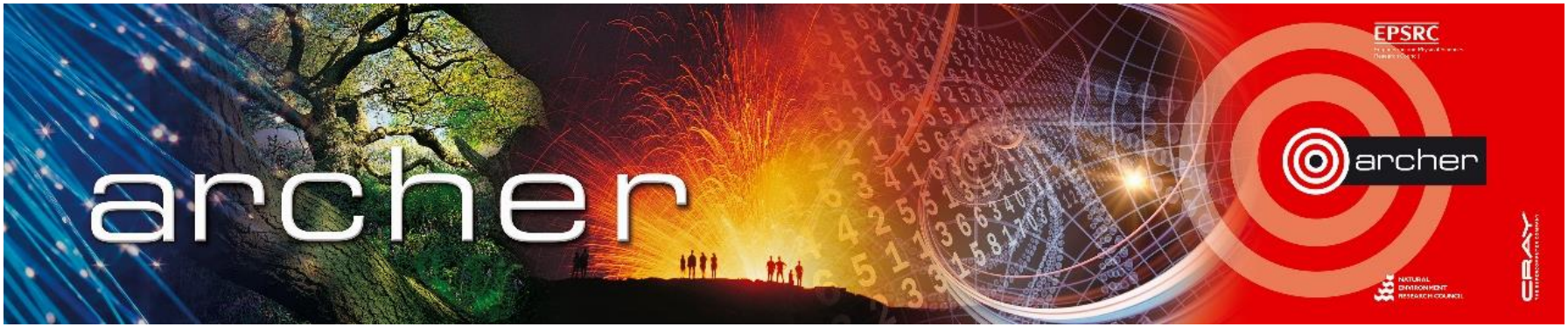


Welcome!

Virtual tutorial starts at 15:00 BST

archer

epcc

THE UNIVERSITY OF EDINBURGH

PBS Job Submission

ARCHER Virtual Tutorial, Wed 9th April 2014

David Henty <d.henty@epcc.ed.ac.uk>

Reusing this material

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

http://creativecommons.org/licenses/by-nc-sa/4.0/deed.en_US

This means you are free to copy and redistribute the material and adapt and build on the material under the following terms: You must give appropriate credit, provide a link to the license and indicate if changes were made. If you adapt or build on the material you must distribute your work under the same license as the original.

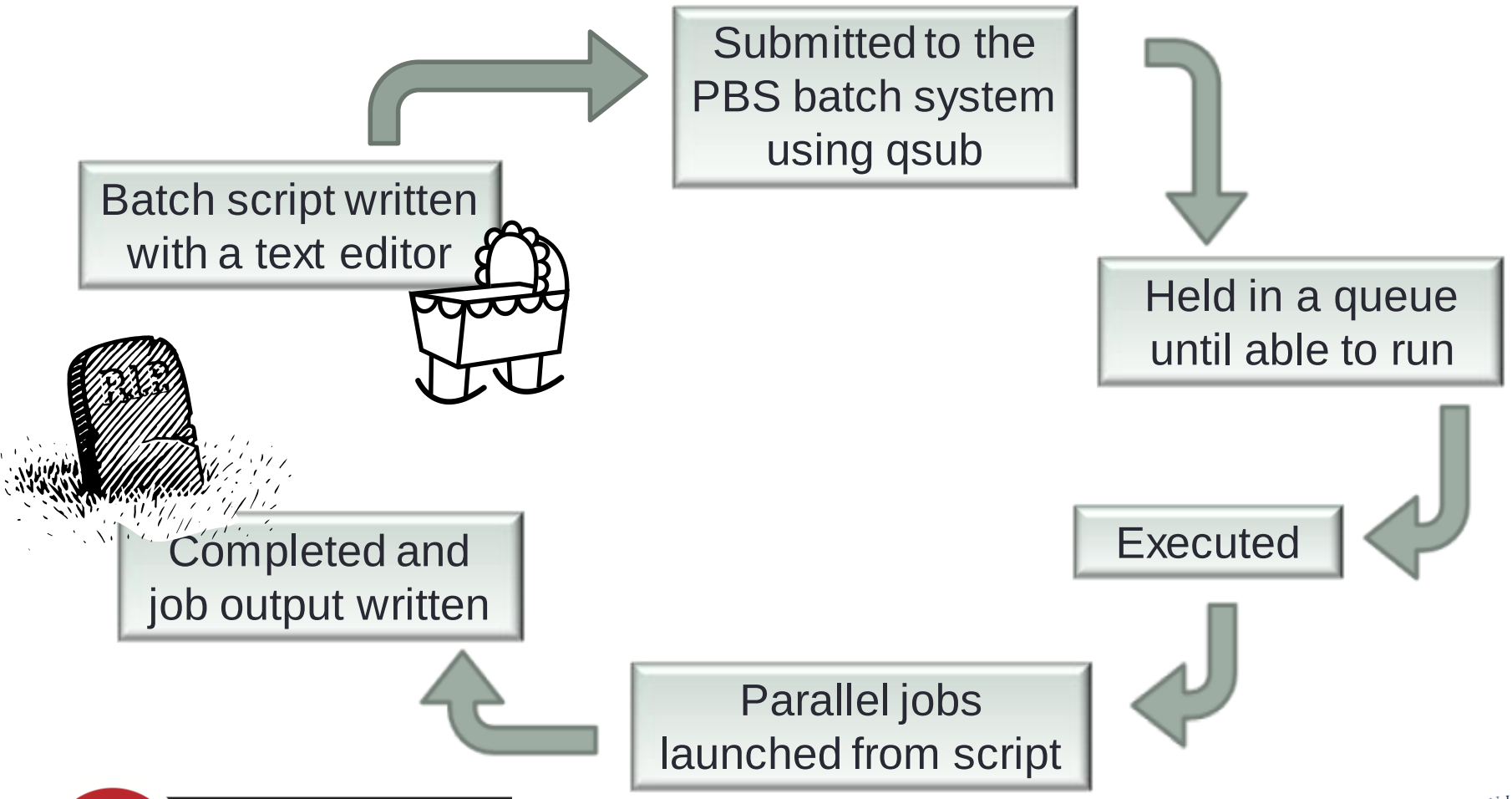
Note that this presentation contains images owned by others. Please seek their permission before reusing these images.

archer

epcc

THE UNIVERSITY OF EDINBURGH

A day in the life of a PBS job



archer

epcc

THE UNIVERSITY OF EDINBURGH

Conception: batch script written …

- What is a batch script?
  - a list of commands that are executed in order exactly as if you typed them into the shell on the command line
  - recommended to use bash
- Lines starting "#" are comments
- Except …
  - **`#!`** is special to operating system
  - **`#!/bin/bash --login`** # Run script as if a bash login session
- and
  - **`#PBS`** is special to batch system
  - **`#PBS -N myjob`** # Pass on as arguments to qsub

Great! So I can run the batch script in advance to check it works since it's just a bunch of commands

I'm afraid it's not that simple

Why?

Your batch script runs on a different computer in a different environment

So some commands that work on the login nodes won't work under PBS?

Actually, some commands that *don't* work on the login nodes *will* work under PBS

archer

epcc

THE UNIVERSITY OF EDINBURGH

Birth: submitted to batch system

```
user@archer> qsub -l select=6 myjob.pbs
123456.sdb
```

- PBS takes a *copy* of your batch script and stores it
  - ascertains resource requirements (e.g. no. of nodes)
    - from command line arguments
    - from **`#PBS`** lines
  - other resources: **`-l walltime=03:00:00`**
- Job is queued until resources are available
  - **qstat** job status set to “Q”

Great! So I can edit the same job script and resubmit straight away?

I'm afraid it's not that simple

Why?

You're probably running an executable from the script and it will see the version that's there at run time - OK for a package but not if you're recompiling

How does it decide when to run my job?

It's a balance of your requested number of nodes and the runtime compared to all the other jobs in the system

So I should take care when specifiying these?

Yes – don't ask for huge amounts of runtime if you don't need it!

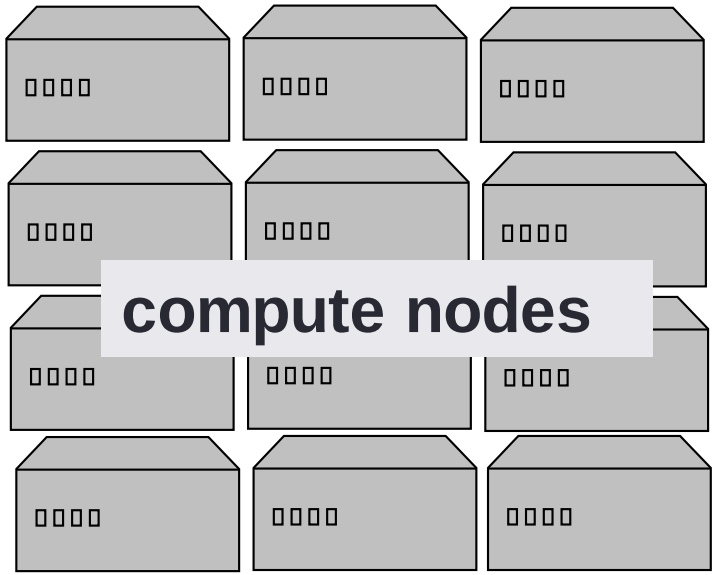
Childhood: job script runs

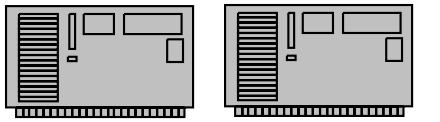
- A set of compute nodes is reserved for your job
  - but your batch script is actually executed on the **MOM nodes**
  - backronym for “Machine Oriented Mini-server”
  - **qstat** job status set to “R”

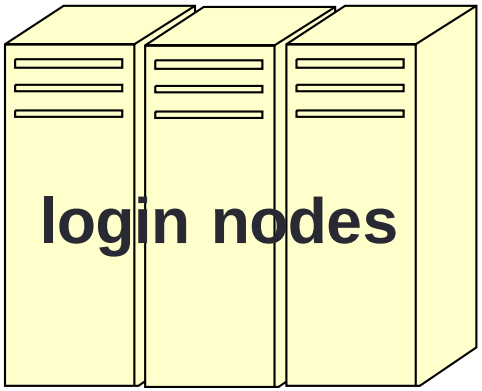
- The **only way** to access the compute nodes is with aprun

```
#PBS -N myjob
...
# Now run the job (I have used select=6)
aprun -n 144 mympiprogram
```

Operating Systems





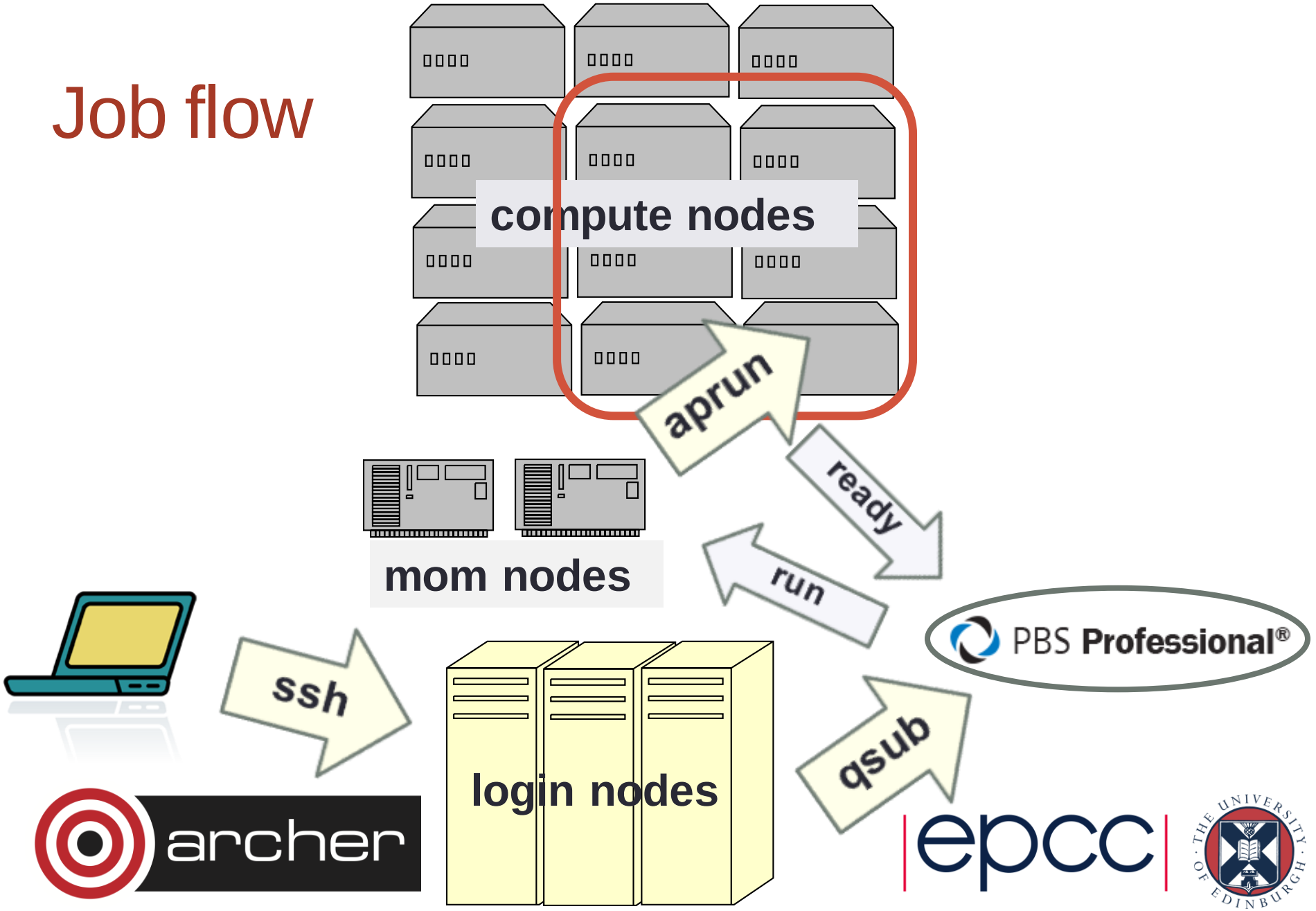


archer

epcc

THE UNIVERSITY OF EDINBURGH

Job flow

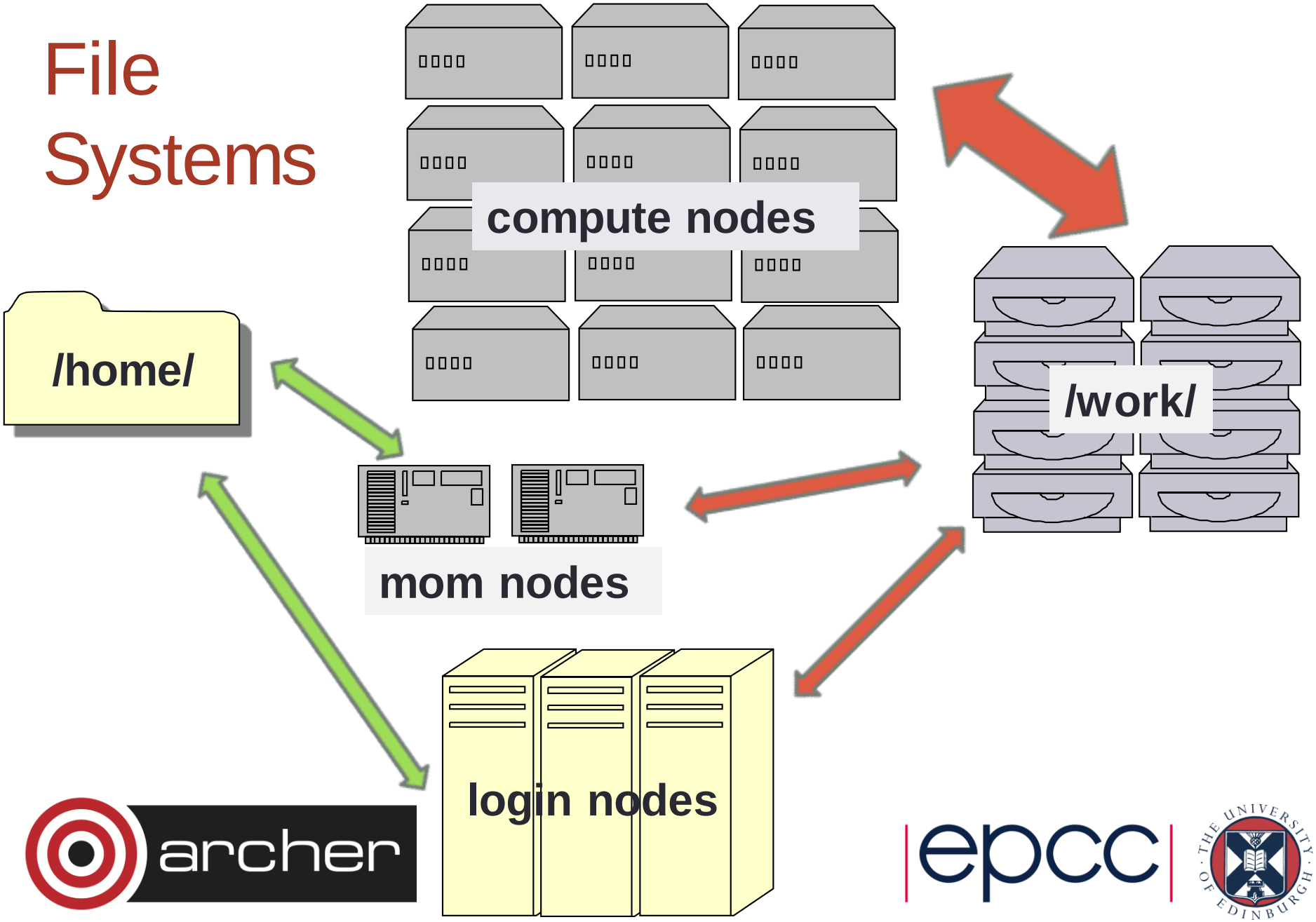

compute nodes
aprun
ready
mom nodes
run
PBS Professional®
ssh
qsub
login nodes
archer
epcc
THE UNIVERSITY OF EDINBURGH

Adulthood: parallel jobs

- Compute nodes reserved for duration of job
  - PBS doesn't care if/how you use them!
  - all commands executed on MOM node
  - aprun on MOM node causes parallel jobs to run on compute nodes

- **`aprun`** does the following
  - **broadcasts the executable** to all the compute nodes
  - **gathers** the standard outputs from all the PEs

File Systems



archer

epcc

THE UNIVERSITY OF EDINBURGH

Retirement: the end of your job

- Job finishes
  - after the all the commands in script have been executed …
  - … or the wallclock limit is exceeded
- All running parallel jobs are killed
  - e.g. wallclock exceeded or aprun running in background (see later)
- Standard outputs collated
  - written to **`myjob.o123456`**
- **qstat** job status set to “E” for some time (annoyingly)
  - then disappears

aprun

- Can issue multiple aprun's in a single job
  - single job + many aprun's may be better than many jobs

```
#PBS -l select=6          # 6*24 = 144 cores
...
aprun -n 144 mympiprogram dataset1
aprun -n 144 mympiprogram dataset2
aprun -n 144 mympiprogram dataset3
...
```

aprun is quite clever

- Can manage multiple parallel jobs simultaneously
  - e.g. imagine we had 4 runs each using 72 cores

```
#PBS -l select=6          # 6*24 = 144 cores
...
aprun -n 72 mympiprogram dataset1
aprun -n 72 mympiprogram dataset2
aprun -n 72 mympiprogram dataset3
aprun -n 72 mympiprogram dataset4
# Incorrect! - all these run sequentially
```

Multiple aprun's in the background (i)

```
aprun -n 72 mympiprogram dataset1 &
aprun -n 72 mympiprogram dataset2 &
aprun -n 72 mympiprogram dataset3 &
aprun -n 72 mympiprogram dataset4 &

# Incorrect: "Job finishes after the all the
# commands in script have been executed".
# Final aprun returns immediately, script
# reaches end and finishes, aprun's killed.
```

archer
epcc
THE UNIVERSITY OF EDINBURGH

Multiple aprun's in the background (ii)

```
aprun -n 72 mympiprogram dataset1 &
aprun -n 72 mympiprogram dataset2 &
aprun -n 72 mympiprogram dataset3 &
aprun -n 72 mympiprogram dataset4

# Incorrect: script finishes when dataset4
# finishes, but other datasets may still be
# running at that time.
```

Multiple aprun's in the background (iii)

```
aprun -n 72 mympiprogram dataset1 &
aprun -n 72 mympiprogram dataset2 &
aprun -n 72 mympiprogram dataset3 &
aprun -n 72 mympiprogram dataset4 &

wait

# Correct! "wait" blocks until all spawned
# processes are complete
```

Task farms with aprun

```
aprun -n 72 mympiprogram dataset1 &
aprun -n 36 mympiprogram dataset2 &
aprun -n 72 mympiprogram dataset3 &
aprun -n 36 mympiprogram dataset4 &
aprun -n 36 mympiprogram dataset5 &

wait

# ordering might be: 1, 2, 4, 3, 5
```

why can my job script see my home files but my MPI program can only see my work files?

Because the script runs on the MOM nodes which see /home but MPI runs on the compute nodes which only see /work

why can I store my MPI executable in /home but not its input files?

aprun broadcasts the executable from /home to the compute nodes, but not any dependent files

how do "interactive" batch jobs work?

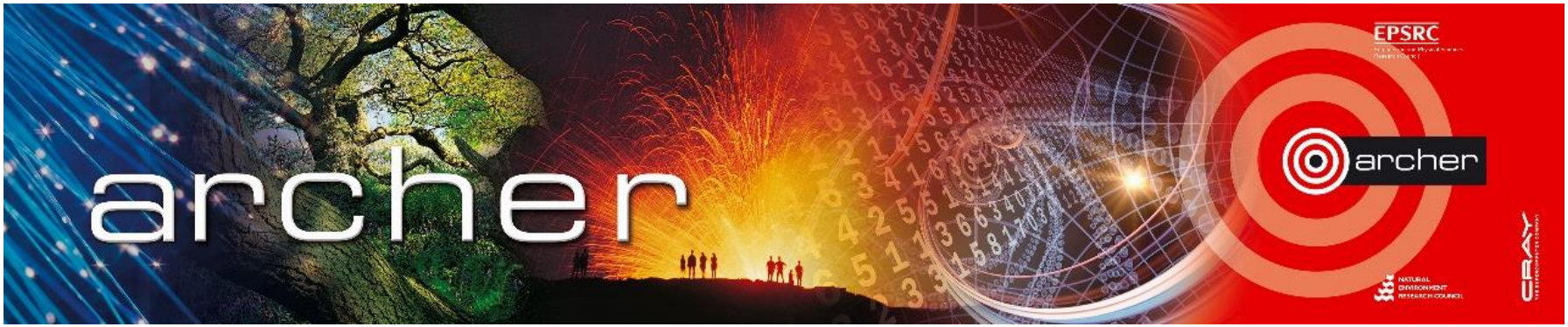
You are effectively submitting a job which runs a bash shell on the MOM nodes

THE END

archer

epcc

THE UNIVERSITY OF EDINBURGH



Goodbye!

Virtual tutorial has finished

archer

epcc

THE UNIVERSITY OF EDINBURGH