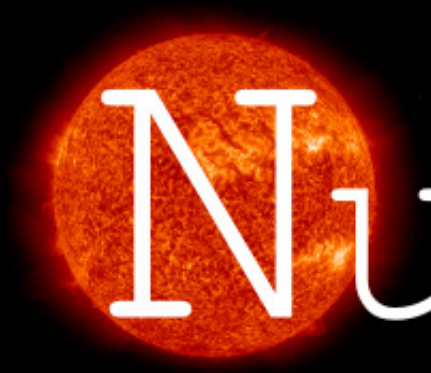


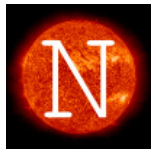
Nu-FuSE

Porting CASTEP to GPGPUs

Adrian Jackson, Toni Collis,
EPCC, University of Edinburgh

Graeme Ackland

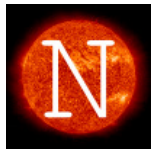
University of Edinburgh



http://www.nu-fuse.com

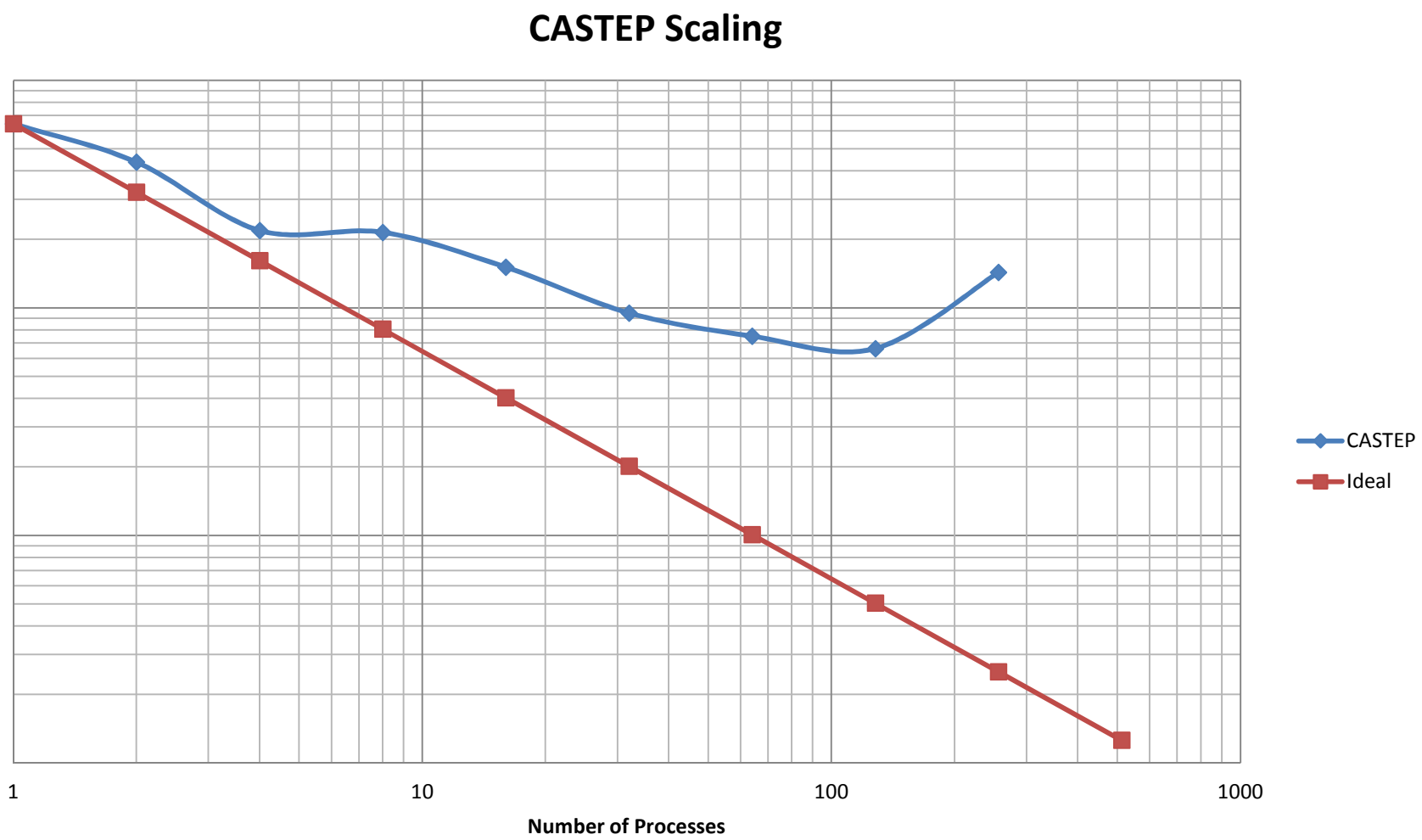
CASTEP

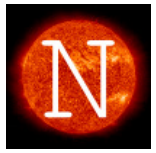
- Density Functional Theory
  - Plane-wave basis set with pseudo potentials
  - Heavy use of FFTs
  - FORTRAN (modern) and MPI for parallelisation
  - plane-waves, k-points and bands data decompositions
- Significant use on UK HPC systems



CASTEP Scaling

http://www.nu-fuse.com

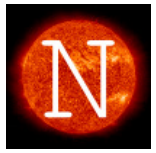




Capabilities

http://www.nu-fuse.com

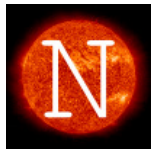
- **Hamiltonians**
- DFT XC-functionals LDA, PW91, PBE, RPBE, PBEsol, WC
- Hybrid functionals PBE0, B3LYP, sX-LDA, the HSE family of functionals (including user-defined parameterisation)
- LDA+U and GGA+U
- Semi-empirical dispersion corrections (DFT+D)
- **Structural methods**
- Full variable-cell geometry optimisation using BFGS, LBFGS and TPSD
- Geometry optimisation using internal co-ordinates
- Geometry optimisation using damped molecular dynamics
- Transition-state search using LST/QST method
- **Molecular Dynamics**
- Molecular Dynamics including fixed and variable-cell MD
- NVE, NVT, NPH and NPT ensembles
- Path-integral MD for quantum nuclear motion
- **Vibrational Spectroscopy**
- Phonon dispersion and DOS over full Brillouin-Zone using DFPT methods
- Phonon dispersion and DOS over full Brillouin-Zone using supercell methods
- IR and raman intensities
- **Dielectric Properties**
- Born effective charges and dielectric permittivity
- Frequency-dependent dielectric permittivity in IR range
- Wannier Functions
- Electrostatic correction for polar slab models
- **Solid-state NMR spectroscopy**
- Chemical Shifts
- Electric Field Gradient tensors
- J-coupling
- **Optical and other Spectroscopies**
- EELS/ELNES and XANES Spectra
- Optical matrix elements and spectra
- **Electronic properties**
- Band-structure calculations
- Mulliken population analysis
- Hirshfeld population analysis
- Electron Localisation Functions (ELF)
- **Pseudopotentials**
- Supports Vanderbilt ultrasoft and norm-conserving pseudopotentials
- Built in "On The Fly" pseudopotential generator
- Self-consistent Pseudpotentials
- (non self-consistent) PAW for properties calculations
- **Electronic Solvers**
- Block Davidson solver with density mixing
- Ensemble DFT for metals



Motivation

http://www.nu-fuse.com

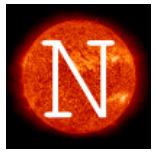
- Demonstrator
  - Investigate whether it makes sense
  - What data transfers are necessary
  - CASTEP 7.0
  - No divergence from mainstream
  - No intrusion into physics
- Single GPU
  - Large simulations on desktop
- Multiple GPU
  - Utilise large GPU'd systems
  - Enable future UK HPC systems to be GPU'd



CASTEP: initial accelerator investigation

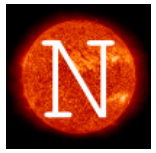
- Replace blas calls with cula
  - (cuda-blas library http://www.culatools.com/)
- Replace fft calls with cufft
  - NLCX and Geometry Optimisation
  - Small simulation, to fit on one CPU, no MPI calls. 4 Ti atoms, 2 O atoms, total of 32 electrons.
    - No device calls runtime = 14.6s
    - Cula blas calls runtime = 31.1s
    - Cula blas and cufft calls runtime = 418s.

Majority of the increased runtime was due to data transfer.



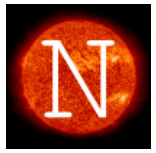
GPU*ification* of CASTEP

- Aim:
  - remove data transfer problems by placing most of the large data structures on the GPU.
  - Use OpenACC kernels, PGI CUDA fortran, cula blas and cufft.
- The process:
  - 'All or nothing' approach, moving large data structures onto the GPU and all affected routines/functions (approximately 50 subroutines)
  - Focus on the serial version first.
  - After initial compilation expect to spend some time optimising, particularly data transfers
  - Move onto mpi version.



OpenACC Directives

- With directives inserted, the compiler will attempt to compile the key kernels for execution on the GPU, and will manage the necessary data transfer automatically.
- Directive format:
  - C: `#pragma acc ….`
  - Fortran: `!$acc ….`
- These are ignored by non-accelerator compilers

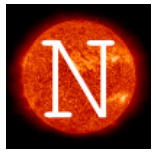


OpenACC

http://www.nu-fuse.com

```
PROGRAM main
   INTEGER :: a(N)
   …
!$acc data copy(a)
!$acc parallel loop
   DO i = 1,N
       a(i) = i
   ENDDO
!$acc end parallel
   loop
   CALL double_array(a)
!$acc end data
   …
END PROGRAM main
```

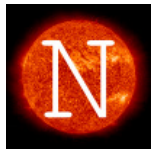
```
SUBROUTINE double_array(b)
   INTEGER :: b(N)
!$acc kernels loop present(b)
   DO i = 1,N
       b(i) = 2*b(i)
   ENDDO
!$acc end kernels loop
END SUBROUTINE double_array
```



GPU*ification* of CASTEP

Data structures on device

- Wavefunctions:
  - `complex(kind=dp) :: Wavefunction%coeffs(:,:,:,:)`
  - `complex(kind=dp) :: Wavefunction%beta_phi(:,:,:,:)`
  - `real(kind=dp) :: Wavefunction%beta_phi_at_gamma(:,:,:,:)`
  - `logical :: Wavefunction%have_beta_phi(:,:)`
  - `complex(kind=dp) :: Wavefunctionslice%coeffs(:,:)`
  - `complex(kind=dp) :: Wavefunctionslice%realspace_coeffs(:,:)`
  - `real(kind=dp) :: Wavefunctionslice%realspace_coeffs_at_gamma(:,:)`
  - `logical :: Wavefunctionslice%have_realspace(:)`
  - `complex(kind=dp) :: Wavefunctionslice%beta_phi(:,:)`
  - `real(kind=dp) :: Wavefunctionslice%beta_phi_at_gamma(:,:)`
- Bands
  - `complex(kind=dp) :: coeffs(:)`
  - `complex(kind=dp) :: beta_phi(:)`
  - `real(kind=dp) :: beta_phi_at_gamma(:)`

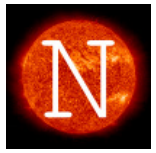


Example use of kernels

http://www.nu-fuse.com

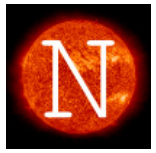
```
subroutine wave_copy_wv_wv_ks
……
!$acc kernels present_or_copy(wvfn_dst, wvfn_src)
!Map  reduced representation of coefficients on k-point
        do nb=1,nbands_to_copy
          recip_grid = cmplx_0
          call
basis_recip_reduced_to_grid(wvfn_src%coeffs(:,nb,nk_s,ns_s),nk_src,recip_grid,'S
TND')
          call
basis_recip_grid_to_reduced(recip_grid,'STND',wvfn_dst%coeffs(:,nb,nk_d,ns_d),nk
_dst)
        end do
……
     ! copy rotation data
……
      do nb=1,nbands_to_copy
        do nb2=1,nbands_to_copy
          wvfn_dst%rotation(nb,wvfn_dst%node_band_index
(nb2,id_in_bnd_group),nk_dst,ns_dst) = &
                                        &
wvfn_src%rotation(nb,wvfn_src%node_band_index(nb2,id_in_bnd_group),nk_src,ns_src
)
        end do
      end do
……
!$acc end kernels

end subroutine wave_copy_wv_wv_ks
```

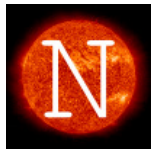


GPU*ification* of CASTEP

- Module procedures used throughout the code
  - Multiple calls for all the core kernels
- Module procedures support different data structures for same call
  - Interface chooses different routines
- CASTEP uses language options that are not supported on devices, such as the use of '`optional`' types when passing data to subroutines followed by '`if present`' statements.
  - Resolved by creating copies of subroutines with and without optional arguments.
- Specifying arrays with `dimension(*)` when passing to subroutines
  - Resolved by specifying correct dimension structure, sometimes requiring multiple copies of subroutines




```
subroutine
basis_real_to_recip_gamma(grid,grid_type,num_grids,gamma)
    real(kind=dp), dimension(*), intent(inout) :: grid
    character(len=*), intent(in) :: grid_type
    complex(kind=dp), dimension(*), intent(out) :: gamma
```



Example modification

http://www.nu-fuse.com

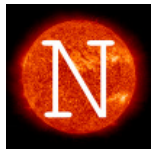
```
interface basis_real_to_recip_gamma
    module procedure basis_real_to_recip_gamma_1d
    module procedure basis_real_to_recip_gamma_2d_grid
    module procedure basis_real_to_recip_gamma_2d_gamma
    module procedure basis_real_to_recip_gamma_2d_grid_2d_gamma
    module procedure basis_real_to_recip_gamma_3d_gamma
    module procedure basis_real_to_recip_gamma_3d_grid_3d_gamma
 end interface
subroutine basis_real_to_recip_gamma_2d_grid_2d_gamma(grid,grid_type,num_grids,gamma)
   implicit none
   integer, intent(in) :: num_grids
   real(kind=dp), dimension(:,:), intent(inout) :: grid
   character(len=*), intent(in) :: grid_type
   complex(kind=dp), dimension(:,:), intent(out) :: gamma

   real(kind=dp), dimension(:), allocatable :: temp_grid
   complex(kind=dp), dimension(:), allocatable :: temp_gamma

   allocate(temp_grid(size(grid)))
   allocate(temp_gamma(size(gamma)))

   temp_grid = reshape(grid,shape(temp_grid))
   temp_gamma = reshape(gamma,shape(temp_gamma))
   call basis_real_to_recip_gamma_inner(temp_grid,grid_type,num_grids,temp_gamma)
   grid = reshape(temp_grid,shape(grid))
   gamma = reshape(temp_gamma,shape(gamma))

   deallocate(temp_grid,temp_gamma)
 end subroutine basis_real_to_recip_gamma_2d_grid_2d_gamma
```



GPU*ification* of CASTEP

- Data that is involved in I/O needs to be taken off the device (copies of data need to be made):

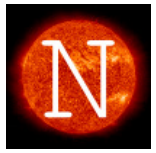
Original code (from ion.CUF):

```
   read(wvfn%page_unit,REC=record,iostat=status)
((wvfn%coeffs(np,nb,1,1),np=1,wvfn%waves_at_kp(nk)),nb=1,wvfn
%nbands_max)
```

New code:

```
read(wvfn%page_unit,REC=record,iostat=status)
((coeffs_tmp,np=1,wvfn%waves_at_kp(nk)),nb=1,wvfn%nbands_max)
wvfn%coeffs(np,nb,1,1) = coeffs_tmp
```

- Sometimes the limitations of what is on and off the device results in multiple `!$acc kernel` regions very close together, and not the entire subroutines, which is not necessarily very efficient. Will require a lot of fine tuning to improve performance.
- Currently still working on successfully compiling the serial code.



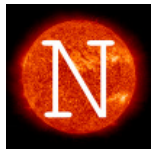
GPU*ification* of CASTEP

http://www.nu-fuse.com

Still an ongoing project

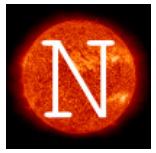
- Very closed to having the first version of the software ported to device
- Expect this to be optimised to improve performance and minimise data transfer
- Using the PGI compiler (in order to use OpenACC) has resulted in multiple compiler issues
  - tmp files not being correctly understood – no clear error message
  - Compiler failing on large files
  - Complex number functions in Fortran not currently compatible with OpenACC.
  - Deep data copy not handled

Next step: OpenACC+MPI implementation.



Reduced port of CASTEP

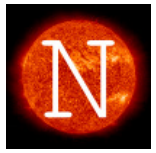
- Porting the full program to difficult
  - Unsupported features and compiler immaturity
  - Low-level changes affected too much code
  - Change approach to port contained functionality
- Particular feature (nlxc calculation)
- Work from bottom up rather than top down
  - Port lowest level kernels, then move data regions successively higher
  - Rather than porting the data structures then altering all associated code to work with those structures



http://www.nu-fuse.com

CASTEP Performance

	Time (s)	Speedup
1 process	6442.45	
2 processes	4368.15	1.47
4 processes	2183.26	2.95
8 processes	2147.57	2.99
16 processes	1489.48	4.32
32 processes	936.59	6.88
64 processes	741.37	8.69
1 GPU	1894.67	3.4



Summary

Porting CASTEP to GPGPUs using OpenACC and CUDA libraries

- Full program defeated us
  - Still porting a large amount of the core kernels but not having to update the whole program
- OpenACC has moved on and so have the compilers
  - Much better now, but still not trivial
- OpenACC is not OpenMP
  - Similar in the sense it is easy to get something to work but harder to get full performance
  - Hides much worse data operations
  - Bad OpenMP will scale a bit
  - Bad (not well structured) OpenACC will go much slower than serial
- How do you cope with modifications in the source code to enable GPU usage?