3. Running & Performance

Weine Olovsson
National Supercomputer Centre (NSC), Linköping University
SNIC-PRACE training, online @NSC 22-23rd Feb 2022

LINKÖPINGS UNIVERSITET

VASP - Best Practices Workshop

NSC

SNIC

PRACE

https://www.nsc.liu.se/

https://www.snic.se/

https://training.prace-ri.eu/

Introduction

- General considerations
- Focus on practical aspects of running VASP

  ...at specific supercomputer centres
- Influential parameters, NPAR/NCORE, ALGO, NSIM, KPAR, ...
- Memory usage
- Benchmarks, examples
- Common problems

... clickable links are underlined

Parallel calculations

Examples for different SNIC HPC clusters

Computation - considerations

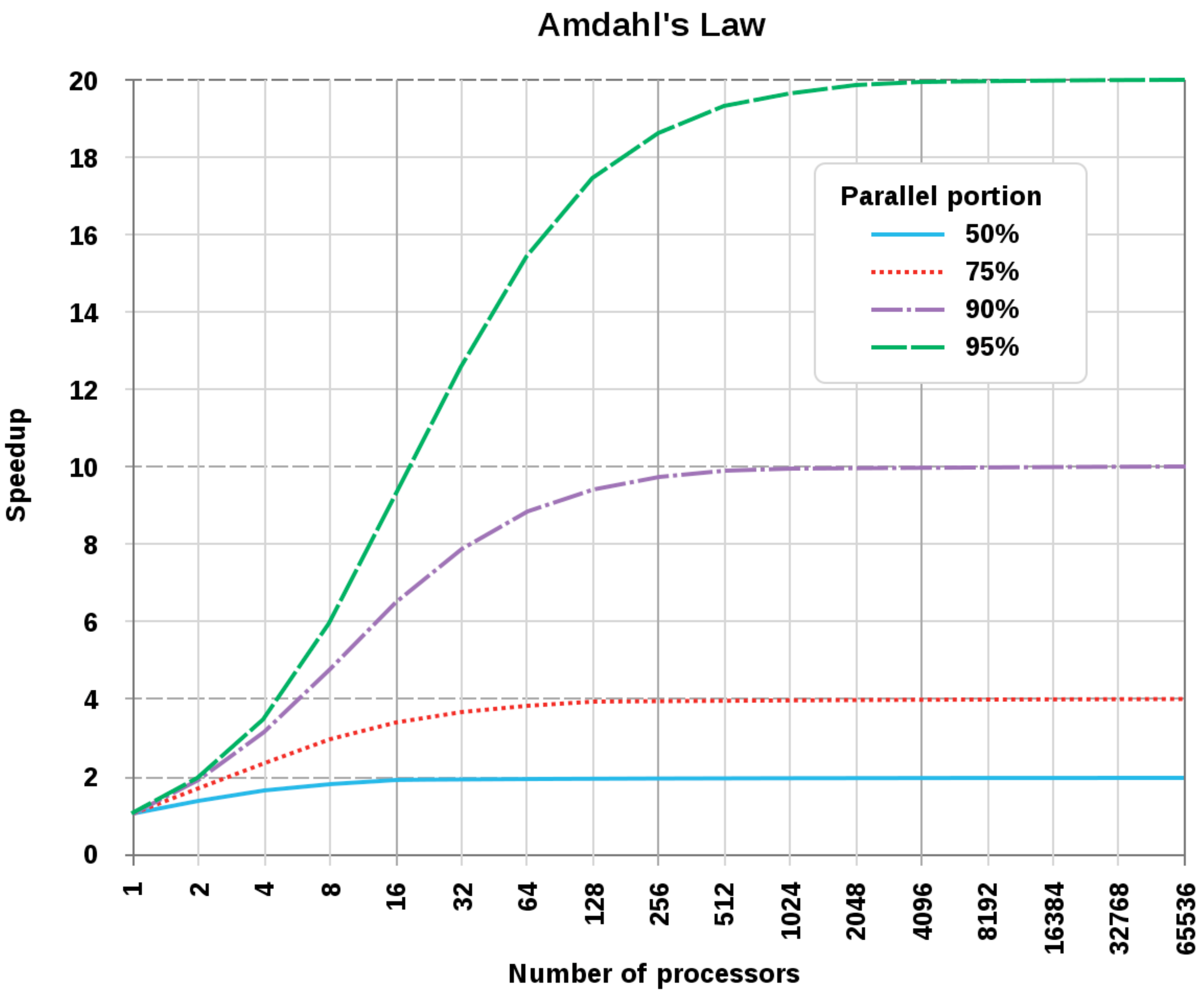
Efficiency:
Running as many jobs as possible for a given allocation of computer time

Speed:
The amount of time (real, “human time”) to run a specific simulation from when it starts

Time-to-solution:
Speed + the time waiting in queue

@Kebnekaise, Tetralith: wall-time limit **7 days**
@Dardel: **24h** (7 day option)

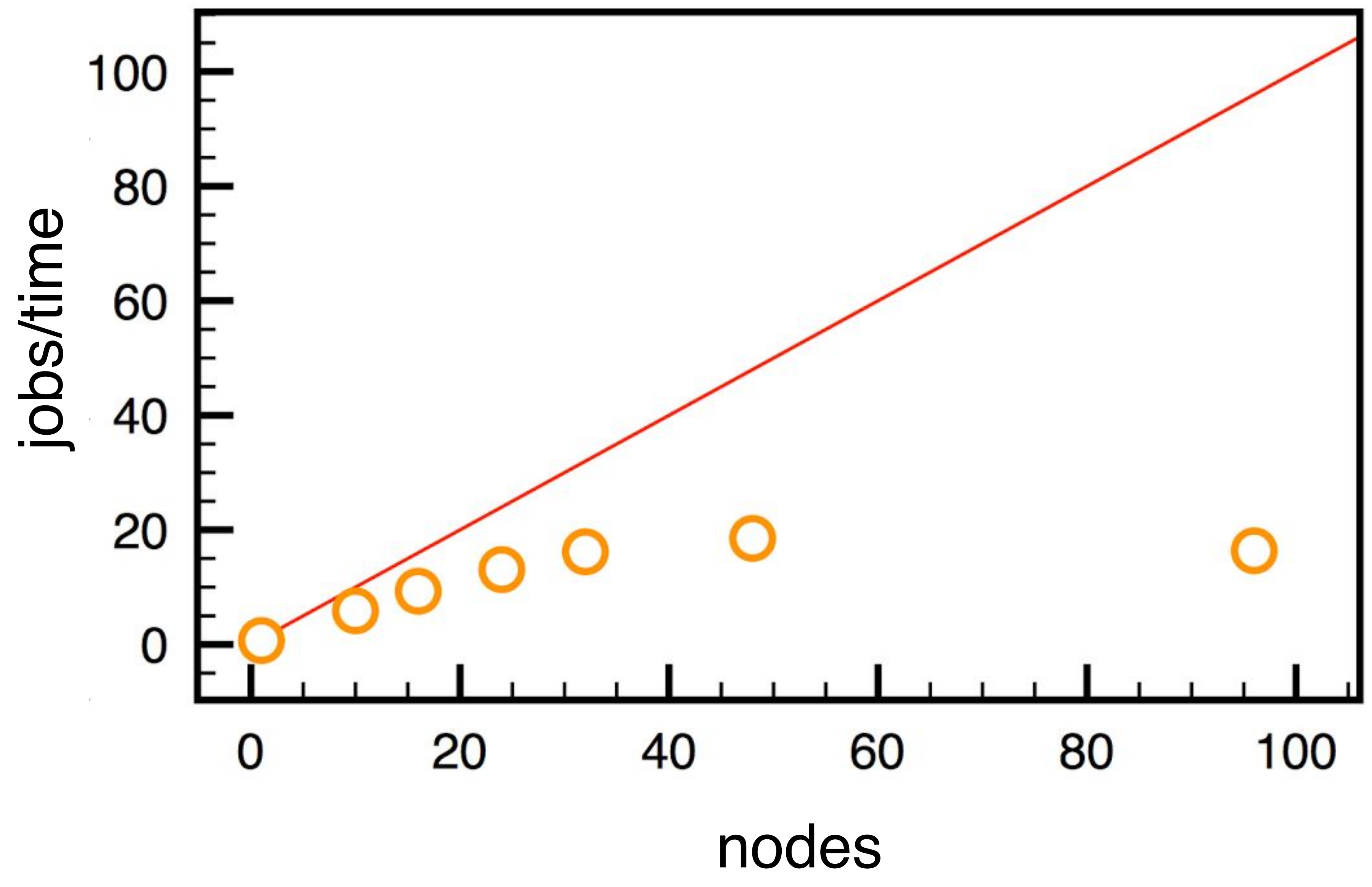
Parallelization - limitations

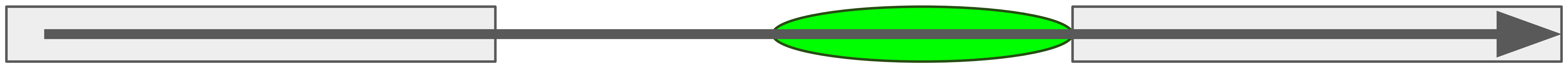


Simple scaling analysis

A minimal scaling analysis can save lots of allocated core hours…

1. Tool your runscript to time your simulation
2. Run an initial best guess number of cores (n)
3. Run the same test on half the number of cores (n/2)
4. Score = time(n/2) / time(n)





Hardware - affects best practices

- Kebnekaise (HPC2N)

432 x 1 node (Intel Xeon E5-2690v4) = 28 cores (128GB RAM)

32 x + 2xGPU (NVidia K80) **vasp-gpu version!**

52 x 1 node (Intel Xeon Gold 6132) = 28 cores (192GB RAM)

10 x + 2xGPU (Nvidia V100) **vasp-gpu version!**

different best practices

36 x 1 KNL (Intel Xeon Phi 7250) node = 68 cores (192GB RAM)

- Abisko (HPC2N)

318 x 1 node (AMD Opteron 6238) = 48 cores (128GB RAM)

Hardware - affects best practices

- **Tetralith** (NSC), Intel Xeon Gold 6130 2.1GHz

  1 node = 32 cores (96GB RAM, fat node 384GB)

  1832 x **60 x**

New!
- **Dardel** (PDC), HPE Cray EX, AMD EPYC 2.25 GHz

  1 node = 128 cores (256GB RAM, 512GB, 1024GB, 2048)

  488 x **20 x** **8 x** **2 x**

Running & performance

Important INCAR settings and benchmarks

Starting advice (reminder)

- Read the documentation!
- VASP default settings   good starting point
- Caution: “inherited” starting files
- Avoid overly complex INCAR
- Possible differences in centres installations

refer to respective webpages / documentation

Quick check your run

- How much/what resources to use?
  - Check NBANDS `$ grep NBANDS OUTCAR`
  - Use ca. 8 bands/core
- How long will it take? `$ grep LOOP OUTCAR` `$ grep LOOP+ OUTCAR`
  - scales with k-points (IBZKPT) `$ grep k-points OUTCAR`
- Does it converge? `$ cat OSZICAR`
- Problems? `$ less slurm*.out`

Quick check your run: tools

- `sacct`
- login to node & run `top`
- @Tetralith: `jobload`, `jobstats` & `jobsh`

`$ man <command>`

"-s r" for running job

`$ sacct --user=<username> -X --format=Elapsed,State,AllocCPUS%9,CPUTimeRaw%20 --starttime=2019-10-01`

`$ sacct -e` example: `--format=JobID,Submit,Start,End,Elapsed,Nodelist,State,AllocCPUS%9,CPUTimeRaw%20`

`$ seff <jobid>`
summary of run

`$ squeue -u <username>`
`$ scancel <jobid>`

@Tetralith:
`$ jobload <jobid>`
`$ jobsh <node>`

INCAR parameters

- PREC - “precision”, ENCUT and FFT grids
- ENCUT - plane wave energy cutoff
- ALGO - wf optimisation
- NBANDS - if not set, auto-determined
- NSIM - for RMM-DIIS algorithm (ALGO)
- NCORE or NPAR - bands treated in parallel
- KPAR - k-point parallel

INCAR parameters

accuracy / method

- PREC - "precision", ENCUT and FFT grids
- ENCUT - plane wave energy cutoff

Completeness of basis-set

Recommended to set!

- ALGO - wf optimisation
- NBANDS - if not set, auto-determined

Must be the same for Etot comparison!

parallel calcs.

- NSIM - for RMM-DIIS algorithm (ALGO)
- NCORE or NPAR - bands treated in parallel
- KPAR - k-point parallel

PREC

- PREC = “precision”, sets ENCUT and FFT grids
- PREC = Normal, **default**
- PREC = Accurate, highly accurate forces
- OBS: Recommended to set ENCUT by hand

More on accuracy

- NGX, NGY, NGZ coarse plane wave FFT grid
  can edit directly (otherwise PREC)
- NGXF, NGYF, NGZF finer FFT grid
  - also see ENAUG
- LREAL=.FALSE. default, might be needed for high accuracy

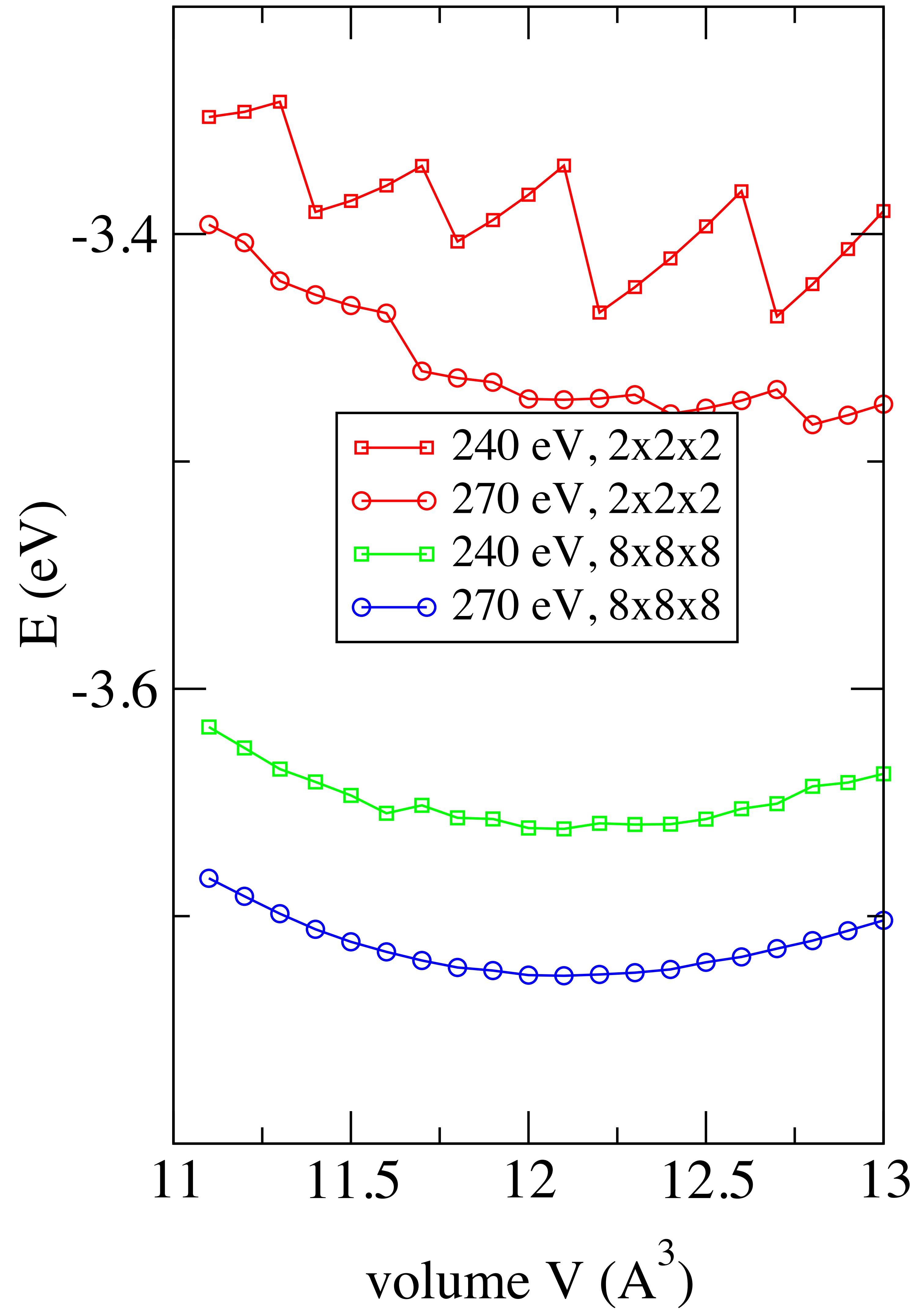
if proj. operators determined in real space, or not

otherwise use faster: LREAL = Auto

In some cases, need VASP installation with no special optimization flags

Convergence, ENCUT and k-mesh

- <u>Cu example by G. Kresse</u>
- Basis-set **changes** with volume
- Cell-shape relaxations, increase ENCUT = ENMAX x1.3
- Read section on <u>structure relaxation</u>



NBANDS

- NBANDS = NELECT/2 + NION/2 (ISPIN=1)
- May change due to parallelization!
- **Easy to divide**, 2^n, 4, 8, 12, 16, …
- select NBANDS = 511 or 512?
- Min limit, 1 band/core
- **Affects Etot!**

Run e.g. quick job to check NBANDS:

```
#SBATCH --reservation=devel
```

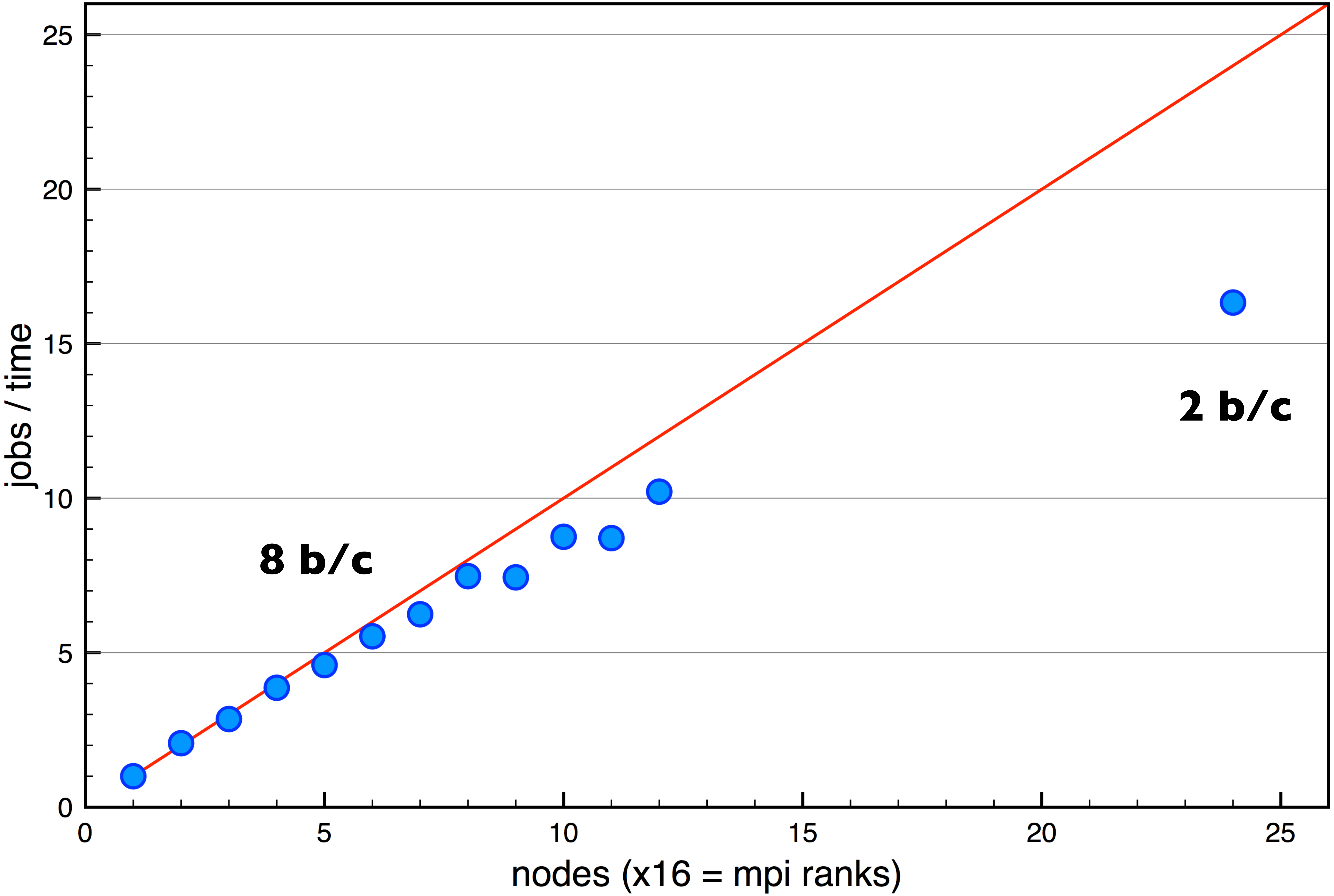
@Tetralith

```
$ grep NBANDS OUTCAR
```

How many cores - efficient and/or fast?

- Start from # of bands, **NBANDS**
- 1 band/core: typically inefficient
- 2 bands/core: ~50% parallel efficiency
- 8 bands/core: good starting point
  - try e.g. **cores ≈ NBANDS / 8**

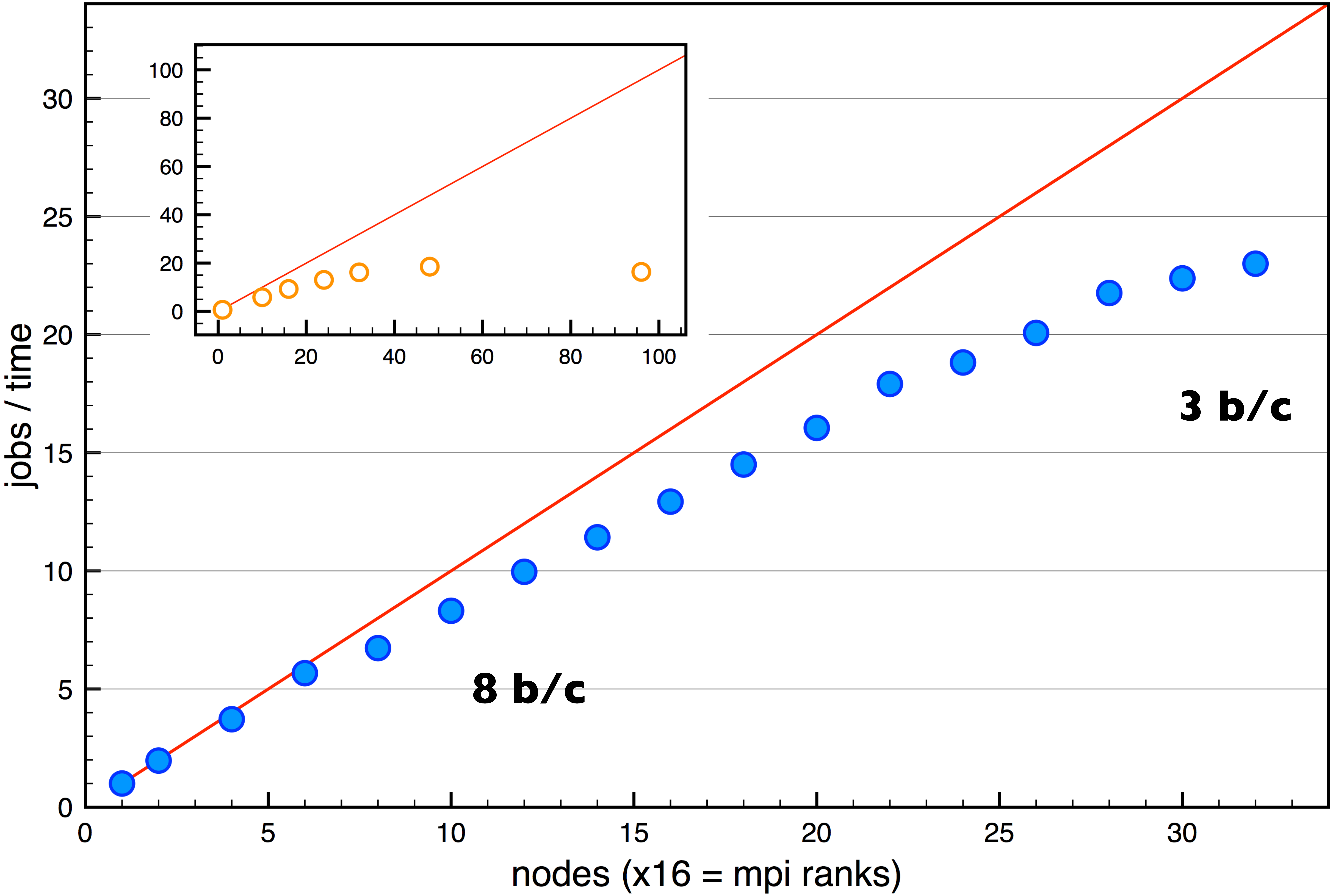
Si-H/Ag(111) 129 atoms, VASP PBE @Triolith (old)



NBANDS=750
4 k-points

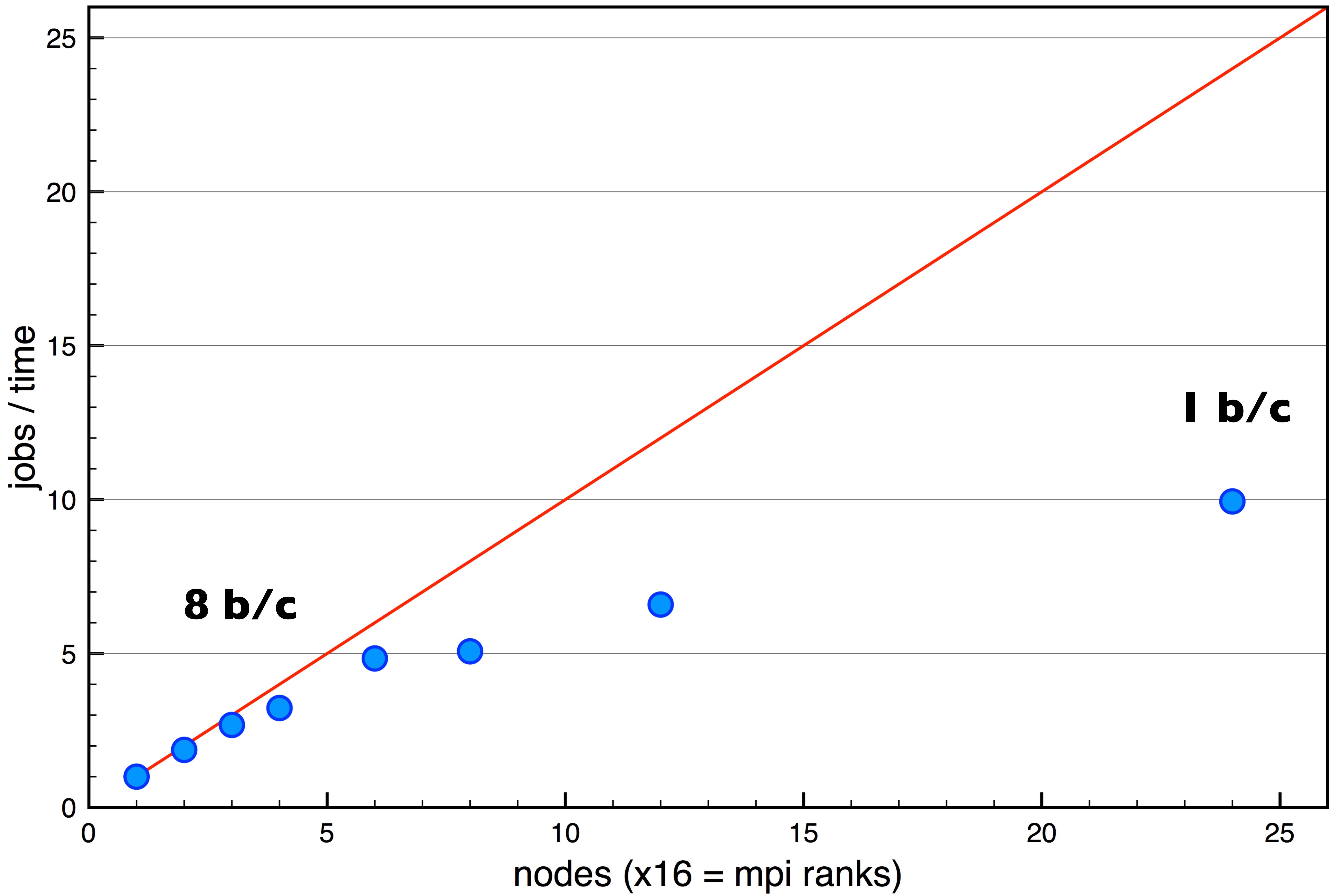
Triolith had **16** c/node
Tetralith: **32** c/node
Kebnekaise: **28** c/node

GaAsBi 512 atoms, VASP PBE @Triolith (old)



NBANDS=1536
4 k-points

GaAsBi 128 atoms, VASP HSE06 @Triolith (old)

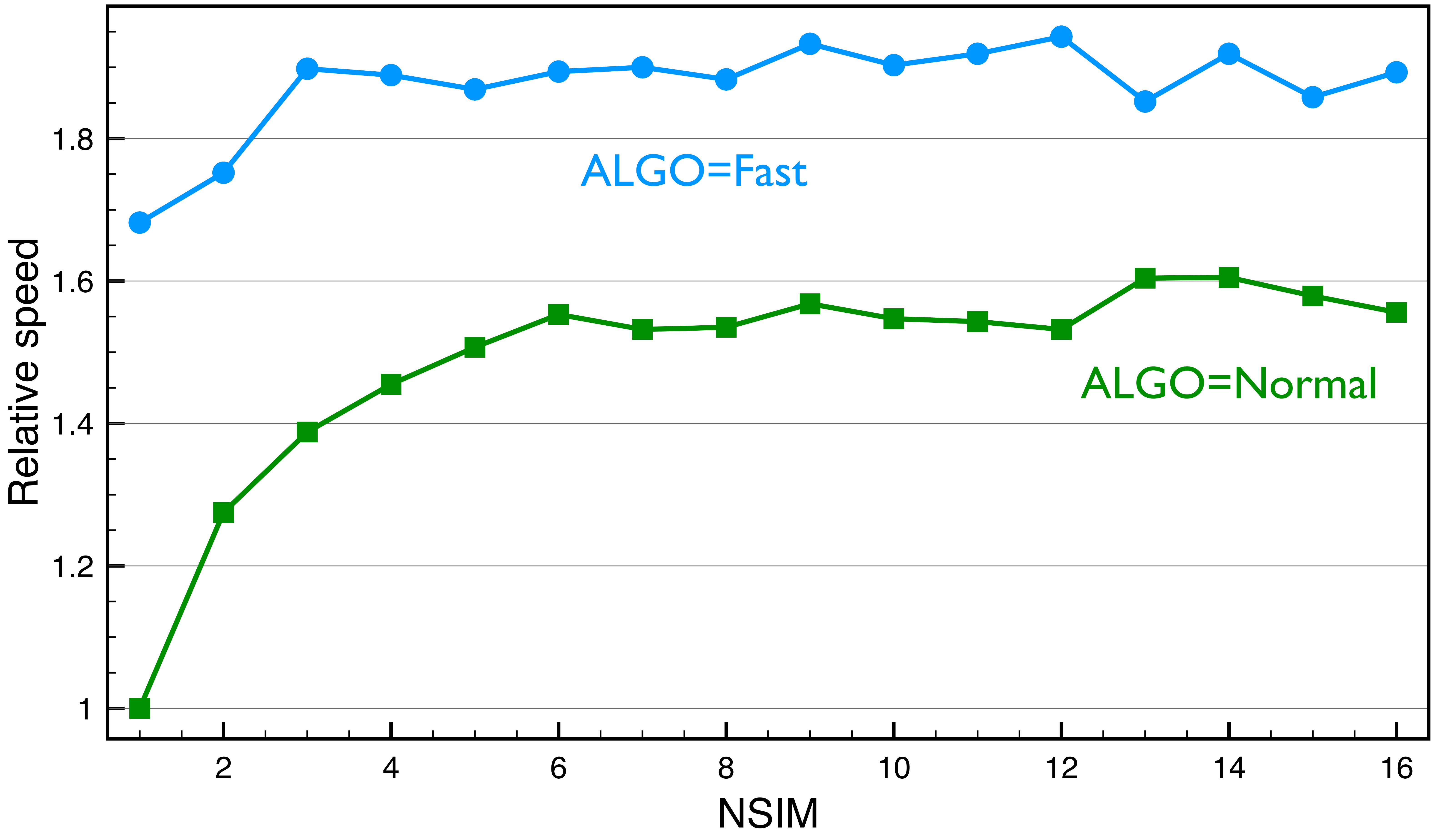


NBANDS=384
12 k-points

ALGO & NSIM

- Blocking mode for RMM-DIIS algorithm
- ALGO = Fast (Dav + R-D) / VeryFast (R-D)
- ALGO = Normal (Davidson algorithm), **default**
- **not for hybrid-DFT**, HSE06 (Damped, All, Normal)
- NSIM = 4, **default** usually good (CPU)
- Kebnekaise/Tetralith: NSIM = 4 (or higher)
- Beskow: NSIM = 2

Si-H/Ag(111) 129 atoms, VASP PBE @Triolith (old)



default NSIM=4 seems OK here

NBANDS=750, 4 k-points

NCORE or NPAR

(default)

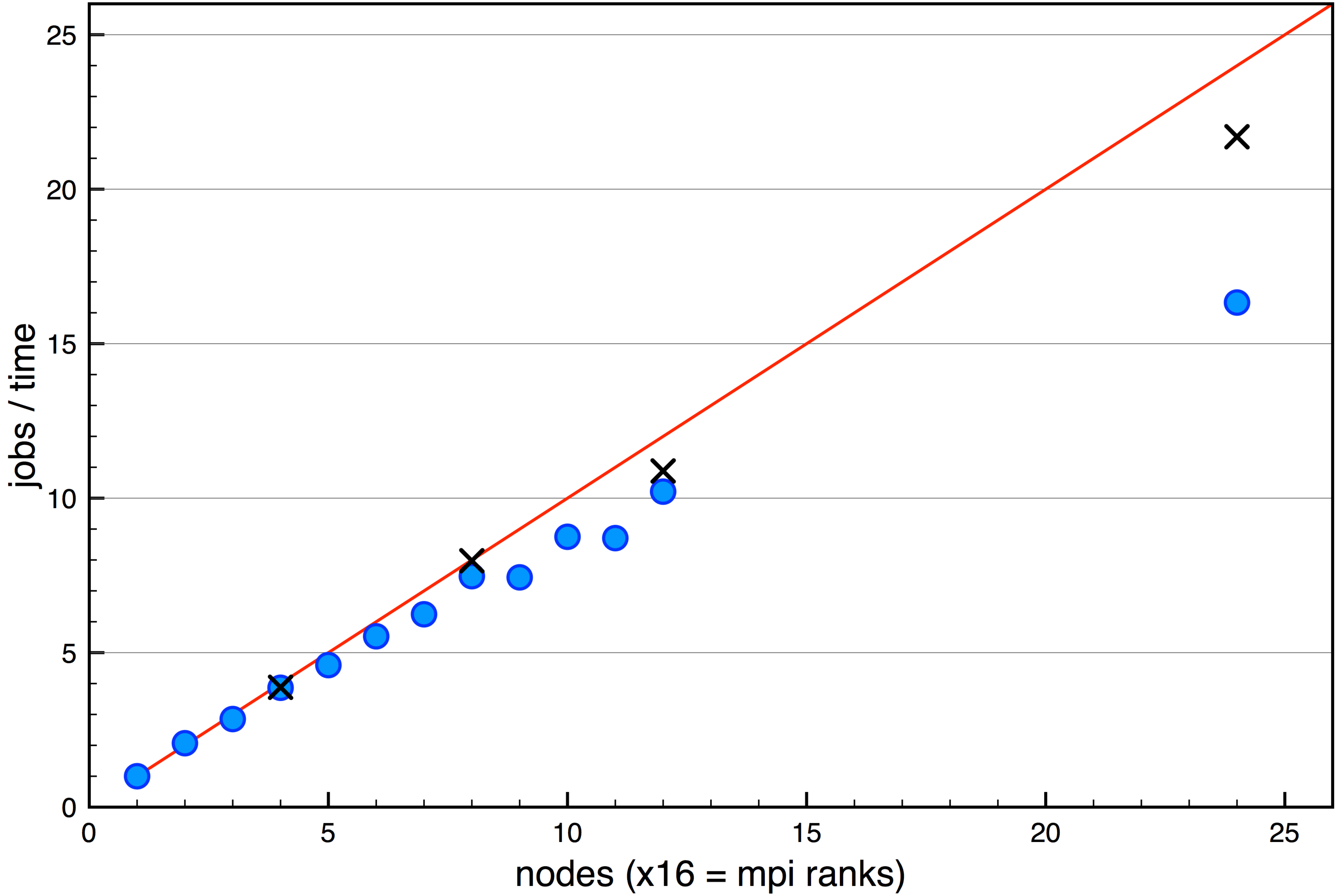
- cores per orbital / bands treated in parallel
- Davidson & RMM-DIIS algorithm
- ALGO = Normal & Fast, VeryFast
- NPAR = 1, saves memory
- NPAR = number of compute nodes
- NCORE = cores per compute node (or socket)

I find it easier to use NCORE, e.g. on Tetralith (if full node):
NCORE=32

KPAR

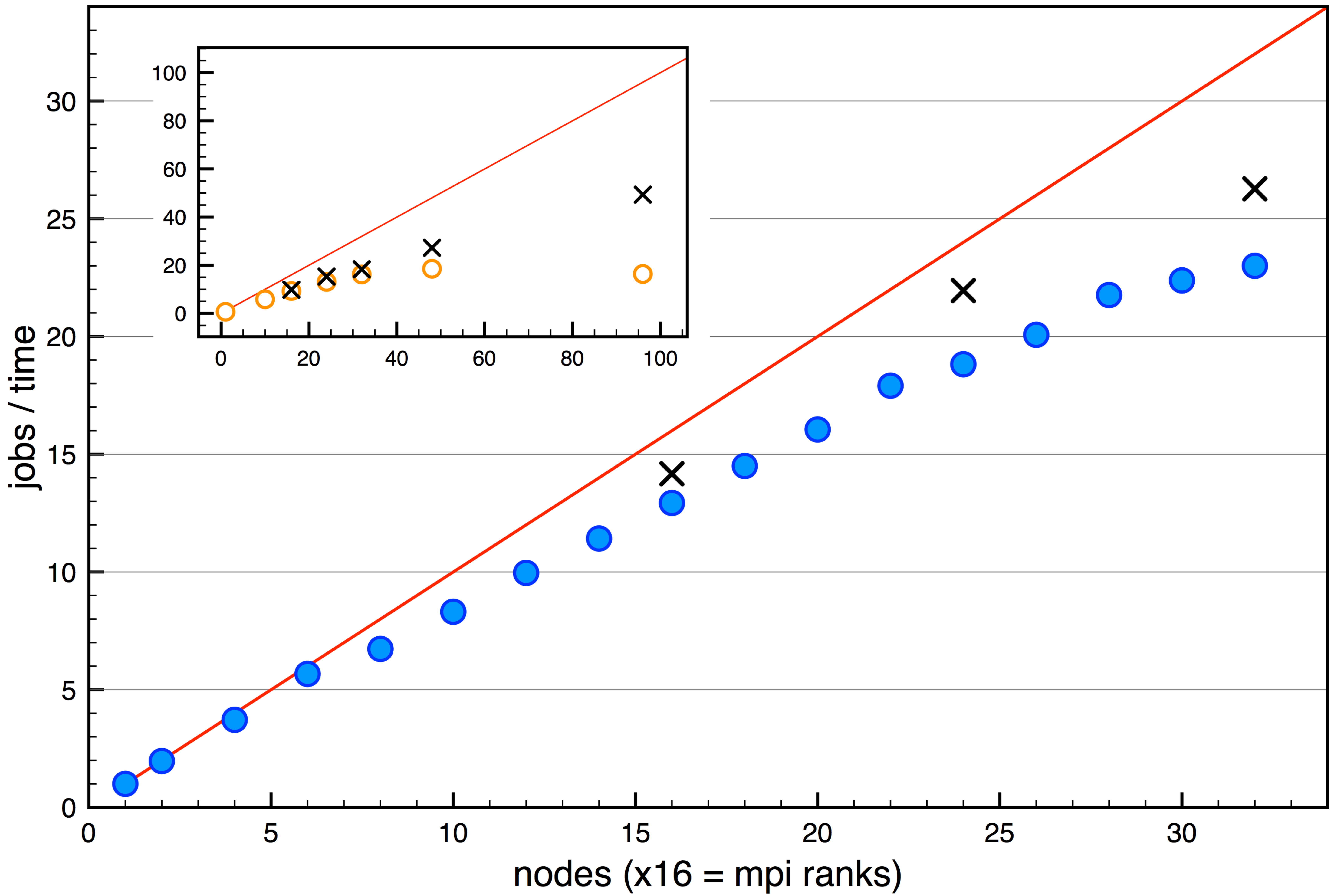
- KPAR = number of k-points treated in parallel
- in particular, good for **hybrid-DFT** jobs
- increase cores at least 2x
- try **KPAR = min (nodes, k-points)**

Si-H/Ag(111) 129 atoms, VASP PBE @Triolith (old)



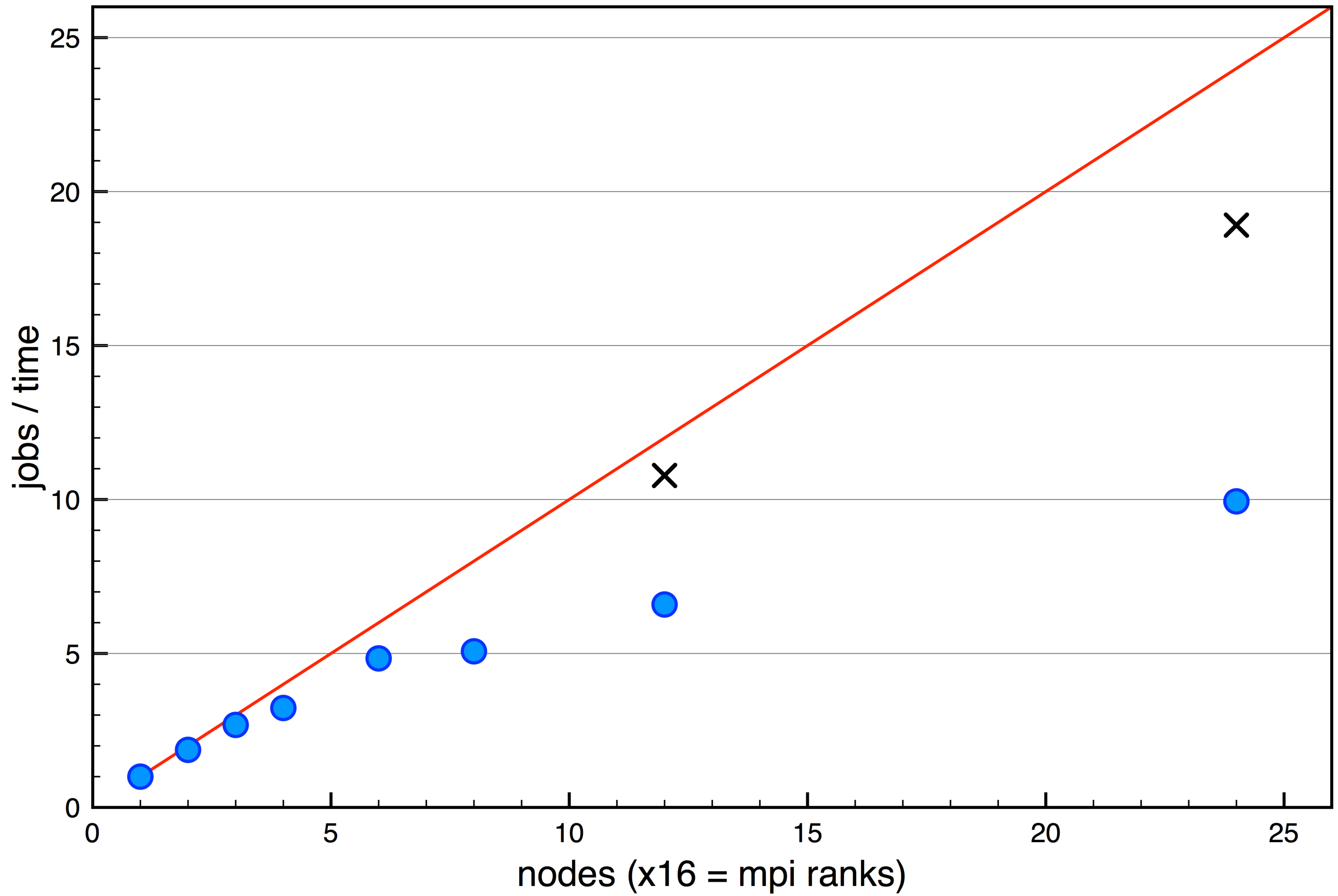
NBANDS=750
4 k-points

GaAsBi 512 atoms, VASP PBE @Triolith (old)



NBANDS=1536
4 k-points

GaAsBi 128 atoms, VASP HSE06 @Triolith (old)



NBANDS=384
12 k-points

Quick comparison

GaAsBi 512 atoms, VASP PBE, NBANDS = 1536, 4 k-points

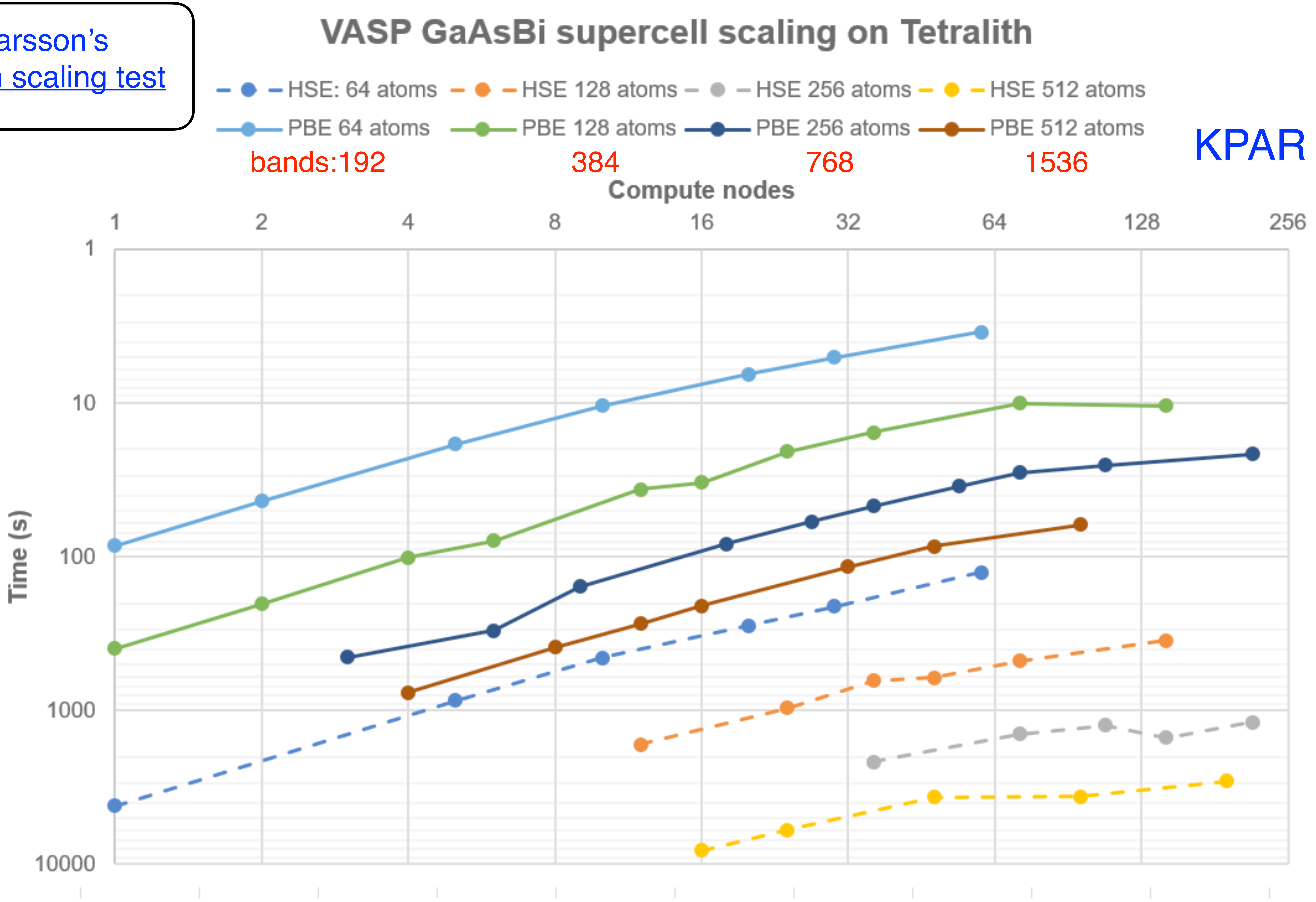
	@Tetralith,	6 nodes, NCORE=32, NSIM=30:	576s
		4:	625s
	@Kebnekaise,	7 nodes, NCORE=28, NSIM=30:	707s
		4:	768s
retired	@Beskow,	6 nodes, NCORE=32, NSIM=4:	1074s
		24 2:	1593s

GaAsBi 128 atoms, VASP HSE06, NBANDS = 384, 3 k-points

	@Kebnekaise,	6 nodes on 24c, NSIM=1:	1598s
retired	@Beskow,	6 nodes on 24c, NSIM=1:	2146s
		on 32c :	2044s

Peter Larsson's
Tetralith scaling test

KPAR is used



As a practical example, let us calculate how many core hours that would be required to run 10,000 full SCF cycles (say 100 geometry optimizations, or a few molecular dynamics simulations). The number of nodes has been chosen so that the parallel efficiency is > 90%:

Atoms	Bands	Nodes	Core hours
64	192	5	8,000
128	384	12	39,000
256	768	18	130,000
512	1536	16	300,000

The same table for 10,000 SCF cycles of HSE06 calculations looks like:

Atoms	Bands	Nodes	Core hours
64	192	10	400,000
128	384	36	2,000,000
256	768	36	6,900,000
512	1536	24	13,000,000

For comparison, a typical large SNAC project might have an allocation of 100,000-1,000,000 core hours per month with several project members, while a smaller personal allocation might be 5,000-10,000 core hours/month. **Thus, while it is technically possible to run very large VASP calculations quickly on Tetralith, careful planning of core hour usage is necessary, or you will exhaust your project allocation.**

GPU

Running VASP on GPUs

GPU

- VASP GPU **Cuda** version @Kebnekaise
- Different optimization than CPU
- KPAR - k-point parallelization ok
- NSIM - very important!
  - e.g. test with NSIM = 16

different from CPU

- NCORE - not supported
- **GPU RAM** possible bottleneck
- VASP6 faster with **OpenACC**, see link

GPU

Table 1. Summary of timing and speedup for the large test cases.

Test case	CPU time (8 nodes)	GPU time (2 V100)	Speedup
GaAsBi	754.8 (LOOP+)	1067.4 (LOOP+)	5.6x
MD-example	6889.6 (LOOP)	18854.5 (LOOP)	2.9x
576_hh_2x2x2_pbe	36822.2 (LOOP+)	89387.1 (LOOP+)	3.3x
128_hh_3x3x3_hse	151539.6 (LOOP) 161687.9 (LOOP+) (*Note: on* ***5 nodes***)	86025.9 (LOOP) 97206.6 (LOOP+)	8.8x 8.3x

Table 2. Summary of system size and speedup for the test cases.

Test case	NIONS	NBANDS	ISPIN	Speedup
GaAsBi	512	1536	1	5.6x
MD-example	128	1536	2	2.9x
576_hh_2x2x2_pbe	574	1440	2	3.3x
128_hh_3x3x3_hse	126	340	2	8.3x

VASP5 GPU Cuda - benchmark by Xin Li (PDC, KTH)

- Use VASP6 OpenACC

GPU

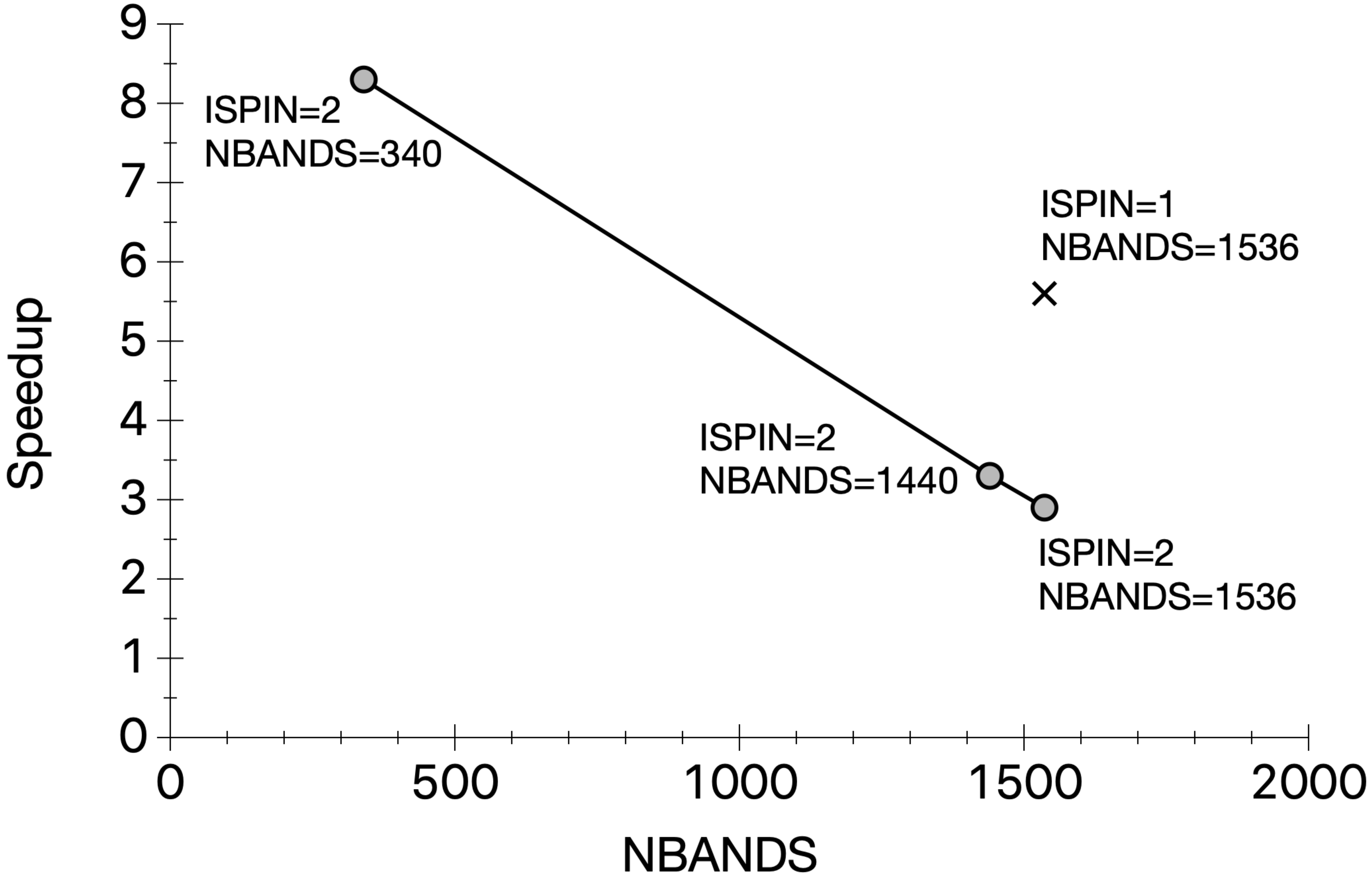


Figure 1. Correlation between ISPIN, NBANDS, and speedup.

VASP5 GPU Cuda - benchmark by Xin Li (PDC, KTH) - Use VASP6 OpenACC

Testing & Benchmarks

In particular on HPC clusters

VASP testing @NSC

- From VASP6, test-suite included
- Peter Larsson's test-suite
- Memory bandwidth important for VASP
- 100's runs for same job -> statistics
- Different builds (flags) for comparison
- Different compiler suites
- Use collection of old problematic jobs…

Problems & Summary

Different problems and their (possible) solution

“Rules of thumb” summary

Possible problems

- Input related many kinds
- Memory (too little)
- **Job size vs. allocation** (mismatch)
  - Inefficient use (wasting core-hours)
- “Difficult” calculations (too costly, not possible)
- **Bugs** (compilers, old versions, …)

  - sometimes from choice of compiler flags which in theory ought to be OK…

Memory (RAM) issues

starting with regular PBE → running HSE06, GW

- changing **type** of calcs.

2x2x2 k-mesh → 4x4x4 k-mesh x 8 k-points

- increasing the **k-mesh**

ENCUT = 400 eV → ENCUT = 600 eV x 1.8

- increasing **energy cutoff**

$$n_{pw} = \propto \mathtt{ENCUT}^{3/2}$$

Memory (RAM) issues …solutions

- Reduce cores/node, e.g. 24c/node, 16c/node
- More nodes (and reduce cores) `#SBATCH --ntasks-per-node=16` `INCAR: NCORE=16`
- @Tetralith: use “fat” memory nodes `#SBATCH -C fat`
- Reduce k-mesh, ENCUT?
- Simplify system?
- Don’t use `--mem` flag

Warning/advice output

Check stdout (slurm-***.out) for warnings!

```
 -----------------------------------------------------------------------------
|                                                                             |
|           W    W    AA    RRRRR   N    N  II  N    N   GGGG   !!!           |
|           W    W   A  A   R    R  NN   N  II  NN   N  G    G  !!!           |
|           W    W  A    A  R    R  N N  N  II  N N  N  G       !!!           |
|           W WW W  AAAAAA  RRRRR   N  N N  II  N  N N  G  GGG   !            |
|           WW  WW  A    A  R   R   N   NN  II  N   NN  G    G                |
|           W    W  A    A  R    R  N    N  II  N    N   GGGG   !!!           |
|                                                                             |
|      ALGO=A and IALGO=5X tend to fail with the tetrahedron method           |
|      (e.g. Bloechls method ISMEAR=-5 is not variational)                    |
|      please switch to IMSEAR=0-n, except for DOS calculations               |
|      For DOS calculations use IALGO=53 after preconverging with ISMEAR>=0   |
|          I HOPE YOU KNOW, WHAT YOU ARE  DOING                               |
|                                                                             |
 -----------------------------------------------------------------------------
```

Common support cases

- complicated INCAR…
- structure (POSCAR)
- k-mesh (KPOINTS)
- NCORE/NPAR, KPAR
- VASP version
- cores
- memory

Common support cases

- complicated INCAR… simplify & try again! ALGO=N
- structure (POSCAR) reasonable/correct?
- k-mesh (KPOINTS) Γ-centered?
- **NCORE**/~~NPAR~~, KPAR simplify (possibly remove)!
- VASP version try latest (possibly "vanilla" version)!

```
$ module add VASP/5.4.4.16052018-nsc1-intel-2018a-eb
```

- cores too few/many?
- memory
  - ENCUT
  - k-mesh

See previous discussion

larger memory nodes:

```
#SBATCH -C fat
```

reduce cores/node:

```
#SBATCH --ntasks-per-node=16
INCAR: NCORE=16
```

Notes / Reminders

- Same NBANDS when comparing E_{tot}?
- Large enough NBANDS? e.g. increase for higher temp.
- Sufficient accuracy for your case?
- Use vasp_gam for 1 k-point calcs.
- LWAVE=.FALSE. WAVECAR might grow very large, don't output if not needed

Summary "rules of thumb"

- job size (max): **total cores ≈ NBANDS / 8**
- **NSIM = 4** (default), or more (2 Beskow)
- **NCORE = cores/node**
- PREC = Accurate - if forces important
- ENCUT = ENMAX x1.5 - "max setting"
- KPAR = min (nodes, k-points) **HSE06, especially useful**
- In general, INCAR default settings OK
- GPU: important to increase NSIM

Resources

- Wiki / Manual
  Check in detail!
- Wiki
  examples, presentations
- Forum

Find the links:
https://vasp.at/

- Peter Larsson's (old) blog at NSC:
  https://www.nsc.liu.se/~pla/

info & discussion

Questions / trouble? support@nsc.liu.se, …

VASP Refs.

- Good presentations by Marsman and Blöchl
- Blöchl PRB **50**, 17953 (1994)
- Blöchl *et al.* https://arxiv.org/abs/cond-mat/0201015v2
- Kresse & Joubert PRB **59**, 1758 (1999)
- Holzwarth *et al.* PRB **55**, 2005 (1997)
- Martin, *Electronic Structure*, Chapter 11.1, 13.2
- https://vasp.at