Any Data, Any Time, Anywhere

Increasing data accessibility for HEP

Brian Bockelman, for the AAA team

NSF PHY-1104664

The Setup

- HEP has an incredibly large data management problem.
  - To make it plausible to handle the sheer size, we must make several processing passes over the data.
  - We must keep several replicas of the data to be able feed enough CPUs.
- This begets complex data management systems for tracking data provenance, location, and replication.

But Wait, There's More!

- While we always talk about physics data due to its volume, there are other data types:
  - *Conditions data*: Information (e.g., alignments) about the detecter at the time the data was taken. Changes as our understanding of the detector improves.
  - *Software*: The software used is complex and large, on the order of millions of lines of code. Multiple releases are available at any given time.

The Problem Statement

- I claim the HEP community has successfully demonstrated the ability to manage its data and processing challenges at its computing sites.
- We need to improve the ability to utilize outside resources - whether a physicist's laptop or an opportunistic grid site.

The Challenge: Make any internet-connected computer useful to the CMS experiment!

Introducing AAA

- The "Any Data, Any Time, Anywhere" project is an NSF-funded initiative to increase data accessibility for the HEP community.
    - We are working to build an data access infrastructure, based on existing components, that makes it possible to run CMS jobs anywhere.

Data Access

Physics Data

- The most challenging aspect is access to physics data.
- CMS manages about 50PB of disk space at its data centers.
- Remote access is mediated through a web-services protocol called "SRM" and transfers via GridFTP.
  - GridFTP typically used to transfer the complete file.
- Access to the data via CMS software is done by specialized protocols or POSIX access.
  - CMS jobs tend to read <1/3 of the file at a rate of 128KB/s to 2MB/s.

The Woe of One Event

- If I want to read a single event, how do I get the data? Options:
  - **Run a grid job on that event**: best case, 15 minutes (create the job, submit it, have it run, fetch results). Worst case, hours of queue time.
  - **Download the file**: First you have to find it and setup the tools. If you're lucky, only 5 minutes to download.

Events per second when jobs are running can be impressive - but the overhead kills things when looking at a single event!

Direct Remote Access is Key

- We turned to the Xrootd project to provide remote, direct access to data stored at sites.
    - Mature project for remote-I/O.
    - Client almost always integrated into ROOT.
    - Has the security mechanisms WLCG needs.
    - Time to open event interactively is limited to network latency.

Key Features

- Client is robust against a multitude of networking failures and misconfigured endpoints.
  - Remote sites are not run by us, so we can't easily control their configurations or software versions.
- Protocol can batch many requests into one network round-trip (routinely hundreds of read requests are batched into one packet).
  - Prefetching and request-batching ("vector reads") are essential to reducing the effect of latency; the whole system depends on this!

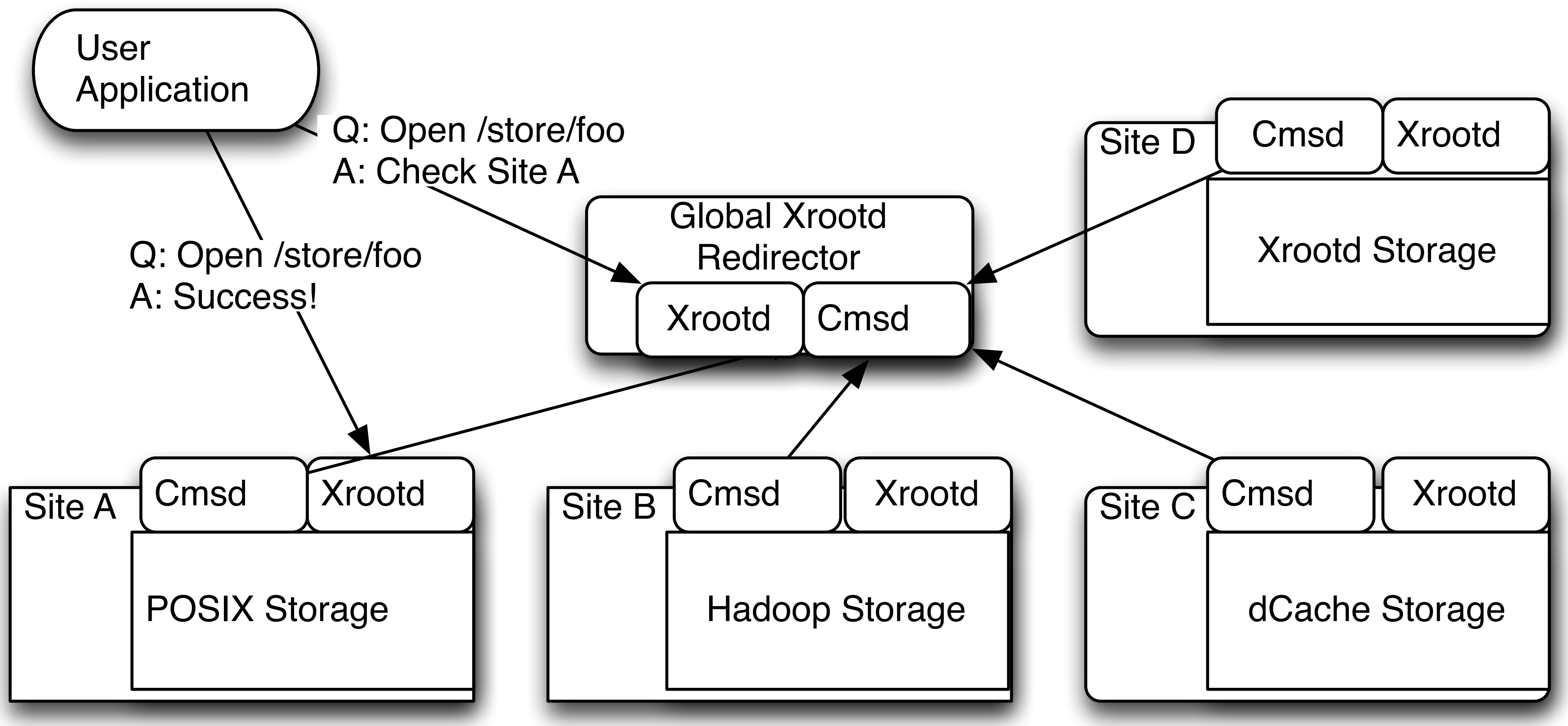
Introducing Federations

- Remote access gives us data for *one* site. We need a federation to access all sites.
- Definition of a **federated storage system***:
  - A collection of disparate storage resources managed by cooperating but independent administrative domains transparently accessible via a common namespace.

* From the Lyon workshop on Federated Data Stores: http://indico.in2p3.fr/conferenceProgram.py?confId=5527

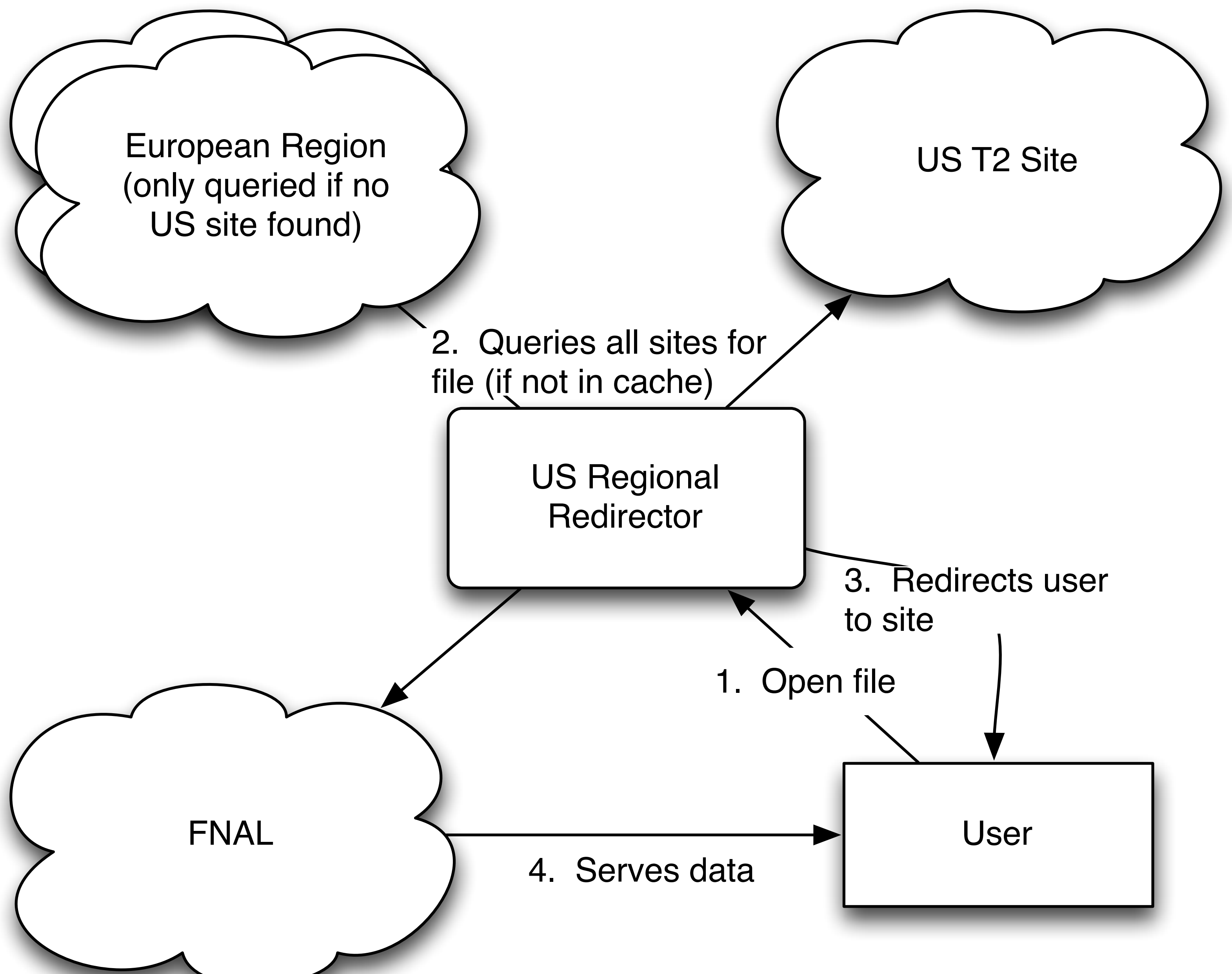
Federating Xrootd

- The simplest kind of federation is illustrated below:



Federation overlays on top of existing storage

Cross-region queries

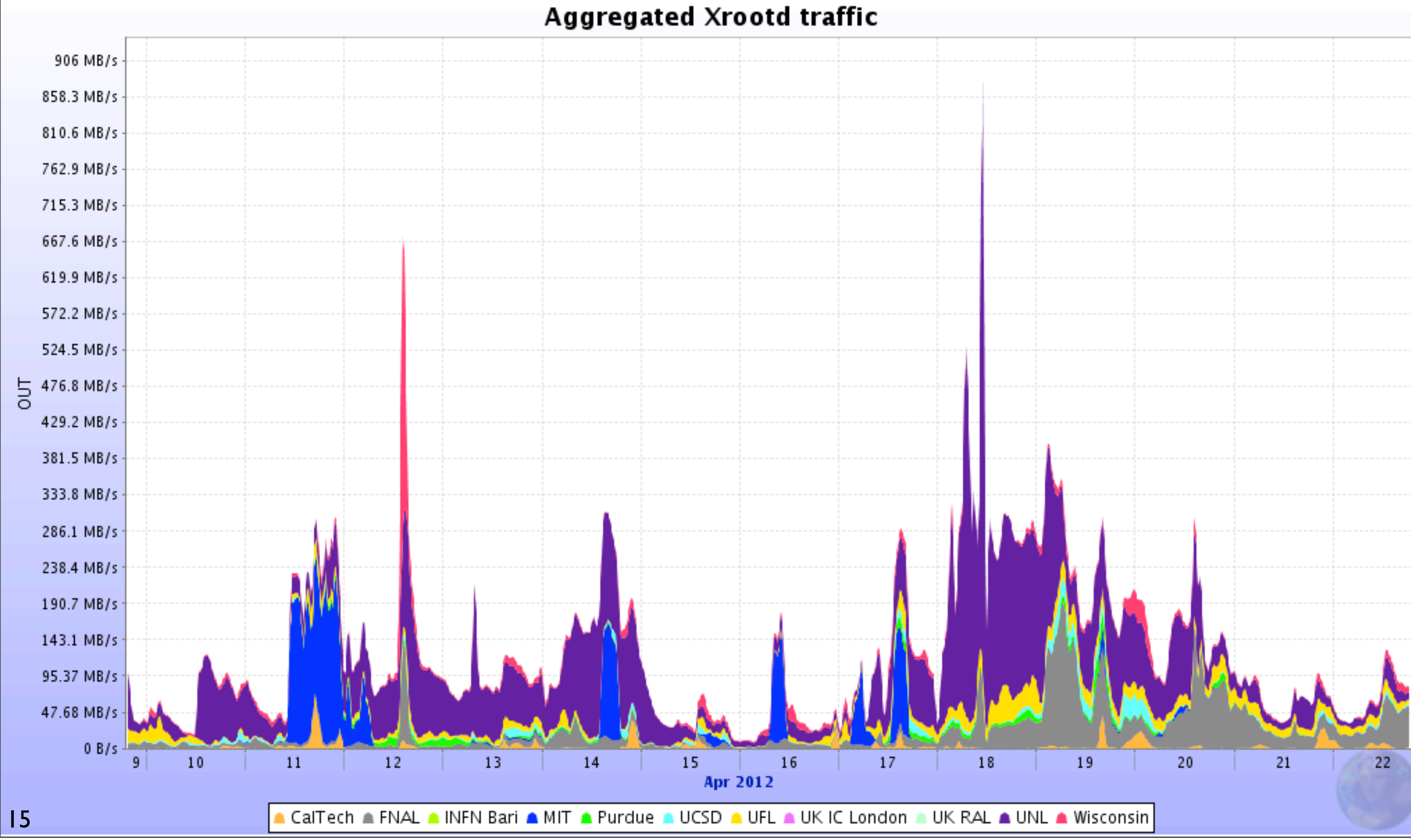


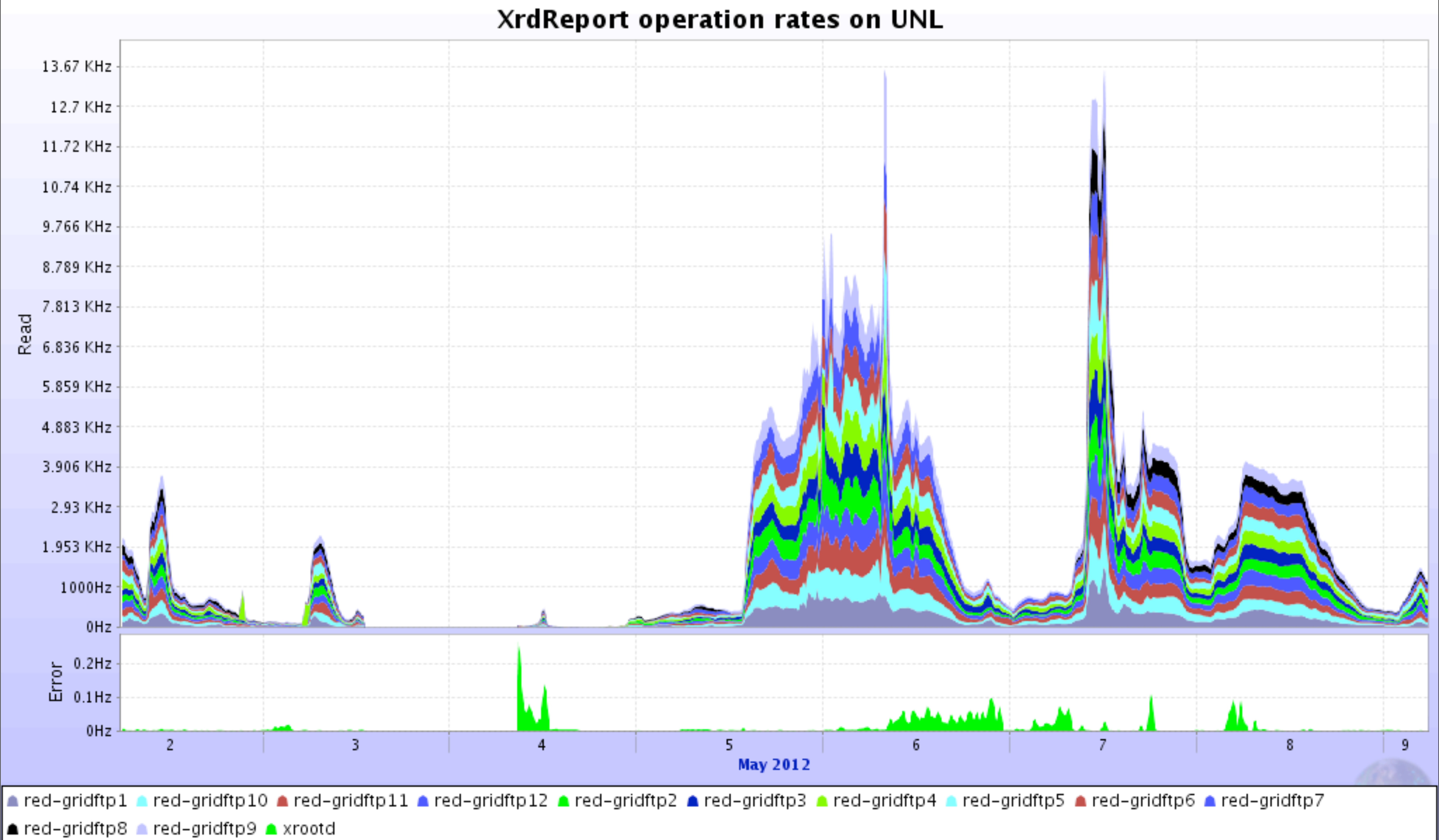
To limit namespace query propagation, queries spill over to other regions only if nothing is found locally.

Deployment

- Currently, redirector at xrootd.unl.edu.
- Includes the FNAL T1 (dCache) and 8 T2s (5 HDFS, 1 dCache, 1 Lustre, 1 L-Store).
- During April, our monitoring recorded:
  - Over 300 unique users,
  - 900K file transfers
  - 300TB moved.

Monitoring


Aggregated Xrootd traffic
906 MB/s
858.3 MB/s
810.6 MB/s
762.9 MB/s
715.3 MB/s
667.6 MB/s
619.9 MB/s
572.2 MB/s
524.5 MB/s
476.8 MB/s
429.2 MB/s
381.5 MB/s
333.8 MB/s
286.1 MB/s
238.4 MB/s
190.7 MB/s
143.1 MB/s
95.37 MB/s
47.68 MB/s
0 B/s
OUT
9 10 11 12 13 14 15 16 17 18 19 20 21 22
Apr 2012
CalTech FNAL INFN Bari MIT Purdue UCSD UFL UK IC London UK RAL UNL Wisconsin



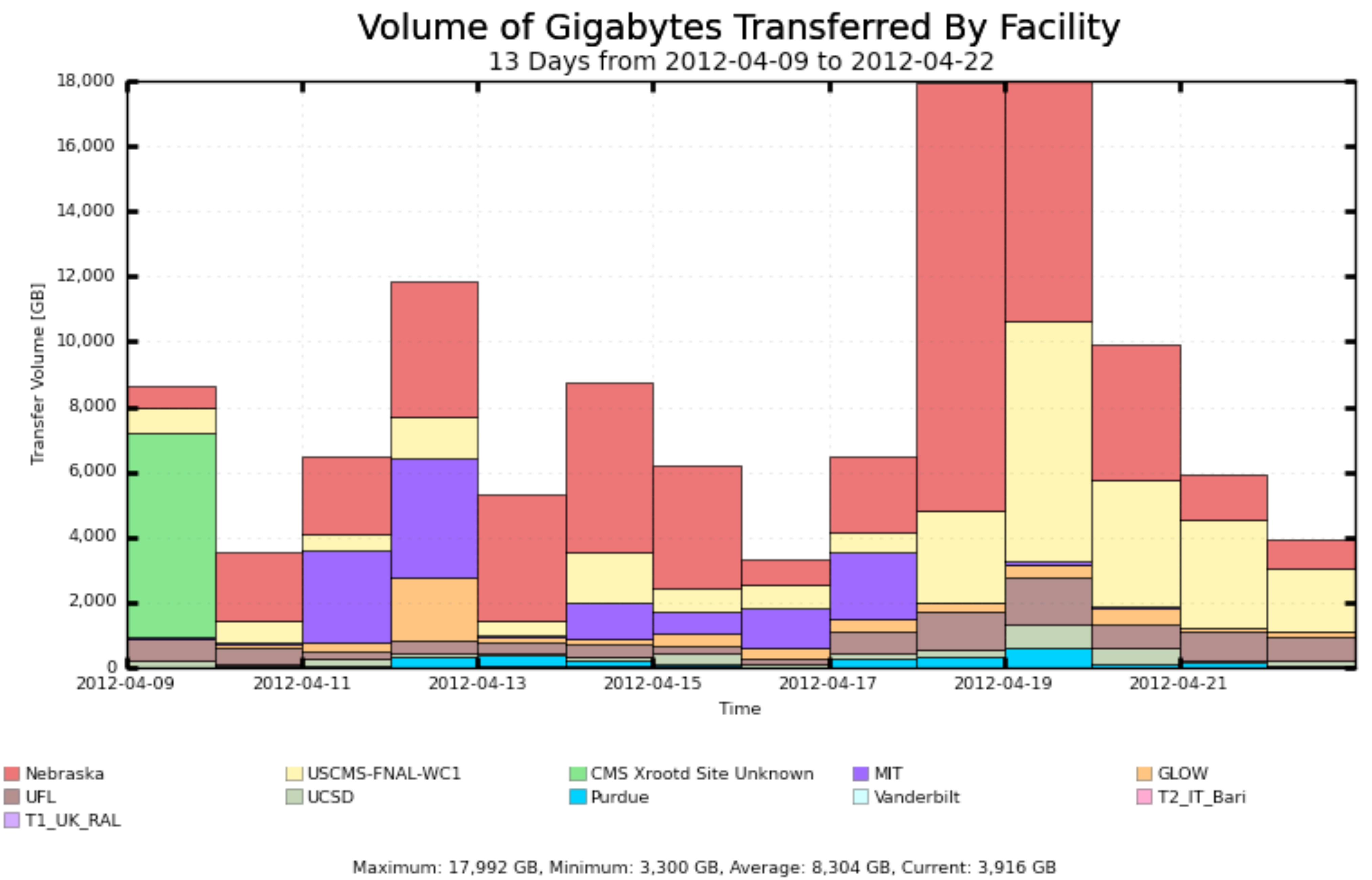
To limit the effects of latency, reads are bundled into large vectors.

Xrd open files [582/3130]

xrootd.t2.ucsd.edu:4243/?no_same_site

Apple | News | CmsXrootdArchitect | Xrootd Nagios | Building a CMSSW re | WLCG TEG Data Man | GlideinFrontend120 | CMS Exit Codes

File Λ	User Hash	Server Domain	Client Domain	Open Ago	Update Ago	Read [MB] Λ	Read [%]	Rate
/store/mc/Fall11	6DB2B1B7	hep.wisc.edu	unl.edu	25:37:11	21:22:11	2728.245	68.827	0.030
/store/mc/Summer11	3C12BE84	fnal.gov	unl.edu	01:06:03	00:05:23	2523.107	62.609	0.637
/store/data/Run2012A	3D8C2B76	fnal.gov	t2.ucsd.edu	04:30:43	00:08:38	2498.118	54.639	0.154
/store/data/Run2012A	3D8C2B76	fnal.gov	t2.ucsd.edu	03:15:18	00:01:08	2474.463	57.433	0.211
/store/data/Run2012A	3D8C2B76	fnal.gov	t2.ucsd.edu	04:00:37	00:01:42	2442.054	57.521	0.169
/store/mc/Summer11	3C12BE84	fnal.gov	unl.edu	00:51:58	00:00:58	2429.944	64.505	0.779
/store/mc/Summer11	3C12BE84	fnal.gov	t2.ucsd.edu	02:14:36	00:04:21	2420.094	59.497	0.300
/store/data/Run2011A	3C12BE84	ihepa.ufl.edu	unl.edu	01:26:01	00:04:41	2413.841	63.327	0.468
/store/data/Run2012A	3D8C2B76	hep.wisc.edu	t2.ucsd.edu	04:00:47	00:10:22	2411.461	52.835	0.167
/store/mc/Summer11	3C12BE84	unl.edu	t2.ucsd.edu	01:31:55	00:00:50	2366.587	64.929	0.429
/store/data/Run2011A	3C12BE84	ihepa.ufl.edu	unl.edu	03:52:41	00:01:56	2365.060	59.156	0.169
/store/data/Run2012A	3AC086B	fnal.gov	t2.ucsd.edu	01:17:18	00:00:18	2320.691	62.855	0.500
/store/data/Run2012A	3D8C2B76	hep.wisc.edu	t2.ucsd.edu	04:41:24	00:08:09	2274.699	48.299	0.135
/store/data/Run2012A	3D8C2B76	hep.wisc.edu	t2.ucsd.edu	04:01:48	00:00:23	2212.339	59.802	0.152
/store/mc/Summer11	3C12BE84	hep.wisc.edu	ultralight.org	02:02:28	00:16:48	2170.151	54.353	0.295
/store/data/Run2012A	3D8C2B76	fnal.gov	t2.ucsd.edu	04:27:46	00:05:21	2164.812	53.054	0.135
/store/data/Run2011A	3C12BE84	ihepa.ufl.edu	unl.edu	01:20:25	00:00:30	2117.791	53.866	0.439
/store/data/Run2011A	3C12BE84	ihepa.ufl.edu	unl.edu	03:52:41	00:11:16	2101.871	52.162	0.151
/store/data/Run2012A	3D8C2B76	fnal.gov	t2.ucsd.edu	04:20:26	00:00:56	1991.785	61.253	0.127
/store/data/Run2012A	3AC086B	ihepa.ufl.edu	t2.ucsd.edu	08:41:15	00:00:05	1976.205	62.562	0.063
/store/mc/Summer11	3C12BE84	unl.edu	ultralight.org	01:17:14	00:10:34	1953.870	53.664	0.422
/store/mc/Fall11	B83B3C94	hep.wisc.edu	ultralight.org	11:32:01	02:49:41	1944.444	52.812	0.047
/store/data/Run2012A	3D8C2B76	hep.wisc.edu	t2.ucsd.edu	03:17:33	00:02:03	1919.433	64.035	0.162
/store/data/Run2011A	3C12BE84	ihepa.ufl.edu	unl.edu	03:38:01	00:00:01	1908.636	50.158	0.146


Volume of Gigabytes Transferred By Facility
13 Days from 2012-04-09 to 2012-04-22
Transfer Volume [GB]
Time
Nebraska
USCMS-FNAL-WC1
CMS Xrootd Site Unknown
MIT
GLOW
UFL
UCSD
Purdue
Vanderbilt
T2_IT_Bari
T1_UK_RAL
Maximum: 17,992 GB, Minimum: 3,300 GB, Average: 8,304 GB, Current: 3,916 GB

Sample Daily Report

```
===========================================================
  Xrootd Summary for 2012-05-09 | 59.92 TB | 37% increase
===========================================================
```

Source Site	Volume GB	# of Transfers	Yesterday Diff	One Week Diff
GLOW	1,080	1,223	55%	1908%
GLOW_Internal	46,053	38,388	65%	32%
MIT	4,950	11,460	103%	95800%
Nebraska	4,195	9,118	-43%	65%
Purdue	415	2,168	279%	374%
T2_IT_Bari	0	6	Unknown	-96%
UCSD	551	4,245	-65%	927%
UFL	1,138	2,066	945%	1250%
USCMS-FNAL-WC1	1,521	2,222	-57%	-32%
Vanderbilt	14	746	400%	598%

Conditions Data

- Conditions data is, happily, mostly taken care of for us.
  - All conditions data in CMS is distributed via a network of HTTP proxies.
  - The actual volume is small (50GB total; small percentage of this is used per job).
- So, only inbound HTTP (preferably, a local HTTP proxy cache) is needed for this.

Software

- The base software install takes about 20GB, and 5-10GB per additional version of CMSSW. 1-2 hours to deploy it "on the fly".
    - Due to disk size and time restraints, impossible to deploy it "on the fly" with the job.
- Given an arbitrary job, only a small portion of the files are used.

How do we solve this? See the next talk!

Use Cases

Interactive Use

- Distributing the software via CVMFS, conditions via HTTP, and physics data via Xrootd, any CMS job can run on an any computer - regardless of whether it is in a CMS data center.
- This covers the "interactive use case", where a physicist is debugging their code or viewing events in the event viewer.
    - A significant percentage of AAA is optimizing the I/O code to be robust in face of high-latency connections.

Fallback

- In CMS, the jobs are sent to a site where the data is stored.
  - Due to unpredictable transient issues, a small percentage is unavailable at any given time.
  - If a job cannot open a file, access fails.
- Now, if a job cannot open a file, it uses the xrootd infrastructure instead.

Overflow

- We use the glideinWMS software to create a heterogeneous Condor pool containing worker nodes from as many sites as possible.
- Condor knows which site each slot is from and matches the jobs according to data locality.
    - Due to transient non-optimal data distribution, there may be slots available with no matching slots.
    - In this case, we will purposely send queued jobs to the "wrong" site if the fallback mechanism can provide the data over the WAN.
        - We're getting close to being able to send jobs to non-CMS sites in production!

Security

- The overflow job is submitted under one identity (the "pilot"), but the actual code is from the user.
    - Needs to run under a different identity, as user can run arbitrary code and the pilot identity is quite powerful.
    - At CMS sites, we have a setuid binary (glexec) to allow identity switching.
    - A bit heavy-handed for opportunistic sites; looking to use parrot identity boxing for isolation.

Thoughts for the Future

- We use CCTools to enable opportunistic grid work.
- Xrootd deployment and client are unique in their abilities to reliably federate multiple sites.
  - But the protocol is niche.
  - We are starting to understand what aspects are lacking in other protocol stacks (e.g., HTTP).
  - It outlines a path that others could take.

The Commoditization of HEP?

- The AAA infrastructure shows CMS can utilize non-CMS sites.
  - We've been able to greatly decrease the "site footprint" of an HEP experiment with respect to the data management.
  - With a significant investment, one could use similar techniques for HTTP. *There is not a straightforward translation*.
- The vision is "Computing as a Service": be able to utilize a research computing site with as light a footprint as possible.
- The computing sites are only partway: the final frontier for the commoditization is showing a HEP experiment can use an off-the-shelf workflow management system.