Differences between ARCHER and ARCHER2

Andy Turner, EPCC, The University of Edinburgh

a.turner@epcc.ed.ac.uk

www.archer2.ac.uk

Reusing this material

Partners

UKRI

Engineering and Physical Sciences Research Council

Natural Environment Research Council

epcc

THE UNIVERSITY of EDINBURGH

CRAY®

a Hewlett Packard Enterprise company

Christopher Ellis

ARCHER2 4 Cabinet Service

ARCHER2 4 cabinet system

- 4 cabinet HPE Cray EX supercomputer
- Hosted at EPCC ACF in Edinburgh
- 1,024 compute nodes
    - 131,072 AMD 7742 cores (2x 64-core processors per node)
    - 256 GiB Memory per node
- 3.7 PB Lustre (1 file system)
- 1 PB home (backed up) storage
- HPE Cray Slingshot interconnect
- 2 login nodes
- Data on 4 cabinet /work file system made available on full system
    - Allow users to copy data over to full system file systems before it is deleted
    - Plan for 3 months of access before data is deleted

ARCHER2 Full System

- HPE Cray EX system
  - (formerly Shasta)
- Peak performance: ~28 PFlops
- 5,848 compute nodes
  - 748,544 AMD cores
- 14.5 PB Lustre (4 file systems)
- 1.1 PB solid state burst buffer
- 1 PB home (backed up) storage
- HPE Cray Slingshot interconnect

Comparison of services

	ARCHER	ARCHER2	ARCHER2 4 Cabinet
Compute nodes (Cores)	4,980 (118,080)	5,848 (748,544)	1,024 (131,072)
High memory nodes	✓	✓	✗
Login nodes	8	4	2
Parallel file systems	3 (4.4 PB total)	4 (14.8 PB total)	1 (3.7 PB total)
Solid state file system	✗	✓	✗
Job scheduler	PBS Pro	Slurm	Slurm
Serial queue (PP nodes)	✓	✓	✗
Compilers	Cray, Gnu, Intel	Cray, Gnu, AMD	Cray, Gnu, AMD
Singularity containers	✗	✓	✓
RDF file system mount	✓	✓	✗

- ARCHER2 Service Provision from EPCC fully operational for 4 cabinet service
  - Project/user administration, service desk, system administration, etc.
- ARCHER2 CSE service from EPCC fully operational for 4 cabinet service
  - Technical support, training, eCSE, etc.

Comparison to ARCHER

Comparison with ARCHER: compute nodes

	ARCHER2	ARCHER	Ratio
Processors	2x AMD EPYC 7742	2x Intel Xeon E5-2697 v2	
Cores per socket (node)	64 (128)	12 (24)	5.3:1
Cores per NUMA region	16	12	
Max. SP GFLOP/s per socket (node)	4608 (9216)	518 (1037)	7.9:1
Max. SP GFLOP/s per core	72	43	1.7:1
Memory per node	256/512 GB	64/128 GB	4.0:1
Memory per core	2.0/4.0 GB	2.7/5.3 GB	0.7:1
Memory channels per socket	8	4	2:1
Memory	DDR 3200	DDR 1866	
Max. memory BW per socket (node)	190.7 (381.4) GB/s	59.7 (119.4) GiB/s	3.2:1
Max. memory BW per core	3.0 GiB/s	5.0 GiB/s	0.6:1

Compute nodes vs ARCHER: Summary

- Much more compute power and memory per node…
- …but the balance is different
  - Many more cores per node
  - Less memory and memory bandwidth on a per-core basis
  - Different cache and NUMA structure
- Need to stop thinking about cores as the key resource indicator
  - Think about the whole node and how you can best exploit it
  - Potentially under-populate the node (i.e. not use all cores)
  - Explore use of multithreading per process
- Different interconnect also changes characteristics…

Software differences

	ARCHER2	ARCHER
Compilers	Cray (CCE): 10.0.3 Gnu (GCC): 10.1.0 AMD (AOCC): 2.1.0.3	Cray (CCE): 8.5.9 Gnu (GCC): 6.3.0 Intel: 17.0.0.098
BLAS/LAPACK/ScaLAPACK	HPE Cray LibSci: 20.8.1.2	Cray LibSci: 16.11.1 Intel MKL: 17.0.0.098
FFTW	FFTW: 3.3.8.7	FFTW: 3.3.6.1 Intel MKL: 17.0.0.098
MPI	HPE Cray MPICH: 8.0.15	Cray MPICH: 7.5.5
Parallel Debugger	gdb4hpc: 4.7.3 valgrind4hpc: 2.7.2	Arm Forge (DDT): 19.0.1
Profiler	CrayPAT: 20.09.0	CrayPAT: 6.4.6 Arm Forge (MAP): 19.0.1
Tools	HPE Cray Python: 3.8.5.0 HPE Cray R: 4.0.2.0 HPE Cray DL Tools: 20.06.01	

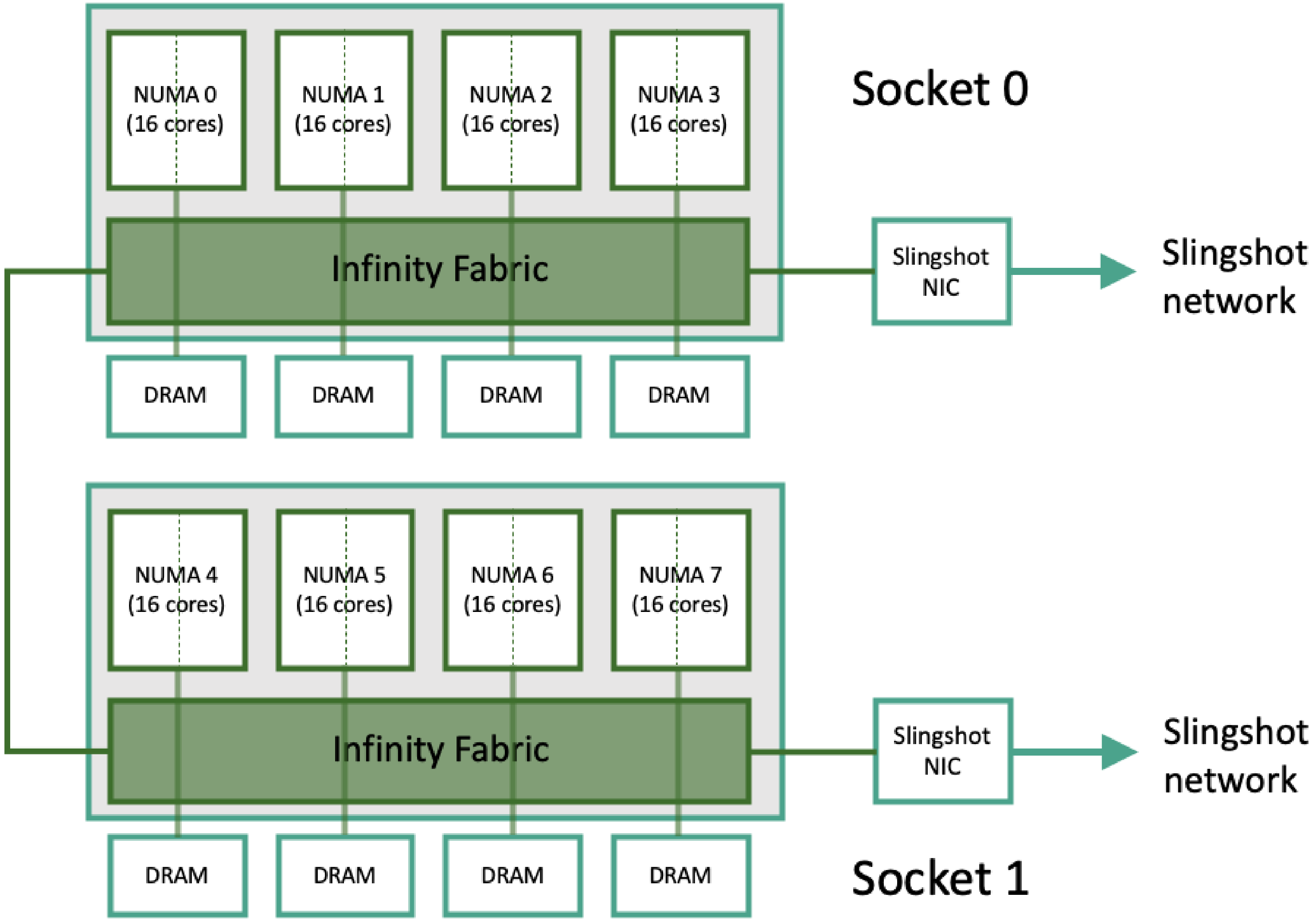
Compilers

- Cray Compiler Environment (CCE), PrgEnv-cray
  - C/C++ compiler is based on Clang
  - Fortran compiler is still the Cray Fortran compiler (crayftn)
- Gnu Compiler Collection (GCC), PrgEnv-gnu
  - C compiler: gcc
  - C++ compiler: g++
  - Fortran compiler: gfortran
- AMD Optimizing Compiler (AOCC), PrgEnv-aocc
  - C/C++ compiler is based on Clang
  - Fortran compiler is based on PGI Flang (**not** f18 Flang)

Barbara Farkas

ARCHER2 Overview

ARCHER2 compute node



Each 16-core NUMA region is made up of 2 8-core complexes

Each 8-core complex contains:

- 8 compute cores with 256-bit AVX2 and FMA
- Each compute core with 512 KB L2 cache
- Shared 64 MB L3 cache

Programming environment and modules

- Initial system based on Environment Modules v4
  - Plan to move to Lmod as soon as it is supported by HPE Cray (likely late 2020)
- Major difference is in how you access different programming environments
  - Use `module restore PrgEnv-*`
  - This is because programming environments are now module collections
- If you need to load programming environments in a batch script then you need to pass a full path
  - e.g. `module restore $PRGENV_DIR/PrgEnv-gnu`
  - e.g. if you want to compile on the compute nodes

HPE Cray provided software

Compilers	Cray, Gnu, AMD
Numerical libraries	HPE Cray LibSci (BLAS, LAPACK, ScaLAPACK) FFTW
IO libraries	HDF5 NetCDF
MPI, OpenSHMEM, CAF, UPC	HPE Cray MPT
Tools	HPE Cray Python (numpy, scipy, mpi4py, dask) HPE Cray R
Debugging	gdb4hpc valgrind4hpc
Profiling	HPE Cray Perftools (CrayPAT, ATP, etc.)

CSE provided software (initial list)

Modelling and simulation	Libraries	Analysis and tools
CASTEP	ARPACK	CDO
Code Saturne	Boost	CGNS
PyChemshell/ChemShell	Eigen	NCL
CP2K	GMP	NCO
ELK	GSL	Paraview
FEniCS	HYPRE	PLUMED
LAMMPS	METIS/ParMETIS	VMD
MITgcm	MUMPS	VTST
NAMD	ParaFEM	
NEMO	PETSc	
NWChem	Scotch/ptScotch	
ONETEP	SLEPC	
OpenFOAM	SUNDIALS	
VASP (5 and 6)	SuperLU/SuperLU_DIST	
	Trilinos	
	Zoltan	

Interconnect: Slingshot

- Ethernet based network
- 2x 100 Gbps Mellanox network interface cards (NICs) per node
- HPE Cray Slingshot switches (200 Gbps per port)
- Two possible underlying software interfaces:
  - Open Fabrics Interface (OFI): craype-network-ofi
    (this is the default on ARCHER2)
  - Unified Communication X (UCX) craype-network-ucx
  - Not yet had a chance to explore any performance differences

Dynamic linking and RUNPATH

- ARCHER2 currently only supports dynamic linking of executables
  - Static linking option may follow in the future
- The compiler wrapper scripts automatically add paths to libraries from the HPE Cray programming environment into the binary RUNPATH
  - These libraries can be found at runtime without needing to load the correct modules
  - If you load a module with a different version from compile time in your job submission script this will supersede the version in RUNPATH
- Unsure if we can make CSE provided libraries behave in the same way - think this is likely possible but we are still testing

Scheduler: MPI jobs

```
#!/bin/bash
#SBATCH --job-name=Example_MPI_Job
#SBATCH --time=0:20:0
#SBATCH --nodes=4
#SBATCH --tasks-per-node=128
#SBATCH --cpus-per-task=1
#SBATCH --account=t01

export OMP_NUM_THREADS=1

srun --cpu-bind=cores my_executable.x
```

Select 4 nodes and 128 (MPI) processes per node

Set OpenMP threads to 1 to prevent libraries from using threading by default

srun uses the distribution from the job options to launch the correct number of MPI processes and place them on the correct nodes and pin to the correct cores

Scheduler: MPI+OpenMP jobs

```
#!/bin/bash
#SBATCH --job-name=Example_MPI_Job
#SBATCH --time=0:20:0
#SBATCH --nodes=4
#SBATCH --tasks-per-node=8
#SBATCH --cpus-per-task=16
#SBATCH --account=t01

export OMP_NUM_THREADS=16
export OMP_PLACES=cores


srun --hint=nomultithread --distribution=block:block my_executable.x
```

Select 4 nodes and 8 (MPI) processes per node, stride of 16 cores between processes to make space for OpenMP threads

Set OpenMP threads to 16 and specify placement to get correct pinning to cores

srun uses the distribution from the job options to launch the correct number of MPI processes and place them on the correct nodes and pin the processes and threads to the correct cores

More complex placements are possible – e.g. MPI processes cyclic across NUMA regions with block distribution of threads within NUMA regions.

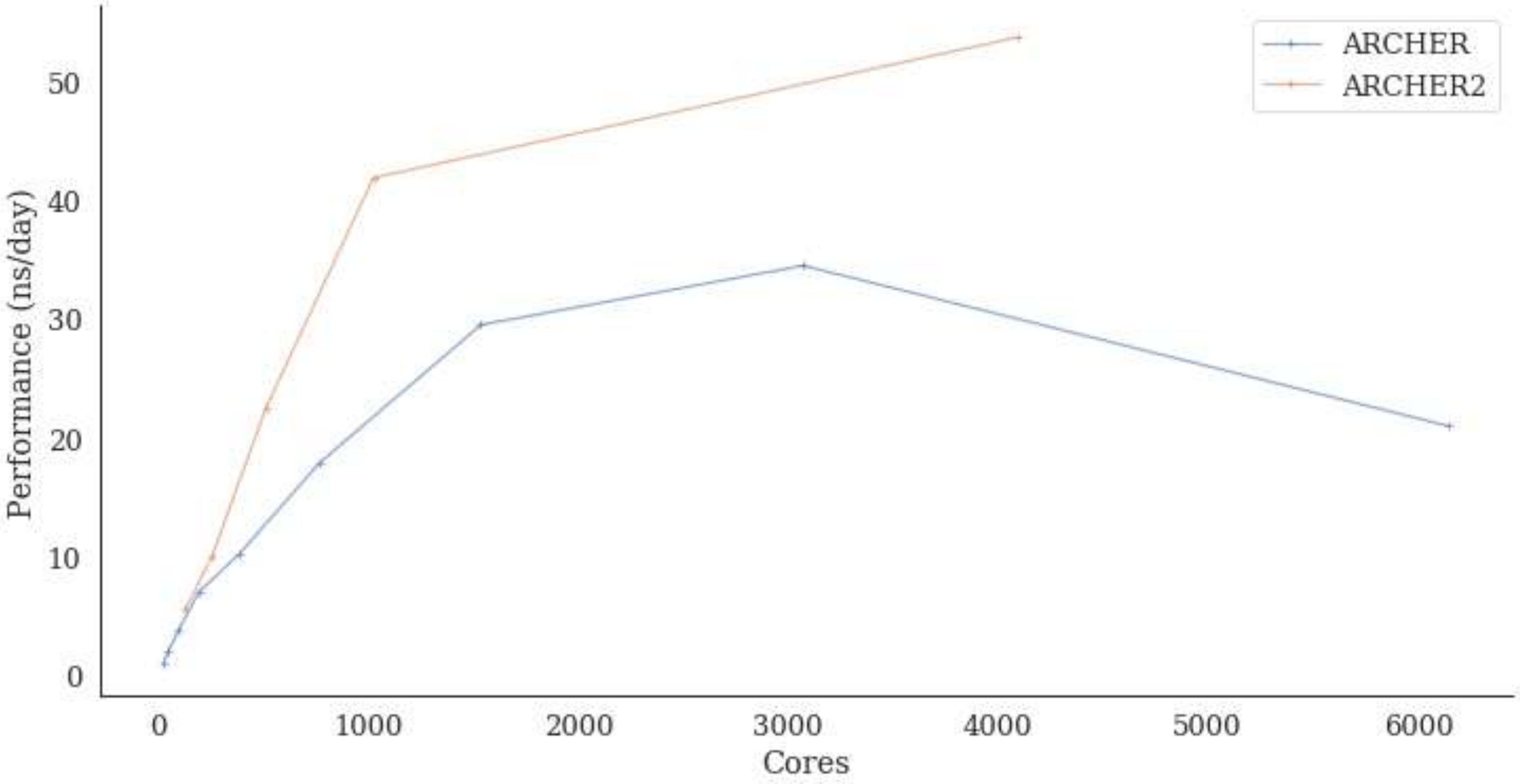
Dr Alfonso Bueno Orovio

Performance

Caveats

- Performed on system with pre-release system software
- No optimisation investigations performed
- Not many (sometimes any) repeat runs to capture variability

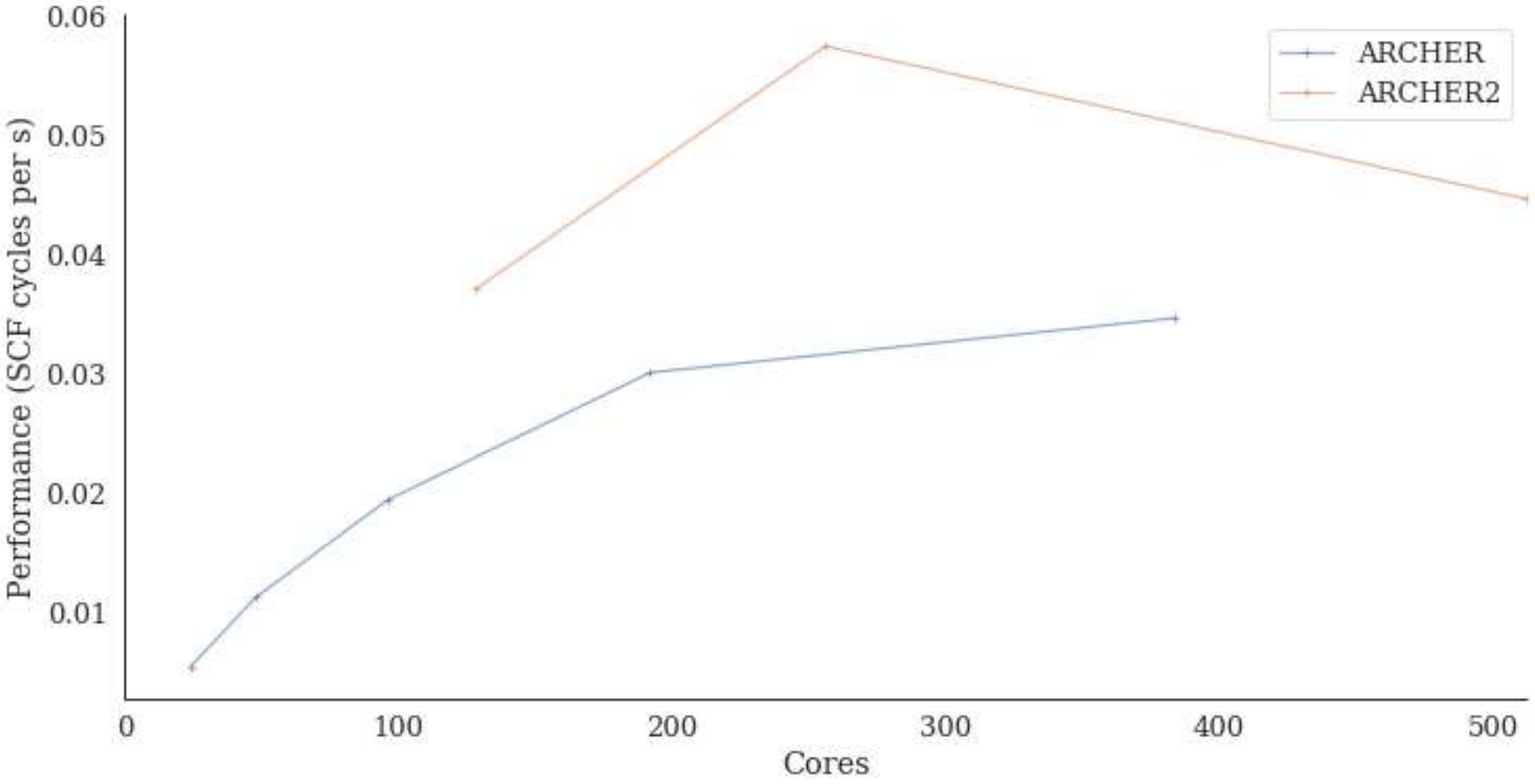
GROMACS: 1400k atoms



Single node comparison

System	Performance (ns/day)	cf. ARCHER
ARCHER	1.049	1.000
ARCHER2	5.581	5.320

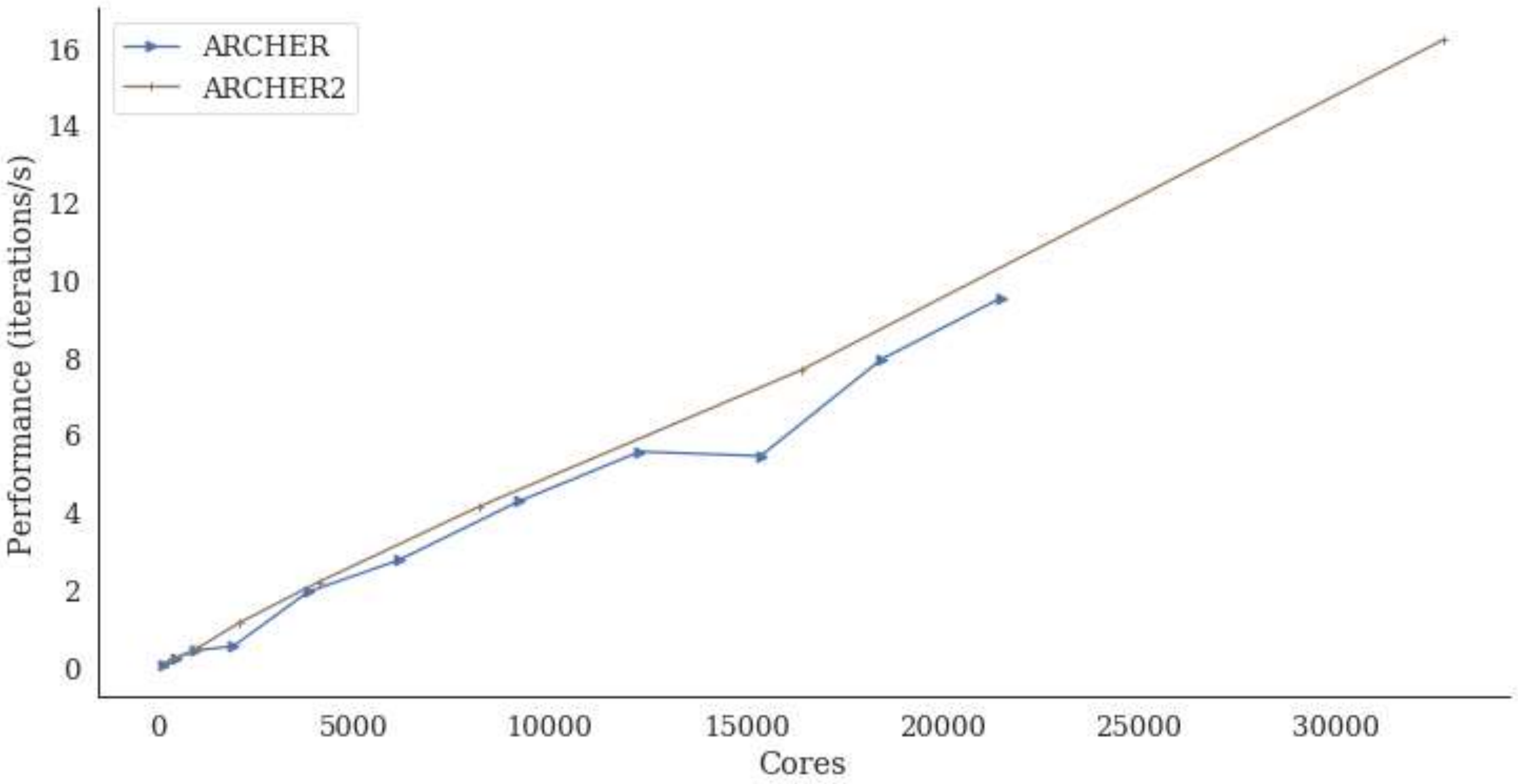
CASTEP: Al Slab benchmark



Single node comparison

System	Performance (SCF cycles/day)	cf. ARCHER
ARCHER	0.00552	1.000
ARCHER2	0.03711	6.724

OpenSBLI: 1024^3 Taylor-Green Vortex



5 node comparison

System	Performance (iterations/s)	cf. ARCHER
ARCHER	0.053	1.00
ARCHER2	0.270	5.09

Summary

Summary

- 4 cabinet service will offer similar core count to ARCHER (fewer nodes)
- Less resiliency and reduced functionality in 4 cabinet service compared to ARCHER
- Some differences in module environment and batch system
- Full set of EPCC support available for 4 cabinet service
- Initial performance looks promising