CINECA SCAI
SuperComputing Applications and Innovation

Running MD on HPC architectures I. Hybrid Clusters

Alessandro Grottesi
Cineca

PRACE CINECA

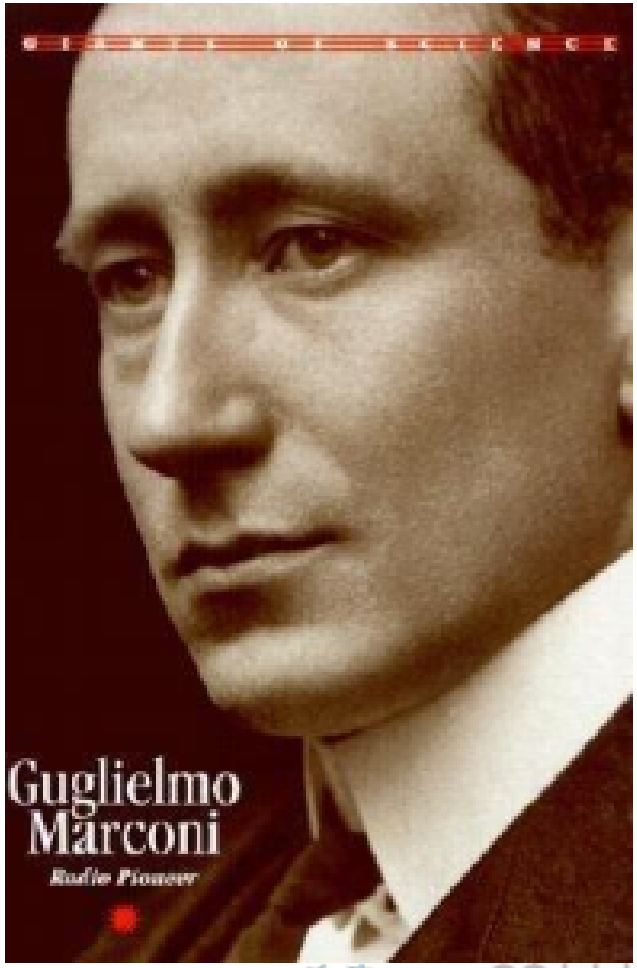
Today's lecture

You will learn:

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

Marconi

- NeXtScale architecture
- nx360M5 nodes:
  - Supporting Intel HSW & BDW
  - Able to host both IB network Mellanox EDR & Intel Omni-Path
  - Twelve nodes are grouped into a Chassis (6 chassis per rack)
- The compute node is made of:
  - 2 x Intel Broadwell (Xeon processor E5-2697 v4)
  - 18 cores, 2,3 HGz
  - 8 x 16GB DIMM memory (RAM DDR4 2400 MHz), 128 GB total
  - 1 x 129 GB SATA MLC S3500 Enterprise Value SSD
  - 1 x link OPA 100GBs
  - 2*18*2,3*16 = 1.325 GFs peak
- The login node (10 servers) – x3550M5 1U
  - 2 x Intel Broadwell (Xeon processor E5-2697 v4)
  - 18 cores, 2,3 HGz
  - 8 x 16GB DIMM memory (RAM DDR4 2400 MHz), 128 GB total
  - 2 x 1 TB SAS
  - 1 x link OPA 100GBs + 1 link 1GbE + 2 link 10GbE
- 24 rack in total:
  - 21 rack ☾ compute
  - 1 rack ☾ service nodes
  - 2 racks ☾ core switch



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

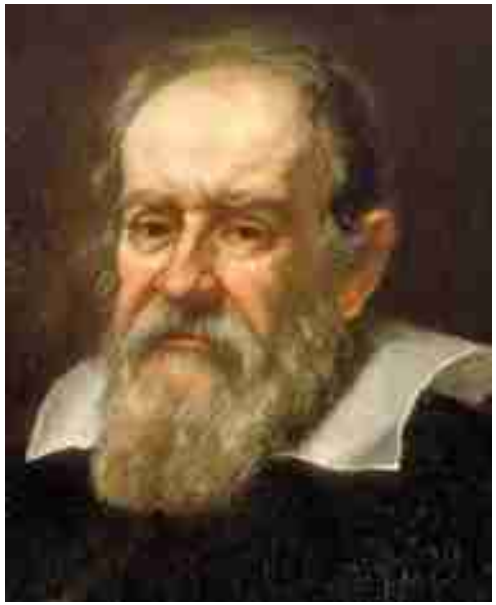
Accelerators: 2 Intel Phi 7120p per node on 384 nodes (768 in total)

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 (69th on TOP500)



Pico

Model: IBM NeXtScale server

Architecture: Linux Infiniband Cluster

Processors Type: Intel Xeon (Ten-Core) E5-2670v2 2.50 GHz (Compute)

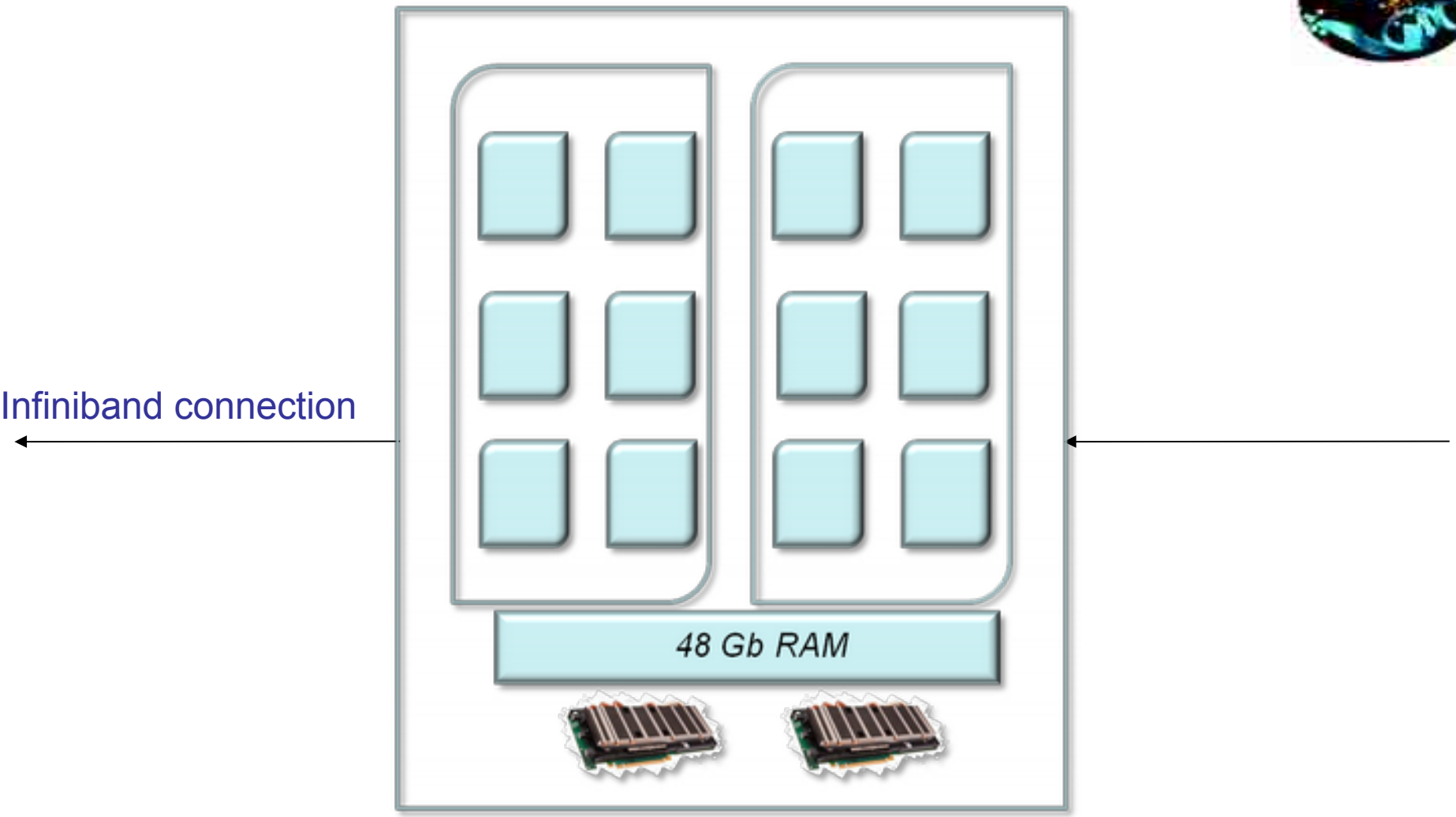
Number of nodes: 54 Compute + 4 visualization + 2 Login + 14 other

Number of cores: 1080 (compute)

Number of accelerators: 4 + 2 (only on viz nodes)

RAM: 128 GB/Compute node (2 viz nodes with 512GB)

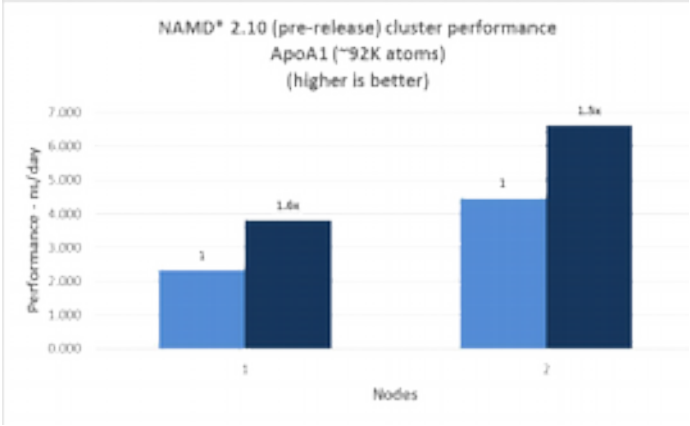
Compute nodes



Molecular Dynamics and accelerators

Intel Xeon Phi

Nvidia K80

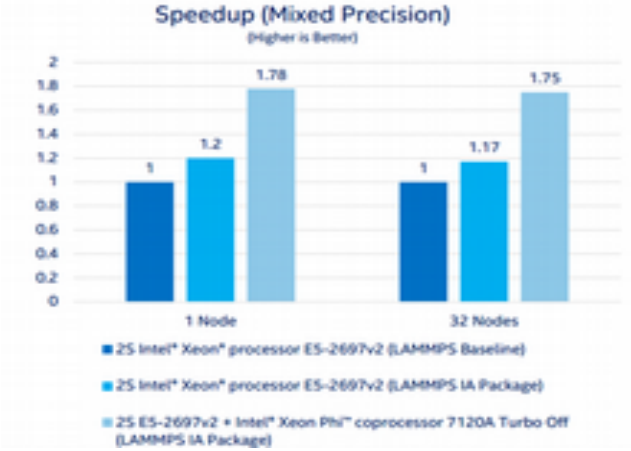


– GROMACS
- Under development, currently (Jan 2015) only native-mode version available.

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

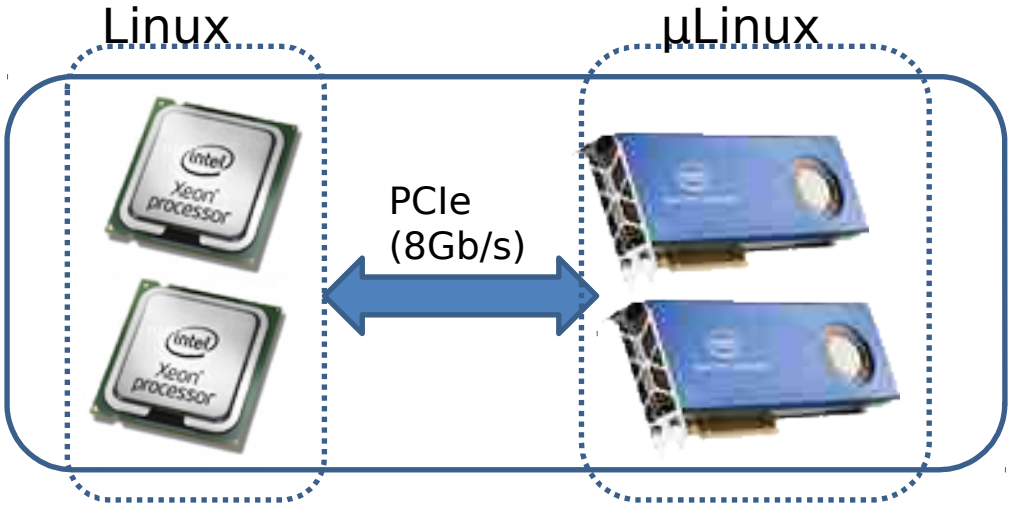
– LAMMPS
- Xeon PHI support available in current downloads for non-bonded calculations. Reported speed-ups of about 1.78x compared to non-accelerated code (one coprocessor/node) for Rhodopsin benchmark. Higher speed-ups obtained with materials simulations.

– AMBER
- Xeon PHI-enabled version released 6 Aug 2014. Waiting for benchmarks.



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.

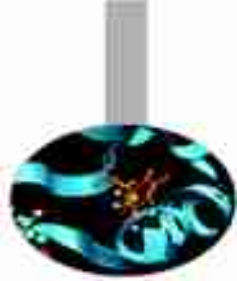
Classical Molecular Dynamics and Intel Xeon PHI

- Development some way behind GPU-CUDA versions of classical MD programs (which started a few years 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):

https://software.intel.com/en-us/articles/namd-for-intel-xeon-phi-coprocessor

Running MD code on Galileo

What for...?



1. Minimizzation
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...

CINECA

Available forcefield in Gromacs (5.x)

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 (CHARM22 plus CMAP for proteins)
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)
14: GROMOS96 54a7 force field (Eur. Biophys. J. (2011), 40, 843-856
15: OPLS-AA/L all-atom force field (2001 aminoacid dihedrals)

Generate topology (Gromacs)

Gromacs:

```
gmx pdb2gmx –f input_file.pdb –ignh –ter
```

input:

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

output:

1. topol.top — system topology
2. posre.itp — position restraints file
3. conf.gro — coordinate file (gro format by default)
4. topolA.itp, topolB.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 (Gromacs)

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:

```
gmx editconf –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 (Gromacs)

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

```
gmx solvate –cp conf.gro –cs spc216.gro –o out.gro -p topol.top
```

Ionic strength: genion

Grid based electrostatic treatment (Ewald sums, PME, etc.) are better performed with system net charge = 0. Namely, make sure that:

solute charge + 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:

```
gmx genion –s topol.tpr –seed XXX –o oution.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 neeed to run an MD simulation.

```
gmx 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:

```
gmx dump –s topol.tpr
```

Output control

Van der Waals and electrostatics

Temperature and pressure coupling

```
title           = Yo
cpp             = cpp
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
```

Output control parameters

nsteps	= 500000	Total number of steps
nstxout	= 5000	coords output frequency for traj.trr
nstvout	= 5000	velocity output frequency for traj.trr
nstlog	= 5000	output frequency for log file (md.log)
nstenergy	= 250	output frequency for energy file ener.edr
nstxtcout	= 250	coords output frequency for traj.xtc
xtc_grps	= Protein	content of file traj.xtc
energygrps	= Protein SOL	energy groups to store in file ener.edr

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.

NAMD input file


```
# standard NAMD benchmark 92k atoms
cellBasisVector1     108.8612 0.0 0.0
cellBasisVector2     0.0 108.8612 0.0
cellBasisVector3     0.0 0.0 77.758
cellOrigin           0.0 0.0 0.0

coordinates          apoa1.pdb
temperature          300
seed                 74269

switching            on
switchdist           10
cutoff               12
pairlistdist         13.5
margin               0
stepspercycle        20

# electrostatics
PME                  on
PMEGridSizeX         108
PMEGridSizeY         108
PMEGridSizeZ         80
```

```
structure            apoa1.psf
parameters           par_all22_prot_lipid.xplor
parameters           par_all22_popc.xplor
exclude              scaled1-4
1-4scaling           1.0

timestep             1.0
fullElectFrequency   4

numsteps             10000
outputEnergies       200
outputtiming         200

outputname           apoa1
```

Amber input file

```
Typical Production MD NVT
&cntrl
  ntx=5, irest=1,
  ntc=2, ntf=2,
  nstlim=2000,
  ntpr=1000, ntwx=1000,
  ntwr=2000,
  dt=0.002, cut=8.,
  ntt=1, tautp=10.0,
  temp0=300.0,
  ntb=2, ntp=1, barostat=2,
  ioutfm=1,
/
```

Run the simulation: Gromacs (serial)

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

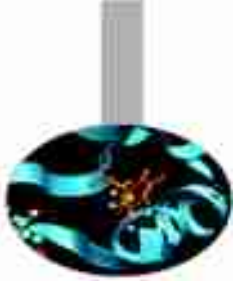
domain decomposition

particle decomposition

The command generates many outpuf filed 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 — chekpoint file for restarting runs.
6. md.log — log file with output control

Run the simulation: NAMD (parallel)



To run NAMD in a pure MPI job:

```
mpirun -np 16 namd2 config.namd > namd.log
```

To run NAMD in a Intel XeonPhi job:

```
mpirun -np 16 namd2.mic config.namd > namd.log
```

To run NAMD on a GPU

```
mpirun -np 16 namd2.cuda -idlepoll +devices 0,1config.namd > namd.log
```

CINECA

Run the simulation: Amber

```
mpirun -np 16 pmemd.MPI -O -i run_md.CPU -o dimero_water_SCP_

mpirun -np 16 pmemd.cuda.MPI -O -i run_md.GPU -o dimero_water_SCP_

mpirun -np 16 pmemd.mic_offload.MPI -O -i run_md.MIC -o dimero_water_SCP_
```

What if it all crashes...?

A .cpt file is produced by mdrun at specified intervals (mdrun -cpt n, where n is the interval in minutes), and contains information on all the state variables in a simulated system. In the case of a crash (hardware failure, power outage, etc), a checkpoint file can be used to resume the simulations exactly as it was before the failure. Simulations can also be extended using a checkpoint file (www.gromacs.org).

```
mdrun -s topol.tpr -cpi state.cpt
```

```
mdrun -s topol.tpr -cpi state.cpt -append
```

Write down coordinates on previous generated files

Gromacs 5.1.2, pure MPI

```
#PBS -N GMX-PureMPI
#PBS -l select=1:ncpus=16:mpiprocs=16:mem=120GB
#PBS -q R1619976
#PBS -l walltime=1:00:00
#PBS -A train_cmdB2016
#PBS -W group_list=train_cmdB2016

cd $PBS_O_WORKDIR                  ==> change to current dir

module load profile/advanced
module load autoload gromacs/5.1.2

export OMP_NUM_THREADS=1           ==> set nr. Of OpenMP threads to 1 per node

mdrun=$(which mdrun_mpi)
cmd="$mdrun -s topol.tpr -v -maxh 1.0 -nb cpu"
mpirun -np 16 $cmd
```

Gromacs 5.1.2, pure MPI

```
File  Edit  View  Search  Terminal  Help
top - 18:59:29 up 191 days,  8:44,  1 user,  load average: 6.03, 6.58, 9.76
Tasks: 521 total,  17 running, 504 sleeping,   0 stopped,   0 zombie
Cpu(s): 99.1%us,  0.9%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  16294372k total,  1183416k used, 15110956k free,    49656k buffers
Swap:  8191992k total,   152304k used,  8039688k free,   184768k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
25080 agrottes  20   0  192m  15m  12m R 100.0  0.1   0:16.71 mdrun_mpi
25082 agrottes  20   0  192m  15m  12m R 100.0  0.1   0:16.71 mdrun_mpi
25083 agrottes  20   0  192m  16m  13m R 100.0  0.1   0:16.72 mdrun_mpi
25088 agrottes  20   0  192m  15m  12m R 100.0  0.1   0:16.72 mdrun_mpi
25089 agrottes  20   0  192m  15m  12m R 100.0  0.1   0:16.73 mdrun_mpi
25091 agrottes  20   0  192m  15m  12m R 100.0  0.1   0:16.72 mdrun_mpi
25093 agrottes  20   0  192m  15m  12m R 100.0  0.1   0:16.74 mdrun_mpi
25094 agrottes  20   0  192m  15m  12m R 100.0  0.1   0:16.72 mdrun_mpi
25079 agrottes  20   0 52.2g  27m  23m R 99.7  0.2   0:16.57 mdrun_mpi
25081 agrottes  20   0  192m  15m  12m R 99.7  0.1   0:16.72 mdrun_mpi
25084 agrottes  20   0  192m  15m  12m R 99.7  0.1   0:16.71 mdrun_mpi
25085 agrottes  20   0  192m  15m  12m R 99.7  0.1   0:16.71 mdrun_mpi
25090 agrottes  20   0  192m  14m  11m R 99.7  0.1   0:16.67 mdrun_mpi
25092 agrottes  20   0  192m  17m  12m R 99.7  0.1   0:16.71 mdrun_mpi
25086 agrottes  20   0  192m  15m  12m R 99.3  0.1   0:16.72 mdrun_mpi
25087 agrottes  20   0  192m  15m  12m R 99.3  0.1   0:16.69 mdrun_mpi
 3923 root       0 -20 8811m 320m  45m S  0.3  2.0   1244:58 mmfsd
19541 abartoli  20   0  172m 5672 1048 S  0.3  0.0   0:00.87 deamon_mic_stat
25120 agrottes  20   0 15292 1576  940 R  0.3  0.0   0:00.04 top
```

Gromacs 5.1.2 MPI+CUDA

```
#!/bin/bash
#PBS -N gmx
#PBS -l select=1:ncpus=2:mpiprocs=2:ngpus=2:mem=15GB
#PBS -l walltime=1:00:00
#PBS -q R1619976
#PBS -A train_cmdB2016
#PBS -W group_list=train_cmdB2016

cd $PBS_O_WORKDIR                    ==> change to current dir

module load profile/advanced
module load autoload cuda/6.5.14
module load gromacs/5.1.2

export OMP_NUM_THREADS=1        ==> set nr. Of OpenMP threads to 1
#                               ==> set total MPI tasks = 2 and bind to two GPUs

mdrun=$(which mdrun_mpi_cuda)
cmd="$mdrun -s topol.tpr -v -maxh 1.0 -gpu_id 01 "
mpirun -np 2 $cmd
```

Gromacs 5.1.2 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  10:45.64 migration/3
   16 root      RT   0     0    0    0 S  0.0  0.0   0:00.01 migration/3
```

Gromacs 5.1.2 MPI/OpenMP+CUDA

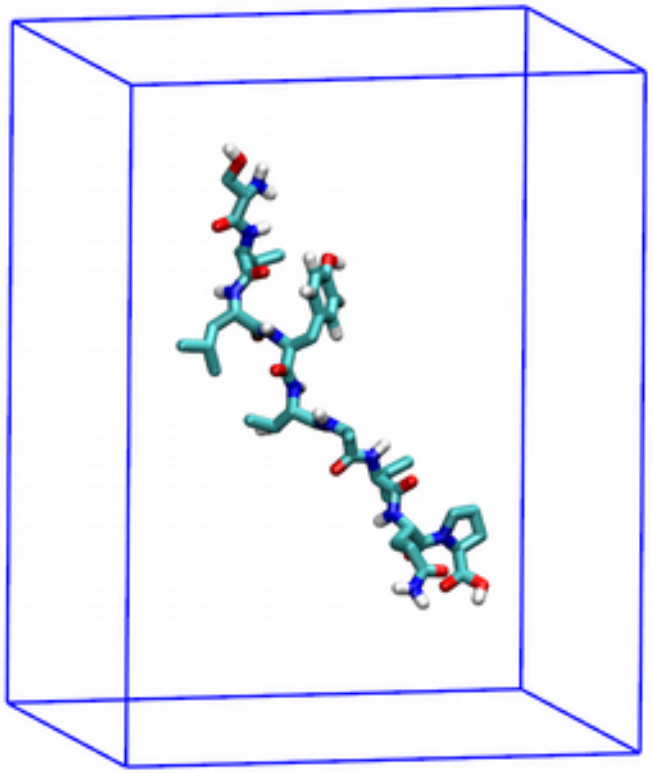
```
#!/bin/bash
#PBS -N gmx
#PBS -l select=1:ncpus=16:mpiprocs=2:ngpus=2:mem=120GB
#PBS -l walltime=1:00:00
#PBS -q R1619980
#PBS -W group_list=train_cmdB2016
#PBS -A train_cmdB2016


cd $PBS_O_WORKDIR                    ==> change to current dir

module load profile/advanced
module load autoload gromacs/5.1.2

export OMP_NUM_THREADS=8             ==> set nr. Of OpenMP threads to 8
#                                    ==> set 2 MPI tasks that bind to two GPUs

mdrun=$(which mdrun_mpi_cuda)
cmd="$mdrun -s topol.tpr -v -maxh 1.0 -gpu_id 01 "
mpirun -np 2 $cmd
```

Gromacs 5.1.2 MPI/OpenMP+CUDA

```
File  Edit  View  Search  Terminal  Help
top - 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
24731 agrottes  20   0 53.4g 153m  92m R 798.6  1.0  11:57.22 mdrun_mpi
24732 agrottes  20   0 53.3g 137m  92m R 798.6  0.9  11:56.41 mdrun_mpi
   67 root      20   0     0    0    0 S  0.3  0.0 128:39.59 events/0
   75 root      20   0     0    0    0 S  0.3  0.0  75:53.93 events/8
19426 abartoli  20   0  178m 6692 1720 S  0.3  0.0   0:02.22 deamon_gpu_stat
    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  12:02.43 ksoftirqd/2
   14 root      RT   0     0    0    0 S  0.0  0.0   0:18.76 watchdog/2
```

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

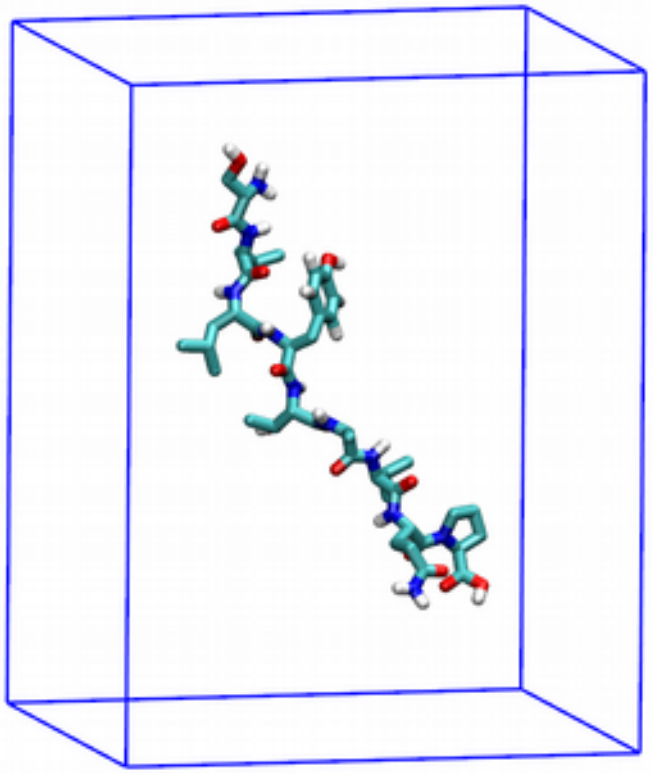
Small peptide in a box of water, ~3300 atoms

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: small peptide in water



- Pure MPI (16 MPI procs) →284.1 ns/day
- MPI-CUDA (2 MPI procs + 2 GPUs) →258.3 ns/day
- MPI/OpenMP/CUDA →457.1 ns/day
  (2 MPI procs + 8 threads + 2 GPUs)
- Intel-Phi (native mode) → 68.6/day
  (16 MPI procs + 34 threads)

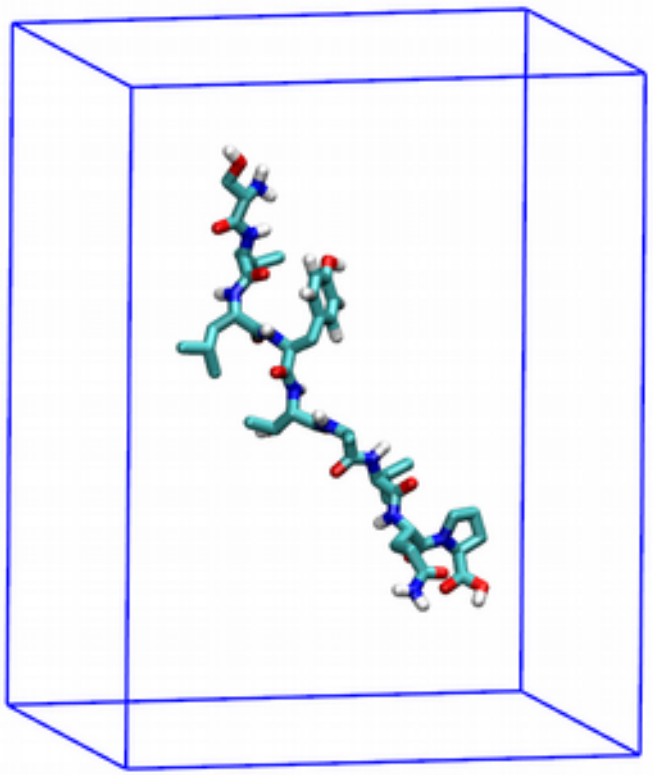
Small peptide in a box of water, ~3300 atoms

Gromacs 5.1.2 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



- MPI-CUDA (2 MPI procs + 2 GPUs) → 148.0 ns/day
- MPI-CUDA (4 MPI procs + 2GPUs) → 304.1 ns/day
- MPI-CUDA (8 MPI procs + 2 GPUs) → 221.5ns/day
- MPI-CUDA (8MPI procs + 2 thrds + 2GPUs) →not applicable
- MPI-CUDA (16MPI + 2 GPUs) → not applicable

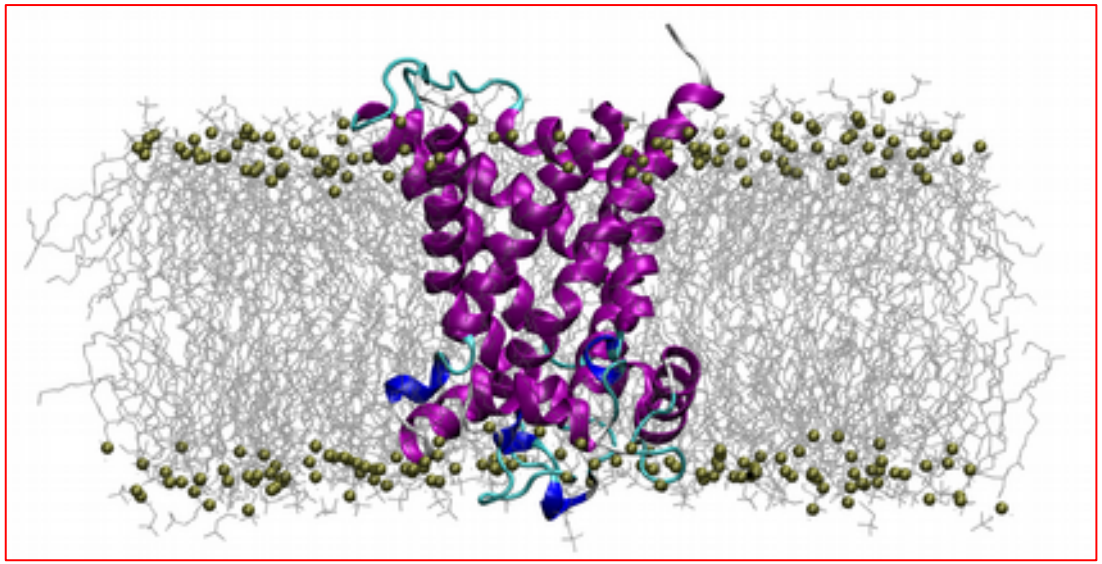
Small peptide in a box of water, ~3300 atoms

Gromacs 4.6.5 with GPU

PME for long electrostatics, 1 nm cut-off, T = 300 K

MD Performance on hybrid CPU-GPU clusters

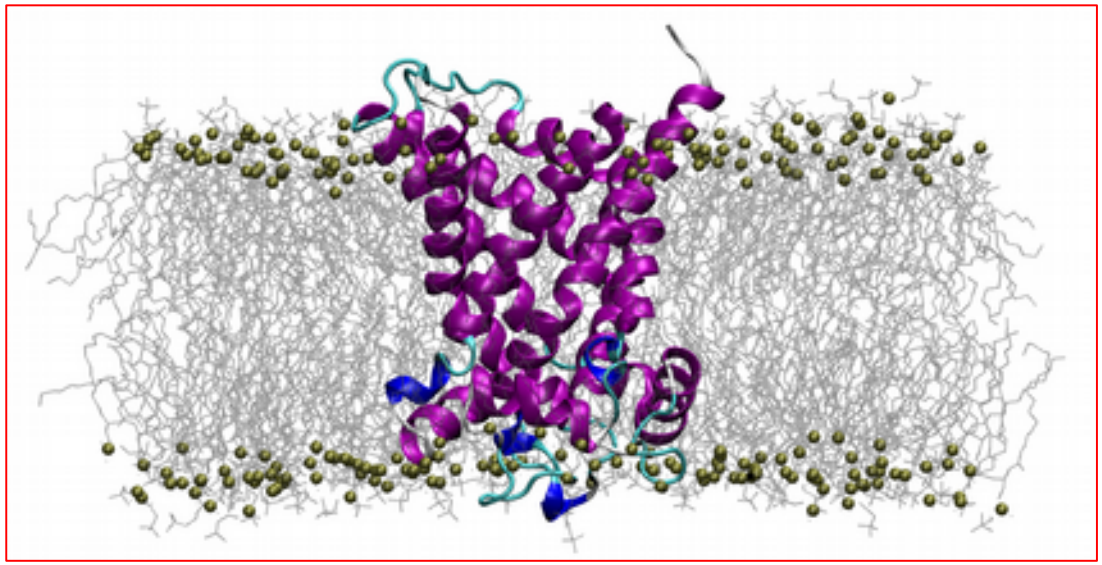
Case study: membrane protein



ATP/ADP Mitochondrial Carrier,
92K atoms
Gromacs 5.0.4 with GPU/MIC
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) → 9.4 ns/day
- MPI/OpenMP/CUDA (2 MPI procs + 8 threads + 2 GPUs) →25.9 ns/day
- MPI + Intel Phi (8 MPI procs + 34 threads, Galileo) →14.6 ns/day

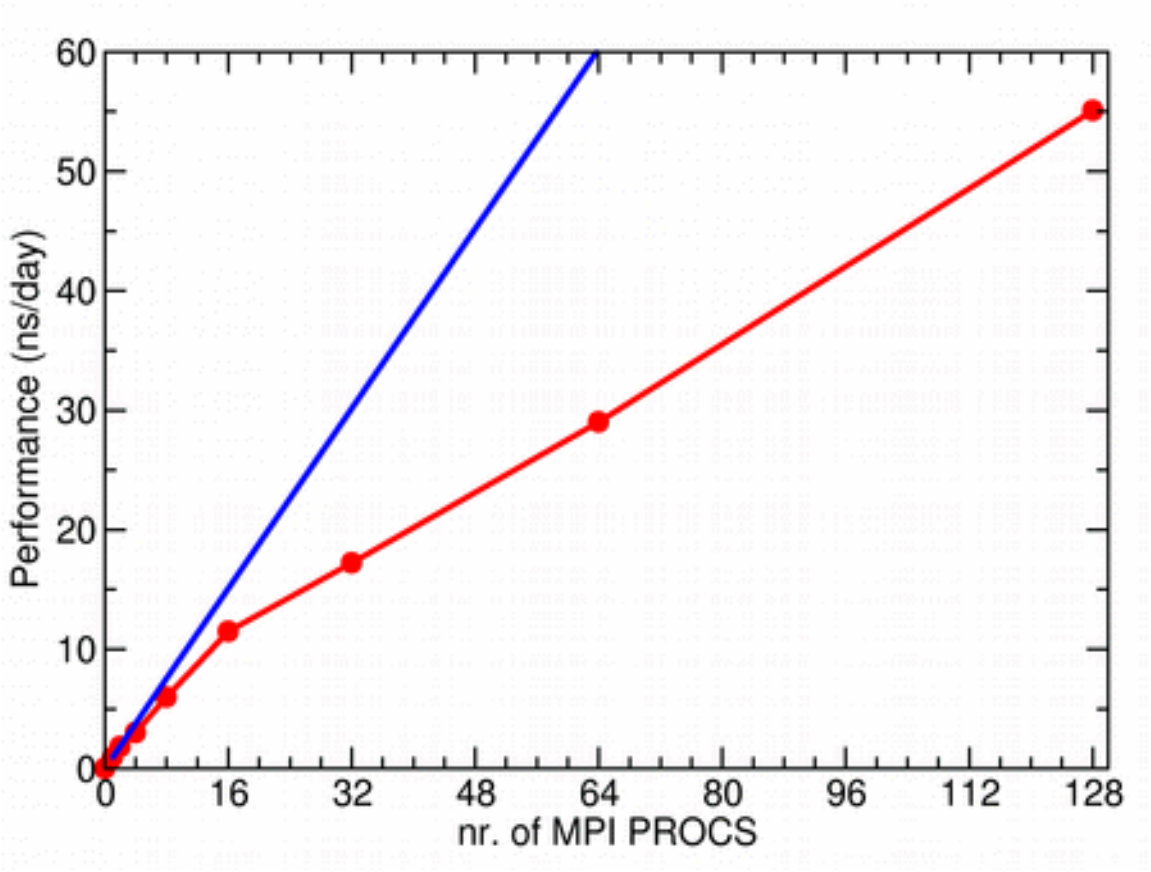
MD Performance on hybrid CPU-GPU clusters

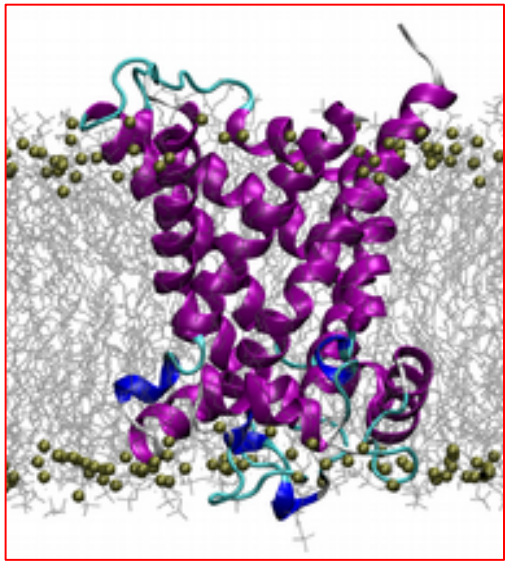


ATP/ADP Mitochondrial Carrier,
92K atoms
Gromacs 5.1.2 with GPU
PME for long electrostatics, 300 K,
Cut-off = 1 nm
Domain Decomposition

- Pure MPI (16 MPI) →11.6 ns/day
- MPI-CUDA (2 MPI procs + 2 GPUs) → 9.4 ns/day
- MPI-CUDA (4 MPI procs + 2GPUs) → 15.2 ns/day
- MPI-CUDA (8 MPI procs + 2 GPUs) → 23.0 ns/day
- MPI-CUDA (16MPI + 2 GPUs) → 28.9 ns/day
- MPI-CUDA (8MPI procs + 2 OpenMP + 2GPUs) → 30.0 ns/day

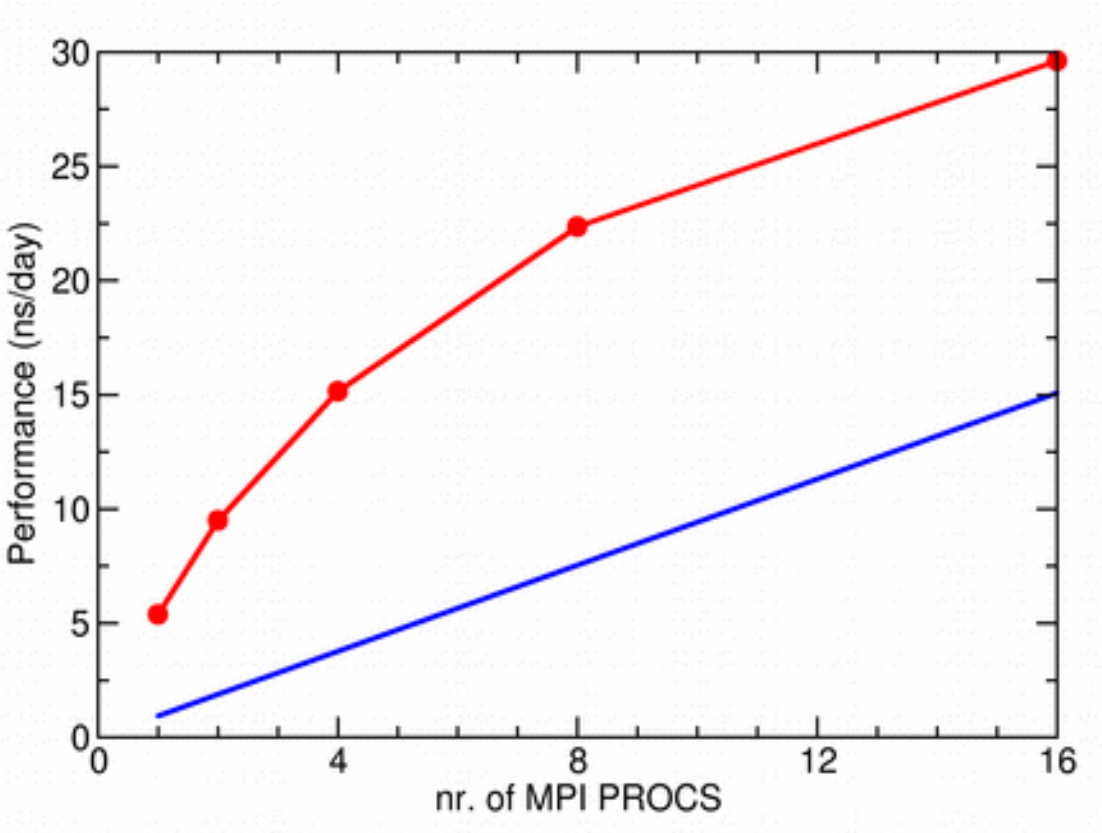
Speed up plot for pure MPI calculation

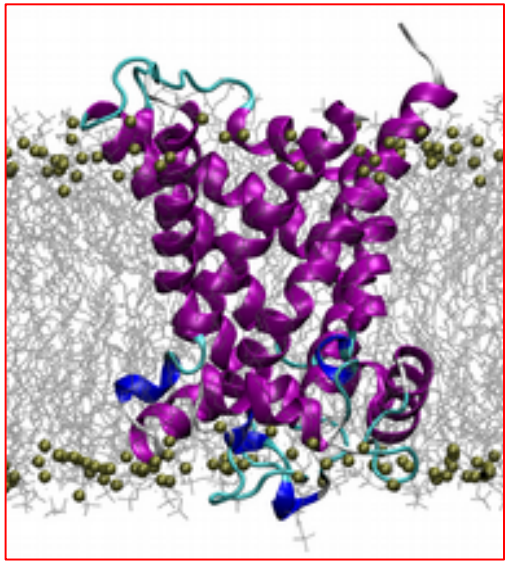




ATP/ADP Mitochondrial Carrier

MD Performance on hybrid CPU-GPU clusters (Eurora)





ATP/ADP Mitochondrial Carrier

GPU acceleration in GROMACS

	Small peptide (3K atoms)	Membrane protein (92K atoms)
Pure MPI	1	1
MPI-CUDA	1.2x	0.8x
multiple MPI ranks/CUDA	2.5x	2.5x
MPI-OpenMP/CUDA	2.0x	2.1x
Intel Phi	1.1x	1.3x

MIC acceleration in NAMD

```
#!/bin/bash
#PBS -S /bin/bash
#PBS -N namd
#PBS -l select=1:ncpus=16:mpiprocs=16:nmics=2:mem=14GB
#PBS -l walltime=00:30:00
#PBS -e job.err
#PBS -o job.out
#PBS -q parallel
#PBS -A train_cmdR2016_0
#PBS -W group_list=train_cmdR2016_0

start_time=$(date +"%s")
cd ${PBS_O_WORKDIR}

module load profile/advanced
module load autoload namd

source $INTEL_HOME/bin/compilervars.sh intel64
mpirun -n 16 namd2.mic md.namd
```

Acceleration in NAMD

(Galileo)

	Apoa1 (3000 atoms) NAMD 2.10	Speed up
Pure MPI	1.43 ns/day	1
MPI-CUDA	4.1 ns/day	2.8
Intel Phi	4.3 ns/day	3.0

MIC acceleration in Amber

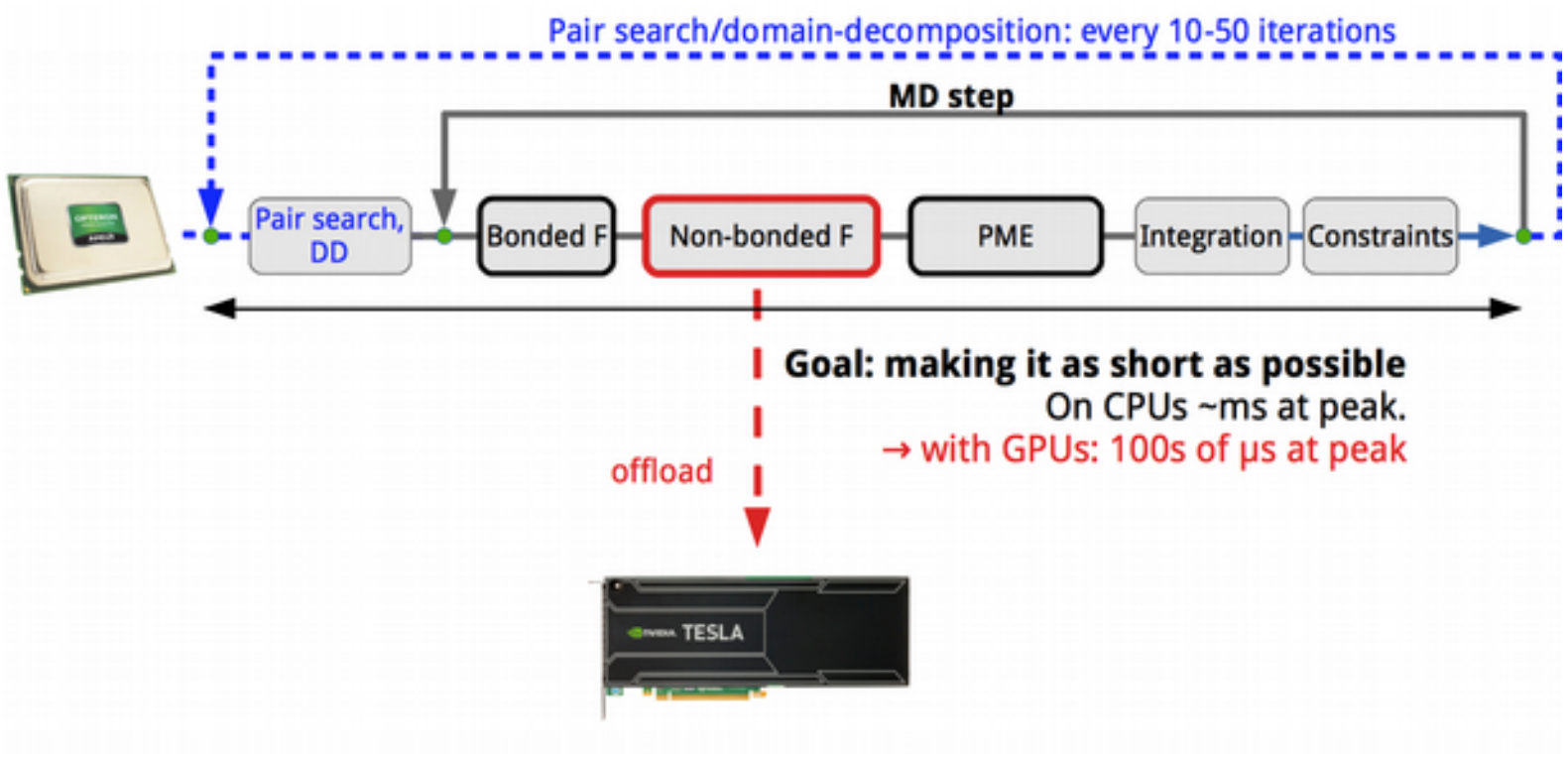
```
#!/bin/bash
#PBS -N Amber
#PBS -l select=1:ncpus=16:mpiprocs=16:nmics=1:mem=14GB
#PBS -q parallel
#PBS -l walltime=1:00:00
#PBS -A train_cmdR2016
#PBS -W group_list=train_cmdR2016

cd $PBS_O_WORKDIR
module load autoload amber/14
exe=$AMBER_HOME/bin/pmemd.mic_offload.MPI
source $INTEL_HOME/bin/compilervars.sh intel64

 mpirun -np 16 $exe -O -i mdin.CPU -o mdout-offload -p prmtop
-c inpcrd
```

MD Optimizzation 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

GPU acceleration in GROMACS

www.gromacs.org



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.

Advanced MD Script for Gromacs (I)

To address the bottleneck caused by multi-threading inefficiencies, it can be advantageous to reduce the number of OpenMP threads per rank. However, to not leave cores empty, this requires using more MPI ranks, hence more PP ranks, and therefore ranks will have to **share GPUs**. GPU sharing is possible by passing a GPU ID to mdrun multiple times, e.g -gpu_id 0011 will allow the first two PP ranks in a compute node to use GPU0 and the third and fourth GPU1.

Example #1:

```
#PBS -l select=1:ncpus=8:mpiprocs=8:ngpus=2:mem=14GB
…
OMP_NUM_THREADS=1
…
mpirun -np 8 mdrun_mpi_cuda -s topol.tpr -maxh 1.0 -deffnm test -gpu_id 00001111
```

Example #2:

```
#PBS -l select=1:ncpus=16:mpiprocs=16:ngpus=2:mem=14GB
…
OMP_NUM_THREADS=1
…
mpirun -np 16 mdrun_mpi_cuda -s topol.tpr -maxh 1.0 -deffnm test -gpu_id 0000000011111111
```