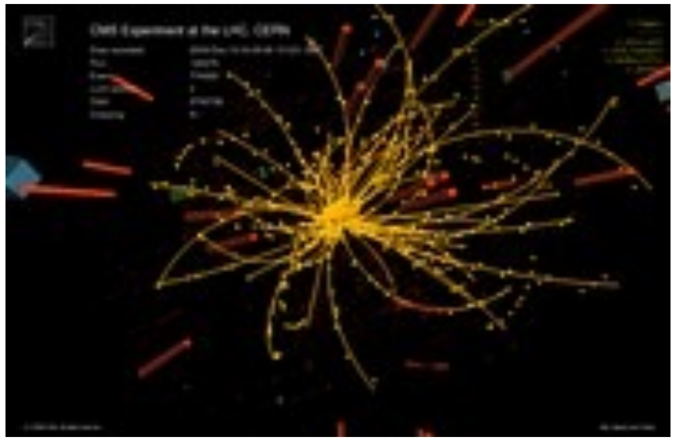
15th International Workshop on advanced computing and analysis techniques in physics
May 16-21, 2013, Beijing, China
http://acat2013.ihep.ac.cn

CMS Use of a Data Federation

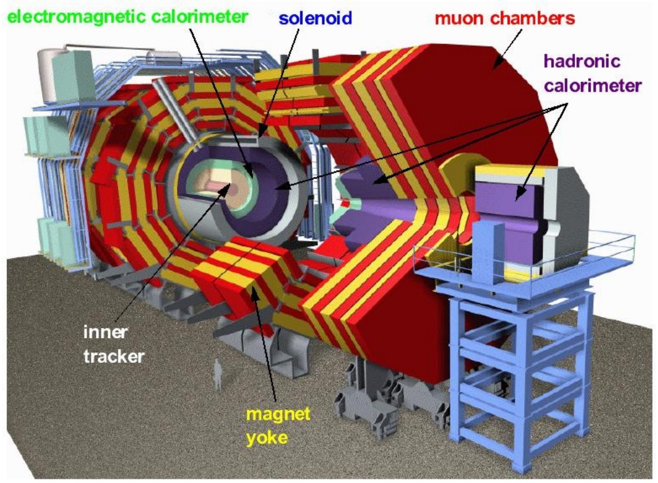
Peter Elmer, Princeton University
(for the CMS Computing group)

CMS at the Large Hadron Collider (LHC)

- General purpose detector for both proton-proton and Heavy Ion collisions at the LHC



- A collaboration of ~3800 physicists and engineers from ~180 institutions in 39 countries



- computing resources are provided to the experiment by about 60 computing centers

CMS Computing Model

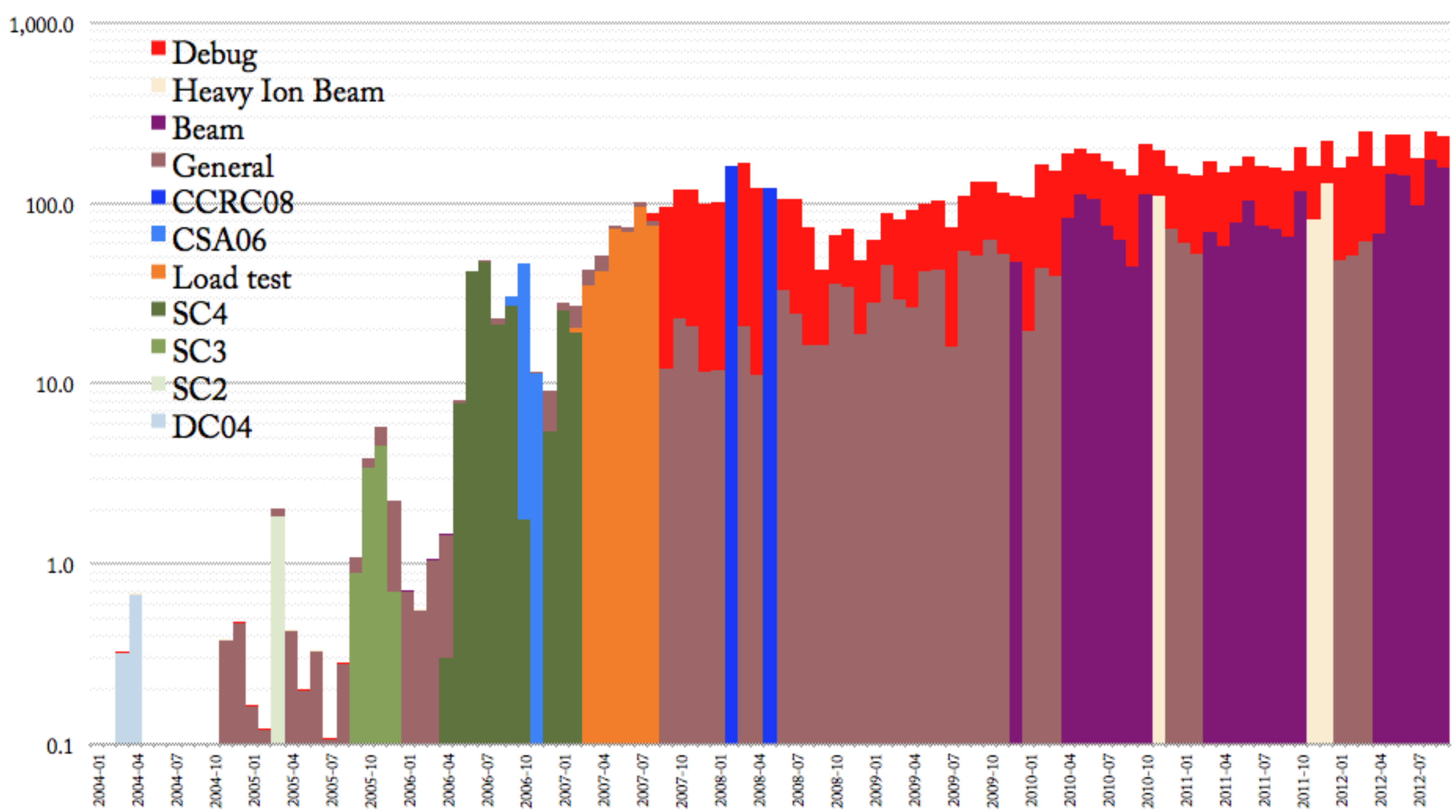
- The baseline CMS Computing Model was developed in 2005: an era of unreliable sites, unreliable storage systems, an unreliable grid and poor monitoring if one wanted to understand what was happening.
- We opted for a very simple model:
  - Datasets are statically placed at sites, either by the central computing operations or by request of analysis groups
  - Jobs go where the data is located and read data locally
  - No data is moved on response to job submissions
- Uses more storage, a tradeoff between efficiency and system complexity

PhEDEx - **Ph**ysics **E**xperiment **D**ata **Ex**port

Generic data transfer and placement system

Design focused on scalability and reliability from the beginning

A transfer management database plus a set of loosely coupled, stateless agents



CMS average data transfer volume (2004-2012)

<u>Volume:</u>
Up to 250TB/day transferred (500k transfer successes/week)

<u>Files:</u>
11M (24M incl. replicas)

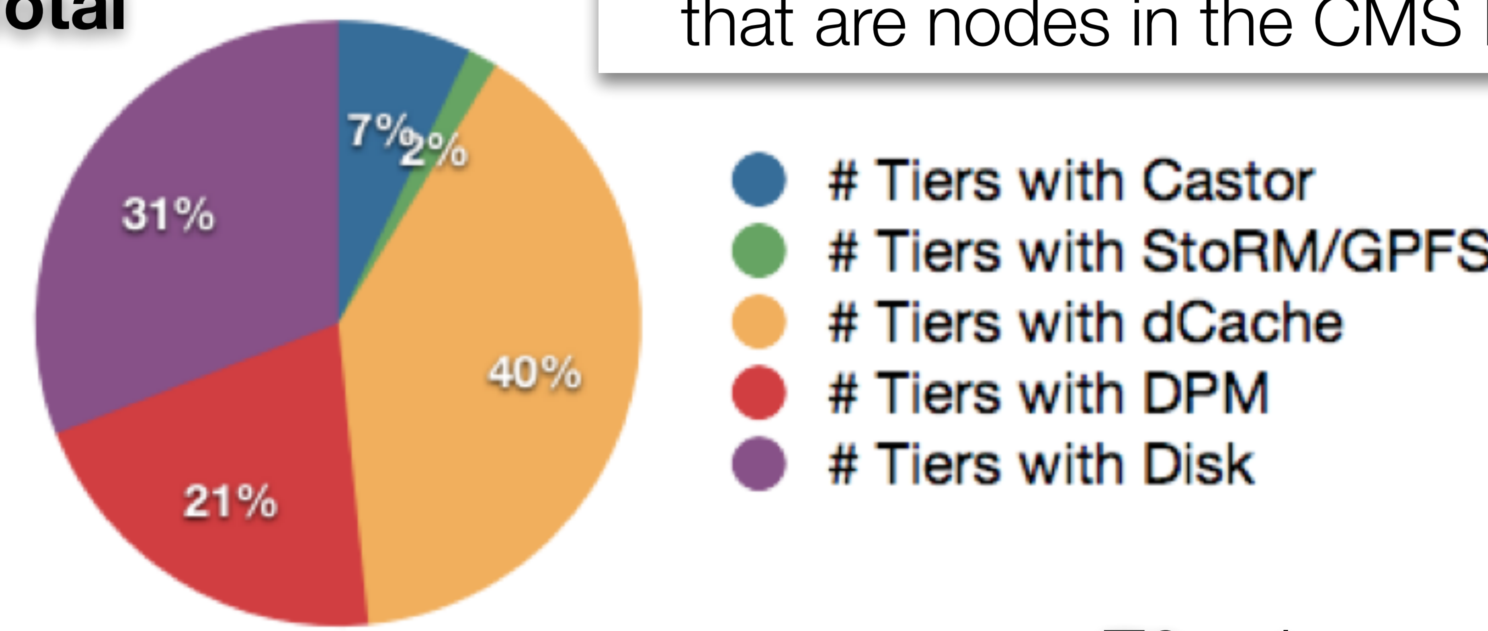
<u>Throughput:</u>
Up to 2-2.5GB/s aggregate in peak weeks

Computing tool with the greatest longevity in CMS

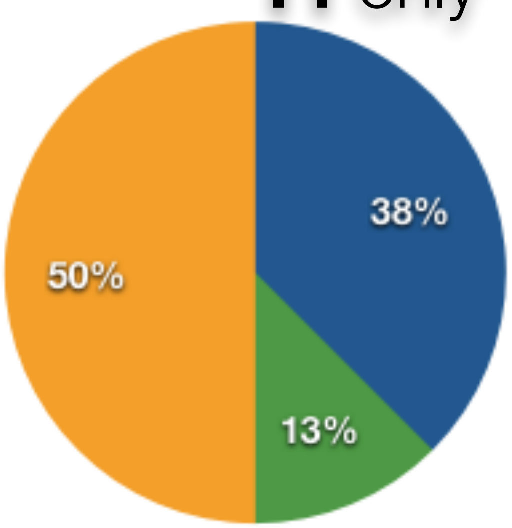
Storage Heterogeneity

Breakdown of **storage solutions** at sites that are nodes in the CMS PhEDEx topology

Total



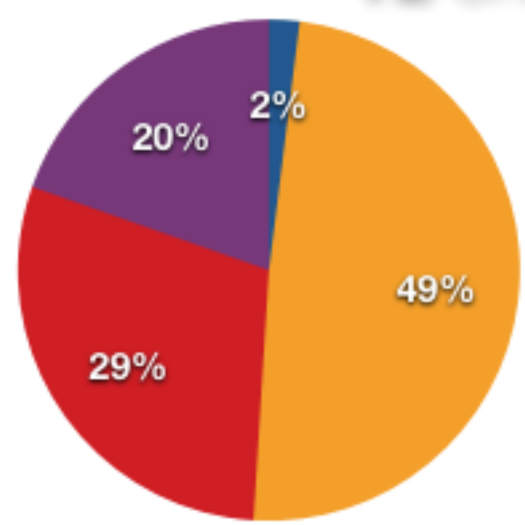
T1 only



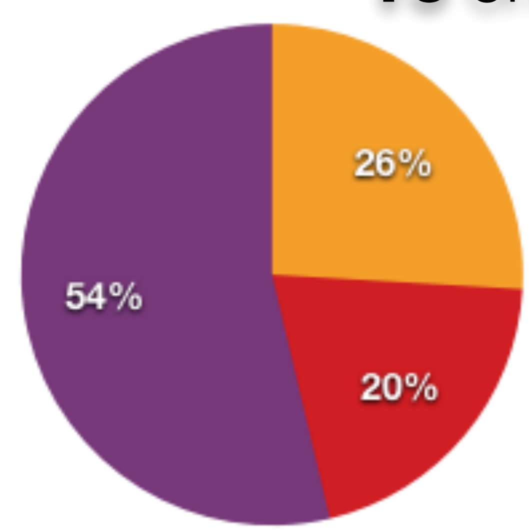
CERN	Castor

ASGC	Castor
CNAF	StoRM/GPFS
FNAL	dCache
IN2P3	dCache
KIT	dCache
PIC	dCache
RAL	Castor

T2 only



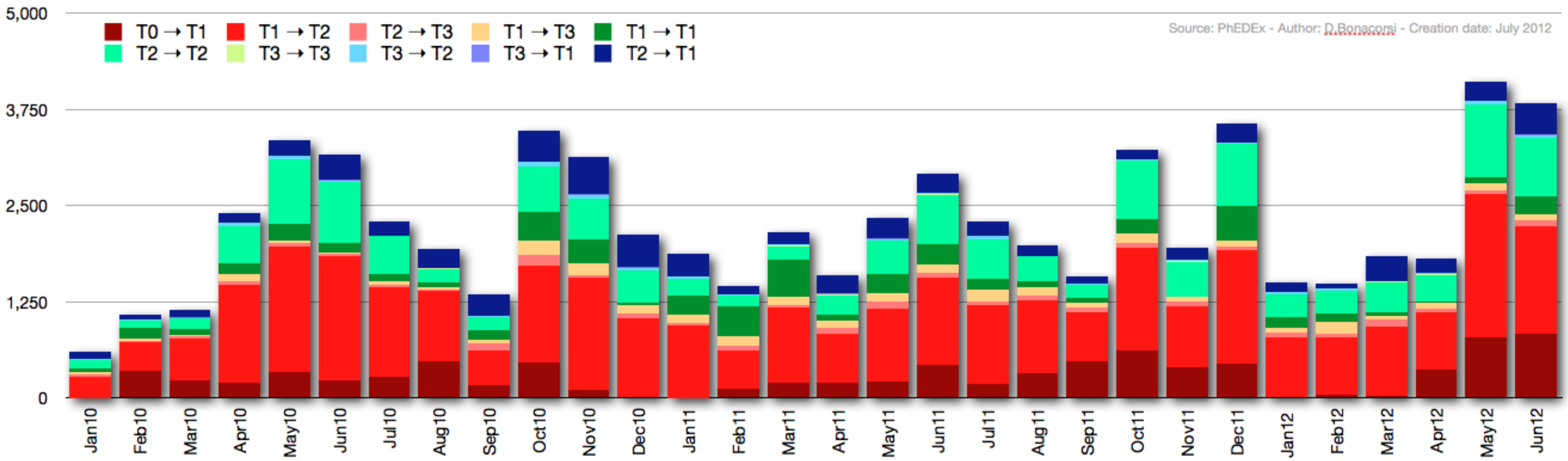
T3 only



Data Federations

- The maturity of the computing system leads to an emphasis on new goals: efficiency of storage use (eventually fewer replicas?), lower latency for end users, greater transparency relative to storage system heterogeneity, improved usability for analysis

- Much of the traffic is to T2 sites, where analysis is happening



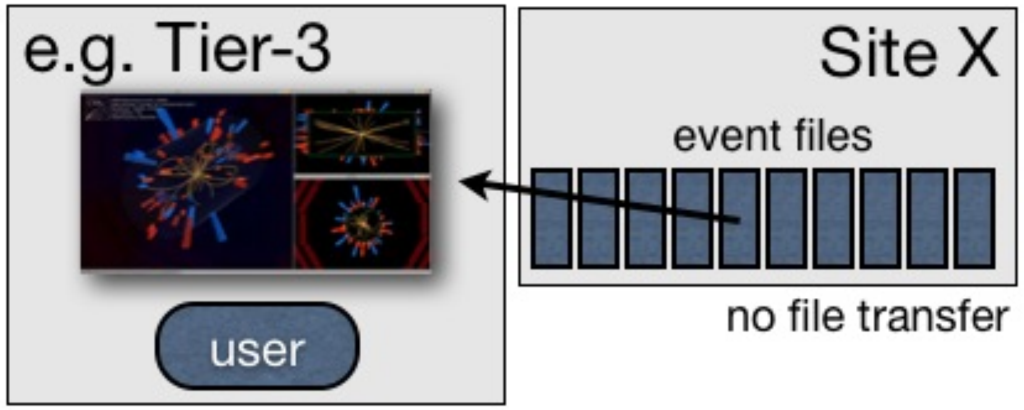
Data Federations

- Exploit WAN data access as a complement to local access: allow jobs running at one site to transparently access files located at other sites
- In short, create a federated data storage system which provides access to files residing in a heterogeneous mix of site storage systems and independent administrative domains.
- Requires a common global namespace. For efficiency applications also need to handle gracefully higher latency file reads and bandwidth limitations.
- A data federation has however several interesting use cases

Use cases

Interactive use-case

- debugging single events / files
  - hosted remotely, event viewer



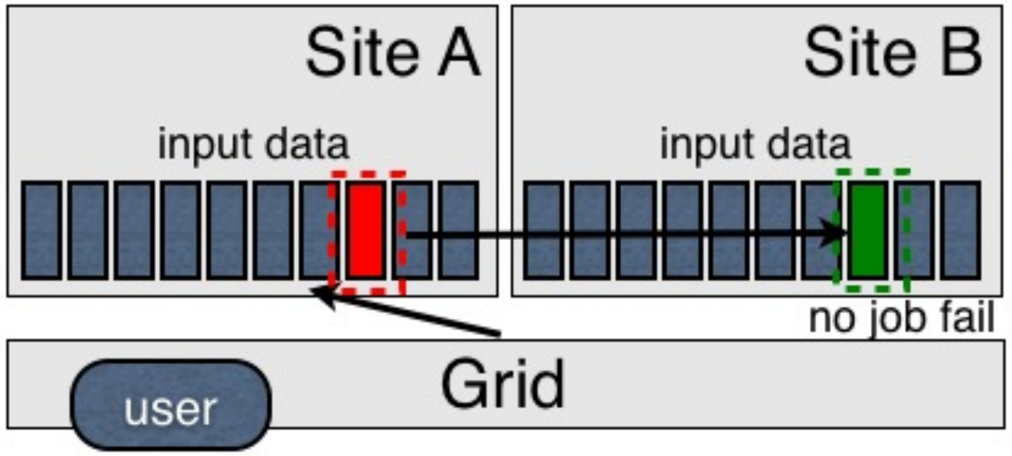
Use cases

Interactive use-case

- debugging single events / files
  - hosted remotely, event viewer

Fallback use-case

- if a grid jobs fails to open a file, no application failure: have it try again remotely
  - loss of efficiency, but no crash

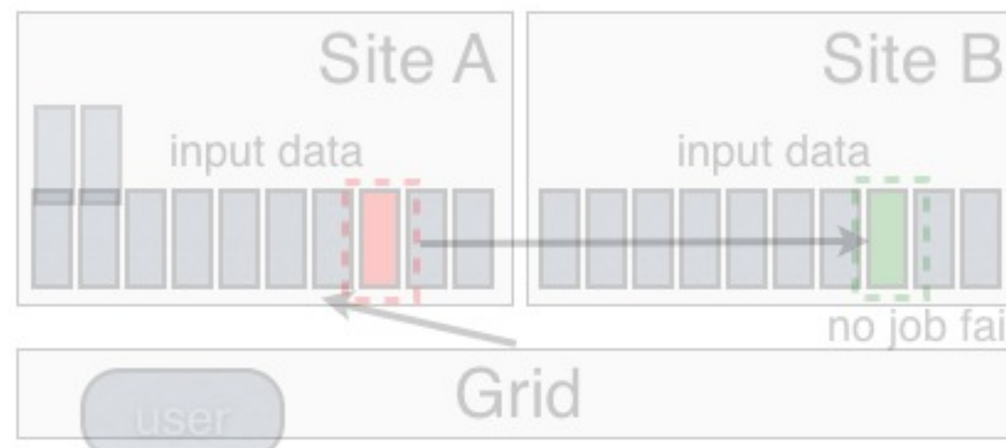


Use cases

Interactive use-case

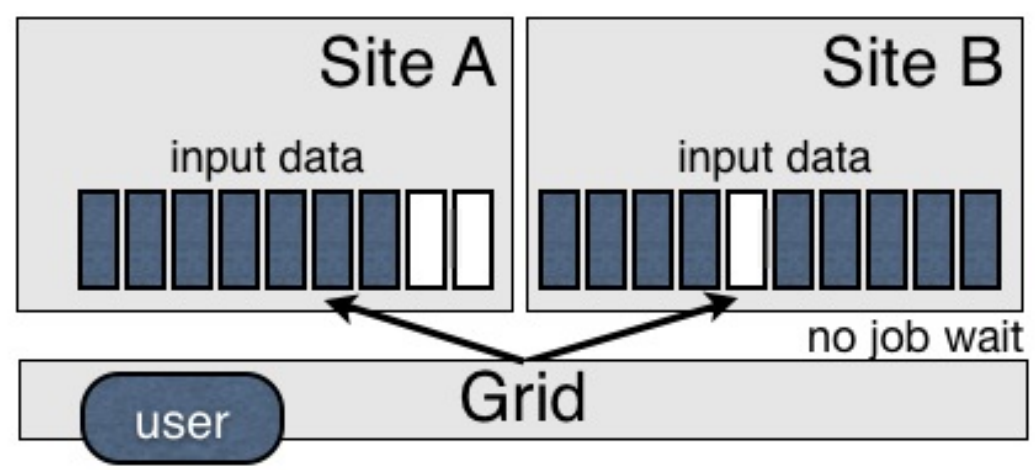
- debugging single events / files
  - hosted remotely, event viewer

Fallback use-case



- if a grid jobs fails to open a file, no application failure: have it try again remotely
  - loss of efficiency, but no crash

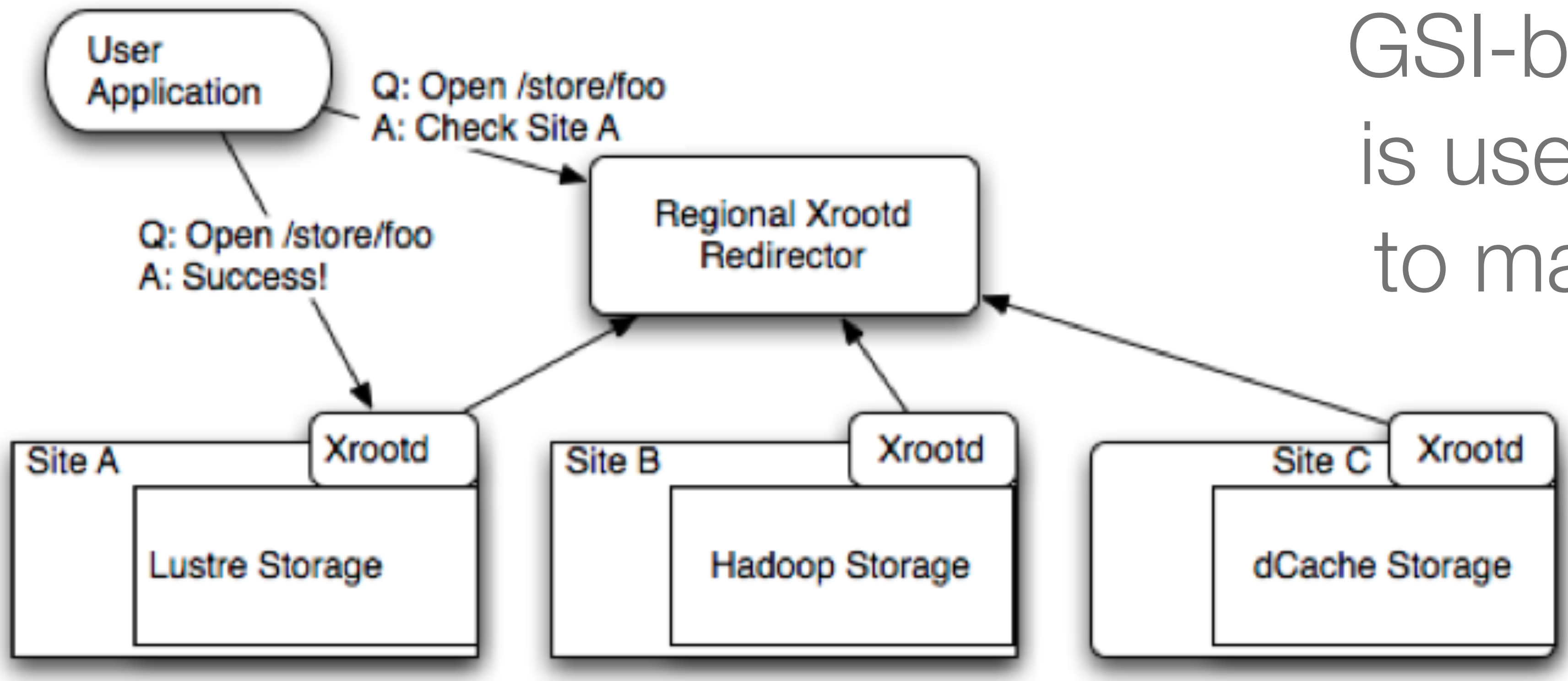
Overflow use-case



- purposely allow a job to go to sites not hosting the full input dataset
  - if one data source is in the federation, a forced fallback will "backfill" otherwise-idle analysis CPUs and work around non-optimal data distribution

xrootd

- The CMS data federation is implemented using xrootd.
- Single entry point for file access is the "redirector". It maintains no permanent file location information, but queries registered xrootd data servers as to whether they have the file. The client is then redirected to one which can provide access.



GSI-based authentication is used, via xrootd plugin to map GSI to username

Global file namespace

- Since 2005 CMS has imposed a single logical file namespace globally. File access (the PFN) at a given site is determined by a site-specific set of transformation rules, i.e. the so-called "trivial file catalog".

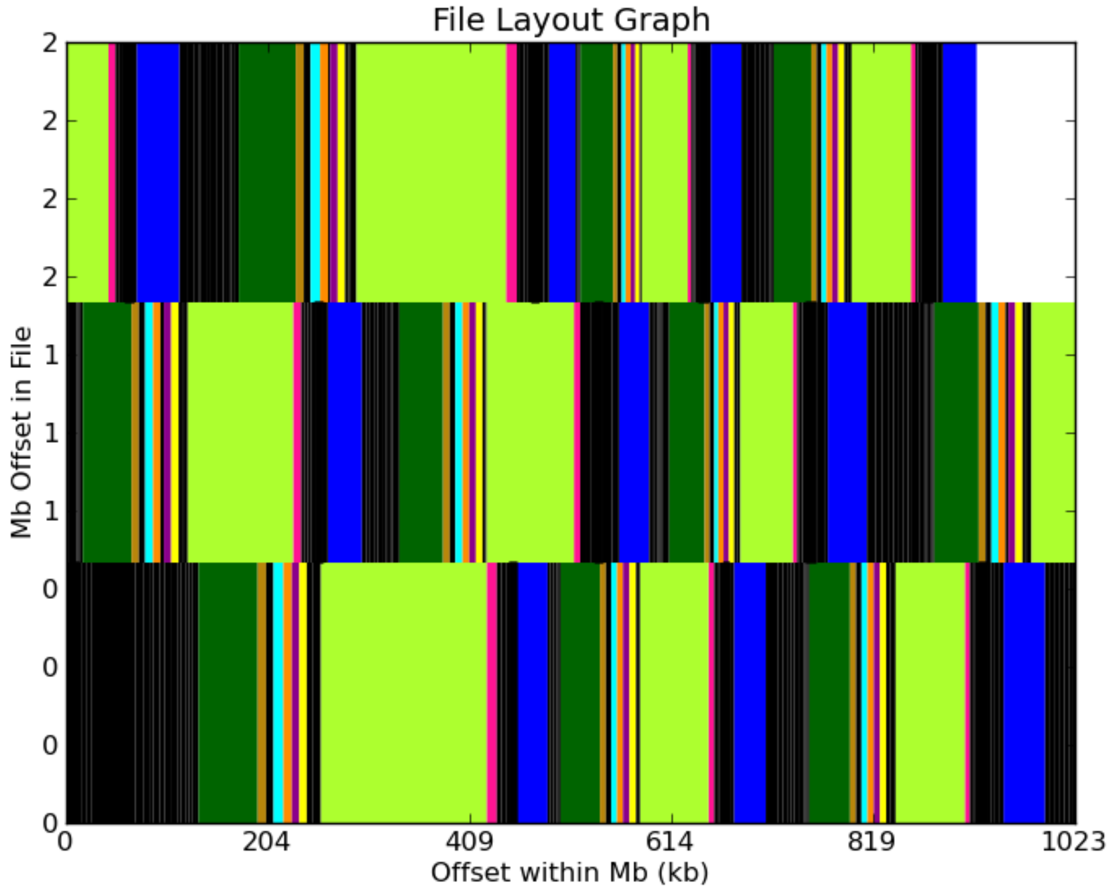
- The use of these rules is setup such that a "fallback" access can be specified. If the job attempts to read the file via the standard (first match) method and it fails, the job will then attempt to open the file via the fallback method. This is configured as the entry point into the global data federation.

- This bit of application-level logic permits the implementation of the use cases mentioned earlier.

Optimizing high-latency file access

- o(50ms) latencies within regions and o(100ms) latency between regions, much greater than local access. Manage reads carefully to avoid small reads.

Use TTreeCache to group many reads into a single, large, vector read. Limit I/O round-trips to one per 512kB read.



Top 10 data products are colored, the rest black

Deployment

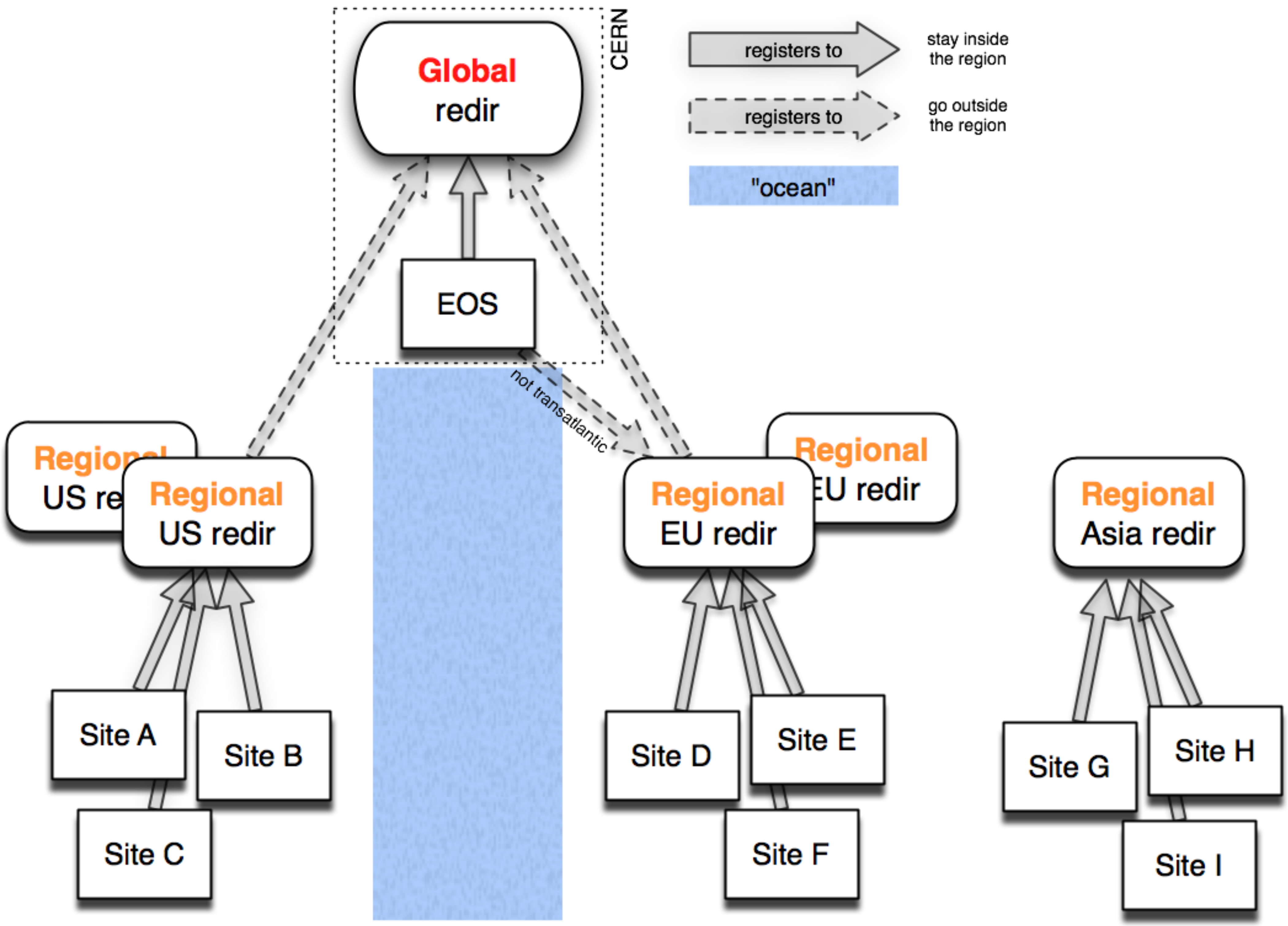
- Funded project "Any Data, Anywhere, Anytime" (AAA) in the US. The first deployment began with sites there as a regional federation.

- Now being expanded to include all CMS sites in a global federation

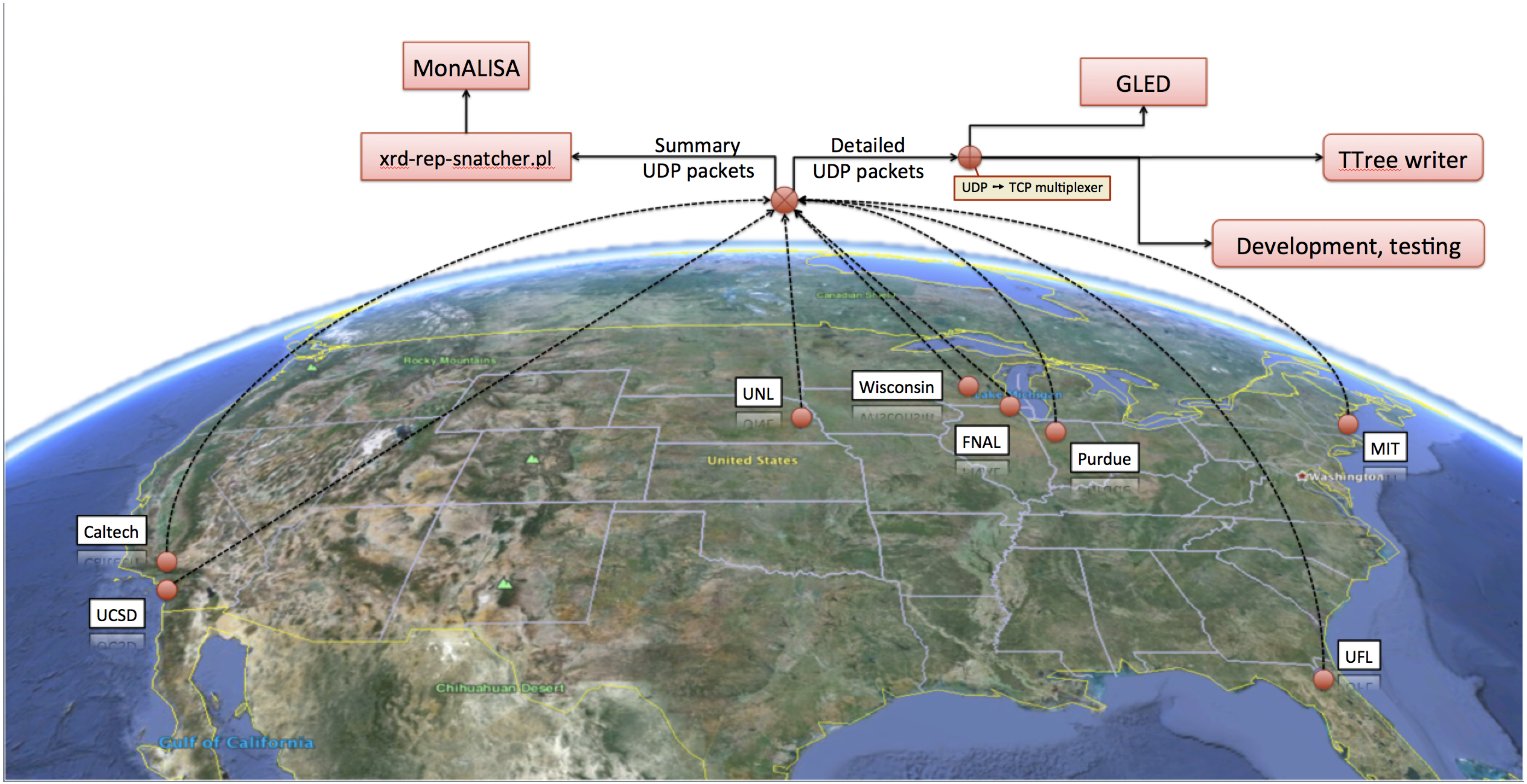
- Access to data in the federation can of course come from anywhere

- The realization of such a system is critically dependent on proper monitoring

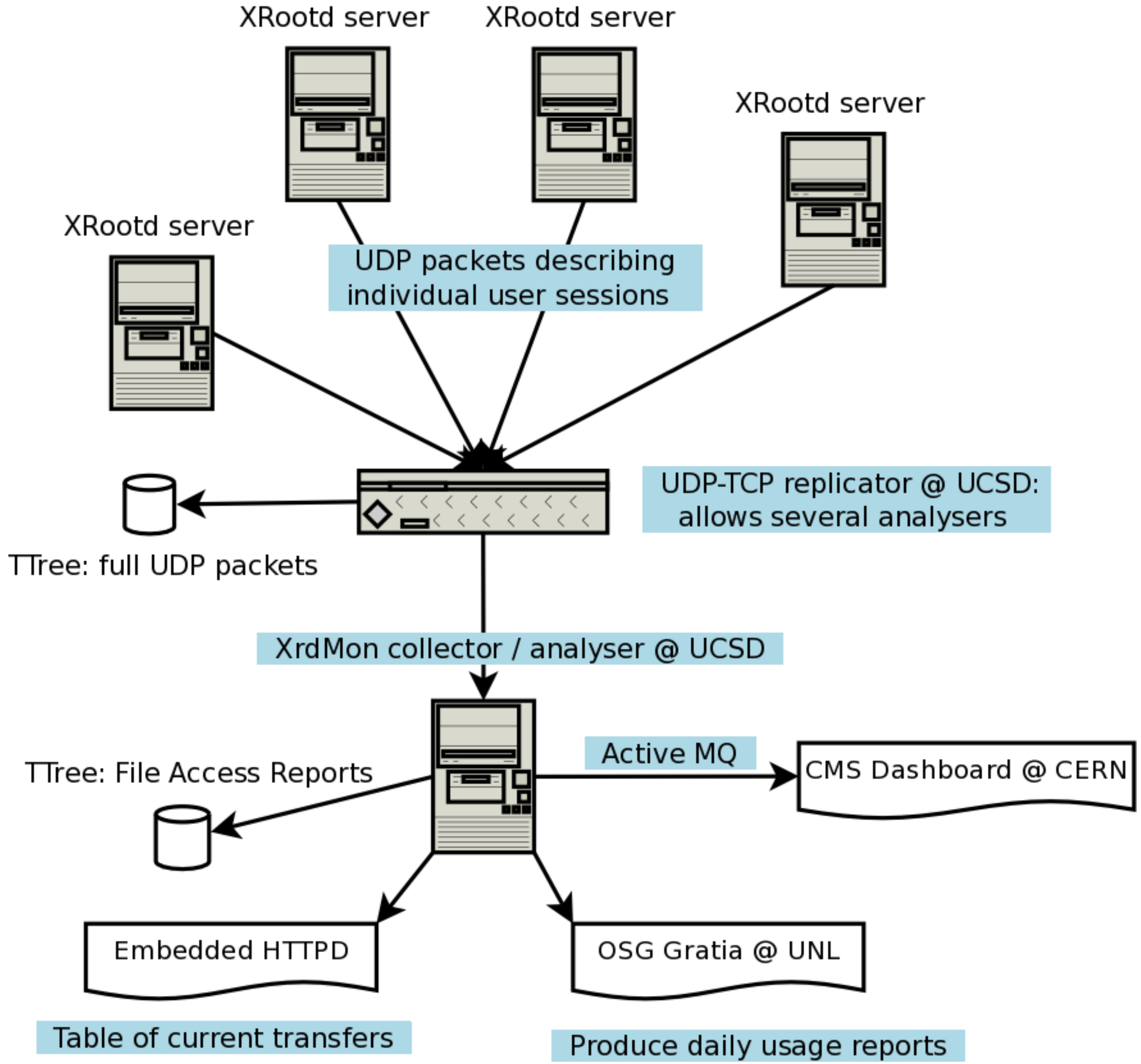
Global Data Federation



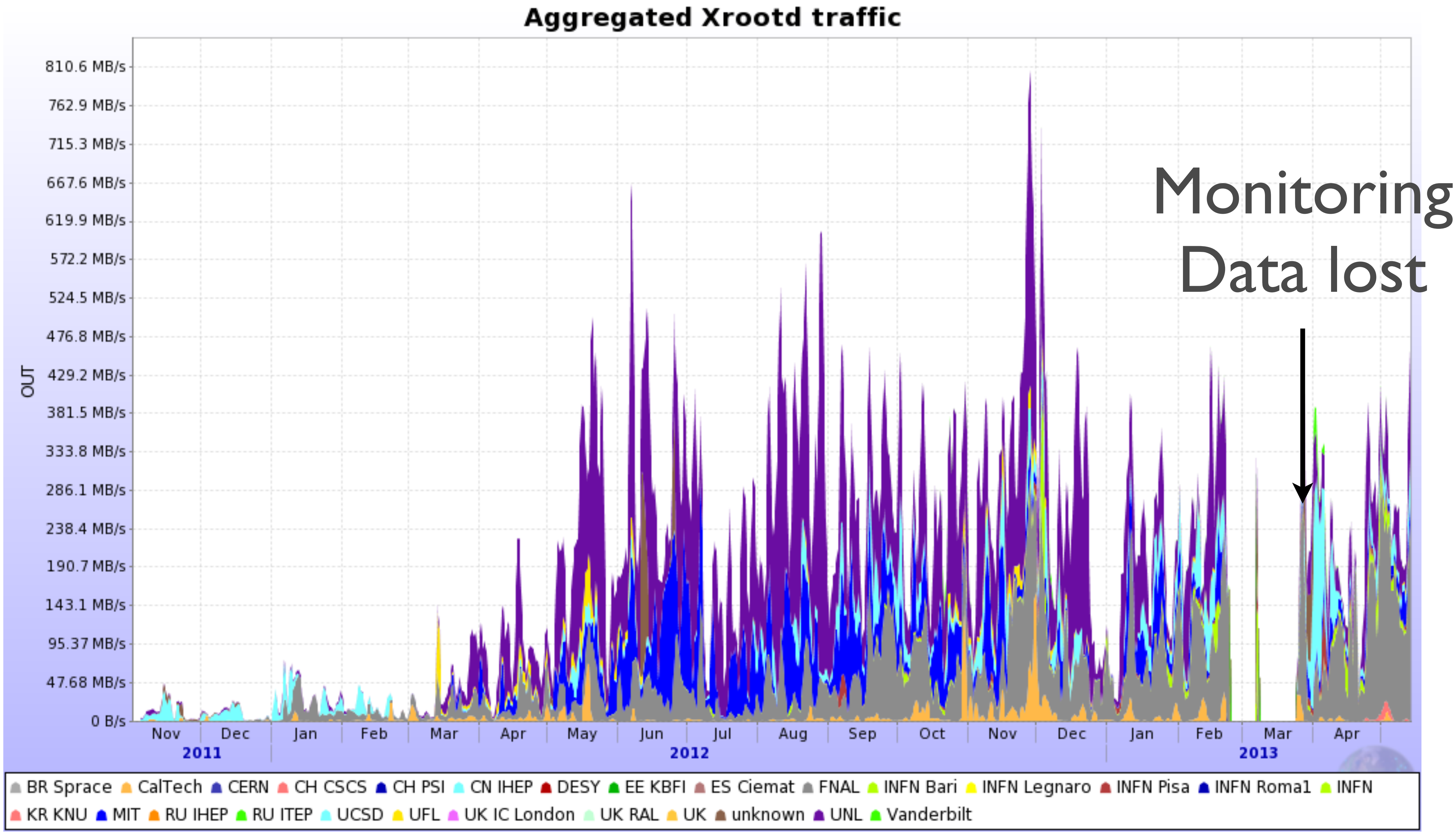
xrootd server side monitoring (UDP)



xrootd monitoring



Growth of remote access data access



30 sites provide access to their data, 37 use as fallback

Realtime monitor

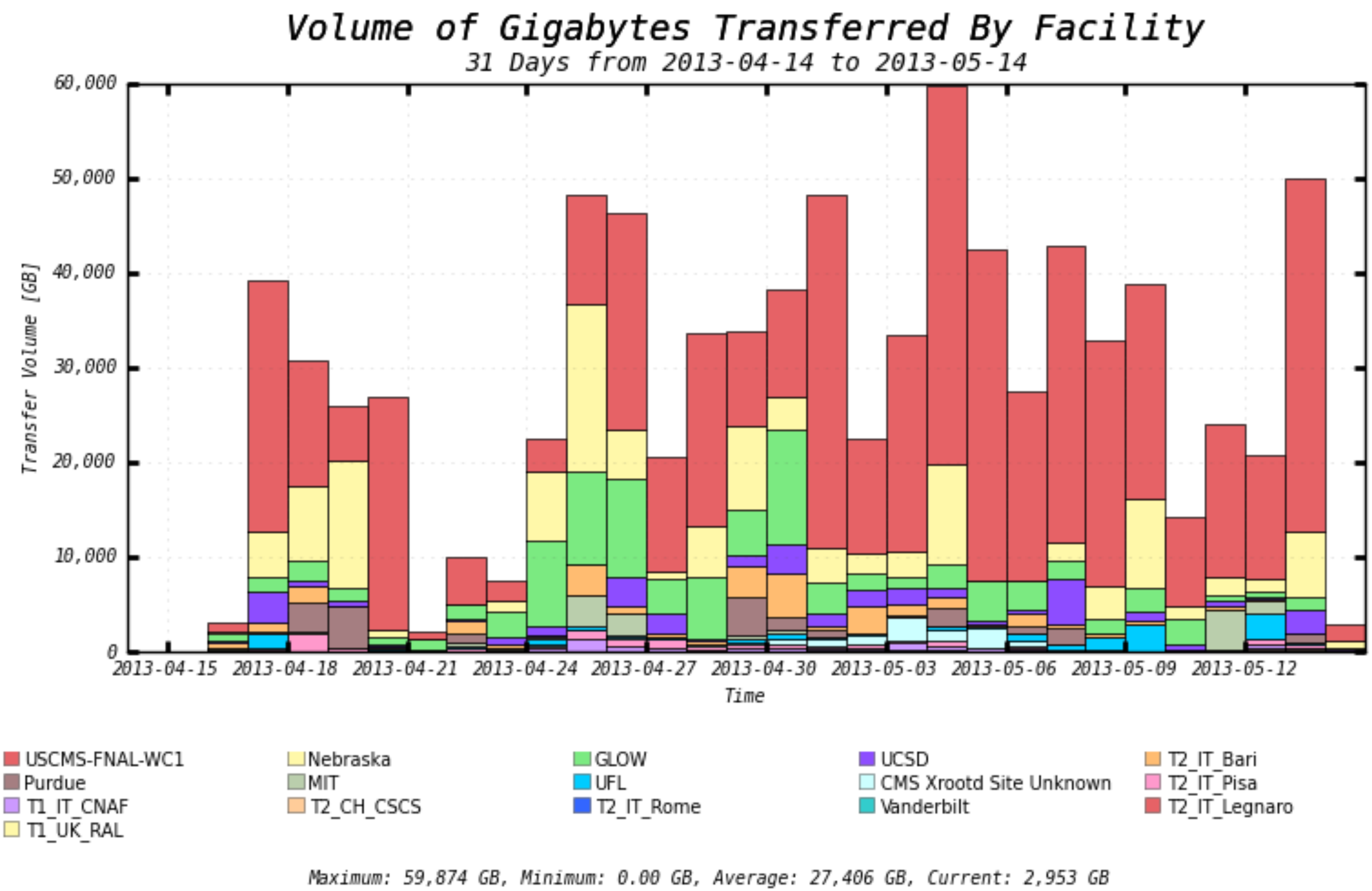
Xrd open files [3513/5322

xrootd.t2.ucsd.edu:4243/xuser/?no_same_site=1

File	User Hash	Server Domain	Client Domain	Open Ago	Update Ago	Read [MB]	Read [%]	Rate [MB/s]	Avg Read [MB]
/store/test/xrootd	3D90840A	rcac.purdue.edu	unl.edu	18:32:39	18:32:39	0.000	0.000	0.000	0.000
/store/data/Run2012D	18D6C056	cmsaf.mit.edu	hep.wisc.edu	12:55:32	12:55:32	0.000	0.000	0.000	0.000
/store/mc/Summer12	D5095D62	t2.ucsd.edu	unl.edu	12:29:53	11:48:03	323.819	8.822	0.129	4.205
/store/mc/Summer12	D5095D62	t2.ucsd.edu	unl.edu	12:26:12	11:21:12	617.432	18.194	0.158	4.642
/store/mc/Summer12	D5095D62	t2.ucsd.edu	unl.edu	12:26:07	11:50:57	348.043	9.292	0.165	3.783
/store/mc/Summer12	D5095D62	t2.ucsd.edu	unl.edu	12:25:59	11:47:28	362.188	10.167	0.157	4.704
/store/mc/Summer12	D5095D62	t2.ucsd.edu	unl.edu	12:23:10	11:12:50	877.042	25.470	0.208	4.363
/store/data/Run2012B	A817F055	hep.wisc.edu	unl.edu	12:17:47	02:35:37	10.604	0.279	0.000	0.080
/store/mc/Summer12	D5095D62	t2.ucsd.edu	unl.edu	12:14:32	11:40:02	294.946	7.233	0.142	4.096
/store/data/Run2012D	96DC057E	hep.wisc.edu	ultralight.org	12:11:41	02:23:01	10.906	0.284	0.000	0.083
/store/mc/Summer12	D5095D62	t2.ucsd.edu	unl.edu	12:09:49	11:25:29	391.817	9.686	0.147	4.664
/store/data/Run2012D	96DC057E	hep.wisc.edu	unl.edu	12:06:42	11:47:12	1776.752	44.453	1.519	5.433
/store/mc/Summer12	D5095D62	t2.ucsd.edu	unl.edu	12:01:11	11:06:51	572.248	14.368	0.176	4.652
/store/data/Run2012D	96DC057E	t2.ucsd.edu	unl.edu	11:44:31	11:42:31	208.446	5.369	1.737	3.722
/store/data/Run2012B	A817F055	t2.ucsd.edu	unl.edu	11:43:41	11:24:11	719.806	19.981	0.615	1.361
/store/mc/Summer12	D5095D62	t2.ucsd.edu	unl.edu	11:39:51	11:26:11	152.566	4.002	0.186	4.015
/store/mc/Summer12	D5095D62	t2.ucsd.edu	unl.edu	11:39:31	11:20:11	195.218	5.599	0.168	0.807
/store/data/Run2012D	96DC057E	hep.wisc.edu	unl.edu	11:38:46	01:56:36	11.103	0.283	0.000	0.084
/store/mc/Summer12	D5095D62	t2.ucsd.edu	unl.edu	11:36:52	08:19:22	1650.219	47.789	0.139	4.854
/store/mc/Summer12	D5095D62	t2.ucsd.edu	unl.edu	11:23:14	11:23:14	0.000	0.000	0.000	0.000
/store/mc/Summer12_DR53X	A011FED1	unl.edu	t2.ucsd.edu	11:19:40	00:09:30	10.482	0.258	0.000	0.069

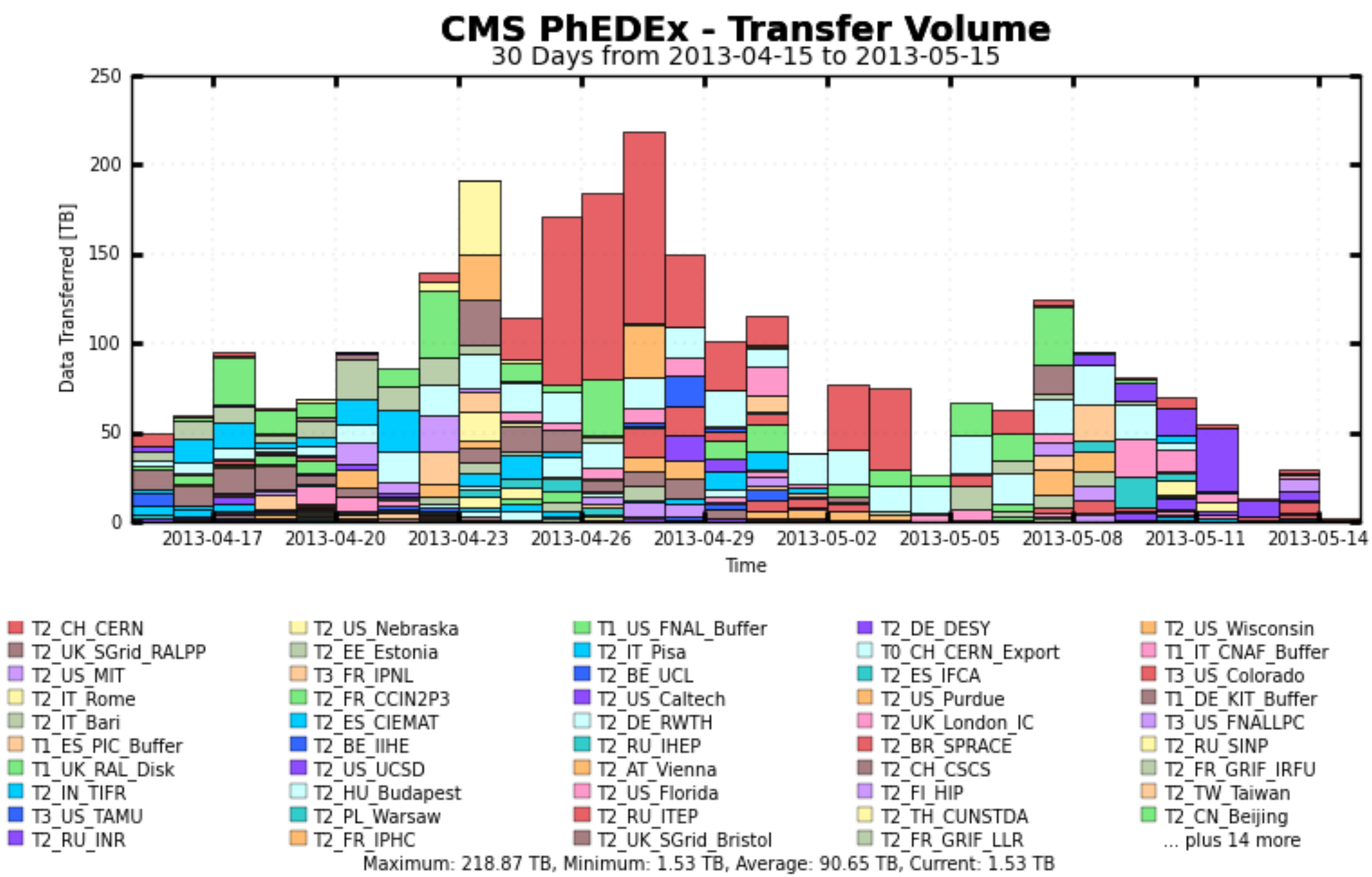
Allows operators/sysadmins to spot jobs causing problems, for example the origin of excessive load

Data outgoing via xrootd by site - Past Month

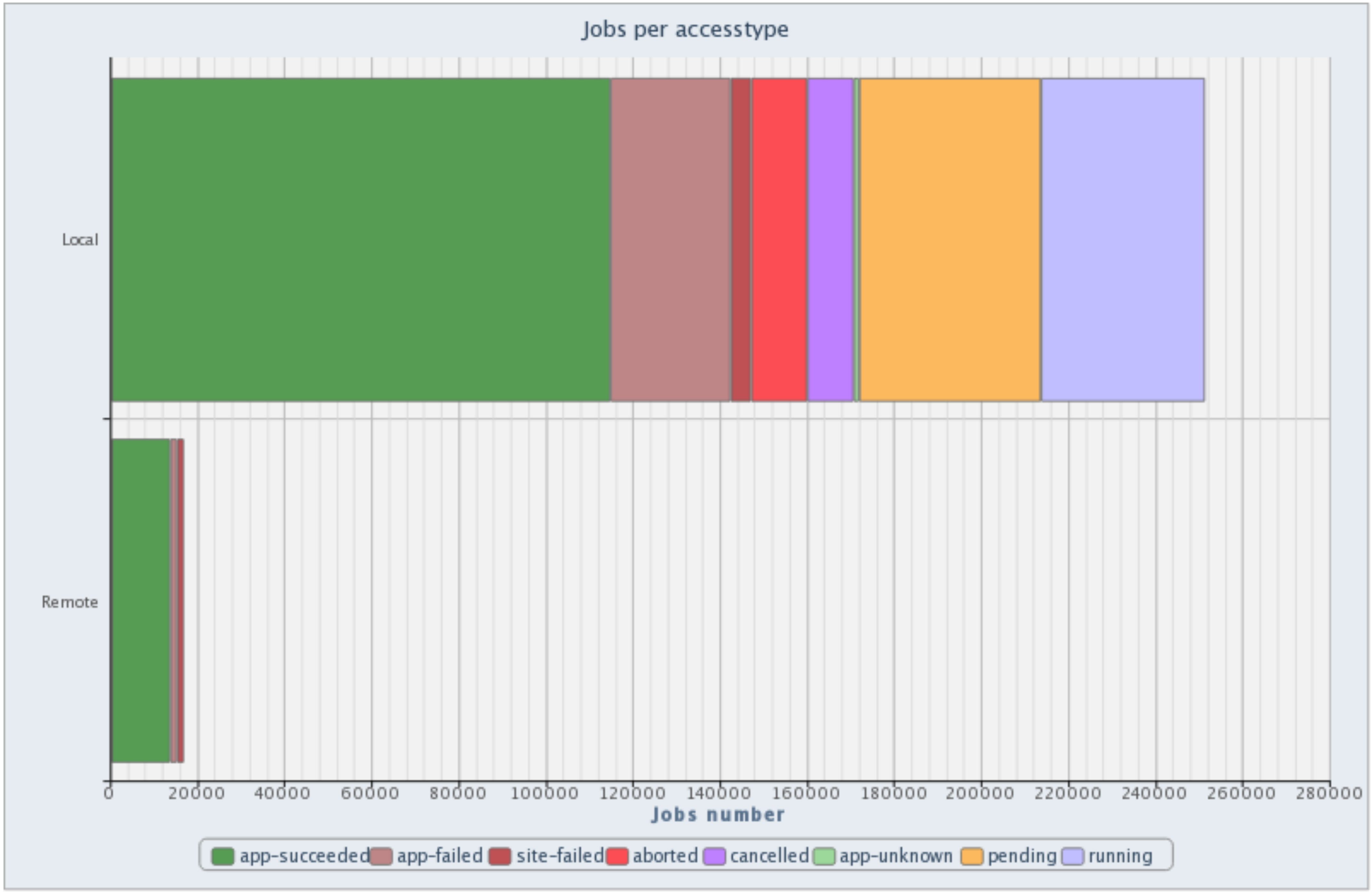


Dominated by a few sites at the moment

Standard PhEDEx Data Transfers - Past Month



CMS Jobs by Access Type



Daily Summary Report for Operators

```
==========================================================
  Xrootd 2013-05-02 | 189.89 TB | 1% increase
==========================================================
```

Source Site	Volume GB	# of Transfers	Yesterday Diff	One Week Diff
CMS Xrootd Site Unknown	966	2,987	29%	2165%
GLOW	2,164	2,920	-35%	-79%
GLOW_Internal	145,069	93,343	22%	79%
MIT	378	6,793	14%	-90%
Nebraska	2,300	11,940	-42%	-90%
Purdue	327	2,670	-77%	-3%
T1_FR_IN2P3	19,227	37,739	16%	237%
T1_IT_CNAF	350	1,585	15%	-74%
T1_UK_RAL	58	332	-40%	-85%
T2_IT_Bari	3,325	10,504	310%	-2%
T2_IT_Pisa	369	793	105%	-61%
UCSD	1,872	5,956	27%	1917%
UFL	98	2,018	-10%	-78%
USCMS-FNAL-WC1	13,219	8,542	-66%	5%
Vanderbilt	160	3,370	-13%	361%

Quick ascii-art/table at-a-glance morning summary of system statistics from the previous day

Summary

- The CMS Computing Model relies on static data placement, job movement to the data and local data access. This simple model has served us well during commissioning and the first LHC run.
- Over the past 1.5 years we have been deploying and commissioning a system, based on xrootd, allowing for WAN access to data by jobs and building a global data federation from heterogeneous resources.
- The system has been growing steadily and we expect that it will be a critical piece of our computing system for the next LHC run

Proper Credits

- UCSD: Frank Wuerthwein, Matevz Tadel, Igor Sfiligoi
- UW: Dan Bradley, Sridhara Dasu
- UNL: Brian Bockelman, Ken Bloom
- The xrootd team: Andy Hanushevsky. Lukasz Janyst, Andreas Peters, Gerri Ganis, et. al.
- dCache implementation: Gerd Behrmann
- Many T1/T2 sysadmins who deployed the system