Available online at www.sciencedirect.com
ScienceDirect
SoftwareX
SoftwareX 1–2 (2015) 19–25
www.elsevier.com/locate/softx

GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers

Mark James Abraham[a,*], Teemu Murtola[d], Roland Schulz[b,c], Szilárd Páll[a], Jeremy C. Smith[b,c], Berk Hess[a], Erik Lindahl[a,d]

[a] *Theoretical Biophysics, Science for Life Laboratory, KTH Royal Institute of Technology, 17121 Solna, Sweden*
[b] *Oak Ridge National Laboratory, 1 Bethel Valley Road, Oak Ridge, TN 37831, United States*
[c] *Department of Biochemistry and Cellular and Molecular Biology, University of Tennessee, M407 Walters Life Sciences, 1414 Cumberland Avenue, Knoxville, TN 37996, United States*
[d] *Center for Biomembrane Research, Department of Biochemistry & Biophysics, Stockholm University, SE-10691 Stockholm, Sweden*

Received 23 February 2015; received in revised form 15 June 2015; accepted 25 June 2015

Abstract

GROMACS is one of the most widely used open-source and free software codes in chemistry, used primarily for dynamical simulations of biomolecules. It provides a rich set of calculation types, preparation and analysis tools. Several advanced techniques for free-energy calculations are supported. In version 5, it reaches new performance heights, through several new and enhanced parallelization algorithms. These work on every level; SIMD registers inside cores, multithreading, heterogeneous CPU–GPU acceleration, state-of-the-art 3D domain decomposition, and ensemble-level parallelization through built-in replica exchange and the separate Copernicus framework. The latest best-in-class compressed trajectory storage format is supported.

Keywords: Molecular dynamics; GPU; SIMD; Free energy

Code metadata

Current code version	5.0.5
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-15-00003
Legal Code License	LGPL v2.1
Code versioning system used	git
Software code languages, tools, and services used	C, C++, CUDA, MPI, OpenMP, CMake
Compilation requirements & dependencies	ANSI C89 and C++98; Unix, Linux, MacOS, Windows
Developer documentation	http://www.gromacs.org/Developer_Zone
User and developer support	http://www.gromacs.org/Support
Bug tracker	http://redmine.gromacs.org

1. Motivation and significance

Molecular dynamics (MD) has greatly expanded the scope of chemistry and several other fields by providing spatial and temporal resolution not available in experiments. Simulations have become more accurate with better force fields, they easily sample molecular motions on the μs scale, and ensemble techniques make it possible to study millisecond scale processes such as protein folding. A typical MD user chooses an initial molecular configuration, describes the atomic interactions and model physics, runs a simulation, and makes observations

* Corresponding author.
E-mail address: mjab@kth.se (M.J. Abraham).

http://dx.doi.org/10.1016/j.softx.2015.06.001

from the trajectory. Such simulations evaluate the millions of interactions of particles for billions of time steps, which can require extraordinary amounts of computational hardware and time—the scientific quality of the result is often proportional to the amount of sampling. The huge application potential has led to implementations of MD in many software packages, including GROMACS [1], AMBER [2], NAMD [3], CHARMM [4], LAMMPS [5], and Desmond [6]. The commoditization of advanced computational techniques by these packages is an important reason for the wide adoption of MD today.

In addition to the thousands of publications using GROMACS every year, one of the most exciting parts of free software is how other people use in ways not anticipated. GROMACS has long been deployed in the Folding@Home distributed computing project [7], and it is frequently used for metadynamics together with PLUMED [8]. Coarse-grained force fields such as MARTINI [9] use the GROMACS infrastructure to implement mesoscale physics models that access otherwise impossible scales of time and distance. Databases of topology file inputs and associated thermochemical results have begun to appear [10,11], and several online services can produce coordinates, parameters and topologies for GROMACS simulations [12,13]. Extending and reusing parts or all of GROMACS code is explicitly encouraged. The free license permits commercial use.

Many implementations (including ours) provide high performance when using large numbers of processors on supercomputers, but a key focus for the development of GROMACS is the fundamental assumption from economic science that *resources are scarce*: No matter how many cores are available, minimizing resource usage makes it possible to run more simulations, e.g. through ensemble methods. GROMACS aims to provide the highest possible absolute performance and efficiency on any hardware, so that both the *maximum achievable* and *real world* throughput is high, to make best use of scarce resources. The package runs fast on every single architecture present in the Top500 supercomputer list, as well as on embedded systems and everyday laptop computers.

In contrast to many other computational challenges, applications in MD typically have an intrinsically fixed problem size. When studying a protein system with 30,000 atoms, it is not relevant that a virus comprising 10 million atoms would scale better. Therefore, weak scaling performance is typically not of primary concern. However, it is critically important to reduce the amount of computer time per unit simulation through optimization, or by improving strong scaling. This improves "time to solution", and also the efficiency, measured as the science performed per amount of hardware or power consumed. While some applications need long individual trajectories, there are also many scientific questions that can be answered by using several trajectories, and for these the overall efficiency will be higher by executing independent simulations in parallel.

Strong scaling of MD is a very difficult software engineering challenge that requires synchronization of computation, coordination, and communication phases down to 100 μs for hundreds of thousands of cores. Hardware advances have been breathtaking, but re-engineering the software to use the new capabilities has been very challenging, and even forces us to reconsider some of the most fundamental MD concepts including the neighbor lists used to track spatial interactions.

This paper will briefly describe historical properties of GROMACS, and then report on recent improvements in GROMACS 4.6 and 5. The source code, as well as a large amount of introductory, tutorial, installation, usage, reference and developer documentation is available from http://www.gromacs.org.

2. Simulation capabilities

Simulations with leap-frog Verlet, velocity Verlet, Brownian and stochastic dynamics are supported, as well as calculations that do energy minimization, normal-mode analysis and simulated annealing. Several techniques are available for regulating temperature and/or pressure. Both SHAKE [14] and P-LINCS [15,16] are available for enforcing holonomic constraints, and the latter can be combined with virtual interaction sites [17] to eliminate enough fast degrees of freedom to allow 5 fs time steps. All widely used molecular mechanics force fields can be used, and 15 flavors of AMBER, CHARMM, GROMOS and OPLS are validated and included. Several community-supported force fields are also available. Non-standard functional forms are supported through user tables.

Simulations may employ several kinds of geometric restraints, use explicit or implicit solvent, and can be atomistic or coarse-grained. `mdrun` can run multiple simulations as part of the same executable, which permits generalized ensemble methods [18] such as replica-exchange [19,20]. Non-equilibrium methods, such as pulling and umbrella sampling, are available, as well as highly powerful alchemical free-energy transformations, and essential dynamics [21]. Many popular simulation file formats can be read natively, or via a VMD plug-in [22].

3. Software description

GROMACS has grown into a very large software project with almost two million lines of code. For a detailed description of the historical development and many algorithms included in the engine, we refer the interested reader to the previous papers published [1,23–28]. For developers, one of the most important changes is that GROMACS 5 is the first release that has moved to C++. While many parts of the code remain in C and it will take a few years to complete the transition, this has led to improvements in code modularity, handling of memory and errors, and enabled much better Doxygen developer documentation and unit testing.

GROMACS 5 works within an elaborate multi-level parallelism (Fig. 1) that distributes computational work across ensembles of simulations, multiple program paths and domains within simulations, multiple cores working on each domain, exploiting instruction-level parallelism across those cores. This design is able to make effective use of all of the available resources when running typical PME simulations on typical hardware. However, the design works less well if the hardware is too unbalanced; GROMACS 5 performance will typically be good

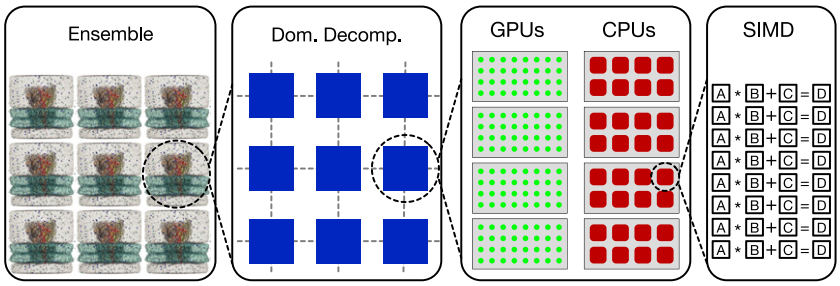

Fig. 1. Multi-level parallelism in GROMACS. SIMD registers are used to parallelize cluster interaction kernels or bonded interactions in each core, and OpenMP multithreading is used for parallelism inside spatial domains while nonbonded interactions are handled by GPUs or other accelerators. MPI with load balancing is used to decompose a single simulation into domains over many nodes in a cluster, and ensemble approaches are used to parallelize with loosely coupled simulations. High performance requires that software targets each level explicitly.

with comparable expenditure on CPU and accelerator, and as many CPU sockets as accelerators.

We expect ensemble-level parallelism to play an increasingly important role in MD algorithm development. While the code scales *down* to a few tens of atoms per core (when only using CPUs), there will always be practical limits on the degree of parallelism achievable. A typical 150,000-atom system has about thirty million particle–particle interactions per MD step, which will not scale to a million-core system because communication and book-keeping costs will dominate. The Copernicus ensemble framework has been developed alongside GROMACS 5, to serve this need and scale to tens of thousands of simulations [29]. It currently supports free-energy calculations, Markov state modeling, and the string method using swarms [30] (http://copernicus.gromacs.org).

Within a simulation, using parallel computers requires splitting the problem into independent units of work. In GROMACS, an "eighth shell" spatial domain decomposition [31,32] efficiently partitions the simulation in a way that preserves locality of reference within each domain. This data parallelization maps each domain to an MPI rank, each of which can in practice have access to various kinds of hardware. Internally, all systems are described with triclinic unit cells, which makes complex geometries such as rhombic dodecahedron, truncated octahedron or hexagonal boxes supported in all parts of the code. This can improve performance up to 40% compared to the same water thickness around a solute in a rectangular box (Fig. 2). Dynamic load balancing between domains is performed in all three dimensions in triclinic geometry; this is critical for high performance. Fig. 2 shows how the larger computational load due to torsions and angles in the protein compared to water leads to significant differences in domain size in the upper left part.

Long-range electrostatics is handled by the particle-mesh Ewald (PME) method [33] by using dedicated MPI ranks for the lattice summation and a two-dimensional pencil decomposition [1] for the required 3D-FFT.

Historically, GROMACS has made use of MPI for domain-level parallel decomposition across nodes, and later CPU cores too, and supplied hand-tuned assembly kernels to access SIMD (single instruction, multiple data) units where available. However, the run-time overheads of the former and the development-time cost of the latter were not sustainable, and there was also the need to incorporate accelerators (such

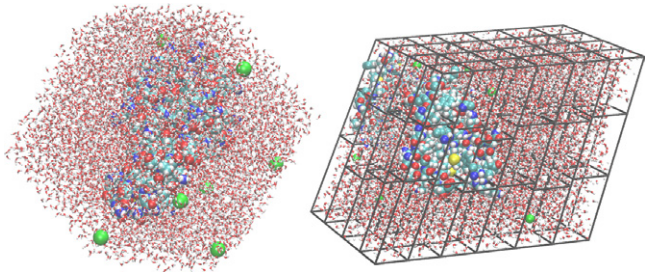

Fig. 2. *Left:* The protein lysozyme (24,119 atoms) in a compact unit cell representation corresponding to a rhombic dodecahedron with close-to-spherical shape. *Right:* Internally, this is represented as a triclinic cell and a load-balanced staggered $6 \times 4 \times 3$ domain decomposition grid on 72 MPI ranks. The PME lattice sum is calculated on a uniform grid of 6×4 MPI ranks (not shown).

as GPUs) into the parallelization strategy. GROMACS 4.6 introduced a native heterogeneous parallelization setup using both CPUs and GPUs. There are two important reasons for still including the CPU: First, the advanced domain decomposition and load balancing would be very difficult to implement efficiently on GPUs (which would hurt scaling). Second, we see it as a huge advantage that *all* algorithms are available in all simulations, even the esoteric or new ones not yet ported to GPUs, and the heterogeneous acceleration makes it possible to completely hide the hardware from the user. There is only a single binary, and by default it will use all available hardware fairly efficiently, with many run-time options available to tune performance.

To make this possible, a new algorithm for evaluating short-ranged non-bonded interaction was implemented, based on Verlet lists with automatic buffering [34]. This recasts the traditional Verlet algorithm to suit modern computer hardware, which permits highly efficient offload of short-ranged work on SIMT-based (simultaneous multithreading) GPUs, as well as efficient SIMD-based CPU implementations. This works well, since the essential requirements for data locality and reuse are similar on both kinds of hardware. This is an important architectural advance, since the same code base and algorithms can be used for all hardware. The key innovation was to transform the standard formulation of the Verlet algorithm that uses lists of particle–particle pairs into lists of interacting small clusters of nearby particles, and to choose the sizes of those clusters at compile time to match

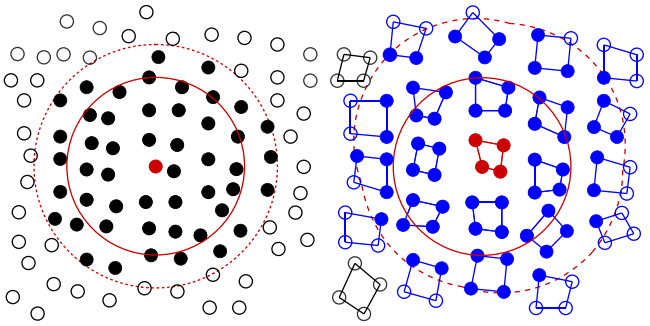

Fig. 3. *Left:* A classical Verlet implementation treats all j-particles within an interaction radius (red) of the central red i-particle, and adds a buffer, also called "skin" (dashed red). Particles outside the buffer (unfilled) are omitted. *Right:* The $M \times N$ scheme builds lists of clusters of N particles (blue), where at least one particle in each cluster is within the buffered interaction radius of any particle in the central red cluster. This envelope has an irregular shape (dashed red), and has an implicit additional buffer (unfilled blue circles) from those particles in clusters where only some particles are within the nominal buffer range. Actual interactions are based on particle distances (red circle, only one shown). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the characteristics of the SIMT or SIMD target hardware. This means there is no requirement for the compiler to recognize the opportunity for vectorization; it is intrinsic to the algorithm and its implementation. Additionally, the cluster sizes are easily adjustable parameters allowing to target new hardware with relatively low effort. Fig. 3 shows how the Verlet scheme recasts the idea of a particle-based pair list into a list of interacting clusters. Fig. 4 illustrates the flow of data in kernels executing on processors of different SIMD widths or GPUs.

Unlike the old "group" scheme, there is no need for special kernels optimized for common molecules such as water. The searching can schedule kernels that will evaluate van der Waals interactions only on the first half of atoms in a given cluster; this runs faster on domains where only some atoms have such interactions, which includes most water models in current use. Naturally, if whole clusters do not have van der Waals or Coulomb interactions, their interactions are evaluated by kernels that skip the corresponding computation entirely. Branches are unavoidable in short-ranged MD kernels, as the model physics permits only interactions within a certain distance to contribute. Implementing such code is most efficient when using selection or branch instructions that produce null results when the interaction should not contribute. This is also useful for other kinds of interaction exclusions used in MD.

The maturation of SIMD intrinsics in compilers made this possible to implement in a new higher-level fashion that retains the performance of the previous raw hand-tuned assembly. To achieve this, we have implemented a SIMD abstraction module that permits us to develop CPU non-bonded kernels in a way that is nearly agnostic about the SIMD width and feature set of the hardware upon which they will run. In particular, we have designed an extensive new internal SIMD math library in both single and double precision that completely avoids both table lookups and integer instructions (which are not available for all SIMD instruction sets). This means that porting to new CPU architectures is a straightforward job of implementing the interface of the SIMD module using the intrinsics suitable for the new CPU, and the old non-bonded kernels can use them correctly. Further, several other modules in GROMACS now use the same SIMD layer and derive the same benefits for performance portability. Crucially, this has reduced the total size of the nonbonded kernels to only a few hundred lines of C, while simultaneously supporting many more SIMD

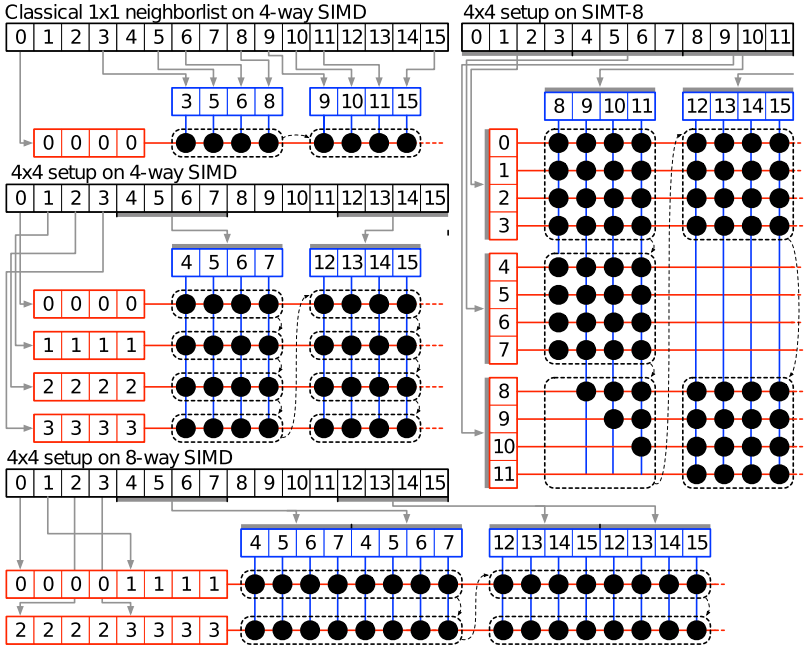

Fig. 4. Illustration of the classical 1×1 neighborlist and the 4×4 non-bonded cluster algorithm setups on processors with different SIMD widths. All numbers are particle indices in a pair list, black dots are interaction calculations, and gray arrows indicate loads. The content of SIMD registers for i- and j-particles (cf. Fig. 3) are shown in red and blue, respectively. Dashed black lines show the computation units, with dotted black arrows indicating their order of execution. The 4×4 setup calculates 4 times as many interactions per memory operation. Unlike the 1×1 setup, the 4×4 setup does not require data shuffling in registers.

instruction sets. GROMACS performance is now more sensitive to the quality of the compiler, which reflects wide-ranging improvements in both. The proliferation of new kinds of SIMD support in CPUs means that GROMACS users have more need to take care to use binaries that match the hardware capabilities.

In particular for SIMD and GPU acceleration, GROMACS makes extensive use of strength-reduction algorithms to permit use of single precision, including a single-sum implementation of the virial calculation [35]. Some molecular simulation packages always compute in double precision; this is available in GROMACS for the few kinds of simulations that require it, but, by default, a *mixed precision* mode is used, in which a few, critical, reductions are performed in double precision. Other high-performance implementations [2] use mixed precision to a larger extent.

The offload model for acceleration creates the need for large groups of atoms in the same spatial region to be treated in the same neighbor search. This conflicts with the former GROMACS model of mapping each CPU core to an MPI rank, and thus a separate domain of atoms close in space. Typically, a CPU has many more cores than it has accelerators, the inefficiency of scheduling separate work for each domain on the accelerator was high, and the existing limitations on the minimum sizes of domains was also problematic. To alleviate this, OpenMP-based multi-threading support was added to GROMACS to permit multiple CPU cores to work on a single domain of atoms. This allows for domains to be larger and thus the overall efficiency to be greatly improved. The resulting OpenMP parallelism is also useful when running on CPU-only hardware, which extends the strong-scaling limit with hybrid MPI/OpenMP, but adds complexity for the user.

The PME algorithm commonly used for molecular simulations is able to shift workload between the short- and long-ranged components with moderate efficiency, while preserving the quality of the model physics. This permits GROMACS 5 to automatically balance the workload for optimal performance. This is particularly useful for the offload model implemented in GROMACS 5 because best throughput is typically obtained when few resources lie idle. Further, when using multiple nodes for a single simulation, the long-ranged component of the PME calculation needs to do global communication for the 3D FFT. This partly drives our choice of which work to offload; doing PME work on a current generation accelerator in a simulation across multiple nodes would accrue latency from data transfers both across the network and from host to device and this would eliminate any performance gain.

One major weakness of the current accelerator-offload implementation in GROMACS is that accelerators are idle once the forces are computed and transferred back to the CPU. Typical schemes for integrating the forces to update the positions often need to enforce holonomic constraints on degrees of freedom such as bond lengths, and such implementations normally feature either iteration or inter-rank communication, which does not suit an offload model of accelerator usage. Overcoming such limitations is a key target for future improvements.

4. Software functionalities

GROMACS is free software distributed under LGPLv2.1, which even allows linking into commercial applications. It requires only standards-conforming C99 and C++98 compilers, and CMake version 2.8.8. Various external libraries are either bundled for convenience, or can be detected (e.g. MKL) or even downloaded automatically (e.g. FFTW) by CMake. Portability is assured via extensive automated continuous integration testing using Jenkins, deployed on Windows, MacOS and Linux, using multiple versions of compilers including those from Intel, GNU, Microsoft, LLVM and IBM. GROMACS supports NVIDIA GPU architectures with compute capabilities 2.0 and later, and the new SIMD module provides native support for a total of 13 different architectures including all x86 flavors (from SSE2 through Xeon Phi, AVX2 and the still unreleased AVX-512F/ER), PowerPC A2 (BlueGene/Q), Sparc VIIIfx (K computer), ARM Neon, IBM VMX (Power7) and VSX (Power8). The latest version can even run inside a browser supporting Google Native Client.

Every single commit during GROMACS development is subject to mandatory code review and automatic regression tests, unit tests and static code analysis before it is added to the public git repository (available via

`git clone git://git.gromacs.org/gromacs.git`).

While the released code is tested on an even larger set of architectures, this makes even the rapidly moving development branch uniquely stable.

4.1. A parallel analysis framework

A new C++ framework for developing GROMACS analysis tools has been introduced, which makes it easy to write new tools that require only a simple loop over all trajectory frames. The framework also provides reusable components for grid-based neighbor searching and common data processing tasks like histograms. Some tools for computing basic geometric properties (distances and angles), as well as surface area calculation, have been converted to the new framework, though much work remains to achieve the full benefits of the new scheme. Future development also aims to support analyzing single trajectory frames in parallel, and Python bindings.

4.2. New simulation features

Just as PME has eliminated cutoff artifacts for electrostatics, there has been increasing attention to cutoff problems and van der Waals interactions. While dispersion corrections alleviate some issues, the fundamental problem is that complex systems such as membranes are neither homogeneous nor isotropic. GROMACS 5 includes a new, very accurate, Lennard-Jones PME implementation [36] whose implementation is only 10%–20% more expensive than short cutoffs in GROMACS, and to the best of our knowledge about an order of magnitude faster than any other alternative. It works for both geometric and Lorentz–Berthelot combination rules, and should enable

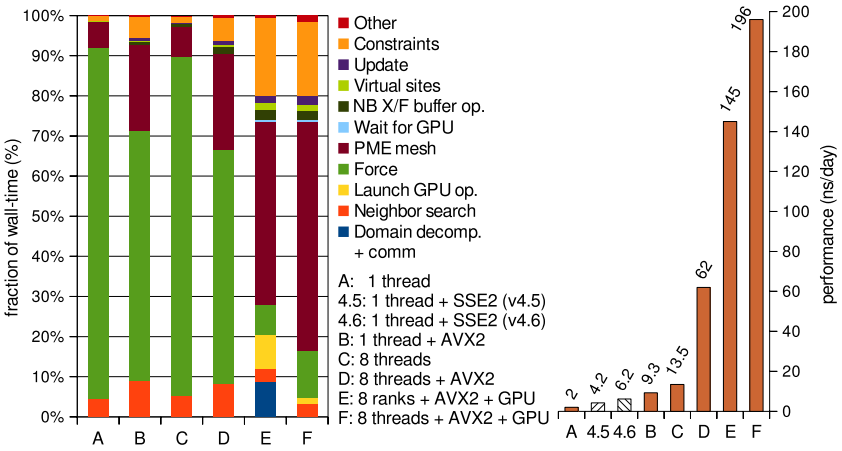

Fig. 5. The anatomy of GROMACS performance. *Left:* The reference setup (A) spends the majority of wall-time in short-range force evaluations, but as SIMD (B), OpenMP (C, D), and GPU (E, F) acceleration are enabled this drops tremendously even on a workstation. With GPUs (E, F), the CPU computes a relatively small amount of bonded interactions. Relying on multi-threading (F) when using GPUs is more efficient than SPMD parallelization with domain-decomposition (E) even though multi-threading in cache intensive code like PME is challenging. *Right:* The absolute performance in GROMACS 5.0 is greatly improved over earlier versions, and more than an order of magnitude higher with accelerators.

much more accurate membrane simulations, free energies, and improved force-field parameterization.

Other new features include Andersen-style thermostats, the Adaptive Resolution Sampling scheme [37] for multi-scale models, Hamiltonian replica exchange, simulated tempering and expanded-ensemble methods [38], rotation of groups with the non-equilibrium pulling module [39], a new computational electrophysiology module [40] that can swap molecules from one side of a membrane to the other, and support for the Interactive Molecular Dynamics [41] protocol to view and manipulate ongoing simulations. The high-quality "counter-based" parallel random number generator Random123 [42] is now used. New bonded interactions were introduced for coarse-grained simulations [43]. Flat-bottomed position restraints were added to avoid perturbing models unnecessarily.

GROMACS 5 also comes with a new highly flexible and efficient compressed file format—TNG [44]. This improves on the previous best-in-class XTC trajectory compression by further exploiting domain knowledge and multi-frame compression, it adds features such as containers for general simulation data, digital signatures, and provides a library to which tool developers may link.

5. Performance & scaling

GROMACS can scale to the largest machines in the world when using gigantic systems, and detailed benchmarks are available on the website. For this work, we want to illustrate the efficiency with more challenging heterogeneous benchmarks used in recent studies: first a very small voltage sensor (VSD) embedded in a united-atom lipid bilayer in a hexagonal box (45,700 atoms) [45], and second a complete ion channel (GluCl) embedded in a larger united-atom bilayer (142,000 atoms) [46]. All simulations use PME and initial cut-offs of 1.0 nm. All bonds were constrained with the LINCS [16] algorithm, and the VSD uses virtual interaction sites constructed

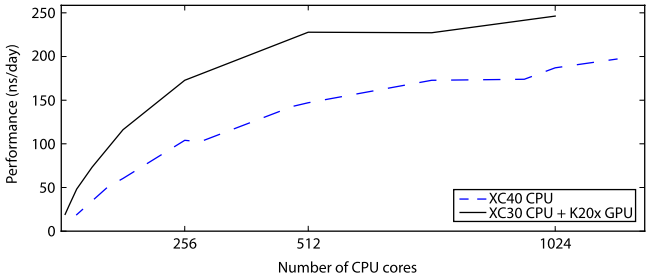

Fig. 6. Absolute scaling performance on a GluCl ion channel of 142,000 atoms, 2.5 fs time steps without virtual sites, with PME and initial cutoffs 1.0 nm, running on Beskow (32 Haswell CPU cores per node, Cray XC40) and Piz Daint (8 Sandy Bridge CPU cores + Tesla K20X per node, Cray XC30).

every step to extend the time step to 5 fs. A stochastic velocity-rescaling thermostat was used [47]. Fig. 5 shows how the fraction of CPU cycles spent on force evaluation in the VSD system drops dramatically when adding SIMD and GPUs. Case "E" reflects the problem with multiple MPI ranks on the CPU and effectively time-sharing the GPU, which introduces decomposition overhead. With GPUs, there is only a small component left for bonded forces; the CPU primarily evaluates the PME mesh part. Absolute performance is much higher with acceleration (explaining the larger fractions for constraints and PME), as illustrated in the right panel, which shows parallelization benefits in different GROMACS versions on a single-socket 8-core Core-i7 5960X desktop with one NVIDIA GTX980 GPU. With SIMD, GPU and OpenMP acceleration, the desktop achieves close to 200 ns/day for the VSD. Fig. 6 shows absolute performance for the larger GluCl system on CPU-only and GPU-equipped Cray clusters. Despite the much faster CPUs with AVX2 support on the XC40, the older XC30 nodes paired with K20x GPUs beat it handily. With similar CPUs and K40 GPUs, we expect accelerated clusters to deliver about 3× the performance of CPU-only ones.

6. Conclusions

The recent efforts to maximize single-core and single-node performance show benefits at all levels, which makes it even more impressive that the relative scaling has also improved substantially. The enhancements described above boost user throughput on all kinds and quantities of hardware, but they were focused on mature technologies. There are many other approaches open for future performance improvements to GROMACS. In particular, an implementation that expresses the algorithm in fine-grained tasks that can be preempted when high-priority work is available, and automatically balanced between otherwise idle executors seems very attractive.

Acknowledgments

This work was supported by the European Research Council (258980, BH), the Swedish Research Council (2013-5901, EL) the Swedish e-Science Research Center, the ScalaLife EU infrastructure (261523), the EU FP7 CRESTA project (287703), Intel Corporation and the ORNL Adaptive Biosystems Imaging project funded by the Biological and Environmental Research of the Office of Science of the U.S. Department of Energy (RS, JCS). Computational resources were provided by the Swedish National Infrastructure for computing (2014/1-30 & 2014/11-33), and the Swiss National Supercomputing Centre CSCS (g43). GROMACS would not be possible without a large and loyal team working on code review, triage, and contributing code and fixes. Thank you!

References

[1] Pronk S, Páll S, Schulz R, Larsson P, Bjelkmar P, Apostolov R, et al. Bioinf 2013;29:845–54.
[2] Case DA, Cheatham TE, Darden T, Gohlke H, Luo R, Merz KM, et al. J Comput Chem 2005;26:1668–88.
[3] Phillips JC, Braun R, Wang W, Gumbart J, Tajkhorshid E, Villa E, et al. J Comput Chem 2005;26:1781–802.
[4] Brooks BR, Bruccoleri RE, Olafson BD, States DJ, Swaminathan S, Karplus M. J Comput Chem 1983;4:187–217.
[5] Plimpton S. J Comp Phys 1995;117:1–19.
[6] Bowers KJ, Chow E, Xu H, Dror RO, Eastwood MP, Gregersen BA, et al., Proceedings of the ACM/IEEE conference on supercomputing; 2006. p. 0–7695–2700–0/06.
[7] Shirts M, Pande VS. Science 2000;290:1903–4.
[8] Tribello GA, Bonomi M, Branduardi D, Camilloni C, Bussi G. Comput Phys Commun 2014;185:604–13.
[9] Marrink SJ, Risselada HJ, Yefimov S, Tieleman DP, de Vries AH. J Phys Chem B 2007;111:7812–24.
[10] Caleman C, van Maaren PJ, Hong M, Hub JS, Costa LT, van der Spoel D. J Chem Theory Comput 2012;8:61–74.
[11] van der Spoel D, van Maaren PJ, Caleman C. Bioinf 2012;28:752–3.
[12] Zoete V, Cuendet MA, Grosdidier A, Michielin O. J Comp Chem 2011;32:2359–68.
[13] Malde AK, Zuo L, Breeze M, Stroet M, Poger D, Nair PC, et al. J Chem Theory Comput 2011;7:4026–37.
[14] Ryckaert JP, Ciccotti G, Berendsen HJ. J Comput Phys 1977;23:327–41.
[15] Hess B, Bekker H, Berendsen HJ, Fraaije JG. J Comput Chem 1997;18:1463–72.
[16] Hess B. J Chem Theory Comput 2008;4:116–22.
[17] Berendsen HJC, van Gunsteren WF. In: AJB, et al., editors. Molecular liquids-dynamics and interactions, NATO ASI C 135. Dordrecht (The Netherlands): Reidel; 1984. p. 475–500.
[18] Mitsutake A, Sugita Y, Okamoto Y. Biopolymers 2001;60:96–123.
[19] Hansmann UH, Okamoto Y. J Comput Chem 1997;18:920–33.
[20] Sugita Y, Okamoto Y. Chem Phys Lett 1999;314:141–51.
[21] Amadei A, Lisnsen A, Berendsen H. Proteins: Struct, Func, Genet 1993;17:412–25.
[22] Humphrey W, Dalke A, Schulten K. J Mol Graph 1996;14:33–8.
[23] Páll S, Abraham MJ, Kutzner C, Hess B, Lindahl E. In: Markidis S, Laure E, editors. Solving software challenges for exascale. Lecture notes in computer science. Springer International Publishing; 2015. p. 3–27.
[24] Bekker H, Berendsen HJC, Dijkstra EJ, Achterop S, van Drunen R, van der Spoel D, et al. In: de Groot RA, Nadrchal J, editors. Physics computing, vol. 92. Singapore: World Scientific; 1993. p. 252–6.
[25] Berendsen HJ, van der Spoel D, van Drunen R. Comput Phys Commun 1995;91:43–56.
[26] Lindahl E, Hess B, van der Spoel D. J Mol Mod 2001;7:306–17.
[27] van der Spoel D, Lindahl E, Hess B, Groenhof G, Mark AE, Berendsen HJ. J Comput Chem 2005;26:1701–18.
[28] Hess B, Kutzner C, van der Spoel D, Lindahl E. J Chem Theory Comput 2008;4:435–47.
[29] Pronk S, Pouya I, Lundborg M, Rotskoff G, Wesén B, Kasson PM, et al. J Chem Theory Comput 2015;11:2600–8.
[30] Pan AC, Roux B. J Chem Phys 2008;129:064107.
[31] Bowers KJ, Dror RO, Shaw DE. J Phys: Conf Ser 2005;16:300.
[32] Bowers KJ, Dror RO, Shaw DE. J Comput Phys 2007;221:303–29.
[33] Darden T, York D, Pedersen L. J Chem Phys 1993;98:10089–92.
[34] Páll S, Hess B. Comp Phys Comm 2013;184:2641–50.
[35] Bekker H, Berendsen HJC, Dijkstra EJ, Achterop S, Drunen RV, Spoel DVD, et al. In: de Groot RA, Nadrchal J, editors. Physics computing, vol. 92. Singapore: World Scientific; 1993. p. 257–61.
[36] Wennberg CL, Murtola T, Hess B, Lindahl E. J Chem Theory Comput 2013;9:3527–37.
[37] Fritsch S, Junghans C, Kremer K. J Chem Theory Comput 2012;8:398–403.
[38] Lyubartsev AP, Martsinovski AA, Shevkunov SV, Vorontsov-Velyaminov PN. J Chem Phys 1992;96:1776–83.
[39] Kutzner C, Czub J, Grubmüller H. J Chem Theory Comput 2011;7:1381–93.
[40] Kutzner C, Grubmüller H, de Groot BL, Zachariae U. Biophys J 2011;101:809–17.
[41] Stone JE, Gullingsrud J, Schulten K. Proceedings of the 2001 symposium on interactive 3D graphics. New York (NY, USA): ACM; 2001. p. 191–4.
[42] Salmon JK, Moraes MA, Dror RO, Shaw DE. Proceedings of 2011 international conference for high performance computing, networking, storage and analysis. New York (NY, USA): ACM; 2011. p. 16:1–:12.
[43] Bulacu M, Goga N, Zhao W, Rossi G, Monticelli L, Periole X, et al. J Chem Phys 2005;123:3282–892.
[44] Lundborg M, Apostolov R, Spångberg D, Gärdenäs A, van der Spoel D, Lindahl E. J Comp Chem 2014;35:260–9.
[45] Henrion U, Renhorn J, Börjesson SI, Nelson EM, Schwaiger CS, Bjelkmar P, et al. Proc Natl Acad Sci 2012;109:8552–7.
[46] Yoluk O, Brömstrup T, Bertaccini EJ, Trudell JR, Lindahl E. Biophys J 2013;105:640–7.
[47] Bussi G, Donadio D, Parrinello M. J Chem Phys 2007;126:014101.