Performance and scaling of the LFRic weather and climate model on different generations of HPE Cray EX supercomputers

Mark Bull[1], Andrew Coughtrie[2], Deva Deeptimahanti[3], Mark Hedley[2], Caoimhín Laoide-Kemp[1], Christopher Maynard[2], Harry Shepherd[2], **Sebastiaan van de Bund[1] ***, Michèle Weiland[1], Benjamin Went[2]

[1] *EPCC, The University of Edinburgh*, Edinburgh (UK)
[2] *Met Office*, Exeter (UK)
[3] *Pawsey Supercomputing Research Centre*, Perth (AU)

[*] s.vandebund@epcc.ed.ac.uk

Met Office

pawsey

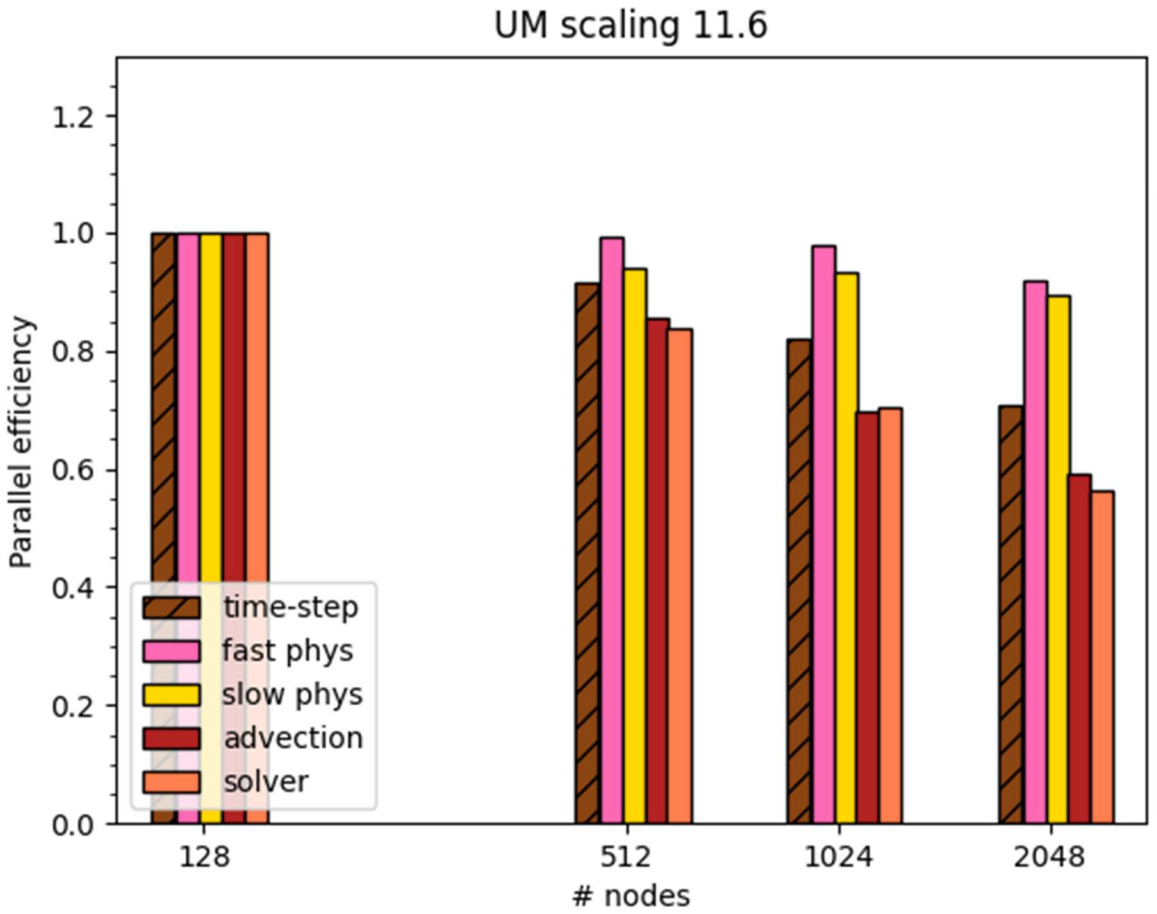
epcc

Why a new model? Unified Model scaling

UM 11.6: current Met Office weather and climate modelling code

N2048 (3072 × 4096)
~ 6.5 km
Cray XC40 intel Broadwell processors
36 cores per node
2048 nodes is 73,728 cores

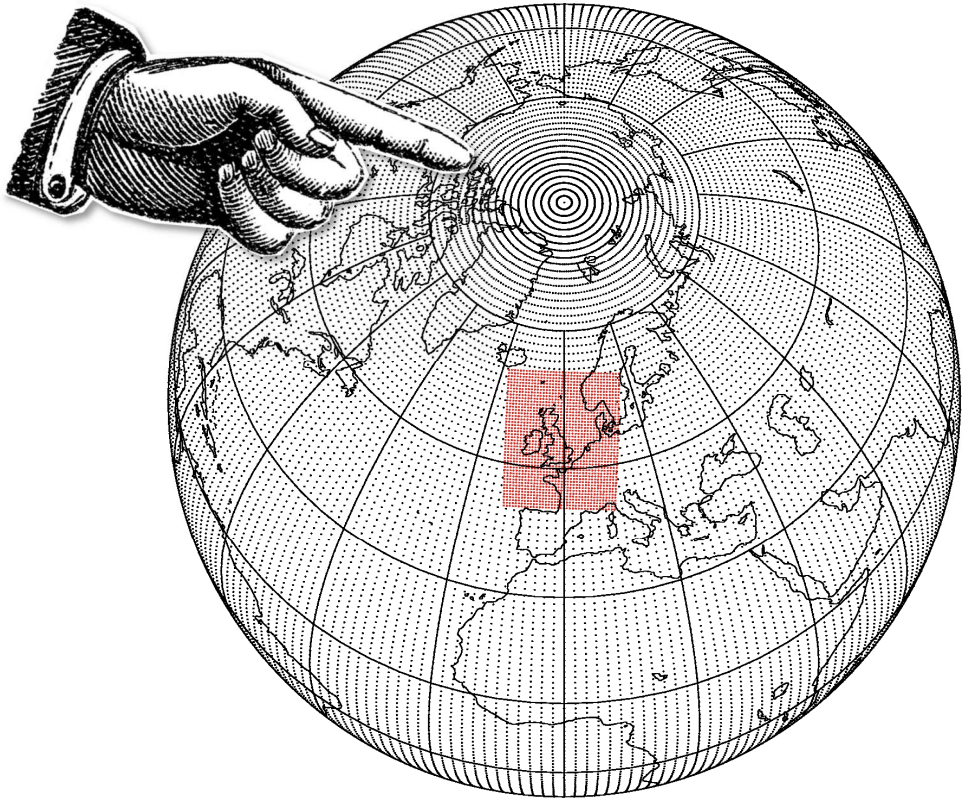
Advection around poles inhibits scaling



The Finger of Blame …

At 25km resolution, grid spacing near poles = 75m

At 10km reduces to 12m!



Semi-Lagrangian Advection
→ Large halos
→ Lots of communication near poles

Next Generation Modelling Systems: LFRic

Met Office is developing new modelling system for Exascale computing

- Replacing (successful) Unified Model (Weather and Climate)
- With **LFRic (Lewis Fry Richardson)** model with **Gung Ho** dynamical core

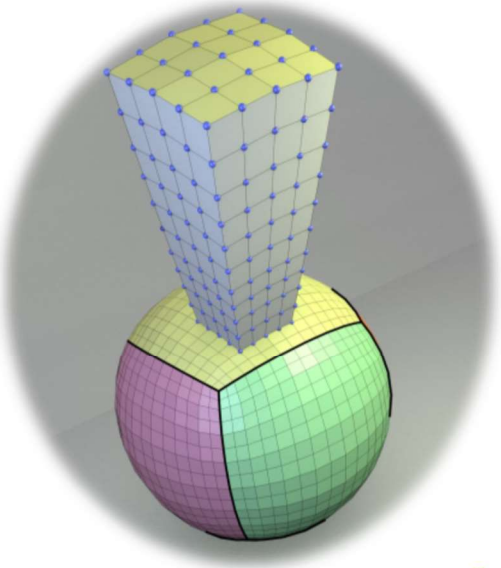
Maintain accuracy

Improve scalability
Exploit other programming models

Lon-lat grid (Poles) *structured*
Finite-difference
Semi-implicit

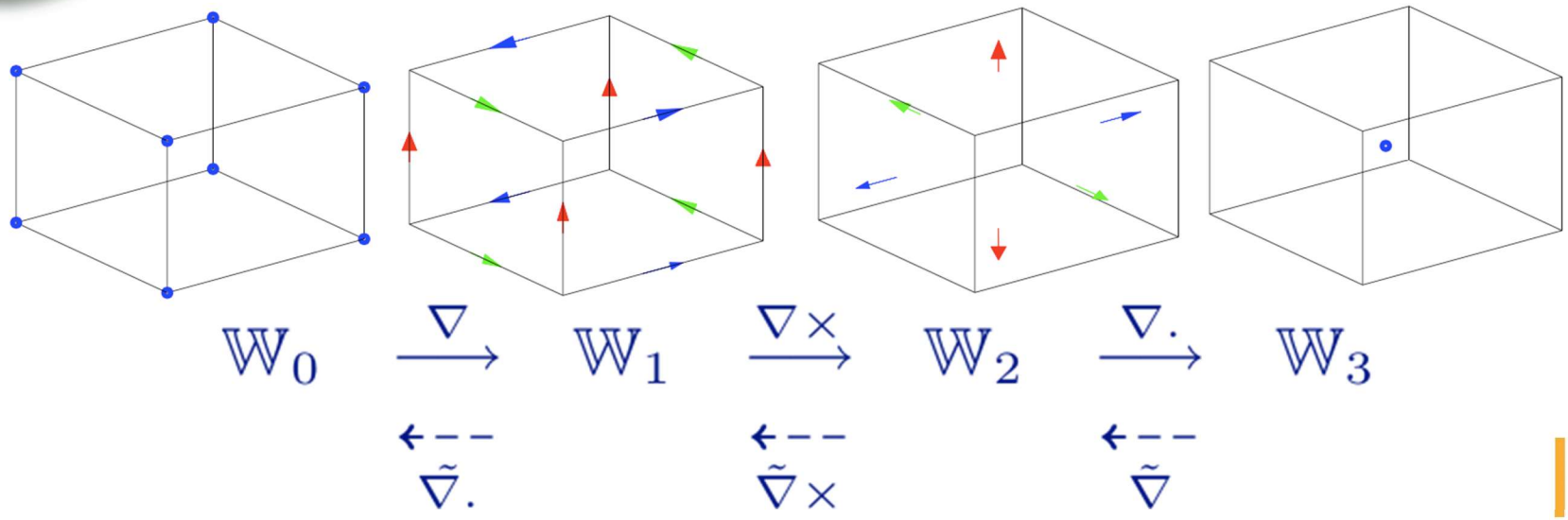
Cubed-sphere mesh – unstructured
Mixed finite element method
Semi-implicit

GungHo dynamical core



Cubed Sphere → no singular poles lon-lat
Unstructured mesh in the horizontal
Structured mesh in vertical with direct memory access
Mixed finite element scheme – *C-Grid*
Exterior calculus *mimetic* properties; semi-implicit in time

DOFs:



Domain Specific Language: PSyclone

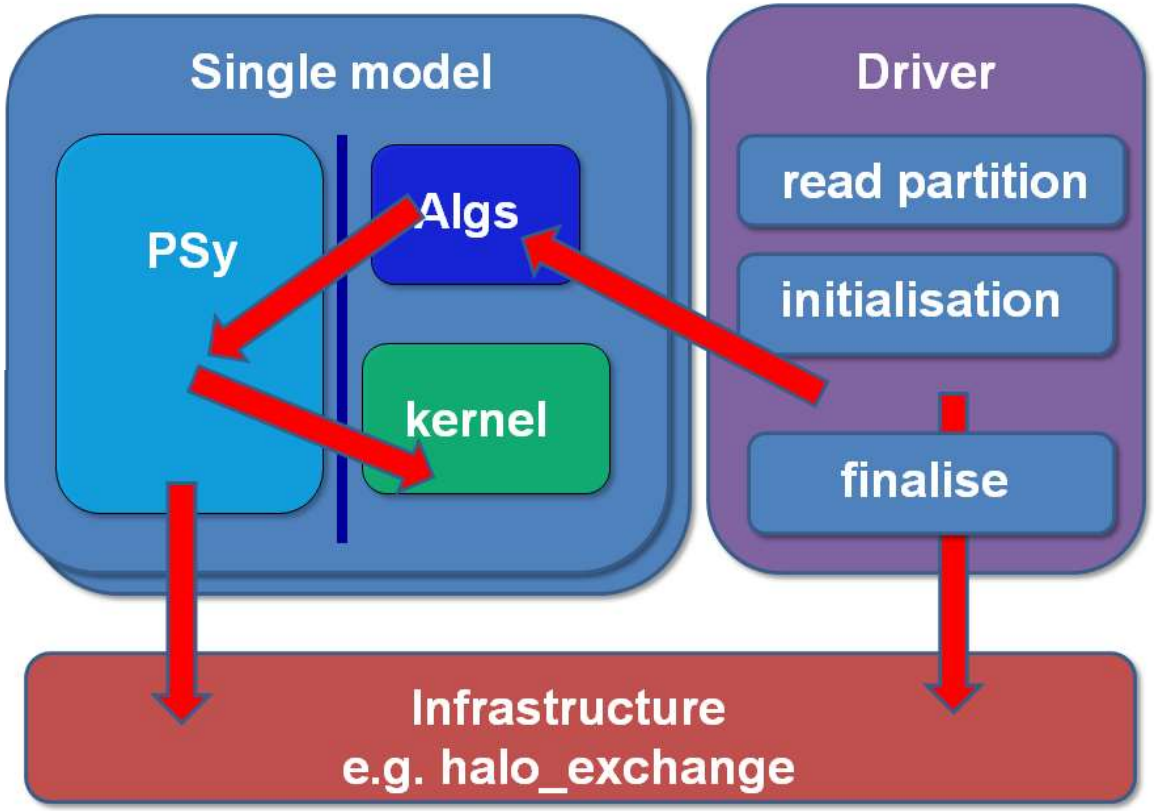
PSyclone

Why?

Too many programming models
Rapid evolution of diverse hardware
Parallel/optimisation code pollution in science code

How?

Restore *separation of concerns* between algorithms, kernels and parallel code
→Automatically generated code for OpenMP, halo exchange (MPI, using YAXT), redundant computation



HPE Cray EX Machines: Overview

System	ARCHER2	Setonix
CPU model	AMD EPYC 7742 "Rome" 64 cores	AMD EPYC 7763 "Milan" 64 cores
CPU clock frequency	2.0GHz (capped)	2.45GHz
CPUs per node	2	2
No. of nodes	5600	1600
Level 3 cache per CPU	16 × 16MB	8 × 32MB
NUMA domains per CPU	4	4
Interconnect	Slingshot 10	Slingshot 11
Cray MPICH version	8.1.23	8.1.25
Cray Fortran compiler version	15.0.0	15.0.1
GNU Fortran compiler version	11.2.0	12.2.0

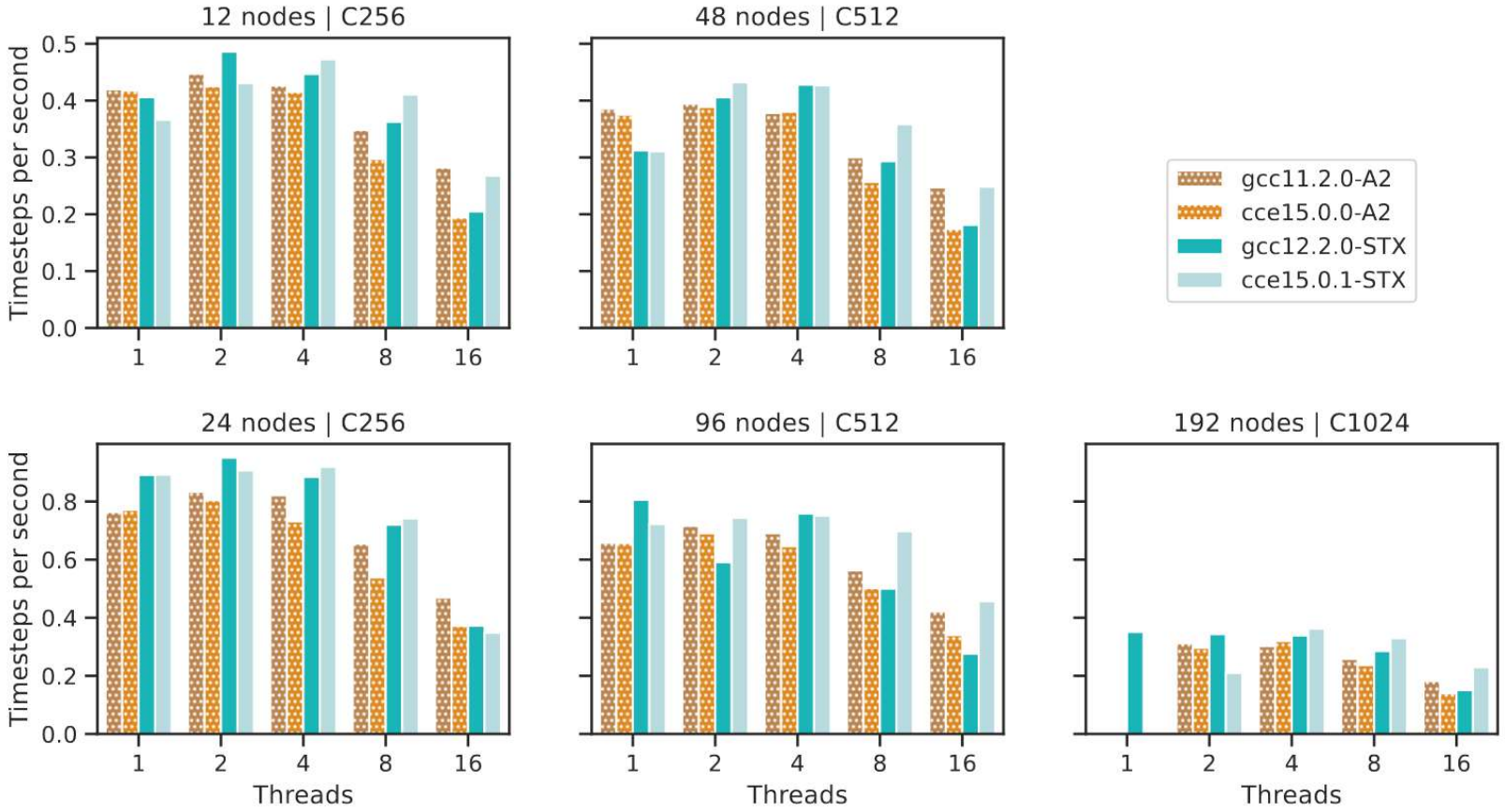
GungHo performance analysis: setup

- 3 different mesh sizes of the cubed sphere mesh: C256, C512 and C1024
- 120 vertical levels
- 96 timesteps
- I/O turned off
- 12 to 192 nodes, 1 to 16 threads

- C.f. operational config: C896 / 70 levels, running on 147 nodes

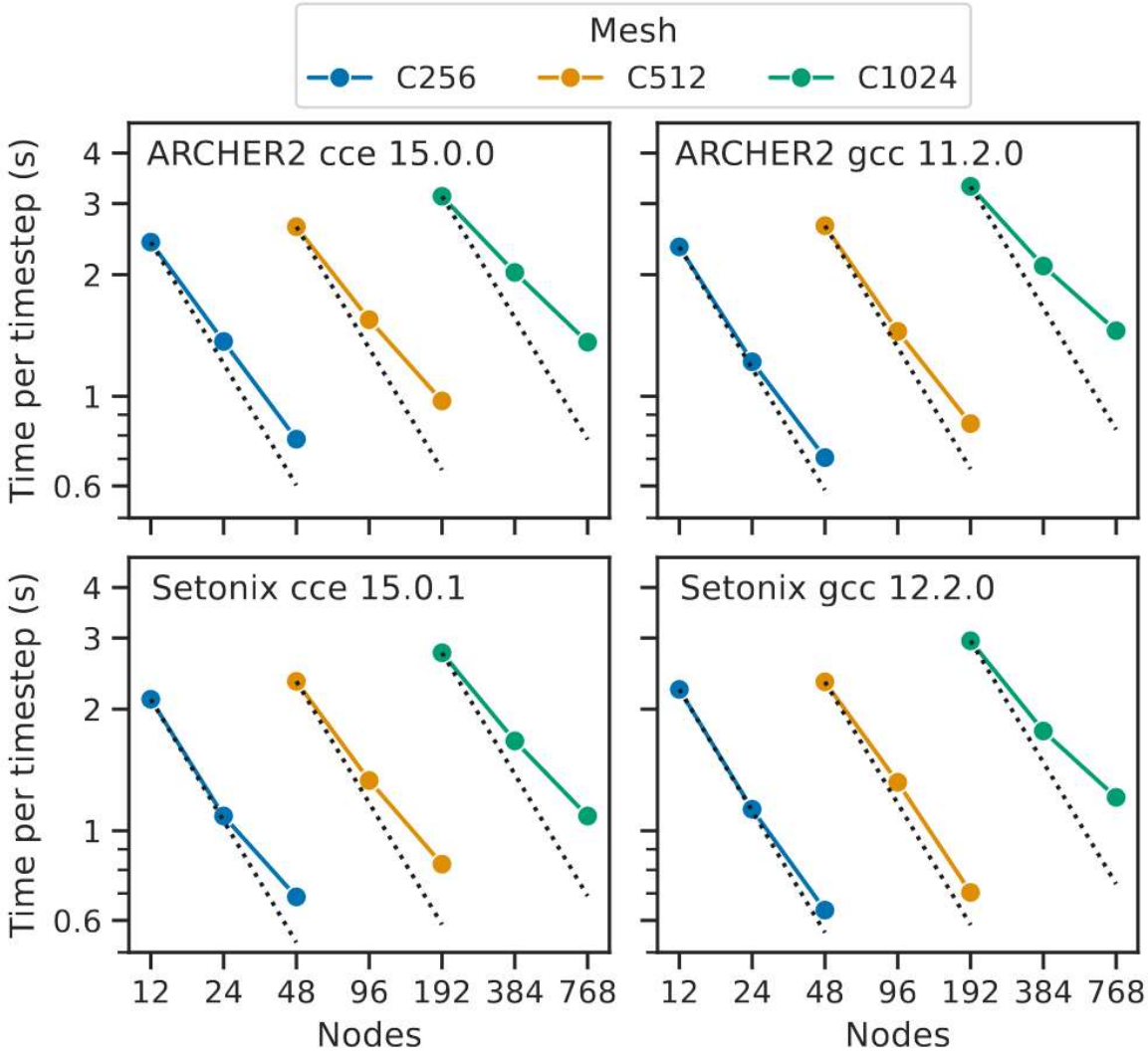
Weak scaling

- Local volume:
  - 256 grid points (top)
  - 128 grid points (bottom)
- 1, 2, 4 threads/rank ideal
- Slightly better performance on Setonix



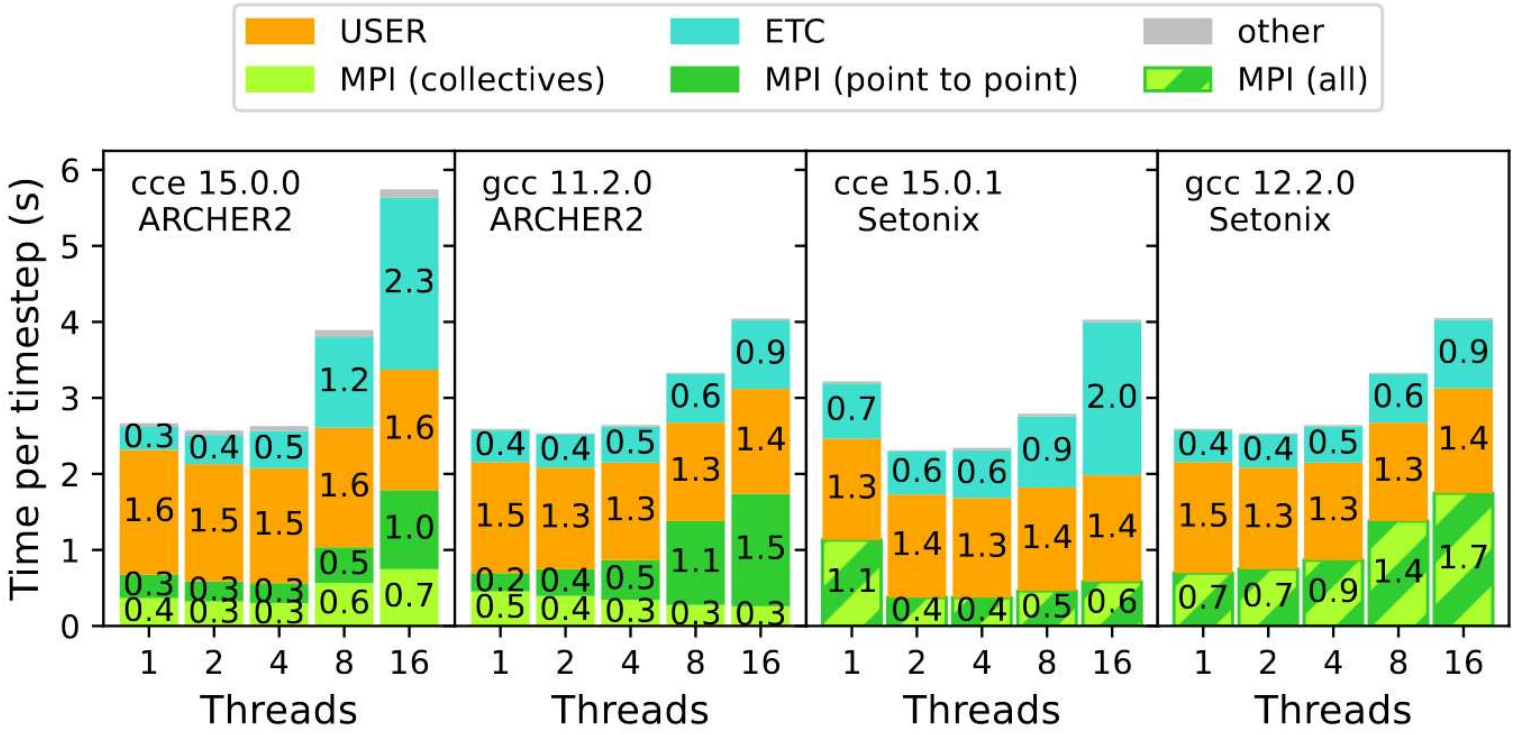
Strong Scaling

- **4 OpenMP threads** (32 ranks per node)
- Comparison of Machines and Compiler Suite
- Higher deviation from ideal scaling for higher mesh sizes



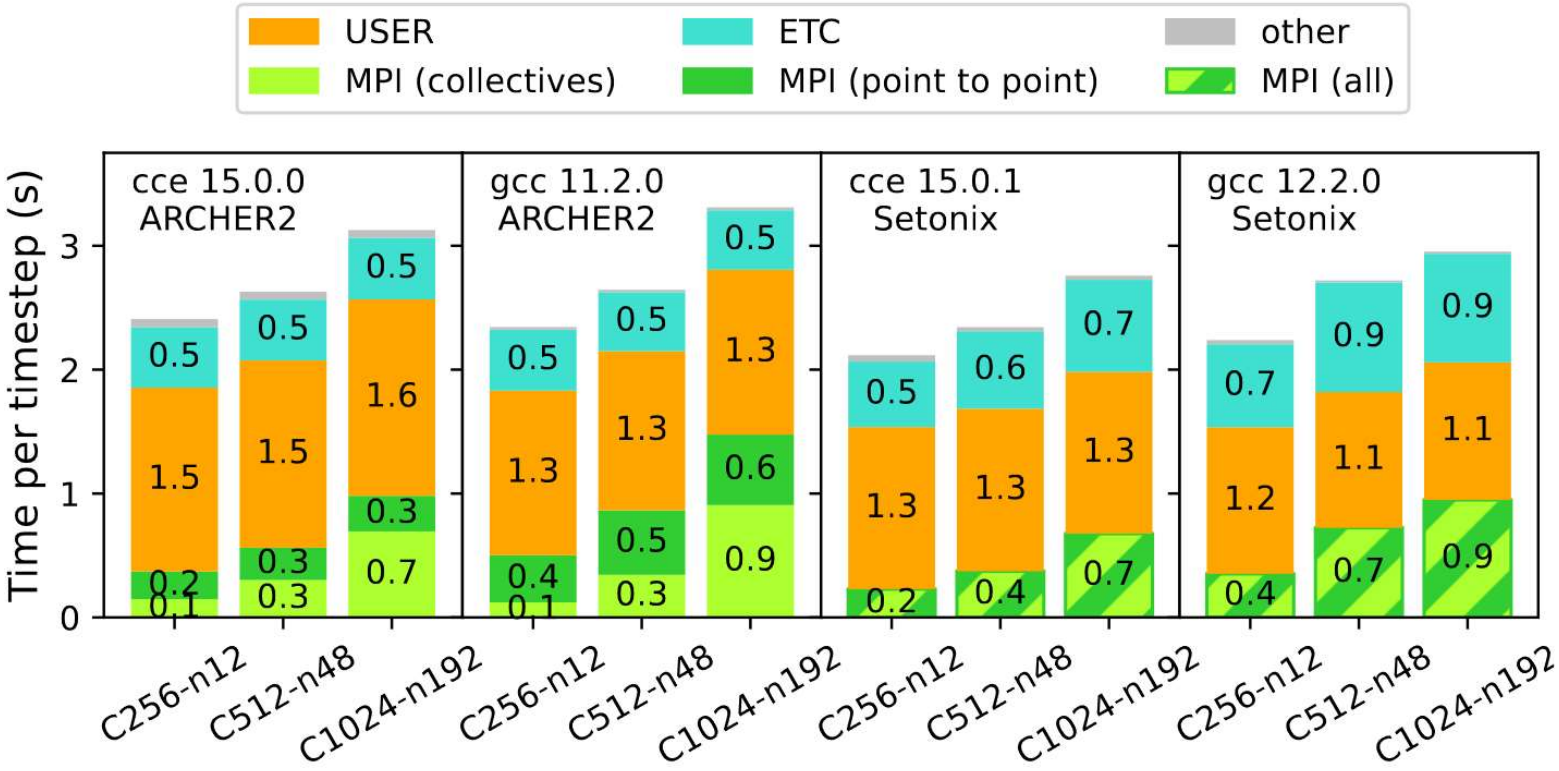
CrayPat Profiling: comms vs compute

- **C512, 48 nodes**
- 2,4 threads: improvement:
  - Fewer redundant halo computations
  - Fewer ranks involved in global sums
- 4+ threads slowdown
  - USER time roughly const.
  - Thread synchronisation issues



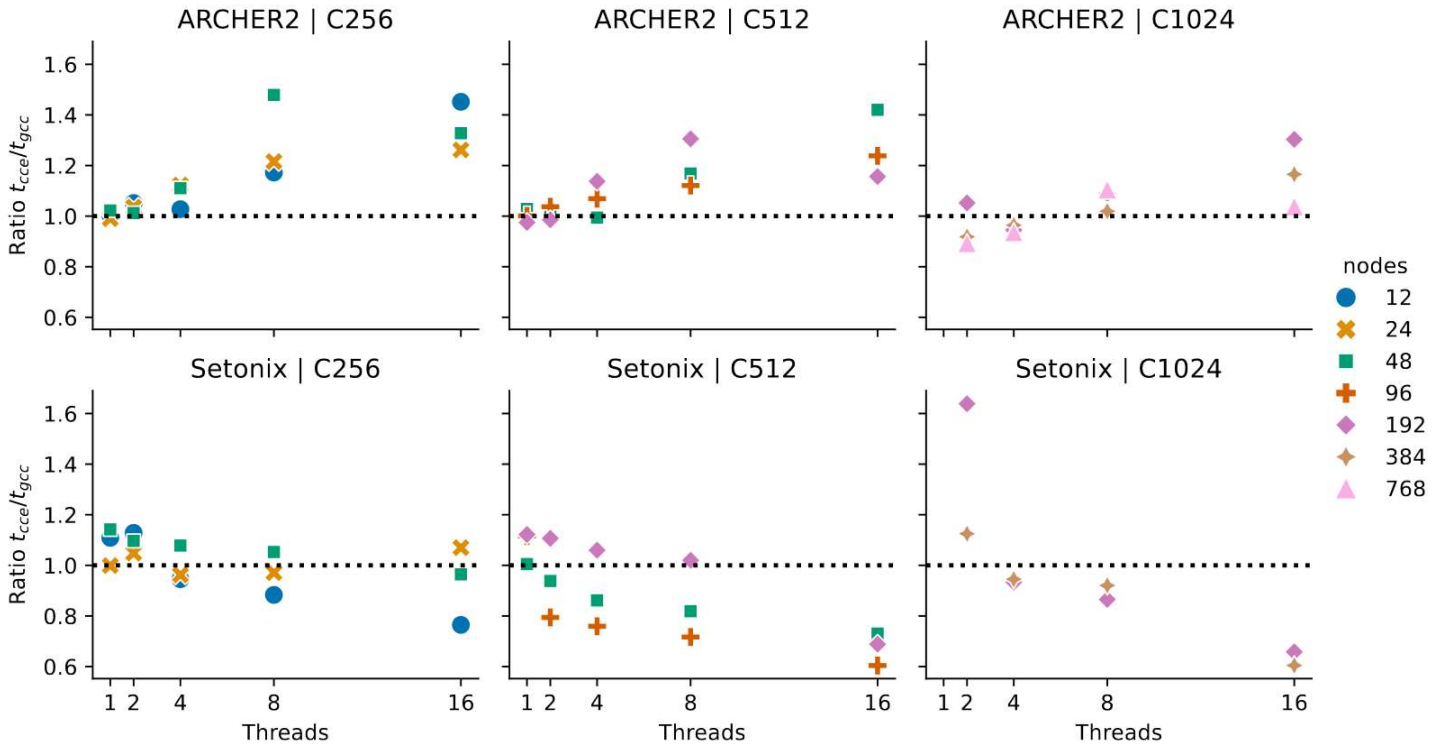
CrayPat Profiling: fixed local volume

- **Constant local volume, 4 threads**
- USER time roughly constant
- Loss of weak scaling due to increased collective comms



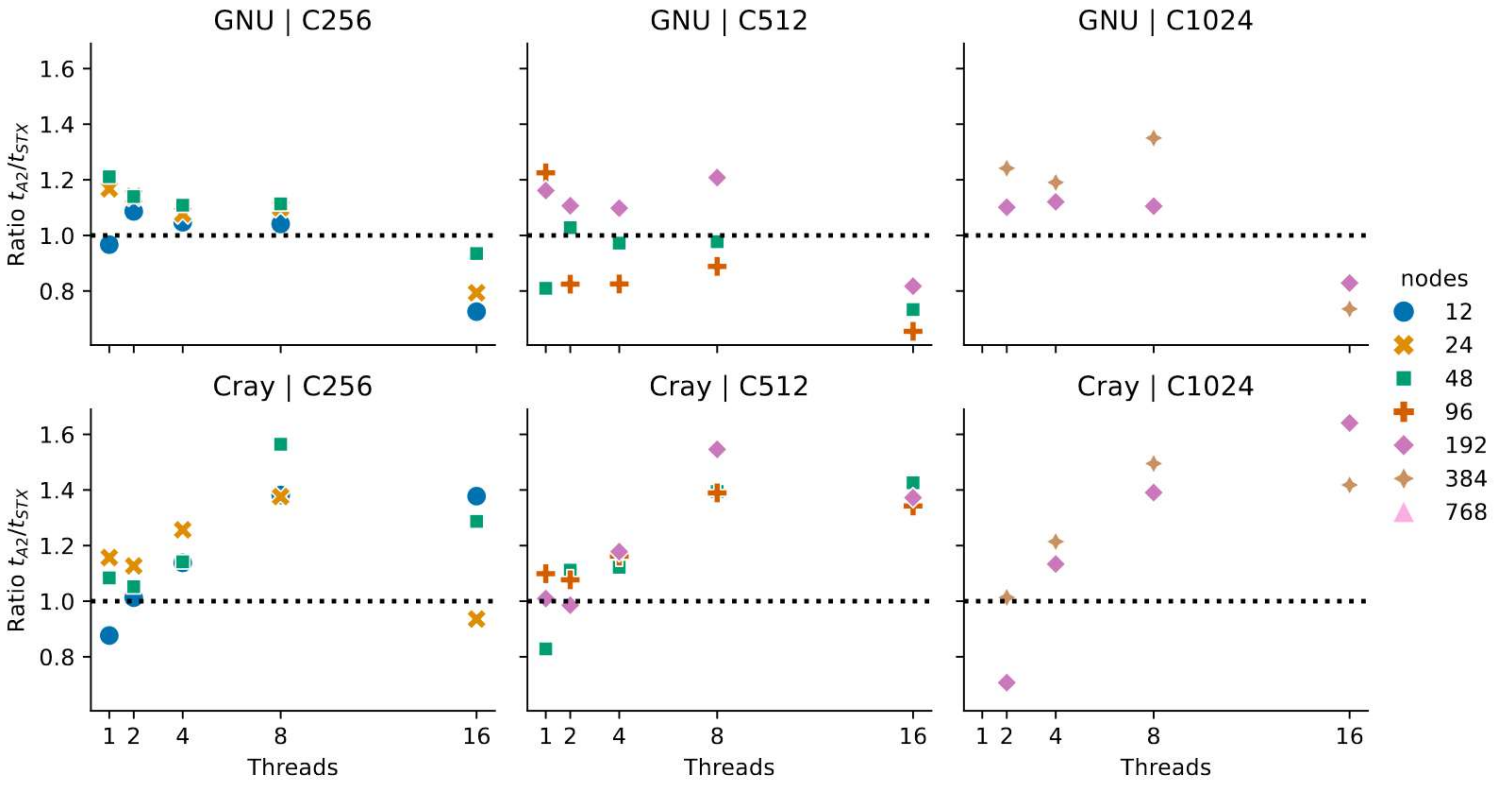
Compiler suite comparison

- Ratio of Cray vs GNU
- Lower → Cray performs better
- Small differences for 1,2,4 threads
- Higher thread counts:
  - GNU tends to be faster on ARCHER2
  - Cray tends to be faster on Setonix



Machine comparison

- ARCHER2 vs Setonix for same compiler suite
- Lower → ARCHER2 performs better
- Setonix generally performs better
- 8 thread differences possibly due to L3 cache differences
  - Shared between 8 cores on Setonix, but 4 on ARCHER2

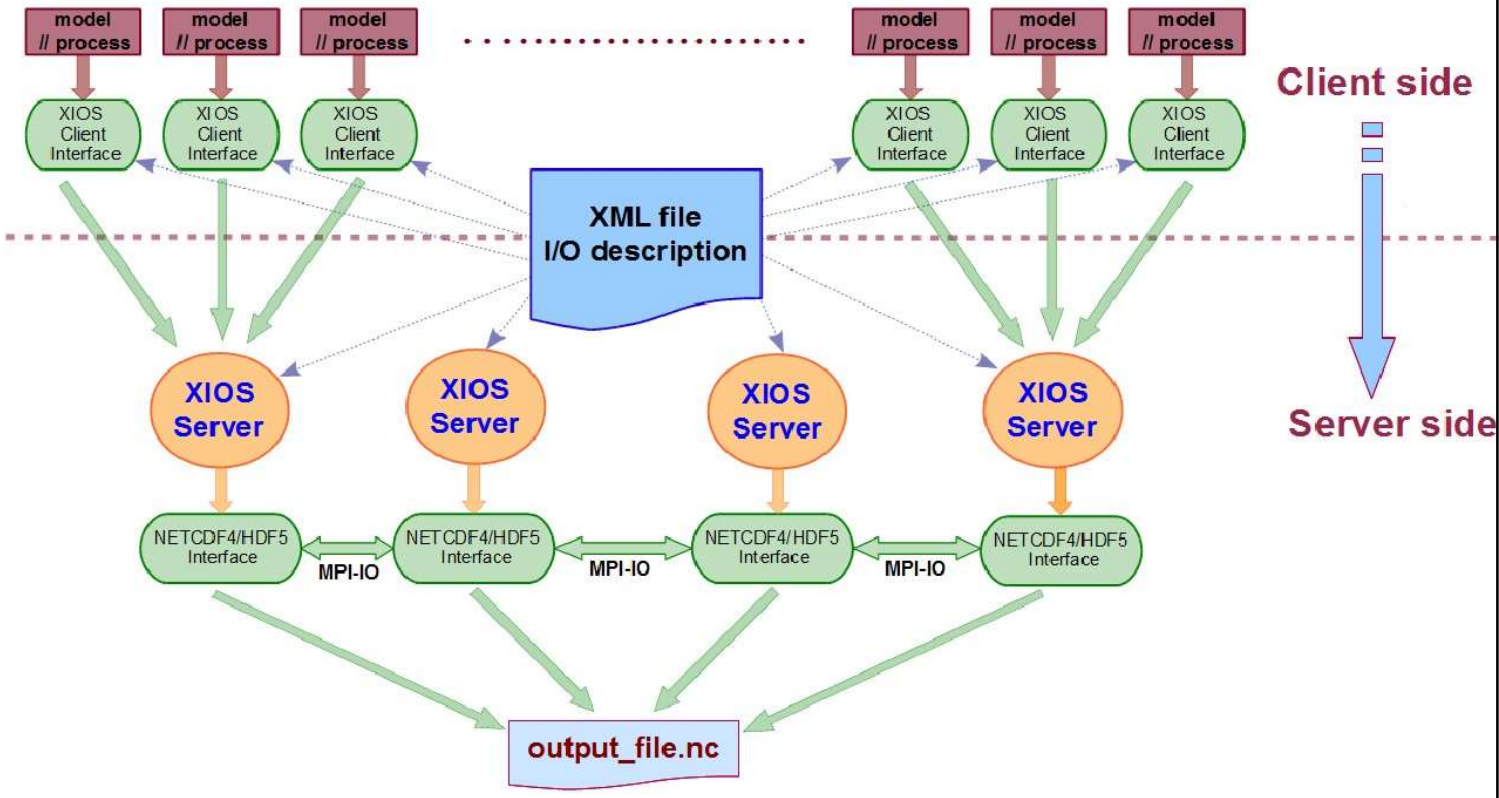


Summary: GungHo performance

- Good scaling up to 768 nodes on C1024/120 levels
  - Operational config: 147 nodes, C896/70 levels
- 2 or 4 threads per MPI/rank gives modest performance gains over MPI only
- Cray/GNU compiler performance broadly similar
- Slightly better performance on Setonix than ARCHER2

XIOS and I/O performance

- LFRic uses XIOS (XML I/O Server), released/maintained by Institut Pierre-Simon Laplace
- Asynchronous client-server data transfer (XIOS2)
- Buffers data and manages parallel reads/writes to netCDF files: try to hide I/O from simulation



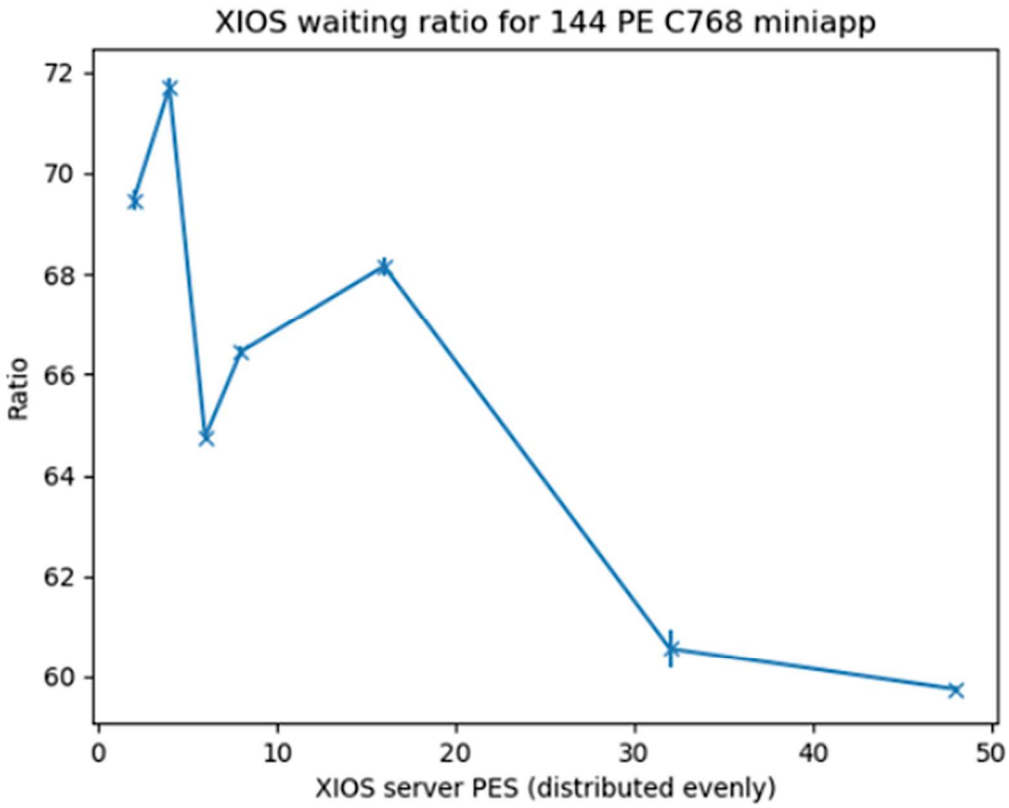
XIOS Scenarios

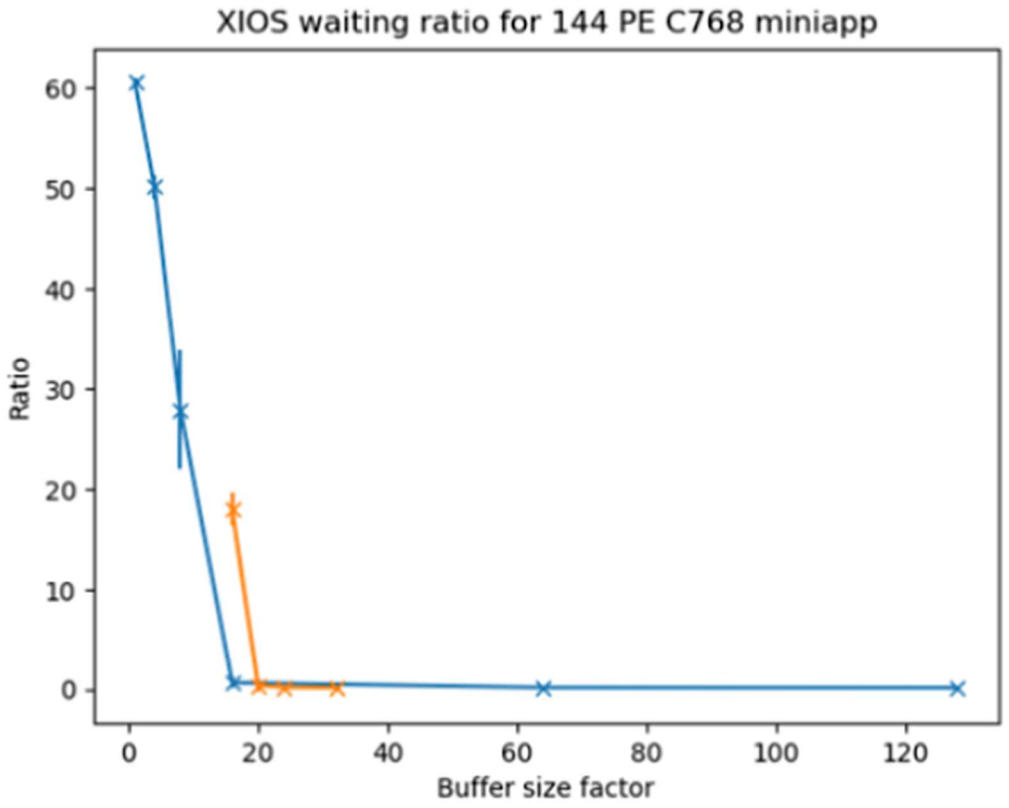
- Results run on Met Office Cray XC40 using XIOS2
- Lustre-based disk storage (similar to ARCHER2/Setonix)
- ***3 test cases:***
  - C768: Varying Processor Numbers and Buffer Sizes
  - C192: Diagnostic Load Tests
  - C896: Operational config

Measures:

- Wall clock time
- XIOS Client buffer wait %
- XIOS server write rate (MiB/second)

C768: Server PEs and buffer size



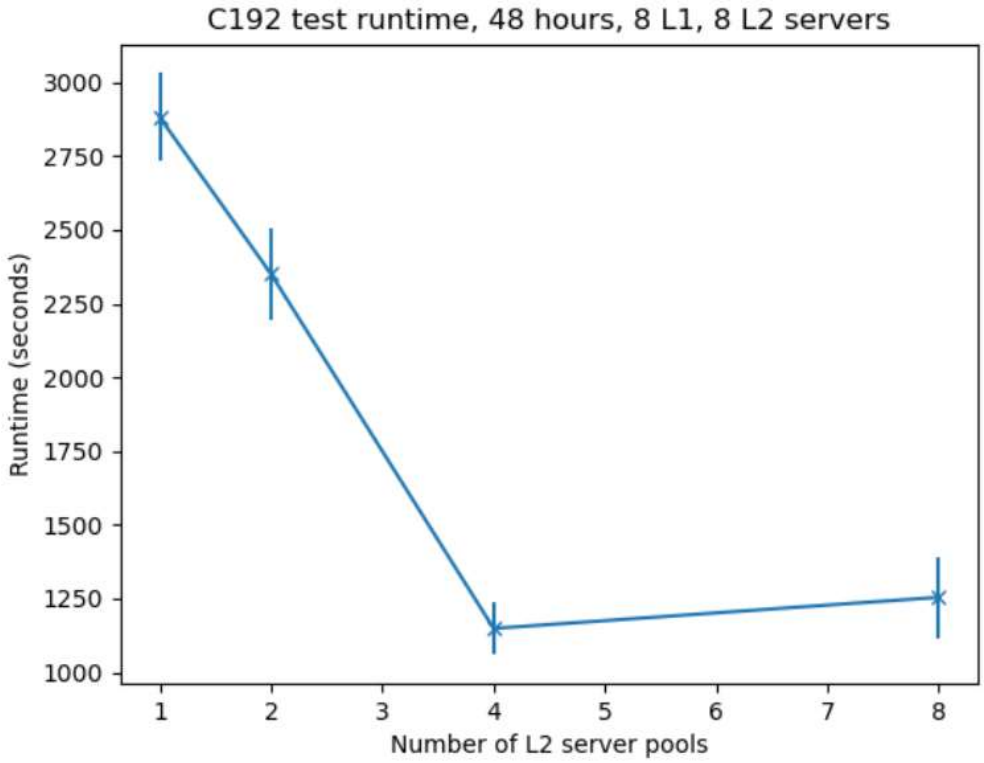


- Much more impact from changing buffer size

C192: Diagnostic Load Tests

- 48 model hours: write 5329 fields (approx. 400 GiB), with 864 LFRic PEs
- Baseline: 72 XIOS PEs:
  - *43.9 mins write time*
  - *15 mins buffer wait time(!)*
- Use XIOS “level 2” servers
- 8 level 1 / 8 level 2 (4 pools of 2 servers each)
  - *21 mins compute*
  - ***0.7 mins** buffer wait time*

- Pool size affects performance
- 2 servers/pool optimal



C896 performance sensitivity

- Writing 1.1TiB of diagnostic data
- Nodes either for simulation or I/O
- Lustre striping leads to significant performance boost

- 25% improvement in wall clock time
- 253% increase in server write rate
- 5× reduction in client wait %

	Baseline	Performance Run
Characteristics		
XIOS Nodes	22 / 153 (14.4%)	24 / 156 (15.5%)
XIOS Ranks per Node	17.81	19
Lustre Striping	None	Full
Log Levels	XIOS:50, LFRic:Info	XIOS:1, LFRic:Warn
Measures		
Test Wall Clock Time:	5140.67 s $\pm\sigma$ 52.12	3861.52 s $\pm\sigma$ 21.36
Data Intensity	0.15 GB per core hour	0.20 GB per core hour
Server Process av. write rate	377.04 MiB/s $\pm\sigma$ 12.23	953.43 MiB/s $\pm\sigma$ 64.87
Client Time Buffer Wait %	7.56 % $\pm\sigma$ 1.60	1.44 % $\pm\sigma$ 0.66
Client time Buffer Wait	382.24 s $\pm\sigma$ 81.25	53.49 s $\pm\sigma$ 24.63

XIOS performance: conclusions

- I/O significant factor in model performance: more proportional impact for high diagnostic loads/high horizontal model resolutions
- Two level server config shows promise
- Significant performance improvements through minor configuration changes

Future work:

- XIOS performance sensitivity testing on Archer2 EX architecture shown promising tuning opportunities

Acknowledgements

- GungHo performance analysis work was undertaken as part of the Met Office Academic Partnership, and used the ARCHER2 UK National Supercomputing Service (https://www.archer2.ac.uk).
- This work was also supported by resources provided by the Pawsey Supercomputing Research Centre https://pawsey.org.au with funding from the Australian Government and the Government of Western Australia.

Met Office

pawsey

epcc

Summary

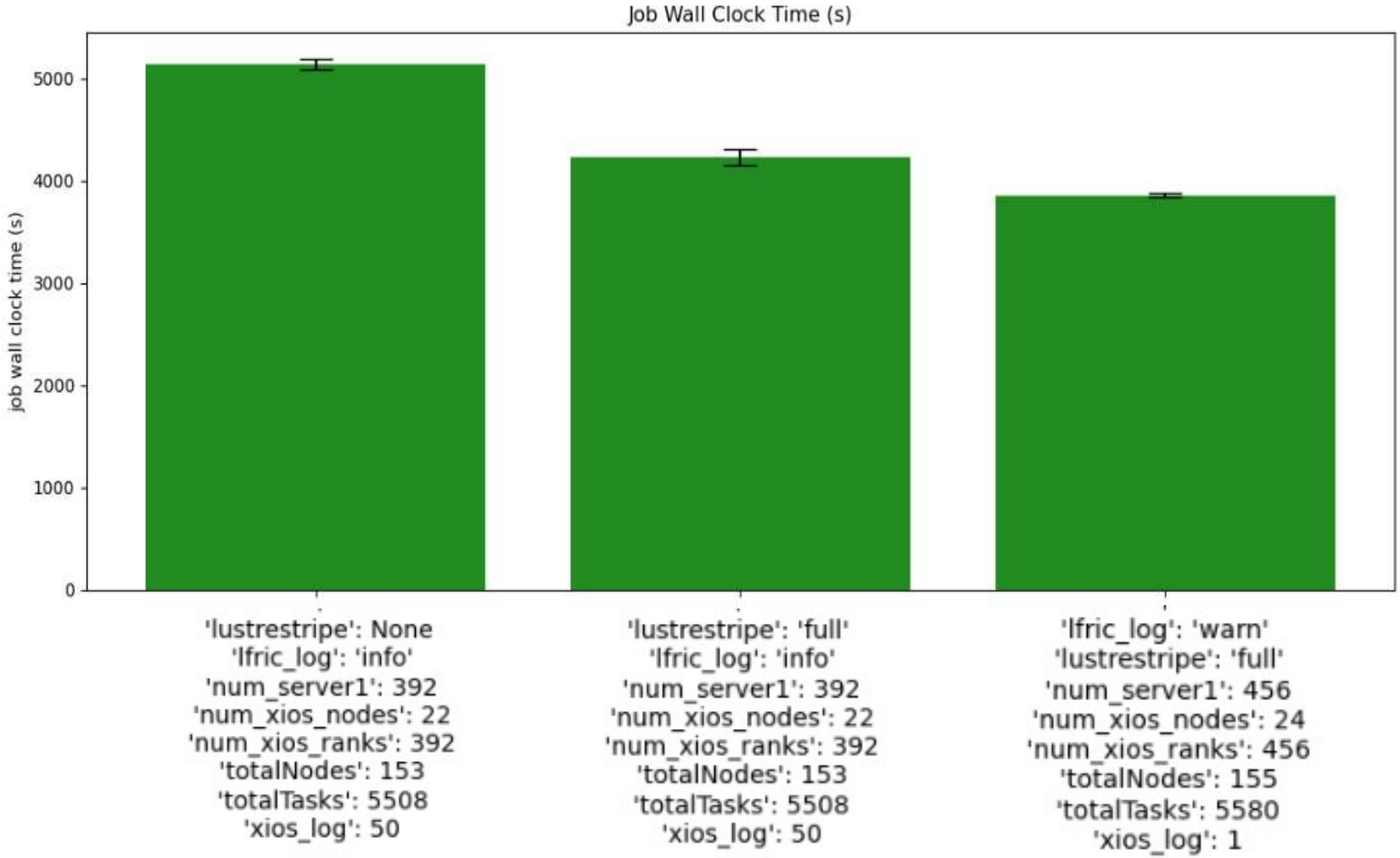
GungHo performance:

- Good scaling up to 768 nodes on C1024/120 levels
  - Operational config: 147 nodes, C896/70 levels
- 2 or 4 threads per MPI/rank gives modest performance gains over MPI only
- Cray/GNU compiler performance broadly similar
- Slightly better performance on Setonix than ARCHER2

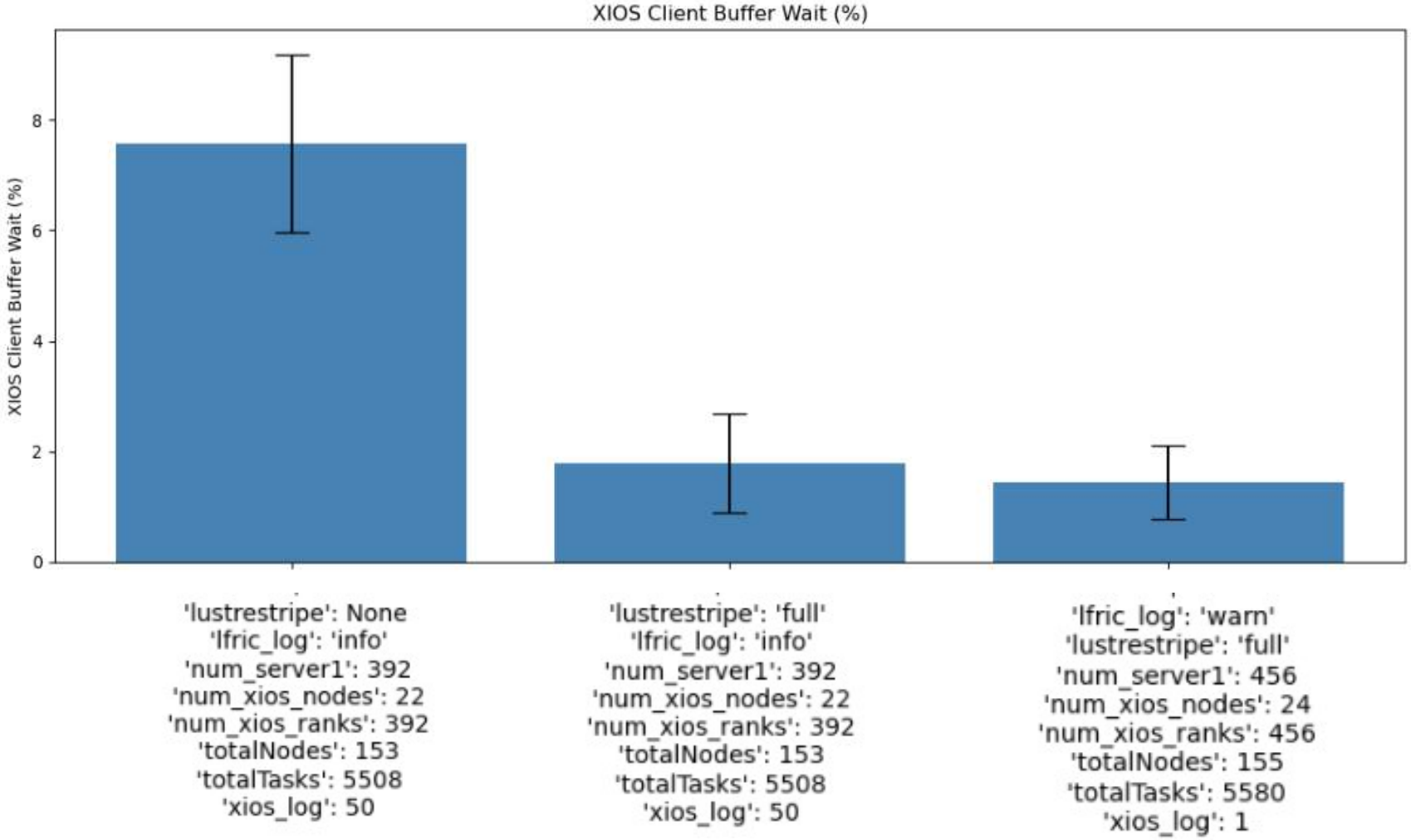
XIOS performance:

- Significant performance improvements through minor configuration changes to model interactions with storage systems.
- Two level server config shows promise
- Significant performance gains and tuning opportunities from testing on ARCHER2

XIOS: Wall clock time



XIOS: Buffer Wait %



XIOS: Server Write Rate