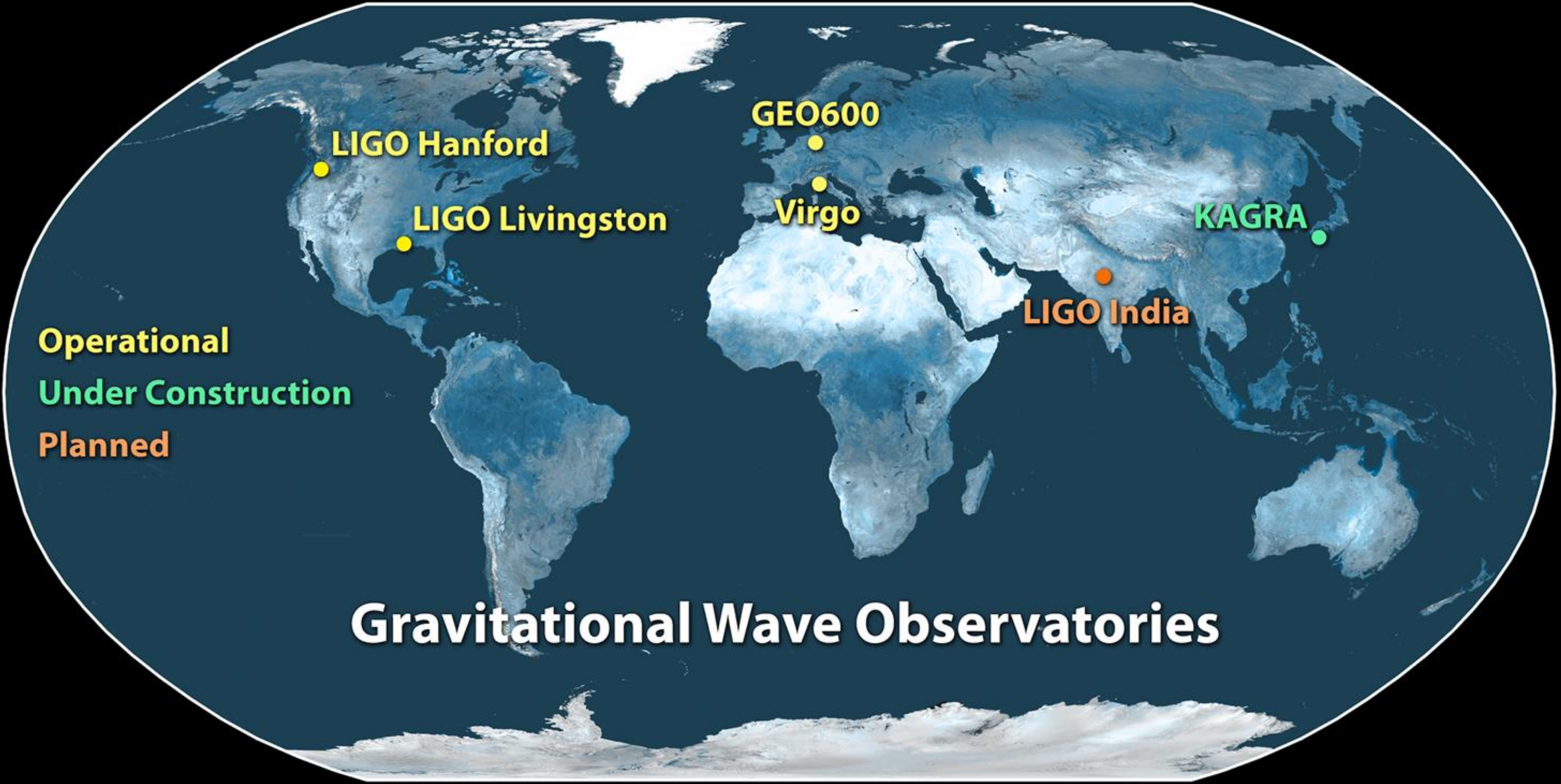
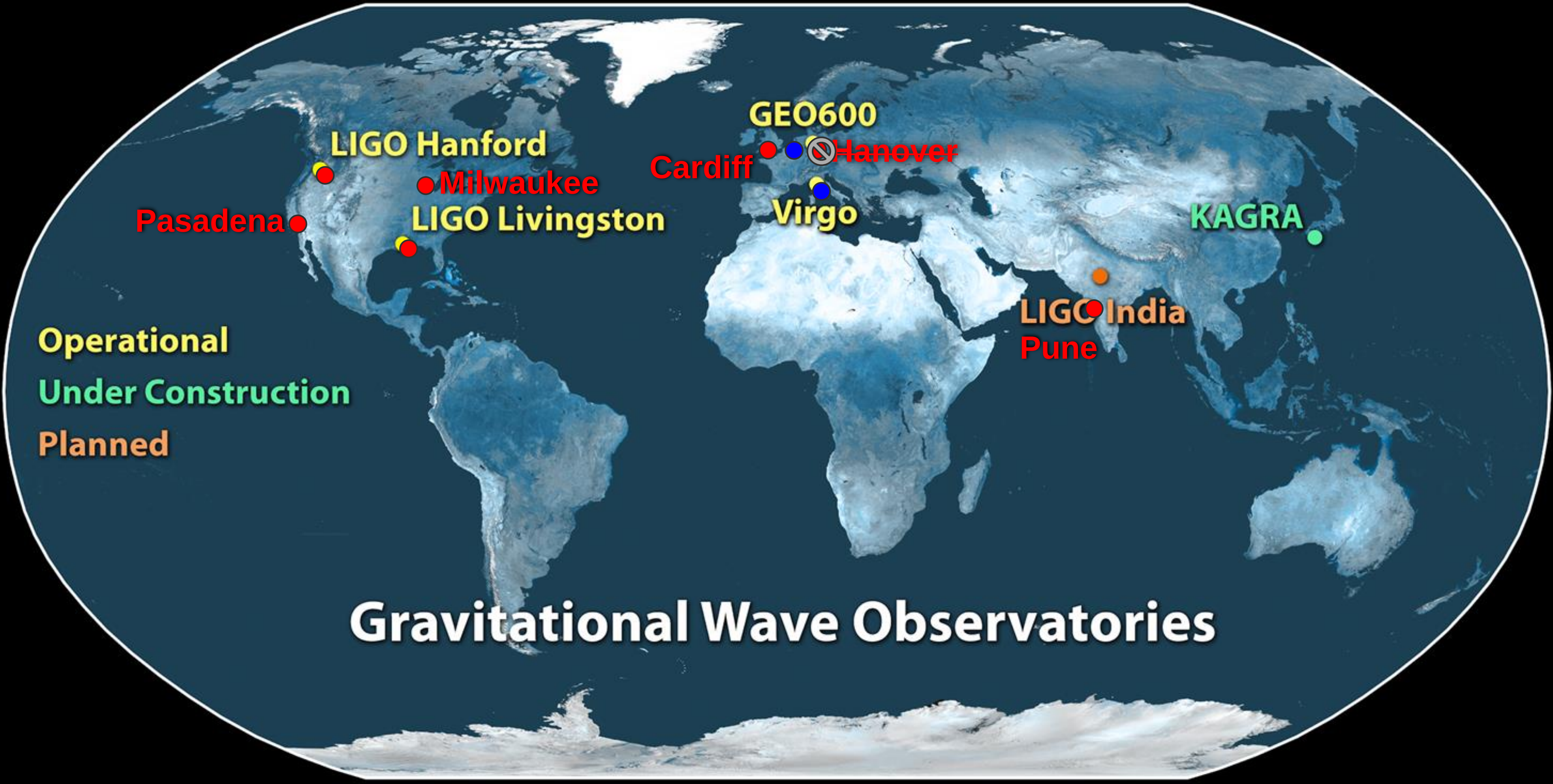
Building a LIGO HTCondor site on top of a shared HPC cluster

Paul Hopkins
Cardiff University and LIGO Scientific Collaboration

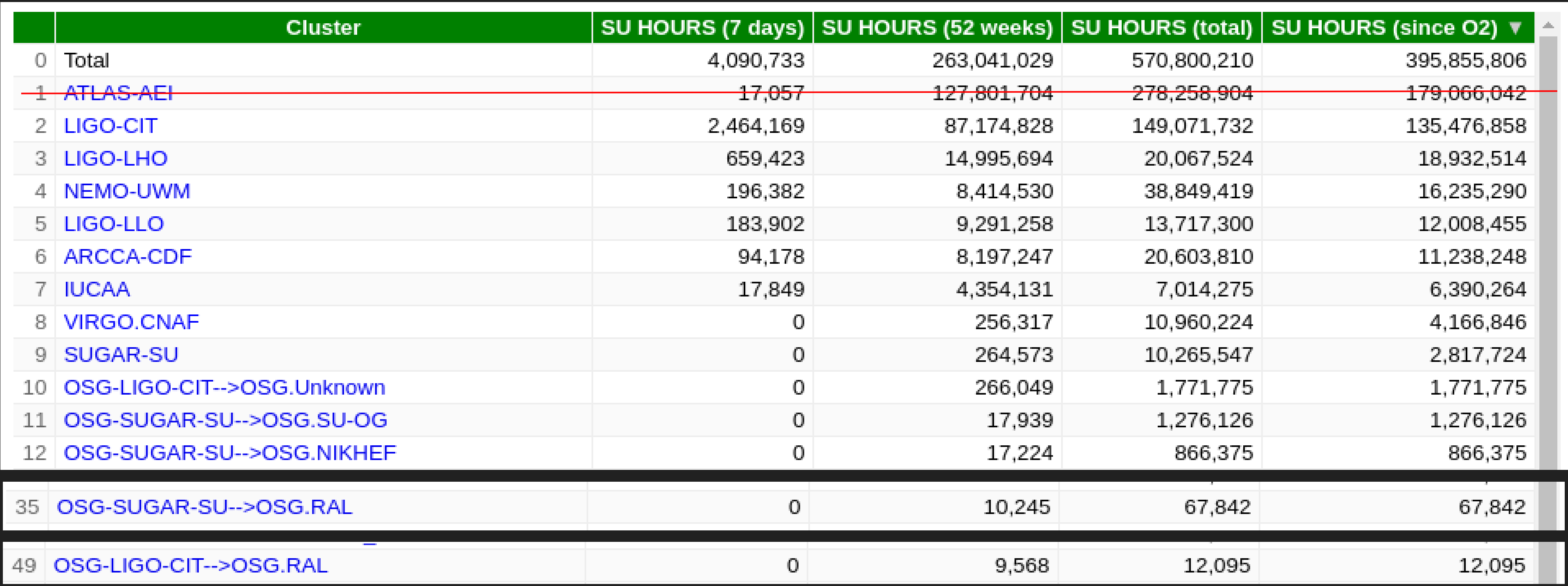
Gravitational Wave Observatories





Gravitational Wave Observatories

LIGO Accounting



	Cluster	SU HOURS (7 days)	SU HOURS (52 weeks)	SU HOURS (total)	SU HOURS (since O2) ▼
0	Total	4,090,733	263,041,029	570,800,210	395,855,806
~~1~~	~~ATLAS-AEI~~	~~17,057~~	~~127,801,704~~	~~278,258,904~~	~~179,066,042~~
2	LIGO-CIT	2,464,169	87,174,828	149,071,732	135,476,858
3	LIGO-LHO	659,423	14,995,694	20,067,524	18,932,514
4	NEMO-UWM	196,382	8,414,530	38,849,419	16,235,290
5	LIGO-LLO	183,902	9,291,258	13,717,300	12,008,455
6	ARCCA-CDF	94,178	8,197,247	20,603,810	11,238,248
7	IUCAA	17,849	4,354,131	7,014,275	6,390,264
8	VIRGO.CNAF	0	256,317	10,960,224	4,166,846
9	SUGAR-SU	0	264,573	10,265,547	2,817,724
10	OSG-LIGO-CIT-->OSG.Unknown	0	266,049	1,771,775	1,771,775
11	OSG-SUGAR-SU-->OSG.SU-OG	0	17,939	1,276,126	1,276,126
12	OSG-SUGAR-SU-->OSG.NIKHEF	0	17,224	866,375	866,375
35	OSG-SUGAR-SU-->OSG.RAL	0	10,245	67,842	67,842
49	OSG-LIGO-CIT-->OSG.RAL	0	9,568	12,095	12,095

What does a LIGO Data Grid site need?

- Access cluster using LIGO credentials
- ★ Software
- Data
- ★ HTCondor Scheduler
- Web Server to view results with LIGO Shibboleth authentication
- JupyterLab Service with “LIGO” kernels
- Shared File System!

Running HTCondor at Cardiff

- Standard RHEL 6.4 head node and compute nodes
- PBSPro scheduler
- Filesystems: `/home, /software (NFS)`; `/scratch` (Lustre) and `/tmp` (local)

- Virtual and Physical head nodes under my control
- HTCondor Central Manager and Submit run on dedicated head nodes
- Only minor modifications allowed on the compute nodes
    - How do deliver them to HTCondor?

HTCondor “Glideins”

- HTCondor borrows whole nodes from PBSPro using custom glidein script:

    -
      ```
      if $idle_jobs_in_condor ; then
                submit_glidein_jobs_to_pbs_queue()
      fi
      ```
- PBSPro then runs jobs on a dedicated queue using an operational user:

    -
      ```
      #!/bin/bash
      #PBS -q ligo-condor
      source /software/tools/condor/8.6.11/condor.sh
      sudo -E $(which condor_master) -d -f
      ```
- `sudo` necessary to switch user to allow access to shared file system
- HTCondor is given whole node, 12 or 24 cores, depending on hardware.
- PBS maximum wall time is 5 or 10 days.

Glidein Config

```
DAEMON_LIST = MASTER STARTD

LOCAL_DIR = /tmp
LOG = $(LOCAL_DIR)/Condor-log
SPOOL = $(LOCAL_DIR)/Condor-spool
EXECUTE = $(LOCAL_DIR)/Condor-execute

UID_DOMAIN = arcca.cf.ac.uk
FILESYSTEM_DOMAIN = arcca.cf.ac.uk
TRUST_UID_DOMAIN = True

STARTD_NOCLAIM_SHUTDOWN = 300

slot_type_1_partitionable = true
slot_type_1 = cpus=$(DETECTED_CORES), mem=$(DETECTED_MEMORY)
num_slots_type_1 = 1
```

Pros / Cons of whole node glidein

- PROS:
- Can use HTCondor to manage whole node:
  - condor_drain, reservation for development and testing
- Due to PBS walltime has a natural de-fragmentation and reassignment to faster nodes.

- CONS:
- Harder for HPC admins to monitor usage and job efficiency

Glidein Alternatives?

- Pyglidein
- ~~Job Transforms to grid universe~~
- ~~Condor Annex~~

LSC Software on Dedicated Sites

- All other LSC sites are dedicated to LIGO so software is installed as system packages, e.g. `/usr/bin/python`
- Two reference OSes; Scientific Linux 7 and Debian 8/9
- Sites carefully updated to ensure library versions are similar
- Software can be easily installed at another dedicated site:
  - `yum install lscsoft-all`
- Some disadvantages:
  - Only one version of package can exist
  - Users are forced to upgrade
  - 3rd party packages within repository often quite old

LSC Software on ARCCA

- Software must be installed from source packages:

  -
    ```
    ./configure
    make
    make install
    ```
  - Software then loaded via environment modules
- From 2012-2016 software was managed by hand
- From 2016-2018 I used Spack (https://spack.io/)
  - Many libraries already packaged (2771), adding packages very easy
  - Allows users to install different versions and “variants” simultaneously
  - Creates modules to load environment

  -
    ```
    spack install lscsoft-all
    ```

Spack - Disadvantages

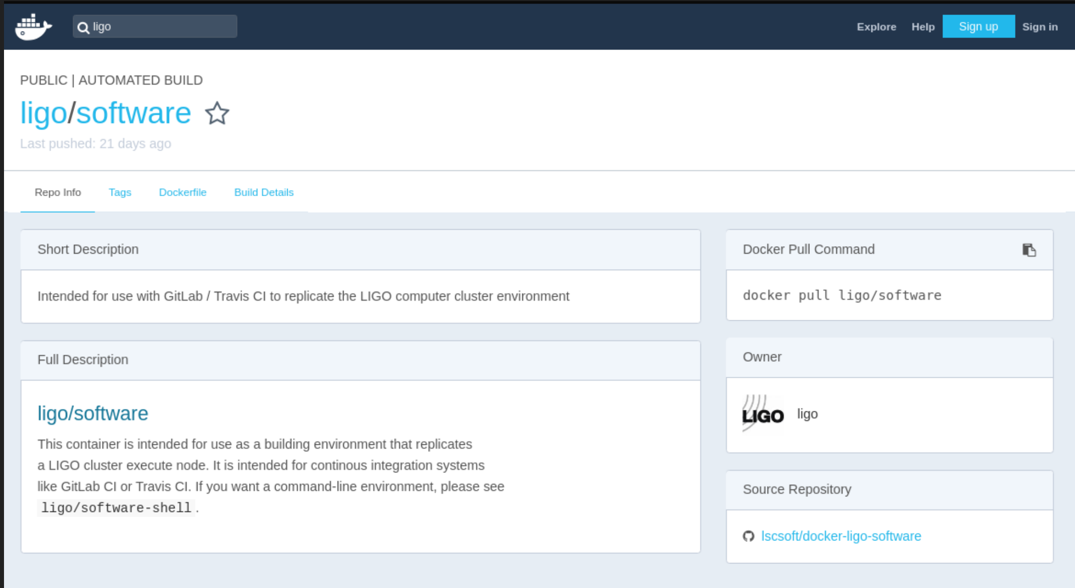
- By default installs ALL dependencies
- For a large number of packages this can be too cumbersome
- Extra work needed to combine into a single installation directory
- Ultimately could not recommend to users
- If I restarted today then I would begin with Anaconda….

Singularity + CVMFS

- As well as containing processes, Singularity presents a complete software environment
- Can be efficiently distributed using CVMFS
- How should I allow users to make use of it?

LIGO + Docker Hub

- LIGO already provide various OS and application images on Docker Hub
- These are unpacked onto CVMFS



PUBLIC | AUTOMATED BUILD

ligo/software

Last pushed: 21 days ago

Short Description

Intended for use with GitLab / Travis CI to replicate the LIGO computer cluster environment

Docker Pull Command

```
docker pull ligo/software
```

Full Description

ligo/software

This container is intended for use as a building environment that replicates a LIGO cluster execute node. It is intended for continous integration systems like GitLab CI or Travis CI. If you want a command-line environment, please see `ligo/software-shell`.

Owner

ligo

Source Repository

lscsoft/docker-ligo-software

```
singularity shell  /cvmfs/ligo-containers.opensciencegrid.org/dockerhub/ligo/software:el7

singularity exec /cvmfs/ligo-containers.opensciencegrid.org/dockerhub/ligo/software:el7 python
```

Automatic Singularity Environment

- Intend to give normal users an automatic Singularity LIGO environment
- But, allow power users to opt out, or to use own custom environment.
- Singularity image path stored in $HOME/.singularity_image:

    -
      ```
      SINGULARITY_IMAGE=$( cat $HOME/.singularity_image )
      ```
- When users login automatically launch shell:

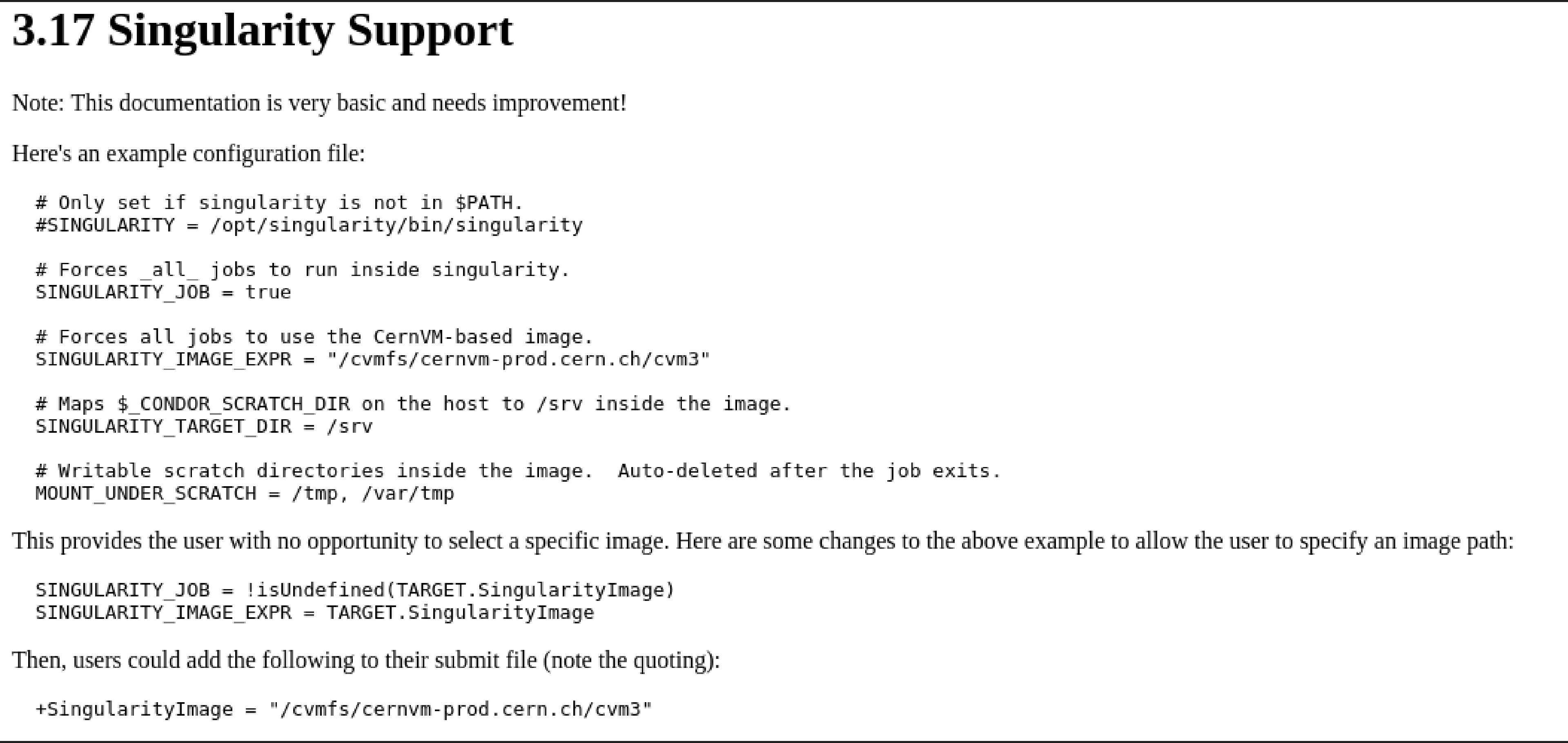
    -
      ```
      singularity shell -s /bin/bash $SINGULARITY_IMAGE
      ```

Example Shell Login

```
paul@HopkinsThinkpad:~$ gsissh ligo.arcca.cf.ac.uk
....
Singularity: Invoking an interactive shell within container…

ligo-headnode/latest spxph@raven14:~$
     python -c "import lalapps; print lalapps.__file__"

/usr/lib/python2.7/dist-packages/lalapps/__init__.pyc
```



3.17 Singularity Support

Note: This documentation is very basic and needs improvement!

Here's an example configuration file:

```
# Only set if singularity is not in $PATH.
#SINGULARITY = /opt/singularity/bin/singularity

# Forces _all_ jobs to run inside singularity.
SINGULARITY_JOB = true

# Forces all jobs to use the CernVM-based image.
SINGULARITY_IMAGE_EXPR = "/cvmfs/cernvm-prod.cern.ch/cvm3"

# Maps $_CONDOR_SCRATCH_DIR on the host to /srv inside the image.
SINGULARITY_TARGET_DIR = /srv

# Writable scratch directories inside the image.  Auto-deleted after the job exits.
MOUNT_UNDER_SCRATCH = /tmp, /var/tmp
```

This provides the user with no opportunity to select a specific image. Here are some changes to the above example to allow the user to specify an image path:

```
SINGULARITY_JOB = !isUndefined(TARGET.SingularityImage)
SINGULARITY_IMAGE_EXPR = TARGET.SingularityImage
```

Then, users could add the following to their submit file (note the quoting):

```
+SingularityImage = "/cvmfs/cernvm-prod.cern.ch/cvm3"
```

Possible Issues and Workarounds

- Care must be taken to ensure HTCondor configuration is the same inside the environment as outside
- Can using `_CONDOR_*` environment variables or bind mount configuration files
- Note that Singularity is executed with `--containall` option:
  - Use minimal /dev and empty other directories (e.g. /tmp and $HOME) instead of sharing filesystems on your host.
  - Contain not only file systems, but also PID, IPC, and environment
- Alternative solution is to use HTCondor USER_JOB_WRAPPER:

```
#!/bin/sh
if [ -n "${_CONDOR_JOB_AD}" ]; then
    owner=$( grep -E '^Owner' $_CONDOR_JOB_AD | cut -d ' ' -f 3 | tr -d '"' )
    SINGULARITY_IMAGE_FILE=$( getent passwd $owner | \
              cut -d: -f6 )/.singularity_image
fi
```

The Pegasus Problem - Scheduler Universe

- Pegasus is used by some LIGO workflows; it is a high-level workflow manager that runs in the scheduler universe.
- BUT, scheduler universe does not support `SINGULARITY_JOB` or USER_JOB_WRAPPER!?
- Tried, local universe, but Pegasus fails. Need to add environment variable:

  ```
  JOB_TRANSFORM_NAMES = SchedulerUniverse
  JOB_TRANSFORM_SchedulerUniverse @=end
  [
    Requirements = JobUniverse == 7;
    SET_JobUniverse = 12;
    EVAL_SET_Env = strcat("CONDOR_ID=", ClusterId, " ", Env);
  ] @end
  ```

Paul’s Lament

- Doing lots of work to give users what they have elsewhere
- Shared file systems and dedicated sites prevent LIGO users from making use of Grid Resources, e.g. OSG and IRIS
- Trying to wean users off it via CVMFS, Singularity, Large scale CVMFS for data, including standard file locations.
- Would like to get Cardiff to accept GRID jobs, but need to convince HPC admins….

Suggestions welcome