UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

High performance N-body simulation on computational grids

Groen, D.J.

Publication date
2010

Link to publication

Citation for published version (APA):
Groen, D. J. (2010). *High performance N-body simulation on computational grids*. [Thesis, fully internal, Universiteit van Amsterdam].

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:19 Jan 2026

1 Introduction

In this thesis we study the simulation of gravitational N-body systems on infrastructures that span across the globe. We give some background information on the subject matter of this thesis in the first part of this section, and an overview of the contents of this thesis in the second part.

Gravity is the dominant force that shapes the universe. It causes objects with mass to attract each other and has a much longer range than the other three fundamental forces (strong interaction, electromagnetic force and weak force). The movements of planets, stars and galaxies are therefore driven by gravity. By understanding the force of gravity, we learn about the dynamical evolution of the solar system, star clusters, galaxies and the universe as a whole. This challenge in understanding the movements of celestial bodies is commonly referred to as the *N-body problem*.

On July 5th 1687, Isaac Newton proposed laws of motion for classical mechanics, as well as a law of gravity in his "Philosophiae Naturalis Principia Mathematica". This law defines the gravitational force F_i on a particular body i as a function of the positions and masses of all other bodies, and is given by

$$F_i = m_i a_i = m_i G \sum_{j \neq i}^{N} \left(m_j \frac{r_j - r_i}{|r_j - r_i|^3} \right). \tag{1.1}$$

Here G is the gravitational constant, and m_j and r_j are respectively the mass and position of a given body j. Although the motion of planets in the Solar system could be reproduced using Kepler's laws of planetary motion [71], Newton's law of gravity introduced a generalized concept of mutual attraction between objects with mass. Using this law it is possible to accurately calculate the movements of celestial bodies in systems with a moderate gravitational field. In the presence of strong gravitational interactions, however, the theory no longer provides correct results.

Between 1907 and 1915, Albert Einstein developed the theory of general relativity, which describes gravity as a geometric property of both space and time. This theory can be applied to accurately calculate the motions of N-body systems with a strong gravitational field and it provides explanations on various astrophysical phenomena, such as gravitational lensing [137, 135] and the existence of black holes [116]. Einstein's theory of relativity has also been applied to support the Big Bang [75], a widely accepted cosmological model that describes the origin and the expansion of the universe.

The breakthroughs achieved by Newton and Einstein provided tools to tackle the N-body problem, as scientists were now able to write down the equations of motion for interacting celestial bodies. By integrating these equations over time it became possible to model the evolution of an N-body system. This integration cannot be performed analytically for $N > 2$. However, we can use a numerical integration method to simulate the dynamics of larger N-body systems. These numerical integrations provide approximate answers and can be applied to systems of any size, if sufficient resources to perform the required computations are available.

1.1 Simulating N-body systems

The first known attempt to reproduce the movement of stars through numerical integration was a brief simulation of a three body problem performed on paper by Strömgren [122].

About 30 years later, the first known N-body simulation consisting of more than a few bodies using a numerical integration scheme was performed by Erik Holmberg in 1941 [63]. He used two sets of 37 light bulbs to simulate the gravitational dynamics of two nebulae (see Fig. 1.1). Such analogue simulations required a vast amount of manual computations and man hours, and were therefore difficult to reproduce.

With the advent of digital computers in the 1950s and 60s, it became considerably easier to numerically integrate N-body systems. Since computers have the capability to perform arithmetic operations and handle data, it became possible to map celestial bodies to digital point particles and perform N-body simulations electronically. The first computer simulations were performed in the 1960s by von Hoerner (1960 [130]), Aarseth (1963 [1]) and van Albada (1968 [126]) and were still proportional in size to Holmbergs acoustic simulation. These simulations used *direct-method integration*, a method where the equations of motion are solved between all particle pairs. Several production-quality direct-method N-body codes have been developed in the subsequent decades. Examples include the `NBODY1` to `NBODY7` codes developed by Aarseth [3], the `kira` integrator in the `Starlab` environment [110] and `phiGRAPE` [50].

1.1.1 Direct-method integration methods

Direct method N-body simulations are highly accurate due to the explicit calculation of the force exchanges between each pair of particles. The high accuracy of direct-method simulations makes them well suited for modelling tightly interacting stellar systems such as dense star clusters (e.g., see [128, 16, 107, 111, 129]) and stars near the Galactic

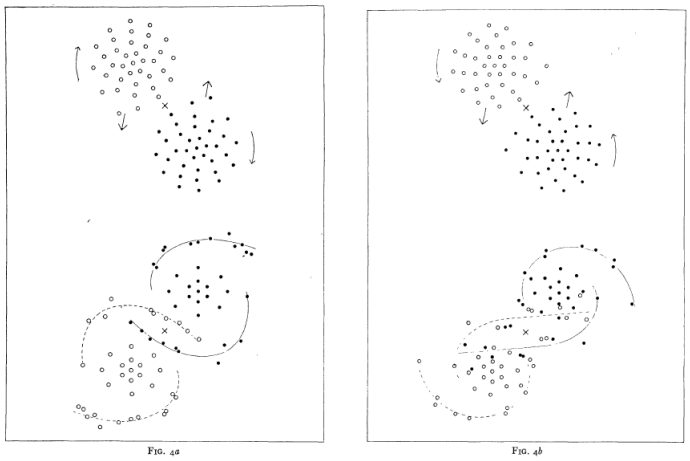

Figure 1.1: Snapshots of two light bulb based N-body simulations of galaxy encounters, one with the galaxies rotating clockwise (left panel), and one with the galaxies rotating counter-clockwise (right panel) [63].

center (e.g., see [35, 80, 102]). More details on N-body problems simulated using direct method can be found in [53, 4].

When simulating an N-body system using a direct method, the motions of particles are commonly approximated using a fourth order Hermite predictor-corrector scheme [85] with either an individual or shared time step scheme. A shared time step scheme imposes the same update interval on all particles while an individual time step scheme assigns an update interval that is specific for each particle. When a particle is selected to update its position and velocity in an individual time step simulation, the positions and velocities of other particles are extrapolated to that point in time. Then, the force exchanges are evaluated between the selected particle and the rest of the system, after which the positions, velocities and time step of the selected particles are updated. Future direct N-body codes may feature sixth or eighth order integration schemes which operate more efficiently and obtain comparable accuracy using fewer time steps [96].

Using a shared or individual time step scheme for N-body integration has disadvantages, especially when the code is run on a parallel or multi-core architecture. Shared time step schemes scale well on parallel architectures, but waste CPU time when tight few body interactions lower the time step length of the whole N-body system. Schemes using individual time steps are more calculation efficient, but scale poorly on parallel architectures due to the time dependencies of gravitational dynamics. The block time-step scheme [90] rounds the individual time step sizes down to a range of discrete values (e.g., powers of two), allowing the integration of multiple particles simultaneously while avoiding unnecessary force calculations. Block time step codes can therefore efficiently be used in parallel [50, 105].

The explicit calculation of forces between all particle pairs in direct-method simula-

tions results in a computational complexity that scales with the particle count squared per integration step ($O(N^2)$). In addition, the number of integration steps required for a full-length simulation also scales with $O(N)$ leading to a total computational complexity of $O(N^3)$. Due to this steep scaling, direct-method integration can only be applied nowadays to problems of up to a few million particles, whereas for example the Milky Way contains about 100 billion stars.

1.1.2 Alternative N-body integration methods

To reduce the computational complexity of N-body simulations, and enable the integration of larger N-body systems, several alternative techniques have been developed. One of the most widely used alternatives is the Barnes-Hut tree method [14], which orders particles into cells of a hierarchical three-dimensional tree. The force interactions over long distances are then calculated using averaged values for mass and position from tree cells instead of using individual particles. The force interactions over short distances are still calculated using individual particles. The accuracy of the tree integration method can be adjusted by modifying the opening angle (θ). A larger opening angle results in fewer force evaluations per particle, and a lower accuracy overall. The application of the Barnes-Hut method results in a reduced computational complexity of $O(N \log N)$ per integration step. A graphical example of the computed interactions in a tree method force calculation can be found in Fig. 1.2.

Another method to speed up N-body integration is the particle-mesh method [92, 62, 91], where all particles are mapped to a mesh by adding weights to the closest mesh vertices. The method then calculates the gravitational potential by converting the mesh to the frequency domain using a Fast Fourier Transform, solving the Poisson equations and then converting it back to the spatial domain. The particle-mesh method is highly accurate in long-distance force calculations, scales linearly with the number of particles and with the number of mesh cells (N_c) as $O(N_c \log N_c)$. The method is less suitable for resolving force interactions over short distances, and is therefore frequently used in combination with an adaptive grid [20] or other integration methods (e.g., PPPM [58]).

One hybrid method that became popular recently in cosmological N-body simulations is the tree/particle-mesh (TreePM) method [133]. This method calculates the force interactions over short distances using the tree method and over long distances using the particle-mesh method. Simulations using the TreePM method are commonly performed to investigate the formation of galaxies in the universe (e.g., see [120, 100, 66]).

1.2 High-performance N-body simulations

The simulation of large N-body systems is computationally expensive, and even approximate integration methods often require more compute power than a single system can provide. We can speed up the N-body calculations by parallelizing the simulation or optimizing it for specialized hardware.

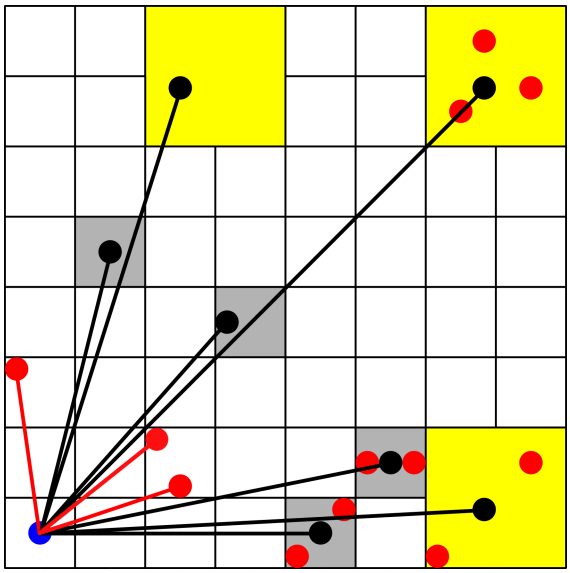

Figure 1.2: 2D example of several force calculations using a tree code. The particle for which tree integration is performed is given by the blue dot in the bottom left corner. The tree cells that are used to integrate this particle are given by the colored boxes and the individual particles are indicated by red dots. The centers of mass of tree cells containing one or more particles are indicated by black dots. Particle-particle force interactions are shown by the red lines, particle-cell interactions are shown by the black lines.

1.2.1 Parallelization and speedup

A parallelized simulation uses multiple computing nodes by distributing the computationally intensive tasks such as force calculations among the processes. It therefore takes less time to complete than a single process simulation. However, some knowledge of parallel programming is required, which can be found in e.g., [32].

If both the single process and the parallel simulation are run using the same problem size, the obtained improvement in performance is referred to as the fixed-size speedup ($S_{\text{fixed}-\text{size}}$). The maximum obtainable fixed-size speedup is constrained by the ratio between the serial execution time V and the parallel execution time T of an otherwise identical N-body simulation [8]:

$$S_{\text{fixed}-\text{size}} \leq \frac{V}{T_s + \frac{T - T_s}{p}}, \tag{1.2}$$

where T_s is the time spent on work that has not been parallelized and p the number of concurrent processes. In this equation, we find that the fixed-size speedup never exceeds the number of processes and will always be less than V/T_s.

If we take the communication overhead required for parallel processing into account

as suggested by [124], we are able to further constrain the maximum speedup to

$$S_{\text{fixed}-\text{size}} \leq \frac{V}{T_s + \frac{T - T_s}{p} + LC_c}. \tag{1.3}$$

Here L is the average point-to-point network latency and C_c the number of blocking communications in the simulation. The number of blocking communications in N-body simulations scales with either $\log p$ or p, because the summation of forces requires communications over all processes. Such global exchanges require $\log p$ blocking communications, or p communications if a ring topology is used. The overhead introduced by inter-process communication results in a decrease in speedup if p is sufficiently large.

The efficiency of a simulation across sites (E) is calculated by dividing V by the product of T and p. It is therefore equal to $S_{\text{fixed}-\text{size}}$ divided by p:

$$E = \frac{V}{Tp} = \frac{S_{\text{fixed}-\text{size}}}{p}. \tag{1.4}$$

A second limitation to the obtained fixed problem size speedup is the degree of parallelism of the application, which indicates the maximum number of concurrent operations that can be performed by the simulation. The maximum degree of parallelism for N-body simulations is typically very large, because a large number of independent force calculations are performed during each integration step.

Modern N-body simulations require an amount of wall-clock time that is comparable to historical runs but, thanks to the advances in compute technology, perform much more calculations during this period. As a result, the scaling we observe historically is one where the compute time of simulations was kept (roughly) constant, rather than the problem size. Gustafson observed this and proposed the concept of fixed-time scaling and fixed-time speedup [47]. Fixed-time (or scaled) speedup is measured by scaling up the problem size as more processes are added to the simulation, maintaining a constant execution time. We then measure the fixed-time speedup using

$$S_{\text{fixed}-\text{time}} = \frac{T_{s1}}{T_{sp}}. \tag{1.5}$$

Here, T_{s1} is the hypothetical execution time of the scaled workload using one process and T_{sp} the execution time of the scaled workload on p processes. Gustafson found that scaled speedup has fewer constraints than fixed-size speedup, and that parallel programs are more efficient when the problem is properly scaled. An alternative method of scaling and measuring scaled speedup is the concept of memory-bound speedup [124]. Here, the problem is scaled such that the memory on all nodes is fully used. This metric is useful for integrators with low computational complexity (such as the TreePM method), as these are more likely to be limited in scaling by the memory size.

1.2.2 Communication in parallel simulations

The performance of a parallel simulation depends on the properties of both the network used and the communication characteristics of the application.

The latency of a network is defined as the time required to send a signal from one endpoint to the other. The latency is heavily dependent on distance because a signal cannot travel faster than the speed of light. Reducing the latency is therefore difficult without modifying the topology of the network. The bandwidth of a network is defined as the number of bytes per second that can be transferred through the network. The bandwidth is limited for a given network, but can be increased for example by upgrading the hardware or adding more network cables. In addition to a low latency and a high bandwidth, a high quality of service and a minimal background load on the network are important to efficiently run a parallel simulation.

The performance of parallel simulations also depends on the communication characteristics of the simulation itself. The time spent on network latency is calculated by the multiplying the number of blocking communication steps in the simulation with the network latency, while the time spent on transferring data is calculated by dividing the communication volume of each process by the available network bandwidth.

1.2.3 N-body simulations using special-purpose hardware

Executing an N-body simulation on special-purpose hardware can lead to large improvements in performance. While general-purpose CPUs have a broad instruction set applicable for a limited number of pipelines, specialized hardware components typically map a more narrow instruction set to a large number of pipelines. This allows a single PC with special-purpose hardware components to achieve a performance comparable to that of dozens of mundane PCs. Several types of special-purpose hardware are currently applied for N-body simulations.

The GRAvity PipE (GRAPE) has been specifically designed to perform fast force calculations, achieving a performance comparable to a hundred PCs at time of launch. The first version was presented in 1990 [123], and broke all performance records once in operation. The latest GRAPE card, the GRAPE-DR [87], was launched in 2009 and delivers 1 TFLOP/s per chip (a card contains up to 8 chips). For nearly two decades, the GRAPE has been the most powerful device for large-scale direct-method N-body simulations.

A notable alternative for the GRAPE emerged in 2007, when the release of the Compute Unified Device Architecture (CUDA) enabled programmers to efficiently use Graphics Processing Units (GPUs) for general-purpose computing. GPUs are designed to produce realistic 3D visualizations, and modern cards are optimized to process as many as 240 floating point operations in parallel [99]. The force calculations required for N-body integration map particularly well to this architecture as they can be performed independently within a given time step and require many repetitions of a limited number of floating point operations. Several groups obtained a simulation performance of up to 200 GFLOP/s per card [18, 49, 40]. Because GPUs offer performance comparable to

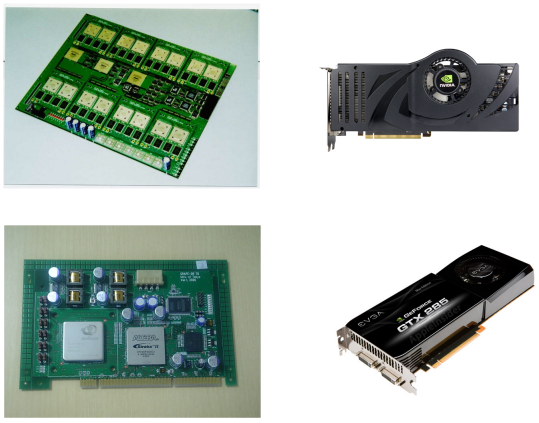

Figure 1.3: Graphical overview of commonly used hardware for N-body integration. The top row shows the generation of hardware used for some of our experiments, which are the GRAPE-6 (top left) and the GeForce 8800 Ultra GPU (top right). The bottom row shows two more recent counterparts, which are the GRAPE-DR (bottom left) and the GeForce GTX285 GPU (bottom right).

specialized hardware at a fraction of the cost, they are now used for a wide range of scientific applications.

The concept of using a high-performance instruction set with limited functionality is not exclusive to special-purpose hardware. Since the release of the 8087 co-processor for floating point operations in 1980 [101], CPU manufacturers have sought to improve the CPU performance for commonly applied computations by introducing specialized processing units. For example, modern commodity processors sold by AMD and Intel provide the SSE1-4 instruction sets to improve floating-point calculation performance. The performance of an application can be improved if it is tuned to use these instructions, but such enhancements are not straightforward to implement. Many compilers are capable of mapping applications to use special-purpose instructions to some extent, but the best performance is obtained by libraries directly written in x86 assembly language. The Phantom GRAPE is a library written in assembler to specifically speed up N-body integration [97], and provides a speedup of ~ 8 compared to regular compiler generated code.

1.3 Distributed infrastructures and computational grids

During the last two decades, the quality and performance of wide area networking has improved at a rate that exceeds the increase in computational power. This has made it increasingly convenient to create a distributed infrastructure by linking machines together over a wide area network (WAN). Distributed infrastructures are often heterogeneous and encompass multiple administrative domains. However, if proper measures are taken to combine the different sites, these infrastructures become

a suitable platform for parallelized simulations.

The most convenient way to combine computational sites into a unified infrastructure is by installing grid middleware such as the Globus Toolkit [31, 30]. Grid middleware adds an abstraction layer on top of the distributed infrastructure, and provides a uniform interface to the full system for all its users. This interface includes a single sign-on method for all sites using proxy certificates, which allows users and programs alike to freely switch between sites once they are granted access to the grid. In addition, grid middleware supports schedulers that distribute computational 'jobs' across the participating sites [5, 55] and mechanisms that allow for automated data replication [22].

The technical barriers for installing grid middleware on computers are trivial to bypass but the political limitations are more difficult to overcome. Especially the larger and more production-oriented supercomputer centers are reluctant to compromise on their scheduling, access and security policies in order to collaborate in a computational grid. Although grid projects consisting of supercomputers do exist (e.g., DEISA [25]), these infrastructures tend to have a more restrictive set of middleware with limited capabilities for resource reservations and no scheduling mechanism for runs across sites. In addition, such grids of supercomputers require dedicated WAN links, as the data volumes exchanged between these machines are generally too large to be efficiently handled by regular internet.

Computational grids are mainly used for high-throughput computing, a discipline where grid infrastructures are used to execute vast numbers of relatively small jobs. However, several projects have developed tools to enable the parallel execution of programs across multiple grid sites. These tools include MPI [118] implementations that specifically support grid security (such as MPICH-G2 [70] and MPIG [89]) or support execution across sites more generally (such as PACX-MPI [38] and OpenMPI [37]). Running a simulation across sites is a potentially attractive way to improve the performance of large-scale N-body simulations, and to enable the simulation of N-body problems of unprecedented scales.

1.4 Thesis overview

The N-body simulations performed up to now provided answers on many research questions, and a variety of integration techniques have been developed to efficiently simulate the dynamical evolution of stellar and cosmological systems. However, up to this point all production N-body simulations have been run on local computing sites only. Using multiple sites in parallel allows the simulation of larger N-body systems, and the continual increase in the number of computing sites will provide a structural boost in growth of N-body simulation performance. However, the computational capabilities provided by a planet that is already covered with computer hardware and networks are not trivially applied to solve large-scale N-body problems. Is it possible to map high performance N-body simulations to infrastructures that span up to several thousands of kilometers? If so, how can we efficiently use these widely distributed resources? The

primary aim of this thesis is to examine the feasibility of simulating N-body systems on wide area computational (grid) infrastructures. We examine a variety of N-body integrators used for increasingly large N-body systems. These integrators range from the high-accuracy direct summation methods used to model dense star clusters to the fast Tree/Particle Mesh methods used to reproduce structure formation in the universe.

1.4.1 Chapter 2

In Chapter 2 we present performance measurements of direct gravitational N-body simulations on the grid, with and without specialized (GRAPE-6) hardware. Our inter-continental virtual organization consists of three sites, one in Tokyo, one in Philadelphia and one in Amsterdam. In many cases, high performance simulations over the entire planet are dominated by network bandwidth rather than latency. With this global grid of GRAPEs our calculation time remains dominated by communication over the entire range of N, which was limited due to the use of three sites. Increasing the number of particles will result in a more efficient execution. Based on these timings we construct and calibrate a model to predict the performance of our simulation on any grid infrastructure with or without GRAPE. We apply this model to predict the simulation performance on the Dutch DAS-3 wide area computer. Equipping the DAS-3 with GRAPE-6Af hardware would achieve break-even between calculation and communication at a few million particles.

1.4.2 Chapter 3

In Chapter 3 we describe the *living application*, a method to autonomously manage applications on the grid. During its execution on the grid, the living application chooses the resources to use in order to complete its tasks. These choices can be based on the internal state, or on autonomously acquired knowledge from external sensors. By giving limited user capabilities to a living application, the living application is able to port itself from one resource topology to another. The application performs these actions at run-time without depending on users or external workflow tools. We demonstrate this new concept in a special case of a living application: the *living simulation*. Today, many simulations require a wide range of numerical solvers and run most efficiently if the solvers are matched to specialized nodes. The idea of the living simulation is that it decides itself which grid machines to use based on the numerical solver currently in use. In this chapter we apply the living simulation to model the collision between two galaxies in a test setup with two specialized computers. This simulation switches at run-time between a GPU-enabled computer in the Netherlands and a GRAPE-enabled machine that resides in the United States. It uses a Barnes-Hut tree N-body code [39] when it runs on the GPU and a direct N-body code [50] when it runs on the GRAPE.

1.4.3 Chapter 4

Chapter 4 reports on CosmoGrid, a large-scale cosmological N-body simulation. Cosmological N-body simulations are an essential tool for evolving density perturbations in

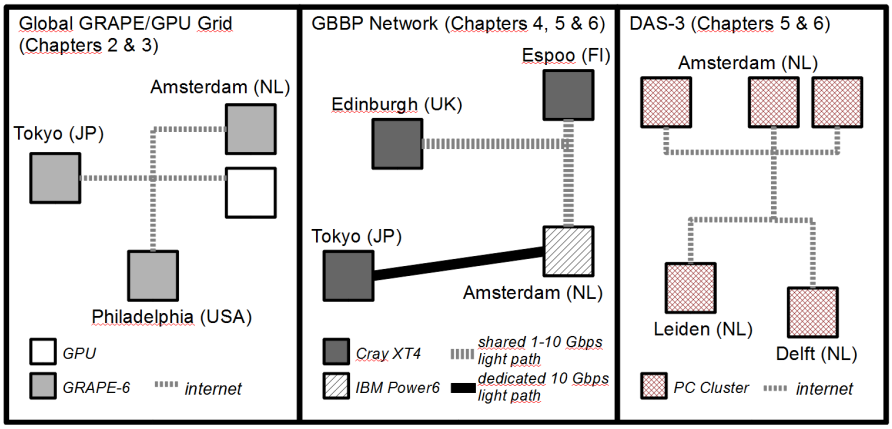

Figure 1.4: Overview of the infrastructures used in this thesis. The left panel shows the systems used in Chapters 2 and 3, the middle panels shows the systems used in Chapters 4 to 6 and the right panel shows the system used additionally in Chapter 5 and 6.

the nonlinear regime. Simulating the formation of large-scale structures in the universe, however, is still a challenge due to the enormous dynamic range in spatial and temporal coordinates, and the enormous computer resources required. The dynamic range is generally handled by the hybridization of numerical techniques. We deal with the computational requirements by connecting two supercomputers through an optical network and make them operate as a single machine. This is challenging, if only for the fact that the supercomputers of our choice are separated by half the planet, as one is located in Amsterdam and the other is in Tokyo. The co-scheduling of the two computers and the gridification of the code enables us to achieve a 90% efficiency for this distributed intercontinental supercomputer. We conclude that running cosmological N-body simulations on a limited number of (≤ 100) processors concurrently on more than 10 supercomputers would provide satisfactory performance and is politically favorable regarding the acquisition of the resources.

1.4.4 Chapter 5

In Chapter 5 we provide a detailed analysis of N-body integrators on supercomputer infrastructures. Here we present an extended version of the GreeM Tree/Particle Mesh integrator which runs on two or more supercomputers. Our code requires no underlying middleware, and can be tuned to efficiently use long distance optical networks. We have performed simulations using up to five sites, and found that the communication overhead of our code increases only marginally as we scale up in number of sites. We also present a time-complexity model for cosmological tree-particle mesh N-body simulations and compare the model predictions with our latest results. We conclude this chapter with a feasibility study of gridification of different N-body integrators. We examine the time complexity models of several integration methods (direct, tree and

TreePM) and apply the models to predict the performance of different N-body solvers on a globally distributed supercomputer.

1.4.5 Chapter 6

In Chapter 6 we present MPWide, a platform independent communication library for performing message passing between computers. Our library allows the coupling of several local MPI applications through a long distance network and is specifically optimized for such communications. The implementation is deliberately kept light-weight, platform independent and the library can be installed and used without administrative privileges. The only requirements are a C++ compiler and at least one open port to a wide area network on each site. In this chapter we present the library, describe the user interface, present performance tests and apply MPWide in a large scale cosmological N-body simulation on a network of two supercomputers, one in Amsterdam and the other one in Tokyo.