Performance of Parallel IO on the 5860-node HPE Cray EX System ARCHER2

David Henty

EPCC, The University of Edinburgh

www.epcc.ed.ac.uk

www.archer2.ac.uk

Reusing this material

Partners

epcc

UKRI

Engineering and Physical Sciences Research Council

Natural Environment Research Council

epcc

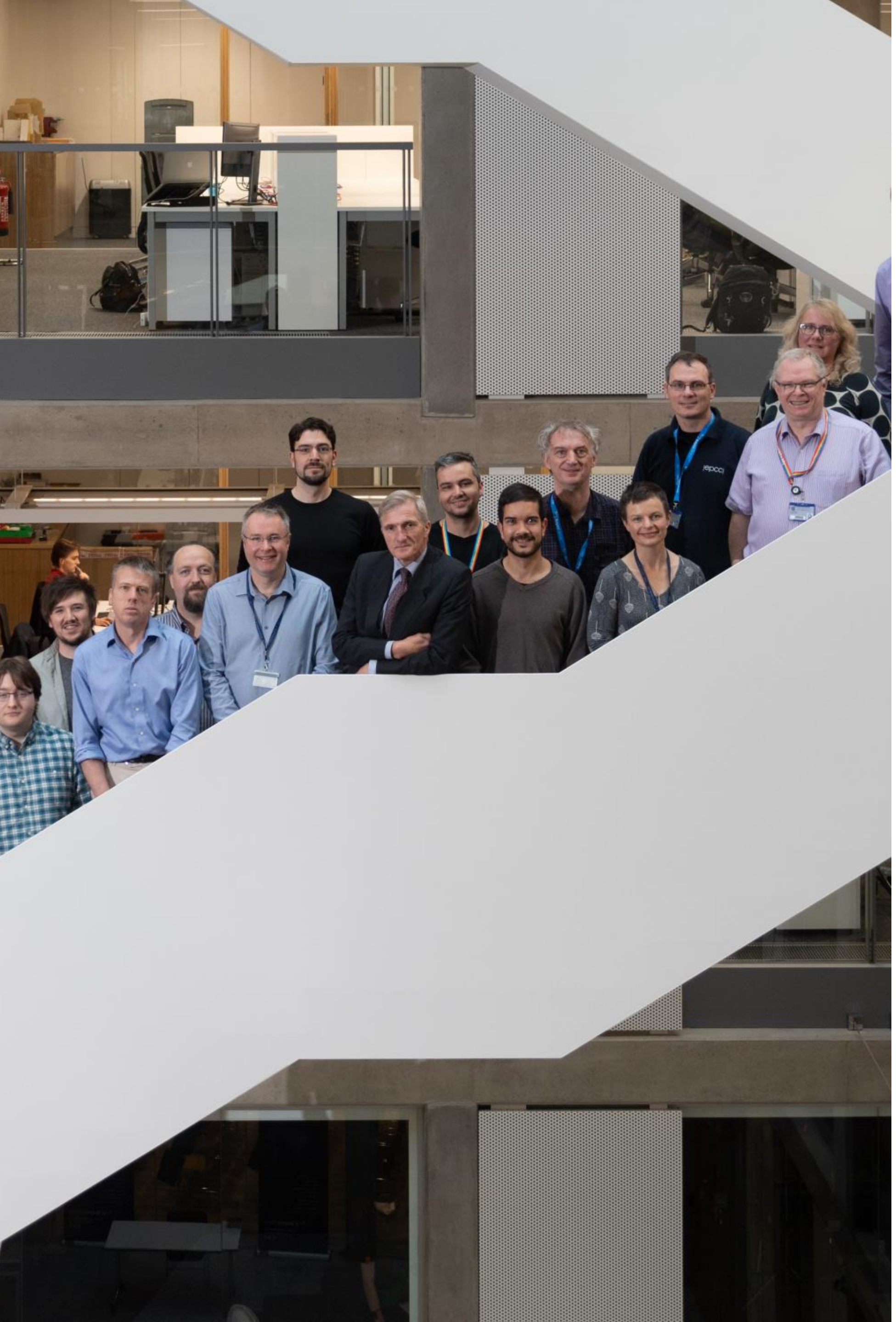
THE UNIVERSITY of EDINBURGH

Hewlett Packard Enterprise

Introduction

- ARCHER2 is the latest UK National Supercomputing Service
  - replaces previous ARCHER service
  - what parallel IO advice should we give ARCHER users for ARCHER2?

	ARCHER Cray XC30	ARCHER2 HPE Cray EX
Compute		
CPU	2× 12-core Intel Ivy-Bridge	2× 64-core AMD EPYC
#nodes	4,920	5,860
#cores	118,080	750,080
network	Cray Aries	HPE Cray Slingshot
Disk		
technology	ClusterStor	ClusterStor L300
#FS	3× Lustre	3× Lustre
#OST / FS	50	12
capacity	4 PiB	13 PiB
NVMe		
technology		ClusterStor E1000F
#FS		1× Lustre
#OST / FS		20
capacity		1 PiB

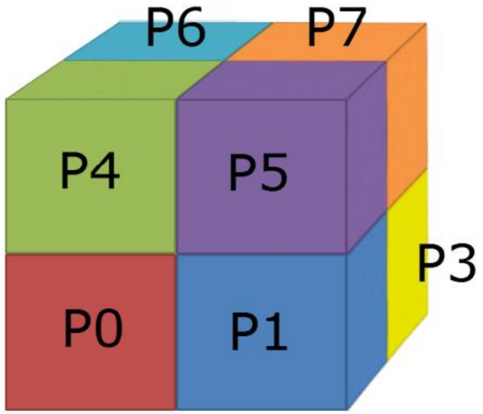


ARCHER2 Service

- Comprehensive support for users from experts at EPCC and HPE
- Application support via ARCHER2 Computational Science and Engineering (CSE) support team
- 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

Benchmarking

- Simple benchio benchmark: https://github.com/davidhenty/benchio
  - written in Fortran for historical reasons
- Large 3D array distributed across 3D process grid
  - writes to a single shared file (SSF): MPI-IO, HDF5 or NetCDF
    - three separate output directories for different filesystem configurations
    - can also write file-per-process (FPP), or single serial file, for comparison
  - surprisingly complicated IO pattern, e.g. 4x4x4 array on 8 processes (2x2x2):

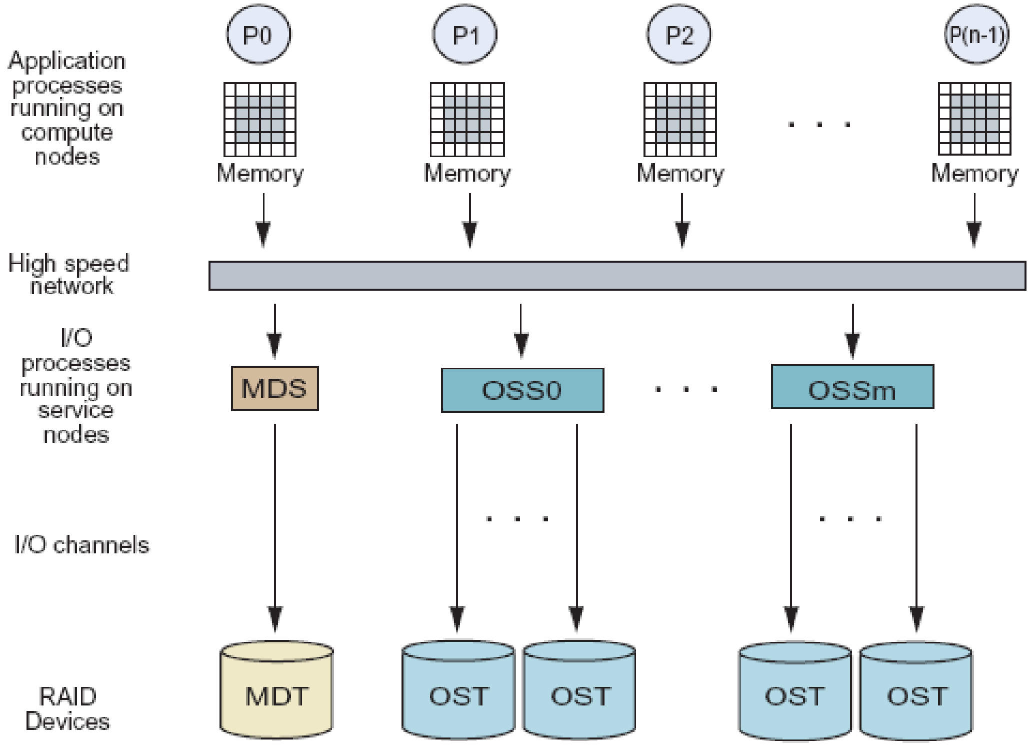


P0	P1	P0	P1	P2	P3	P2	P3	P0	P1	P0	P1	P2	P3	P2	P3
P4	P5	P4	P5	P6	P7	P6	P7	P4	P5	P4	P5	P6	P7	P6	P7

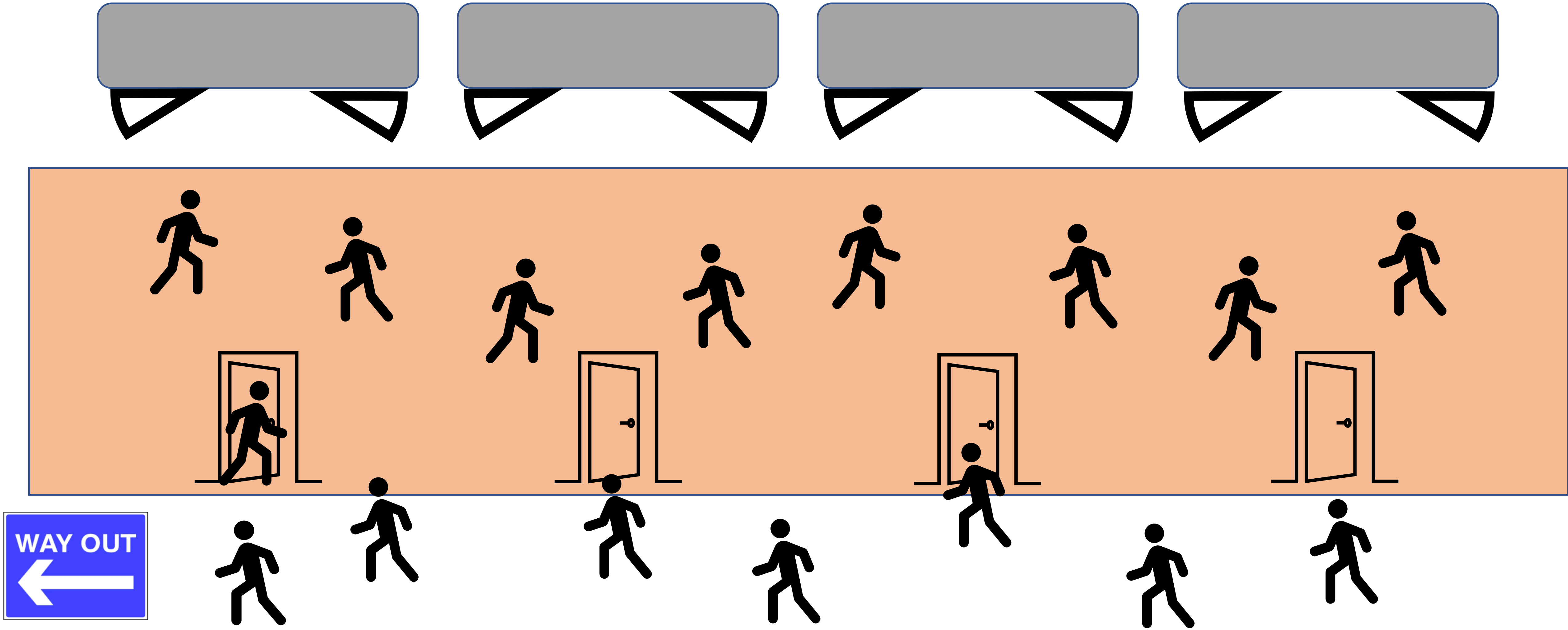
Lustre

- One Lustre filesystem has many disks (strictly, Object Storage Targets)
  - controlled by a single MetaData Server, each node a separate Lustre client
  - ARCHER filesystems had around 50 OSTs
  - ARCHER2 disk filesystems have 12 OSTs
    - NVMe (solid state) filesystem has 20 OSTs

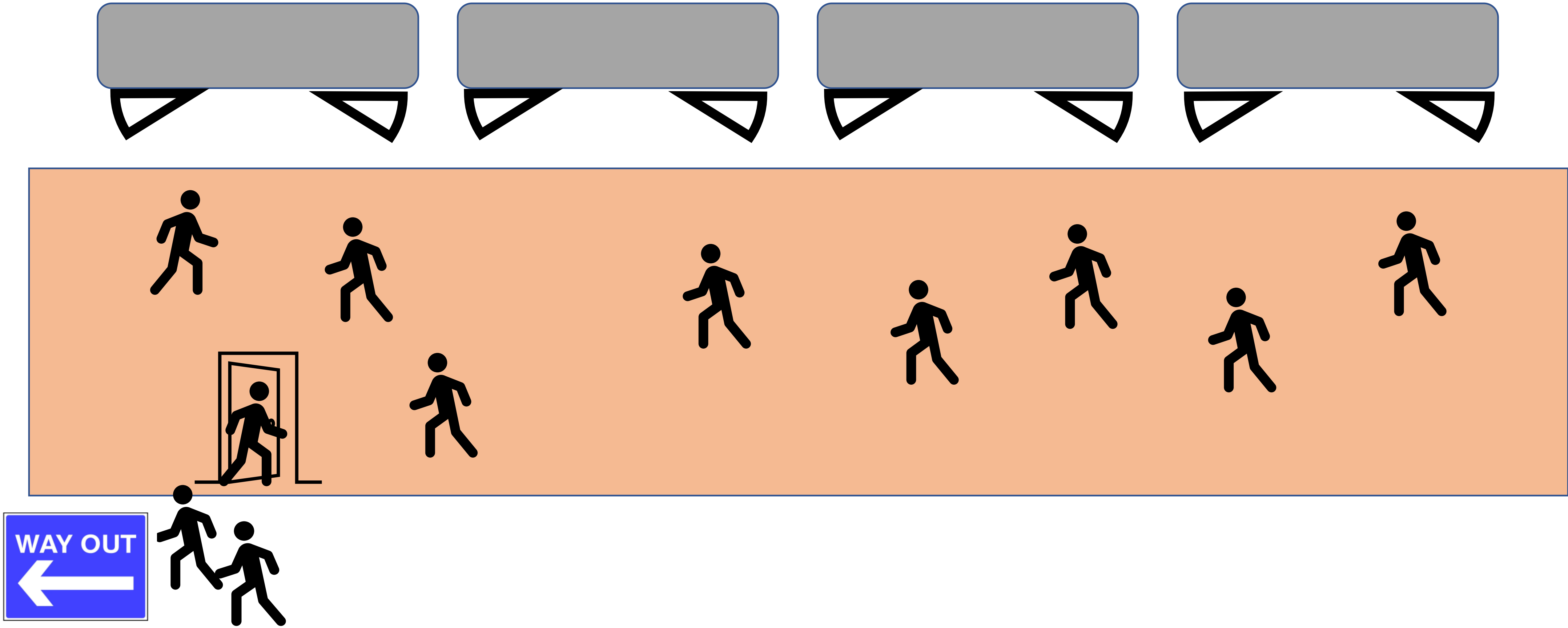
- Multi-disk parallelism in two ways
  - single file stored on many OSTs
    - Lustre calls this "striping"
  - files can be stored on a single OST
    - Lustre will use different OSTs for each file
    - benefits serial IO if there are many files written simultaneously from many nodes



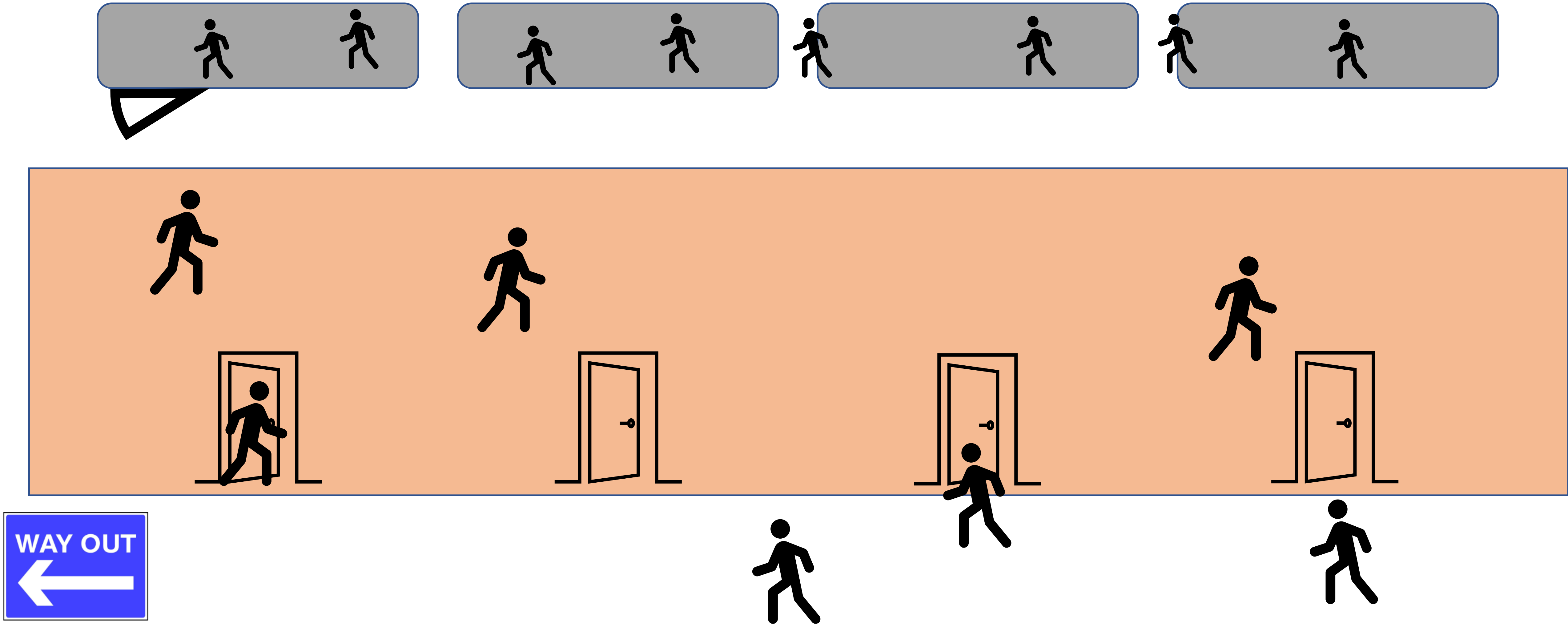
Train Analogy



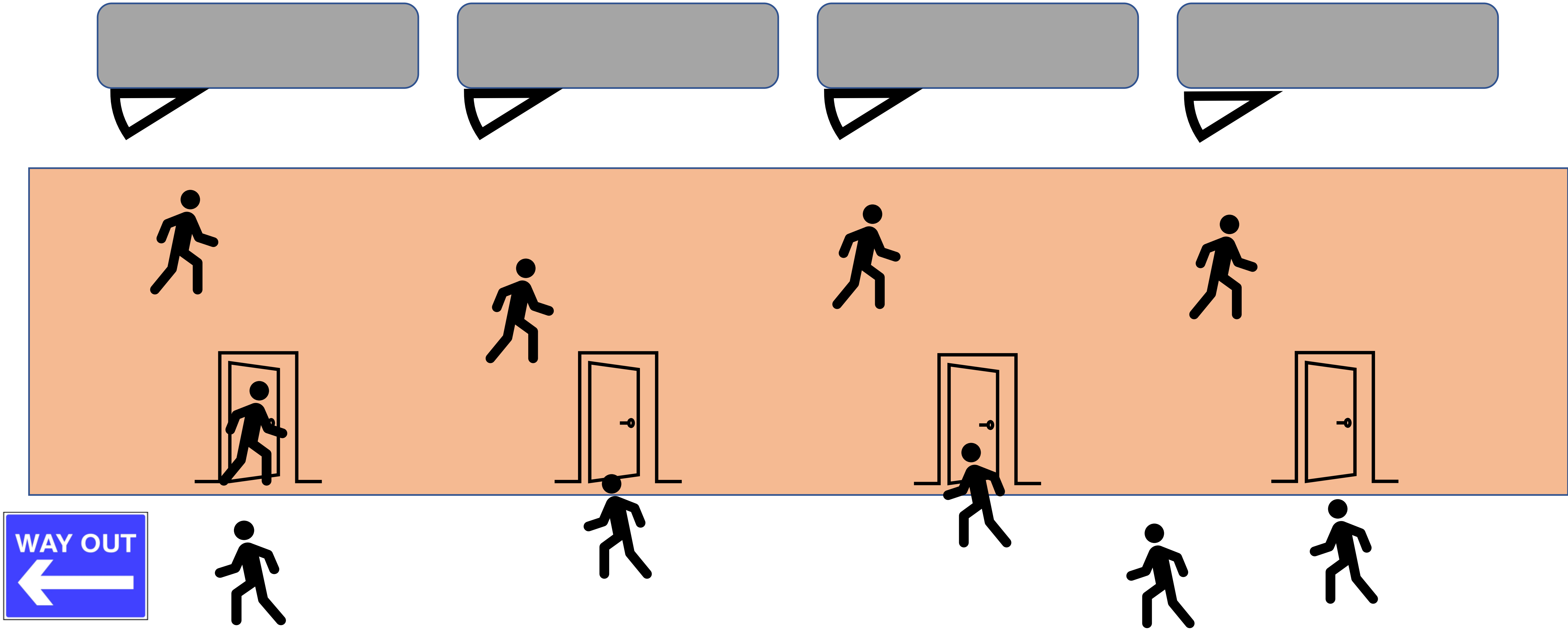
Serial Exit Door



Serial Train Door

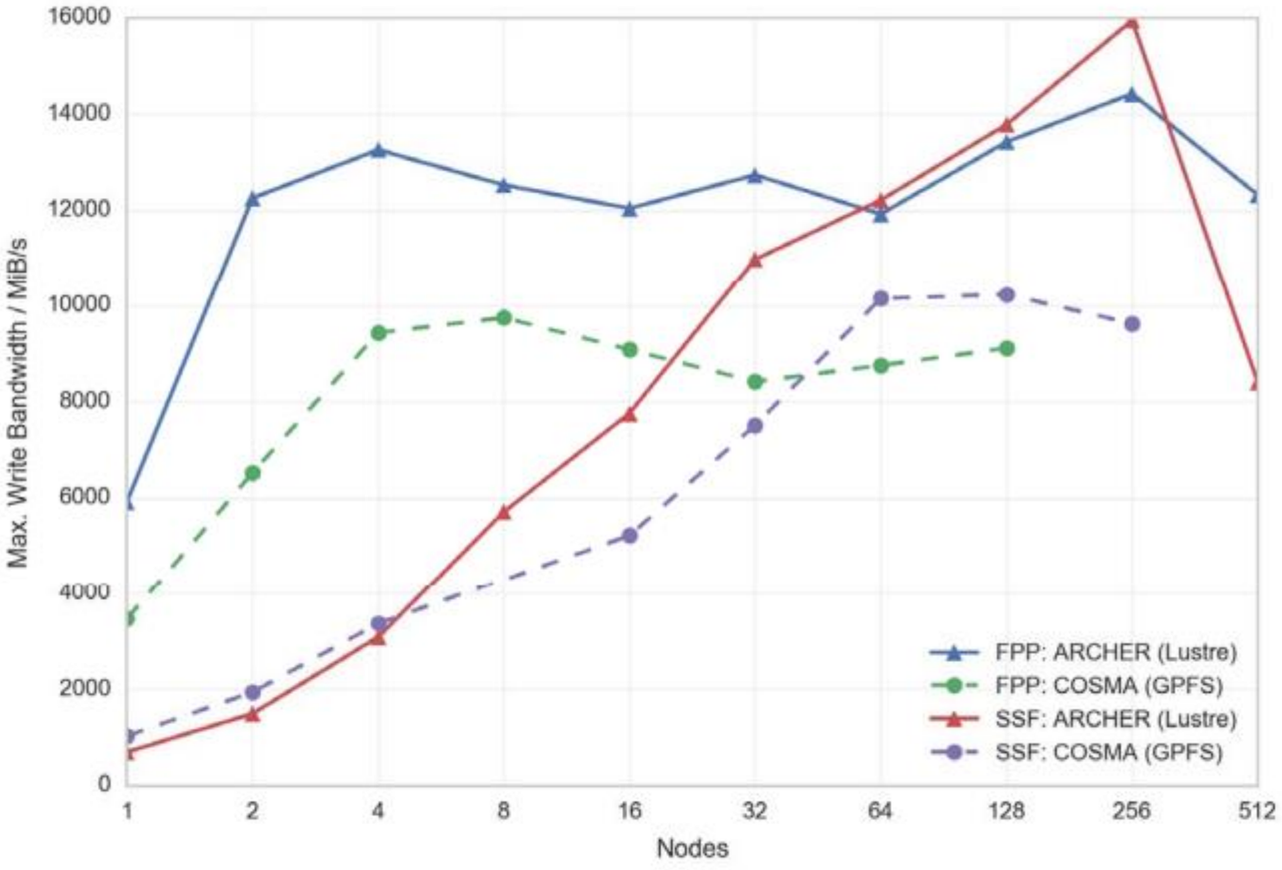


Reduced Train Doors



Where are the bottlenecks?

- From “Parallel IO on ARCHER” at www.archer.ac.uk/training/virtual/



- 500 MiB/s from single process
  - MPI-IO assigns 1 writer / stripe
- Consistent with
  - per-node limit around 6 GiB/s
    - see FPP on 1 node
  - per-OST limit around 700 MiB/s
    - linear scaling of SSF up to 8 nodes
  - about 50% efficiency on all OSTs
    - both SSF and FPP can achieve 15 GiB/s when using all OSTs
    - requires at least 4 nodes for FPP
    - requires at least 64 nodes for SSF

Summary on ARCHER

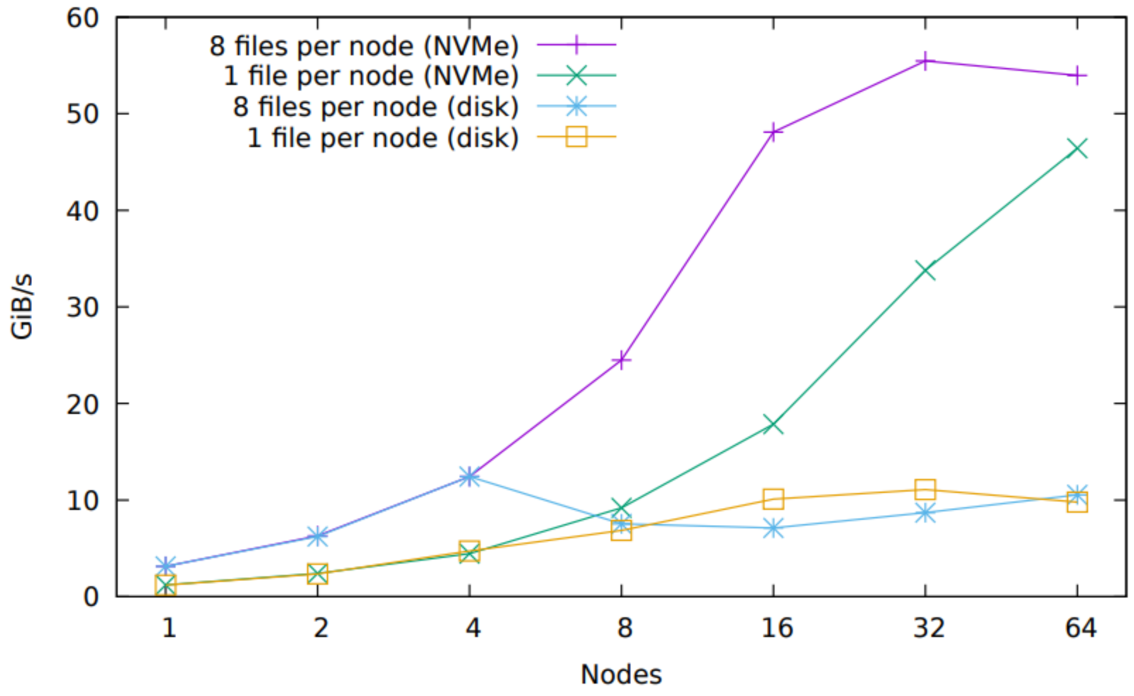
- Peak rates
  - single process can write at 500 MiB/s
  - single node can write at 6 GiB/s
  - single OST can sustain 700 MiB/s
- MPI-IO assigns single process to write per stripe (on different nodes)
  - does not seem optimal as a node can sustain an order of magnitude more
- But
  - single OST bandwidth very similar to single process bandwidth
  - MPI-IO can saturate filesystem with more nodes than OSTs
    - i.e. for 64 or more nodes (as there are 50 OSTs)
  - Contention at scale gives parallel efficiency around 50%
    - maximum aggregate bandwidths around 15 GiB/s for serial (FFP) and parallel (SSF) IO
    - HDF5 and NetCDF largely track MPI-IO: NetCDF calls HDF5 which uses MPI-IO

ARCHER2 investigation

- Range of stripe settings: `lfs setstripe -c <stripecount> <directory>`
  - unstriped/ (`-c 1`) single OST
  - striped/ (`-c 4`) four OSTs
  - fullstriped/ (`-c -1`) all OSTs (12 on disk, 20 on NVMe)
- Run 10 times and use maximum IO rate
  - around 10% standard deviation on disk, less on NVMe as no user service yet
- System software
  - PrgEnv-cray/8.0.0
  - Cray Fortran compiler
  - Cray MPI, MPI-IO, HDF5 and NetCDF libraries

ARCHER2 file-per-process (1 GiB/node)

- FPP results on ARCHER2 difficult to interpret (caching?)
  - over 500 GiB/s for both filesystems (single process achieves around 1 GiB/s)
- Try writing to a single OST (Lustre configuration option)
  - need to restrict the number of files due to contention



- Consistent with:
  - 12 GiB/s max per OST for disk
  - 55 GiB/s max per OST for NVMe
- Hardware limits from HPE
  - 11 GiB/s and 55 GiB/s !
- No clear per-node limit
  - disk and NVME data differ for small node counts

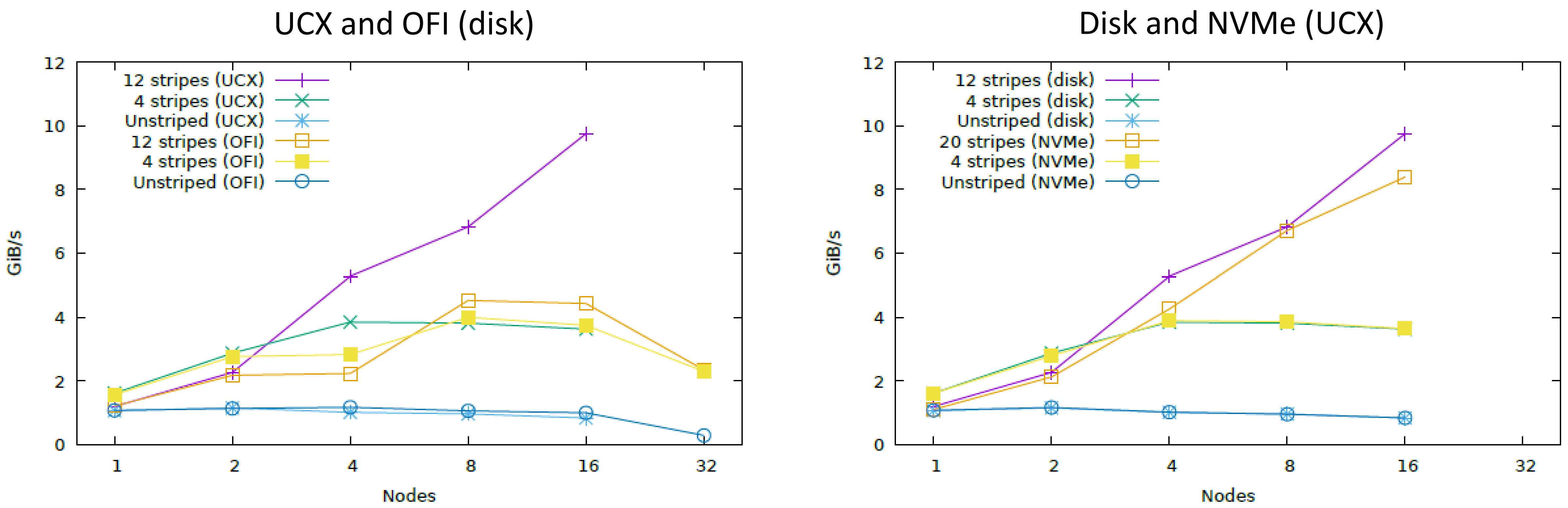
MPI-IO

- Default performance was terrible
  - no benefits from parallelism (multiple nodes or OSTs)

nodes	stripes	GiB/s
1	1	1.07
1	2	1.58
1	12	1.22
2	1	0.01
2	2	0.26
2	12	N/A

  - Two approaches
    - tune MPI collectives for large buffers: **`export FI_OFI_RXM_SAR_LIMIT=64K`**
    - use non-default UCX transport layer (default is Open Fabrics Interface OFI)

OFI vs UCX MPI – strong scaling with 16 GiB file



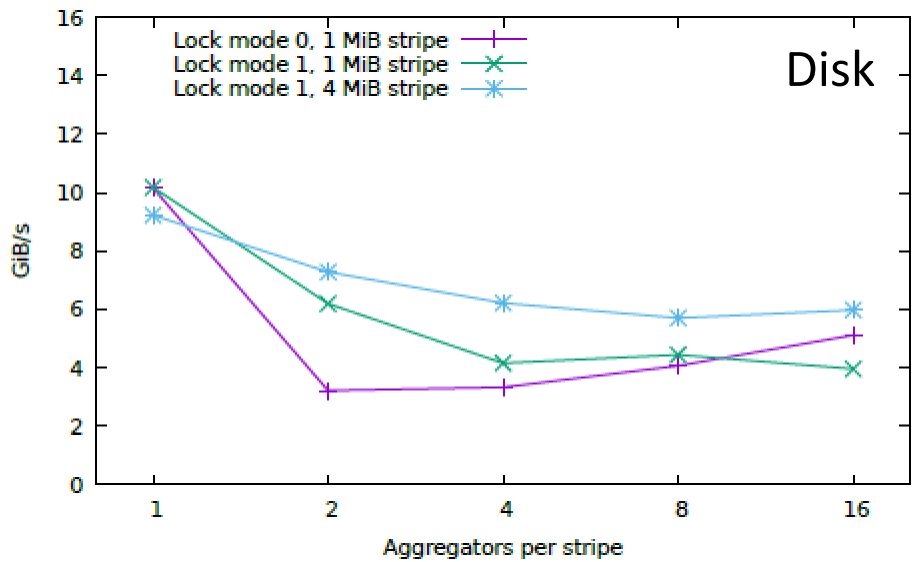
- UCX is better than OFI
  - although benchmark hangs for UCX on 32 nodes or more ...
- Scaling of UCX MPI-IO on ARCHER2 the same as MPI-IO on ARCHER
  - parallel bandwidth = serial bandwidth * min(#stripes, #nodes)

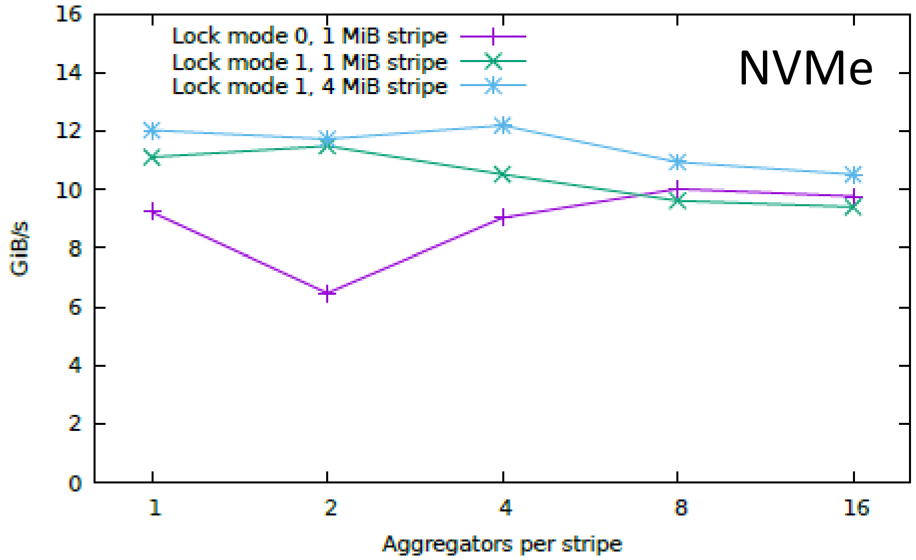
MPI-IO summary

- Parallel IO results very disappointing
  - changing default stripe size of 1 MiB had very little effect
  - note that on both systems collective IO calls are essential
- MPI-IO uses one writer or "aggregator" per Lustre stripe (i.e. per OST)
  - parallel bandwidth of 10 GiB/s limited by per-process IO limit of 1 GiB/s
  - cf. disk and NVMe totals of 12*11 = 132 GiB/s and 20*55 = 1.1 TiB/s !
- On ARCHER
  - could saturate Lustre because OST limit was similar to per-process limit
  - high aggregate bandwidth from large number (50) of slow OSTs
- On ARCHER2
  - have many fewer (12 and 20) OSTs but they are much faster
  - MPI-IO not configured for this situation (HDF5 and NetCDF suffer similarly)

Changing aggregator settings

- Clearly need to have more than one aggregator per node
  - `export MPICH_MPIIO_HINTS = *:cray_cb_nodes_multiplier=2`
  - note that useful stats printed using `export MPICH_MPIIO_STATS=1`
  - multiple aggregators per OST leads to file locking (lock mode 0)
  - can relax this for collective MPI-IO: `*:cray_cb_write_lock_mode=1`
- Results for disk and NVMe, maximal striping (also vary stripe size)





Conclusions

- MPI-IO results disappointing
  - SSF parallel MPI-IO around 10% and 1% of peak disk and NVMe bandwidths
    - requires UCX MPI which may affect MPI comms performance in a real application
    - HDF5 and NetCDF similar
  - user **can** saturate Lustre filesystem using file-per-process
    - but not a practical approach at scale
  - MPI-IO was able to saturate Lustre on ARCHER
    - large number of slow OSTs compared to ARCHER2's small number of fast OSTs
- Single IO aggregator per stripe/OST far from optimal on ARCHER2
  - increasing this did not help, nor did changing locking mode
  - note that relaxed locking **not** an option for NetCDF or HDF5 as they perform some non-collective IO even in collective mode (for metadata?)

Further work

- Work with HPE to try and address the poor performance
  - resolve issues with UCX on 32 nodes

- Extend benchio to use ADIOS2 library
  - ADIOS2 can use MPI-IO, HDF5 or its own file format
  - we have seen good performance elsewhere using BP4

- Initial results (from other work)
  - MPI-IO and HDF5 write the same file in parallel as in serial
  - ADIOS2 BP4 appears adaptive, e.g. sometimes writes multiple files
    - has the same concept of "aggregators" as MPI-IO – default seems to be one per node
    - possibility of much improved bandwidth if aggregators write to different files and therefore avoid issues around file locking