High performance N-body simulation on computational grids

Groen, D.J.

Citation for published version (APA):
5 High Performance Gravitational $N$-body Simulations on a Planet-wide Distributed Supercomputer

Based on:
D. Groen, S. Portegies Zwart, T. Ishiyama and J. Makino
High Performance Gravitational $N$-body simulations on a Planet-wide Distributed Supercomputer,
submitted to Computational Science and Discovery.

In this chapter we present our TreePM $N$-body simulations on a planet-wide network of supercomputers connected by dedicated and shared lightpaths. The experiments were performed using the MPWide communication library.

5.1 Introduction

Some applications for large scale simulations require a large amount of compute power. This is often hard to acquire on a single machine. Combining multiple supercomputers to do one large calculation can lift this limitation, but such wide area computing is only suitable for certain algorithms. And even then the political issues, like arranging the network, acquiring the compute time, making reservations, scheduling runtime and synchronizing the run start, and technical limitations are profound. Earlier attempts based on interconnecting PC clusters were quite successful [29, 46, 89, 12, 13], but lacked the raw supercomputer performance required for our application.

Running simulations across multiple supercomputers has been done a few times before [98, 112, 121, 104], though the performance of simulations across three or more supercomputers has not yet been measured in detail. Here we report on the performance of our parallel astronomical simulations which use up to 4 supercomputers and predict the performance for simulations which use 5 or more supercomputers.

In our experiments we use an international infrastructure of supercomputers. These machines include an IBM Power6 supercomputer located at SARA in Amsterdam (the
Netherlands) and three Cray XT-4 supercomputers located at the Edinburgh Parallel Computing Centre in Edinburgh (United Kingdom), the IT Center for Science in Espoo (Finland) and the Center For Computational Astrophysics in Tokyo (Japan). The machines are connected to an optical network which uses more than 32,000 km of cable in total. The Edinburgh site is equipped with a 1 Gbps interface while the other three sites are equipped with a 10 Gbps interface. We achieved a peak performance of 0.610 TFLOP/s and a sustained performance of 0.375 TFLOP/s using 120 cores distributed over 4 sites. To provide a comparison with the international tests we also run the code over up to 5 sites on a national grid of Beowulf clusters. Our wide area simulations are realized with the development of a software environment for Simulating the Universe Structure formation on Heterogeneous Infrastructures, or SUSHI for short.

5.2 Overview of SUSHI

Our code is based on the GreeM cosmological $N$-body integrator, which was originally developed for special-purpose GRAPE hardware [134]. The code integrates the equations of motion for dark matter particles using a shared and adaptive time step scheme and a Tree/Particle-Mesh (TreePM) force calculation method [133] which assumes periodic boundary conditions. The short range force interactions are resolved using a Barnes-Hut tree algorithm [14] while the long range interactions are resolved using a Particle Mesh (PM) algorithm [58].

The tree integration method places particles in a three-dimensional sparse octree structure, where each cell contains the center of mass and the mass aggregate of the particles therein. The method then resolves long range force interactions using particles and tree cells instead of using direct particle-particle evaluation. The accuracy of the tree integration method can be tuned by changing the opening angle ($\theta$), which determines how small and distant a group of particles needs to be to use the approximate particle-tree cell evaluation. A higher value for $\theta$ results in fewer particle-particle evaluations, and a lower accuracy of the simulation. Particle integration using a tree algorithm is more compute-intensive than integration using a PM algorithm, but we speed up the calculations by a factor $\sim 8$ using assembler-level optimizations [97]. The PM algorithm maps the particles to a grid of mesh cells and calculates the gravitational potential using a Fast Fourier Transform. It accurately calculates the forces of long distance interactions, but is less accurate in computing forces over short distances, for which the tree algorithm is applied instead.

The code has been modified to allow simulations on massively parallel machines [65], in which the code uses a recursive multi-section scheme [84] to divide the workload over the processes. The workload is redistributed during each step of the simulation so that the force calculation time remains equal for all processes.

5.2.1 Parallelization across supercomputers

We have developed SUSHI to efficiently use multiple supercomputers for our simulations. We coupled the TreePM code with the MPWide communication library and developed
a cross-supercomputer parallelization scheme. Because the wide area network has performance and topological characteristics that are different from local networks, the communication scheme between sites is different from the scheme used between nodes. When SUSHI is deployed across sites, each site is connected to two neighboring sites to form a ring topology.

5.2.1.1 Communication scheme

A simulation using SUSHI consists of four communication phases per step. During these phases the simulation:

1. Exchanges mesh densities.
2. Collects sample particles to determine the site boundaries,
3. Exchanges tree structures with neighboring sites.
4. Migrates particles between neighboring sites.

When exchanging mesh densities, the mesh cells from all sites are aggregated to obtain the global mesh density. This mesh density is then used to perform PM integration. At the time of writing the PM integration is still a serial operation, but we are in the process of implementing a parallelized version. The mesh densities are gathered using a ring communication over all sites. The force calculation time and time step information of each site are also accumulated during this phase.

At each step the site boundaries are updated, based on the obtained force calculation times and the current particle distribution. To gain information on the particle distribution, the communication processes on each site gather sample particles from all other processes. These sampled particles are then gathered from all sites using a ring communication.

Before the tree force calculations can be performed, each site constructs a local essential tree structure. This local essential tree is a set of particles and tree cells which are used to compute the force exchanges, and partially resides on neighboring sites. To obtain a complete local essential tree, each site therefore requires the missing tree data from its neighbors. The simulation gathers the tree data using one neighbor exchange for two-site runs, or two exchanges for runs across three or more sites.

After the force calculations have been performed, the simulation updates the positions of all particles. At this point, some particles may be located outside the site boundaries and need to be migrated to a neighboring site. This communication requires one neighbor exchange for two site runs, or two exchanges for runs across three or more sites.

5.2.1.2 Domain decomposition

We have implemented a hierarchical decomposition scheme to distribute the particles among supercomputers. This scheme uses a one-dimensional slab decomposition to
distribute the particles among the sites, and a recursive multi-section scheme over three dimensions to distribute the particles among the processes. Because the domain decomposition between sites is one-dimensional, each supercomputer only exchanges particles and tree structures with two other machines. The data exchanges between sites can therefore be done efficiently in a ring topology. Most supercomputers are connected by optical paths in either a ring or star topology, if they are connected at all.

The simulation adjusts the distribution of particles among supercomputers at runtime, such that the force calculation time is kept equal on all sites. The number of particles \( N_i \) on a single site \( i \) for a run performed over \( s \) sites is therefore given by

\[
N_i \sim \frac{N \left( t_{\text{calc},i} \right)^{-1}}{\sum_{j=0}^{s-1} \left( t_{\text{calc},j} \right)^{-1}}.
\]  

(5.1)

The force calculation time on site \( i \) during the previous step is given by \( t_{\text{calc},i} \). The load balancing algorithm can be suppressed by explicitly limiting the boundary moving length per step, thereby limiting the communication traffic between sites.

5.2.1.3 Implementation of communication routines

We use the MPWide communication library, which is described in Chapter 6, to perform wide area message passing within SUSHI. The implementation of the communication routines has few external dependencies, which makes it easy to install on different platforms. The wide area communications in MPWide are performed using parallel TCP streams. In cases where the supercomputers can only be indirectly connected, we use MPWide-based port forwarding programs on the intermediate nodes to establish a communication path. During the development of SUSHI, we found that it is not trivial to obtain optimal communication performance between supercomputers. Therefore we added several features that can be used to improve the communication performance. The communication routines within SUSHI can be customized for individual paths between supercomputers. Settings that can be adjusted for each connection include the number of parallel TCP streams, the TCP buffer sizes and the size of data packages that are written to or read from the sockets. To improve the performance on some long distance networks, MPWide also supports software-based packet pacing.

5.3 Performance model

We have developed a performance model for SUSHI. The model can be applied to predict the execution time and scalability of simulations that run across supercomputers. To make an accurate prediction we require several architecture-dependent parameters. These include machine-specific parameters such as the time spent on a single tree interaction (given by \( \tau_{\text{tree}} \)) and a single particle mesh interaction (\( \tau_{\text{pm}} \)). In addition, we need a few parameters for the networks used in the simulation, which are the round-trip time (given by \( \lambda_{\text{lan}} \) and \( \lambda_{\text{wan}} \)) and the available bandwidth (\( \sigma_{\text{lan}} \) and \( \sigma_{\text{wan}} \)). The values
of these parameters can be obtained through minor network tests and a small single process test simulation on each site. The values used for our experiments are found in Tab. 5.5 for each supercomputer, in Tab. 5.2 for each national grid site and in Tab. 5.6 for the local and wide area networks.

5.3.1 Single supercomputer

The time required for one TreePM integration step using $p$ processes on a single supercomputer ($t_{\text{exec}}(1,p)$) consists of time spent on tree force calculations ($t_{\text{tree}}$), time spent on PM integration ($t_{\text{pm}}$) and the communication overhead ($t_{\text{comm}}$):

$$t_{\text{exec}}(1,p) = t_{\text{tree}} + t_{\text{pm}} + t_{\text{comm}}. \quad (5.2)$$

The time spent on tree integration ($t_{\text{tree}}$) is dominated by force calculations. The force calculation time is obtained by multiplying the time required to perform a single force interaction ($\tau_{\text{tree}}$) with the total number of tree interactions ($n_{\text{int}}$) and dividing it by the number of processes ($p$). Creating interaction lists and constructing the tree introduce additional overhead that scales with the number of interactions. To account for this in a simplified way, we multiply the time spent on force calculations with a factor $1.2 \, \frac{1}{1}$. The time spent on tree integration then becomes

$$t_{\text{tree}} = 1.2 \times \frac{\tau_{\text{tree}} n_{\text{int}}}{p}. \quad (5.3)$$

The number of interactions per simulation step depends on many parameters including, but not limited to, the number of particles ($N$), the opening angle of the tree integration ($\theta$) and the number of mesh cells ($M$). We have performed several runs over a single supercomputer and fitted the number of interactions for cosmological datasets, which results in

$$n_{\text{int}} = 4.6 \times 10^{-4} N^{1.0667} \frac{N^{1/12}}{\theta^{1.35}} \frac{M^{1/12}}{\sqrt{2}}. \quad (5.4)$$

In general, $n_{\text{int}} \propto \theta^{-1.35}$ although this estimate may not be accurate if $\theta \gtrsim 0.2$ or $\theta \lesssim 0.75$. In these regimes, the number of interactions depends more strongly on other tree integration settings, such as the maximum number of particles allowed to share interaction lists.

We calculate the time spent on PM integration ($t_{\text{pm}}$) by multiplying the number of mesh cells ($M$) with the time required to do PM integration for one mesh cell ($\tau_{\text{pm}}$). The total time spent on PM integration then becomes

$$t_{\text{pm}} = \tau_{\text{pm}} M. \quad (5.5)$$

$^1$This value is based on timings from single site runs using $N = 128^3$ up to $N = 2048^3$
We decompose the communication time \( t_{\text{comm}} \) into time spent to initially traverse the networks \( t_l \), which is latency-bound and time spent on data throughput \( t_b \), which is limited by the available bandwidth of the local network. Therefore,

\[
t_{\text{comm}} = t_l + t_b. \tag{5.6}
\]

For each step, the code performs 18 collective operations containing \( \log p \) communication steps and two all-to-all communications with \( p \) communication steps. The time spent in latency \( t_l \) is calculated by multiplying the number of communication steps with the network round-trip time \( \lambda_{\text{lan}} \). As a result,

\[
t_l = \lambda_{\text{lan}} (18 \log p + 2p), \tag{5.7}
\]

We determine the time spent on data throughput \( t_b \) by dividing the data volume of the local area communications by the network bandwidth \( \sigma_{\text{lan}} \). The communication data volume consists of three dominant parts. These are the grid of mesh cells (which consist of \( 4M \) bytes in total), the local essential tree structures (estimated to be \( (48/\theta + 24) N^{2/3} \) bytes for \( 0.2 < \theta < 0.75 \)), and the sample particles which are used to determine the node boundaries (12\( N r_{\text{samp}} \) bytes in total). Here, \( r_{\text{samp}} \), which we set to 1/10000 for large calculations, is the ratio of sampled particles relative to \( N \). The time spent on data throughput is then

\[
t_b = \frac{4M + (48/\theta + 24) N^{2/3} + 12N r_{\text{samp}}}{\sigma_{\text{lan}}}. \tag{5.8}
\]

Additional communication is required to migrate particles between sites. The data volume of this communication is relatively large during initial simulation steps, but becomes negligible once sufficient steps have been taken to adequately balance the workload. A detailed review of the communication characteristics of the code is presented in [65].

### 5.3.2 Multiple supercomputers

We calculate the wall-clock time required for a single TreePM integration step using \( p \) processes in total across \( s \) supercomputers \( (t_{\text{exec}}(s,p)) \) by adding the wide area communication overhead \( w_{\text{comm}} \) to the time spent on tree integrations \( t_{\text{tree}} \), the time spent on PM integration \( t_{\text{pm}} \) and the time spent on local area communications \( t_{\text{comm}} \). The execution time per step is therefore

\[
t_{\text{exec}}(s,p) = t_{\text{tree}} + t_{\text{pm}} + t_{\text{comm}} + w_{\text{comm}}. \tag{5.9}
\]

Here, we calculate \( t_{\text{tree}} \) using Eq. 5.3, \( t_{\text{pm}} \) using Eq. 5.5 and \( t_{\text{comm}} \) using Eq. 5.6. The communication overhead on the wide area network \( w_{\text{comm}} \) consists of the time...
spent in latency \((w_l)\) and the time spent on data throughput \((w_b)\) on the wide area network. As a result,

\[
w_{\text{comm}} = w_l + w_b. \tag{5.10}
\]

The code performs five blocking gather operations over all sites per step. These gathers are performed using a ring scheme, which requires \(s - 1\) neighbor exchanges per gather. We also require four blocking exchanges with each of the two neighboring sites. The total number of exchanges is then equal to \(5s + 3\) and, the total time spent in latency \((w_l)\) then becomes

\[
w_l = \lambda_{\text{wan}} (5s + 3). \tag{5.11}
\]

Here, \(\lambda_{\text{wan}}\) is the network round-trip time between sites.

We calculate the time spent on wide area data throughput \((w_b)\) by dividing the data volume of the wide area communications by the bandwidth capacity of the wide area network \((\sigma_{\text{wan}})\). The volume of the exchanged data between sites is similar to the data volume between nodes with two exceptions. First, the exchange of mesh densities requires one float per mesh cell per site and second, because SUSHI uses a 1D decomposition between sites in the x direction we exchange and compare only the Cartesian x coordinates of sampled particles. The data volume for the exchange of sampled particles is therefore three times smaller. The total time spent on wide area data throughput is

\[
w_b = \frac{4sM + (48/\theta + 24) N^{2/3} + 4N r_{\text{samp}}}{\sigma_{\text{wan}}} . \tag{5.12}
\]

### 5.3.3 Scalability across sites

The speedup of a simulation across sites, \(S(s)\), is defined by dividing the time required for an integration step on 1 site using \(p\) processes \((t_{\text{exec}}(1,p))\) by the time required for an integration step over \(s\) sites using a total of \(s \cdot p\) processes \((t_{\text{exec}}(s,sp))\). It is therefore given by

\[
S(s) = \frac{t_{\text{exec}}(1,p)}{t_{\text{exec}}(s,sp)}. \tag{5.13}
\]

The efficiency of a simulation across sites, \(E(s)\), is calculated by dividing the time require for an integration step on 1 site using \(p\) processes by the time required for an integration step over \(s\) sites using a total of \(p\) processes \((t_{\text{exec}}(s,p))\). The efficiency is then

\[
E(s) = \frac{t_{\text{exec}}(1,p)}{t_{\text{exec}}(s,p)}. \tag{5.14}
\]
Table 5.1: Initial condition and accuracy parameters used for our test simulations. The maximum number of particles allowed to share an interactions list in the tree integration is given by \( n_{crit} \).

5.4 Experiments

We have tested SUSHI for performance on a grid of 5 Beowulf clusters, as well as an infrastructure consisting of four supercomputers. Each simulation lasts for 100 integration steps, and uses an opening angle of \( \theta = 0.3 \) when \( z > 10 \) and \( \theta = 0.5 \) when \( z \leq 10 \). For each opening angle we measured the total wall-clock time and communication time per step, averaged over 10 steps. All measurements were made near \( z = 10 \), approximately 460 million years after the Big Bang. A full listing of the simulation parameters and initial condition characteristics of our experiments is given in Tab. 5.1.

We compare the results of our experiments with predictions from our performance model. To do so, we measured the value of several machine constants using local tests and provide them in Tab. 5.2 and Tab. 5.5. The network constants used in the performance model are given in Tab. 5.6. As all simulations lasted for only 100 integration steps, particle exchanges were still performed to improve the distribution of work. We have added the measured average data volume of these exchanges to the data volume for wide area communications in our model. For full-length simulations, the data volume of these particle exchanges is a negligible part of the total communication volume.

5.4.1 DAS-3 experiment setup

The Distributed ASCI Supercomputer 3 (DAS-3 [24]) is a Dutch infrastructure that consists of 5 PC clusters within The Netherlands. The clusters use 10 Gbps networking internally, while the head nodes of each site are connected to regular internet in a star topology. Using end-to-end message passing tests we were able to achieve a performance of up to 1.25 Gbps between sites. The exact specifications of the five DAS-3 sites can
5.4 Experiments

5.4.2 DAS-3 results

The timing measurements of our experiments can be found respectively in the left panel of Fig. 5.1 for the runs with $N = 256^3$ and in Fig. 5.2 for the runs with $N = 512^3$. Here we see that the communication overhead becomes marginally higher as we increase the number of sites in the simulation with $N = 256^3$ and $M = 128^3$ mesh cells, both for runs with $\theta = 0.3$ and with $\theta = 0.5$. The measured communication overhead for simulations with $N = 512^3$ increases more steeply with $s$, because the larger communication data volume results in a higher traffic load on the internet lines between the sites.

The calculation time per step is almost twice as high for runs using $\theta = 0.3$ compared to runs with $\theta = 0.5$, while the communication overhead is approximately equal. Integration steps with a smaller $\theta$ therefore scale more favorably when increasing $s$.

We provide a comparison between our timing results from the experiments and predictions from our performance model in Tab. 5.3 and Tab. 5.4. Here we see that the achieved performance roughly matches the model predictions. For the runs with $N = 512^3$, the model provides slightly optimistic predictions for $M = 128^3$ and slightly pessimistic predictions for $M = 256^3$. This may be caused by the time required for
Figure 5.1: Wall-clock time (given by circles) and communication time (given by squares) per step averaged over 10 steps as a function of the number of sites $s$ for two different simulations. Results are given for a simulation with $N = 256^3$ and $M = 128^3$ over 60 processes using the DAS-3 (left panel) and across multiple supercomputers (right panel). The runs using $\theta = 0.5$ are given by open symbols and runs using $\theta = 0.3$ are given by solid symbols. The standard deviation of each measurement is shown by an error bar.
Figure 5.2: As Fig. 5.1, but for simulations on the DAS-3 using $N = 512^3$ and $M = 128^3$ (left panel), and simulations using $N = 512^3$ and $M = 256^3$ (right panel). All simulations were run over 120 processes in total.
High Performance Gravitational $N$-body Simulations on a Planet-wide Distributed Supercomputer

5.4.3 Gravitational Billion Body Project experiment setup

We have run a number of test simulations across multiple supercomputers to measure the performance of our code and to test the validity of our performance model. The simulations, which use datasets consisting of $N = 256^3$, $N = 512^3$ and $N = 1024^3$ dark matter particles, were run across up to four supercomputers. We provide the technical characteristics of each supercomputer in Tab. 5.5. The three European supercomputers are connected to the DEISA shared network, which can be used without prior reservation although some user-space tuning is required to get acceptable performance. The fourth supercomputer resides in Japan and is connected to the other three machines with a 10 Gbps intercontinental light path.

<table>
<thead>
<tr>
<th>$N^{1/3}$</th>
<th>$M^{1/3}$</th>
<th>np</th>
<th>sites</th>
<th>comm.</th>
<th>exec.</th>
<th>$t_{\text{comm}}$</th>
<th>$t_{\text{exec}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>128</td>
<td>60</td>
<td>1</td>
<td>0.64</td>
<td>13.6</td>
<td>0.22</td>
<td>12.8</td>
</tr>
<tr>
<td>256</td>
<td>128</td>
<td>60</td>
<td>2</td>
<td>1.28</td>
<td>14.8</td>
<td>0.83</td>
<td>13.9</td>
</tr>
<tr>
<td>256</td>
<td>128</td>
<td>60</td>
<td>3</td>
<td>2.15</td>
<td>16.3</td>
<td>1.73</td>
<td>14.3</td>
</tr>
<tr>
<td>256</td>
<td>128</td>
<td>60</td>
<td>4</td>
<td>1.70</td>
<td>15.5</td>
<td>2.96</td>
<td>15.5</td>
</tr>
<tr>
<td>256</td>
<td>128</td>
<td>60</td>
<td>5</td>
<td>2.16</td>
<td>16.2</td>
<td>4.50</td>
<td>17.1</td>
</tr>
<tr>
<td>512</td>
<td>128</td>
<td>120</td>
<td>1</td>
<td>2.28</td>
<td>62.8</td>
<td>0.58</td>
<td>65.8</td>
</tr>
<tr>
<td>512</td>
<td>128</td>
<td>120</td>
<td>2</td>
<td>4.75</td>
<td>73.4</td>
<td>3.26</td>
<td>71.2</td>
</tr>
<tr>
<td>512</td>
<td>128</td>
<td>120</td>
<td>3</td>
<td>11.1</td>
<td>84.6</td>
<td>6.24</td>
<td>71.4</td>
</tr>
<tr>
<td>512</td>
<td>128</td>
<td>120</td>
<td>4</td>
<td>19.0</td>
<td>94.0</td>
<td>9.53</td>
<td>74.8</td>
</tr>
<tr>
<td>512</td>
<td>128</td>
<td>120</td>
<td>5</td>
<td>22.8</td>
<td>98.7</td>
<td>13.2</td>
<td>78.4</td>
</tr>
<tr>
<td>512</td>
<td>256</td>
<td>120</td>
<td>1</td>
<td>4.42</td>
<td>53.1</td>
<td>1.14</td>
<td>61.2</td>
</tr>
<tr>
<td>512</td>
<td>256</td>
<td>120</td>
<td>2</td>
<td>8.37</td>
<td>61.9</td>
<td>6.06</td>
<td>68.4</td>
</tr>
<tr>
<td>512</td>
<td>256</td>
<td>120</td>
<td>3</td>
<td>8.75</td>
<td>58.0</td>
<td>13.5</td>
<td>73.6</td>
</tr>
<tr>
<td>512</td>
<td>256</td>
<td>120</td>
<td>4</td>
<td>18.7</td>
<td>75.6</td>
<td>23.5</td>
<td>84.0</td>
</tr>
<tr>
<td>512</td>
<td>256</td>
<td>120</td>
<td>5</td>
<td>27.0</td>
<td>80.2</td>
<td>36.1</td>
<td>96.8</td>
</tr>
</tbody>
</table>

Table 5.3: List of runs performed on the DAS-3, which use $\theta = 0.3$. The cube root of the number of particles and mesh cells are given in the first and second column, the number of processes and the number of sites involved in the third and fourth column. The last four columns contain respectively the measured average communication time per step over 10 steps, the average wall-clock time per step over 10 steps, and the communication and wall-clock time as predicted by our performance model.

the exchange of the local essential tree. In our model the size of the local essential tree is only dependent on $N$ and $\theta$. However, in our experiments we also find a minor correlation with the number of mesh cells used, as the range of the tree integration is equal to three mesh cell lengths.
5.4 Experiments

<table>
<thead>
<tr>
<th>$N^{1/3}$</th>
<th>$M^{1/3}$</th>
<th>np</th>
<th>sites</th>
<th>comm. measure</th>
<th>exec. measure</th>
<th>$t_{\text{comm}}$ model</th>
<th>$t_{\text{exec}}$ model</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>128</td>
<td>60</td>
<td>1</td>
<td>0.63</td>
<td>8.35</td>
<td>0.18</td>
<td>6.83</td>
</tr>
<tr>
<td>256</td>
<td>128</td>
<td>60</td>
<td>2</td>
<td>0.88</td>
<td>8.49</td>
<td>0.71</td>
<td>7.61</td>
</tr>
<tr>
<td>256</td>
<td>128</td>
<td>60</td>
<td>3</td>
<td>2.13</td>
<td>9.56</td>
<td>1.53</td>
<td>8.18</td>
</tr>
<tr>
<td>256</td>
<td>128</td>
<td>60</td>
<td>4</td>
<td>1.59</td>
<td>9.45</td>
<td>2.68</td>
<td>9.38</td>
</tr>
<tr>
<td>256</td>
<td>128</td>
<td>60</td>
<td>5</td>
<td>1.94</td>
<td>9.98</td>
<td>4.14</td>
<td>10.9</td>
</tr>
<tr>
<td>512</td>
<td>128</td>
<td>120</td>
<td>1</td>
<td>1.80</td>
<td>32.7</td>
<td>0.42</td>
<td>33.5</td>
</tr>
<tr>
<td>512</td>
<td>128</td>
<td>120</td>
<td>2</td>
<td>3.90</td>
<td>38.2</td>
<td>2.78</td>
<td>37.2</td>
</tr>
<tr>
<td>512</td>
<td>128</td>
<td>120</td>
<td>3</td>
<td>7.08</td>
<td>43.1</td>
<td>5.44</td>
<td>38.5</td>
</tr>
<tr>
<td>512</td>
<td>128</td>
<td>120</td>
<td>4</td>
<td>11.1</td>
<td>49.3</td>
<td>8.41</td>
<td>41.5</td>
</tr>
<tr>
<td>512</td>
<td>128</td>
<td>120</td>
<td>5</td>
<td>15.7</td>
<td>54.1</td>
<td>11.7</td>
<td>44.9</td>
</tr>
<tr>
<td>512</td>
<td>256</td>
<td>120</td>
<td>1</td>
<td>2.78</td>
<td>31.5</td>
<td>0.98</td>
<td>34.0</td>
</tr>
<tr>
<td>512</td>
<td>256</td>
<td>120</td>
<td>2</td>
<td>9.07</td>
<td>40.9</td>
<td>5.58</td>
<td>39.8</td>
</tr>
<tr>
<td>512</td>
<td>256</td>
<td>120</td>
<td>3</td>
<td>9.59</td>
<td>38.9</td>
<td>12.7</td>
<td>45.8</td>
</tr>
<tr>
<td>512</td>
<td>256</td>
<td>120</td>
<td>4</td>
<td>13.3</td>
<td>46.0</td>
<td>22.4</td>
<td>55.9</td>
</tr>
<tr>
<td>512</td>
<td>256</td>
<td>120</td>
<td>5</td>
<td>26.5</td>
<td>57.7</td>
<td>34.7</td>
<td>68.3</td>
</tr>
</tbody>
</table>

Table 5.4: As Tab. 5.3, but with $\theta = 0.5$ instead of $\theta = 0.3$.

5.4.3.1 Network configuration

In the shared DEISA network we applied the following settings to MPWide: First, all communication paths used at least 16 parallel streams and messages were sent and received in chunks of 256 kB per stream. These settings allow us to reach $\sim 100$ MB/s sustained throughput on the network between Amsterdam and Edinburgh. Second, we used software-based packet pacing to reduce the CPU usage of MPWide on the communication nodes. This had little impact on the communication performance of the application, but was required because some of the communication nodes were non-dedicated.

Although the light path between Amsterdam and Tokyo did not have an optimal TCP configuration, we were able to achieve a sustained throughput rate of $\sim 100$ MB/s by tuning our MPWide settings. To accomplish this throughput rate, we used 64 parallel TCP streams, limited our burst exchange rate to 100 MB/s per stream using packet pacing and performed send/receive operations in chunks of 8kB per stream. In comparison, when using a single TCP stream, our throughput was limited to 10 MB/s, even though the TCP buffering size was set to more than 30 MB on the end nodes. We believe that this limitation arises from TCP buffer limitations on one of the intermediary nodes on the light path.

Since most of the supercomputers are connected to the high speed network through specialized communication nodes, we are required to forward our messages through these nodes to exchange data between supercomputers. This forwarding is done in user space with MPWide Forwarder programs. A graphical overview of the network topology,
Table 5.5: Technical specifications of the IBM Power 6 supercomputer in Amsterdam (The Netherlands) and the Cray-XT4 supercomputers in Espoo (Finland), Edinburgh (United Kingdom) and Tokyo (Japan). Note that Huygens is connected to both the 10Gbps DEISA network and the 10Gbps light path between Amsterdam and Tokyo.

including the communication nodes as well as latency and bandwidth characteristics for each network path, can be found in Fig. 5.3.

5.4.4 GBBP results

The timing measurements of several simulations using \( N = 256^3 \) are given in the right panel of Fig. 5.1 and measurements of simulations using \( N = 512^3 \) are given in Fig. 5.4. We provide the timing results and model predictions for all our experiments in Tables 5.7 and 5.8.

We have performed a number of runs with \( N = 256^3 \) across up to three supercomputers. The runs over two supercomputers in the DEISA network have a communication overhead between 1.08 and 1.54 seconds. This constitutes between 8 and 13\% of the

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda_{\text{lan}} )</td>
<td>( 1.0 \times 10^{-4} )</td>
<td>[s]</td>
</tr>
<tr>
<td>( \lambda_{\text{wan}} )</td>
<td>( 3.0 \times 10^{-3} )</td>
<td>[s]</td>
</tr>
<tr>
<td>( \sigma_{\text{lan}} )</td>
<td>( 1.0 \times 10^{8} )</td>
<td>[bytes/s]</td>
</tr>
<tr>
<td>( \sigma_{\text{wan}} )</td>
<td>( 5.0 \times 10^{7} / s - 1 )</td>
<td>[bytes/s]</td>
</tr>
</tbody>
</table>

Table 5.6: List of network parameters used for modelling the performance of our runs. The name of the constant is given in the first column, followed by a brief description of the constant in the second column, the value used for modelling the DAS-3 runs in the third column, the value used for modelling the GBBP runs in the fourth column and the unit used in the fifth column. Since the wide area network in the DAS-3 resembles a star topology, we divide the available bandwidth by \( s - 1 \).
5.4 Experiments

Figure 5.3: Network overview of a cosmological simulation across four supercomputers. The network latency and maximum sustained throughput for 64 MB message exchanges over each network path is given on the left, and an overview of the network topology is given on the right. Here, the communication nodes are indicated by the solid gray boxes.

Figure 5.4: As Fig. 5.2, but these simulations were run on the planet-wide network of supercomputers $s$. The wall-clock time is given as a function of the number of supercomputers. The ordering and technical specifications of the supercomputers we used are given in Tab. 5.5.
Figure 5.5: Snapshots of a simulation with $N = 512^3$ over four supercomputers. The snapshots are taken at $z = 5.65$, approximately one billion years after the Big Bang, when some dark matter structures can already be observed. The slices have been placed to match the volumes on the supercomputers that reside in Edinburgh (top left), Espoo (top right), Tokyo (bottom left) and Amsterdam (bottom right).
total runtime for runs with $\theta = 0.3$ and between 15 and 20\% for runs with $\theta = 0.5$. The run with $N = 256^3$ between Edinburgh and Amsterdam has less communication overhead than the run between Espoo and Amsterdam, despite the 1 Gbps bandwidth limitation on the connection to Edinburgh. Our run between Amsterdam and Espoo suffered from a high background load, which caused the communication time to fluctuate over 10 steps with $\sigma \sim 0.25$ s, compared to $\sigma \sim 0.04$ s for the run between Amsterdam and Edinburgh. The run over three sites has a higher overhead than the runs over two sites, mainly due to using the direct connection between Edinburgh and Espoo, which is poorly optimized.

The wall-clock time of the simulations with $N = 512^3$ is generally dominated by calculations, although the communication overhead becomes higher as we increase $s$. The runs over two sites (Espoo and Amsterdam) spend about 4-5 seconds on communication, which is less than 10 percent of the total execution time for the run using $\theta = 0.3$. For simulations with $M = 128^3$, the use of three supercomputers rather than two does not significantly increase the communication overhead. However, when we run simulations with $N = 512^3$ and $M = 256^3$ the use of a third supercomputer doubles the communication overhead. This increase in overhead can be attributed to the larger mesh size, as the now larger data volume of the mesh exchange scales with $s$.

We provide an overview of the time spent per simulation step for the run with $N = 512^3$ and $M = 128^3$ over three sites (Edinburgh, Espoo and Amsterdam) in Fig. 5.6. For this run the total communication overhead of the simulation code remains limited and relatively stable, achieving slightly lower values at later steps where $\theta = 0.5$. A decomposition of the communication overhead for this run is given in the bottom panel of Fig. 5.6. The time required for exchanging the local essential tree constitutes about half of the total communication overhead. The simulation changes the opening angle at step 46, from $\theta = 0.3$ to $\theta = 0.5$. As a result, the size of the local essential tree becomes smaller and less time is spent on exchanging the local essential trees. The time spent on the other communications remains roughly constant throughout this run.

We have had a brief opportunity to run across all four supercomputers, which allowed us to run a simulation with $N = 512^3$. We provide a density plot of the particle data present at each site at the end of this run in Fig. 5.5. In addition, the timing measurements of the run across four sites are given in the left panel of Fig. 5.4. Here we observe an increase in communication time from 3.8-4.8 seconds for a run over the three DEISA sites to 12.9-19.3 seconds for the run over all four sites. However, the runs over three sites relay messages between Edinburgh and Espoo through the supercomputer in Amsterdam, whereas the four site run uses the direct connection between these sites, which is poorly optimized. If we use the direct connection for a run across three sites the communication overhead increases from 3.8-4.8 seconds to 10.1-13.2 seconds per step (see Tables 5.7 and 5.8). Therefore, the communication overhead of a run across four sites may be reduced by a factor of two if we would relay the messages between Espoo and Edinburgh over Amsterdam.

We also ran several simulations with $N = 1024^3$. The load balancing increases the communication volume of these larger runs considerably, as 850MB of particle data was exchanged on average during each step. However, the communication time
Figure 5.6: Performance results of a simulation with $N = 512^3$ and $M = 128^3$ using 120 processes in total across supercomputers in Amsterdam, Edinburgh and Espoo as a function of the simulation step number. The wall-clock time spent per step on calculations (top red area) and communications (bottom black area) is given in the top figure. Time spent on the four communication phases is given in the bottom figure. These phases are (from top to bottom) the migration of particles between sites, the exchanges of sample particles for determining the site boundaries, the local essential tree exchanges (PP) and the mesh cell exchanges (PM).
Table 5.7: List of runs performed for GBBP that use $\theta = 0.3$. The cube root of the number of particles and mesh cells are given in the first and second column, the number of processes and the supercomputers involved in the third and fourth column. Here, the letters correspond to supercomputers in Espoo (given by “H”), (E)dinburgh, (A)msterdam and (T)okyo. For runs over three and four sites, we relayed all traffic through the supercomputer in Amsterdam except for the runs marked with an asterisk, which used the direct network between Edinburgh and Espoo. The last four columns contain respectively the measured average communication time per step over 10 steps, the average wall-clock time per step over 10 steps, and the communication and wall-clock time as predicted by our performance model.
High Performance Gravitational $N$-body Simulations on a Planet-wide Distributed Supercomputer

<table>
<thead>
<tr>
<th>$N^{1/3}$</th>
<th>$M^{1/3}$</th>
<th>np</th>
<th>sites</th>
<th>comm. measure</th>
<th>exec. measure</th>
<th>$t_{\text{comm}}$</th>
<th>$t_{\text{exec}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[s]</td>
<td>[s]</td>
<td>[s]</td>
<td>[s]</td>
</tr>
<tr>
<td>256</td>
<td>128</td>
<td>60</td>
<td>A</td>
<td>0.20</td>
<td>6.45</td>
<td>0.03</td>
<td>5.81</td>
</tr>
<tr>
<td>256</td>
<td>128</td>
<td>60</td>
<td>HA</td>
<td>1.04</td>
<td>7.09</td>
<td>0.97</td>
<td>6.75</td>
</tr>
<tr>
<td>256</td>
<td>128</td>
<td>60</td>
<td>EA</td>
<td>1.48</td>
<td>7.57</td>
<td>0.97</td>
<td>6.75</td>
</tr>
<tr>
<td>256</td>
<td>128</td>
<td>60</td>
<td>AT</td>
<td>3.58</td>
<td>9.86</td>
<td>4.14</td>
<td>9.93</td>
</tr>
<tr>
<td>256</td>
<td>128</td>
<td>60</td>
<td>HEA*</td>
<td>5.83</td>
<td>12.1</td>
<td>1.58</td>
<td>7.36</td>
</tr>
<tr>
<td>512</td>
<td>128</td>
<td>120</td>
<td>A</td>
<td>0.99</td>
<td>35.8</td>
<td>0.05</td>
<td>30.0</td>
</tr>
<tr>
<td>512</td>
<td>128</td>
<td>120</td>
<td>HA</td>
<td>4.52</td>
<td>32.8</td>
<td>2.82</td>
<td>28.7</td>
</tr>
<tr>
<td>512</td>
<td>128</td>
<td>120</td>
<td>HEA</td>
<td>3.80</td>
<td>29.0</td>
<td>3.88</td>
<td>28.5</td>
</tr>
<tr>
<td>512</td>
<td>128</td>
<td>120</td>
<td>HEA*</td>
<td>10.9</td>
<td>40.5</td>
<td>3.88</td>
<td>28.5</td>
</tr>
<tr>
<td>512</td>
<td>128</td>
<td>120</td>
<td>HEAT*</td>
<td>12.9</td>
<td>39.7</td>
<td>9.23</td>
<td>33.7</td>
</tr>
<tr>
<td>512</td>
<td>256</td>
<td>120</td>
<td>A</td>
<td>1.31</td>
<td>26.8</td>
<td>0.07</td>
<td>27.9</td>
</tr>
<tr>
<td>512</td>
<td>256</td>
<td>120</td>
<td>HA</td>
<td>5.38</td>
<td>29.5</td>
<td>5.09</td>
<td>29.3</td>
</tr>
<tr>
<td>512</td>
<td>256</td>
<td>120</td>
<td>EA</td>
<td>6.99</td>
<td>29.4</td>
<td>5.09</td>
<td>29.5</td>
</tr>
<tr>
<td>512</td>
<td>256</td>
<td>120</td>
<td>AT</td>
<td>9.64</td>
<td>32.2</td>
<td>8.26</td>
<td>33.4</td>
</tr>
<tr>
<td>512</td>
<td>256</td>
<td>120</td>
<td>HEA</td>
<td>12.7</td>
<td>36.1</td>
<td>7.26</td>
<td>30.4</td>
</tr>
<tr>
<td>1024</td>
<td>256</td>
<td>240</td>
<td>EA</td>
<td>19.1</td>
<td>128</td>
<td>22.6</td>
<td>144</td>
</tr>
<tr>
<td>2048</td>
<td>256</td>
<td>750</td>
<td>AT</td>
<td>46.5</td>
<td>483</td>
<td>17.7</td>
<td>464</td>
</tr>
</tbody>
</table>

Table 5.8: As Tab. 5.7 but with $\theta = 0.5$ instead of $\theta = 0.3$. The run with $N = 2048^3$ used an older version of the code and different settings, and is described in Chapter 4.
still constitutes only one-tenth of the total run-time for the simulation across three supercomputers. We also performed one run with \( N = 2048^3 \) using an older version of the code and where \( r_{\text{samp}} \) was set to 5000. The run used 500 processes in Amsterdam and 250 processes in Tokyo and is described in Chapter 4. We measured a communication overhead of \( \sim 10\% \).

### 5.5 Scalability of \( N \)-body simulations across supercomputers

In this section we use our performance model to predict the scalability of \( N \)-body code across grid sites. In the first part of this section we examine the predicted speedup and efficiency when scaling three cosmological \( N \)-body problems across multiple supercomputers. In the second part we apply existing performance models for tree codes (with block and shared time step schemes) and direct-method codes to predict the scalability of three stellar \( N \)-body problems across supercomputers. In the third part we predict the efficiency of cosmological \( N \)-body simulations over 8 sites as a function of the available bandwidth between supercomputers. We provide an overview of the three cosmological problems in Tab. 5.9, and an overview of the three stellar problems in Tab. 5.11.

These problems are mapped to a global grid infrastructure, which has a network latency of 0.3 s and a bandwidth capacity of 400 MB/s between supercomputers. The machine constants are similar to the ones we used for our runs, and can be found in Tab. 5.10. To limit the complexity of our analysis, we assume an identical calculation speed for all cores on all sites. Our performance predictions use an opening angle \( \theta = 0.5 \).

#### 5.5.1 Speedup and efficiency predictions for TreePM simulations

The predicted speedup \( S(s) \) (as defined in Eq. 5.13) for three cosmological simulations as a function of the number of supercomputers \( s \) can be found in Fig. 5.7. As the number of processes per site remains fixed, the total number of processes \( p \) increases linearly with \( s \). All example problems run efficiently over up to 3 sites. The simulation

<table>
<thead>
<tr>
<th>Integrator</th>
<th>( N )</th>
<th>( M )</th>
<th>np 1</th>
<th>np 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>TreePM 2048</td>
<td>2048^3</td>
<td>256^3</td>
<td>128</td>
<td>2048</td>
</tr>
<tr>
<td>TreePM 2048</td>
<td>2048^3</td>
<td>1024^3</td>
<td>128</td>
<td>2048</td>
</tr>
<tr>
<td>TreePM 8192</td>
<td>8192^3</td>
<td>1024^3</td>
<td>4096</td>
<td>32768</td>
</tr>
</tbody>
</table>

Table 5.9: Description of the cosmological \( N \)-body problems used for the scalability analysis across supercomputers. The type of integrator is given in the first column, the number of particles in the second column and the number of mesh cells in the third column. The last two columns contain respectively the number of processes per site for the speedup analysis and the total number of processes for the efficiency and bandwidth analysis.
with $N = 2048^3$ and $M = 256^3$ scales well as $s$ increases and obtains a speedup of $\sim 13$ when $s = 16$. When scaling up beyond $s \sim 25$, the speedup diminishes as the simulation becomes dominated by communication.

The simulation with $N = 2048^3$ and $M = 1024^3$ does not scale as well and only achieves a good speedup when run across a few sites. Here, the speedup curve as a function of $s$ begins to flatten at $s = 5$, due to the serial integration of the larger mesh. For $s \gtrsim 16$, the communication overhead begins to dominate performance and the speedup decreases for higher $s$. The speedup of the run with $N = 8192^3$ and $M = 1024^3$ scales better with $s$ than the speedup of the run with $N = 2048^3$ because it spends more time per step on tree force calculations.

We provide the predicted efficiency of three simulations over $s$ supercomputers relative to a simulation over one supercomputer, $E(s)$, using the same number of processes in Fig. 5.8. Here, the run with $N = 2048^3$ and $M = 256^3$ and the run with $N = 8192^3$ and $M = 1024^3$ retain an almost identical efficiency as we scale up with $s$. If $s \gtrsim 10$, the run with $N = 8192^3$ is slightly less efficient than the simulation with $N = 2048^3$ and $M = 256^3$. The simulation using $N = 2048^3$ and $M = 1024^3$ is less efficient than the other two runs. The data volume of the mesh cell exchange is 64 times higher than that of the run with $M = 256^3$, which results in an increased communication overhead.

Cosmological production simulations use a vast number of particles, and require compute resources that provide sufficient memory for particle storage and integration. The memory required for tree integration is equal to 60 bytes per particle and 52 bytes per tree node. As we use an $N_{\text{leaf}}$ value of 10 (where $N_{\text{leaf}}$ is the number of particles where the interaction tree will not be divided further), we have on average 0.75 tree nodes per particle, and therefore require a total of 99 bytes of memory per integrated particle (see [65] for further details).

Using 99 bytes per particle, a simulation with $N = 2048^3$ requires 850 GB of RAM for tree integration while a run with $N = 8192^3$ requires at least 54 TB of RAM. In addition, 4.5 bytes per mesh cell is required to do PM integration. These memory constraints place a lower limit on the number of processes that can be used, and indirectly determine the memory required on communication nodes. For particularly

<table>
<thead>
<tr>
<th>Name of constant</th>
<th>Value</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_{\text{tree}}$</td>
<td>$5.0 \times 10^{-9}$</td>
<td>[s]</td>
</tr>
<tr>
<td>$\tau_{\text{pm}}$</td>
<td>$1.6 \times 10^{-7}$</td>
<td>[s]</td>
</tr>
<tr>
<td>$\lambda_{\text{lan}}$</td>
<td>$8.0 \times 10^{-5}$</td>
<td>[s]</td>
</tr>
<tr>
<td>$\lambda_{\text{wan}}$</td>
<td>$3.0 \times 10^{-1}$</td>
<td>[s]</td>
</tr>
<tr>
<td>$\sigma_{\text{lan}}$</td>
<td>$2.3 \times 10^9$</td>
<td>[bytes/s]</td>
</tr>
<tr>
<td>$\sigma_{\text{wan}}$</td>
<td>$4.0 \times 10^8$</td>
<td>[bytes/s]</td>
</tr>
</tbody>
</table>

Table 5.10: List of network parameters used for the scalability predictions of our code. The name of the constant can be found in the first column, the value used for our global grid model in the second column and the units used for each value in the third column.
5.5 Scalability of $N$-body simulations across supercomputers

Figure 5.7: Predicted speedup $S(s)$ of simulations using the TreePM method as a function of the number of sites $s$ in a global grid. The total number of processes scales linearly with $s$.

large exchange volumes, the code can conserve memory on the communication nodes by communicating in multiple steps.

5.5.2 Speedup and efficiency predictions for tree and direct-method simulations

We present predictions of three example $N$-body simulations of stellar systems, each of which uses a different integration method. The integration methods we use in our models are a Barnes-Hut tree algorithm with shared time steps, a tree algorithm using block time steps [90] and a direct-method algorithm [85] using block time steps. For the tree algorithm we choose the problem sizes and process counts previously used for the cosmological simulation models with $N = 2048^3$. Modelling the performance of direct-method simulations using $N = 2048^3$ is unrealistic, because such a run would take many years. We instead predict the performance of direct-method simulations using a more realistic problem size of 5 million particles.

We model the tree algorithm using a slightly extended version of the models presented in [84, 82] and the direct-method algorithm using the grid-enabled model presented in Chapter 2. An overview of the three problems used for our predictions is given in Tab. 5.11.

The simulations using the tree algorithm are mapped to the same global grid infrastructure that we used for modelling the cosmological simulation (see Tab. 5.10
Figure 5.8: Predicted efficiency $E(s)$ of simulations using the TreePM method as a function of the number of sites $s$ in a global grid. The total number of processes is kept fixed for all simulations.

Table 5.11: Description of the stellar $N$-body problems used for the scalability analysis across supercomputers. The type of integrator is given in the first column and the number of particles in the second column. The number of processes per site for respectively the speedup analysis and the efficiency analysis are given in the third and fourth column. The time step scheme used is given in the last column. Note that the direct-method simulation is modelled for a grid of GPUs.
5.5 Scalability of N-body simulations across supercomputers

for the machine constants used). The direct-method simulation is modelled using the network constants from the global grid model, but here we assume that each node has a Graphics Processing Unit (GPU) with a force calculation performance of 200 GFLOP/s. As each force interaction requires 66 FLOPs on a GPU, we spend $3.3 \times 10^{-10}$ seconds per force interaction (see [40] for details on direct N-body integration on the GPU).

5.5.2.1 Performance model for the tree algorithm

We extend the tree code models given in [84, 82] to include the grid communication overhead by adding the equations for wide area communication of the tree method from our SUSHI performance model. In accordance with our previous model, we define the wide area latency-bound time for the tree method ($w_{l,\text{tree}}$) as,

$$w_{l,\text{tree}} = \lambda_{\text{wan}} (4(s-1) + 4).$$  \hspace{1cm} (5.15)

The bandwidth-bound communication time ($w_{b,\text{tree}}$) consists of the communication volume for the local essential tree exchange, which we estimate to be double the size used in TreePM simulations (see Section 5.3.1) due to the lack of PM integration, and the communication volume required for particle sampling. It is therefore given by,

$$w_{b,\text{tree}} = \frac{(96/\theta + 48) N^{2/3} + 4Nr_{\text{samp}}}{\sigma_{\text{wan}}},$$  \hspace{1cm} (5.16)

where $r_{\text{samp}} = 1/10000$ and $\theta = 0.5$. We use the equation given in [82] for Plummer sphere data sets to calculate the total number of force interactions.

5.5.2.2 Modelling of block time steps

Tree and direct N-body integrators frequently use a block time step scheme instead of a shared time step scheme. Block-time step schemes reduce the computational load by integrating only a subset of particles during each step [54, 81]. To equalize the number of integrated particles between simulations, we therefore compare a single shared time-step integration with $N/n_b$ steps of a block time-step integration, where $n_b$ is the average block size. We adopt an average block size of $n_b = 0.2 N^{0.81}$ for all block time-step integrators, the same value that we used in Chapter 2.

5.5.2.3 Predictions

We give the predicted speedup $S(s)$ of the three simulations as a function of the number of supercomputers $s$ in Fig. 5.9. Here, the tree code with shared time steps scales similarly to the cosmological simulation with $N = 2048^3$ and $M = 256^3$ (see Fig. 5.7). We predict a speedup of $\sim 13$ when $s = 16$. When we model the tree code with a block time step scheme, the scalability becomes worse because it requires $N/n_b$ times as many communication steps to integrate $N$ particles. The large number of communications combined with the high latency of wide area networks result in a high communication
Figure 5.9: Predicted speedup $S(s)$ of $N$-body simulations using the tree and direct method as a function of the number of sites $s$ in a global grid. The total number of processes scales linearly with $s$. 
5.5 Scalability of \( N \)-body simulations across supercomputers

overhead. The direct \( N \)-body run on a global grid of GPUs also does not scale well over \( s \), as predicted earlier in Chapter 2, due to the use of block time steps.

We give the predicted efficiency of the three simulations over \( s \) supercomputers relative to a simulation over one supercomputer, \( E(s) \) in Fig. 5.10. The efficiency of these runs is mainly limited by the latency-bound communication time. The tree code with shared time steps retains a high efficiency for \( s \lesssim 16 \), while the simulations with block time steps are less efficient due to the larger number of communications. We predict a slightly higher efficiency when \( s \gtrsim 4 \) for the direct-method simulation than for the tree code simulation with block time steps. This is because the lower \( N \) results in a lower number of communication steps, and therefore in less communication overhead.

5.5.3 Bandwidth analysis for cosmological simulations

We have shown that our TreePM code scales well across up to \( \sim 16 \) sites if the ratio between the number of particles and the number of mesh cells is sufficiently large. Here we examine the efficiency of four cosmological simulations over 8 sites. The efficiency compared to a single-site run is predicted as a function of the available bandwidth. We have included three predictions for a global grid with 0.3 s network latency as well as one prediction for a simulation with \( N = 2048^3 \) and \( M = 256^3 \) over a medium-range grid with 30 ms network latency. The details of these simulations are given in Tab. 5.9,
and the results are shown in Fig. 5.11.

We find that the efficiency of cosmological simulations across sites is heavily dependent on the available bandwidth. The run with \( N = 2048^3 \) and \( M = 256^3 \) on a global grid has \( E(8) = 0.9 \) (where \( E(8) \) is the efficiency as defined in Eq. 5.14) when the supercomputers are connected with a 100 MB/s network. Using a network with a higher bandwidth has little effect on the achieved efficiency, as the communication overhead is then dominated by network latency. The effect of network latency is clearly visible when we look at the prediction for the same simulation on a grid with a shorter baseline. When using a grid with 30 ms network latency the simulation reaches \( E(8) = 0.97 \) if the wide area network achieves a throughput of 1000 MB/s (which is possible with a fine-tuned and dedicated 10 Gbps optical network). We predict an efficiency \( E(8) > 0.8 \) for both simulations if the available network bandwidth between sites is 50 MB/s.

The run with \( N = 2048^3 \) and \( M = 1024^3 \) is more communication intensive, and requires a network throughput of at least 700 MB/s to achieve an efficiency of 0.8. The simulation using \( N = 8192^3 \) and \( M = 1024^3 \) runs more efficiently than the simulation using \( N = 2048^3 \) and \( M = 1024^3 \) independent of the obtained bandwidth. Although the exchanged data volume is larger in the run with \( N = 8192^3 \), this increased overhead is offset by the higher force calculation time per step. The large communication volume reduces the efficiency considerably for low bandwidth values, and an average network throughput of at least 150 MB/s is required to achieve an efficiency of \( E(8) = 0.8 \).

### 5.6 Conclusion

We have run a few dozen cosmological \( N \)-body simulations and analyzed the scalability of our SUSHI integrator on a national distributed computer and across a global network of supercomputers. Our results confirm that SUSHI is able to efficiently perform simulations across supercomputers. We were able to run a simulation using 1024\(^3\) particles across three supercomputers with \( \sim 10\% \) communication overhead. The communication performance can be further improved by tuning the optical networks.

Based on our model predictions we conclude that a long-term cosmological simulation using 2048\(^3\) particles and 256\(^3\) mesh cells scales well over up to \( \sim 16 \) sites, given that sufficient bandwidth is available and the number of cores used per site is limited to \( \sim 256 \). We also predict that tree codes with a shared time step scheme run efficiently across multiple supercomputers, while tree codes with a block time step scheme do not.

Considerable effort is still required to obtain acceptable message passing performance through a long distance optical network. This is due to three reasons. First, it may take up to several months to arrange an intercontinental light path. Second, optical networks are generally used for high volume data streaming such as distributed visualization or bulk data transfer, and are therefore not yet tuned to achieve optimal message passing performance. Third, intercontinental networks traverse a large number of different institutes, making it politically difficult for users to diagnose and adjust settings on individual sections of the path. For our experiments we therefore chose to optimize the wide area communications by tuning our application, rather than requesting system-level
5.6 Conclusion

Figure 5.11: Predicted efficiency of four $N$-body simulations using the TreePM method over 8 sites ($E(8)$) as a function of the wide area network throughput ($\sigma_{wan}$) in MB/s. Three simulations are run on a global grid. One simulation uses $N = 2048^3$ particles and $M = 256^3$ mesh cells (given by the black dashed line), one uses $N = 2048^3$ and $M = 1024^3$ (black dotted line) and one run uses $N = 8192^3$ and $M = 1024^3$ (thick black line). A simulation using $N = 2048^3$ and $M = 256^3$ modelled for a grid with 30 ms network latency between sites is given by the thin gray line. Runs with $N = 2048^3$ particles are predicted using a total of 2048 processes, runs with $N = 8192^3$ particles are predicted using a total of 32768 processes.

modifications to the light path configuration.

The main challenges in running simulations across supercomputers are now political, rather than technical. During the GBBP project, we were able to overcome many of the political challenges in part due to good will of all organizations involved and in part through sheer patience and perseverance. However, orchestrating a reservation spanning across multiple supercomputers is a major political undertaking. The use of a meta-scheduler and reservation system for supercomputers and optical networks greatly reduces this overhead, and also improves the workload distribution between supercomputer centers. Once the political barriers are overcome, we will be able to run long lasting and large scale production simulations over a grid of supercomputers.