Band Parallelism in CASTEP:

Scaling to More Than 1000 Cores

P.J. Hasnip[1] M. Ashworth[2] M. Plummer[2]
M.I.J. Probert[1] K. Refson[3]

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

Cray User Group Meeting 2009

Outline

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 Basics

Castep solves a set of Schrödinger equations,

$$\mathrm{H}_k[n]\psi_{bk}(\mathbf{r}) = \epsilon_{bk}\psi_{bk}(\mathbf{r})$$

where n is the electronic density and $\{\psi_{bk}\}$ are the *bands*.

$$n(\mathbf{r}) = \sum_{bk} 2w_k \left|\psi_{bk}(\mathbf{r})\right|^2$$

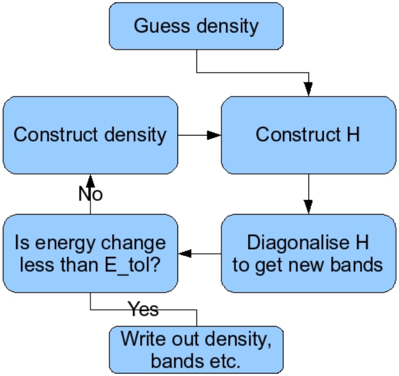
Self-consistency

$$\mathrm{H}_k[n]\psi_{bk}(\mathbf{r}) = \epsilon_{bk}\psi_{bk}(\mathbf{r})$$

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

We need to solve this eigenvalue equation iteratively until we have *self-consistency*.

How Castep Works


Guess density
Construct density
Construct H
No
Is energy change less than E_tol?
Diagonalise H to get new bands
Yes
Write out density, bands etc.

A useful basis set

We expand ψ_{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:

```
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

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

If we increase the size of our simulation system:

- The size of the smallest **G**-vector decreases
- The number of **G**-vectors, N$_G$, *increases*
- On HPC machines N$_G$ might be O(100,000)

The **k**-points

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

$$|\mathbf{k}| < \frac{1}{2}\left|\mathbf{G}_{smallest}\right| .$$

We increase the **k**-point density until our calculation converges.

If we increase the size of our simulation system:

- The size of the smallest **G**-vector decreases
- The number of **k**-points we need *decreases*
- On HPC machines N_k might be O(1)

Orthogonalisation

We find ψ_{bk} by varying $\{c_{Gbk}\}$ to minimise ϵ_{bk}.

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

The computational time per **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 $\longrightarrow$ 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

$$\mathrm{H}_k[n]\psi_{bk}\left(r\right) = \epsilon_{bk}\psi_{bk}\left(r\right)$$

The eigenvalue equations for different **k**-points are only weakly coupled.

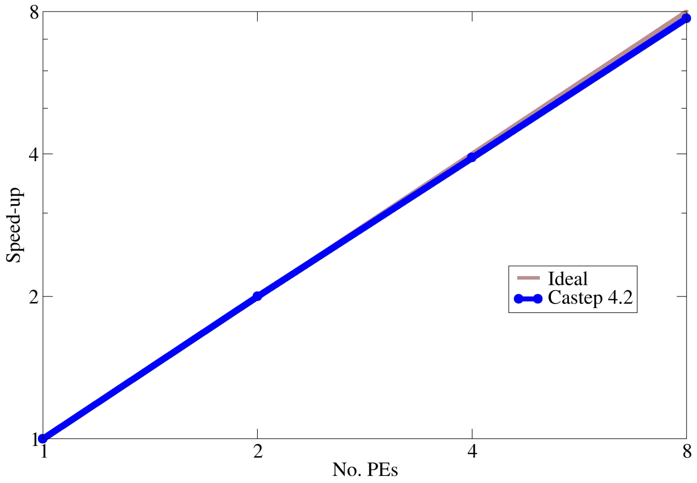
- Distribute data and workload by **k**-point
- Gives near-perfect scaling
- Large calculations only need O(1) **k**-points
  $\Longrightarrow$ run out of them very quickly!

TiN Benchmark

The TiN simulation is a small standard benchmark

- 33 atoms
- 8 **k**-points
- 164 bands
- 10,972 **G**-vectors

TiN **k**-point Parallel

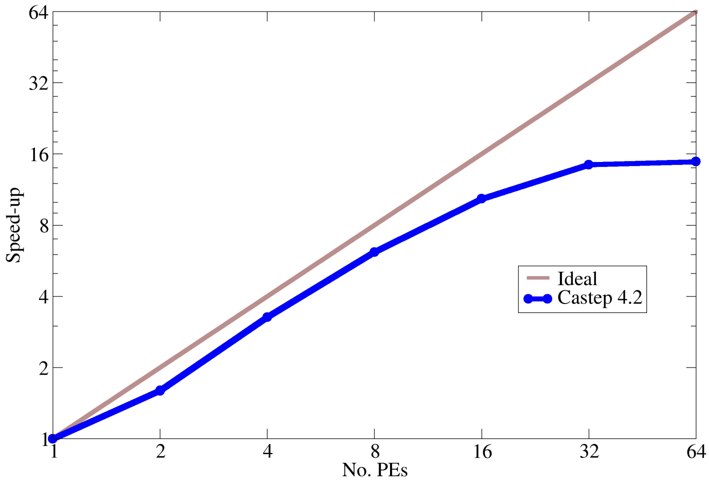


G-vector parallelism

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

- Distribute the data and workload over the **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

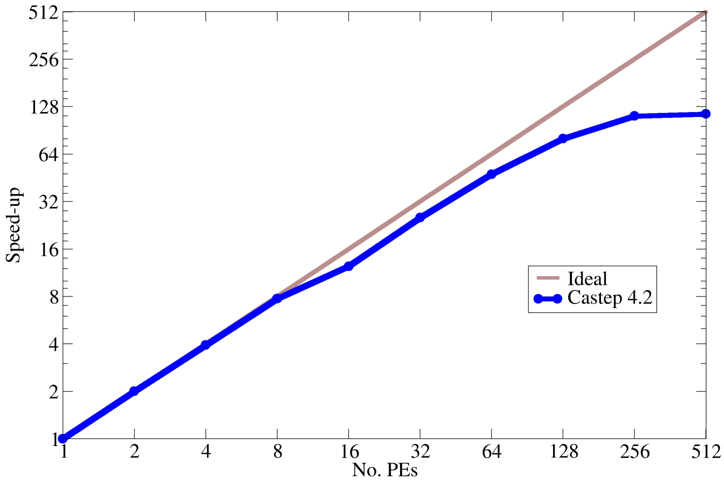
TiN **G**-vector Parallel



Mixed Parallelism

- **k**-point parallelism near-perfect to 8 cores
- **G**-vector parallelism good to 16 or 32 cores
- We allow both simultaneously

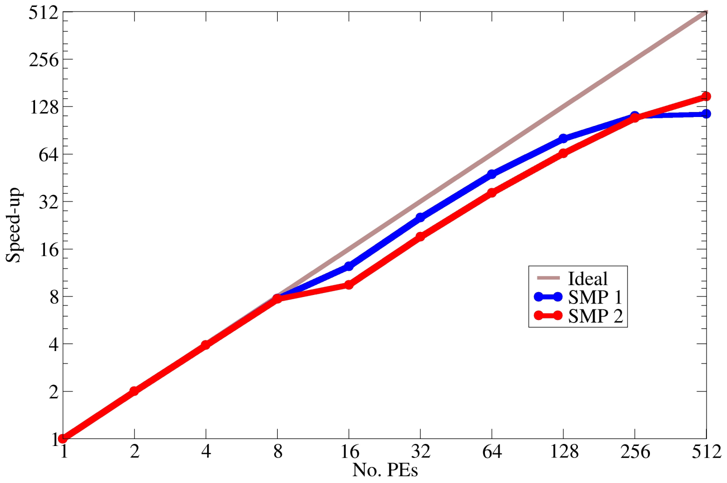
TiN in Mixed **k**- and **G**-Parallel



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
  $\Rightarrow$ can set `num_proc_in_smp` to 1 or 2

TiN with all-to-all optimisations



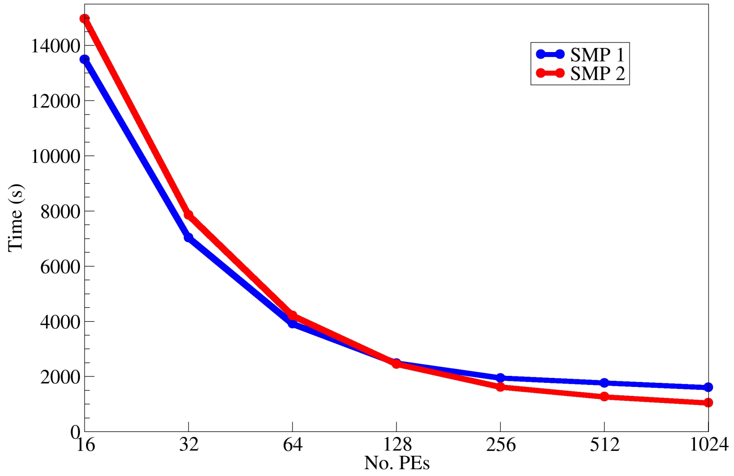
Al_2O_3-3x3 benchmark

The TiN benchmark is quite small. A larger standard benchmark system Al_2O_3 slab (3×3 surface):

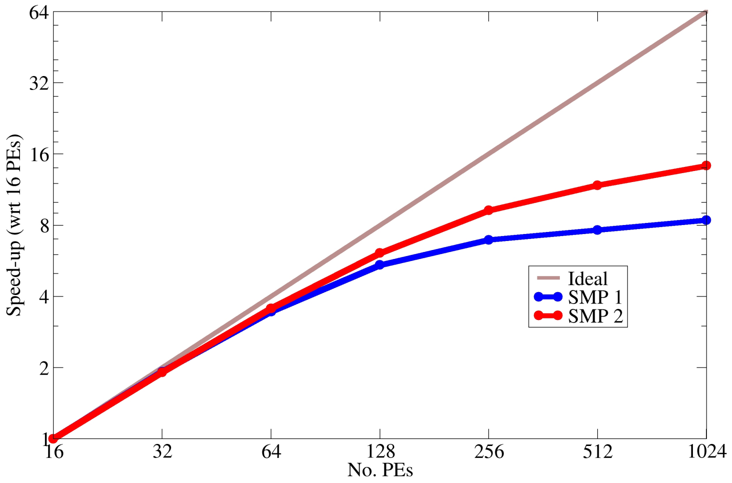
- 270 atoms
- 2 **k**-points
- 778 bands (1296 electrons)
- 88,184 **G**-vectors

Too large to run in serial, so performance measured wrt 16 cores.

Al_2O_3-3x3 time



Al_2O_3-3x3 parallel scaling



Band Parallel Proposal

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

- 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

Castep Profile

Using Cray PAT we profiled the Al2O3 benchmark on 512 cores.

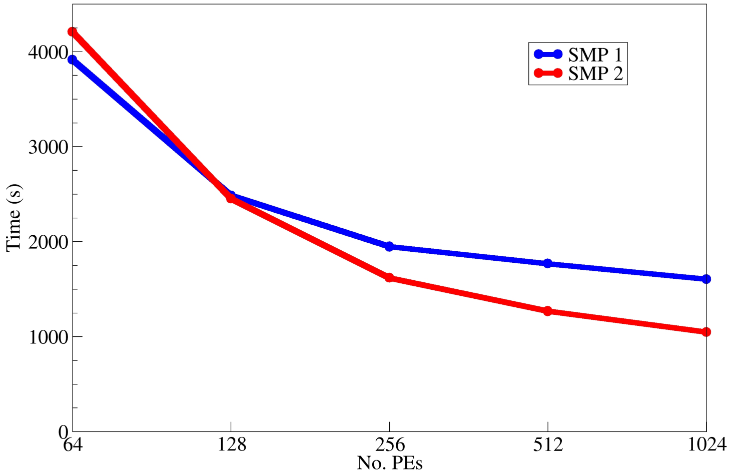
- 30% of time in FFTs
- 30% of time in ZGEMM of one subroutine

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

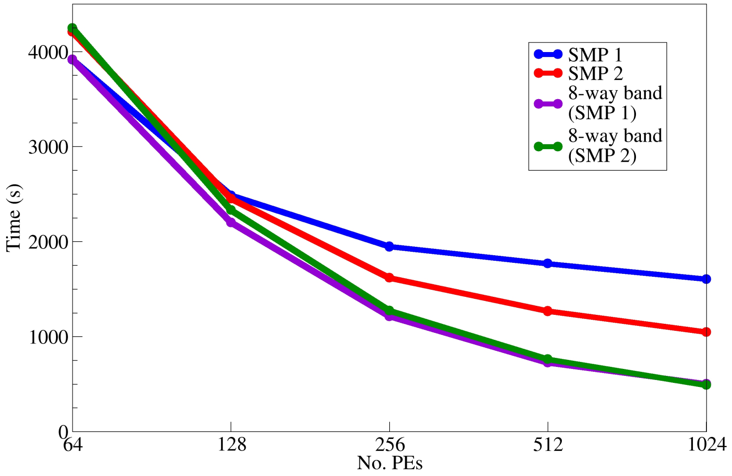
Band Parallel Implementation

- Band-parallelism implemented as an *additional* level of parallelism
- Bands distributed round-robin for load-balancing
- ScaLAPACK used for parallel matrix diagonalisation/inversion

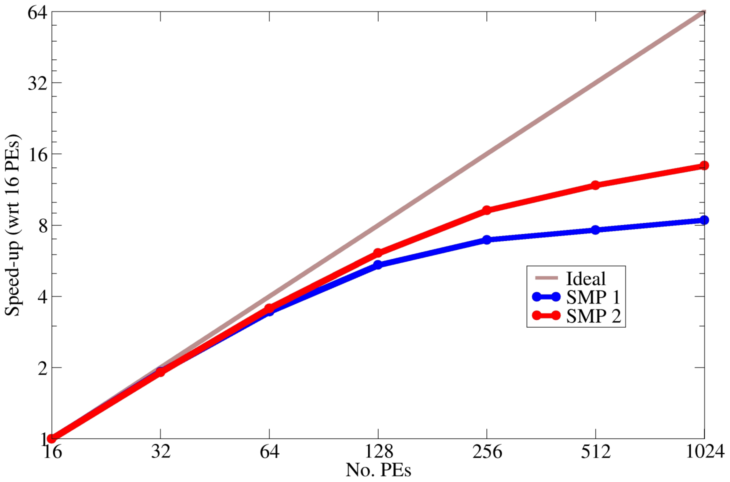
Results for Al_2O_3 3×3



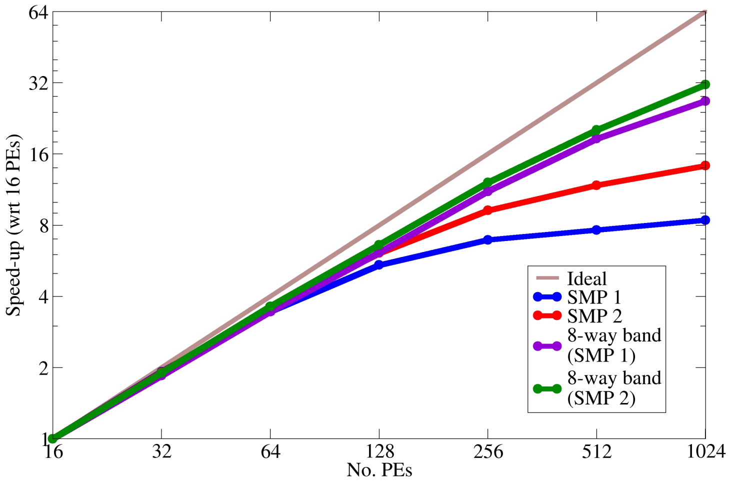
Results for Al_2O_3 3×3



Results for Al_2O_3 3×3



Results for Al_2O_3 3×3



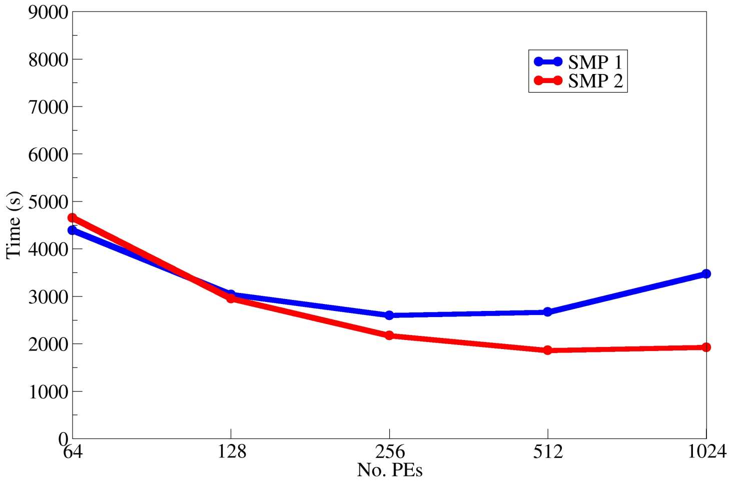
A real system

So far we've only looked at benchmarks–now we'll look at something more interesting:

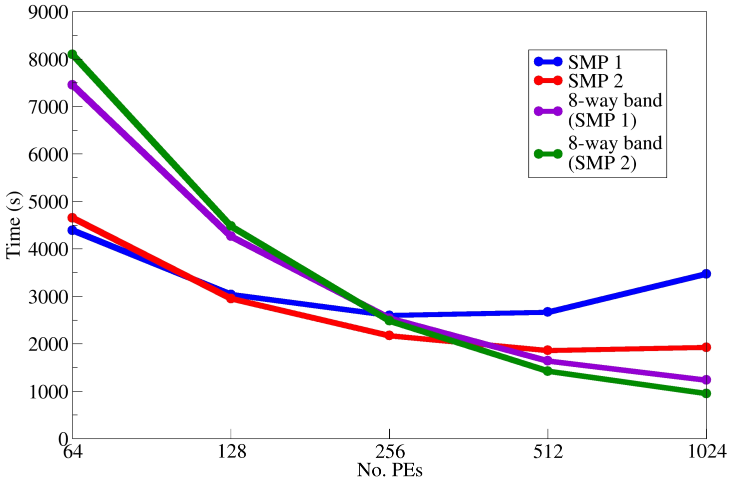
- Immidazolium chloride–a room-temperature ionic liquid
- 408 atoms
- 1 **k**-point
- 662 bands
- 137,728 **G**-vectors
- Want to run a molecular dynamics simulation

Can only be run on 64 PEs or more.

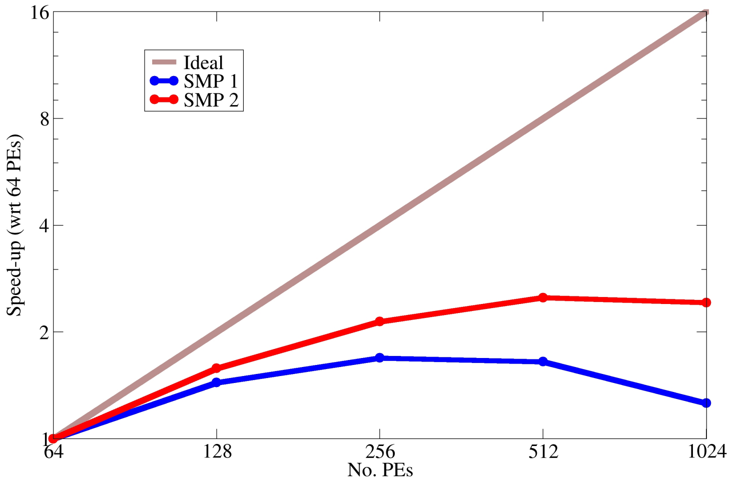
Immidazolium chloride



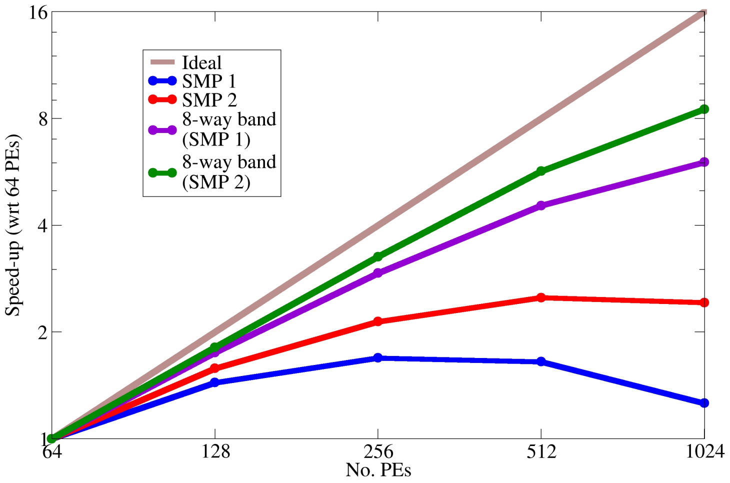
Immidazolium chloride with band-parallelism



Immidazolium chloride speed-up



Immidazolium chloride speed-up



Immidazolium 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.

Summary

- 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

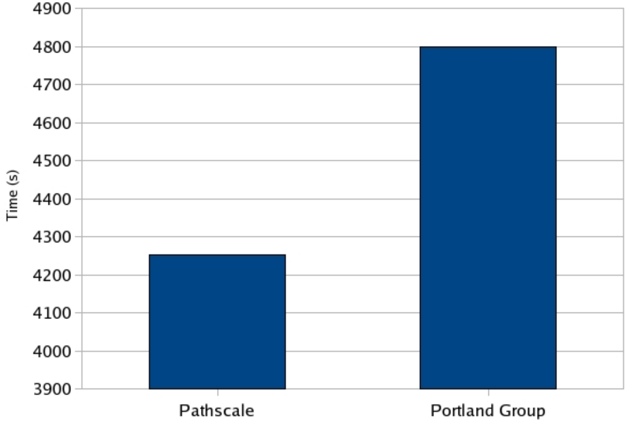
Castep on HECToR

We investigated the impact on performance of...

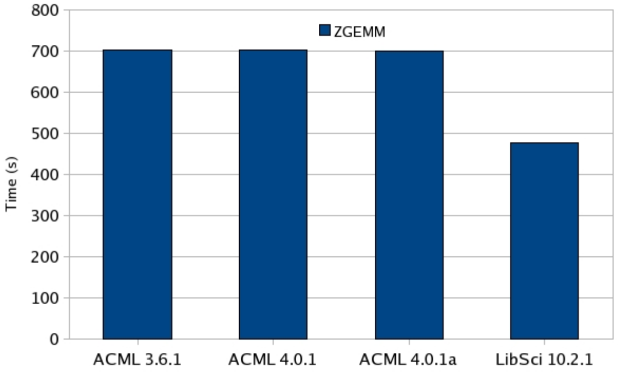
- Compiler
- BLAS/LAPACK libraries
- FFT libraries

Results shown are for the smaller TiN benchmark.

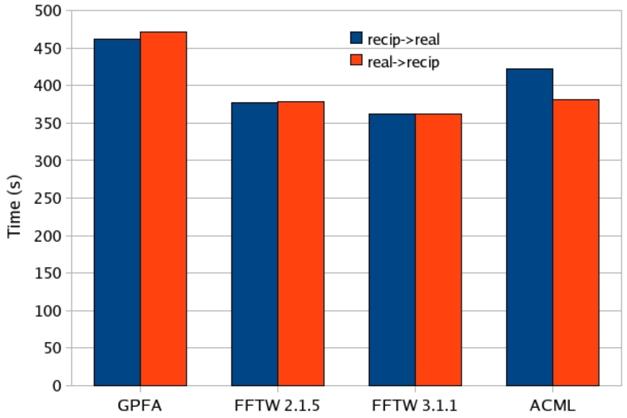
Compiler


Time (s)
4900
4800
4700
4600
4500
4400
4300
4200
4100
4000
3900
Pathscale
Portland Group

BLAS/LAPACK


800
700
600
500
400
300
200
100
0
Time (s)
ZGEMM
ACML 3.6.1
ACML 4.0.1
ACML 4.0.1a
LibSci 10.2.1

FFT


500
450
400
350
300
250
200
150
100
50
0
Time (s)
recip->real
real->recip
GPFA
FFTW 2.1.5
FFTW 3.1.1
ACML