University of Sussex EasyBuild setup

Leonardo Rojas Amaya

Research Computing Officer
IT Services
University of Sussex

HIP-SIG Spin-off workshop on EasyBuild
Brimingham – Oct 22nd 2018

Our team

ITS Research team

University's HPC development, implementations and maintenance.

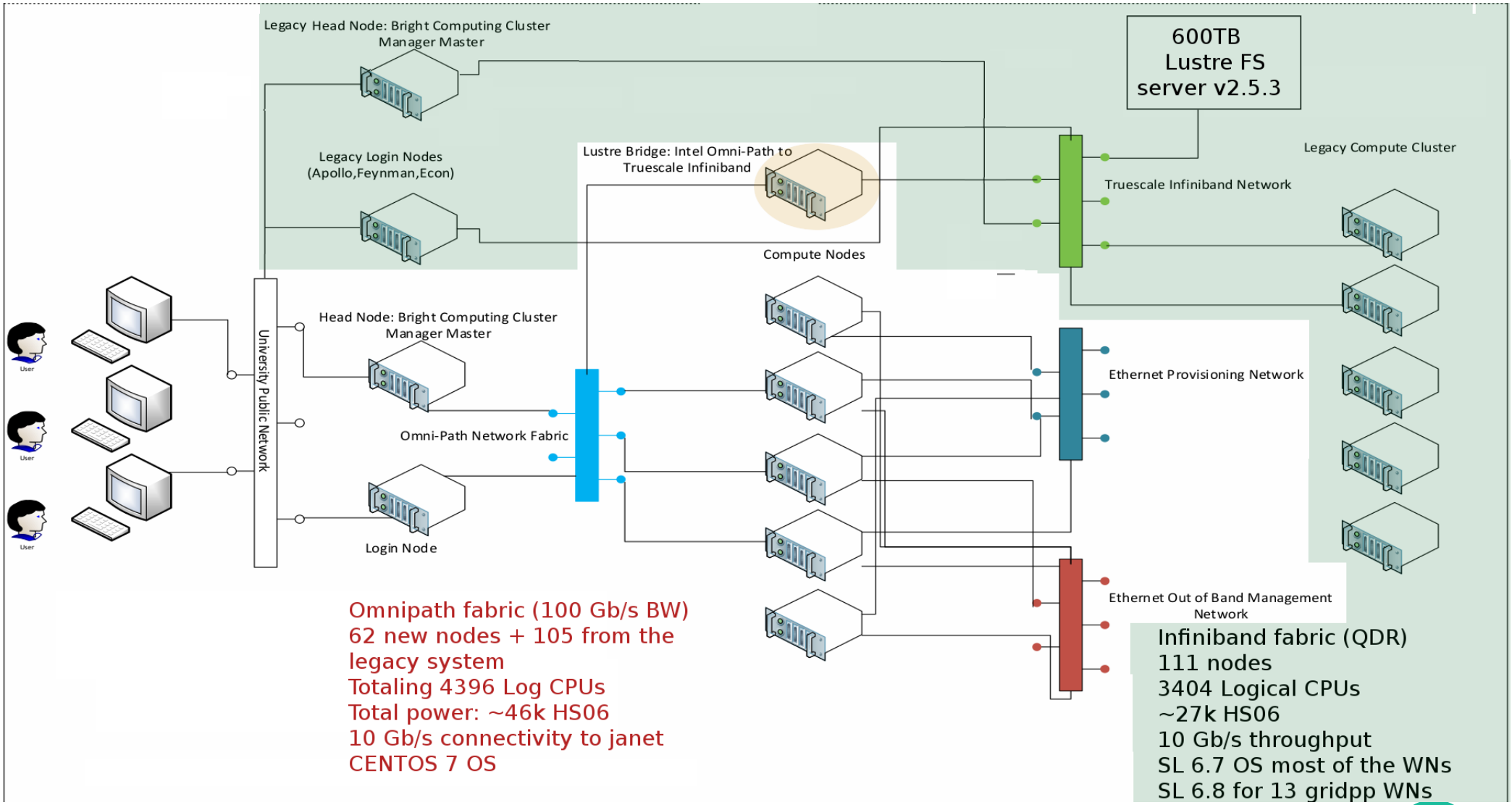
- Jeremy Maris – Research Computing Manager
- Alhamdu Bello – Research Computing Officer
- Bernie Broughton – Research Computing Officer

Formerly:
I have a background in Physics and an MSc in Advance Computational Methods for Aeronatics (Computational Fluid Dynamics stuff).

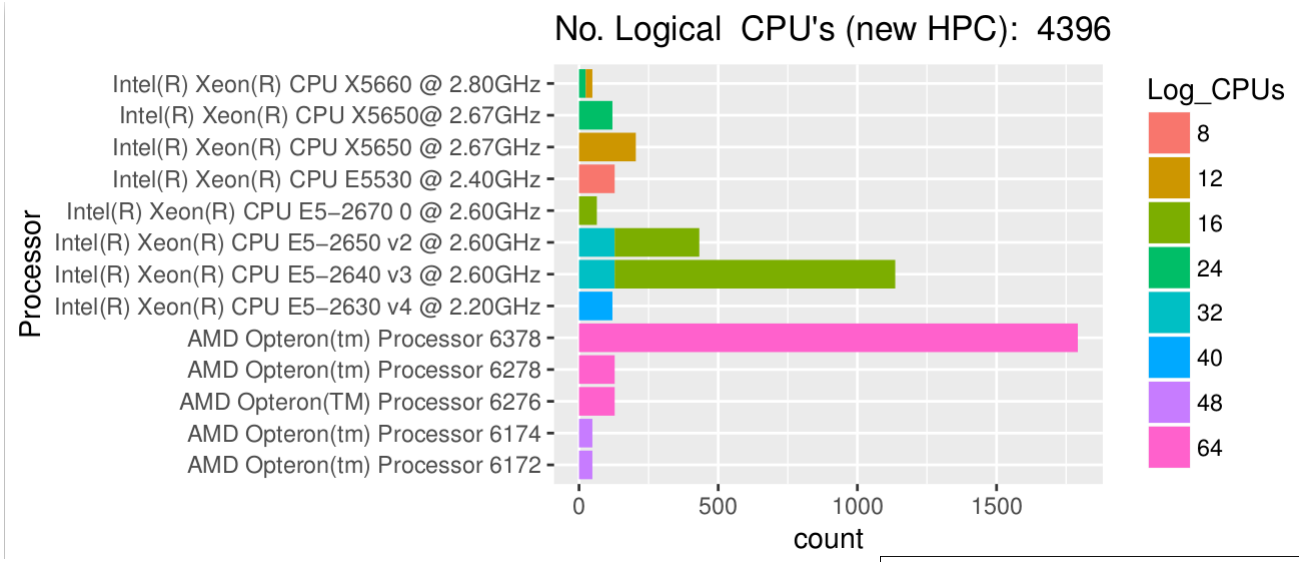
I’m in the process of leaving my role as Linux Technical Analyst at the Mathematical and Physical Sciences School at the University of Sussex, I’ve been there for almost 2 years

- I was the gridpp cluster manager
- Provided Research computing support
- And dealt with the HPC administratin on behalf of the school

Our Current HPC system



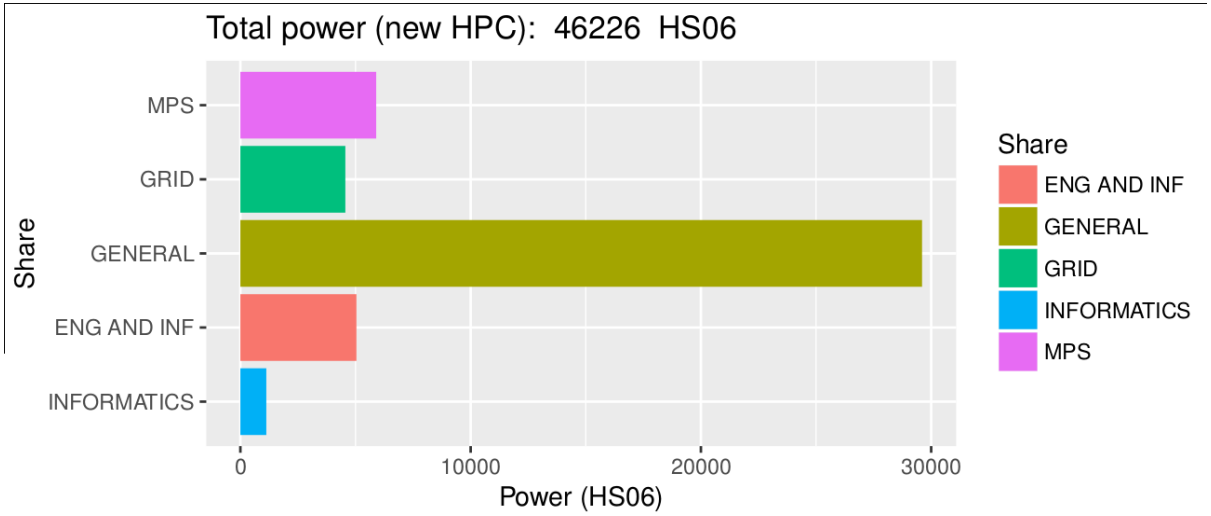
Compute power: our hybrid HPC system



Processor model distribution for our new consolidated cluster.

We have AMDs, Intel supporting AVX, AVX2 and AVX512

Compute power across schools



Legacy software stack main weaknesses

1. On the legacy cluster the **building software workflow was basically manual**, following the standard configure / make / make install approach.

2. **Module creation was heavily manual** too. Once a software was built, the system administrator created the module from scratch and had to ensure it loaded all the necessary dependencies, and correctly set all the env variables for it to work.

3. There is **no track for building recipies** or dependencies, so that reproducing previous builds was quite hard when possible.

4. **Lack of standard toolchains** software was built against arbitrary compilers depending on individual situations.

5. Having a **heterogeneous cluster**, **lack of proper support for different architectures**. Modules presented to the users may fail on certain nodes on illegal instruction errors.

Legacy software stack

```
$ module avail openmpi
--------- /cm/shared/modulefiles ----------------
openmpi/gcc/64/1.6.5                  openmpi/gcc-6.2/2.1.1
openmpi/gcc/64/1.7.3                  openmpi/intel/64/1.6.5
openmpi/gcc/64/2.0.1                  openmpi/intel/64/1.7.3
openmpi/gcc/64/2.1.1                  openmpi/intel/64/current
openmpi/gcc/64/disable-dlopen/2.1.1 openmpi/open64/64/1.6.5

--------- /mnt/pactsw/etc/modulefiles -----------
openmpi/2.0.1/gcc/4.9.4
openmpi/2.0.1/gcc/5.4.0

$ module  avail python/
-------- /cm/shared/modulefiles  ----------------
python/2.7.10             python/2.7.3
python/3.5.1              python/current
python/test-2.7.5         python/2.7.11
python/2.7.5              python/3.5.1-libshared
python/intel/2.7.12       python/2.7.12
python/3.4.3              python/anaconda/2-5.1.0
python/intel/3.6.039
```

New cluster eb set-up

```
$ [hpcbuild@apollo-master2:~ ] $ eb --show-config

buildpath                (F) = /cm/shared/easybuild/build
containerpath            (D) = /cm/shared/easybuild/containers
git-working-dirs-path    (F) = /cm/shared/easybuild/github-working-dir
installpath              (F) = /cm/shared/easybuild/default_install/
packagepath              (F) = /cm/shared/easybuild/packages
repositorypath           (F) = /cm/shared/easybuild/ebfiles_repo.git
sourcepath               (F) = /cm/shared/easybuild/source
robot-paths              (D) = /cm/shared/apps/easybuild/3.6.2/software
                               /EasyBuild/3.6.2/lib/python2.7/site-
                                packages/easybuild_easyconfigs-3.6.2-
                                py2.7.egg/easybuild/easyconfigs
repository               (F) = GitRepository
github-user              (F) = sussex-hpcbuild
module-syntax            (F) = Tcl
modules-tool             (F) = EnvironmentModulesC
```

Compiler tool-chains

```
$ eb -S intel-2017a | wc -l
274

$ eb intel-2017a.eb -xr | grep -i missing
* parallel_studio_xe_2017_update1_composer_edition_for_cpp.tgz (MISSING)
* parallel_studio_xe_2017_update1_composer_edition_for_fortran.tgz (MISSING)
* l_mpi_2017.1.132.tgz (MISSING)
* l_mkl_2017.1.132.tgz (MISSING)

$ module avail
   /mnt/ebinstall/modules/all:
   - Package -+- Versions -
     Anaconda3/4.0.0
     Boost/1.63.0-intel-2017a-Python-2.7.13
     HDF5/1.10.0-patch1-intel-2017a
     HDF5/1.8.18-intel-2017a
     netCDF/4.4.1.1-intel-2017a
     OpenFOAM/4.1-intel-2017a
     ParaView/5.2.0-intel-2017a-mpi
     Python/2.7.13-intel-2017a
     Python/3.6.1-intel-2017a
     Qt/4.8.7-intel-2017a
     R/3.4.0-intel-2017a-X11-20170314
     Singularity/2.4-GCC-6.3.0-2.27
     SQLite/3.17.0-GCCcore-6.3.0
     Tensorflow/1.1.0-intel-2017a-Python-2.7
     Tensorflow/1.2.0-intel-2017a-Python-3.6
     Tk/8.6.6-intel-2017a
```

Compiler tool-chains for users

A user wanted to run **C2ray-3D** (a conservative, casual ray tracing method) to calculate the transfer of ionizing radiation in astrophysical processes

```
$ module load easybuild/software
$ module load intel/2017a
$ export I_MPI_F90=ifort
$ make C2Ray_3D_cubep3m_kyl_periodic_omp_mpi
mpif90  -c -fpp -O3 -u -fpe0 -ipo -DIFORT -shared-intel   -qopenmp
-DMY_OPENMP  -lpthread -DMPI -DMPILOG  -DQUASARS  ../precision.f90
…
```

Her Jobscript:

```
#$ -pe impi 36
#$ -l exclusive=true
#$ -l h_vmem=30G
#$ -binding pe striding:2:8
module add easybuild/software
module add intel/2017a

export OMP_NUM_THREADS=8
mpirun -n $NSLOTS ./C2Ray_3D_cubep3m_kyl_periodic_omp_mpi input
```

Software stack replicas

The one-size-fits-all approach will waste all the benefits from newest processors' optimizations. One option is to create software stack replicas:

```
[apollo-master2->device]% pexec -n node065,node063 "mount | grep ebinstall"
[node065]:
nfs004.hpc.susx.ac.uk:/srv/ebinstall/core-avx2 on /mnt/ebinstall type nfs ...

[node063]:
nfs004.hpc.susx.ac.uk:/srv/ebinstall/avx on /mnt/ebinstall type nfs  …
```

To install then you just repeat your build eb command on each architecture to generate working replicas of the software

```
[hpcbuild@node063:~]$  eb Python-2.7.14-intel-2018a.eb --installpath=/mnt/ebinstall/
[hpcbuild@node065:~]$  eb Python-2.7.14-intel-2018a.eb --installpath=/mnt/ebinstall/
```

So wherever users' jobs are dispatched
they are going to run compatible binaries!

```
[hpcbuild@node063:~ ] $ module display  easybuild/software
   -------------------------------------------------------------------
   /cm/shared/modulefiles/easybuild/software:
   module-whatis  Adds software built with EassyBuild to your Module path
   module         use /mnt/ebinstall/modules/all
   -------------------------------------------------------------------
```

Singularity containers with eb

A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings.

```
# eb  --allow-use-as-root-and-accept-consequences  Singularity-2.5.2-GCC-6.3.0-2.27.eb --installpath=/mnt/ebinstall/ --robot=./
```

A user is then able to create their own container on his workstation

```
[root@user-workstation] #  time singularity build  cosmosis-openmpi.simg SingularityFile
```

Copy it across to the cluster and run it using eb modules

Tip: In terms of performance it is better to build the container within the same hardware it is to be run, it also prevents processor architecture incompatibilities in case users have newer processors on their workstations!

The jobscript again is very neat:

```
#$ -pe openmpi 64
#
module add easybuild/software
module add foss/2017a Singularity/2.5.2-GCC-6.3.0-2.27

mpirun -np $NSLOTS singularity exec $HOME/singularity/cosmosis-openmpi.simg cosmosis --mpi /path/to/imput/r01.ini
```

Tip:it is not advisable to install Singularity as a module, since having old versions of it expose your system to security flaws. If you do, keep it up to date and remove older versions from your system!

Still struggling!

> **Users are requesting software packages that are unavailable on eb**:
  * Latest version of the Fenics solver,
  * Deal.ii,
  * Singularity 2.5.2,
  * Tensorflow with GPU support

  We need to start editing and creating our own eb files and be able to edit existing eb blocks or create our own

Tip: Some of these are now available on eb v3.7

Our approach at the moment is:

1. look for the package on **eb**
2. if there is a package on a different version **try modifying the eb file (Singularity)**
3. if there is no package and it is a python env, **try using the eb Anaconda Module (Fenics)**
4. otherwise look for **a docker/singularity container (GPU capable Tensorflow, Fenics)**
5. if all of these fail, still **build the software manually!**

Tip: With Singularity and Anaconda you have to think carefully about the performance implications. Depending on how you use them you could be running generic pre-compiled binaries that do not take full advantage of your hardware.

> We have encountered software packages that we have not been able to compile **(R-3.5.0-iomkl-2018a)**

Tip: With the most recent versions of R in particular, you can get compilation errors on some libraries. I that happened you could still install the module (`eb R.eb --module-only`) then you can attempt to fix the problem and proceed with the installation with (`eb R.eb --skip --force`)

.

Thanks!

Any questions?