Article

GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation

Berk Hess, Carsten Kutzner, David van der Spoel, and Erik Lindahl

J. Chem. Theory Comput., **2008**, 4 (3), 435-447• DOI: 10.1021/ct700301q • Publication Date (Web): 02 February 2008

JCTC Journal of Chemical Theory and Computation

GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation

Berk Hess*

Max-Planck Institute for Polymer Research, Ackermannweg 10, D-55128 Mainz, Germany

Carsten Kutzner

Department of Theoretical and Computational Biophysics, Max-Planck-Institute of Biophysical Chemistry, Am Fassberg 11, D-37077 Göttingen, Germany

David van der Spoel

Department of Cell and Molecular Biology, Uppsala University, Husargatan 3, Box 596, SE-75124 Uppsala, Sweden

Erik Lindahl

Stockholm Center for Biomembrane Research, Stockholm University, SE-10691 Stockholm, Sweden

Received November 7, 2007

Abstract: Molecular simulation is an extremely useful, but computationally very expensive tool for studies of chemical and biomolecular systems. Here, we present a new implementation of our molecular simulation toolkit GROMACS which now both achieves extremely high performance on single processors from algorithmic optimizations and hand-coded routines and simultaneously scales very well on parallel machines. The code encompasses a minimal-communication domain decomposition algorithm, full dynamic load balancing, a state-of-the-art parallel constraint solver, and efficient virtual site algorithms that allow removal of hydrogen atom degrees of freedom to enable integration time steps up to 5 fs for atomistic simulations also in parallel. To improve the scaling properties of the common particle mesh Ewald electrostatics algorithms, we have in addition used a Multiple-Program, Multiple-Data approach, with separate node domains responsible for direct and reciprocal space interactions. Not only does this combination of algorithms enable extremely long simulations of large systems but also it provides that simulation performance on quite modest numbers of standard cluster nodes.

I. Introduction

Over the last few decades, molecular dynamics simulation has become a common tool in theoretical studies both of simple liquids and large biomolecular systems such as proteins or DNA in realistic solvent environments. The computational complexity of this type of calculations has historically been extremely high, and much research has therefore focused on algorithms to achieve single simulations that are as long or large as possible. Some of the key early work was the introduction of holonomic bond length constraints[1] and rigid-body water models[2,3] to enable longer integration time steps. However, one of the most important general developments in the field was the introduction of parallel molecular simulation implementations during the late

10.1021/ct700301q CCC: $40.75
Published on Web 02/02/2008

1980s and early 1990s.[4−7] The NAMD program by the Schulten group[8] was the first to enable scaling of large molecular simulations to hundreds of processors, Duan and Kollman were able to complete the first microsecond simulation of a protein by creating a special parallel version of Amber, and more recently Fitch et al. have taken scaling to the extreme with their BlueMatter code which can use all tens of thousands of nodes on the special BlueGene hardware.[9]

On the other hand, an equally strong trend in the field has been the change of focus to statistical properties like free energy of solvation or binding of small molecules and, e.g., protein folding rates. For this class of problems (limited by sampling) the main bottleneck is single-CPU performance, since it is typically always possible to achieve perfect scaling on any cluster by starting hundreds of independent simulations with slightly different initial conditions. This has always been a central theme in GROMACS development and perhaps best showcased by its adoption in the Folding@Home project, where it is running on hundreds of thousands of independent clients.[10] GROMACS achieves exceptional single-CPU performance because of the manually tuned SSE, SSE2, and ALTIVEC force kernels, but there are also many algorithmic optimizations, for instance single-sum virials and strength-reduced algorithms to allow single-precision floating-point arithmetic in all places where it still conserves energy (which doubles memory and cache bandwidth).[11,12] In the benchmark section we show that GROMACS in single precision matches the energy conservation of a double precision package.

Unfortunately it is far from trivial to combine raw single-CPU performance and scaling, and in many cases there are inherent tradeoffs. It is for instance straightforward to constrain all bond lengths on a single CPU, but in parallel it is usually only applied to bonds involving hydrogens to avoid (iterative) communication, which in turn puts a lower limit on the possible time step.

In this paper, we present a completely reworked parallelization algorithm that has been implemented in GROMACS. However, rather than optimizing relative scaling over N CPUs we have focused on (*i*) achieving the highest possible absolute performance and (*ii*) doing so on as few processors as possible since supercomputer resources are typically scarce. A key challenge has therefore been to make sure all algorithms used to improve single-CPU performance through longer time steps such as holonomic bond constraints, replacing hydrogens with virtual interaction sites,[13] and arbitrary triclinic unit cells also work efficiently in parallel.

GROMACS was in fact set up to run in parallel on 10Mbit ethernet from the start in 1992[7] but used a particle/force decomposition that did not scale well. The single-instruction-multiple-data kernels we introduced in 2000 made the relative scaling even worse (although absolute performance improved significantly), since the fraction of remaining time spent on communication increased. A related problem was load imbalance; with particle decomposition one can frequently avoid imbalance by distributing different types of molecules uniformly over the processors. Domain decomposition, on the other hand, requires automatic load balancing to avoid deterioration of performance. This load imbalance typically occurs in three cases: The most obvious reason is an uneven distribution of particles in space, such as a system with a liquid−vapor coexistence. A second reason is imbalance due to different interaction densities. In biomolecular systems the atom density is usually nearly uniform, but when a united-atom forcefield is used hydrocarbon segments (e.g., in lipid chains) have a three times lower particle density and these particles have only Lennard-Jones interactions. This makes the computation of interactions of a slab of lipids an order of magnitude faster than a slab of water molecules. Interaction density imbalance is also an issue with all-atom force fields in GROMACS, since the program provides optimized water−water loops for standard SPC/TIP3P/TIP4P waters with Lennard-Jones interactions only on the oxygens.[2,3] (In principle it is straightforward to introduce similar optimization for the CHARMM-style modified TIP water models with Lennard-Jones interactions on the hydrogens too, but since there is no clear advantage from the extra interactions we have not yet done so.) A third reason for load imbalance is statistical fluctuation of the number of particles in a domain decomposition cell. This primarily plays a role when cells only contain a few hundred atoms.

Another major issue for simulation of large molecules such as proteins was the fact that atoms connected by constraints could not be split over processors (holonomic constraints) a problem shared with all other biomolecular simulation packages (the alternative being shorter time-steps, possible coupled with multiple-time-step integration). This issue is more acute with domain decomposition, since even small molecules in general do not reside in a single domain.

Finally, the last challenge was the nonimpressive scaling of the Particle Mesh Ewald (PME) electrostatics[14] as implemented in the previous GROMACS version. Since PME involves two 3D fast Fourier transforms (FFTs), it requires global all-to-all communication where the number of messages scale as the square of the number of nodes involved. There have been several attempts at parallelizing PME using iterative solvers instead of using FFTs. A different algorithm that reduces communication is fast multipole expansion.[15] However, presently none of these methods combine the efficiency of PME using FFTs with good scaling up to many processors.

We have addressed these four issues by devising an eighth-shell domain decomposition method coupled to a full dynamic load-balancing algorithm with a minimum amount of communication, a parallel version of the constraint algorithm LINCS that enables holonomic constraints without iterative communication, and splitting off the PME calculation to dedicated PME processors. These four key advances will be described in the next three sections, followed by a description of other new features and a set of benchmarks to illustrate both absolute performance and relative scaling.

II. Domain Decomposition

Recently, the D. E. Shaw group has performed several studies into general zonal methods[16] for parallelization of particle-based interactions. In zonal (or neutral territory) methods, forces between particles i and j are not necessarily calculated

on a processor where either of particles i or j resides. Somewhat paradoxically, such methods can be significantly more efficient than traditional domain decomposition methods since they reduce the total amount of data communicated. Two methods achieve the least communication when the domain size is not extremely small compared to the cutoff radius; these two methods were termed *eighth shell*[17] and *midpoint* methods[18] by Shaw and co-workers. In the half shell method, interactions between particle i and j are calculated in the cell where i or j resides. The minimum communication required for such a method is half of the volume of a boundary of a thickness equal to the cutoff radius. The eighth shell method improves on this by also calculating interactions between particles that reside in different communicated zones. The communicated volume of the eighth shell method is thus a subset of that of the half shell method, and it also requires less communication steps which helps reduce latency.

The basic eighth shell method was already described in 1991 by Liem et al.,[19] who implemented communication with only nearest neighbors. In GROMACS 4 we have extended this method for communication with multiple cells and staggered grids for dynamic load balancing. The Shaw group has since chosen to use the midpoint method in their Desmond code since it can take advantage of hardware where each processor has two network connections that simultaneously send and receive. After quite stimulating discussions with the Shaw group we chose not to switch to the midpoint method, primarily not only because we avoid the calculation of the midpoint, which has to be determined binary identically on multiple processors, but also because not all hardware that GROMACS will run on has two network connections. With only one network connection, a single pair of send and receive calls clearly causes less latency than two such pairs of calls.

Before going into the description of the algorithm, the concept of charge groups needs to be explained; these were originally introduced to avoid electrostatic artifacts. By grouping several partially charged atoms of a chemical group into a neutral charge group, charge−charge interactions entering and leaving the cutoff are effectively replaced by short-range dipole−dipole interactions. The location of a charge group in GROMACS is given by the (non-mass-weighted) average of the coordinates of the atoms. With the advent of the PME electrostatics method this is no longer an issue. But charge groups can also speed up the neighbor search by an order of magnitude; given a pair of water molecules for instance, we only need to determine one distance instead of nine (or sixteen for a four-site water model). This is particularly important in GROMACS since the neighbor searching is much slower than the force loops, for which we typically use tuned assembly code. Since charge groups are used as the basic unit for neighbor searching, they also need to be the basic unit for the domain decomposition. In GROMACS 4, the domains are rebuilt every time neighbor searching is performed, typically every 10 steps.

The division of the interactions among processors is illustrated in Figure 1. Consider the processor or cell that has the charge groups in zone 0 as home charge groups, i.e.,

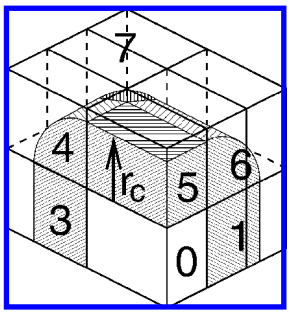

Figure 1. A nonstaggered domain decomposition grid of 3 × 2 × 2 cells. Coordinates in zones 1 to 7 are communicated to the corner cell that has its home particles in zone 0. r_c is the cutoff radius.

it performs the integration of the equations of motion for the particles in these charge groups. In the eighth shell method each cell should determine the interactions between pairs of charge groups of which, for each dimension, the minimum cell index of the two charge groups corresponds to the index of that cell. This can be accomplished by the following procedure. Cell 0 receives the coordinates of the particles in the dashed zones 1 to 7, by communication only in one direction for each dimension. When all cells dimensions are larger than the cutoff, each zone corresponds to part of a single, neighboring cell. But in general many cells can contribute to one zone. Each processor calculates the interactions between charge groups of zone 0 with zones 0 to 7, of zone 1 with zones 3 to 6, of zone 2 with zone 5, and of zone 3 with zones 5 and 6. If this procedure is applied for all processors, all pair interactions within the cutoff radius are calculated.

Interactions involving three or more atoms cannot be distributed according to the scheme described above. Bonded interactions are distributed over the processors by finding the smallest x, y, and z coordinate of the charge groups involved and assigning the interaction to the processor with the home cell where these smallest coordinates reside—note that this does not require any extra communication between the processors. This procedure works as long as the largest distance between charge groups involved in bonded interactions is not larger than the smallest cell dimension. To check if this is the case, we count the number of assigned bonded interactions during domain decomposition and compare it to the total number of bonded interactions in the system. When there are only two cells in a certain dimension and the corresponding box length is smaller than four times the cutoff distance, a cutoff criterion is required for any pair of particles involved to avoid that bonded interactions are assigned to multiple cells. Unlike the midpoint method, this procedure limits the distances involved in bonded interactions to the smallest cell dimension. For atomistic simulations this is not an issue, since distances in bonded interactions are usually smaller than 0.5 nm, leading to a limit of 10 to 20 atoms per cell, which is beyond the scaling of GROMACS 4. For coarse-grained simulations bonded distances can be larger, but because of the lower interaction density this also does not limit the scaling.

For full dynamic load balancing the boundaries between cells need to be adjusted during the simulation. For 1D

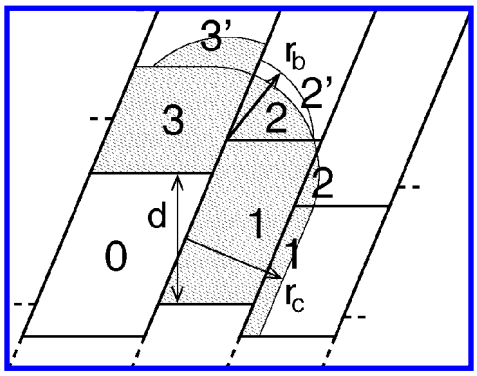

Figure 2. The zones to communicate to the processor of zone 0, see the text for details. r_c and r_b are the nonbonded and bonded cutoff radii, respectively, and d is an example of a distance between following, staggered boundaries of cells.

domain decomposition this is trivial, but for a 3D decomposition the cell boundaries in the last two dimensions need to be staggered along the first dimensions to allow for complete load balancing (see the next section for details). Figure 2 shows the communicated zones for 2D domain decomposition in the most general case, namely a triclinic unit cell with dynamic load balancing. Zones 1, 2, and 3 indicate the parts of neighboring cells that are within the nonbonded cutoff radius r_c of the home cell of zone 0. Without dynamic load balancing this is all that would need to be communicated to the processor of zone 0. With dynamic load balancing the staggering can lead to an extra volume 3′ that needs to be communicated, due to the nonbonded interactions between cells 1 and 3 which should be calculated on the processor of cell 0. For bonded interactions, zones 1 and 2 might also require expansion. To ensure that all bonded interaction between charge groups can be assigned to a processor, it is sufficient to ensure that the charge groups within a sphere with a radius r_b, the cutoff for bonded interactions, are present on at least one processor for every possible center of the sphere. In Figure 2 this means we also need to communicate volume 2′. When no bonded interactions are present between charge groups, such volumes are not communicated. For 3D domain decomposition the picture becomes quite a bit more complicated, but the procedure is analogous apart from more extensive bookkeeping. All three cases have been fully implemented for general triclinic cells. GROMACS 4 does not (yet) take full advantage of the reduction in the communication due to rounding of the zones. Currently zones are only rounded in the 'forward' directions, for example part 3′ in Figure 2 is replaced by the smallest parallelogram enclosing it.

The communication of the coordinates and charge group indices can be performed efficiently by 'pulsing' the information in one direction simultaneously for all cells one or more times. This needs to be repeated for each dimension. The number of pulses n_p in a dimension is given by the cutoff length in that direction divided by the minimum cell size. In most cases n_p will be one or two. Consider a 3D domain decomposition where we decompose in the order x, y, z; meaning that the x boundaries are aligned, the y boundaries are staggered along the x direction, and the z boundaries are staggered along the x and y directions. Each processor first sends the zone that its neighboring cell in $-z$ needs to this cell. This process is done $n_p(z)$ times. Now each processor can send the zone its neighboring cell in $-y$ needs, plus the part of the zone it received from $+z$, that is also required by the neighbor in $-y$. The last step consists of $n_p(x)$ pulses in $-x$ where (parts of) 4 zones are sent over. In this way $n_p(x) + n_p(y) + n_p(z)$ communication steps are required to communicate with $n_p(x) \times n_p(y) \times n_p(z) - 1$ processors, while no information is sent over that is not directly required by the neighboring processors. The communication of the forces happens according to the same procedure but in reversed order and direction.

Another example of a minor complication in the communication is virtual interaction sites constructed from atoms in other charge groups. This is used in some polymer (anisotropic united atom) force fields, but GROMACS can also employ virtual sites to entirely remove hydrogen vibrations and construct the hydrogens in their equilibrium positions from neighboring heavy atoms each time step.[13] Since the constructing atoms are not necessarily interacting on the same node, we have to track the virtual site coordinate dependencies separately to make sure they are both available for construction and that forces are properly communicated back. The communication for virtual sites is also performed with pulses but now in both directions. Here only one pulse per dimension is required, since the distances involved in the construction of virtual sites are at most two bond lengths.

III. Dynamic Load Balancing

Calculating the forces is by far the most time-consuming part in MD simulations. In GROMACS, the force calculation is preceded by the coordinate communication and followed by the force communication. We can therefore balance the load by determining the time spent in the force routines on each processor and then adjusting the volume of every cell in the appropriate direction. The timings are determined using inline assembly hardware cycle counters and supported for virtually all modern processor architectures. For a 3D decomposition with order x, y, z the load balancing algorithm works as follows: First the timings are accumulated in the z direction to the processor of cell $z = 0$, independently for each x and y row. The processor of $z = 0$ sums these timings and sends the sum to the processor of $y = 0$. This processor sums the timings again and sends the sum to the processor of $x = 0$. This processor can now shift the x boundaries and send these to the $y = 0$ processors. They can then determine the y boundaries, send the x and y boundaries to the $z = 0$ processors, which can then determine z boundaries, and send all boundaries to the processors along their z row. With this procedure only the necessary information is sent to the processors that need it and global communication is avoided.

As mentioned in the Introduction, load imbalance can come from several sources. One needs to move boundaries in a conservative fashion in order to avoid oscillations and instabilities, which could for instance occur due to statistical fluctuations in the number of particles in small cells. Empirically, we have found that scaling the relative lengths of the cells in each dimension with 0.5 times the load

imbalance, and a maximum scaling of 5%, produced efficient and stable load balancing. For large numbers of cells or inhomogeneous systems two more checks are required: A first restriction is that boundaries should not move more than halfway an adjacent cell (where instead of halfway one could also choose a different value). This prevents cells from moving so far that a charge group would move two cells in a single step. It also prevents load balancing issues when there are narrow zones of high density in the system. A second problem is that due to the staggering, cell boundaries along neighboring rows could shift to such an extent that additional cells would enter the cutoff radius. For load balanced simulations the user can set the minimum allowed cell size, and by default the nonbonded cutoff radius is used. The distance between following, staggered cell boundaries (as indicated by *d* in Figure 2) should not be smaller than this minimum allowed cell size. To ensure this, we limit the new position of each boundary to the old limit plus half the old margin. In this way we make sure that one boundary can move up and independently an adjacent staggered boundary can move down, without extra communication. The neighboring boundaries are communicated after load balancing, since they are needed to determine the zones for communication. When pressure scaling is applied, the limits are increased by 2% to allow the system to adjust at the next domain decomposition before hitting the cutoff restrictions imposed by the staggering.

In practical tests, load imbalances of a factor of 2 on several hundreds of processors were reduced to 2% after a few load balancing steps or a couple of seconds of simulation time.

IV. Parallel Holonomic Constraints

There are two strong reasons for using constraints in simulations: First, a physical reason that constraints can be considered a more faithful representation of chemical bonds in their quantum mechanical ground state than a classical harmonic potential. Second, a practical reason because rapid bond vibrations limit the time step. Removing these vibrations by constraining the bonds thus allows us to increase the time step and significantly improve absolute simulation performance. A frequently used rule-of-thumb is 1 fs without constraints, 1.4 fs with bonds to hydrogens constrained, and 2 fs when all bonds are constrained. Unfortunately, the common SHAKE[1] constraint algorithm is iterative and therefore not very suitable for parallelization—in fact, there has previously not been any efficient algorithm that could handle constraints connected over different processors due to domain decomposition. Most biomolecular packages therefore use constraints only for bonds involving hydrogens.

By default, GROMACS uses a noniterative constraints algorithm called *LINear Constraint Solver* (LINCS), which proved much easier to fully parallelize as hinted already in the original paper.[20] The details of the parallel LINCS algorithm P-LINCS are described elsewhere,[21] so we will only give a brief overview here. In the algorithm, the range of influence of coupled constraints is set by the order of the expansion for the matrix inversion. It is only necessary to communicate a subset of the old and new unconstrained coordinates between neighboring cells before applying the constraints. The atoms connected by up to "one plus the expansion order" bonds away need to be communicated. We can then constrain the local bonds plus the extra bonds. The communicated atoms will not have the final correctly constraint positions (since they interact with additional neighbors), but the local atoms will. The beauty of the algorithm is that normal molecular simulation only requires a first, linear correction and a single iterative step. In both these steps updated positions are communicated and adjustment forces calculated locally. The constraint communication can be accomplished with a single forward and backward pulse of the decomposition grid in each dimension, similar to the domain decomposition communication. The results of P-LINCS in GROMACS are binary identical to those of the single processor version.

In principle a similar method could be used to parallelize other constraint algorithms. However, apart from multiple communication steps for iterative methods such as SHAKE,[1] another problem is that one does not know a priori which atoms need to be communicated, because the number of iterations is not fixed. To our best knowledge, this is the first efficient implementation of an holonomic constraint algorithm for domain decomposition.[21]

The accuracy of the velocities of constrained particles has further been improved both for LINCS and SHAKE using a recently described procedure based on Lagrange multipliers.[22] For SETTLE[23] we have applied the slightly less accurate method of correcting the velocities with the position corrections divided by the time step. These changes significantly improve long-term energy conservation in GROMACS, in particular for single precision simulations.[21] With domain decomposition, SHAKE and SETTLE can only be used for constraints between atoms that reside in the same charge group. SETTLE is only used for water molecules though, which are usually a single charge group anyway.

The virtual interaction sites described earlier require rigid constraint constructs, and the implementation of parallel holonomic constraints was therefore critical to enable virtual sites with parallel domain decomposition. This enables the complete removal of hydrogen angle vibrations, which is normally the next fastest motion after bond length oscillations. Full rotational freedom of CH_3/NH_2/NH_3 groups is still maintained by using dummy mass sites,[13] which enables time steps as long as 5 fs. It has been shown that removing the angle vibrations involving hydrogens has a minor effect on the geometry of intraprotein hydrogen bonds and that properties such as the number of hydrogen bonds, dihedral distributions, secondary structure, and rmsd are not affected.[13] Note that simply constraining all angles involving hydrogens effectively also constrains most of the other angles in a molecule, which would affect the dynamics of molecules significantly.[24] In contrast, replacing hydrogens by virtual interaction sites does not affect the angular degrees of freedom involving heavy atoms. This hydrogen-removal procedure generates uncoupled angle constraints for hydrogens in alcohol groups. These angle constraints converge twice as slow in LINCS as normal constraints. To bring the accuracy of uncoupled angle constraints up to that of bond

constraints, the LINCS expansion order for angle constraints has been doubled (see the P-LINCS paper[21] for details). In the benchmark section we show that a time step of 4 fs does not deteriorate the energy conservation.

V. Optimizing Memory Access

The raw speed of processors in terms of executing instructions has increased exponentially with Moore's law. However, the memory access latency and bandwidth has not kept up with the instruction speed. This has been partially compensated by added fast cache memory and smart caching algorithms. But this only helps for repeated access of small blocks of memory. Random access of large amounts of memory has become relatively very expensive. In molecular dynamics simulations of fluid systems, particles diffuse over time. So even when starting out with an ordered system, after some time particles that are close in space will no longer be close in memory. This results in random memory access through the whole coordinate array during the neighbor search, force calculation, and the PME charge and force assignment. Meloni et al. have shown that spatially ordering atoms can significantly improve performance for a Lennard-Jones system.[25]

We have implemented a sorting scheme that improves upon that of Meloni et al. by ordering the charge groups according to their neighbor search cell assignment. Ordering using the neighbor search cell assignment provides the optimal memory access order of atoms during the force calculations. In this way, nearly all coordinates in memory are used along a cell row with a fixed minor index. For major indices there are some jumps, but the number of jumps is now the number of different major row indices instead of the number of charge group pairs. Effectively each part of the coordinate array needs to be read from memory to cache only once, instead M^2 times where M is the total number of charge groups divided by the number of charge groups that fit in cache. This approach requires that the charge groups are resorted at every step where neighbor searching is performed. For optimal performance with PME, the major and minor dimensions for the indexing of the neighbor search cells and the PME grid should match.

A second reason for ordering is to allow for exact rerunning of part of a simulation. Due to the domain decomposition the order of the local charge groups on each processor changes. This order affects the rounding of the least significant bit in the summation of forces. To exactly reproduce part of a simulation the local atom order should be reproducible when restarting at any point in time. To define a unique order, we sort the charge groups within each neighbor search cell according to the order in the topology. Since charge groups only move a short distance between neighbor list updates, few particles cross cell boundaries, and the sorting can be done efficiently with a linear algorithm.

Optimization of memory access becomes particularly important in combination with the assembly kernels, since the SIMD instructions are extremely fast and therefore memory access can be a significant bottleneck. To quantify this we have simulated a 2 M NaCl(aq) solution[26] using SPC/E water[27] with reaction-field and PME electrostatics. The effect of the sorting is shown in Table 1. The sorting ensures a nearly constant performance, independent of the system size. Without sorting there is a 10% performance degradation at 10^4 atoms per core and a factor of 2 at $2-3 \times 10^5$ atoms. For a Lennard-Jones system of 10^5 atoms the difference is a factor of 4. Note that sorting actually decreases the scaling efficiency with the number of processors, since for low parallelization (more atoms per processor) the absolute performance increases more than for high parallelization, but it obviously always helps absolute performance.

Table 1. Number of MD Steps per Second with and without Spatial Sorting of Charge Groups[a]

		number of atoms per core			
sorting	electrostatics	1705	8525	34100	272800
yes	reaction field	241	48.5	11.9	1.39
no	reaction field	238	44.0	9.6	0.60
yes	PME	102	22.5	5.4	0.61
no	PME	101	20.6	4.8	0.33

[a] As a function of the number of atoms per core for a 2 M NaCl(aq) solution on a 2.2 GHz AMD64 CPU.

VI. Multiple-Program, Multiple-Data PME Parallelization

The typical parallelization scheme for molecular simulation and most other codes today is Single-Program, Multiple-Data (SPMD) where all processors execute the same code but with different data. This is an obvious solution to decompose a system containing hundreds of thousands of similar particles. However, particularly for the now ubiquitous PME algorithm this approach has some drawbacks: First, the direct space interactions handled through classical cutoffs and the reciprocal space lattice summation are really independent and could be carried out in parallel rather than partitioning smaller work-units over more processors. Second, the scaling of PME is usually limited by the all-to-all communication of data during the parallel 3D FFT.[28] While the total bandwidth is constant, the number of messages and latencies grow as N^2, where N is the number of nodes over which the FFT grid is partitioned.

Apart from rewriting and tuning the parallel PME algorithm to support domain decomposition, we have addressed this problem by optionally supporting Multiple-Program, Multiple-Data (MPMD) parallelization where a subset of processors are assigned as dedicated PME processors, while the direct space interactions and integration are domain decomposed over the remaining processors. On most networks the newly added communication step between real and reciprocal space processors is more than compensated by better 3D FFT scaling when the number of nodes involved in the latter is reduced a factor of 3−4. The optimal ratio for real space to reciprocal processors is usually between 2:1 and 3:1. Good load balancing for a given ratio can be reached by moving interactions between direct and reciprocal space to ensure load balance, as long as the real space cutoff and grid cell size are adjusted by the same factor the overall accuracy remains constant.[14] In future versions of GROMACS this procedure may be automated.

We assume that the PME processor count is never higher than the number of real space processors. In general, each PME processor will receive coordinates from a list of real space peers, after which the two sets of nodes start working on their respective (separate) domains. The PME processors communicate particle coordinates internally if necessary, perform charge spreading on the local grid, and then communicate overlapping grid parts with the PME neighbors. The actual FFT/convolution/iFFT is performed the standard way but now involving much fewer nodes. After force interpolation the forces corresponding to grid overlap are communicated to PME neighbors again, after which we synchronize and send communicate all forces back to the corresponding real space processors (energy and virial terms only need to be communicated to one of the processors).

With current multicore processors and multisocket motherboards the MPMD approach is particularly advantageous. The costly part is the redistribution of the 3D FFT grid, which is done twice for the forward and twice for the backward transform. This redistribution requires simultaneous communication between all PME nodes, which occurs when the real space nodes are not communicating, and to make use of this GROMACS interleaves the PME processors with the real space processors on nodes. Thus, on a machine where two cores share a network connection, with MPMD only one PME process uses a single network connection instead of two PME processes, and therefore the communication speed for the 3D FFT is doubled. For a real space to PME processor ratio of 3:1, with four cores sharing a network connection, MPMD quadruples the communication speed for the 3D FFT, while simultaneously decreasing the number of process pairs that need to exchange FFT grid information by a factor 16.

VII. The MD Communication

Previous GROMACS versions used a ring communication topology, where half of the coordinates/forces were sent over half the ring. To be frank, the only thing to be said in favor of that is that it was simple. Figure 3 shows a flowchart of the updated communication that now relies heavily on collective and synchronized communication calls available, e.g., in MPI. Starting with the direct space domain (left), each node begins by communicating coordinates necessary to construct virtual sites and then constructs these. At the main coordinate communication stage, data are first sent to peer PME nodes that then begin their independent work. In direct space, neighboring nodes exchange coordinates according to the domain decomposition, calculate interactions, and then communicate forces. Since the PME virial is calculated in reciprocal space, we need to calculate the direct space virial before retrieving the forces from the PME nodes. Finally, the direct space nodes do integration, parallel constraints (P-LINCS), and energy summation. The reciprocal domain nodes start their work when they get updated coordinates from their peer direct space nodes and exchange data with their neighbors to achieve a clean 1D decomposition of the charge grid. After spreading the charges the overlapping parts are communicated and summed, and 3D FFT, convolution, and 3D inverse FFT are performed in parallel. Finally local forces are interpolated, communicated back to the correct PME processor, and sent back to the direct space processor it came from. Whenever possible we use collective MPI operations, e.g., to enable binary-tree summation, and pulsing operations use combined send-receive operations to fully utilize torus networks present on hardware such as IBM BlueGene or Cray XT4.

VIII. Other New Features

Previously, GROMACS only supported neighbor list updates at fixed intervals, but the use of potentials that are switched exactly to zero at some finite distance is increasing, mainly to avoid cutoff effects. To be sure that no interaction is missed, the neighbor list can be updated heuristically in GROMACS 4. The neighbor list is then updated when one or more atoms have moved a distance of more than half the buffer size from the center of geometry of the charge group they belong to, as determined at the last neighbor search (note that without charge groups this is just the position of the atom at the last neighbor search). Coordinate scaling due to pressure coupling is taken into account.

GROMACS can now also be used very efficiently for coarse-grained simulations (see benchmarks section) or many nonstandard simulations that require special interactions. User defined nonbonded interactions that can be assigned independently for each pair of charge groups were already supported, and we have now additionally implemented user defined bonds, angles, and dihedrals functions. Thus, a user now has full control over functional form as well as the parameters of all interactions. Just as for the tabulated nonbonded interactions, cubic spline interpolation is used, which provides continuous and consistent potentials and forces.

In addition to systems without periodic boundaries and with full 3D periodicity, systems with only 2D periodicity in x and y are now also supported. The 2D periodicity can be combined with one or two uniform walls at constant-z planes. The neighbor searching still uses a grid for dimensions x and y and with two walls, also in z, for optimal efficiency. The walls are represented by a potential that works only in the z-direction, which can be, e.g., 9−3, 10−4, or a user defined tabulated potential, with coefficients set individually for each atom type.

Restraining (using an umbrella potential) or constraining the center(s) of mass of a group or groups of atoms can now be done in parallel. One can restrain or constrain absolute positions or relative distances between groups. The center of mass of a group of atoms can be ill-defined in a periodic system. To determine the center of mass a reference atom is chosen for each group. The center of mass of each group relative to its reference atom is then determined, and the position of the reference atom is added to obtain the center of mass position. This provides a unique center of mass, as long as all atoms in the group are within half the smallest box dimension of the reference atom. Since there are no a priori limits on the distances between atoms in a group, global communication is required. There are two global communication steps: one to communicate the reference atom positions and one to sum the center of mass contribu-

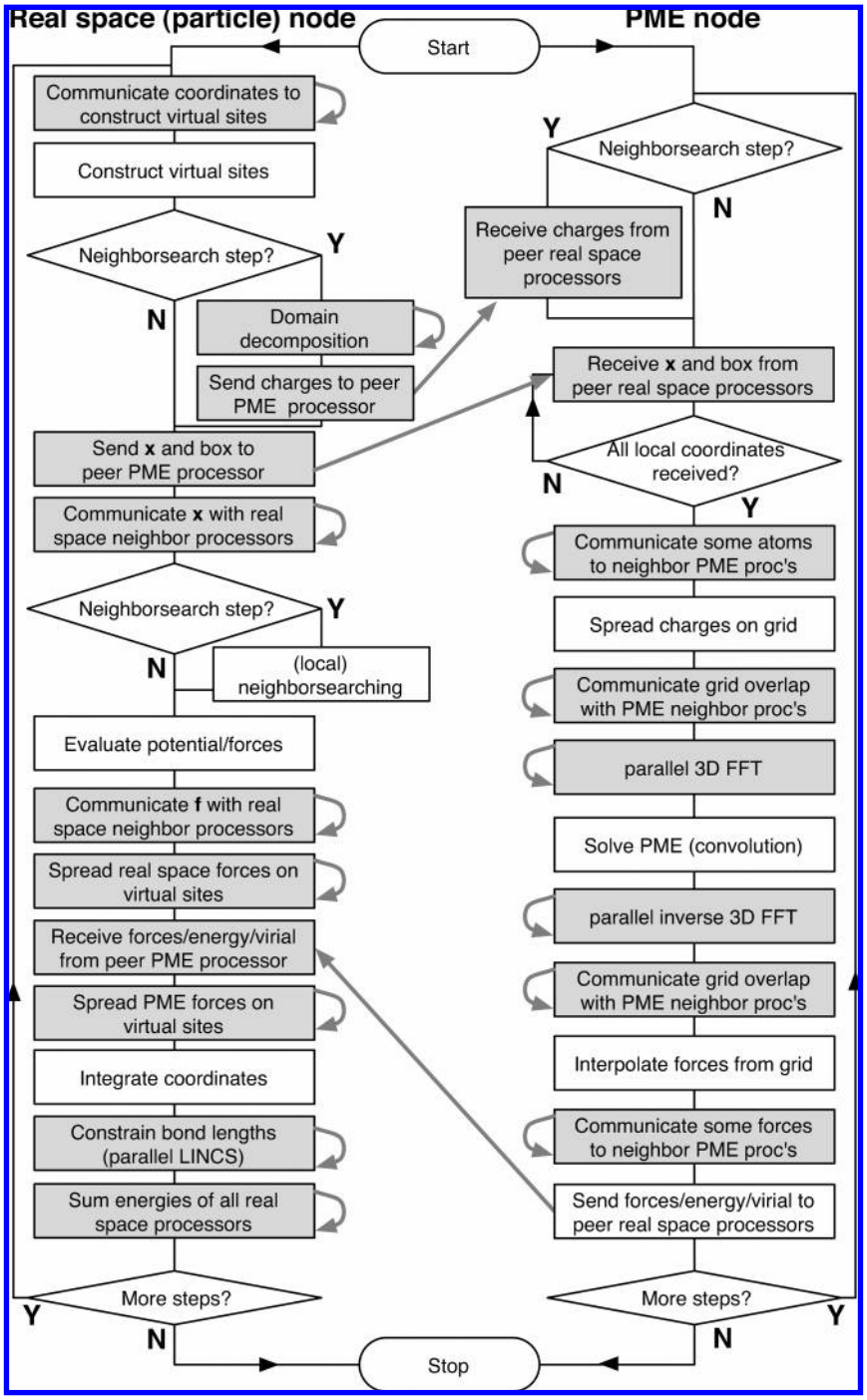

Figure 3. Flowchart for a typical simulation step for both particle and PME nodes. Shaded boxes involve communication, with gray arrows indicating whether the communication only involves similar types of nodes or synchronization between the two domains.

tions over the cells. The restraint or constraint force calculation can then be performed locally.

IX. Benchmarks

The presented benchmarks were performed in the NVT ensemble, using a reversible Nosé-Hoover leapfrog integrator,[29] single precision and dynamic load balancing, unless stated otherwise. Single precision position, velocity and force vectors, combined with some essential variables in double precision is accurate enough for most purposes. In the P-LINCS paper[21] it is shown that with single precision and the constraint velocity correction using the Lagrange multipliers, the energy drift can be reduced to a level unmeasurable over 1 nanosecond. If required, GROMACS can also be compiled in full double precision.

First we will examine the scaling of the basic domain decomposition code, without communication for constraints and virtual sites. To illustrate the basic scaling for all-atom type force fields, we used an OPLS all-atom methanol model,[30] which leads to an interaction density close to that inside a protein. The results for weak scaling, i.e., when the system size grows proportionally with the number of CPUs,

Table 2. Performance in MD Steps per Second for 200 Methanol Molecules (1200 Atoms) per Core[a]

					number of cores				
elec.	prec.	CPU	GHz	cpn	1	2	8	32	128
	S	AMD	2.2	8	167	168	166		
RF	S	Intel	2.33	8	211	216	214		
	S	Intel	3.0	4	274	281	277	265	237
	S	Intel	3.0	2	274	281	284	284	274
RF3	S	Intel	3.0	4	272	274	208	87	44
RF	D	Intel	3.0	4	167	169	159	144	123
	D	Intel	3.0	2	167	169	165	161	153
	S	AMD	2.2	8	103	101	98		
PME	S	Intel	2.33	8	128	127	122		
	S	Intel	3.0	4	172	172	156	150	134
	S	Intel	3.0	2	172	172	162	152	145
PME	D	Intel	3.0	4	112	110	95	90	85
	D	Intel	3.0	2	112	110	100	91	90

[a] With a cutoff of 1 nm, with reaction field (RF), reaction field with GROMACS 3.3 (RF3) and PME with a grid-spacing of 0.121 nm, in single (S) and double (D) precision on AMD64 and Intel Core2 machines with 8 cores per node (cpn) or 4 and 2 cores per node with Infiniband.

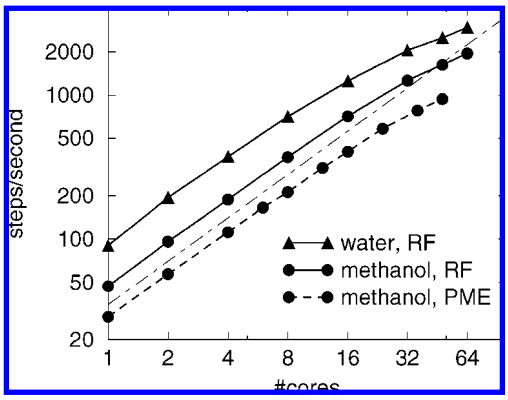

Figure 4. Scaling for a methanol system of 7200 atoms (circles) and an SPC/E water system of 9000 atoms (triangles), with a cutoff 1 nm, with reaction field (solid lines) and PME (dashed line) with a grid-spacing of 0.121 nm (36 × 36 × 36 grid) on a 3 GHz Intel Core2 cluster with Infiniband. The dot-dashed line indicates linear scaling.

are shown in Table 2. With reaction-field electrostatics the computational part of the code scales completely linear. When going from 1 to 2 or 8 cores frequently superlinear scaling can be observed, this is primarily because the charge group sorting is not implemented for single processor simulations. Without PME, the scaling is close to linear, unlike GROMACS 3.3 which already slows down by a factor of 3 on 32 cores. The small drop in performance at 128 processors is caused by the local coordinate and force communication, especially in double precision, and by the global communication for the summation of energies, which is required for temperature coupling. The time spent in the summation increases with the number of processors, since there are more processors to sum over. Unfortunately MPI implementations are often not optimized for the currently typical computing clusters: multiple cores sharing a network connection. With MVAPICH2 on 16 nodes with 4 cores each, the MPI_Allreduce () call takes 120 μs; when we replaced this single call by a two-step procedure, first within each node and then between the nodes, the time is reduced to 90 μs. This global communication is unavoidable for any algorithm that uses global temperature and/or pressure coupling, but the severity depends on the MPI implementation quality. With PME electrostatics linear scaling is impossible, since PME inherently scales as $N \log(N)$. However, in practice the scaling of PME is limited more by the communication involved in the 3D-FFT. However, as evident from Table 2, scaling with PME is still very good, particularly when the high absolute performance is taken into account. Furthermore the difference between 2 and 4 cores per node is quite small. This is because for the communication between the PME processes there is no difference in network speed, as in both cases there is only one PME process per node. With 4 cores per node the real space process to PME process communication all happens within nodes. When one puts the real space and PME processes on separate nodes, the performance with 32 processes decreases by 16%, mainly because each PME process needs to communicate over the network with 3 real space processes while sharing its network connection with 3 other PME processes. Without the MPMD PME implementation the scaling would be much worse, since the FFT grid would need to be redistributed over 4 times as many processors. Still, the 3D-FFT algorithm is one of the points we will focus future performance work on. When switching from single to double precision the performance is reduced by a factor of 1.6. This is not due to the higher cost of the floating point operations but more due to doubling of the required memory bandwidth, both for the force computation and the communication. The PME mesh part becomes relatively cheaper in double precision; therefore, one could optimize the simulation setup to obtain a slightly higher performance. This has not been done for this benchmark.

To illustrate strong scaling we used the same methanol system mentioned before with 1200 molecules as well as a 3000 SPC/E water[27] system. For water with reaction field the scaling is nearly linear up to 2000 MD steps per second, where there are 200 atoms per core (Figure 4). Without PME, the main bottleneck is the summation of energies over all the processors. For the 3000 water system, the summation of energies over 64 cores takes 17% of the total run time. Water runs about twice as fast as methanol, due to the optimized SSE water loops. With PME, methanol scales in the same way but at about 2/3 of the absolute speed of the reaction-field simulations. In contrast to weak scaling, the relative cost of the latency in the coordinate and force communication increases linearly with the number of processors. However, the summation of the energies is still the final bottleneck, since the *relative* cost of this operation increases faster than linear. Thus, the current limit of about 200 atoms per core is due to the communication latency of the Infiniband network.

It is impossible to quantify the general GROMACS performance for coarse-grained systems, since the different levels and ways of coarse-graining lead to very different types of models with different computational demands. Here, we chose a coarse-grained model for polystyrene that uses

Table 3. Number of Steps per Second for a Coarse-Grained Polystyrene Model[a]

package	thermostat	machine	1	2	8	32	64	96
GROMACS	Nosé-Hoover	3 GHz	126	241	964	2950	4120	
GROMACS	Langevin	3 GHz	106	204	829	2860	4760	6170
GROMACS	Langevin	2.33 GHz	80	155	593			
ESPResSo	Langevin	2.33 GHz	41	85	254			

[a] With 9600 beads as a function of the number of cores on a 3 GHz Intel Core2 cluster with 2 cores per Infiniband connection and an 8 core 2.33 GHz Intel Core2 machine.

nonstandard interactions for the bonded as well as the nonbonded interactions.[31] This model uses 2 beads per repeat unit, which leads to a reduction in particles with a factor of 8 compared to an all-atom model and a factor of 4 compared to a united-atom model. The beads are connected linearly in chains of 96 repeat units with bond, angle, and dihedral potentials. The benchmark system consists of a melt of 50 such chains, i.e., 9600 beads, in a cubic box of 9.4 nm. Since the particle density is 8 times lower and the 0.85 nm neighbor list cutoff shorter than that of an atomistic simulation, the computational load per particle for the nonbonded interactions is roughly 10 times less. For this model, the nonbonded and bonded interactions use roughly equal amounts of computational time. This is the only system for which we did not use dynamic load balancing. Because there are so few interactions to calculate, dynamic load balancing slows down the simulations, especially at high parallelization. The benchmark results with a Nose-Hoover and a Langevin thermostat[32] are shown in Table 3. Also shown is a comparison with the ESPResSo package[33] (Extensible Simulation Package for Research on Soft matter). GROMACS is twice as fast as ESPResSo and shows better scaling. This system scales to more than 6000 MD steps per second. The Langevin integrator used requires four random Gaussian numbers per degree of freedom per integration step. With a simpler integrator, as used by Espresso, the performance increases by 18% one 1 core and by 10% on 96 cores. One can see that at low parallelization Langevin dynamics is less efficient, since generating random numbers is relatively expensive for a coarse-grained system. But above 32 cores, or 300 beads per core, it becomes faster than the Nose-Hoover thermostat. This is because the summation of energies is not required at every step for the local Langevin thermostat. Here one can clearly see that simulations with global thermo- and/or barostats in GROMACS 4 are limited by the efficiency of the MPI_Allreduce() call. With the Langevin thermostat the scaling on an Infiniband cluster is only limited by the latency of the coordinate and force communication.

As a representative protein system, we chose T4-lysozyme (164 residues) and the OPLS all-atom force field. We solvated it in a rhombic dodecahedron (triclinic) unit cell with a minimum image distance of 7 nm, with 7156 SPC/E water molecules and 8 Cl^- ions, giving a total of 24119 atoms. The cutoff was 1 nm, and the neighbor list was updated every 20 fs. For electrostatics we used PME with a grid of 56 × 56 × 56 (0.125 nm spacing). Without virtual sites we used a time step of 2 fs and for LINCS 1 iteration

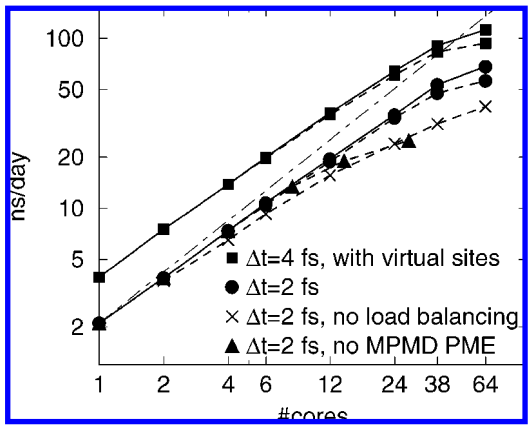

Figure 5. Performance for lysozyme in water (24119 atoms) with OPLS-aa and PME on a 3 GHz dual core Intel Core2 cluster with 2 (solid lines) and 4 (dashed lines) cores per InFIniband interconnect. The dot-dashed line indicates linear scaling.

and an expansion order of 4. With virtual sites we used a time step of 4 fs, a single LINCS iteration (expansion order 6). We ran the benchmarks on a 3 GHz Intel Core2 ("Woodcrest") system with Infiniband interconnects. The real space to PME process ratio for this system is 2:1, except for 38 processes (14 PME) and 64 processes (28 PME). This is the only benchmark that actually communicates with more than one cell in each dimension ($n_p = 2$). Results with 2 and 4 cores per Infiniband connection are shown in Figure 5. When all the presented algorithms are used, the scaling is close to linear up to 38 processors. Without dynamic load balancing the performance is reduced by a factor of 1.5 on 38 processors. When all nodes participate in the PME mesh part, the scaling is limited to 14 processors. With a time step of 2 fs a maximum performance of 68 ns/day is reached, and with a time step of 4 fs this increases to 112 ns/day. Up to 12 processors there is no difference between 2 or 4 cores sharing an Infiniband connection, while at 38 processors the difference is 14%. It is worth mentioning that the repartitioning of the domain decomposition, reassigning charge groups to cells, spatial sorting, setting up the zones, assigning the bonded interactionsm and setting up P-LINCS, always takes a negligible amount of time. The percentage of the total run time spent in repartitioning is 2% with a time step of 2 fs and 4−5% with a time step of 4 fs; the difference is mainly due to the difference in neighbor list update frequency.

For a similar sized protein system we performed a comparison to other simulation packages. We chose one of the most commonly used systems: the joint Amber-CHARMM benchmark DHFR (dihydrofolate reductase) of 23558 atoms in a cubic box of 6.2 nm. Choosing the setup for a benchmark that compares different simulation packages is a difficult issue. Different packages support different features, and the parameter settings for optimal performance can differ between packages. One clear example of this is the box shape. GROMACS can use any triclinic box shape without loss of performance, and one would therefore always choose to solvate a spherical protein in a rhombic dodecahedron unit-cell, which reduces the volume by a factor of

Table 4. Parameters for the DHFR Benchmark and the Energy Drift per Degree of Freedom

package	cutoff (nm)	PME grid	PME freq	time step (fs)	constraints	virtual sites	energy drift (k_BT/ns)
GROMACS	0.96	60 × 60 × 60	1 step	1	none	no	0.011
				2.5	H-bonds	no	0.005
				4	all bonds	yes	0.013
Desmond	0.90	64 × 64 × 64	2 steps	1	none	no	0.017
				2.5	H-bonds	no	0.001
NAMD	0.90	64 × 64 × 64	2 steps	1	none	no	0.023

$\sqrt{2}$ compared to a cubic unit-cell with the same periodic image distance. An important aspect of the setup is the nonbonded interaction treatment. The joint Amber-Charmm benchmark uses interactions that smoothly switch to zero at the cutoff combined with a buffer region. Such a setup is required for accurate energy conservation. But it is questionable if such accurate energy conservation is required for thermostatted simulations. GROMACS loses relatively more performance in such a setup than other packages, since it also calculates all interactions with the buffer region, even though they are all zero. Furthermore, we think that the PME settings for this benchmark (see Table 4) are somewhat conservative; this means the PME-mesh code has a relatively high weight in the results. But since determining the sampling accuracy of molecular simulations goes beyond the scope of this paper, we decided to use the same accuracy and aim for energy conservation. Timings for the Desmond and NAMD[34] packages were taken from the Desmond paper.[35] As Desmond, we used the OPLS all-atom force-field with the TIP3P water model.[3] Note that NAMD and Desmond calculate the PME mesh contribution only every second step, while GROMACS does it every step. We chose to increase the cutoff from 0.9 to 0.96 nm and scale the PME grid spacing accordingly, which provides slightly more accurate forces and a real to reciprocal space process ratio of 3:1. The neighbor list was updated heuristically with a buffer of 0.26 nm. The simulation settings and energy drift are shown in Table 4; note that we took the energy drift values for Desmond and NAMD from the ApoA1 system,[35] which uses a 1.2 nm cutoff and should therefore provide comparable or lower drift. The energy drift for GROMACS is 0.01 k_BT/ns per degree of freedom. This is slightly better than NAMD and Desmond without constraints. With constraints the energy drift with Desmond is an order of magnitude smaller. These results show that codes like GROMACS and Desmond that mainly use single precision do not have larger integration errors than NAMD which uses double precision vectors. It also shows that the use of a time step of 4 fs in GROMACS does not deteriorate the energy conservation. Unfortunately we did not have an identical cluster at our disposal. We also ran the GROMACS benchmarks on a dual core cluster with Infiniband but with 3 GHz Intel Core2 nodes instead of 2.4 GHz AMD64 nodes. Timings for DHFR are shown in Figure 6. If we look at the 1 fs time step results, we can see that, per clock cycle, GROMACS is 2 times faster than Desmond and 3−4 times faster than NAMD, even though the benchmark settings are unfavorable for GROMACS. Additionally GROMACS can be another factor 1.5 faster by increasing the time step from 2.5 to 4 fs, which is made possible by

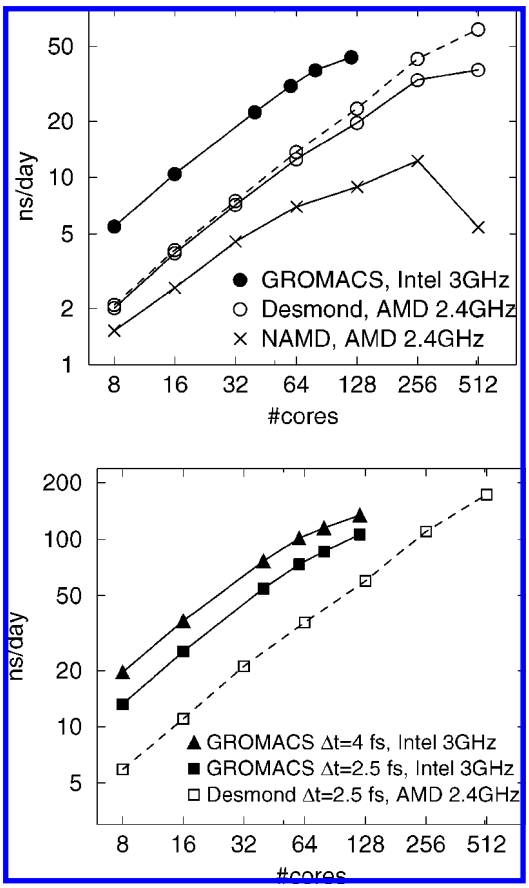

Figure 6. Performance for DHFR in water (23558 atoms) with a 1 fs time step (top panel) and longer time steps (bottom panel) using GROMACS, Desmond, and NAMD. The dashed lines for Desmond show the performance with a tuned Infiniband library.

constraining all bonds and converting hydrogens to virtual sites. With MPI, Desmond shows similar scaling to GROMACS, whereas NAMD scales worse. With a special Infiniband communication library, Desmond scales much further than GROMACS in terms of number of cores but only slightly further in terms of actual performance. GROMACS would certainly also benefit from such a library.

Finally we show the scaling for a large system, which was somewhat of a weak point in earlier GROMACS versions. The system in question is a structure of the Kv1.2 voltage-gated ion channel[36] placed in a 3:1 POPC:POPG bilayer mixture and solvated with water and ions. The OPLS all-atom force field with virtual site hydrogens is used for the ion channel (18,112 atoms), lipids are modeled with the Berger united-atom force field (424 lipids, 22159 atoms),

Table 5. Simulation Speed in ns/day with GROMACS 4 Domain Decomposition and GROMACS 3.3 Particle Decomposition for the Membrane/Protein System (121449 Atoms)[a]

cores	cpn	4	8	16	32	64	128
GROMACS 4	2	3.1	6.1	11.8	22.3	39.3	65.5
GROMACS 4	4	3.1	6.0	11.6	21.6	38.0	60.1
GROMACS 3.3	2	2.8	4.8	7.7	9.5		
GROMACS 3.3	4	2.7	4.8	7.0	8.4		

[a] With a time step of 5 fs on a 3 GHz Intel Core2 Infiniband cluster with 2 and 4 cores per node/network connection (cpn).

and the total system size is 13 × 13 × 8.8 nm, with 119,913 atoms. We used a cutoff of 1.1 nm and a PME grid of 96 × 96 × 64 (spacing 0.136 nm), giving a real space to PME process ratio of 3:1. Removing the hydrogen vibrations by using virtual sites allows for a time step of 5 fs. The neighbor list was updated every 6 steps (30 fs), since the dynamics in the important membrane region is slower than in water. In Table 5 one can see near linear scaling up to 128 processors, where a performance of slightly more than 60 ns/day is reached. With GROMACS 3.3 the system scales up to 32 processors, where it runs at less than half the speed of the domain decomposition; GROMACS 4 reaches an order of magnitude higher performance. The scaling limitation for this type of system is currently the PME FFT implementation.

X. Conclusions

We have shown that the eighth shell domain decomposition and the dynamic load balancing provide very good scaling to large numbers of processors. Dynamic load balancing can provide a 50% performance increase for typical protein simulations. Another important new feature is the Multiple-Program, Multiple-Data PME parallelization, which lowers the number of processes between which the 3D FFT grid needs to be redistributed, while simultaneously increasing the effective communication speed on systems where multiple cores share a network connection. Since the optimal real space to PME process ratio is often 3:1, the benefit of MPMD is higher with 4 or 8 nodes per core than with 1 or 2. This is advantageous, since having more cores per node decreases the cost and space requirements of computing clusters. MPMD allows simulations with PME to scale to double the number of processors and thereby doubles the simulation speed. The P-LINCS and virtual site algorithms allow a doubling of the time step.

But what makes a biomolecular MD package tick is not just a single algorithm but a combination of many efficient algorithms. If one aspect has not been parallelized efficiently, this rapidly becomes a bottleneck—not necessarily for relative scaling but absolute performance. From the benchmarks above, we believe we have largely managed to avoid such bottlenecks in the implementation described here. Not only do the presented algorithms provide very good scaling to large numbers of processors but also we do so without compromising the high single-node performance or any of the algorithms to extend time steps. Together, these features of GROMACS 4 allow for absolute simulation speed that is an order of magnitude larger than previously.

How good the scaling is depends on three factors: the speed of the computational part in isolation, the efficiency of the parallel and communication algorithms, and the efficiency of the communication itself. The first two factors we have been optimized extensively. The single processor performance of GROMACS is unrivaled. This makes good relative scaling extremely difficult, since communication takes relatively more time. Nevertheless, the benchmarks show that the scaling is now nearly linear over a large range of processor counts. The scaling is usually limited by the third factor, the efficiency of the communication. This is given by the network setup and its drivers. With PME the scaling of GROMACS 4 is limited by communication for the 3D FFT. Without PME the scaling is limited by one single communication call per MD step for summing the energies. For any MD code the latter issue cannot be avoided when a global thermostat or barostat is used every step. As a rough guideline one can say that with modern commodity processors connected by an Infiniband network, GROMACS 4 scales close to linear up to 2000 steps per second for simple liquids without PME, while for complex membrane protein simulations (no optimized water kernels) with PME and constraints it scales up to 500 steps per second. There are still alternatives with even more impressive *relative* scaling,[9] and dedicated-hardware implementation might provide extremely high performance if cost is no issue. However, for all normal cases where resources are scarce and absolute performance is the only thing that matters, we believe the implementation presented here will be extremely attractive for molecular simulations.

Acknowledgment. The authors thank the RRZE in Erlangen and especially Georg Hager for providing computing resources for benchmarks and Vagelis Harmandaris for providing the ESPResSo benchmarks. This work was supported by grants from the Swedish Foundation for Strategic Research and the Swedish Research Council (E.L.).

References

(1) Ryckaert, J. P.; Ciccotti, G.; Berendsen, H. J. C. *J. Comput. Phys.* **1977**, *23*, 327.

(2) Berendsen, H. J. C.; Postma, J. P. M.; van Gunsteren, W. F.; Hermans, J. In *Intermolecular Forces*; Pullman, B., Ed.; D. Reidel Publishing Company: Dordrecht, 1981; pp 331−342.

(3) Jorgensen, W. L.; Chandrasekhar, J.; Madura, J. D.; Impey, R. W.; Klein, M. L. *J. Chem. Phys.* **1983**, *79*, 926.

(4) Fincham, D. *Mol. Simul.* **1987**, *1*, 1.

(5) Raine, A. R. C.; Fincham, D.; Smith, W. *Comput. Phys. Commun.* **1989**, *55*, 13.

(6) Clark, T.; McCammon, J. A.; Scott, L. R. In *Proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing*: Dongarra, J., et al., Eds.; SIAM: Philadelphia, 1991; pp 338−344,

(7) Bekker, H.; Berendsen, H. J. C.; Dijkstra, E. J.; Achterop, S.; van Drunen, R.; van der Spoel, D.; Sijbers, A.; Keegstra, H.; Reitsma, B.; Renardus, M. K. R. In *Physics Computing 92*; de Groot, R. A., Nadrchal, J., Eds.; World Scientific: Singapore, 1993; pp 252−256,

(8) Nelson, M.; Humphrey, W.; Gursoy, A.; Dalke, A.; Kalé, L.; Skeel, R.; Schulten, K. *Int. J. High Perform. Comput. Appl.* **1996**, *10*, 251.

(9) Fitch, B.; Germain, R.; Mendell, M.; Pitera, J.; Pitman, M.; Rayshubskiy, A.; Sham, Y.; Suits, F.; Swope, W.; Ward, T.; Zhestkov, Y.; Zhou, R. *J. Parallel Distributed Comput.* **2003**, *63*, 759.

(10) Rhee, Y. M.; Sorin, E. J.; Jayachandran, G.; Lindahl, E.; Pande, V. S. *Proc. Natl. Acad. Sci. U.S.A.* **2004**, *101*, 6456.

(11) Lindahl, E.; Hess, B.; van der Spoel, D. *J. Mol. Model.* **2001**, 7, 306.

(12) van der Spoel, D.; Lindahl, E.; Hess, B.; Groenhof, G.; Mark, A. E.; Berendsen, H. J. C. *J. Comput. Chem.* **2005**, *26*, 1701.

(13) Feenstra, K. A.; Hess, B.; Berendsen, H. J. C. *J. Comput. Chem.* **1999**, *20*, 786.

(14) Essmann, U.; Perera, L.; Berkowitz, M. L.; Darden, T.; Lee, H.; Pedersen, L. G. *J. Chem. Phys.* **1995**, *103*, 8577.

(15) Greengard, L.; Rokhlin, V. *J. Comput. Phys.* **1987**, *73*, 325.

(16) Bowers, K. J.; Dror, R. O.; Shaw, D. E. *J. Comput. Phys.* **2007**, 221, 303.

(17) Bowers, K. J.; Dror, R. O.; Shaw, D. E. *J. Phys. Conf. Ser.* **2005**, *16*, 300.

(18) Bowers, K. J.; Dror, R. O.; Shaw, D. E. *J. Chem. Phys.* **2006**, *124* (18), 184109.

(19) Liem, S. Y.; Brown, D.; Clarke, J. H. R. *Comput. Phys. Commun*. **1991**, *67* (2), 261.

(20) Hess, B.; Bekker, H.; Berendsen, H. J. C.; Fraaije, J. G. E. M. *J. Comput. Chem.* **1997**, *18*, 1463.

(21) Hess, B. *J. Chem. Theory Comput.* **2008**, *4* (1), 116.

(22) Lippert, R. A.; Bowers, K. J.; Dror, R. O.; Eastwood, M. P.; Gregersen, B. A.; Klepeis, J. L.; Kolossvary, I.; Shaw, D. E. *J. Chem. Phys.* **2007**, *126*, 046101.

(23) Miyamoto, S.; Kollman, P. A. *J. Comput. Chem.* **1992**, *13*, 952.

(24) van Gunsteren, W. F.; Karplus, M. *Macromolecules* **1982**, *15*, 1528.

(25) Meloni, S.; Rosati, M. *J. Chem. Phys.* **2007**, *126*, 121102.

(26) Weerasinghe, S.; Smith, P. E. *J. Chem. Phys.* **2003**, *119* (21), 11342.

(27) Berendsen, H. J. C.; Grigera, J. R.; Straatsma, T. P. *J. Phys. Chem.* **1987**, *91*, 6269.

(28) Kutzner, C.; van der Spoel, D.; Fechner, M.; Lindahl, E.; Schmitt, U. W.; de Groot, B. L.; Grubmuller, H. *J. Comput. Chem.* **2007**, *28*, 2075.

(29) Holian, B. L.; Voter, A. F.; Ravelo, R. *Phys. Rev. E* **1995**, *52* (3), 2338.

(30) Jorgensen, W. L.; Maxwell, D. S.; Tirado-Rives, J. *J. Am. Chem. Soc.* **1996**, *118*, 11225.

(31) Harmandaris, V. A.; Reith, D.; van der Vegt, N. F. A.; Kremer, K. *Macromolecules* **2007**, *208*, 2109.

(32) van Gunsteren, W. F.; Berendsen, H. J. C. *Mol. Simul*. **1988**, *1*, 173.

(33) Limbach, H.-J.; Arnold, A.; Mann, B. A.; Holm, C. *Comput. Phys. Commun*. **2006**, *174* (9), 704.

(34) Phillips, J. C.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kalé, L.; Schulten, K. *J. Comput. Chem.* **2005**, *26* (16), 1781.

(35) Bowers, K. J.; Chow, E.; Xu, H.; Dror, R. O.; Eastwood, M. P.; Gregersen, B. A.; Klepeis, J. L.; Kolossvary, I.; Moraes, M. A.; Sacerdoti, F. D.; Salmon, J. K.; Shan, Y.; Shaw, D. E. In *ACM/IEEE SC 2006 Conference (SC'06)*; 2006; p 43.

(36) Long, S.; Campbell, E. B.; MacKinnon, R. *Science* **2005**, *309*, 897.

CT700301Q