RUNNING CP2K IN PARALLEL ON ARCHER

Iain Bethune (ibethune@epcc.ed.ac.uk)

Overview

- Introduction to ARCHER
  - Parallel Programming models
- CP2K Algorithms and Data Structures
- Running CP2K on ARCHER
- Parallel Performance
- CP2K Timing Report

Introduction to ARCHER

- UK National Supercomputing Service
- Cray XC30 Hardware
  - Nodes based on 2×Intel Ivy Bridge 12-core processors
  - 64GB (or 128GB) memory per node
  - 3008 nodes in total (72162 cores)
  - Linked by Cray Aries interconnect (dragonfly topology)
- Cray Application Development Environment
  - Cray, Intel, GNU Compilers
  - Cray Parallel Libraries (MPI, SHMEM, PGAS)
  - DDT Debugger, Cray Performance Analysis Tools

Introduction to ARCHER

- EPSRC
  - Managing partner on behalf of RCUK
- Cray
  - Hardware provider
- EPCC
  - Service Provision (SP) – Systems, Helpdesk, Administration, Overall Management (also input from STFC Daresbury Laboratory)
  - Computational Science and Engineering (CSE) – In-depth support, training, embedded CSE (eCSE) funding calls
  - Hosting of hardware – datacentre, infrastructure, etc.

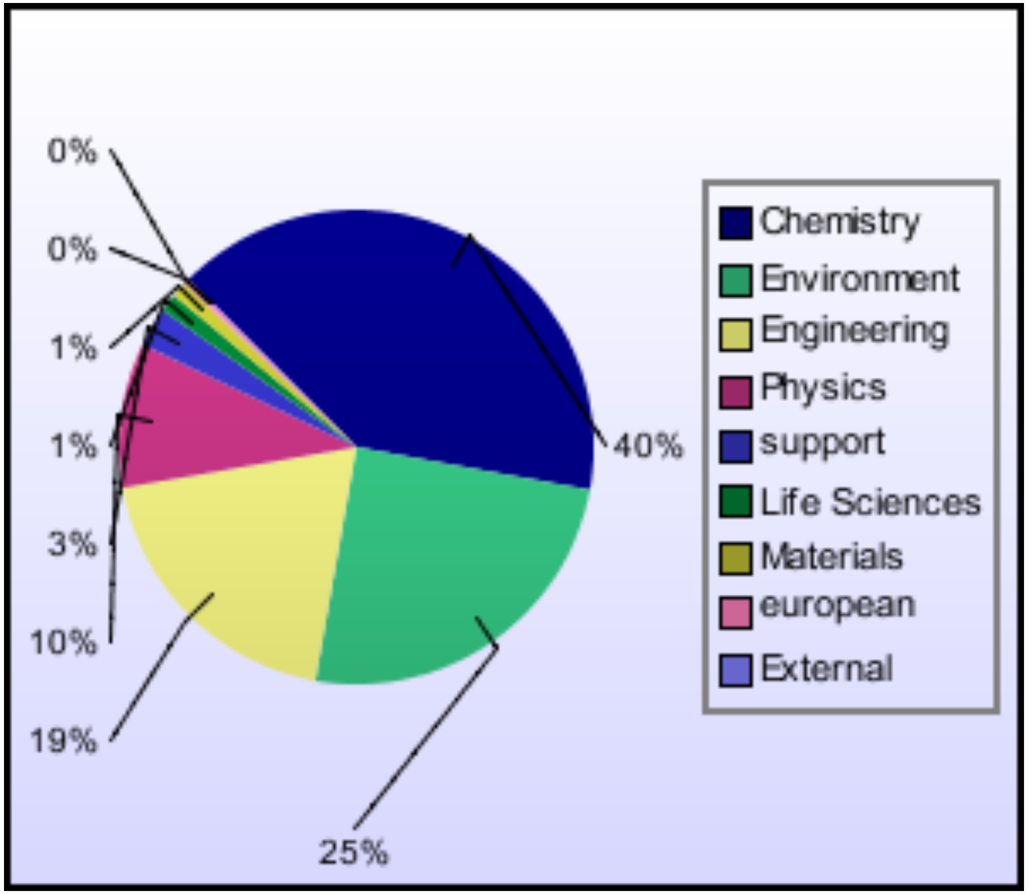
Introduction to ARCHER

Slater (NSCCS)	ARCHER
Intel Ivy Bridge 2.6GHz	Intel Ivy Bridge 2.7GHz
8-core CPU	24 cores per node (2×12-core NUMA)
4 TB total memory (8 GB/core)	64GB per node (2.66 GB/core) or 128GB per node (5.33 GB/core)
64 CPUs (512 cores)	3008 nodes (72,192 cores)
NUMAlink network	Cray Aries / Dragonfly
	2 Post-processing nodes: 48 core SandyBridge 1TB Memory

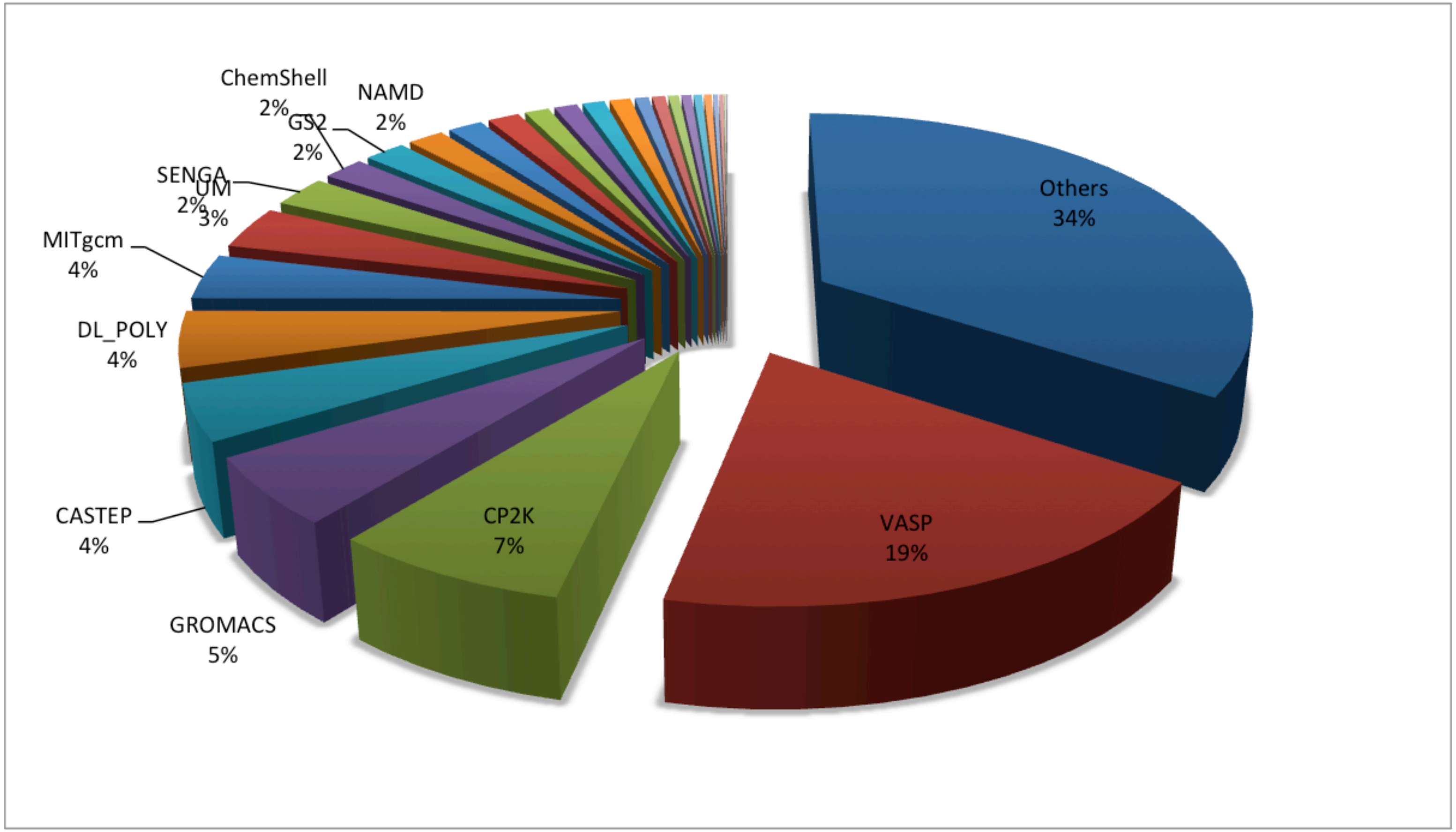
Introduction to ARCHER

- /home – NFS, not accessible on compute nodes
  - For source code and critical files
  - Backed up
  - > 200 TB total
- /work – Lustre, accessible on all nodes
  - High-performance parallel filesystem
  - Not backed-up
  - > 4PB total
- RDF – GPFS, not accessible on compute nodes
  - Long term data storage

Introduction to ARCHER



Introduction to ARCHER



epcc

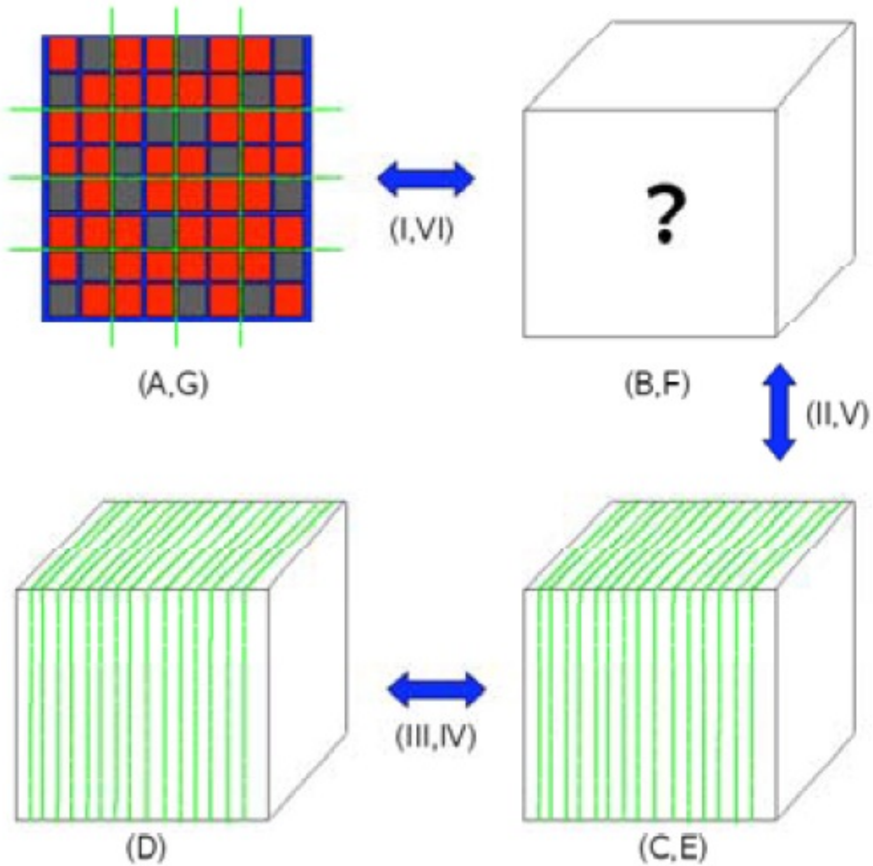
Introduction to ARCHER: Parallel Programming Models

- MPI
  - Message Passing Interface (www.mpi-forum.org)
  - Library supplied by Cray (or OpenMPI, MPICH …)
  - Distributed Memory model
  - Explicit message passing
  - Can scale to 100,000s of cores
- OpenMP
  - Open Multi-Processing (www.openmp.org)
  - Code directives and runtime library provided by compiler
  - Shared Memory model
  - Communication via shared data
  - Scales up to size of node (24 cores)

CP2K Algorithms and Data Structures

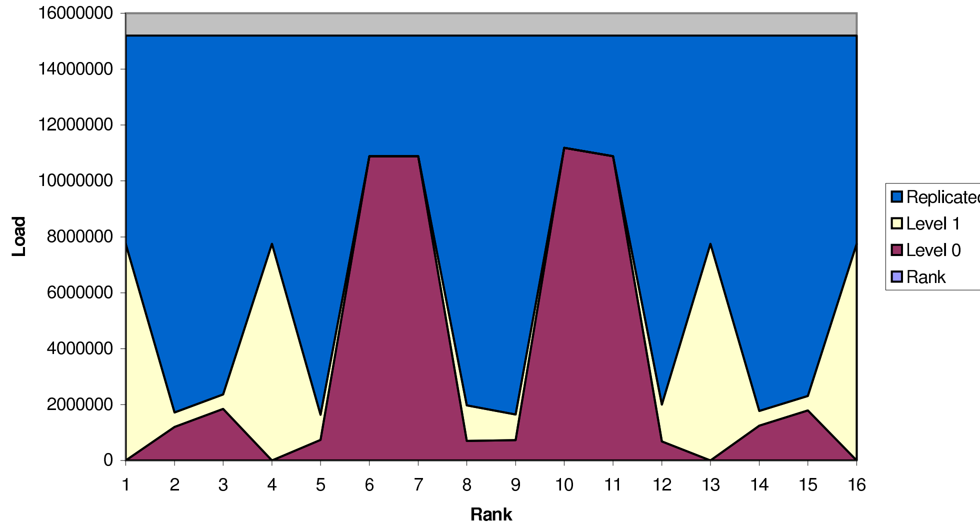
- (A,G) – distributed matrices
- (B,F) – realspace multigrids
- (C,E) – realspace data on planewave multigrids
- (D) – planewave grids

- (I,VI) – integration/ collocation of gaussian products
- (II,V) – realspace-to-planewave transfer
- (III,IV) – FFTs (planewave transfer)

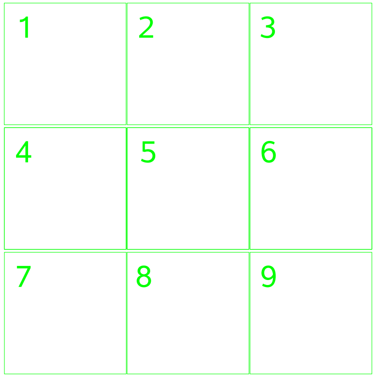


CP2K Algorithms and Data Structures

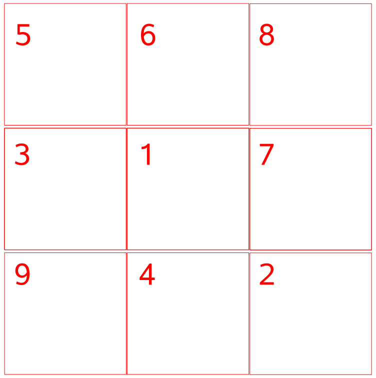
- Distributed realspace grids
  - Overcome memory bottleneck
  - Reduce communication costs
  - Parallel load balancing
    - On a single grid level
    - Re-ordering multiple grid levels
    - Finely balance with replicated tasks



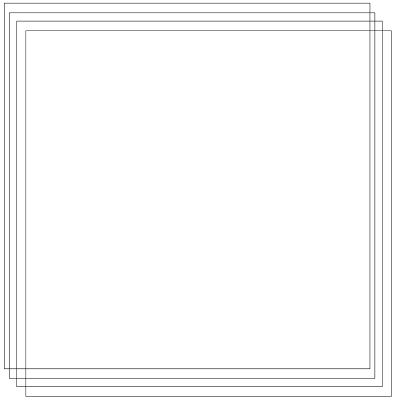
Level 1, fine grid, distributed



Level 2, medium grid, dist



Level 3, coarse grid, replicated



CP2K Algorithms and Data Structures

- Fast Fourier Transforms
  - 1D or 2D decomposition
  - FFTW3 and CuFFT library interface
  - Cache and re-use data
    - FFTW plans, cartesian communicators
- DBCSR
  - Distributed MM based on Cannon's Algorithm
  - Local multiplication recursive, cache oblivious
  - `libsmm` for small block multiplications

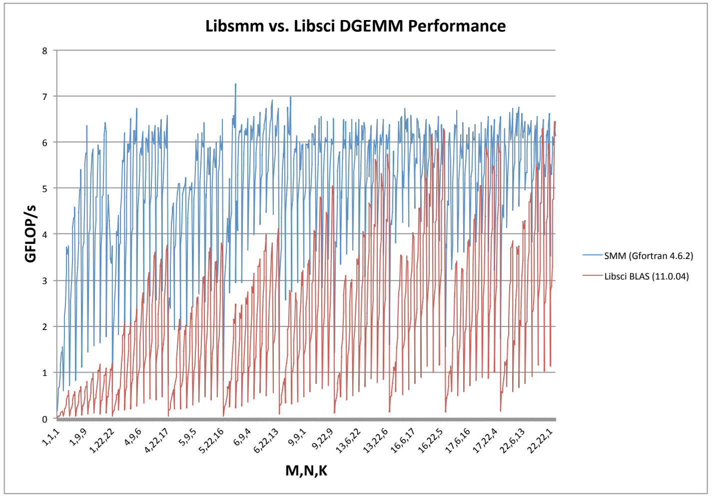
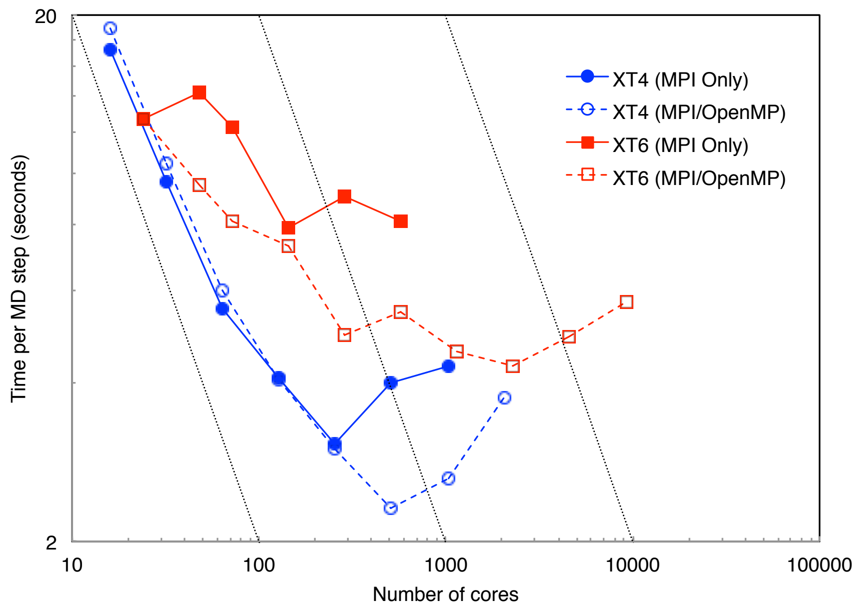


Figure 5: Comparing performance of SMM and Libsci BLAS for block sizes up to 22,22,22

CP2K Algorithms and Data Structures

- OpenMP
  - Now in all key areas of CP2K
  - FFT, DBCSR, Collocate/ Integrate, Buffer Packing
  - Incremental addition over time



- Dense Linear Algebra
  - Matrix operations during SCF
  - GEMM - ScaLAPACK
  - SYEVD – ScaLAPACK / ELPA

Running CP2K on ARCHER

- Full details in the instruction sheet

- Access via (shared) login nodes
- CP2K is installed as a ‘module’

```
~> module load cp2k
```

- Do not run time-consuming jobs on the login nodes

```
~> $CP2K/cp2k.sopt H2O-32.inp
```

✗

```
~> $CP2K/cp2k.sopt --check H2O-32.inp
```

✓

Running CP2K on ARCHER

- To run in parallel on the compute nodes…

- Create a PBS Batch script:
  - Request some nodes (24 cores each) `#PBS -l select=1`
  - For a fixed amount of time `#PBS -l walltime=0:20:0`

- Launch CP2K in parallel:

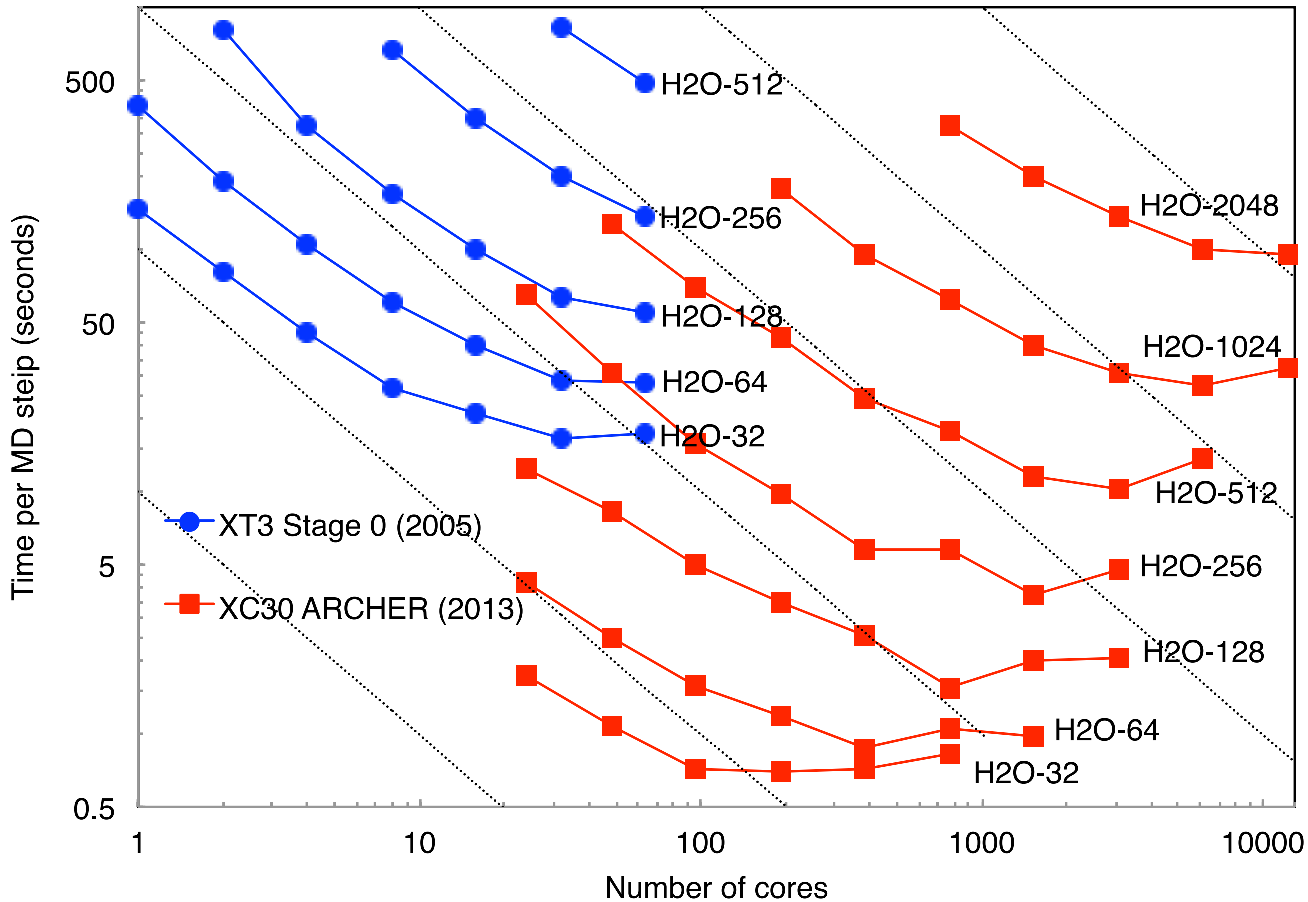
```
module load cp2k

aprun -n 24 $CP2K/cp2k.popt H2O-32.inp
```

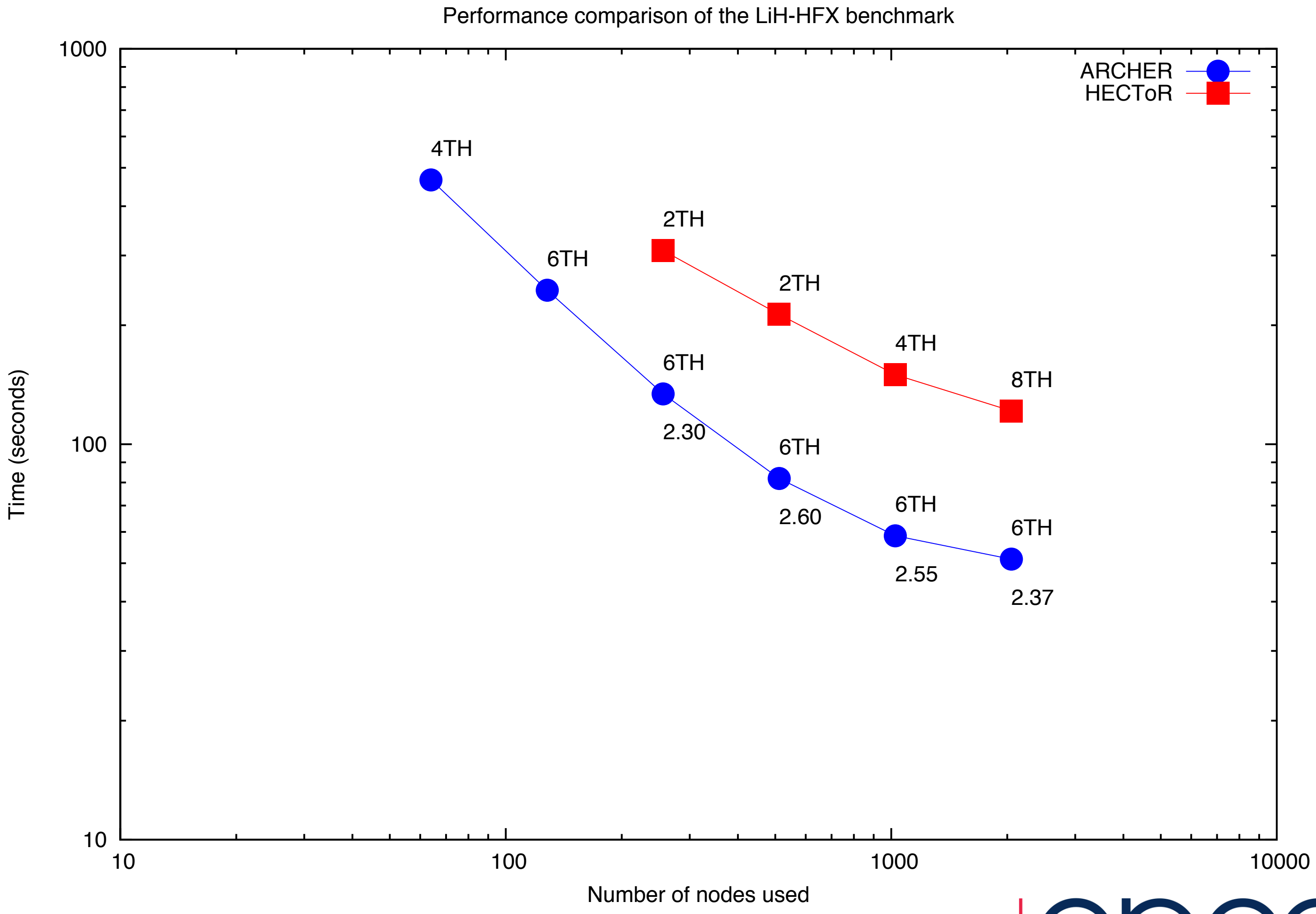
Parallel Performance

- Different ways of comparing time-to-solution and compute resource…
- Speedup: $S = T_{ref} / T_{par}$
- Efficiency: $E_p = S_p / p$,   good scaling is $E > 0.7$
- If $E < 1$, then using more processors uses more compute time (AUs)
- Compromise between overall speed of calculation and efficient use of budget
  - Depends if you have one large or many smaller calculations

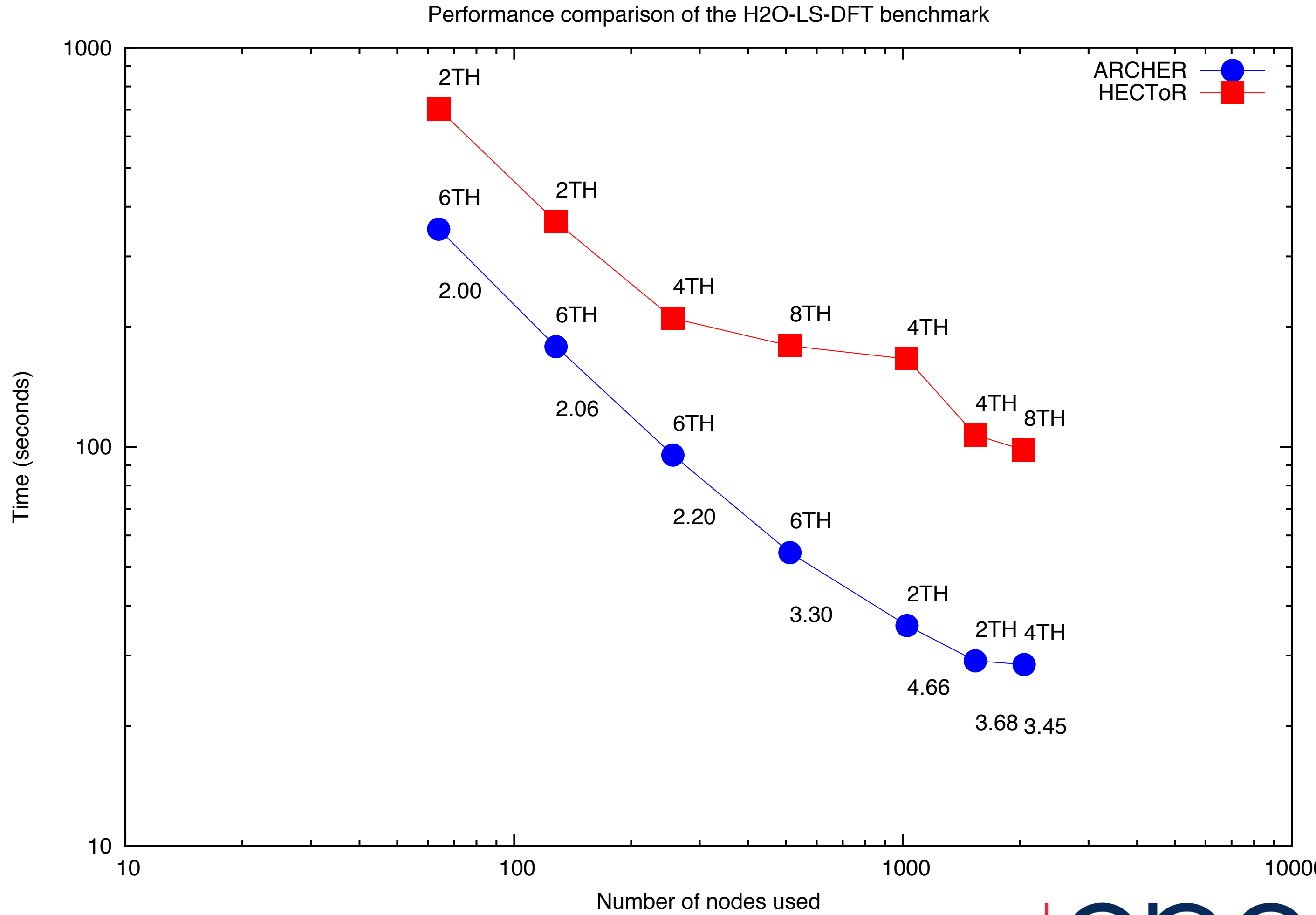
Parallel Performance : H2O-xx



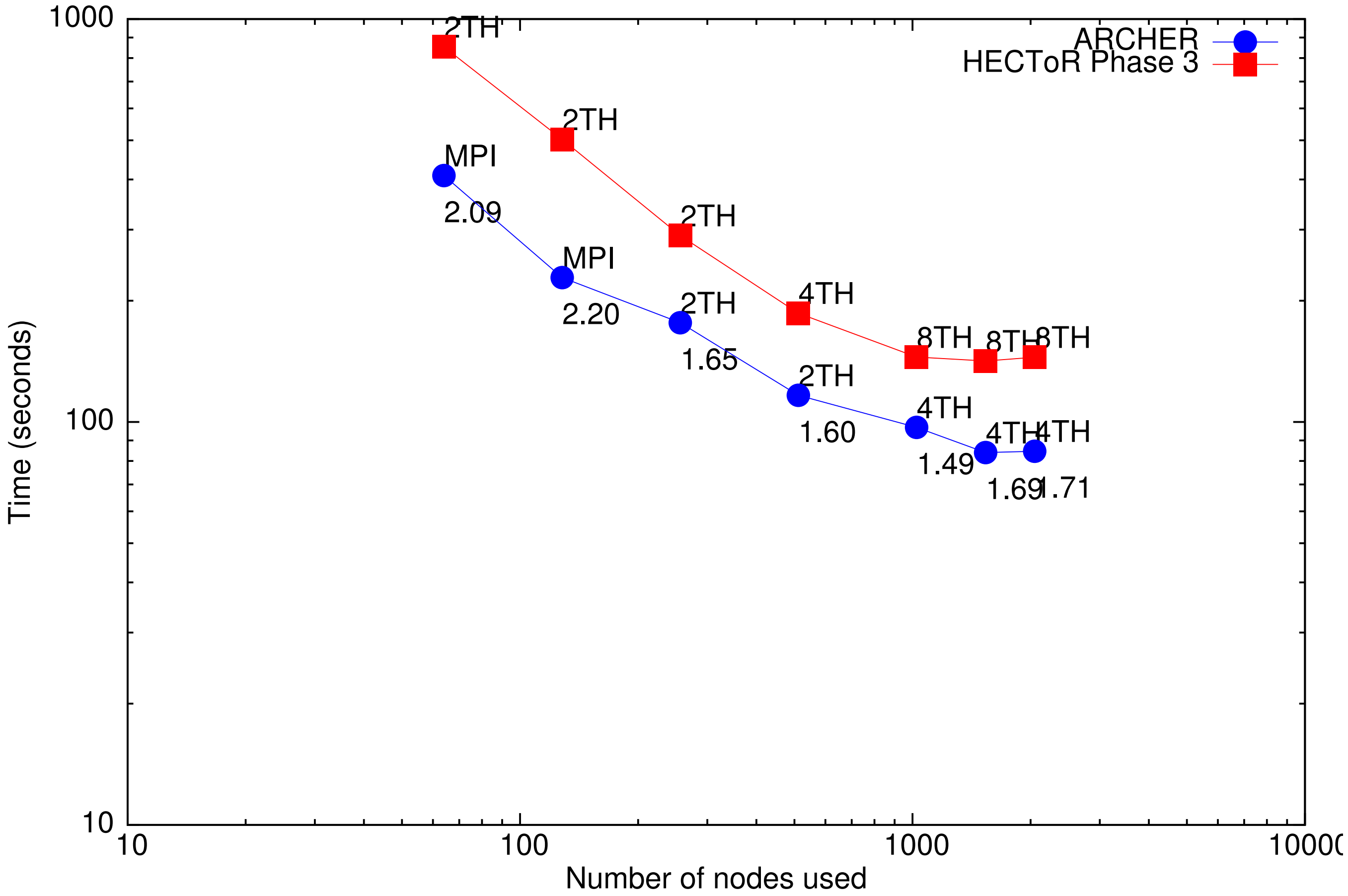
Parallel Performance: LiH-HFX



Parallel Performance: H2O-LS-DFT



Parallel Performance: H2O-64-RI-MP2



CP2K Timing Report

- CP2K measures are reports time spent in routines and communication
  - timing reports are printed at the end of the run

```
-------------------------------------------------------------------------------
-                                                                             -
-                          MESSAGE PASSING PERFORMANCE                        -
-                                                                             -
-------------------------------------------------------------------------------

ROUTINE                 CALLS  TOT TIME [s]  AVE VOLUME [Bytes]  PERFORMANCE [MB/s]
MP_Group                    4         0.000
MP_Bcast                  186         0.018             958318.             9942.82
MP_Allreduce             1418         0.619               2239.                5.13
MP_Gather                  44         0.321              21504.                2.95
MP_Sync                  1372         0.472
MP_Alltoall              1961         5.334          323681322.           119008.54
MP_ISendRecv           337480         0.177               1552.             2953.86
MP_Wait                352330         5.593
MP_comm_split              48         0.054
MP_ISend                39600         0.179              14199.             3147.38
MP_IRecv                39600         0.100              14199.             5638.21
-------------------------------------------------------------------------------
```

CP2K Timing Report

```
-------------------------------------------------------------------------------
-                                                                             -
-                                T I M I N G                                  -
-                                                                             -
-------------------------------------------------------------------------------
SUBROUTINE                               CALLS  ASD         SELF TIME        TOTAL TIME
                                       MAXIMUM       AVERAGE  MAXIMUM  AVERAGE  MAXIMUM
CP2K                                         1  1.0    0.018    0.018   57.900   57.900
qs_mol_dyn_low                               1  2.0    0.007    0.008   57.725   57.737
qs_forces                                   11  3.9    0.262    0.278   57.492   57.493
qs_energies_scf                             11  4.9    0.005    0.006   55.828   55.836
scf_env_do_scf                              11  5.9    0.000    0.001   51.007   51.019
scf_env_do_scf_inner_loop                   99  6.5    0.003    0.007   43.388   43.389
velocity_verlet                             10  3.0    0.001    0.001   32.954   32.955
qs_scf_loop_do_ot                           99  7.5    0.000    0.000   29.807   29.918
ot_scf_mini                                 99  8.5    0.003    0.004   28.538   28.627
cp_dbcsr_multiply_d                       2338 11.6    0.005    0.006   25.588   25.936
dbcsr_mm_cannon_multiply                  2338 13.6    2.794    3.975   25.458   25.809
cannon_multiply_low                       2338 14.6    3.845    4.349   14.697   15.980
ot_mini                                     99  9.5    0.003    0.004   15.701   15.942
-------------------------------------------------------------------------------
```

CP2K Timing Report

- Not just for developers!
  - Check that communication is < 50% of total runtime
  - Check where most time is being spent:
    - Sparse matrix multiplication - `cp_dbcsr_multiply_d`
    - Dense matrix algebra – `cp_fm_syevd`, `cp_fm_cholesky_*`, `cp_fm_gemm`
    - FFT – `fft3d_*`
    - Collocate / integrate – `calculate_rho_elec`, `integrate_v_rspace`
- Control level of granularity

```
&GLOBAL
  &TIMINGS
    THRESHOLD 0.00001
  &END TIMINGS
&END GLOBAL
```

Default is 0.02 (2%)

After lunch: try it out for yourself in the computer lab…

Any questions?