ARCHER2 Capability Days

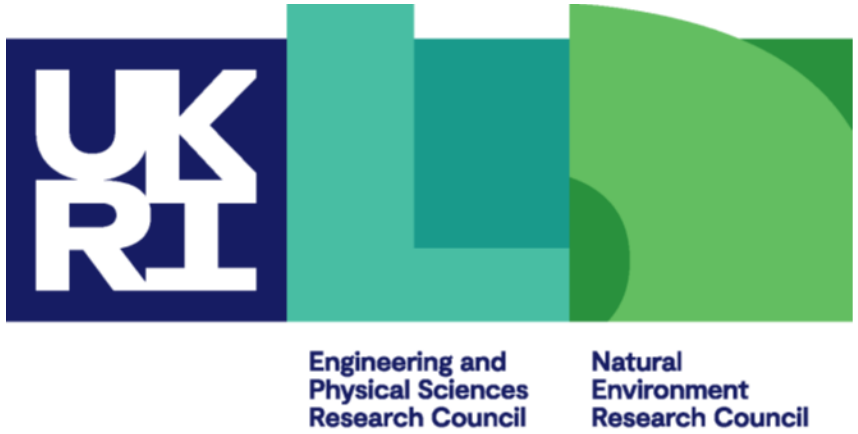
James Richings
EPCC, The University of Edinburgh
j.richings@epcc.ed.ac.uk
www.archer2.ac.uk

archer2

epcc

ARCHER2 Partners

epcc



epcc

THE UNIVERSITY of EDINBURGH

Hewlett Packard Enterprise

Capability Days (CD) 3: save the date

- Capability Days 3: 08:00 Tue 26 – 14:00 Thu 28 Sep 2024

- Capability Days 2: 08:00 Tue 4 – 14:00 Thu 6 Jun 2024
- Capability Days 1: 09:00 Thu 14 – 09:00 Fri 15 Mar 2024

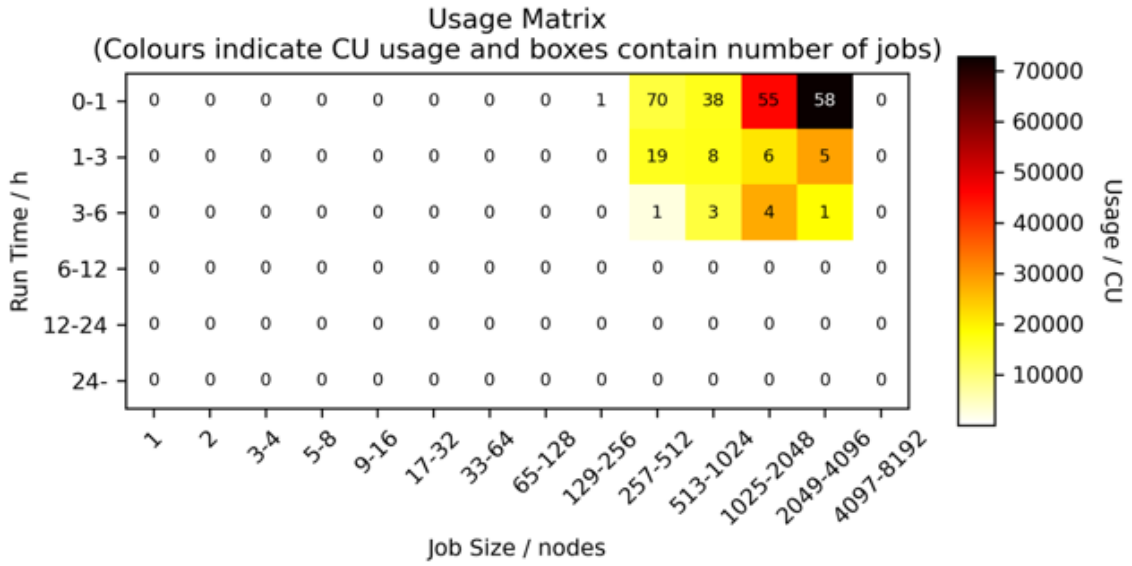
Motivation

- Provide a facility that can be used to test at scale to help prepare software and communities for future resources
- Enable capability use cases not possible on other UK HPC services
- Enhance world-leading science from ARCHER2 by enabling modelling and simulation at scales that are not otherwise possible

Addressing barriers

- Some technology issues with running at scale
  - Situation substantially improved since major upgrade in first half of 2023
  - Large jobs are much more reliable
- Lack of available budget and storage resources
  - Capability Days allows large jobs to run free of charge
  - Availability of scratch solid state storage allows users to go above usual quota
- Potential impact on day-to-day use of ARCHER2
  - Aim to reduce disruption to usual day-to-day use of ARCHER2
  - Some standard resource remains available during Capability Day period

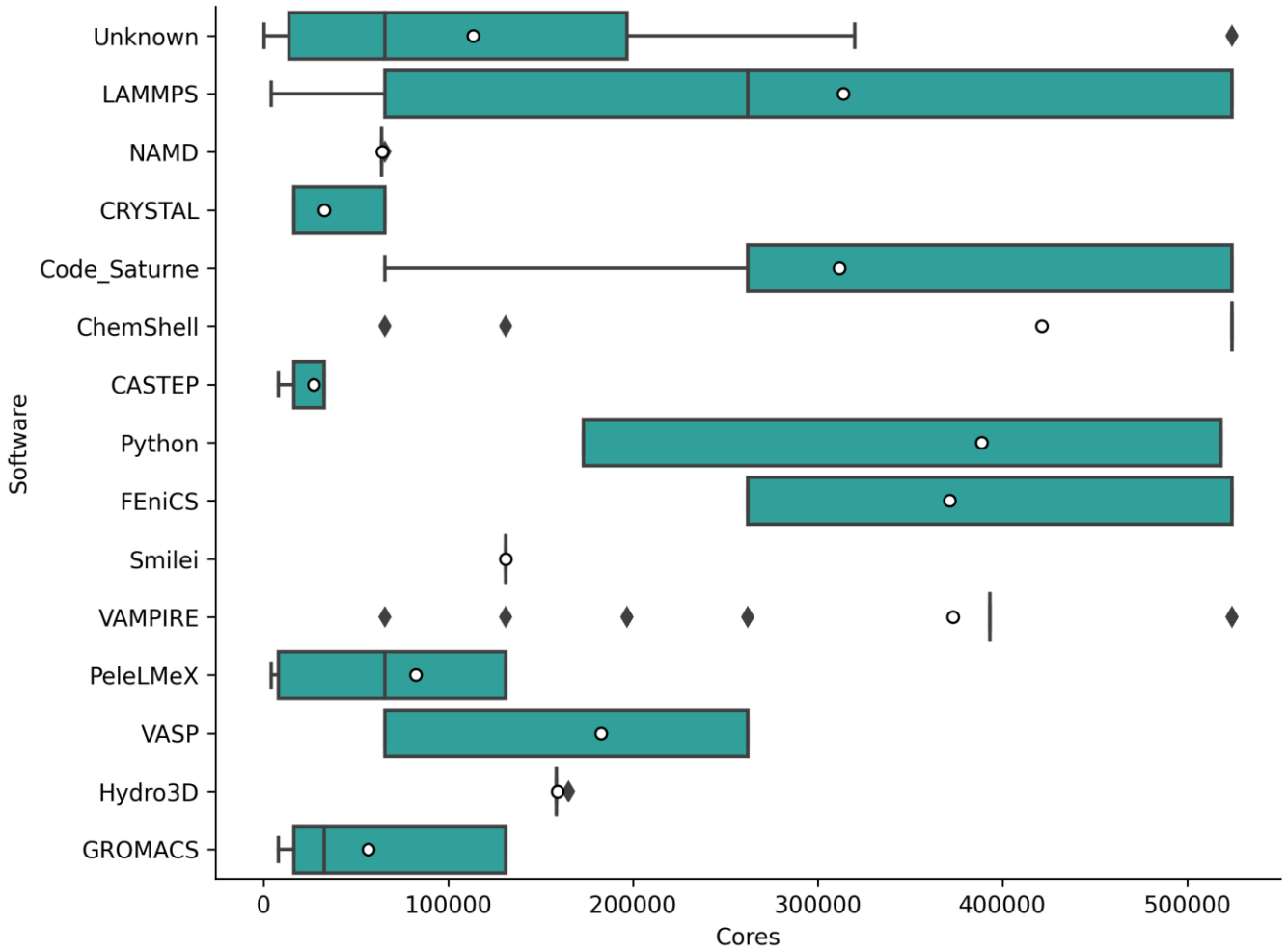
Stats from previous Capability Days



Broad range of different software types and research areas

Note: maximum MPI process count on Frontier (exascale system) likely around 75,000 (8 MPI processes per node, 1 per GPU) – equivalent to just under 600 fully-populated ARCHER2 nodes.

Job size distribution by software



Capability Days summary

Session	Runs	Options	Job size limits	Job time limits	Notes
Pre-capability Day	08:00 – 20:00 Tue 24 Sep	`--qos=pre-capabilityday`	256 – 1024 nodes	20 mins	+ve budget required
NERC Capability	08:00 – 16:00 Tue 24 Sep	`--reservation=NERCcapability --qos=reservation`	Up to 1024 nodes	N/A	NERC only +ve budget required
Capability Day	20:00 Tue 24 – 14:00 Thu 26 Sep	`--qos=capabilityday`	512 – 4096 nodes	1 hour	+ve budget required

- Jobs that run during any of these sessions will be uncharged
- Users can queue up work beforehand
- Reduced number of nodes available for standard ARCHER2 jobs – users should expect reduced throughput and longer wait times for standard ARCHER2 jobs

Practical information

- Documentation
  - https://docs.archer2.ac.uk/user-guide/scheduler/#capability-days
  - Includes full details on limits and example job submission scripts
- Limits on number of jobs queued and running based on budget code, not user
  - Aim to try to give a wide variety of projects access
- May want to use scratch (NVMe) storage for jobs
  - For both performance at scale and increased quota
  - Access needs to be requested via service desk in advance if not already setup
  - Check your user account page in SAFE to see if you have access

z19 resources	home (a2fs-home3)	1 GiB		
	general (rdfaas_general)	248 GiB	4,995 GiB	2,996 Files
	epsrc (rdfaas_epsrc)	1 GiB		24 Files
	work (a2fs-work1)	2,163 GiB	75,000 GiB	124,822 Files
	work (a2fs-work2)	43,906 GiB	49,989 GiB	48,476,601 Files
	work (a2fs-work3)	1,746 GiB	44,999 GiB	830 Files
	scratch (a2fs-nvme)	1 GiB	250,000 GiB	262,717 Files


Tips and scaling advice
archer2
epcc

General tips

- Plan what you want to test ahead of time
- Test job submission scripts thoroughly before submitting
  - Frustrating to get to head of the queue and then fail due to a typo
  - Can be done using `short` QoS on small node count without running full applications
- If possible, test scaling (even on very short tests) up to 1024 nodes in pre-capability day or NERCcapability
  - Gives confidence that application and startup are working as expected
  - Startup may take longer than you expect as node count increases
- Consider combining runs at fixed node count
  - Wait once, run multiple tests
- If possible, submit jobs ahead of time

MPI at scale

- Review your MPI communication pattern to understand if it has the potential to scale
  - All-to-all patterns are unlikely to scale well to large node counts
- MPI transport protocol: experience shows that OFI (default) works more reliably at large scale than UCX
- UCX:
  - If you want to use UCX, the default ”ud” (unrealiable datagram) protocol sometimes requires a longer timeout
  - You may also want to try different protocols (may lead to a performance reduction)

Switching to UCX from OFI:

https://docs.archer2.ac.uk/user-guide/dev-environment/#switching-to-alternative-ucx-mpi-implementation

```
export UCX_TLS=ud,self,sm
export UCX_UD_MLX5_TIMEOUT=20m
export UCX_UD_TIMEOUT=20m
```

```
export UCX_TLS=self,rc,dc,sm
```

IO at scale

- Think about your IO pattern and whether it will scale to the node counts you want to use
- MPIIO can involve alltoall-like patterns
  - If all MPI processes are involved in MPIIO calls, scaling may be poor
- File-per-process or similar can scale well but care needed
  - If each process is reading/writing multiple very small files (< 10 kB) per iteration this can cause issues for the file system
  - Especially if each time the file is accessed it is opened/closed
  - Especially if all the files are in a single directory (should be no more than around 1000 files per directory)
  - Understand how many files you are likely to read/write, at what interval, what size they are and how they are structured (in directories)
  - Potential mitigation strategies
    - Reduce IO: read/write fewer files or read/write files less often in scaling tests
    - Consolidate IO calls: read/write more data per operation, do not open/close/stat files every time you perform IO (Python standard IO often causes issues like this)
    - Use solid state storage: read/write using the scratch NVMe file system (still need to pay attention to other points, this is not a “magic bullet”)

Python at scale

- Large numbers of processes importing Python modules can incur a large overhead
  - Consider using Spindle to manage startup
  - https://docs.archer2.ac.uk/user-guide/python/#running-python-at-scale
- Note that standard IO in Python can cause issues as individual IO operations include multiple file metadata operations. If you are accessing large numbers of files and/or accessing files at very high frequency this can overload the file system
  - Look to reduce IO load by reducing file count or frequency of access
  - Use dedicated IO libraries within Python (e.g. NetCDF, HDF5)

Summary

epcc

Summary

epcc

- Dates:
  - 08:00 – 20:00 Tue 24 Sep: pre-capabilityday session
  - 08:00 – 16:00 Tue 24 Sep: NERCcapability reservation
  - 20:00 Tue 24 – 14:00 Thu 26 Sep: capabilityday session
- Docs: https://docs.archer2.ac.uk/user-guide/scheduler/#capability-days
- Test before launch
  - Use the short QoS ahead of time to check script correctness
  - Use pre-capabilityday or NERCcapability to check application startup at scale
- Review MPI approach to understand if it may have scaling issues
  - All-to-all communication patterns unlikely to scale to high process counts
  - OFI or UCX reliable protocols recommended
- Review IO strategy
  - Ensure that you are not doing something that may cause file system overload issues
  - Consider reducing IO load by reducing frequency or amount of IO
  - Make use of scratch (NVMe) storage
- Feedback:
  - There will be a survey after this iteration of capability days please send any feedback you have to help us improve this for next time.