OSG-CAF

A single point of submission for CDF to the Open Science Grid

Matthew Norman, Shih-Chieh Hsu, Elliot Lipeles, Mark Neubauer, Subir Sarkar, Igor Sfiligoi, Frank Würthwein,
Presented by Matthew Norman
University of California, San Diego

Introduction to OSG-CAF

What is the OSG-CAF?

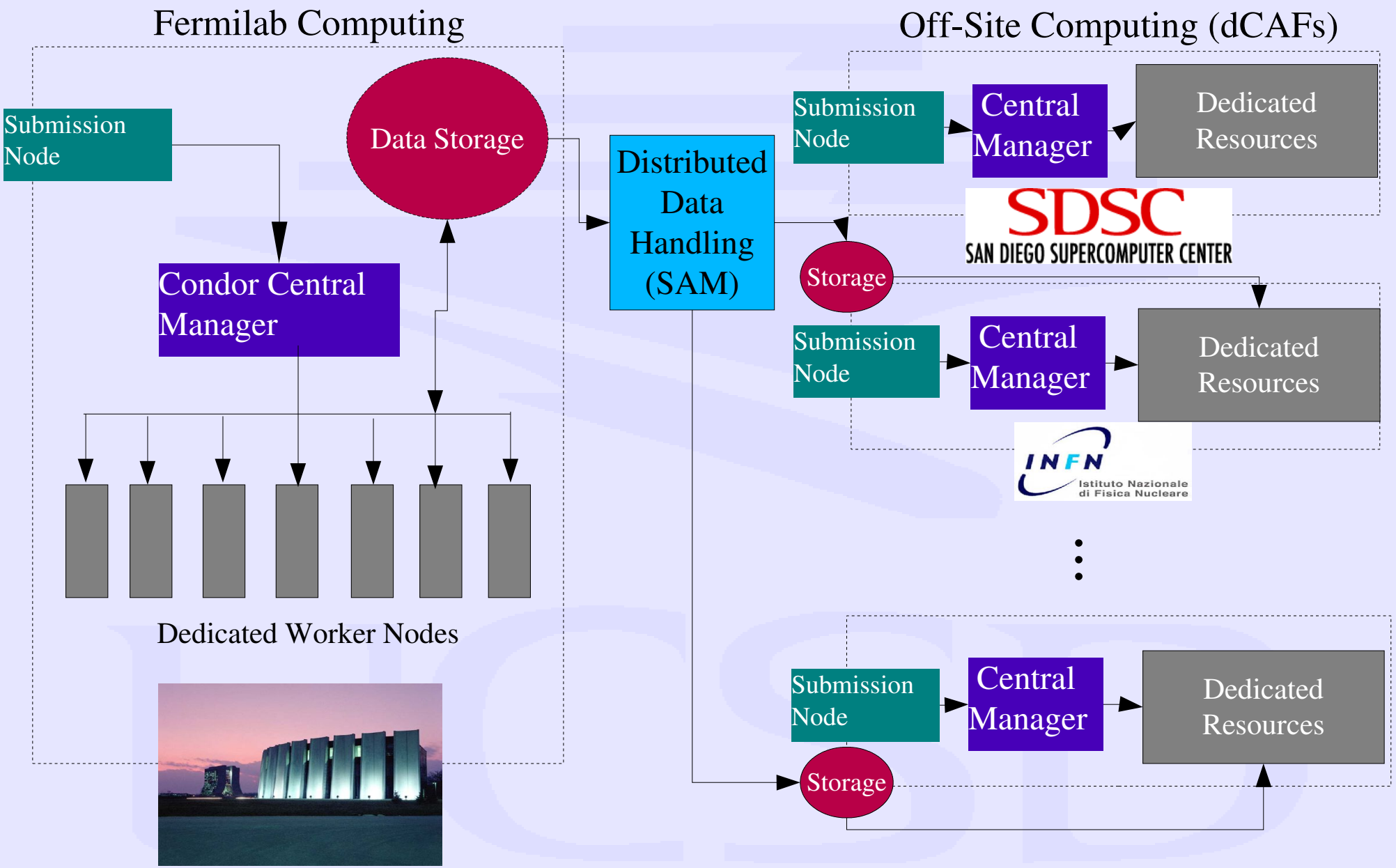
- A logical expansion of the CAF (CDF Analysis Farms) system.
- A way to allow CDF users convenient access to OSG resources.
- A production facility for CDF Monte Carlo.

Why do we need it?

- To handle increased computing demands due to higher luminosity.
- To deal with our limited ability to host more dedicated resources.
- To gain experience in the transition to the grid paradigm

CDF's portal to the OSG

Original Distributed Computing Plan



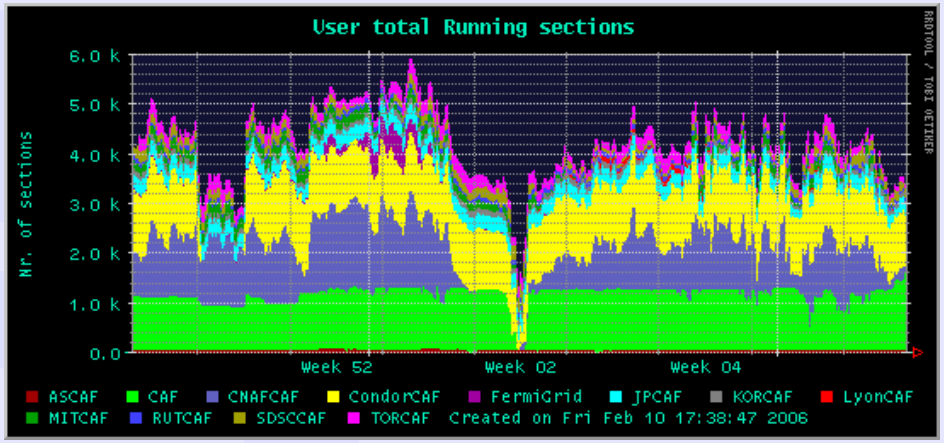
Changes to the Plan

The CAF system has performed well:

•dCAFs have expanded: 10 sites on 3 continents, and still growing!

• Computing power at Fermilab has increased dramatically.

However, we've hit the limit.

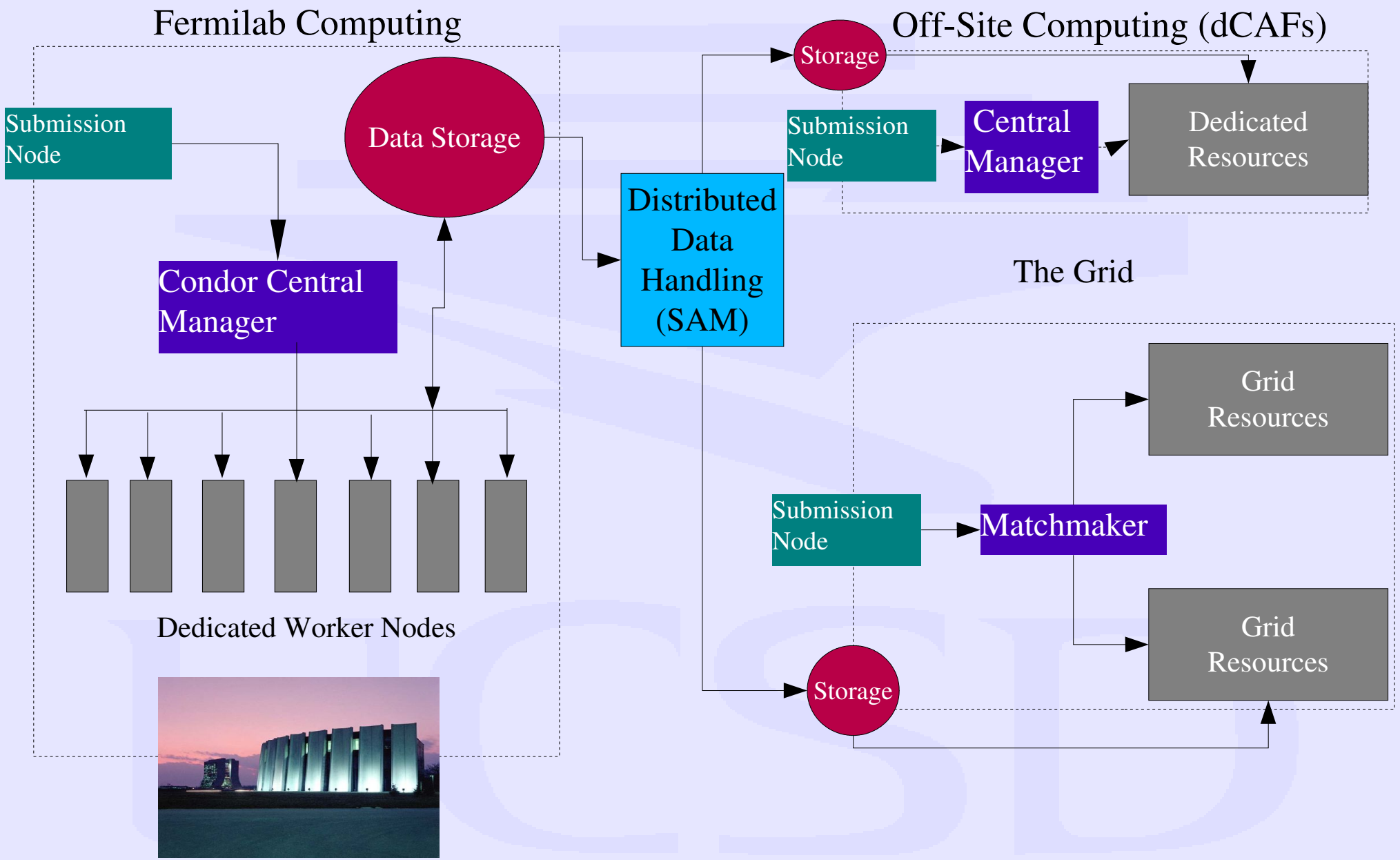


• Increased luminosity has increased computing demand as fast as capability has grown.

• Acquiring and maintaining dedicated resources is much more difficult than investing in Grid resources.

CDF must go to the Grid!

Updated Distributed Computing Plan



The Grid Blueprint for CDF

- Access to the Grid must use the same user authentication scheme as all other CDF interactions (Fermilab kerberos)

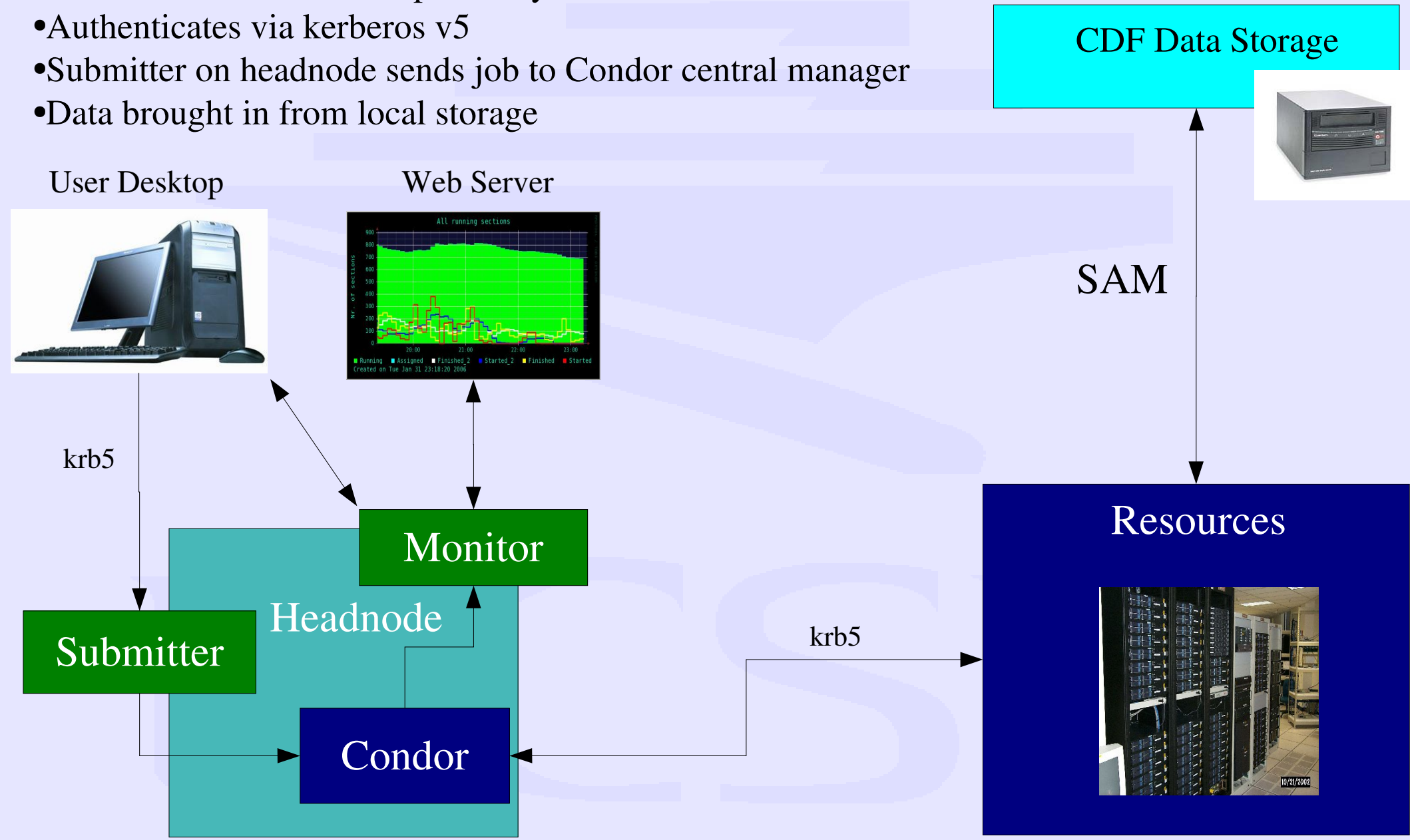
- Running outside of Fermilab must not prevent the user from accessing some of the data at FNAL.

- All actual grid mechanics should be taken care of “under the hood”

- Submission and access must be centralized for convenience.

Standard CAF

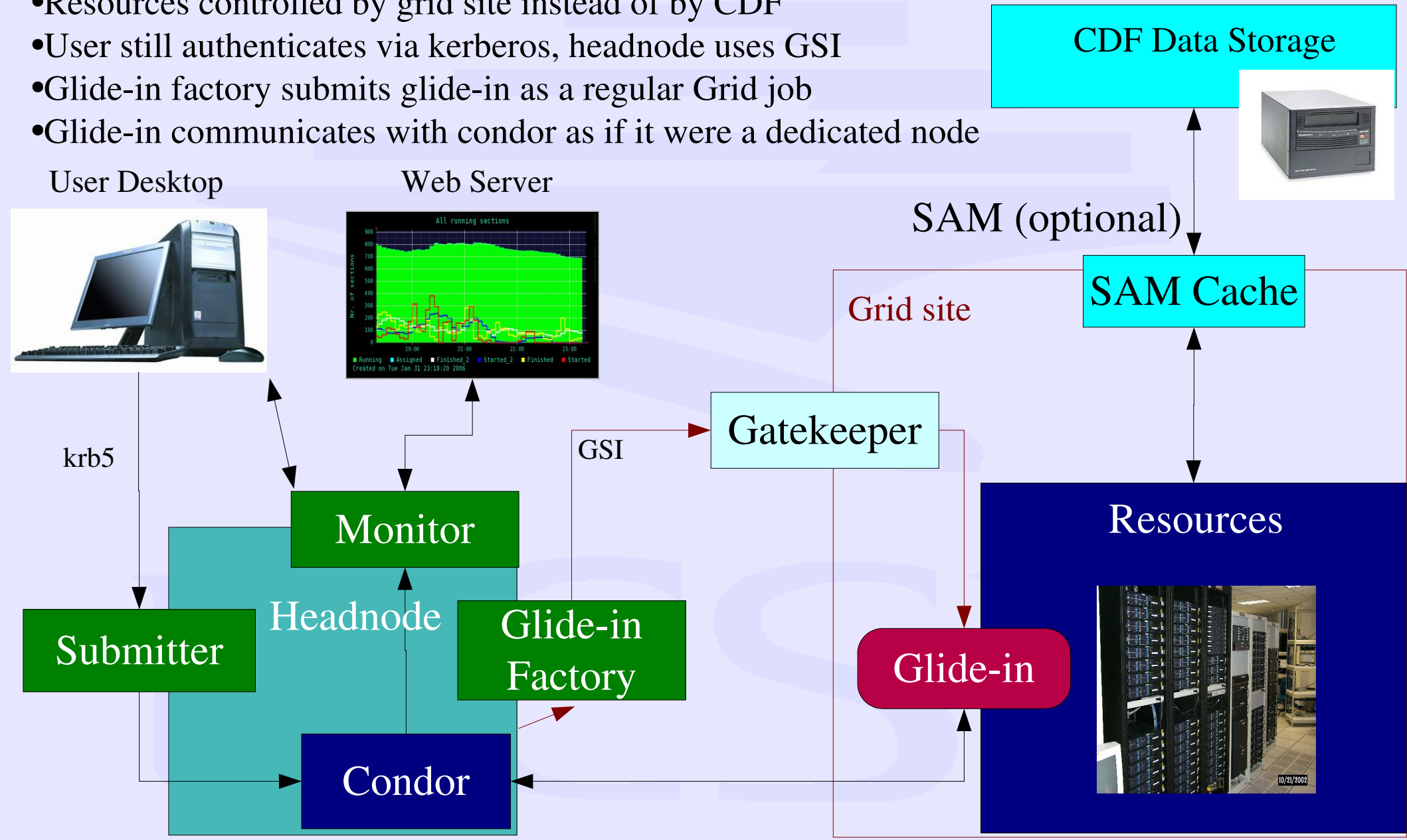
- User submits from desktop directly to submitter
- Authenticates via kerberos v5
- Submitter on headnode sends job to Condor central manager
- Data brought in from local storage



GlideCAF

See GlideCAF talk for more information

- Resources controlled by grid site instead of by CDF
- User still authenticates via kerberos, headnode uses GSI
- Glide-in factory submits glide-in as a regular Grid job
- Glide-in communicates with condor as if it were a dedicated node



Difficulties with Global GlideCAF

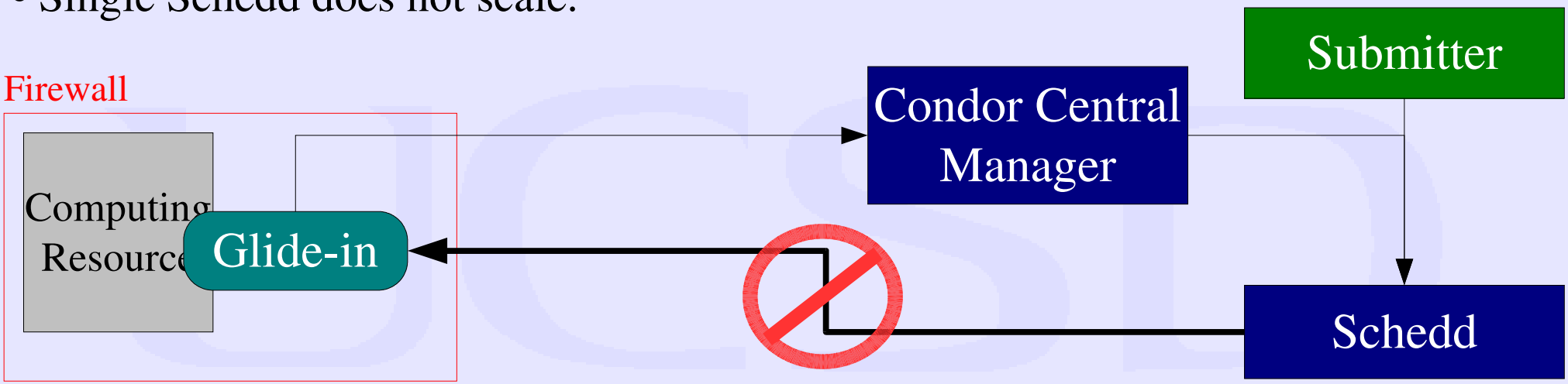
There are several problems with the global deployment of the GlideCAF over the WAN:

- Cannot operate if firewall does not allow the Condor Central Manger to contact the Glide-in! Most Grid sites have firewalls or NAT which prevents incoming connections.

- Condor depends on UDP packets, which frequently get lost over a WAN.

- Distributing CDF-specific software to worker nodes not trivial.

- Single Schedd does not scale.



Firewall traversal

Our Solution: GCB (Generic Connection Brokering)

GCB allows applications to communicate freely over firewall boundaries

1) At submission, the glide-in opens a persistent connection with the GCB Server. The system notes that it is using GCB.

Condor Central Manager

GCB Server

Persistent

Firewall

Glide-in

Computing Resource

2) When a job is assigned, the Schedd asks the GCB Server to open a connection.

Submitter

Schedd

3) The GCB Server tells the glide-in to call-out to the Schedd

4) The Glide-in then calls out to the Schedd, opening the necessary communications.

GCB Deployment

Preliminary tests allowed us to launch a glide-in through the OSG Gatekeeper in San Diego and communicate with it, enabling us to launch simple batch jobs.

GCB has also communicated with glide-ins at:

- University of Wisconsin
- FermiGrid
- University of Iowa
- Academia Sinica (Taiwan)
- Brookhaven National Lab

This list is being expanded now.

This method requires the Glide-in to be able to call out. GCB can be modified to operate on the network boundary on both sides for isolated sites.

Local Bandwidth Considerations

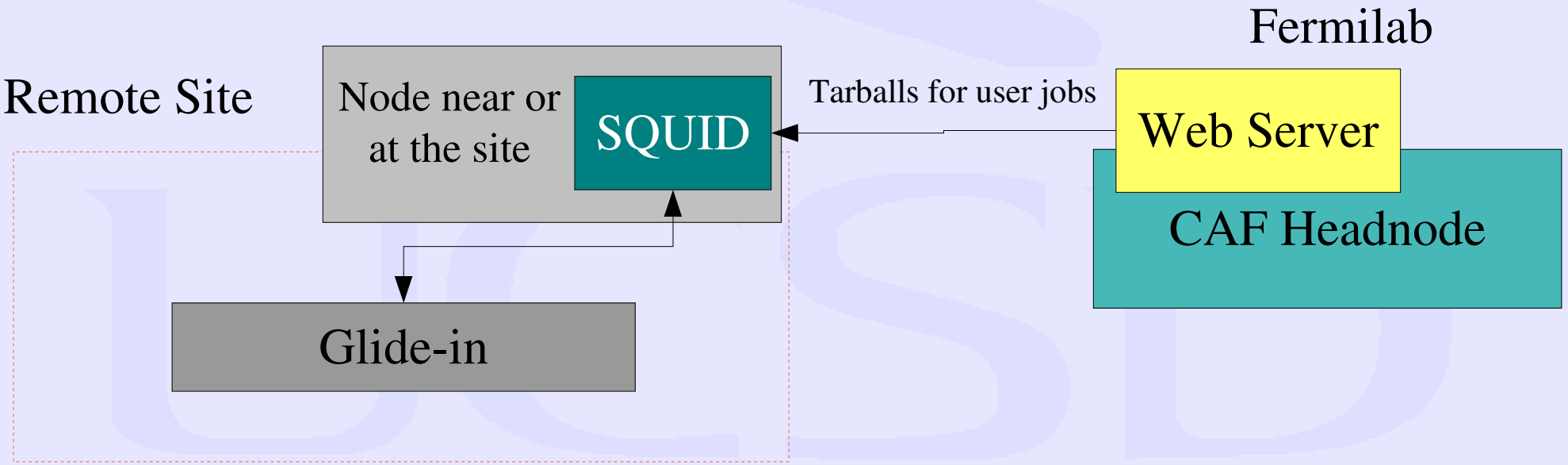
One difficulty with running a large CAF (over two or three thousand VMs) is the load on local bandwidth and the headnode memory caused by user tarballs.

- For CDF analysis, the size of the job tarball can be formidable, tarballs run up to the allowed 250MB limit.
- Most jobs have multiple sections, all using the same tarball.
- Continually sending these tarballs drains the resources of the headnode and seriously taxes the wide-area bandwidth.

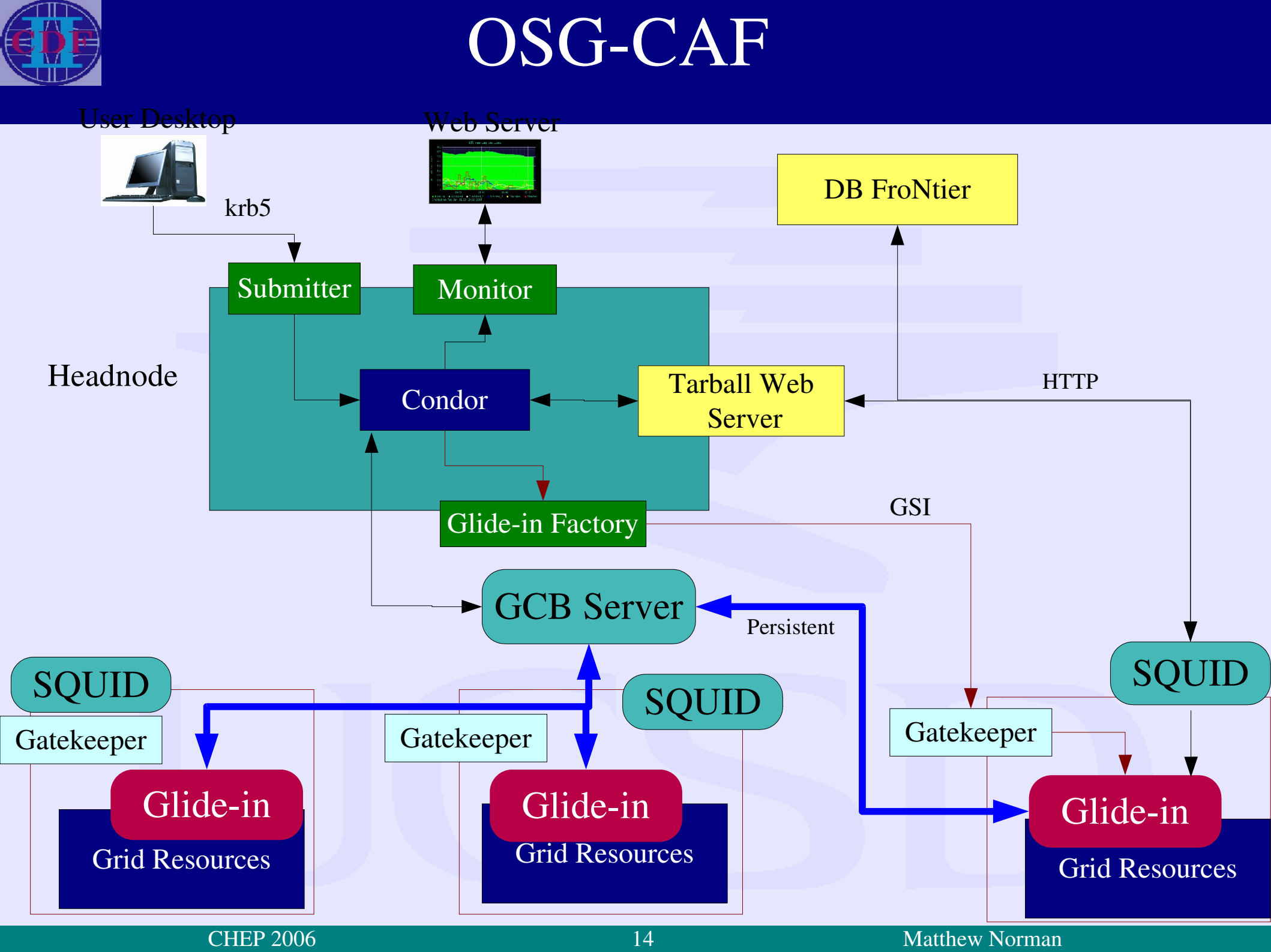
Our Solution? Caching and transfer via HTTP via SQUID.

Tarball Distributions

- Tarballs can be distributed to the world via a web server on the headnode
- To allow worker nodes to access tarballs rapidly, we want local caching
- The intention is to install SQUID on sites we intend to use.
- SQUID is a widely used caching and proxy application, proposed for use at all OSG sites.
- SQUID can also be used in the DB FroNtier service.
- A single user tarball can be cached, and sent from the SQUID multiple times
- Greatly reduces headnode load and WAN traffic.



OSG-CAF


User Desktop
Web Server
krb5
DB FroNtier
Submitter
Monitor
Headnode
Condor
Tarball Web Server
HTTP
Glide-in Factory
GSI
GCB Server
Persistent
SQUID
SQUID
SQUID
Gatekeeper
Gatekeeper
Gatekeeper
Glide-in
Grid Resources
Glide-in
Grid Resources
Glide-in
Grid Resources

Software Distribution via Parrot

Most CDF jobs require CDF software distribution.
How to deliver it?

Option1:
Installing on each and every grid site.
Complex and error prone.

Option2:
Use a shared file system (AFS).
Not all sites support it.

Our solution:
Deliver via HTTP from central server. Use **Parrot** for file access.
Jobs see software distribution as being local,
Parrot converts POSIX I/O calls to appropriate remote commands.
Together with Squid, ideal for our needs.

See the Parrot poster presentation for more details

Scalability Issues

User Desktop

Central Manager

Collector

Negotiator

Headnode

Monitor

Submitter

Condor Schedd

DAGman

Job Section

Job Section

Job Section

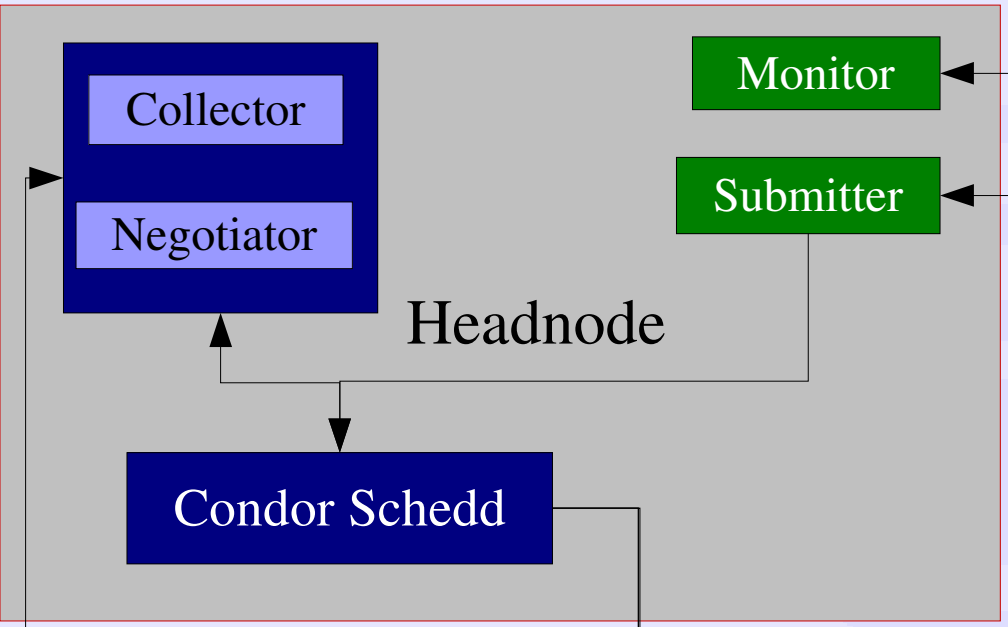
Job Section

Resources

The traditional CAF relies heavily upon a single Schedd, which tracks all the jobs submitted to that CAF. Tests reveal that a single Schedd is not scalable past 2,500 sections for hundreds of users.

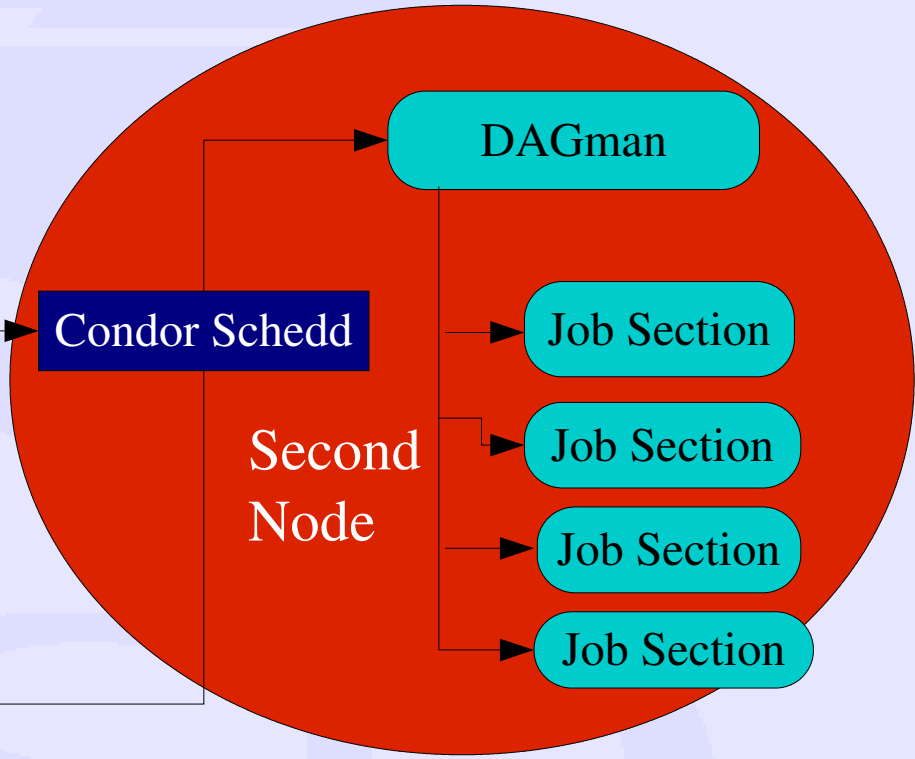
Multiple Schedds can be run on one node, but this puts a very high load on that machine.

Condor-C Scalability



User Desktop

Main Schedd allows us to manage jobs and assign consecutive JIDs.



Condor Schedd

Condor Schedd

Resources

In our Condor-C system, we reduce the load on the headnode by distributing jobs to secondary Schedds located on different nodes. The second node than handles all the CPU load of actually managing the job.

Summary

• CDF is beginning the shift from dedicated resources to the Grid.

• The CDF Analysis Farms portal has been integrated with several applications in order to meet CDF's Grid needs:

- Condor Glide-in and GCB, to gather Grid resources
- Condor-C to allow scaling to large numbers of users and VMs
- HTTP server and HTTP cache (SQUID) to reduce network traffic
- Parrot, to distribute software

•The OSG-CAF has undergone low-scale testing, and is moving towards production.

• CDF has developed, and is nearing deployment, of a single point of submission system for accessing the resources of the Grid.

Backup Slides

OSG-CAF Status

Currently Tested:

- GlideCAFs allow us to extend the CAF infrastructure to use Grid resources.
- GCB allows us to communicate with glide-ins over firewalls, and without using UDP over WAN.
- Transferring user tarballs via HTTP, which, combined with caching, reducing load on headnode and WAN.

Challenges for the Future:

- Load testing with GCB over multiple Grid sites.
- Security and load testing with Condor-C for scalability to full Grid.
- Deployment of Parrot for sites that require code from central repository.

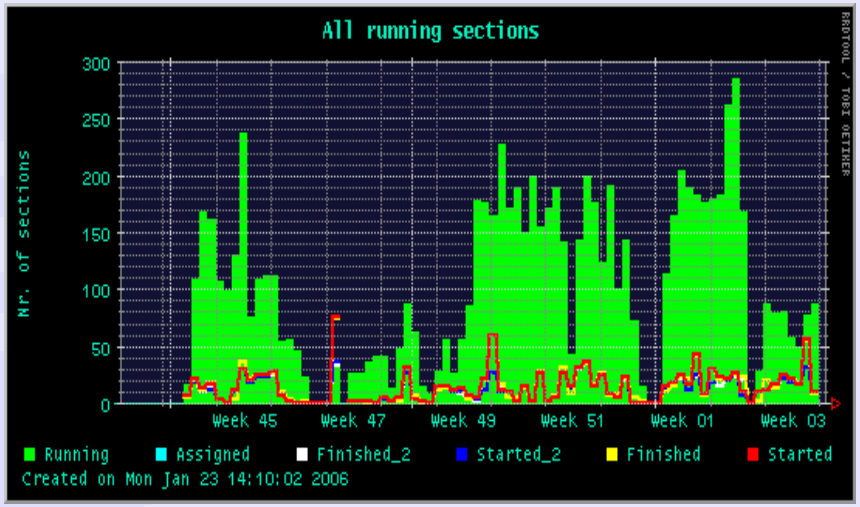
Current GlideCAF Usage

GlideCAF technology in use:

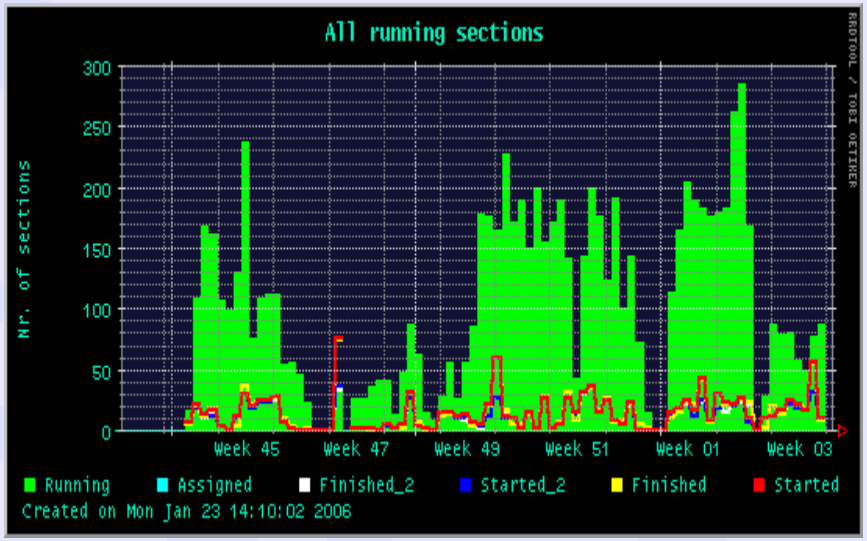
- CNAF
  - Runs at CNAF tier-1 center
  - Maximum of almost 2,000 active VMs

- FermiGlideCAF
  - Runs on GPFarms, etc.,
  - Maximum of over 400 active VMs

- SDSC
  - Runs on UCSD tier-2 OSG cluster
  - Maximum of over 200 active VMs

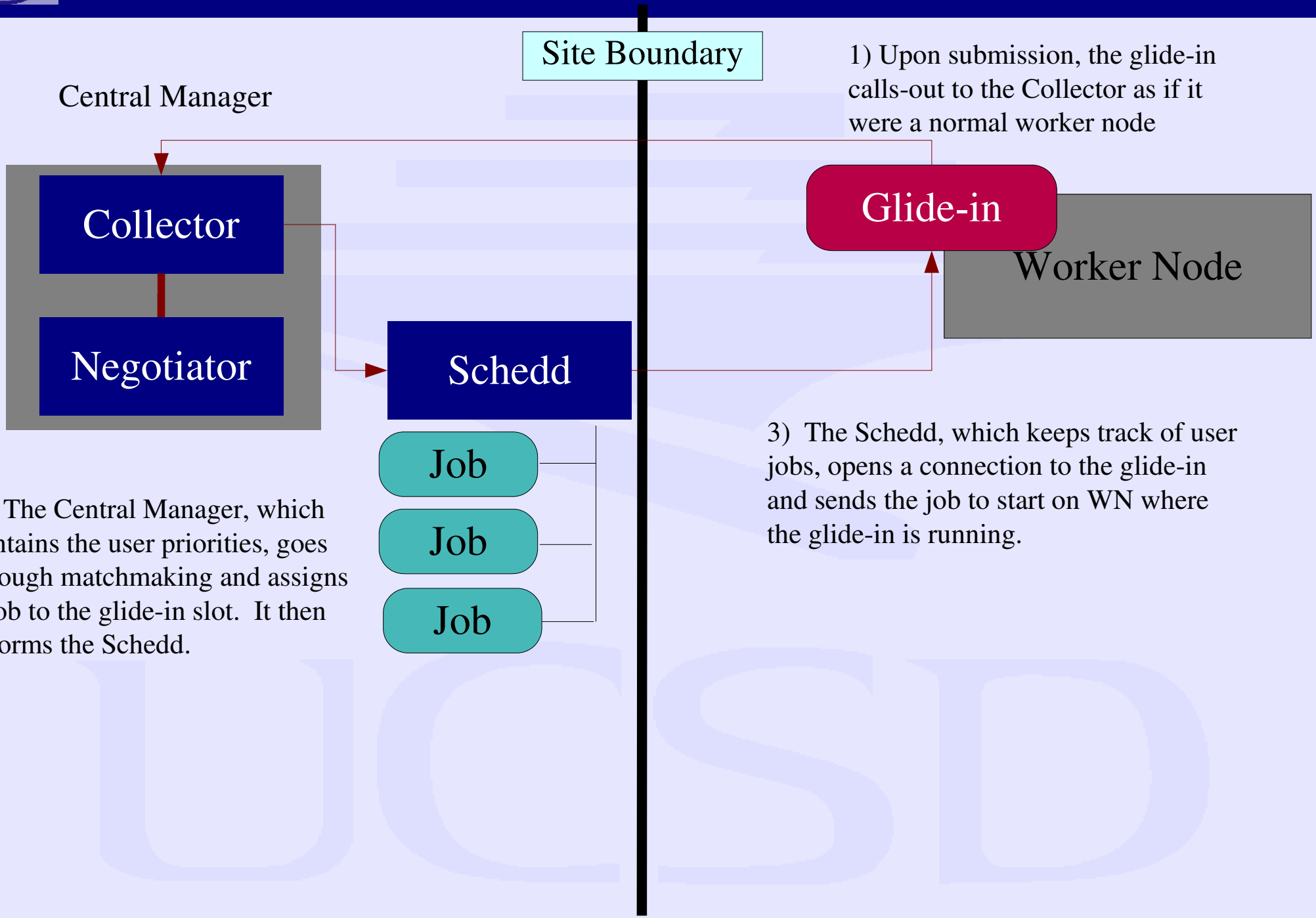


CNAF



SDSC

Glide-in Mechanics



1) Upon submission, the glide-in calls-out to the Collector as if it were a normal worker node

2) The Central Manager, which contains the user priorities, goes through matchmaking and assigns a job to the glide-in slot. It then informs the Schedd.

3) The Schedd, which keeps track of user jobs, opens a connection to the glide-in and sends the job to start on WN where the glide-in is running.

Condor-C Performance

Condor-C was tested as a preliminary to unifying control of computer resources at FNAL With four thousand VMs, three thousand of them were assigned in approximately three hours, an acceptable start up time. The entire farm could be filled up to capacity and could maintain itself for several days without appreciable strain on the headnode.

The system was tested repeatedly over a series of several weeks in order to iron out bugs. During this period, the Condor system retained stability, and handled large numbers of jobs with very low latency.

Some problems remain:

- Security Issues: Condor-C does not work well with Kerberos authentication. A fix is forthcoming.
- Operational Issues: Some commands do not work well when issued through the schedd hierarchy. The problem has been reported.
- Hardware Issues: Condor-C system has not been fully tested for response to hardware failure.

Data Distribution

Plan: Use SAM with SRM to distribute data.

- Current testing will be started with MC jobs-low data load via DB FroNtier.
- SAM-SRM interface will allow us to move data sets efficiently via the OSG's infrastructure without burdening the WAN.
- SAM Cache appears promising.

Transfer of Output

Currently, output is transferred back to FNAL (or wherever the user desires) using kerberized rcp.

Output will be transferred back to FNAL originally via SAM-upload. Eventually we intend to use SAM-SRM to make a series of short hops to get the output back to Fermilab.

End