Band Parallelism in CASTEP:
Scaling to More Than 1000 Cores

P.J. Hasnip¹  M. Ashworth²  M. Plummer²  
M.I.J. Probert¹  K. Refson³

¹University of York  ²STFC, Daresbury  ³STFC, Rutherford Appleton Laboratory

Cray User Group Meeting 2009
Outline

1. Introduction
   - What is Castep?
   - How does Castep work?
   - Castep in Parallel

2. Band Parallelism
   - Proposal
   - Castep development
   - Results
Castep is...

- A general-purpose ‘first principles’ atomistic modelling program
- Based on density functional theory
- Used on many HPC machines, including the UK National HPC Service, HECToR (XT4)
Castep can...

- Compute the electronic density
- Determine the groundstate atomic configuration and cell
- Simulate molecular dynamics (path-integrals, variable cell)
- Calculate band-structures and density of states
- Compute various spectra (optical, IR, Raman, NMR, XANES...)
- plus linear response, population analysis, ELF, etc.
 Castep is written using...

- Fortran 95
- BLAS/LAPACK for linear algebra
- FFT libraries (where available)
- MPI for parallel communication

Portable and well optimised (achieves 37-40% peak on XT4).
Castep solves a set of Schrödinger equations,

\[ H_k[n] \psi_{bk}(r) = \epsilon_{bk} \psi_{bk}(r) \]

where \( n \) is the electronic density and \( \{\psi_{bk}\} \) are the bands.

\[ n(r) = \sum_{bk} 2w_k |\psi_{bk}(r)|^2 \]
Self-consistency

\[ H_k \[ n \] \psi_{bk} (r) = \epsilon_{bk} \psi_{bk} (r) \]

- \( H_k \) depends on \( n(r) \)
- \( n(r) \) depends on \( \{ \psi_{bk} \} \)

We need to solve this eigenvalue equation iteratively until we have self-consistency.
How Castep Works

1. Guess density
2. Construct density
3. Construct H
4. Is energy change less than E_tol?
   - No: Go back to Construct density
   - Yes: Diagonalise H to get new bands
5. Write out density, bands etc.
We expand $\psi_{bk}$ in a plane-wave (Fourier) basis,

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

The fundamental data object in CASTEP is the wavefunction data type, which stores the complex coefficients $c_{Gbk}$ for all bands at all k-points:

$$\text{wvfn\%coeffs}(1:nG,1:nbands,1:nkpts)$$

Thus for a given k-point, any band is just a vector of length $N_G$ and the Hamiltonian $\hat{H}_k$ is a $N_G \times N_G$ matrix.
A useful basis set

\[ \psi_{bk}(\mathbf{r}) = \sum_{\mathbf{G}} c_{Gb_k} e^{i(\mathbf{G}+\mathbf{k}) \cdot \mathbf{r}} \]

If we increase the size of our simulation system:

- The size of the smallest $\mathbf{G}$-vector decreases
- The number of $\mathbf{G}$-vectors, $N_G$, *increases*
- On HPC machines $N_G$ might be $O(100,000)$
The \( \mathbf{k} \)-points

The vectors \( \{ \mathbf{k} \} \) sample the region of reciprocal-space

\[
|\mathbf{k}| < \frac{1}{2} |\mathbf{G}_{\text{smallest}}|.
\]

We increase the \( \mathbf{k} \)-point density until our calculation converges.

If we increase the size of our simulation system:

- The size of the smallest \( \mathbf{G} \)-vector decreases
- The number of \( \mathbf{k} \)-points we need decreases
- On HPC machines \( N_k \) might be \( O(1) \)
We find $\psi_{bk}$ by varying $\{c_{Gbk}\}$ to minimise $\epsilon_{bk}$.

To prevent all the bands heading for the lowest energy one, we explicitly orthogonalise them to each other.

The computational time per $\mathbf{k}$-point scales as

$$N_G N_b^2.$$  

i.e. cubically for large systems (recall $N_k = O(1)$).

This cost dominates in large calculations.
Fourier Transforms

$H$ is a $N_G \times N_G$ matrix, and $N_G = O(100,000)$
$\Rightarrow$ too large to compute, store or apply explicitly.

Some contributions to $H$ are diagonal in $G$-space, and some in $r$-space $\rightarrow$ store and apply these separately.

FFTs are used to switch between $r$- and $G$-space. The computational time per $k$-point scales as

$$N_G \ln (N_G) N_b$$

i.e. approximately quadratically for large systems.
**k-point parallelism**

\[ H_k[n] \psi_{bk}(r) = \epsilon_{bk} \psi_{bk}(r) \]

The eigenvalue equations for different \( k \)-points are only weakly coupled.

- Distribute data and workload by \( k \)-point
- Gives near-perfect scaling
- Large calculations only need \( O(1) \) \( k \)-points
  \( \implies \) run out of them very quickly!
The TiN simulation is a small standard benchmark

- 33 atoms
- 8 $k$-points
- 164 bands
- 10,972 $G$-vectors
G-vector parallelism

\[ \psi_{bk}(r) = \sum_G c_{Gbk} e^{i(G+k) \cdot r} \]

- Distribute the data and workload over the \( \mathbf{G} \)-vectors
- \( N_G \) is large, and increases with system size
- Fourier transforms require all-to-all communications
- Good scaling for moderate numbers of cores
TiN G-vector Parallel

![Graph showing speed-up vs. number of PEs]

- Speed-up vs. No. PEs
- Two lines: Ideal and Castep 4.2

**Legend:**
- Ideal
- Castep 4.2

**Axes:**
- Y-axis: Speed-up
- X-axis: No. PEs

---

P. Hasnip, M. Ashworth, M. Plummer, M. Probert, K. Refson

Band Parallelism in CASTEP
Mixed Parallelism

- \( \text{k} \)-point parallelism near-perfect to 8 cores
- \( \text{G} \)-vector parallelism good to 16 or 32 cores
- We allow both simultaneously
TiN in Mixed \( k \)- and \( G \)-Parallel
Reducing All-to-All Communications

- Ultimate scaling dominated by all-to-all
- Castep already optimised to minimise FFTs
- Split into two phases:
  - local core-to-core within each node
  - node-to-node
- Reduced all-to-all, but additional comms phase
- Controlled by `num_proc_in_smp` parameter
- HECToR has one dual-core CPU per node
  ⇒ can set `num_proc_in_smp` to 1 or 2
TiN with all-to-all optimisations

![Graph showing speed-up vs number of PEs for different configurations.]

- **Ideal**
- **SMP 1**
- **SMP 2**

**Axes:**
- Y-axis: Speed-up
- X-axis: No. PEs

**Legend:**
- Ideal
- SMP 1
- SMP 2

**Observations:**
- The speed-up increases with the number of PEs.
- SMP 1 and SMP 2 show similar trends, with SMP 2 slightly higher.
- The ideal performance is represented by a straight line.

**Note:**
- The graph illustrates the efficiency of different parallel configurations.
- The performance improvement is more pronounced with higher numbers of PEs.
The TiN benchmark is quite small. A larger standard benchmark system $\text{Al}_2\text{O}_3$ slab (3×3 surface):

- 270 atoms
- 2 $\mathbf{k}$-points
- 778 bands (1296 electrons)
- 88,184 $\mathbf{G}$-vectors

Too large to run in serial, so performance measured wrt 16 cores.
Al$_2$O$_3$-3x3 time

The graph shows the time (in seconds) taken to compute Al$_2$O$_3$-3x3 configurations with different numbers of processing elements (PEs) for two different configurations, labeled SMP 1 and SMP 2. The time decreases as the number of PEs increases, indicating improved performance with parallel processing.

- SMP 1:
  - 16 PEs: Time is significantly high.
  - 1024 PEs: Time decreases significantly.

- SMP 2:
  - 16 PEs: Time is relatively high.
  - 1024 PEs: Time decreases significantly.

The graph implies that Castep in parallel can significantly reduce computation time for large-scale simulations.
**Band Parallel Proposal**

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

- Distribute data and workload over the bands
- \( N_b \) is moderately large, and increases with system size
- On HPC machines \( N_b \) might be \( O(1000) \)
- Band-rotations now require all-to-all communications
The Band Parallel Project

8 month project to:

- Investigate Castep performance on HECToR XT4
- Implement band-parallelism
- Parallelise costly non-distributed operations
Castep development

Castep 4.2 was used as the base for this project:

- 334,395 lines of Fortran 90
- 54 modules
- Already has multiple levels of parallelism
Using Cray PAT we profiled the Al2O3 benchmark on 512 cores.

- 30% of time in FFTs
- 30% of time in ZGEMM of one subroutine

Scaling is limited by all-to-all, as expected (MPI_AlltoAllv).
Band Parallel Implementation

- Band-parallelism implemented as an additional level of parallelism
- Bands distributed round-robin for load-balancing
- ScaLAPACK used for parallel matrix diagonalisation/inversion
Results for $\text{Al}_2\text{O}_3$ $3 \times 3$
Results for $\text{Al}_2\text{O}_3 \times 3$
Results for $\text{Al}_2\text{O}_3 \ 3 \times 3$
Results for $\text{Al}_2\text{O}_3$ $3 \times 3$
A real system

So far we’ve only looked at benchmarks—now we’ll look at something more interesting:

- Imidazolium chloride—a room-temperature ionic liquid
- 408 atoms
- 1 $k$-point
- 662 bands
- 137,728 $G$-vectors
- Want to run a molecular dynamics simulation

Can only be run on 64 PEs or more.
Immidazolium chloride
Immidazolium chloride with band-parallelism
Imidazolium chloride speed-up

![Graph showing speed-up with increasing number of PEs for Ideal, SMP 1, and SMP 2 cases.]
Immidazolium chloride speed-up

- Ideal
- SMP 1
- SMP 2
- 8-way band (SMP 1)
- 8-way band (SMP 2)
Imidazolium chloride performance

- Poorer performance for 64 and 128 PEs due to:
  - Lack of Hermitian subroutines in ScaLAPACK
  - Enforce same no. updates for each band
- Good scaling to 512 cores even with only 1 k-point
- Achieves 1 SCF cycle per minute on 1024 cores
Remaining work

- Extend band-parallelism to non-groundstate calculations (e.g. NMR, linear response)
- Develop ‘band-local’ optimisers to improve scaling
Acknowledgements

- Alan Simpson (EPCC)
- Christof Vömel (Lawrence Berkeley National Lab)
- NAG (Oxford), especially Guy Robinson, Ian Reid, Edward Smyth, Sarfraz Nadeem and Phil Ridley.
- Financial assistance from EPSRC (UK) via dCSE grant.
Band parallelism implemented on top of existing parallelism
4 times more cores can now be used efficiently
Code quicker even for moderate numbers of cores
We investigated the impact on performance of...

- Compiler
- BLAS/LAPACK libraries
- FFT libraries

Results shown are for the smaller TiN benchmark.
BLAS/LAPACK

![Bar chart showing time in seconds for different versions of ACML and LibSci libraries for ZGEMM operation.]

ACML 3.6.1, ACML 4.0.1, ACML 4.0.1a, LibSci 10.2.1
Introduction
Band Parallelism
Summary

FFT

P. Hasnip, M. Ashworth, M. Plummer, M. Probert, K. Refson

Band Parallelism in CASTEP