Parallel I/O Performance

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

Acknowledgements

- Contributers:
  - 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?

Motivation (cont.)

- HPC users and application developers often poorly understand parallel I/O performance in terms of:
  - what “good” performance actually is on a particular file system;
  - what a particular application does;
  - what benchmarks illustrate.

Benchmarks

IOR and benchio

Benchmarks: IOR

- Widely used for testing parallel file system performance from multiple processes
  - Supports: MPI-IO, HDF5, NetCDF
- Investigated as a benchmark but not chosen because:
  - the software is opaque with many different options; this hinders understanding on what I/O operations are actually being performed;
  - the naïve parallel I/O patterns implemented in IOR do not provide a good representation of user I/O so the results from the benchmark do not provide a useful measure of where the limits of performance are for users.

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

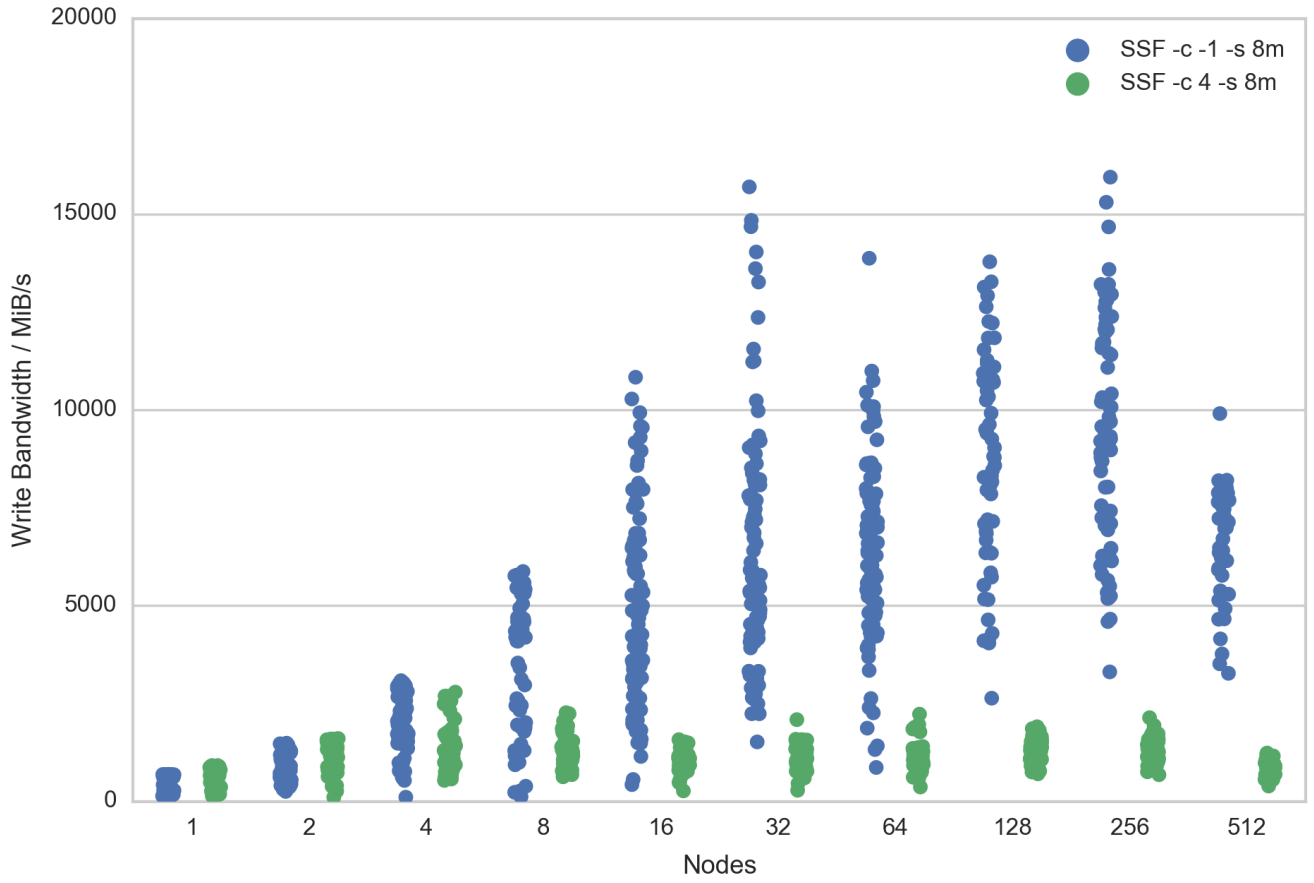
Benchmark setup (cont.)

- 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

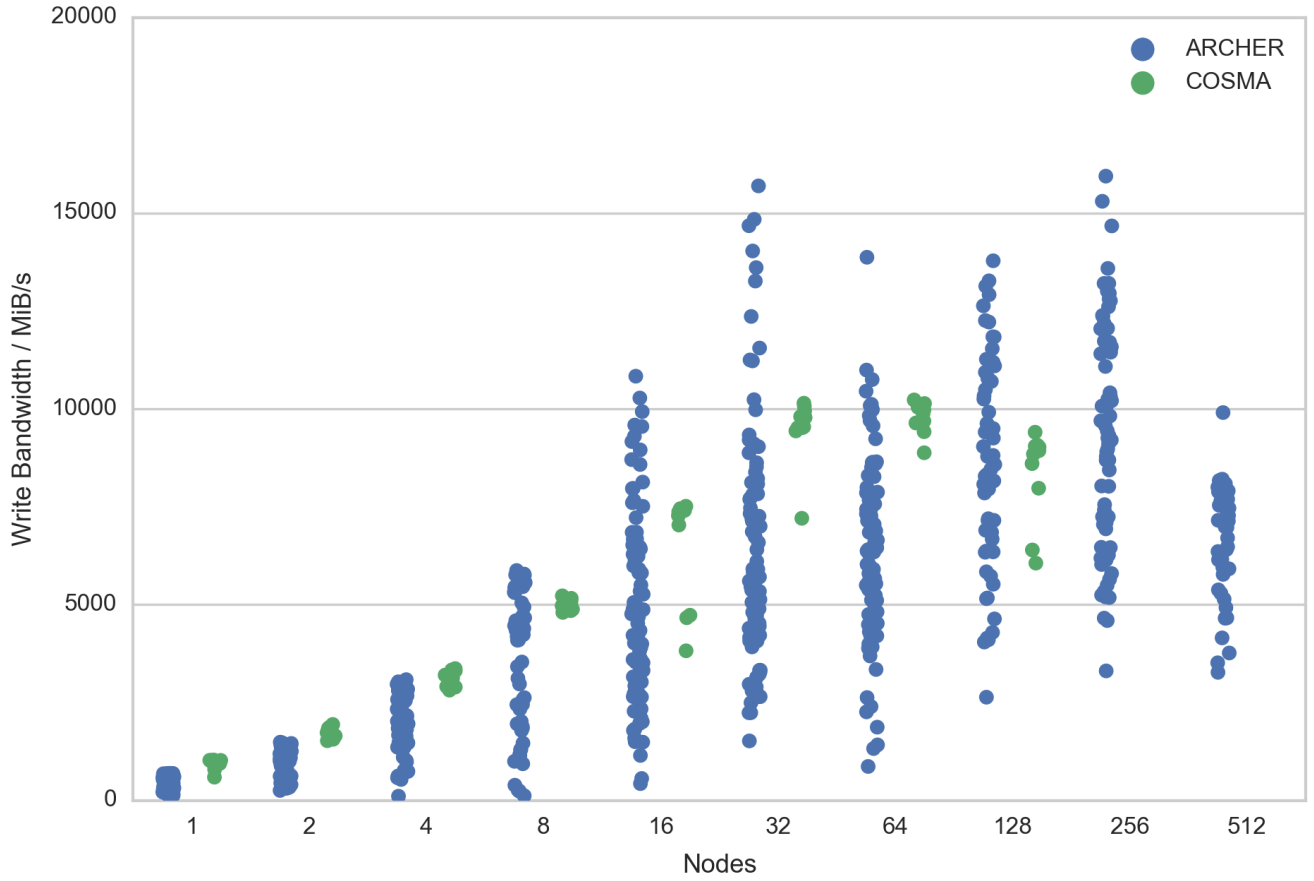


archer

epcc

THE UNIVERSITY OF EDINBURGH

SSF: Lustre/GPFS Comparison



archer

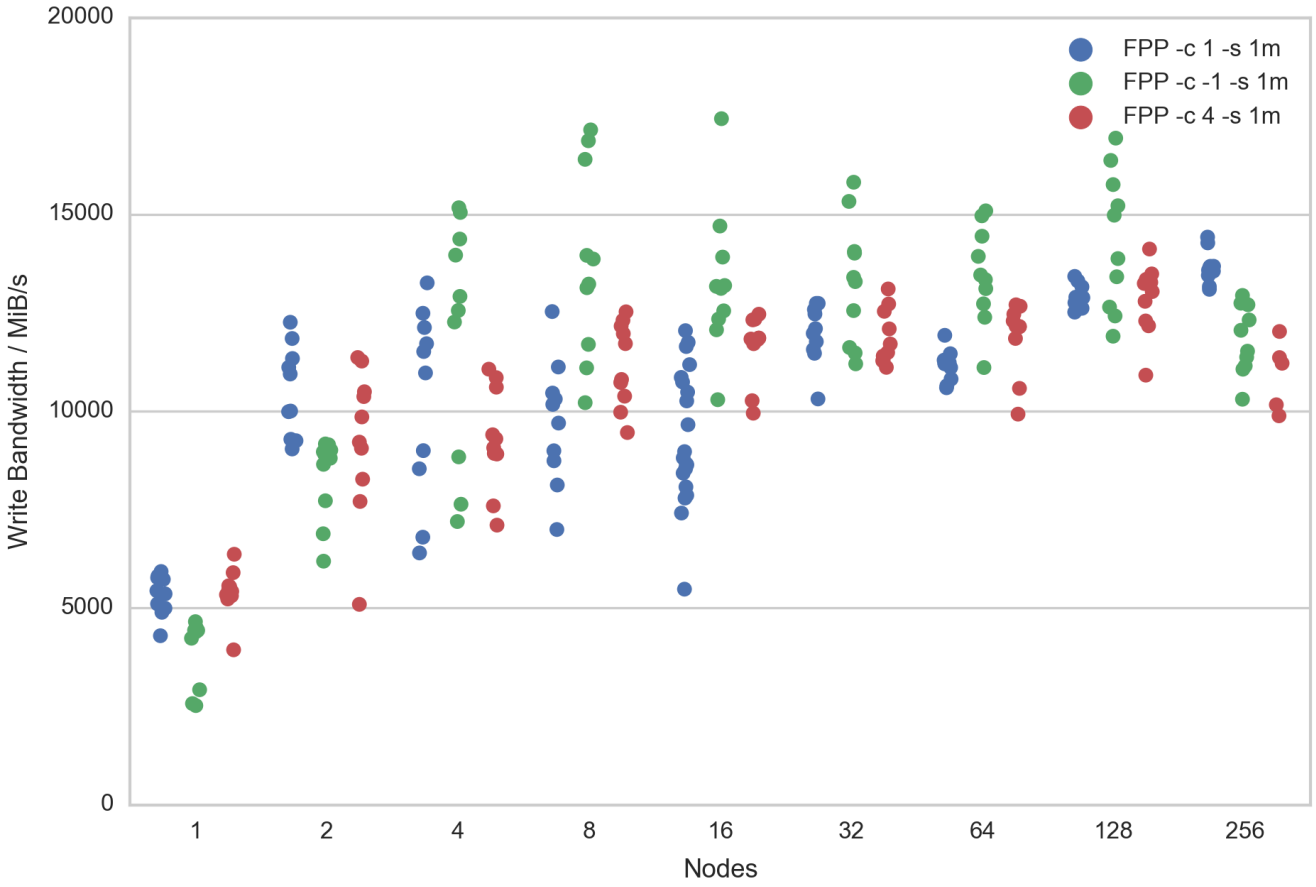
epcc

THE UNIVERSITY OF EDINBURGH

File Per Process (FPP)

Fortran Binary File Comparisons

FPP: ARCHER Lustre

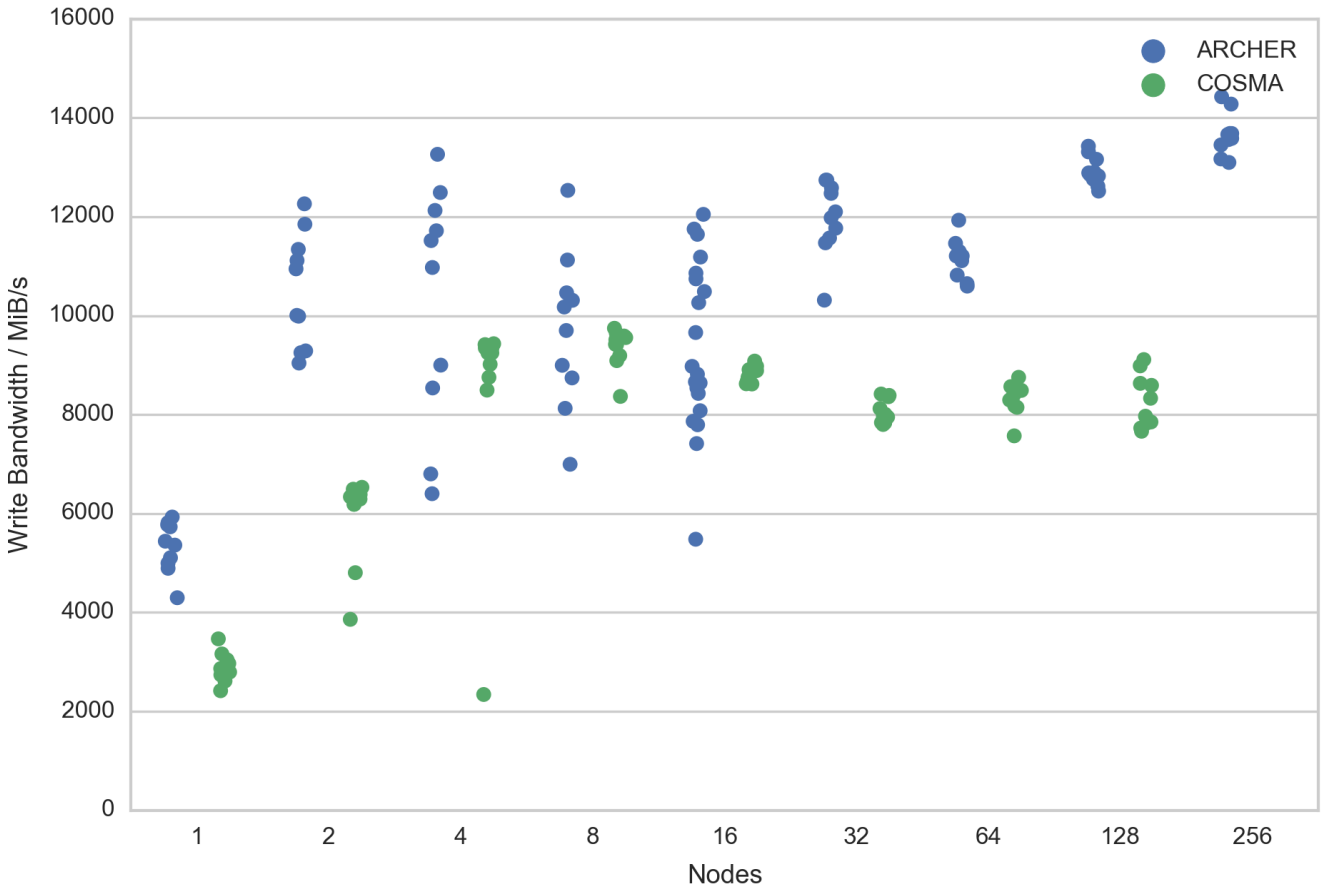


archer

epcc

THE UNIVERSITY OF EDINBURGH

FPP: Lustre/GPFS Comparison



archer

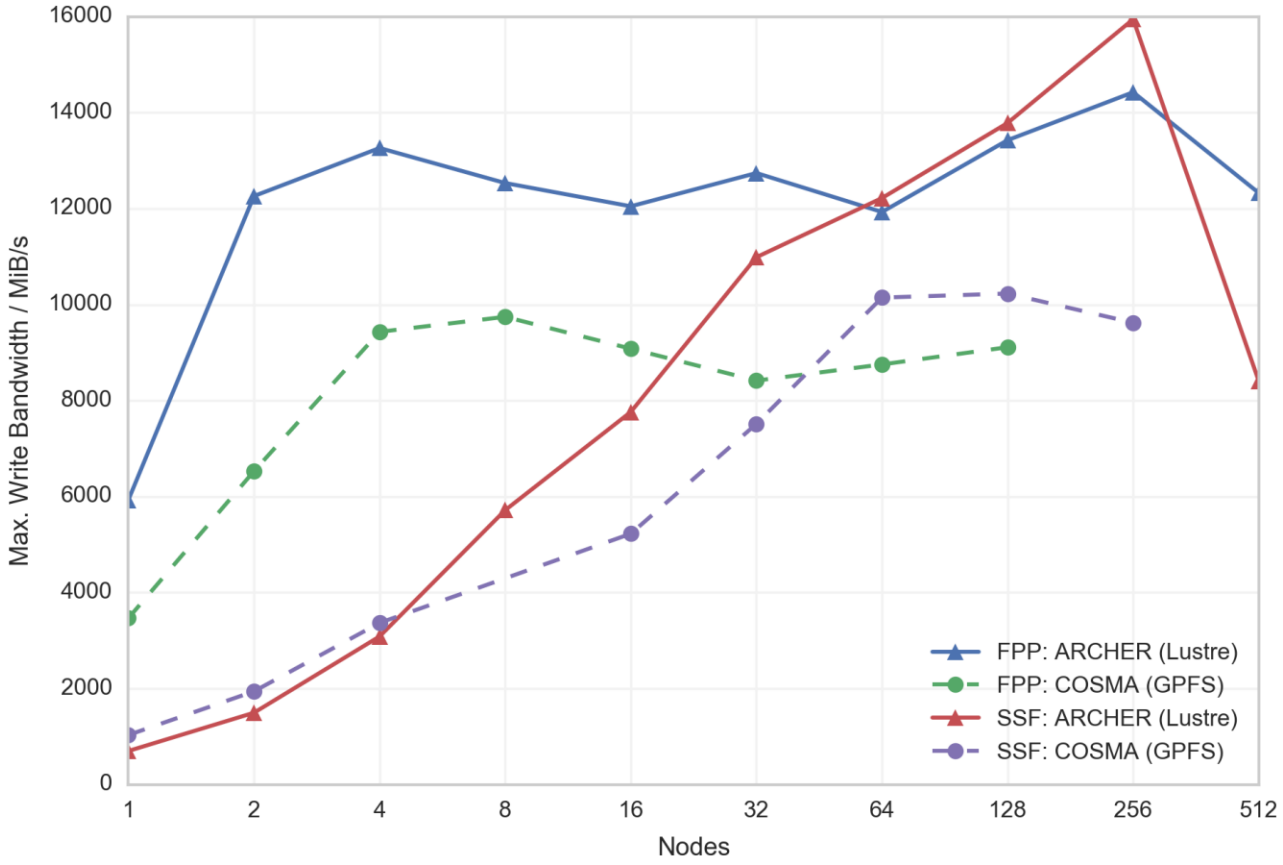
epcc

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 and Further Work

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

Useful Tools

- Pandas:
  - Python statistical analysis library
  - Invaluable for exploring data with multiple classification characteristics
- Seaborn:
  - Python statistical plotting library
  - Visual exploration of data
  - Simple plotting of complex data visualisations

Efficient Parallel I/O Course

- ARCHER Training
- 29-30 March 2017
- University of Durham
- Free for all attendees

- http://www.archer.ac.uk/training/
- Provide access to ARCHER during the course but lessons are generic for all parallel file systems