LIGO on OSG

Brian Bockelman
OSG AHM 2016

THIS IS NOT A LIGO TALK

(In fact, the LIGO collaboration meeting is this week on the west coast)

This is a talk about enabling LIGO on OSG

Ancient History:

- LIGO was an OSG stakeholder in the early days.
- However, we scared them off for a few reasons, including:
  - **OSG was hard to use**: Payload jobs were sent to GRAM using Condor-G. Quite unreliable and a foreign interface to users.
  - **No solution for software / data**: We asked sites to provide NFS mounts ($OSG_APP, $OSG_DATA) but these were inconsistently deployed and had no management tools.
  - **User-unfriendly requirement of certificates**: The process of getting a DOEGrid certificate was grueling.
- Running opportunistically on OSG required more effort / expertise / blood / sweat / tears than LIGO had to spare. Cost/Benefit didn't make sense!

The Challenge:

A decade later, can OSG do better?

Application Details - PyCBC

- PyCBC is a software suite to search LIGO data for Compact Binary Coalescence (CBC) events.
    - Looking for two large things spinning fast, then hitting each other.
- Driver written in python; compiles then invokes JIT-compiled C++ code.
- Workflow managed by Pegasus and executed using HTCondor.

The Solution - the players

- **Resource Provisioning**: GlideinWMS.
- **Job management**: HTCondor.
- **Data distribution**: Xrootd (originally GridFTP) from Nebraska.
- **Software distribution**: OASIS.

Job Management

- Resources were provisioned out of the OSG opportunistic pool: unique twist was LIGO required us to verify UID separation from other users.

- Additionally, XD allocations at Stampede were used (see https://indico.fnal.gov/contributionDisplay.py?contribId=33&confId=10571).

- Overall, about 4M CPU hours were used in fall 2015

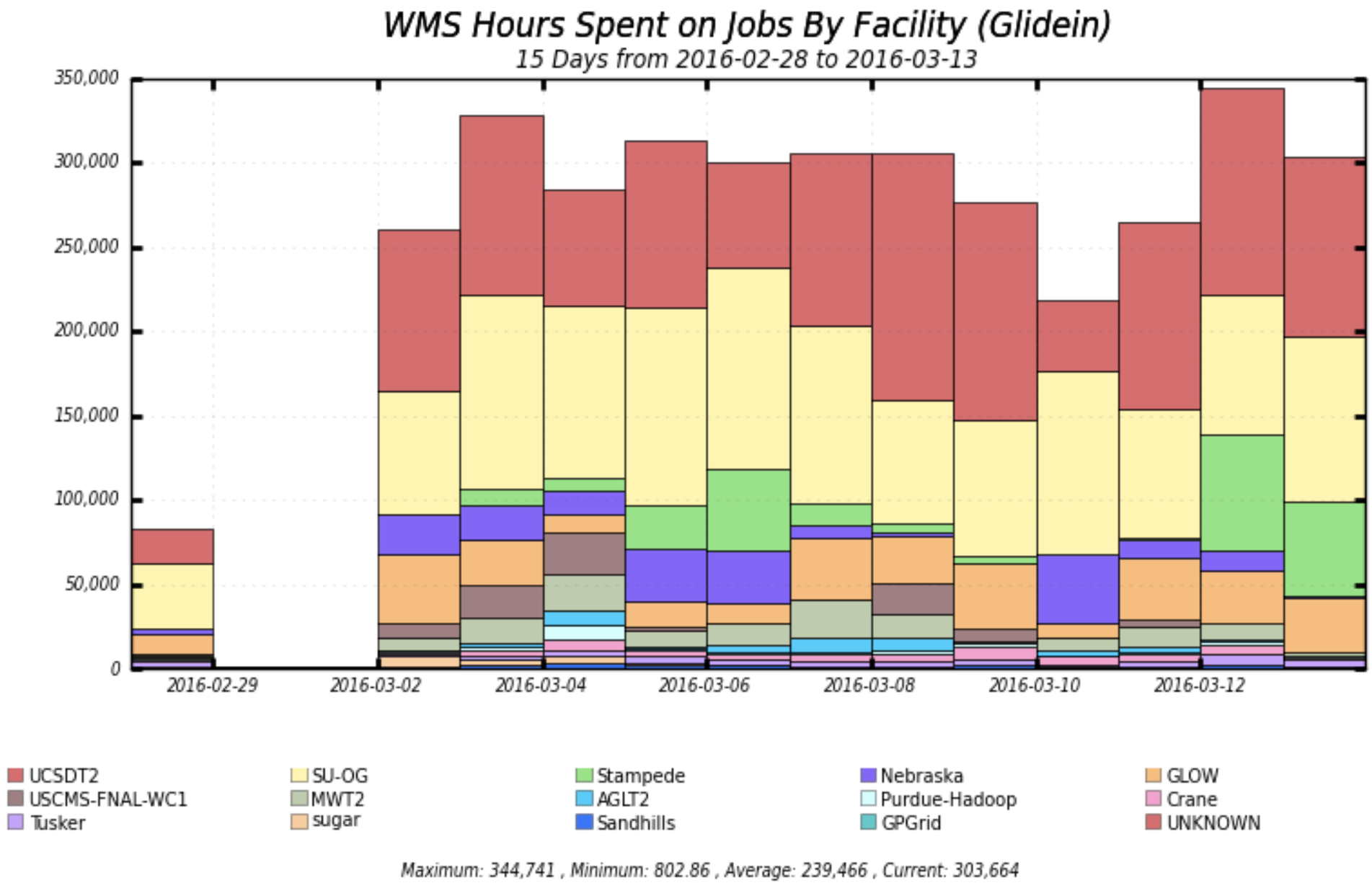
- After that, it was a “normal HTCondor pool” - just as LIGO users were used to.

```
[root@sugar-dev2 ~]# condor_q -totals


-- Schedd: sugar-dev2.phy.syr.edu : <128.230.146.12:18755>

22568 jobs; 0 completed, 0 removed, 10532 idle, 12036 running, 0 held, 0 suspend
ed
[root@sugar-dev2 ~]#
```

Last two weeks - 4.5M hours



(As a comparison, XD allocation is 2M SUs)

Data Distribution

- The relevant LIGO dataset is about 5TB and divided into 500MB files.

- Pegasus knows all files that a given job will need; each job will need 1-2 files.

- Jobs are several hours long: aggregate transfer rate works out to be about 1Mbps.

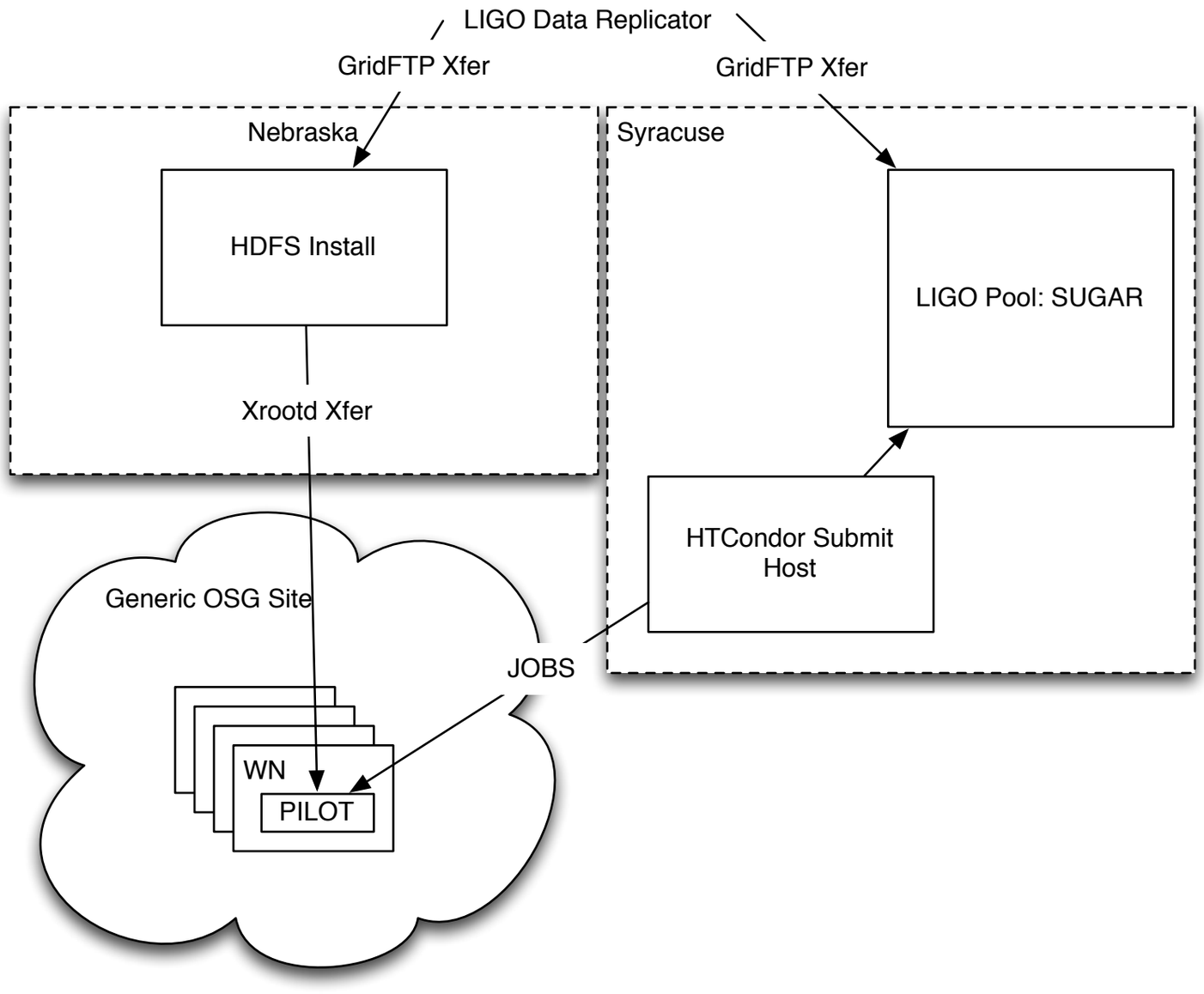
- **Solution**: opportunistically use the GridFTP/Xrootd infrastructure at the Nebraska CMS Tier-2; stream data remotely to jobs.

  - No staging of data to each site was necessary.

Data Distribution

As LIGO usually gets about 10k cores, typical throughput from Nebraska is 10Gbps.

From October to December, about 1PB of data was transferred.



Looking Forward

- We “got lucky” in our data distribution solution:
    - PyCBC’s data volume requirements were small enough to opportunistically use Nebraska.
        - Transfer rates were small enough to
    - PyCBC was using Pegasus, which can stage data.
    - Several just-in-time bugfixes from OSG and Pegasus team to glue it all together!
- Can we do better?

Looking Forward - ligo.osgstorage.org

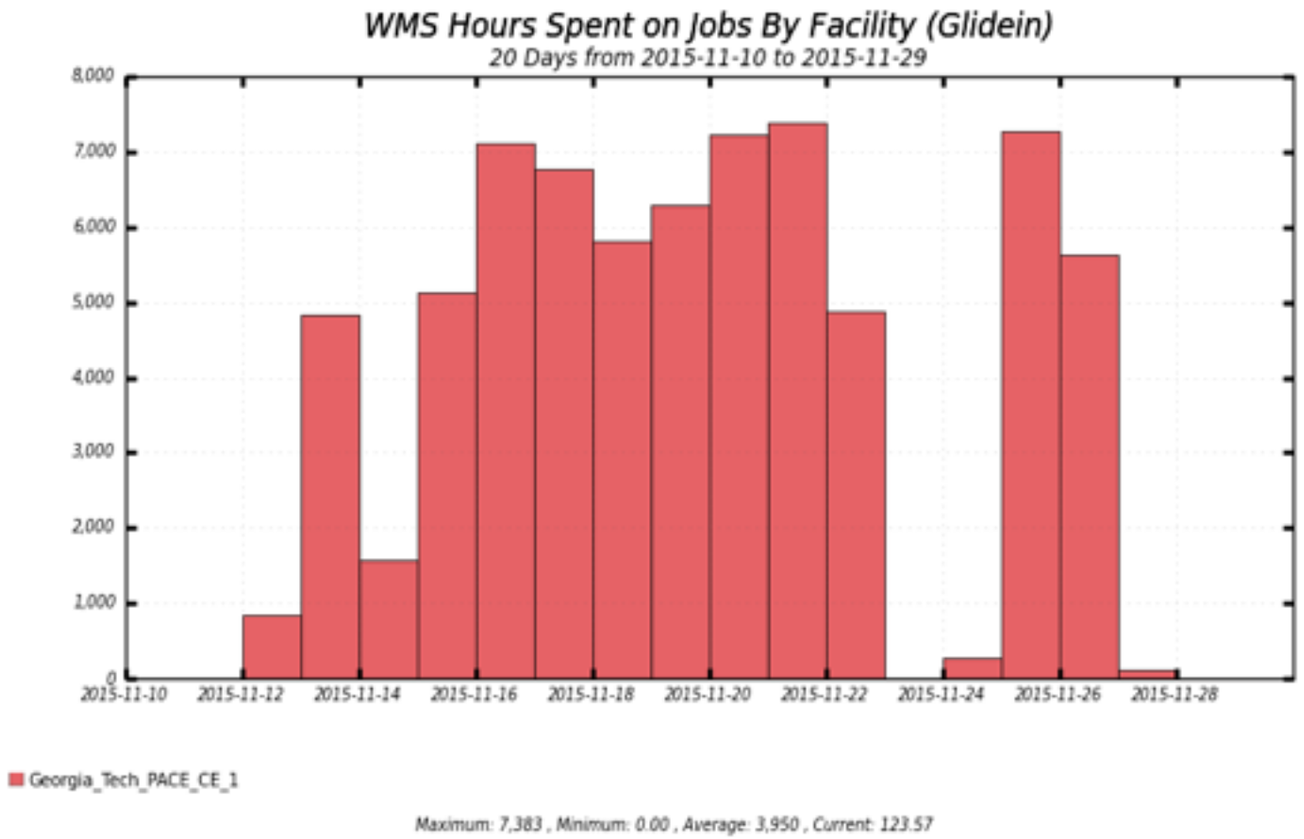
- We have been working with the CVMFS team to add features for exporting a data federation via CVMFS.
    - This allows us to securely provide global a POSIX mount with all of LIGO's data.
        - How? See Derek's talk tomorrow!
- Will require CVMFS 2.2.2 on the worker node: will take a few months for sites to upgrade. Probably large-scale usage by late summer.
- The underlying "data federation" will be the Nebraska T2 until Stash does authenticated exports.

A Second Challenge

- LIGO's dedicated resources have required a very specific host OS environment. Scientific software is distributed via RPMs and installed into the system.
    - This is quite pleasant for dedicated users but painful for shared clusters.
- Can OSG help here?
    - GA Tech was the first cluster we started with.

Well, sorta…

- Even though OSG can coexist better with a university cluster, there's still a high hill to climb.
  - Hard for busy sysadmins to keep things sustained!
- Not enough LIGO pipelines are converted to effectively utilize available resources.



Conclusions

- Using OSG services, LIGO has been able to provision LIGO owned, OSG opportunistic, and XD allocation-based resources at large-scale.
    - This has made a significant contribution to the PyCBC work.
- We were able to quickly execute this through reuse of solutions done for the OSG VO.
- OSG helped LIGO access contributed resources without requiring them to convert to LIGO Data Grid sites.
- Challenges remain:
    - PyCBC was the “best” workflow for OSG. Can we move others?
    - We could do a much better job of utilizing GA Tech.

So, did OSG discover Gravitational Waves?

- No!
  - It turns out the initial discovery was a “loud” signal that was noticed immediately.
    - However, OSG opportunistic computing helped support the analysis and improve the resulting paper.
- The CBC team is currently analyzing the remainder of the science run ending in mid-January.
  - Hopefully the OSG opportunistic contribution will lead to significant new results!