Monitoring and evaluating I/O performance of HPC systems

Emmanouil Farsarakis - @efarsarakis
Michèle Weiland - @micheleweiland
Adrian Jackson - @adrianjhpc
Mark Parsons – @mark_epcc

EASC, April 2016, Stockholm

NEXTGenIO facts

Project

- Research and innovation
- 36 month duration
- €8.1 million
- Approx. 50% committed to hardware development

Exascale is VERY challenging

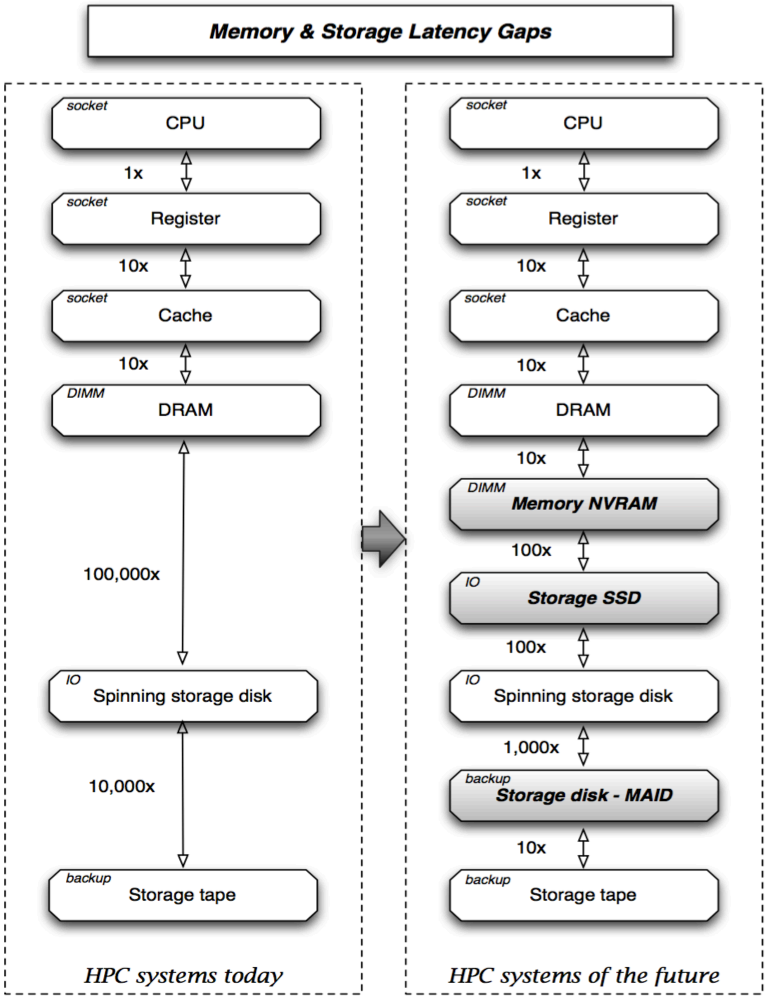
- In 1990 EPCC's 800 processor Meiko CS-1 delivered 8×10^{6} flops peak
- In 2015 our 118,080 core Cray XC30 delivers 2.5×10^{15} flops peak
- A 3.1 million times increase!
- Transition from Mega ➤ Giga ➤ Tera ➤ Peta has been challenging but largely incremental
- We've lived in a golden age of stability

I/O is a key exascale challenge

- Parallelism beyond 100 million threads demands a new approach to I/O
- Today's Petascale systems struggle with I/O
  - Inter-processor communication limits performance
  - Reading and writing data to parallel filesystems is a major bottleneck
- New technologies are needed
  - To improve inter-processor communication
  - To help us rethink data management and processing on capability systems

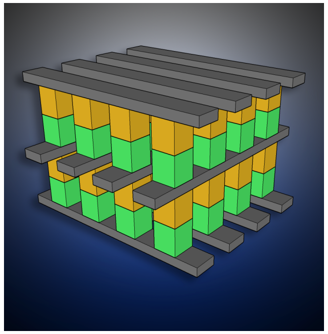
A new hierarchy

- Next generation NVRAM technologies will profoundly changing memory and storage hierarchies
- HPC systems and Data Intensive systems will merge
- Profound changes are coming to ALL Data Centres
- ...but in HPC we need to develop software – OS and application – to support their use



NEXTGenIO Objectives

- Develop a new server architecture using next generation processor and memory advances
  - Based on Intel Xeon and 3D XPoint technologies

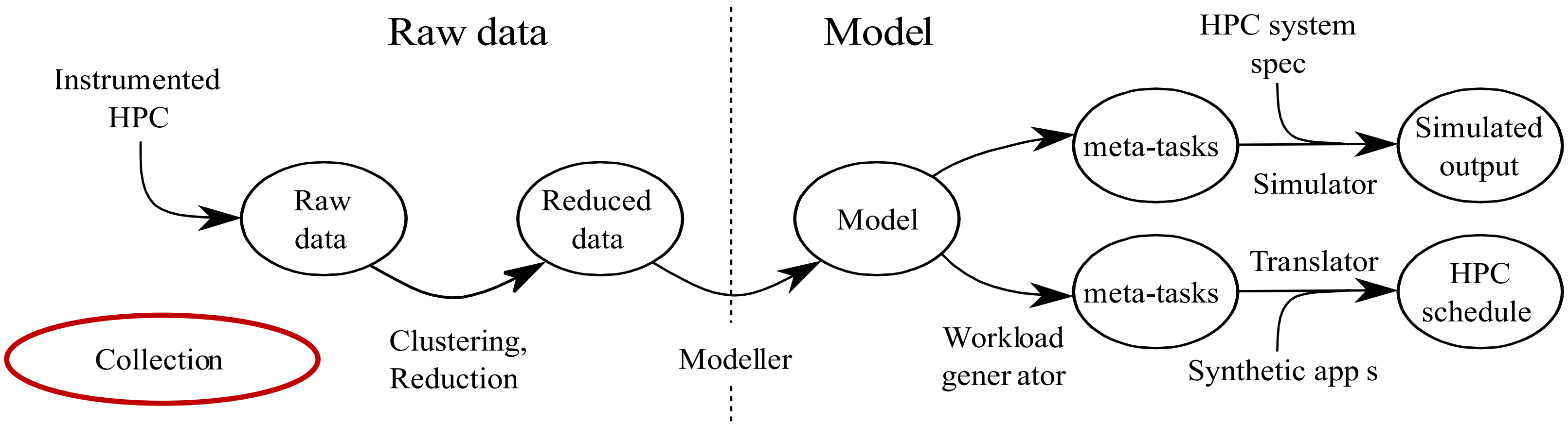


- Investigate the best ways of utilising these technologies in HPC
  - Develop the systemware to support their use at the Exascale
- Model three different I/O workloads and use this understanding in a co-design process
  - Representative of real HPC centre workloads

New hardware design and assessment

- I/O workload simulator (IOWS)
  - Experiment with system setups and configurations
  - Understand impact of changes on overall system throughput and performance
- Need to understand current systems
  - How are they being used?
  - What are the bottlenecks?
- We are looking at multiple systems
  - ARCHER → this talk
  - ECMWF
  - Arctur

IOWS architecture



- ***Data gathering*.** The HPC systems will be instrumented.
- ***Data reduction*.** The collected data will be classified into "classes" of similar jobs, and the parameter space reduced.
- ***Workload model/schedule*.** The workload will be mapped onto a set of idealised tasks (meta-tasks) and a scalable schedule will be designed
- ***Model simulation.*** Given a concrete HPC specification (scheduler, node topology, etc.), the execution of the meta-tasks is simulated.
- ***Model execution*.** A real schedule of synthetic applications is run on a real HPC system according to the meta-tasks.

ARCHER in a nutshell

- UK's national HPC facility
- Used by >3500 scientists from wide range of domains
- ~90% utilisation

- Cray XC30
- 4920 compute nodes
- High Performance Lustre filesystem

Required metrics

Vast amount of parameters can be collected:

- Many components in HPC stack
- Myriad types of running jobs

↓

We can’t assume we know what we need before we have measured & analysed it!

Required metrics

Collected data will generally fall into one of two categories:

1. System workflow data
   - Submission, queueing, execution times, …
2. Job behaviour on the system
   - Resources used, type of I/O, amount of I/O, …

Methodology

- Ideally we want system wide metrics for all data collected
  - ➞ This may not be possible
  - ➞ PBS, ALPS, …
- Alternative means of collecting data by indirect estimation
  - ➞ Profiling important codes
  - ➞ CrayPat, Allinea tools, …

Methodology walkthrough

As previously mentioned:

There are many many parameters to consider.

First step:

Monitor system for given timeframe

Keep:

PBS information

ALPS information

Cray I/O monitoring tool

- Submission time
- Execution time
- Req. resources
- Executable name
- MPI procs
- OpenMP threads
- Node list
- KB read/write
- Read/Write ops
- Metadata
- CPU utilisation
- Memory use
- Comms

Methodology walkthrough

PBS

Provides little information other than:

- **Job ID**
- Requested **number of nodes**
- Submit/execute/complete **times**

- Submission time
- Execution time
- Req. resources
- Executable name
- MPI procs
- OpenMP threads
- Node list
- KB read/write
- Read/Write ops
- Metadata
- CPU utilisation
- Memory use
- Comms

Methodology walkthrough

PBS

Can be taken from PBS logs directly or…

ARCHER safe:

- Can generate reports in mutliple formats from DB with PBS logs

- Submission time
- Execution time
- Req. resources
- Executable name
- MPI procs
- OpenMP threads
- Node list
- KB read/write
- Read/Write ops
- Metadata
- Comms

Job-ID	Submit	Start	End	User	Budget	Machine	JobName	Tasks	CPUs	Nodes	Queue
3642878.sdb	22/04/16 01:17	22/04/16 02:06	22/04/16 02:11	adrianj2	e281-sla	ARCHER	run3072	3072	3072	128	standard

Methodology walkthrough

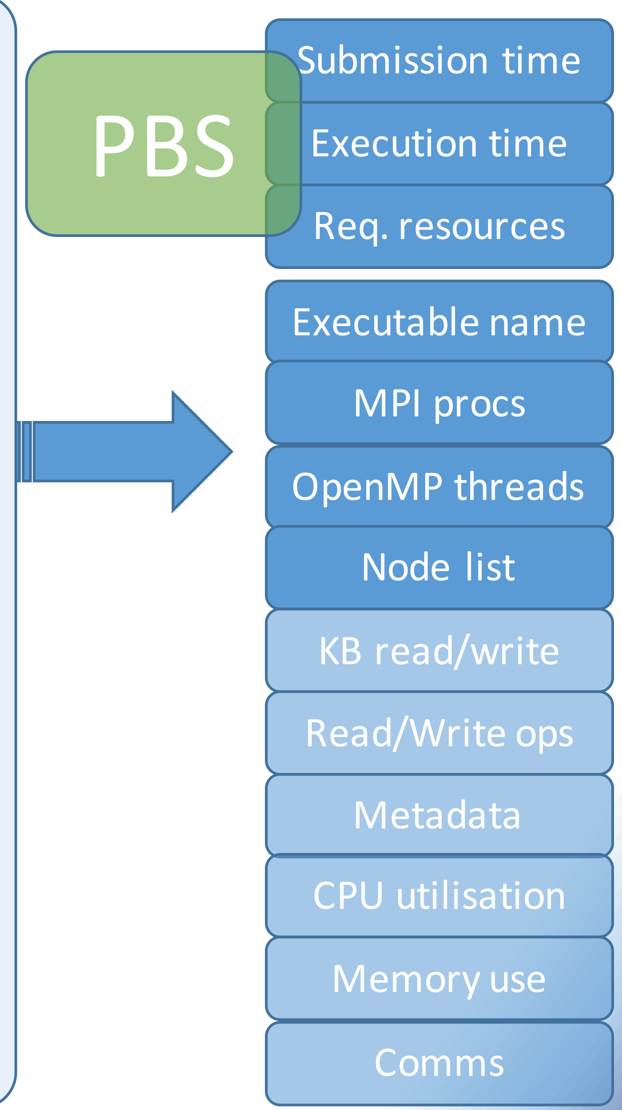
ALPS

(Application Level Placement Scheduler)

Provides more detail on:

- Executable **name**
  - What application is running?
- **Node** list
- **MPI** processes
- **OMP_NUM_THREADS***
- Thread/proc **placement***

*Can be assumed from “aprun” command



Methodology walkthrough

ALPS

```
$: more alps.log
<150>1 2016-04-22T01:17:49.441684+00:00 c2-1c0s0n1 aprun 9020 p0-20160127t131237 [alps_msgs@34] apid=20775696, Finishing, user=61000, batch_id=3642878.sdb, cmd_line="aprun -n 1536 /work/z01/z01/ajackson2/gs2/gs2 test.in", num_nodes=128, node_list=2673-2801,2887, cwd="/proj/z01/z01/ajackson2/gs2"
…
```

← PBS link

- Parsing can be **very** tedious
  - These are sys-admin logs NOT performance/monitoring logs

PBS

- Submission time
- Execution time
- Req. resources

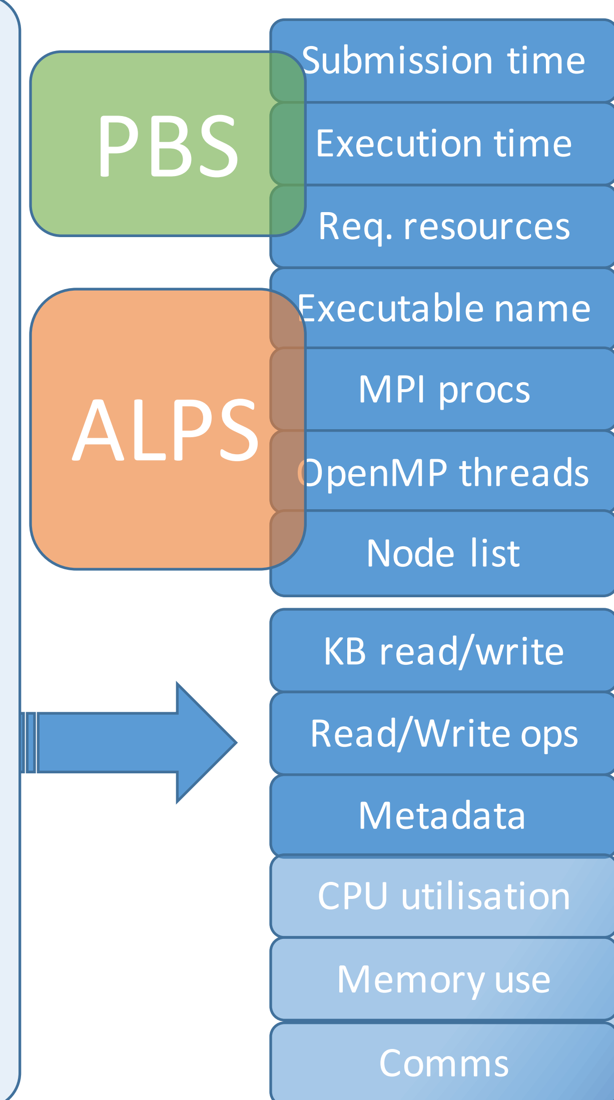
- Executable name
- MPI procs
- OpenMP threads
- Node list
- KB read/write
- Read/Write ops
- Metadata
- CPU utilisation
- Memory use
- Comms

Methodology walkthrough

Cray I/O monitoring tool

Use node allocation and times from PBS and ALPS to correlate/aggregate per-node information:

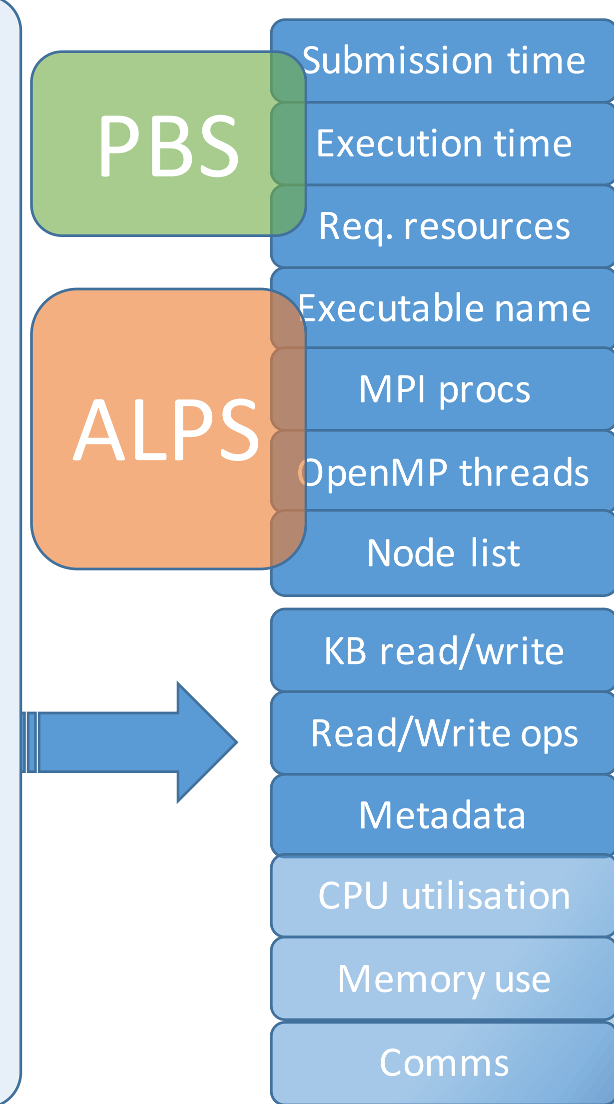
- **KB** read/write
- Read/Write **ops**
- **Metadata**
  - mkdir, open, rm, …



Methodology walkthrough

Cray I/O monitoring tool

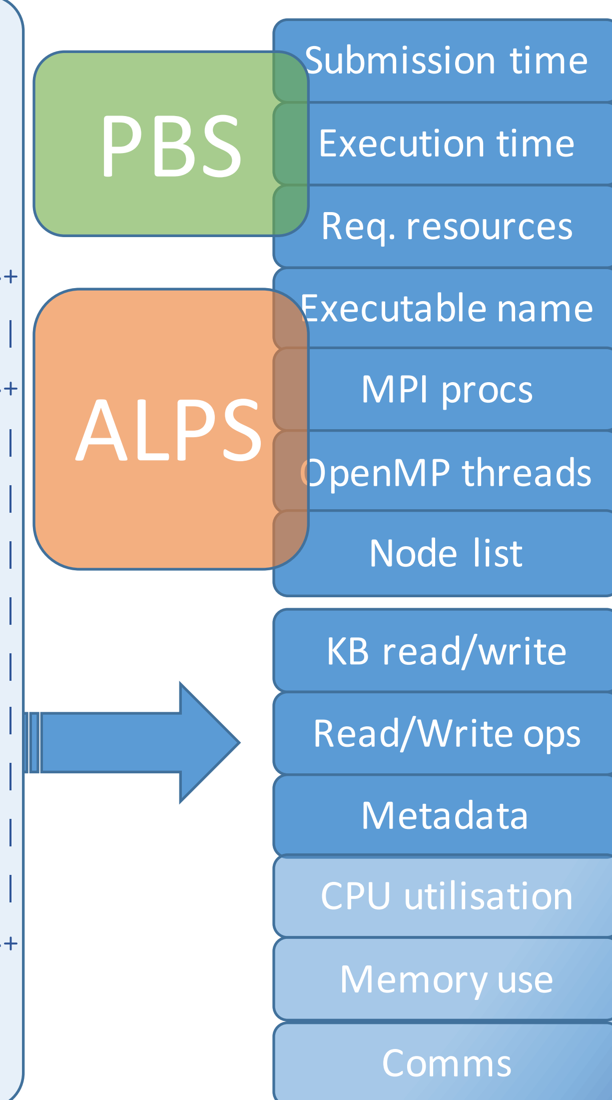
- Exploits Lustre I/O counters on each OST of each OSS and for each compute node
- I/O granularity ~ 3mins
- Metadata granularity ~ 30s
- Data stored on SQL DB
- Can query data as required (per job, node, filesystem, …) as long as we have the required information (i.e. node list for a job)



Methodology walkthrough

SQL table for I/O

```
mysql> desc oss;
+-----------+--------------------+------+-----+---------+-------+
| Field     | Type               | Null | Key | Default | Extra |
+-----------+--------------------+------+-----+---------+-------+
| time      | int(10) unsigned   | YES  | MUL | NULL    |       |
| read_kb   | int(10) unsigned   | YES  |     | NULL    |       |
| read_ops  | int(10) unsigned   | YES  |     | NULL    |       |
| write_kb  | int(10) unsigned   | YES  |     | NULL    |       |
| write_ops | int(10) unsigned   | YES  |     | NULL    |       |
| other     | int(10) unsigned   | YES  |     | NULL    |       |
| client    | smallint(5) unsigned | YES |    | NULL    |       |
| oss       | smallint(6)        | YES  |     | NULL    |       |
| gni       | tinyint(3) unsigned | YES |     | NULL    |       |
+-----------+--------------------+------+-----+---------+-------+
```



Methodology walkthrough

Example data from oss table

```
mysql> desc oss;
mysql> select * from oss limit 2
    -> ;
```

time	read _kb	read _ops	write _kb	write _ops	other	client	oss	gni
1453890520	0	0	0	0	0	2175	14	0
1453890520	0	0	0	0	0	369	14	0

PBS

- Submission time
- Execution time
- Req. resources

ALPS

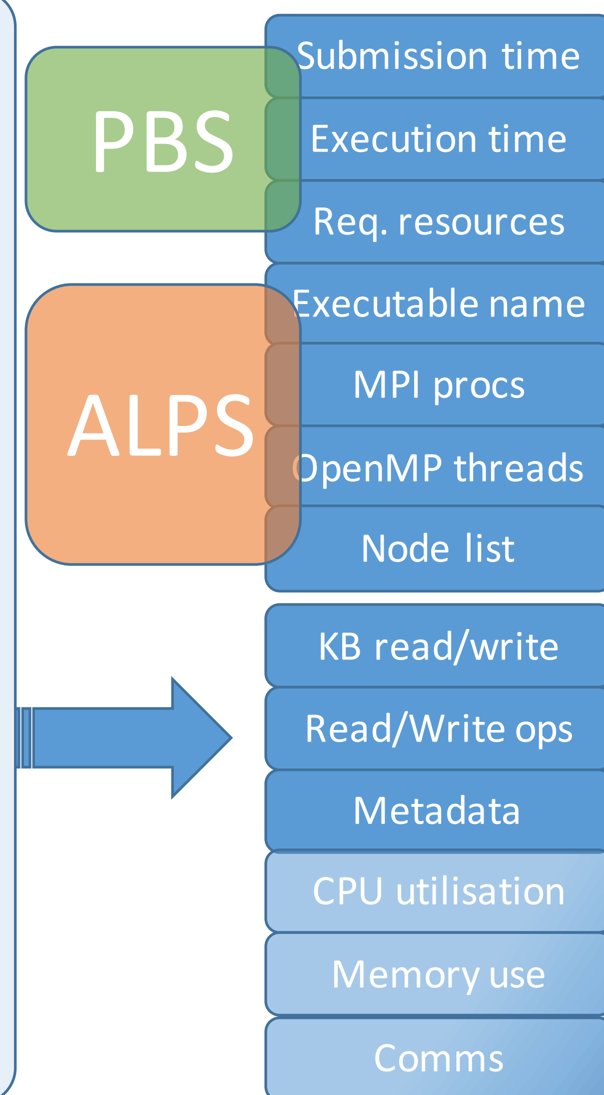
- Executable name
- MPI procs
- OpenMP threads
- Node list

- KB read/write
- Read/Write ops
- Metadata
- CPU utilisation
- Memory use
- Comms

Methodology walkthrough

SQL table for metadata

```
+----------+----------------------+------+-----+---------+-------+
| Field    | Type                 | Null | Key | Default | Extra |
+----------+----------------------+------+-----+---------+-------+
| time     | int(10) unsigned     | YES  |     | NULL    |       |
| open     | int(10) unsigned     | YES  |     | NULL    |       |
| close    | int(10) unsigned     | YES  |     | NULL    |       |
| mknod    | int(10) unsigned     | YES  |     | NULL    |       |
| link     | int(10) unsigned     | YES  |     | NULL    |       |
| unlink   | int(10) unsigned     | YES  |     | NULL    |       |
| mkdir    | int(10) unsigned     | YES  |     | NULL    |       |
| rmdir    | int(10) unsigned     | YES  |     | NULL    |       |
| ren      | int(10) unsigned     | YES  |     | NULL    |       |
| getattr  | int(10) unsigned     | YES  |     | NULL    |       |
| setattr  | int(10) unsigned     | YES  |     | NULL    |       |
| getxattr | int(10) unsigned     | YES  |     | NULL    |       |
| setxattr | int(10) unsigned     | YES  |     | NULL    |       |
| statfs   | int(10) unsigned     | YES  |     | NULL    |       |
  . . .
```

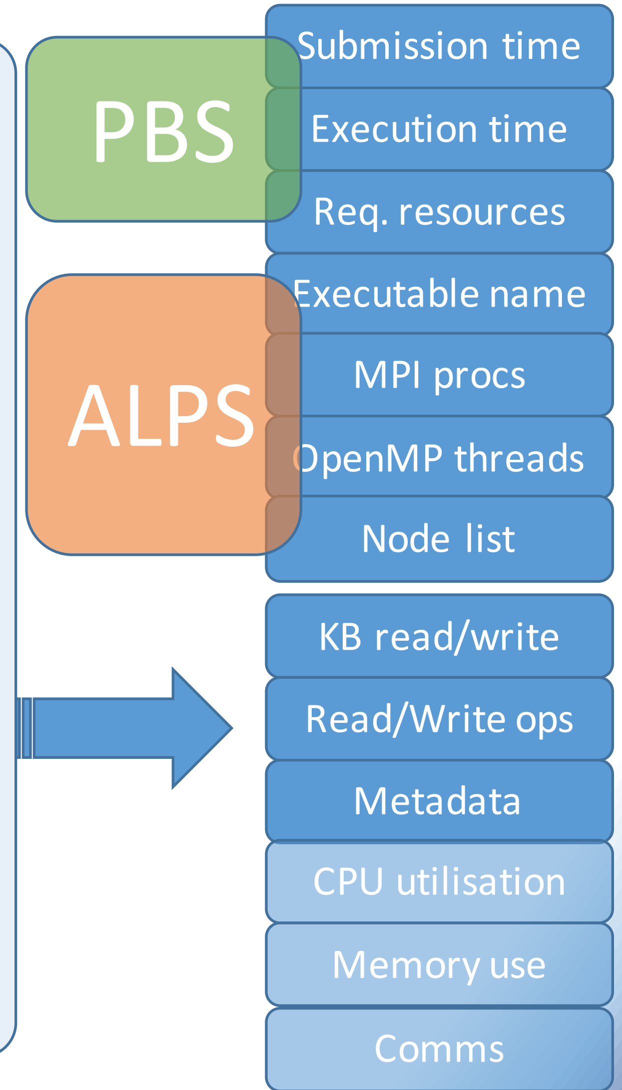


Methodology walkthrough

Example data from mds table

```
mysql> select * from mds limit 2
    -> ;
```

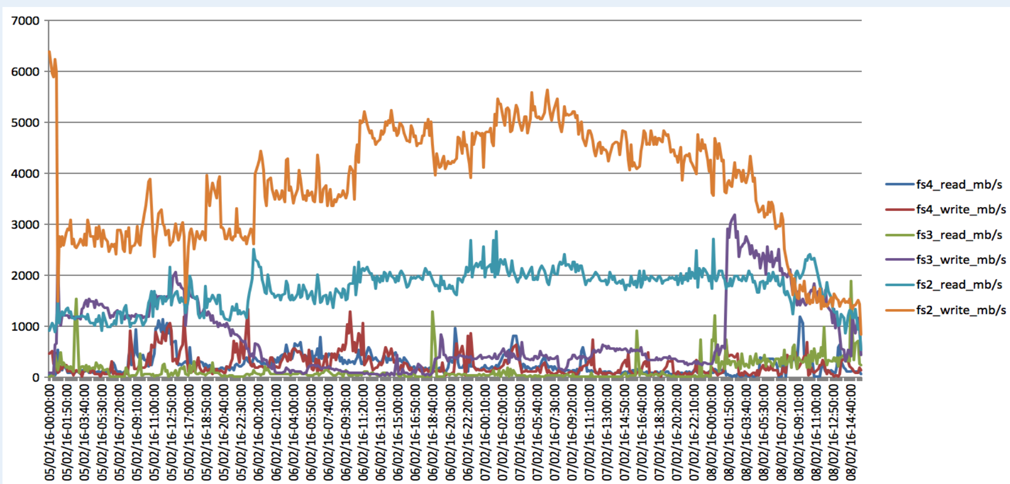
time	open	close	mkdir	rmdir	client	mds	…
1453890525	0	0	0	0	2175	3	…
1453890525	0	0	0	0	369	3	…



Methodology walkthrough

Cray I/O monitoring tool

- Example of per-filesystem I/O query for ~85hour window



- Metadata can show if FS2 problems are due to poor I/O strategy (i.e. many small files)

PBS

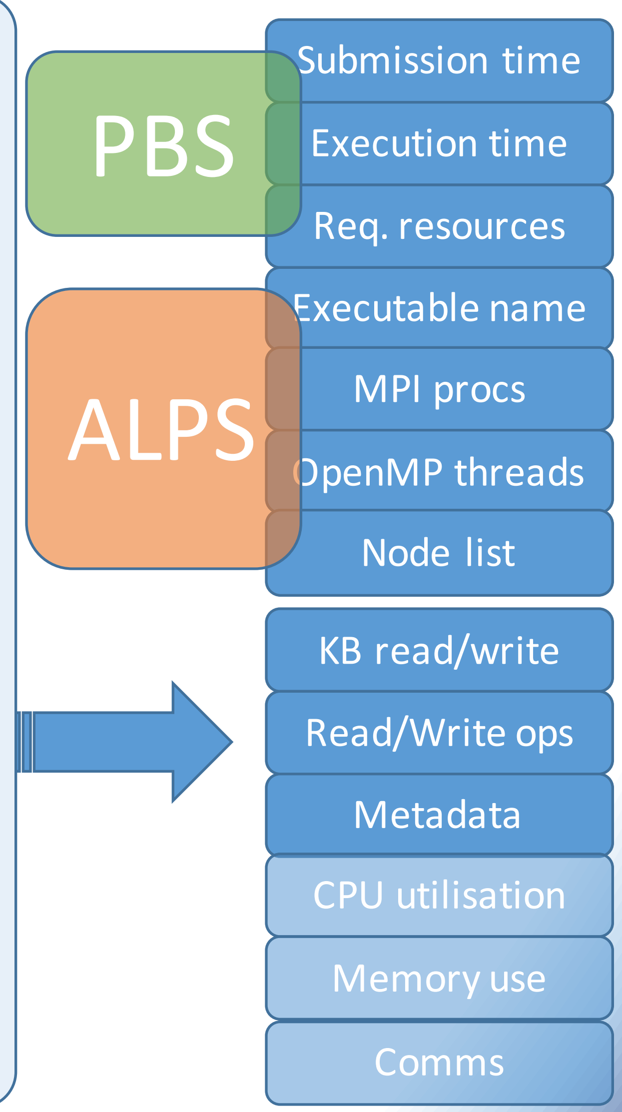
ALPS

- Submission time
- Execution time
- Req. resources
- Executable name
- MPI procs
- OpenMP threads
- Node list
- KB read/write
- Read/Write ops
- Metadata
- CPU utilisation
- Memory use
- Comms

Methodology walkthrough

Cray I/O monitoring tool

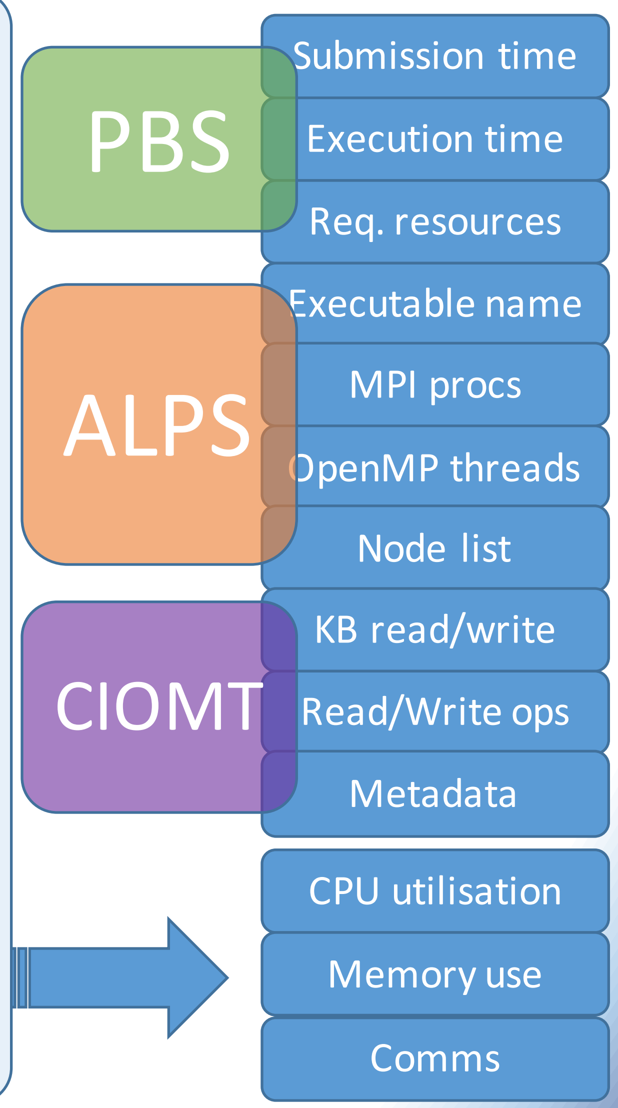
- Must link to ALPS and PBS data in order to query by job_id, user, group, …
- Currently can only be done manually



Methodology walkthrough

Intra-node metrics and comms

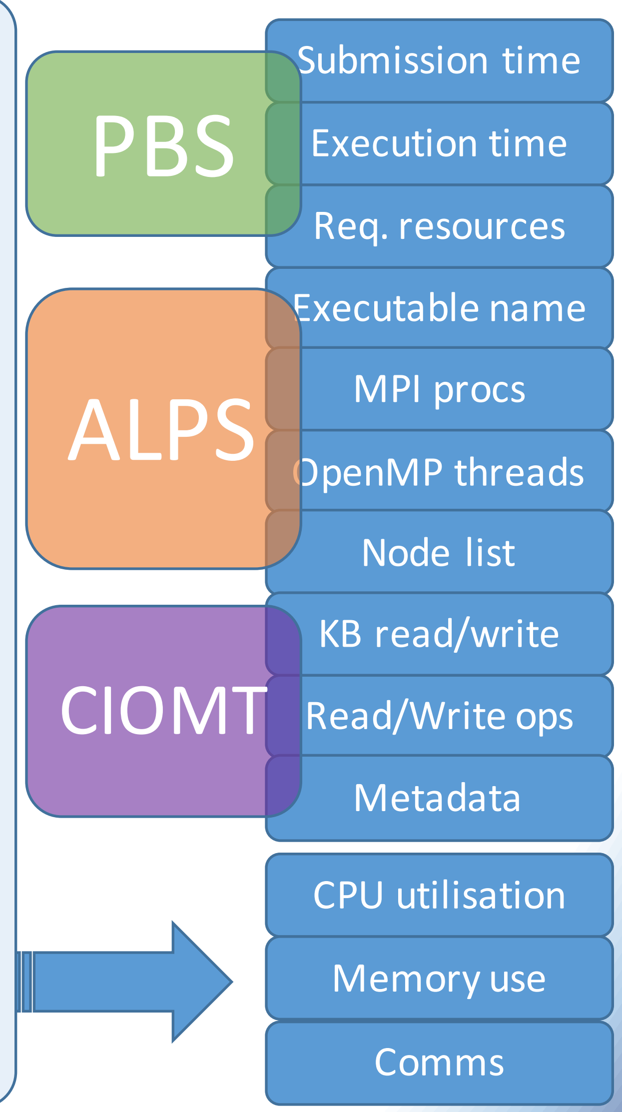
- Ideally would like to use profiler tool across the system for a (short) period of time:
  - Requires recompilation for codes using dynamic libraries
  - Would lead to ~5% overhead
    - Benchmark data unusable
    - Cost for user
  - Would require top level approval…



Methodology walkthrough

Indirect metrics

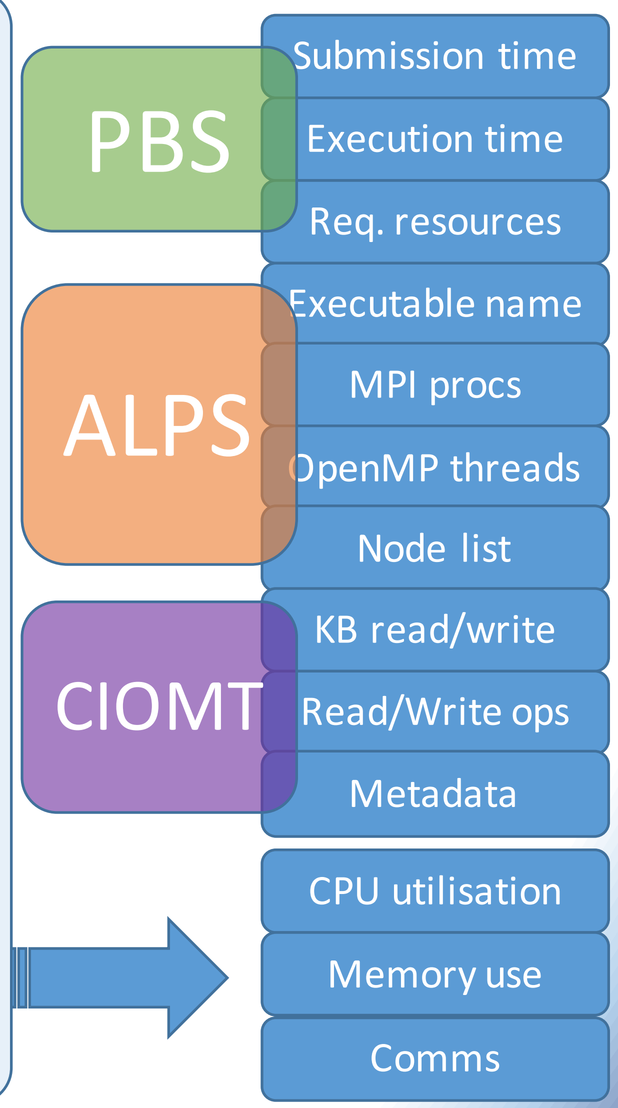
- **Identify** high utilisation **apps** using data from PBS (resources, time) and ALPS (application name)
- **Profile** applications using tools such as CrayPat and/or MAP.
- **Estimate** system-wide metrics based on results/utilisation



Methodology walkthrough

Profiler tools

Tool / Functionality	CrayPat	Allinea Performance Reports	Allinea MAP
Available for all partners	No	Yes	Yes
Power/Energy	Yes	No	Yes
Overhead	Low	Low	~5%
Tool support	Cray	Project Partner	Project Partner
time-trace	No	No	Yes



Methodology walkthrough

Which codes to profile

- Previous work at EPCC → correlate jobs with specific applications with 85-90% confidence.

Code type	Node hours	% Time
Materials Science	4683957	38.81
Climate/Ocean Modelling	2645213	21.92
Computational Fluid Dynamics	1608832	13.33
Biomolecular Simulation	1258204	10.43
Plasma Science	222644	1.84
Other defined	436130	3.61
Unknown	1899283	15.74

Q4 2015 ARCHER usage per application domain. Generated using https://github.com/aturner-epcc/archer-monitor

PBS
- Submission time
- Execution time
- Req. resources

ALPS
- Executable name
- MPI procs
- OpenMP threads
- Node list

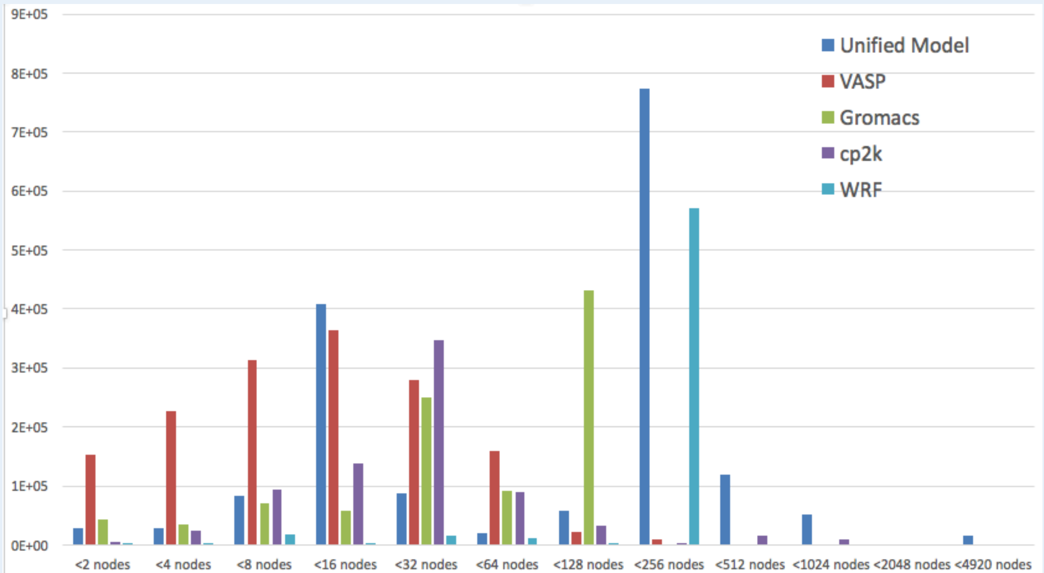
CIOMT
- KB read/write
- Read/Write ops
- Metadata

MAP
- CPU utilisation
- Memory use
- Comms

Methodology walkthrough

Which codes to profile

- Can identify job size distributions, by job numbers, node hours, …



Q4 2015 ARCHER top utilisation codes job size distribution. Generated using https://github.com/aturner-epcc/archer-monitor

PBS
- Submission time
- Execution time
- Req. resources

ALPS
- Executable name
- MPI procs
- OpenMP threads
- Node list

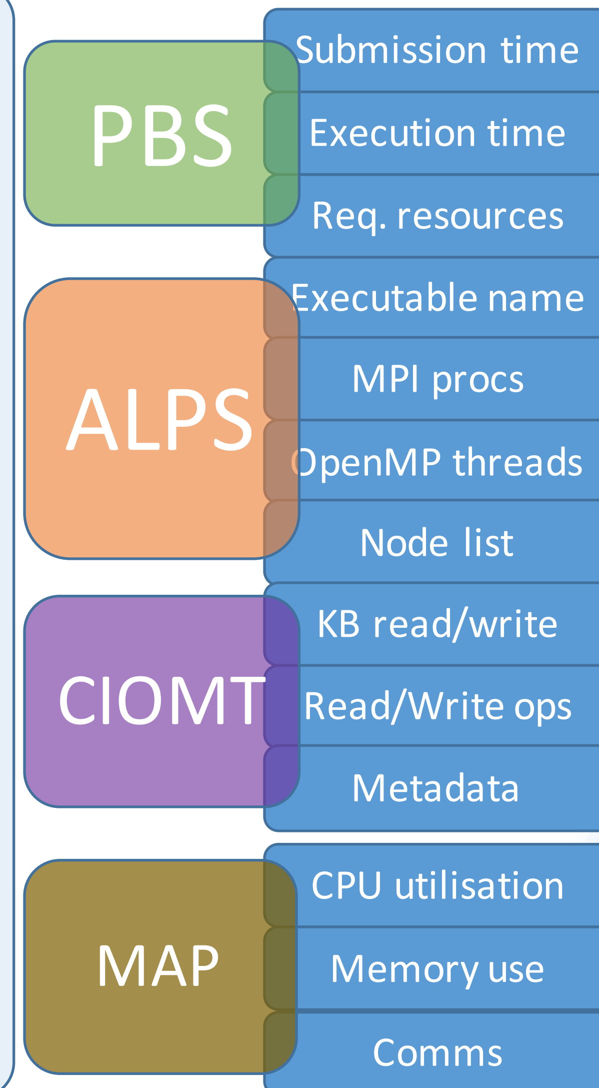
CIOMT
- KB read/write
- Read/Write ops
- Metadata

MAP
- CPU utilisation
- Memory use
- Comms

Methodology walkthrough

What about energy/power?

- Currently investigating the feasibility of using RUR on ARCHER
- Would log data on:
  - Energy consumption
  - Memory
- Alternatively can extract per-node power/energy data using CrayPat
  - Does not include disk or network power



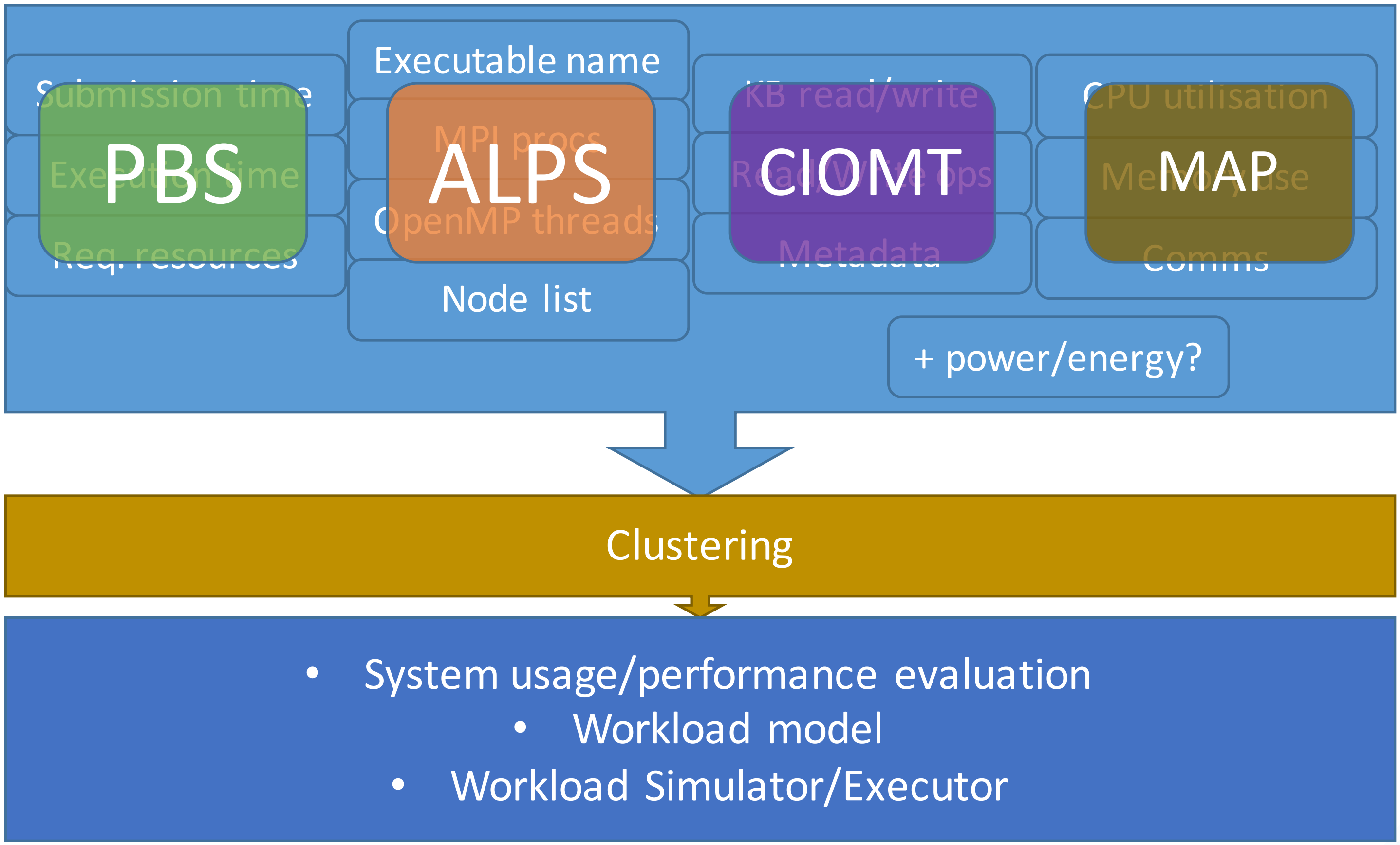
Methodology walkthrough

Clustering

- A clustering algorithm is used to cluster a set of samples according to selected parameters.
- It will be used to reduce the size of the workload parameter space
- Several algorithms/packages exist and will be explored within this project (e.g. python SciKit)
- Clustering can be applied to the HPC workload at several levels and should reflect sensible grouping strategies of the data.
    - **High level job meta-data:** number of processes, user ID, user behaviour, date/time, etc...
    - **Characteristics at application level**: discretized representation of time-traces of CPU, memory, IO metrics

Methodology walkthrough

Bring everything together



Conclusion

- Monitoring large-scale HPC systems is very complex
- There is a vast amount of data which may influence the performance of a system
- There are multiple sources from which to extract that data
- Difficult to balance
  - Need to **monitor** system
  - Not (significantly) ***disrupting*** performance
- Post-processing will be a challenging task

Thank you

Questions?

Michèle Weiland
@micheleweiland
m.weiland@epcc.ed.ac.uk
epcc