High performance cosmological simulations on a grid of supercomputers

Derek Groen
Centre for Computational Science
University College London
London, United Kingdom
Email: d.groen@ucl.ac.uk

Steven Rieder, Simon Portegies Zwart
Leiden Observatory
Leiden University
Leiden, the Netherlands
Email: rieder@strw.leidenuniv.nl,
spz@strw.leidenuniv.nl

Abstract—We present results from our cosmological N-body simulation which consisted of 2048x2048x2048 particles and ran distributed across three supercomputers throughout Europe. The run, which was performed as the concluding phase of the Gravitational Billion Body Problem DEISA project, integrated a 30 Mpc box of dark matter using an optimized Tree/Particle Mesh N-body integrator. We ran the simulation up to the present day (z=0), and obtained an efficiency of about 0.93 over 2048 cores compared to a single supercomputer run. In addition, we share our experiences on using multiple supercomputers for high performance computing and provide several recommendations for future projects.

Keywords-high-performance computing; distributed computing; parallelization of applications; cosmology; N-body simulation

I. INTRODUCTION

Cosmological simulations are an efficient method to gain understanding of the formation of large-scale structures in the Universe. Large simulations were previously applied to model the evolution of dark matter in the Universe [1], and to investigate the properties of Milky-Way sized dark matter halos [2], [3]. However, these simulations are computationally demanding, and are best run on large production infrastructures. We have previously run a cosmological simulation using two supercomputers across the globe [4] with the GreeM integrator [5], [6], and presented the SUSHI N-body integrator [7], which we used to run simulations across up to four supercomputers. The simulations we ran in the Gravitational Billion Body Project produced over 110 TB of data, which we have used to characterize the properties of ultra-faint dwarf galaxies [8], and to compare the halo mass function in our runs to analytical formulae for the mass function. Among other things, we found that the halo mass function in our runs shows good agreement with the Sheth and Tormen function [9] down to $\sim 10^7$ solar mass.

Here we present the performance results of a production simulation across three supercomputers, as well as several other runs which all use an enhanced version of SUSHI. The production simulation ran continuously for ~ 8 hours, using 2048 cores in total for calculations as well as 4 additional cores for communications. We achieved a peak performance of 3.31×10^{11} tree force interactions per second, a sustained performance of 2.19×10^{11} tree force interactions per second

and a wide area communication overhead of less than 10% overall.

We briefly reflect on the improvements made to SUSHI for this work in Section 2, while we report on tests performed on a single supercomputer in Section 3. In Section 4 we describe our experiments across three supercomputers and present our performance results. We reflect on our experiences on using multiple supercomputers for distributed supercomputing simulations, and provide several recommendations for users and resource providers in Section 4 and present our conclusions in Section 5.

A. Related work

There are a several other projects which have run high performance computing applications across multiple supercomputers. These include simulations of a galaxy collision [10], a materials science problem [11] as well as an analysis application for arthropod evolution [12]. A larger number of groups performed distributed computing across sites of PCs rather than supercomputers (e.g., [13], [14], [15]). Several software tools have been developed to facilitate high performance computing across sites of PCs (e.g., [16], [17], [18], [19], [20]) and within volatile computing environments [21]. The recently launched MAPPER EU-FP7 project [22] seeks to run multiscale applications across a distributed supercomputing environment, where individual subcodes periodically exchange information and (in some cases) run concurrently on different supercomputing architectures.

II. IMPROVEMENTS TO SUSHI

Based on results of our earlier simulations and in preparation for the production run across three supercomputers we made several modifications to the SUSHI distributed N-body integrator. In our previous experiments a relatively large amount of computation and communication time was spent on (non-parallelize) particle-mesh integration. To reduce this bottleneck we now parallelized the particle-mesh integration routines using the parallel FFTW2 library [23] and a one dimensional slab decomposition. We also optimized the communications of the particle-mesh integration by introducing a scheme where sites only broadcast those mesh cells which have actual particle content. This

Table I
INITIAL CONDITION AND ACCURACY PARAMETERS USED FOR OUR
SIMULATIONS WITH 2048^3 PARTICLES.

Parameter	Value
Matter density parameter (ω_0)	0.3
Cosmological constant (λ_0)	0.7
Hubble constant (H_0)	70.0 km/s/Mpc
Mass fluctuation parameter (σ_8)	0.8
Box size	$(30\text{Mpc})^3$
Softening for 2048^3 particle run.	175 pc
Sampling rate for 2048^3 particle run.	20000

Table II
OVERVIEW OF EXPERIMENTS PERFORMED WITH THE ENHANCED
SUSHI CODE ON THE HUYGENS SUPERCOMPUTER. THE TIME SPENT
ON COMMUNICATION IS GIVEN IN THE FOURTH COLUMN, WHILE THE
TOTAL RUNTIME IS GIVEN IN THE FIFTH COLUMN. ALL TIMES ARE
MEASURED PER STEP, AVERAGED OVER STEPS 1-11. IN ADDITION WE
INCLUDED THE TIMING RESULTS OF THE LAST 10 STEPS OF THE
SIMULATION RUNNING ON 2048 CORES (BOTTOM ROW).

N	p	θ	comm. t [s]	runtime [s]	z range $\times 10^{-3}$	speedup
2048^3	512	0.5	19.18	501.3	2.5-2.4	1
2048^3	1024	0.5	13.96	258.2	2.5-2.4	1.94
2048^3	2048	0.5	22.34	151.0	2.5-2.4	3.32
2048^3	2048	0.5	16.22	143.7	0.1-0.0	-

optimization reduced the size of the mesh communications by a factor roughly equal to the number of sites used, in the case of an equal domain distribution.

In some of the larger previous runs we also observed load imbalances if the code was run across two machines with different architectures, despite the presence of a load balancing scheme. This result has led us to further optimized the load balancing in SUSHI, taking into account not only the force integration time, but also the number of particles stored on each node. In addition to these changes, we also seized the opportunity to plug in a more recent MPWide [24] version into SUSHI. This newer version contains several optimizations to improve the wide area communication over networks with a high latency.

III. TESTS ON A SINGLE SITE

A. Setup

We performed a number of runs on the Huygens super-computer to validate the scalability of our new implementation, and to provide performance measurements against which we can compare our results using multiple sites. More information on the Huygens machine can be found in the second column of Tab. III. The initial conditions for this simulation is the snapshot at redshift $z = 0.0026$ from the CosmoGrid simulation (described in [4]). We also use the simulation parameters chosen for the CosmoGrid simulation, which are summarized in Tab. I. Here the first four parameters are constants which are derived from WMAP observations (with a slight-roundoff) and the physical size of our simulated system is given by the fifth parameter (Box size). The softening in our simulation (i.e. a length value added to reduce the intensity of close interactions) and the sampling rate are given by the last two parameters. The sampling rate is the ratio of particles in the simulation divided by the number of particles sampled by the load balancing scheme. Our simulation used a mesh size of 512^3 cells. We ran the simulation using respectively 512 cores and 1024 cores until $z = 0.0024$, and using 2048 cores until the simulation completed (at $z = 0$). The number of force calculations per step in the simulation varies for different z values, though these variations are neglishible for $z < 0.01$.

B. Results

The performance results of our runs are shown in Tab. II. In addition, the total runtime of the run using 2048 cores is given by the light blue line in Fig. 2. The overall performance of the code is dominated by calculations, with the communication overhead ranging from ~5% for 512 cores to ~10-15% for 2048 cores. During the run using 2048 cores, several snapshots were written. This resulted in a greatly increased execution time during two steps of the run.

IV. TESTS ACROSS THREE SITES

A. Setup

We performed our main run using a total of 2048 cores across three supercomputers, which are listed in Tab. III. These machines include Huygens in the Netherlands (1024 cores), Louhi in Finland (512 cores), and HECToR in Scotland (512 cores). The sites are connected to the DEISA shared network with either a 1Gbps interface (HECToR) or a 10Gbps interface (Huygens, Louhi). The initial conditions and simulation parameters chosen are identical to those of the runs using 1 supercomputer, although we use a mesh of 256^3 cells. The use of a smaller mesh size results in a slightly higher calculation time as tree interactions are calculated over a longer range, but a somewhat lower time spent on intra-site communications. We configured MPWide to use 64 parallel `tcp` streams per path for the wide area communication channels, each with a `tcp` buffer size set at 768 kB and packet-pacing set at 10 MB/s maximum. We enabled some load balancing during the run, though we had to limit the boundary moving length per step to 0.00001 of the box length due to memory constraints on our communication nodes and the presence of dense halos in our initial condition.

In addition to the main run, we also performed three smaller runs using the same code across the same three supercomputers. These include one run with 1024^3 particles using 80 cores per supercomputer, and two runs with 512^3 particles using 40 cores per supercomputer. These runs also used a mesh size of 256^3, though we did reduce the

Table III
PROPERTIES OF THE THREE SUPERCOMPUTERS USED FOR OUR RUN.
THE MEASURED PEAK NUMBER OF TREE FORCE INTERACTIONS (IN
MILLIONS) PER SECOND PER CORE IS GIVEN FOR EACH SITE IN THE
BOTTOM ROW.

Name	Huygens	Louhi	HECToR
Location	Amsterdam	Espoo	Edinburgh
Vendor	IBM	Cray	Cray
Architecture	Power6	XT4	XT4
# of cores	3328	4048	12288
CPU [GHz]	4.7	2.3	2.3
RAM / core [GB]	4/8	1/2	2
force calcs. / core [Mints/s]	185	256	250

Table IV
OVERVIEW OF EXPERIMENTS PERFORMED WITH THE ENHANCED
SUSHI CODE ACROSS ALL THREE SUPERCOMPUTERS. ALL TIMES ARE
MEASURED PER STEP, AVERAGED OVER 10 STEPS.

N	p	θ	comm. WAN	time total	runtime	z range
512^3	120	0.3	6.925	7.312	39.70	11.8-10.1
512^3	120	0.5	5.982	6.335	24.60	9.9-8.8
1024^3	240	0.3	12.09	14.04	214.5	17.0-14.9
2048^3	2048	0.5	15.40	24.77	167.7	0.0026-0.0025
2048^3	2048	0.5	14.62	23.13	155.2	0.0001-0

sampling rate to respectively 10000 and 5000 for the runs with 1024^3 and 512^3 particles. The force softening used for these runs were respectively 1.25kpc and 2.5kpc, and we set the boundary moving length limit to 0.01 of the box length. Some of the measurements were made using an opening angle θ of 0.3, rather than 0.5. Using a smaller opening angle results a higher accuracy of the force integration on close range, but also results in a higher force calculation and tree structure communication time per step.

B. Results

The timing results of our production run are shown in Fig. 1. Here, we also added the wall-clock time results of the simulation run using 2048 cores on Huygens as reference. The simulation run across three sites is only $\sim 9\%$ slower per step than the single-site run, despite the slightly higher force calculation time due to the lower number of mesh cells. The peaks in wall-clock time of the single site run are caused by the writing of snapshots during those steps (we only wrote one snapshot at the end of the three site run). The total wide area communication overhead of our run is $\lesssim 10\%$ at about $15s$ per step. Most of this time is required to exchange the tree structures between sites, though the communications for the parallelized particle-mesh require an additional $\sim 2.5s$ per step. Despite the use of a shared wide area network, the communication performance of our run shows very little jitter and no large slowdowns. We provide a snapshot of the final state of the simulation (at $z = 0$), distributed across the three supercomputers, in Fig. 2.

We also provide a numerical overview of the production run performance, as well as that of several other runs which use the new code, in Tab.IV. The communication overhead for the runs with 512^3 particles is less than 20%, while the overhead for the run with 1024^3 particles is just 6.5%. The parallelization of the particle-mesh integration and the enhanced load balancing greatly improved the performance of these runs, especially in the case with 1024^3 particles. Here, the communication overhead was reduced by $\sim 60\%$ and the overall runtime by more than 25% compared to the previous version [7].

V. USER EXPERIENCES

We have presented results from several cosmological simulations which run across three supercomputers, including a production run lasting for 8 hours. In the process of seeking a solution for wide area message passing between supercomputers, requesting allocations, arranging network paths and preparing for the execution of these simulations, we have learned a number of valuable lessons.

Primarily, we found that it is structurally possible to do high performace computing across multiple supercomputers. During the GBBP project we have run a considerable number of large-scale simulations using two or more supercomputers, with results improving as we were able to further enhance the N-body integrator and optimize the MPWide communication library for the wide area networks that we used.

The cooperation of the resource providers was particularly crucial in this project, as they enabled previously unavailable network paths and provided us with means to initiate simulations concurrently at the different sites. However, reserving networks and orchestrating concurrent supercomputer runs currently does require a disproportionate amount of time and effort, which makes performance optimization and debugging a challenging task. The effort required to run applications across supercomputers can be greatly reduced if resource providers were to adopt automated resource reservation systems for their supercomputers, and maintain shared high-bandwidth networking between sites. The persistent DEISA shared network connections helped greatly in our case, as we could use it at will without prior network reservations.

The software environment across different supercomputers, even within the same distributed infrastructure, is very heterogeneous. This made it unattractive to use existing middleware or message passing implementations to make different sites interoperable. We chose to use a modular approach where we connected platform-specific optimized versions of the SUSHI code with the MPWide communication library. With MPWide being a user-space tool that requires no external libraries or administrative privileges, we are able to install and run the simulation code in the locally preferred software environments on each site without

needing any additional (grid) middleware. We recommend adopting a similar modular software approach in future distributed supercomputing efforts for its ease of installation and optimization, at least until resource providers present a homogeneous and interoperable software environment for distributed supercomputing.

This paper focuses on the calculation and communication performance aspects of a single application run across supercomputers. However, the methods presented here can be applied for several other purposes. During this project we were confronted with additional overhead introduced by disk I/O, as can be observed in Figure 1. With supercomputer disk performance and capacity improving at a much slower rate than the compute power, the deployment of an application across sites may help to eliminate a disk I/O performance bottleneck, though a detailed investigation will be needed to quantify such potential benefit. Additionally, the communication technique could be used to facilitate periodic exchanges between different simulation codes, each of which runs on a different site and tackles a different aspect of a complex multiscale or multiphysics problem.

VI. CONCLUSION

Our results show that cosmological production simulations run efficiently across supercomputers for a prolonged time. The political effort required to arrange cross-supercomputer runs is considerable, and is an important reason why few people have attempted to run production simulations across supercomputers. We have shown that the added overhead of using a network of supercomputers is rather marginal for at least one optimized production application and that given the right (political) environment, supercomputers can be conveniently connected to form even larger high performance computing resources.
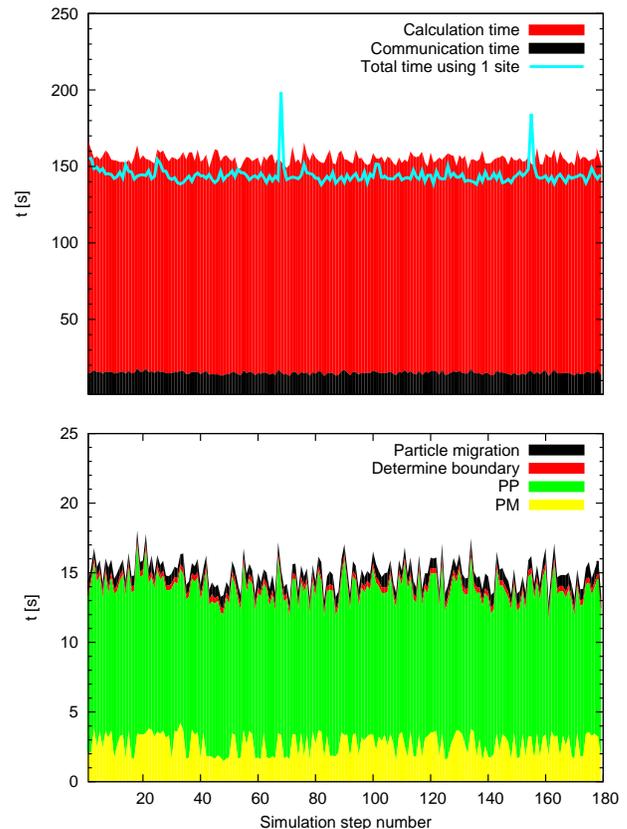
Figure 1. Performance results of the production simulation across three sites. In the top figure we provide the total time spent on calculation per step in red, and on communication per step in blue. Here, the total wall-clock time of an identical simulation using 2048 processes only on Huygens is given by the light blue line. 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). See [7] for full details on the communication routines of the code.

REFERENCES

[1] V. Springel, S. D. M. White, A. Jenkins, C. S. Frenk, N. Yoshida, L. Gao, J. Navarro, R. Thacker, D. Croton, J. Helly, J. A. Peacock, S. Cole, P. Thomas, H. Couchman, A. Evrard, J. Colberg, and F. Pearce, "Simulations of the formation, evolution and clustering of galaxies and quasars," *Nature*, vol. 435, pp. 629–636, Jun. 2005.

[2] V. Springel, J. Wang, M. Vogelsberger, A. Ludlow, A. Jenkins, A. Helmi, J. F. Navarro, C. S. Frenk, and S. D. M. White, "The Aquarius Project: the subhaloes of galactic haloes," *MNRAS*, vol. 391, pp. 1685–1711, Dec. 2008.

[3] T. Ishiyama, T. Fukushige, and J. Makino, "Variation of the Subhalo Abundance in Dark Matter Halos," *ApJ*, vol. 696, pp. 2115–2125, May 2009.

[4] S. Portegies Zwart, T. Ishiyama, D. Groen, K. Nitadori, J. Makino, C. de Laat, S. McMillan, K. Hiraki, S. Harfst, and P. Grosso, "Simulating the universe on an intercontinental grid," *Computer*, vol. 43, pp. 63–70, 2010.

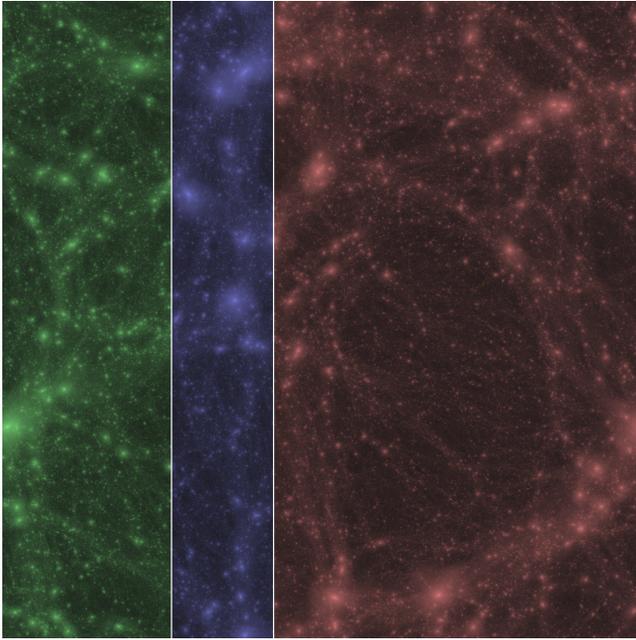[5] K. Yoshikawa and T. Fukushige, "PPPM and TreePM Methods on GRAPE Systems for Cosmological N-Body Sim-

Figure 2. The final snapshot of the production simulation across three supercomputers, taken at $z = 0$. The size of the box is 30x30x30 Mpc and the contents are colored to match the particles residing in Espoo (green, left), Edinburgh (blue, center) and Amsterdam (red, right) respectively.

ulations," *Publications of the Astronomical Society of Japan*, vol. 57, pp. 849–860, Dec. 2005.

[6] T. Ishiyama, T. Fukushige, and J. Makino, "GreeM: Massively Parallel TreePM Code for Large Cosmological N -body Simulations," *Publications of the Astronomical Society of Japan*, vol. 61, pp. 1319–1330, Dec. 2009.

[7] D. Groen, S. Portegies Zwart, T. Ishiyama, and J. Makino, "High Performance Gravitational N-body simulations on a Planet-wide Distributed Supercomputer," *Computational Science and Discovery*, vol. 4, no. 015001, Jan. 2011.

[8] T. Ishiyama, J. Makino, S. Portegies Zwart, D. Groen, K. Nitadori, S. Rieder, C. de Laat, S. McMillan, K. Hiraki, and S. Harfst, "The Cosmogrid Simulation: Statistical Properties of Small Dark Matter Halos," *ArXiv e-prints (submitted to PASJ)*, Jan. 2011.

[9] R. K. Sheth and G. Tormen, "Large-scale bias and the peak background split," *MNRAS*, vol. 308, pp. 119–126, Sep. 1999.

[10] M. Norman, P. Beckman, G. Bryan, J. Dubinski, D. Gannon, L. Hernquist, K. Keahey, J. Ostriker, J. Shalf, J. Welling, and S. Yang, "Galaxies collide on the i-way: an example of heterogeneous wide-area collaborative supercomputing," *International Journal of High Performance Computing Applications*, vol. 10, no. 2-3, pp. 132–144, 1996.

[11] T. J. Pratt, L. G. Martinez, M. O. Vahle, and T. V. Archuleta, "Sandia's network for supercomputer '96: Linking supercomputers in a wide area asynchronous transfer mode (atm) network," Sandia National Labs., Albuquerque, NM (United States), Tech. Rep., 1997.

[12] C. Stewart, R. Keller, R. Repasky, M. Hess, D. Hart, M. Muller, R. Sheppard, U. Wossner, M. Aumuller, H. Li, D. Berry, and J. Colbourne, "A global grid for analysis of arthropod evolution," in *GRID '04: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 328–337.

[13] A. Gualandris, S. Portegies Zwart, and A. Tirado-Ramos, "Performance analysis of direct n-body algorithms for astrophysical simulations on distributed systems." *Parallel Computing*, vol. 33, no. 3, pp. 159–173, 2007.

[14] H. Bal and K. Verstoep, "Large-scale parallel computing on grids," *Electronic Notes in Theoretical Computer Science*, vol. 220, no. 2, pp. 3 – 17, 2008, proceedings of the 7th International Workshop on Parallel and Distributed Methods in verifiCation (PDMC 2008).

[15] P. Bar, C. Coti, D. Groen, T. Herault, V. Kravtsov, M. Swain, and A. Schuster, "Running parallel applications with topology-aware grid middleware," in *Fifth IEEE international conference on e-Science and Grid computing: Oxford, United Kingdom*. Piscataway, NJ: IEEE Computer Society, December 2009, pp. 292–299.

[16] N. Karonis, B. Toonen, and I. Foster, "Mpich-g2: A grid-enabled implementation of the message passing interface," *Journal of Parallel and Distributed Computing*, vol. 63, no. 5, pp. 551 – 563, 2003, special Issue on Computational Grids.

[17] E. Gabriel, M. Resch, T. Beisel, and R. Keller, "Distributed computing in a heterogeneous computing environment," in *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, ser. Lecture Notes in Computer Science, vol. 1497. Springer, 1998, pp. 180–187.

[18] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall, "Open MPI: Goals, concept, and design of a next generation MPI implementation," in *Proceedings, 11th European PVM/MPI Users' Group Meeting*, Budapest, Hungary, September 2004, pp. 97–104.

[19] S. Manos, M. Mazzeo, O. Kenway, P. V. Coveney, N. T. Karonis, and B. R. Toonen, "Distributed mpi cross-site run performance using mpig," in *HPDC*, 2008, pp. 229–230.

[20] S. Sundari M., S. S. Vadhiyar, and R. S. Nanjundiah, "Morco: middleware framework for long-running multi-component applications on batch grids," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '10. New York, NY, USA: ACM, 2010, pp. 328–331. [Online]. Available: http://doi.acm.org/10.1145/1851476.1851522

[21] B. Rood, N. Gnanasambandam, M. J. Lewis, and N. Sharma, "Toward high performance computing in unconventional computing environments," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '10. New York, NY, USA: ACM, 2010, pp. 627–635. [Online]. Available: http://doi.acm.org/10.1145/1851476.1851569

[22] MAPPER, "Multiscale applications on european e-infrastructures: http://www.mapper-project.eu," Jul. 2011.

[23] M. Frigo and S. Johnson, "The design and implementation of fftw3," in *Proceedings of the IEEE*, vol. 93, Feb 2005, pp. 216–231.

[24] D. Groen, S. Rieder, P. Grosso, C. de Laat, and P. Portegies Zwart, "A light-weight communication library for distributed computing," *Computational Science and Discovery*, vol. 3, no. 015002, Aug. 2010.