THE UNIVERSITY of York

# Parallelization in CASTEP

Matt Probert
*August-Wilhelm Scheer Visiting Prof TUM 2015*
Condensed Matter Dynamics Group
Department of Physics,
University of York, U.K.
http://www-users.york.ac.uk/~mijp1

# Overview

- Bottlenecks
- How to parallelize a plane-wave DFT code
  - K-points, G-vectors and bands
- Parallel efficiency
- Summary

# Before you parallelize anything

- Do you know where is the code spending its time?
- Do you know what are the key data structures / algorithms?
- Have you got good serial performance?
- Are you limited by run time and/or available memory etc?
- Only if answer is ‘yes’ to all these questions is it worth going further …

# The Bottlenecks

# CASTEP bottlenecks

- We saw in last lecture that the key algorithms in CASTEP are:
  - Applying $H$ – and with smart use of real/ reciprocal space cost ~$O(N_G N_B)$
  - Cost of FFT ~$O(N_G N_B \ln N_G)$
  - Orthogonalization of bands ~$O(N_B^3)$
- Where
  - Number of plane waves = $N_G$
  - Number of bands = $N_B$
  - And $N_G >> N_B$ and in general $N_G \alpha N_B$

# DFT recap

- The above simplification misses some important details
- Remember what we are trying to do – solve the K-S equations in periodic system, i.e.

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

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

# Bloch's Theorem

- The potential must be periodic:

$$V(\mathbf{r} + \mathbf{L}) = V(\mathbf{r})$$

- So the wavefunction is 'quasi-periodic':

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

  - 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.

- $\boldsymbol{k}$ is also a wave-vector and represents a point in the Brillouin Zone

# 3D Plane Waves

- Since $u_{bk}(\mathbf{r})$ is periodic we can express it as a Fourier Series too:

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

  - where $c_{Gbk}$ are complex coefficients
- Hence we have

$$\psi_{bk}(\mathbf{r}) = \sum_G c_{Gbk} e^{i(\mathbf{G}+\mathbf{k}).\mathbf{r}}$$

# Brillouin Zone

- What is the value of ***k*** ?
  - Need to cover all values within the Brillouin Zone – the reciprocal space dual of the unit cell
  - Need to integrate over all ***k*** to calculate density
  - But bands vary slowly so can replace integral by sampling:

$$\rho(\mathbf{r}) = \sum_b \int |\psi_{bk}(\mathbf{r})|^2 d^3\mathbf{k}$$

$$\approx \sum_{bk} |\psi_{bk}(\mathbf{r})|^2$$

# *k*-point sampling

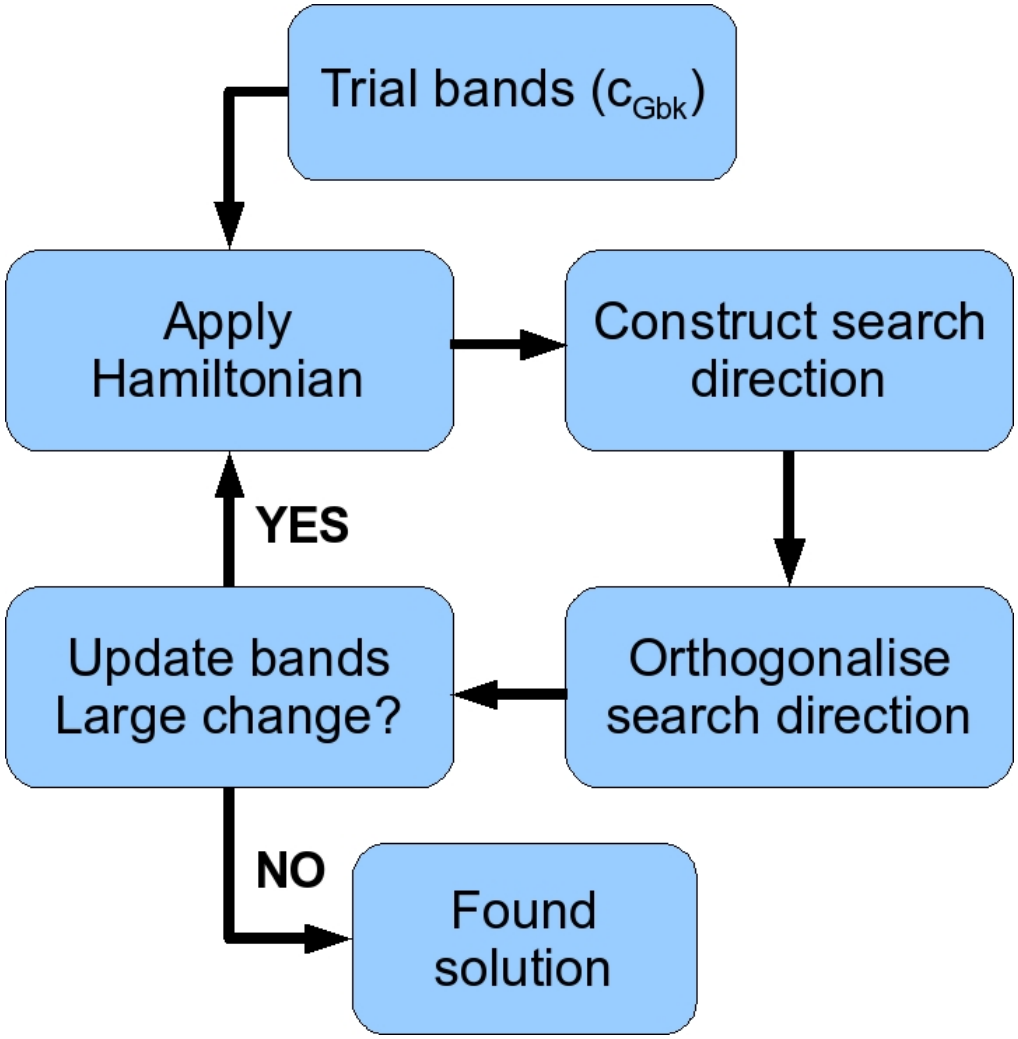
- So in CASTEP we need to make sure have got correct ***k***-point sampling
  - A user controlled convergence parameter
- And the bands at each ***k***-point are independent of each other:

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

- But it means that to do a solid requires more work – summing over ***k*** – that is not necessary in aperiodic system (e.g. molecule)

# Key Algorithms

# Solving the K-S Equations



# CASTEP and FFTs

- In a bit more detail …
- To apply *H* we need to 3D FFT from real to reciprocal space & v.v.
- Time to transform 1 band $\psi_{bk}(\mathbf{G}) \longleftrightarrow \psi_{bk}(\mathbf{r})$ is ~$O(N_G \ln N_G)$
- But we need to do this for every ***k***-point and band
- Hence FFT time ~$O(N_G N_B N_k \ln N_G)$

# Orthogonalization

- We construct the *band overlap* matrix at each ***k***-point: $S_{nmk} = \langle \psi_{nk} \mid \psi_{mk} \rangle$
  - Time to construct ~$O(N_G N_B^2 N_k)$
- Then we invert *S* matrix at each ***k*** to construct orthogonalizing transformation
  - Time to invert ~$O(N_B^3 N_k)$
- Then apply $S^{-1}$ to get orthogonal bands
  - Time to apply ~$O(N_G N_B^2 N_k)$

# Computational Bottlenecks

- For a small system, we have $N_G$ and $N_B$ small, $N_k$ big
- All bottlenecks ~ $N_k$ so parallelize over ***k***
- For big system, we have $N_k$ small, $N_G$ and $N_B$ big, so orthogonalization ~$N_G N_B^2$ wins
  - Key cost in large systems
  - Parallelize over $N_G$ and/or $N_B$
- Different parallel strategies depending on problem size …

# Key Data Structures

# Data Structures

- Key data structures are wavefunction and density – in both real and reciprocal space
  - Functions of plane waves, bands and k-points
- Need to *distribute* data across compute cores to reduce memory required per core
- And choose distribution to fit the key algorithms: matrix orthogonalization, multiplications and FFTs

# *k*-point parallelism

# *k*-point parallelism

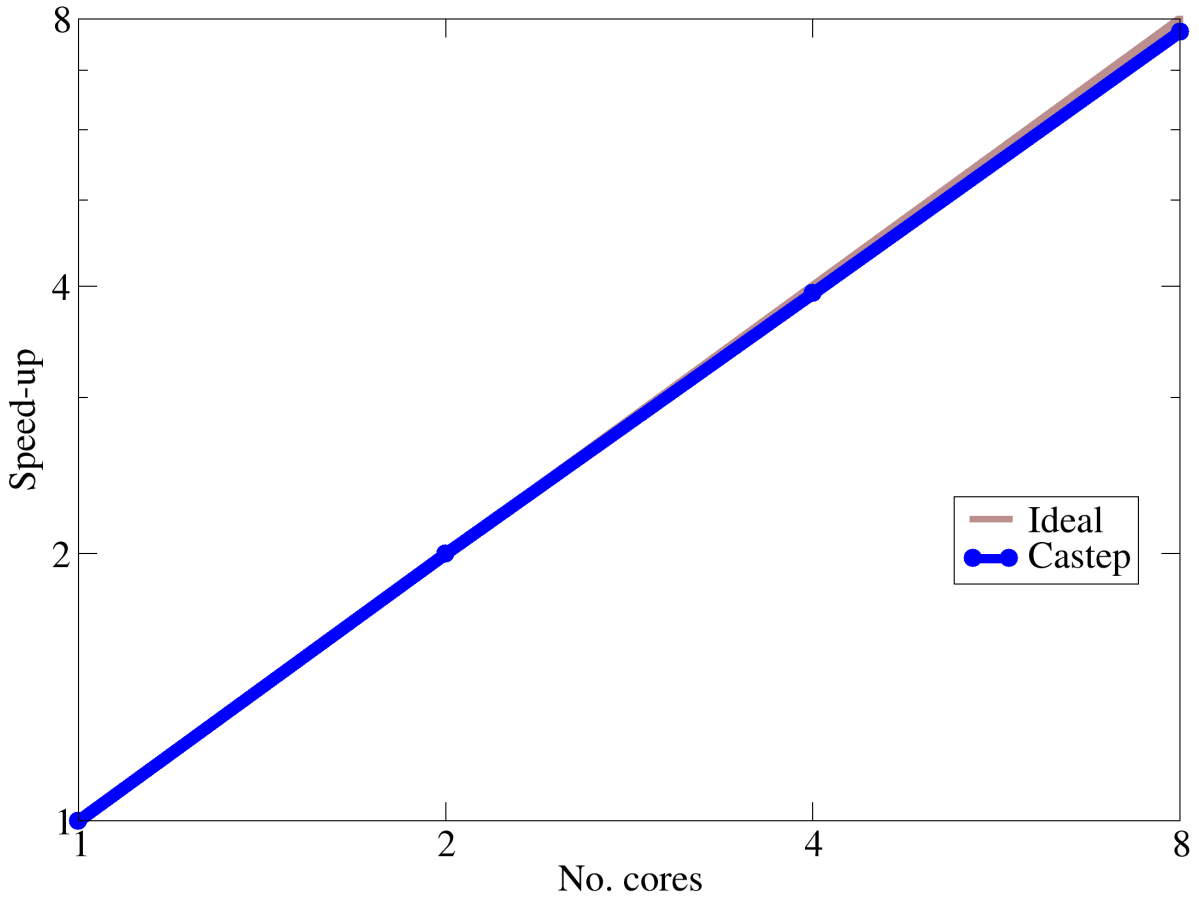
- Simplest approach is ***k***-point parallelism
- Bands at different ***k***-points are almost entirely independent
- Only need to communicate when constructing density as

$$\rho(\mathbf{r}) = \sum_{bk} |\psi_{bk}(\mathbf{r})|^2$$

- Hence give each core a subset of ***k***-points and solve a subset of K-S equations …

# 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
- BUT as go to bigger system sizes, have bigger unit cell -> smaller BZ -> need less ***k***-points -> less scope for parallelism!
  - The bigger the system the less cores we can use!
  - In limit of very big systems $N_k = 1$

# G-vector parallelism

# *G*-vector parallelism

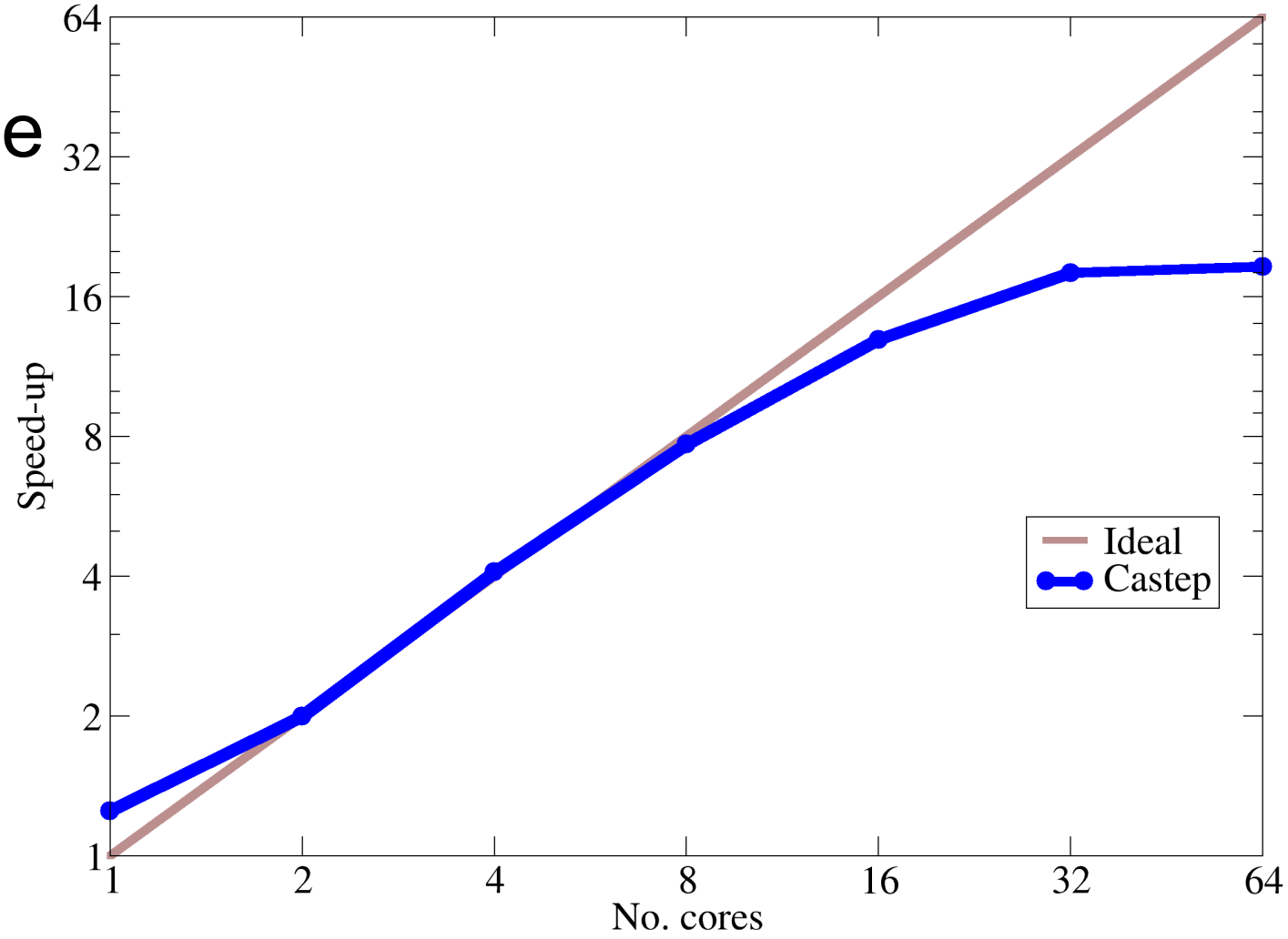
- Large systems dominated by cost of band orthogonalization with *S* matrix:

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

- 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 fine-grain communications than ***k***-point
  - Hence more sensitive to interconnect
  - Need low latency network (ethernet bad)
- But working on different part of data structures to ***k***-point parallelism so can combine them …

# *k*+*G* parallelism

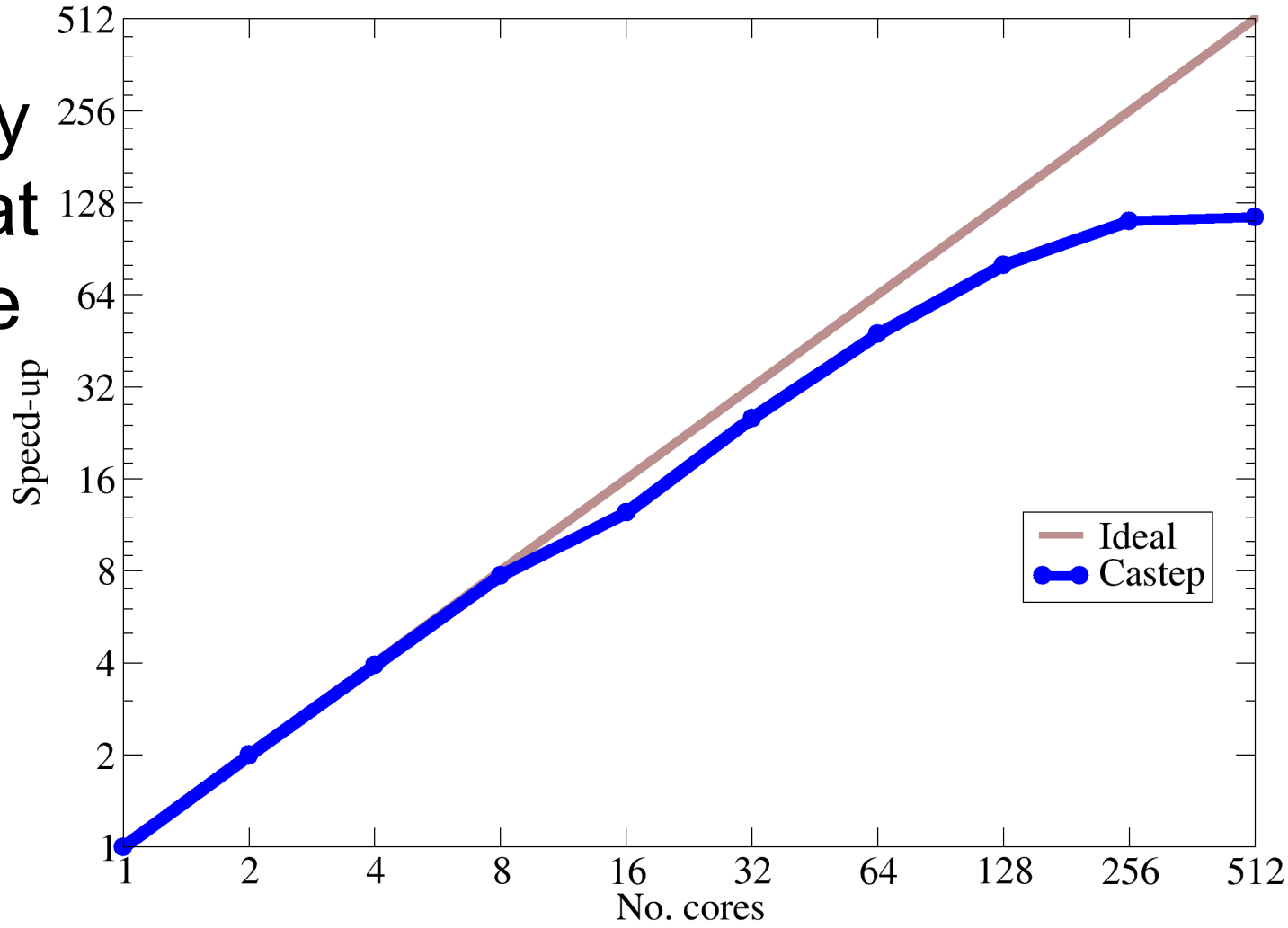
- Independent parallelisation schemes
- E.g. if $N_k$=2, $N_G$=9000 and $N_{core}$=6:

| 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 (e.g. if $N_k$=35 choose $N_{core}$ = 5 or 7 for good scaling)
- And then use ***G***-vector
  - E.g. with $N_k$=35 can run on $N_{core}$=70 (but 2-way ***G***-vector is not best) or $N_{core}$=105, 140 …
  - Can also work with $N_{core}$=20 and having multiple ***k***-points per core

# More parallelism

# More parallelism

- Is there anything else we can parallelise over?

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

- 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-parallel orthogonalization

- 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 *S* matrix rows over cores

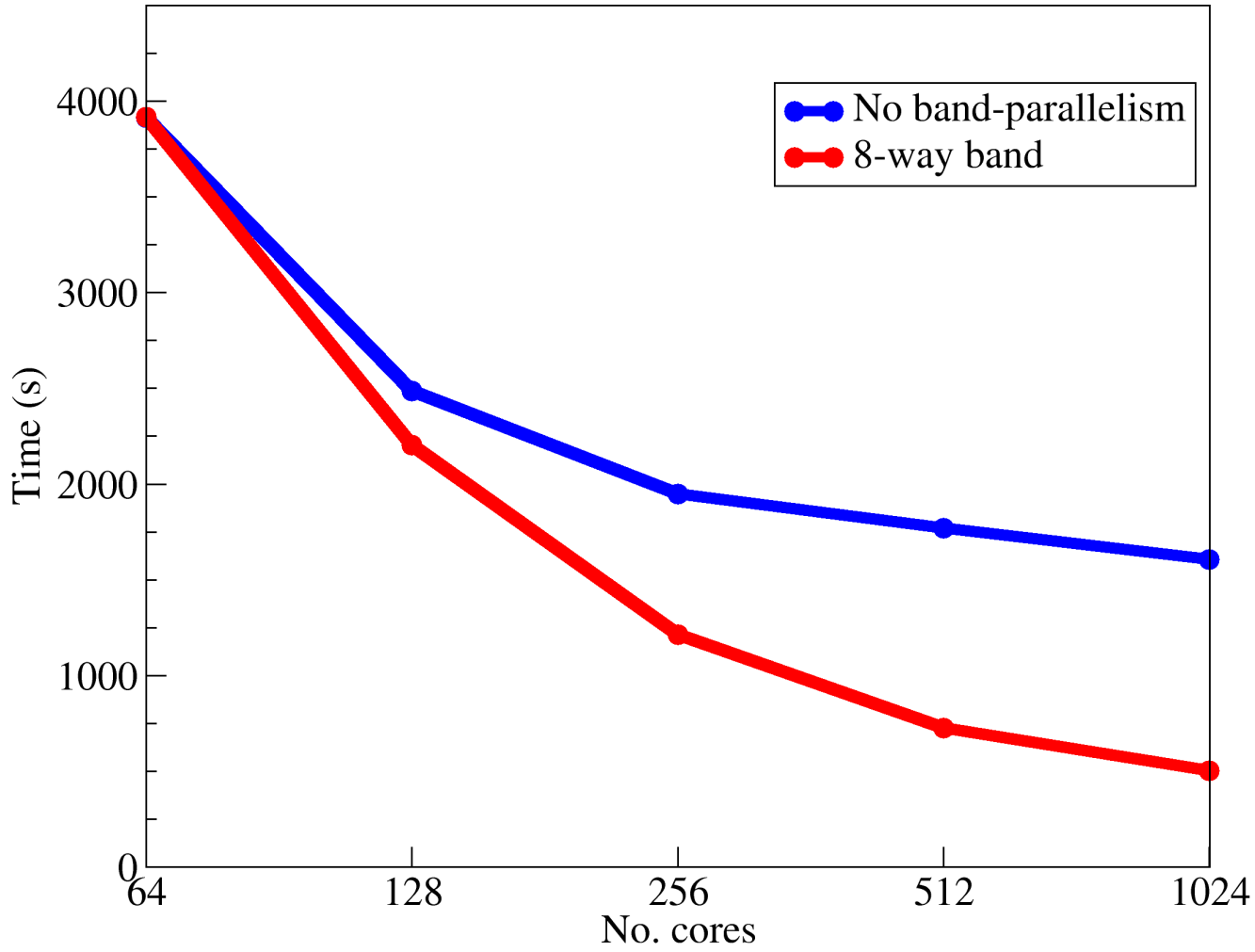
# Mixing k, G and B parallelism

- ***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 orthogonalization, needs high bandwidth interconnect

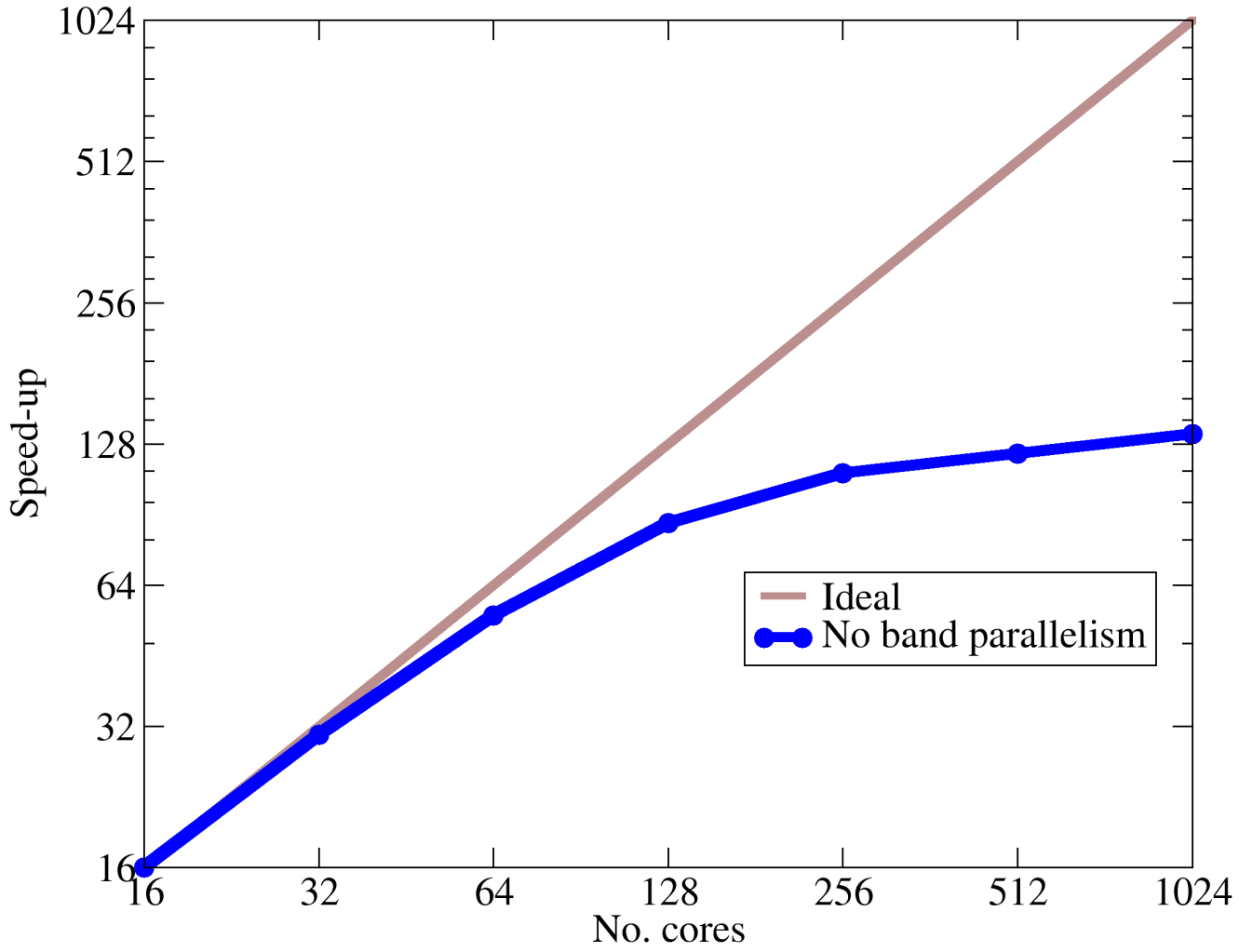
# A bigger benchmark

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

# $Al_2O_3$ time

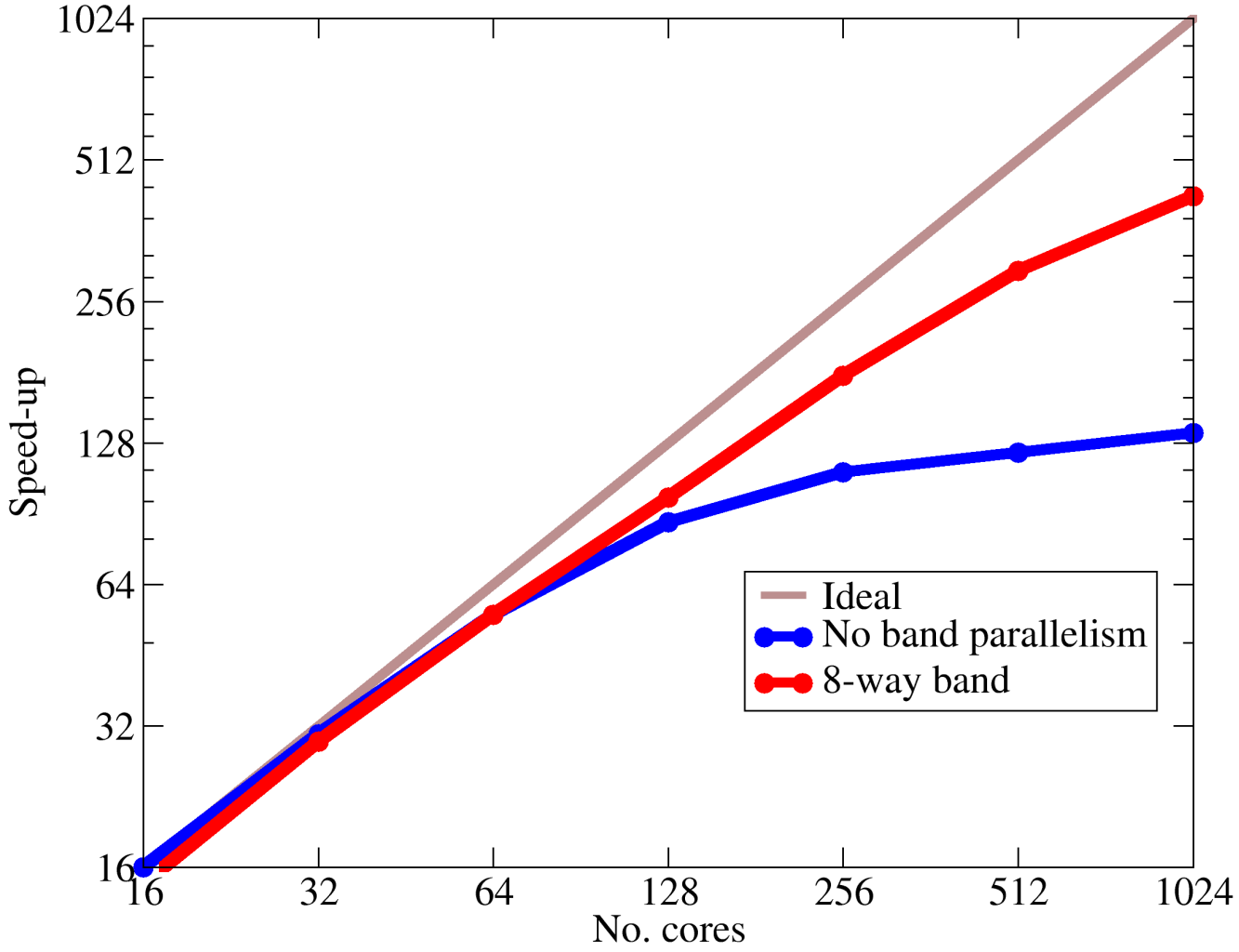

No band-parallelism
8-way band
Time (s)
0
1000
2000
3000
4000
64
128
256
512
1024
No. cores

# $Al_2O_3$ 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!

# Γ-point optimization

# Very large system sizes

- For isolated systems, or very large unit cells, only need 1 k-point
  - Can choose to be $\boldsymbol{k}$ = (0,0,0) = $\Gamma$-point of BZ
- Why?
  - At $\Gamma$ the bands are real in real-space
    - Fourier coefficient at $c(-\boldsymbol{G}) = c^*(+\boldsymbol{G})$
    - So only need ½ the Fourier transforms
  - And inner products are real so no need to compute imaginary parts
  - x2 speed on FFT and x8 on orthogonalization

# Γ-point calculations

- CASTEP can automatically detect Γ-point calculations and switch internally
  - Saving in memory and time
  - Orthogonalization gain bigger than FFT so may look like scaling worse but still faster!
- Need to be careful on science – is Γ-point really good enough?
  - Sometimes the speed gain means it is better to go to a larger cell to exploit Γ-point

# 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-parallelize* a calculation – can go *slower* if put in too many cores as comms cost will dominate