IceCube

OSG AHM March 2018

D. Schultz, V. Brik, H. Skarlupka, P. Meade, G. Merino

Outline

- CVMFS on Kubernetes
- Pyglidein - running jobs on the Grid
- IceProd - IceCube simulation production and data processing framework
- Supercomputers - XSEDE, TITAN
- Data Management - long term archive

CVMFS on Kubernetes

/cvmfs/icecube.opensciencegrid.org/

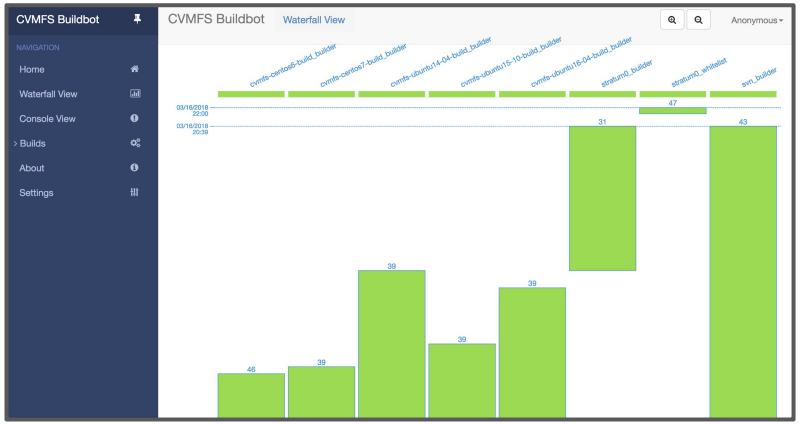
CVMFS - Stratum-0 containerized deployment

Infrastructure as code

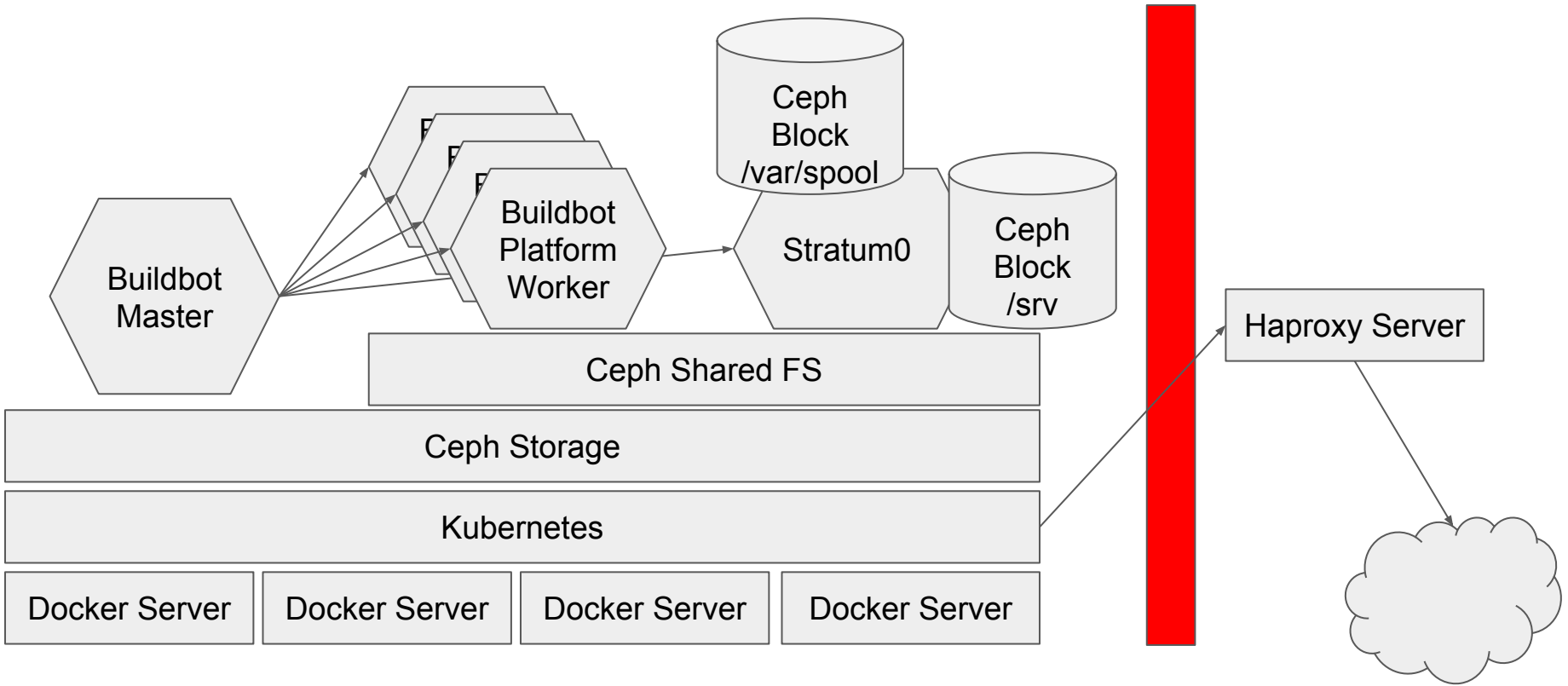
- Easy to deploy / change infrastructure
- Adding new platforms becomes simple

Automation

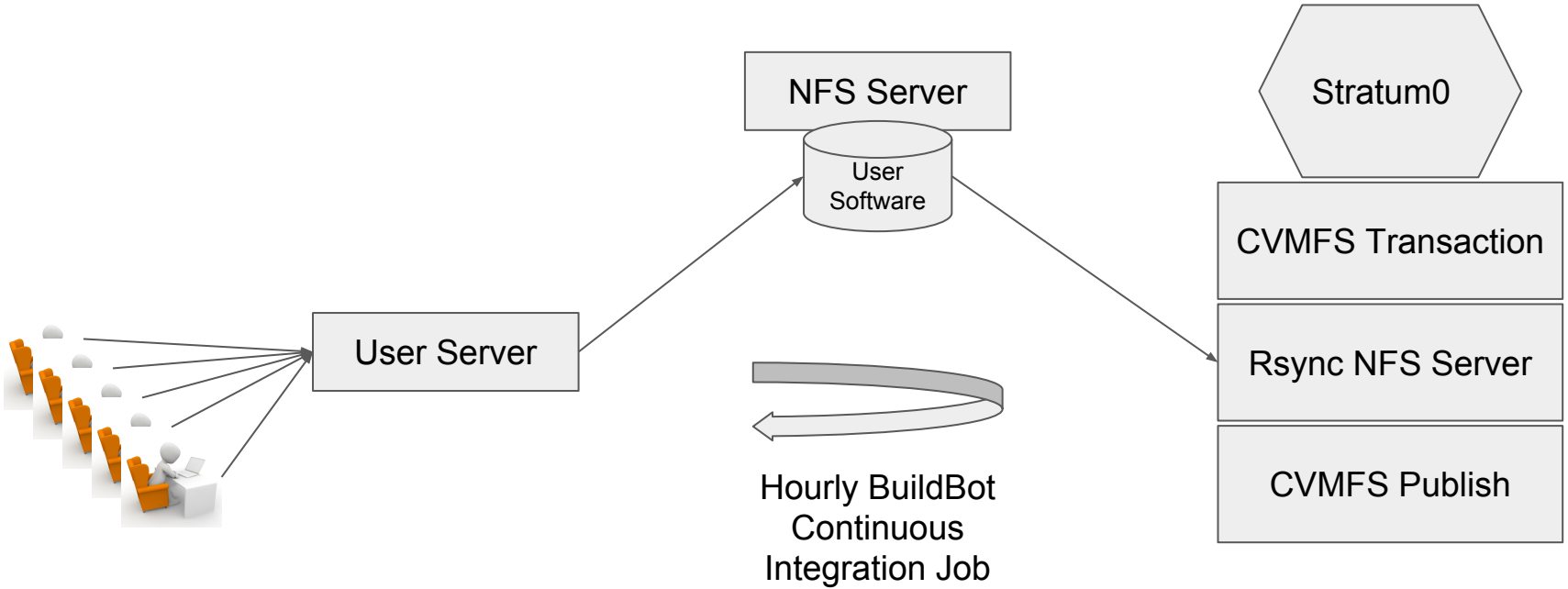
- Push-button build / publish
- Git hooks to deploy new software releases
- Nightly build of stable trunk



CVMFS on Kubernetes



CVMFS user space (in preparation)



Running Jobs

Pyglidein

Pyglidein

A python server-client pair for submitting HTCondor glidein jobs on remote batch systems.

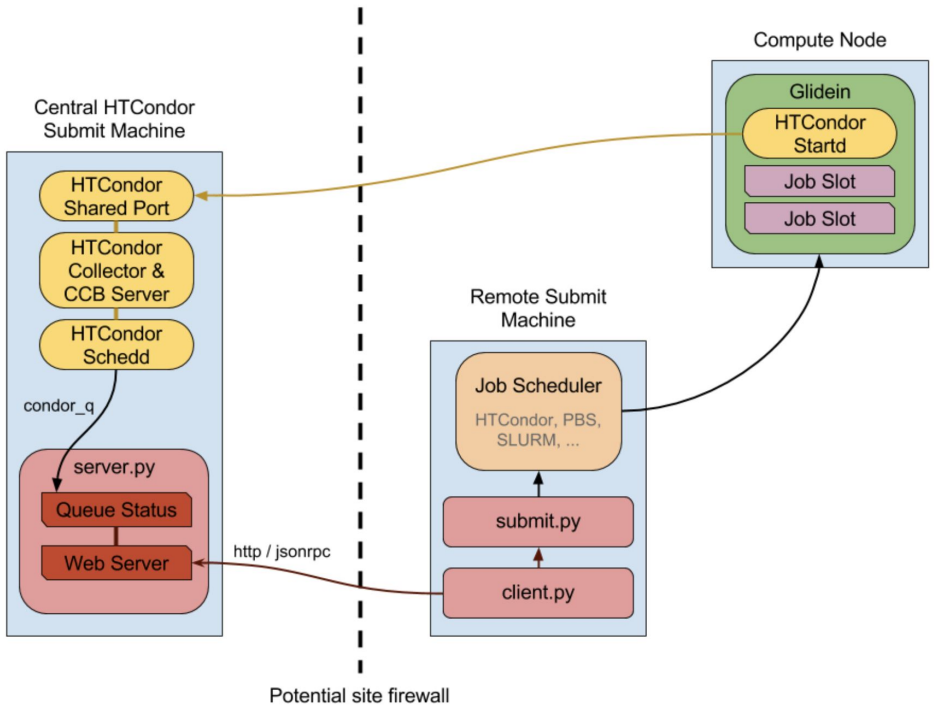
https://github.com/WIPACrepo/pyglidein

Motivation / requirements:

- easy for remote sites

```
$ pip install pyglidein
```

- address 2-factor auth



Pyglidein - S3 Logging

Enabled with client config flag - turn on for debugging

Presigned GET and PUT S3 url for each glidein - access keys only stored in the submit node

- HTCondor logs uploaded every 5 minutes
- GET url stored in startd classad, injected into each job

Job history stored in Elasticsearch

- Can query for failed jobs, get S3 log url
- S3 logs persist up to 90 days

Pyglidein - Blackhole prevention

Add HTCondor StartD Cron scripts to Pyglidein

- **PYGLIDEIN_RESOURCE_<NAME>** = True/False classad
    - Checks for CVMFS, GridFTP access, GPU functionality
- **PYGLIDEIN_METRIC_TIME_PER_PHOTON** classad
    - IceCube GPU speed benchmarking (photon propagation)

Still in alpha testing

- Users currently add Requirements based on these classads
- GPU benchmarking - we want to do normalized accounting

Pyglidein - What is next

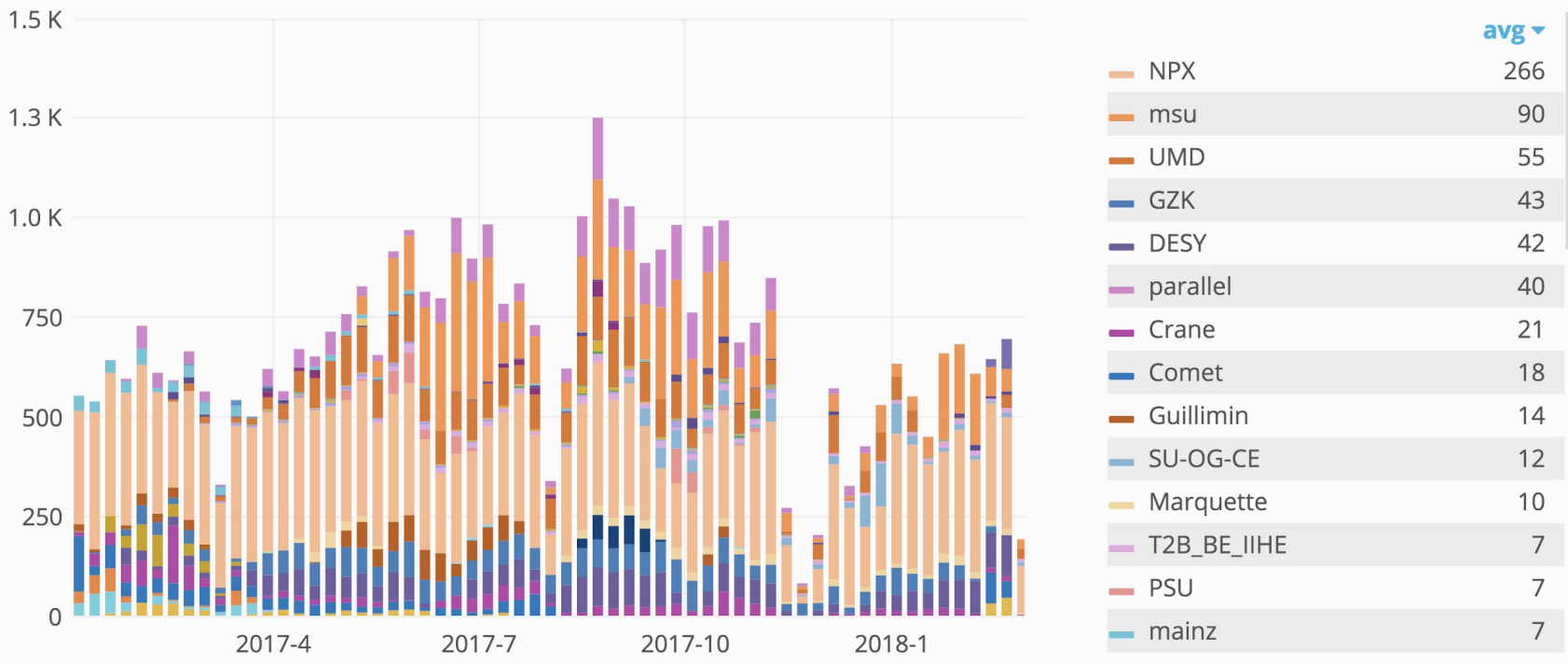
Use a newer version of HTCondor and Parrot to be able to use Singularity

- same functionality as already present in GlideinWMS, so users can use Singularity without caring about the glidein “flavor”

Want to test new GPUsUsage feature in HTCondor 8.7.7

- need a patch in order to use it inside a glidein (has been already submitted to HTCondor)

GPU usage per week



Data Processing Framework

IceProd

IceProd - Dataset bookkeeping

What is it:

- Dataset submission framework
- Keeps track of metadata, software config, versioning
- Monitors job status, resource usage
  - could be done as part of glidein infrastructure (more later ...)
- Retries failed jobs
  - Can resubmit with different requirements

IceProd - Dataset bookkeeping

What do we use it for:

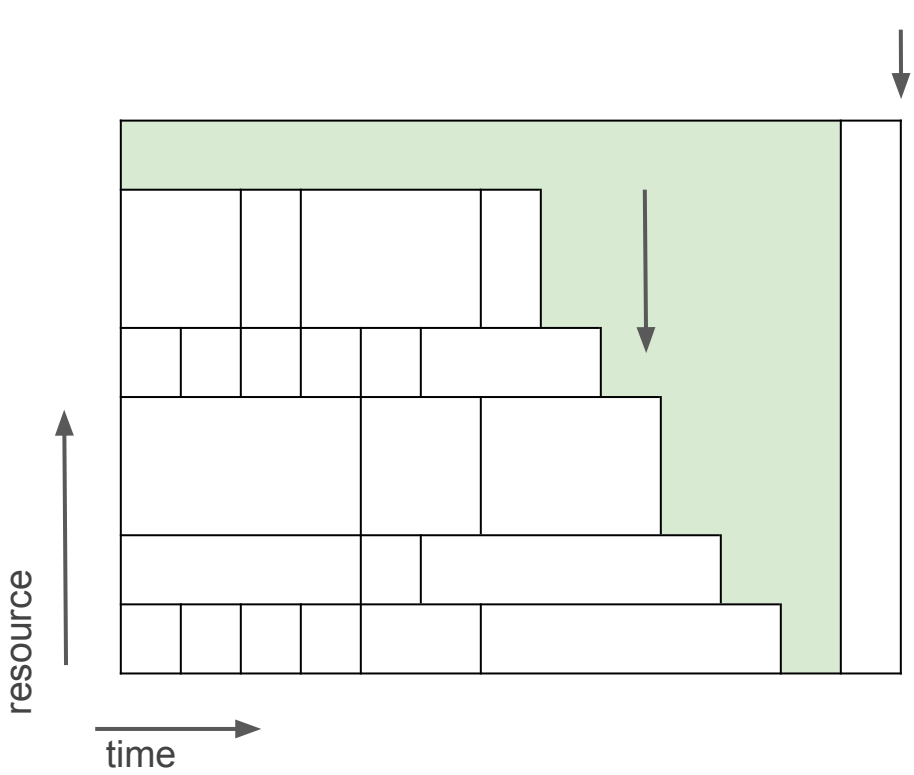
- Simulation production
- Experimental data processing (pre-analysis)
- Increasingly higher levels of common processing

IceProd - Internal Pilot

We run a **pilot** inside the HTCondor job - Things it does / plans to do:

- Aggregate communications with the IceProd server
  - IceProd pilots are whole-node jobs: one communication link per node
- Resource monitoring in real-time
  - cpu, gpu, memory, disk usage, time
- Asynchronous file transfer
  - stage in/out files for next/prev jobs while jobs execute
- Dynamically resizable “jobs”

Dynamically resizable slots



give more resources to a long-running job to try and complete it before the end of the glidein

Supercomputers

XSEDE 2017 allocation

June 2017 - start of our 2nd research allocation. Target GPUs

Represents ~15% of our GPU capacity at UW-Madison

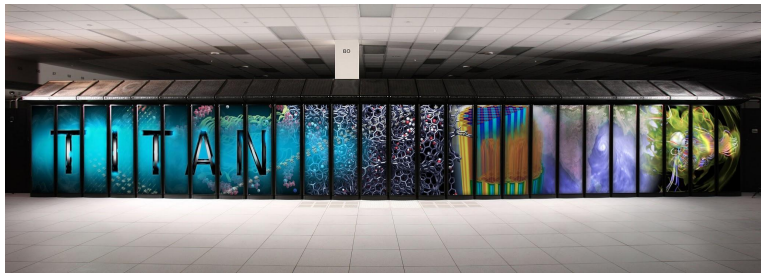
A good opportunity to exercise integration of supercomputers and extended period operations (run Pyglidein clients on submit nodes)

System	***Service Units awarded***	***GPU nodes***	***Used so far***
XStream	*475,000*	*65x8 K80*	*~16%*
Bridges GPU	*157,500*	*16x2 K80 + 32x2 P100*	*~6%*
Comet GPU	*146,000*	*36x4 K80 + 36x4 P100*	*~35%*
OSG	*4,560,000*		*>100%*

TITAN

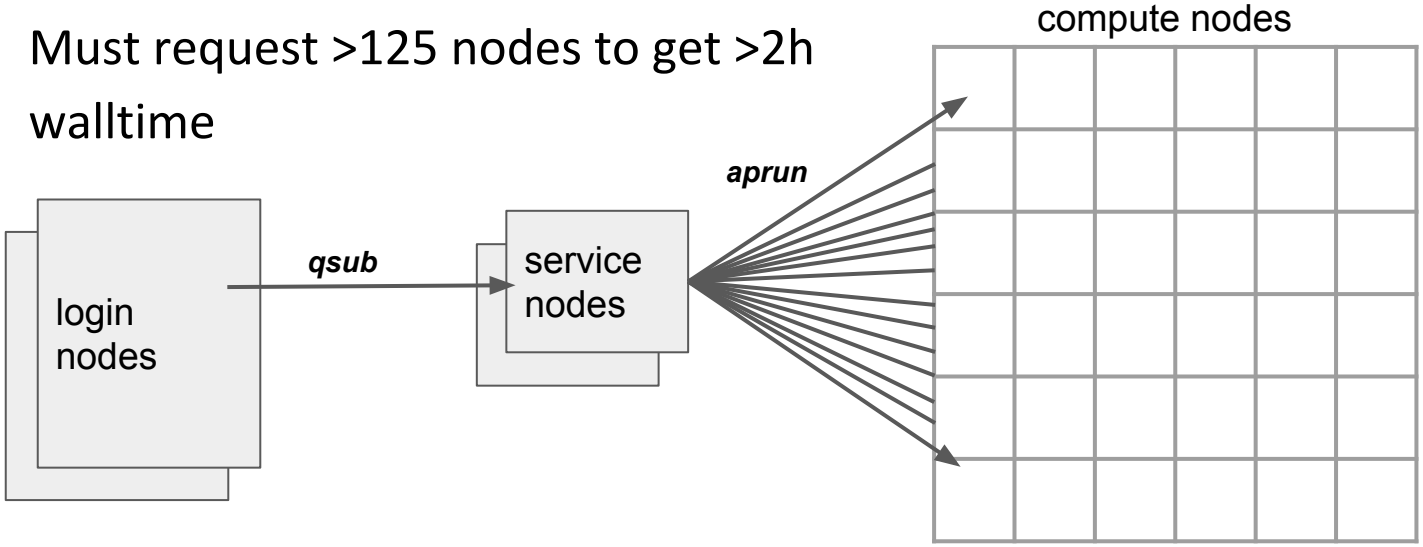
Cray XK7 at Oak Ridge Leadership Computing Facility

- 18,688 physical compute nodes
  - 16-core AMD Opteron, 32GB RAM
  - 6GB nVidia Kepler GPU
- ~300,000 cores, 600 TB RAM, 18.7k GPUs
- “Cray Linux Environment”: regular linux on service nodes, Linux microkernel on compute



TITAN

- Execute nodes on isolated network
- Very MPI-oriented
- Policies adverse to smaller jobs
  - Only 2 concurrent jobs <=125 nodes
  - Must request >125 nodes to get >2h walltime



HTCondor on Titan - Singularity jobs

Singularity container with HTCondor and subset of IceCube software

- ~40 minutes to rebuild container from scratch and upload to Titan
  - annoying for small changes
  - load as much as possible from outside container during development
- sshd for interactive debugging since Titan doesn’t provide it
- Few bash scripts scripts to start and manage condor pool

HTCondor on Titan: 1 Titan job = 1 Condor pool

Each node does real-time logging of resource usage (GPU, CPU, mem …)

Pool shuts down automatically a few minutes after running out of idle jobs (control cost, reduce wasted nodes)

- Pool resumed in a new job that may request different resources
- Condor state stored on shared file system

Pool can be monitored by sshing into central manager

Log files on shared filesystem

- Can be watched or analyzed from a login node

IceCube on Titan - Notes

Singularity support on Titan extremely, extremely useful

Development iterations on Titan can be slow. Debugging is inconvenient.

- Interactive access to running containers makes life less painful

HTCondor on TITAN works well

- So far, 45k simulations done (26% of allocation), 2.1 TB output

Working in close collaboration with ATLAS experts to test using Panda for running IceCube jobs in TITAN

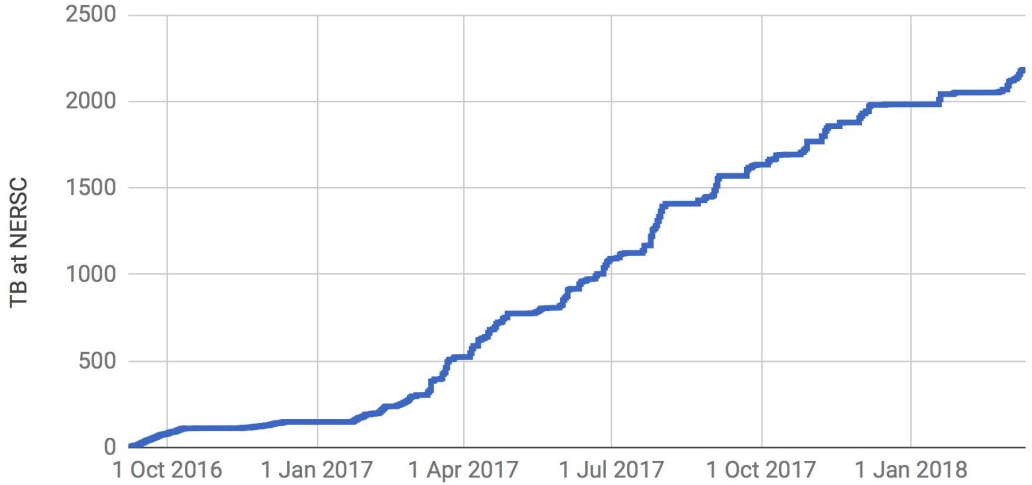
- Work in progress. Expect to wrap-up in the next 3 months.

Data Management

Long Term Archive

JADE Long Term Archive software components:

- Indexer - Metadata collection and validation
- Bundler - Creates large (>500GB) archives
- Mirror - Transfer via Globus from UW-Madison to DESY/NERSC



Long Term Archive - Issues

Operations are still very labor-intensive

- Indexing/Bundling - submitting HTCondor jobs to the local cluster
- Transfer - transfers scheduled as bundle files are produced
- Taping - moving archival bundle at NERSC from disk to HPSS

Reporting Tools

- Somewhat Primitive, Somewhat Buggy

Ongoing development. Can we use existing tools to do part of the work and concentrate in the IceCube specifics?

Rucio

We attended the OSG data management mini-workshop in January

Rucio seems to do several things that our data management system JADE also has to do - e.g. transfer file from A to B …

- We are more than happy to consider delegating some of these tasks to a 3rd party service
  - not interested in reinventing the wheel
- Some JADE functionality will continue to be “custom”
  - ingest from DAQ, satellite transfer ...

Rucio evaluation

We have proposed OSG to do a limited-scope evaluation of Rucio in the next months

- Goal: learn the details of Rucio by running it and find out ways to integrate with JADE

Scope: replicate the IceCube-2016 Level2 & Level3 datasets from UW-Madison to DESY-ZN (~150 TB) - a real-life task

Prototype service for this test provided by OSG (thanks!):

rucio-icecube.grid.uchicago.edu

Summary

New continuous integration system for nightly builds & CVMFS publication

- kubernetes very useful to get an easy and flexible deployment

Pyglidein new features

- Startd logs to S3, Startd cron for blackhole detection, GPU benchmarking

IceProd: dataset processing framework on top of HTCondor. Focus on dataset bookkeeping

- Also, workflow management functionality, pilot based

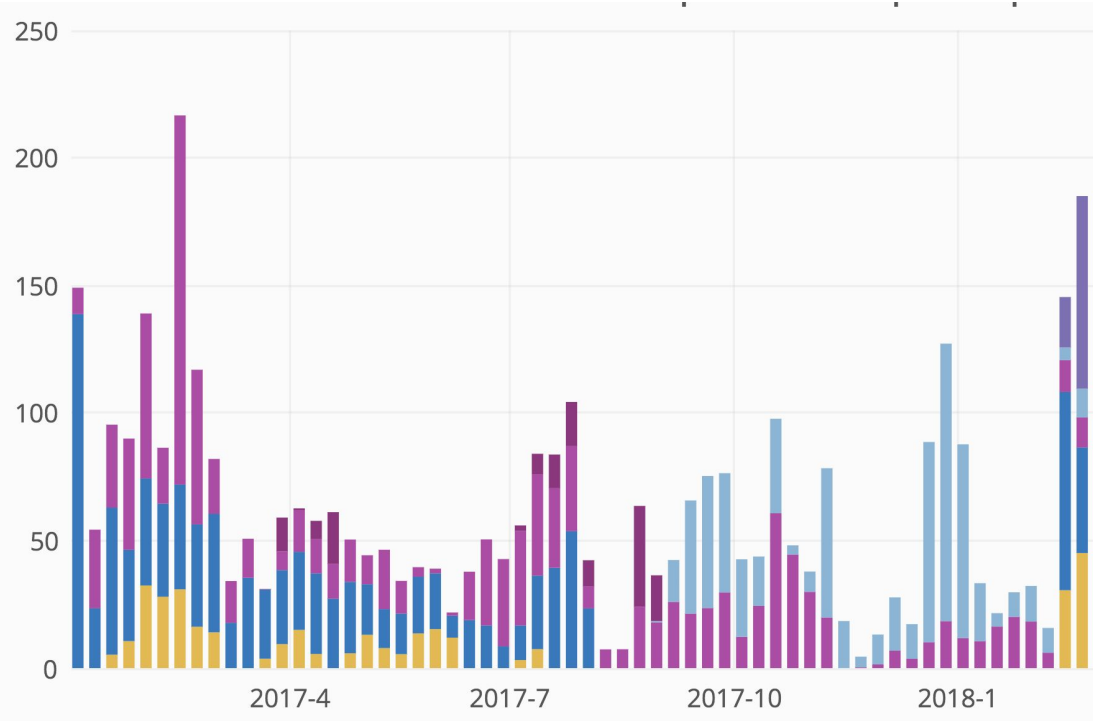
Supercomputers: Additional GPU capacity

- XSEDE mostly glidein/CVMFS friendly
- Currently testing different approaches in TITAN

thank you

backup slides

GPU used per week - OSG and XSEDE



Crane	21
Comet	18
Guillimin	14
SU-OG-CE	12
Bridges	5
Xstream	2
xstream	2

Metadata catalog

Motivation: various services that “handle files”

- Iceprod, JADE - multiple instances of each
- Each doing independent bookkeeping in internal DBs

Goal: Collect all file metadata in a central catalog

- Build a central catalog that contains all file metadata
  - Users could query it to find data files
- Can be used by all these services - keep one consistent view of the metadata for all IceCube files

Metadata catalog

Structure

- Python web server with REST API - speaks JSON
- MongoDB backend

Schema

- Core metadata (uuid, name, location, checksum, …)
- Specific metadata in sub-objects; can have more than one

v1.0 “beta” release soon - exposes a few useful queries

- files in a simulation dataset, good files in an IceCube run, files that contain a specific Event ID ...

```
$ condor_q 2723657.0 -af MachineAttrPRESIGNED_GET_URL0
https://pyglidein-logging-comet.s3.amazonaws.com/Comet_ef41844a-4b34-48a8-bc02-61fe4b17128d.tar.gz?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAJA5UTAFCQSD57VNA%2F20180317%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20180317T093004Z&X-Amz-Expires=604800&X-Amz-SignedHeaders=host&X-Amz-Signature=53163897eac0e008bb90075b5a106db0171c0d5ee227ebd988914ac77dada829

$ wget -O comet.tar.gz
"https://pyglidein-logging-comet.s3.amazonaws.com/Comet_ef41844a-4b34-48a8-bc02-61fe4b17128d.tar.gz?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAJA5UTAFCQSD57VNA%2F20180317%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20180317T093004Z&X-Amz-Expires=604800&X-Amz-SignedHeaders=host&X-Amz-Signature=53163897eac0e008bb90075b5a106db0171c0d5ee227ebd988914ac77dada829"

$ tar tvfz comet.tar.gz
drwxr-xr-x heaths/wis142     0 2018-03-17 08:42 log.198.202.117.246-175631/
-rw-r--r-- heaths/wis142 47826 2018-03-17 08:36 log.198.202.117.246-175631/StarterLog.slot1_2
-rw-r--r-- heaths/wis142 42951 2018-03-17 08:16 log.198.202.117.246-175631/StarterLog.slot1_4
-rw-r--r-- heaths/wis142  2068 2018-03-17 05:31 log.198.202.117.246-175631/MasterLog
-rw-r--r-- heaths/wis142 47707 2018-03-17 08:32 log.198.202.117.246-175631/StarterLog.slot1_5
prw------- heaths/wis142     0 2018-03-17 08:42 log.198.202.117.246-175631/procd_address
-rw-r--r-- heaths/wis142 42964 2018-03-17 08:20 log.198.202.117.246-175631/StarterLog.slot1_6
-rw-r--r-- heaths/wis142 51579 2018-03-17 08:36 log.198.202.117.246-175631/StartLog
-rw-r--r-- heaths/wis142 42060 2018-03-17 08:36 log.198.202.117.246-175631/XferStatsLog
-rw-r--r-- heaths/wis142 353029 2018-03-17 08:36 log.198.202.117.246-175631/startd_history
prw------- heaths/wis142      0 2018-03-17 08:42 log.198.202.117.246-175631/procd_address.watchdog
-rw-r--r-- heaths/wis142   1513 2018-03-17 04:31 log.198.202.117.246-175631/StarterLog
-rw-r--r-- heaths/wis142 934891 2018-03-17 08:42 log.198.202.117.246-175631/ProcLog
-rw-r--r-- heaths/wis142  47592 2018-03-17 08:35 log.198.202.117.246-175631/StarterLog.slot1_1
-rw------- heaths/wis142      0 2018-03-17 04:31 log.198.202.117.246-175631/InstanceLock
-rw------- heaths/wis142    134 2018-03-17 08:16 log.198.202.117.246-175631/.startd_claim_id.slot1
-rw-r--r-- heaths/wis142    134 2018-03-17 04:31 log.198.202.117.246-175631/.startd_address
-rw-r--r-- heaths/wis142  47829 2018-03-17 08:36 log.198.202.117.246-175631/StarterLog.slot1_3
-rw-r--r-- heaths/wis142    134 2018-03-17 04:31 log.198.202.117.246-175631/.master_address
```

GPU benchmarking info

```
$ condor_status -con 'DetectedGPUs > 0' -af Machine PYGLIDEIN_RESOURCE_GPU PYGLIDEIN_METRIC_TIME_PER_PHOTON GPU_NAMES

rad-7.icecube.wisc.edu true 12.40467981504772 GeForce GTX 1080
rad-8.icecube.wisc.edu true 12.37946663285881 GeForce GTX 1080
gzk-3.chtc.wisc.edu true 12.48916970311749 GeForce GTX 1080
gzk-4.chtc.wisc.edu true 12.1040769367873 GeForce GTX 1080
gtx-41.icecube.wisc.edu true 19.38806129013316 GeForce GTX 980
gtx-39.icecube.wisc.edu true 19.5445995407605 GeForce GTX 980
gtx-4.icecube.wisc.edu true 56.13695514464422 GeForce GTX 690
gtx-6.icecube.wisc.edu true 57.80035978324592 GeForce GTX 690
```

Long Term Archive - Globus

DESY identified an issue in the Globus-dCache interface

- Affects transfer efficiency

  https://github.com/dCache/dcache/issues/3545

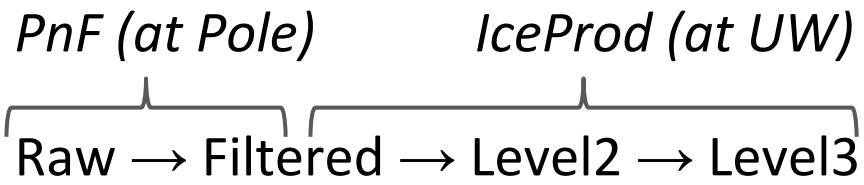
- dCache developers recently released a patch (thanks!)

Also, concerned about the long-term direction of Globus

- Going closed-source, support for gridftp not clear …

Reproducibility/traceability

The IceCube data processing pipeline is centrally managed down to Level3



User analysis starts at Level3 or Level2. Generates various data products which are finally used to produce results in publications.

- Tracing the data provenance is a challenge
- Goal: being able to track which data files went into generating a particular scientific result (publication)

Reproducibility/traceability