N-Body Simulation
What is N-body Simulation?

Simulation of a dynamical system of particles, usually under the influence of physical forces, such as gravity.

\[ F = \frac{G m_1 m_2}{r^2} \]
Objective

• Simulate the gravitational forces acting between a number of bodies in space.

• Barnes-Hut Tree algorithm for optimization of the force calculation.

• Implementation of the project using MPI.

• Comparison of different approaches.
The Barnes-Hut Algorithm

• Speeding up the brute force n-body algorithm is to group nearby bodies and approximate them as a single body.

• If the group is sufficiently far away, we can approximate its gravitational effects by using its center of mass.

• Two bodies \((x_1, y_1)\) of mass ‘\(m_1\)’, and \((x_2, y_2)\) of mass ‘\(m_2\)’.

\[
\begin{align*}
m &= m_1 + m_2 \\
x &= \frac{x_1 m_1 + x_2 m_2}{m} \\
y &= \frac{y_1 m_1 + y_2 m_2}{m}
\end{align*}
\]
The Barnes-Hut Algorithm

- It recursively divides the set of bodies into groups by storing them in a *quad-tree*.
- The topmost node represents the whole space, and its four children represent the four quadrants of the space.
The Barnes-Hut Algorithm
The Barnes-Hut Algorithm

- Determine if \( (s / d) < \Theta \)

- \( s \) is the width of the region represented by the internal node,

- \( d \) is the distance between the body and the node’s center-of-mass

- \( \Theta \) can change the speed and accuracy of the simulation. Typically, 0.5.
The Barnes-Hut Algorithm

Constructing the Barnes-Hut tree:

To insert a body $b$ into the tree rooted at node $x$, use recursive procedure:

- **If node $x$ does not contain a body, put the new body $b$ here.**

- **If node $x$ is an internal node, update the center-of-mass and total mass of $x$. Recursively insert the body $b$ in the appropriate quadrant.**

- **If node $x$ is an external node, subdivide the region further by creating four children. Then, recursively insert both $b$ and $c$ into the appropriate quadrant(s).**

- **Finally, update the center-of-mass and total mass of $x$.**
Initial Conditions
5 nodes to be added; start with empty root node

Add body A
fits in nw corner of root node

Add body B
fits in ne corner of root node

Add body C
ne corner occupied by B; insert branch and add both B and C as children

Add body D
fits in se corner of branch node

Add body E
fits in se corner of root node
Our Attempt

1. Master – Worker Configuration:

- **Parallel Tree Formation**
  - Every node reads data from input file.
  - Formation of quad-tree at all nodes in parallel.

- **Parallel Force Calculation**
  - Every processor selects bodies from input file based on its rank.
  - Calculate force on the selected bodies and their new position due to the force.

- **Merge Partial Results**
  - Merge the partial results from all the nodes at master node to get the final result.
  - Broadcast the new dataset to all nodes.
1. Master – Worker Configuration:

- Initially data is at all processors
- Each processor on individual machines will create Tree.
- Force for divided points will be calculated parallelly
- New Points will be gathered at processor 1
- Points will be multicast on all processors
1. Master – Worker Configuration:

### Number of Bodies vs Number of Cores (8 Cores/Node)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>0.00415</td>
<td>0.0034</td>
<td>0.00379</td>
<td>0.00412</td>
<td>0.00658</td>
<td>0.069261</td>
<td>0.007926</td>
<td>1.164081</td>
<td>9.328586</td>
<td>15.090703</td>
</tr>
<tr>
<td>1024</td>
<td>0.007804</td>
<td>0.006331</td>
<td>0.006141</td>
<td>0.006481</td>
<td>0.00813</td>
<td>0.01089</td>
<td>0.024016</td>
<td>0.054247</td>
<td>3.547129</td>
<td>9.35108</td>
</tr>
<tr>
<td>4,000</td>
<td>1.05145</td>
<td>0.546004</td>
<td>0.367569</td>
<td>0.47367</td>
<td>0.148349</td>
<td>0.084429</td>
<td>0.14190</td>
<td>1.348655</td>
<td>6.092180</td>
<td>10.570122</td>
</tr>
<tr>
<td>10,000</td>
<td>7.204272</td>
<td>3.585363</td>
<td>2.414663</td>
<td>1.829134</td>
<td>1.721207</td>
<td>3.879273</td>
<td>11.297868</td>
<td>16.450013</td>
<td>33.001744</td>
<td>70.34019</td>
</tr>
<tr>
<td>40,000</td>
<td>125.70631</td>
<td>63.20303</td>
<td>42.38242</td>
<td>33.10332</td>
<td>25.78558</td>
<td>27.124613</td>
<td>36.43922</td>
<td>48.252101</td>
<td>79.216301</td>
<td>114.675226</td>
</tr>
</tbody>
</table>
1. Master – Worker Configuration:

Time vs No of Cores (8 Cores/Node)
1. Master – Worker Configuration:
   Time vs No of Cores (8 Cores/Node)
1. Master – Worker Configuration:

**Number of Bodies vs Number of Cores (1 Core/Node)**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>0.01806</td>
<td>0.00616</td>
<td>0.00452</td>
<td>0.00532</td>
<td>0.00325</td>
<td>0.00369</td>
<td>0.00456</td>
<td>0.00379</td>
<td>0.0058</td>
<td>0.00359</td>
</tr>
<tr>
<td>1024</td>
<td>0.07366</td>
<td>0.05778</td>
<td>0.0604</td>
<td>0.05033</td>
<td>0.07467</td>
<td>0.07871</td>
<td>0.07971</td>
<td>0.07617</td>
<td>0.07723</td>
<td>0.07723</td>
</tr>
<tr>
<td>4000</td>
<td>1.15893</td>
<td>1.13307</td>
<td>1.10315</td>
<td>1.09626</td>
<td>1.09492</td>
<td>1.09624</td>
<td>1.09624</td>
<td>1.09402</td>
<td>1.09466</td>
<td>1.0961</td>
</tr>
<tr>
<td>20000</td>
<td>84.6857</td>
<td>83.6755</td>
<td>82.6016</td>
<td>83.0337</td>
<td>82.21</td>
<td>83.5951</td>
<td>82.9271</td>
<td>48.6264</td>
<td>51.0667</td>
<td>52.4524</td>
</tr>
<tr>
<td>40000</td>
<td>378.088</td>
<td>270.424</td>
<td>268.216</td>
<td>280.575</td>
<td>281.261</td>
<td>281.266</td>
<td>279.194</td>
<td>280.503</td>
<td>278.525</td>
<td>278.534</td>
</tr>
</tbody>
</table>
1. Master – Worker Configuration:
   Time vs No of Cores (1 Core/Node)
1. Master – Worker Configuration:

Time vs No of Cores (1 Core/Node)
2. All to All Configuration:

- **Parallel Tree Formation**
  - Every node reads data from input file.
  - Formation of quad-tree at all nodes in parallel.

- **Parallel Force Calculation**
  - Every processor selects bodies from quad-tree based on its rank.
  - Calculate force on the selected bodies and their new position due to this force.

- **Merge Partial Results**
  - Every node gathers partial result from all the other nodes using **MPI_Allgather**.
2. All to All Configuration:

Initially data is on all Processors

Each processor on individual machines will create Tree.

Force for divided points will be calculates parallelly

MPI_ALLGATHER
# 2. All to All Configuration:

Number of Bodies vs Number of Cores (8 Cores/Node)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0.00314</td>
<td>0.0020</td>
<td>0.00171</td>
<td>0.00199</td>
<td>0.00282</td>
<td>0.004832</td>
<td>0.010829</td>
<td>0.018858</td>
<td>0.043453</td>
<td>0.167657</td>
</tr>
<tr>
<td>4000</td>
<td>0.00517</td>
<td>0.0042</td>
<td>0.00339</td>
<td>0.005657</td>
<td>0.00627</td>
<td>0.008842</td>
<td>0.016211</td>
<td>0.031286</td>
<td>0.064436</td>
<td>0.139774</td>
</tr>
<tr>
<td>8000</td>
<td>0.05313</td>
<td>0.0640</td>
<td>0.05303</td>
<td>0.052184</td>
<td>0.051377</td>
<td>0.095169</td>
<td>0.176234</td>
<td>0.261428</td>
<td>0.25843</td>
<td>0.225114</td>
</tr>
<tr>
<td>10000</td>
<td>0.06300</td>
<td>0.06113</td>
<td>0.06096</td>
<td>0.060151</td>
<td>0.05019</td>
<td>0.036377</td>
<td>0.037995</td>
<td>0.046847</td>
<td>0.130437</td>
<td>0.254299</td>
</tr>
<tr>
<td>20000</td>
<td>0.27814</td>
<td>0.2573</td>
<td>0.25487</td>
<td>0.24071</td>
<td>0.10119</td>
<td>0.095223</td>
<td>0.048549</td>
<td>1.10658</td>
<td>1.90124</td>
<td>2.579095</td>
</tr>
<tr>
<td>40000</td>
<td>0.56598</td>
<td>0.51989</td>
<td>0.50833</td>
<td>0.50402</td>
<td>0.20842</td>
<td>0.170834</td>
<td>0.19661</td>
<td>0.243217</td>
<td>2.532225</td>
<td>3.993428</td>
</tr>
<tr>
<td>100000</td>
<td>1.798</td>
<td>1.30278</td>
<td>1.209313</td>
<td>0.98268</td>
<td>0.42435</td>
<td>0.230418</td>
<td>0.471686</td>
<td>0.759693</td>
<td>4.263693</td>
<td>6.618865</td>
</tr>
<tr>
<td>10000000</td>
<td>71.4634</td>
<td>43.5849</td>
<td>29.95473</td>
<td>21.3642</td>
<td>12.5039</td>
<td>46.167598</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2. All to All Configuration:
Time vs No of Cores (8 Cores/Node)
2. All to All Configuration:
   Time vs No of Cores (8 Cores/Node)
2. All to All Configuration:

Number of Bodies vs Number of Cores (1 Core/Node)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0.00322</td>
<td>0.0029</td>
<td>0.0021</td>
<td>0.0020</td>
<td>0.00367</td>
<td>0.005291</td>
<td>0.012746</td>
<td>0.037138</td>
<td>0.07351</td>
<td>0.60512</td>
</tr>
<tr>
<td>4000</td>
<td>0.00497</td>
<td>0.0044</td>
<td>0.00413</td>
<td>0.003879</td>
<td>0.007364</td>
<td>0.010942</td>
<td>0.02389</td>
<td>0.042596</td>
<td>0.092569</td>
<td>0.97594</td>
</tr>
<tr>
<td>8000</td>
<td>0.05302</td>
<td>0.07591</td>
<td>0.06739</td>
<td>0.064371</td>
<td>0.06075</td>
<td>0.06753</td>
<td>0.0760231</td>
<td>0.162841</td>
<td>0.277403</td>
<td>1.52179</td>
</tr>
<tr>
<td>10000</td>
<td>0.07310</td>
<td>0.07589</td>
<td>0.07089</td>
<td>0.070012</td>
<td>0.133816</td>
<td>0.0984027</td>
<td>0.3053662</td>
<td>0.539712</td>
<td>2.71329</td>
<td></td>
</tr>
<tr>
<td>20000</td>
<td>0.25491</td>
<td>0.22671</td>
<td>0.21561</td>
<td>0.20683</td>
<td>0.195752</td>
<td>0.184693</td>
<td>0.204915</td>
<td>1.73914</td>
<td>2.6319</td>
<td>4.3891</td>
</tr>
<tr>
<td>40000</td>
<td>0.56071</td>
<td>0.53892</td>
<td>0.52593</td>
<td>0.519768</td>
<td>0.37516</td>
<td>0.298305</td>
<td>0.24730</td>
<td>2.13840</td>
<td>3.86032</td>
<td>7.09152</td>
</tr>
<tr>
<td>100000</td>
<td>1.8065</td>
<td>1.5491</td>
<td>1.3118</td>
<td>1.1863</td>
<td>1.05293</td>
<td>0.91742</td>
<td>0.76491</td>
<td>3.57921</td>
<td>5.64190</td>
<td>9.42618</td>
</tr>
<tr>
<td>10000000</td>
<td>72.0049</td>
<td>54.7293</td>
<td>41.89721</td>
<td>29.36642</td>
<td>17.83714</td>
<td>62.81534</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2. All to All Configuration:
   Time vs No of Cores (1 Core/Node)
2. All to All Configuration:
   Time vs No of Cores (1 Core/Node)
Observations

1. Master-Worker Configuration:
   - Best result for 8 cores per node is achieved with 4-8 cores.
   - Best results for 1 core per node:
     - For 128 bodies, best result achieved with 3 cores. Increasing cores after that did affect performance much.
     - For 1,024 and 4,000 bodies, best result achieved with 4 cores.
     - For 10,000 and 40,000 bodies, best result achieved with 64 cores.
     - For 20,000 bodies, best result achieved with 2-3 cores.
Observations

2. All to All Configuration:
   • Best results for 8 cores per node:
     • For small datasets (1000-8000 bodies) best result is achieved with 8 cores.
     • For medium datasets (10,000 – 1 Million bodies) best result is achieved with 16 cores.
     • For large datasets (10 Million bodies) best result is achieved with 8 cores.
   • Best results for 1 core per node:
     • For small datasets (1000-10,000 bodies) best result is achieved with 4 cores.
     • For medium datasets (20,000 – 1 Million bodies) best result is achieved with 32 cores.
     • For large datasets (10 Million bodies) best result is achieved with 8 cores.
   • After the best configuration, adding more cores increases running time due to communication overhead.
Conclusion

1. Master-Worker Configuration:
   - Load Distribution: Better than All-to-All configuration as the dataset is distributed for force calculation.
   - Running time more than All-to-All configuration due to communication overhead.
   - Due to sending of whole dataset from the master to other nodes, could not run on datasets having more than 40000 bodies.

2. All to All Configuration:
   - Load Distribution: Worse than Master-Worker configuration as each core processes a subset of tree for force calculation and number of bodies may vary in each part of tree.
   - Running time less than Master-Worker configuration due to less communication overhead as only partial results are sent.
   - Due to less communication overhead, running program with larger datasets was possible.
References

- The Barnes-Hut Algorithm - TOM VENTIMIGLIA & KEVIN WAYNE - http://arborjs.org/docs/barnes-hut
- Planar Decomposition for Quadtree Data Structure - PINAKI MAZUMDER
- An Effective Way to Represent Quadtrees - JAMES FOLEY