

Applications' usage in HPC2N

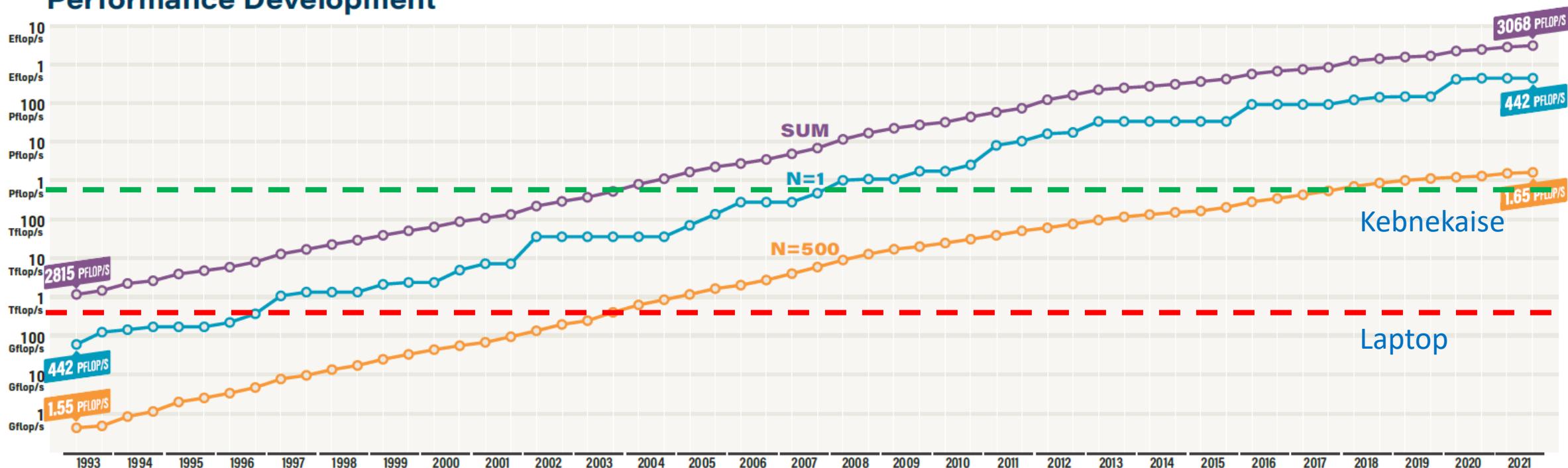
P. Ojeda, B. Brydsö

Top500 list

NOVEMBER 2021

				SITE	COUNTRY	CORES	R _{MAX} PFLOP/S	POWER MW
1	Fugaku	Fujitsu A64FX (48C, 2.2GHz), Tofu Interconnect D		RIKEN R-CCS	Japan	7,630,848	442.0	29.9
2	Summit	IBM POWER9 (22C, 3.07GHz), NVIDIA Volta GV100 (80C), Dual-Rail Mellanox EDR Infiniband		DOE/SC/ORNL	USA	2,414,592	148.6	10.1
3	Sierra	IBM POWER9 (22C, 3.1GHz), NVIDIA Tesla V100 (80C), Dual-Rail Mellanox EDR Infiniband		DOE/NNSA/LLNL	USA	1,572,480	94.6	7.44
4	Sunway TaihuLight	Shenwei SW26010 (260C, 1.45 GHz) Custom Interconnect		NSCC in Wuxi	China	10,649,600	93.0	15.4
5	Perlmutter	HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10 (274 GB)		LBNL	USA	761,856	70.9	2.58

Performance Development



Why do we need HPC?

Scientific codes use loops extensively:

```
do j=1,N  
  do i=1,N  
    do k=1,N  
      p(i,j) = p(i,j) + v1(i,k)* v2(k,j)  
    enddo  
  enddo  
enddo
```

Heavy loops
MD, QM, CFD, ...

```
!$omp parallel do private(i,j,k) shared(p,v1,v2) collapse(2)  
  do j=1,N  
    do i=1,N  
      s=0.0d0  
      do k=1,N  
        s = s + v1(i,k)* v2(k,j)  
      enddo  
      p(i,j) = s  
    enddo  
  enddo  
 !$omp end parallel do
```

Parallelized loops

Processes communication

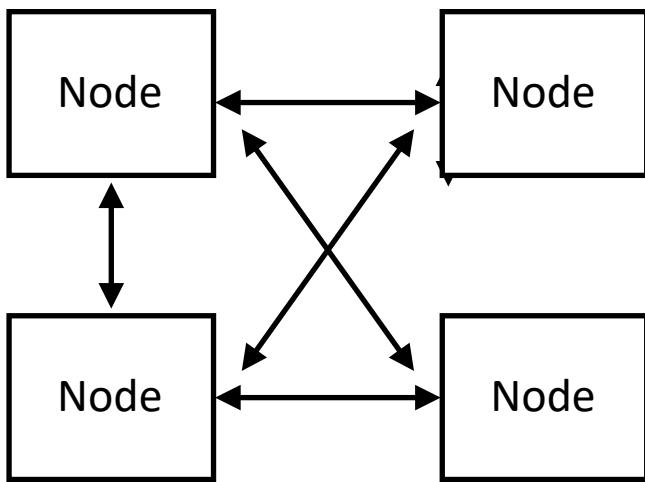


Figure : Nodes.

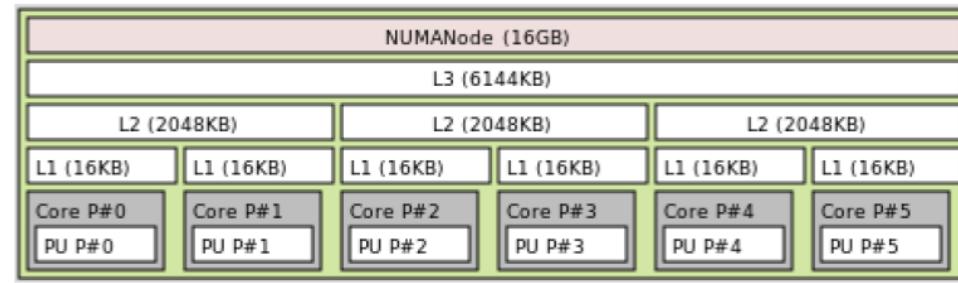
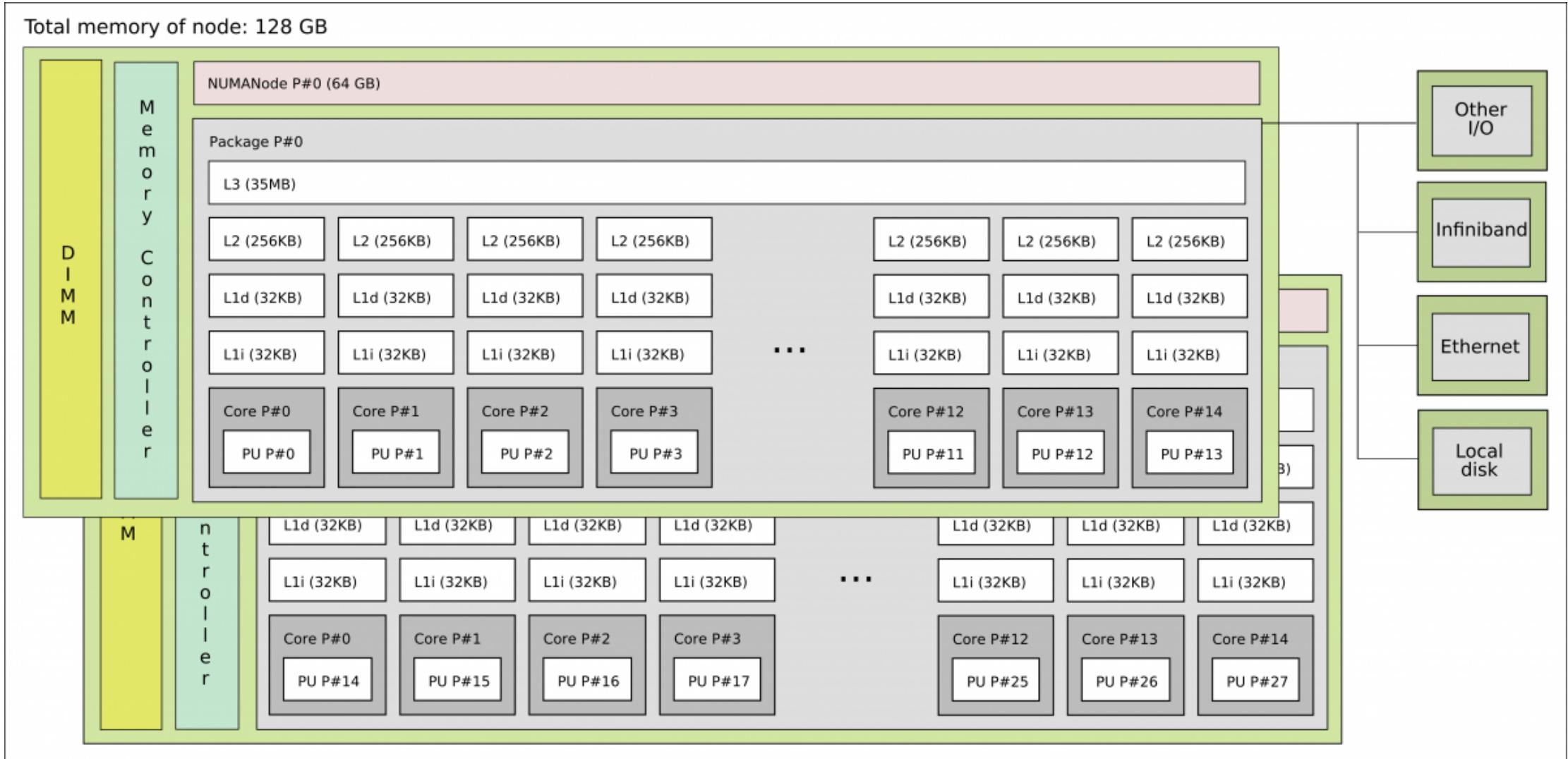


Figure : NUMA machine.

$$T_p = \frac{T_s}{P} + T_{comm}$$

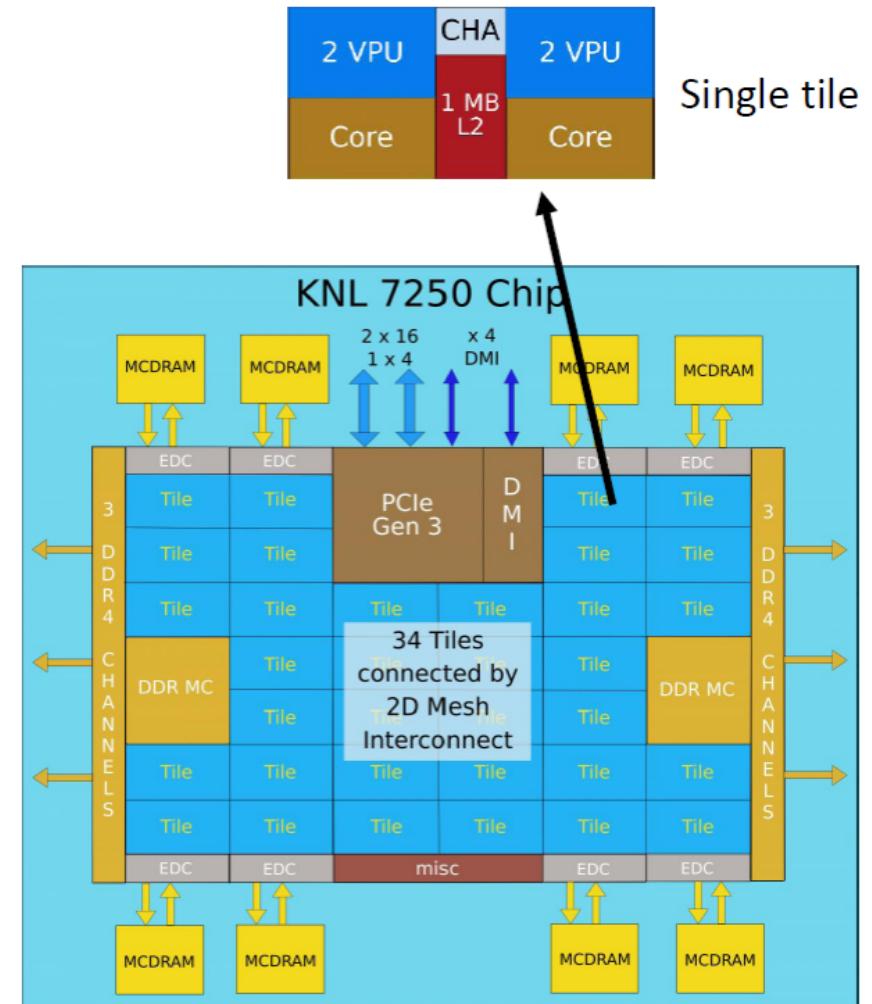
Broadwell node on Kebnekaise



Accelerators



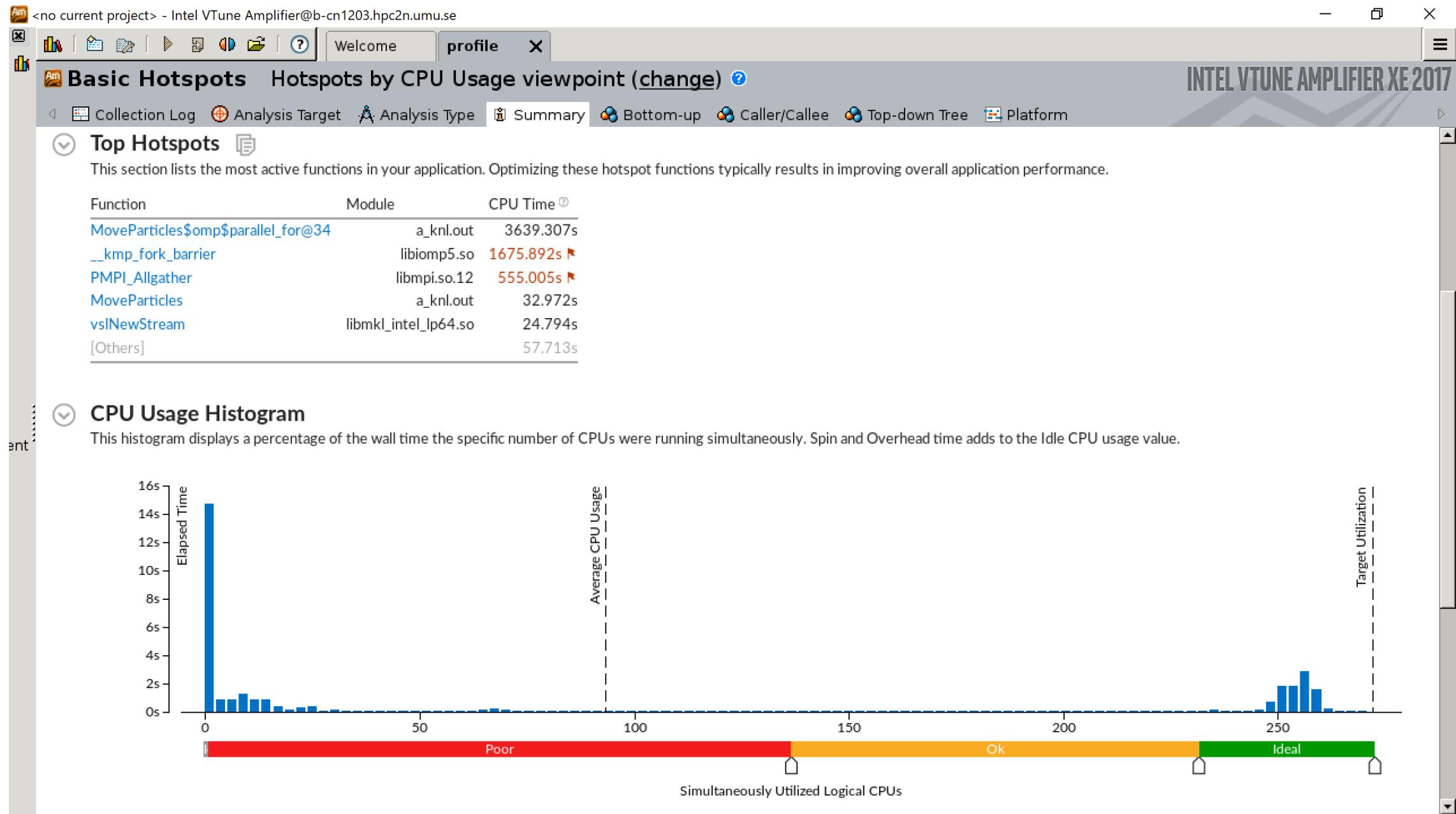
GPU showing the independent units
Streaming Multiprocessors (SM).



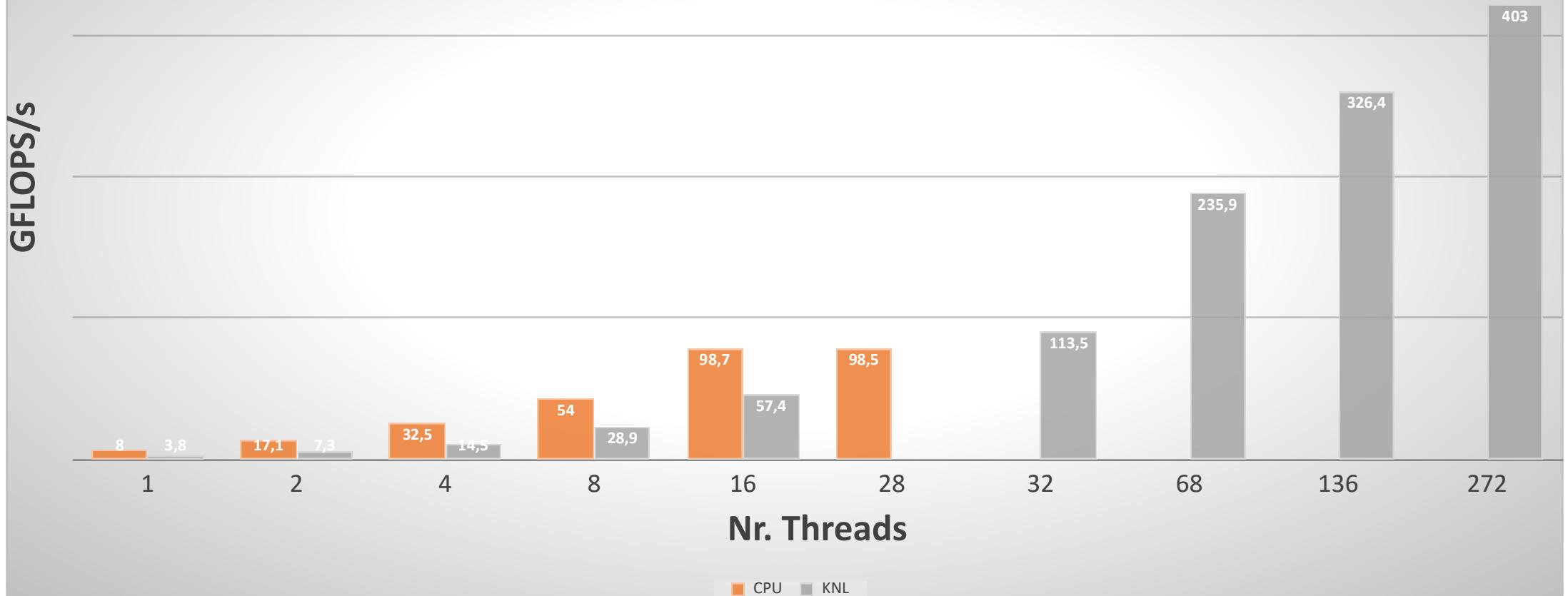
KNL, composed of several Tiles

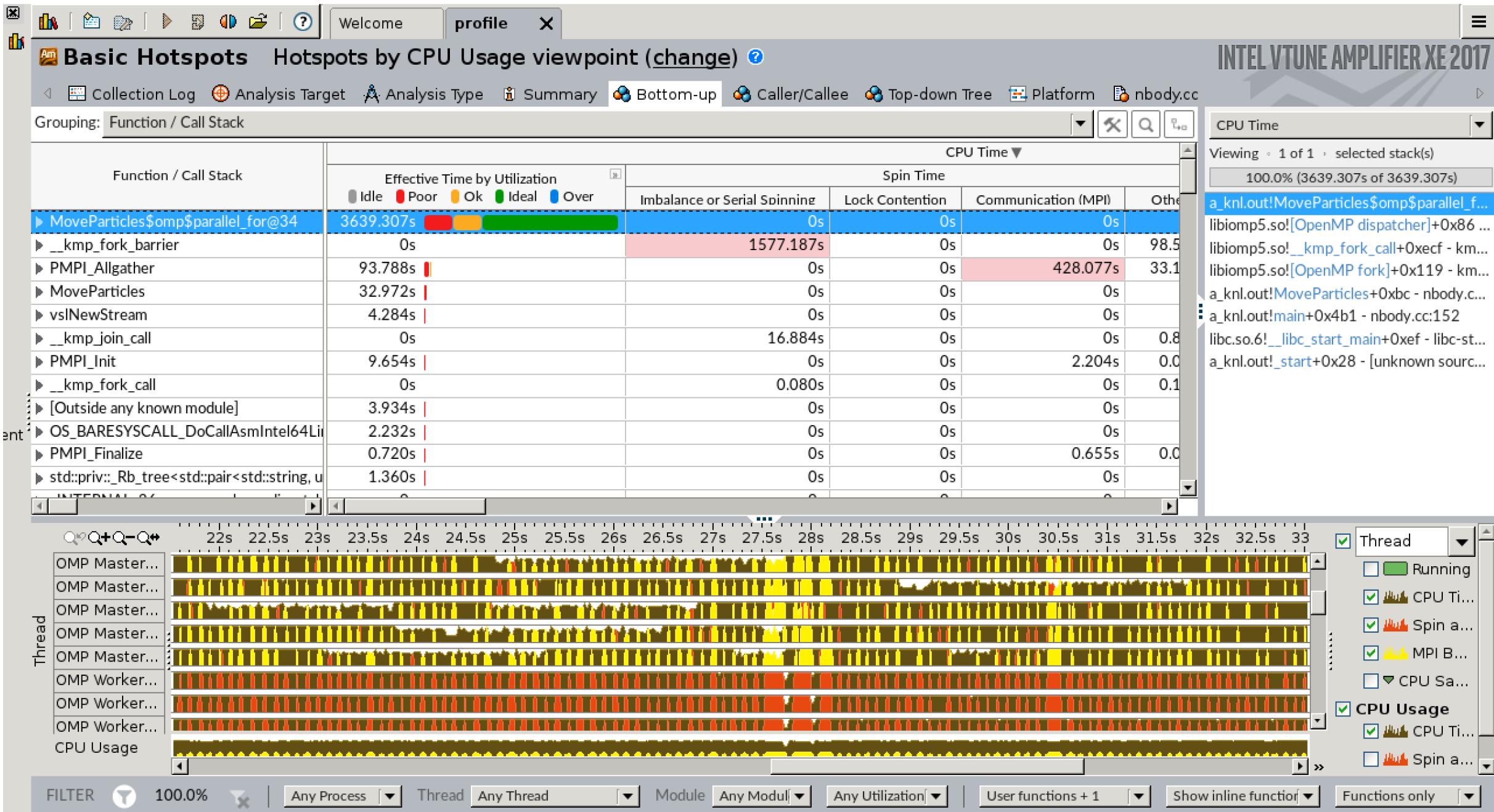
$$Z(i) = A * X(i) + Y(i) \quad (\text{Vector Op.})$$

Tuning your own Applications



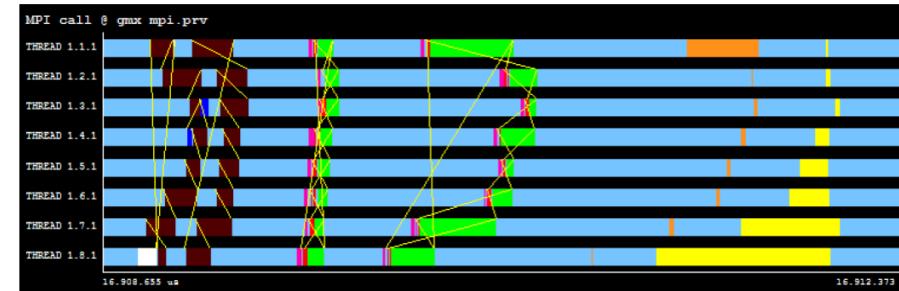
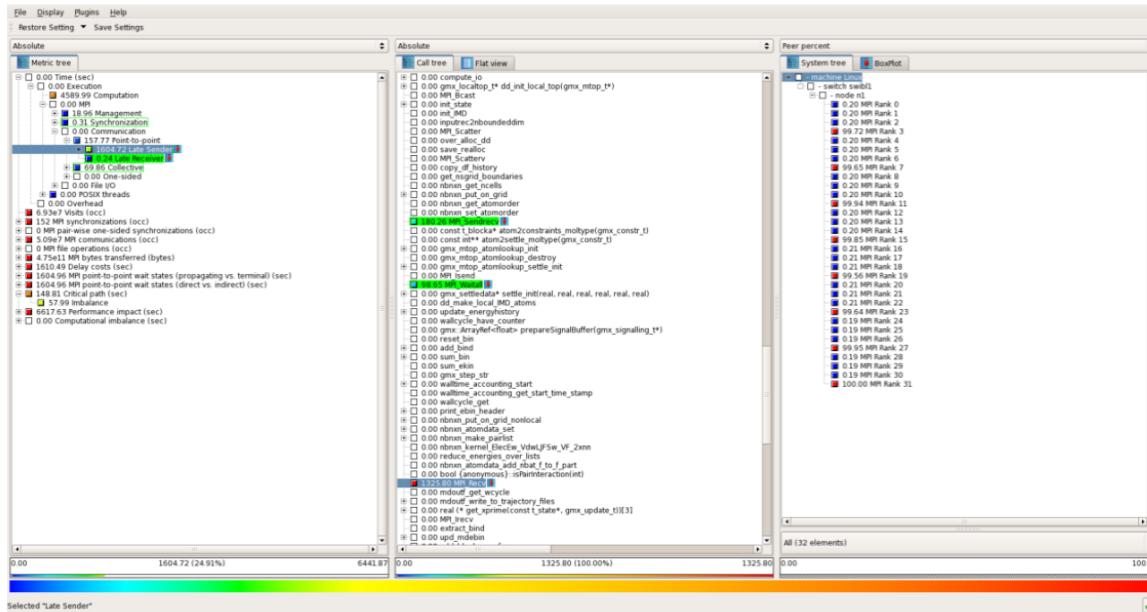
N-body MD





Profiling Tools at HPC2N

- Intel Vtune, Intel Inspector
- SCALASCA, Score-P, Cube
- Extrae/Paraver
- Allinea DDT
- Application Expert support on demand



J. Eriksson, P. Ojeda, T. Ponweiser, T. Steinreicher (2017)

Performance measurement

Nr. cores (n)	Time (min)	Speedup (S)	dS/dn
1	19.5	1	
2	10.8	1.80	0.80
4	5.5	3.54	0.86
8	2.8	6.96	0.85
16	1.5	13	0.75
32	0.9	21.66	0.54
48	0.9	21.66	0

Timing results of a AMBER simulation.

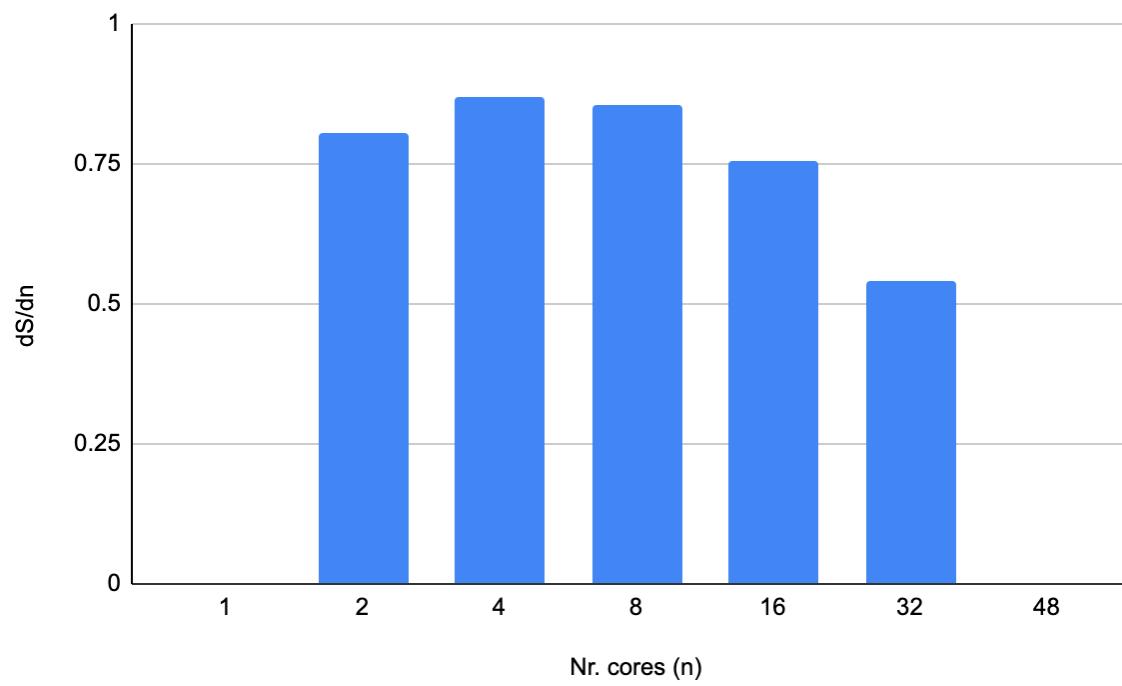
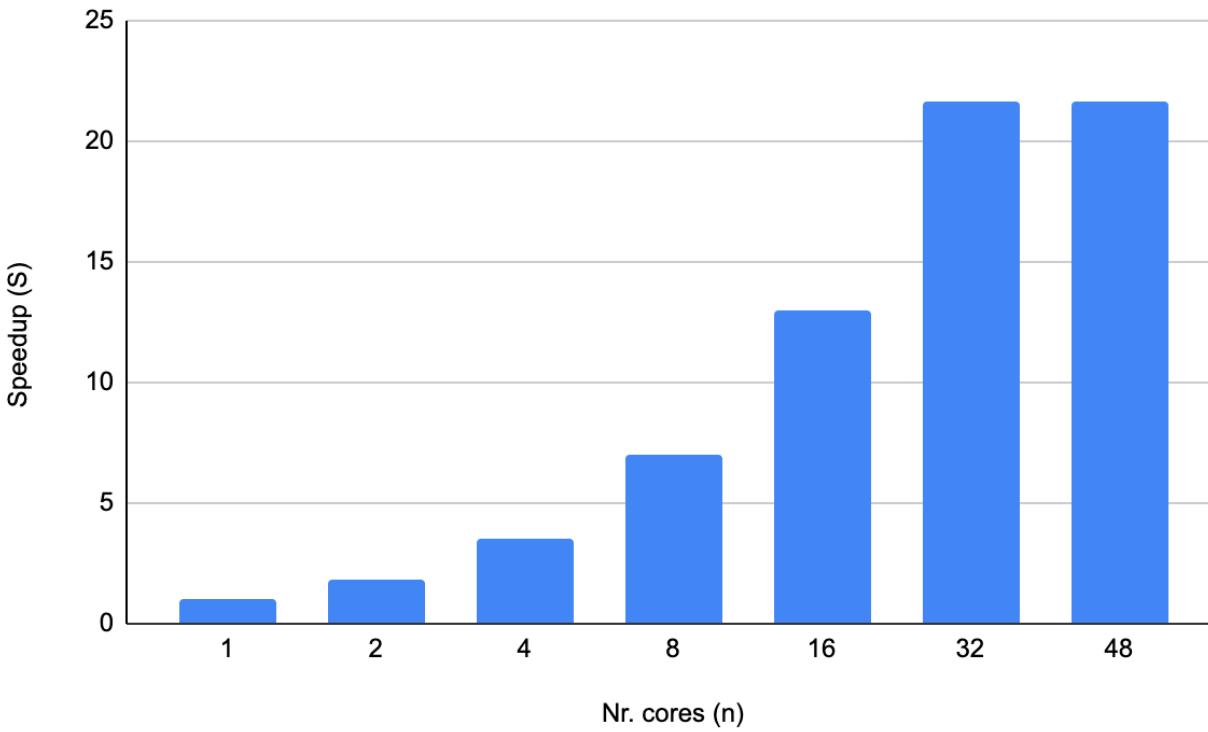
$$S(x \text{ cores}) = \text{Time}(1 \text{ core})/\text{Time}(x \text{ cores})$$

Example:

$$S(2 \text{ cores}) = 19.5 \text{ min}/10.8 = 1.8$$

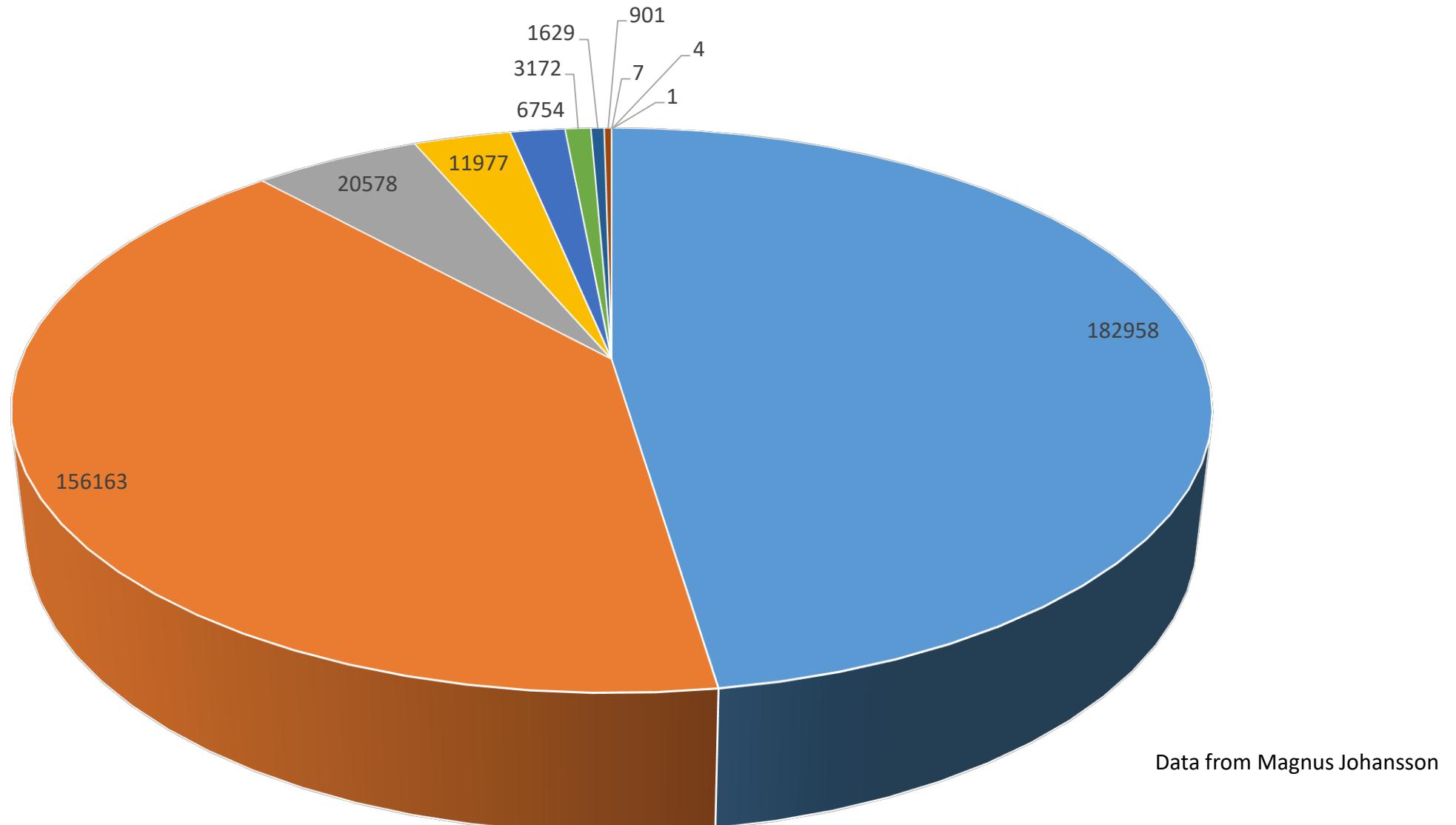
Derivative example

$$\frac{\partial S}{\partial n} \Big|_{n=32} = (21.66 - 13) / (32 - 16) = 0.54$$



Applications

Software on Kebnekaise



- VASP ■ GROMACS ■ LAMMPS ■ GAUSSIAN ■ GPAW ■ R
- NWChem ■ MATLAB ■ AMBER ■ OpenFOAM ■ Wannier90

R/Rstudio

\$ml dependencies_of_R

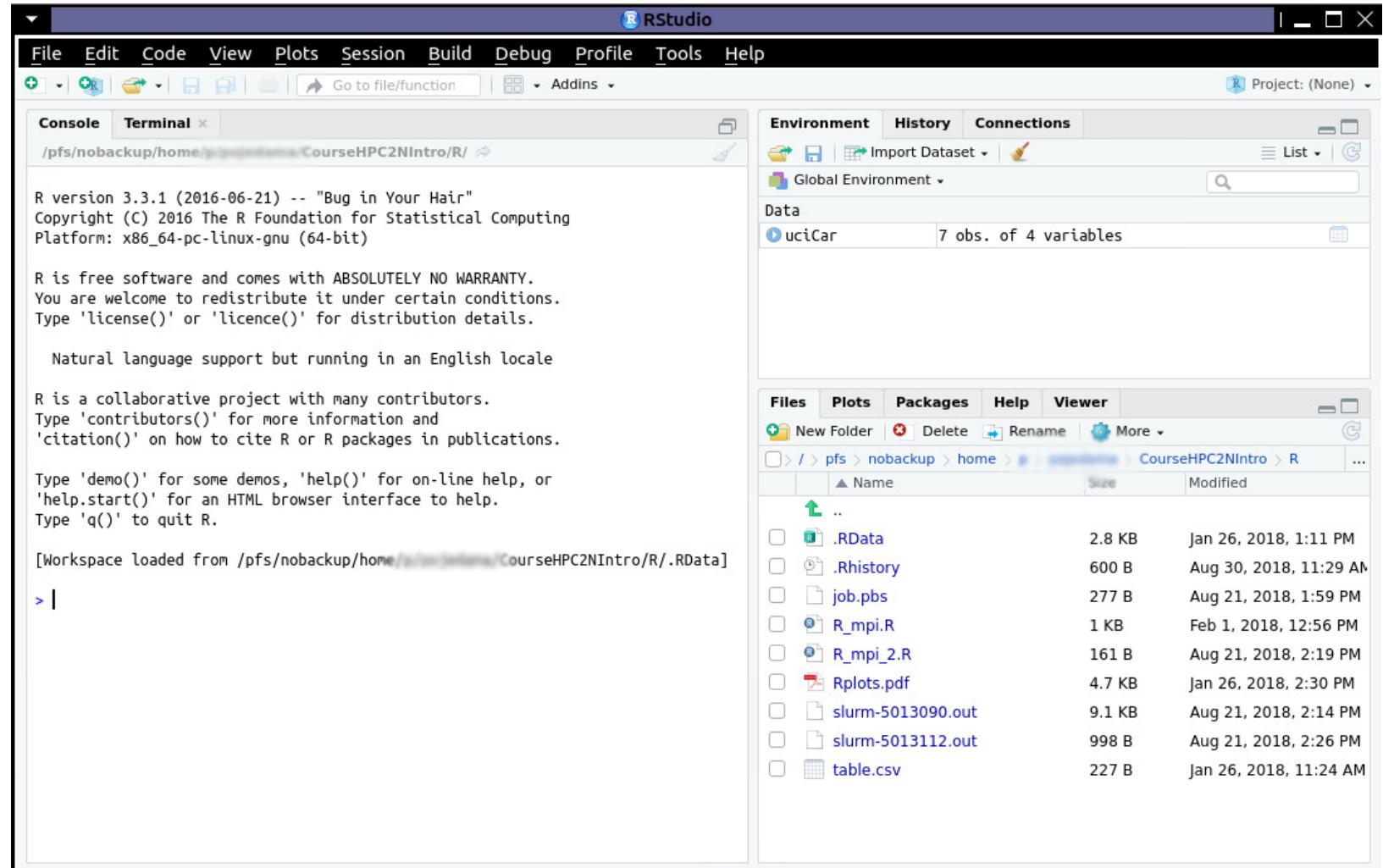
\$ml R/version

\$rstudio

Rstudio will use the loaded R version

Please use the GUI just for
lightweight tasks, otherwise
use the batch queues.

Heavy jobs will be cancelled!



R/Rstudio

Installing packages

In case you need to install some R package that is not included in the installed R version, you need to follow this guideline:

https://www.hpc2n.umu.se/resources/software/user_installed/r

Otherwise, the batch system won't have access to the manually installed packages.

R serial job

```
#!/bin/bash
### SNAC project number, enter your own
#SBATCH -A SNICXXXX-YY-ZZ
# Asking for one core
#SBATCH -n 1
#SBATCH --time=00:10:00

ml dependencies_of_R
ml R/version

R --no-save --quiet < input.R > Rexample.out
```

Further information: <https://www.hpc2n.umu.se/resources/software/r>

R parallel job (Rmpi)

File: Rmpi.R

```
library("Rmpi")
mpi.spawn.Rslaves(nslaves=5)

x <- c(10,20,30,40,50)
mpi.apply(x,runif)

# Close down the MPI processes and quit R
mpi.close.Rslaves()
mpi.finalize()
```

Further information: <https://www.hpc2n.umu.se/resources/software/r>

```
#!/bin/bash
#SBATCH -A project_ID
#Asking for 10 min.
#SBATCH -t 00:10:00
#SBATCH -N 1
#SBATCH -n 6

export OMPI_MCA_mpi_warn_on_fork=0

ml dependencies_of_R
ml R/version

R -q --slave -f Rmpi.R
```

R parallel job (doParallel)

```
#!/bin/bash
#SBATCH -A project_ID
#Asking for 10 min.
#SBATCH -t 00:10:00
##SBATCH -N 1
#SBATCH -n 8
##SBATCH --exclusive

ml dependencies_of_R
ml R/version
```

```
R -q --slave -f doParallel_ML.R
```

doParallel_ML.R

```
library(doParallel)
registerDoParallel(cores=8)
getDoParWorkers()
```

```
library(caret)
library(MASS)
library(klaR)
library(nnet)
library(e1071)
library(rpart)
```

Further information: <https://www.hpc2n.umu.se/resources/software/r>

R parallel job (doParallel)

doParallel_ML.R

```
#!/bin/bash
#SBATCH -A project_ID
#Asking for 10 min.
#SBATCH -t 00:10:00
##SBATCH -N 1
#SBATCH -n 8
##SBATCH --exclusive

ml dependencies_of_R
ml R/version

R -q --slave -f doParallel_ML.R
```

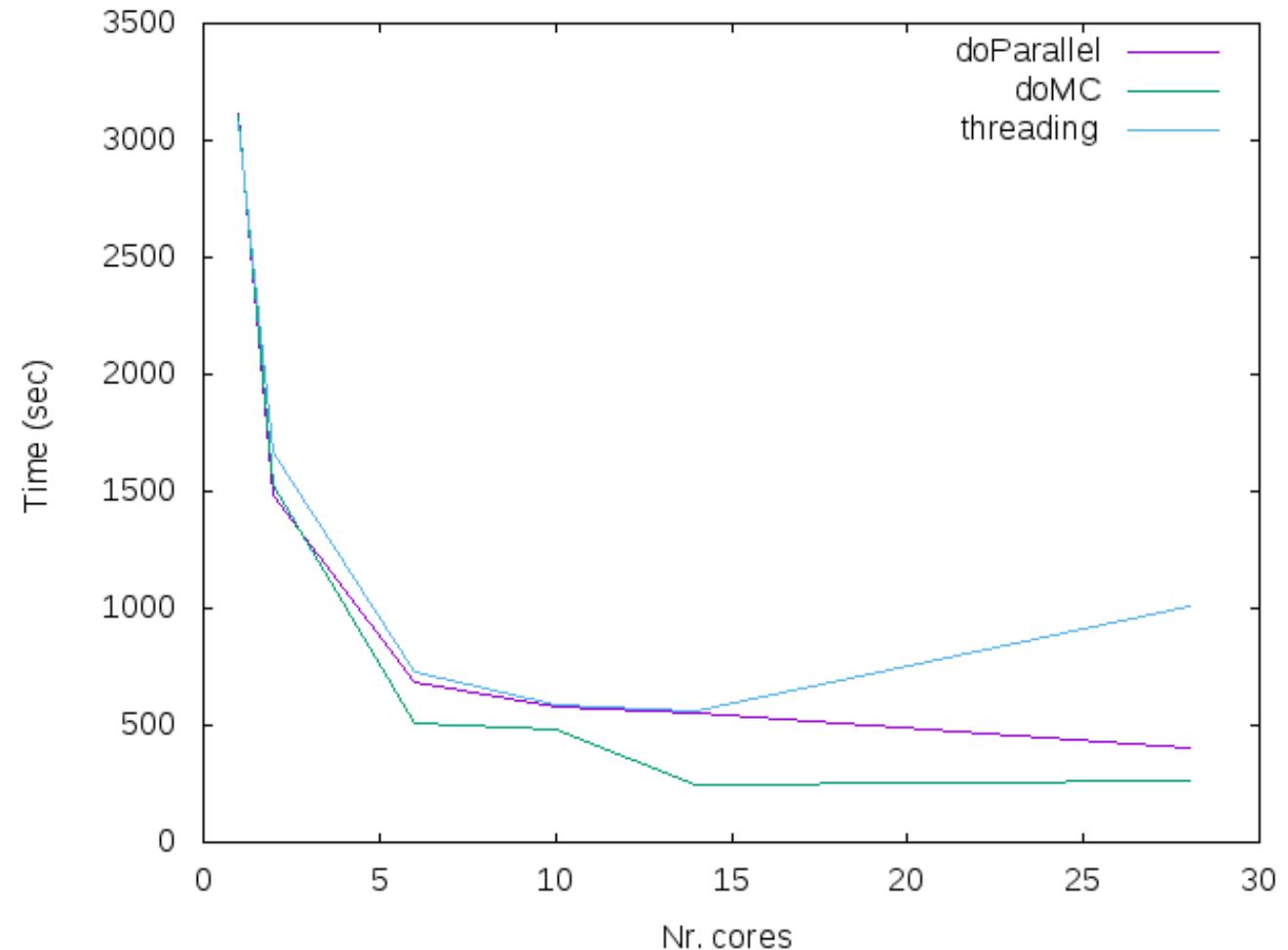
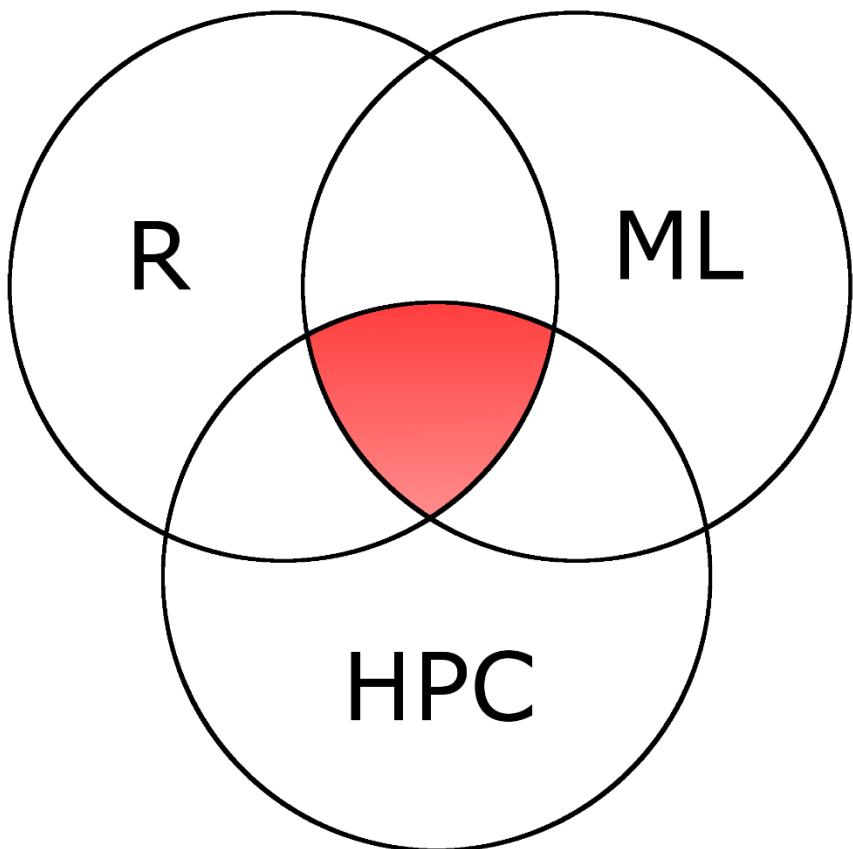
```
#Serial mode
sr <- foreach(1:runs, .combine = rbind) %do% evaluation()

#Parallel mode
pr <- foreach(1:runs, .combine = rbind) %dopar% evaluation()
```

Further information: <https://www.hpc2n.umu.se/resources/software/r>

R machine learning

Example: Boosted classification model using “xgboost” (7 tuning parameters). Using R package on Kebnekaise (details: <http://appliedpredictivemodeling.com/blog/2018/1/17/parallel-processing>).



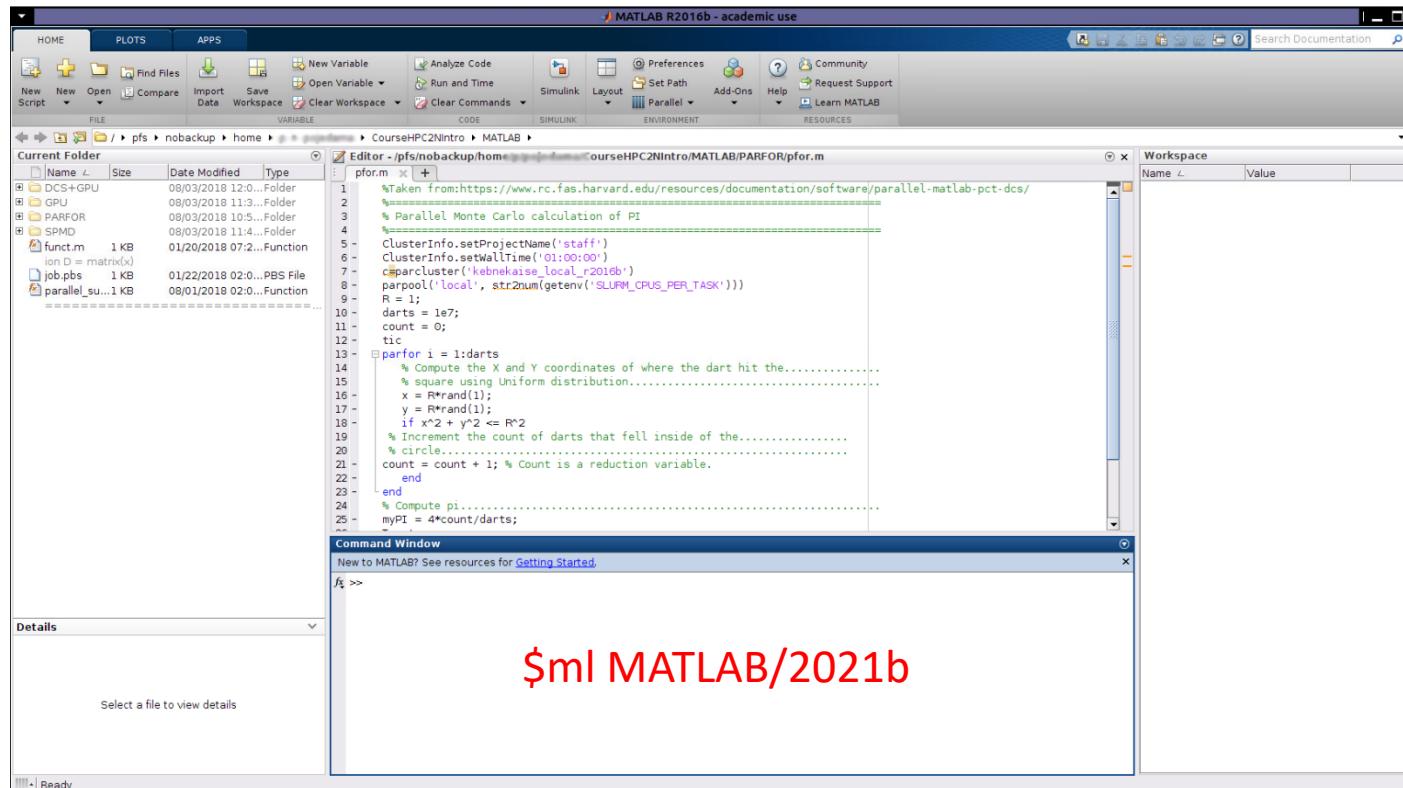
MATLAB

One needs to configure MATLAB the first time one uses it:

*MATLAB 2018b and later

(<https://www.hpc2n.umu.se/resources/software/matlab>)

(<https://www.hpc2n.umu.se/resources/software/configure-matlab-2018>)



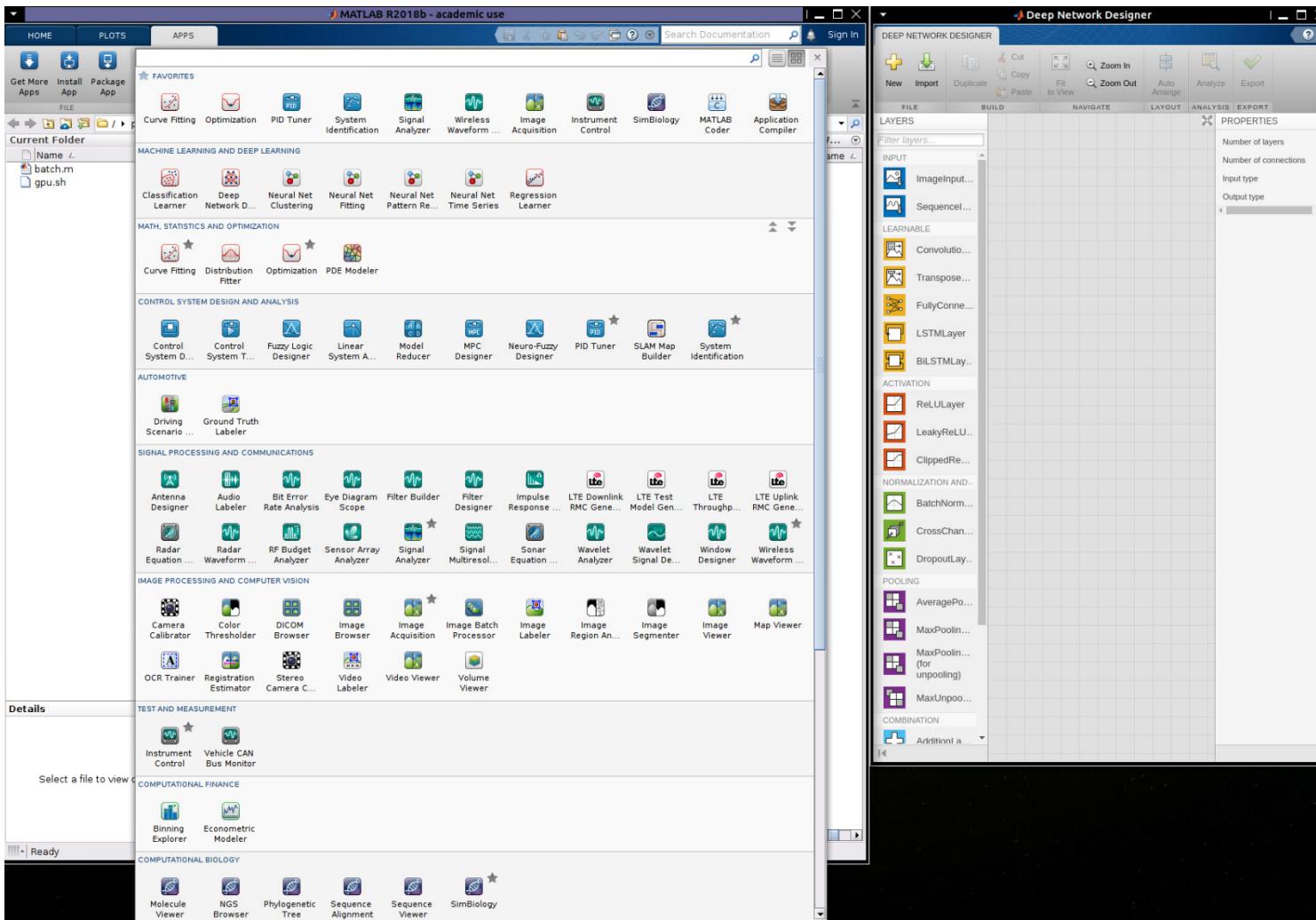
Please use the GUI just for
lightweight tasks, otherwise
use the batch queues.

Heavy jobs will be cancelled!

Start Matlab with:
matlab -singleCompThread

Further information: <https://www.hpc2n.umu.se/resources/software/matlab>

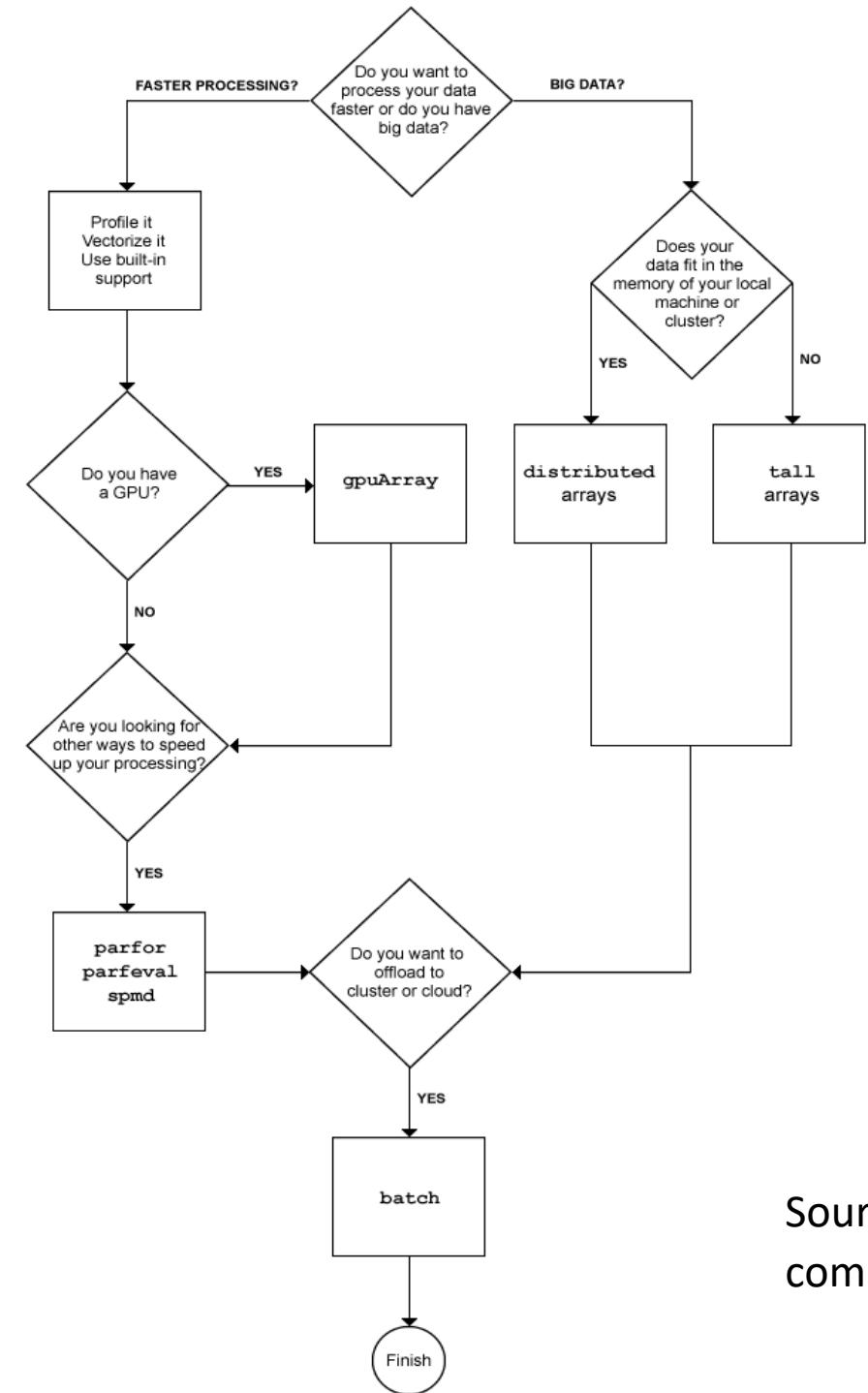
MATLAB



\$ml MATLAB/2021b

Includes improved tools for Machine Learning and Deep Learning

Further information: <https://www.hpc2n.umu.se/resources/software/matlab>



Parallel computing tools in MATLAB

Source: <https://se.mathworks.com/help/parallel-computing/choosing-a-parallel-computing-solution.html>

MATLAB

- Parallel Computing Toolbox (PCT)
 - Parfor loops
 - Single Program Multiple Data, Labs are used to solve a task
- Distributed Computing Server (DCS)

Serial job

File name: funct.m

```
function D  
  
format long  
A1 = rand(10000,10000);  
tic;  
B1 = fft(A1);  
time1 = toc;  
  
filename = 'log.out';  
mid=fopen(filename,'w');  
fprintf(mid,'Time = %16.12f\n',time1);  
fclose(mid);
```

```
#!/bin/bash  
#SBATCH -A project_ID  
#Asking for 10 min.  
#SBATCH -t 00:10:00  
#SBATCH -n 1  
#SBATCH -p batch  
#Load modules necessary for running MATLAB  
ml MATLAB/2021b
```

srun matlab -nodisplay -singleCompThread -r
"funct;exit" > out.log

exit 0

Parfor

File name: pfor.m

```
parpool('local', str2num(getenv('SLURM_CPUS_PER_TASK')))
```

```
parfor i = 1:darts
```

```
x = rand(1);
```

```
y = rand(1);
```

```
if x^2 + y^2 <= R^2
```

```
count = count + 1; % reduction variable.
```

```
end
```

```
end
```

Batch script:

```
#SBATCH -c 4
```

```
#SBATCH -p batch
```

```
#Load modules necessary for running MATLAB  
ml MATLAB/2019b.Update2
```

```
srun matlab --nodisplay -r "pfor"
```

Timing	
Nr. Workers	Time(sec)
1	26.43
2	17.17
3	11.82
4.	7.71
28	1.82

SPMD

File: spmdex.m

```
parpool('local', str2num(getenv('SLURM_CPUS_PER_TASK')))
```

```
% The expression that we wish to integrate:
```

```
f = @(x) 4./(1 + x.^2);
```

```
spmd
```

```
% Choose a range over which to integrate based on my unique index:
```

```
range_start = (labindex - 1) / numlabs;
```

```
range_end = labindex / numlabs;
```

```
% Calculate my portion of the overall integral
```

```
my_integral = quadl( f, range_start, range_end );
```

```
% Aggregate the result by adding together each value of ?my_integral?
```

```
total_integral = gplus( my_integral );
```

```
end
```

```
total_int = total_integral{1};
```

```
fprintf('The computed value of pi is %8.7f.\n',total_int);
```

```
delete(gcp);
```

```
#!/bin/bash
```

```
#SBATCH -A staff
```

```
# Asking for 10 min.
```

```
#SBATCH -t 00:10:00
```

```
#SBATCH -c 4
```

```
#SBATCH -p batch
```

```
# Load modules necessary for MATLAB
```

```
ml MATLAB/2021b
```

```
srun matlab -nodisplay -r "spmdex" > out.log
```

MATLAB+GPU

On the MATLAB GUI:

```
c = parcluster;
ClusterInfo.setName('staff');
ClusterInfo.setWallTime('00:10:00');
% Tell the scheduler that you want to use the GPUs
ClusterInfo.setUseGpu(true)

% number of GPUs per node to use
ClusterInfo.setGpusPerNode(2)
ClusterInfo.setUserDefinedOptions('--gres=gpu:k80:2,mps')
%gpuDevice

j = c.batch(@parallelex, 1, {1}, 'pool', 2);
```

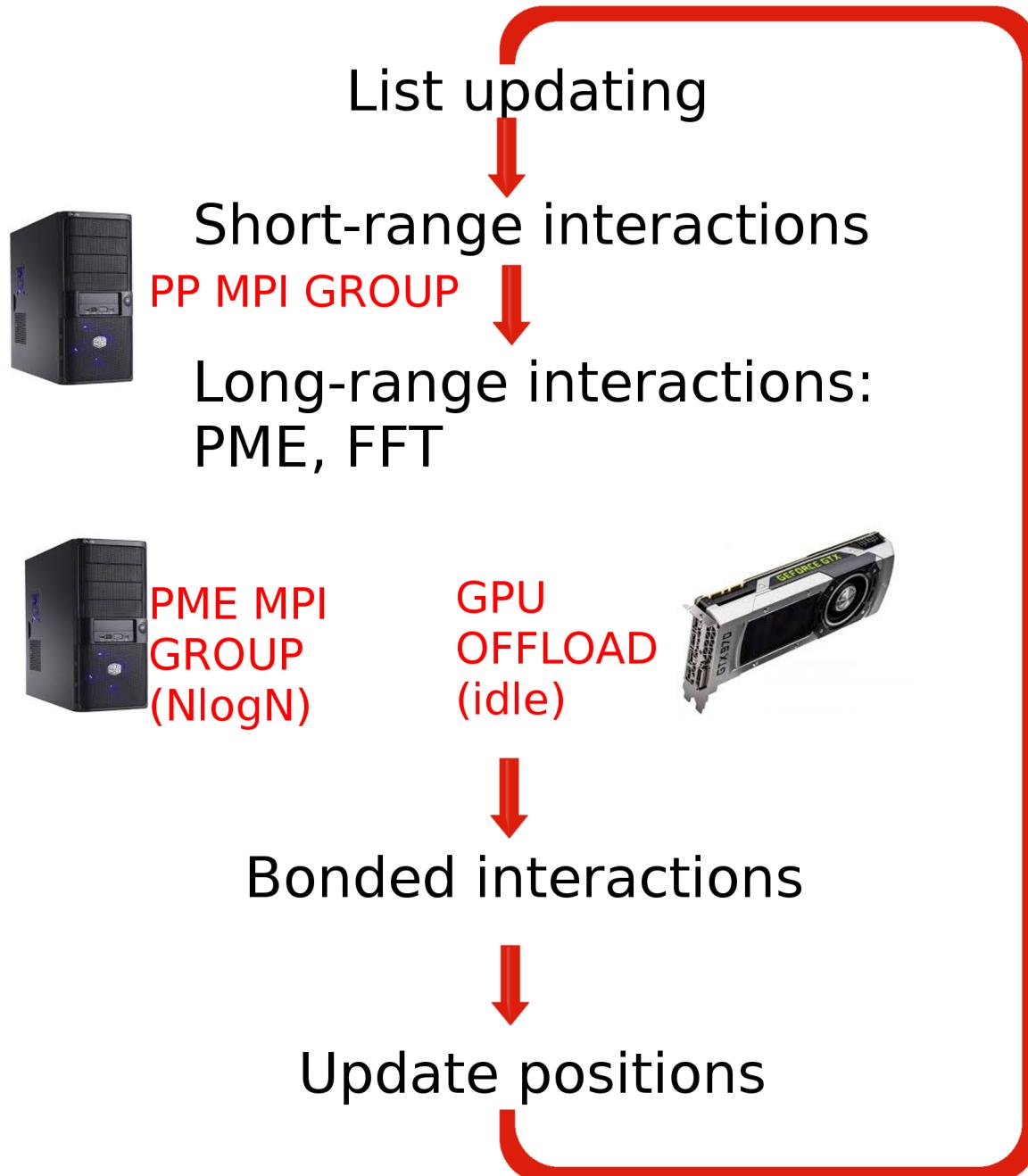
File: parallelex.m

```
function speedUp = parallel(x)
format long
A1 = rand(10000,10000);
tic;
B1 = fft(A1);
time1 = toc;

A2 = gpuArray(A1);
tic;
B2 = fft(A2);
time2 = toc;

speedUp = time1/time2;
end
```

MD Codes



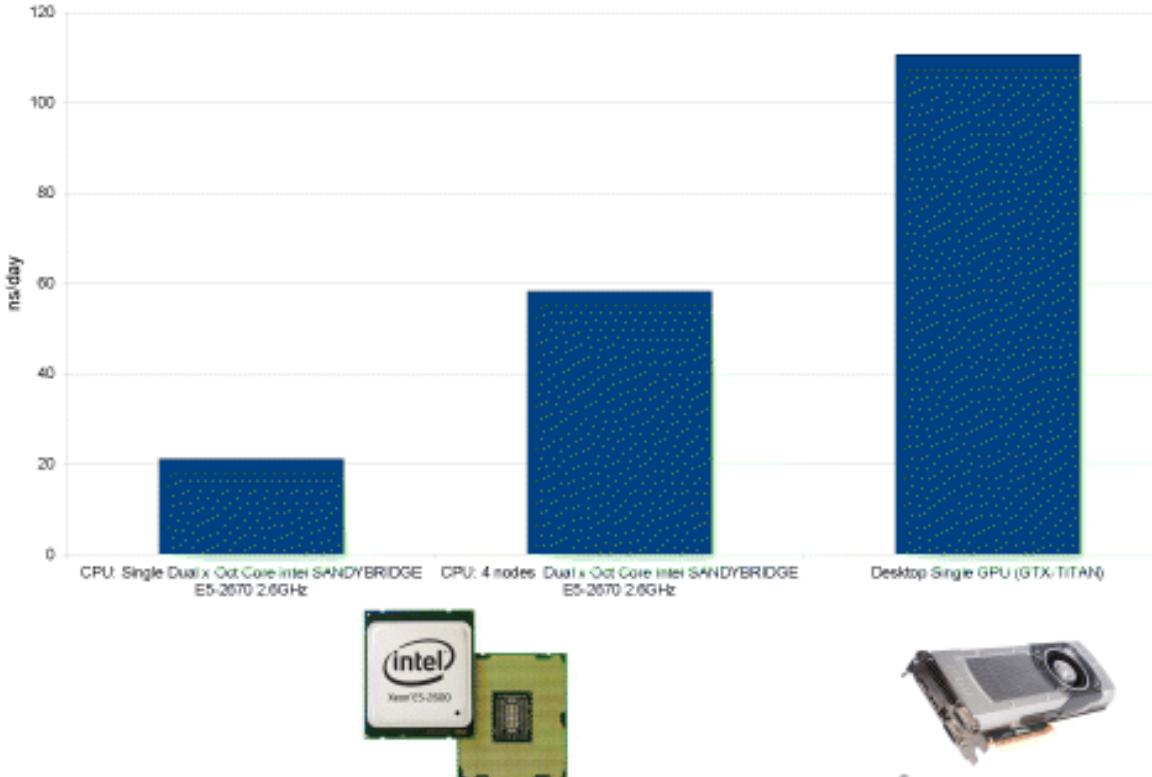
Benchmark

- Solvated protein
- 158945 atoms
- 1.2 nm cutoff radius
- 1 fs time step
- PME for electrostatics

<http://www.charmm-gui.org/>

Amber

- Collection of independent routines
- It uses Sander/ PMEMD for solving the Newton's equations
- It offers a robust set of analysis tools



```
module load dependencies_of_amber  
ml Amber/version  
srun pmemd.MPI -O -i input.mdin  
srun pmemd.cuda.MPI -O -l input.mdin
```

AMBER Tools

- For setting up a simulation (initial structure, solvation, ions, ...):

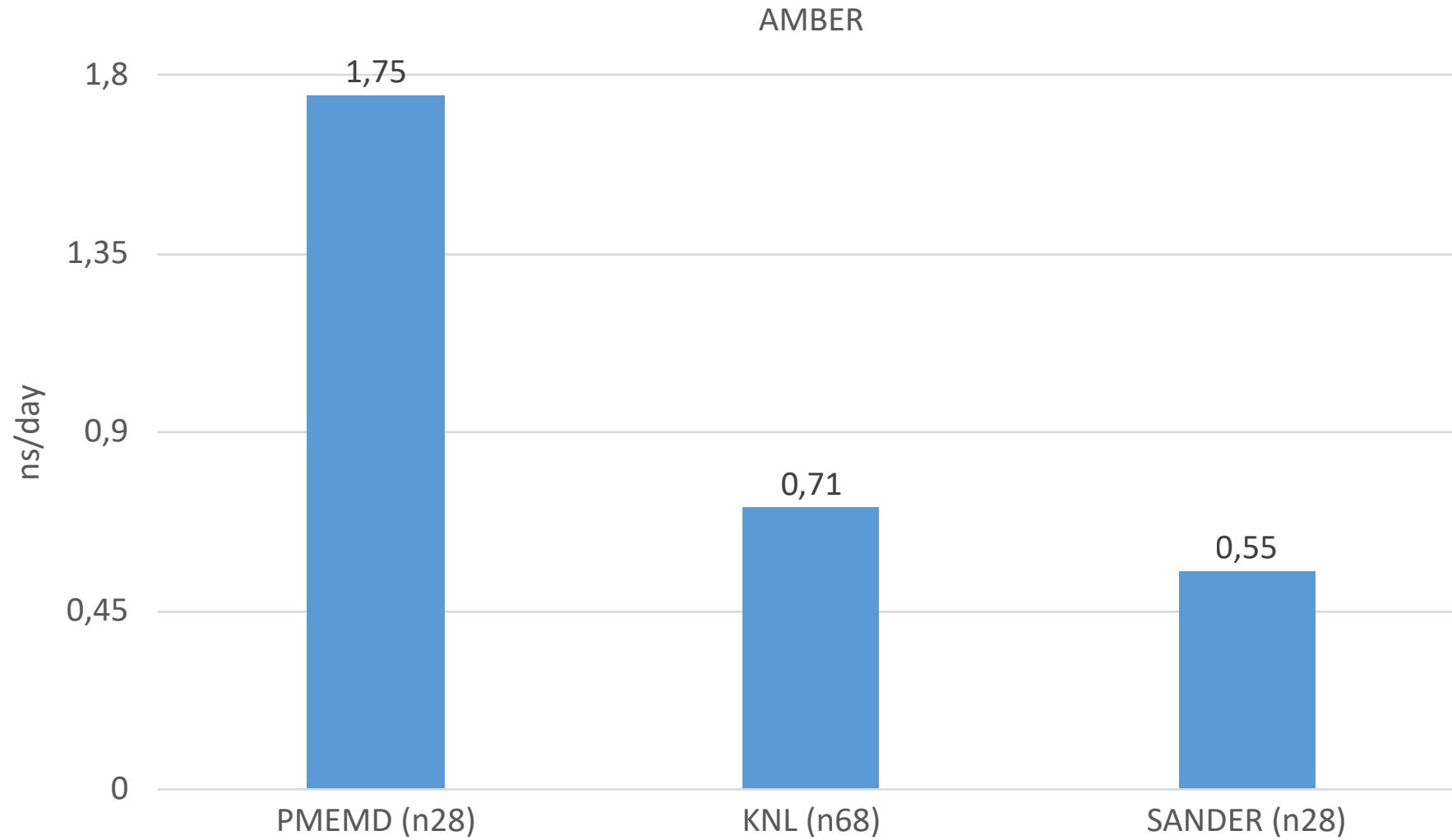
Load the modules:

```
$ml dependencies_of_amber  
$ml Amber/version
```

On the command line:

```
$tleap, antechamber, cpptraj, ...
```

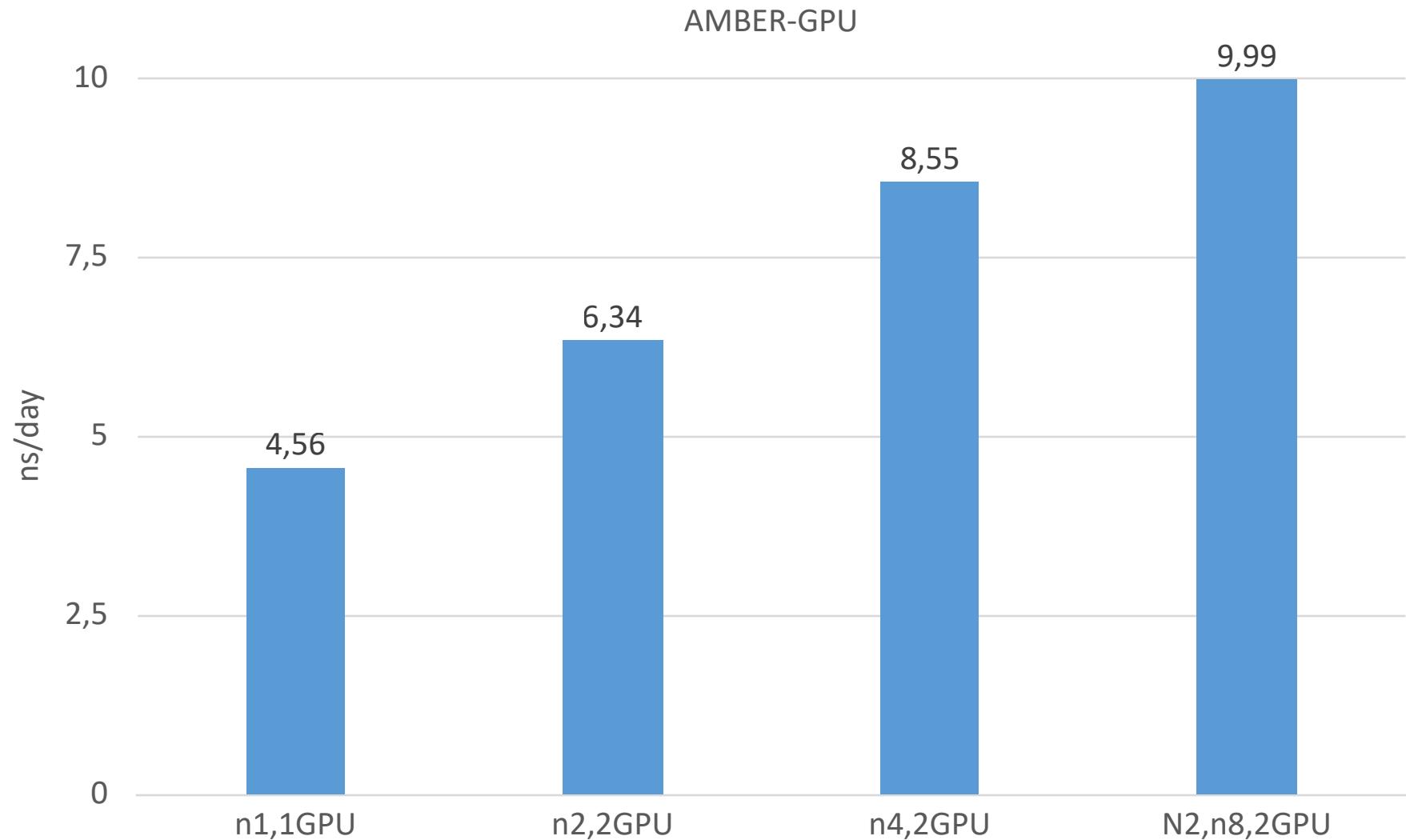
AMBER



AMBER

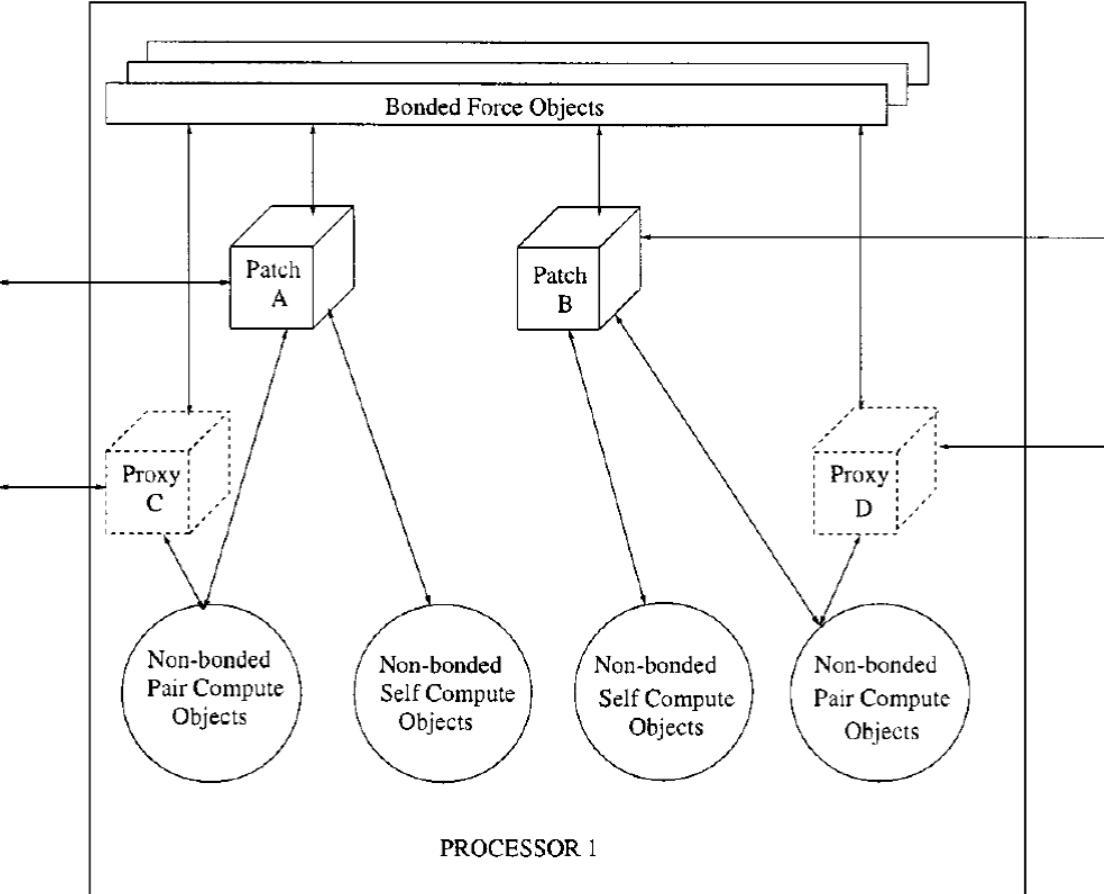
On Kebnekaise,
best performance
is achieved with 4
MPIs/Node and
using 2 GPU cards

Notice that, the
remaining CPUs
are not used.



NAMD

- Based on charm++ communication protocol
- It is object-oriented
- Versions: single node, multi-node, GPU, and KNL
- Highly scalable



NAMD (GPU) (single node)

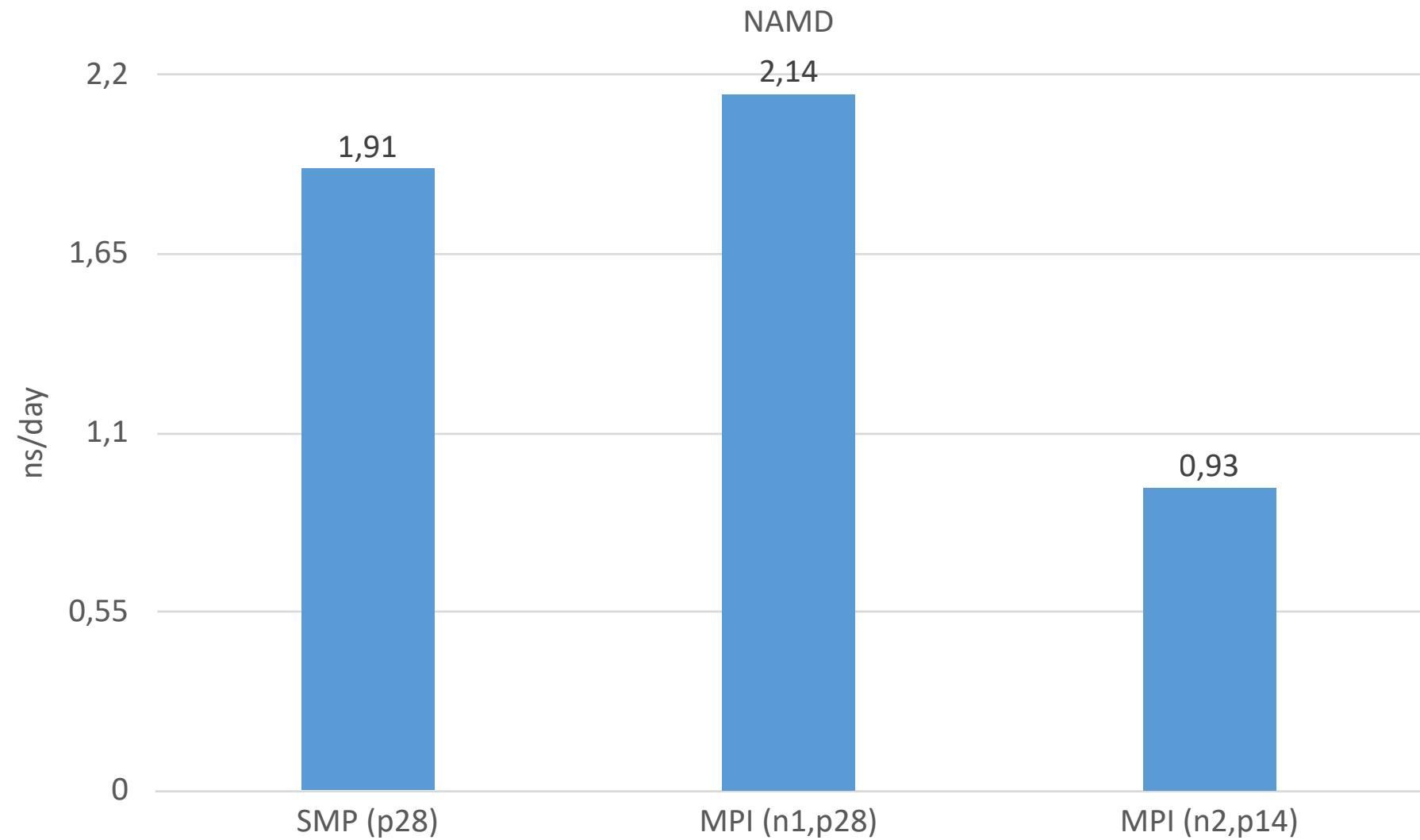
```
#!/bin/bash
#SBATCH -A staff
#SBATCH -t 00:50:00
#SBATCH -N 1
#SBATCH -n 28
#SBATCH --exclusive
#Ask for 2 GPU cards
#SBATCH --gres=gpu:k80:2
#Load modules necessary for running NAMD
ml GCC/9.3.0 CUDA/11.0.2 OpenMPI/4.0.3
ml NAMD/2.14-nompi
#Execute NAMD
namd2 +p 28 +setcpuaffinity step4_equilibration.inp > output_smp.dat
```

NAMD

- Colvars module for free energy calculations can be run on GPUs.

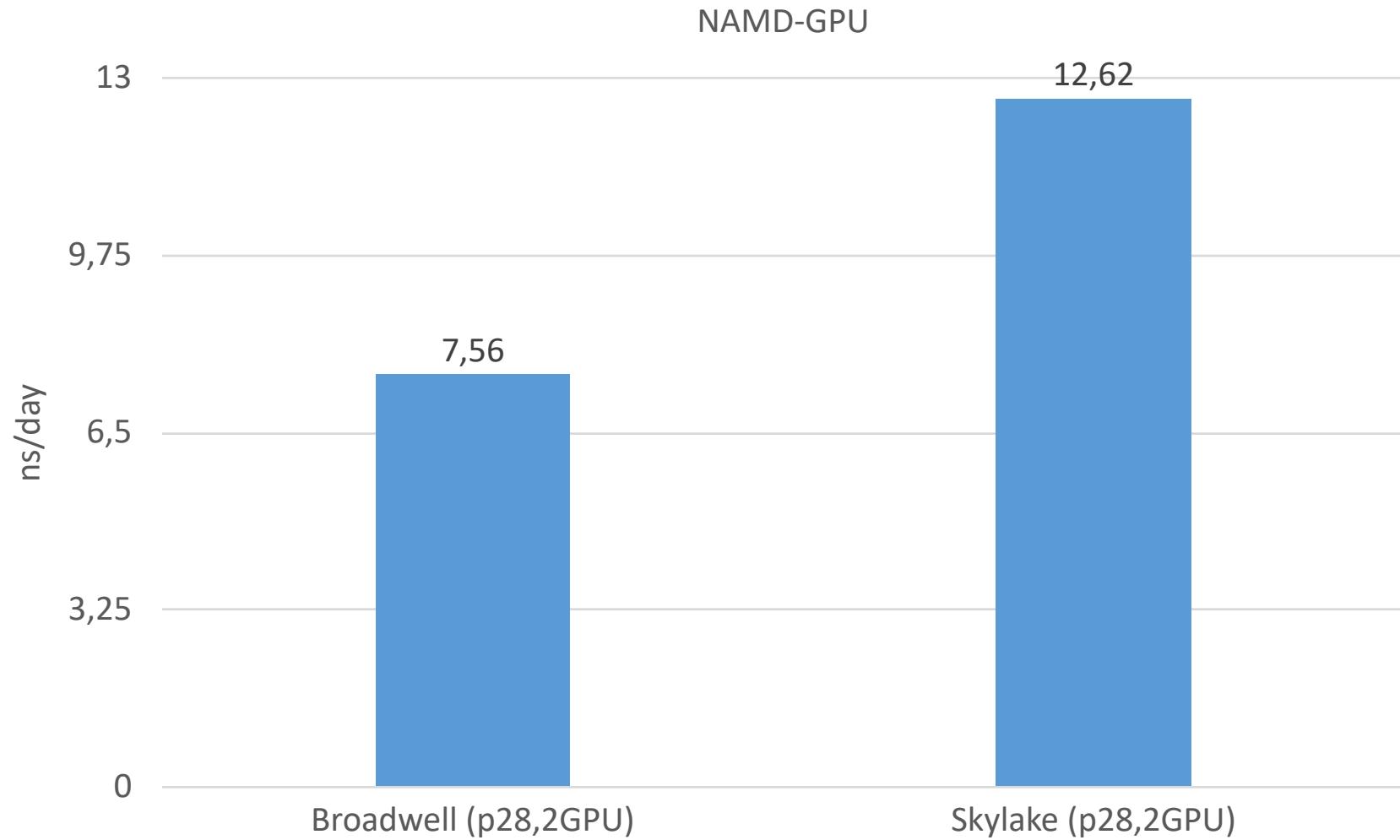
NAMD

Use the `+setcpuaffinity`
flag for a **10% speedup**



NAMD

With the new Volta GPU cards one can speed up the simulation by **1.66x**



GROMACS

```
#SBATCH -n 4
#SBATCH -c 7
# Asking for 2 GPUs
#SBATCH --gres=gpu:k80:2
#SBATCH -p batch
```

```
ml dependencies_of_gromacs
ml GROMACS/version
```

```
if [ -n "$SLURM_CPUS_PER_TASK" ]; then
    mdargs="-ntomp $SLURM_CPUS_PER_TASK"
else
    mdargs="-ntomp 1"
fi
```

GROMACS recognizes the number of available GPU cards

```
srun -n $SLURM_NTASKS gmx_mpi mdrun $mdargs -npme 0 -dlb yes -v -deffnm step4.1_equilibration
```

gmx tune_pme

Individual timings for input file 0 (npt_bench00.tpr):

PME ranks	Gcycles	ns/day	PME/f	Remark
0	4355.019	57.776	-	OK.
0	4547.105	55.335	-	OK.
0	4289.420	58.659	-	OK.
-1(0)	4455.791	56.469	-	OK.
-1(0)	4440.157	56.668	-	OK.
-1(0)	4275.551	58.850	-	OK.

Tuning took 7.7 minutes.

Summary of successful runs:

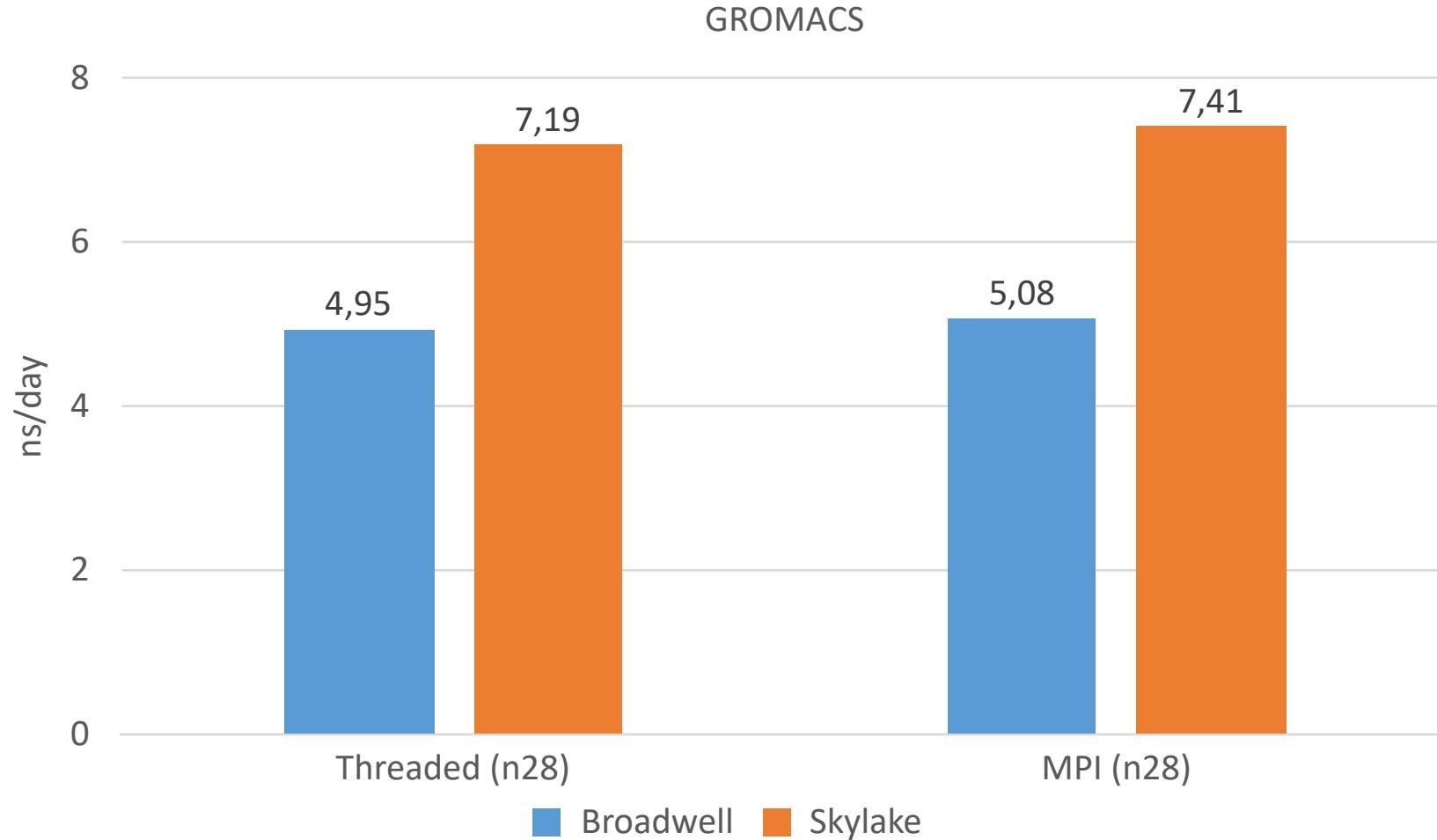
Line	tpr	PME ranks	Gcycles	Av.	Std.dev.	ns/day	PME/f	DD grid
0	0	0	4397.181	133.917	57.257	-	4 1 1	
1	0	-1(0)	4390.500	99.855	57.329	-	4 1 1	

Best performance was achieved with the automatic number of PME ranks (see line 1)

Please use this command line to launch the simulation:

mpirun -np 4 gmx_mpi mdrun -npme -1 -s npt.tpr -ntomp 7 -dlb yes

GROMACS



GROMACS KNL

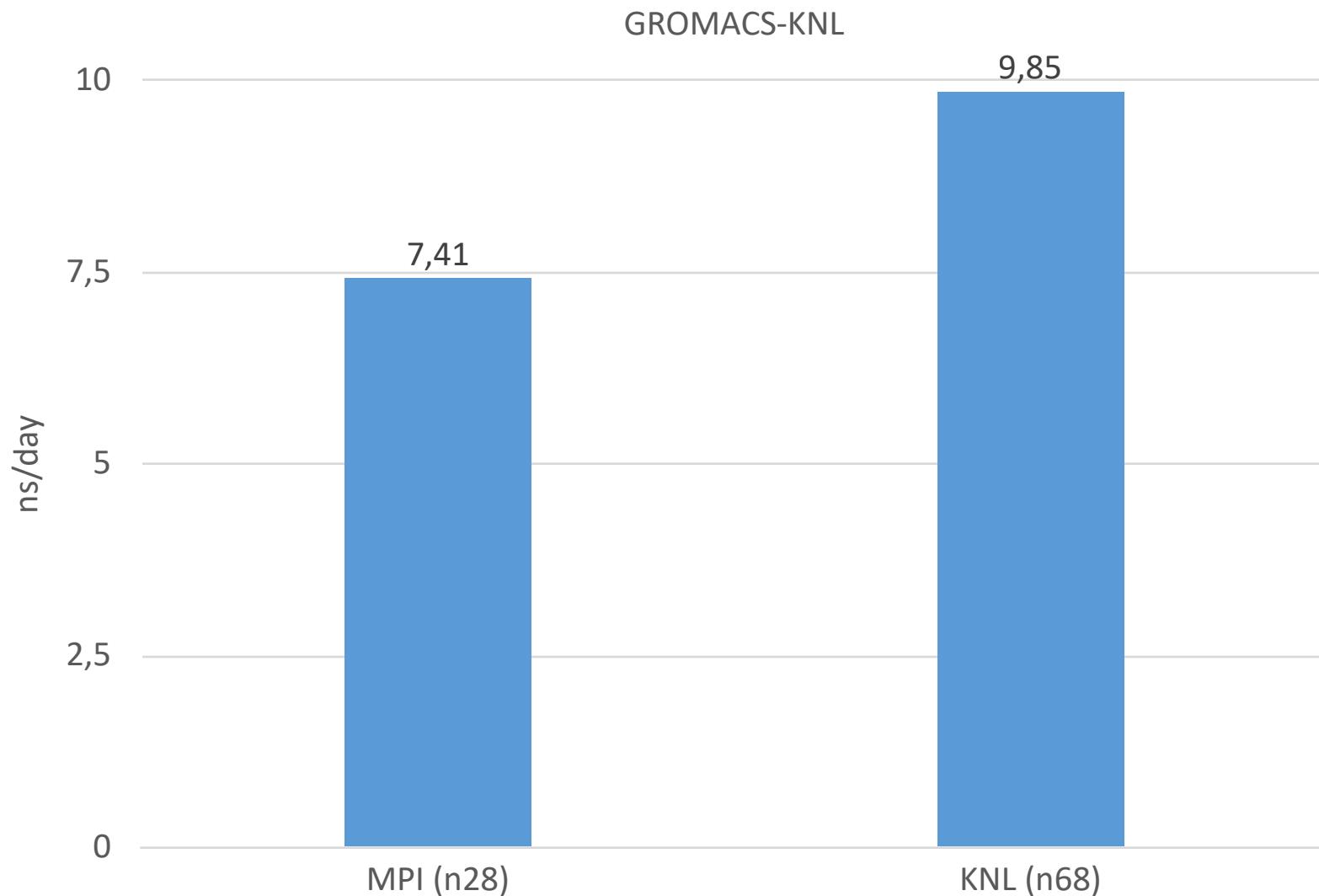
```
#!/bin/bash
#SBATCH -A project_ID
#SBATCH -t 00:50:00
#SBATCH -N 1
#SBATCH -n 68
#SBATCH -p knl
#SBATCH --constraint=cache,quad
#SBATCH --exclusive

ml dependencies_of_gromacs
ml GROMACS/version

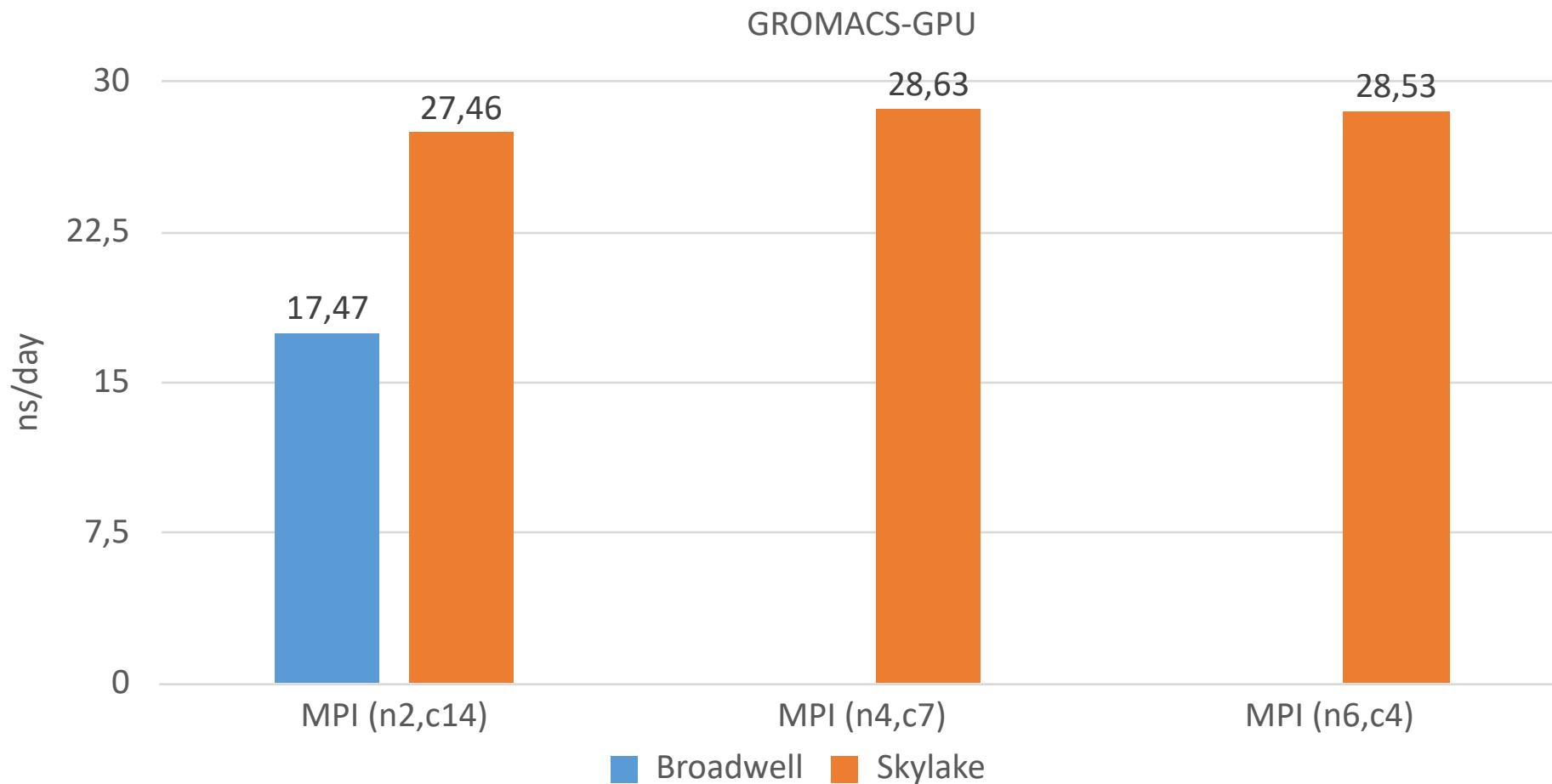
export OMP_NUM_THREADS=2
gmx mdrun -ntmpi 68 -npme 18 -ntomp 2 -pin on -pinoffset 0 -pinstride 2 -dlb auto -v -deffnm step4.1_equilibration
```

GROMACS KNL

- Login to kebnekaise-knl.hpc2n.umu.se and submit your script from there
- KNL queue is most of the time available compared to GPU one



GROMACS



GROMACS

- PLUMED, threaded MPI version is not supported.

Instead of:

`gmx mdrun –ntmpi X`

Use:

`mpirun gmx_mpi ...`

- Avoid writing energies
- Offloading features starting from 2018 version

<https://www.hpc2n.umu.se/events/courses/md-course-spring-2019>

LAMMPS (MPI)

```
#!/bin/bash
#SBATCH -A staff
#Asking for 10 min.
#SBATCH -t 02:10:00
#Number of nodes
#SBATCH -N 1
#Ask for 28 processes
#SBATCH -n 28

#Load modules necessary for running LAMMPS
module load dependencies_of_lammps
module load LAMMPS/version

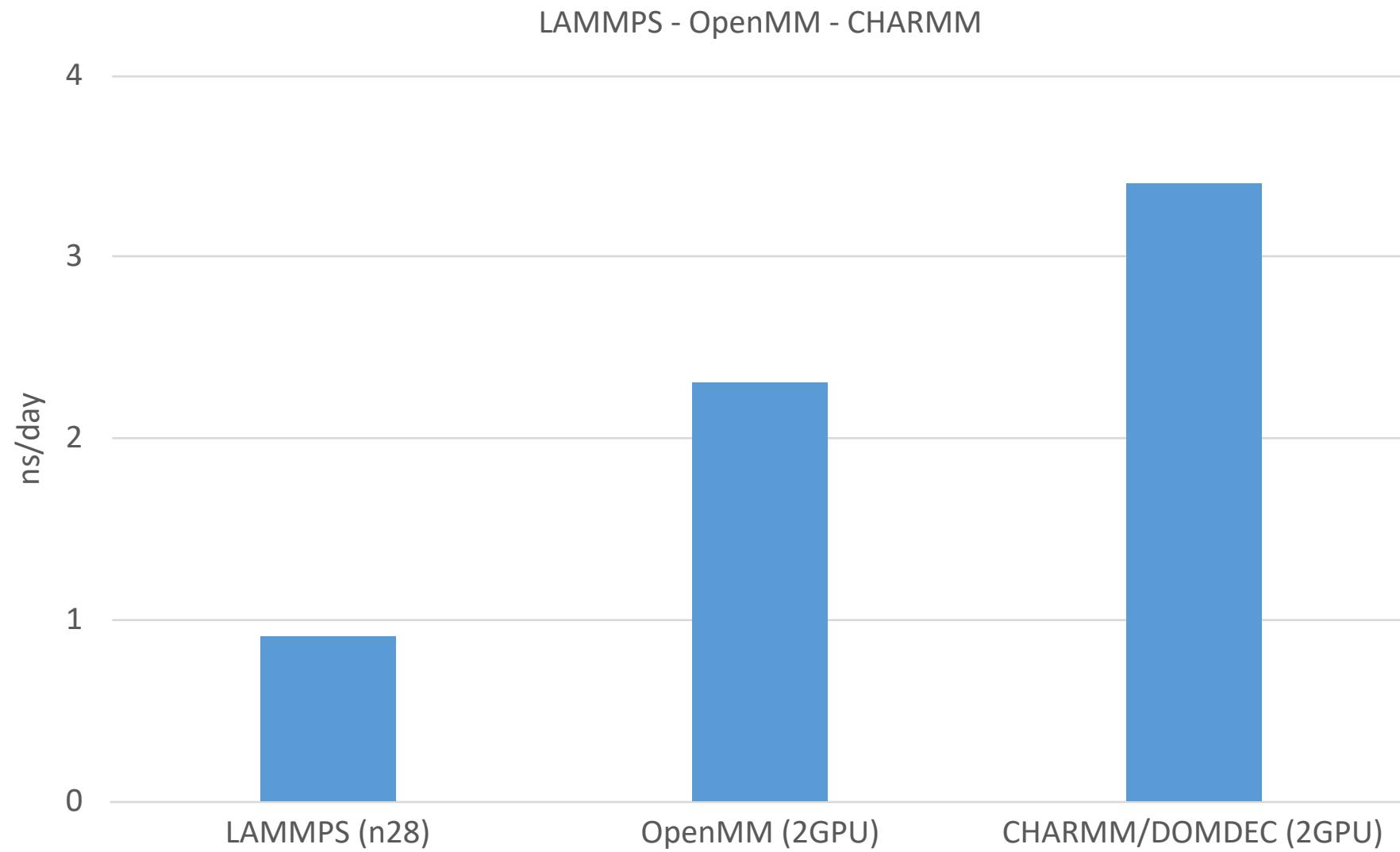
#Execute LAMMPS
srun lmp_intel_cpu -in step4.1_equilibration.inp
```

LAMMPS (OpenMP)

```
#!/bin/bash
#SBATCH -A staff
#Asking for 10 min.
#SBATCH -t 02:10:00
#Number of nodes
#SBATCH -N 1
#Ask for 28 processes
#SBATCH -n 14
#SBATCH -c 2

#Load modules necessary for running LAMMPS
module load dependencies_of_lammps
module load LAMMPS/version
export OMP_NUM_THREADS=2
#Execute LAMMPS
srun lmp_intel_cpu -in step4.1_equilibration.inp
```

LAMMPS - OpenMM - CHARMM



Scratch directory

- If the program writes frequently data, you can get an additional speed up by using the local scratch directory:

```
rm -rf /scratch/test  
export parent=/pfs/nobackup/home/u/username/benchmarks/  
mkdir /scratch/test  
rsync -avzh $parent/ /scratch/test/
```

```
cd /scratch/test
```

```
namd2 +p 28 +setcpuaffinity input.inp > output.dat
```

```
cd $parent
```

```
rsync -avzh /scratch/test/ $parent/
```

```
rm -rf /scratch/test
```

~10-15% speedup

Abisko (all nodes): 352 GB

Kebnekaise, standard nodes: 171 GB

Kebnekaise, GPU nodes: 171 GB

Kebnekaise, Largemem nodes: 352 GB
(a few have 391 GB)

Suggestions

- Check if the software version you are trying is MPI (-n), OpenMP (-c), or hybrid (-n -c). Also, if you are using a GPU version.
- Is your simulation scaling properly?
- Is the cutoff radius for long-range interactions appropriate? PME (gmx tune_pme for GROMACS)?

Tensorflow

```
#!/usr/share/python

import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
print(sess.run(hello))
```

```
#!/bin/bash
#SBATCH -A staff
#SBATCH -t 00:50:00
#SBATCH -N 1
#SBATCH --exclusive
#SBATCH --gres=gpu:k80:2
```

```
ml dependencies_of_Tensorflow
ml Tensorflow/version
python tensor.py
```

```
name: Tesla K80
major: 3 minor: 7 memoryClockRate (GHz) 0.8235
pciBusID 0000:0f:00.0
Total memory: 11.17GiB
Free memory: 11.10GiB
2017-12-05 17:13:03.555397: W
Found device 1 with properties:
name: Tesla K80
>>> print(sess.run(hello))
b'Hello, TensorFlow!'
```

Tensorflow

```
#!/usr/share/python
import tensorflow as tf

#Parameters
W = tf.Variable([.3],tf.float32)
b = tf.Variable([-3],tf.float32)
#Input and output
x = tf.placeholder(tf.float32)
linear_model = W*x+b
y = tf.placeholder(tf.float32)

#Loss
square_delta = tf.square(linear_model-y)
loss = tf.reduce_sum(square_delta)
#Optimize
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(loss)

init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

for i in range (1000):
    sess.run(train,{x:[1,2,3,4],y:[0,-1,-2,-3]})

print(sess.run([W,b]))
```

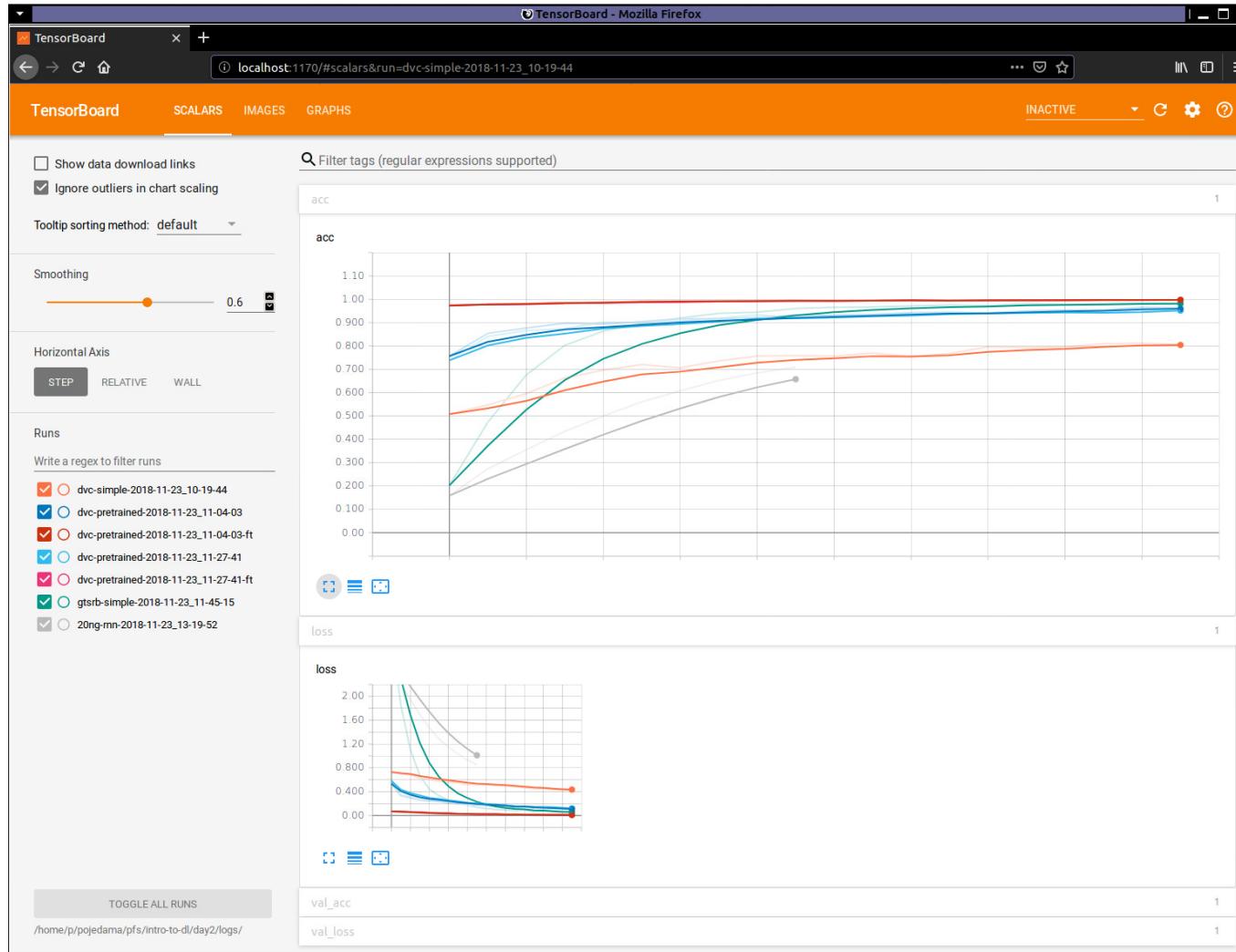
Computing the loss

name: Quadro K6000
major: 3 minor: 5 memoryClockRate (GHz) 0.9015
pciBusID 0000:06:00.0
Total memory: 11.92GiB
Free memory: 11.82GiB
Creating TensorFlow device (/gpu:0) -> (device: 0, name: Quadro K6000, pci bus id: 0000:06:00.0)

[array([-0.9999969], dtype=float32), array([0.99999082], dtype=float32)]

Tensorflow+Tensorboard

```
module load GCC/7.3.0-2.30 CUDA/9.2.88 OpenMPI/3.1.1 #On the command line execute these instructions  
module load TensorFlow/1.10.1-Python-2.7.15 Keras/2.2.2-Python-2.7.15  
tensorboard --logdir=/home/p/<path-to-directory>/logs --port=1170
```



To monitor the behavior of your jobs open a web browser and type:

`localhost:1170`

GAUSSIAN+GPU

Initial input file:

```
$more input_bk.com
%chk=geom_optim.chk
%mem=16GB
#UB3LYP/6-31+G(d) OPT=(ModRedun) SCF=(MaxCycle=256) pop=none NoSymm

45 atoms structure, RESP

+5 15
0      3.744336      -1.126487      6.111505
P      2.893853      -1.251776      4.246949
O      4.150424      -3.154051      4.078061
```

Corrected input file:

```
$more input.com
%cpu=0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27
%gpucpu=0-3=0,7,14,21
%chk=geom_optim.chk
%mem=16GB
#UB3LYP/6-31+G(d) OPT=(ModRedun) SCF=(MaxCycle=256) pop=none NoSymm
```

```
#!/bin/bash
#SBATCH -A staff
#SBATCH -N 1
#SBATCH -c 28
#SBATCH --exclusive
#SBATCH --gres=gpu:k80:2
#SBATCH --time=00:10:00

module add gaussian/16.A.03-AVX2
# Assume that the job file are located
g16.set-cpu+gpu-list input.com
time g16 input
```

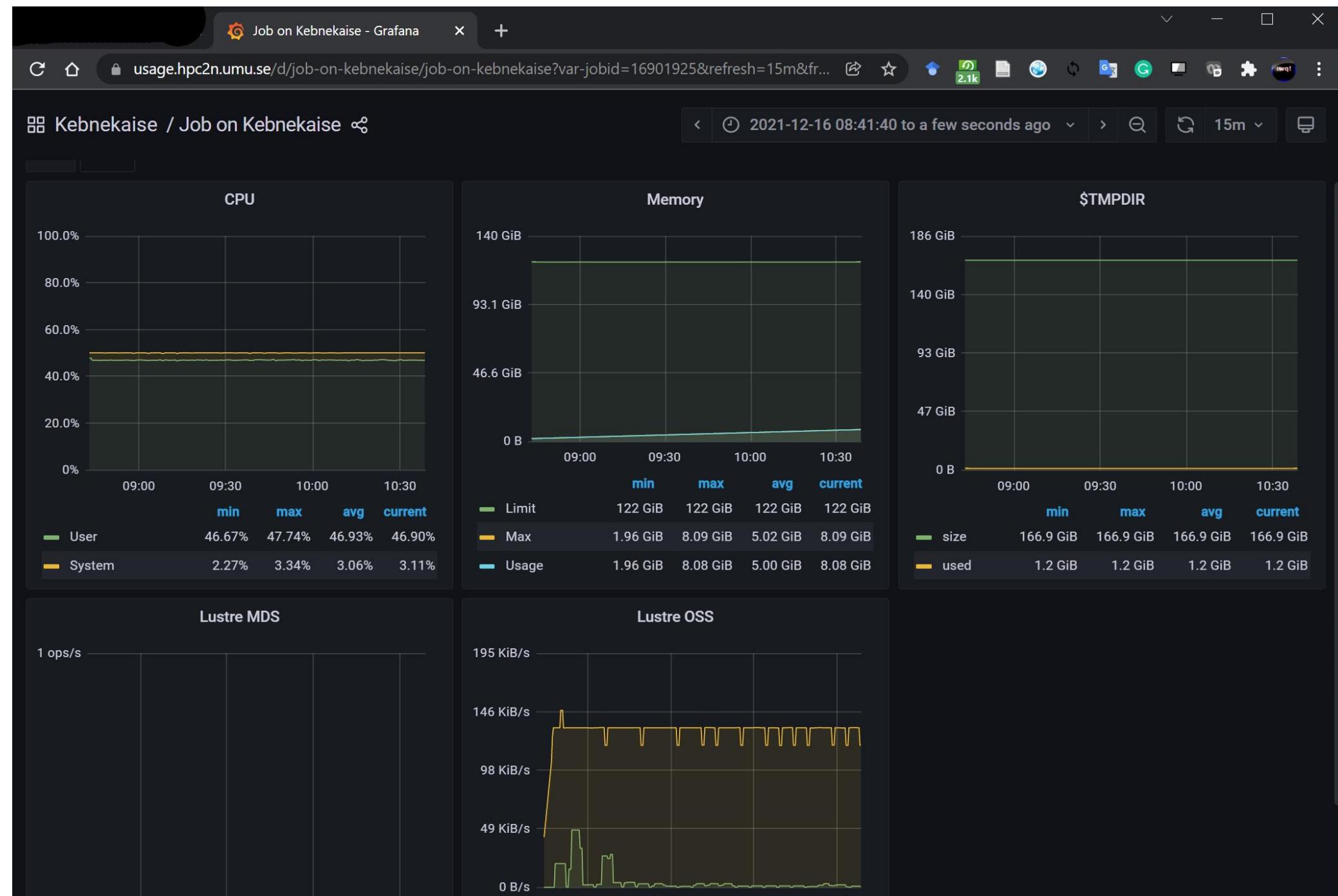
Integral=(FineGrid,Acc2E=10) Constants=2006

Suggestions

- One common issue with Comp. Chem. Software is related to the available memory. A possible solution is to request a larger number of cpus (with –n or –c in SLURM) but make use of less than that number.

Final comments

- Check if the software version you are trying is MPI (-n), OpenMP (-c), or hybrid (-n -c). Also, if you are using a GPU version.
- Is the number of cores/nodes you request optimal? Trying a short simulation could help.
- Is your simulation scaling well?
- Could KNL nodes speed up your workflow?
- Use **job-usage** script on the command line, this will allow you to monitor the resources usage in your local browser



Final comments

- Upon asking a question to the support team: **Make a folder with the case which displays the issue. Including all relevant files would be useful.**
- Keep the topics of the support tickets independent (one ticket per case)
- Avoid using SLURM batch scripts from other centers, use those provided in our web site.
- Don't load modules in your .bashrc file, this can caused troubles with software.
- Don't forget to cite HPC2N and the SNIC project upon publishing.