Parellelizing N-Body simulations with MPI

Nicholas Molyneaux

Project of Program Parallelization on PC clusters - Autumn 2015

Computational Science and Engineering

N-Body problem

The interaction between two bodies can be modeled with

$$\|\mathbf{F}\|_{m_1} = G\frac{m_1 m_2}{r^2}$$

where the components of the force can be expressed as

$$F_{m_1,x} = \|\mathbf{F}\|_{m_1}\frac{p_{2,x} - p_{1,x}}{r} \text{ and } F_{m_1,y} = \|\mathbf{F}\|_{m_1}\frac{p_{2,y} - p_{1,y}}{r}$$

hence we have

$$a_{m_1,x} = G(p_{2,x} - p_{1,x})\frac{m_2}{r^3} \text{ and } a_{m_1,y} = G(p_{2,y} - p_{1,y})\frac{m_2}{r^3}$$

Solutions & complexity

- Brute-force approach (abbreviated BF)
  - Straight forward approach: calculate interactions between all bodies
  - Complexity of $\mathcal{O}(N^2) \Rightarrow$ very expensive for large problems
  - Easy to implement and parallelize
  - Load balanced by construction
- Barnes-Hut algorithm (abbreviated QT)
  - Use Quad-Tree to store bodies
  - For "large distances", approximate bodies by center of mass
  - Complexity of $\mathcal{O}(NlogN)$
  - **Need to define "large distances" and guarantee load balancing**

Barnes-Hut Implementation

Barnes-Hut implementation

```
if (my_rank == 0)
    load data
broadcast data to all nodes
create quadTree

for (t < tFinal)
    assign branches to nodes

    for (bodies in myBranches)
        calculate force on body
    end for

    update myPositions
    scatter my Positions into all Positions
    if (my_rank == 0)
        write allPositions to file

    rebuild tree for next time step
end for
```

"Large Distance"

From what point do we approximate a group of bodies by the center of mass ?

When the normalized distance between the body and the node goes above a threshold.

```
if size(quad) / distance < theta
    calculateForce(body, node)
else
    calculateForce(body, body)
end if
```

Load Balancing

For a parallel implementation, how to guarantee that each node does (approximately) the same amout of work ?

$$\Downarrow$$

Splitting the tree into "branches" $\Rightarrow$ distributing among nodes.

To maximize performance: distribute branches close to each other to same node

- Concept of bins: each compute node has a capacity
- When full, start giving branches to next compute node... and so on
- Perform a tree walk (depth first)

Load Balancing - algorithm

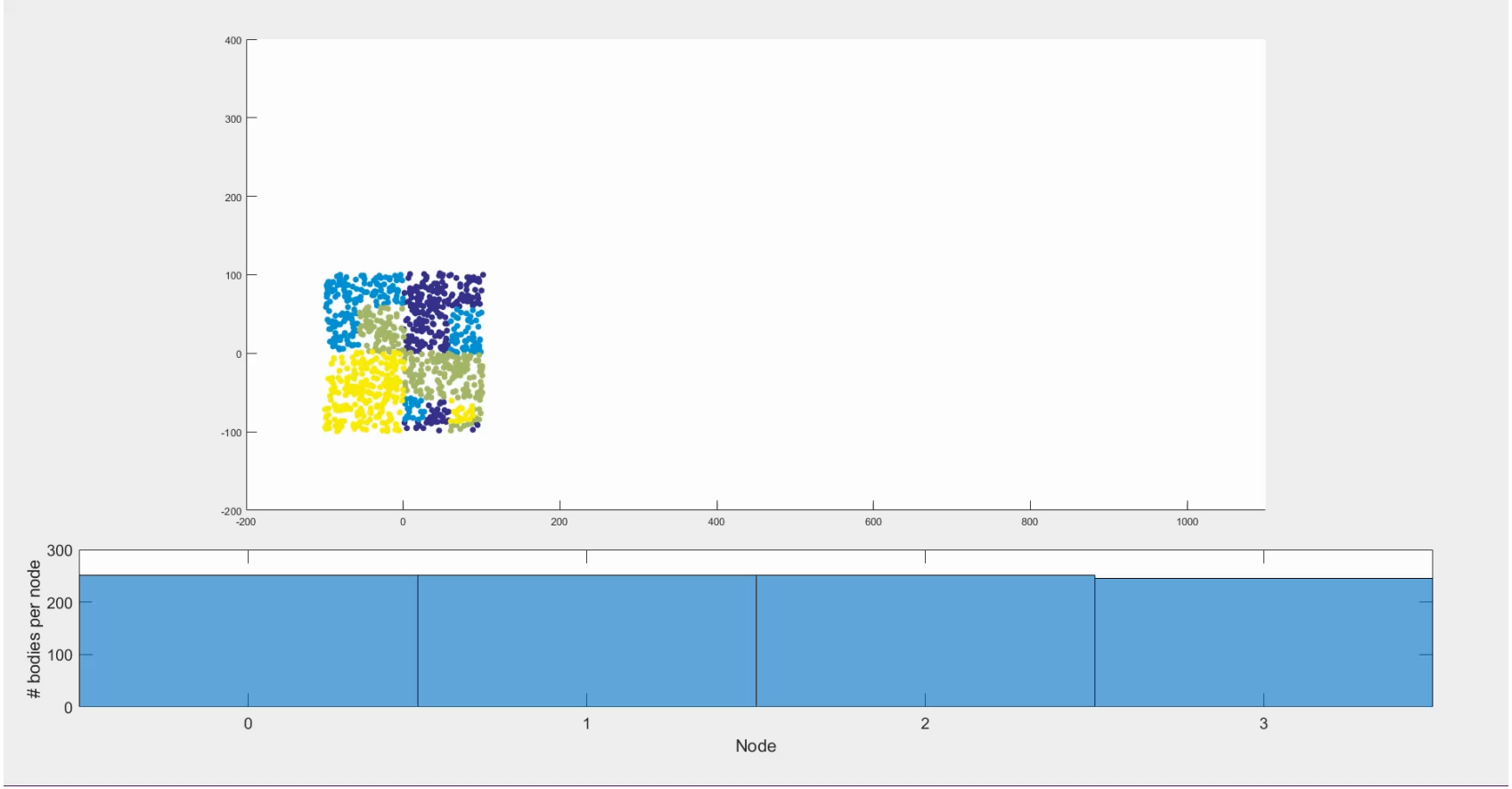
```
if node.nbBodies > computeNode.capacity
    move down to children
else
    while node.nbBodies + computeNode.nbBodies > capacity
        move to next computeNode
    end while

    if node.nbBodies + LASTComputeNode.nbBodies > capacity
        move down to child
    else
        assign this branch to computeNode
        update computeNode.nbBodies
    end if
end if
```

Results

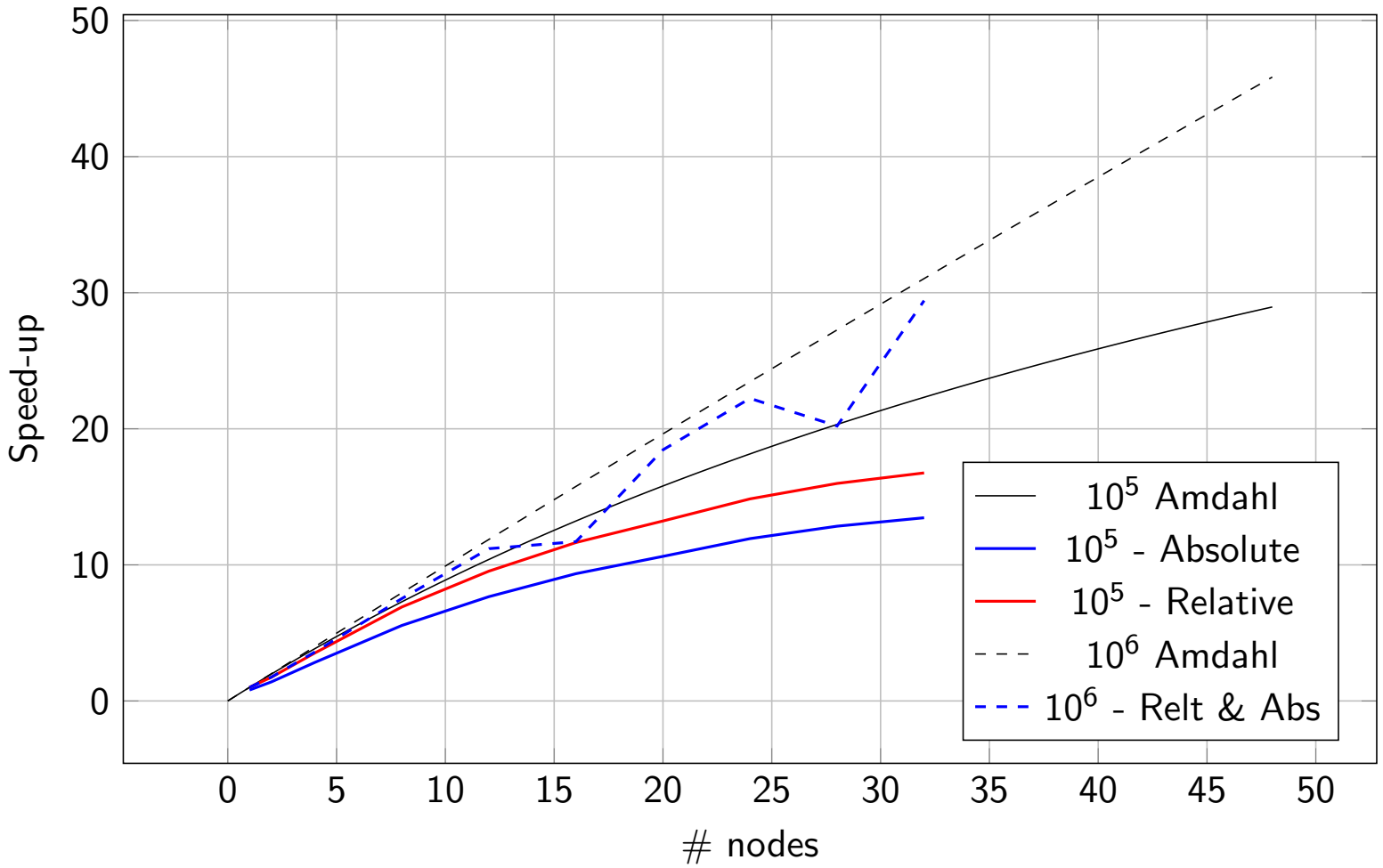
Running times

# bodies	serial time	4 nodes	16 nodes	32 nodes
10^5 - BF	441 s	125 s	38 s	26 s
10^5 - QT	15 s	10.8 s	8.2 s	8.3 s
10^6 - BF	17 h	4.5 h	1.3 h	0.5 h
10^6 - QT	133 s	82 s	80 s	79 s

Load Balancing



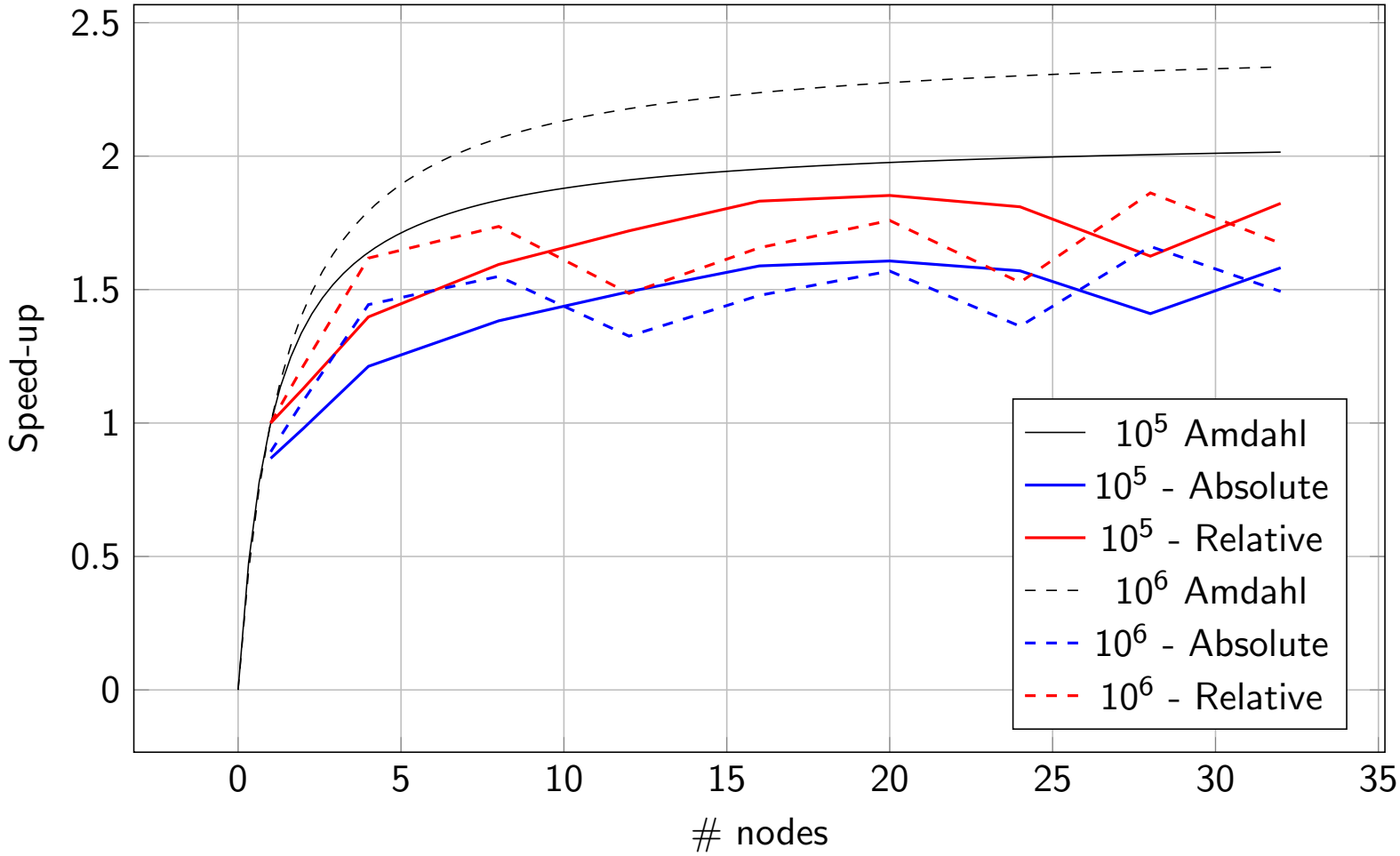
Brute-force speed-up



Brute-Force speed-up

- For both sizes, Amdahl's law over estimated the real speedup
- Even for 10^5 bodies, more than 32 nodes can still be interesting
- Irregular speedup curve possibly due to splitting of nodes between racks, hence significantly larger communicatoin times for some cases

Barnes-Hut (QT) speed-up



Barnes-Hut (QT) speed-up

- Amdahl predicts a very poor speedup... which is respected
- The problem comes from writing the data at each time step
- But significantly faster than brute-force approach

Conclusion

Conclusion

- Quad tree rebuilt from scratch at each time step
- Quad tree could be built in parallel: efficient if bodies are sorted
- Spatial distribution of body assignment (load balancing) could be improved

- A smart serial algorithm is **much** faster than a parallelized trivial approach
- Upper bound of speedups can easily be reached, hence controlling serial and communcation times is critical