Compchem weekly seminar 23/01/2018

Gromacs benchmarks

Gromacs 2019 Compilation

- Many different possibilities
  - icc vs gcc compiler
  - Build own FFTW or MKL
  - SIMD instructions on new CPUs
  - MPI or not
  - Infiniband support (or lack thereof)
  - GPU support or not, CUDA and drivers versions…

Executables

Executable	Uses	Compiler options
gmx	Serial run, preparation, analysis	gcc, build own fftw
mdrun_mpi	MD on “normal”	icc, mkl, avx_256
mdrun_avx2	MD on “cluster-e5v4”	icc, mkl, avx**2**_256
mdrun_gpu	MD on “gpu” (K20, not yet)	icc, mkl, avx_256, CUDA 10
mdrun_gpu_avx2	MD on “gpu-umr850” (K40) MD on “gpu-umr1248-gtx1080”	icc, mkl, avx**2**_256, CUDA 10

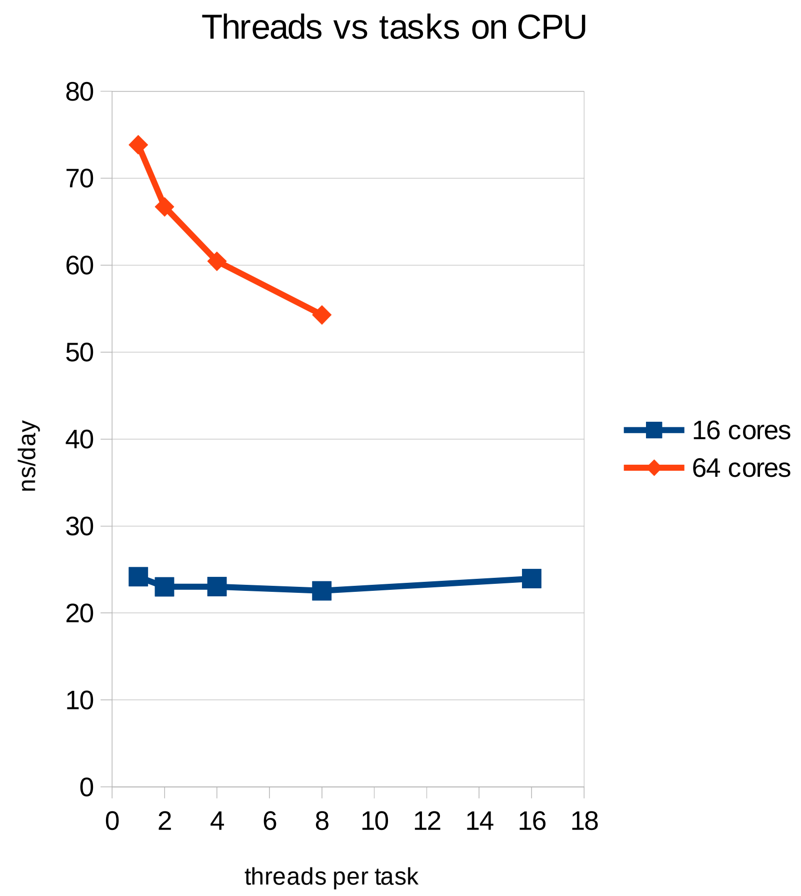
Benchmarks

- System
  - 128 POPC, Slipids, NPT, free simulations, 35000 atoms, 200 ps
- Parallelization options
  - MPI tasks
    - OpenMP threads
      - CPU cores
  - En pratique tasks * threads = ncores
  - Can be run on several nodes

MPI	Task 1				Task 2			
Open MP	Thread 1-1		Thread 1-2		Thread 2-1		Thread 2-2	
Core	1	2	3	4	5	6	7	8

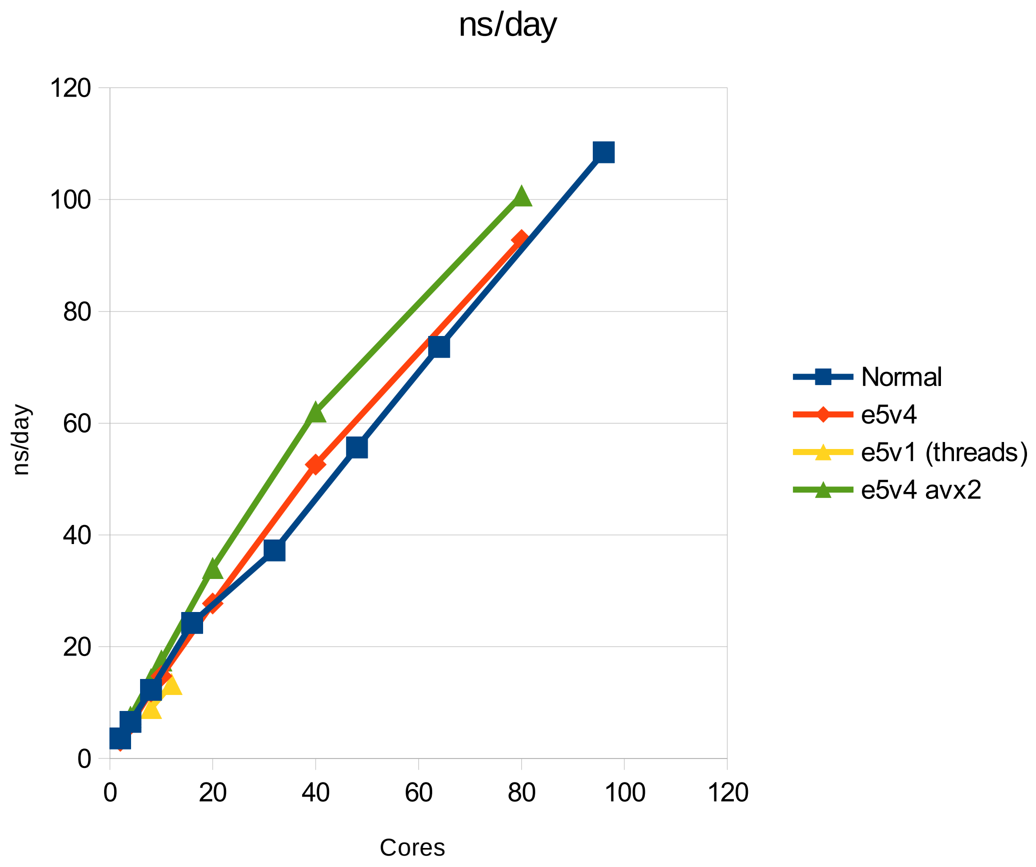
CPU : tasks vs threads

- Use as many tasks as cores
  - #SBATCH --cpus-per-task=1
  - #SBATCH --threads-per-core=1



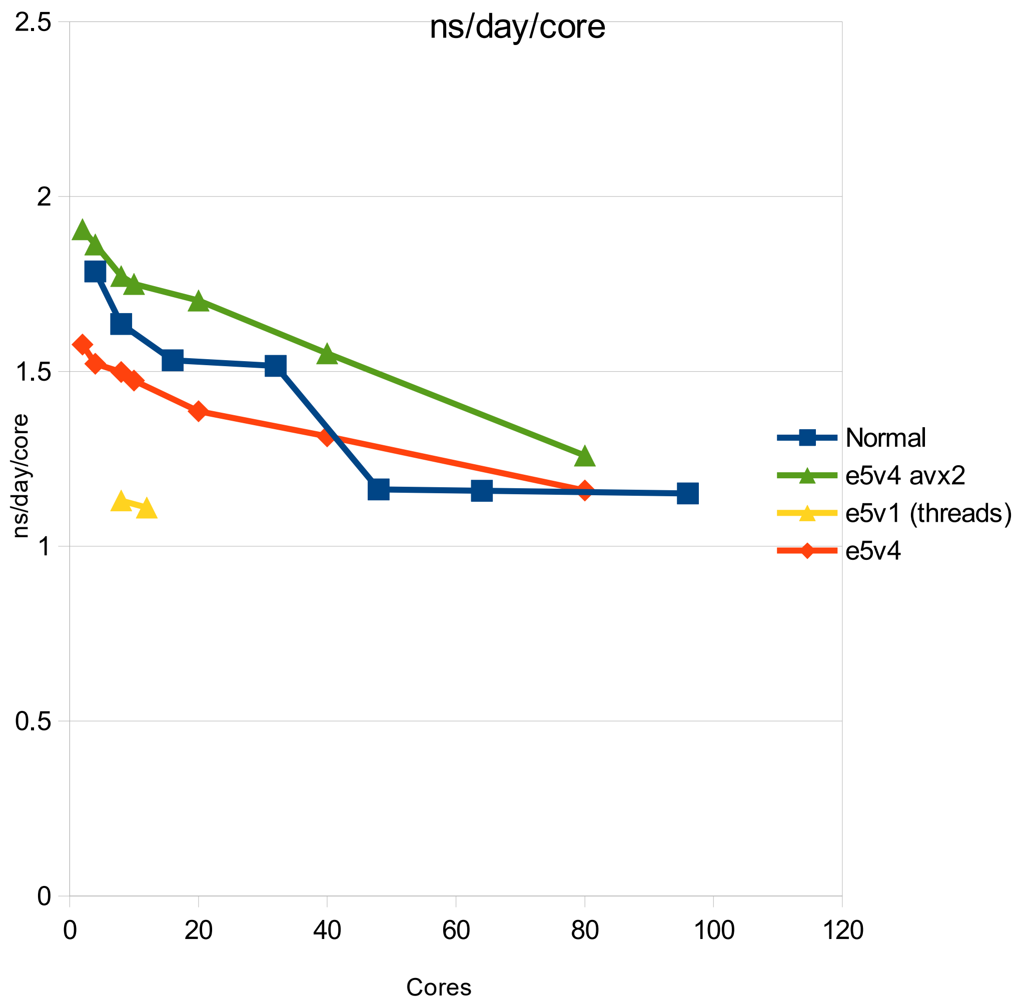
CPU performance

- Almost linear scaling up to what we can use
- Newer CPUs (e5v4) are slightly better
  - especially if they benefit from latest SIMD instructions



CPU performance

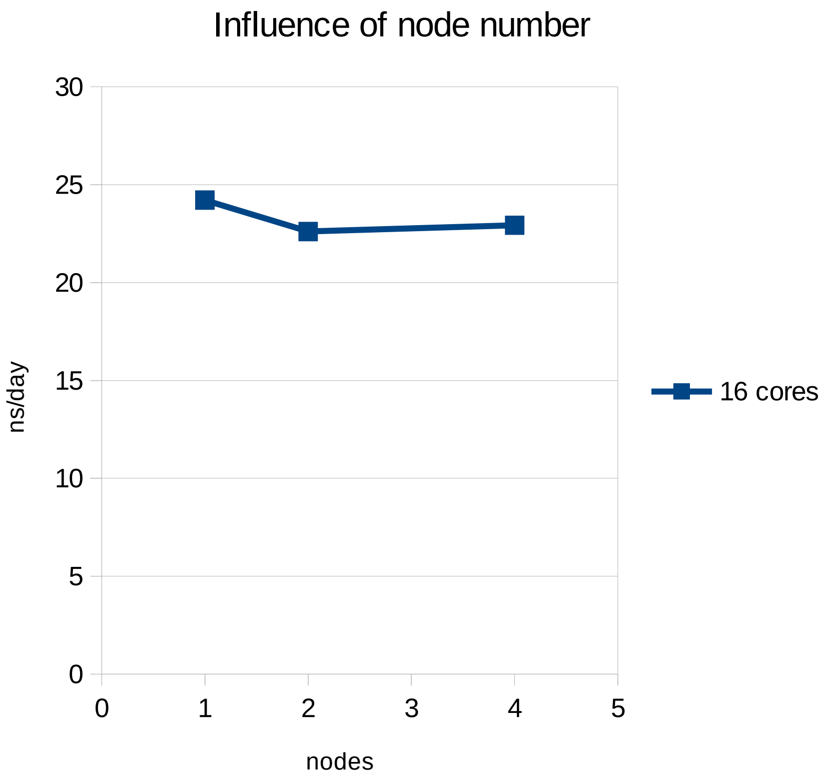
- Almost linear scaling up to what we can use
- Newer CPUs (e5v4) are slightly better
  - especially if they benefit from latest SIMD instructions
- Performance degradation over multiple nodes with “normal”



CPU Multiple nodes

- Again, better use as few nodes as possible, even if infiniband limits the penalty

- The tests were performed on empty nodes, expect higher performance degradation on 100% cluster use

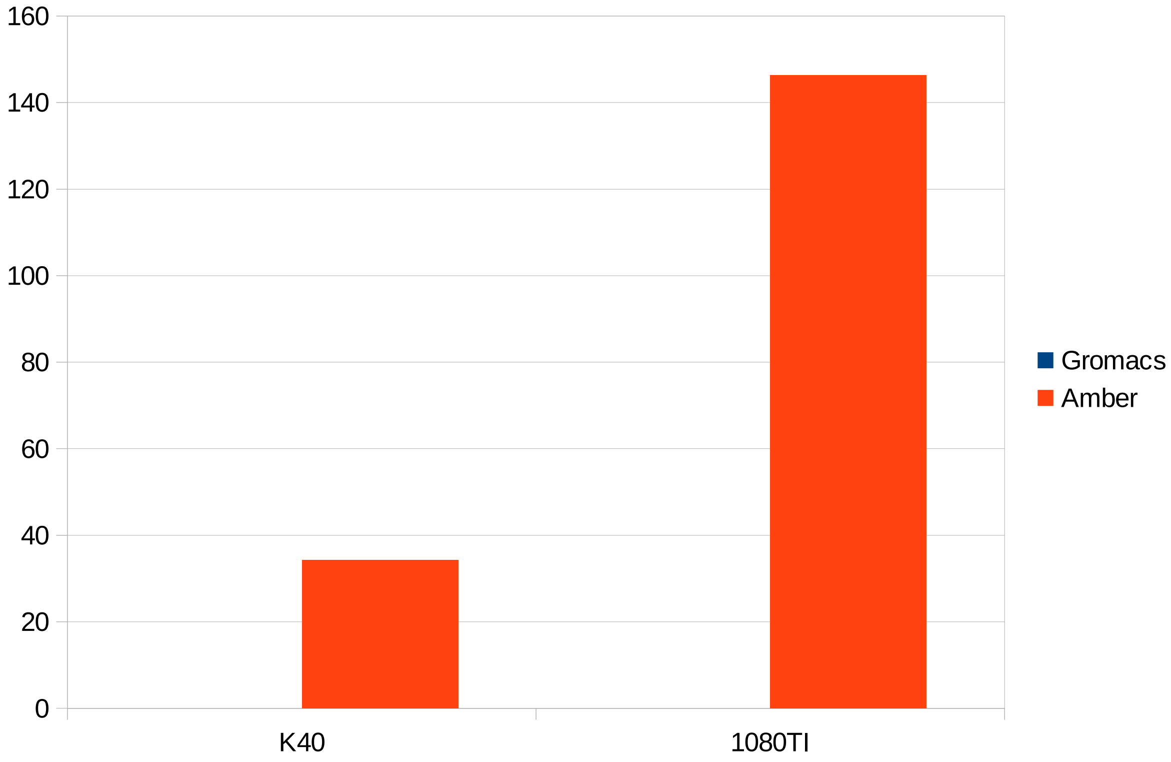


GPU parameters

- GPU itself (1080, K20, K40…)
- Cores : tasks vs threads
- Cores per GPU
- What is done by the GPU?
  - Bonded, non-bonded, PME…
- Number of GPUs
- Simultaneous jobs on the node

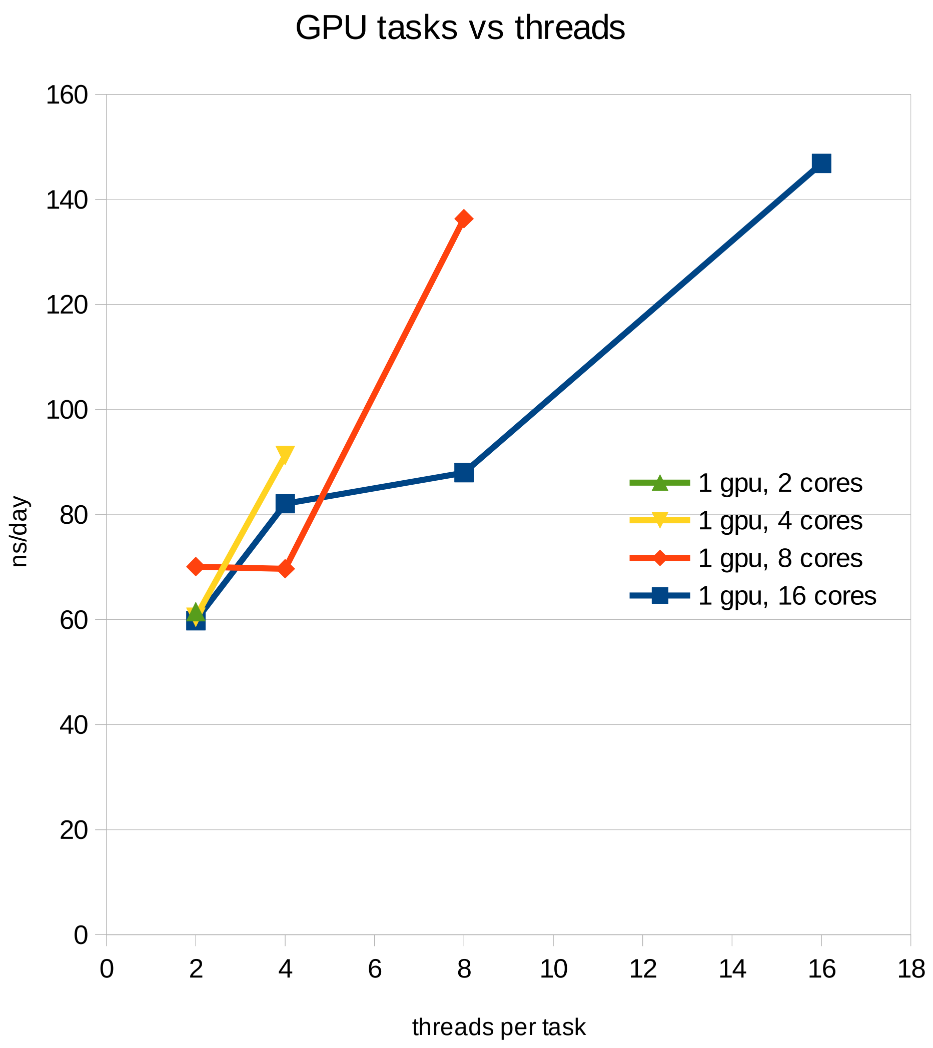
Which GPU ?

- Tested only with amber yet
- 1080ti are 4x faster than K40



GPU : tasks vs threads

- Use as many threads as possible
- 1 GPU can benefit from more cores
  - But other 7 GPUs are idle…
- On our nodes, ideal is 1 task of 2 threads per GPU



What is done by the GPU?

- 1 GPU (1080), 2 threads

non-bonded	PME	bonded	ns/day
CPU	CPU	CPU	5
GPU	CPU	CPU	36
GPU	GPU	CPU	61
GPU	GPU	GPU	85

Number of GPUs per job

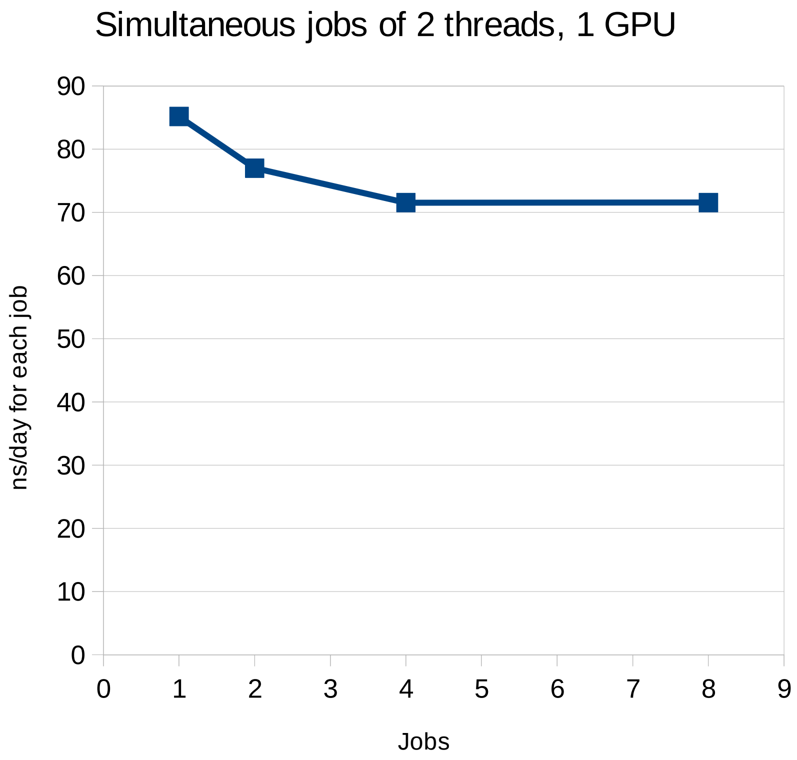
- Always do PME on GPU
- To do PME on GPUs using multiple GPUs, you have to dedicate one GPU to it.
- No benefit of using 2 GPUs because one is waiting for the other
- Benefits start from 4 GPU
- Most efficient use is 1 GPU

Cores	GPU	PME	ns/day	ns/day/ GPU
2	1	GPU	85	85
4	2	CPU	67	33.5
4	2	GPU	81	40.5
8	4	GPU	144	36
16	8	CPU	119	15
16	8	GPU	210	26

Simultaneous jobs

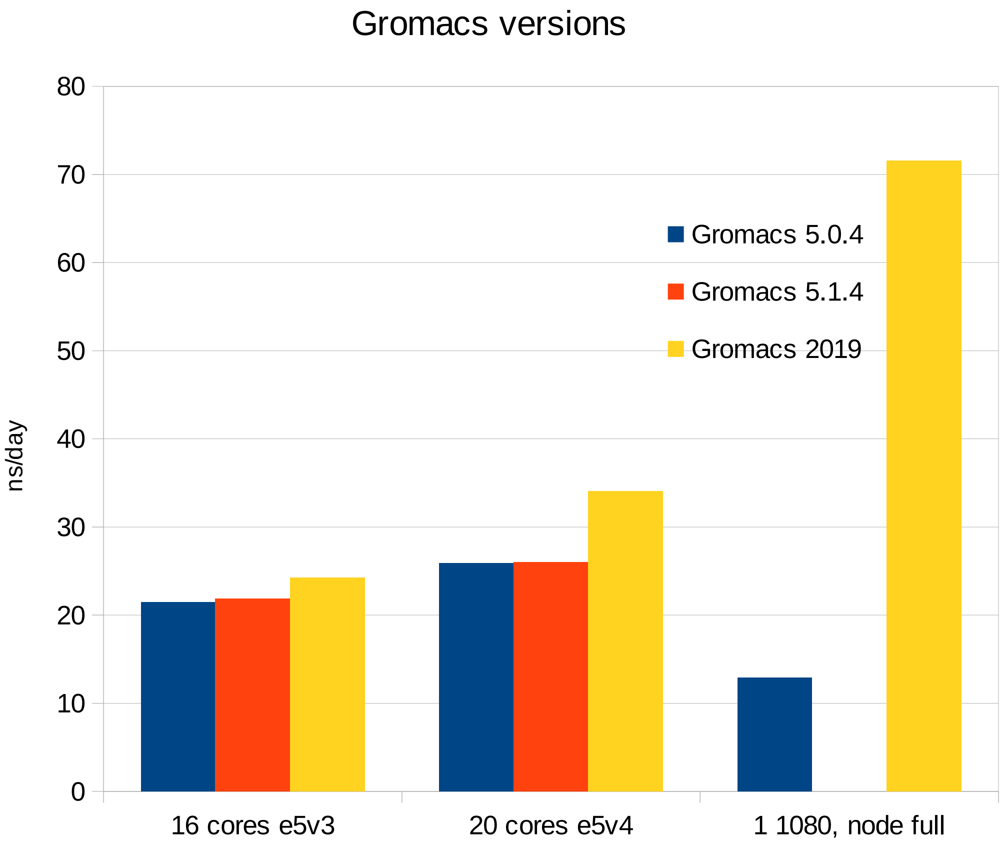
- All previous numbers are jobs running on empty nodes

- 16% performance decrease when node is full or half-full



Gromacs versions

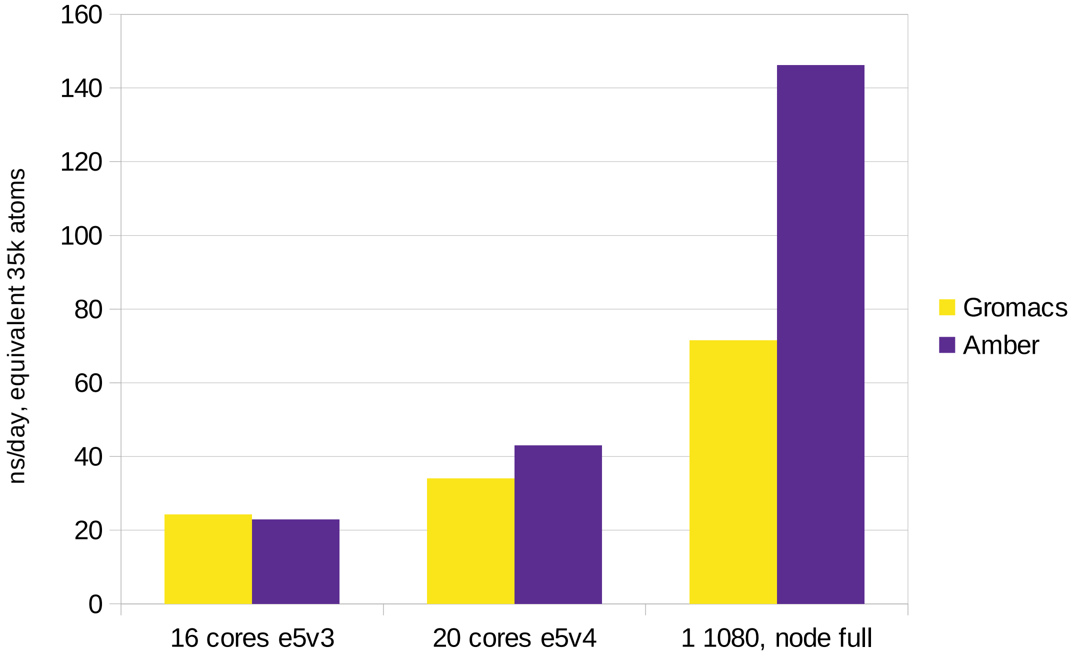
- Gromacs 2019 is :
  - 13% better on “normal”
  - 32% better on “e5v4”
  - 455% better on GPU (1080)



Gromacs 2019 VS Amber

- CPU : about the same
- 1 GPU : Amber wins (twice as fast)

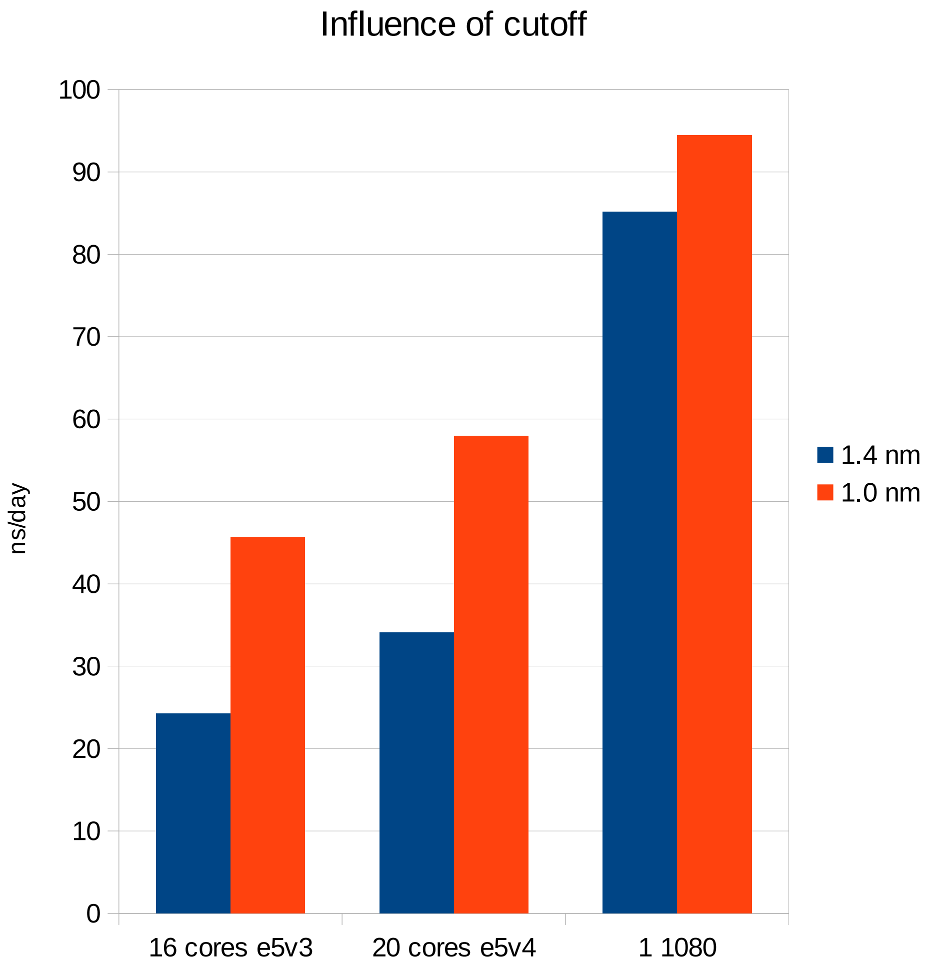
- Not exactly identical systems and parameters
- Amber performance scaled to number of atoms



Influence of cutoff

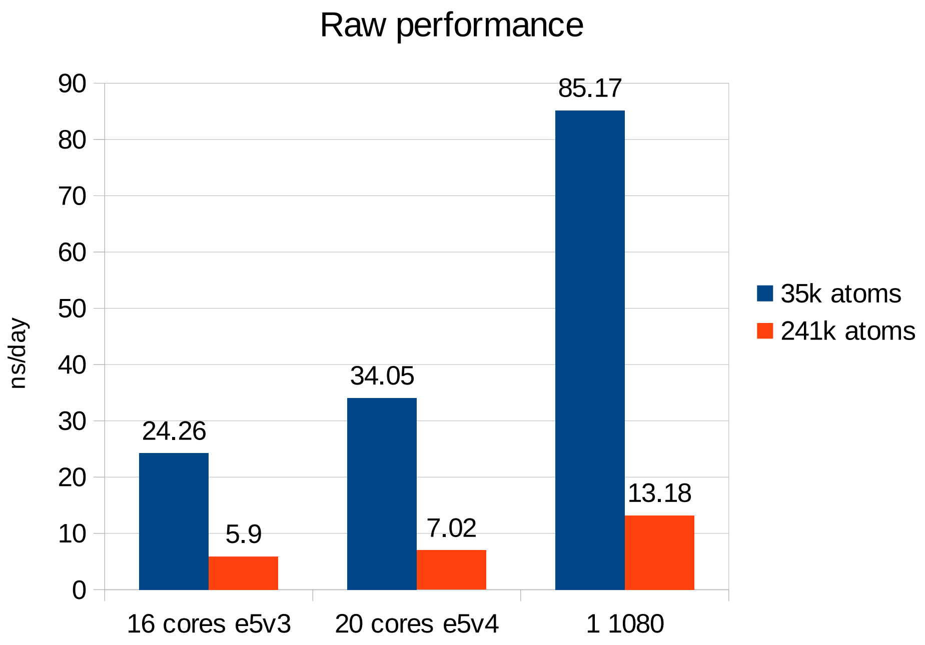
- Slipids uses 1.4 nm cutoffs
- Their latest paper suggests to try 1.0 nm

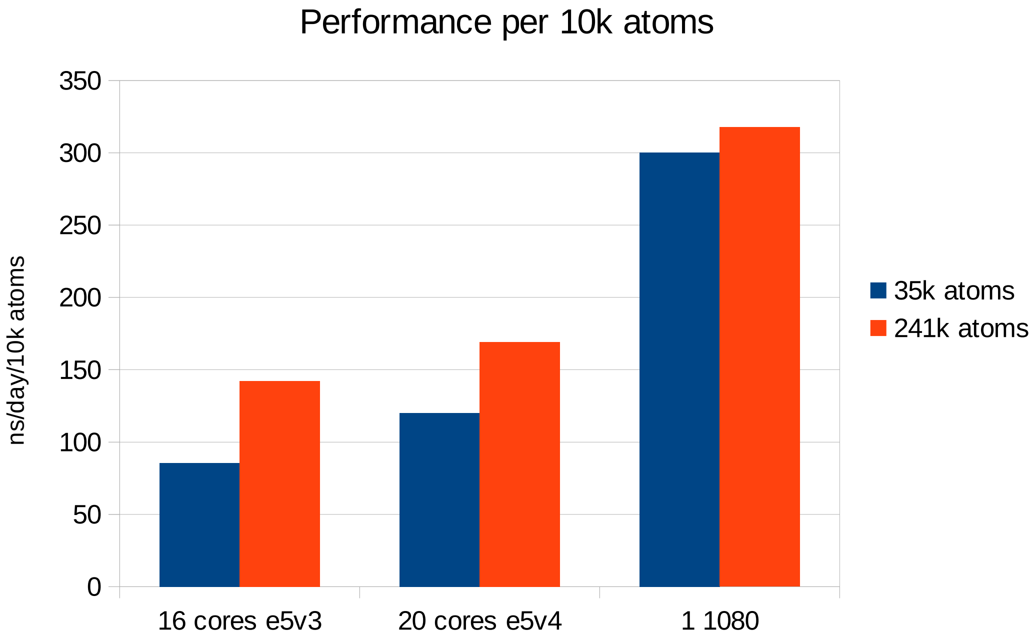
- 88% faster on e5v3
- 70% faster on e5v4
- 10% faster on GPU



System size

- 241k atoms: transporter with amberff and shorter cutoff





Summary

- If you use Gromacs, switch to 2019
  - I will provide optimized sample batch files in my $HOME
- Gromacs is now much faster on GPU
  - 1 1080TI is 2.5x faster than 20 CPU cores
  - Amber is still 2x faster than Gromacs
  - Gromacs team is working on it
- Our systems scale relatively well with number of cores or atoms
  - You can choose to go faster on 1 job or do more jobs
  - Try reducing cutoff and evaluate the effects
- Which new nodes to buy? It all depends on cost