ATLAS and CMS Data Management Tools and Federated Data Store Implementations

Daniele Bonacorsi

[University of Bologna, Italy - deputy CMS Computing coordinator]

[Credits for material and images are given slide by slide, wherever possible, far from exhaustiveness]

http://indico.cern.ch/conferenceOtherViews.py?view=lcg&confId=215393

Outline

Data Management

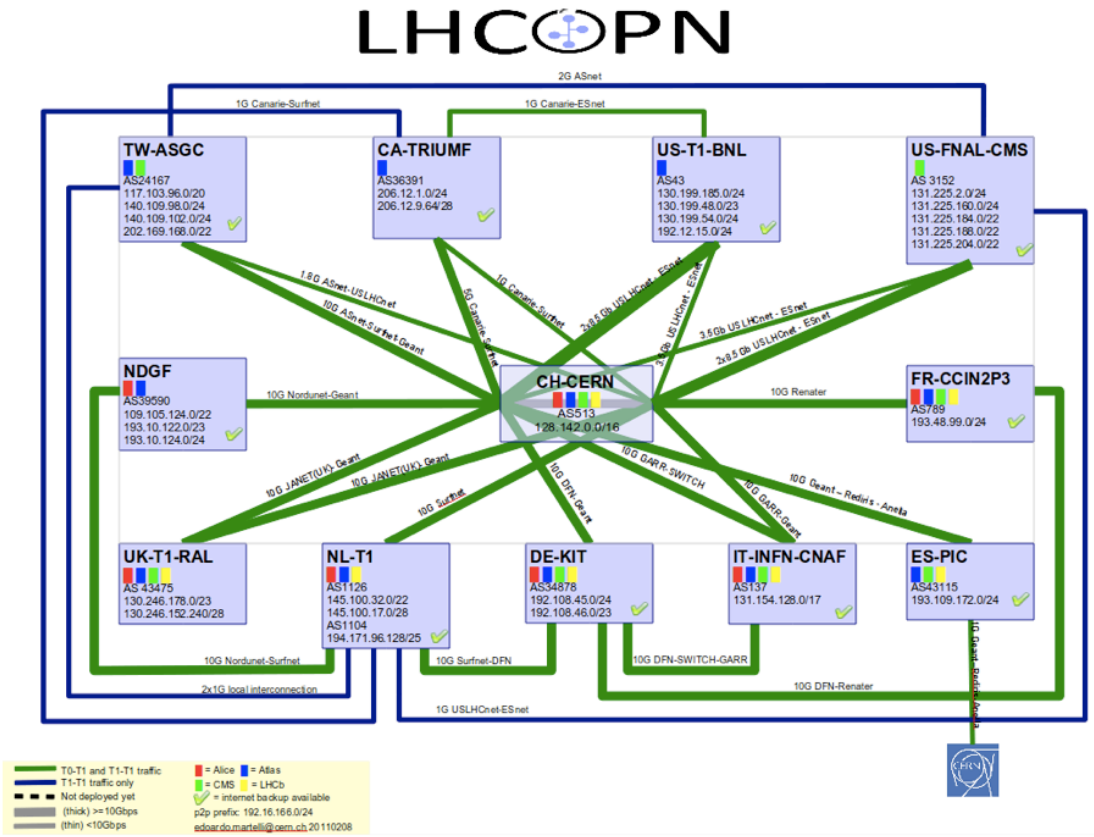
- ATLAS
- CMS
- commonalities and differences
- network-awareness?

Federations

- CMS
- ATLAS
- networks?

[some references throughout the slides]

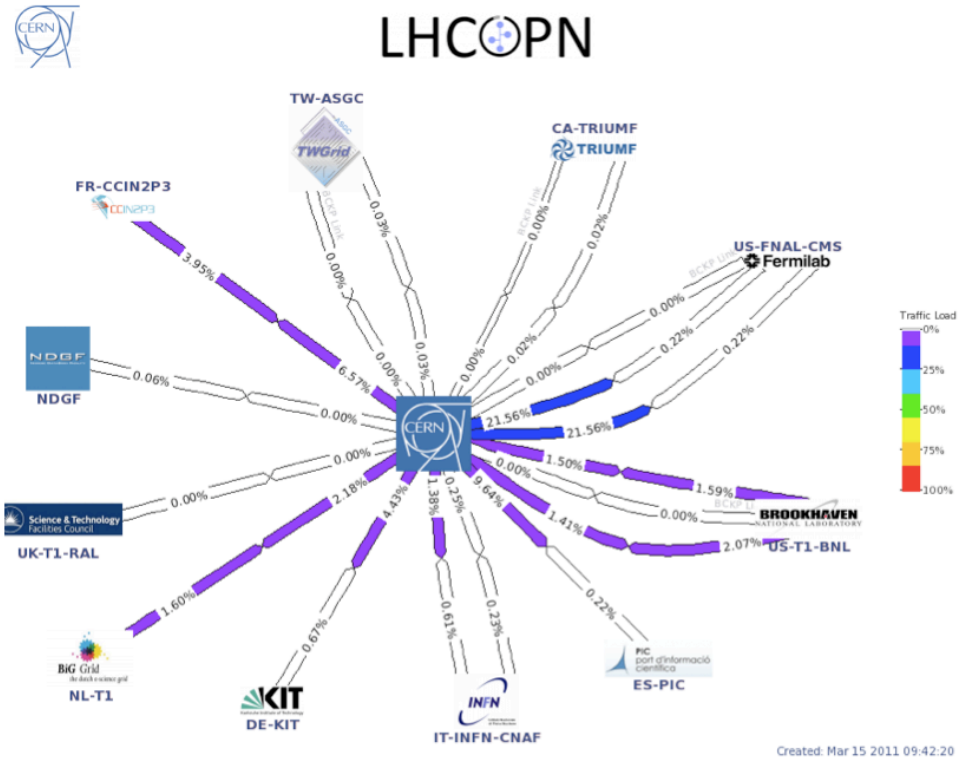
Networks



Probably our most reliable service

- OPN links now fully redundant
  - Means no service interruptions

Excellent monitoring systems



Are ATLAS and CMS DM sectors so different?

ATLAS and CMS have similar needs and deployed similar solutions:

- in the ATLAS Distributed Data Management (DDM) project, Don Quijote 2 (DQ2) is used to handle transfer tasks, exploiting a file-level transfer service
- in the CMS Computing project, the Physics Experiment Data Export (PhEDEx) system is used to handle transfer tasks, exploiting a file-level transfer service

The file-level transfer service behind is the same for both

- the gLite (now EMI) File Transfer Service (FTS) handles the vast majority of ATLAS/CMS transfers worldwide

The two experiment-specific DM systems on top are still different, though. Why?

- experiments and dataset definitions were and are different
- experiments grew up with Grid middleware: sometimes slower, sometimes faster

Data Management commonalities

Basic concepts which are common:

- jobs run where data is, and importance of data placement
- high-level dataset replication systems that use low-level transfer systems
- need of LFN-to-PFN mappings and replica catalogues

But they differ in the “details” (and guess where devil is..)

- See next slides

Brief overview, and then back to comparison.

DM

- ATLAS
- CMS

ATLAS DQ2: concepts

ATLAS Distributed Data Management (DDM) project since 2005

- charged with managing ATLAS data on the Grid
- current DDM system is Don Qijote 2 (DQ2)
  - developed on the basis of ops experience with previous tools
- DQ2 in ATLAS handles all file-based data, it is the tool for defining and handling datasets
  - design objectives were to meet scalability, robustness, flexibility needed by ATLAS to manage the complete data flow, from RAW data handling, through globally managed production, down to individual physics analyses
    - from the ATLAS Computing TDR: "(...) *The scope of the system encompasses the management of file-based data of all types (event data, conditions data, user-defined filesets containing files of any type.*"

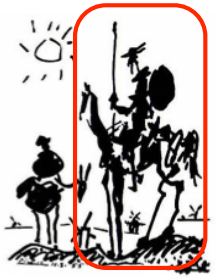
As from the DM concepts applied in DQ2 [1,2,3]:

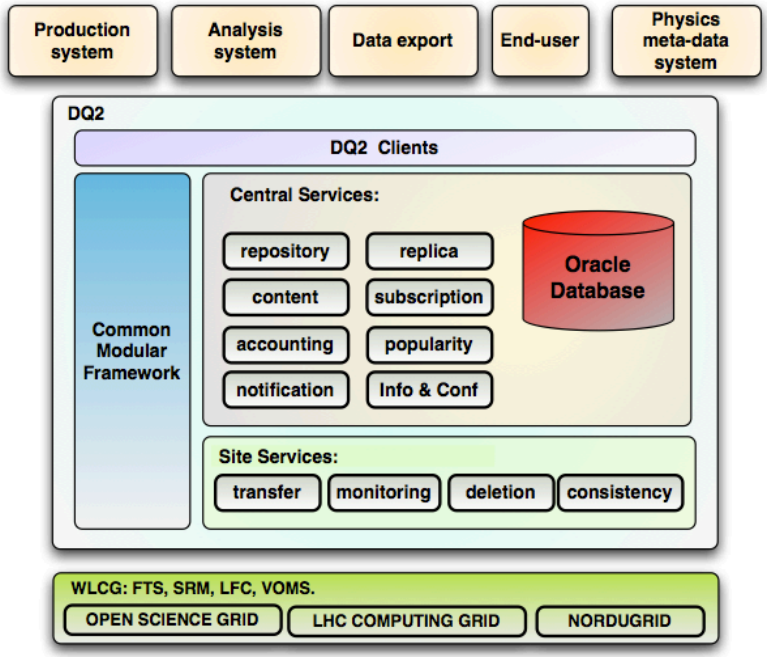
- a **dataset** is an aggregation of data that serve collectively as input/output of a computation process
  - it provides the granularity of data handling that users typically manipulate, and the unit for bulk data management
- dataset constituents are **files**
  - overlaps are possible
- datasets live in a **flat namespace**, they have a **version**, and a **mutability state**
  - *open* = the latest dataset version is open (new data can be appended, or existing data portions removed)
  - *closed* = the latest dataset version is closed, new data added results in a new version being opened
  - *frozen* = the latest dataset version is closed, no new versions can be added (i.e. immutable)
- can be aggregated into **container** objects

Many of these concepts have been revamped within Rucio (see next slides).

[1] "The ATLAS Distributed Data Management project", V.Garonne et al, CHEP'12
[2] "Managing ATLAS data on a petabyte-scale with DQ2", J.Phys.Conf.Series 119 (2008) 062017
[3] "DQ2 - Data Distribution with DQ2 in ATLAS", K. Leffhalm, DESY, 2008

ATLAS DQ2: implementation





"The ATLAS Distributed Data Management project", V.Garonne et al, CHEP'12

System architecture

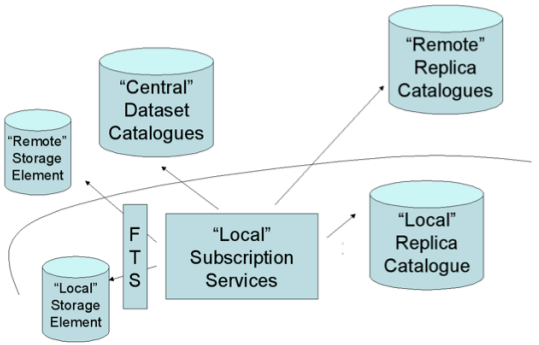
- a combination of central bookkeeping services, distributed site services and agents over a foundation of basic file handling Grid middleware
  - internally to DQ2: clients, central services, site services well separated

Central services and DQ2 clients

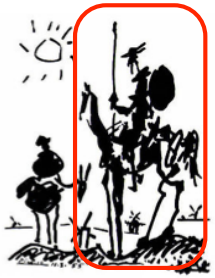
- HA architecture, multiple stateless FEs, Oracle RAC for BE DBs
  - The core is several central catalogues (databases) storing information about datasets
    - **Content** Catalog: datasets content in terms of files (GUID, LFN, size, cksum, ...)
    - **Repository** Catalog: datasets metadata (name, owner, creation time, state, versions, ...)
    - **Location** Catalog: the "replica catalogue" (versions aware)
    - **Subscription** Catalog: user subscriptions for transfers and their status.
- interfaces provided via CLI and python API

Site Services

- autonomous agent-based frameworks with concurrent agents acting on central DBs
- Site Services installed on VO-boxes at CERN, one VO-box per "ATLAS cloud"
- Examples of agents:
  - Fetcher: polls Subscription Catalog and finds subscriptions to a site
  - Replica Resolver: finds and chooses a source among all replicas available as from the Location Catalog
  - Partitioner: hand-shaking with FTS, polls FTS channel status, dimension requests accordingly
  - ...



ATLAS DQ2 in production



Successfully serving the ATLAS needs

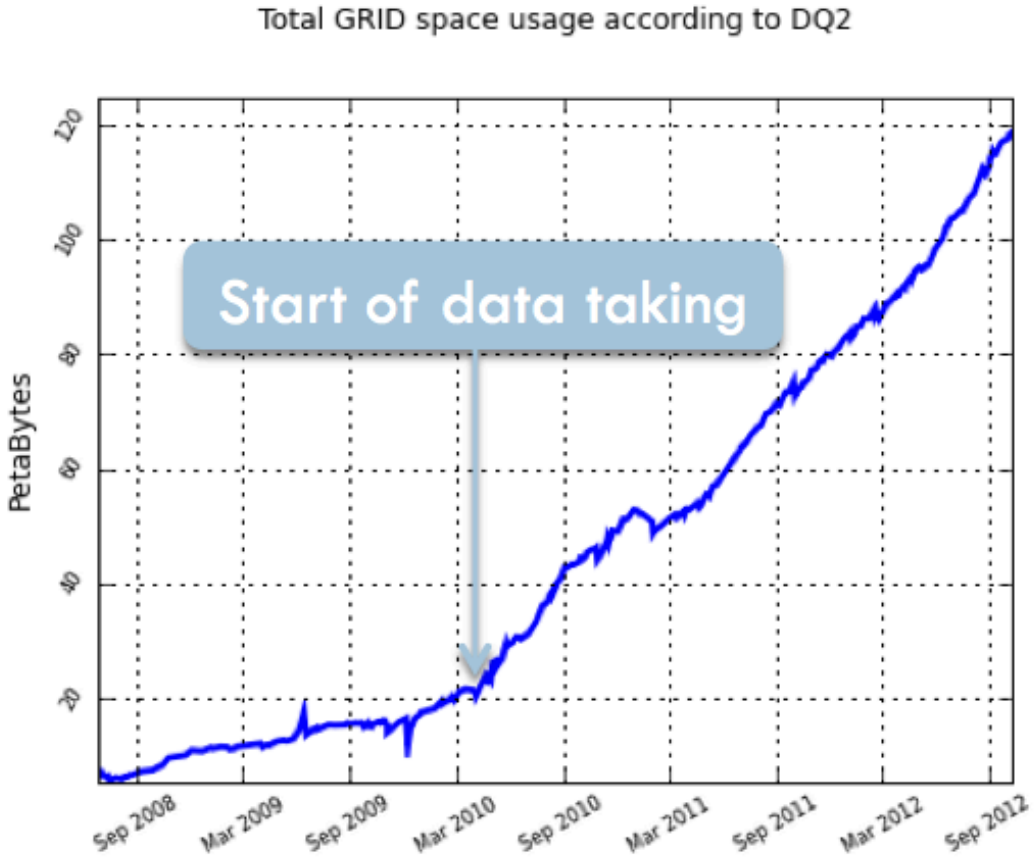
- managing datasets and files at the required scale
- in operations, constant interaction with WLCG sites

DQ2 in numbers:

- **130** PBs
- **600k** datasets, **355M** files
- **130** sites (**700** endpoints)
- **800** active users

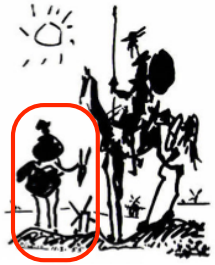
- Grid file accesses: **5M** /day
  - linear increase (factor 2 /yr)

- Central Services
  - **12M** read queries /day
  - **0.6M** write requests /day



[1] "The ATLAS Distributed Data Management project", V.Garonne et al, CHEP'12
[2] R.Vigne, M.Lassnig at the pre-GDB on storage interfaces and access, Oct 2012
[3] "Rucio", V.Garonne, this week's ATLAS Jamboree

ATLAS DDM evolution: Rucio



Lessons learned in DQ2 operations

- successful operations but very manpower-intensive
- not easily scalable, due to complex components interaction
- difficult to extend (i.e. add new features, adapt to new technologies)
  - originally designed for SRM + FTS

Moving towards the next-generation DDM system in ATLAS: **Rucio** [1,2]

- address the items above
  - i.e. ensure scalability and adaptability; reduce operational overheads; support new ATLAS use-cases; use free, open, and standard technologies
- Timeline
  - 2011: user surveys, technical meetings, use-cases collections, first conceptual model
  - 2012: development, pilot server delivered in November 2012
  - 2013: functional testing, gradual migration DQ2→Rucio, and external applications also (e.g. PanDA)
  - 2014: in production at the end of LS1

In the context of this workshop?

- the Replica management is interesting for example...
  - See later

[1] private communications on Rucio conceptual model
[2] R.Vigne, M.Lassnig at the pre-GDB on storage interfaces and access, Oct 2012

CMS PhEDEx: concepts

CMS data is placed at Computing Tiers according to experiment policies

- Both RAW data from the detector and subsequent re-processed data
- Event data are in **files**, grouped in **fileblocks** (for bulk DM reasons), grouped in **datasets** (whose definition is physics driven)
  - $\mathcal{O}(10^6)$ files/yr (≥ GB); $\mathcal{O}(10^5)$ blocks/yr (1-10 TB); datasets are ~0.1-100 TB

CMS uses the Physics Experiment Data Export (**PhEDEx**) system [1,2,3,4,5]

- a reliable and scalable dataset (fileblock-level) replication system
- developed by CMS physicists and software engineers, not CMS-specific though
- the most long-lived production-quality Computing tool in CMS (dates back to 2003/04)

Design goals

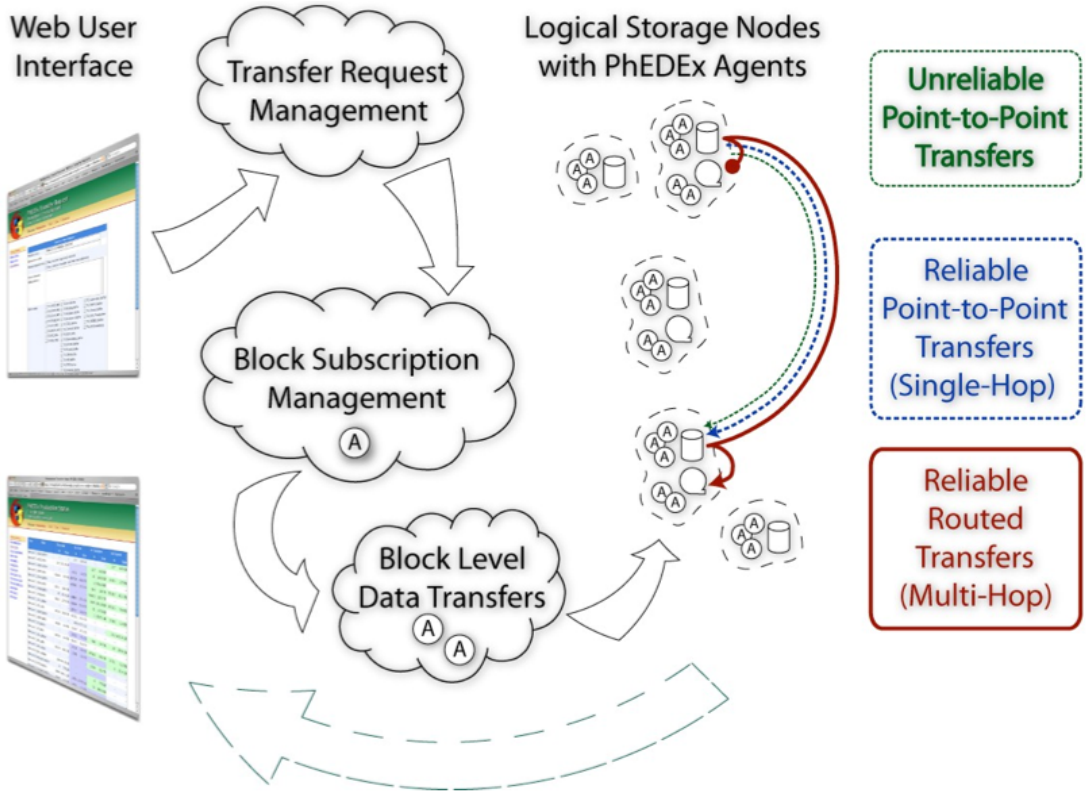
- Reliability
  - robustness & self-healing capabilities, integrity of replicated data
- Flexibility
  - deal with different transfer priorities/queues, support of any transfer protocols & storage systems
- Scalability
  - adequate for the LHC data taking needs
- Solidity of monitoring
  - solid and consistent, user-friendly

[1] "Performance studies and improvements of CMS Distributed Data Transfers", R. Kaselis, CHEP'12
[2] see several contributions by T.Wildish to CHEP'12
[3] "Scaling CMS data transfer system for LHC start-up", L.Tuura, CHEP'07
[4] "The CMS data transfer test environment in preparation for LHC data taking", N.Magini, IEEE NSS-MIC 2008
[5] "Techniques for high-throughput, reliable transfer systems: break-down of PhEDEx design", D.Bonacorsi, CHEP'06

CMS PhEDEx: implementations

Architecture in a nutshell

- a set of loosely-coupled, stateless, highly-specialized software **agents**
  - historically at sites, but today could be run elsewhere
  - each designed to perform a specific “simple” and insulated task in a reliable way
    - e.g. routing, actual replication, file checks, tape migration, new files injection, ...
- Agents inter-communicate with a transfer management database (**TMDB**)
  - all agents inter-communicate with this central blackboard, central at CERN
  - block replica location and subscriptions, file metadata information, transfer states, ...



Successfully met the requirements

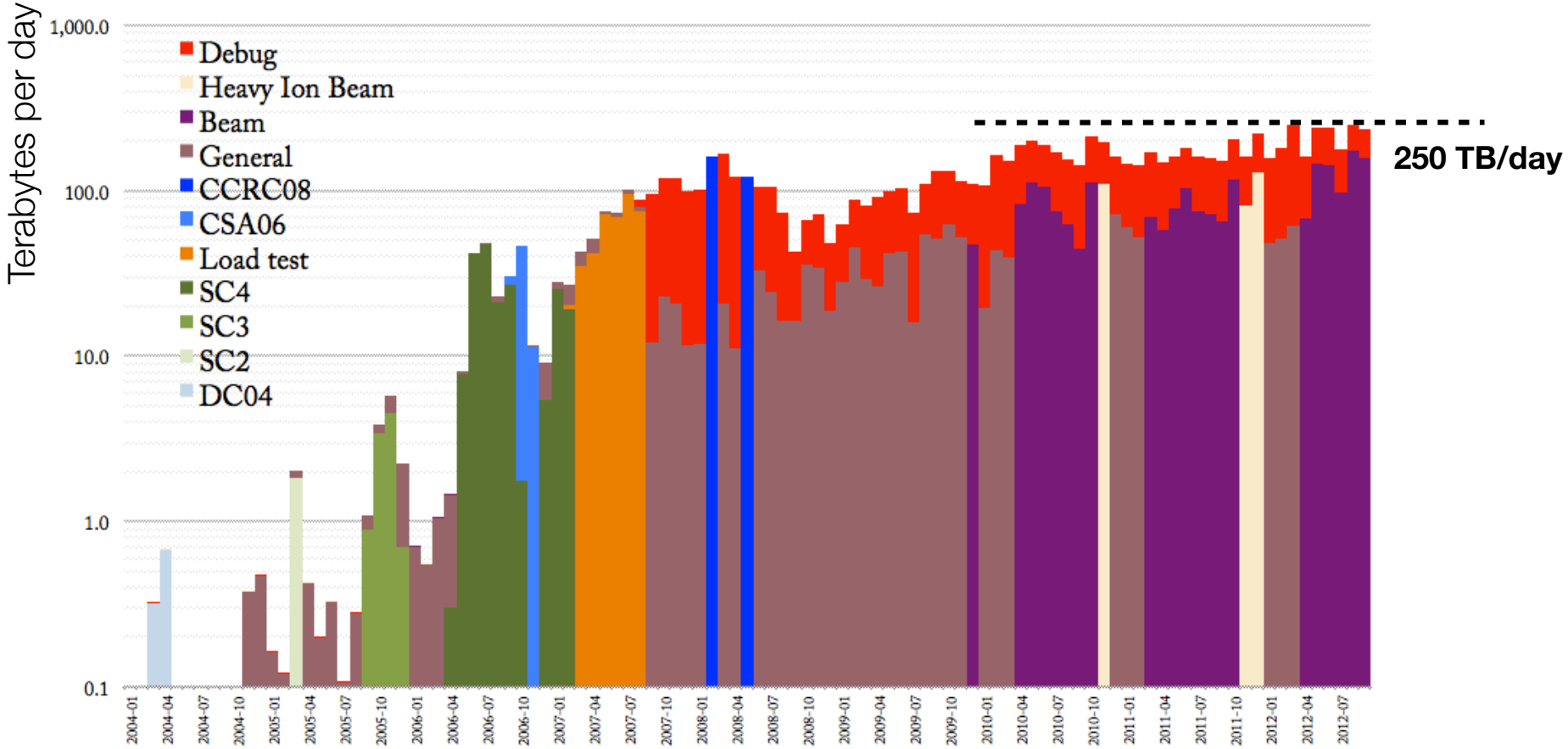
- Reliability: designed under assumption that any operation might fail
  - efforts in monitoring, size and cksum checks, automatic cool-off for failed transfers, self-healing, solid error recovery, fail-over tactics
- Flexibility: agnostic in most of the non-core choices
  - no limits in the variety of adoptable file-level transfer mechanisms, intelligent routing with fall-back mechanism
- Scalability: well designed schema + highly-optimized queries + Oracle RAC
  - agents in themselves are extremely thin and stateless, demonstrated to scale well beyond the LHC needs, very rare service interruptions over years (including critical data taking times and under any load conditions)
- Monitoring: tailored around users’ and system managers’ needs
  - solid web interface, offering user-oriented and customizable views of transfer rates, queue volumes, transfer quality, detailed error log, status of all components, ... (both real time and historical)

CMS PhEDEx in production

After a massive commissioning effort, PhEDEx served CMS very well in LHC data taking times

- volume: **~250 TB/day** among dozens of Tiers (~**500k** transfer successes /week)
- # files: **~19M** logical files (but total of replicas so far is **~27M**)
- throughput: **2-2.5 GB/s** aggregate (weekly averages) in peak weeks in 2012

CMS average data transfer volume (2004-2012)



Data management

- commonalities

Differences in DM implementation choices (or status)

ATLAS handles a larger **number of files** in the DM system than CMS

- ATLAS injects all files in their system, including in particular all user file. As a result, ATLAS transfers many more files with DDM than CMS does with PhEDEx, and have a lot of small files in the system

In ATLAS **a file can belong to multiple datasets**, while in CMS the datasets contain unique files

- CMS deletes on the dataset level or data block level, while ATLAS deletes on the file level

ATLAS and CMS use different **catalogues** for mapping purposes

- ATLAS currently uses a central catalog (LFC) to perform LFN-to-PFN mapping. This is going to change in Rucio during LS1, it will use implicit LFN-to-PFN conversion rules, more similar to what CMS does

ATLAS is going towards a less MONARC-hierarchical and "full **mesh**" transfer model like CMS

- ATLAS uses a more hierarchical model of T0→T1→T2 transfers still, similar to what CMS had in ~2008. Many T2s are still restricted to sharing data only with their "regional" T1, though now ATLAS is also moving to a "full mesh" like CMS and there are also many T2s that transfer data to/from any T1 (so-called "T2D"s). Following from the previous point, ATLAS has enabled multi-hop transfers to get data from T2_X to T2_Y through some T1

CMS is now moving to **disk vs tape separation** like ATLAS (and others) already did

- CMS has not yet finalized a full separation among disk and tape resources at the T1 level, still sending most of the data to tape backend. ATLAS has separated "T1 Disk" from "T1 tape" through space tokens a long time ago

ATLAS system is somehow more "**dynamic**" than CMS, soon doing something similar

- A large fraction of ATLAS data placement is automated, with dynamic subscription and deletion of datasets based on data popularity rather than relying on human action on requests. CMS has equipped their system with data popularity evaluation mechanisms, and will soon (2013) enable deletions, but so far the system is less "dynamic"

ATLAS operates systems more **centrally**, CMS has a **distributed** deployment of components

- ATLAS DDM is operated centrally at CERN for all sites, talking to the site SEs only with "remote" tools (i.e. SRM, FTS). CMS PhEDEx is a distributed transfer system with central agents at CERN and site agents at each site, managed by the CMS contacts on site (despite it has been demonstrated years ago that technically all agents could be run elsewhere)

Not all items are relevant to the topic of this workshop, but a couple are.

Another bit in common

- the underlying file-level transfer service

gLite File Transfer Service

File Transfer Service (FTS) is a low-level data movement service

- designed for point to point movement of (sets of) physical files between SEs
- multi-VO by design, it allows sites to control the network resource usage
  - via dedicated interfaces (+ display statistics of ongoing transfers)

Concepts in a nutshell

- FTS performs file transfers on **channels** among “sites” (SRM and gsiftp endpoints)
  - a channel is a network “pipe” for transferring files between two endpoints
  - STAR channels are foreseen (i.e. not dedicated, but shared with other applications)
- an FTS **instance** serves a configurable set of channels
- the channels are **unidirectional** and their configuration is **mandatory**
  - different instances may intentionally serve opposite directions of a network link between two sites
- typically, **transfer jobs** are submitted to the service
  - asynchronous bulk transfer requests, now served via third-party gridFTP of SRM Copy
  - each comprises a list of files, a set of config parameters (e.g. gridFTP parameters for TCP buffer size and # streams), a security token
- **deployment** suggestions originally indicated by developers, and consolidated over time
  - typically, FTS servers are deployed at (large) sites where there are (large) amounts of data to be transferred
- **monitoring** of FTS channels status and performances
  - via FTM, plus FTS also publishes messages to a message bus for every transfer and a periodic message for channel statistics, ratios of active/ready transfers, etc

FTS roadmap [1]

(...)

FTS version 2.2.4 (project: gLite 3.2) [2]

- bug-fix version

FTS version 2.2.6 (project: EMI1) [3]

- support of transfers between source elements not supporting SRM interface
  - to transfer between SRM-less endpoints, the service must be properly published in BDII [4]
- support of restarting / partially-resuming failed transfers

FTS version 2.2.8 (project: EMI1) [5]

- support FTS monitoring messages
- change FTS file overwrite logic
- support for gridftp endpoints
- Oracle 11g integration
- nagios probes for FTS

FTS version 2.2.9 not out yet (roadmap: [6])

- bug-fix version

(...)

FTS 3 [7]

[1] https://svnweb.cern.ch/trac/glitefts
[2] https://svnweb.cern.ch/trac/glitefts/wiki/FTSRelease_2_2_4
[3] https://svnweb.cern.ch/trac/glitefts/wiki/FTSRelease_2_2_6
[4] https://svnweb.cern.ch/trac/glitefts/wiki/PublishGsiftpInBDII
[5] https://svnweb.cern.ch/trac/glitefts/wiki/FTSRelease_2_2_8
[6] https://svnweb.cern.ch/trac/glitefts/roadmap
[7] https://svnweb.cern.ch/trac/fts3

FTS 3

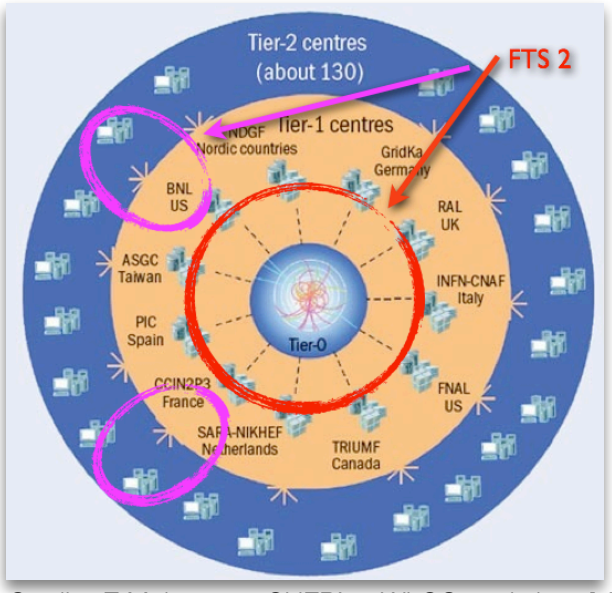
FTS 2 has served LHC well by transferring the large bulk of LHC files

- ATLAS and CMS (not alone) have interfaced their high-level data management systems to use FTS2 as the file-level transfer system

FTS 3 announced [1,2] as the new generation WLCG transfer job scheduler for scientific experiments handling large amounts of data

FTS2 itself has been somehow design around the MONARC model assumptions

- focus on T0→T1, T1↔T1 as well as T1↔T2
- i.e. exploit LHCOPN strength and full redundancy, probably making Ops models weaker the more you move to non-T0/T1 levels



Credits: Z.Molnar, pre-CHEP'12 WLCG workshop [2]

The FTS3 project is currently in prototype 1 phase

- installed at CERN, RAL, BNL, ASGC for functional testing purposes

Roadmap foresees a roll at the end of LS1 (more details in [1])

[1] https://svnweb.cern.ch/trac/fts3
[2] http://indico.cern.ch/getFile.py/access?contribId=436&sessionId=4&resId=1&materialId=slides&confId=149557

FTS3 concepts

Main changes seem to be driven by the hierarchy→"mesh" change

- less-structured data placement models grow, Tier levels operationally more similar
- MONARC-like models imply management of $\mathcal{O}(10)$ channels, mesh would diverge

A shift of paradigm towards something scalable in the new realm

- moving away from "channels", "site groups", "STAR channels"
  - pair config → point config
  - channel share → SE, site share
  - channel parameter → SE, site, network parameter
- Done configuring key-value pairs, JSON-formatted for each (group of) SEs, e.g.:
  - {"se":"se.cern.ch","active":true,"in":{"share":[{"cms":5},{"atlas":5},{"public":5}],"protocol":[{"nostreams":10},{"urlcopy_tx_to":3000}]},"out":{"share":[{"cms":6},{"atlas":5},{"public":4}],"protocol":[{"nostreams":12},{"urlcopy_tx_to":3600}]}}
- Deployment choices are wide open
  - just one FTS3 server to rule them all? distributed servers, with overlaps possible?
  - hybrid way, with FTS2-like control kept for T0-T1?

Adding network knowledge?

- dynamic adaptation of config implemented, needs testing
  - FTS3 supports "transfer auto-tuning", i.e. dynamically adjust the number of tcp streams, tcp buffer size, # of active transfers per source→dest based on the success/failure rate of previous transfers and the throughput achieved

Adding SE state?

- FTS3 will be checking BDII/MyOsg info services for the state of a SE before starting a transfer
  - only considered if it exists in any information system

Transfers: network-awareness? [1/2]

Where data management could become network-aware?

Level 1: "*high*"-level i.e. **activity planning**

- in some sense, above both Data and Workload Management
- the planning (e.g. dependencies, completion times, ..) drive workflow scheduling and executions
  - network bandwidth reservation could be triggered in advance based on planning details/needs

Level 2: "*medium*"-level i.e. **transfer "routing"**

- *(NOTE: "routing" here intended at the experiment application level, not at the network level)*
- static subscriptions are executed by selecting the "best" source(s) to a destination
- the choice is now based on internal transfer stats (e.g. transfer rates, failures, .. over last days/hrs)
  - network information could be used instead, or additionally

Level 3: "*low*"-level i.e. **file-level transfer**

- could be at the transfer agent level (e.g. FileDownload for CMS PhEDEx) or indeed the underlying file transfer service (FTS)
- all subscriptions and routing would be done in a traditional, network-unaware manner
  - bandwidth allocation may be triggered when the file transfer service needs to deal with a long transfer queue on a link (e.g. threshold?)

Examples? See next slide.

Transfers: network-awareness? [2/2]

Level 1: "*high*"-level i.e. **activity planning**

- *subscriptions in Rucio may be an interesting candidate for a choice at this level?*
  - replica management based on **Replication Rules** defined on datasets/containers. Each rules is owned by a Rucio "**account**", and defines the minimum # of replicas that have to be available on a Rucio Storage Element (**RSE**), i.e. a storage space with attributes. RSEs can be grouped in logical ways (e.g. CLOUD=US, or Tier=1). Accounts manage (and are charged) for their own data with replication rules defined on datasets/containers and lists of RSEs
  - *Could a translation of such a rule into a concrete list of transfer tasks be engineered to be optimized on the basis of network-aware information? (e.g. naively: "choose the source RSE with best connection to the destination RSE"?)*

Level 2: "*medium*"-level i.e. **transfer "routing"**

- *ATLAS Site Services or PhEDEx FileRouter could use network info at this level?*

Level 3: "*low*"-level i.e. **file-level transfer**

- *e.g. FDT used as the backend in the FileDownload agent in PhEDEx on the /Debug instance on just one link may be an existing proof of concept of a choice at this level?*

Food for thoughts...

Federations

- CMS
- ATLAS

Scaling up *vs* Scaling down

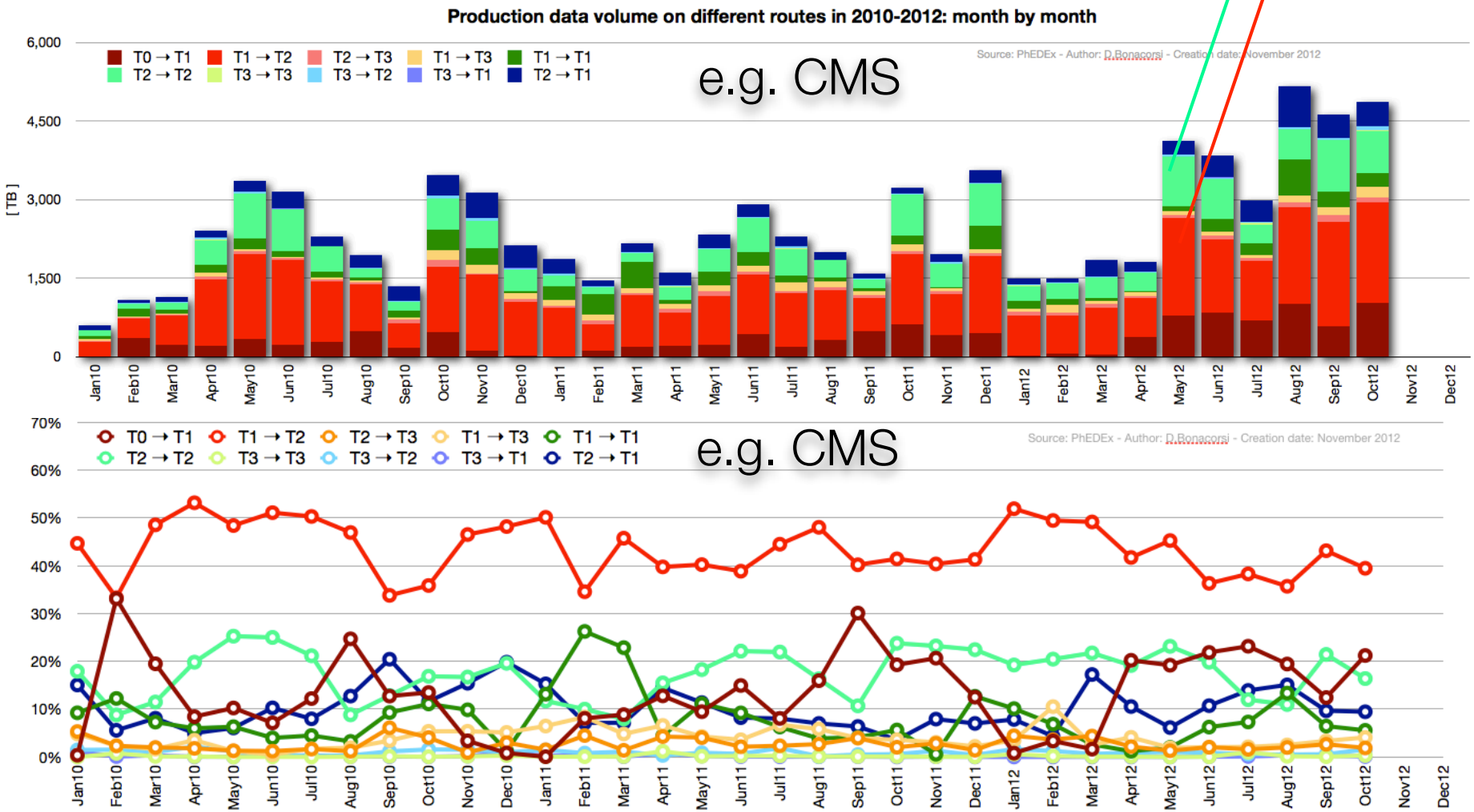
We are successfully *scaling up*...

- to huge transfer volumes and large # jobs concurrently running on Grids

... but are we *scaling down*?

- in terms of usefulness and usability for the physicists doing analysis?

E.g. a lot of "T2 population traffic"
(and T2 is the CMS analysis level...)



Data Federations

WAN access to complement **local** access

- under certain circumstances, a remote access can be an interesting option

Main drivers of this evolution are:

- reduce the "overall" data access latency perceived by CMS analysis teams
- protect analysis jobs from crashing
- eventually reduce the data replication factor?

An "**evolution**" (more than a *revolution*) in ATLAS/CMS data management

- the usage of CMS/PhEDEx and ATLAS/DQ2 just stand as major successes
- the majority of access will remains with local files

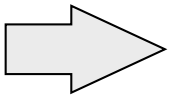
A **Federated store** is:

- *"a collection of disparate storage resources managed by cooperating but independent administrative domains transparently accessible via a common namespace"*

Rolling out **federated data infrastructures**

- Goal is to provide data access to all disk-resident data from a single virtual endpoint ("redirector")
  - regardless of implementation (local storage choices) and location (everywhere worldwide)
- Based on the *xrootd* software suite
  - Federations accessible by any LHC analysis environment (xrootd client included in most distributions of the ROOT framework)
- ALICE pioneered. ATLAS and CMS are working on similar projects.

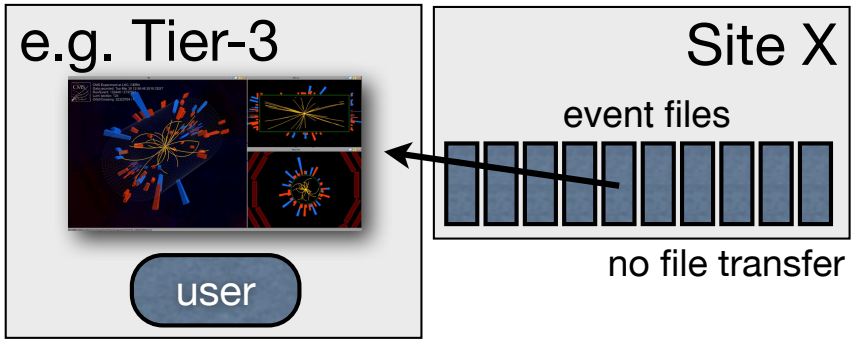
What about the most interesting **use-cases**?



Use cases

Interactive use-case

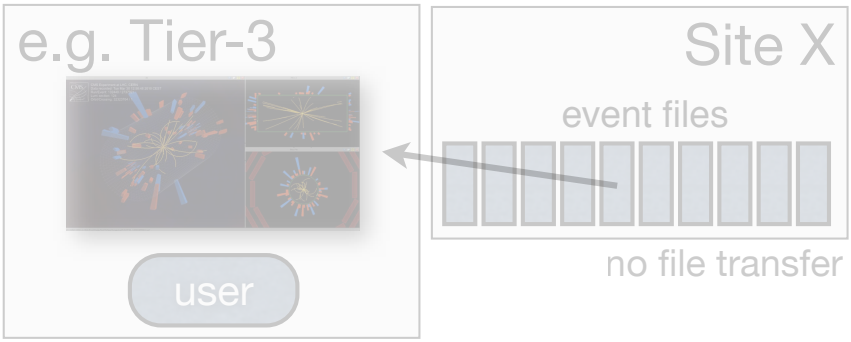
- debugging single events / files
  - hosted remotely, event viewer



Use cases

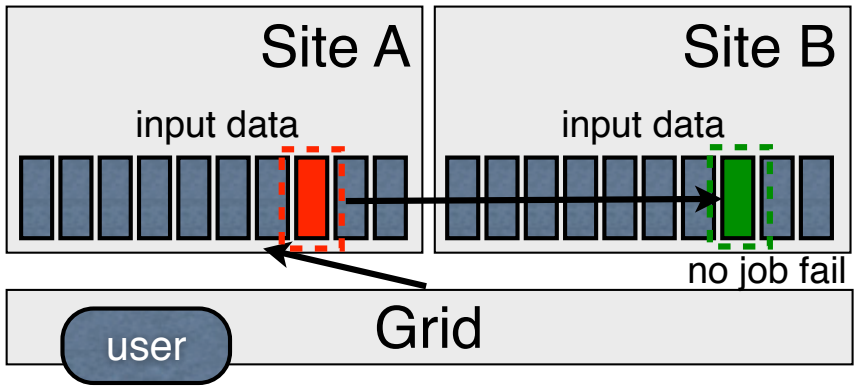
Interactive use-case

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



Fallback use-case

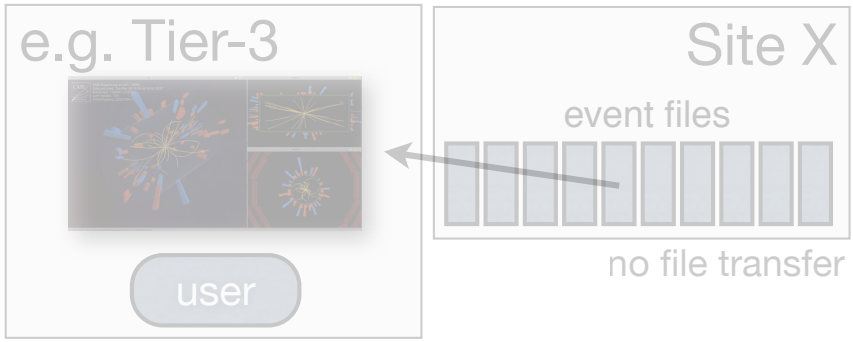
- if a grid jobs fails to open a file, no application failure: have it try again remotely
  - loss of efficiency, but no crash



Use cases

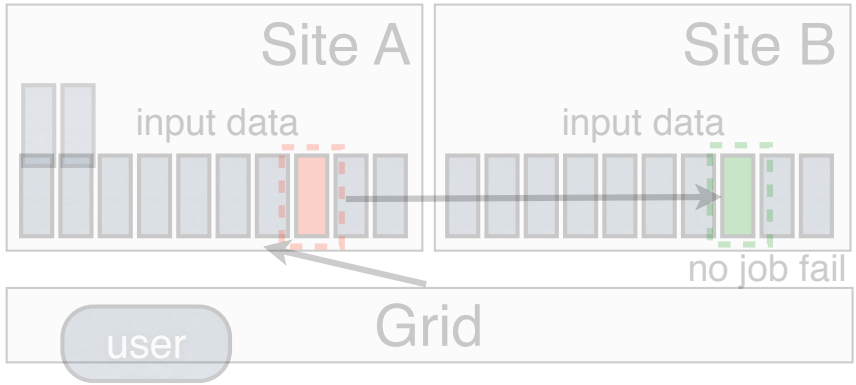
Interactive use-case

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



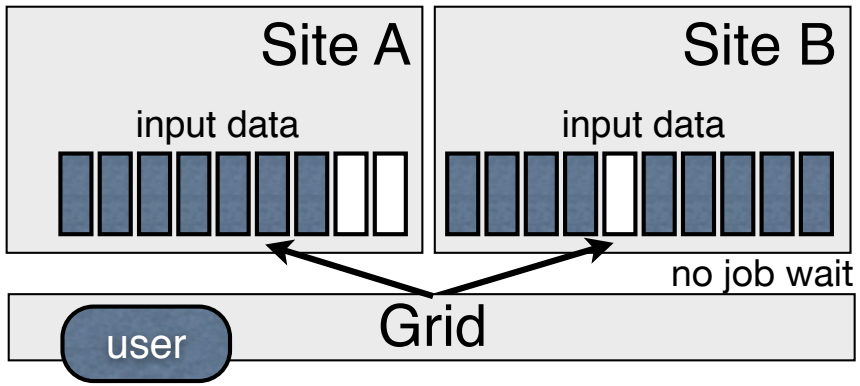
Fallback use-case

- 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



CMS AAA

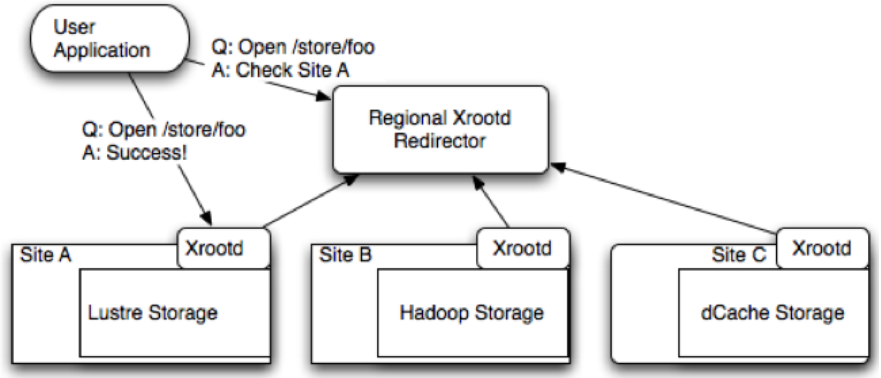
US-CMS pioneered this effort, with the “Any data, anywhere, anytime (**AAA**)” project

Regional approach helpful as a start, to control network latencies

- Same namespace, irrespective of data location
  - TFile::Open(“root://xrootd.unl.edu//store/foo”)

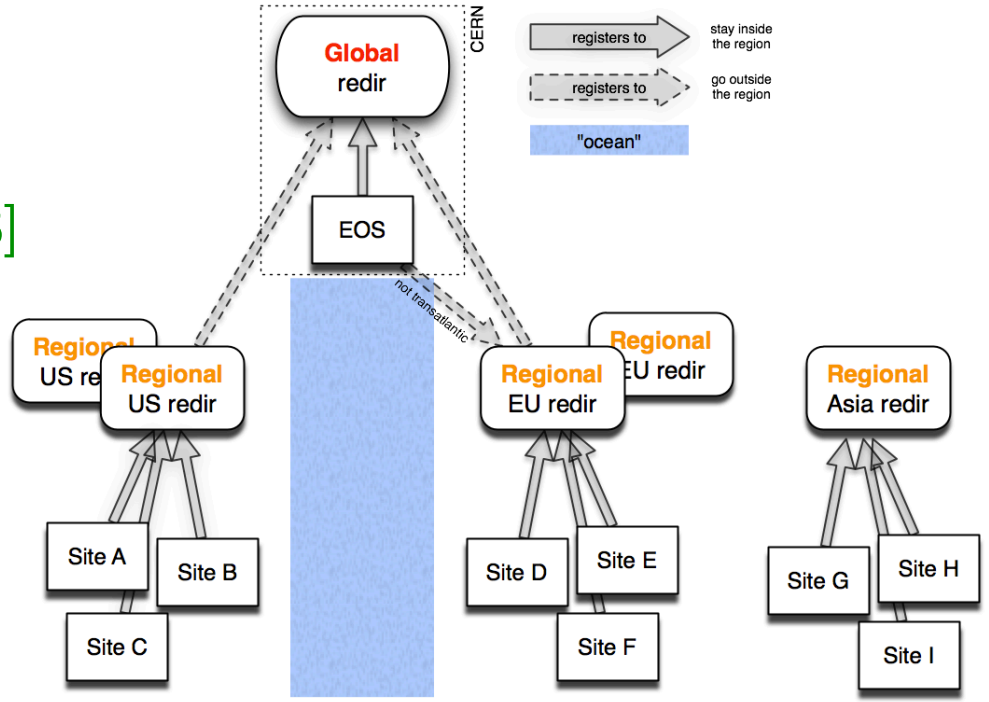
Then, go global, by deployment stages

- add extra-layer(s) of redirectors
  - as operational solidity increase (or driven by needs)



Xrootd-based federations are entering large-scale, day-to-day use for CMS [1,2,3]

- Global and regional redirectors in US and EU, about 15 sites worldwide
- ~10% of the CMS access is now over such infrastructure
- Health monitoring (SAM tests, Nagios probes, ...)
  - Components, Traffic and load, Successful data paths, Versioning and tracking versions
- Support and installation documentation getting more and more solid



[1] “Xrootd Monitoring for the CMS experiment”, B. Bockelman, CHEP’12
[2] “Using Xrootd to Federate Regional Storage”, B. Bockelman et al, CHEP’12
[3] “CMS Storage Federations”, D.Bonacorsi, IEEE NSS-MIC 2012

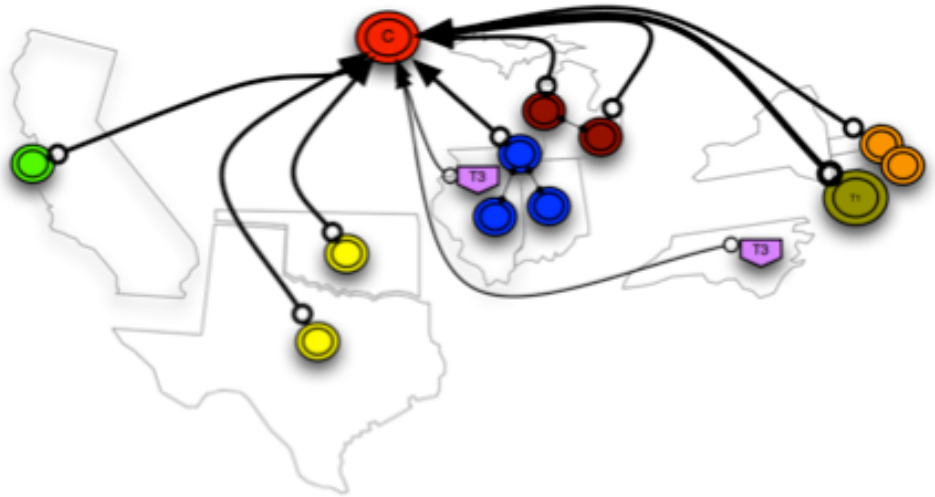
ATLAS FAX

Goals of the Federating ATLAS data stores using Xrootd (**FAX**) project [1,2]:

- common ATLAS namespace across all federated storage nodes
- easiness of use, homogeneity and ubiquitousness of access to ATLAS data
- use as a failover for existing systems
- gain access to more CPUs otherwise free
- explore as caching mechanism at sites to reduce local data management tasks

Status

- from R&D to deployment over US Tiers (see list) in 2011
  - Feasibility testing monitoring, site integrations
- extended to EU sites as an ATLAS-wide project in June 2012



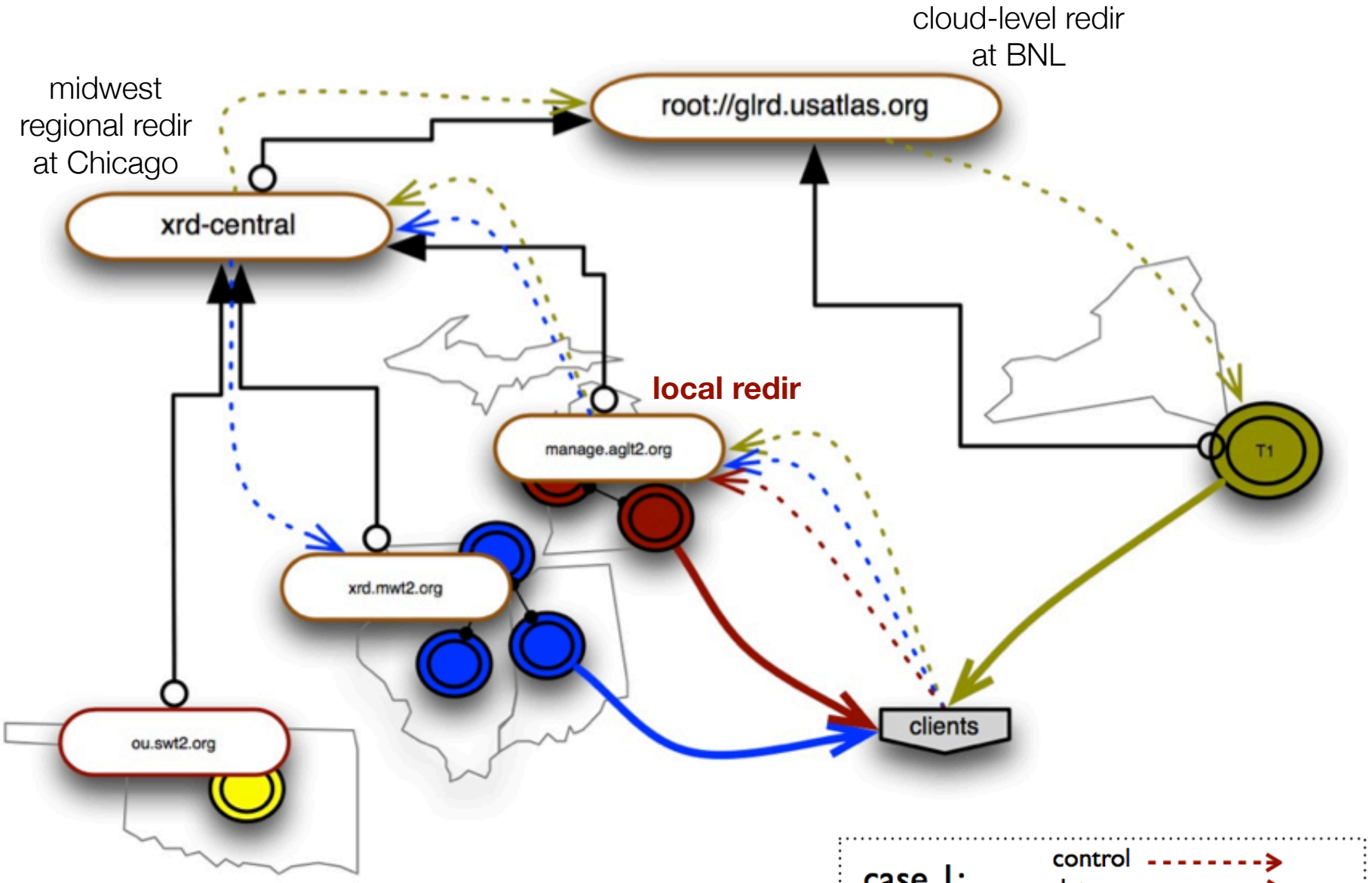
Deployment (as of August 2012 [2])

- US cloud: cloud-level redirector at BNL, midwest regional redirector at Chicago, T3 redirector hosted at BNL
- UK and DE cloud, EU super-cloud and ATLAS global: redirectors set-up at CERN

[1] "Using Xrootd to Federate Regional Storage", B. Bockelman et al, CHEP'12
[2] "Federating ATLAS Data Stores using Xrootd", R.Gardner, WLCG Federations Meeting, Aug 2012

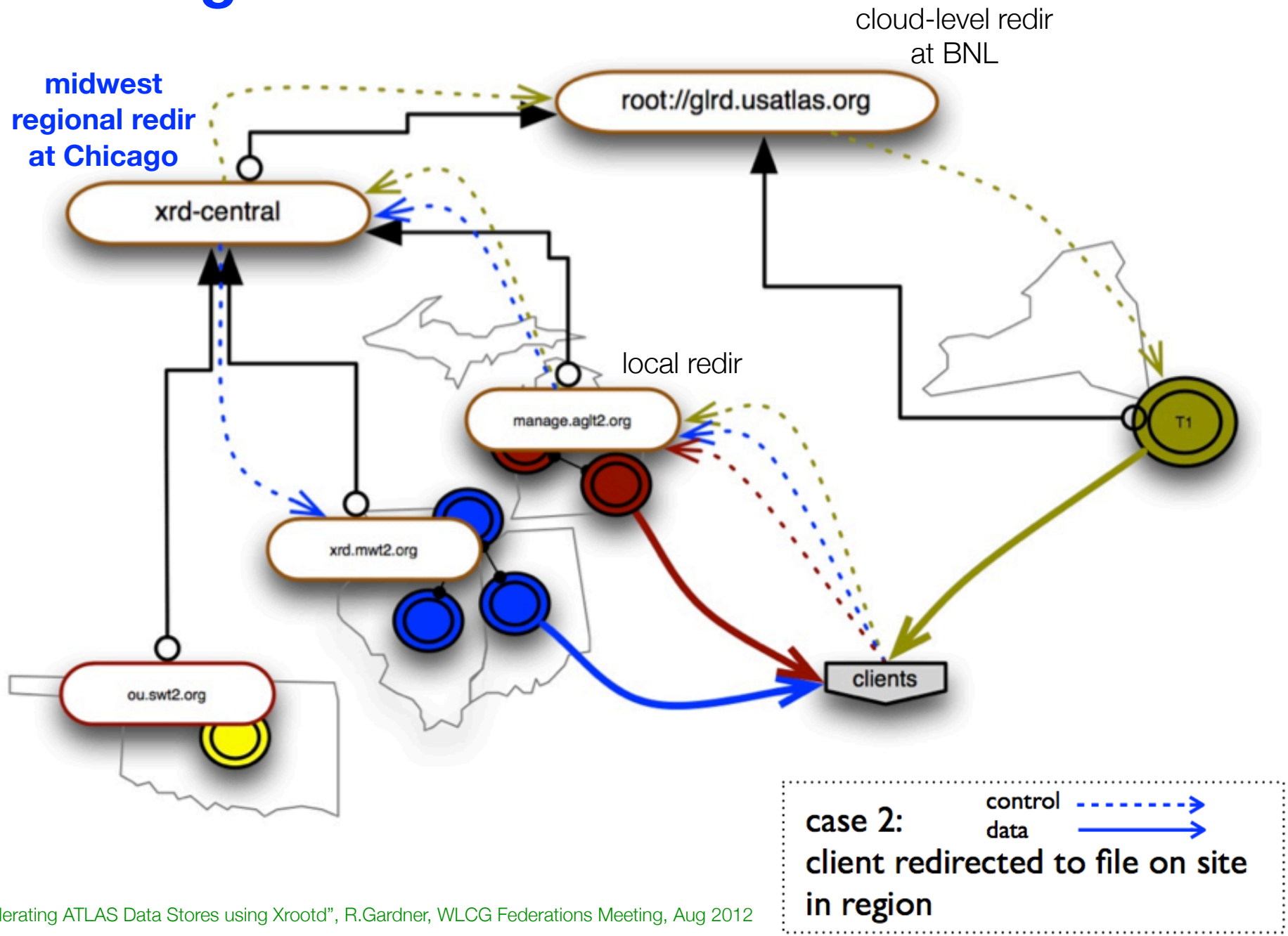
Note: FAX/xrootd is not the only federation being implemented, ATLAS DDM is also going the HTTP/WebDAV redirection route with already existing standard protocols and tools.

FAX local redirection



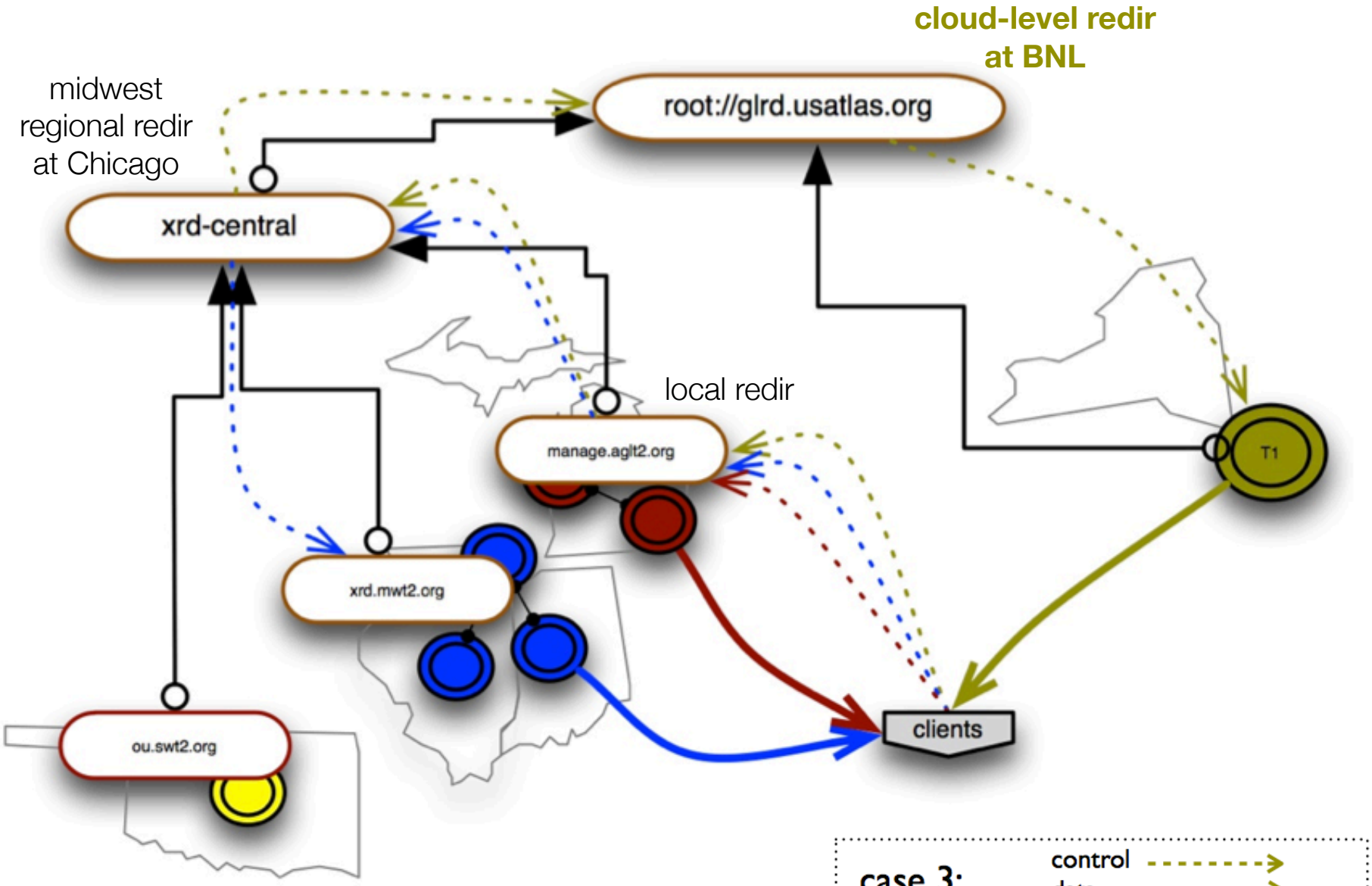
"Federating ATLAS Data Stores using Xrootd", R.Gardner, WLCG Federations Meeting, Aug 2012

FAX regional redirection



"Federating ATLAS Data Stores using Xrootd", R.Gardner, WLCG Federations Meeting, Aug 2012

FAX cloud-level redirection

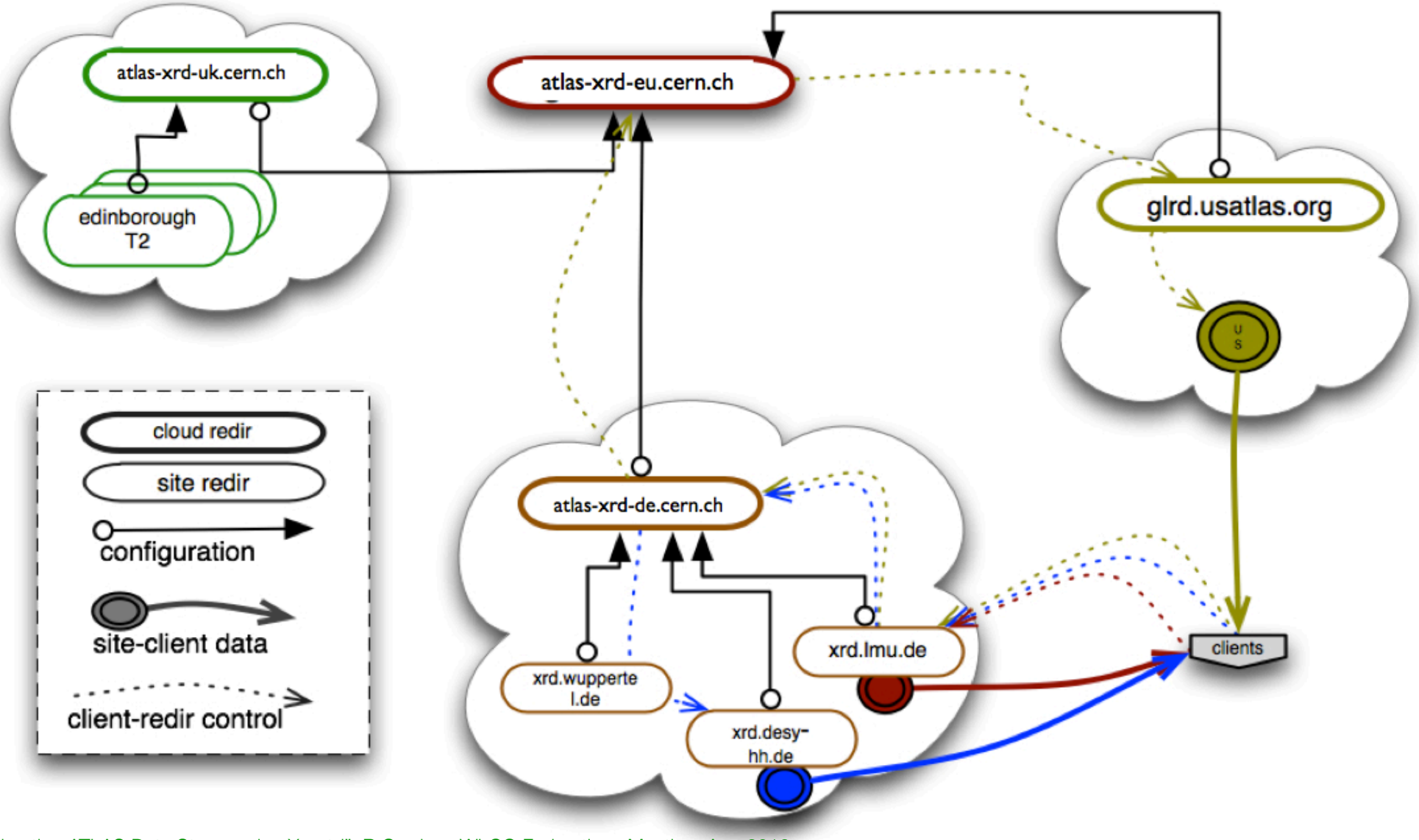


"Federating ATLAS Data Stores using Xrootd", R.Gardner, WLCG Federations Meeting, Aug 2012

FAX cloud to global

As we said:

- UK and DE cloud: redirector set-up at CERN
- EU super-cloud redirector set-up at CERN
- ATLAS global redirector at CERN



"Federating ATLAS Data Stores using Xrootd", R.Gardner, WLCG Federations Meeting, Aug 2012

Thoughts..

Among the main federations prerequisite:

- Remote vs local efficiency is the key
  - The ability to WAN xrootd access is dependent on the application code's ability to efficiently process data over large latency links. Effort by the experiment is a prerequisite to many of the use-cases work
- a pervasive and performant network across the federation nodes
  - how do we want xrootd redirectors point to file location? "network proximity"? This is an interesting point to expand in the context of this workshop

Among the obstacles:

- site integration is a hard job, and networks are part of the difficulty. Apart from inhomogeneous BE technologies at WLCG Tiers (dCache, DPM, GPFS, Lustre, Castor, ...), federations are affected by firewalls (lab, campus, cluster), storage pools on public vs private networks, ...
  - many site-specific configs and a variety of different performances (proxy, DNS balancing over doors, redirections directives, ...). E.g. ATLAS working hard to smoothen the deployment to new sites

What can networks do for experiments and their data federations?

- i.e. having high-performance links between all federated sites would just suffice?
  - 10 Gbps regions report smooth operations with xrootd-based data federations
  - e.g. "give us LHCONE, maintain it like the OPN, allow it to grow" ?

What can experiment do themselves to match network evolutions?

- can we make our workflows more predictable?
  - e.g. give more structure to the redirector-driven traffic flows, and let the network help you out
- there is a sort of "symmetry breaking" over the whole mesh around well connected Tier-2 sites
  - networks should improve to support their outbound traffic more than anything else?
  -

Conclusions

Do ATLAS+CMS need the same in terms of network evolutions?

- it may be not obvious that both would interact to PtP/PtMP scheduling in the same way. They certainly could (probably also should?) but we do need to carefully and pragmatically look to see if we are actually in that position, or if it requires significant work, more than the gain we would obtain e.g. in transfer operations.

But: LS1 is an *excellent* opportunity window

- if not now, when?
- experiments may be interested in trying to evaluate one or more approaches

Which approach? Choice driven by:

- the level of pragmatic and productive interactions with the Network community
  - the WLCG Network group is now having a productive and positive role in this crucial interaction
  - start from experiments needs: we can't consider what to build until we know what experiments want
- the quality, cost, features of the interfaces offered by the Network experts
  - e.g. is what is being offered adequate/convenient for the experiment transfer operations needs?
  - e.g. when is one experiment accounted for the use (or lack of use) of a preallocated bandwidth?
- the ATLAS/CMS (and not only!) manpower available to perform meaningful tests
  - Computing Models evolution and optimization comes through investments, not for free

This workshop is an excellent starting point, and happens at a right time

- a body (panel?) to follow-up with a wise and balanced composition would potentially make a big difference