

# OSG Info Services: Philosophy and Technology

Brian Bockelman

# Preamble

- About three years ago, OSG decided to start on the path of retiring the BDII.
  - At first, the focus was to reduce existing use cases and avoiding new use cases.
  - This is about when we decided to not upgrade to GLUE2.
- About 12 months ago, OSG started to work on a replacement for some core functionality, the OSG Collector.
  - First widely-usable version 6 months ago.
- We will work with stakeholders migrate to the new systems; no intent to “pull the rug” out from anyone.

# The OSG Approach

- We think it is important to differentiate between who provides the data, the frequency of update, and where it should enter the system.
- To address each case, we must decide if the information is collected:
  - Centralized or decentralized.
  - Automated or human-entered.
  - With a timestamp and an expiration date.
- The ideal data formats may differ.

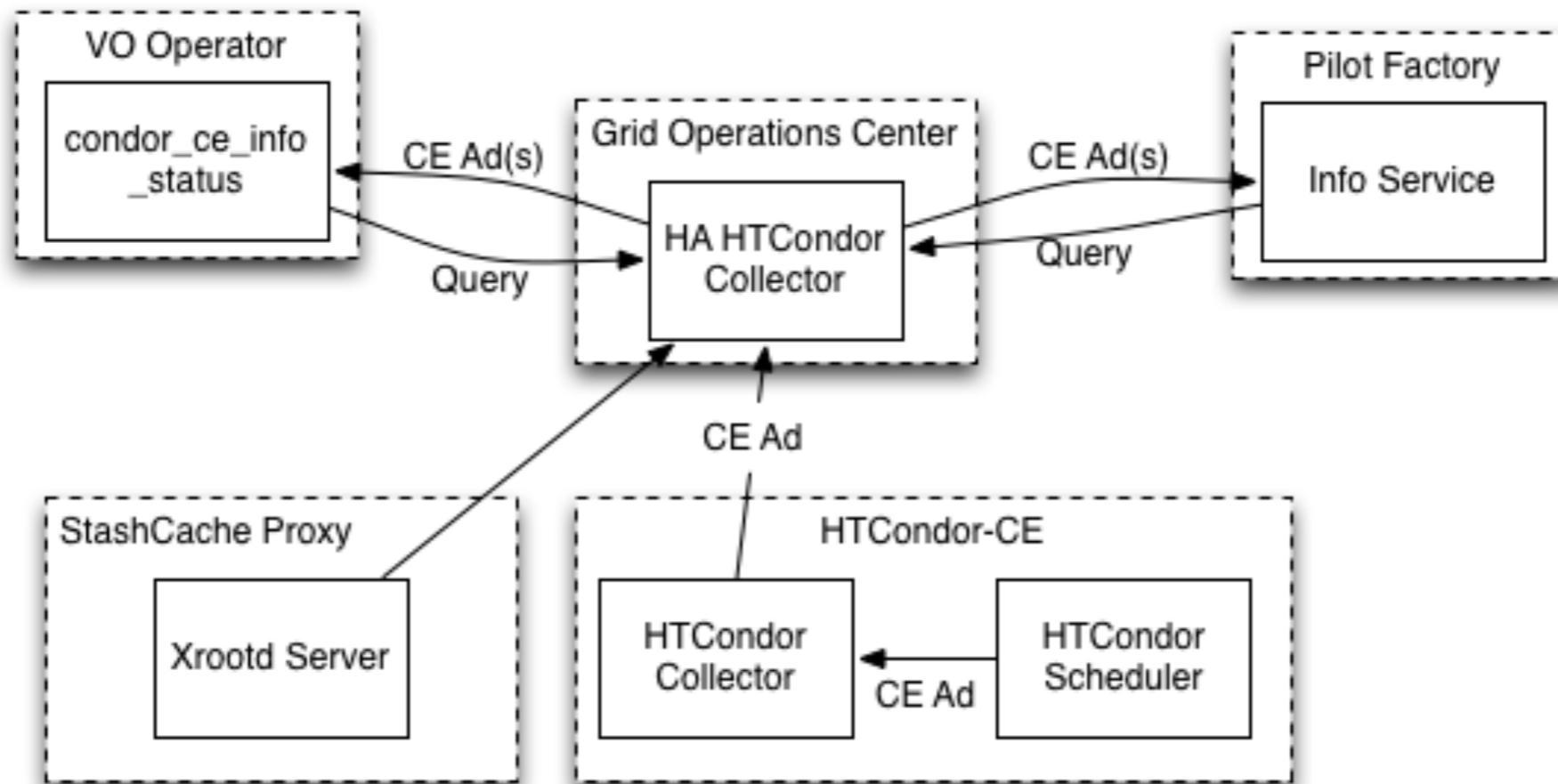
# Centralized, Static Data

- The OSG Information Manager (OIM) is where humans can enter static data.
  - Defines topologies of sites, resources, and services.
  - Resource information like pledges, normalization factors.
  - Provides appropriate access controls; not everyone has permission to edit all information.
  - Very analogous to GOCDB.
- The publicly accessible interface to OIM is MyOSG.
  - Primarily exports XML in a custom schema.

# OSG Collector

- The OSG Collector provides dynamic data about running services.
- Each service is represented by a single HTCondor ClassAd.
- The implementation is nothing more than a `condor_collector` daemon running centrally.

# Component Architecture



# Example: HTCondor-CE ClassAds

- The HTCondor-CE publishes a ClassAd providing information about:
  - Install details: backend batch system, version, OIM-registered site name.
  - Contact details: How to locate the running service.
  - Resource catalog: resource types.

# Resource Catalog in ClassAd is equivalent to this page:

The screenshot shows the AWS console page for Amazon EC2 Instance Types, specifically for Compute Optimized C4 instances. The page includes a navigation sidebar on the left, a main content area with a description and features, and a table of instance types on the right.

**Amazon Web Services**  
PRODUCTS & SERVICES

- Amazon EC2 >
- Product Details >
- Instances** >
- Previous Generation Instances >
- Purchasing Options >
- Pricing >
- Getting Started >
- FAQs >
- Developer Resources >
- Amazon EC2 SLA >
- AWS Management Portal for vCenter >

**RELATED LINKS**

- Amazon EC2 Spot Instances
- Amazon EC2 Reserved Instances

## Compute Optimized

### C4

C4 instances are the latest generation of Compute-optimized instances, featuring the highest performing processors and the lowest price/compute performance in EC2.

**Features:**

- High frequency Intel Xeon E5-2666 v3 (Haswell) processors optimized specifically for EC2
- EBS-optimized by default and at no additional cost
- Ability to control processor C-state and P-state configuration on the c4.8xlarge instance type
- Support for [Enhanced Networking](#) and Clustering

Model	vCPU	Mem (GiB)	Storage	Dedicated EBS Throughput (Mbps)
c4.large	2	3.75	EBS-Only	500
c4.xlarge	4	7.5	EBS-Only	750
c4.2xlarge	8	15	EBS-Only	1,000
c4.4xlarge	16	30	EBS-Only	2,000
c4.8xlarge	36	60	EBS-Only	4,000

### C3

# Resource Matching

- Resource catalog is a list of sub-ClassAds. Each one:
  - Describes a resource type accessible behind the CE.
  - Describes any specific pilot requirements to access the site.
  - Describes what must be in the pilot job submission to access the given resource.
- Great further reading: <https://indico.fnal.gov/getFile.py/access?contribId=19&sessionId=8&resId=0&materialId=slides&confId=8580>

# Schema-Free

- While there is a defined set of attributes, an important aspect of using ClassAds is they are schema-free.
  - If a VO needs specific information from associated sites, they can define.
- For example, if USATLAS would like to define a “JobType” attribute, all sites that want to support USATLAS can fill in this attribute appropriately.
  - No need to formally define “JobType” outside the USATLAS context or involve middleware devs.

# Routing - No queues!

- The HTCondor-CE is meant to expose no batch configuration information (such as queues!).
- The pilot job describes the resources it needs.
- A “job route”, internal to the CE, converts it to the appropriate batch system job.
- ... or VM or docker containers or \$FUTURE\_TECHNOLOGY.

# Targeting Resource Provisioning

- Ads in collector provide information to **pilot factories** about what resources are accessible and how to access them.
- Not meant to provide monitoring of job statuses.
- There's significant per-CE information in the BDII *beyond* this use case. When implementing the OSG Collector-to-GLUE translator for CERN, we noticed these are no longer present:
  - Total cores in the batch system (is this needed? Can the information from MyOSG be used instead)
  - Site normalization constants (have per-resource-type, but provide no way to convert to site-wide).
- To provide a full GLUE description, we would need to aggregate site information from the site & central systems, automated & manual systems. More on that later.

# Accessing CE Information

- Collaborating with CERN IT, we have a prototype-level script to translate OSG Collector ads to GLUE.
  - Not perfect: there are different semantics between the two. For the use cases in our document, I think there's sufficient overlap.
- Loss of information when translating to GLUE attributes are not provided, particularly in the schema-free attributes.
  - Again, not a problem if we stick to stated use cases.
- Idea is to provide an easier route for adoption. VOs don't need to understand collector ads if they already understand GLUE.
  - Note this is different from OSG running a BDII service!
- **IMPORTANT:** See GLUE as an interoperable language. Distinct from both BDII and CE- / site-centric generation of GLUE!

# Storage

- In the storage domain, within the OSG there's little demand for more than a service registry (MyOSG), particularly outside WLCG.
  - And, within the WLCG, we're still struggling a bit to understand what is needed.
  - Seems nothing for CMS; not clear on ATLAS. This is our next project with Ale!
- Hence, storage in the OSG-Collector is still a bit of playground.
  - We are working on making sure the OSG-run Xrootd federation (StashCache) has its distributed endpoints represented here.
    - Very nascent, we'll see where it goes.

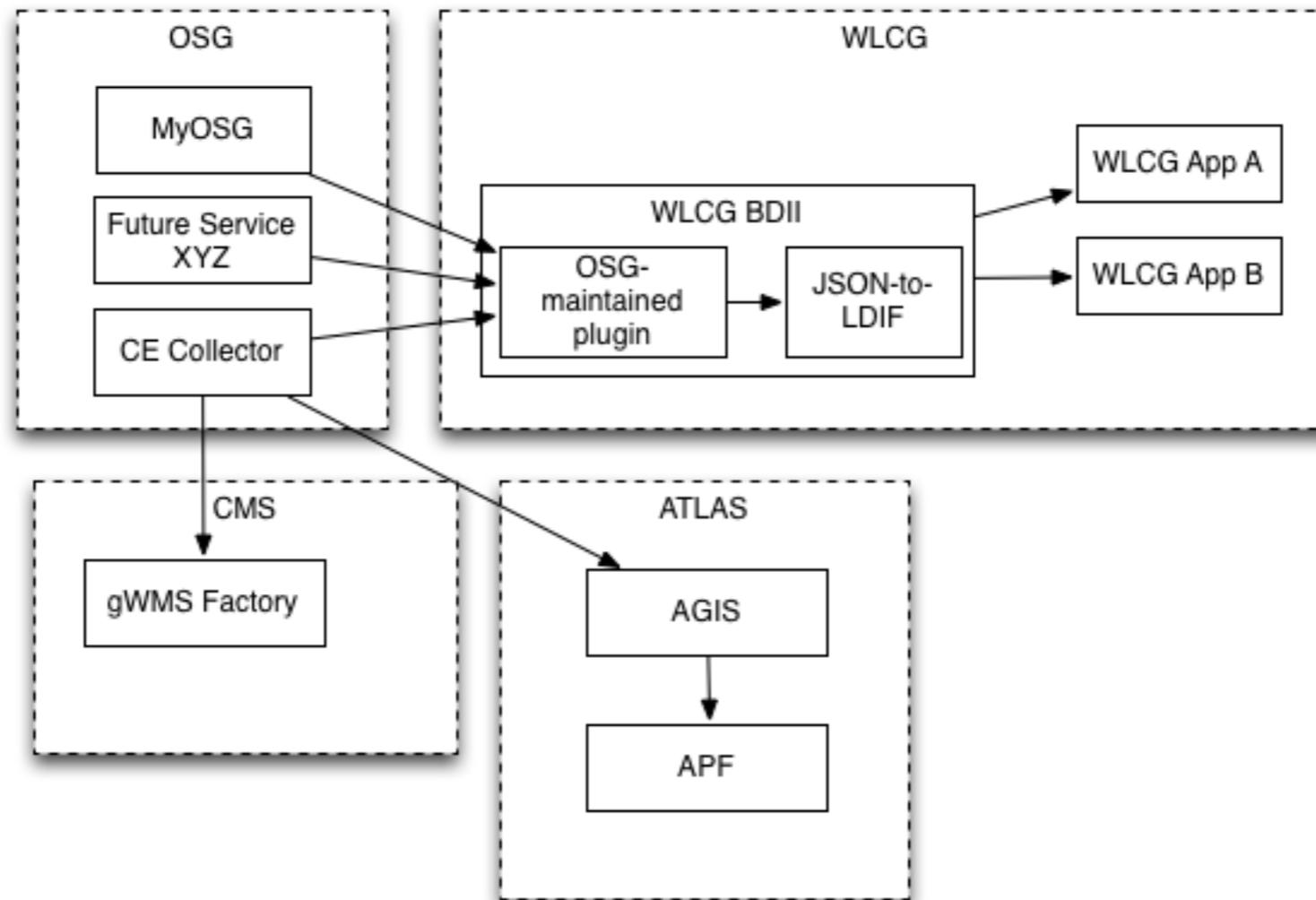
# Futures

- OSG Collector's first and foremost role is to provide information for resource provisioning within OSG.
- One direction of growth is adding new resource types. As noted, we have (very new) integration with Xrootd servers.
- Work remaining to do with explicit policies for data lifetime and offline services.
- Strong emphasis in migrating OSG stakeholders to new services.

# Futures

- It's an important to differentiate what OSG considers the appropriate system for its ecosystem, use cases, and stakeholders ...
  - ... and what's right for interoperating with the WLCG.
  - We can continue to develop and mature translators from our internal representation (ClassAds, XML) and systems (OSG-Collector, MyOSG) to the schema WLCG picks.
  - In terms of workforce and expertise, there's little long-term future for OSG running a BDII service. Preference to provide information via HTTP / JSON representation.
- Just recall OSG management will ask me:
  - What stakeholder (USCMS or USATLAS?) needs this?
  - How will this help our stakeholders meet their MoU commitments or improve their science throughput?
- I need to come armed with good answers!

# One Possible Approach



Just one potential approach - no commitments!

Relevant questions:

- Push or pull? Pull is illustrated above.
- Would experiments participate as part of WLCG or standalone?

# Appendix: Info mapping

- (Very incomplete)
- List of OSG sites / topology -> MyOSG.
- Installed capacity -> MyOSG.
- Contact information for a given CE -> OSG Collector.
- CEs that can provide 24GB / 8 core pilots -> OSG Collector.
- Batch slot normalization constants -> MyOSG.
- Administrator contact info -> MyOSG.