Enhancing scalability of the gyrokinetic code GS2 by using MPI Shared Memory for FFTs

Lucian Anton[1], Ferdinand van Wyk[2,4], Edmund Highcock[2],
Colin Roach[3], Joseph T. Parker[5]

[1]Cray UK, [2]University of Oxford,
[3]CCFE Culham, [4]STFC Daresbury, [5]STFC RAL

CRAY User Group 2016,
London, May 10-12 2016

Outlook

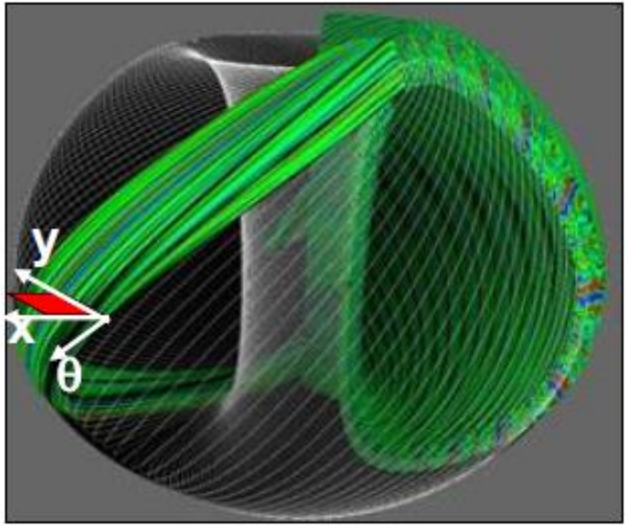
- Introduction to GS2
- Accelerating FFTs with shared memory in GS2
- Implementation details & scaling performance
- Conclusions

GS2 Introduction: code description

- GS2 is a parallel physics application, developed to study low-frequency turbulence in magnetized plasma
  - Large computation scale well to O(1000) MPI ranks
- Collaborative project with main contributors and users from: Culham Centre for Fusion Energy, Princeton Plasma Physics Laboratory, University of Maryland, University of Oxford, University of York, EPCC
- GS2 is written (mainly) in Fortran, the code source and documentation are available online
  - http://gyrokinetics.sourceforge.net/wiki

GS2 Introduction: basic physics

- Charged particle in strong magnetic field have a highly anisotropic motion:
  - The fast rotation around the magnetic line can be averaged out
  - The centre of gyration moves along the magnetic line
  - Drift motion across the field lines due to field geometry and electromagnetic turbulence



- The particle equilibrium distribution is often unstable to small scale instabilities
  - Full plasma state is described by perturbed distribution named **g**
  - **g** evulotion is described by the Gyrokinetic Equation (GKE)
  - Used to compute physical quantities: heat and particle fluxes, electromagnetics fields

GS2 Introduction: Gyrokinetics multiscale computational challenges

- GKE for ITER simulation at ion scale:
  - Must resolve the time for an ion to cross a turbulent eddy ~ 1 μs,
  - turbulence saturation time τ_{sat} ~10 ms
- Desirable simulations
  - Confinement time ~ 100 τ_{sat}
    - ~10^{12} grid points
    - ~170 Pflops.hours (~ a week on 10^5 cores)
  - Coupled electron–ion turbulence ~60^3 factor.
- Fusion is hard also in bits not only in atoms, but simplified approaches are useful
- Large, scalable FFTs are an essential requirement for these kind on computations

GS2 Introduction: data layouts

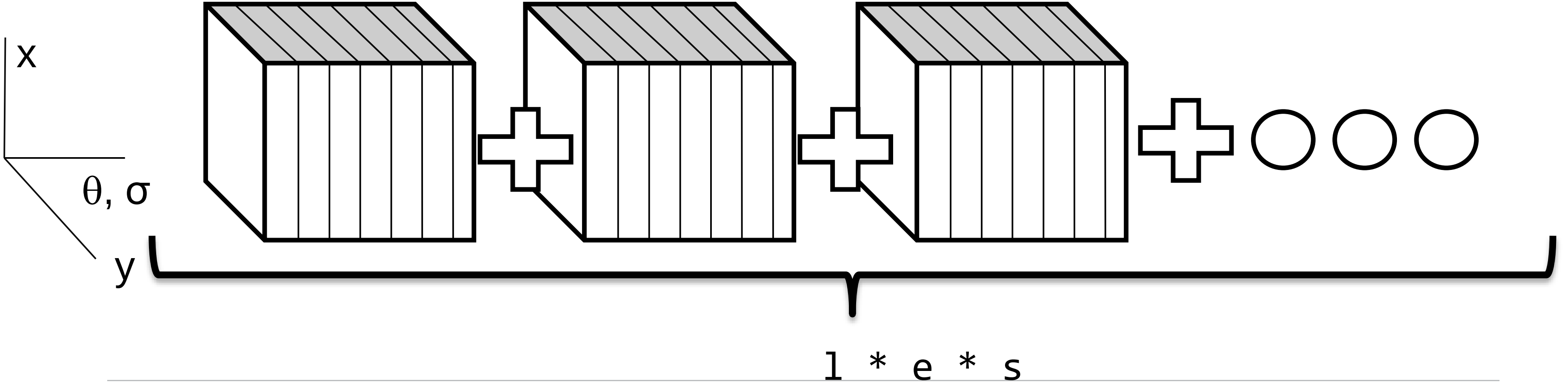
- **g(θ,σ,x,y,l,e,s)** is computed by solving numerically GKE
  - **θ** space coordinate along magnetic field
  - **σ** direction of motion along magnetic field (+/-1)
  - **x,y** space coordinates perpendicular to the magnetic field
  - **l,e** describe particle energy
  - **s** particle species
- Computation of GKE terms requires several discretized grid memory layouts with various subsets of variables ranges local to the MPI ranks
  - Time advance : g_lo(θ,σ::\`xyles\`)
  - FFT : xxf_lo(x::y,θ,σ,\`les\`) yxf_lo(y::x,θ,σ,\`les\`)
  - Collision layouts :…
- Costly redistributes between layouts at large counts of MPI ranks saturate parallel scaling because of the all to all communication pattern

GS2 Introduction: Fourier transforms

- In current computation GS2 needs to compute O(100) of 2D FT on O(100) x O(100)
  - (x,y) coordinates
  - uses dealiasing
- In general FT can be computed in GS2 in xxf_lo and yxf_lo
  - Poor scalability at high MPI rank counts
- In certain cases computation can be done in g_lo
  - MPI communication is avoided

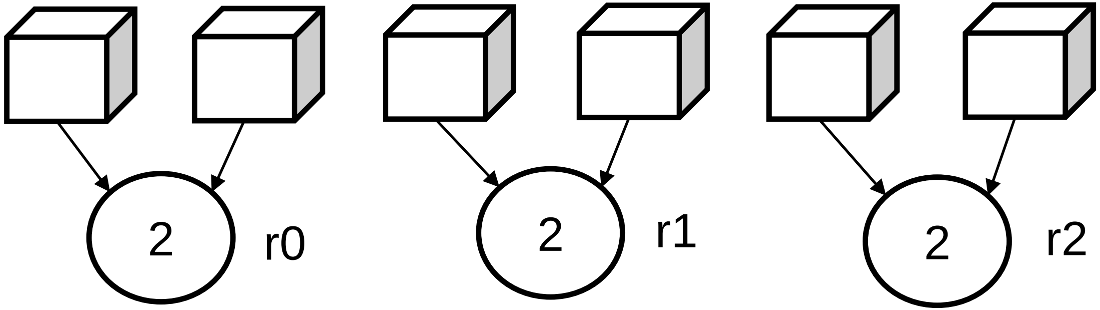
GS2 accelerated FFTs

- **g_lo (θ,σ::`yxles`)** layout can be thought as ‘`l*e*s`’ set of 3D subarrays
  - First dimension aggregates θ and σ
- FFTs are computed in the (x,y) plane of each block

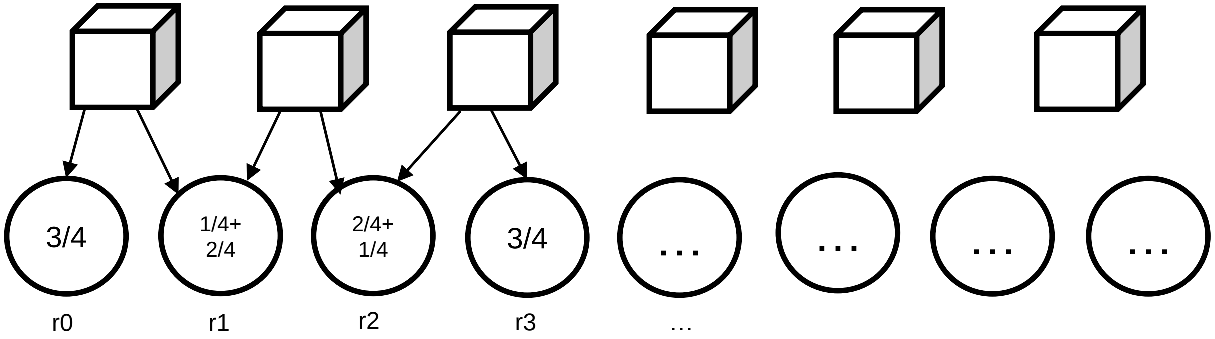


GS2 FFT parallel algorithm

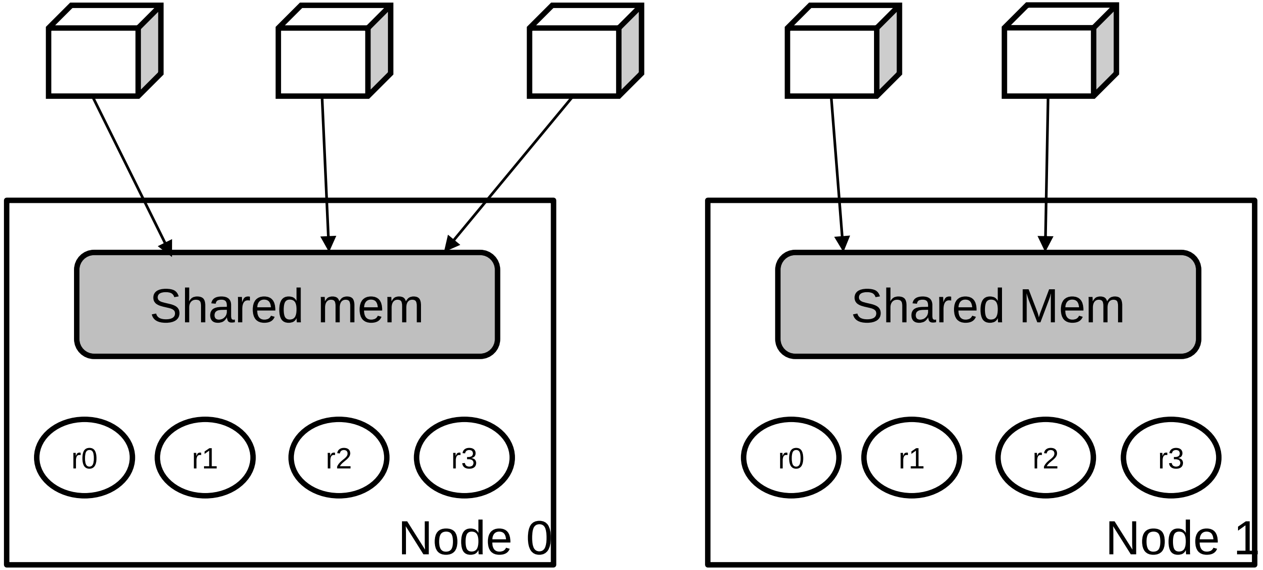
Accelerated FFTs:
les % Nproc =0
2D FFTW plan



Distributed FFTs:
Uses MPI comm
+ 2 1D FFTW
plans

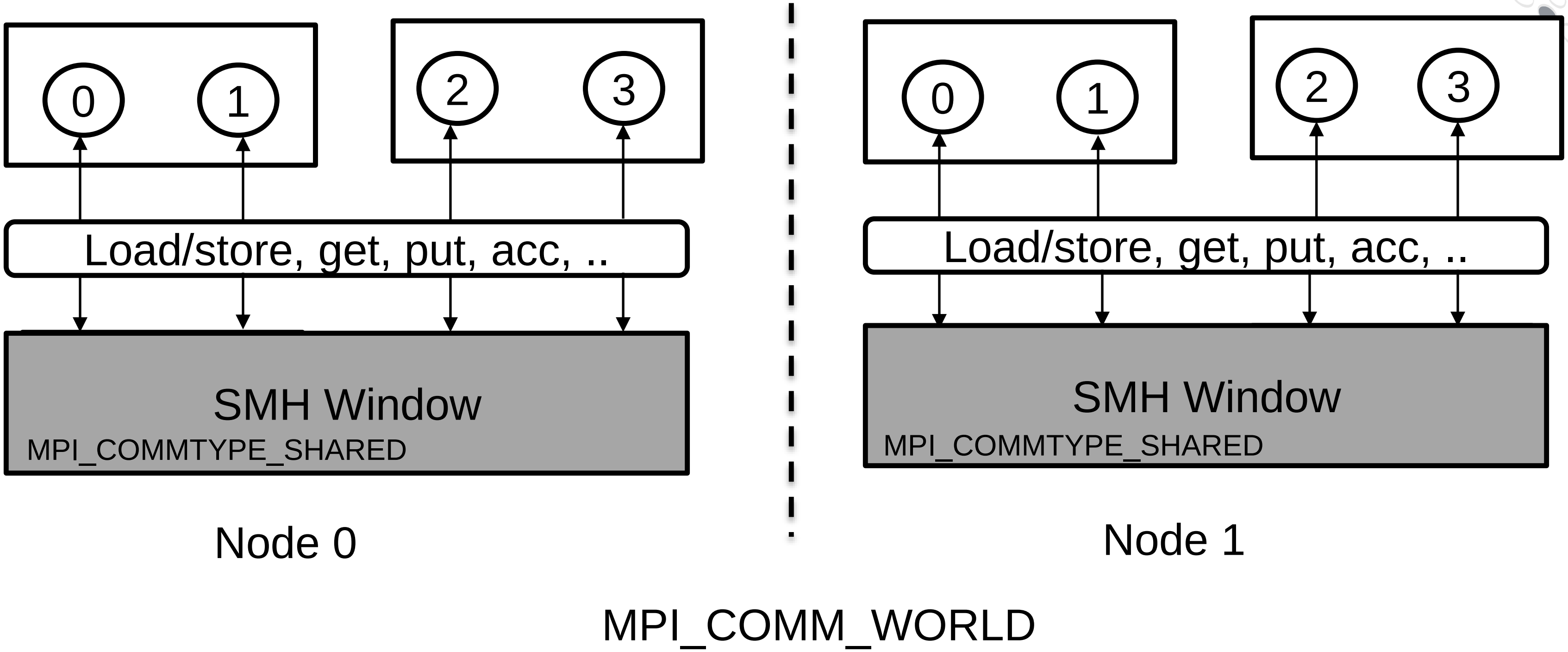


Accelerated FFTs with SHM



- We choose load imbalance to save MPI comm
- All ranks from a SHM-node work on one 'les' block at a time
- 2x1D FFTW plans found to be faster than 2D plan

Shared Memory in MPI 3



T. Hoefler *et al* " MPI + MPI: a new hybrid approach to parallel programming with MPI plus shared memory",*Journal of Computing*, 2013

Implementation detail: SHM module

- Fortran module for shared memory

```
type shm_info_t
    integer comm, wcomm, size, id
    integer, allocatable :: wranks(:)
 end type shm_info_t
  public :: shm_init, shm_alloc, shm_free, &
      shm_onnode, shm_node_id, shm_get_node_pointer, &
      shm_node_barrier, shm_clean, shm_fence
```

- I. J. Bush, *New Fortran Features: The Portable Use of Shared Memory Segments*, HPCx Consortium, Tech. Rep., 2007.

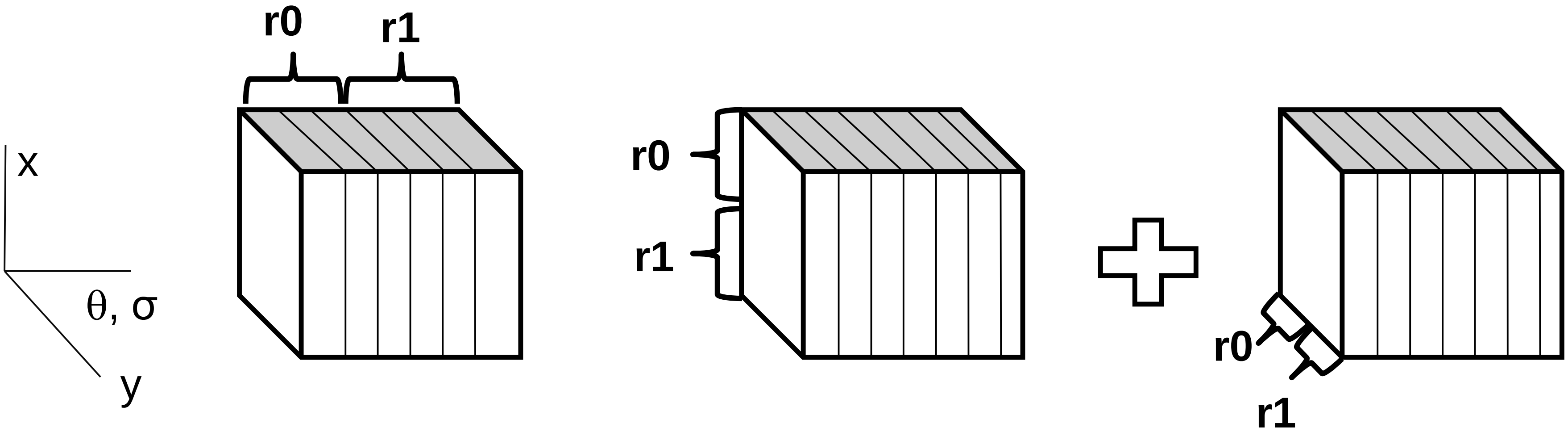
http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0701.pdf

Implementation details: GS2 code changes

- **GS2 data layout adaptation**
  - Insert SHM-node info in the layout descriptors
  - New FFTW plans and 2 new subroutines for shared computation of FFTW on FFT-block
  - Change a few allocatable arrays to pointers
- **Shared memory is compatible with OpenMP**
  - Not limited by the possible limited OpenMP scaling of other numerical kernels

Implementation details: 2D vs 1D FFTW plans(I)

- FFTs are to be computed for every (x,y) plane of each block



Implementation details: 2D vs 1D FFTW plans (II)

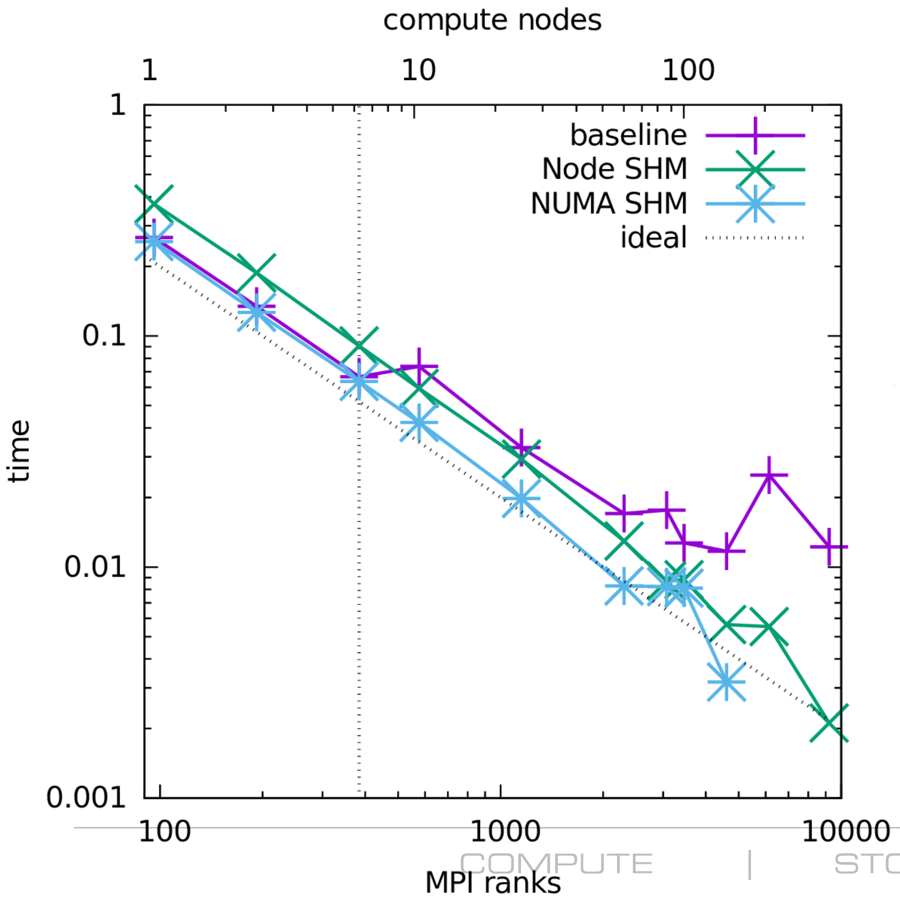
```
if (use_2d_fftw_plan) then
     call fftw_execute_dft(one_node_plan2d,a(1+team%nt_shift:1+team%nt_shift,1,1,1), &
          b(1+team%nt_shift:1+team%nt_shift,1,1,1))
   else
     do j = shm_info%id+1, gy, shm_info%size
       call fftw_execute_dft(planfx1, a(:,:,j,1), b(:,:,j,1))
     enddo

     call shm_flush(b(1,1,1,1))

     do i = shm_info%id+1, nx, shm_info%size
       buffy1nd(:,:) = b(:,i,:,1)
       call fftw_execute_dft(planfy1, buffy1nd,buffy1nd)
       b(:,i,:,1) = buffy1nd(:,:)
     enddo
…….
```

Implementation details: 2D vs 1D FFTW plans (III)

```
FFTW / fftw_execute_dft
------------------------------------------------------------------
 Time                                   0.500982 secs
  Imb. Time                                0.095712
sec
TLB utilization                  25.34 refs/miss
0.05 avg uses
  D1 cache hit,miss ratios          80.0% hits
20.0% misses
  D1 cache utilization (misses)       5.01 refs/miss
0.63 avg hits
  D2 cache hit,miss ratio          54.3% hits
45.7% misses
  D1+D2 cache hit,miss ratio          90.9% hits
9.1% misses
  D1+D2 cache utilization          10.95 refs/miss
1.37 avg hits
```

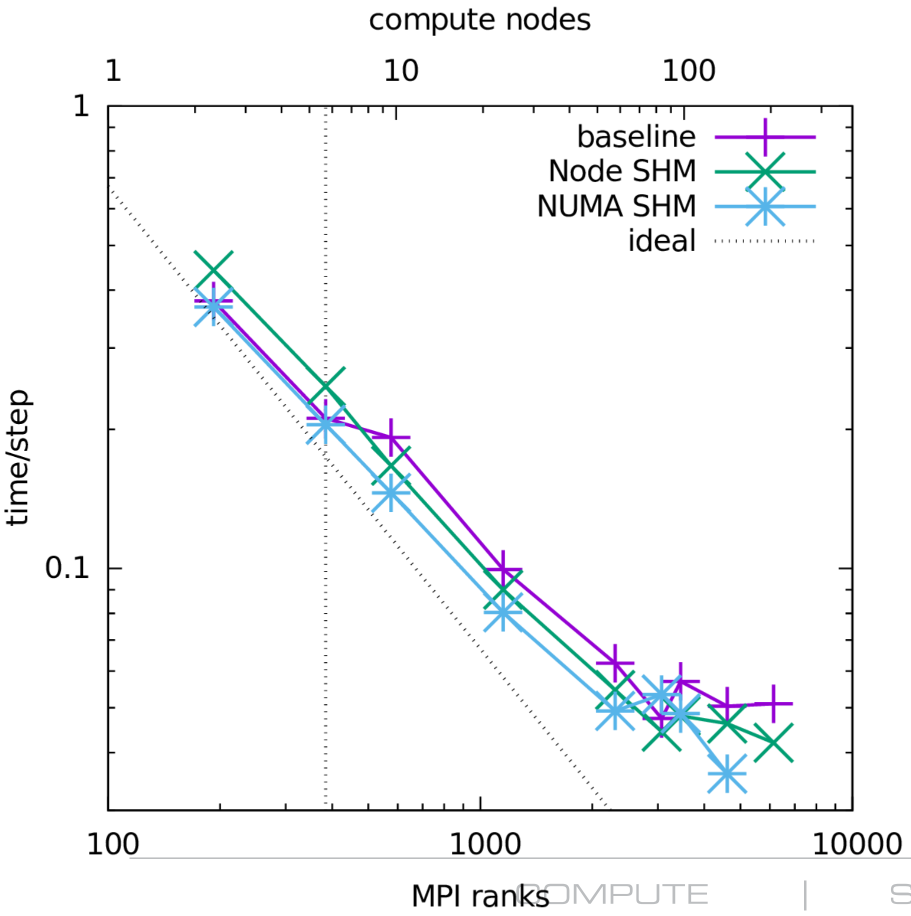
```
FFTW / fftw_execute_dft
----------------------------------------------------------------------
Time                                    0.277788 secs
  Imb. Time                                 0.021013
secs
TLB utilization                  1,115.55 refs/miss
2.18 avg uses
  D1 cache hit,miss ratios          91.3% hits
8.7% misses
  D1 cache utilization (misses)      11.54 refs/miss
1.44 avg hits
  D2 cache hit,miss ratio          45.4% hits
54.6% misses
  D1+D2 cache hit,miss ratio          95.3% hits
4.7% misses
  D1+D2 cache utilization          21.13 refs/miss
2.64 avg hits
```

Scaling performance: FFT benchmark



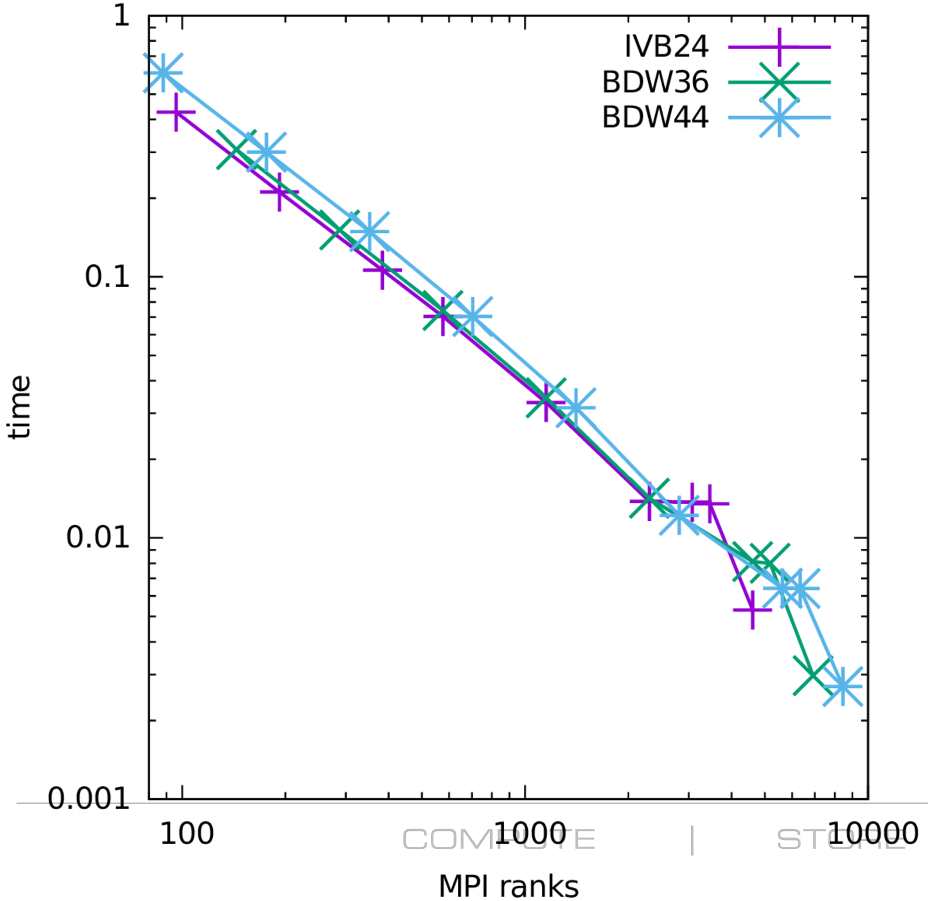
- 384 `les` blocks 106x128x128 FFT block
- Use SHM segments at node level and socket level
- Runs done on ARCHER
  - Intel compiler 15.0.0.163
  - FFTW 3.3.4.5
- **Perfect scaling for small load imbalance**
- **SHM extends the good scaling regime increased ~ 2 times**

Scaling performance: GS2 Collisonless run



- **Scaling saturates because of the linear term**
    - Load imbalance for NUMA-SHM takes place in the same range

Scaling performance: Broadwell vs Ivy Bridge



- **Showing only NUMA-SHM**
- **BDW:**
  - 36 cores/node, 2.1 GHz,
  - 44 cores/node, 2.2 GHz
- **IVB:**
  - 24 cores/node, 2.7 GHz
- **Performance per core is similar for all chips**

Conclusions

- Accelerated FFTs+SHM scale ~ 10,000 MPI ranks on ARCHER (scaling range more than doubled + speed up)
- GS2 collisionless computation speeds up ~ 25% at large core counts over a good range of MPI ranks
  - Load imbalance affects the scaling
    - More elaborate algorithms are worth exploring
- Minimal intrusion in the original source code
    - Compatible with OpenMP
- Scaling is preserved for the larger Broadwell nodes

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, and URIKA. The following are trademarks of Cray Inc.: APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, REVEAL,THREADSTORM. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.

Acknowledgments

- This work was partially supported by:
  - the Plasma HEC Consortium [EPSRC grant number EP/L000237/1],
  - CCP Plasma[EPSRC grant number EP/M022463/1],
  - the UK Engineering and Physical Sciences Research Council (EPSRC) through the Software Outlook Programme.
  - RCUK Energy Programme [grant number EP/I501045].
- This work used the ARCHER UK National Supercomputing Service (http://www.archer.ac.uk).
- Cray

Q&A

Lucian Anton lanton@cray.com

Colin Roach colin.roach@ukaea.uk