OpenFabrics and UCX

Performance on the ARCHER2 HPE Cray EX system

Michael Bareford, David Henty, William Lucas, Andy Turner
EPCC, The University of Edinburgh
www.archer2.ac.uk

Reusing this material

Partners

epcc

UKRI

Engineering and Physical Sciences Research Council

Natural Environment Research Council

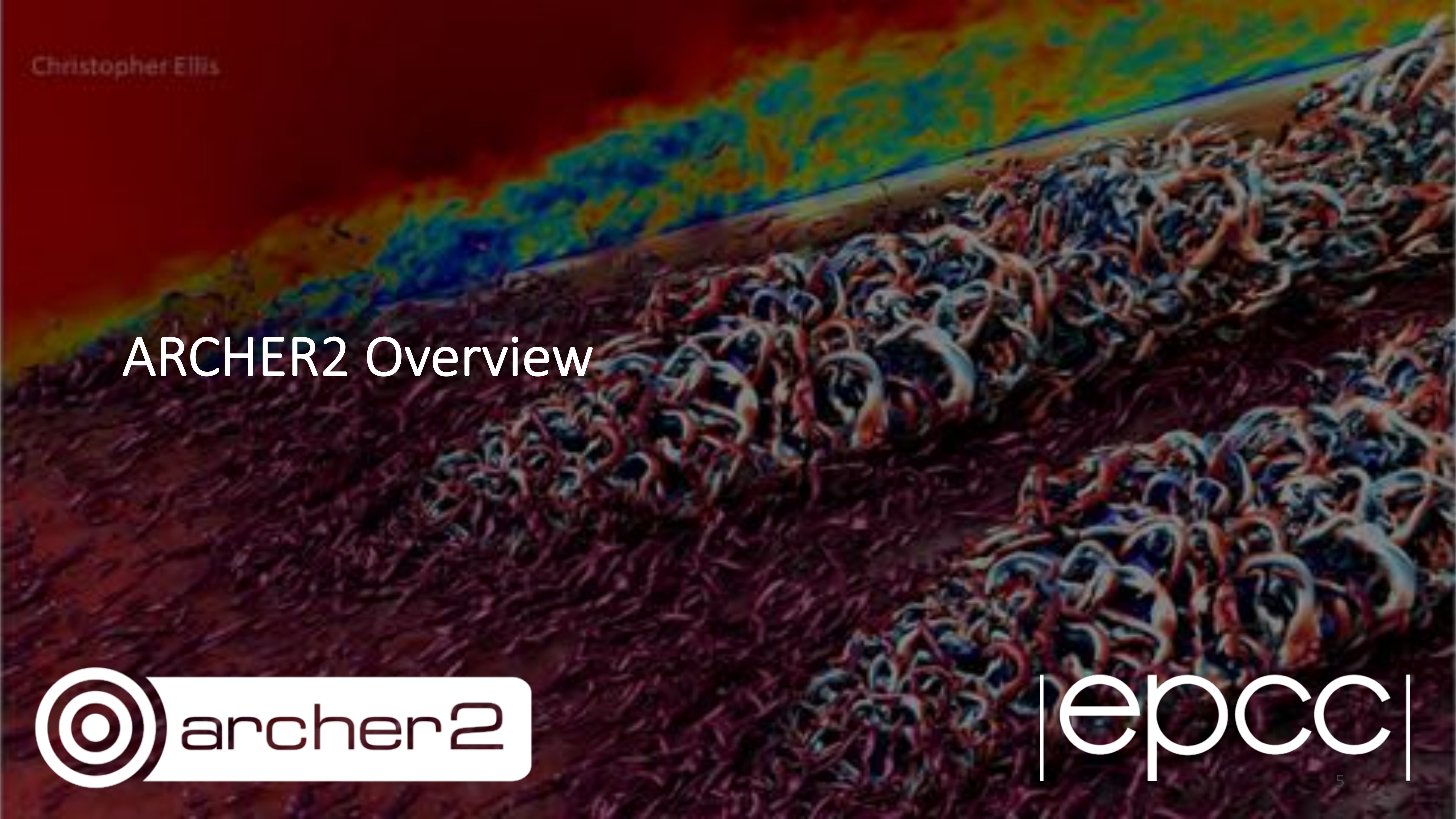
epcc

THE UNIVERSITY of EDINBURGH

Hewlett Packard Enterprise

Introduction

- ARCHER2 is the latest UK National Supercomputing Service
  - Based around a large (750,000 core) HPE Cray EX system
- Compare the performance of OpenFabrics (OFI) and Mellanox UCX
  - Transport protocols that underly the MPI library on ARCHER2
  - Can be selected at runtime by users
- Motivation:
  - Provide good advice to users on which protocol to choose and when
  - Understand the performance characteristics of Slingshot interconnect
  - Identify problems and areas for improvement – particularly at large scale



ARCHER2 Overview

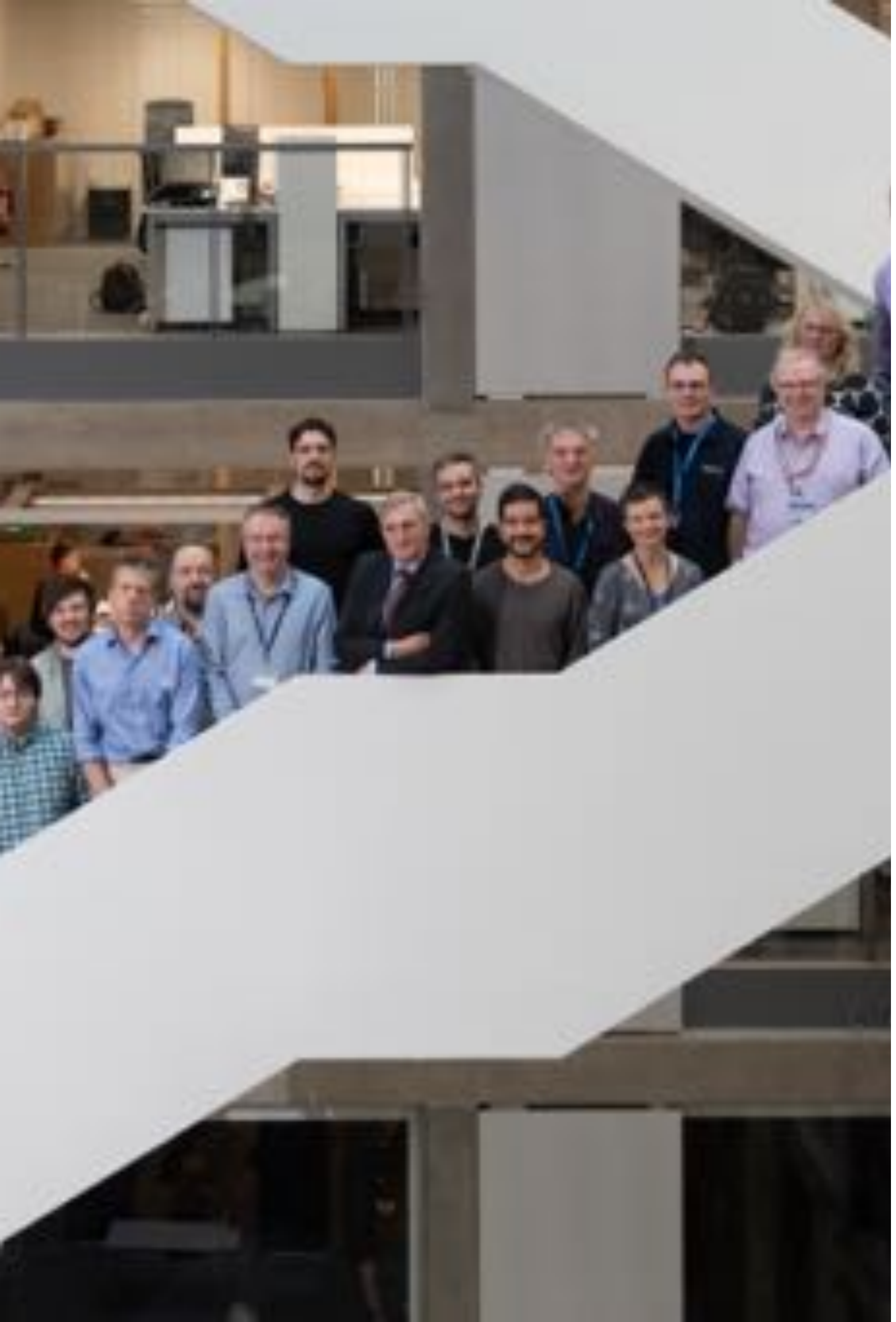
- UK National Supercomputing Service – based at EPCC at The University of Edinburgh
- Service designed to enable world-leading research for a wide range of research areas in the UK
- User base of over 3000 users
- Aim to have at least 10x the research throughput of its predecessor (ARCHER) and provide new capabilities for researchers

Application Type	Approx. % Use	Example Applications
Quantum Materials Modelling	40%	CASINO, CASTEP, CP2K, QE, VASP
Earth Systems Modelling	20%	Met Office UM, MITgcm, NEMO, WRF
Computational Fluid Dynamics	15%	OpenFOAM, Nektar++, OpenSBLI, Code_Saturne
Biomolecular Modelling	15%	GROMACS, NAMD
Classical Materials Modelling	5%	LAMMPS
Plasma Physics	3%	EPOCH, GS2, OSIRIS
Quantum Chemistry	2%	NWChem

- Huge range of software: top 10 codes ~50%, top 40 ~75%, 100s of others make up the rest

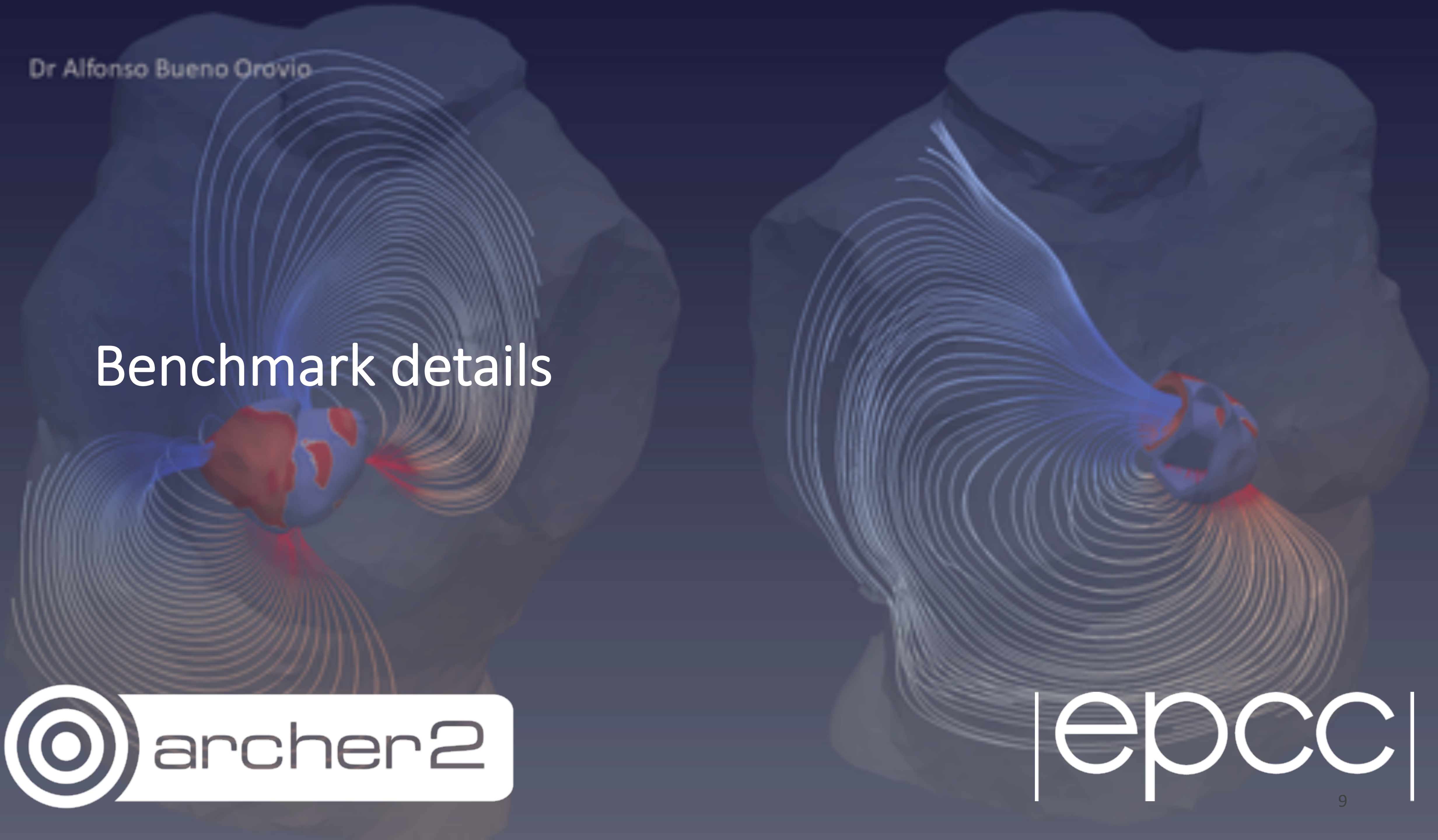
ARCHER2 system

- HPE Cray EX Supercomputer
- Hosted at EPCC, The University of Edinburgh
- 5,860 compute nodes (750,080 CPU compute cores)
- HPE Cray Slingshot 10 interconnect
- Compute nodes:
  - Dual socket AMD EPYC^{TM} 7742 (Rome), 64c, 2.25 GHz
  - 256 GIB / 512 GiB memory per node
  - Two 100 Gbps Slingshot interfaces per node
- 3x ClusterStor L300 Lustre file systems, each 3.6 PB
- 1 PB ClusterStor E1000F solid state storage
  - Not available at start of service
  - Will be available to users via Slurm BB directives
- 4x NetApp FAS8200A file systems, 1 PB total



ARCHER2 Service

- Comprehensive support for users from experts at EPCC and HPE
- Extensive training programme that is free to researchers
  - Wide range of courses from entry level to advanced
- Support to employ Research Software Engineers to improve codes
  - These can be RSEs in the community or provided by EPCC
- Outreach and engagement with the public and wider research community



Benchmark details

Full details

- Full details of the benchmarks available on Github:
  - https://github.com/ARCHER2-HPC/performance_ofi-ucx
- Includes:
  - Benchmark descriptions and input decks
  - Job submission scripts
  - Build instructions and versions of software used
  - Raw results and outputs from benchmark runs
  - Output from profiles on different node counts (CrayPAT-lite)
  - Analysis scripts
- Multiple runs (typically 5) of each benchmark at each point.
  - Best result used for analysis

Benchmarks

- Synthetic:
  - Ohio State University (OSU) MPI micro-benchmarks
    - https://mvapich.cse.ohio-state.edu/benchmarks/
- Applications:
  - CASTEP: plane wave DFT materials science
    - https://www.castep.org
  - CP2K: localised basis set DFT materials science
    - https://www.cp2k.org
  - GROMACS: classical biomolecular modelling
    - https://www.gromacs.org
  - NEMO: oceanographic modelling code
    - https://www.nemo-ocean.eu/
  - OpenSBLI: CFD via domain specific language
    - https://opensbli.github.io/
  - VASP: DFT materials science
    - https://vasp.at/

Application benchmarks: key comms. routines

- CASTEP:
  - Large DNA benchmark: from CASTEP developers, http://www.castep.org/CASTEP/DNA
  - MPI_Alltoallv
- CP2K:
  - LiH exact exchange: from CP2K, https://www.cp2k.org/performance#lih-hfx
  - ScaLAPACK/BLACS – generally translates to MPI_Allreduce
- GROMACS:
  - benchPEP (12M atoms, peptides in water): from Dept. of Theoretical and Computational Biophysics, Max Planck Institute for Multidisciplinary Sciences, Göttingen, https://www.mpinat.mpg.de/grubmueller/bench
  - Point-to-point MPI communications
- NEMO:
  - Double gyre system in the northern hemisphere (GYRE_PISCES benchmark): from NEMO, https://forge.ipsl.jussieu.fr/nemo/wiki/Users/ReferenceConfigurations/GYRE_PISCES
  - Detached mode (2 I/O servers and 48 ocean tasks per node).
  - MPI_Allgather, MPI_Allreduce, MPI_Alltoallv
- OpenSBLI:
  - Taylor-Green vortex calculation, from UK Turbulence Consortium
  - Point to point MPI communications
- VASP:
  - TiO_2 supercell, pure DFT, from Materials Chemistry Consortium
  - MPI_Alltoallv, MPI_Allreduce

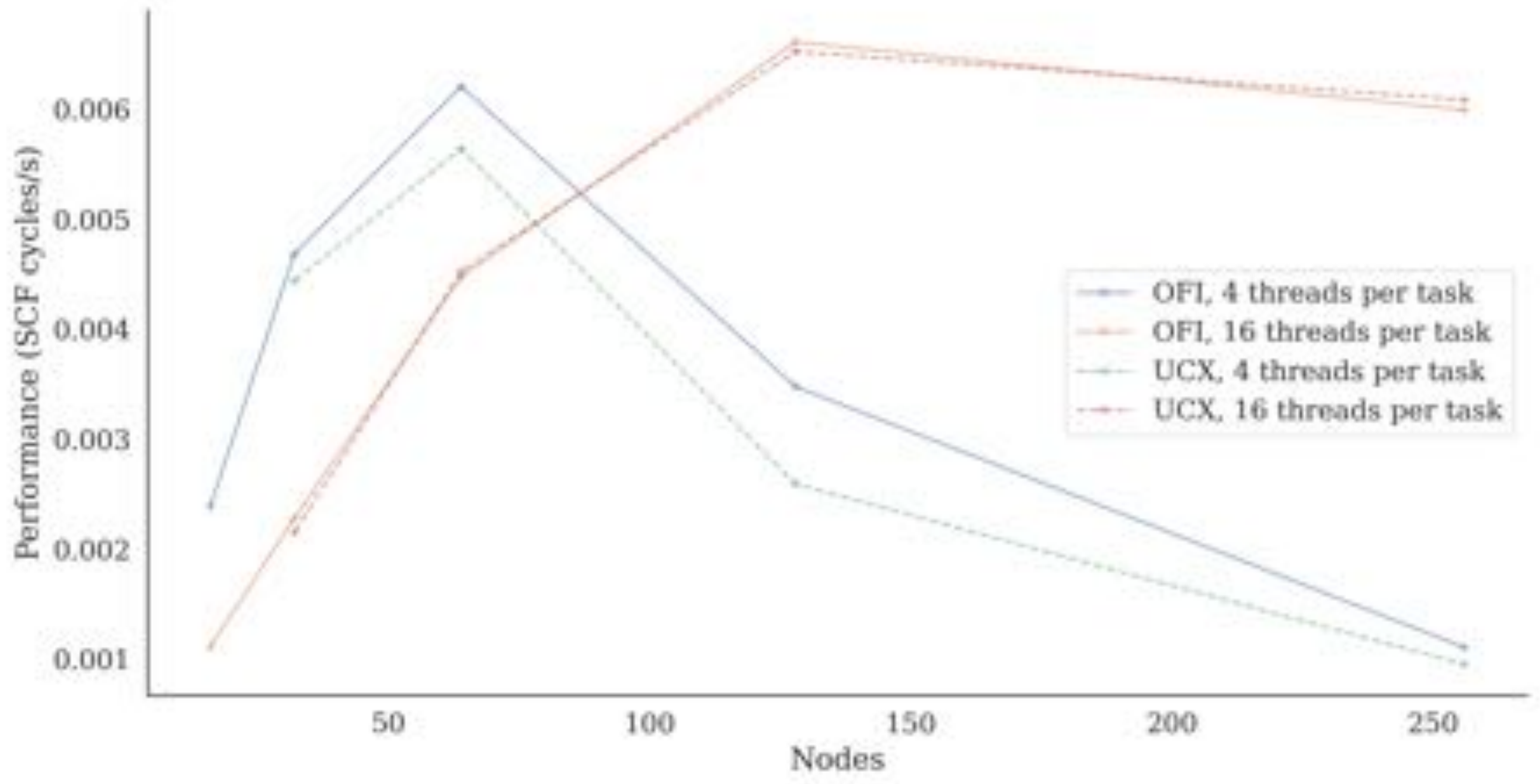
Build environment summary

- System software: Shasta 1.4
- HPE Cray Programming Environment (CPE) 21.09
  - HPE Cray MPICH 8.1.9
  - OpenFabrics: 1.11.0.4.71 (libfabric)
  - UCX: 1.9.0 (from Mellanox HPCX 2.7.0)
- Compilers:
  - GCC 11.2.0: OSU MPI, CP2K, GROMACS, NEMO, VASP
  - CCE 12.0.3: OpenSBLI, CASTEP, NEMO
- Other CPE components
  - FFTW 3.3.8.11: CASTEP, CP2K, VASP
  - XIOS 2.5: NEMO

Results and analysis

CASTEP

All use 128 cores per node

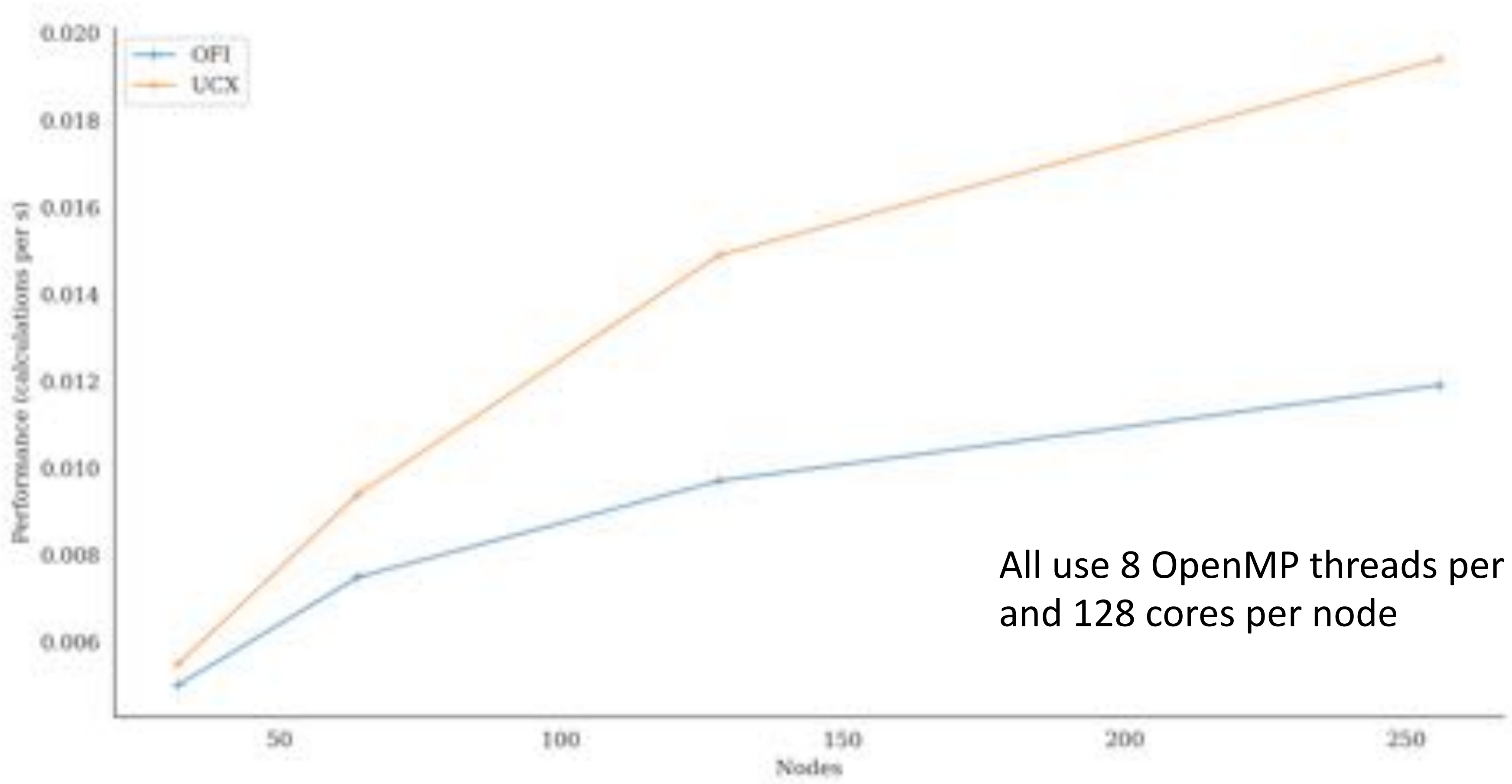


CASTEP

- OFI performs better with fewer threads per task, but with more threads MPI's reduced importance allows OFI and UCX to perform very similarly.
- OFI and UCX profiles are similar to one another independent of number of nodes.

Nodes	OFI	UCX
32	SCF time: 2568.39s 50.3% MPI 42.1% Alltoallv 5.0% Scatter 2.6% Allreduce	SCF time: 2867.39s 55.6% MPI 43.1% Alltoallv 7.7% Scatter 4.1% Allreduce
128	SCF time: 4020.28s 91.8% MPI 85.4% Alltoallv 3.9% Scatter 1.9% Allreduce	SCF time: 5764.28s 94.3% MPI 86.8% Alltoallv 4.7% Scatter 2.5% Allreduce

CP2K: performance



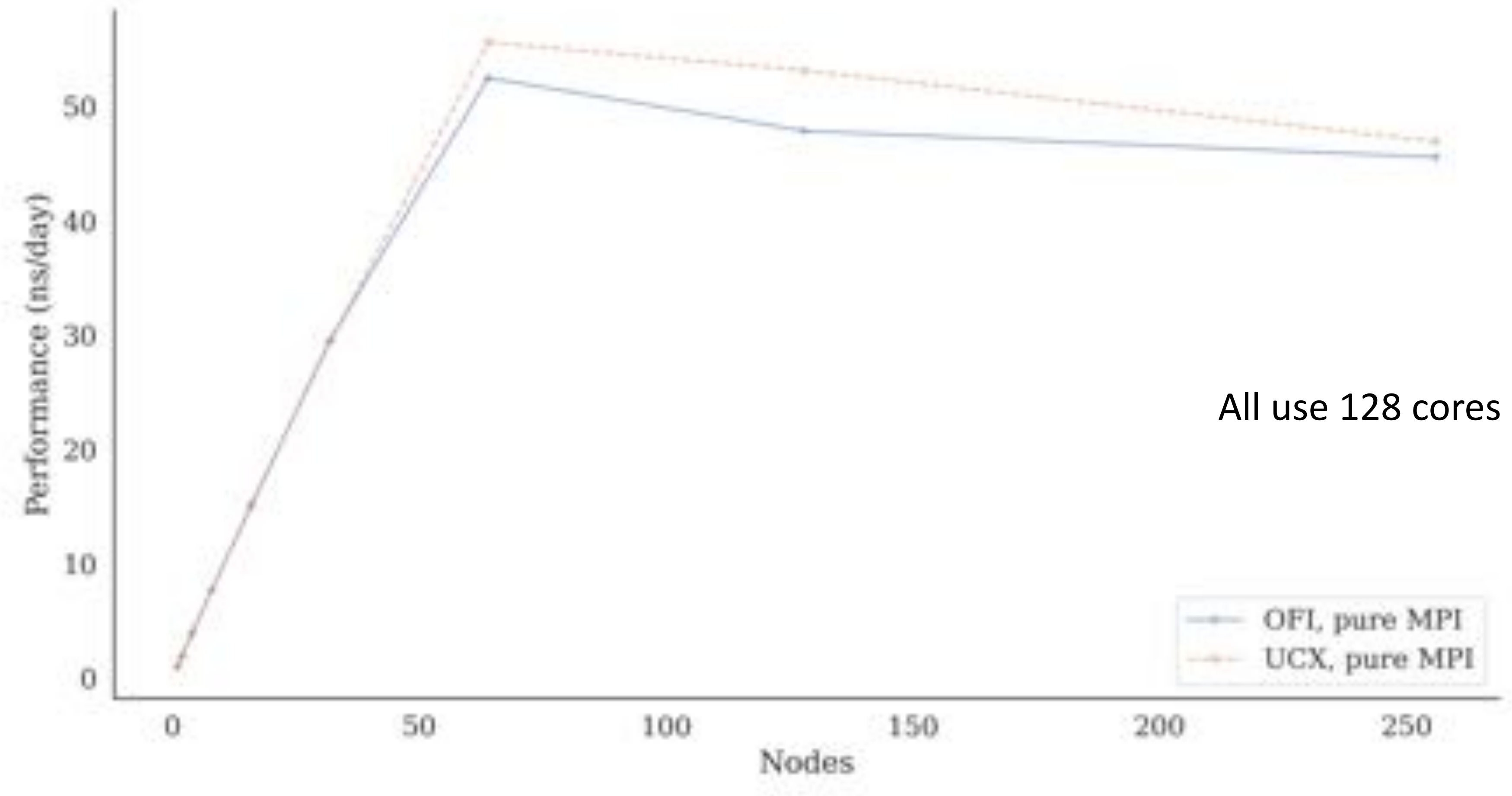
All use 8 OpenMP threads per MPI process and 128 cores per node

CP2K: profile

- Much better scaling with UCX
- MPI_Alltoallv appears in profiles for OFI but not UCX

Nodes	OFI	UCX
64	Runtime: 419.0s 16.3% MPI 5.5% Alltoallv 3.4% Waitall 3.0% Barrier 2.3% Allreduce	LOOP+: 421.0s 11.8% MPI 3.5% Waitall 3.2% Barrier 2.8% Allreduce
128	Runtime: 103.082s 19.8% MPI 6.4% Barrier 5.3% Alltoallv 2.7% Allreduce 2.6% Waitall 1.1% Isend	Runtime: 67.164s 14.9% MPI 6.6% Barrier 3.3% Allreduce 3.0% Waitall

GROMACS

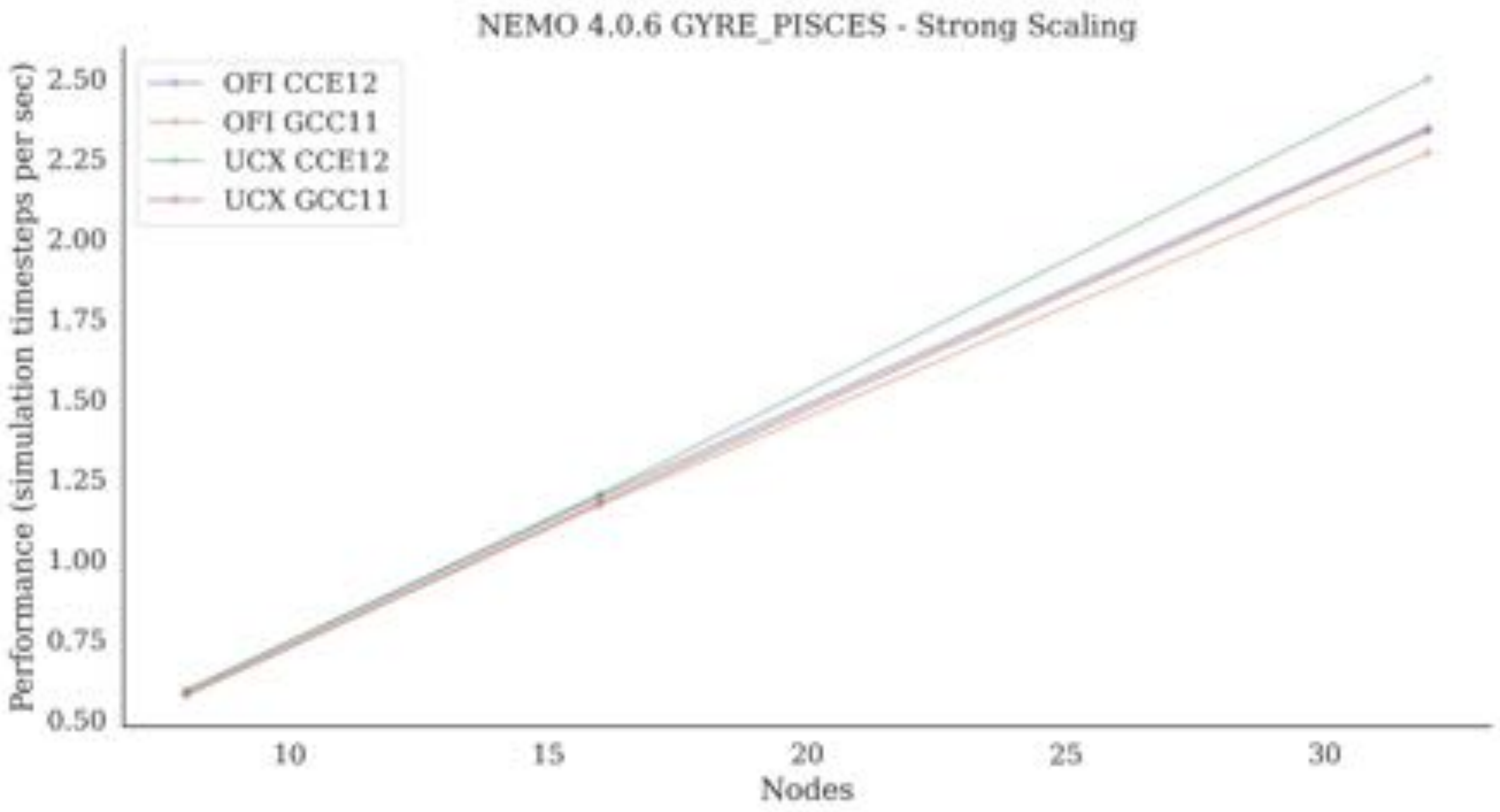


All use 128 cores per node

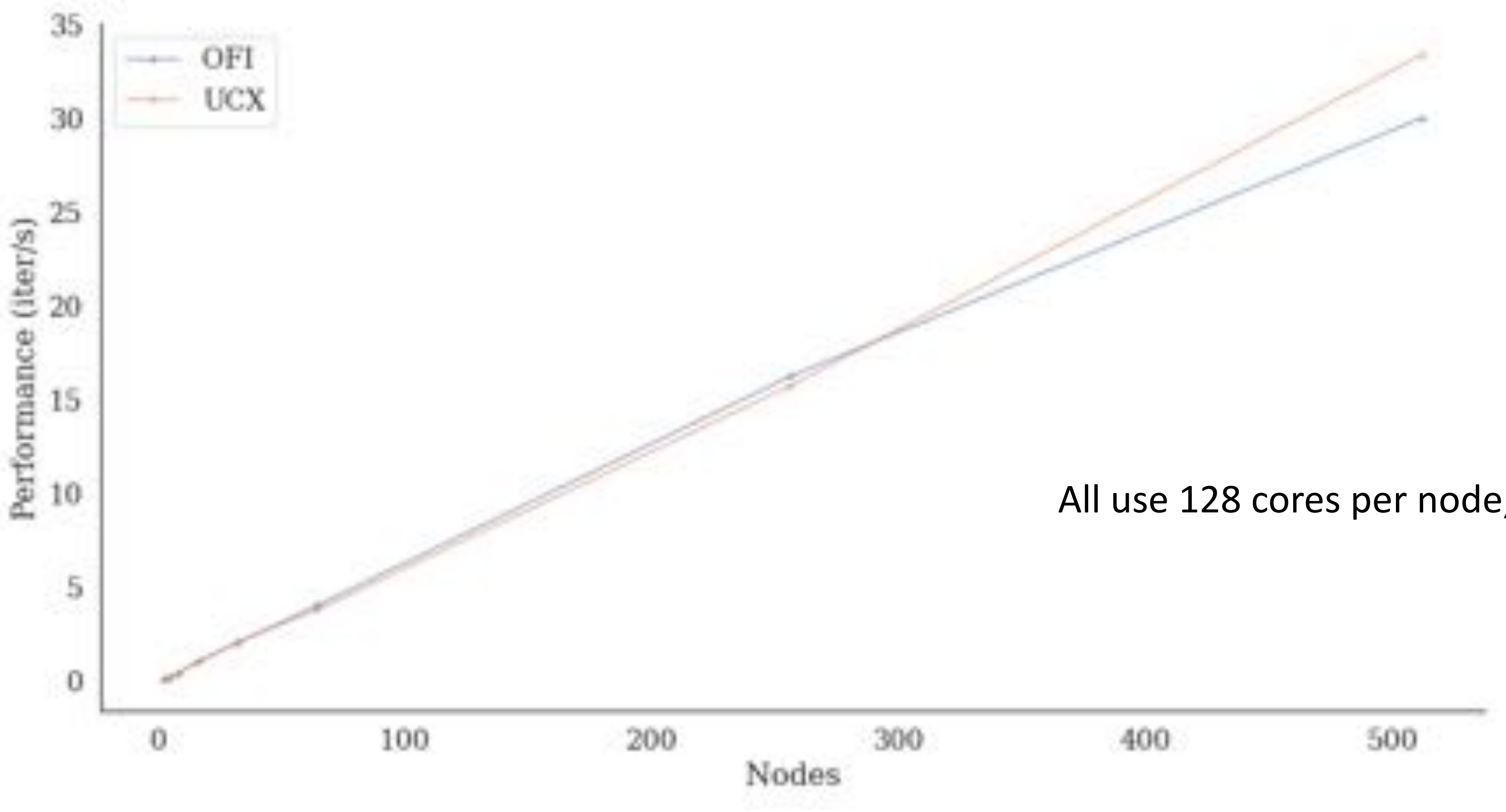
GROMACS profiling

- Performance very similar between OFI and UCX at lower node counts. At larger node counts, UCX is slightly faster.
- Marginally better performance with UCX at 8 nodes pure MPI due to less time spent in MPI_Bcast and MPI_Recv.
- Using 64 nodes and pure MPI, MPI_Scatterv takes most of the time with OFI, but doesn't appear for UCX.

Nodes	OFI	UCX
8	Runtime: 228.219s 31.4% MPI 12.7% Waitall 5.0% Sendrecv 3.4% Bcast 3.2% Recv 2.8% Alltoall	Runtime: 224.501s 27.1% MPI 14.4% Waitall 5.3% Sendrecv 2.8% Alltoall 1.9% Recv 1.1% Bcast
64	Runtime: 35.971s 88.9% MPI 58.4% Scatterv 21.2% Recv 2.3% Sendrecv 1.7% Comm_split 1.7% Alltoall	Runtime: 32.465s 48.1% MPI 11.2% Waitall 9.1% Sendrecv 6.2% Comm_split 6.0% Alltoall 5.7% Bcast


NEMO 4.0.6 GYRE_PISCES - Strong Scaling
OFI CCE12
OFI GCC11
UCX CCE12
UCX GCC11
Performance (simulation timesteps per sec)
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
10
15
20
25
30
Nodes

OpenSBLI: performance



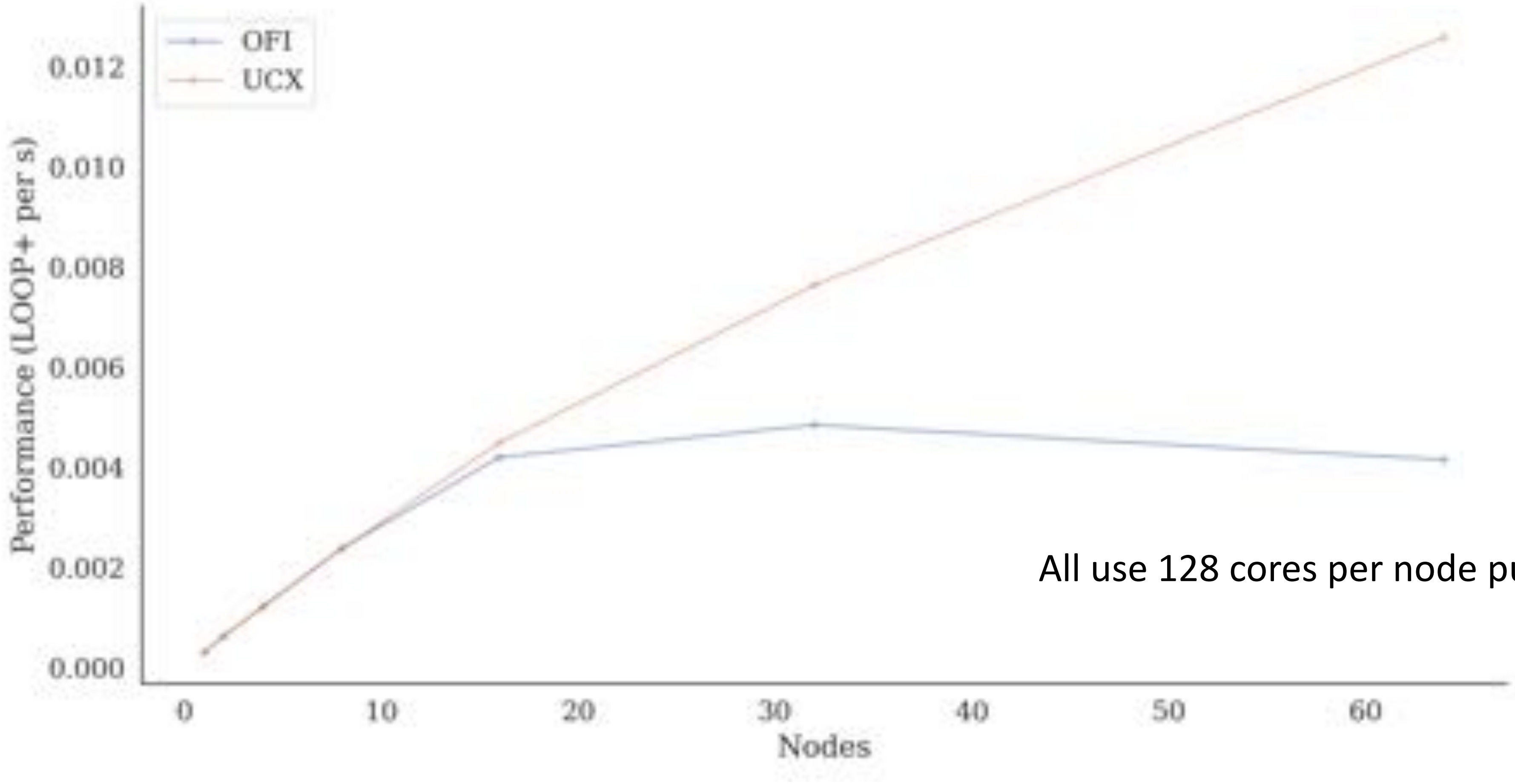
All use 128 cores per node, pure MPI

OpenSBLI: analysis

- Performance and scaling very similar for OFI and UCX
- Profiles shows similar timings between OFI and UCX
- MPI parts are based around point-to-point comms and are not critical to scaling
    - The MPI_Waitall calls are part of calculation initialization and can be reduced by running for longer

Nodes	OFI	UCX
32	Iteration: 0.483s 22.1% MPI 21.5% Waitall	Iteration: 0.475s 26.4% MPI 26.3% Waitall
256	Iteration: 0.061s 42.1% MPI 40.2% Waitall 1.0% Isend	Iteration: 0.058s 38.0% MPI 37.3% Waitall

VASP: performance



All use 128 cores per node pure MPI

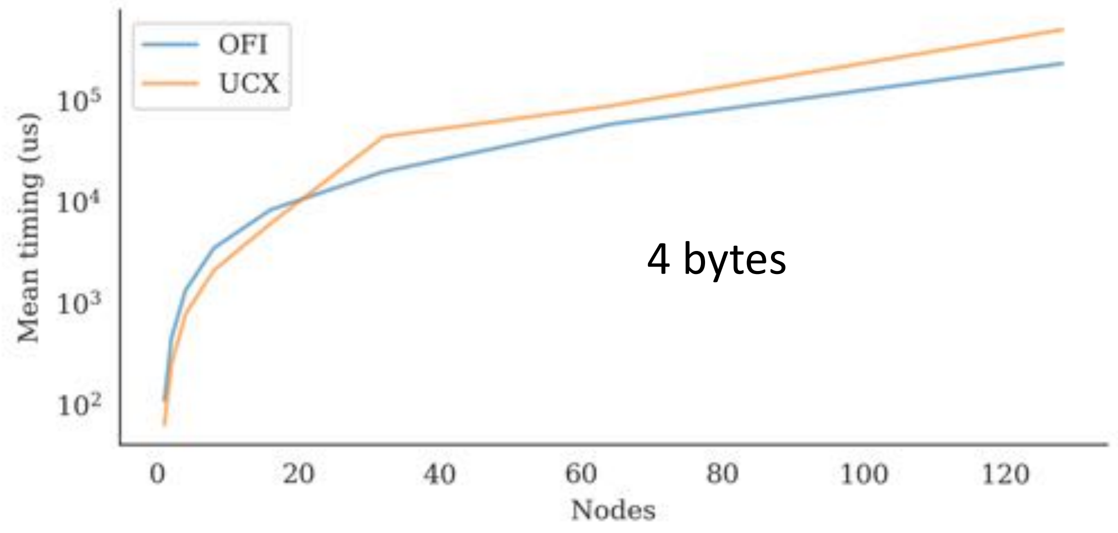
VASP: analysis

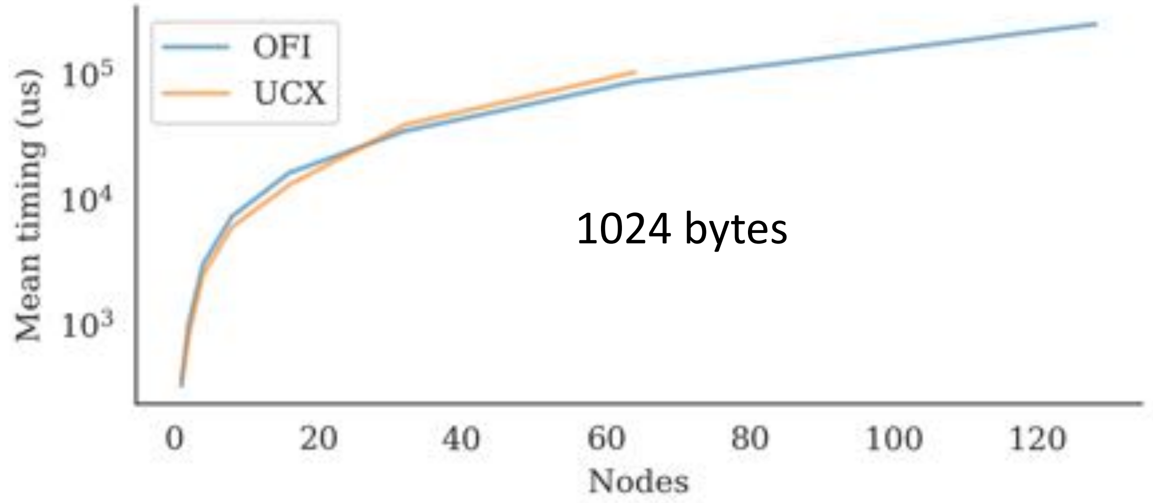
- Much better scaling with UCX version above 16 nodes
- For OFI, MPI_Alltoallv becomes the dominant routine as number of nodes increases
  - Same behaviour is not seen for UCX version

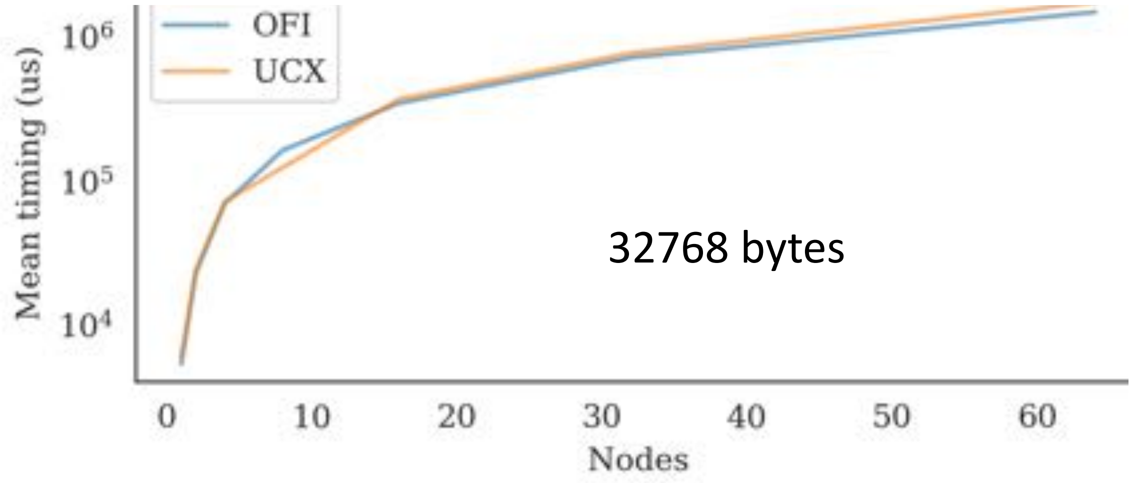
Nodes	OFI	UCX
8	LOOP+: 419.0s 27.7% MPI 8.3% Alltoallv 7.9% Bcast 5.1% Allreduce 4.6% Barrier	LOOP+: 421.0s 28.8% MPI 9.6% Bcast 5.2% Alltoallv 4.9% Barrier 4.6% Allreduce 4.5% Alltoall
64	LOOP+: 240.5s 60.5% MPI 43.2% Alltoallv 7.2% Bcast 5.6% Allreduce 4.0% Barrier	LOOP+: 79.4s 50.4% MPI 22.0% Bcast 10.2% Barrier 8.5% Allreduce 7.7% Alltoall 2.1% Alltoallv

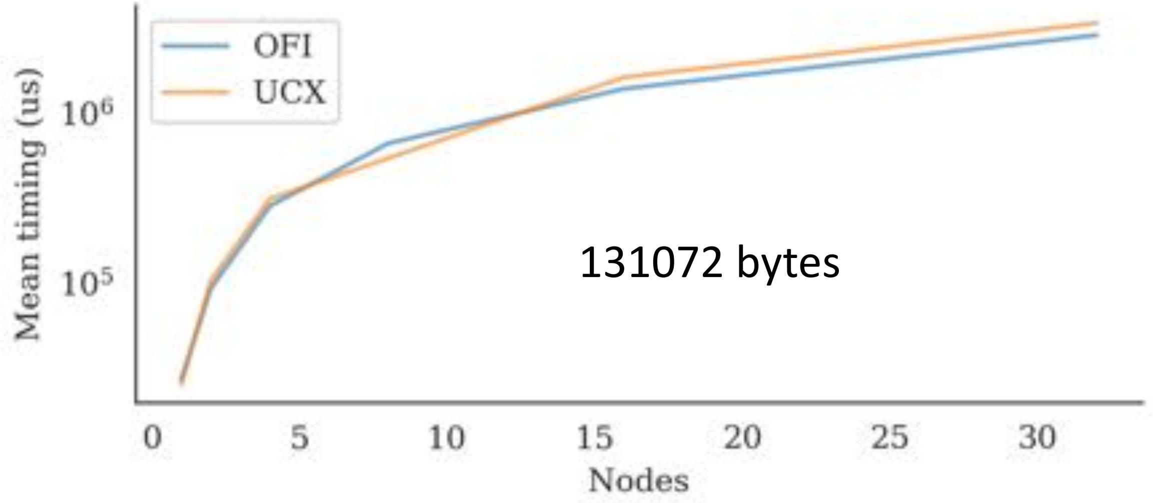
OSU micro-benchmarks: MPI_Alltoallv

All use 128 cores per node, pure MPI









OSU micro-benchmarks

- Basic MPI_Alltoallv benchmarking shows little difference between OFI and UCX performance across a range of message sizes
- Does not tally with the differences in performance seen for CP2K and VASP where reduced OFI scaling/performance seems linked to slower MPI_Alltoallv operations
- More detailed analysis needed to look at how the benchmarks are using MPI_Alltoallv and why the differences are not reflected in OSU micro-benchmarks

Next steps and summary

Next steps

- Further investigations of the characteristics of MPI_Alltoallv use in application benchmarks to better understand the performance issues for OFI
  - Can we find a synthetic benchmark that demonstrates the issues?
  - What is the difference between Alltoallv in CASTEP and OSU micro-benchmarks compared to CP2K and VASP?
- Extend to larger node counts
- Repeat tests as system software is updated (Shasta version, Slingshot software, MPI library)

Summary

- The choice of the OpenFabrics or UCX transport protocol can have a large effect on application performance
- Particularly large effects have been observed for benchmarks that use MPI_Alltoallv collective operations
  - Using UCX typically gives much better performance/scaling than OFI in these cases
  - The effect is not seen in OSU MPI_Alltoallv micro-benchmarks
- Benchmarks that rely on point-to-point MPI routines show much smaller variation in performance between OFI and UCX

https://github.com/ARCHER2-HPC/performance_ofi-ucx