Automatized Performance Tuning for GROMACS

Carsten Kutzner and Helmut Grubmüller

Dep. of Theoretical and Computational Biophysics, Max-Planck-Institute for Biophysical Chemistry, Göttingen, Germany

Abstract

Parallel molecular dynamics calculations with PME[1,2] electrostatics enjoy a considerable performance increase in Gromacs 4.0[3] compared to older versions. Part of that increase results from the fact that a subgroup of the parallel computer is assigned for the PME calculation which requires all-to-all communication among all participants. All-to-all communication is one of the major scaling bottlenecks of parallel PME calculations[4].

For optimum performance it is essential that every processor gets assigned an equal amount of work. Most critical in this respect is that both processor subgroups complete each time step simultaneously such that no group has to wait for the other group. This can be achieved by carefully balancing the ratio of PME to direct-space processors. Gromacs tries to predict this ratio based upon hard-wired reference numbers. However, it does not know about the underlying network or processor speeds, facts which can influence the forecast seriously. As a consequence, the optimum number of PME nodes can only be derived with the help of test runs.

This is where g_tune_pme comes into play: It will do the required test runs for you and subsequently launch the simulation with the best possible settings. All it needs is the tpr file prepared for the simulation.

g_tune_pme

- finds optimum number of PME nodes
- checks whether performance can be increased by shifting work between real and reciprocal space
- direct launch of simulation after tuning
- provided with Gromacs 4.1. Stand-alone version for version 4.0 is available at www.mpibpc.mpg.de/home/grubmueller/projects/MethodAdvancements/Gromacs/

Example call

```
g_tune_pme -np 64 -s protein.tpr -launch
```

Example benchmark

As a typical example we chose the DPPC membrane system from the Gromacs benchmark suite available at www.gromacs.org > benchmarks > gmxbench-3.0.tar.gz. The system contains 1024 DPPC lipids plus 23,552 SPC water molecules and thus altogether 121,856 particles.

For the benchmark we set

```
coulombtype = PME
r_coul = 1.0 [nm]
r_vdW = 1.0 [nm]
fourierpacing = 0.135 [nm]
```

yielding a standard PME grid of 132*140*48 points. For a scaling factor of 1.1 for cutoffs and fourierspacing the resulting PME grid is 120*125*42 points.

The benchmarks were performed on a cluster of 8 processor Intel L5430@2.66 GHz (Harpertown) nodes connected by 4xDDR Infiniband (20 Gbit/s).

References

(1) T. Darden, D. York, L. Pedersen, 1993. Particle mesh Ewald: An N*log(N) method for Ewald sums in large systems, J. Chem. Phys. 98 (12), 10089-10092

(2) U. Essmann, L. Perera, M.L. Berkowitz, T. Darden, H. Lee, G. Pedersen, 1995. A smooth particle mesh Ewald method, J. Chem. Phys. 103 (19), 8577-8593

(3) B. Hess, C. Kutzner, D. van der Spoel, E. Lindahl, 2008. GROMACS 4: algorithms for highly efficient, load-balanced, and scalable molecular simulation. JCTC 4 (3), 435-447

(4) C. Kutzner, D. van der Spoel, M. Fechner, E. Lindahl, U. Schmitt, B. de Groot and H. Grubmüler, 2007. Speeding up parallel GROMACS on high-latency networks. J. Comp. Chem. 28 (12), 2075-2084

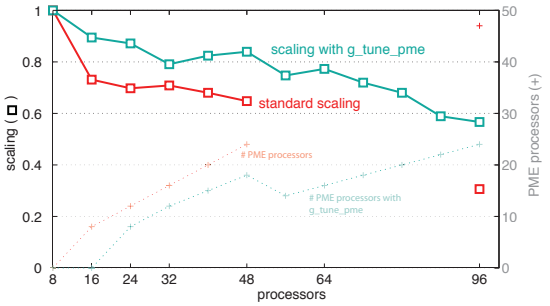

Fig. 2: Squares (left scale) show the scaling which is the execution time on a single node divided by *n* times the execution time on *n* nodes.
Plus signs (right scale) show the number of PME processors. Up to 48 processors the tpr file was left untouched whereas from 56 processors on a slight shift (factor 1.1) of work from reciprocal to real space came out to be advantageous for the performance. (Between 48 and 96 processors the required number of PME processors could not be estimated by mdrun.)

Performance with and without tuning

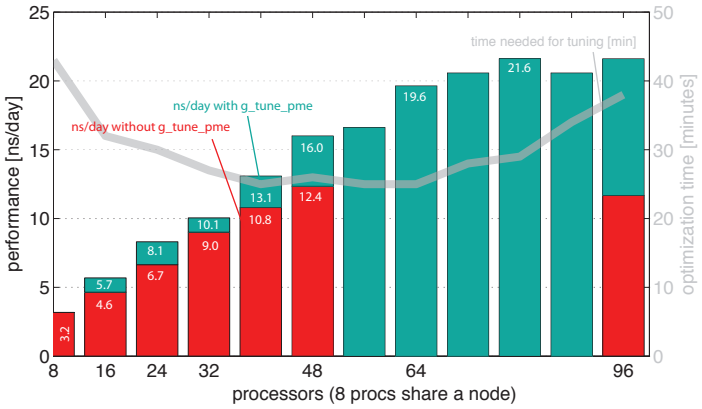

Fig. 1: Red bars (left scale) depict the DPPC performance [ns/day] at a 2 fs timestep when the number of PME processors is estimated by mdrun. The green bars show the gain after tuning.
The grey line (right scale) shows the wallclock time needed for the tuning, which was about half an hour here.

How it works

The tool reads in a simulation tpr file and writes several benchmark tpr files where a) the number of steps is set to the benchmark value–typically 2500–and b) computational load is shifted between real and reciprocal part of the Ewald sum. This can be achieved by multiplying both the direct-space cutoff as well as the PME grid spacing by a small factor, typically 1.0, 1.1 and 1.2. The higher the factor, the more work is shifted from the PME to the direct-space processors. This is done because the reciprocal space calculations are most efficient on a small number of processors.
Subsequently, the newly created tpr files are benchmarked whereas the number of PME processors is varied between 1/4 and 1/2 of all processors. For better statistics this can be done several times. The best performance-wise parameters are determined and finally the simulation with these parameters is launched.

Summary and outlook

- The more processors, the more can typically be gained by g_tune_pme
- Tuning already pays off when a simulation takes more than a few hours of wallclock time
- In most cases performance can be improved notably. For the presented benchmark the performance could be raised by a factor of 1.2 on average for 16–48 processors.
- Outlook: Incorporate an error estimation for the PME part. This way, load can be shifted between real and reciprocal space at a constant absolute error. (The currently assumed linear relationship is backed up by numerical tests.) Florian Dommert, Frankfurt Institute for Advanced Studies, currently works on an analytical error estimate for PME.