Science & Technology Facilities Council

Configuring and Optimizing the Weather Research and Forecast Model on the Cray XT

Andrew Porter and Mike Ashworth
Computational Science and Engineering Department
STFC Daresbury Laboratory

andrew.porter@stfc.ac.uk

24th May 2010
Cray User Group, Edinburgh

Overview

- Introduction
- Machines
- Benchmark Configuration
- Choice of Compiler/Flags
- MPI Versus Mixed Mode (MPI/OpenMP)
- Memory Bandwidth Issues
- Tuning Cache Usage
- Input/Output
  - Default scheme
  - pNetCDF
  - I/O servers & process placement

Introduction - WRF

- Regional- to global-scale model for research and operational weather-forecast systems
- Developed through a collaboration between various US bodies (NCAR, NOAA...)
- Finite difference scheme + physics parametrisations
- F90 [+ MPI] [+ OpenMP]
- 6000 registered users (June 2008)

Science & Technology Facilities Council

Introduction – this work

- WRF accounts for significant fraction of usage of UK national facility (HECToR)
- Aim here is to investigate ways of ensuring this use is efficient
- Mainly through (the many) configuration options
- Code optimization when/if required

Machines Used

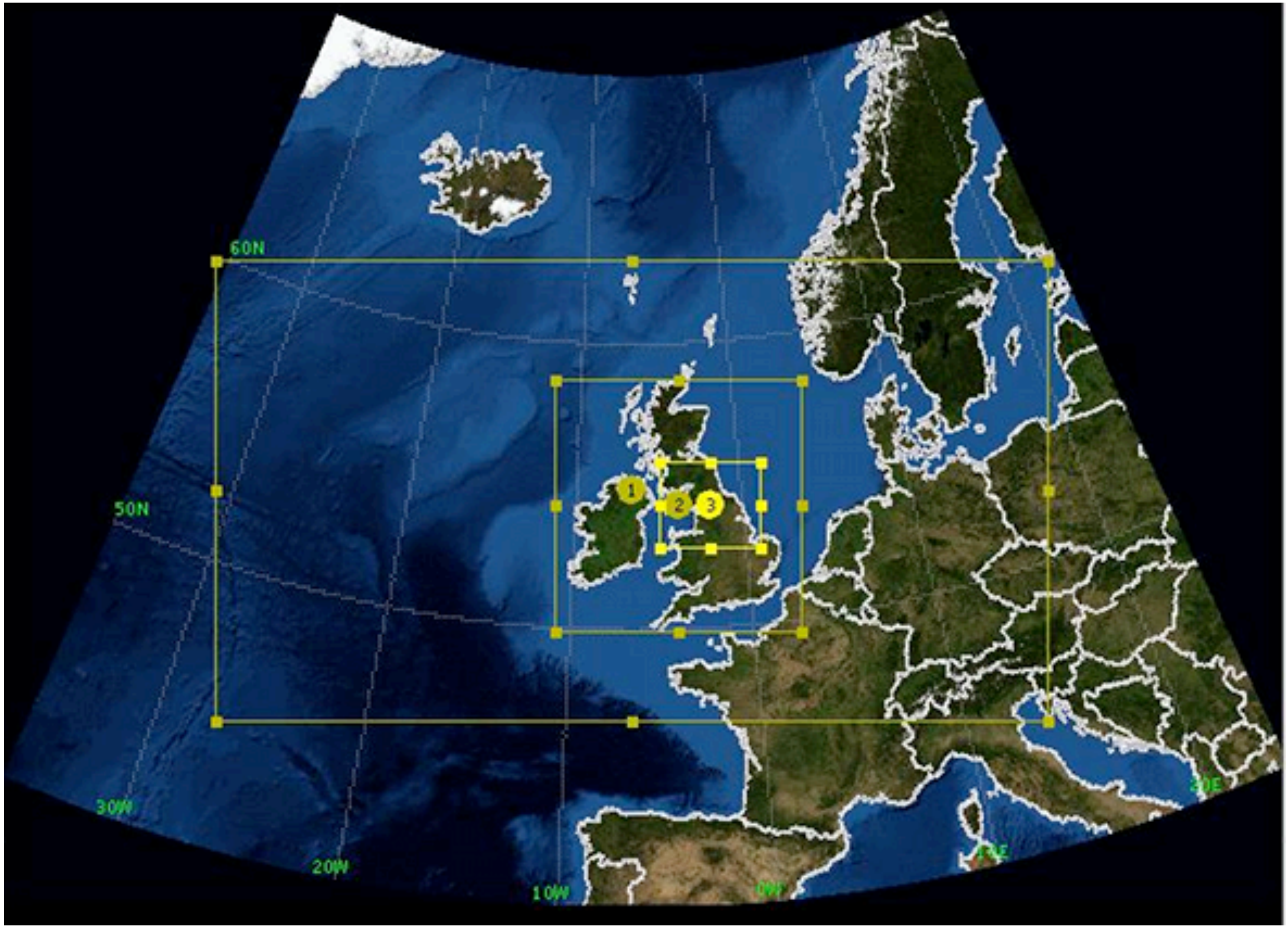
- HECToR – UK national academic supercomputing service
  - Cray XT4
  - 1x AMD Barcelona 2.3GHz quad-core chip per compute node
  - SeaStar2 interconnect
- Monte Rosa – Swiss National Supercomputing Service (CSCS)
  - Cray XT5
  - 2x AMD Istanbul 2.4GHz hexa-core chips per compute node
  - SeaStar2 interconnect

Benchmark Configuration
"Great North Run"

Three nested domains with two-way feedback between them:

D1 = 356 x 196
D2 = 319 x 322
D3 = 391 x 328

D3 gives 1Km-resolution data over Northern England.



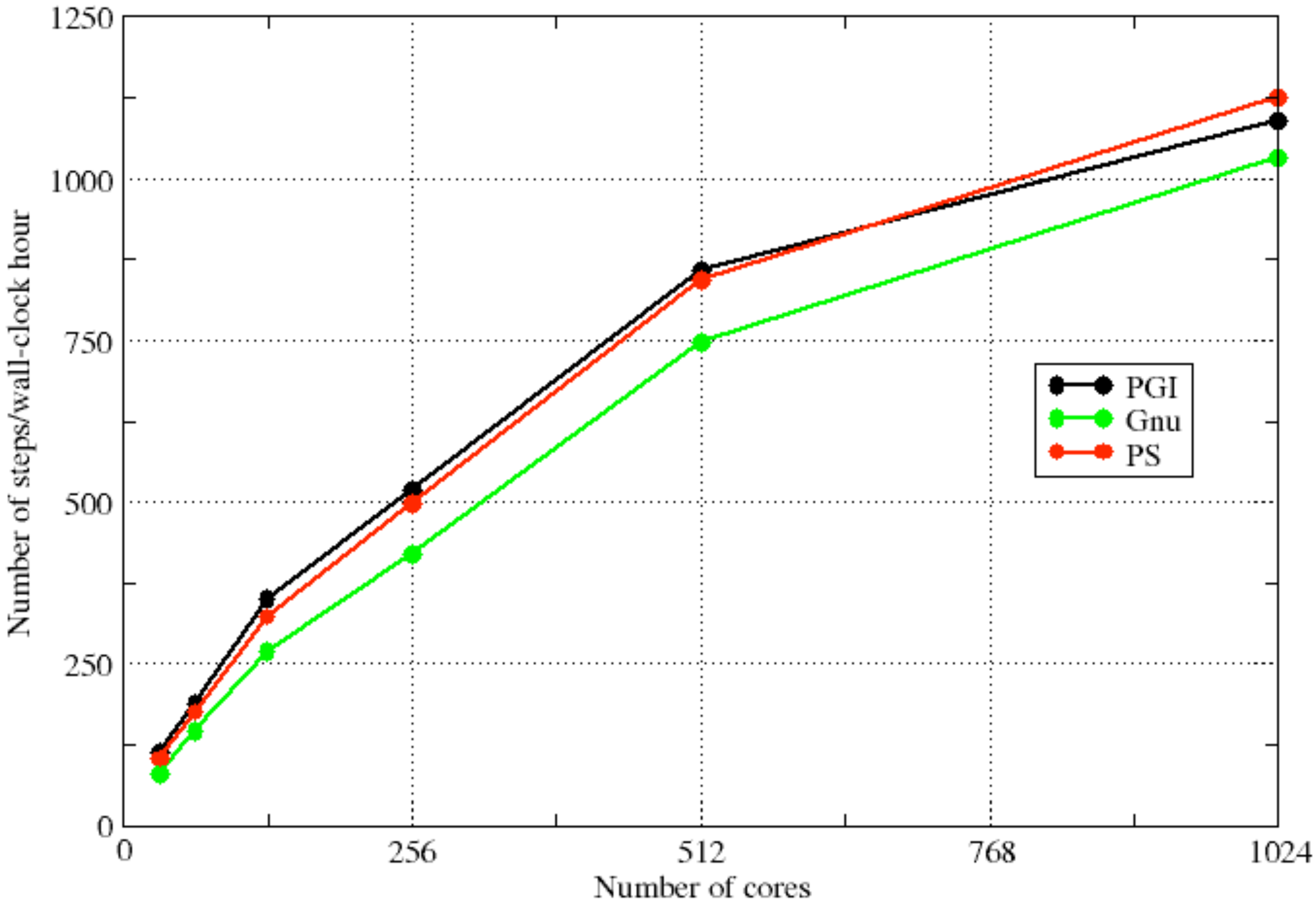
Science & Technology Facilities Council

Choice of Compiler/Flags

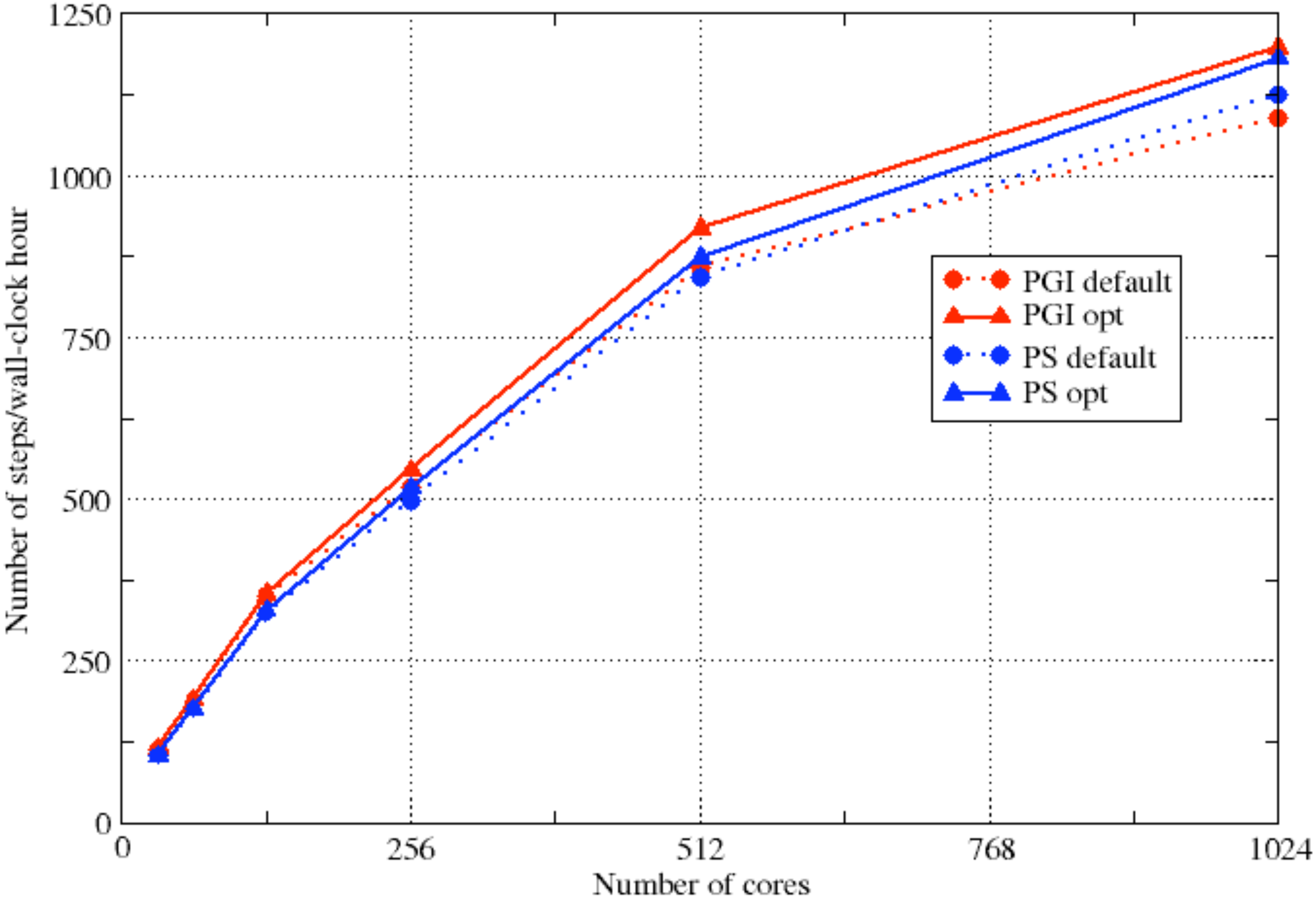
- HECToR offers four different compilers!
  - Portland Group (PGI)
  - Pathscale (recently bought by Cray)
  - Cray
  - Gnu (gcc + gfortran)
- WRF can be built in serial, shared-memory (sm), distributed-memory (dm) and mixed (dm+sm) modes...

Science & Technology Facilities Council

Initial Compiler Comparison for dm (MPI) build



Effect of Extra Flags

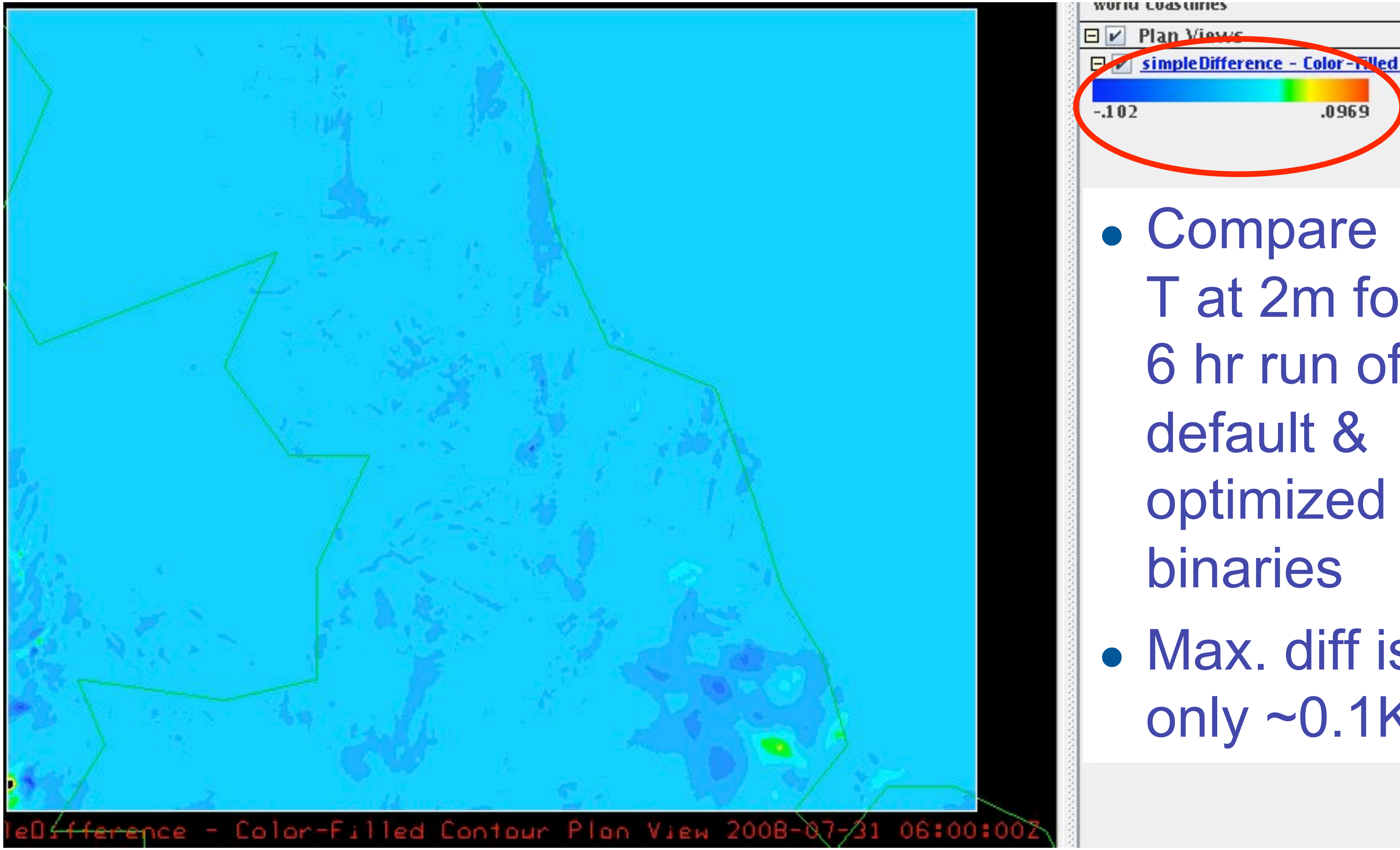


Science & Technology Facilities Council

Compiler notes I

- 1.1K -> 1.2K time-steps/wall-clock hour on 1024 cores from increasing optimization with PGI
  - -O3 –fast to –O3 –fastsse –Mvect=noaltcode –Msmartalloc –Mprefetch=distance:8 -Mfprel
- 1.2K -> 1.3K by re-building to remove array init'n prior to each inter-domain feedback stage
- PS with extra optimization flags only very slightly slower than PGI
- Gnu (default) is 25% slower than PGI (default) on 256 cores but only 10% slower on 1024
- Deficit much larger when extra optimization turned on for PGI

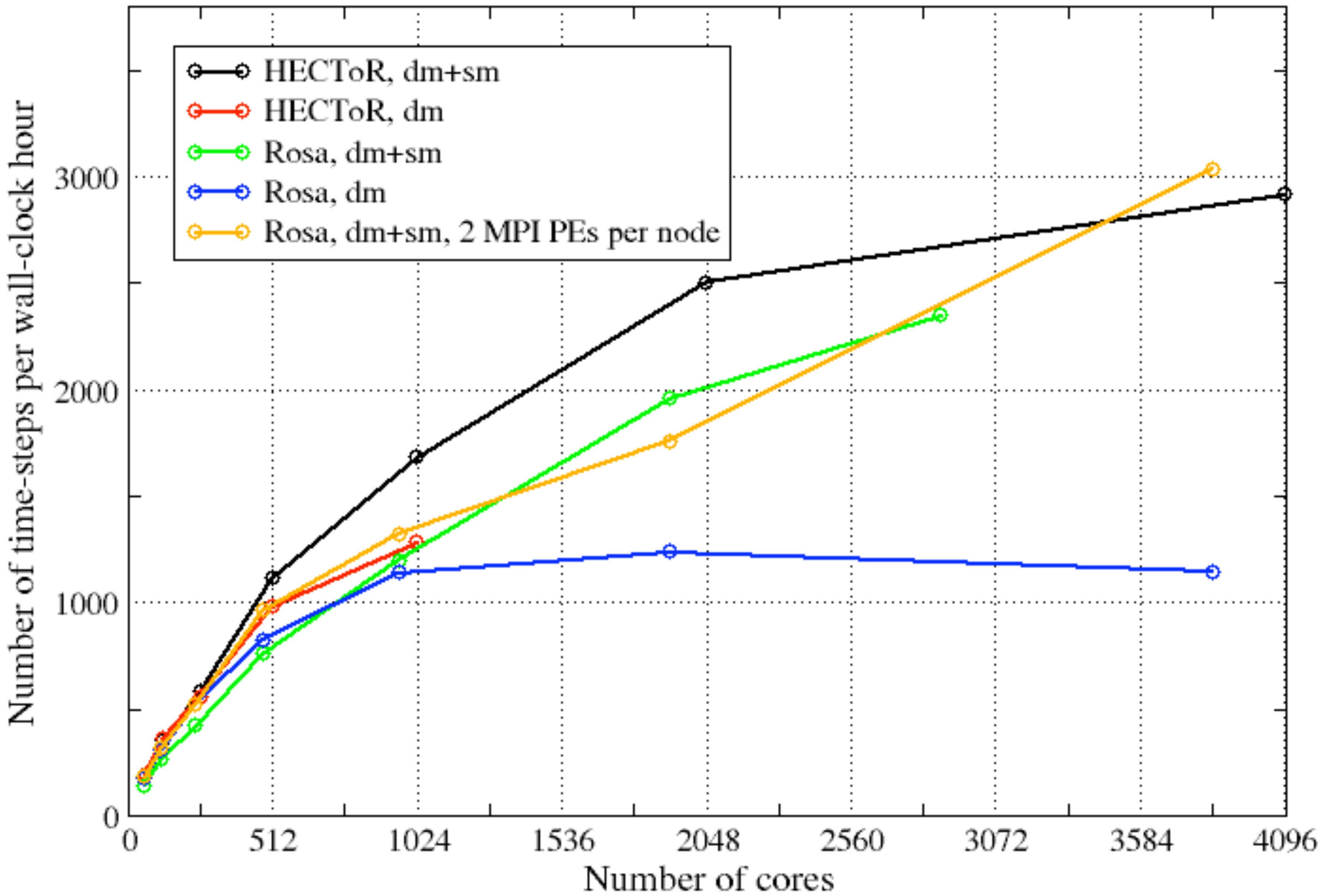
Verification of Results



- Compare T at 2m for 6 hr run of default & optimized binaries
- Max. diff is only ~0.1K

Science & Technology Facilities Council

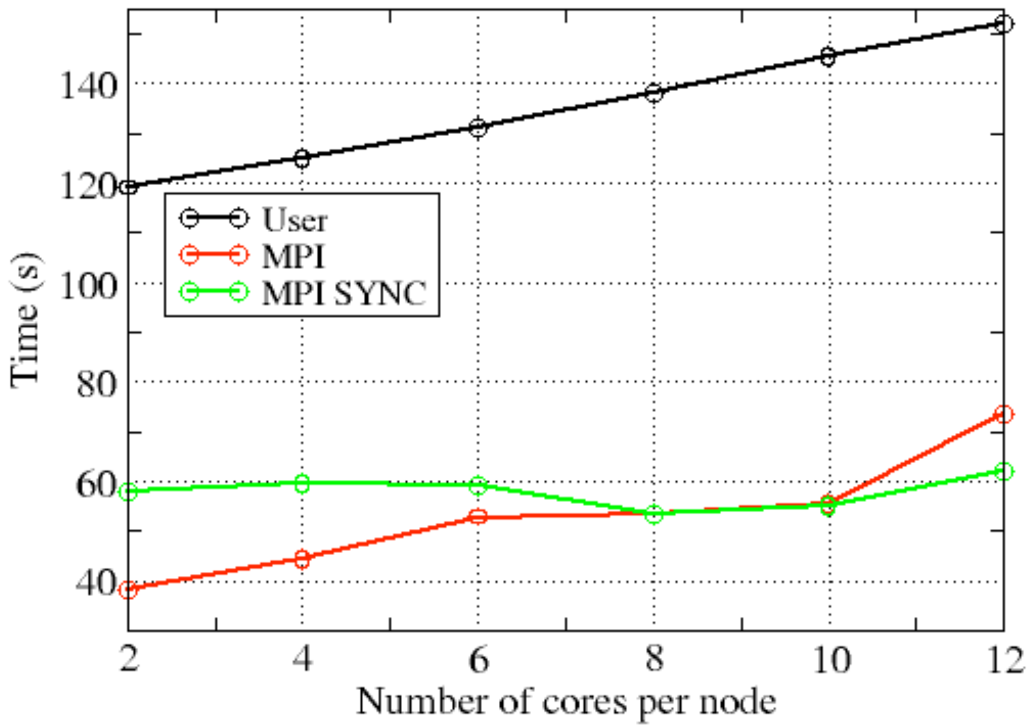
Mixed mode versus dm on XT4 and XT5

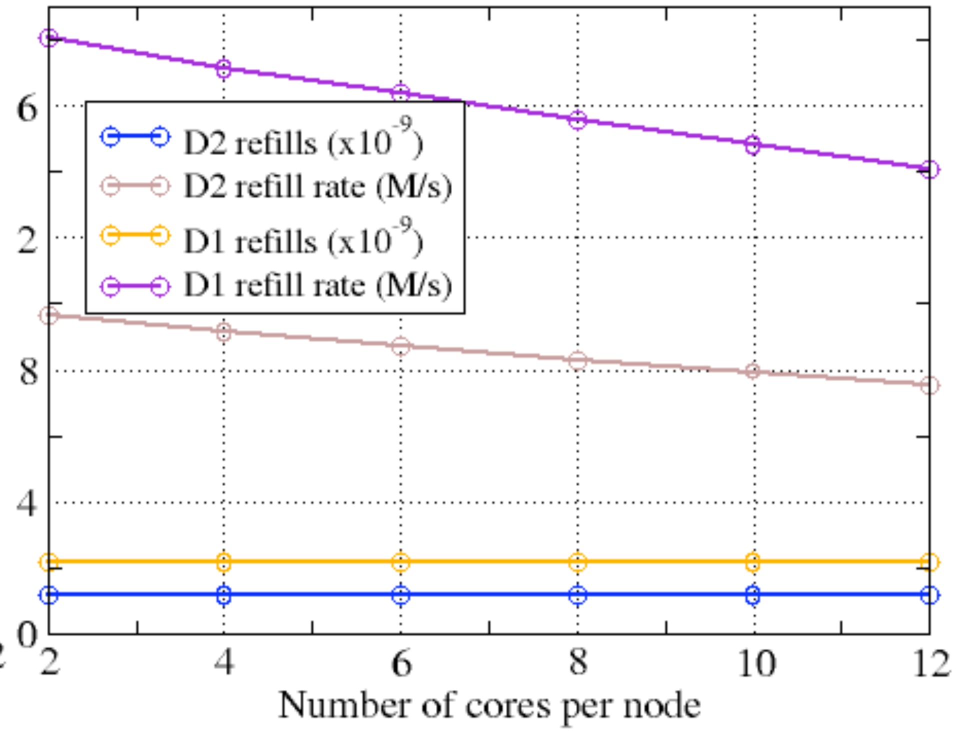


Compiler notes II

- PS dm+sm binary faster than PGI version
- dm+sm faster than dm on 512+ cores
  - Reduced MPI communications
  - Better use of cache
- WRF generally faster on 2.3 GHz quad-core XT4 than on 2.4 GHz hexa-core XT5
  - Only dm+sm version comes close to overcoming the difference

Under-populating XT5 nodes



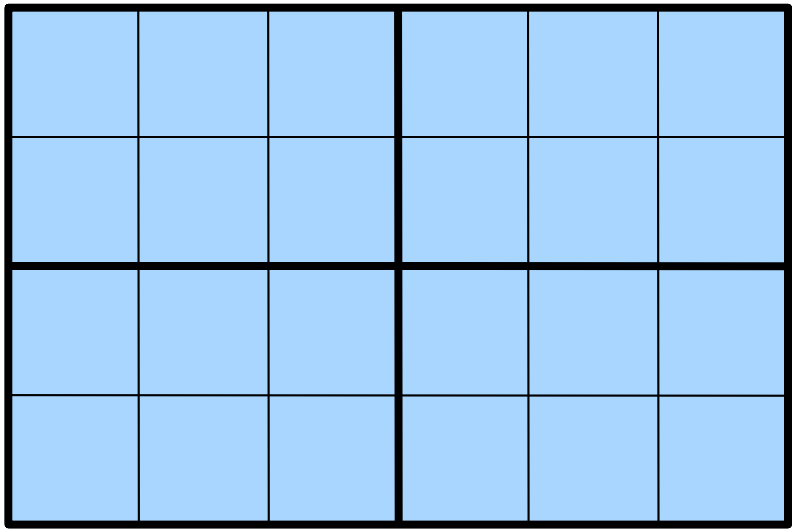


- De-populating steadily reduces time in both user and MPI code
- Rate of cache fills for user code steadily increases: **'memory wall'**

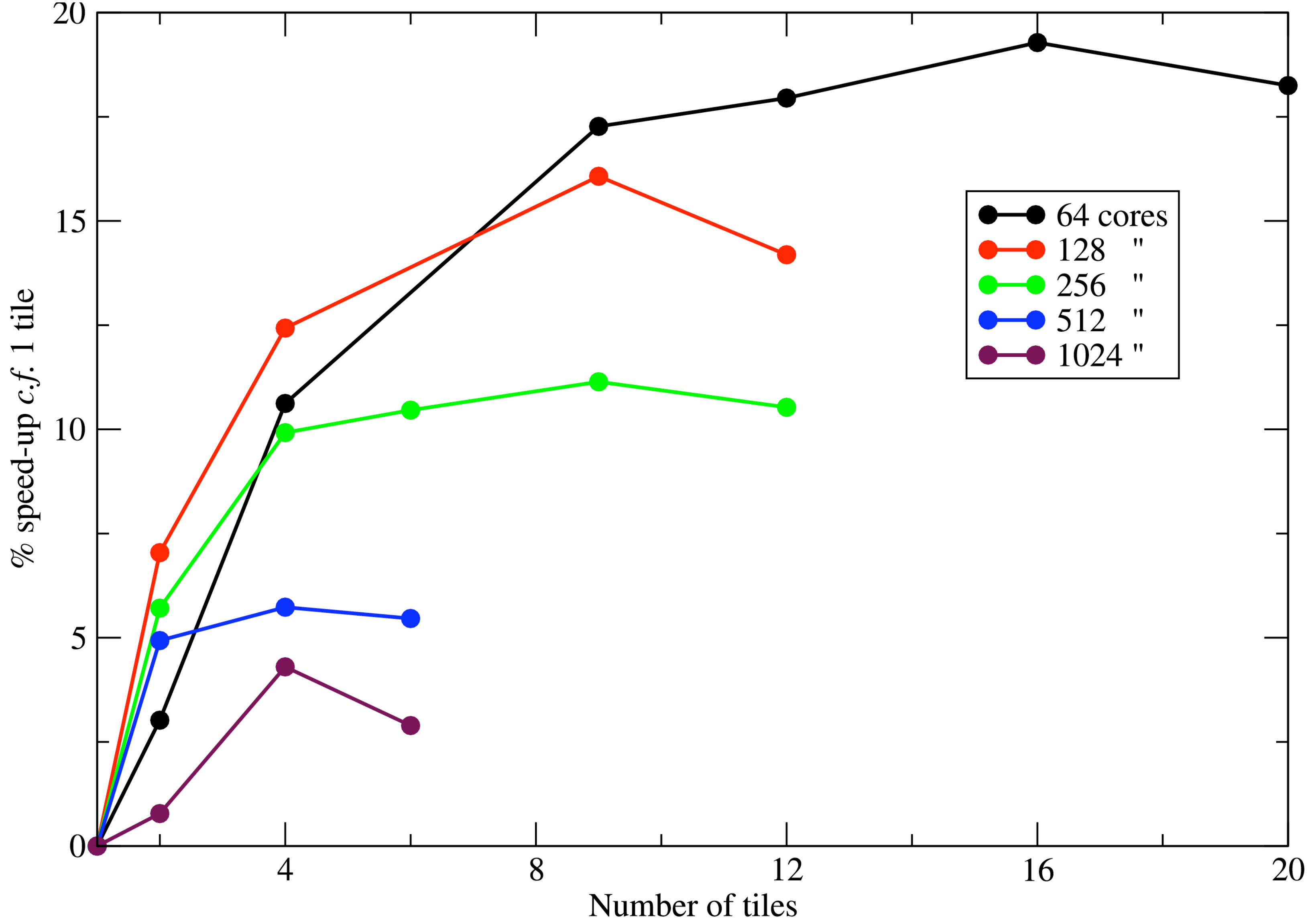
Improving cache usage

- Efficient use of large, on-chip memory cache is very important in getting high performance from x86-type chips
- Under MPI, WRF gives each process a 'patch' to work on. These patches can be further decomposed into 'tiles' (used by the OpenMP implementation)

e.g. decomposition of domain into four patches with each patch containing six tiles:



Performance variation with tiling



Notes on tiling performance

- Most effect on low core-count jobs because these have large patches and thus large array extents
- In this case, still get ~5% speed-up by using four tiles for both 512- and 1024-core MPI jobs
- HWPC data shows that improvement is largely due to better use of L2 ‘victim’ cache (20% hit rate => 70+% hit rate)

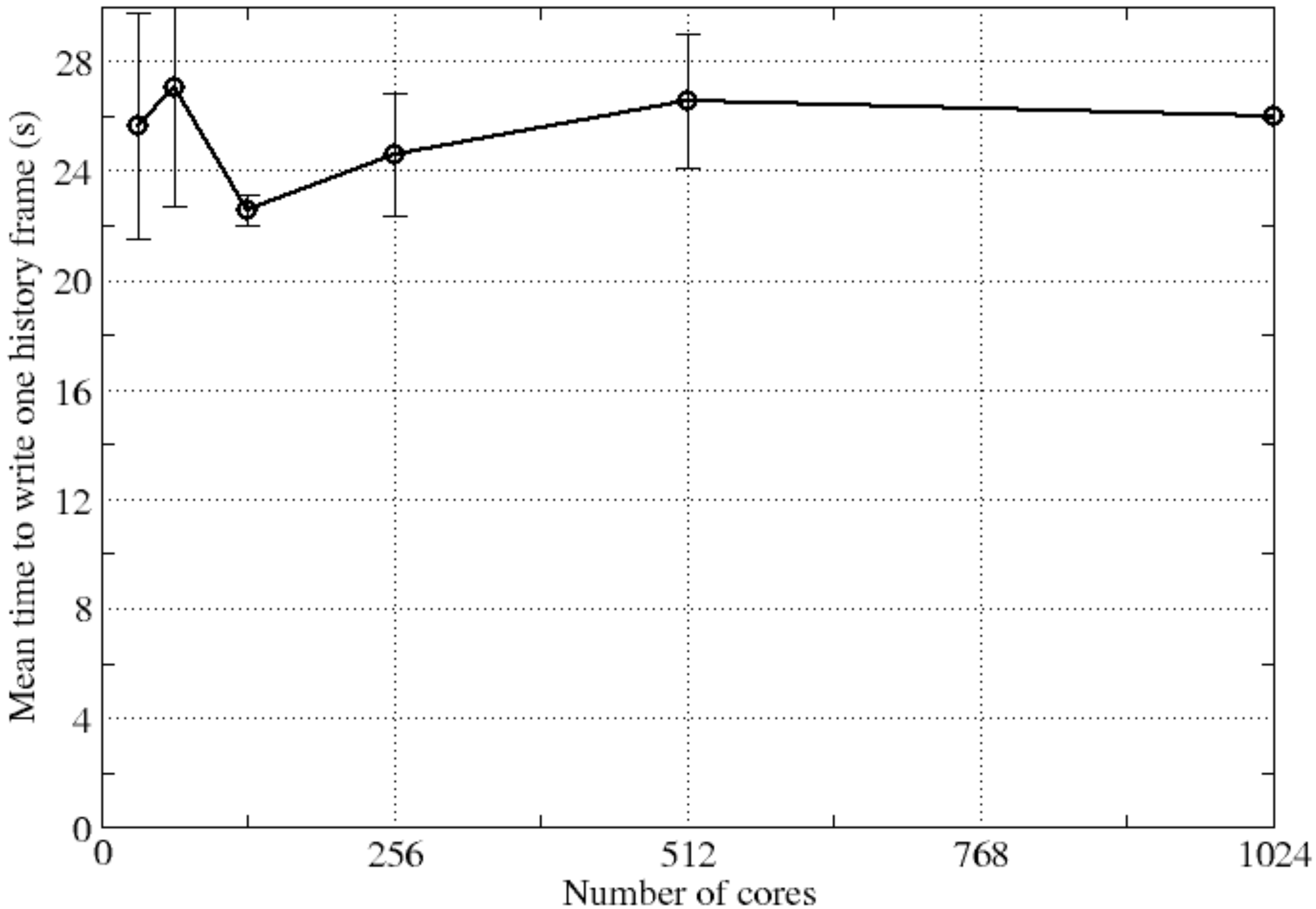
I/O Considerations

- All benchmark results presented so far carefully exclude effects of doing I/O
- But, MUST write data to file for job to be scientifically useful…
- Data written as ‘frames’
  - a snapshot of the system at a given point in time
  - One frame for GNR is ~1.6GB in total but this is spread across 3 files (1 per domain) and many variables

Science & Technology Facilities Council

Approaches to I/O in WRF

- **Default:** data for whole model domain gathered on 'master' PE which then writes to disk
- All PEs block while master is writing
- Does not scale
- Memory limitations



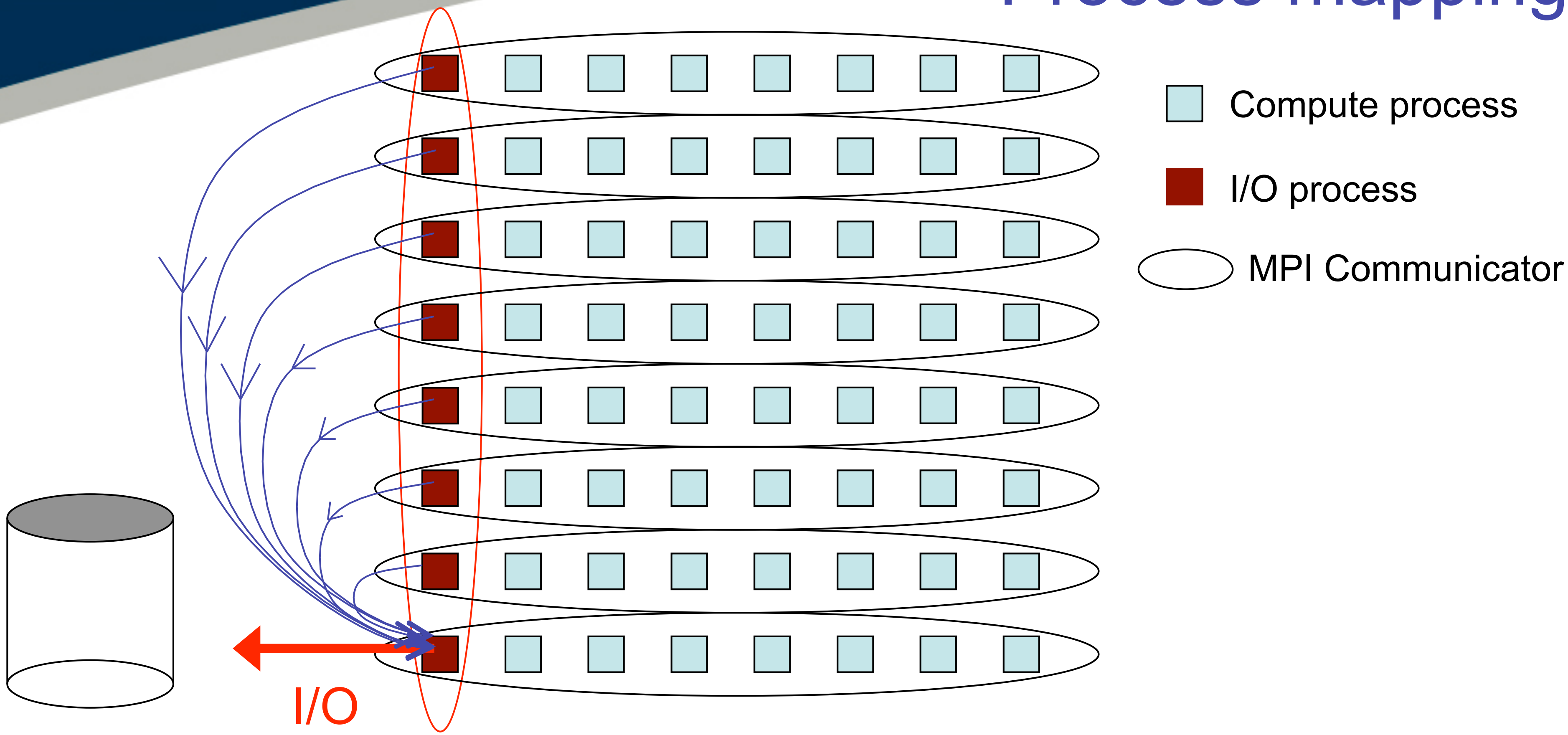
Parallel netCDF (pNetCDF)

- Uses the pNetCDF library from Argonne
- *Every* PE writes
- Current method of last resort when domain won't fit into memory of single PE
  - Will become more of a problem as model sizes and numbers of cores/socket increase
- Slow
  - Lots of small writes
  - *e.g.* 256-core job, mean time to write domain 3 with default method = 12s. Increases to 103s with parallel netCDF!

I/O Quilting

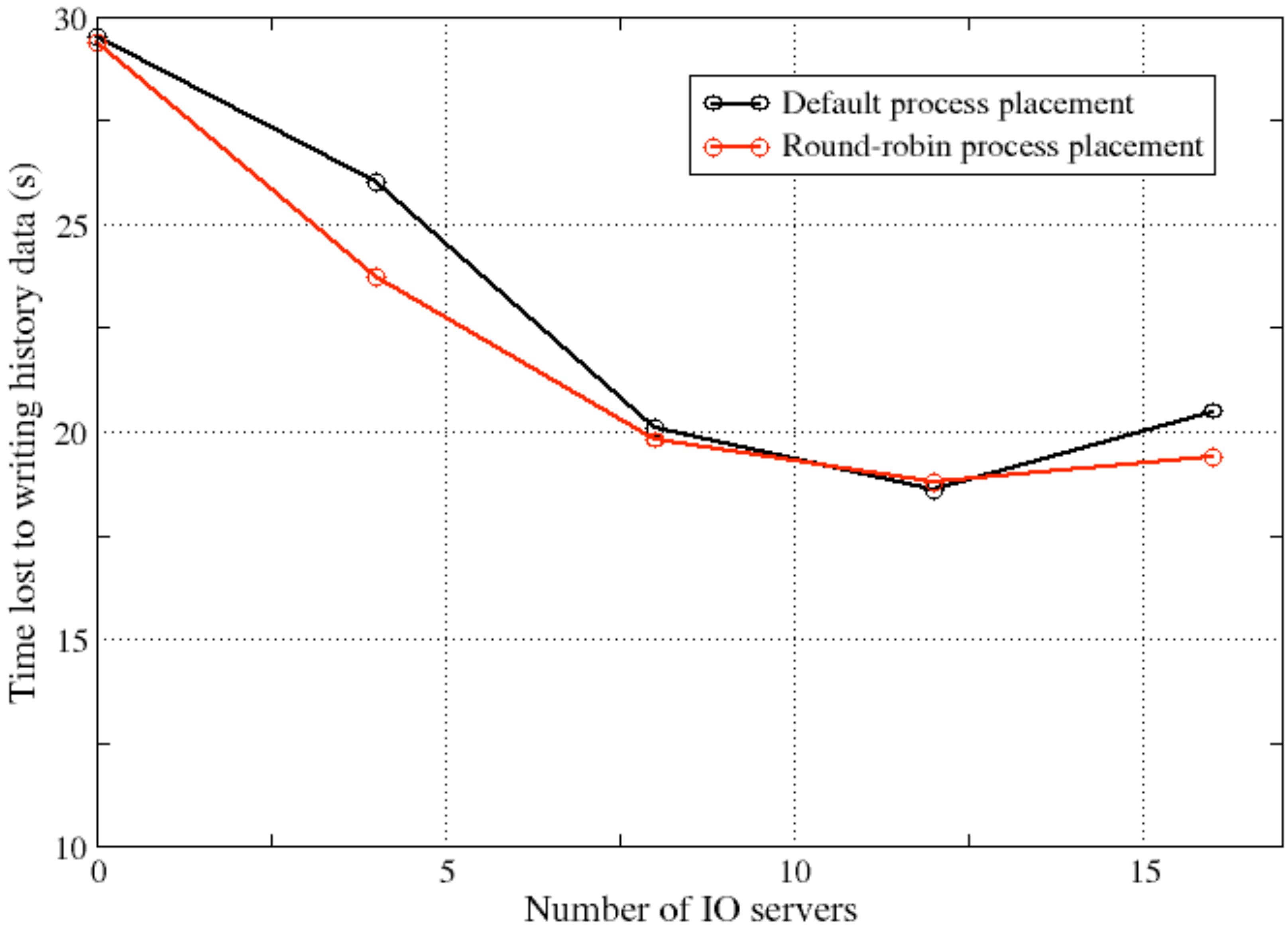
- Use dedicated ‘I/O servers’ to write data
- Compute PEs free to continue once data sent to I/O servers
- No longer have to block while data is sent to disk
- Number of I/O servers may be tuned to minimise time to gather data
- Only ‘master’ I/O server currently writes
  - Domain must still fit into memory

Process mapping

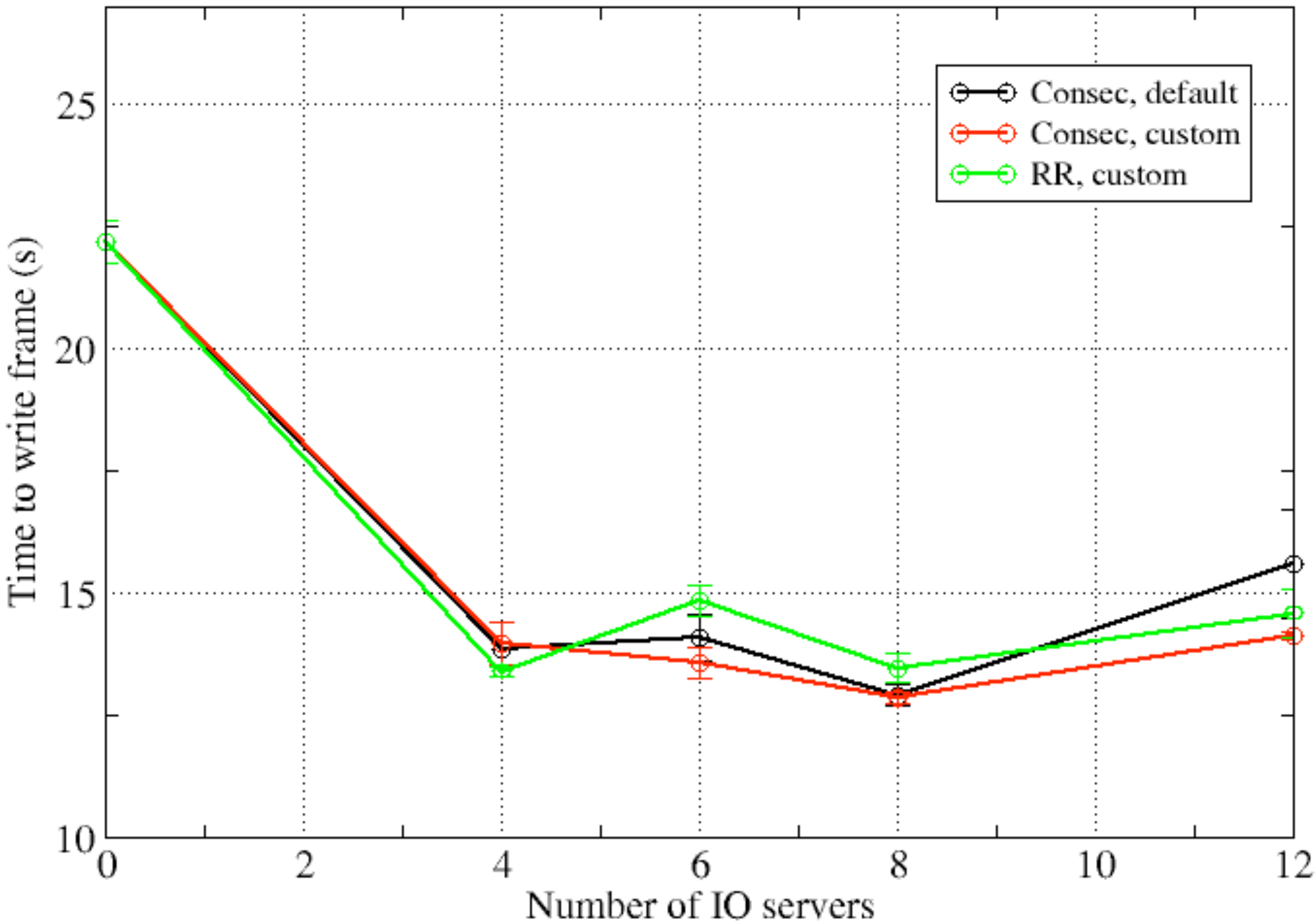


- How best to assign compute PEs to I/O servers?
- By default, all I/O servers end up grouped together on a few compute nodes

I/O quilting performance



Effect of process mapping



Science & Technology Facilities Council

Conclusions

- PGI best for dm build, PS for sm+dm
- sm+dm scales best; performs much better than dm on fatter nodes of XT5
  - Less MPI communication
  - Better cache usage
- Codes like WRF that are memory-bandwidth bound are not well-served by proliferation of cores/socket
- I/O quilting reduces time lost to I/O and is *insensitive* to process placement/mapping

Science & Technology Facilities Council

Acknowledgements

- EPSRC and NAG, UK for funding
- Alan Gadian, Ralph Burton (University of Leeds) and Michael Bane (University of Manchester) for project direction
- John Michelakes (NCAR) for problem-solving assistance and advice

andrew.porter@stfc.ac.uk