/
https://en.wikipedia.org/wiki/Z-order_curve
Data distribution in \( n \)-body problems

- Let us consider a two-dimensional space with bodies/particles in it
Data distribution in $n$-body problems

- Let us consider a two-dimensional space with bodies/particles in it
Data distribution in \( n \)-body problems

- Let us consider a two-dimensional space with bodies/particles in it

Quad-tree: not all nodes are shown
Load balance and grain size

- **Load balance**: try to balance the amount of work (computation) assigned to different threads/processes
  - Bring ratio of maximum to average load as close to 1 as possible
  - Secondary consideration: also load balance amount of communication

- **Grain size**: ratio of computation-to-communication
  - Coarse-grained (more computation) vs. fine-grained (more communication)