Running MD on HPC architectures I. Hybrid Clusters

Alessandro Grottgesi
Cineca
Today's lecture

You will learn:

• Gromacs @ CINECA: set up and launch of simulations
• Launch MD code (GROMACS, NAMD)
• Optimize performance and benchmarking
• Tutorial (later this afternoon...)

SuperComputing Applications and Innovation
Eurora

Model: Eurora prototype
Architecture: Linux Infiniband Cluster

Processors Type:
- Intel Xeon (Eight-Core SandyBridge) E5-2658 2.10 GHz (Compute)
- Intel Xeon (Eight-Core SandyBridge) E5-2687W 3.10 GHz (Compute)
- Intel Xeon (Esa-Core Westmere) E5645 2.4 GHz (Login)

Number of nodes: 64 Compute + 1 Login
Number of cores: 1024 (compute) + 12 (login)

Number of accelerators: 64 nVIDIA Tesla K20 (Kepler) + 64 Intel Xeon Phi (MIC)

RAM: 1.1 TB (16 GB/Compute node + 32GB/Fat node)

OS: RedHat CentOS release 6.3, 64 bit
Model: IBM NeXtScale server
Architecture: Linux Infiniband Cluster
Processors Type: Intel Xeon (Ten-Core) E5-2670v2 2.50 GHz (Compute)
Number of nodes: 54 Compute + 4 visualization + 2 Login + 14 other
Number of cores: 1080 (compute)
Number of accelerators: 4 + 2 (only on viz nodes)
RAM: 128 GB/Compute node (2 viz nodes with 512GB)
Galileo

Model: IBM NeXtScale
Architecture: Linux Infiniband Cluster
Nodes: 516
Processors: 8-cores Intel Haswell 2.40 GHz (2 per node)
Cores: 16 cores/node, 8256 cores in total
Accelerators: 2 Intel Phi 7120p per node on 384 nodes (768 in total)
RAM: 128 GB/node, 8 GB/core
Internal Network: Infiniband with 4x QDR switches
Disk Space: 2,500 TB of local storage
Peak Performance: \(~1\) Pflop/s (69\textsuperscript{th} on TOP500)
Compute nodes

Infiniband connection

48 Gb RAM
Molecular Dynamics and accelerators

- **GROMACS**
  - Under development, currently (Jan 2015) only native-mode version available.

- **NAMD**
  - Pre-release version of NAMD 2.10 has Xeon PHI support but still under development. Speed-ups < 2 for ApoA1 and STMV benchmarks.

- **LAMMPS**
  - Xeon PHI support available in current downloads for non-bonded calculations. Reported speed-ups of about 1.78x compared to non-accelerated code (one coprocessor/node) for Rhodopsin benchmark. Higher speed-ups obtained with materials simulations.

- **AMBER**
Intel Xeon PHI overview

• Intel product line based on Intel’s Many Integrated Core (MIC) technology where many low, power cores (>50) are packed on a single chip.

• Currently available device (Knight’s Corner or KNC) can be seen as a co-processor, in direct competition to NVIDIA GPU for HPC.
  – connection to host CPU via PCI-eXpress link.

• But unlike GPU technology is not too dissimilar from host CPU → not essential to rewrite code, in principle just re-compile (with Intel compilers). Should lead to shorter development path in most cases.

• Doesn’t mean though that code does need not porting – to obtain peak performance some optimisation is needed.

Most powerful supercomputer in TOP500 (Tianhe-2) uses 48000 Xeon PHI cards.
Classical Molecular Dynamics and Intel Xeon PHI

• Development some way behind GPU-CUDA versions of classical MD programs (which started about 4 years ago).
• But given that there is no need to rewrite in new languages (e.g CUDA) development path should be shorter.
• Most current Xeon PHI versions seem to be based on the off-load model to exploit CUDA developments.
• Off-loaded calculations invariably involve non-bonded dispersion interactions may also include PME, energy calculation etc.
• Intel maintains a list of “recipes” for building Xeon PHI applications (not just MD):

Running MD code on Euria
(GROMACS)
GROMACS: what for...?

1. Minimization
2. Molecular Dynamics (classic, brownian, Langevin)
3. Normal Mode Analysis
4. Essential Dynamics and Sampling
5. Free Energy calculations (FEP, Umbrella sampling, AFM)
6. Replica Exchange Molecular Dynamics
7. Coarse-Grained MD
8. Metadynamics
9. Much more...
Available forcefield in Gromacs (4.6.5)

1. AMBER03 protein, nucleic AMBER94 (Duan et al., J. Comp. Chem. 24, 1999-2012, 2003)
2. AMBER94 force field (Cornell et al., JACS 117, 5179-5197, 1995)
3. AMBER96 protein, nucleic AMBER94 (Kollman et al., Acc. Chem. Res. 29, 461-469, 1996)
4. AMBER99 protein, nucleic AMBER94 (Wang et al., J. Comp. Chem. 21, 1049-1074, 2000)
5. AMBER99SB protein, nucleic AMBER94 (Hornak et al., Proteins 65, 712-725, 2006)
6. AMBER99SB-ILDN protein, nucleic AMBER94 (Lindorff-Larsen et al., Proteins 78, 1950-58, 2010)
7. AMBERGS force field (Garcia & Sanbonmatsu, PNAS 99, 2782-2787, 2002)
8. CHARMM27 all-atom force field (with CMAP) - version 2.0
9. GROMOS96 43a1 force field
10. GROMOS96 43a2 force field (improved alkane dihedrals)
11. GROMOS96 45a3 force field (Schuler JCC 2001 22 1205)
12. GROMOS96 53a5 force field (JCC 2004 vol 25 pag 1656)
13. GROMOS96 53a6 force field (JCC 2004 vol 25 pag 1656)
15. OPLS-AA/L all-atom force field (2001 aminoacid dihedrals)
16. [DEPRECATED] Encad all-atom force field, using full solvent charges
17. [DEPRECATED] Encad all-atom force field, using scaled-down vacuum charges
18. [DEPRECATED] Gromacs force field (see manual)
19. [DEPRECATED] Gromacs force field with hydrogens for NMR
To convert a structure pdb file into a Gromacs topology:

```
pdb2gmx -f input_file.pdb -ignh -ter
```

input:
1. file_in.pdb initial set of coordinates (either pdb or gro format)

output:
1. topl.top system topology
2. posre.itp position restraints file
3. conf.gro coordinate file (gro format by default)
4. toplA.itp, toplB.itp, etc topology of chain A, B, etc...
5. posreA.itp, posreB.itp, etc position restraints file for chain A, B, etc...
How to generate the box: editconf

Structure generated file has to be immersed in a box of water molecules (or alternative solvent) prior to run an MD simulation. Different types of box are available in Gromacs (triclinic, cubic, dodecahedron or octahedron) and can be generated by the command:

```bash
eeditconf –f conf.gro -bt triclinic –d 0.8 –o output.gro
```

Box type: triclinic in this case

Minimal solute-solvent distance along box axes.
Box solvation: genbox

Once defined, box has to be physically soaked with water (or alternative solvent). This can be easily performed by running the command:

```
genbox –cp conf.gro –cs spc216.gro –o out.gro -p topol.top
```
Grid based electrostatic treatment (Ewald sums, PME, etc.) are better performed with system net charge = 0. Namely, make sure that:

\[
\text{solute charge} + \text{solvent charge} = 0
\]

To set up box neutrality we can replace as many water molecule with corresponding positive or negative ions to generate a total charge = 0. To do so, we can run the genion command as follows:

```
genion –s topol.tpr –seed XXX –o ution.gro
   –nn 20 –np 10 -p topol.top
```

This command replace randomly a total of 30 water molecules with 20 negative ions (chloride) and 10 positive ions (sodium) and updates the topol.top file with the new list of atoms.
grompp: the GROMACS preprocessor

Command grompp generates a binary input file with all structural info and forcefield parameters needed to run an MD simulation.

```plaintext
  grompp -f param.mdp -c coord.gro -n index.ndx -p topol.top -o topol.tpr
```

Grompp output is a binary file called topol.tpr that can be used as input for running the calculation. To visualize and check all info stored in the topol.tpr file we can use the following command:

```plaintext
  gmxdump -s topol.tpr
```
Output control

cpp = Yo
Include = -I../top
Cefine = -DPOSRES
Integrator = md
Dt = 0.002
Nsteps = 500000
Nstxout = 5000
Nstvout = 5000
Nstlog = 5000
Nstenergy = 250
Nstxtcou = 250
Xtc_grps = Protein
Energygrps = Protein  SOL
Nstlist = 10
Ns_type = grid
Rlist = 0.8
Coulombtype = PME
Rcoulomb = 1.4
Rvdw = 0.8
Tcoupl = V-Rescale
Tc-grps = Protein  SOL
Tau_t = 0.1 0.1
Ref_t = 300 300
Pcoupl =
Tau_p = 1.0
Compressibility = 4.5e-5
Ref_p = 1.0
Gen_vel = yes
Gen_temp = 300
Gen_seed = 173529
Constraints = all-bonds

Van der Waals and electrostatics

Temperature and pressure coupling
Output control parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nsteps</td>
<td>500000</td>
<td>Total number of steps</td>
</tr>
<tr>
<td>nstxout</td>
<td>5000</td>
<td>coords output frequency for traj.trr</td>
</tr>
<tr>
<td>nstvout</td>
<td>5000</td>
<td>velocity output frequency for traj.trr</td>
</tr>
<tr>
<td>nstlog</td>
<td>5000</td>
<td>output frequency for log file (md.log)</td>
</tr>
<tr>
<td>nstenergy</td>
<td>250</td>
<td>output frequency for energy file ener.edr</td>
</tr>
<tr>
<td>nstxtcout</td>
<td>250</td>
<td>coords output frequency for traj.xtc</td>
</tr>
<tr>
<td>xtc_grps</td>
<td>Protein</td>
<td>content of file traj.xtc</td>
</tr>
<tr>
<td>energygrps</td>
<td>Protein SOL</td>
<td>energy groups to store in file ener.edr</td>
</tr>
</tbody>
</table>

File `traj.xtc` contains coordinates of our simulated system. Atomic coordinates are saved in a compressed format so that to reduce file size. This file is the main trajectory file used for simulation analysis.

File `traj.trr` contains atomic coordinates, velocities and forces of our simulated system. These data are saved as 4 digits floating point numbers and are usefull to recover coordinates and velocities after a job crash or if we need velocities and forces for special analyses.
Electrostatics control

; Method for doing electrostatics
coulombtype = PME
rcoulomb-switch = 0
rcoulomb = 1.2
; Relative dielectric constant for the medium and the reaction field
epsilon_r = 1
epsilon_rf = 1
; Method for doing Van der Waals
vdw-type = Cut-off
; cut-off lengths
rvdw-switch = 0
rvdw = 1.2
; Spacing for the PME/PPPM FFT grid
fourierspacing = 0.150
; FFT grid size, when a value is 0 fourierspacing will be used
fourier_nx = 0
fourier_ny = 0
fourier_nz = 0
; EWALD/PME/PPPM parameters
pme_order = 4
ewald_rtol = 1e-05
ewald_geometry = 3d
epsilon_surface = 0
optimize_fft = no
Parameter for temperature and pressure coupling

; Temperature coupling
Tcoupl = V-rescale
; Groups to couple separately
tc-grps = System
; Time constant (ps) and reference temperature (K)
tau_t = 0.1
ref_t = 250.0

; Pressure coupling
Pcoupl = Parrinello-Rahman
Pcoupltype = isotropic
; Time constant (ps), compressibility (1/bar) and reference P (bar)
tau_p = 0.5
compressibility = 4.5e-5
ref_p = 1.0

; Scaling of reference coordinates, No, All or COM
refcoord_scaling = No
; Random seed for Andersen thermostat
andersen_seed = 815131
Run the simulation: mdrun

```
mdrun -s topol.tpr -dd dx dy dz -pd -npme N
```

The command generates many output files at the end of the job. Among them:

1. confout.gro  final coordinates file (gro format)
2. traj.xtc  simulation trajectory file (compressed)
3. traj.trr  simulation trajectory file (coord+velocity, high prec.)
4. ener.edr  energy file
5. state.cpt  checkpoint file for restarting runs.
6. md.log  log file with output control

Domain decomposition and particle decomposition.
Gromacs 5.0.4, pure MPI on Eurora

#!/bin/bash

#PBS -N gmx
#PBS -l select=1:ncpus=16:mpiprocs=16:mem=14GB
#PBS -l walltime=1:00:00
#PBS -q R1660526
#PBS -W group_list=train_cmd32015
#PBS -A train_cmd32015

cd $PBS_O_WORKDIR

module load profile/advanced
module load autoload gromacs/5.0.4

export OMP_NUM_THREADS=1

mdrun=$(which mdrun_mpi)
cmd="$mdrun -s topol.tpr -v -maxh 1.0 -nb cpu"
mpirun -np 16 $cmd
Gromacs 5.0.4, pure MPI on Eurora
#!/bin/bash

#PBS -N gmx

#PBS -l select=1:ncpus=2:mpiprocs=2:ngpus=2:mem=14GB
#PBS -l walltime=1:00:00
#PBS -q R1660526
#PBS -W group_list=train_cmd32015
#PBS -A train_cmd32015

cd $PBS_O_WORKDIR

module load profile/advanced
module load autoload gromacs/5.0.4

export OMP_NUM_THREADS=1

mdrun=$(which mdrun_mpi_cuda)

# => change to current dir

cmd="$mdrun -s topol.tpr -v -maxh 1.0 -gpu_id 01 "

mpirun -np 2 $cmd
Gromacs 5.0.4 MPI+CUDA on Eurora

```
File Edit View Search Terminal Help

top - 18:55:58 up 191 days, 8:40, 1 user, load average: 2.63, 8.31, 11.05
Tasks: 509 total, 5 running, 504 sleeping, 0 stopped, 0 zombie
Cpu(s): 12.5%us, 0.1%sy, 0.0%ni, 87.4%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 16294372k total, 1306364k used, 14988008k free, 49428k buffers
Swap: 8191992k total, 152304k used, 8039688k free, 185300k cached

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
24907 agrottes 20 0 52.3g 106m 91m R 99.7 0.7 0:20.38 mdrun_mpi
24908 agrottes 20 0 52.2g 103m 90m R 99.4 0.7 0:20.51 mdrun_mpi
24915 agrottes 20 0 15292 1572 940 R 0.3 0.0 0:00.04 top
  1 root 20 0 19352 660 432 S 0.0 0.0 0:17.84 init
  2 root 20 0 0 0 0 S 0.0 0.0 0:00.64 kthreadd
  3 root RT 0 0 0 0 S 0.0 0.0 21:41.03 migration/0
  4 root 20 0 0 0 0 S 0.0 0.0 34:50.30 ksoftirqd/0
  5 root RT 0 0 0 0 S 0.0 0.0 0:00.01 migration/0
  6 root RT 0 0 0 0 S 0.0 0.0 0:42.24 watchdog/0
  7 root RT 0 0 0 0 S 0.0 0.0 15:43.68 migration/1
  8 root RT 0 0 0 0 S 0.0 0.0 0:00.01 migration/1
  9 root 20 0 0 0 0 S 0.0 0.0 17:55.57 ksoftirqd/1
 10 root RT 0 0 0 0 S 0.0 0.0 0:28.06 watchdog/1
 11 root RT 0 0 0 0 S 0.0 0.0 11:14.94 migration/2
 12 root RT 0 0 0 0 S 0.0 0.0 0:00.01 migration/2
 13 root 20 0 0 0 0 S 0.0 0.0 12:02.44 ksoftirqd/2
 14 root RT 0 0 0 0 S 0.0 0.0 0:18.76 watchdog/2
 15 root RT 0 0 0 0 S 0.0 0.0 16:45.64 migration/3
 16 root RT 0 0 0 0 S 0.0 0.0 0:00.01 migration/3
```
#!/bin/bash

#PBS -N gmx

#PBS -l select=1:ncpus=16:mpiprocs=2:ngpus=2:mem=14GB
#PBS -l walltime=1:00:00
#PBS -q R1660526
#PBS -W group_list=train_cmd32015
#PBS -A train_cmd32015

cd $PBS_O_WORKDIR

module load profile/advanced
module load autoload gromacs/4.6.5

export OMP_NUM_THREADS=8

mdrun=$(which mdrun_mpi_cuda)
cmd="$mdrun -s topol.tpr -v -maxh 1.0 -gpu_id 01"

mpirun -np 2 $cmd
### System Status

**Top Command Output**

- **Uptime:** 18:53:13 up 191 days, 8:37, 1 user, load average: 25.69, 13.10, 12.81
- **Tasks:** 509 total, 3 running, 506 sleeping, 0 stopped, 0 zombie
- **Cpu(s):** 95.1% us, 4.9% sy, 0.0% ni, 0.0% id, 0.0% wa, 0.0% hi, 0.0% si, 0.0% st
- **Mem:** 16294372k total, 1387360k used, 14907012k free, 49240k buffers
- **Swap:** 8191992k total, 152304k used, 8039688k free, 187840k cached

#### PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PR</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>%CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>24731</td>
<td>agrottes</td>
<td>20</td>
<td>0</td>
<td>53.4g</td>
<td>153m</td>
<td>92m</td>
<td>R</td>
<td>798.6</td>
<td>1.0</td>
<td>11:57.22</td>
<td>mdrun_mpi</td>
</tr>
<tr>
<td>24732</td>
<td>agrottes</td>
<td>20</td>
<td>0</td>
<td>53.3g</td>
<td>137m</td>
<td>92m</td>
<td>R</td>
<td>798.6</td>
<td>0.9</td>
<td>11:56.41</td>
<td>mdrun_mpi</td>
</tr>
<tr>
<td>67</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.3</td>
<td>0.0</td>
<td>128:39:59</td>
<td>events/0</td>
</tr>
<tr>
<td>75</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.3</td>
<td>0.0</td>
<td>75:53:93</td>
<td>events/8</td>
</tr>
<tr>
<td>19426</td>
<td>abartoll</td>
<td>20</td>
<td>0</td>
<td>178m</td>
<td>6592</td>
<td>1720</td>
<td>S</td>
<td>0.3</td>
<td>0.0</td>
<td>0:02:22</td>
<td>daemon_gpu_stat</td>
</tr>
<tr>
<td>1</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>19352</td>
<td>666</td>
<td>432</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0:17:84</td>
<td>init</td>
</tr>
<tr>
<td>2</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0:00:64</td>
<td>kthreadd</td>
</tr>
<tr>
<td>3</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>21:41:03</td>
<td>migration/0</td>
</tr>
<tr>
<td>4</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>34:50:30</td>
<td>ksoftirqd/0</td>
</tr>
<tr>
<td>5</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0:00:01</td>
<td>migration/0</td>
</tr>
<tr>
<td>6</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0:42:24</td>
<td>watchdog/0</td>
</tr>
<tr>
<td>7</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>15:43:68</td>
<td>migration/1</td>
</tr>
<tr>
<td>8</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0:00:01</td>
<td>migration/1</td>
</tr>
<tr>
<td>9</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>17:55:57</td>
<td>ksoftirqd/1</td>
</tr>
<tr>
<td>10</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0:28:06</td>
<td>watchdog/1</td>
</tr>
<tr>
<td>11</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>11:14:94</td>
<td>migration/2</td>
</tr>
<tr>
<td>12</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0:00:01</td>
<td>migration/2</td>
</tr>
<tr>
<td>13</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>12:02:43</td>
<td>ksoftirqd/2</td>
</tr>
<tr>
<td>14</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0:18:76</td>
<td>watchdog/2</td>
</tr>
</tbody>
</table>
MD Performance on hybrid CPU-GPU clusters

Case study: small peptide in water

- Pure MPI (12 MPI procs) → 123.9 ns/day
- MPI-CUDA (2 MPI procs + 2 GPUs) → 148.0 ns/day
- MPI/OpenMP/CUDA (2 MPI procs + 6 threads + 2 GPUs) → 253.0 ns/day
- MPI + Intel-Phi (12 MPI procs + 34 threads) → 134.1 ns/day

Small peptide in a box of water, ~3300 atoms
Gromacs 4.6.5 with GPU
PME for long electrostatics, 1 nm cut-off, T = 300 K
MD Performance on hybrid CPU-GPU clusters
multiple MPI ranks with 2 GPUs

Small peptide in a box of water, ~3300 atoms
Gromacs 4.6.5 with GPU
PME for long electrostatics, 1 nm cut-off, T = 300 K
MD Performance on hybrid CPU-GPU clusters (Eurora)

**Case study: membrane protein**

ATP/ADP Mitochondrial Carrier,
92K atoms
Gromacs 5.0.4 with GPU
PME for long electrostatics, 300 K,
Cut-off = 1 nm
Domain Decomposition

- Pure MPI (16 MPI procs) → 11.6 ns/day
- MPI-CUDA (2 MPI procs + 2 GPUs) → 9.5 ns/day
- MPI/OpenMP/CUDA (2 MPI procs + 8 threads + 2 GPUs) → 24.6 ns/day
- MPI + Intel Phi (8 MPI procs + 34 threads) → 14.6 ns/day
MD Performance on hybrid CPU-GPU clusters (Eurora)

ATP/ADP Mitochondrial Carrier, 92K atoms
Gromacs 5.0.4 with GPU
PME for long electrostatics, 300 K,
Cut-off = 1 nm
Domain Decomposition

- Pure MPI (16 MPI) → 11.6 ns/day
- MPI-CUDA (2 MPI procs + 2 GPUs) → 9.5 ns/day
- MPI-CUDA (4 MPI procs + 2 GPUs) → 14.7 ns/day
- MPI-CUDA (8 MPI procs + 2 GPUs) → 22.2 ns/day
- MPI-CUDA (16 MPI + 2 GPUs) → 27.9 ns/day
- MPI-CUDA (8 MPI procs + 2 OpenMP + 2 GPUs) → 29.2 ns/day
Speed up analysis
pure MPI job

ATP-Carrier

Parallel Efficiency with 32 MPI procs = 57.2 %
MD Performance on hybrid CPU-GPU clusters (Eurora)

![Graph showing performance vs number of MPI processes.](image)

- **Performance (ns/day)**
- **nr. of MPI PROCESSES**

- **Pure MPI**

**ATP-Carrier**
## GPU acceleration in GROMACS (Galileo)

<table>
<thead>
<tr>
<th></th>
<th>Small peptide (3K atoms)</th>
<th>Membrane protein (92K atoms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pure MPI</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MPI-CUDA</td>
<td>1.2x</td>
<td>0.8x</td>
</tr>
<tr>
<td>multiple MPI ranks/CUDA</td>
<td>2.5x</td>
<td>2.5x</td>
</tr>
<tr>
<td>MPI-OpenMP/CUDA</td>
<td>2.0x</td>
<td>2.1x</td>
</tr>
<tr>
<td>Intel Phi</td>
<td>1.1x</td>
<td>1.3x</td>
</tr>
</tbody>
</table>
MD Optimization on hybrid CPU-GPU Clusters

- System size and composition:
- Choice of PME vs. Cut-off based electrostatics
- Larger cut-off radius means a larger Verlet list ==> GPU is better than CPU
- Pure MPI jobs are suitable for small-sized systems
The idea behind the native GPU acceleration in GROMACS is that we offload the heavy nonbonded force calculation to an accelerator (either a GPU or Xeon Phi), while the CPU does bonded forces and lattice summation (PME) in the mean time.