Improved GROMACS scaling on Ethernet switched clusters

C. Kutzner[a], D. van der Spoel[b], M. Fechner[a], E. Lindahl[c], U.W. Schmitt[a], B.L. de Groot[a], H. Grubmüller[a]

[a]Dep. of Theoretical and Computational Biophysics, Max-Planck-Institute for Biophysical Chemistry, Göttingen, Germany
[b]Department of Cell and Molecular Biology, Uppsala University, Sweden
[c]Stockholm Bioinformatics Center, Stockholm University, Stockholm, Sweden

Abstract

We investigated the prerequisites for decent scaling of the GROMACS 3.3 molecular dynamics (MD) code on Ethernet Beowulf clusters.

GROMACS[1] uses the Message Passing Interface (MPI) standard for communication between the processors. The long range electrostatic forces are evaluated with the Particle-Mesh-Ewald[2,3] (PME) method which is significantly more accurate than a simple cutoff treatment. The code scales well on shared memory supercomputers like the IBM p690 (Regatta) and on Linux clusters with a special interconnect like Myrinet or Infiniband. Exemplary speedups for an 80k atom test system are Sp_8=6.2, Sp_{16}=10 on Myrinet, Sp_{16}=11 on Infiniband and Sp_{32}=21 on a Regatta node. On Ethernet switched clusters, however, the scaling typically breaks down as soon as more than two computational nodes are involved (central figure, orange). Each node contains 1 or 2 CPUs in this study.

1. As the main scaling bottleneck we identified all-to-all communication, which is needed twice each time step. It performs a parallel transpose within the Fast Fourier Transformation, the latter being essential for PME. During an all-to-all, each process sends a unique message (however, of same size) to every other process. When using the LAM MPI implementation, a huge amount of simultaneous and therefore colliding messages „floods" the network, resulting in frequent TCP packet loss and time consuming re-trials. The performance of the MPI_Alltoall routine was systematically investigated. We benchmarked both the entire application (with a typical MD system) as well as the isolated all-to-all routine. Here we show results for LAM-MPI (as it is mostly adopted for GROMACS on Ethernet) but all tests were also done with MPICH.

2. We find that the MPI_Alltoall (and thus GROMACS) performs significantly better when IEEE 802.3x flow control[4] is activated in the Ethernet switch (green).

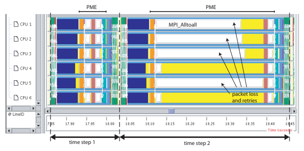
3. For network switches not supporting flow control, we have implemented an ordered all-to-all routine for multi-CPU nodes that performs reliably well for typical GROMACS all-to-all message sizes. Thus the scaling significantly improves, even for switches that lack flow control (blue).

4. In addition, for the common HP ProCurve 2848 switch we find that for optimum all-to-all performance it is essential how the nodes are connected to the switch's ports. This is also demonstrated in the example of the Car-Parinello MD code[7].

Summary

- on Ethernet Beowulf clusters, MPI_Alltoall network congestion typically prevents scaling of GROMACS to > 2 nodes
- enabling flow control overcomes this bottleneck for up to 16 nodes
- for switches that do not support flow control, incorporating the ordered all-to-all into the code serves as a fallback mechanism
- in both cases a speedup of >8 on 16 single-CPU nodes is reached for the 80k atom test system
- Packet loss within the switch can also heavily degrade parallel performance. For the common HP 2848 switch this can be prevented by using a maximum of 9 out of 12 consecutive ports.
- these findings are applicable to other programs using all-to-all communication over Ethernet

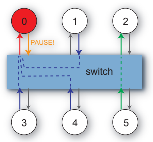
1. The problem: congestion during all-to-all



Two GROMACS time steps as logged with the MPE tools[5]. White bars indicate MPI_Alltoall.

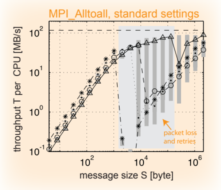
2. How flow control helps

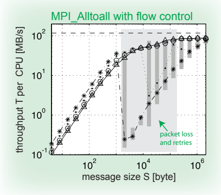
The IEEE 802.3x standard[4] defines a flow control mechanism at the link level. A receiver may send a PAUSE frame to tell the source to stop sending.

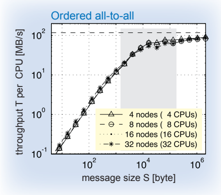


single-CPU nodes

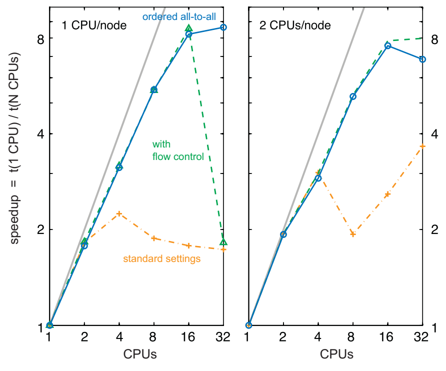
Throughput T for LAM 7.1.1 all-to-all communication.
In an all-to-all on N processors, a given process sends N−1 messages of size S to the other processes. Let the transfer of these altogether N(N−1) messages take the time Δt.
We then define the throughput T per CPU as T = (N−1)S/Δt.
Typical MD systems yield sizes S in the blue area.







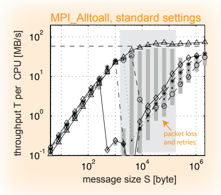
GROMACS speedups on Gigabit Ethernet

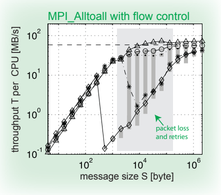


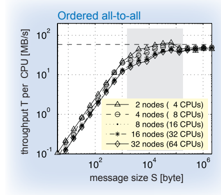
GROMACS speedups with standard switch settings (orange), with flow control (green) and with ordered all-to-all (blue). Test system is a protein tetramer embedded in a lipid bilayer membrane surrounded by water, approx. 80k atoms altogether

dual-CPU nodes

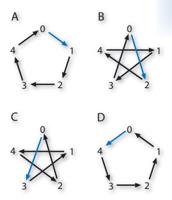
AMD Opterons, Linux 2.6.5,
Broadcom NetXtreme BCM5704 onboard NICs



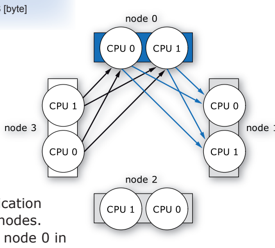




3. Preventing congestion with ordered communication

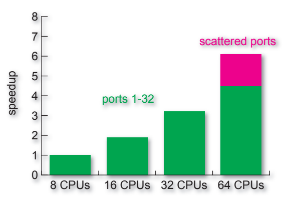


Left: An ordered communication scheme for 5 processes[6].

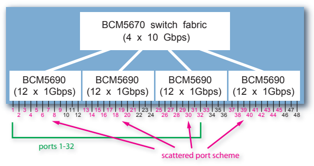


Right: Our multi-CPU node communication scheme for the case of 4 dual-CPU nodes. Shown is just the traffic to and from node 0 in a single phase (corresponding to phase A of single-CPU scheme).

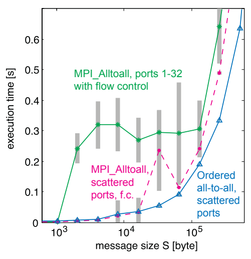
4. The switch – yet another bottleneck



CPMD 3.9.1[7] speedups for a 64 water system inside a cubic box with length 1.242 nm using the LAM MPI_Alltoall (with flow control). Grey: ports 1-32 on the switch. Green: scattered ports 1-9, 13-21, 25-33, 37-41.



The internal 10 Gbps connections of the common HP ProCurve 2848 switch are a bottleneck since each subswitch generates up to 12 Gbps traffic. We experimentally determined that packet loss can be prevented if not more than 9 out of each group of 12 ports are used.



Green: Execution time of the MPICH-2 1.0.3 all-to-all for 64 processes on 32 nodes using the ports 1-32 on the switch (with flow control). Magenta: Same on scattered ports 1-9, 13-21, 25-33, 37-41. Blue: Our ordered multi-CPU all-to-all on scattered ports.

References

(1) D. van der Spoel, E. Lindahl, B. Hess, G. Groenhof, A.E. Mark, H.J.C. Berendsen, 2005. GROMACS: Fast, Flexible, and Free. J. Comput. Chem. 26.
(2) 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.
(3) 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.
(4) IEEE, 2000. Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements–Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications. IEEE 802.3, New York, pp. 1472–1481, Annex 31B: MAC control PAUSE operation.
(5) O. Zaki, E. Lusk, W. Gropp, D. Swider 1999. Toward scalable performance visualization with Jumpshot. Int. J. High Perform. Comput. Appl. 13 (3).
(6) A. Karwande, X. Yuan, D. Lowenthal 2005. An MPI prototype for compiled communication on Ethernet switched clusters. J. Parallel Dist. Comp. 65
(7) http://www.cpmd.org/