# VC3 - R&D for Virtual Clusters

Paul Brenner, Rob Gardner, Mike Hildreth, John Hover, Kevin Lannon, David Miller, Doug Thain

BROOKHAVEN NATIONAL LABORATORY

THE UNIVERSITY OF CHICAGO

UNIVERSITY OF NOTRE DAME

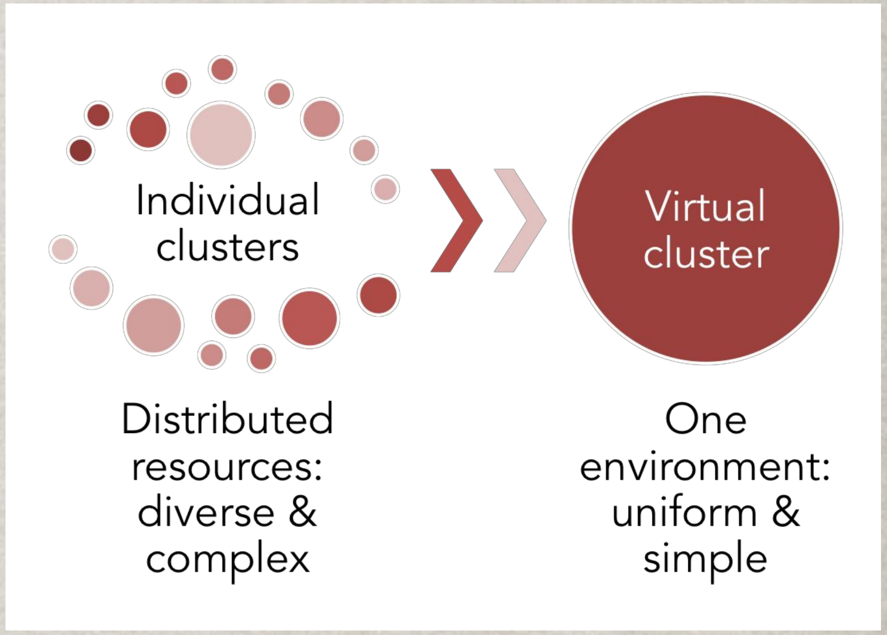
## My Perspective

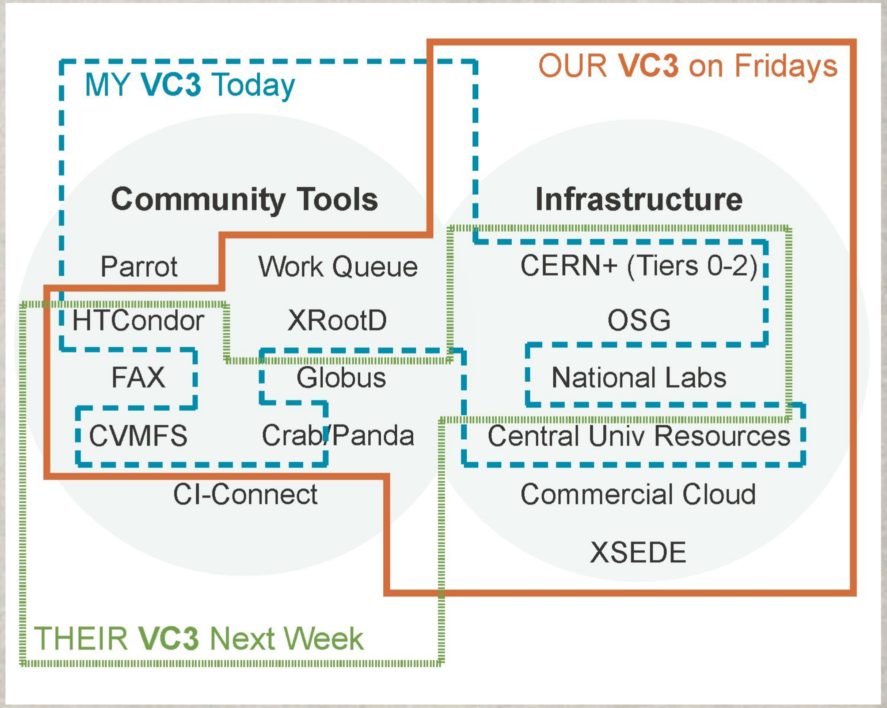
- Member of CMS collaboration
  - ~3000 collaborating scientists
  - Well-developed computing infrastructure making extensive use of OSG
  - Physicists, not computer scientist
- Faculty at Notre Dame
  - ND Center for Research computing hosts 25k cores of computing mostly owned by individual PIs
  - Cycles made available opportunistically via local condor pool, but not on OSG

## What is VC3?

### Virtual Clusters for Community Computation

- Cluster = optimal computing abstraction
- Self-service model inspired by commercial clouds
  - Users allocate raw computing resources from providers
  - Users responsible for configuring resources and/or adapting workloads
- Pros: Maximum flexibility to deploy applications and collaborate with users
- Cons: Greater burden on end users ⇒ Need better tools!


Individual clusters
Virtual cluster
Distributed resources: diverse & complex
One environment: uniform & simple



## Who Needs This?

- Driven by HEP science use case (ATLAS + CMS)
  - But can be generalized to many other use cases
- Focus on smaller groups: individual PIs or small collaborations of experimentalists and/or theorists
  - Large-scale, experiment-wide production already well served by existing tools
- Concrete examples:
  - PI with a campus-based resource not shared on the OSG
  - PI wins an allocation at an HPC LCF/AWS grant/allocation on campus cluster, and wants to collaborate with a small group
  - Organizers of a workshop want to provide a temporary computing environment combining resources from several universities

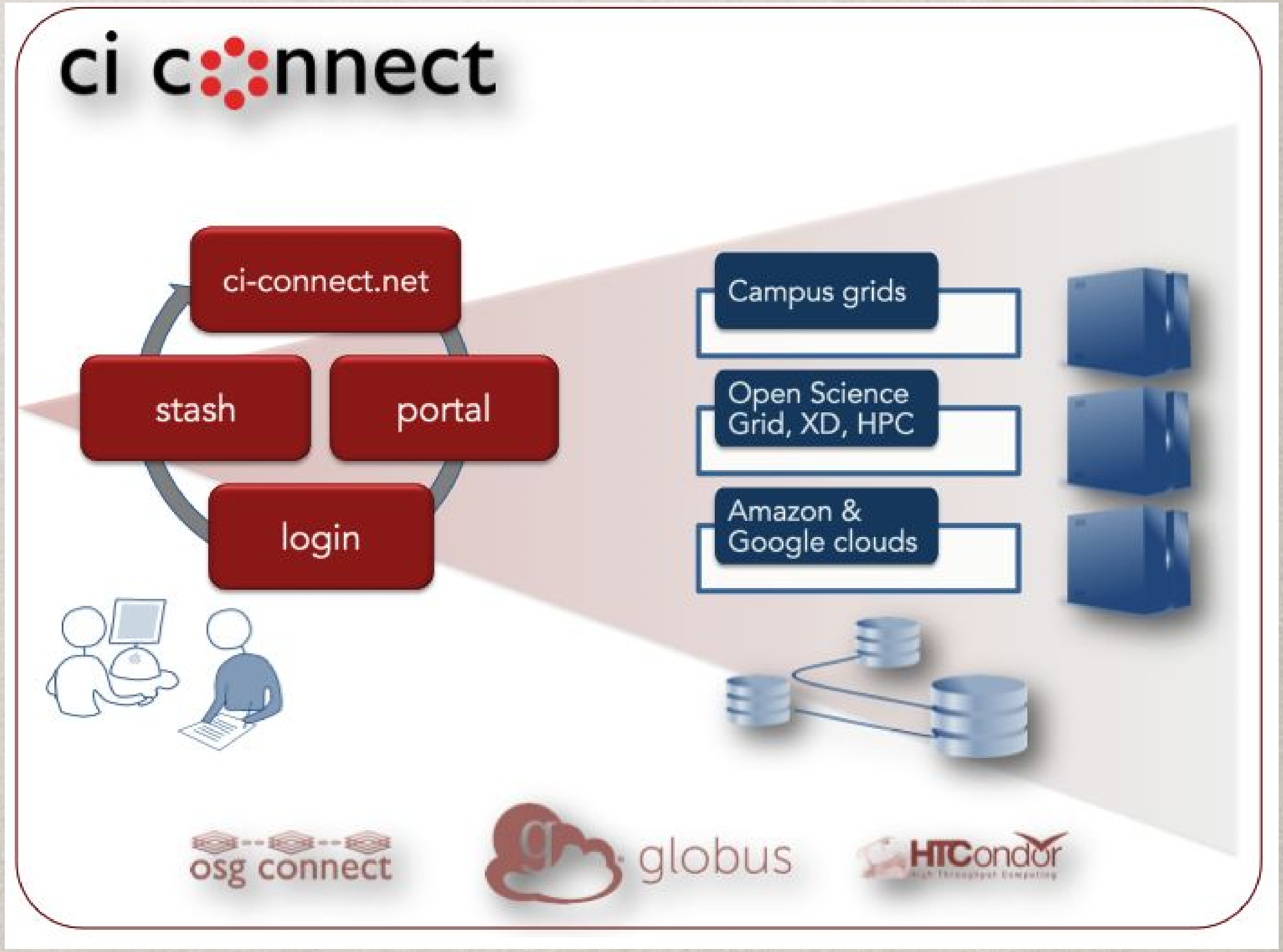
## How Do We Get There?

- Guiding principles:
  - Concentration of expertise for service deployment and operations
  - Minimizing footprint at resource endpoints $\Rightarrow$ maximizing reach for science community
- Necessary components (Existing example)
  - Virtual cluster service (CI Connect)
  - Provisioning Factories (AutoPyFactory)
  - Self-Configuring Workloads (Lobster based on CCTools)
  - Data Access and Caching (xrootd/AAA/FAX)
- Build up on existing technology to make progress as quickly as possible.

## Building Blocks: CI-Connect

- OSG as a service
- Used in OSG, ATLAS and CMS Connect (see K. Hurtado's talk from earlier today)
- Key features
  - Login host: identity management via InCommon/Globus
  - Group remote resources into a common HTCondor queue
  - Stash: network accessible storage

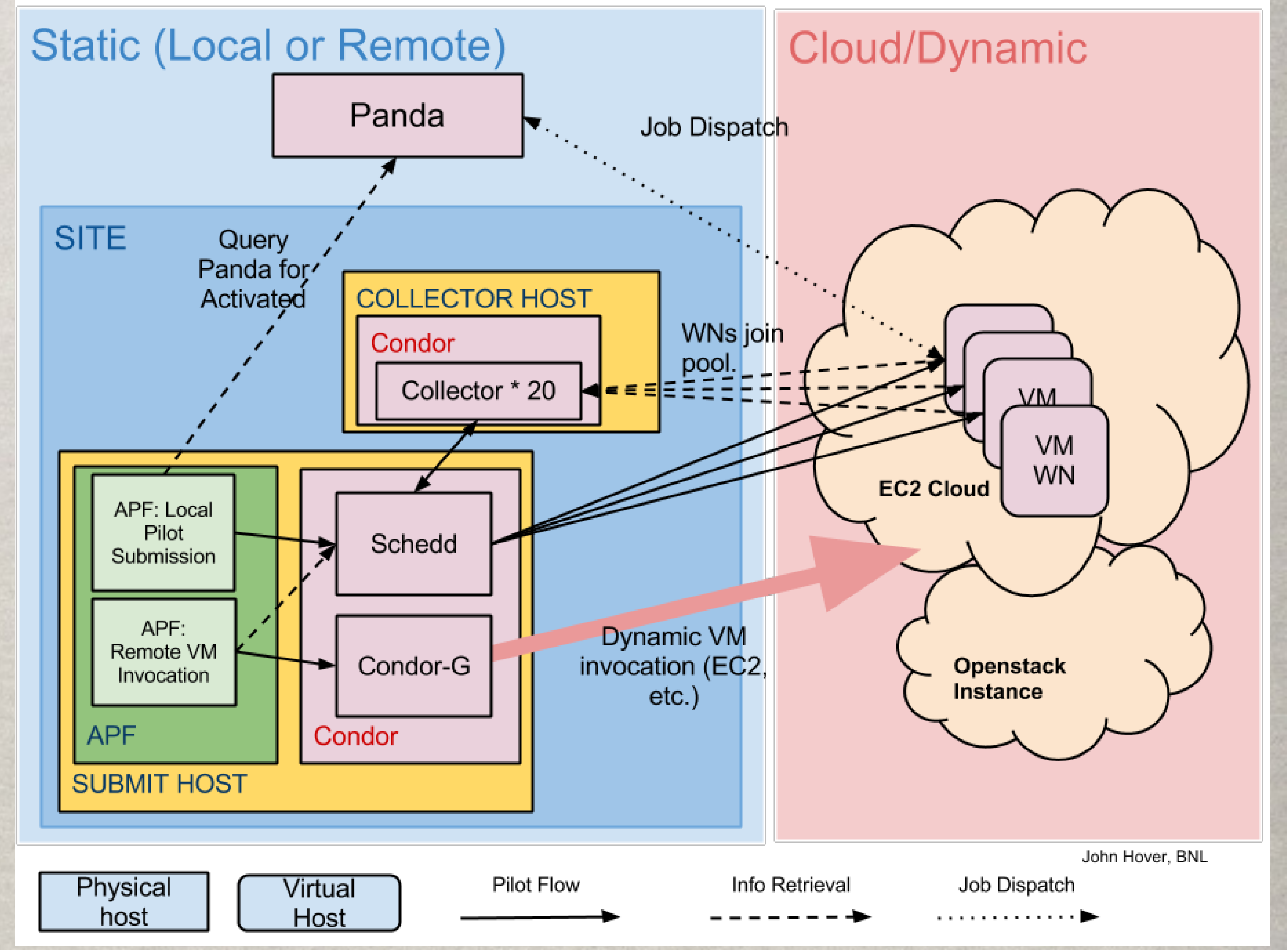
CI Connect generic pattern, implemented in OSG, ATLAS, and CMS Connect services



## Building Blocks: AutoPyFactory

- Part of ATLAS workload management system (PanDA), but designed to be general
- Plugin design enables adaptation to different submission infrastructures (batch, grid, cloud) and resource targets
- Handles VM lifecycle management for cloud resources, prioritization among different resources
- See talk by J. Hover from earlier today on use with AWS

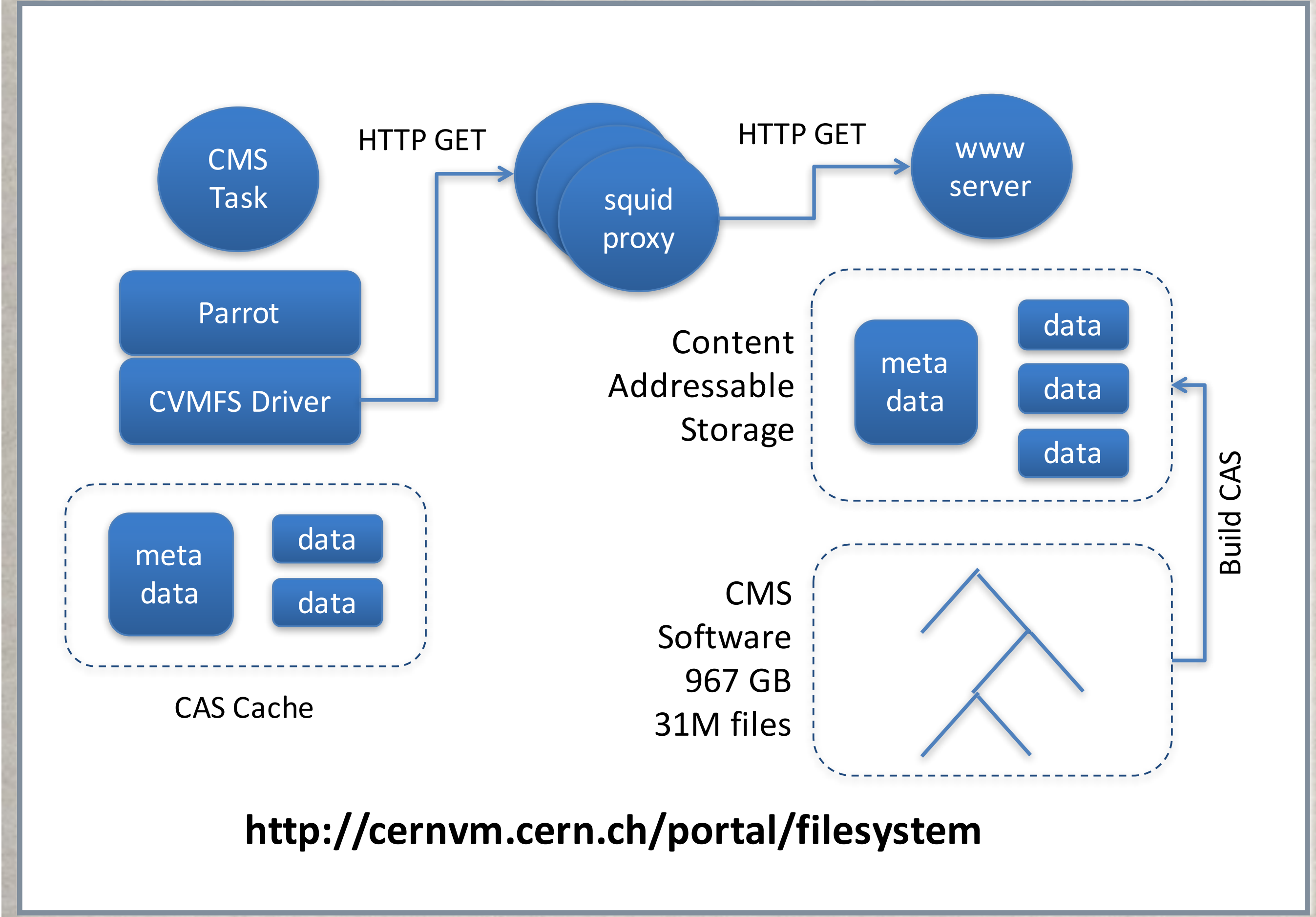
AutoPyFactory: Pilot submission framework designed to be reliable, scalable, and flexible



# Building Blocks: CCTools

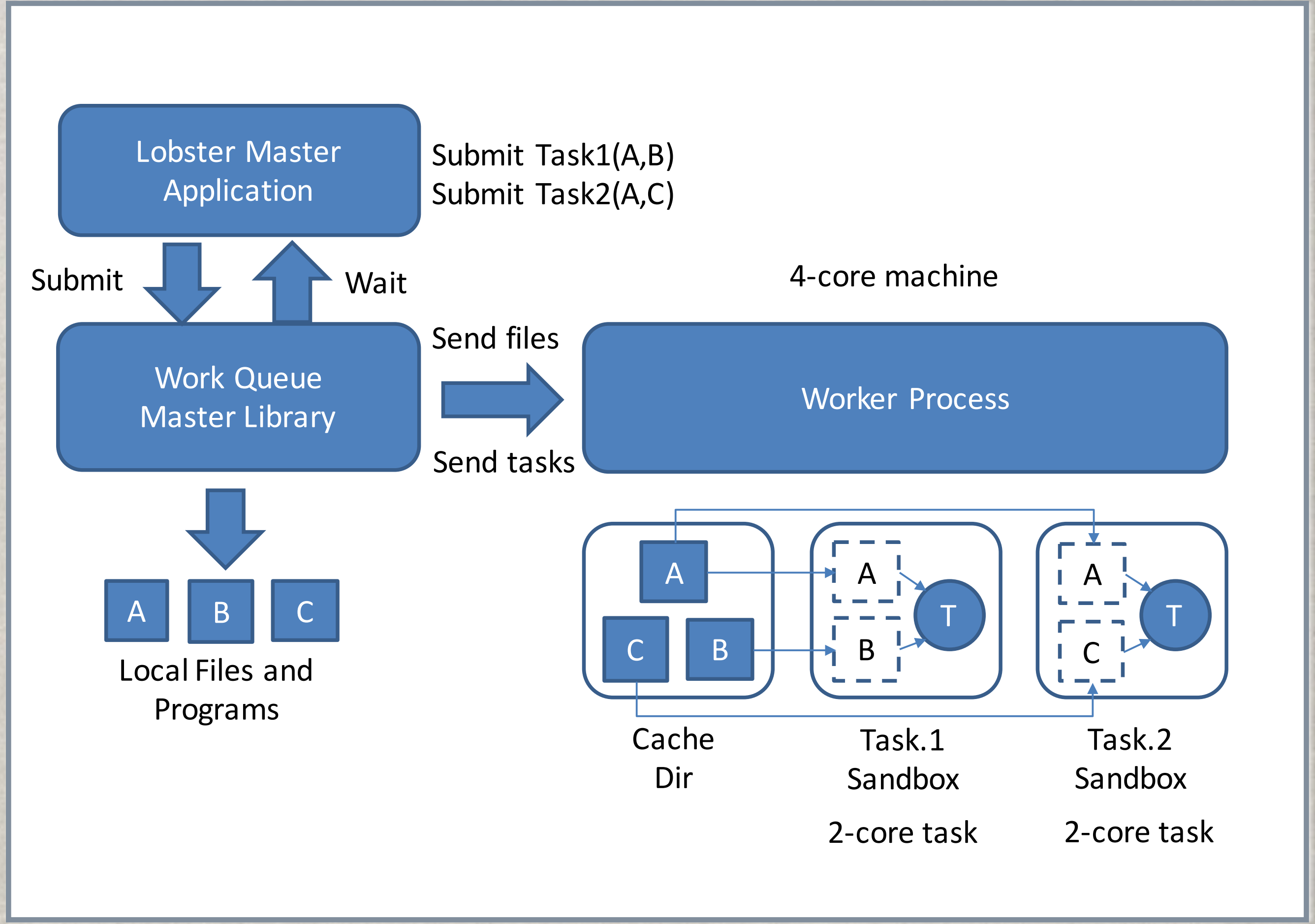
- The Cooperative Computing Lab (CCL) provides open source tools for leveraging distributed computing systems with emphasis on operating with only user-level permissions

Parrot+CVMFS provides access to software over the network (see talk by B. Tovar later today)



**http://cernvm.cern.ch/portal/filesystem**

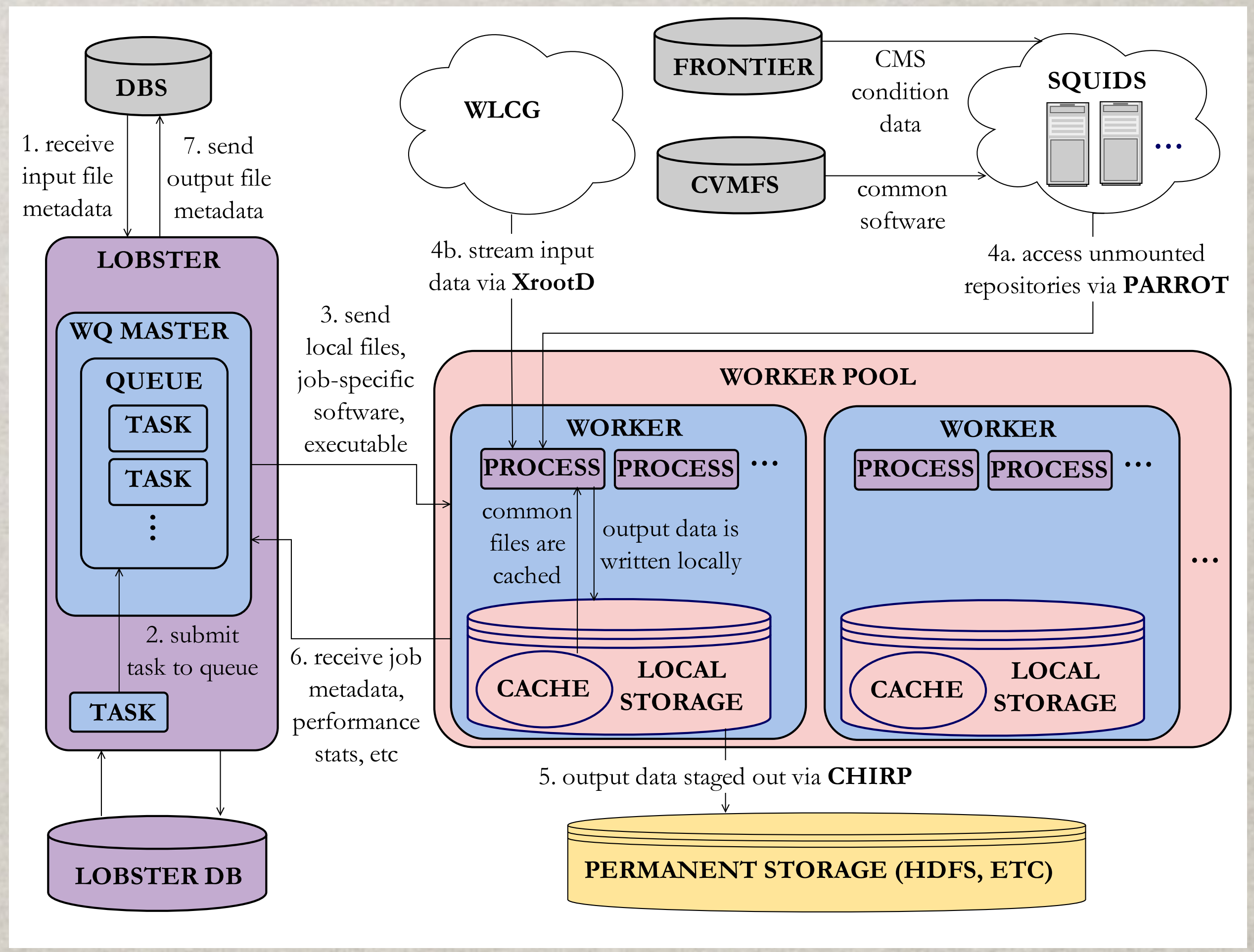
Work Queue provides task management with local caching on worker resources



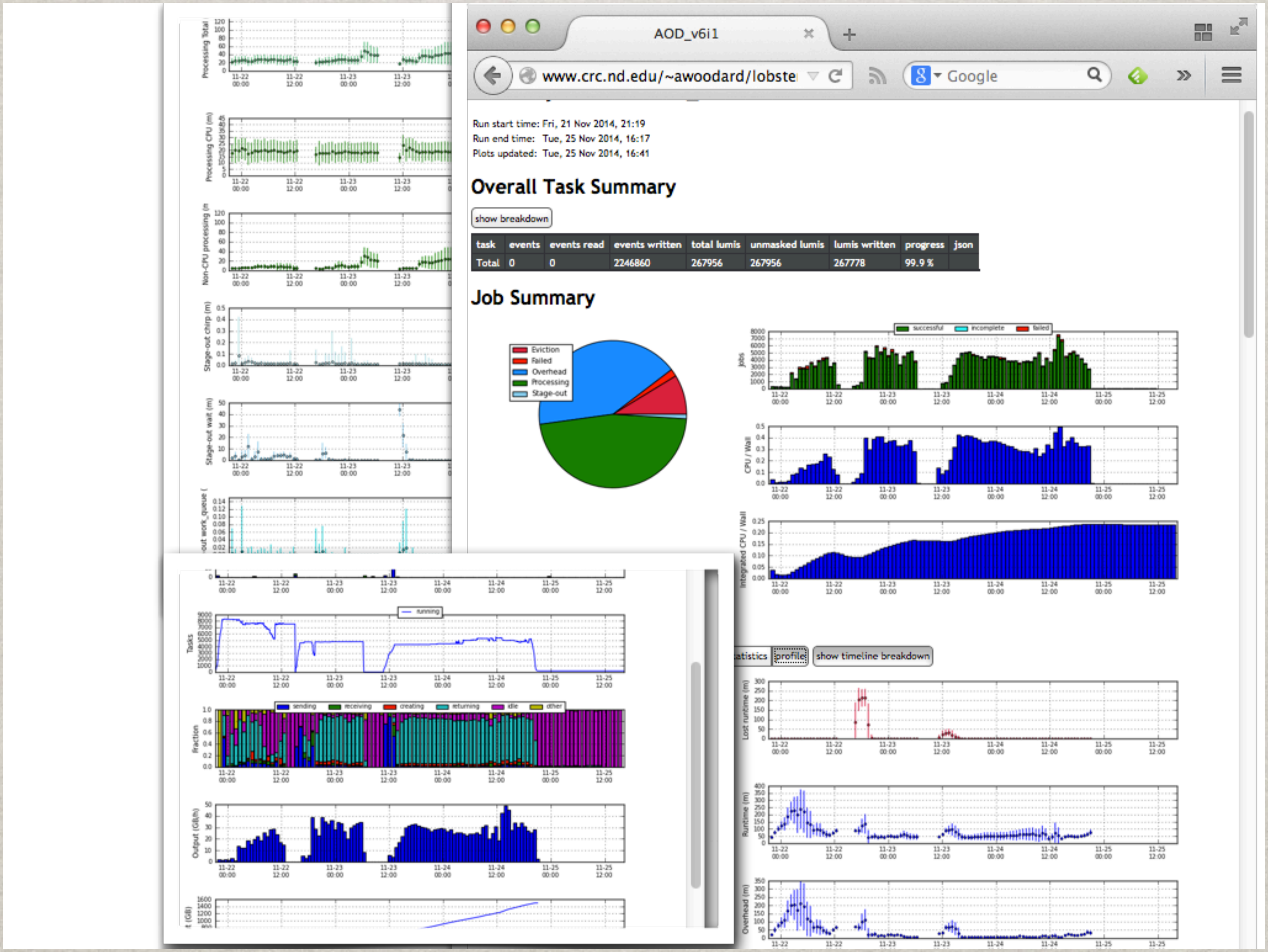
# Building Blocks: Lobster

- Goal: Run CMS analysis jobs on non-dedicated, opportunistic resources (at ND), including dealing well with eviction
- Designed to be deployed using only user-privileged processes + some essential services (squid, CVMFS)
- Manages workloads, but leaves resource management to others (i.e. user submitting workers to local queue directly)
- Monitoring is **key**: monitor as many aspects of task execution as possible and report to user
- Has been used to run successfully on up to 26k CPU cores of opportunistic resources.

**L**arge-scale **O**pportunistic **B**atch **S**ubmission **T**oolkit for **E**xploiting **R**esources (Lobster) builds on CCTools, OSG, xrootd, etc., to enable CMS analysis jobs to run on non-dedicated resources at scale.


DBS
1. receive input file metadata
7. send output file metadata
WLCG
FRONTIER
CMS condition data
SQUIDS
CVMFS
common software
4b. stream input data via XrootD
4a. access unmounted repositories via PARROT
LOBSTER
WQ MASTER
QUEUE
TASK
TASK
3. send local files, job-specific software, executable
WORKER POOL
WORKER
PROCESS
PROCESS
common files are cached
output data is written locally
2. submit task to queue
6. receive job metadata, performance stats, etc
CACHE
LOCAL STORAGE
TASK
5. output data staged out via CHIRP
LOBSTER DB
PERMANENT STORAGE (HDFS, ETC)

# Example Monitoring Plots



Run start time: Fri, 21 Nov 2014, 21:19
Run end time: Tue, 25 Nov 2014, 16:17
Plots updated: Tue, 25 Nov 2014, 16:41

Overall Task Summary

| task | events | events read | events written | total lumis | unmasked lumis | lumis written | progress | json |
|---|---|---|---|---|---|---|---|---|
| Total | 0 | 0 | 2246860 | 267956 | 267956 | 267778 | 99.9 % | |

Job Summary

## OSG Oasis Essential!

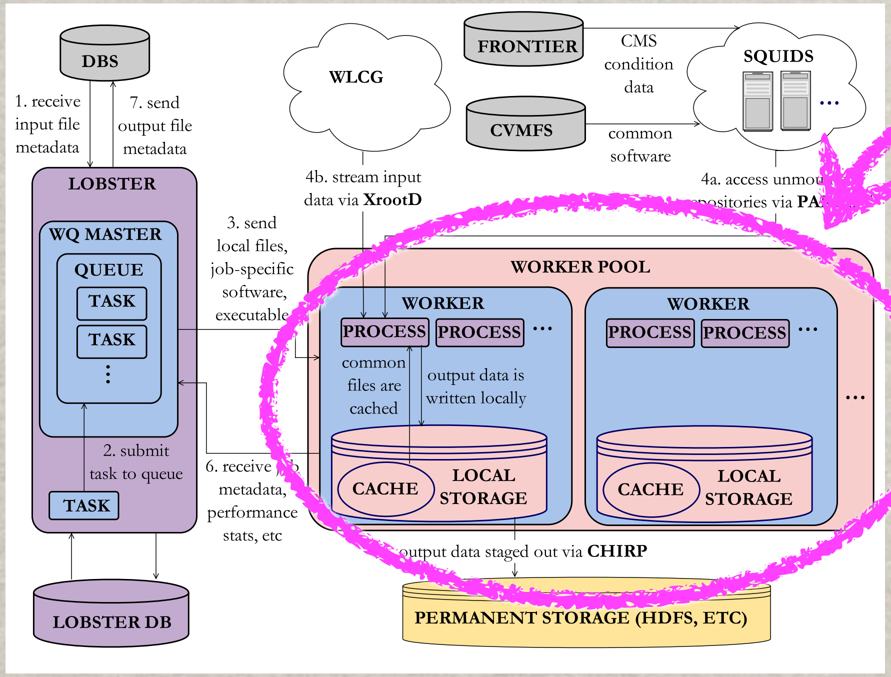
- Lobster relies on Oasis CVMFS repository to provide access to OSG software stack
- OSG software ***IS NOT*** installed locally on worker nodes
- Specifically using
  - osg-software/osg-wn-client/
  - mis/certificates
- Without this resource, could not access the large (up to 25k CPU core) opportunistic resources at ND
- **Thank you OSG for providing this service!**

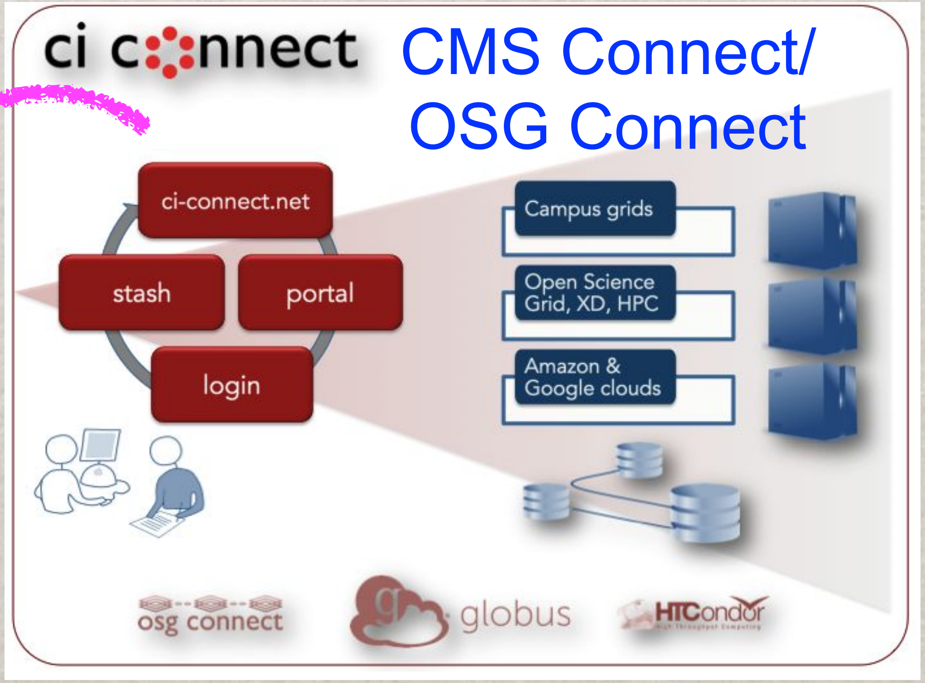
## Assembling Pieces

- Clearly many ways the above pieces could be combined
  - CI Connect + AutoPyFactory = Virtual clusters that access a broader range of different resources
  - CI Connect + Lobster = Running at scale on virtual cluster made of non-dedicated, opportunistic resources

# Assembling Pieces

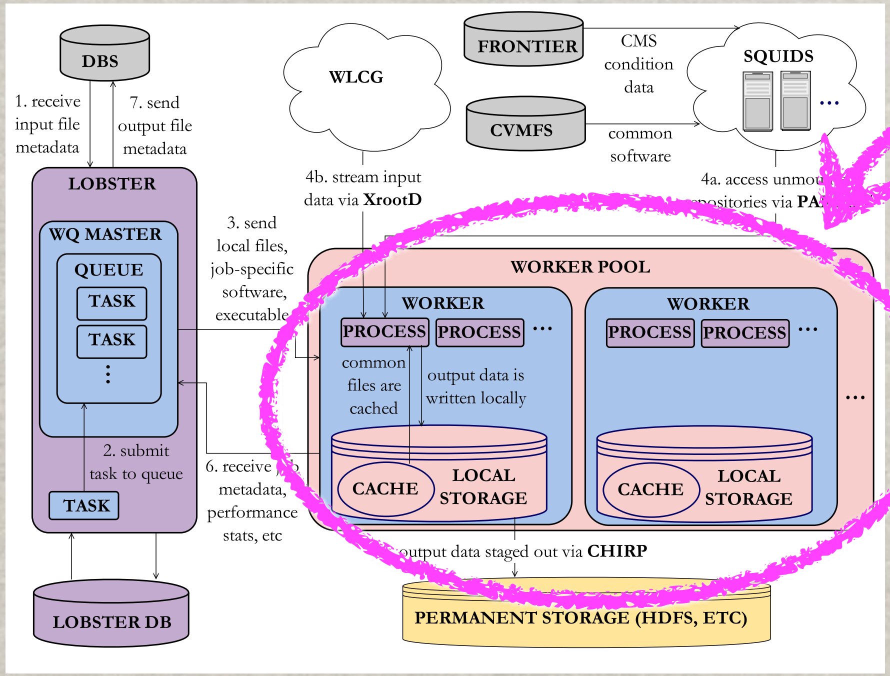
- Clearly many ways the above pieces could be combined
  - CI Connect + AutoPyFactory = Virtual clusters that access a broader range of different resources
  - **CI Connect + Lobster = Running at scale on virtual cluster made of non-dedicated, opportunistic resources**

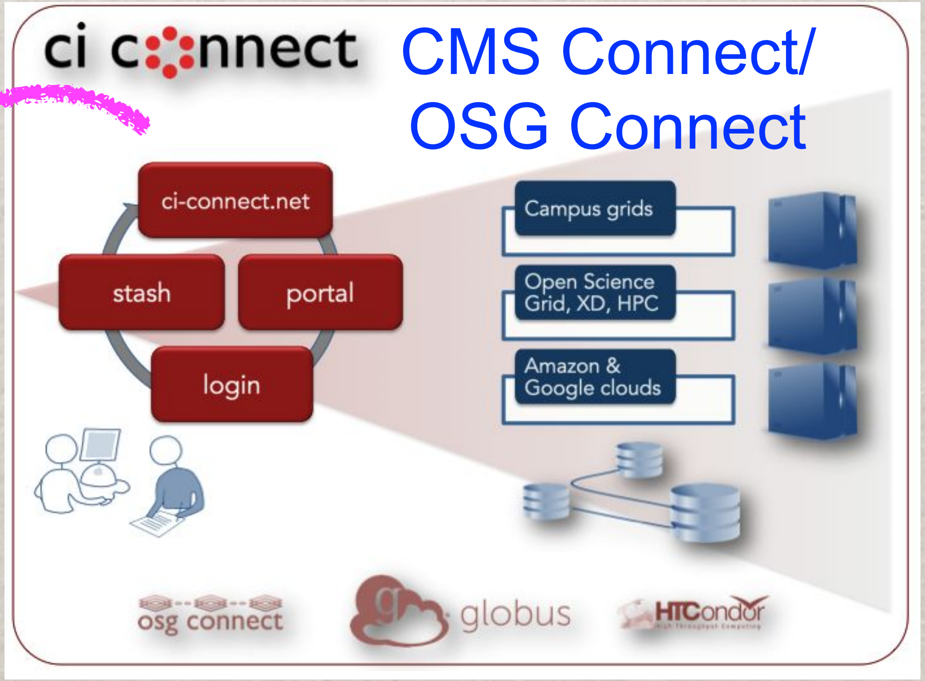

DBS
1. receive input file metadata
7. send output file metadata
LOBSTER
WQ MASTER
QUEUE
TASK
TASK
2. submit task to queue
TASK
LOBSTER DB
3. send local files, job-specific software, executable
6. receive job metadata, performance stats, etc
WLCG
4b. stream input data via XrootD
FRONTIER
CMS condition data
CVMFS
common software
SQUIDS
4a. access unmounted repositories via PARROT
WORKER POOL
WORKER
PROCESS
PROCESS
common files are cached
output data is written locally
CACHE
LOCAL STORAGE
output data staged out via CHIRP
PERMANENT STORAGE (HDFS, ETC)



# Assembling Pieces

- Clearly many ways the above pieces could be combined
  - CI Connect + AutoPyFactory = Virtual clusters that access a broader range of different resources
  - **CI Connect + Lobster = Running at scale on virtual cluster made of non-dedicated, opportunistic resources**


DBS
1. receive input file metadata
7. send output file metadata
LOBSTER
WQ MASTER
QUEUE
TASK
TASK
2. submit task to queue
TASK
LOBSTER DB
WLCG
FRONTIER
CMS condition data
SQUIDS
CVMFS
common software
4b. stream input data via XrootD
4a. access unmounted repositories via PARROT
3. send local files, job-specific software, executable
6. receive job metadata, performance stats, etc
WORKER POOL
WORKER
PROCESS
PROCESS
common files are cached
output data is written locally
CACHE
LOCAL STORAGE
WORKER
PROCESS
PROCESS
CACHE
LOCAL STORAGE
5. output data staged out via CHIRP
PERMANENT STORAGE (HDFS, ETC)



Worked!

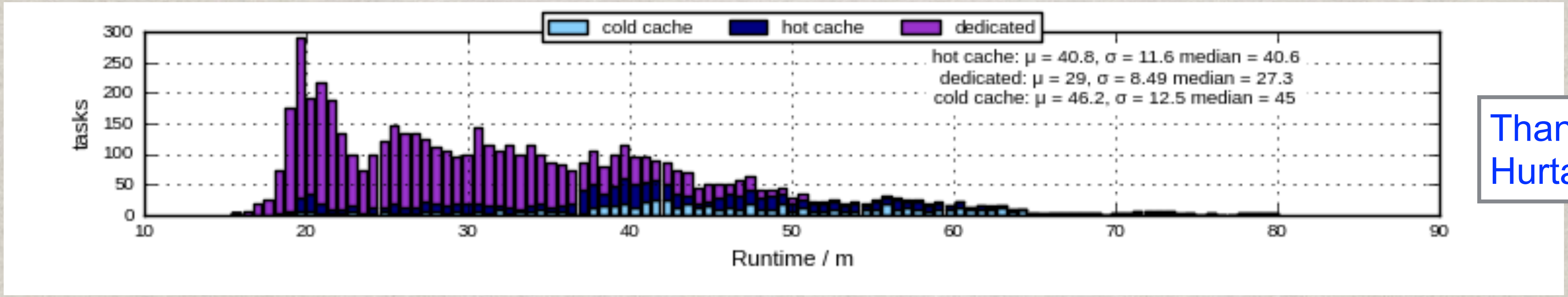
K. Lannon

UNIVERSITY OF NOTRE DAME

# Lobster+CI-Connect

- Run successfully using both OSG and CMS Connect services

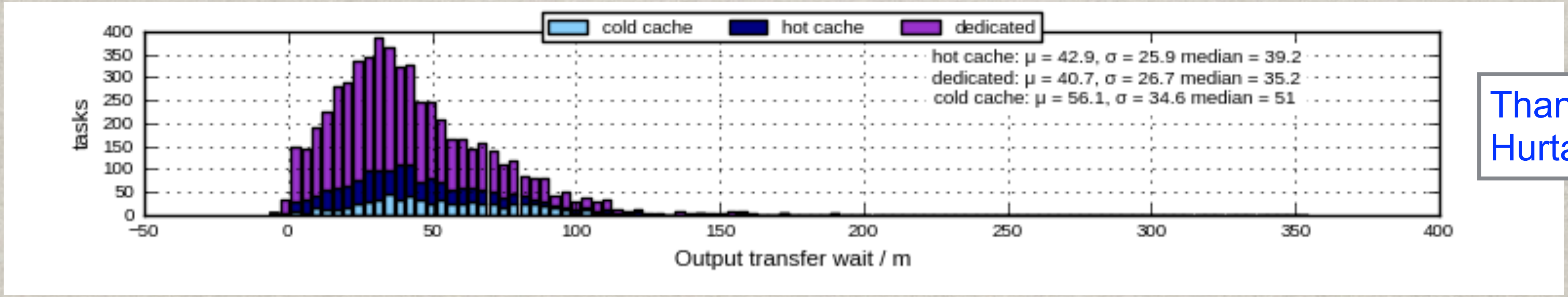
Example from OSG Connect: Many jobs running successfully!



Thanks to K. Hurtado for plots!

**Dedicated** = jobs run on node providing CVMFS directly; **Cold Cache** = jobs run on node not providing CVMFS, used Parrot to access; **Hot Cache** = jobs ran on worker that had already populated CVMFS as part of previous job

Not everything was smooth: Jobs *waiting* an average of 40-50 min to transfer (small) output.



Thanks to K. Hurtado for plots!

Tracked down to issue with one cluster failing jobs at high rate, thereby monopolizing the Work Queue master's bandwidth. Solutions available in Work Queue, but work best if we can deploy specialized jobs (foreman) at each site where workers run. Requires integration between CI Connect and Work Queue.

## How Could This Work Better?

- In principle all pieces can be used together today as is
- However, significant gains to be realized if pieces are integrated (i.e. taught to communicate, etc.)
- Examples
  - To do intelligent caching of files, workload management tool and virtual cluster service need to communicate on resource topology
  - To avoid overwhelming network capacity of various resources, workload management and virtual cluster need to combine network monitoring on task and system level
  - To adapt running parameters to resource conditions (e.g. eviction rate, etc.), virtual cluster layer can provide historical statistics for resources to workload management layer
- Goal of VC3 to work out how to better integrate these pieces

## Summary and Outlook

- Virtual clusters provide users with a convenient interface to distributed resources, but at the cost of higher user burden for configuring workloads to different resources
- Better workload management tools can overcome these limitations and make it easier for users to deploy complex applications
- Integrating these tools makes possible dynamic deployment of virtual clusters utilizing a broad class of dedicated and non-dedicated, opportunistic resources
- First steps show the potential of this approach, but lots of work ahead!
  - Short term: service and tools development work
  - Longer term: deploy test cases and engage user community