How Resources are Shared and Used through the OSG

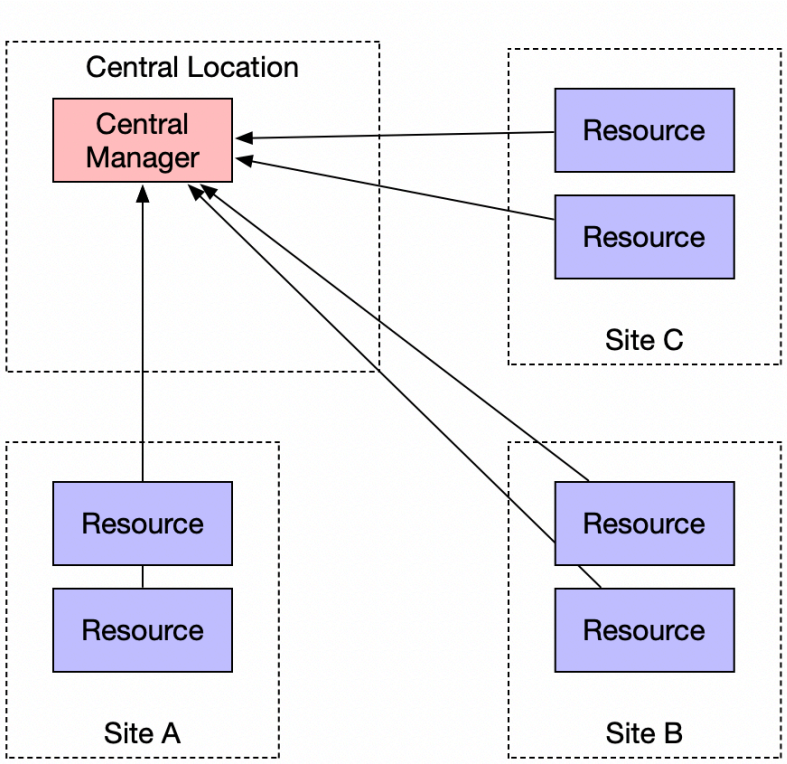
Brian Bockelman
OSG Technology Area Coordinator
Associate Scientist,
Morgridge Institute for Research

Sites and OSG

- A site on the OSG is a collection of resources – clusters – that are run by a specific administrative domain.
  - **Example**: the Holland Computing Center at Nebraska is an OSG site that makes several clusters available as OSG resources.
  - **Example**: A new CC* award recipient would like to make their new compute cluster available to LIGO and backfilled by other scientific work on the OSG.
- *Goal for today*: explain how OSG facilitates sites to share resources with the communities they are interested in.
  - Ideally, this helps meet your CC* goal of making 20% of your resources available for external communities.
  - A later presentation
  - Some technical details are present but this is meant to be fairly high-level.

Resource Sharing on the OSG

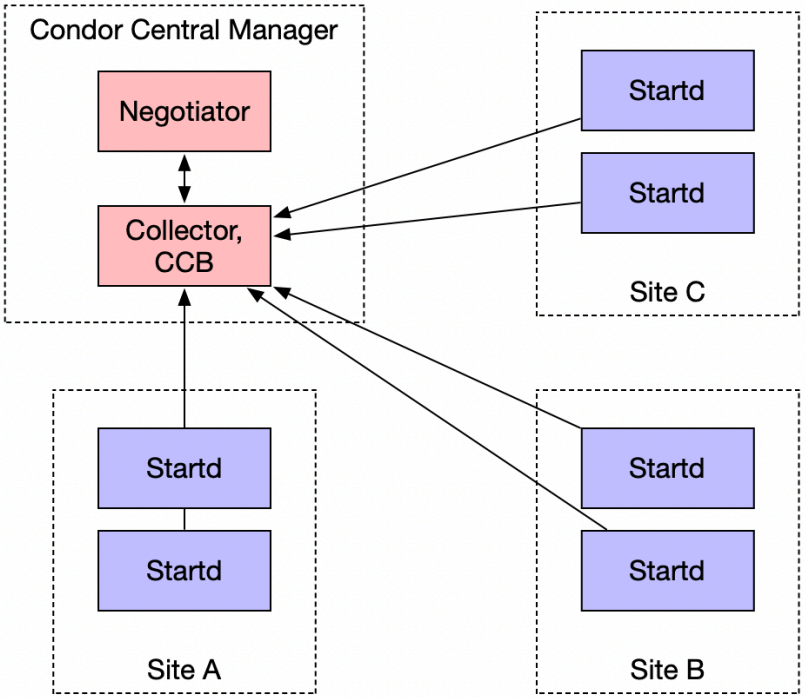
- OSG shares resources via the concept of an “overlay pool”.
  - Disparate worker node resources are allocated (think: starting a VM).
  - Some piece of software (“pilot”) starts on the worker node which subsequently connects to a central pool.
  - Work – batch jobs! - from a the outside is pulled down to the worker node and executed. These are “payload” jobs.
- Technologies we’ll see today:
  - HTCondor manages the jobs and central pool.
  - glideinWMS helps decide where and when to allocate resources.



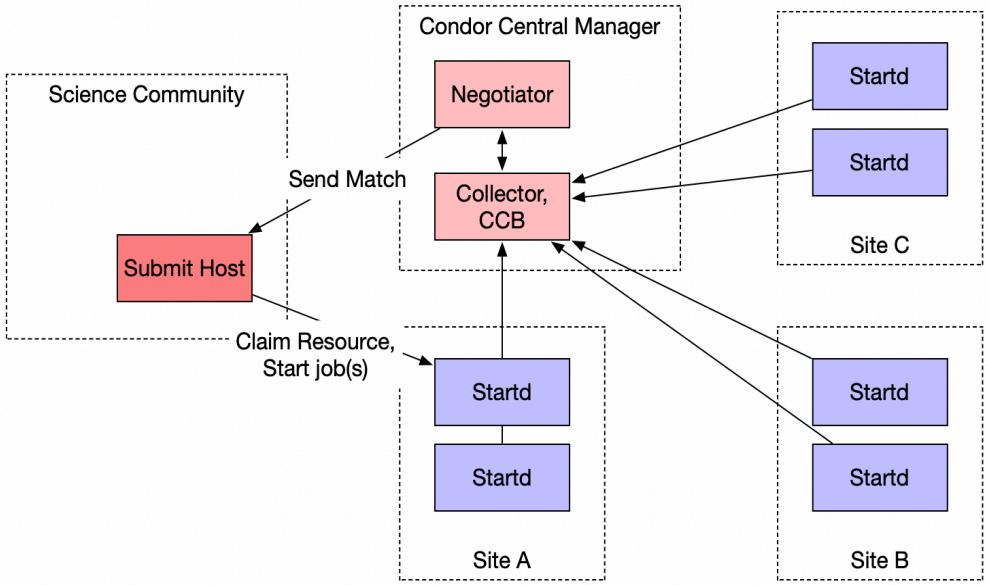
Now with Jargon!

In HTCondor Jargon:

- The software the pilot deploys on the site's worker node is called the **startd**.
- The central manager has three primary components:
  - **Collector**: daemon where all the resource descriptions are uploaded from the worker node.
  - **Condor Connection Broker (CCB)**: Used to manage network connections, allowing **startd** to be behind a NAT.
  - **Negotiator**: Implements policy and allocates the share of resources to different users.



Distributed HTCondor in 2 minutes



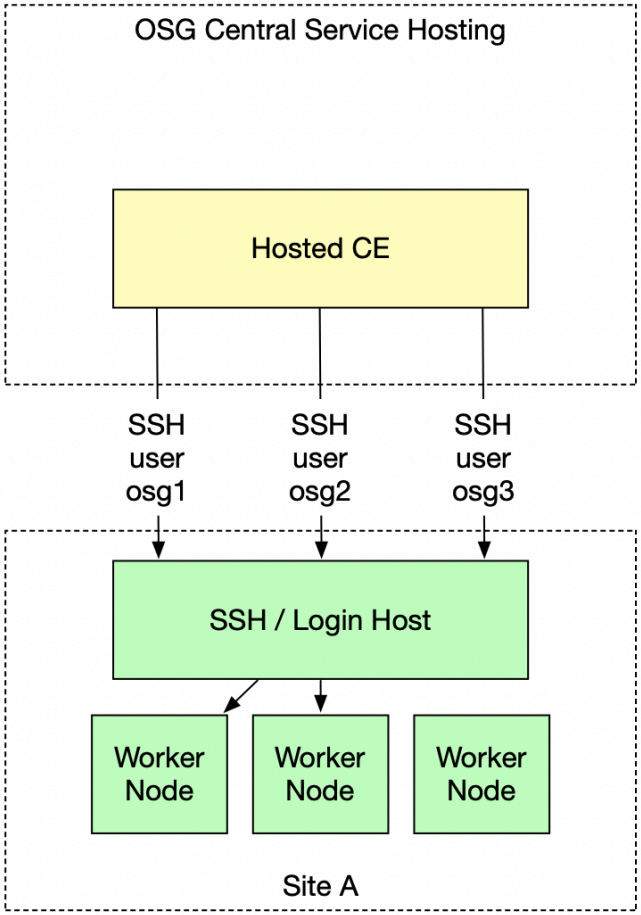
Complexities not shown:

- A science community may have several job submit hosts.
- Different communities (particularly, large ones) may decide to run their own pools.
  - OSG runs a pool for individual PI groups.
  - OSG also runs pools for larger collaborations such as LIGO.
- In this picture, all three sites are only supporting one community.

Allocating Resources

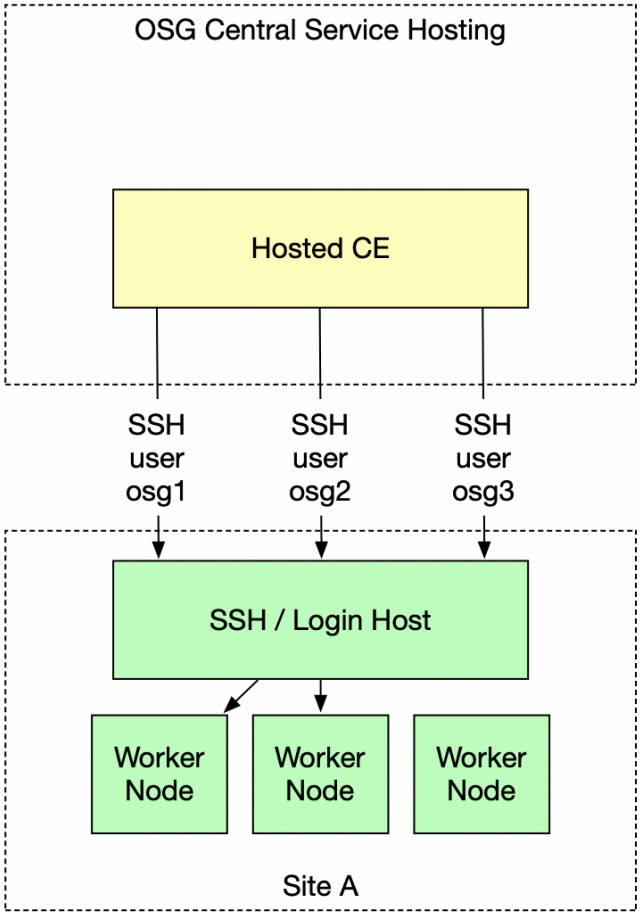
"Resources Allocations" are really batch jobs.

- These batch jobs arrive from an OSG-hosted "Compute Element" to the site login host.
  - One CE per SSH host.
- You can ban or prioritize individual communities without involving OSG as each community is a distinct batch user.
- OSG is here to facilitate resource sharing, not demand specific resource allocations.
  - There is a "special community" (confusingly, also called the "OSG"), run by OSG, which redistributes resources with an emphasis on single PIs.
  - The "OSG Community" also include XSEDE allocations, scientific collaborations -- all under the umbrella of 'open science' and research.
    - We have simple demographics and accounting for this community available to you.



Allocating Resources

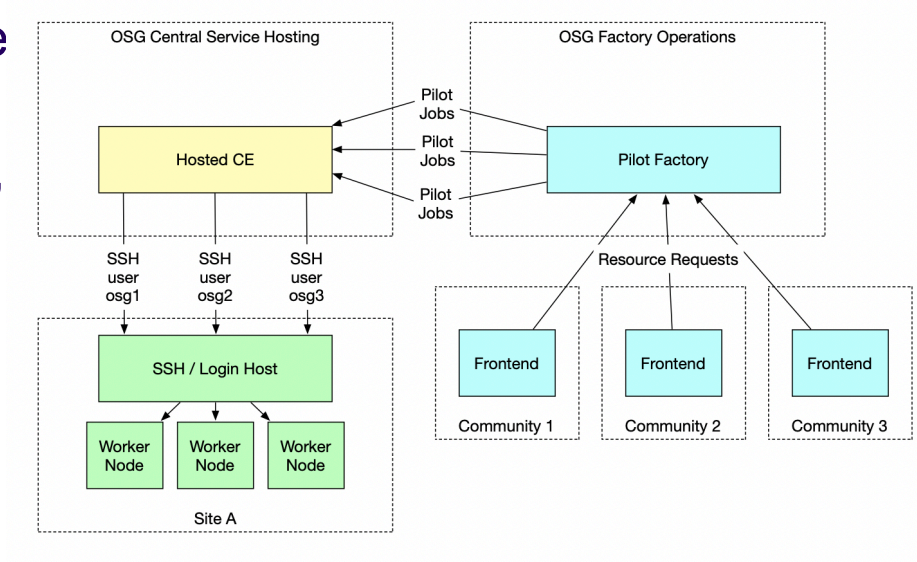
- Each community is mapped to a **different Unix account**.
- For ease of configuration, we request these be of the form osg01, osg02, … osg20.
  - **By pre-creating these accounts, we hope to never ask for new ones in the future**.
  - If this is an undue burden, let us know – we aim to keep startup costs low!
  - The mapping from "osgNN" Unix account to community name is found on our website.
  - We will provide an SSH public key – more later.



Allocating Resources

The OSG "factory" creates pilot jobs for each CE to submit to the local batch system.

- Most, but not all, communities use the OSG-run factory. Some run their own.
- Each community has a "frontend," which determines the resource requests per site based on their current job load and profile.
- The factory translates those resource requests to pilot jobs to various CEs.
  - The hosted OSG-CE will transform these to jobs appropriate for your site's batch system and submit over SSH.



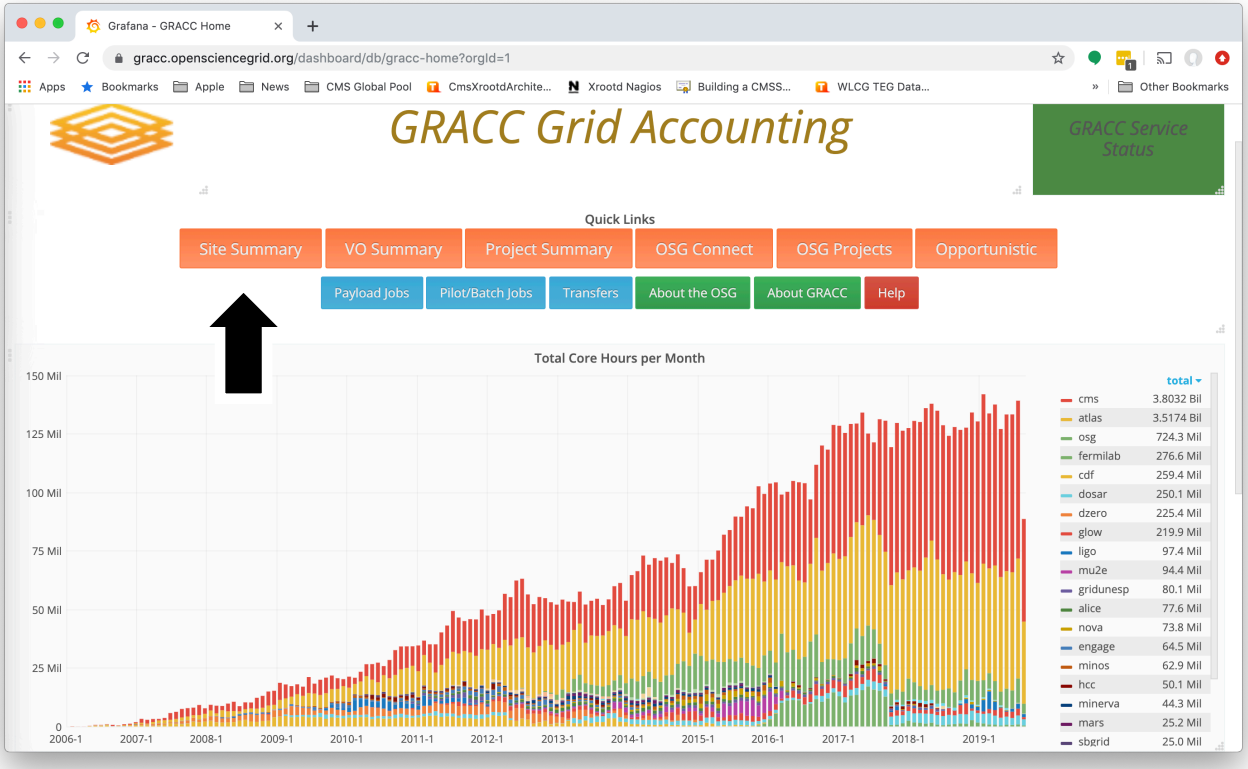
Resource Accounting

- CC* program aims to make available 20% of the resources to external communities.
  - Both “make available” and “20%” are surprisingly hard to define. How you interpret this is between you and ~~god~~ NSF – not OSG’s business!
  - OSG helps provide input to this process by making some simple accounting numbers available.
- Unfortunately, accounting is surprisingly subtle on OSG:
  - **Pilot** accounting tells us what compute resources were made available via the batch system.
  - **Payload** accounting tells us how the communities use the allocated resources.
- Ideally, these are identical: **in practice, they are not**!
  - Example: a pilot may start up but find that all payload jobs are already gone.
  - In most cases, these numbers are within 10% of each other.

Other Accounting Gotchas

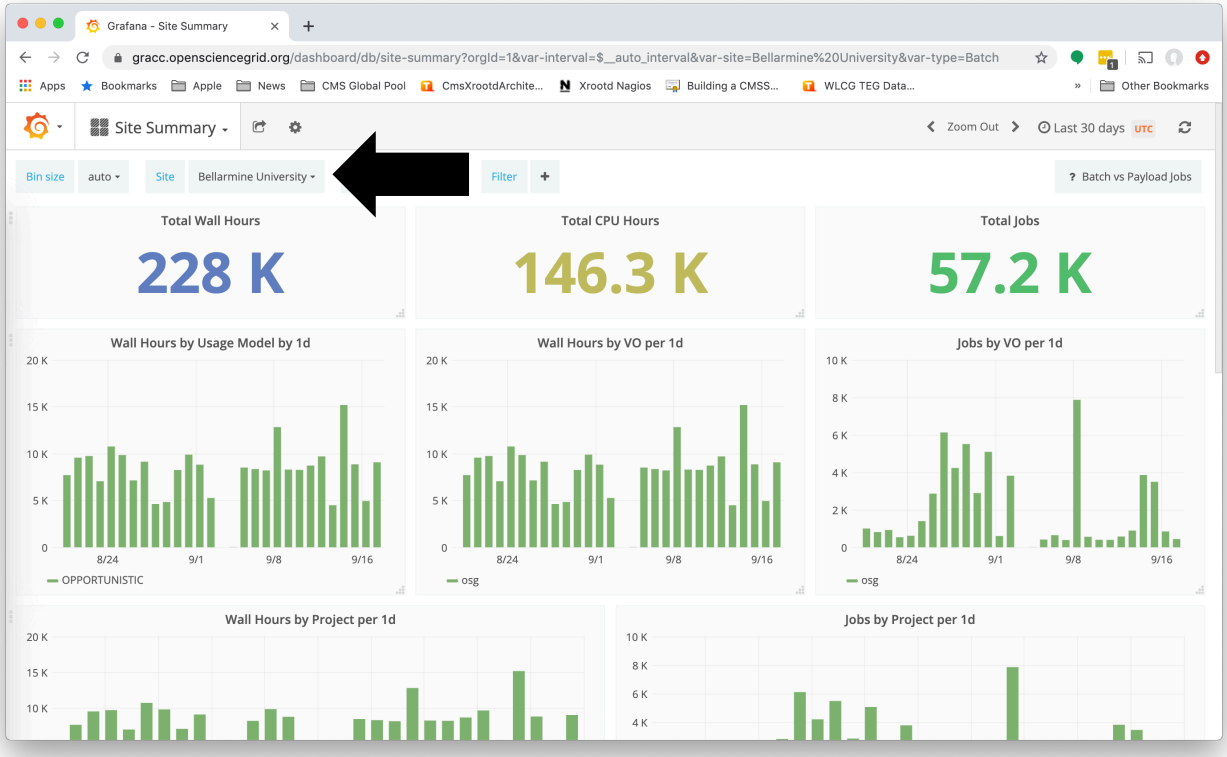
- A few other notable “gotchas”
  - Not all communities report the same details to OSG on how they use the resources.
    - **Example**: LHC community does not provide payload details.
  - Some communities utilize the OSG services to reach non-US sites.
    - **Example**: we only get payload information from European sites running IceCube, not pilot.
- Because this is complex, it’s useful to think carefully about what question you want to ask the system.
  - I’ll walk through a few screenshots on what I think is most important.

Grid Resource ACCounting (GRACC) portal



Accessible at https://gracc.opensciencegrid.org or by clicking on "explore our accounting portal" on the homepage.

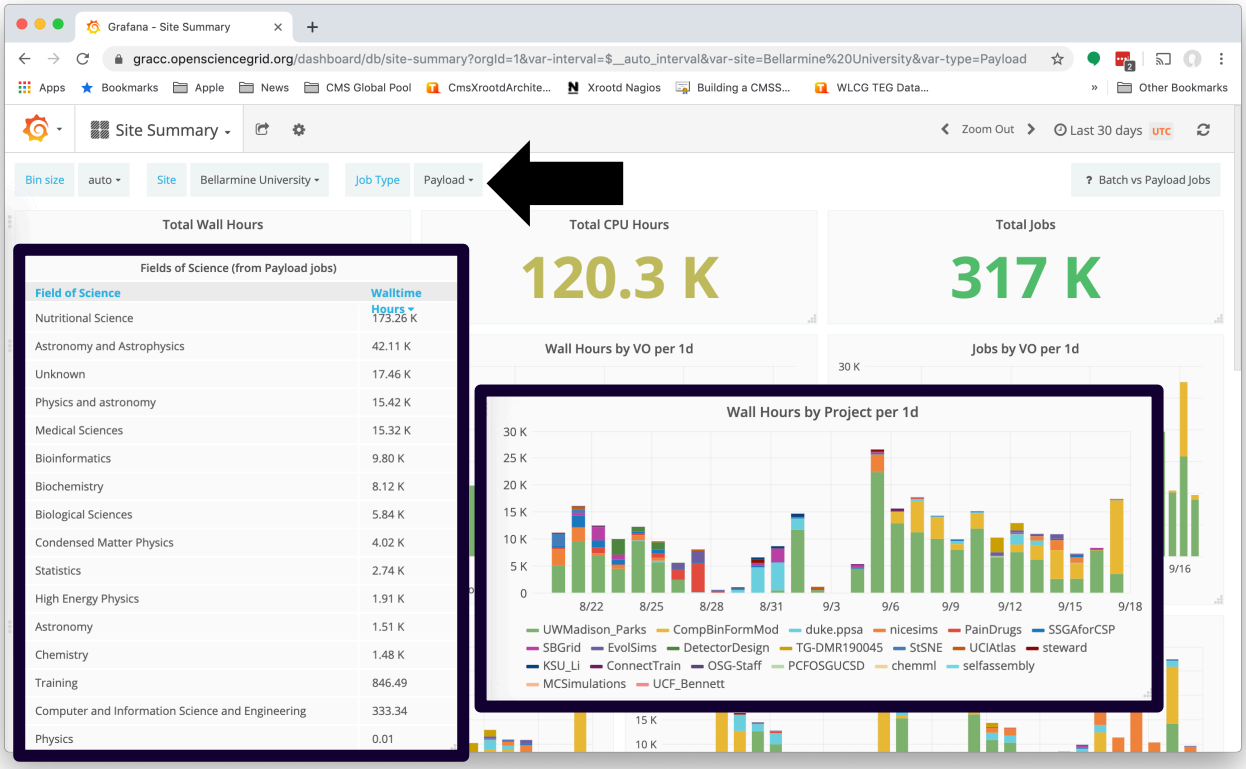
Grid Resource ACCounting (GRACC) portal



The “Site Summary” page defaults to all sites; use the drop-down to select your site name.

- You get to pick your site name. Names like “Bellarmine University” tend to be more descriptive than “KR-KISTI-GSDC-02”.
- This shows the view by community; in this case, only the special “osg” community was run.

Grid Resource ACCounting (GRACC) portal



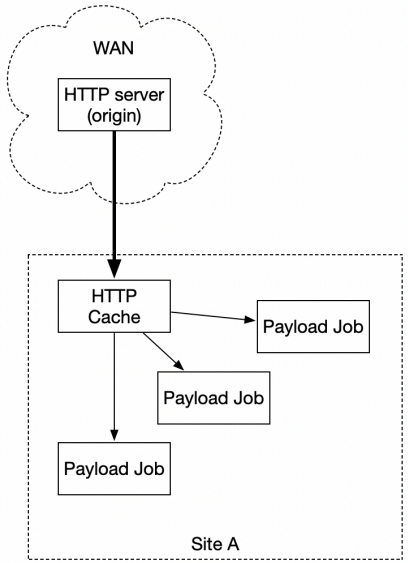
- Switch to the “payload” job type to get information about the resource usage, including
- The projects names inside the community.
- The corresponding fields of science.

Sharing Resources More Effectively

- So far, we've talked about the simplest way to share resources via the OSG. Requires no site-run services and aims to minimize required site effort.
- There are services you can run at your site to either (a) improve the efficiency of jobs or (b) attract a broader range of jobs. Your site does more science – but more effort is involved!
  - **HTTP Cache**: helps avoid frequent retransfer of many (small) resources over HTTP.
  - **CVMFS**: Global, read-only, caching filesystem for distributing containers and software (data is moved via the HTTP cache).
  - **Singularity**: Allows us to launch jobs inside containers.
  - **Data Caches**: Helps jobs avoid moving large data repeatedly.
- I'll include links to documentation; tackle these if desired (and after the basics are working).

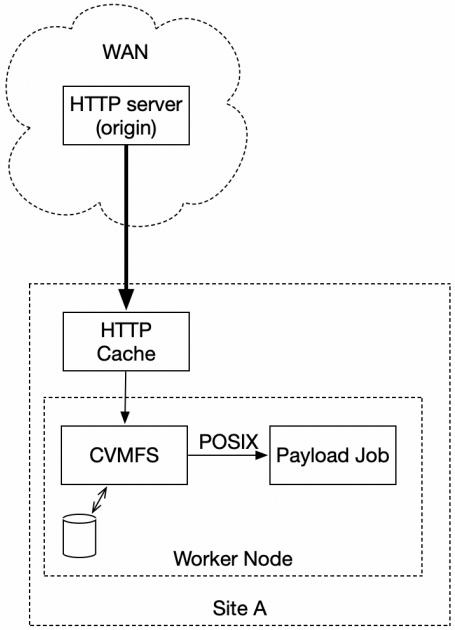
Sharing Resources More Effectively - HTTP Cache

- A broad set of data – software, configurations, job inputs – can be moved to the worker node via HTTP. A significant amount is very frequently reused.
  - By placing a HTTP cache on-site, repeated data use only goes over the LAN instead of the WAN.
  - Any caching HTTP proxy can work for OSG. However:
    - Not all scale well in terms of concurrency.
    - Most are tuned for HTML files, not objects in the >1MB range.
  - We work with the Frontier project to support a special configuration of the venerable Squid software, frontier-squid. Monitoring, logging, and configuration are tuned specifically for OSG usage.
- Sysadmin Documentation. Complexity level: **Easy**.



Sharing Resources More Effectively – Distributed Filesystem (CVMFS)

- CVMFS is a global, integrity-checked, read-only POSIX filesystem. This achieves scale by distributing data through HTTP caches and a CDN.
  - Everything is cached – filesystem metadata and data – all the way to the local worker node. Data is moved to the worker node only on access.
  - In OSG, we use this as a mechanism to distribute science community containers and software. Given the popularity of containers, many jobs require this.
  - **Downside**: This is software that is run on the worker node, which adds complexity.
  - **Downside**: Implemented using FUSE and autofs, two tricky technologies.
  - **Good news on the horizon**: In CentOS 8, this can be done by the batch job completely unprivileged. Nothing to install or monitor on the worker node.
- Sysadmin Documentation. Complexity level: **Moderate**.



Sharing Resources More Effectively – Containers (Singularity)

- Singularity is a container runtime that aims to fit the needs of running containers inside a batch system. No running daemon like Docker – just a process inside the batch job.
  - For full functionality - and on RHEL6 – a `setuid` (extra privileges) binary is needed.
  - For OSG use cases – and on RHEL7 – we recommend using unprivileged.
- The pilot will invoke Singularity prior to starting the payload; this way, the pilot sees the host operating system and the payload sees the container of its choice.
  - The containers are typically distributed via CVMFS.
- OSG provides targeted support for the "community edition" of Singularity; most of the core developers work for a startup company (Sylabs).
- Sysadmin Documentation. Complexity level: **Easy**.

Sharing Resources More Effectively – Data Caches

- The Frontier-Squid software targets the distribution of “small-ish” objects – less then 1GB:
  - Is inefficient to use for files over 1GB.
  - Does not provide a mechanism to securely cache proprietary scientific data. (Note: OSG does not provide mechanisms suitable for HIPAA data.)
- We have a separate software (XCache, a special configuration of the XRootD software) to fill this role.
  - Designed for delivering 1-10GB of data to jobs where there is cache-friendly access and the total working set size of a workflow is <10TB.
  - Provides mechanisms to authenticate and authorize
  - In the end, still transfers data via HTTP / HTTPS.
- Sysadmin Documentation. Complexity Level: Moderate / Hard.

Save This Slide!

- What do we need in order to run at your site?
  - User account(s) setup.
  - Worker nodes need outgoing network connectivity to the central pool and submit hosts. NAT is fine!
    - The outgoing IP addresses will vary from community to community.
  - An automated, external, component will need to submit jobs to the batch system. OSG hosts this service; jobs are submitted over a SSH connection.
    - Some large, complex sites host their own; not a great place to start.
- What additional site services are useful?
  - HTTP cache: helps avoid frequent retransfer of many (small) resources over HTTP.
  - CVMFS: Global, read-only, caching filesystem for distributing containers and software. Runs on worker nodes using FUSE.
  - Singularity: Allows us to launch jobs inside containers.
  - Data Caches: Helps jobs avoid moving large data repeatedly
- Individual science communities (such as LHC) may require additional services; let us know who you want to support and we can provide more details.

Questions?

This material is based upon work supported by the National Science Foundation under Grant No. 1148698. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.