Monitoring and evaluation of I/O performance of HPC systems

Prof Mark Parsons, Project Coordinator

Dr Michèle Weiland

Mr Adrian Jackson

Mr Emmanouil Farsarakis

epcc The University of Edinburgh

NEXTGenIO project

Project

- Research & Innovation Action
- 36 month duration
- €8.1 million
- Approx. 50% committed to hardware development
- Prototype system available from Month 27

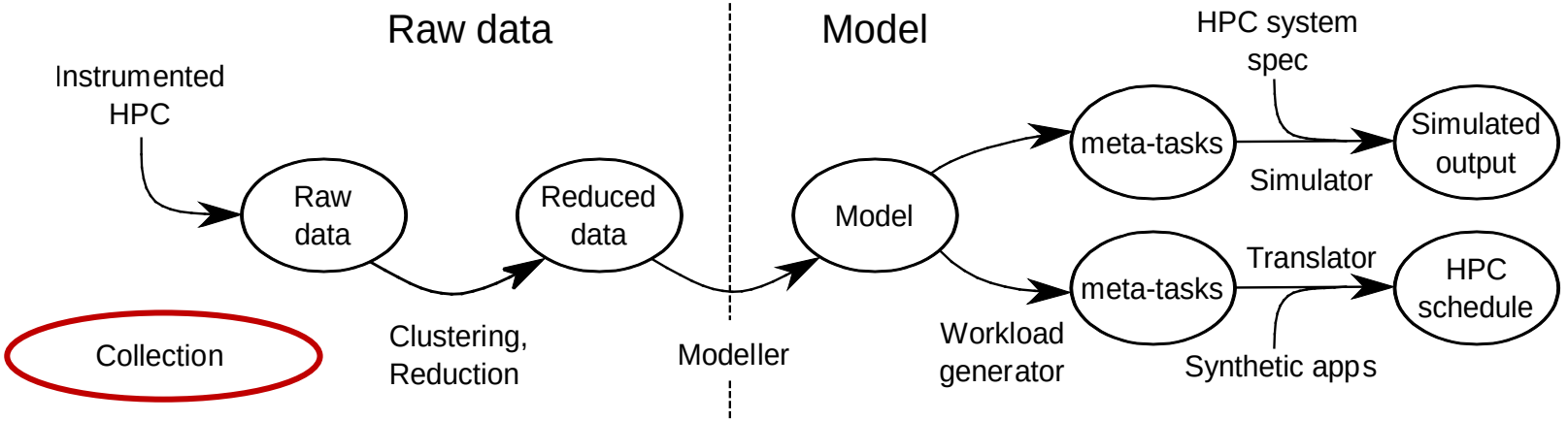
Partners

- EPCC
- INTEL
- FUJITSU
- BSC
- TUD
- ALLINEA
- ECMWF
- ARCTUR

I/O Workload Simulator

- 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.

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
  - → But 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

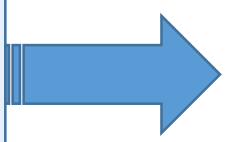
- Submission time
- Execution time
- Req. resources
- Executable name
- MPI procs
- OpenMP threads
- Node list
- KB read/write
- Read/Write ops
- Metadata
- CPU utilisation
- Memory use
- Comms

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

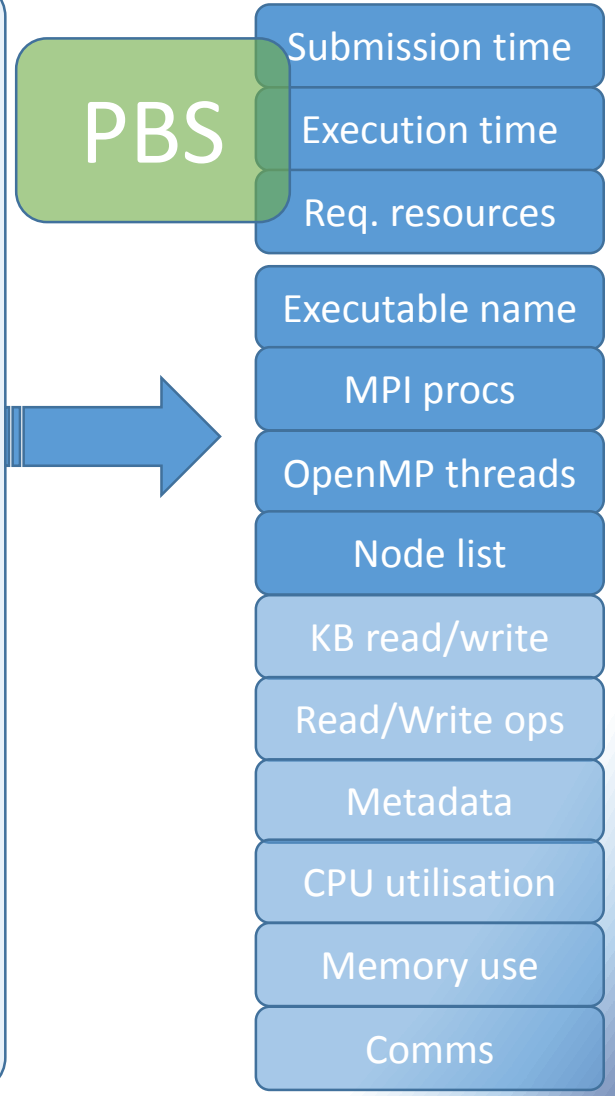
ALPS

(Application Level Placement Scheduler)

Provides more detail on:

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

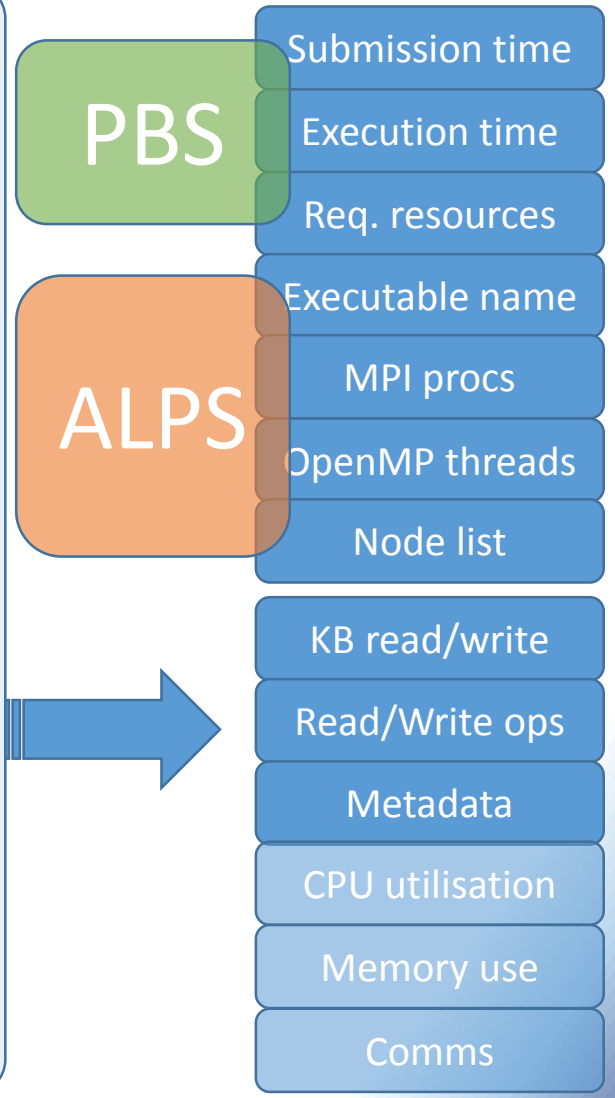
*Can be assumed from “aprun” command



Methodology walkthrough

Cray I/O monitoring tool

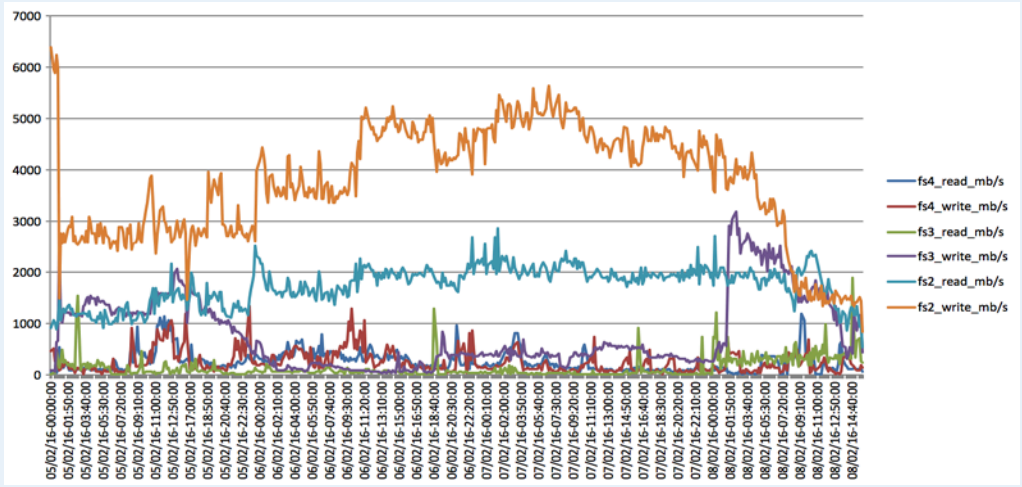
- Exploits Lustre I/O counters on each OST of each OSS and for each compute node
- I/O granularity ~ 3mins
- Metadata granularity ~ 30s
- Data stored on SQL DB
- Can query data as required (per job, node, filesystem, …) as long as we have the required information (i.e. node list for a job)



Methodology walkthrough

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

- Submission time
- Execution time
- Req. resources

ALPS

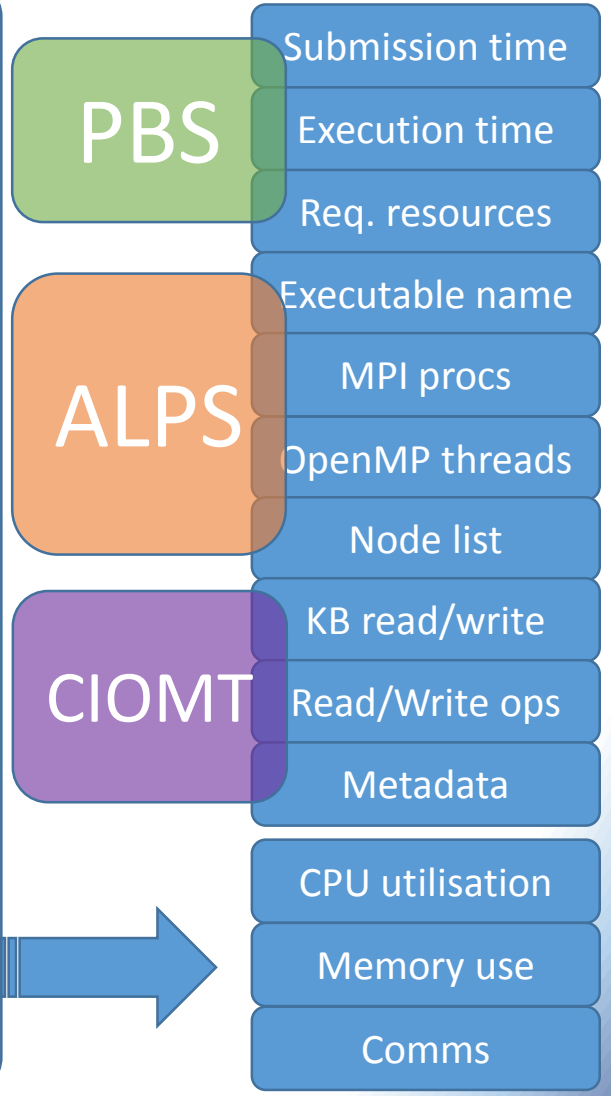
- Executable name
- MPI procs
- OpenMP threads
- Node list

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

Methodology walkthrough

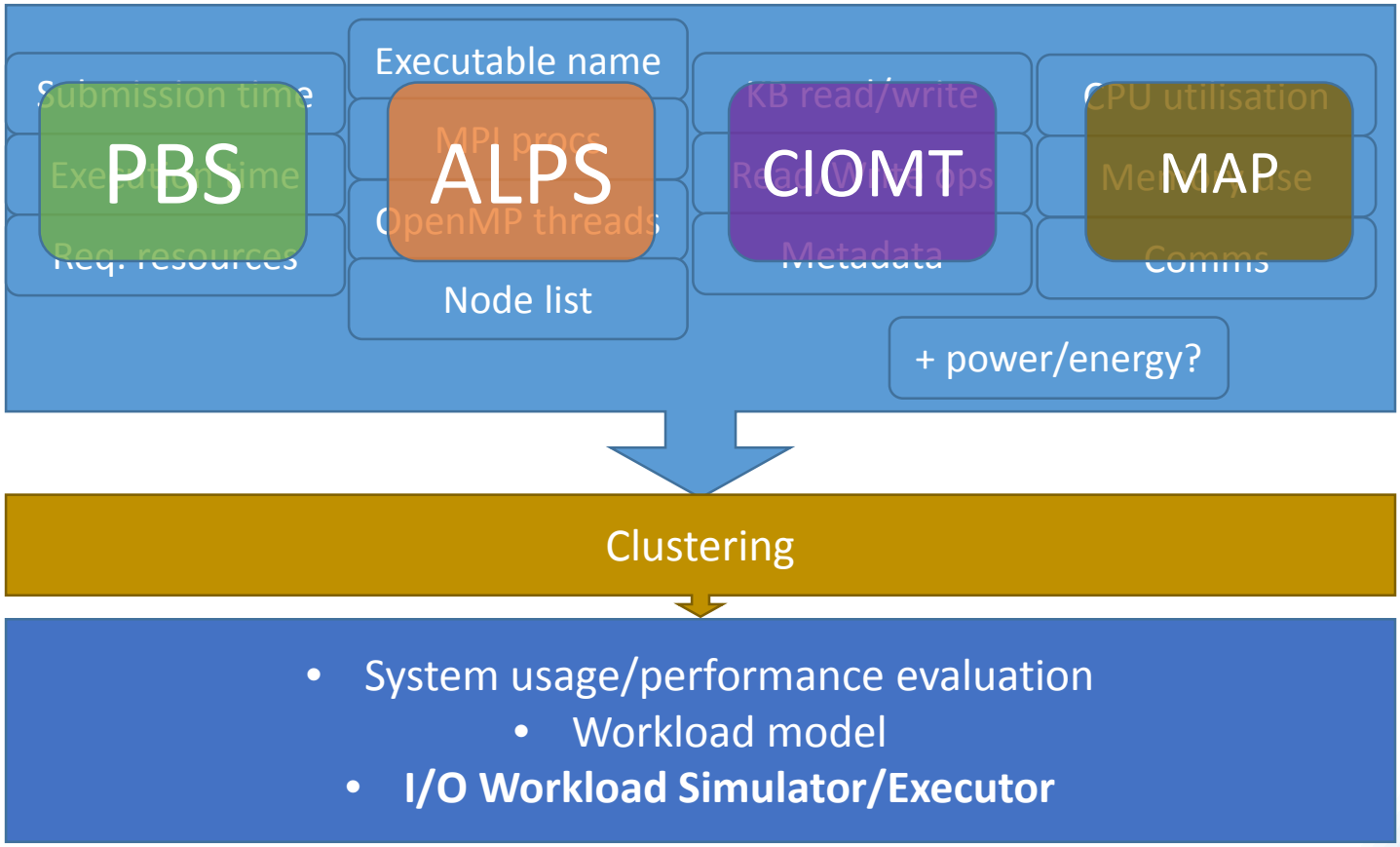
Indirect metrics

- **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

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