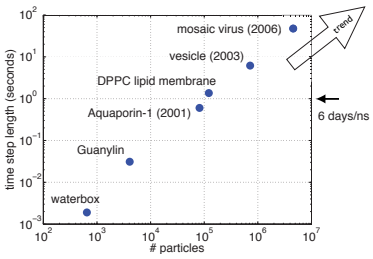
# Optimising GROMACS for parallel performance

C. Kutzner[a], D. van der Spoel[b], E. Lindahl[c], Berk Hess[d], Jakob Pichlmeier[e], 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
[d]Max-Planck-Institute for Polymer Research, Mainz, Germany
[e]IBM Munich, Germany

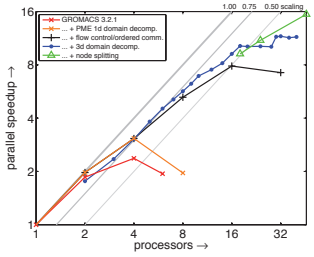
## The need for speed




Simulation systems are getting larger and larger. Since we cannot wait years for the results, multiple processors have to work on it in parallel.

- The **GROMACS** molecular dynamics (MD) code has an extremely **high single-processor performance**
- However, large systems are feasible only with parallel processing
- The more processors take part in a simulation, the more the parallel efficiency is degraded by
  - **network bottlenecks**
  - **communication overheads**
  - **uneven load balancing**
- most of time is spent in calculation of non-bonded forces. Calculation is split into a short range (SR) and a long range (LR) part. GROMACS uses the efficient Particle-Mesh-Ewald (PME) method to evaluate the Coulomb forces.
- Long range part needs **all-to-all** communication for FFT transposes. In an all-to-all on N procs, $N^2$ messages are transferred!

## Scaling improvements




Due to communication overheads and uneven loads the maximal speedup is limited. It can be raised by careful code optimisations.

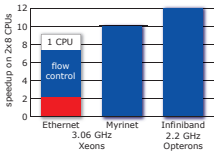
# Project overview

## Hide communication

- SR and LR Coulomb forces can be calculated independently
- LR part needs excessive communication (including 2x all-to-all), which can possibly be **overlapped** with SR calculation

## Network optimisations

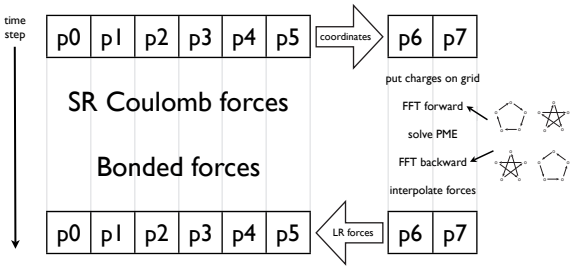
- **flow control** was found to be a prerequisite for good scaling on Ethernet
- GROMACS speedups with and without flow control, compared to high-performance interconnects:




- switch can become a bottleneck, too
- enlarge bandwidth by using multiple network interfaces per node, as we will see more and more processors on a node in the future
- reduce latency with a low-overhead protocol (GAMMA)

## Node splitting

- do **PME on a subgroup of all processors**. All-to-all communication and the FFT is much more efficient on smaller number of CPUs. Also the grids have to be divisible by the number of nodes.




future projects

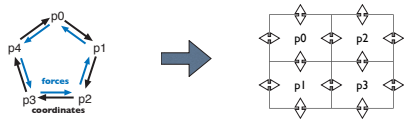
## PME frequency reduction

- since the long-range part of the Coulomb potential anyway varies slowly, construction of the potential grid at every time step might be overkill
- if PME is only done every 2nd time step, the average time step length would be reduced by approx. 20%!
- of course, careful error estimations have to be performed

ongoing projects

## 3d domain decomposition
**(is implemented by B. Hess)**

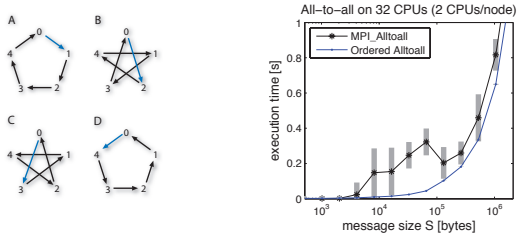
- each processor gets assigned a part of the space, not a part of the atoms
- makes the number of communication steps for the short range part **independent of the number of processes** (N)




- N/2 systolic pulses -> only 3 pulses to transfer coordinate / force data

## Ordered communication

- Problem: All-to-all communication on a large number of processors can easily overload the network
- our **ordered all-to-all algorithm** prevents that and thereby reduces communication time




past projects

## PME 1d domain decomposition
**(was implemented by J. Pichlmeier)**

- after sorting the charges in x-direction only the PME grid boundaries have to be communicated (instead of the whole grids)
- the boundaries can be transferred in just 2 communication steps, independent of the number of processes!
- as a result the communication volume within the PME part is significantly reduced