CSE Benchmarking

Andy Turner, EPCC
a.turner@epcc.ed.ac.uk

Reusing this material

archer

epcc

THE UNIVERSITY OF EDINBURGH

www.epcc.ed.ac.uk
www.archer.ac.uk

Overview

- ARCHER Scaling Benchmarks
  - Motivation
  - Selection process
  - Application benchmark results and conclusions
- KNL Performance Comparisons
  - Motivation
  - Applications covered
  - Results and conclusions
- Parallel I/O benchmarking
  - Brief overview
  - Results and conclusions

ARCHER Scaling Benchmarks

Acknowledgements

- Contributors:
  - Gordon Gibb, EPCC
  - Elena Breitmoser, EPCC
  - David Scott, EPCC
  - Iain Bethune, EPCC (now Hartree)
  - Alan Gray, EPCC (now NVIDIA)
- Thanks:
  - ARCHER Scientific Consortia members
  - Louise Tillman, EPSRC

Motivation

- Update ARCHER benchmarks to represent current and future use
- Exercise HPC systems at scale with real applications and use cases
- Involve user community in proposing benchmarks and providing cases
- Compare different HPC systems around the UK (and beyond)
- Provide information to users:
  - Measured performance variation
  - Reference application profiles
  - Effect of new versions, compilers and/or libraries

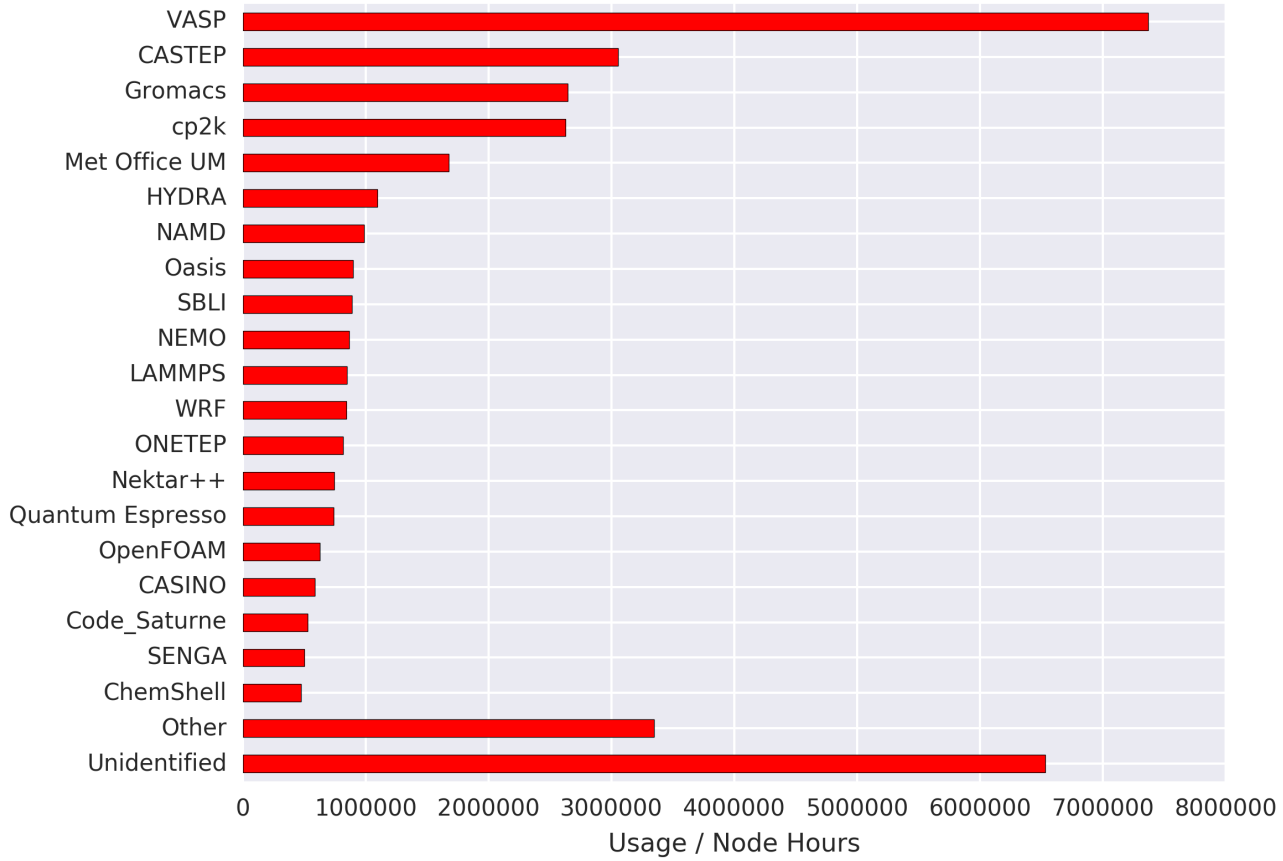
Selection Process

- Analyse current use
- Propose to research councils and major user groups
- Gather feedback and produce final proposal

Previous Benchmarks

- CASTEP: Periodic electronic structure (Fortran, MPI+OpenMP)
- CP2K: Periodic electronic structure (Fortran, MPI+OpenMP)
- DL_POLY: Classical molecular mechanics (Fortran, MPI+OpenMP)
- SENGA: CFD with combustion (Fortran, MPI)
- Met Office UM: Climate modelling (Fortran, MPI+OpenMP)

Current Use



- CASTEP, CP2K, UM still high usage
- SENGA low usage (< 2%)
- DL_POLY < 1% usage

- Biomolecular simulation are heavily used (GROMACS/NAMD)
- C++ compilers should be stressed by benchmarks

archer

epcc

THE UNIVERSITY OF EDINBURGH

Feedback from user groups

- CFD community proposed OpenSBLI as the application that will be most representative of future work
  - Tests domain-specific language (DSL) setup
- Materials modelling community asked for both CASTEP and CP2K to be kept as they operate in different ways
- Climate and ocean modelling communities asked for coupled model (OASIS3+UM+NEMO) to replace UM
  - Benchmark should include I/O component of application
- Accepted that GROMACS should replace DL_POLY
  - Stresses C++ compiler and represents biomolecular simulation community

Benchmark Set

- CASTEP
  - DNA, 1536 atoms in large simulation cell
- CP2K
  - Bulk LiH, hybrid functional
- OpenSBLI
  - Taylor-Green vortex on a 1024x1024x1024 grid
- GROMACS
  - 42 million atom simulation
- OASIS3
  - Includes UM (atmosphere) and NEMO (ocean)
- See: http://www.archer.ac.uk/documentation/white-papers/benchmarks/UK_National_HPC_Benchmarks.pdf

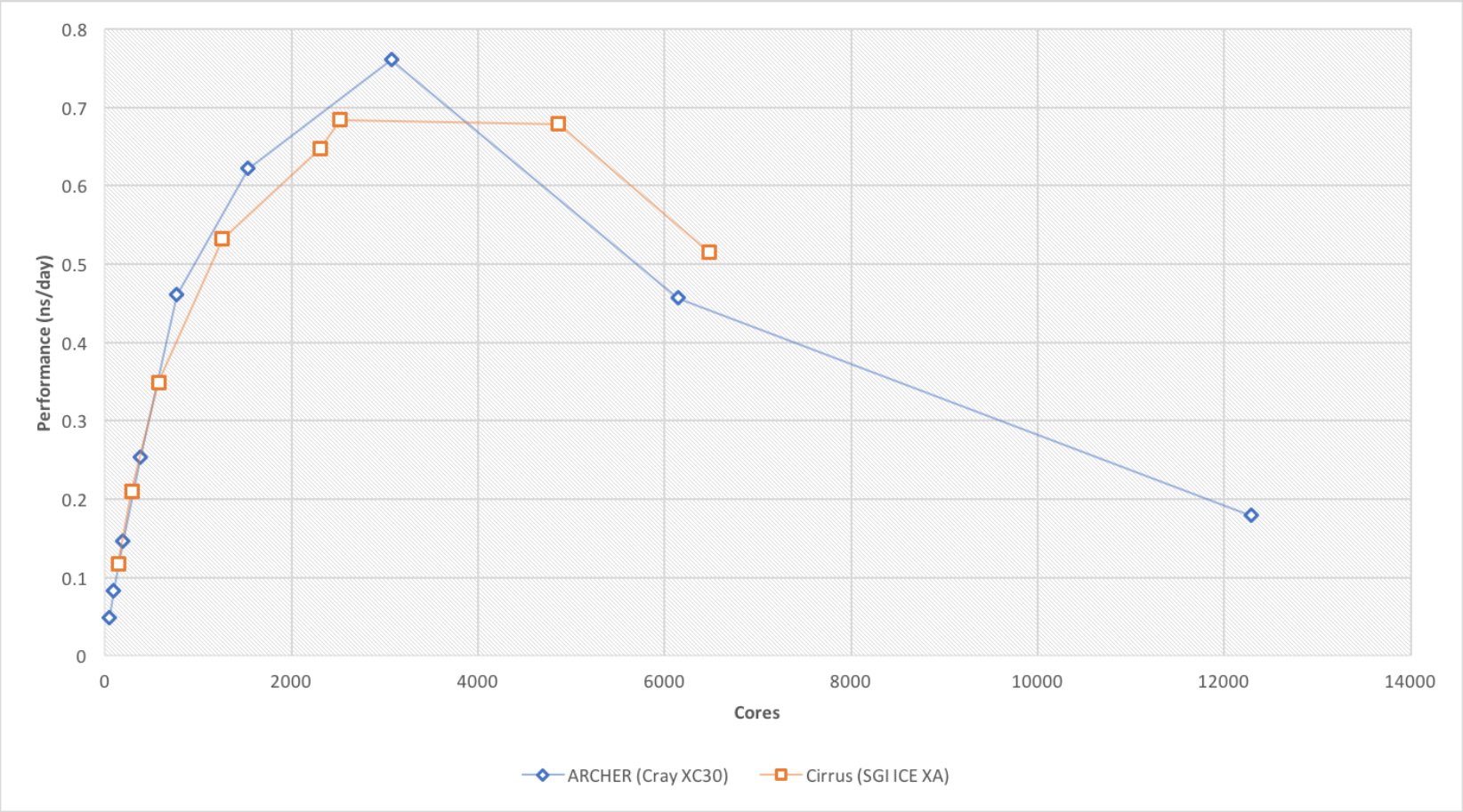
Systems benchmarked so far

- Cray XC30
- 2x 2.7 GHz 12-core Xeon “Ivy Bridge” per node
- 64 GiB DDR3 RAM per node
- Cray Aries interconnect, dragonfly topology

- HPE/SGI ICE XA
- 2x 2.1 GHz 18-core Xeon “Broadwell” per node
- 256 GiB DDR4 RAM per node
- Infiniband FDR interconnect, hypercube topology

Benchmark data: http://www.archer.ac.uk/community/benchmarks/archer/

Results: GROMACS

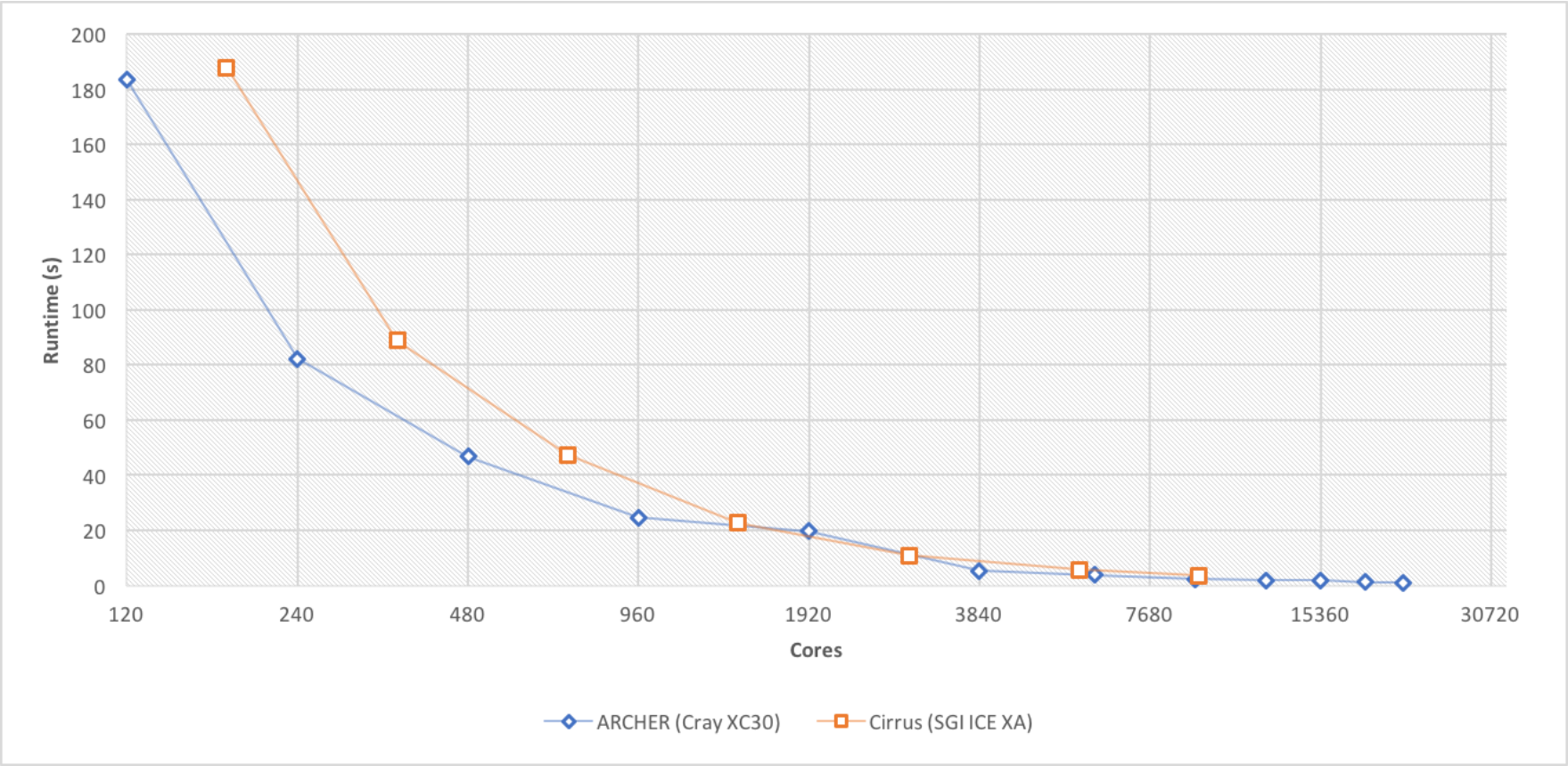


archer

epcc

THE UNIVERSITY OF EDINBURGH

Results: OpenSBLI

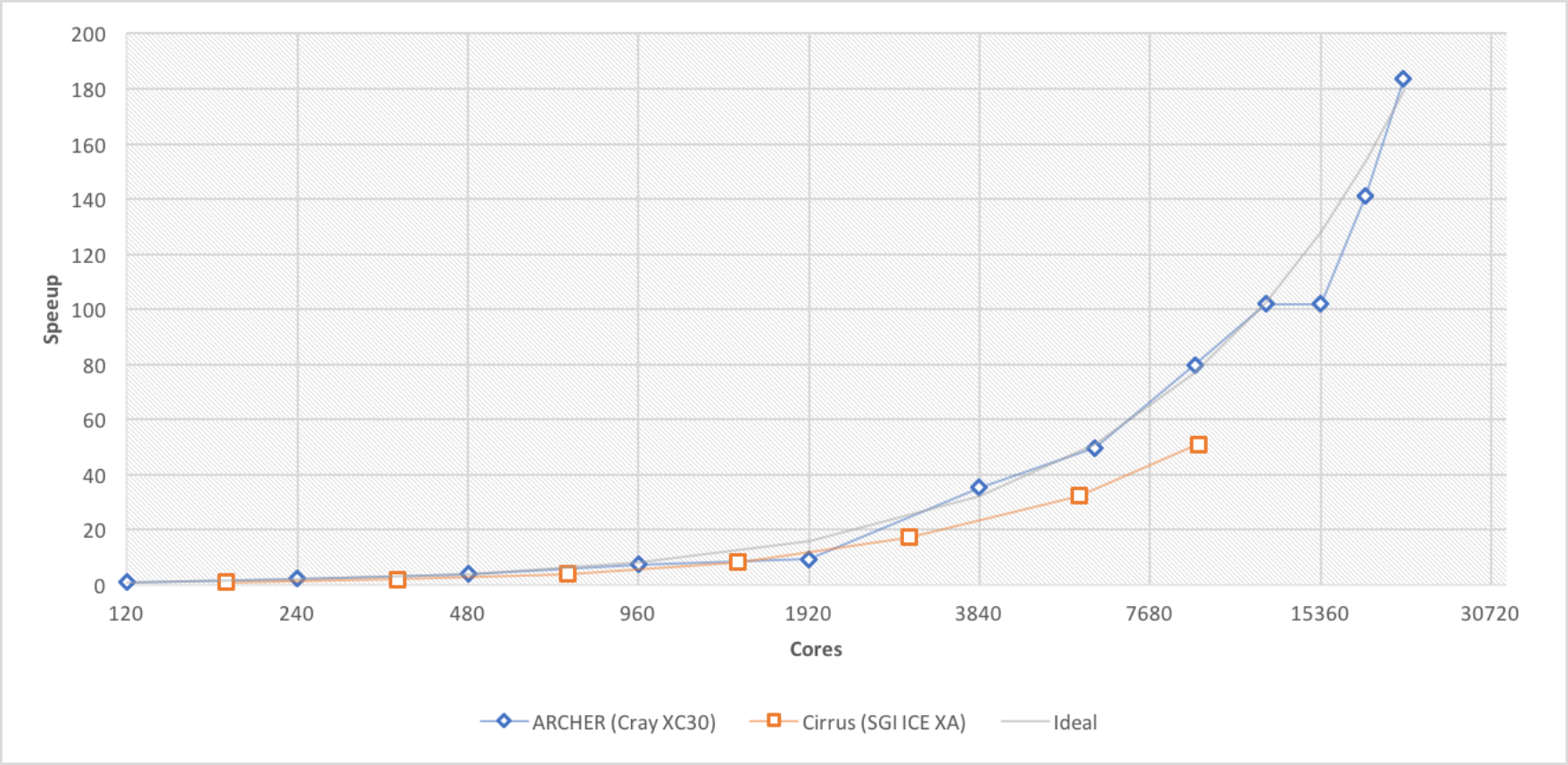


archer

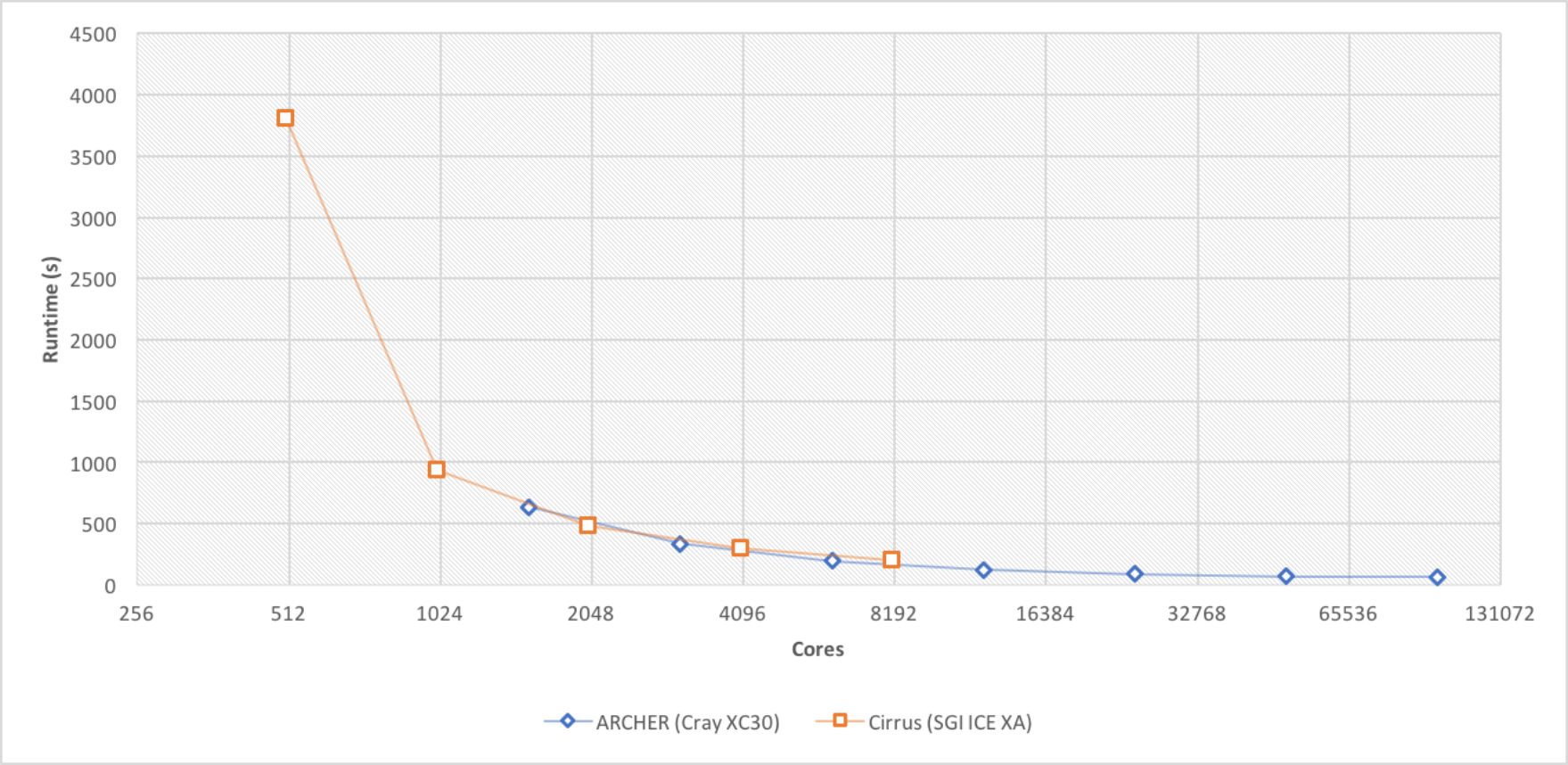
epcc

THE UNIVERSITY OF EDINBURGH

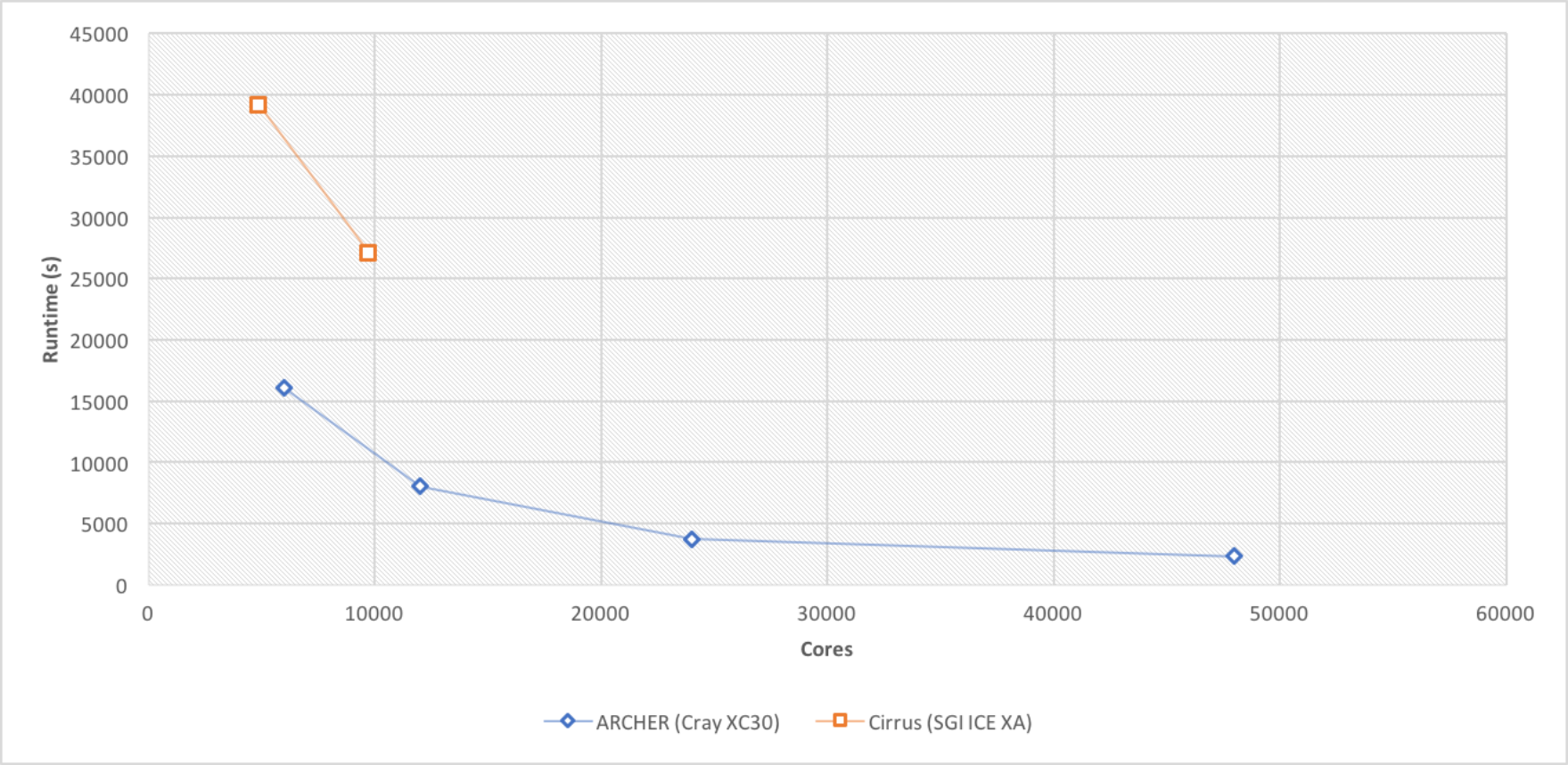
Results: OpenSBLI speedup



Results: CP2K



Results: CASTEP



archer

epcc

THE UNIVERSITY OF EDINBURGH

Conclusions so far

- Broadwell Xeon do not give obvious boost over Ivy Bridge
  - (At low core core counts most are memory bound.)
  - Increase in memory bandwidth is balanced by increased core count?
- Performance converges as core count increases
  - Applications become communication bound
  - Interconnect latencies and bandwidths are similar
- CASTEP shows very poor performance on Cirrus
  - Software issue with AlltoAll collective in SGI stack? Or Cray stack is very good at this? Or something else?
  - Currently investigating…

Next steps

- Continue investigations into CASTEP performance
  - Run on other Tier-2 Broadwell Xeon systems
  - Benchmark MPI collective performance
- Run OASIS benchmark
  - Only just become available
- Automatic, periodic runs of benchmarks
  - Track variations in performance
- Produce profiles of applications running benchmarks
  - Provide additional information for users and developers
  - Identify where differences lie between platforms

KNL Performance Comparison

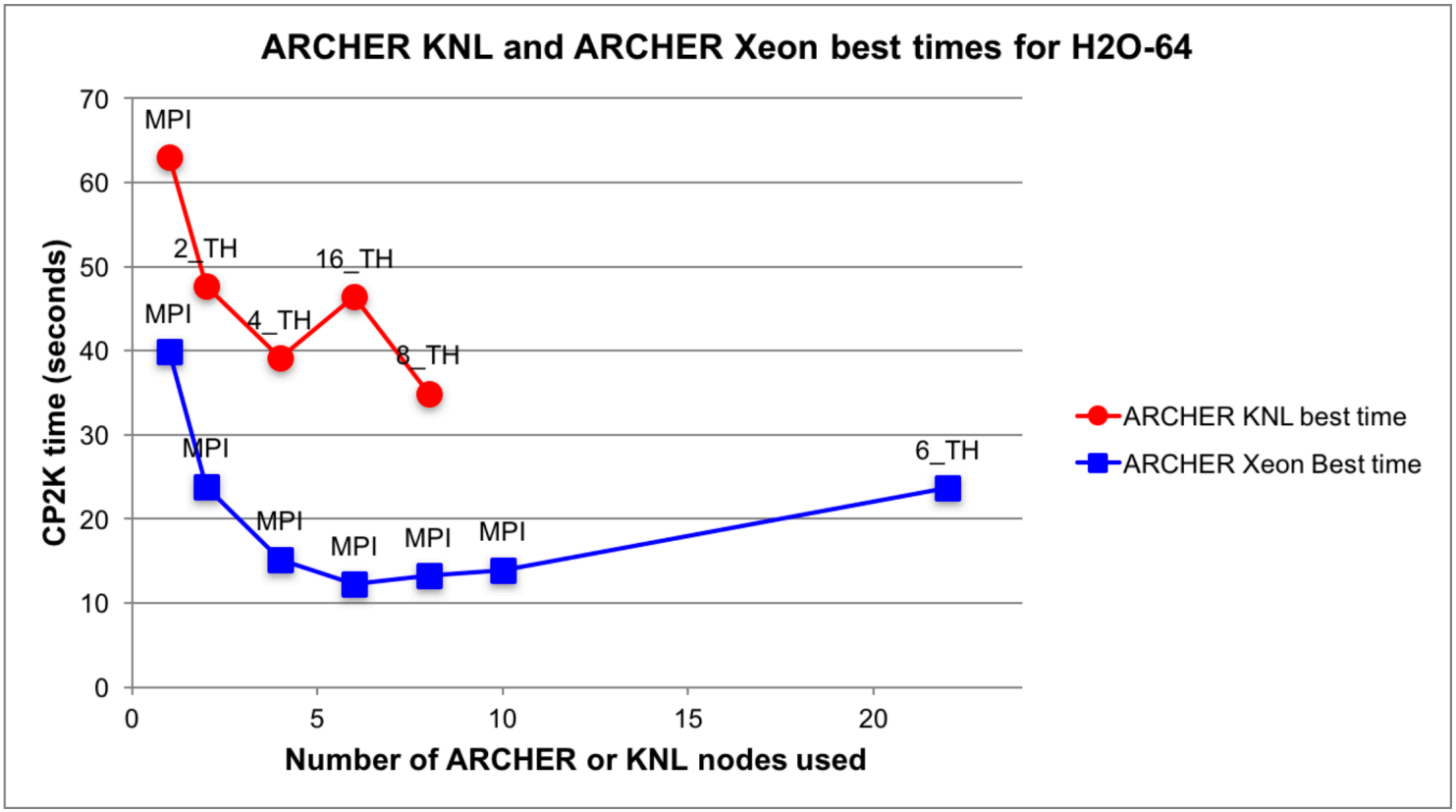
Acknowledgements

- CASTEP: Gordon Gibb, EPCC
- COSA: Adrian Jackson, EPCC
- CP2K: Fiona Reid, EPCC
- GaitSym: Bill Sellers, University of Manchester
- GS2: David Dickinson, University of York
- HLBM: George Barakos, University of Glasgow
- Incompact3d: Sylvain Laizet, Imperial College, London
- LAMMPS: Luis Cebamanos, Kevin Stratford, EPCC
- NAMD: Andy Turner, EPCC
- OpenSBLI: Luis Cebamanos, EPCC; Christian Jacobs, University of Southampton
- Quantum Espresso: Paolo Di Cono, King's College London
- R and SPRINT: Adrian Jackson, EPCC
- SENGA2: Neelofer Banglawala, EPCC
- UCNS3D: Panagiotis Tsoutsanis, Antonios Antoniadis, Cranfield University

Motivation

- Compare performance of ARCHER Xeon nodes to KNL nodes
  - For naïve porting effort rather than large amounts of work
  - Gives an idea of performance levels that real users would see
- Broad overview of different codes
- User engagement with KNL test facility
- Provide useful small benchmarks to compare range of technologies
  - Meets a different need to the scaling benchmarks previously mentioned – mostly comparing node-level performance here

Sample results: CP2K



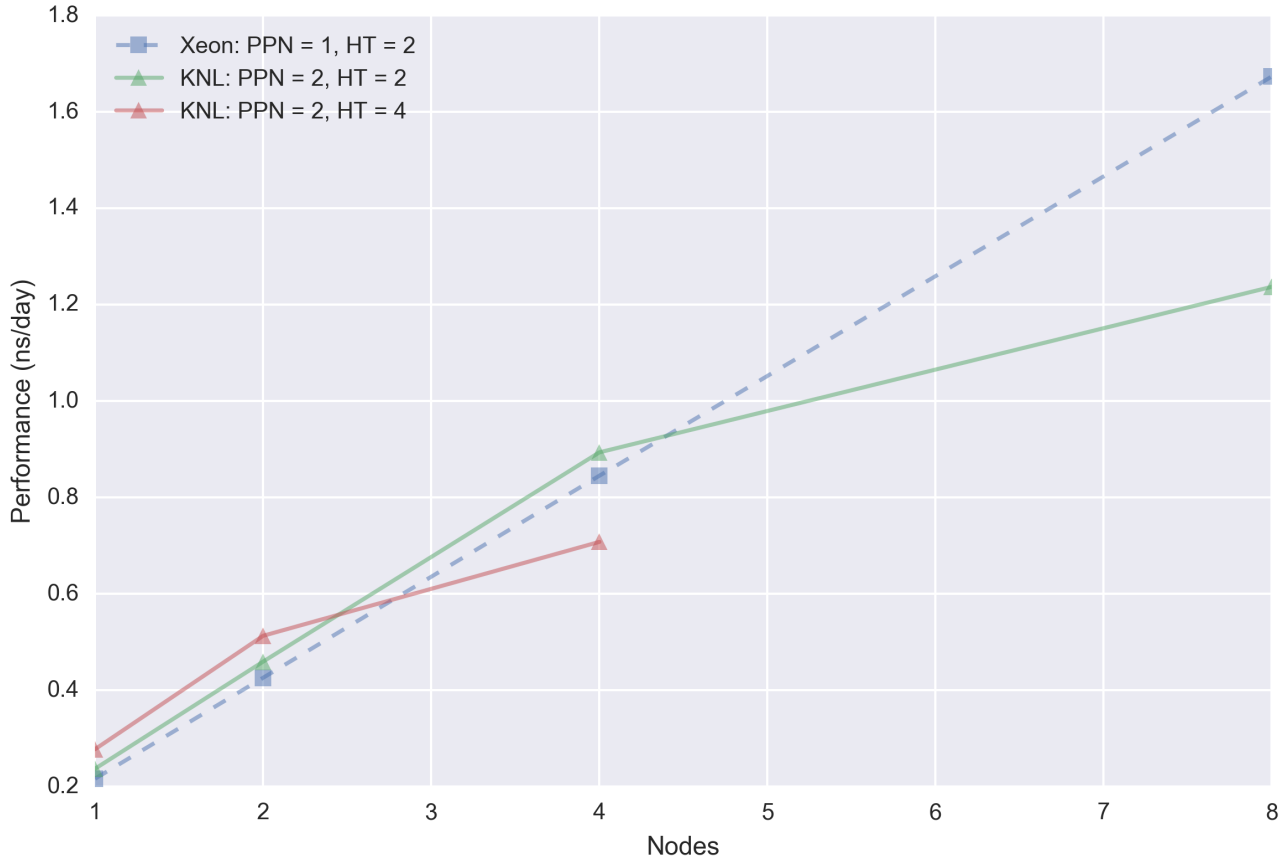
Fiona Reid, EPCC

archer

epcc

THE UNIVERSITY OF EDINBURGH

Sample results: NAMD

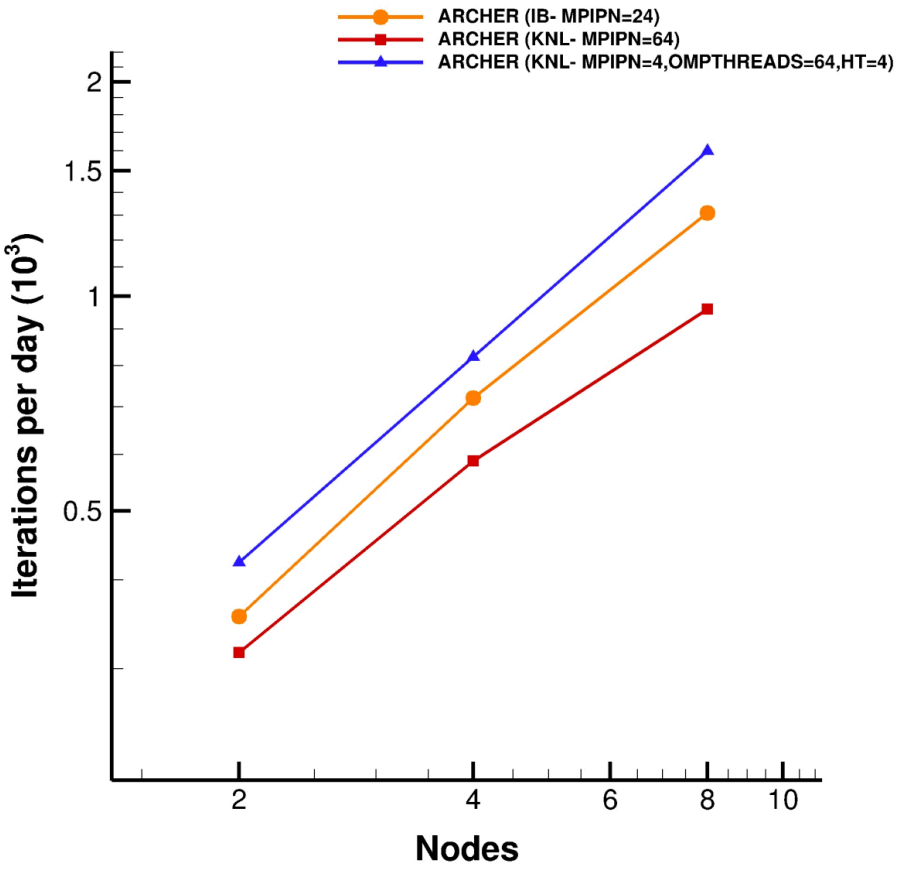


archer

epcc

THE UNIVERSITY OF EDINBURGH

Results: UCNS3D



Panagiotis Tsoutsanis,
Antonios Antoniadis,
Cranfield University

archer

epcc

THE UNIVERSITY OF EDINBURGH

Conclusions so far

- Portability quite simple for most applications
- Some applications do worse on KNL than Xeon…
- …some do about the same…
- …and some do better on KNL than Xeon
  - (note this is comparison to old Xeon processor)
- Applications that are memory bound and can fit well in HBM as cache see good performance
  - Without large amounts of application re-writing
- People report performance in lots of different ways!
  https://www.epcc.ed.ac.uk/blog/2017/05/11/archer-developers-and-presenting-performance

Next steps

- Profile selected applications on KNL and on Xeon
  - Where are differences and similarities?
- Compare performance on other Tier-2 systems
  - Compare different processors
  - Do different node architectures make a difference?

- Reports at: http://www.archer.ac.uk/community/benchmarks/archer-knl/

Parallel I/O benchmarking

Acknowledgements

- Contributors:
  - Dominic Sloan-Murphy, EPCC
  - David Henty, EPCC
  - Harvey Richarson, Cray
- Thanks:
  - Lydia Heck, DiRAC COSMA Durham
  - Bryan Lawrence, JASMIN

Motivation

- I/O performance is becoming more critical for HPC application performance as applications scale up
  - Many applications now have an I/O-bound phase
- What is the maximum performance you can expect from the ARCHER parallel file systems in production?
  - Compared to other HPC parallel I/O setups?
- What are the best file layouts and Lustre striping settings for different scenarios?
- How do MPI-IO, NetCDF and HDF5 write performance compare?
  - …and how do they compare to naïve file-per-process?

Benchmarks

Benchmarks: benchio (SSF)

- Simple Fortran program:
  - Writes 3D distributed dataset to single shared file (SSF)
  - MPI-IO, HDF5, NetCDF
- Advantages:
  - Small number of options: dataset size, number of processes, simple to understand what program is doing
  - Data distribution closer to many I/O-bound user applications
- Disadvantages:
  - Write performance only (read added soon)
- https://github.com/EPCCed/benchio

Why not use IOR?

- IOR is like Linpack
  - data decomposition designed to measure maximum IO bandwidth
  - imagine 64 data elements on 8 processes

  - IOR file: 8 large blocks of 8 contiguous items:

- benchio uses more realistic (but still simple) decomposition
  - leads to surprisingly complicated IO patterns
- Imagine 4x4x4 grid split evenly across 8 processes (2x2x2)
  - benchio file contains multiple interleaved small blocks of 2 items

archer

epcc

THE UNIVERSITY OF EDINBURGH

Benchmarks: benchio_fpp (FPP)

- FPP = File Per Process
- Derived from benchio:
  - Each process writes to own Fortran binary file
  - No HDF5, NetCDF support yet
- Need to be careful to ensure that buffering is not used by writing large amounts of data per process
  - MPI-IO bypasses buffering so not a problem for SSF version

Systems and Setup

Systems

- ARCHER
  - Cray Sonexion Lustre
  - Theoretical peak bandwidth: 30 GiB/s
- COSMA5
  - DDN GPFS
  - Theoretical peak bandwidth: 20 GiB/s
- Also small-scale systems (results not included here)
  - RDF – DDN GPFS, single node
  - JASMIN – PANASAS, small process counts

Benchmark setup

- Single Shared File (SSF)
  - MPI-IO collective
  - 128 MiB written per process
  - Lustre stripe counts: 1 (unstriped), 4 (default), -1 (maximum)
  - Lustre stripe sizes: 1 MiB, 4 MiB, 8 MiB
- File Per Process (FPP)
  - Fortran binary write (STREAM)
  - 1024 MiB written per process (*i.e.* per file)
  - Lustre stripe counts: 1 (unstriped), 4 (default), -1 (maximum)
  - Lustre stripe sizes: 1 MiB

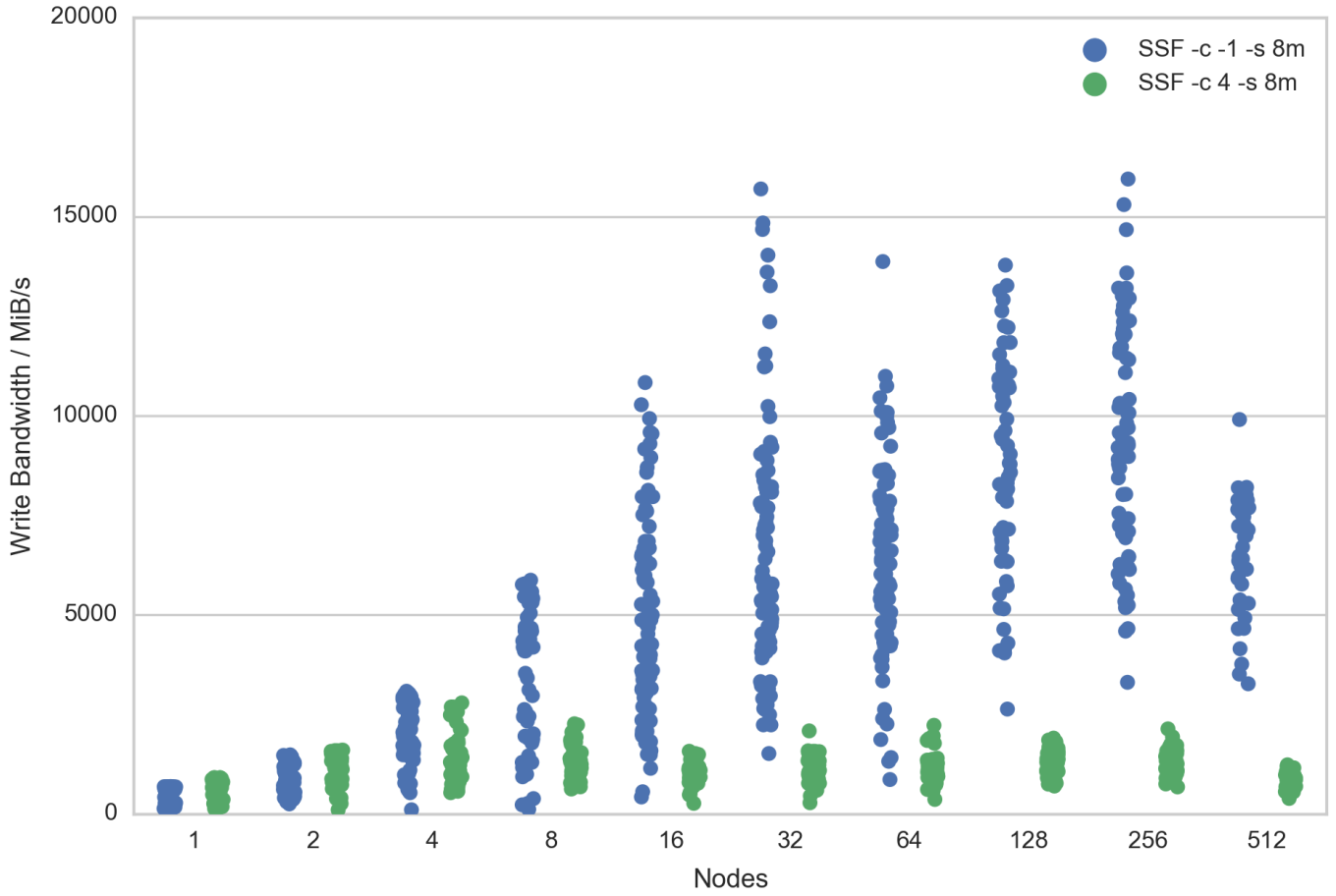
Benchmark setup (cont.)

- MPI-IO collective operations used in all cases
  - Previous experience shows that this is required for performance
- All compute nodes fully populated
  - This is typically how users use the system
- All runs performed during production
  - Subject to same contention as all users

Single Shared File (SSF)

MPI-IO Comparisons

SSF: ARCHER Lustre

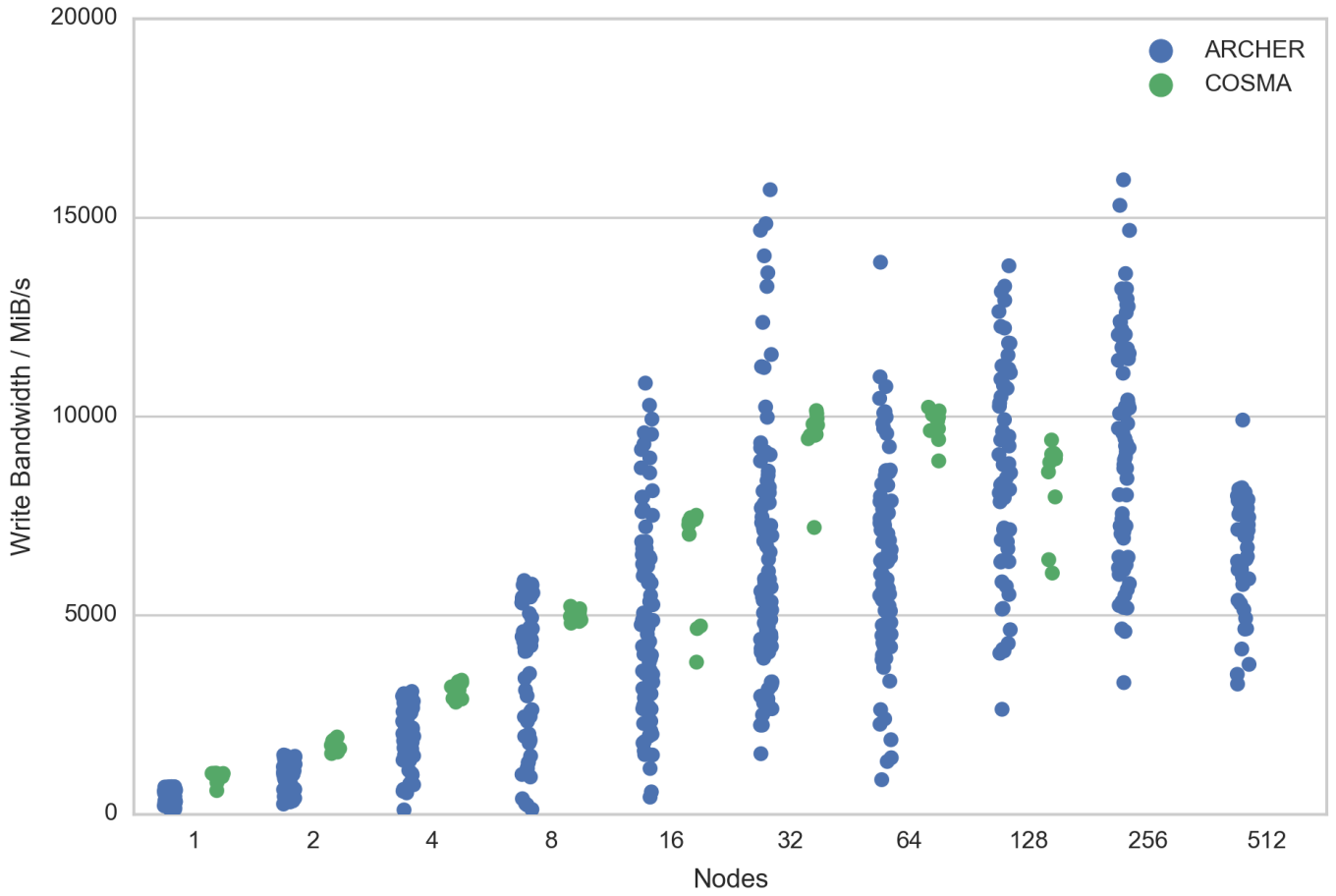


archer

epcc

THE UNIVERSITY OF EDINBURGH

SSF: Lustre/GPFS Comparison



archer

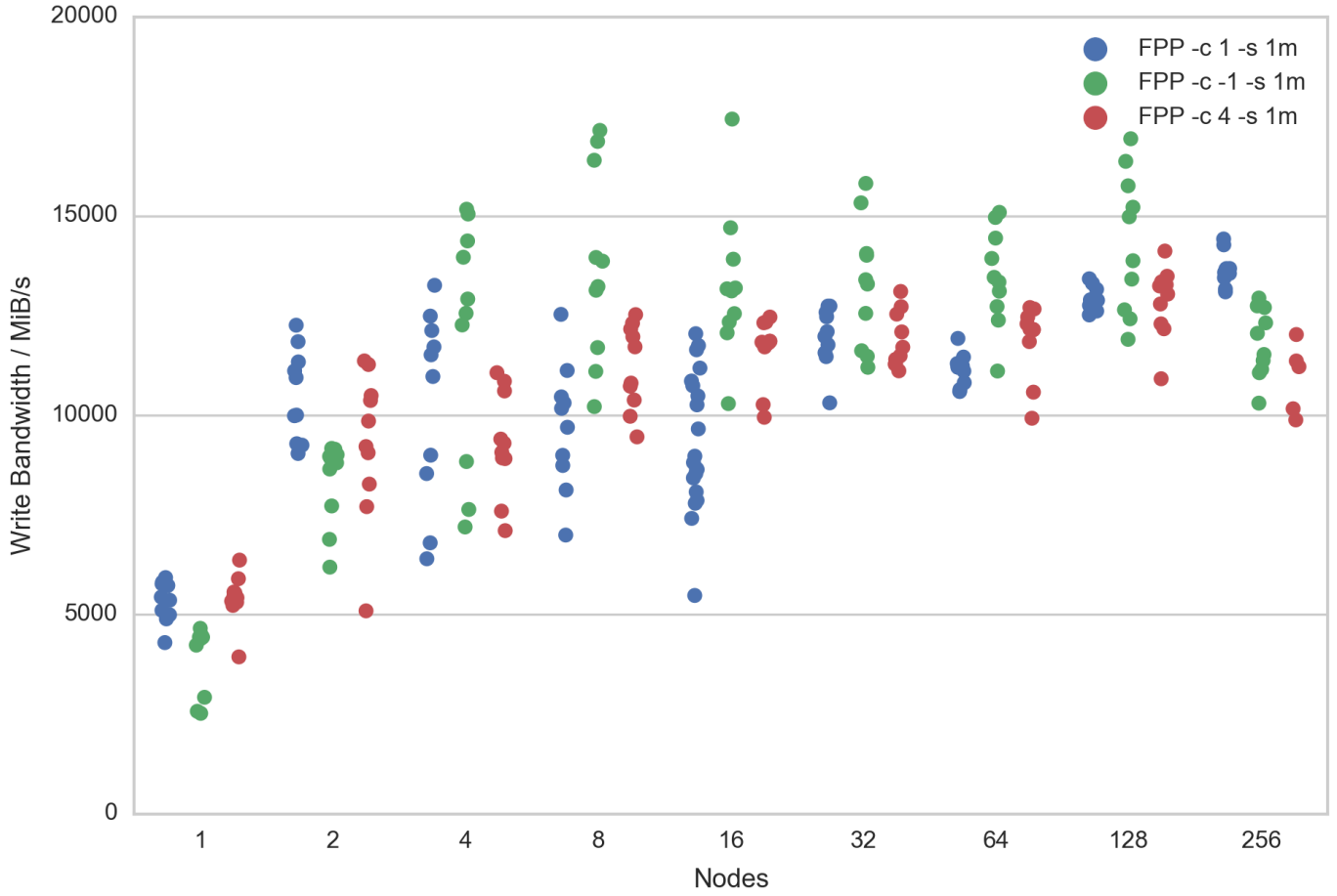
epcc

THE UNIVERSITY OF EDINBURGH

File Per Process (FPP)

Fortran Binary File Comparisons

FPP: ARCHER Lustre

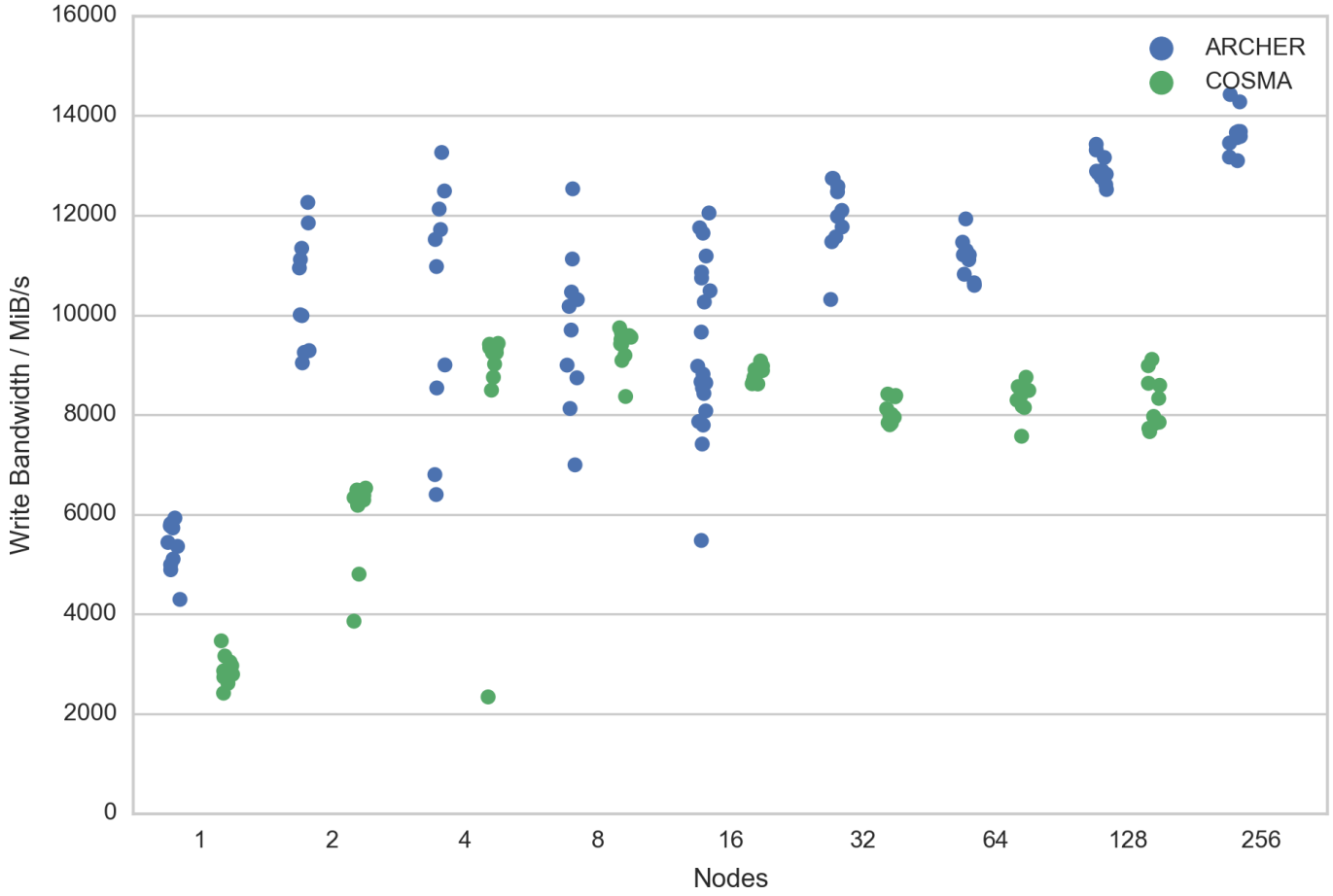


archer

epcc

THE UNIVERSITY OF EDINBURGH

FPP: Lustre/GPFS Comparison



archer

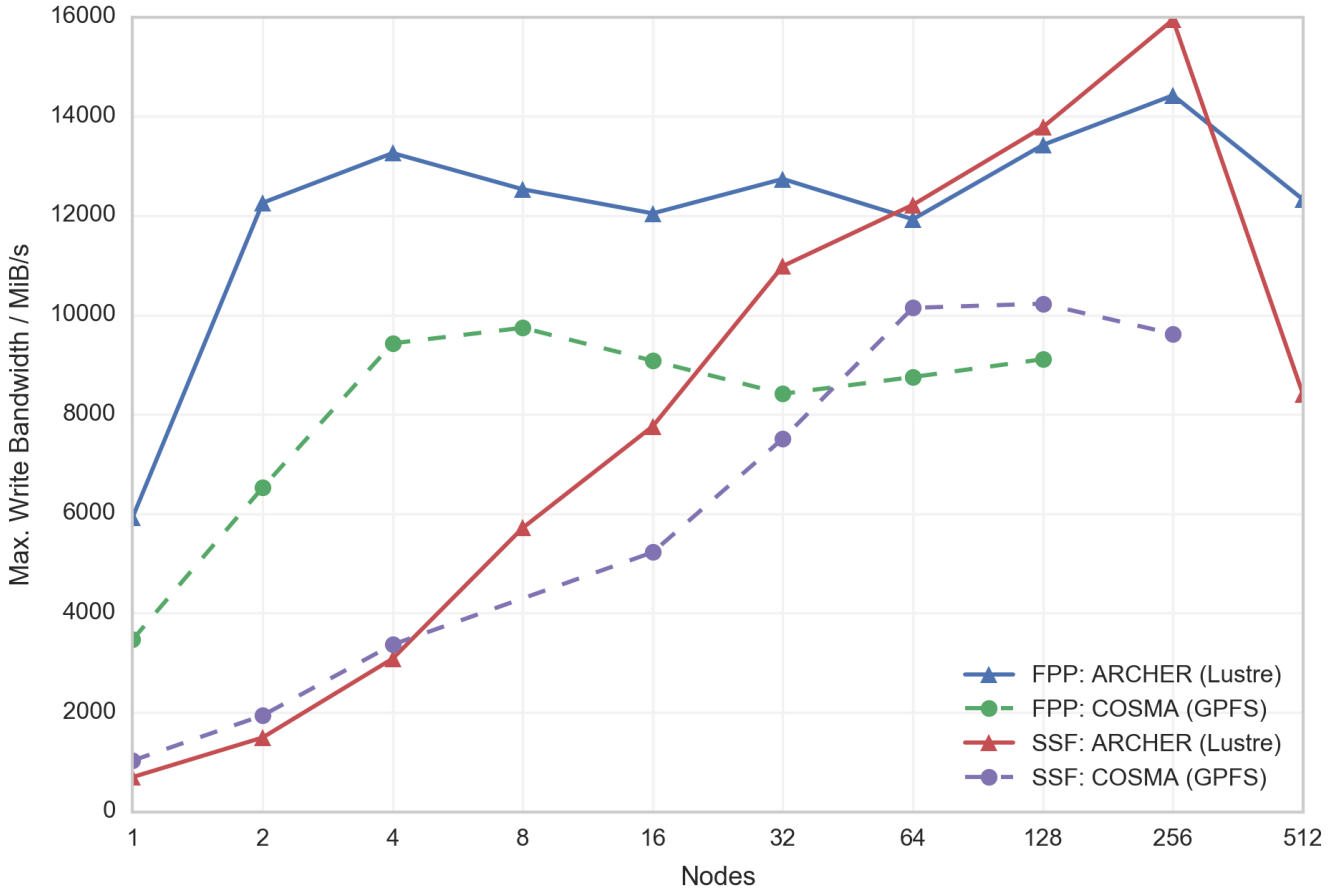
epcc

THE UNIVERSITY OF EDINBURGH

SSF vs. FPP

Comparisons

SSF vs. FPP: max. peformance



archer

epcc

THE UNIVERSITY OF EDINBURGH

SSF vs. FPP

- Simple file-per-process gives better performance at lower node counts…
  - …and is similar to shared file at higher node counts
- Should always use single striping for FPP on ARCHER:
  - Get random failures due to excessive metadata operations otherwise
- Disadvantages to FPP:
  - You will probably have to reconstruct the data for any analysis
  - For checkpoints, you must use identical decomposition
- FPP worth considering if you can live with constraints
- Both schemes achieve a maximum of ~50% of peak for both GPFS and Lustre in production

SSF vs. FPP (cont.)

- SSF can give excellent performance:
  - Each I/O client (node) writes a single block of data
  - Usually requires significant internal communication to reorganise data layout
  - Contingent on using well written parallel I/O libraries to perform this reorganisation…
  - …and this requires parallel collective I/O (without this the performance can be orders of magnitude less)
  - Advantage that data is often in a format that can be analysed or reused at the end of the simulation

Summary

- File-per-process is simple and performs well up to high node counts
  - Probable cost in data reconstruction
  - May be useful for pure checkpointing
- Shared file competitive at high node counts
  - Must use MPI-IO collectives
  - Must use maximum stripe count on Lustre
  - Stripe size has small effect
- Maximum of ~50% peak file system performance

Further Work

- Understand where ~50% maximum performance limit comes from
- Analyse results for HDF5 and NetCDF
- Extend benchio to benchmark read performance
- Run I/O-bound application benchmarks
- Analyse automatically-gathered Lustre performance statistics