THE UNIVERSITY of York

Parallel CASTEP

Matt Probert with thanks to Phil Hasnip
Condensed Matter Dynamics Group
Department of Physics,
University of York, U.K.
http://www-users.york.ac.uk/~mijp1

The story so far…

- CASTEP solves the Kohn-Sham equations for a periodic system (potential), i.e.

$$\hat{H}[\rho]\psi_b = E_b\psi_b$$

where particle *b* has the *b*th solution (band) at Brillouin zone sampling point **k**, and

$$\hat{H}[\rho] = -\frac{\hbar^2}{2m}\nabla^2 + \hat{V}_{HXC}[\rho] + \hat{V}_{ext}.$$

Bloch’s Theorem

- Recall that Bloch’s theorem lets us write:

$$\psi_k(\mathbf{r}) \quad = \quad e^{i\mathbf{k}.\mathbf{r}} u_k(\mathbf{r})$$

- Where $u_k(\mathbf{r}+\mathbf{L}) = u_k(\mathbf{r})$ is periodic and $e^{i\mathbf{k}.\mathbf{r}}$ is an arbitrary phase factor. We express $u_{bk}(\mathbf{r})$ as a Fourier series:

$$u_{bk}(\mathbf{r}) \quad = \quad \sum_{G} c_{Gbk} e^{i\mathbf{G}.\mathbf{r}}$$

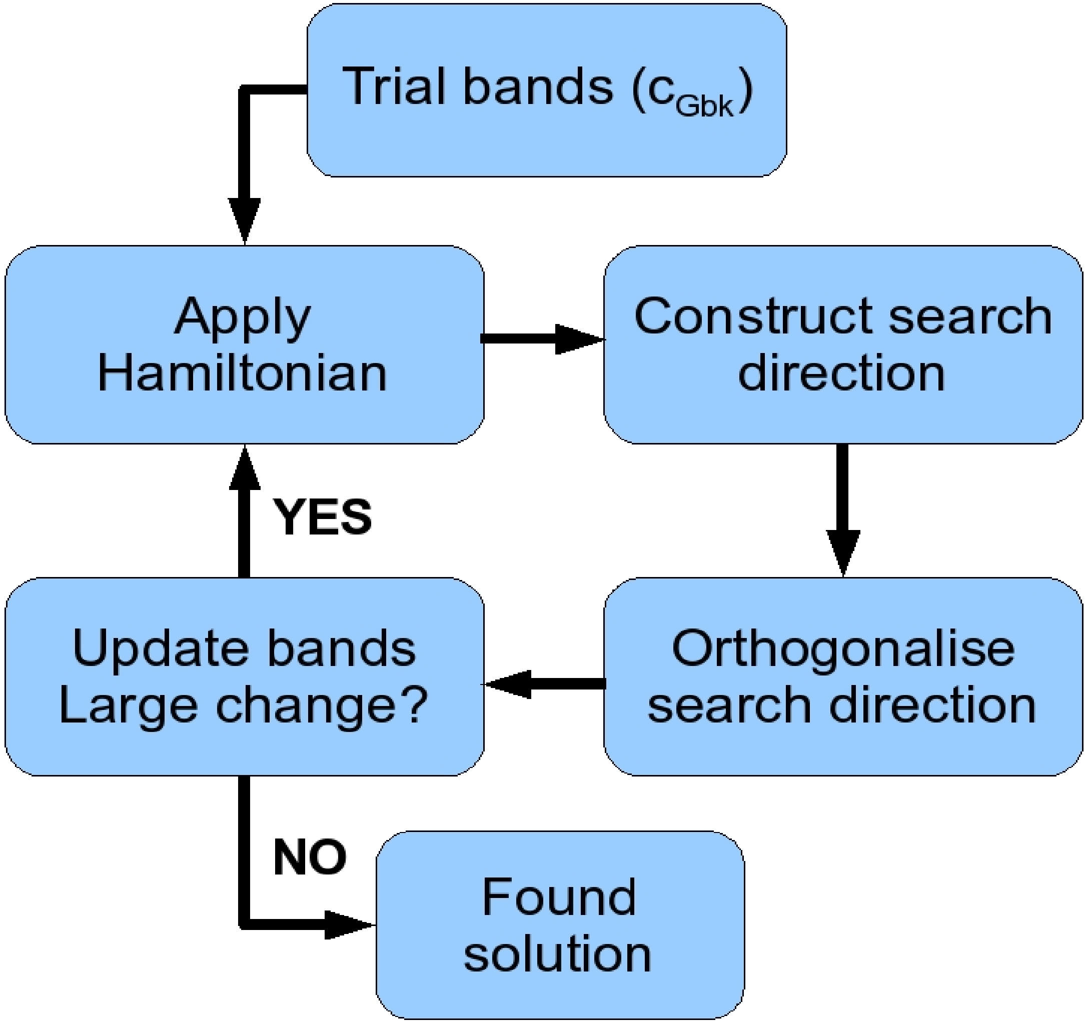
- Where $c_{\mathbf{G}b\mathbf{k}}$ are complex Fourier coefficients.

The wavefunction

$$\psi_{b\mathbf{k}} = \sum_{\mathbf{G}} c_{Gb\mathbf{k}} e^{i(\mathbf{G}+\mathbf{k}).\mathbf{r}}$$

- The complex coefficients $c_{\mathbf{G}b\mathbf{k}}$ are what CASTEP computes, and take up a lot of the computer’s memory (RAM).
- **G**: a reciprocal lattice vector (“G-vector”)
- b: a band index
- ***k***: a Brillouin zone sampling point (“***k***-point”)

Where does CASTEP spend its time?


Trial bands (c_{Gbk})
Apply Hamiltonian
Construct search direction
Orthogonalise search direction
Update bands Large change?
YES
NO
Found solution

Where does CASTEP spend its time?

- Applying *H*
  - Kinetic energy applied in reciprocal-space
  - Local potential applied in real-space so need to (fast) Fourier transform between the two spaces.
- Orthogonalising wavefunctions
  - Need to make trial bands orthogonal to each other
  - Compute the band-overlap matrix, and transform to an orthonormal set.

CASTEP and FFTs

- To apply *H* we need to 3D FFT from real to reciprocal space & vice versa.

$$\psi_{bk}(\mathbf{G}) \longleftrightarrow \psi_{bk}(\mathbf{r})$$

- Time to transform 1 band at 1 **k**-point with N_G G-vectors (plane-waves) is ~$\mathcal{O}(N_G \ln N_G)$
- Therefore to transform each of the N_b bands at each of the N_k ***k***-points takes a total FFT time ~$\mathcal{O}(N_G N_b N_k \ln N_G)$

Orthogonalisation

- We construct the *band overlap* matrix at each ***k***-point: $S_{nmk} = \langle \psi_{nk} \,|\, \psi_{mk} \rangle$
  - Time to construct ~$\mathcal{O}(N_G N_b^2 N_k)$
- Then we decompose this *S* matrix at each ***k*** to construct orthogonalising transformation
  - Time to decompose ~$\mathcal{O}(N_b^3 N_k)$
- Then apply transformation to get orthogonal bands
  - Time to apply ~$\mathcal{O}(N_G N_b^2 N_k)$

Scaling of quantities with system size

- For small systems:
  - N_G small
  - N_B small
  - N_k big

  Time usually dominated by the Fourier transform.

  Both the Fourier transform and orthonormalisation scale as ~ N_k so parallelise over $\boldsymbol{k}$

K-point parallelism

k-point parallelism

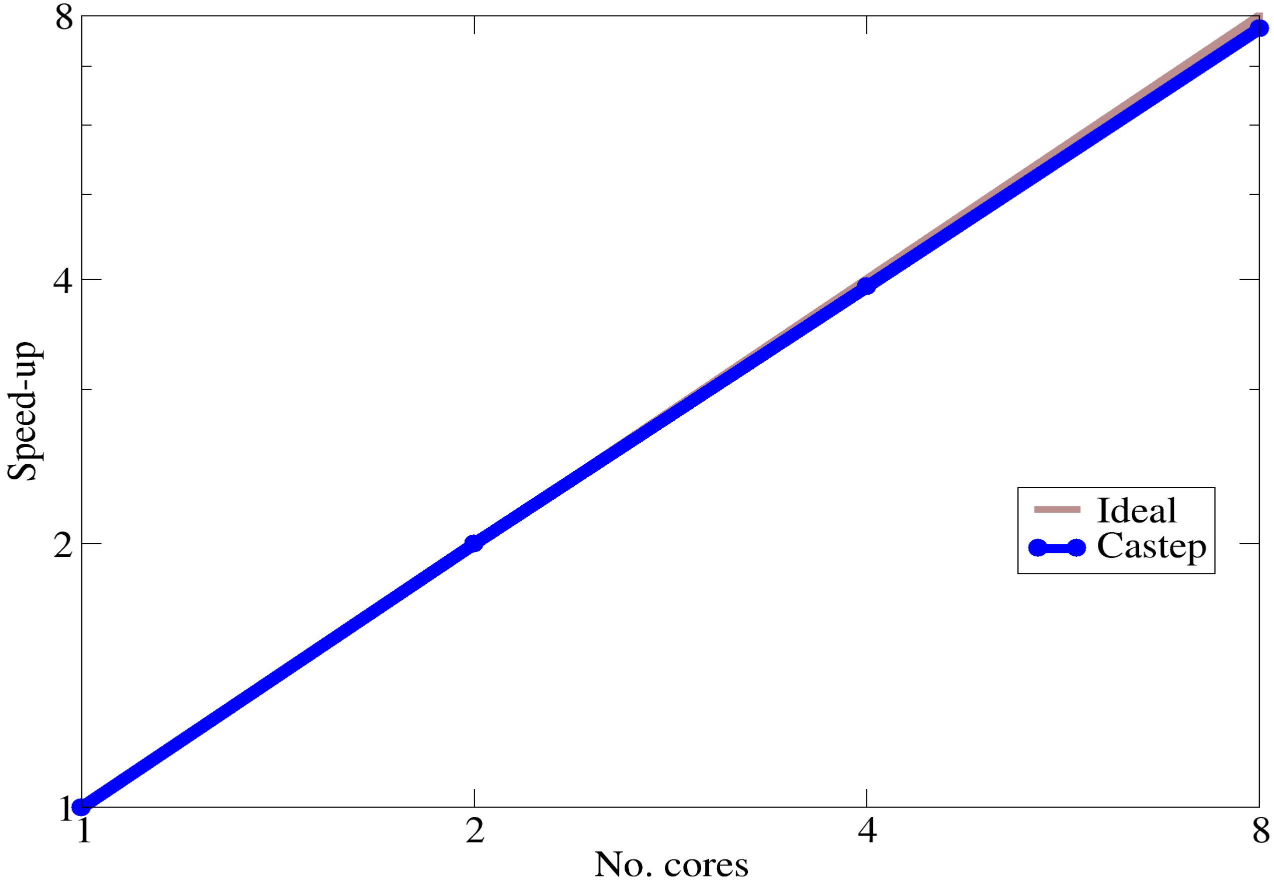
- The bands at each ***k***-point are almost independent of each other:

$$\hat{H}_k[\rho]\psi_{bk} = E_{bk}\psi_{nk}$$

- Can give each core a subset of ***k***-points and solve a subset of the equations
- Why “almost” independent? They are coupled via the density $\rho\left(\mathbf{r}\right) = \sum_{b\mathbf{k}} f_{b\mathbf{k}} \left|\psi_{b\mathbf{k}}\left(\mathbf{r}\right)\right|^2$
- where $f_{b\mathbf{k}}$ is the band occupancy.

Example – small benchmark

- TiN is a standard small benchmark:
  - 33 atoms
  - 8 ***k***-points
  - 164 bands
  - 10962 Gv



k-parallel and big systems

- ***k***-parallelism is almost perfect
  - Puts very little demand on communication infrastructure so scales well over ethernet
  - Use **`--dryrun`** flag to see how many ***k***-points
- BUT as go to bigger system sizes, have bigger unit cell -> smaller BZ -> need fewer ***k***-points -> less scope for parallelism!
  - The bigger the system the fewer cores we can use!
  - In limit of very big systems $N_k = 1$

Scaling of quantities with system size

- For big systems:
  - N_G big
  - N_b big
  - N_k small

  Time dominated by orthogonalisation

  $\sim N_G N_b^2 N_k$

  Need to parallelise over something else…

G-vector parallelism

G-vector parallelism

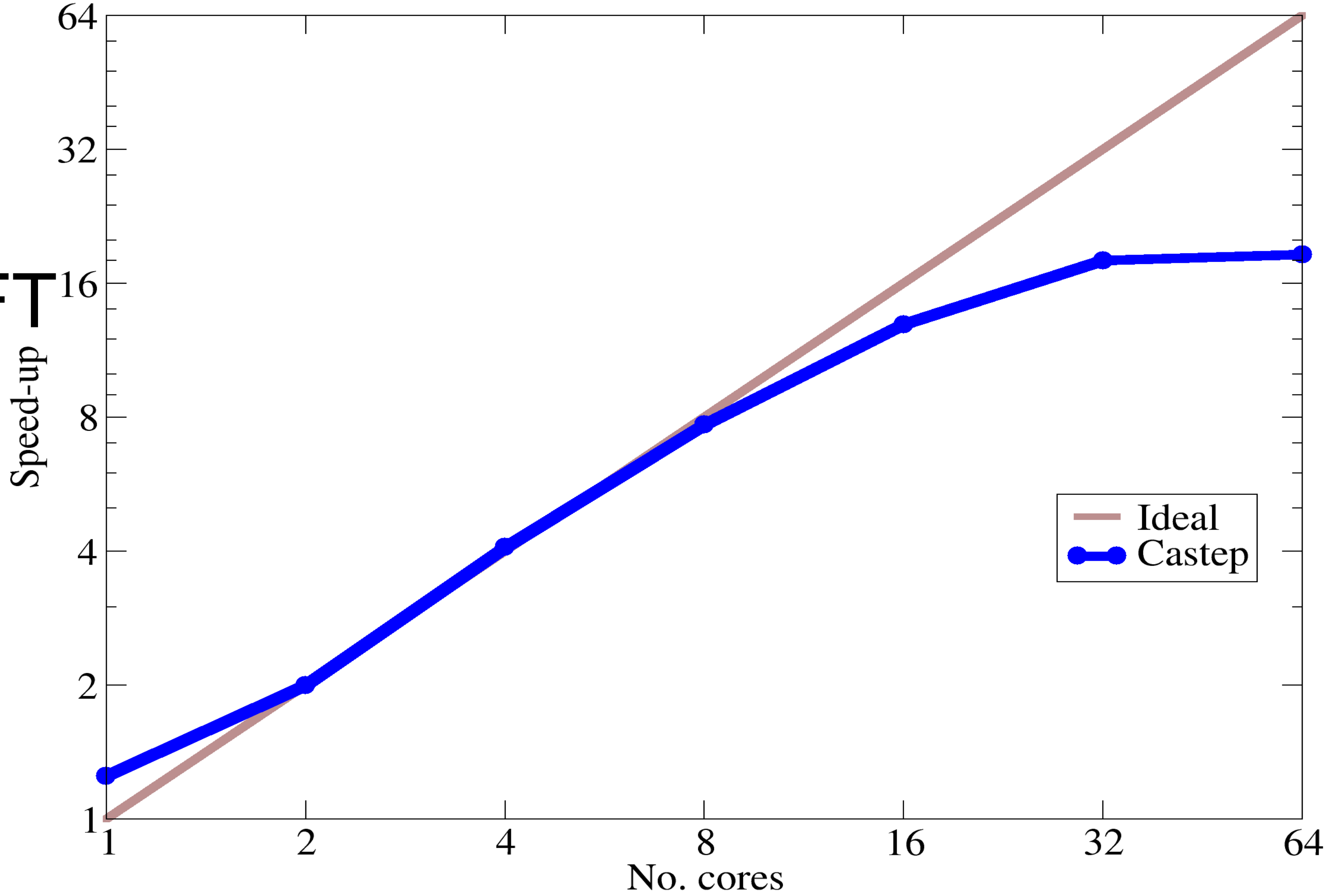
- Large systems dominated by cost of band orthogonalisation with S matrix:

$$\begin{aligned} S_{nmk} &= \langle \psi_{nk} \,|\, \psi_{mk} \rangle \\ &= \sum_{G} c^{\star}_{Gnk} c_{Gmk} \end{aligned}$$

- Distribute ***G***-vectors over cores
- Contributions to S summed over cores
- N_G increases with system size

G-vector parallelism in action

- TiN again
- 1 core faster due to non-parallel FFT
- Effect of comms



Analysis of ***G***-vector parallelism

- ***G***-vector parallelism requires much more finely-grained communications than ***k***-point
  - More sensitive to interconnect
  - Need low-latency network (ethernet is bad!)
- But working on different part of data structures to ***k***-point parallelism so can combine them …

k*+*G parallelism

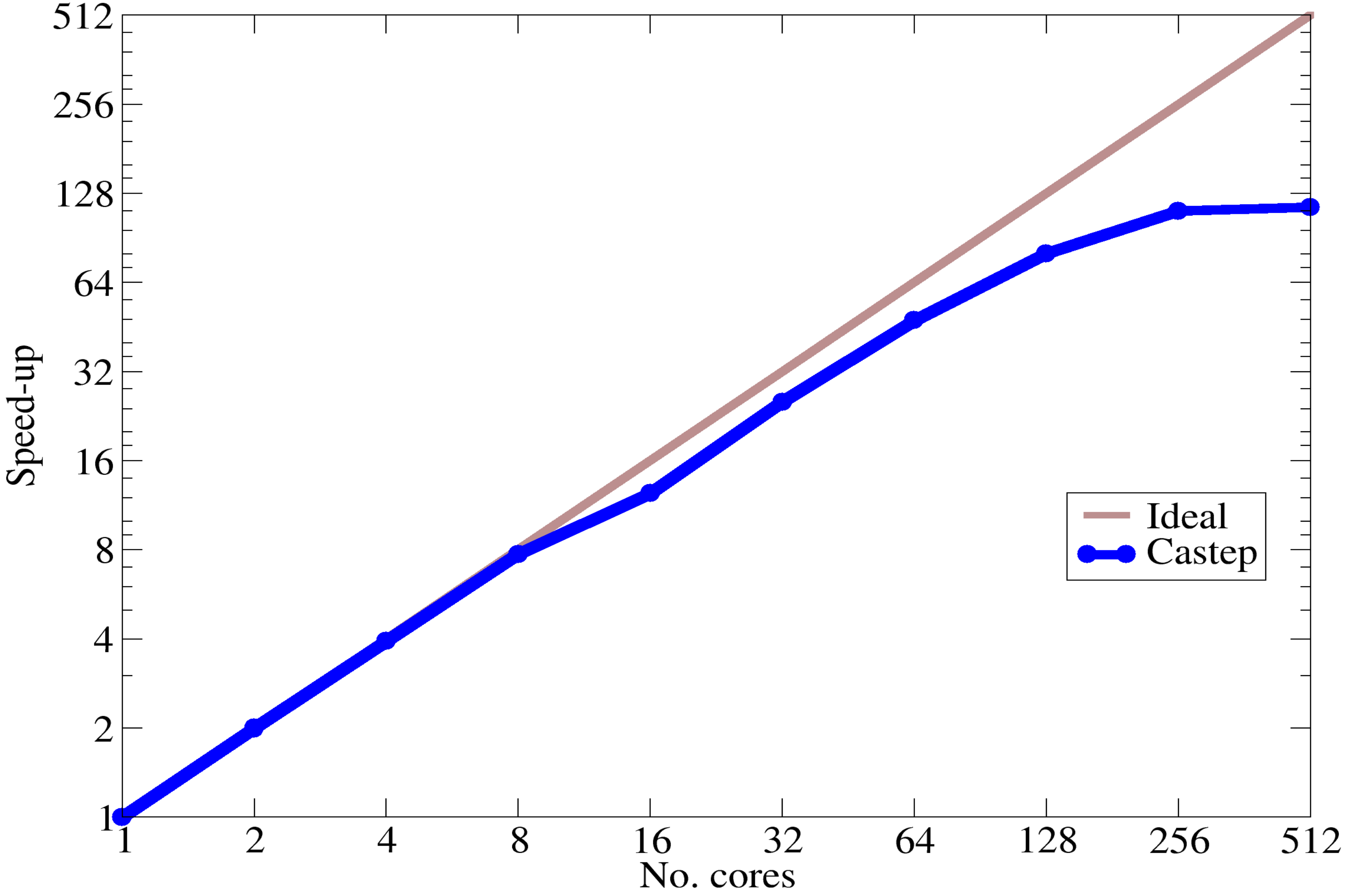
- Independent parallelisation schemes
- E.g. if N_k=2, N_G=9000 and N_{core}=6:

Data	***k*-point 1**	***k*-point 2**
G-vecs 1-3000	Core 1	Core 4
G-vecs 3001-6000	Core 2	Core 5
G-vecs 6001-9000	Core 3	Core 6

- For any ***k***-point the ***G***-vector data is split across 3 cores, i.e. 3-way ***G***-vector parallel
- For any subset of ***G***-vectors the data is split across 2 cores, i.e. 2-way ***k***-point parallel

k+G parallelism in action

- TiN again
- Scaling limited by comms at high core counts



CASTEP optimal performance

- Always use ***k***-point parallelism if it is there
  - Hence run on $N_{core} = N_k$
  - Or if that is not practical/feasible choose a high common factor for ***k***-point and then use ***G***-vector
  - E.g. $\mathrm{N_k}$=35

Cores	Parallel distribution
70	Each pair of cores gets one **k**-point **G**-vectors are distributed within each pair
36	One core left idle; CASTEP uses 35 cores
35	Each core gets one **k**-point
21	Cores split into 7 triplets Each triplet of cores gets 5 **k**-points **G**-vectors are distributed within each triplet
5	Each core gets 7 **k**-points

Parallel Fourier Transforms

- Why does **G**-vector parallelism have poorer performance?
  - In **G**-vector parallel, do 3D FFT as three 1D FFTs
  - Each core has all **G**-vectors in a z-column
  - Do 1D FFT along z
  - All cores swap data so each has y-column data
  - Do 1D FFT along y
  - Now swap to get x-column data and do final FFT
  - Each core has real-space data along x.

Parallel Fourier Transforms

- The actual 1D FFTs are distributed well
- When the cores swap data, all cores communicate with all other cores
- For *P* cores this “data transposition” requires P^2

  Communications of 1/ P^2 data each
- As *P* increases we end up with huge numbers of tiny messages – strongly latency-bound!
- On a cluster with multicore nodes, cores often share interconnect with others on same node: contention
- Time scales as P^2 and Fourier transform dominates

  computational time for large core counts.

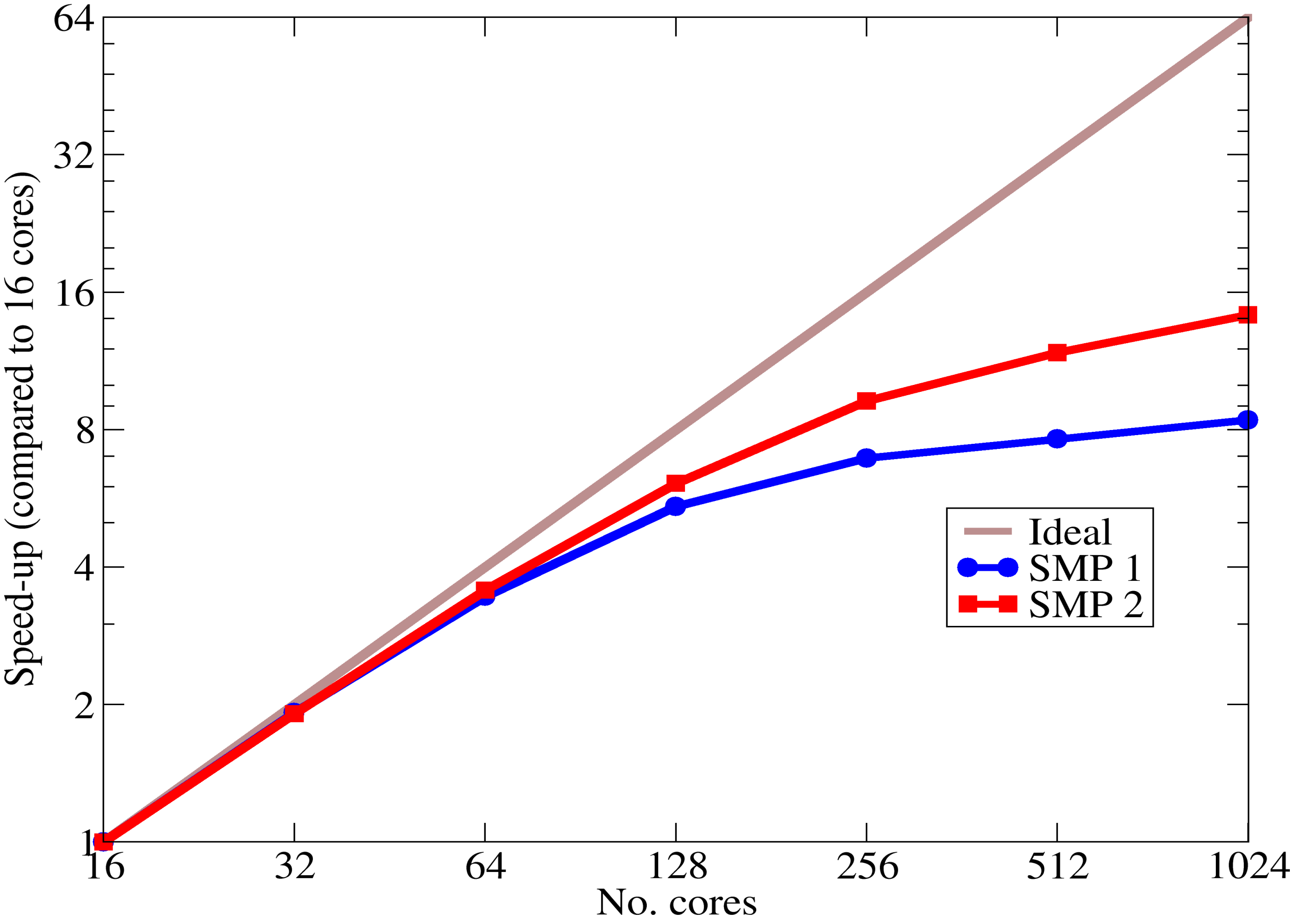
Parallel Fourier Transforms

- Contention can be reduced by aggregating messages
- Cores on same node designate a “master” core
- Cores give data to master
- All masters communicate
- Masters pass data back to cores on their node
- Leads to fewer, longer messages between nodes, so less latency-bound
- Reduces contention
- Activate via `.param` file e.g.:

```
num_proc_in_smp : 2
```

THE UNIVERSITY *of York*

Parallel FFTs with SMP optimisation



Band-parallelism

- Is there anything else we can parallelise over?

$$\psi_{b\mathbf{k}} = \sum_{\mathbf{G}} c_{Gb\mathbf{k}} e^{i(\mathbf{G}+\mathbf{k}).\mathbf{r}}$$

- Done ***G*** and ***k*** so what about *b*?
  - N_b grows with system size
  - Same *H* for different bands at same ***k***
  - Fourier transforms of different bands independent –> perfect scaling here?

band parallelism

Band-parallel orthogonalisation

- Need to construct *S* matrix at each ***k***-point

$$S_{nm} = \langle \psi_n | \psi_m \rangle$$

- Inner product is between all pairs of bands
  - Need all-to-all communication
  - Need high-bandwidth interconnect
  - Will limit scaling at high core counts
  - Distribute rows of *S* matrix over cores

Band-parallel functionality

- Band-parallelism is the newest form of parallelism and is not turned on automatically
- Not all functionality supports band parallism – depends on CASTEP version
- Can control exact parallel distribution via a devel_code setting in `.param` e.g.:

```
%block devel_code
PARALLEL: kpoint=2 gvector=2 band=2 :END_PARALLEL
%endblock devel_code
```

Mixing k, G and b parallelism

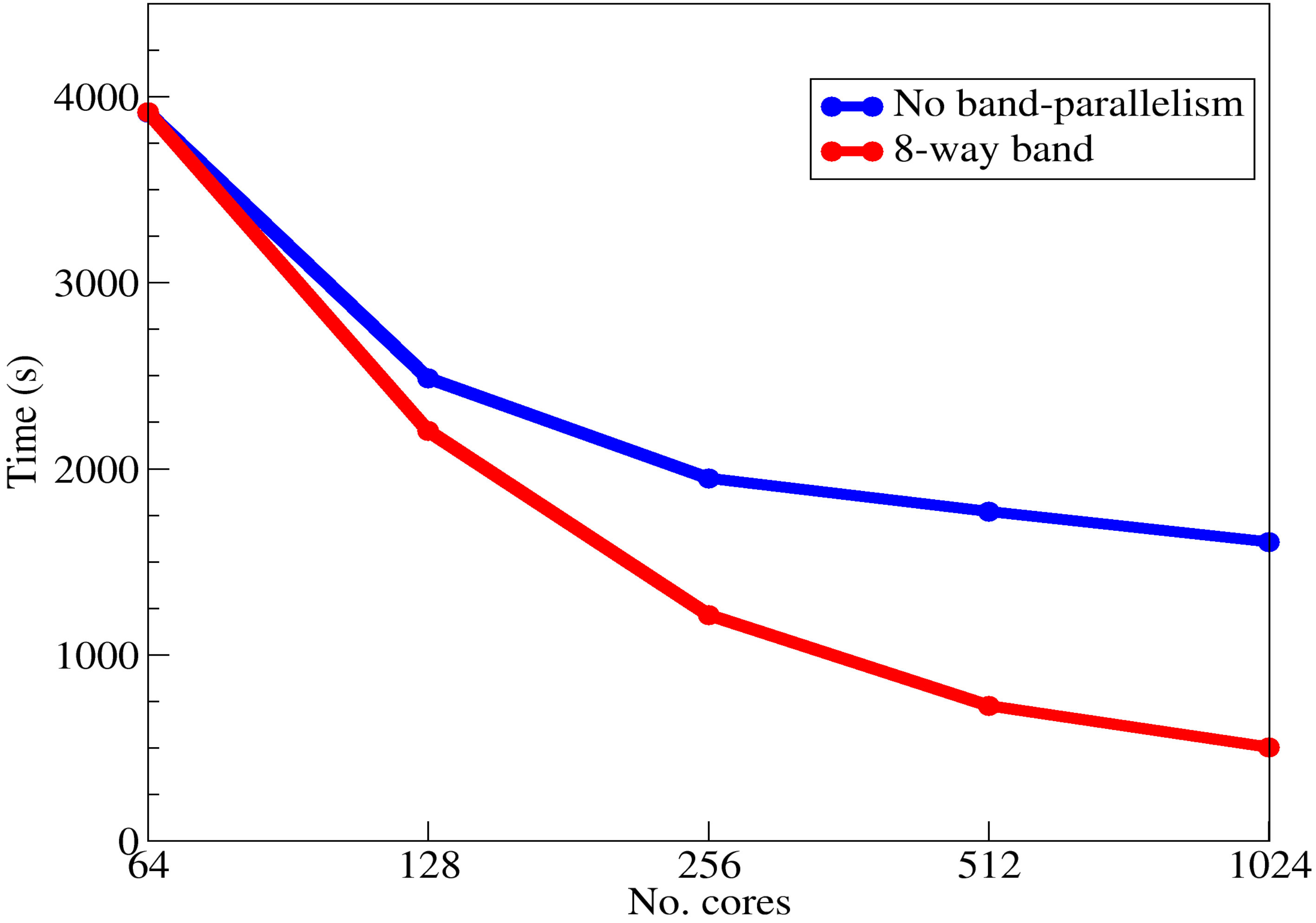
- ***k***-point, ***G***-vector and band-parallelism are all independent -> can combine all 3
  - ***k***-point scales perfectly, OK on poor interconnect
  - ***G***-vector dominated by comms in FFT: needs low-latency interconnect
  - Band-parallel dominated by comms in orthogonalisation: needs high-bandwidth interconnect

Putting it all together …

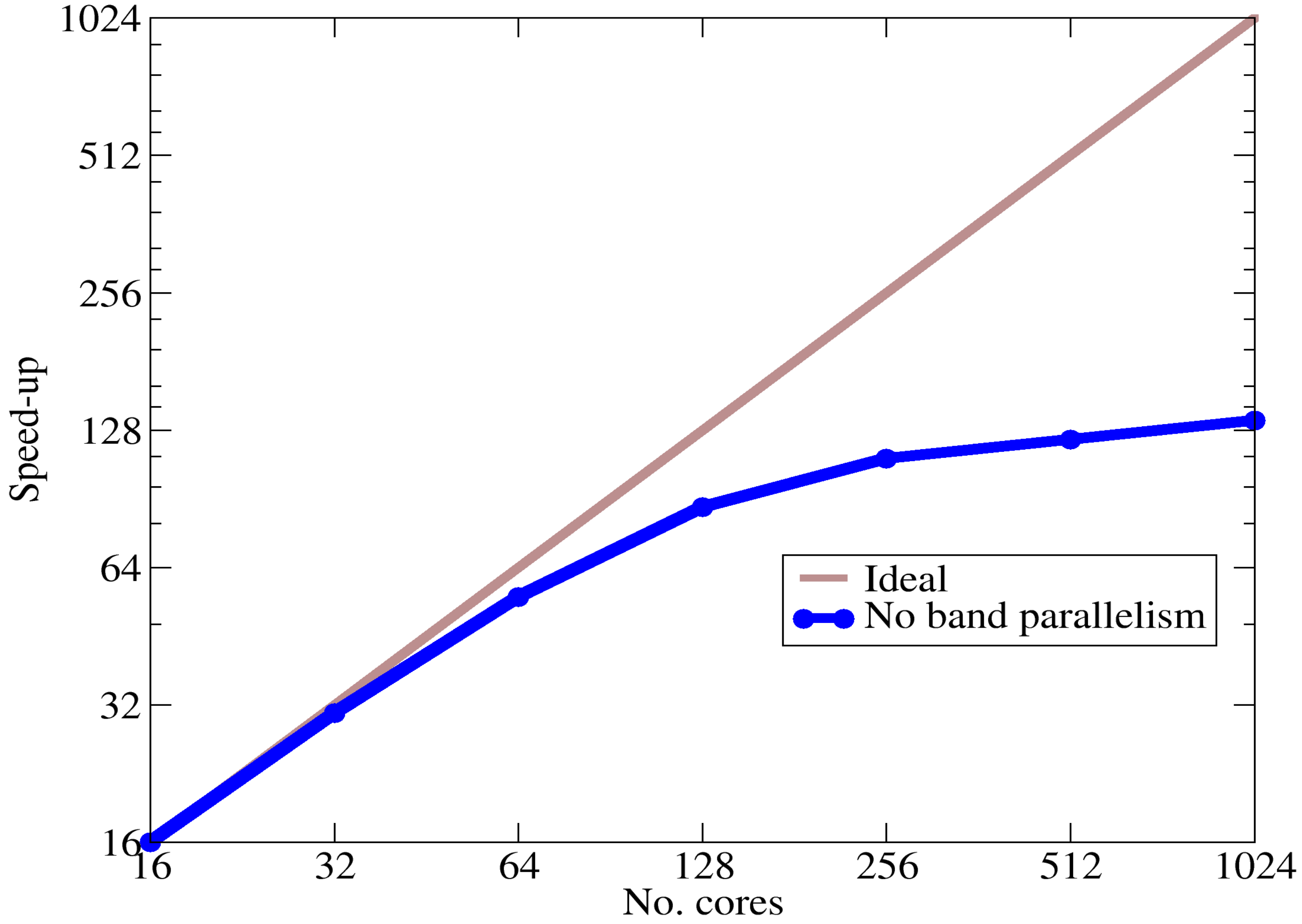
A bigger benchmark

- Al_2O_3-3x3 surface slab:
  - 270 atoms
  - 2 ***k***-points
  - 778 bands
  - 88184 ***G***-vectors

Al_2O_3 time

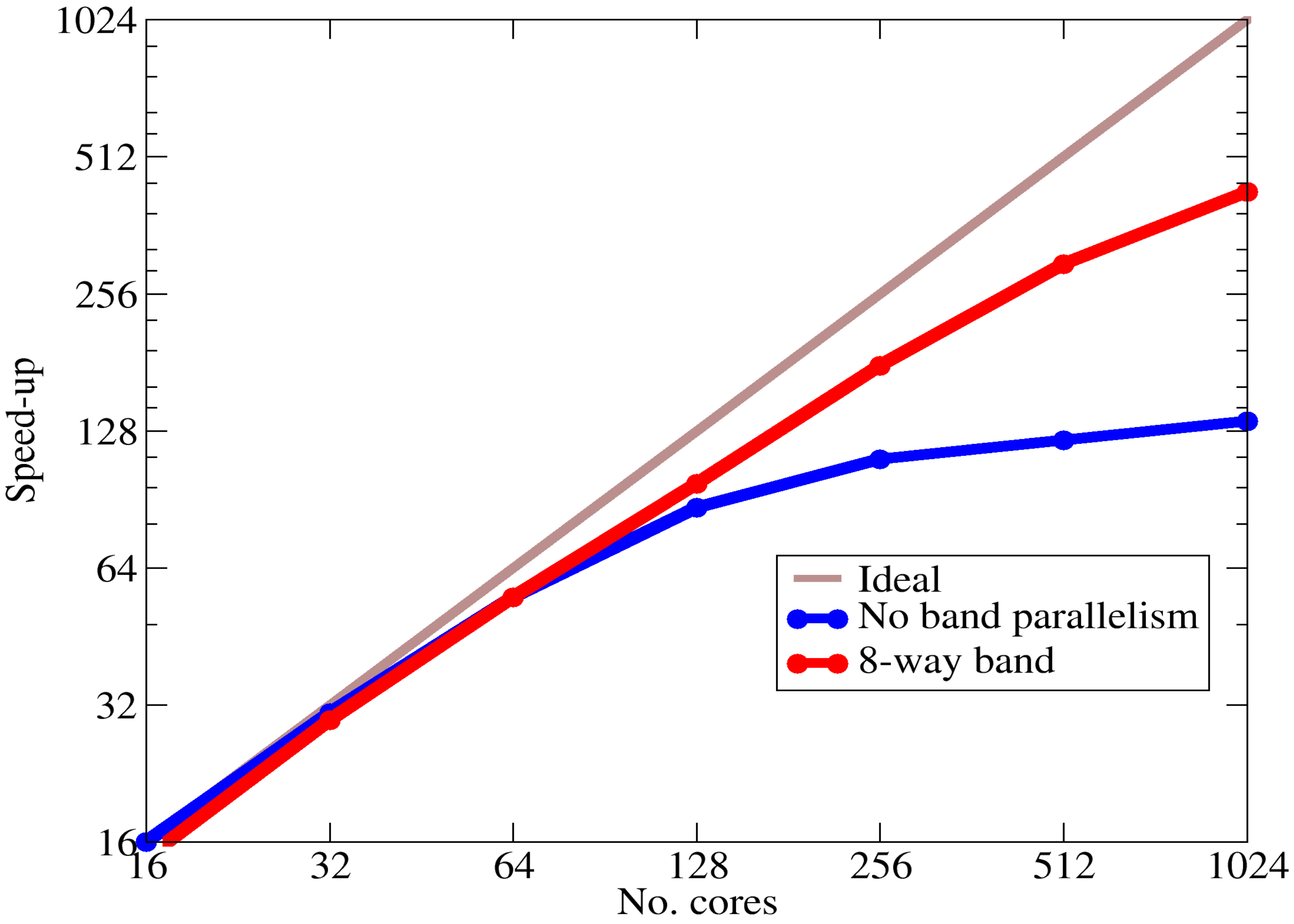

No band-parallelism
8-way band
Time (s)
4000
3000
2000
1000
0
64
128
256
512
1024
No. cores

Al_2O_3 parallel speedup



Use 16 core reference as too big to run on anything smaller!

Al_2O_3 parallel speedup

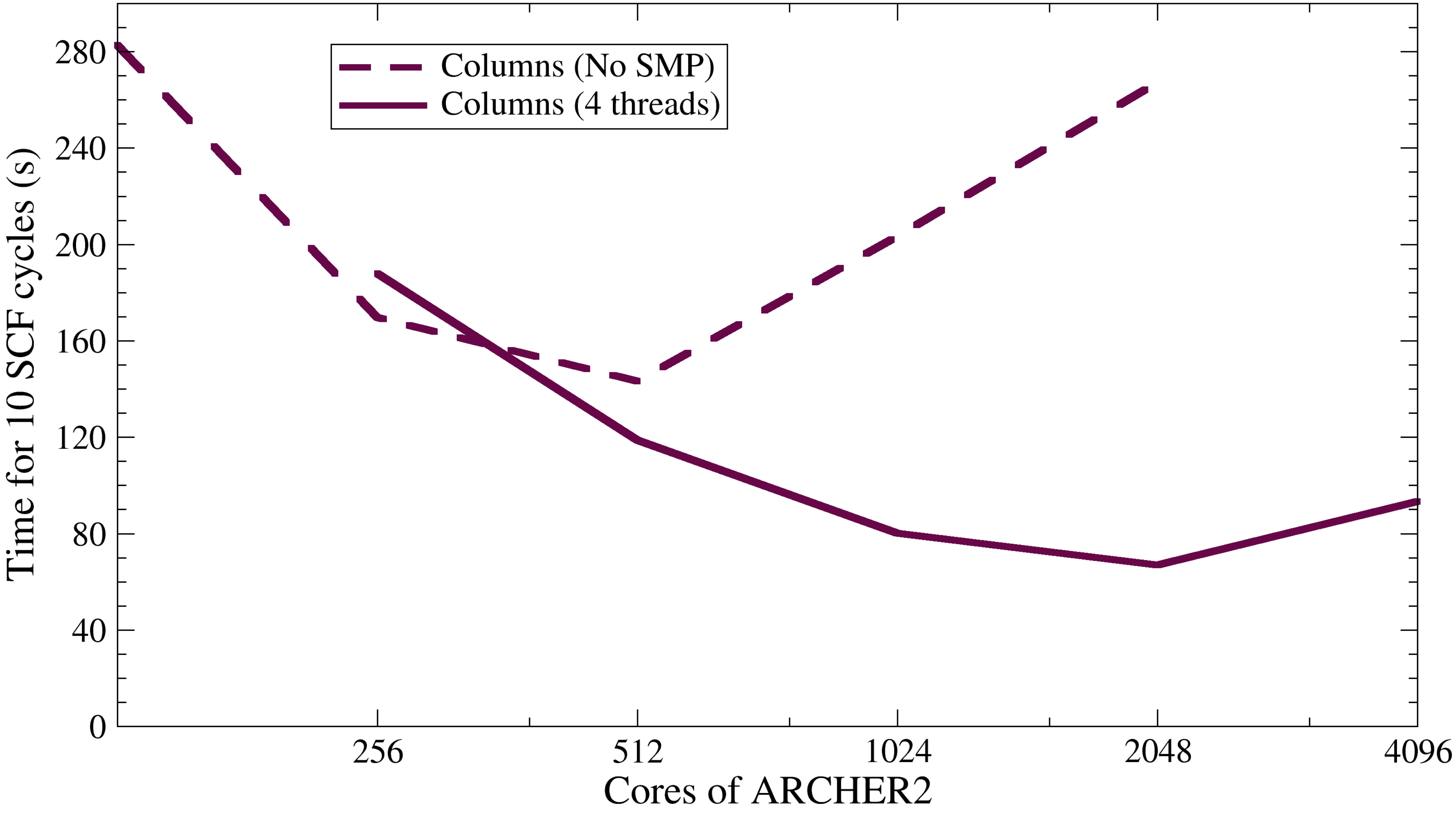


Use 16 core reference as too big to run on anything smaller!

Threading

- So far we’ve mostly considered distributed-memory parallelism
- Can also use shared-memory (OpenMP) parallelism
- Initial implementation released in CASTEP 16.1
- Activate by setting an *environment variable* `CASTEP_NUM_THREADS` to a value greater than 1
- Reduces memory usage per node
- Modest speed-ups for some systems
- Can be combined with usual MPI parallelism
- Expect further optimisations in the future

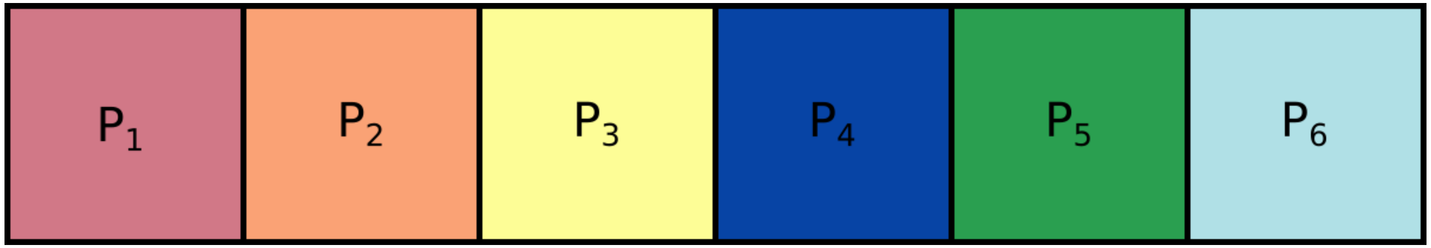
Al_2O_3 with threading



Latest developments …

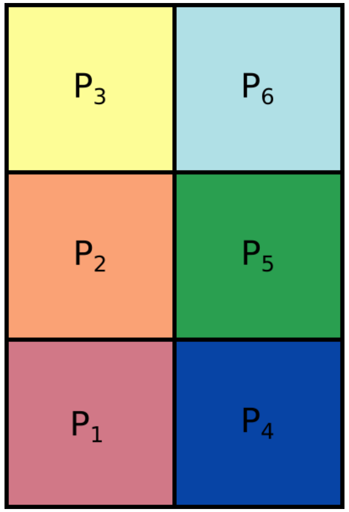
Process grid

- FFT limits scaling for large calculations as it requires all-to-all comms
- With *P* processors comms time scales ~ P^2



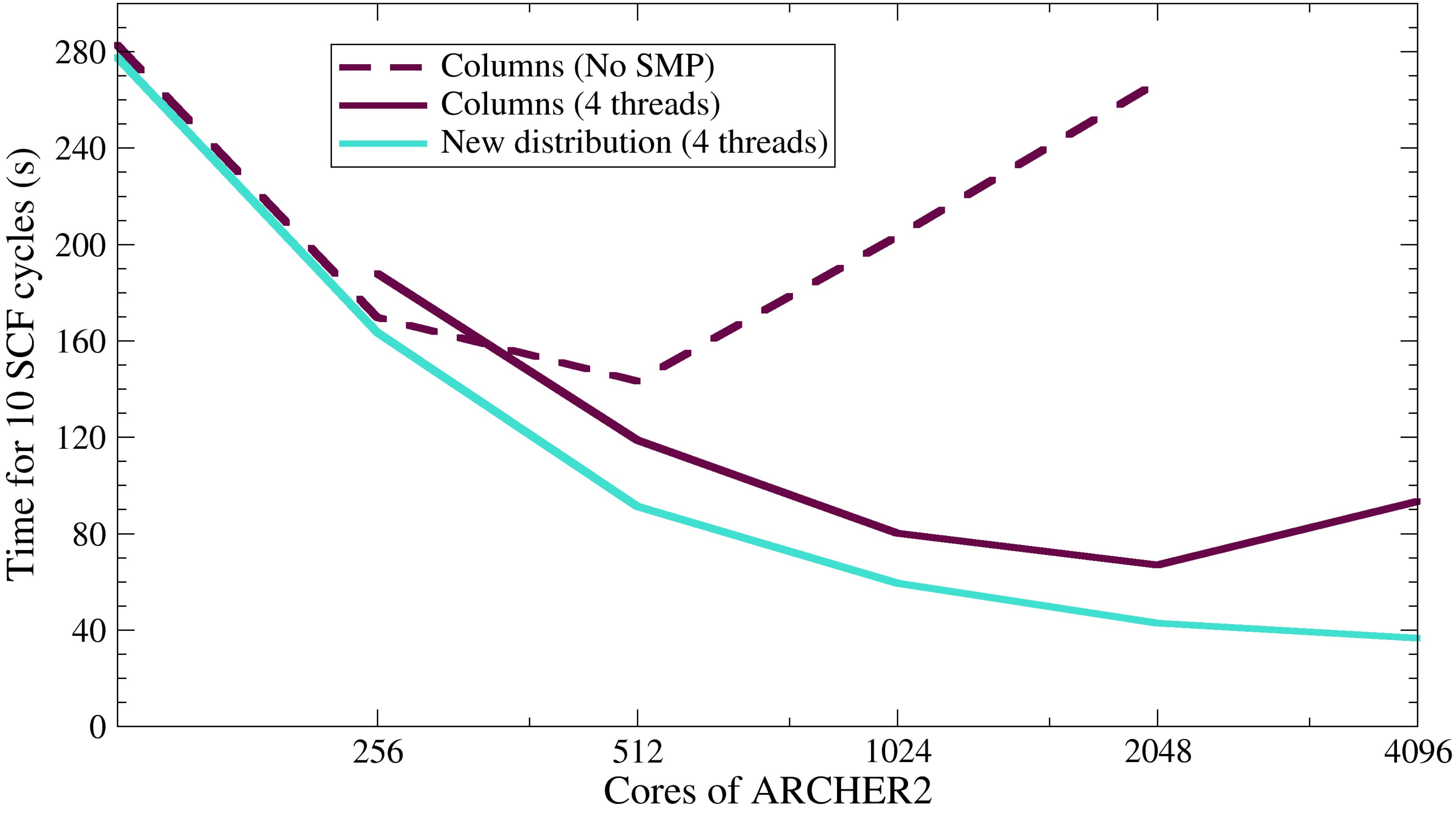
$O(P^2)$

- V24 put into a process grid
- Now only need comms in a row or a column of the grid
- Less comms & better scaling!



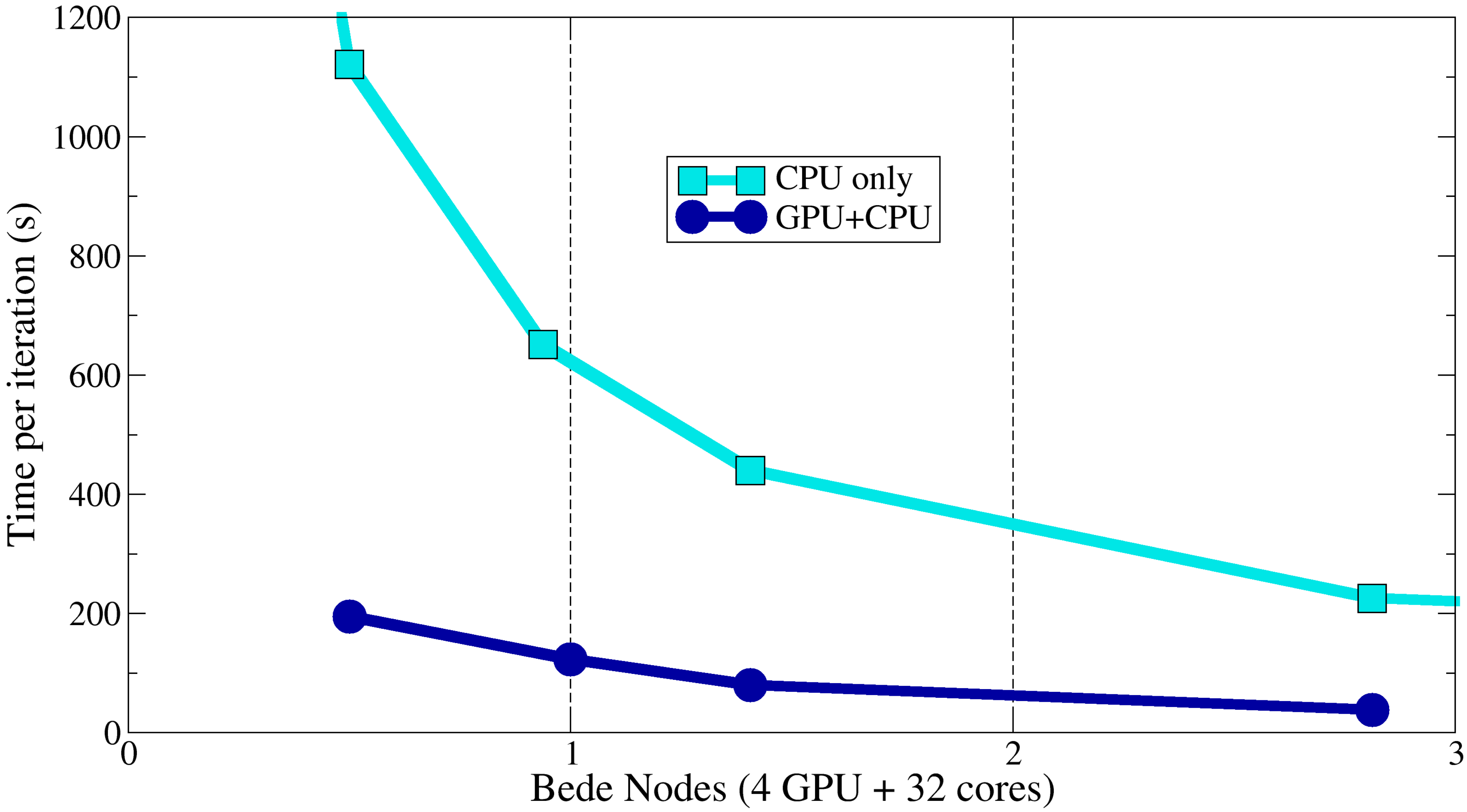
$O(P)$

Al_2O_3 with process grid



THE UNIVERSITY *of York*

GPU port in progress …



Summary

Summary

- Plane-wave DFT in CASTEP has lots of parallelism potential
  - Can parallelise over ***k***-points, ***G***-vectors and bands
  - Choose which scheme depending on material system size / features
  - Also depends on interconnect in computer
  - BEWARE: you can *over-parallelise* a calculation – can go *slower* if put in too many cores as comms cost will dominate