3 Molecular Dynamics with GROMACS

Peicho Petkov, Stoyan Markov, Valentin Pavlov
National Centre of Supercomputing Application, NCSA, Bulgaria
peicho@phys.uni-sofia.bg

3.1 Introduction

GROMACS[46,47,48,49,50,51,52] is one of the fastest molecular dynamics simulators in the world. It is used mainly for soft matter molecular dynamics (MD) simulations with implementation in life sciences. GROMACS tracks the trajectories of a system of particles (atoms) that evolves in time by solving differential equations of motion at each time step. The coordinates and velocities of the particles are calculated by using their values from the previous time frame. In each time step, it calculates the forces acting on each atom, which is indeed the most time-consuming operation. From the computational point of view, the force acting on a particle is a result of a summation over:

[46] H. J. C. Berendsen, D. van der Spoel and R. van Drunen, "GROMACS: A message-passing parallel molecular dynamics implementation," *In Computer Physics Communications, ISSN 0010-4655, DOI: 10.1016/0010-4655(95)00042-E,* vol. 91, no. 1-3, pp. 43-56, 1995.

[47] E. Lindahl, B. Hess, D. van der Spoel and J. Mol, "GROMACS 3.0: a package for molecular simulation and trajectory analysis," *Molecular modeling annual, DOI: 10.1007/s008940100045,* vol. 7, no. 8, pp. 306-3017, 2001.

[48] D. van der Spoel, E. Lindahl, B. Hess, G. Groenhof, A. E. Mark and H. J. C. Berendsen, "GROMACS: Fast, flexible, and free," *Journal of Computational Chemistry, DOI: 10.1002/jcc.20291,* vol. 26, no. 16, p. 1701–1718, 2005.

[49] B. Hess, C. Kutzner, D. van der Spoel and E. Lindahl , "GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation," *Journal of Chemical Theory and Computation, DOI: 10.1021/ct700301q,* vol. 4, no. 3, p. 435–447, 2008.

[50] S. Pronk, S. Páll , R. Schulz, P. Larsson, P. Bjelkmar, R. Apostolov, M. R. Shirts, J. C. Smith, P. M. Kasson, D. van der Spoel, B. Hess and E. Lindahl, "GROMACS 4.5: a high-throughput and highly parallel open source molecular simulation toolkit," *Bioinformatics, https://doi.org/10.1093/bioinformatics/btt055,* vol. 29, no. 7, pp. 845-854, 2013.

[51] S. Páll , M. J. Abraham, C. Kutzner, B. Hess and E. Lindahl, "Tackling Exascale Software Challenges in Molecular Dynamics Simulations with GROMACS," *Markidis S., Laure E. (eds) Solving Software Challenges for Exascale. EASC 2014. Lecture Notes in Computer Science,* pp. pp 3-27, 2015

[52] M. J. Abraham, T. Murtola, R. Schulz , S. Páll , J. C. Smith, B. Hess and E. Lindahl, "GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers," *SoftwareX, DOI: 10.1016/j.softx.2015.06.001,* Vols. 1-2, pp. 19-25, 2015.

- pairs of particles connected by “bonds” – bonded interactions,
- pairs of particles satisfying some distance criteria – short-range non-bonded interactions,
- long-range corrections including calculations where data of all the simulated particles are needed – long-range interactions.

Usually, pairs of atoms are defined in a predefined cut-off radius calculating short-range interactions[53], while the long-range interactions are calculated using Fast Fourier Transform (FFT) based algorithms.

3.2 Application structure

GROMACS can run on almost every modern computing architecture[54]. The simulation program uses multi-level parallelism to utilize the computational power offered (see Figure 3.1). By means of domain decomposition, the calculations are spread along the distributed memory computational resources – compute nodes. On the domain decomposition level, MPI is used. At MPI rank level, a shared memory model is implemented with both OpenMP and GPU accelerators. Finally, SIMD registers are used to vectorise the calculations in each CPU core.

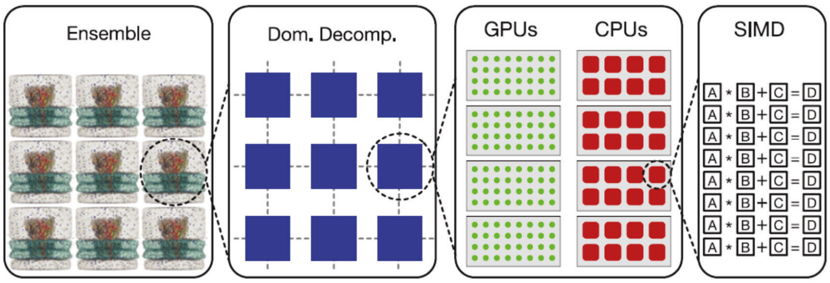

Figure 3.1: Multi-level parallelism in GROMACS.[54]

The Particle-mesh Ewald (PME) algorithm uses FFT to solve long-range electrostatic contributions to real-space direct Coulomb sums. The reader should consider the fact that the specific implementation involving All-to-All MPI communications causes the performance scalability to drop. The latter can be solved by overlapping real-space calculations and Fourier-space calculations. In GROMACS the MPI ranks are divided into two groups: one for real-space calculations (PP nodes) and the rest being dedicated to PME calculations (PME nodes).

[53] Computer Physics Communications 184 (2013) 2641–2650

[54] Abraham, et al. (2015) SoftwareX 1-2 19-25.

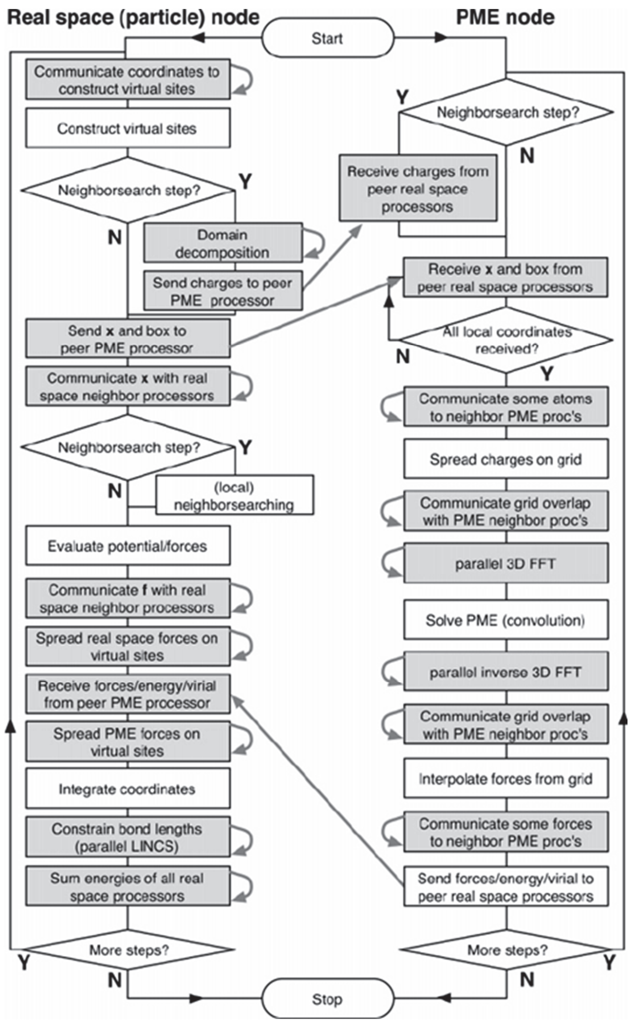
Real space (particle) node
Start
PME node
Communicate coordinates to construct virtual sites
Construct virtual sites
Neighborsearch step?
Y
N
Domain decomposition
Send charges to peer PME processor
Send x and box to peer PME processor
Communicate x with real space neighbor processors
Neighborsearch step?
Y
N
(local) neighborsearching
Evaluate potential/forces
Communicate f with real space neighbor processors
Spread real space forces on virtual sites
Receive forces/energy/virial from peer PME processor
Spread PME forces on virtual sites
Integrate coordinates
Constrain bond lengths (parallel LINCS)
Sum energies of all real space processors
More steps?
Y
N
Neighborsearch step?
Y
N
Receive charges from peer real space processors
Receive x and box from peer real space processors
All local coordinates received?
N
Y
Communicate some atoms to neighbor PME proc's
Spread charges on grid
Communicate grid overlap with PME neighbor proc's
parallel 3D FFT
Solve PME (convolution)
parallel inverse 3D FFT
Communicate grid overlap with PME neighbor proc's
Interpolate forces from grid
Communicate some forces to neighbor PME proc's
Send forces/energy/virial to peer real space processors
More steps?
Y
N
Stop

Figure 3.2: GROMACS flowchart for a typical simulation step for both particle and PME nodes.[55]

As shown in Figure 3.2, the resulting flowchart in an MD step can be described in the following manner. Each PP node has a corresponding PME node. At the beginning of the time step, each PP node sends coordinates and charges to its corresponding PME node and once the PME calculations are completed, each PME node sends the

[55] B. Hess, C. Kutzner, D. van der Spoel, E. Lindahl. GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation, Journal of Chemical Theory and Computation 4, 435-447 (2008)

resulting forces back to the corresponding PP node. Meanwhile, all collective communications proceed only between PME nodes as well as only between PP nodes and overlapping the FFT All-to-All communications (exchanged between PME nodes only) with real-space calculations. Consequently, one must optimise the number of PP and PME nodes in such a way that PME nodes need to send the forces that they have calculated in the exact moment when the PP nodes need Fourier-space forces, energies, etc. The GROMACS tool called *tune_pme* enables users to scan different combinations and start the simulation with an optimal PP/PME nodes ratio whilst the *mdrun* simulator further tunes the PME mesh and cut-off radius at the beginning of the MD simulation run. It should be noted that one can choose to execute bonded, non-bonded and long-range interactions on CPU or GPU devices.

3.3 Application mapping

How GROMACS is mapped to the Modular Supercomputing Architecture (MSA) depends on the simulation problem size and aims at optimizing the computational load. There are three computationally expensive components of the force to be calculated, namely bonded interactions, short-range non-bonded interactions, and long-range interactions (the same applies to the neighbour list construction, but this is not done every time step), and each of them can be run either on a CPU or on a GPU accelerator.

3.3.1 MD simulations of less than 10^4 particles (CM)

MD simulations of few tens of thousands of particles, e.g. many simulations of small peptide monomers in aqueous solution, should run efficiently on one node in the CM. In this case calculation time is comparable with communication time (computing node-to-computing node or host-to-device communications of small data buffers) and the performance scalability is limited. The overall flowchart of one MD integration step is show in Figure 3.3.

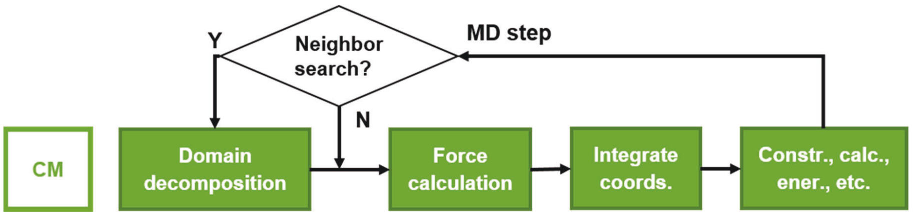

Figure 3.3: Schematic workflow of MD simulations with less than 10^4 particles in the MSA

3.3.2 MD simulations with number of particles of the order of 10^5 (ESB/DAM)

The ESB and DAM offer better performance for MD simulations consisting of hundreds of thousands of atoms (Figure 3.4), where particle-particle interactions are calculated on the GPU accelerators, while long-range electrostatic interactions run on the CPUs.

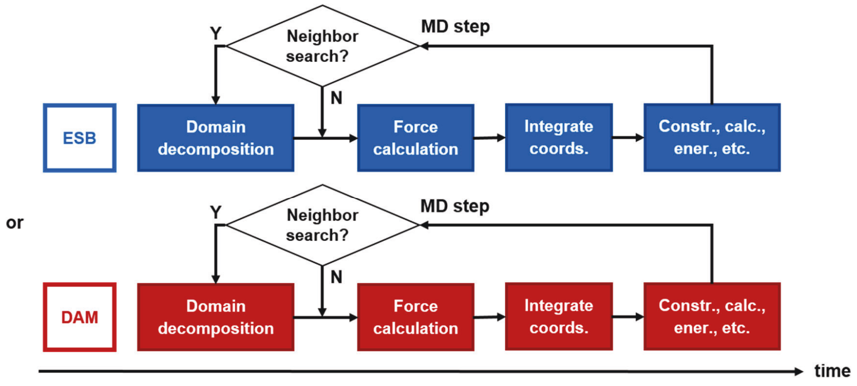

Figure 3.4: Schematic workflow of MD simulations with 10^5 particles in the MSA

3.3.3 MD simulations of millions of particles (ESB-CM)

MD simulations of large macromolecules and their complexes at reasonable time scales[56] demand computational resources with good enough performance scalability. When simulating the time evolution of systems consisting of more than several millions of particles, one should use thousands of cores/MPI ranks in the CM or tens/hundreds of nodes with GPU accelerators (like those in DAM). In the first case, the performance scalability saturates due to the enormous number of MPI process. In the second case, pair interactions go on the GPU accelerators while PME ranks run either on the CPUs or on a single GPU (GROMACS does not support PME calculations over multiple GPUs due to need for multiple GPU-to-CPU and vice versa data transfers for 3D FFT implementation). Single GPU performance is insufficient for PME calculations to deliver long-range forces to the PP nodes at the required speed, and would introduce imbalance causing performance scalability degradation. Moreover, for larger atomic systems PME calculations should be conducted on as few nodes as possible to minimise the time spent on All-to-All MPI communications. Therefore, the PME part of the simulation should run on nodes with powerful CPUs and good inter-node network, namely the CM. GPUs are very suitable for doing PP calculations, as mentioned above,

[56] Curr Opin Struct Biol. 2015 Apr; 31: 64–74

so the MSA offers good resource utilization and management when running very large MD simulations with GROMACS in ESB-CM configuration (see Figure 3.5). Both modules will communicate through the network.

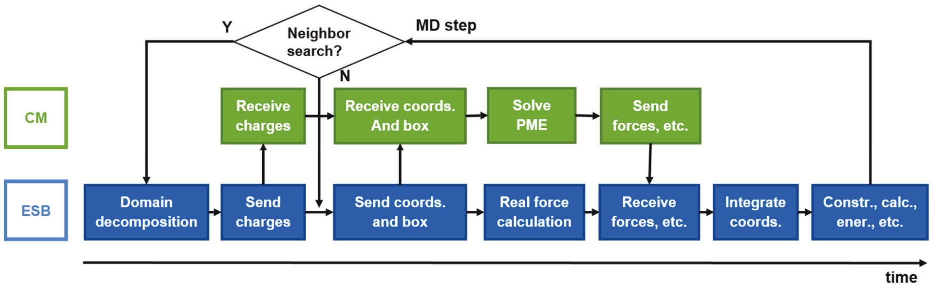

Figure 3.5: Schematic workflow of MD simulations with millions of particles in the MSA

3.3.4 MD simulations with big volumes (several million nm³)

Two different offload modes were investigated:

Run PME on ESB and PP on CM: The option to run PME on ESB and PP on CM is not supported natively in GROMACS; it only includes a single-node GPU implementation of PME. This option was implemented and tested but the alternative FMM option (see below) delivered better performance and efficiency.

Replace PME with FMM running on ESB or CM: The primary limiting factor in the PME method is that it utilizes a uniform mesh on the problem domain and the spacing of the mesh is a function of the cut-off radius at fixed accuracy. Solving for big volumes – in the order of several million nm^3, where the mesh size becomes larger than e.g. 1,000x1,000x1,000 – quickly becomes inefficient or even impossible. The Fast Multipole Method (FMM)[57] is gaining significant attention in the MD community lately, namely because of its O(N) complexity compared with the O(N*logN) complexity of PME-style methods. Due to its large multiplicative constant, it usually fails to achieve the execution times of PME-style methods on CPUs. However, the GPUs utilized in ESB promise to reduce by an order of magnitude the calculation times, and thus make the FMM method competitive. Additionally, the boundary element method (BEM)

[57] Rokhlin, Vladimir (1985). "Rapid Solution of Integral Equations of Classic Potential Theory." J. Computational Physics Vol. 60, pp. 187–207.

formulation of the continuum electrostatic model[58], local alternative charge distributions treatment with minimal overhead, and λ-dynamics module[59] have been applied. The result is a GPU-accelerated fast multipole method for GROMACS[60] [61]. In DEEP-EST we also include a multiple-GPU FMM implementation that runs on the ESB, and an FMM implementation for multiple-CPUs that can run on the CM. The respective application partitioning is depicted in Figure 3.6.

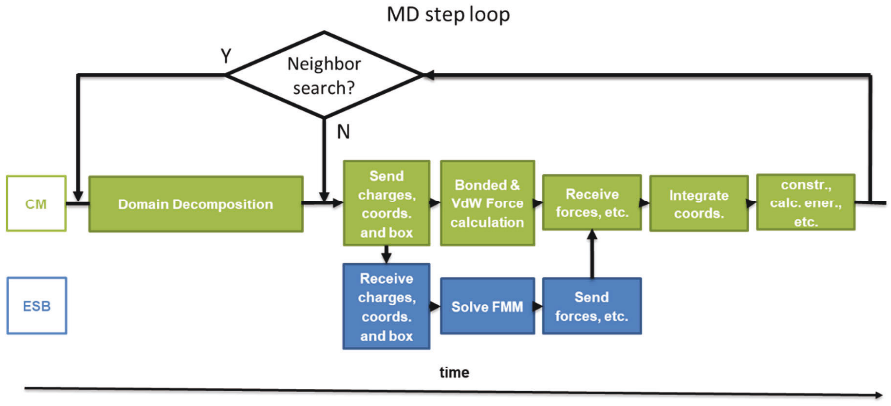

Figure 3.6: Schematic workflow of MD simulations with very big volumes in the MSA

3.4 Porting experience

The MSA can be utilized in ways not yet supported by the native GROMACS implementation, such as using the ESB to run PME calculations while CM runs PP calculation. Additionally, the computing power offered by the ESB can be harnessed for an efficient implementation of the FMM, which for certain MD simulation volumes can prove beneficial over PME. The SPPEXA project[62] already released a GROMACS

[58] Rio Yokota, Tsuyoshi Hamada, Jaydeep P. Bardhan, Matthew G. Knepley, Lorena A. Barba: Biomolecular Electrostatics Simulation by an FMM-based BEM on 512 GPUs. CoRR abs/1007.4591 (2010)

[59] Kohnke B. et al. (2020) GROMEX: A Scalable and Versatile Fast Multipole Method for Biomolecular Simulation. In: Bungartz HJ., Reiz S., Uekermann B., Neumann P., Nagel W. (eds) Software for Exascale Computing - SPPEXA 2016-2019. Lecture Notes in Computational Science and Engineering, vol 136. Springer, Cham. https://doi.org/10.1007/978-3-030-47956-5_17

[60] http://www.sppexa.de

[61] Kohnke, B., Kutzner, C., & Grubmüller, H. (2020). A GPU-accelerated fast multipole method for GROMACS: Performance and accuracy. *Journal of Chemical Theory and Computation, 16*(11), 6938-6949. doi:10.1021/acs.jctc.0c00744.

[62] http://www.sppexa.de/

version with a single-GPU implementation of FMM. In DEEP-EST we investigated the possibility of utilizing multi-GPU FMM for larger simulation volumes.

In order to study the scalability and performance of different approaches for long-range electrostatics treatment in the context of MSA, and to enable flexibility according to users' needs, we linked GROMACS with the standalone IRIS electrostatics library[63]. This non-invasive approach allows us to experiment with, and provide the user with solutions targeting MSA, while at the same time keeping the original optimized and certified GROMACS codes mostly intact. The newly developed features are available for GROMACS users by linking GROMACS to IRIS with minimal interventions.

Development of the IRIS library started in the PRACE-5IP[64] projects, with the main goal to provide MD code developers with an offloading of long-range electrostatic calculation to a dedicated group of MPI ranks in the manner that it is done in GROMACS. Such separation of short- and long-range interactions allows for better scalability of the MD application. Initially IRIS included a CPU-only version of the P3M[65] algorithm with 3D domain decomposition. It is similar to the SPME implementation in GROMACS, up to a slightly different Green function and interpolation scheme. In DEEP-EST we implemented the following changes to IRIS:

- 1D and 2D domain decomposition of the mesh, which greatly increases the overall performance and scalability compared to the already existing 3D version;
- Port to CUDA to support execution of the long-range contribution on multiple ESB nodes;
- Parallel CPU and parallel GPU versions of the FMM method, which allows running either on the CM or on the ESB.

The main challenges faced during the implementation of the aforementioned changes are related to:

- The unavoidable collective communication pattern inherent in the nature of the long-range electrostatic interaction;
- Memory transfers between the host and the device for the GPU versions of P3M and FMM.

Both these issues lead to poor scalability, since the time spent waiting for their completion cannot be reduced. In order to mitigate their impact, we used the following techniques:

[63] https://github.com/vpavlov/iris

[64] https://cordis.europa.eu/project/id/730913

[65] Roger W. Hockney; James W. Eastwood (1988). "Particle-Particle-Particle-Mesh (P^{3}M) Algorithms". *Computer simulation using particles*. CRC Press. pp. *267–304*. *ISBN* *9780852743928*.

- Overlap the collective communication with computation where possible by using CUDA asynchronous kernel execution and memory transfers;
- Utilising the CUDA-aware MPI implementation on the ESB to optimize the data transfer between GPU memory on different ESB nodes.

The implementation of the required changes consists of roughly 15,000 lines of code and together with testing and bug fixing it took about 7 PM.

3.5 Scalability

We measured the time to solution (T1), the time spent in MPI calls (T2), the time in GPU kernels (TG), the number of MPI messages, the volume of the exchanged data and the total duration of the MPI call of one and the same type. Based on these measurements, we estimated the load balance as:

$$\textbf{Load balance} = \frac{(1/N)\sum_{p=1}^{N}(T1_p - T2_p)}{MAX(T1_p - T2_p)} \times \mathbf{100}\%,$$

where N is the number of MPI ranks, $T1_p$ is the time to solution, and $T2_p$ is the time spent in MPI calls by MPI rank *p*. MAX() takes the maximum value among the all MPI ranks.

Dedicated test runs were performed to measure the performance scalability and parameters listed in the tables below for MD simulations of different size conducted on different number of codes in a single module – ESB, CM and multiple modules – ESB+CM. GROMACS performance was estimated based on 10,000 MD steps long runs, while the communications' profiling was done for 1,000 MD steps.

3.5.1 ESB Scalability results

Strong scaling

The timing data and calculated values of the parameters defined in the beginning of the section are shown in Table 3.1, Table 3.2, Table 3.3 and Table 3.4 for MD simulations with 1.25M, 20M, 40M and 80M atoms, respectively. The data for single nodes are not included due to the different computation model involved: on a single node all calculations are done in the GPU. When multiple nodes are used for a parallel simulation, particle-to-particle calculations run on the GPU, while the CPUs do PME calculations, using 8 MPI ranks per node and 1 OpenMP thread per MPI rank. The performance data are plotted in Figure 3.7 with the parallel efficiency shown in Figure 3.8. For MD simulations performance is measured as the amount of simulated time (nanoseconds) that can be calculated in one day (higher is better).

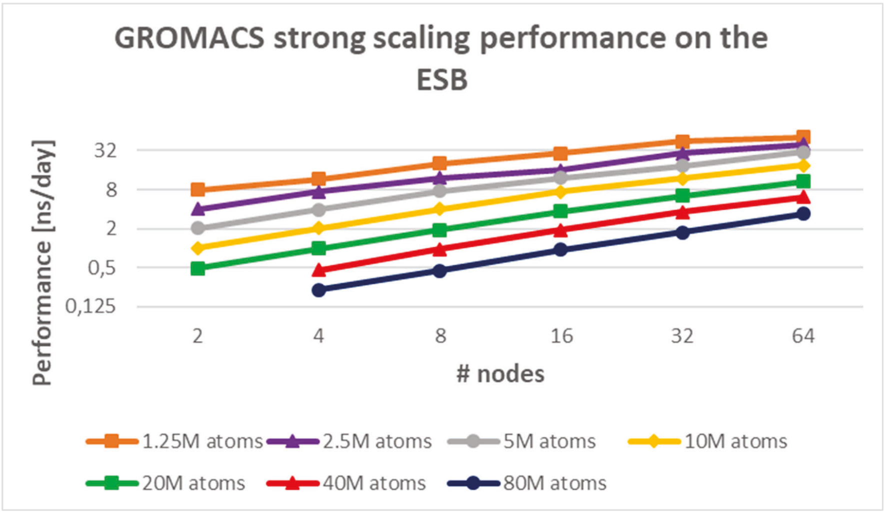

Figure 3.7: GROMACS strong scaling performance on ESB for MD simulations of different size

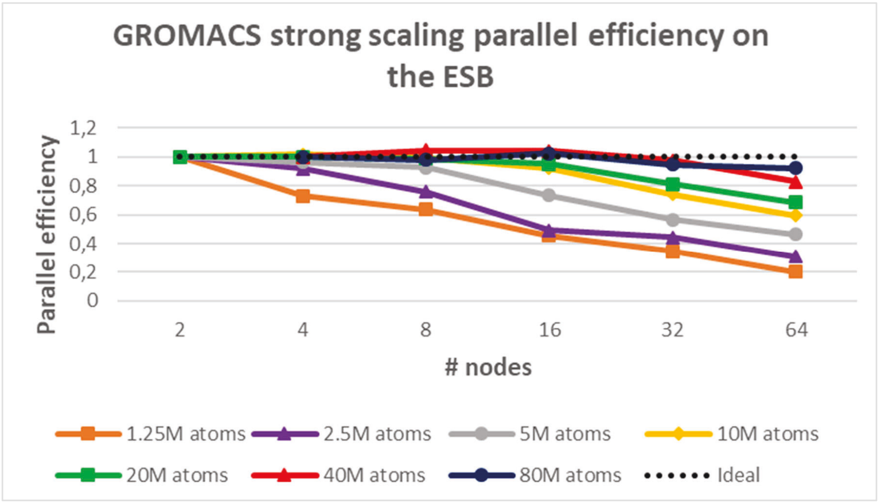

Figure 3.8: GROMACS parallel efficiency (strong scaling) on ESB for MD simulations of different size

# nodes	2	4	8	16	32	64
T1: Time to solution [s]	22.4	12.8	7.4	5.2	4.1	3.7
T2: Time spent in MPI [s]	4.3	3.3	2.2	2.0	2.0	2.2
TG: Time in GPU kernels [s]	22.4	12.8	7.4	5.2	4.1	3.7
Load balance [%]	93	94	93	86	85	77
# MPI messages [count]	451,779	897,767	1,796,201	3,837,839	8,309,987	27,067,919
MPI data volume [GByte]	98	135	89	133	194	262
Most important MPI operation	MPI_Sendrecv	MPI_Sendrecv	MPI_Sendrecv	MPI_Sendrecv	MPI_Sendrecv	MPI Sendrecv
Most important MPI operation [%]	76	82	80	77	73	69
2nd most important MPI operation	MPI_Alltoall	MPI_Alltoall	MPI_Alltoall	MPI_Alltoall	MPI_Alltoall	MPI_Alltoall
2nd most important MPI operation [%]	23	16	19	22	26	28
3rd most important MPI operation	MPI_Recv	MPI_Recv	MPI_Recv	MPI_Recv	MPI_Recv	MPI_Recv
3rd most important MPI operation [%]	0.7	1.3	0.4	0.5	0.5	1.9

Table 3.1: GROMCAS strong scaling measurements for the Bombinin test case with 1.25M atoms

As seen in the Table 3.1, the scalability of the MD simulation with 1.25M atoms saturates at 32 nodes, when the communication time becomes roughly equal to the computation time and even longer than computation time – for 64 nodes. For all other simulations, the computations takes longer than the communications and the load balance is above 93%. The most important MPI communication call is `MPI_Sendrecv`, taking more than 69% of the time spent in the MPI calls in all cases. The second most important MPI communication call is `MPI_Alltoall`, which takes almost all the rest of the MPI time, while the third most important MPI communication call takes less than 4% of the MPI time.

# nodes	2	4	8	16	32	64
T1: Time to solution [s]	352	177	91	47	27	16
T2: Time spent in MPI [s]	58	30	19	10	7	6
TG: Time in GPU kernels [s]	352	177	91	47	27	16
Load balance [%]	98	97	96	96	94	94
# MPI messages [count]	426,039	876,291	1,703,475	3,545,936	7,144,548	24,431,296
MPI data volume [GByte]	1025	1299	1622	2156	2876	3561
Most important MPI operation	MPI_Sendrecv	MPI_Sendrecv	MPI_Sendrecv	MPI_Sendrecv	MPI_Sendrecv	MPI_Sendrecv
Most important MPI operation [%]	80	73	66	62	62	60
2nd most important MPI operation	MPI_Alltoall	MPI_Alltoall	MPI_Alltoall	MPI_Alltoall	MPI_Alltoall	MPI_Alltoall
2nd most important MPI operation [%]	19	24	31	37	36	35
3rd most important MPI operation	MPI_Recv	MPI_Recv	MPI_Recv	MPI_Recv	MPI_Recv	MPI_Recv
3rd most important MPI operation [%]	0.9	1.9	2.1	0.4	0.6	2.4

Table 3.2: GROMACS strong scaling measurements for the Bombinin test case with 20M atoms

# nodes	4	8	16	32	64
T1: Time to solution [s]	262	178	92	49	28
T2: Time spent in MPI [s]	69	31	20	12	9
TG: Time in GPU kernels [s]	262	1782	922	492	282
Load balance [%]	96	98	93	95	94

# MPI messages [count]	824,046	1,723,736	3,323,946	7,198,028	14,259,647
MPI data volume [GByte]	2081	2599	3270	4273	5761
Most important MPI operation	MPI_Sendrecv	MPI_Sendrecv	MPI_Sendrecv	MPI_Sendrecv	MPI_Sendrecv
Most important MPI operation [%]	76	62	60	53	52
2nd most important MPI operation	MPI_Alltoall	MPI_Alltoall	MPI_Alltoall	MPI_Alltoall	MPI_Alltoall
[%]	21	35	38	46	46
3rd most important MPI operation	MPI_Recv	MPI_Recv	MPI_Recv	MPI_Bcast	MPI_Recv
[%]	1.9	1.5	1.8	0.5	0.7

Table 3.3: GROMACS strong scaling measurements for the Bombinin test case with 40M atoms

# nodes	**4**	**8**	**16**	**32**	**64**
T1: Time to solution [s]	752	376	188	95	51
T2: Time spent in MPI [s]	156	87	46	22	13
TG: Time in GPU kernels [s]	752	376	188	95	51
Load balance [%]	94	96	97	96	96
# MPI messages [count]	854,543	1,635,915	3,380,131	6,683,527	14,329,559
MPI data volume [GByte]	3564	4256	5275	6536	8552
Most important MPI operation	MPI_Sendrecv	MPI_Sendrecv	MPI_Sendrecv	MPI_Sendrecv	MPI_Alltoall
Most important MPI operation [%]	77	77	59	52	52

2nd most important MPI operation	MPI_Alltoall	MPI_Alltoall	MPI_Alltoall	MPI_Alltoall	MPI_Sendrecv
2nd most important MPI operation [%]	19	21	39	46	47
3rd most important MPI operation	MPI_Recv	MPI_Recv	MPI_Recv	MPI_Recv	MPI_Bcast
3rd most important MPI operation [%]	4.1	1.5	1.3	1.0	0.6

Table 3.4: GROMACS strong scaling measurements for the Bombinin test case with 80M atoms

The strong scalability gets better when increasing the problem size as seen in Table 3.3 and Table 3.4 for MD simulation with 40M and 80M atoms, respectively. The parallel efficiency is close to the ideal one as show in Figure 3.8. This trend holds until the duration of the PME calculations running on the CPUs of the ESB nodes would not exceed the duration of the PP calculations running on the GPUs of the ESB. Such a condition ensures overlapping communications in the PME part (on the CPUs) with the particle-to-particle calculation (on the GPUs).

Weak scaling

Figure 3.9 and Figure 3.10 show the weak scalability of the application for the evaluated simulations. In this scenario the volume of work per node is kept constant by running the 2.5M system on 2 nodes, the 5M system on 4 nodes, the 10M system on 8 nodes, the 20M system on 16 nodes, the 40M system on 32 nodes, and the 80M system on 64 nodes. As visible from Table 3.2 and Table 3.4, the `MPI_Alltoall` share of the total communication time rises from 37% on 16 nodes to 52% on 64 nodes, which limits the weak scalability when increasing the number of nodes.

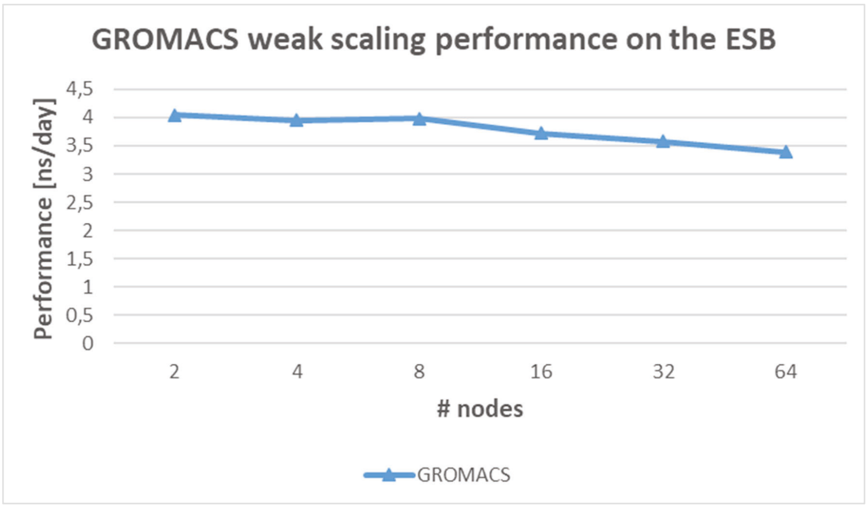

Figure 3.9: GROMACS weak scaling performance on the ESB

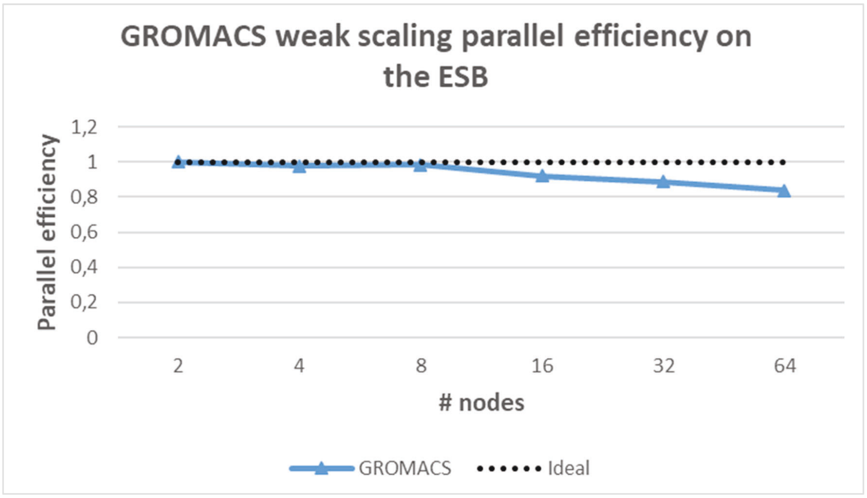

Figure 3.10: GROMACS weak scaling parallel efficiency on the ESB

3.5.2 CM Scalability results

Strong scaling

The strong scaling performance of MD simulations with 2.5M, 20M, and 80M atoms when running GROMACS on the CM is shown in Figure 3.11 and the corresponding parallel efficiency is shown in Figure 3.12. For these experiments 24 MPI ranks (18 for PP and 6 for PME calculations) per node and 2 OpenMP threads per MPI rank were used. The MD simulations with numbers of atoms between 300k and 2M show good scalability; bigger simulations presented worse strong scalability due to the limiting effect of collective communications between the PME ranks. Overall, the single-module MD simulations on both CM and ESB show good scalability for the entire range of simulation sizes up to several millions of atoms.

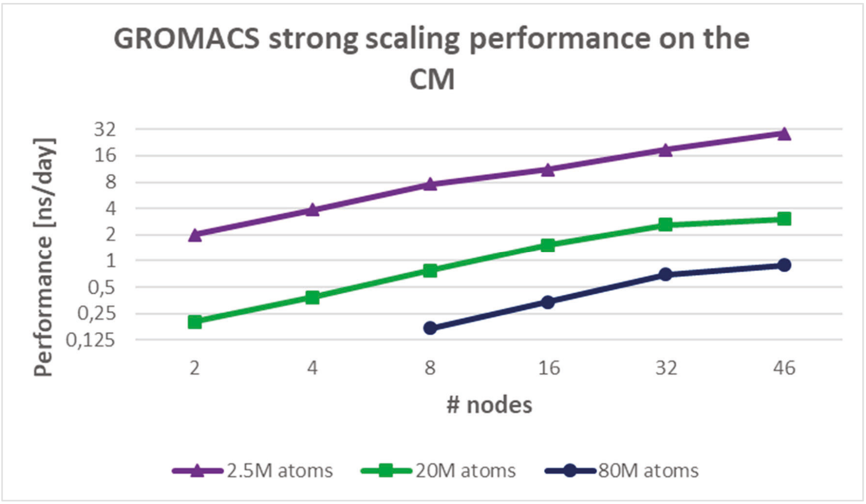

Figure 3.11: GROMACS strong scaling performance on the CM for MD simulations of different size

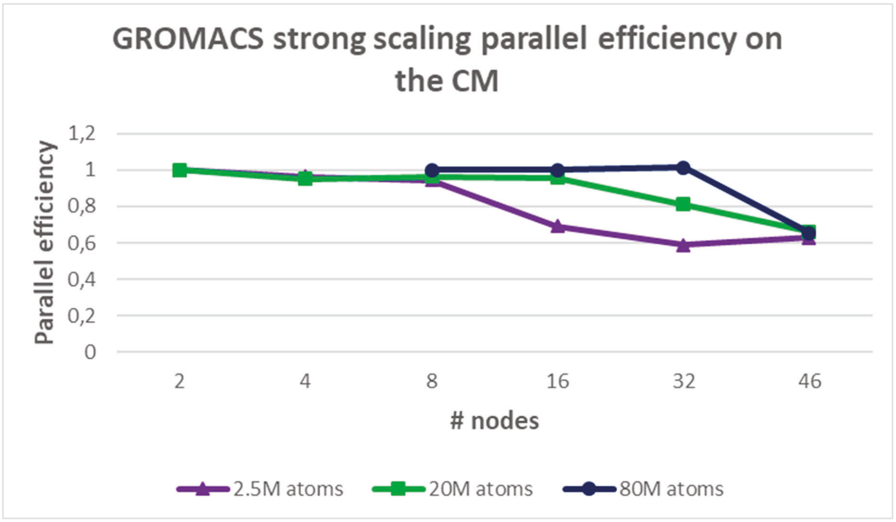

Figure 3.12: GROMACS strong scaling parallel efficiency on the CM for MD simulations of different size

3.5.3 ESB+CM Scalability results

Strong scaling GROMACS with PME

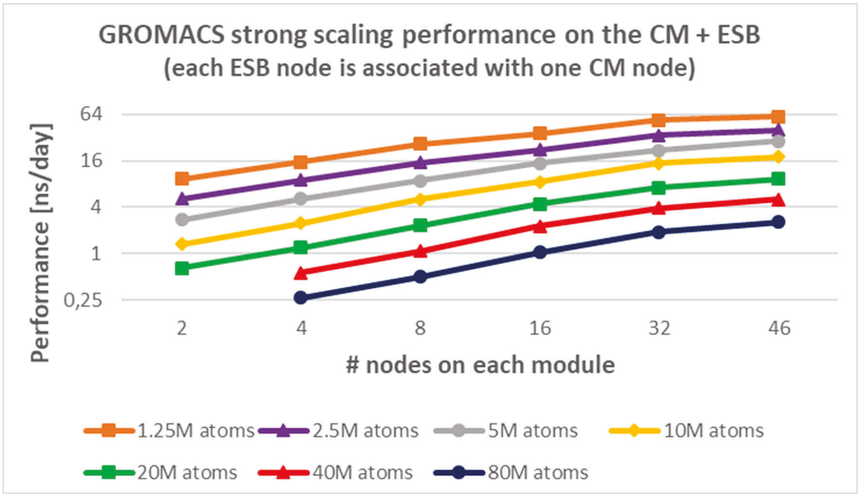

Figure 3.13: GROMACS strong scaling performance in Cluster-Booster configuration on the ESB (PP) and the CM (PME) for MD simulations of different size

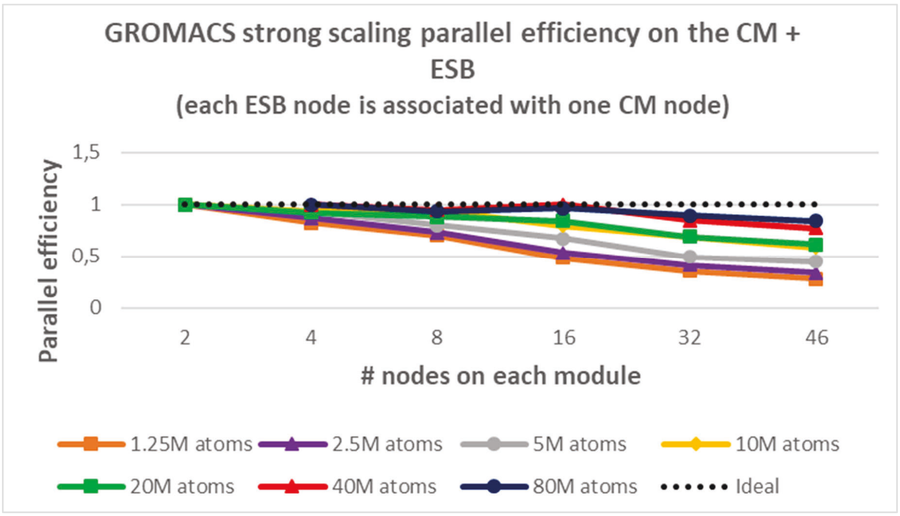

Figure 3.14: GROMACS strong scaling parallel efficiency in Cluster-Booster configuration on the ESB (PP) and the CM (PME) for MD simulations of different size

As discussed above, MD simulations in Cluster-Booster configuration run the particle-to-particle (PP) calculations on the ESB and long-range electrostatics calculations with PME method on the CM. The optimal performance was reached with a 1:1 ratio of ESB nodes to CM nodes. The GROMACS performance of MD simulations with 1.25M, 2.5M, 5M, 10M, 20M, 40M, 80M atoms is plotted in Figure 3.13 and the corresponding parallel efficiency in Figure 3.14. Detailed comparison of the corresponding performance of the ESB-only runs (plotted in Figure 3.7) showed performance gain of between 10% and 40% as depicted in Figure 3.15. The relative performance gain was calculated according to the following formula:

$$RPG = \frac{P_{ESB+CM} - P_{ESB}}{P_{ESB}} \times 100\%,$$

where *RPG* denotes the Relative Performance Gain, P_{ESB+CM} the performance in Cluster-Booster configuration, and P_{ESB} the performance on the ESB module.

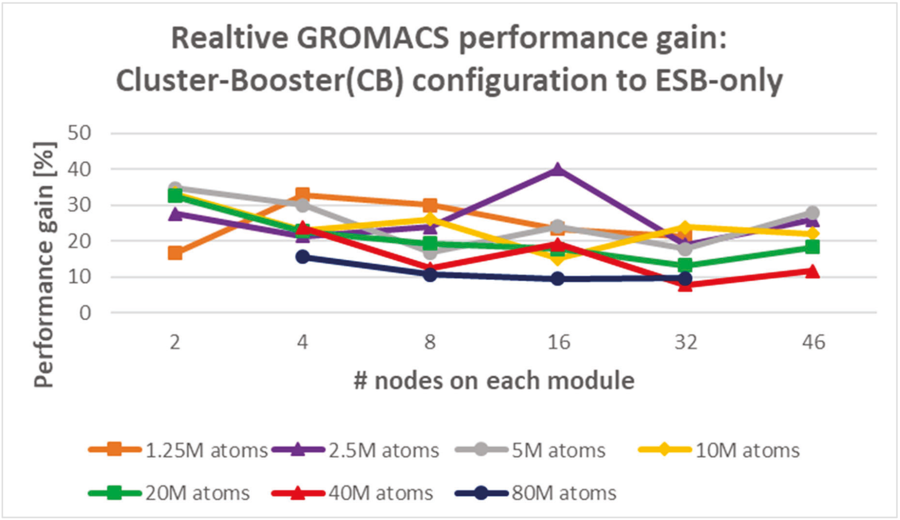

Figure 3.15: Relative performance gain of GROMACS in Cluster-Booster configuration to single module configuration (ESB) for MD simulations with different size

Weak scaling GROMACS with FMM

In most cases, PME outperforms FMM, while the latter starts to become beneficial for large volumes of the simulation box. However, GROMACS has an intrinsic hard limit for the input number of atoms in a system (~100 million). For dense systems such as the ones used in life sciences research this limit is reached at approximately ~1 Million nm^3 (Mnm^3) volumes. In order to perform weak scalability of FMM for much larger volumes, a sparse system needs to be taken as a test case. To evaluate the weak scalability of the newly implemented multi-GPU FMM method integrated with GROMACS, a starting aerosol problem containing 75 droplets of water[66] (217,326 atoms) in a simulation box of volume ~5 Mnm^3 is n-folded up to 32 times. The largest problem obtained in this way contains 6,954,432 atoms in a simulation box with a volume of ~160 Mnm^3. The problem is simulated for 200 MD steps on increasing numbers of ESB nodes. The number of MPI tasks on the CM is determined in order to minimize energy usage while keeping the performance balance between the ESB and CM. The amount of CM nodes needed is reduced because they do not need to calculate the pair-wise Coulomb interactions, which are already included in the FMM algorithm. The execution becomes therefore generally faster. Table 3.5 shows the obtained measurements.

[66] https://www.mpibpc.mpg.de/17532883/03_aerosol.tgz

# nodes CM module	1	1	1	2	3	3
# nodes ESB module	1	2	4	8	16	32
Full System						
T1: Time to solution [s]	11	16	35	11	18	39
T2: Time spent in MPI [s]	4.3	8.0	13.4	3.8	8.7	16.2
Load balance [%]	69	52	62	69	62	62
# MPI messages	4,672	12,843	16,162	97,064	289,524	532,028
MPI data volume [GByte]	1.5	4.0	11.4	20.8	47.3	120.5
Module CM						
T1: Runtime on Module CM [s]	11	16	35	11	18	39
T2: Time spent in MPI [s]	4.3	9.9	19.8	4.6	10.6	23.5
# MPI messages CM	1,686	5,144	5,212	31,436	55,411	82,498
MPI data volume [GB]	0.02	0.09	0.18	0.47	1.15	1.58
Load balance [%]	95	96	96	81	95	89
Module ESB						
T1: Runtime on Module ESB [s]	11	16	35	11	18	39
T2: Time spent in MPI [s]	0.2	0.3	0.5	0.7	0.8	1.7
T_GPU: Time spent in GPU kernels [s]	8	14	31	8	15	34
# MPI messages ESB	200	1,000	1,000	1,000	1,000	1,000
MPI data volume [GB]	0.0	1.0	5.4	8.7	23.0	72.0
Load balance [%]	100.00	99.99	99.91	99.59	99.68	99.32
Inter-modular communication						
MPI data transfer time [s]	0.2	0.1	0.3	0.1	0.3	0.5
# of MPI messages	2,786	6,699	9,950	64,628	232,942	448,530
MPI data volume [GB]	1.5	2.9	5.9	11.6	23.2	46.9
Most important MPI operation	MPI_Recv	MPI_Recv	MPI_Waitall	MPI_Waitall	MPI_Waitall	MPI_Waitall

Most important MPI operation [%]	59	72	61	70	74	59
2nd most important MPI operation	MPI_Waitall	MPI_Waitall	MPI_Recv	MPI_Recv	MPI_Recv	MPI_Recv
2nd most important MPI operation [%]	38	24	37	25	22	36
3rd most important MPI operation	MPI_Isend	MPI_Isend	MPI_Isend	MPI_Isend	MPI_Isend	MPI_Isend
3rd most important MPI operation [%]	1.8	3.0	2.1	3.5	3.0	3.5

Table 3.5: GROMACS + IRIS/FMM weak scaling measurements for the aerosol test case

Figure 3.16 shows the time to solution for the different cases. A distinctive pattern is observed, which at first might lead to the conclusion that this method does not scale at all. However, it is misleading to perform a weak scaling test of the FMM method by doubling the number of processors and problem size; instead, weak scaling should be performed by multiplying the number of processors and problem size by 8 each time.

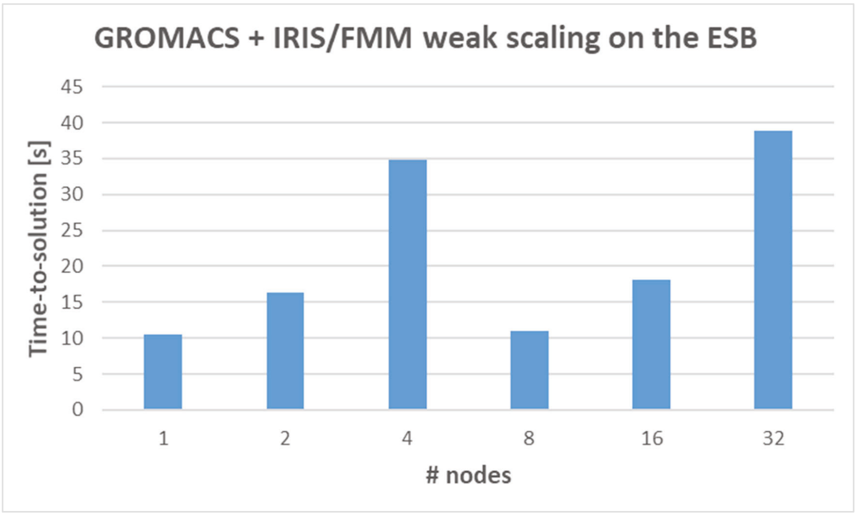

Figure 3.16: GROMACS + IRIS/FMM weak scaling time to solution (11 MD steps). Tree depth for 1-, 2- and 4- ESB node cases is 4, while for 8-, 16- and 32- ESB node cases is increased to 5

The FMM method relies on dividing the domain into an oct-tree up to certain configurable depth. Each cell in the tree (except for the leaf nodes) has exactly 8 children. From algorithmic point of view, it is not beneficial to increase the depth unless there are 8 times more processors. This is why the tree depth for the 1-, 2- and 4-ESB node cases is kept constant (depth 4) and for the 8-, 16- and 32-ESB node cases it is increased to 5. By doubling the simulation system size but keeping the depth constant, we end up with leaf cells containing twice the number of atoms compared with the previous case, which inevitably leads to increased simulation time. However, comparing the cases 1-node to 8-node, as well as 2-node to 16-node and 8-node to 32-node, we can see the true weak scalability, as depicted on Figure 3.17. The obvious result from these measurements is that weak scalability for larger number of nodes is preserved, if the depth is increased each time that the number of nodes grows by 8×.

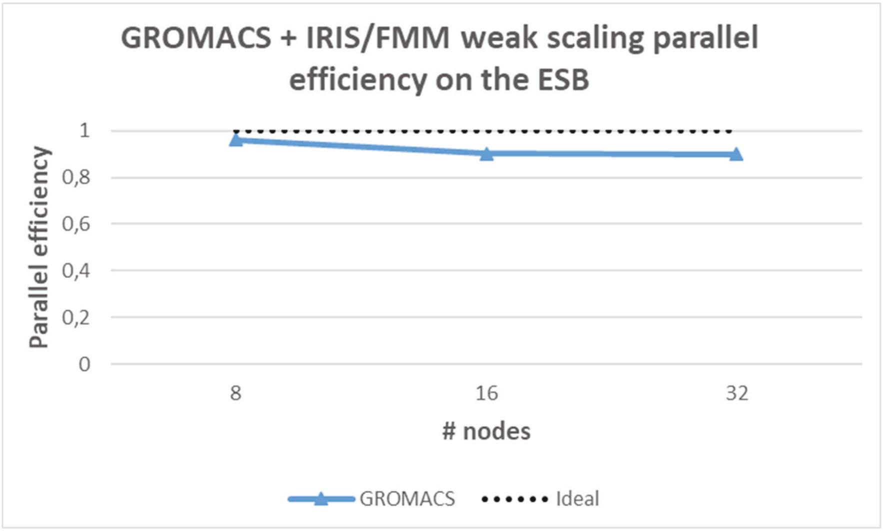

Figure 3.17: GROMACS + IRIS/FMM weak scalability, comparing 8- to 1- ESB nodes, 16- to 2- ESB nodes and 32- to 4- ESB nodes

Strong scaling FMM (Standalone IRIS/FMM)

The sparse aerosol system used for the weak scalability tests cannot be used to measure accurately the strong scalability of the FMM method, since there is large load imbalance due to its inhomogeneity. This load imbalance becomes even larger with increasing number of processors, which hinders the scalability. A dense problem is more suitable for strong scalability tests. Moreover, there are previous results[67] showing strong scalability of standalone multi-GPU FMM implementation for a problem consisting of 100M particles. To this end, the test case chosen for strong scalability tests of the FMM method consists of 100M atoms worth of water molecules in 1 Mnm^3 simulation box. Due to the hard limit for the input size in GROMACS such a system cannot be fully simulated. Instead, only the FMM part as implemented in the IRIS library was run on the ESB nodes. The results obtained this way show the strong scaling potential of the developed FMM code itself.

The problem is simulated for 11 MD steps on increasing number of ESB nodes. The CM nodes are used only to load the input data and send it through to the ESB nodes for calculation, thus better representing the situation in an eventual full-simulation scenario. One MPI CM rank corresponds to 1 ESB node. Table 3.6 shows the results. Note that the data for the CM nodes is not representative because of the above comment.

# nodes CM module	**1**	**1**	**1**	**1**	**1**	**2**
# nodes ESB module	1	2	4	8	16	32
Full system						
T1: Time to solution [s]	2,204	1,137	564	287	149	77
T2: Time spent in MPI [s]	1,096	579	289	144	73	38
Load balance [%]	51	52	52	51	52	55
# MPI messages	145	255	519	1,572	5,796	22,692
MPI data volume [GByte]	36	42	48	57	65	78
Module CM						
T1: Runtime on Module CM [s]	2,204	1,137	564	287	149	78
T2: Time spent in MPI [s]	2,144	1,127	557	285	143	72
# MPI messages CM	0	0	0	0	0	0
MPI data volume [GB]	0	0	0	0	0	0
Load balance [%]	100.0	99.8	99.5	98.5	97.1	99.2
Module ESB						

[67] Rio Yokota, Tsuyoshi Hamada, Jaydeep P. Bardhan, Matthew G. Knepley, Lorena A. Barba: Biomolecular Electrostatics Simulation by an FMM-based BEM on 512 GPUs. CoRR abs/1007.4591 (2010)

T1: Runtime on Module ESB [s]	2,204	1,137	564	287	149	78
T2: Time spent in MPI [s]	49	31	22	3	3	3
T_GPU: Time spent in GPU kernels [s]	2,120	1,113	532	275	143	71
# MPI messages ESB	11	55	55	55	55	55
MPI data volume [GB]	4.10E-07	5.3	12	20	29	41
Load balance [%]	100	99.8	99.6	99.2	98.5	97.3
Inter-modular communication						
MPI data transfer time [s]	6.1	3.6	1.7	0.7	0.3	0.2
# of MPI messages	134	200	464	1,517	5,741	22,637
MPI data volume [GB]	36.3	36.3	36.3	36.3	36.3	36.3
Most important MPI operation	MPI_Recv	MPI_Recv	MPI_Recv	MPI_Recv	MPI_Recv	MPI_Recv
Most important MPI operation [%]	78	69	75	63	66	57
2nd most important MPI operation	MPI_Wait	MPI_Wait	MPI_Wait	MPI_Wait	MPI_Wait	MPI_Wait
2nd most important MPI operation [%]	20	30	22	33	30	36
3rd most important MPI operation	MPI_Iprobe	MPI_Isend	MPI_Isend	MPI_Isend	MPI_Isend	MPI_Isend
3rd most important MPI operation [%]	1.9	0.9	2.0	3.4	4.7	6.3

Table 3.6: Standalone IRIS/FMM strong scaling measurements for the 100M test case

The presented data shows that the strong scalability of the multi-GPU FMM code has a nearly perfect parallel efficiency (see Figure 3.19). The time spent in GPU kernels also scales near ideally and dominates the execution time, as shown in Figure 3.18. The communication between ESB ranks is completely overlapped by the calculations on the GPU kernels (more specifically P2P self-interactions) and does not contribute to the total step time. Moreover, the MPI time is less than 4% of the total execution time.

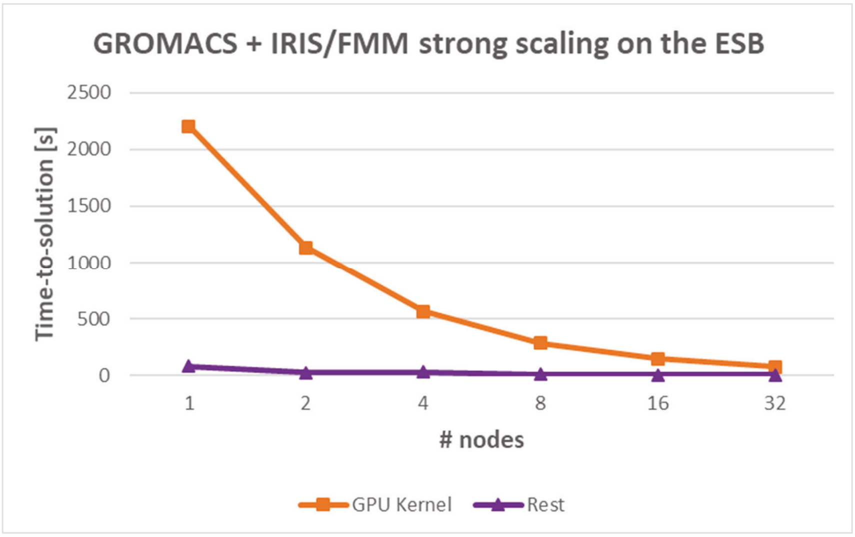

Figure 3.18: Standalone IRIS/FMM time to solution for the GPU kernels and the complete ESB step. GPU kernels time is shown in blue, while all the rest of the activities (data preparation, CPU activities, data transfer) is shown in orange

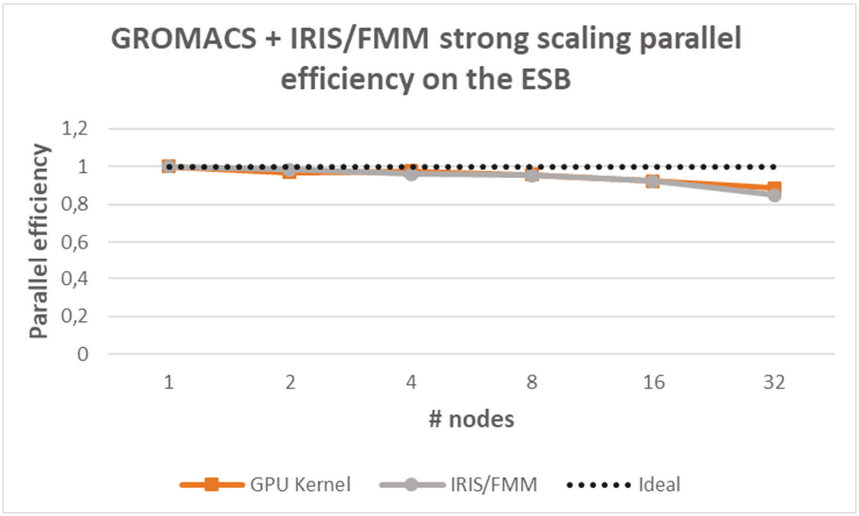

Figure 3.19: Standalone IRIS/FMM parallel efficiency for the GPU kernels (shown in orange) and the complete ESB step (shown in grey)

3.5.4 Our path to Exascale

In order to extrapolate towards Exascale we need to look at the details of the computation load and communication patterns of a single MD step and assess their inherent scalability.

3.5.4.1 P3M/PME

The PME/P3M MD step consists of the following components:

- Receive input data;
- Particle to Mesh;
- Halo exchange;
- Forward 3D FFT, including remap;
- Calculate reciprocal space electrostatic energy;
- 3x Backward 3D FFT, including remap;
- Mesh to Particle;
- Send output results.

The ***Receive input data*** and ***Send output results*** component involves asynchronous point-to-point communication only (non-blocking send to blocking receive) and its duration depends on the amount of data transferred and the latency and throughput of the network.

The ***Particle to Mesh*** and ***Mesh to Particle*** components perform only computations. Their complexity depends on the number of atoms per rank and size of the computation mesh. Both strong and weak scaling should not be limited.

The ***Halo exchange*** involves point-to-point communication only and the amount of data to be transferred depends on the accuracy defined by the user. In the strong scaling case, the scalability is limited by the interconnecting network bandwidth.

There are two main subcomponents in the ***3D FFT***, namely 2D or 1D FFT local calculations and collective communications to remap the mesh to prepare it for the FFT in the remaining dimension(s). All ranks are involved in the collective communications and they are the main source of scalability saturation in both strong and weak scaling cases, while the FFT calculations do not influence the performance scalability. Moreover, the duration of the collective all-to-all communication heavily depends on the size of the computational mesh, the whole span of which needs to be exchanged, and this greatly influences weak scalability.

3.5.4.2 FMM

The FMM MD step consists of the following components:

- Receive input data;
- Local tree construction;
- Exchange of the Local Essential Tree;
- Dual tree traversal;
- Send output results.

The ***Receive input data*** and ***Send output results*** components involve asynchronous point-to-point communication only (non-blocking send to blocking receive) and their duration depends on the amount of data transferred and the latency and throughput of the network.

For the strong scaling case the amount of data transferred to a single MPI task is reduced as the number of ranks is increased. For the weak scaling case the amount of data transferred to a single MPI rank is constant. The total amount of data is increased, along with the number of messages. In both cases the scalability of the component is limited mainly by the latency and throughput of the network.

The ***Local tree construction*** component involves only computation and all steps are of O(N) complexity. Both strong and weak scaling should not be limited.

The ***Exchange of the Local essential tree*** component involves two all-to-all communications: one for exchanging the P2P halo atoms and one for exchanging the cells needed by other processors to perform M2L kernels. These exchanges are overlapped with the P2P in-cell interactions computed on the GPU. The solver can be optimally parametrized by the user so that this communication is completely hidden. For the CPU implementation however, this is the main bottleneck of the method.

This component also involves additional calculations for reconstructing the non-local part of the tree shared by all nodes. For both strong and weak scaling, the additional calculations in the local essential tree stays generally of the same order and does not scale, but their duration can be made relatively small if the performance of the CPU/GPU is high enough. Thus, the scalability is limited by the single node FPU performance.

The ***Dual tree traversal*** component involves only computation and all steps involved are of O(N) complexity. Both strong and weak scaling should not be limited.

3.5.4.3 What are the limitations? – Can they be fixed?

For MD simulations as a whole, the main scalability limiting factor is the number of atoms per MPI rank. When the number of atoms per node (CPU only) goes below roughly 1,000, the communication starts dominating.

The main limitation of the **PME** method is its weak scalability when the problem volume approaches millions of nm^3 in realistic scenarios with Fourier spacing not exceeding 2.2 Ǻ; in this case the all-to-all MPI communications necessary for the 3D FFT become a severe limiting factor. This limitation cannot be fixed since it is an inherent characteristic of the method and these communications cannot be overlapped with meaningful computation. This is the main reason for developing the multi-GPU FMM code as part of this project. Apart from that limitation, PME shows good strong scalability and excellent performance for most MD simulations required in life sciences nowadays and can be used exceptionally well in ensemble simulations.

FMM provides a viable alternative for large problems[68] and enables simulations that are not feasible with PME nowadays. Its strong scalability is limited by the computation vs. data transfer ratio, and specifically in the GPU case by the computation vs. memory transfer ratio. For the hardware used in the ESB nodes this happens when the single step wall-clock time starts approaching ~100ms (3.5 ns/day at 4 fs MD time step).

Another important factor that limits both the strong and weak scalability in MD simulations, regardless of the method used, is load imbalance due to problem inhomogeneity. In GROMACS there is a dynamic load-balancer that aims to mitigate this problem by rescaling the local domains.

3.5.4.4 How to use future Exascale systems

According to the application's present status, the Exascale performance could be reached either by a combination of ensemble and strong scaling, or by weak scaling.

In drug design, they investigate the interaction of a particular protein with many ligands, which results in running many MD simulations to solve drug candidate discovery. The strong scaling limit of the single MD simulation determines the number of nodes per MD simulation, and the number of simultaneously running simulations depends on the available resources. The I/O operations per MD simulation, namely writing trajectories, are done once per second on average, so they are not expected to play a limiting role.

Large MD simulations are used in molecular biology, polymer science, material science, etc. Such MD simulations include increasingly larger space volumes with

[68] Tchipev N, Seckler S, Heinen M, et al. The International Journal of High Performance Computing Applications. 2019;33(5):838-854. doi:10.1177/1094342018819741

number of particles in the order of hundreds of millions or even billions[69]. In such cases, the number of particles per node is kept optimal, and the MD simulation is run on the corresponding number of nodes (increasing the number of nodes). Such large volumetric problems can be solved using the FMM and having in mind that its strong scalability starts deteriorating when the wall-clock time of the single step starts approaching ~100 ms on current hardware. We can therefore conclude that it can become possible to simulate a problem involving billions of particles on pre-Exascale and Exascale systems with at least 5 ns/day performance (at 4 fs/step).

3.5.4.5 Where did the DEEP-EST project help on the way to Exascale?

The MSA idea behind the DEEP-EST project allows MD simulation packages to run algorithmically different parts of the problem on more appropriate computing architecture to optimize the price/performance ratio of the computation. It adds versatility and allows choosing the right combination of nodes depending on the simulation size in order to achieve the best performance with as little energy as possible. This would be impossible in a homogenous system with a unique type of nodes.

The multi-GPU FMM implementation, which enables the computation of very large problems, further benefits from the MSA idea by utilizing the low-performance CPUs of the ESB nodes to do the FMM-related housekeeping tasks, like dual-tree traversal, LET construction and communications. In the meantime, the high-performance CPUs of the CM nodes are busy with bonded interactions and Van der Waals computations. Moreover, there is flexibility to tune the number of CM nodes against ESB nodes for better load balance. Keeping the particle-to-particle ranks and the FMM ranks on separate modules allows the user to bundle the particle-to-particle ranks on a smaller number of CM nodes, thus reducing the network load by keeping most of the interactions in memory.

3.6 Energy consumption

The energy consumption was measured for runs shown in Section 3.5. Here, only the data for MD simulations of 2.5M, 20M and 80M atoms are presented to illustrate the energy consumption for medium, big and large simulations. The data collected for 10,000 MD steps runs on the CM, ESB and Booster-Cluster configuration (ESB+CM) for different MD simulations sizes are plotted in Figure 3.20, Figure 3.21, and Figure 3.22 for 2.5M, 20M and 80M atoms, respectively. In ESB+CM configuration the number

[69] Jung, J., Nishima, W., Daniels, M., Bascom, et al. J. Comput. Chem. 2019, 40, 1919– 1930. DOI: 10.1002/jcc.25840

of ESB nodes equals the number of CM nodes and the sum of both kinds of nodes is plotted. Long-range electrostatics was calculated with PME.

These plots show that the ESB consumes the lowest amount of energy for all runs, while the CM has about 4 times greater consumption on average. The ESB-only and ESB+CM configurations show relatively good and constant behaviour in the strong scaling scenario for 20M and 80M MD simulations.

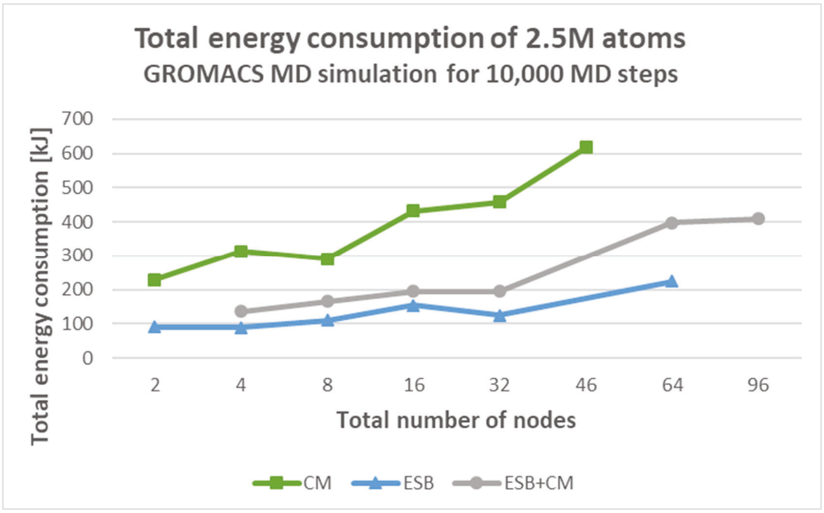

Figure 3.20: Total energy consumption of 2.5M atoms GROMACS MD simulation for 10,000 MD steps

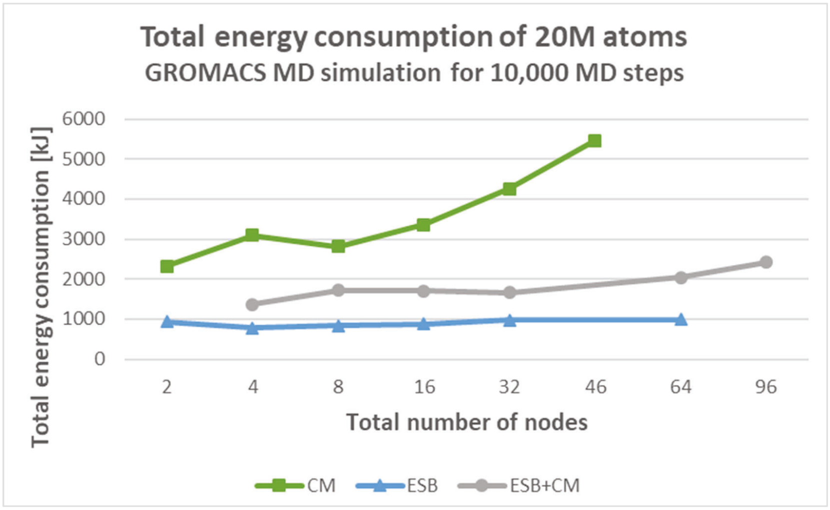

Figure 3.21: Total energy consumption of 20M atoms GROMACS MD simulation for 10,000 MD steps

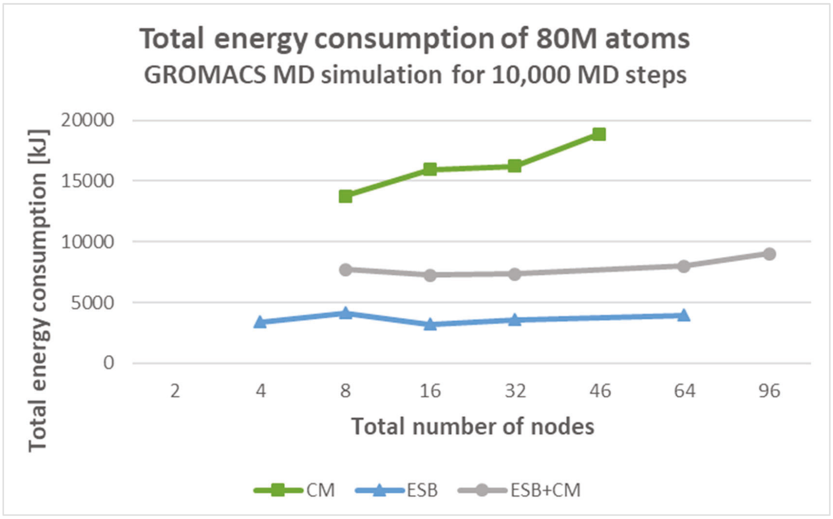

Figure 3.22: Total energy consumption of 80M atoms GROMACS MD simulation for 10,000 MD steps

3.7 Performance comparison

This sections gives an overview on the energy/performance ratio and the comparison of the old and new algorithms.

3.7.1 Energy/Performance ratio

The Energy vs. Performance ratio is measured in MJ/ns (MegaJoule/nanosecond) and estimates the energy spent to simulate one nanosecond of time evolution. Energy vs. Performance of the CM, ESB and ESB+CM configuration for different MD simulation sizes are plotted in Figure 3.23, Figure 3.24, and Figure 3.25 for 2.5M, 20M and 80M atoms, respectively. In Cluster-Booster configuration the number of ESB nodes equals the number of CM nodes and the total number of nodes is plotted. Long-range electrostatics was calculated with PME.

The ESB has the best Energy vs. Performance ratio and stays constant in the strong scaling scenario for MD simulations with more than 2.5M atoms, as it does the Cluster-Booster configuration. The increase in the performance of ESB+CM configuration shown in Figure 3.15 comes at the expense of higher energy consumption.

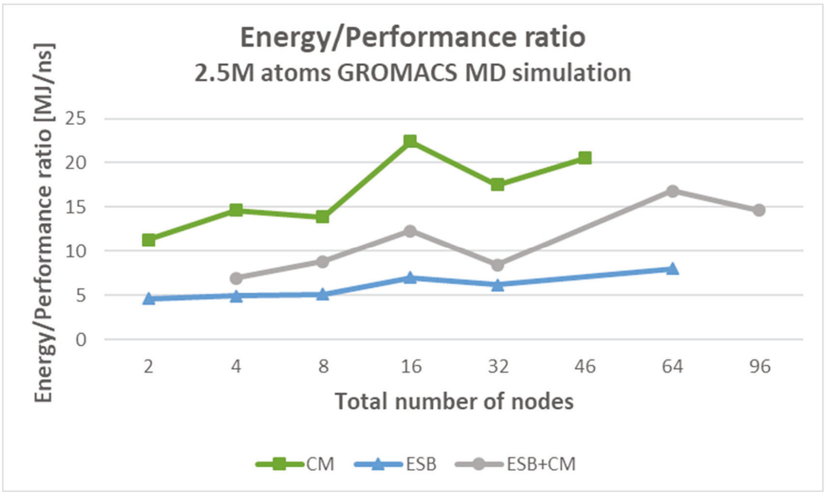

Figure 3.23: Energy vs. Performance ratio of 2.5M atoms GROMACS MD simulation

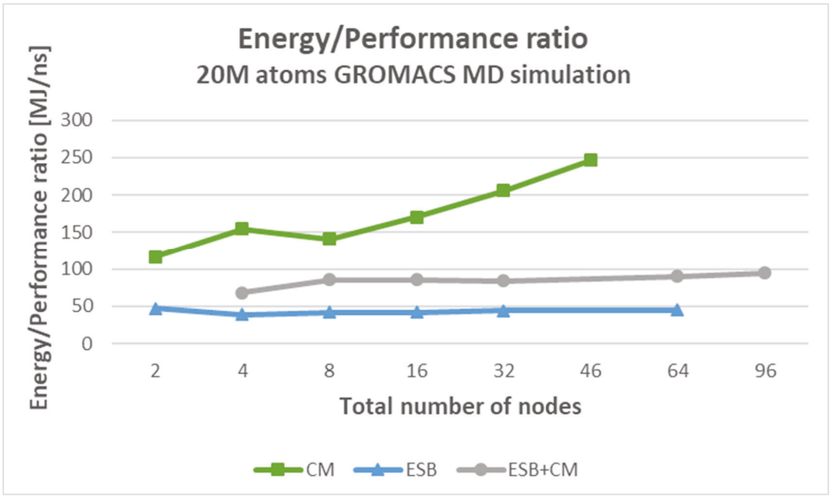

Figure 3.24: Energy vs. Performance ratio of 20M atoms GROMACS MD simulation

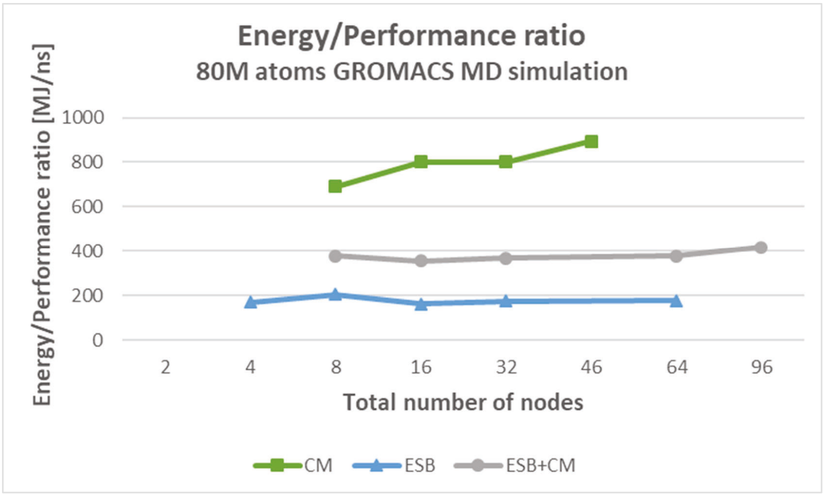

Figure 3.25: Energy vs. Performance ratio of 80M atoms GROMACS MD simulation

3.7.2 Performance of the newly implemented algorithms

Direct performance comparison between the existing PME method and the newly implemented multi-GPU FMM is impractical since these methods have a non-overlapping domain of application. PME method outperforms FMM for all problems where it is applicable. On the other hand, FMM enables solving MD problems with very large volumes and number of particles which PME simply cannot handle within reasonable timeframes. We expect such large problems to be solved on pre-Exascale and Exascale systems using FMM, while PME is used for ensemble simulations.

3.8 Conclusion

The DEEP-EST project provided means to further enhance the capabilities of MD software. In computer-aided drug design or life sciences on the MSA one can optimize the price vs. performance ratio by choosing the appropriate configuration of nodes for each particular task. For example, MD simulations of several thousand atoms should run on the CM, while the ESB is beneficial for MD simulations of millions of atoms. In certain cases, the Cluster-Booster configuration shows up to 30% better performance than using ESB nodes only, albite at higher energy cost. The applicability of such trade-off can be considered by the user when the time to solution is more important than the price of the solution itself.

The multi-GPU FMM developed as part of this project is to the best of our knowledge the first such implementation integrated with GROMACS, while a single-GPU version had been developed as part of the SPPEXA project[70]. This new functionality that allows the utilization of FMM on large number of GPUs opens new possibilities for GROMACS to perform very large simulations in fields like material science, polymer science, molecular biology, nanostructures and condensed systems. Results obtained for the MSA architecture show good promise that by using the newly implemented multi-GPU FMM such large simulations consisting of billions of particles may be run at reasonable performance. Future work for this implementation includes overcoming the hard limit on the size of the MD simulation and further optimization of the code both in terms of performance and capabilities. For biological systems in life science research the existing PME method already provides excellent performance on the MSA. The software and hardware work together to establish GROMACS as an even more versatile tool, applicable in a wide range of fields, strongly competing with non-European tools already existing in these areas.

In summary, the MSA employed in the project is suitable for a wide range of applications in the MD domain. Together with the modular hardware architecture, the additions implemented in the application provide extra flexibility to the end-users for selecting the optimal hardware and software configuration depending on their simulation needs.

[70] *J. Chem. Theory Comput.* 2020, 16, 11, 6938–6949