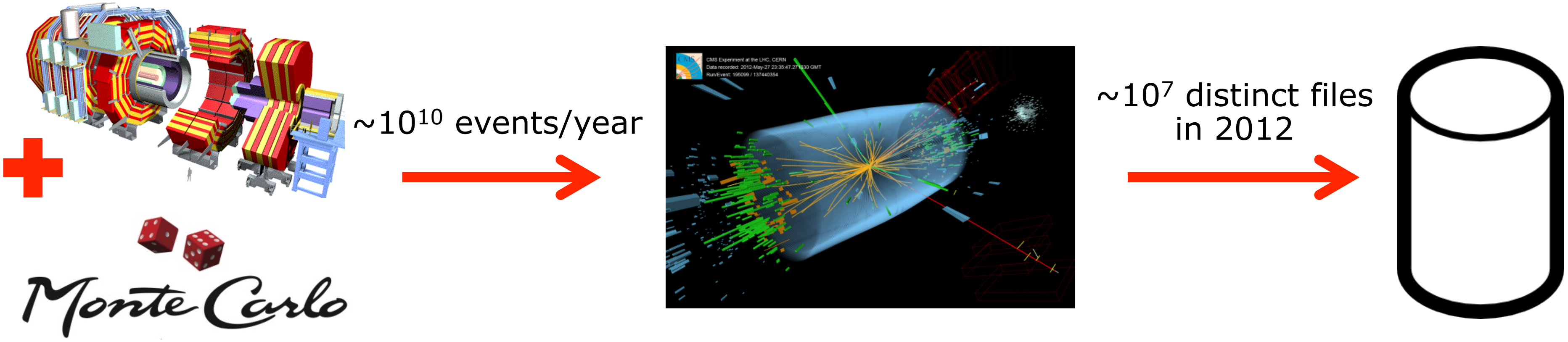
The CMS Data Management System

M. Giffels, Y. Guo, N. Magini,
T. Wildish, V. Kuznetsov

Outline

- Introduction
- Data Management in CMS: performance during LHC Run 1 and planned improvements for LHC Run 2
  - Architecture
  - Core components
    - Data bookkeeping – DBS
    - Data transfers – PhEDEx
    - Query and aggregation service – DAS
  - Additional services
- Summary

CMS Data


Monte Carlo
CMS Experiment at the LHC, CERN
Data recorded: 2012-May-27 23:35:47.271030 GMT
Run/Event: 195099 / 137440354

~10^{10} events/year

~10^{7} distinct files in 2012

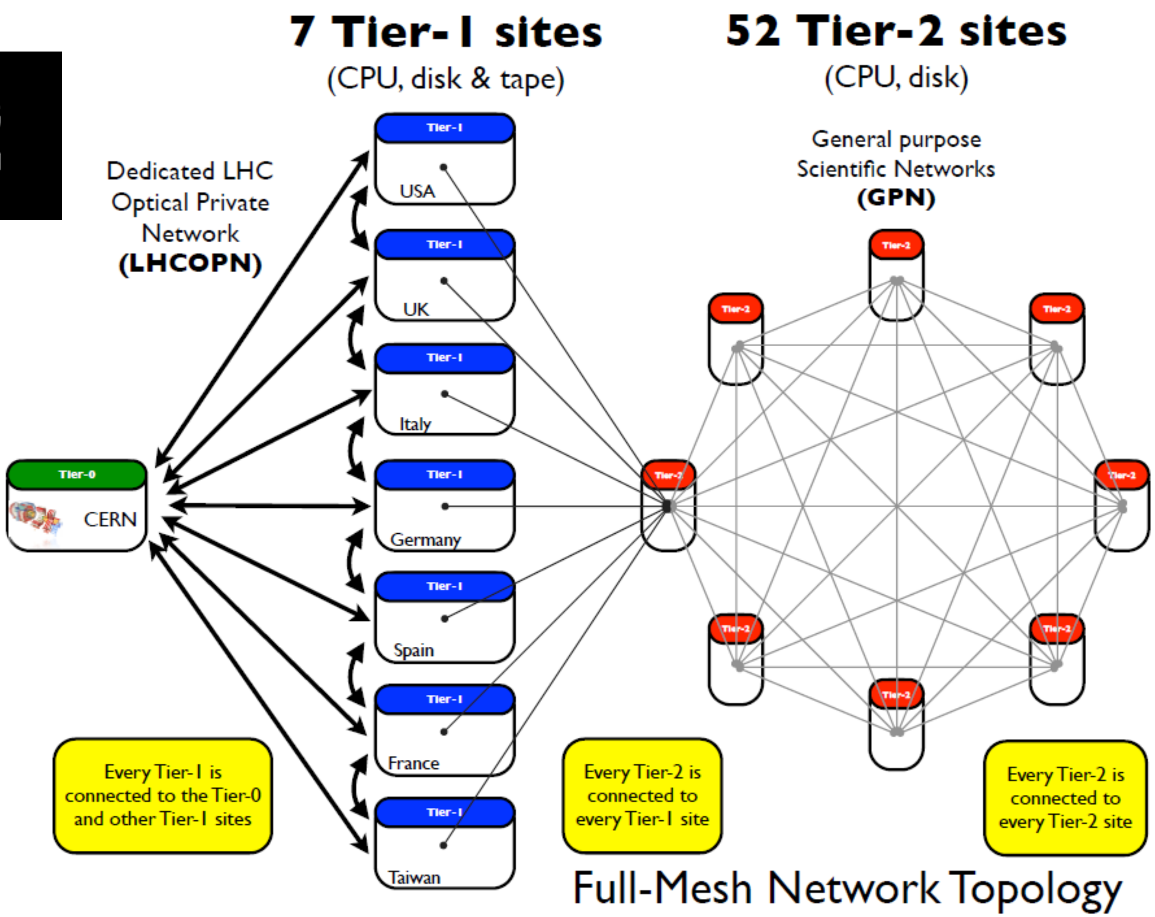
- Event data in files
  - average file size reasonably large ~2.5 GB
    - Output merged to help scaling in catalogs and storages
- Files are grouped in file blocks to manage them in bulk
  - ~10-1000 files/block
- File blocks are grouped by physics content in datasets of variable size (0.1–100 TB)

CMS Computing Infrastructure

WLCG
Worldwide LHC Computing Grid

Tier-0

Custodial storage of first copy of the data
Disk storage for prompt reconstruction and calibration
Distribution of data to Tier-1s



7 Tier-I sites
(CPU, disk & tape)

52 Tier-2 sites
(CPU, disk)

Dedicated LHC Optical Private Network **(LHCOPN)**

General purpose Scientific Networks **(GPN)**

Every Tier-I is connected to the Tier-0 and other Tier-I sites

Every Tier-2 is connected to every Tier-I site

Every Tier-2 is connected to every Tier-2 site

Full-Mesh Network Topology

Tier-1s

Shared custodial storage of second copy of the data
Disk storage for data reconstruction and reprocessing
Distribution of analysis data to Tier-2s

Tier-2s

Disk storage for end-user physics analysis and Monte Carlo simulation

Data Management system

- Tasks:
  - Data bookkeeping catalog: *what are the data?*
  - Data location catalog: *where are the data?*
  - Data placement and transfer management
    - For archiving and before processing: CMS job submission is *mostly* based on data location
- CMS Architecture:
  - Independent core components dedicated to different tasks
  - Interacting with each other and with external services via webservices
  - Data from different components aggregated through dedicated service

‘Trivial’ File Catalog

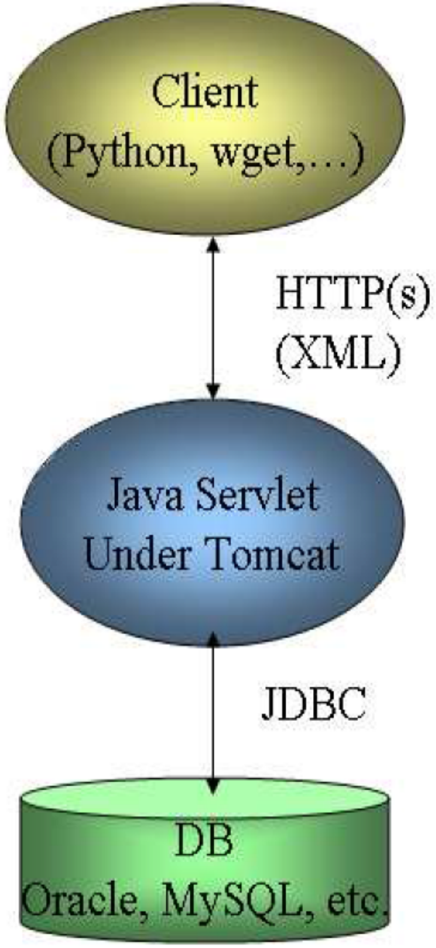
- *Keep site configuration local*: no global catalog of physical file replicas
- Central services only keep track of the site location of the replicas of logical files
- **TrivialFileCatalog** to map logical to physical file names on local storages: just an xml with a set of regexp rules published by the site
  - Running jobs don’t need to contact DM system for file access, they just need to read a local config file
  - Sites can change their storage backend transparently for CMS – just publish a new xml
  - Increased flexibility for sites who need complicated storage setups (e.g. different storages for different namespaces), and great simplicity for sites who don’t

Dataset Bookkeeping System (DBS)

- Data bookkeeping: **DBS**
- Data definition
  - What are the data, how they were produced
    - configuration, run number/luminosity, parentage
- Data discovery
  - Which data exist, how they are organized in datasets/blocks/files

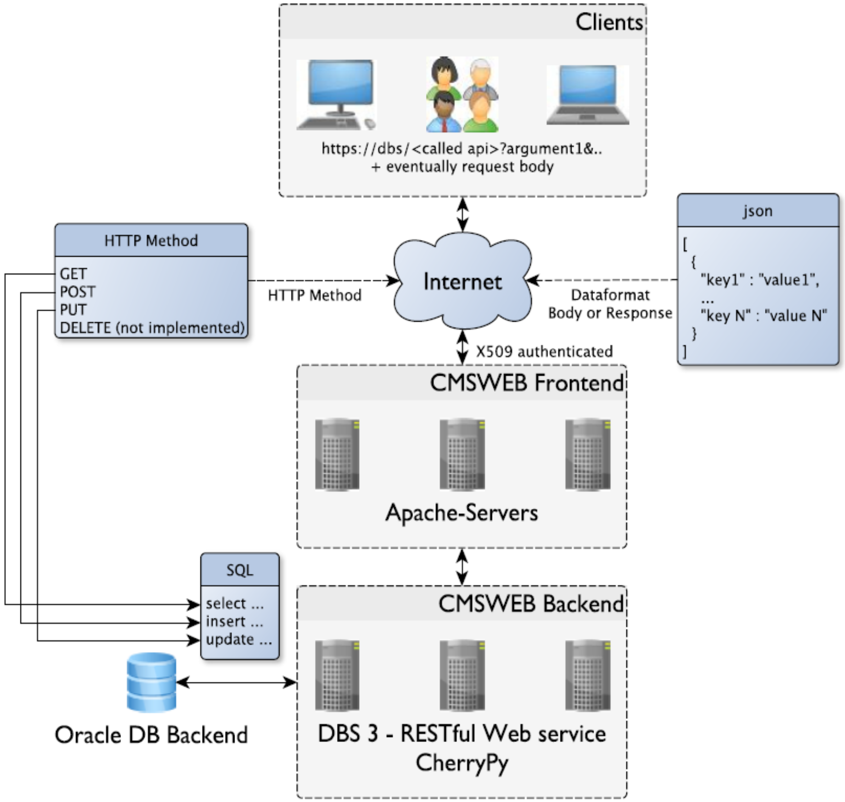
Dataset Bookkeeping System (DBS)

- Current implementation: **DBS2**
  - SQL DB backend
    - Oracle currently used, more supported
    - Multiple instances for different scopes: **Global DBS** holds all official CMS data
  - Java servlets under Tomcat
  - XML format for client payload, requires client-side API
- DBS2 sustained the load in LHC Run 1 with some scaling issues
  - At startup the DBS webpage was the main interface for user data discovery, this had to be dropped



Dataset Bookkeeping System (DBS)

- New version: **DBS3**
  - Rescoped schema in Oracle DB to improve scaling, dropping information that did not belong there e.g. data location
  - REST API based on standard cmsweb libs, SQLAlchemy and CherryPy
  - Lightweight information exchange with JSON
- Currently deployed in parallel to DBS2 for validation before final switch[1]



[1] M. Giffels: Data Bookkeeping Service 3 - Providing event metadata in CMS

PhEDEx

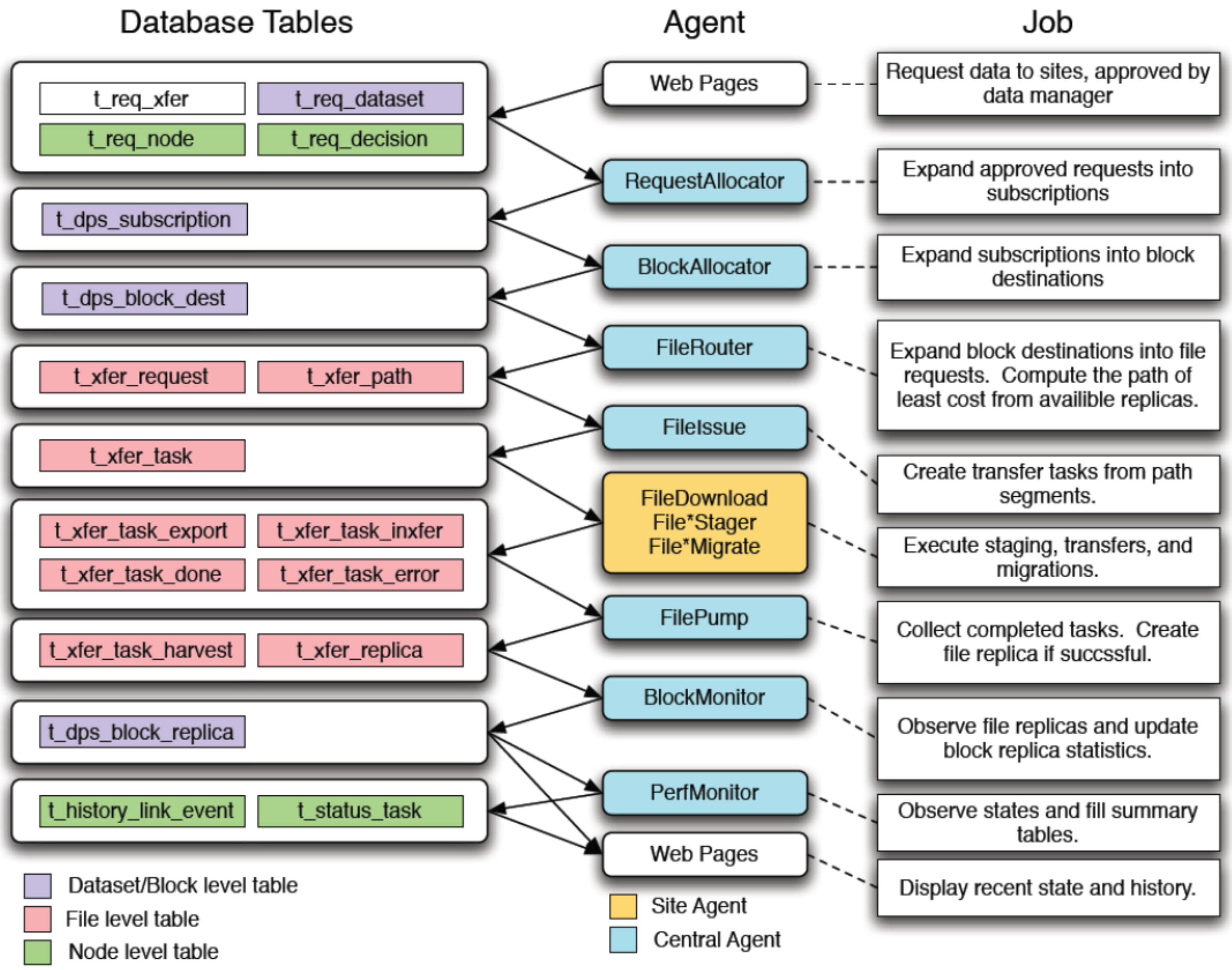
- Data location and placement: **PhEDEx** - Physics Experiment Data Export
- Each CMS site runs a set of agents
  - Independent, specialized perl daemons dedicated to fulfill a specific “simple” task
    - Central agents: routing, task assignment, …
    - Site-specific agents: download, mass storage staging and migration, deletion, consistency checks, …
- Agents intercommunicate through a central Transfer Management DB (TMDB)
  - Oracle SQL backend
- Web data service and interactive site

TMDB

Oracle SQL DB for:

- Replica location catalog
  - Tracked at the level of blocks of files except during transfers, for scaling
  - Only site location of replicas is tracked
- Transfer state blackboard
  - Highly volatile tables for file-level information on active transfer tasks
  - Monitoring tables with aggregated data

PhEDEx workflow



Database Tables

- t_req_xfer, t_req_dataset, t_req_node, t_req_decision
- t_dps_subscription
- t_dps_block_dest
- t_xfer_request, t_xfer_path
- t_xfer_task
- t_xfer_task_export, t_xfer_task_inxfer, t_xfer_task_done, t_xfer_task_error
- t_xfer_task_harvest, t_xfer_replica
- t_dps_block_replica
- t_history_link_event, t_status_task

🟪 Dataset/Block level table

🟥 File level table

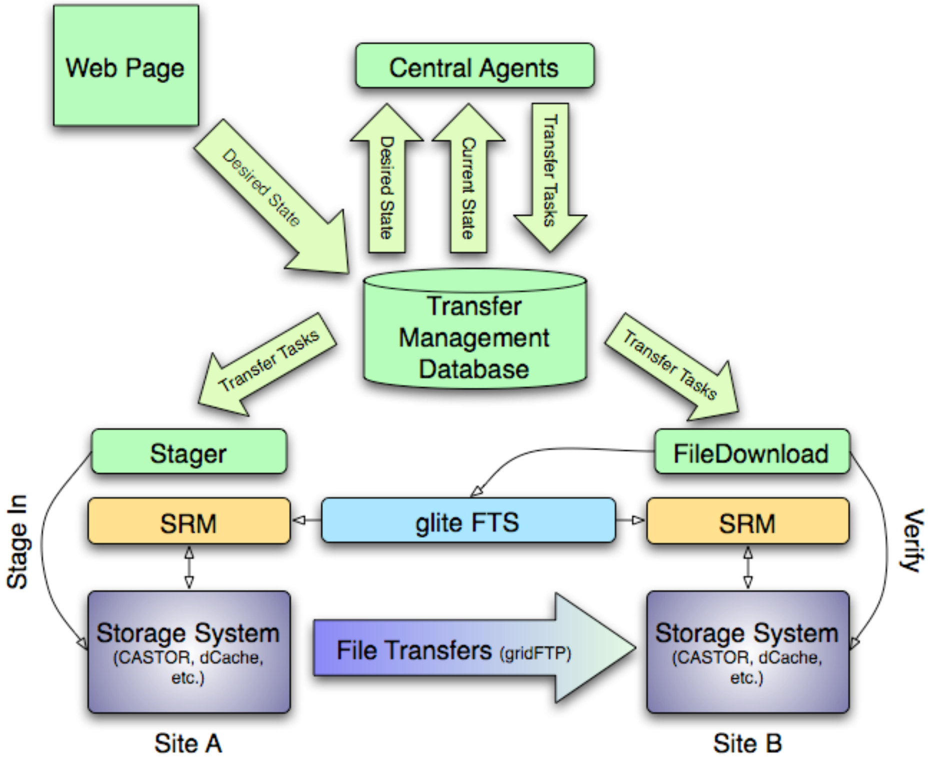
🟩 Node level table

Agent	Job
Web Pages	Request data to sites, approved by data manager
RequestAllocator	Expand approved requests into subscriptions
BlockAllocator	Expand subscriptions into block destinations
FileRouter	Expand block destinations into file requests. Compute the path of least cost from availible replicas.
FileIssue	Create transfer tasks from path segments.
FileDownload File*Stager File*Migrate	Execute staging, transfers, and migrations.
FilePump	Collect completed tasks. Create file replica if succssful.
BlockMonitor	Observe file replicas and update block replica statistics.
PerfMonitor	Observe states and fill summary tables.
Web Pages	Display recent state and history.

🟨 Site Agent

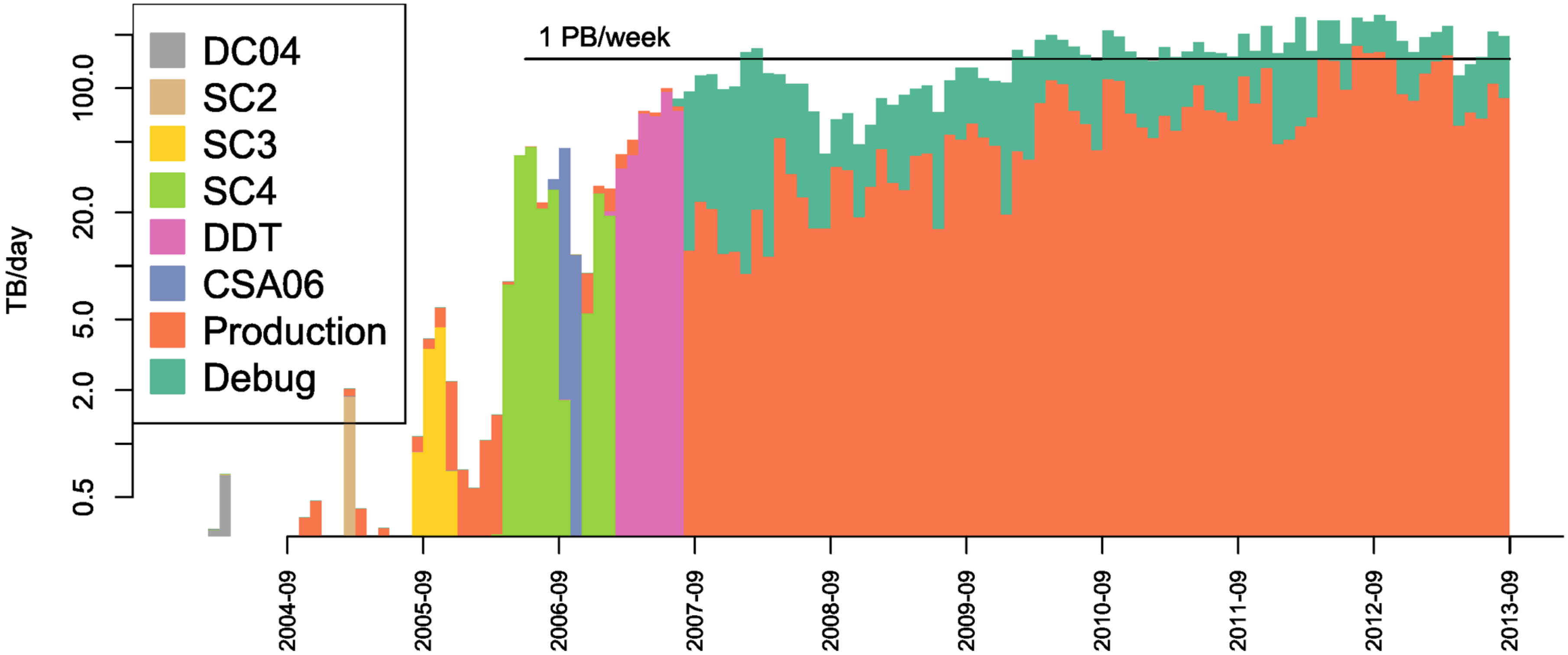
🟦 Central Agent

PhEDEx transfer workflow



- Central PhEDEx agents are middleware-agnostic
- Site agents integrated through plugins with WLCG DM middleware – e.g FTS or SRM – to execute transfers

PhEDEx performance



- Up to 77 PB of replicas, ~450k transfers/day in Production

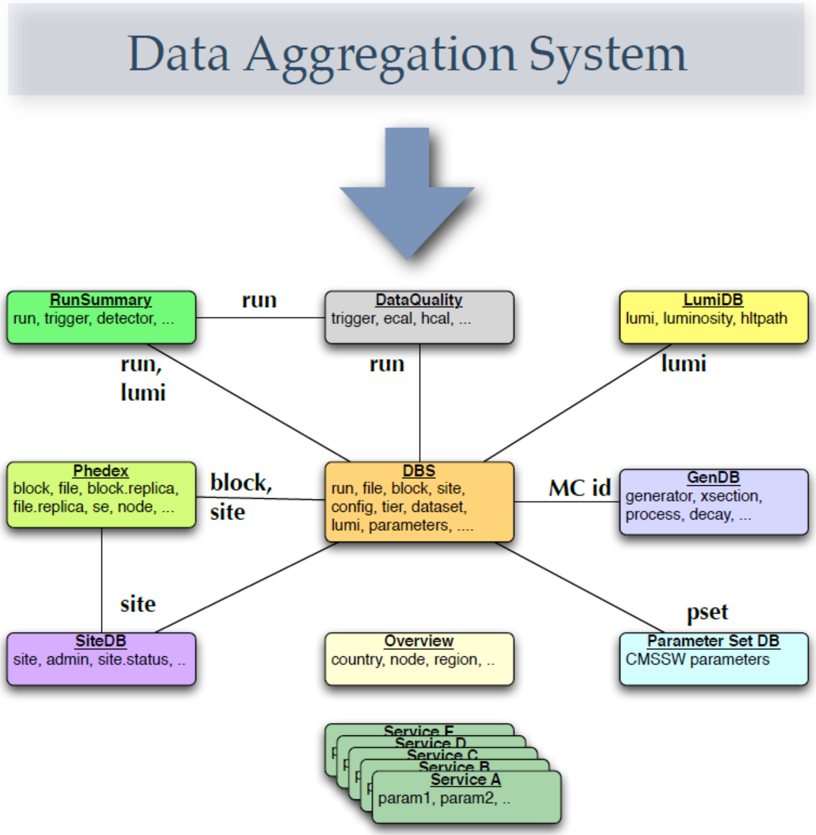
PhEDEx improvements

- PhEDEx scaling proved in simulation testbeds with rates/data volumes ~100x higher than production[1]
- Recent and planned changes focused on adding support for more flexible workflows
  - Improved support for block transfers
  - Support for requesting generic operator actions e.g. consistency checking[2]
  - Support for dynamic networking[3]

1. T. Wildish: Integration and validation testing for PhEDEx, DBS and DAS with the PhEDEx LifeCycle agent
2. C-H Huang: Request for All - Generalized Request Framework for PhEDEx
3. T. Wildish: Challenging data and workload management in CMS Computing with network-aware systems

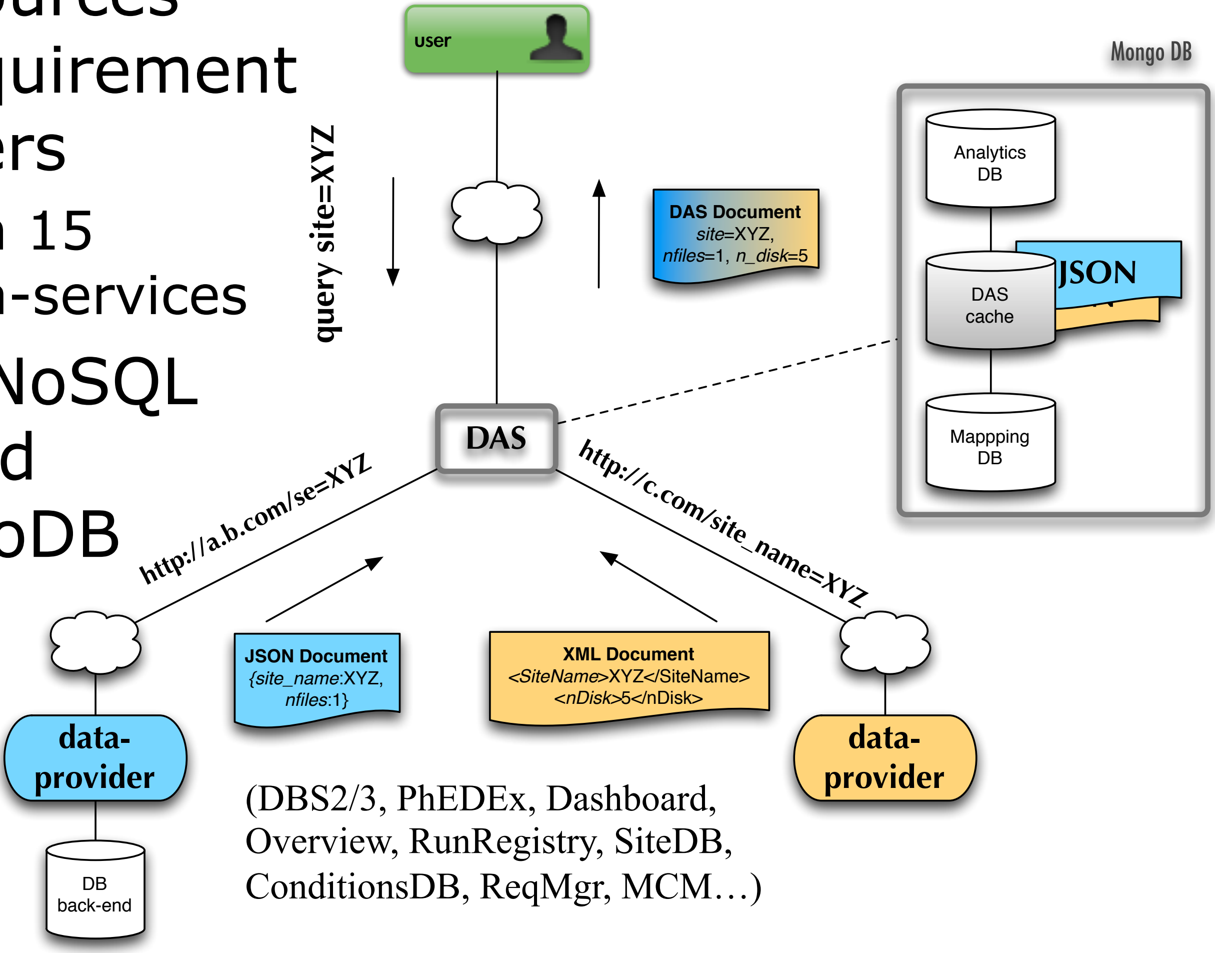
Data aggregation

- Deploying independent, dedicated services ensures that each can be optimized for its own task
- Disadvantage: users and services need to query multiple sources to get combined information, e.g.
  - find all sites hosting files for run=XXX
- Solution: **DAS** Data Aggregation System


Data Aggregation System
RunSummary
run, trigger, detector, ...
run
DataQuality
trigger, ecal, hcal, ...
LumiDB
lumi, luminosity, hltpath
run, lumi
run
lumi
Phedex
block, file, block.replica, file.replica, se, node, ...
block, site
DBS
run, file, block, site, config, tier, dataset, lumi, parameters, ...
MC id
GenDB
generator, xsection, process, decay, ...
site
pset
SiteDB
site, admin, site.status, ..
Overview
country, node, region, ..
Parameter Set DB
CMSSW parameters
Service E
Service D
Service C
Service B
Service A
param1, param2, ..

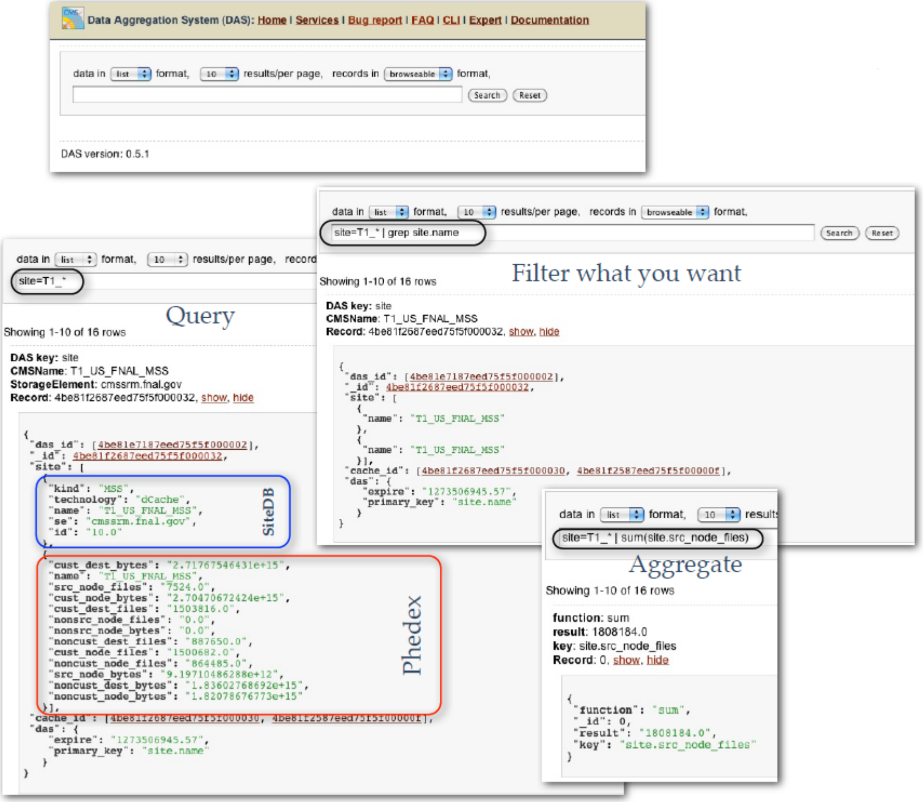
DAS

- Aggregating data from multiple web sources without any requirement on data providers
  - DAS works with 15 distributed data-services
- Data stored in NoSQL document-based database MongoDB
  - For caching and storing aggregated results



(DBS2/3, PhEDEx, Dashboard, Overview, RunRegistry, SiteDB, ConditionsDB, ReqMgr, MCM…)

DAS interface


Data Aggregation System (DAS): Home | Services | Bug report | FAQ | CLI | Expert | Documentation
data in list format, 10 results/per page, records in browseable format,
Search
Reset
Query
SiteDB
Phedex
Filter what you want
Aggregate

DAS version: 0.5.1

site=T1_*

Showing 1-10 of 16 rows

DAS key: site
CMSName: T1_US_FNAL_MSS
StorageElement: cmssrm.fnal.gov
Record: 4be81f2687eed75f5f000032, show, hide

```
{
 "das_id": [4be81e7187eed75f5f000002],
 "_id": 4be81f2687eed75f5f000032,
 "site": [
  {
   "kind": "MSS",
   "technology": "dCache",
   "name": "T1_US_FNAL_MSS",
   "se": "cmssrm.fnal.gov",
   "id": "10.0"
  },
  {
   "cust_dest_bytes": "2.71767546431e+15",
   "name": "T1_US_FNAL_MSS",
   "src_node_files": "7524.0",
   "cust_node_bytes": "2.70470672424e+15",
   "cust_dest_files": "1503816.0",
   "nonsrc_node_files": "0.0",
   "nonsrc_node_bytes": "0.0",
   "noncust_dest_files": "887650.0",
   "cust_node_files": "1500682.0",
   "noncust_node_files": "864485.0",
   "src_node_bytes": "9.19710486288e+12",
   "noncust_dest_bytes": "1.83602768692e+15",
   "noncust_node_bytes": "1.82078676773e+15"
  }],
 "cache_id": [4be81f2687eed75f5f000030, 4be81f2587eed75f5f00000f],
 "das": {
   "expire": "1273506945.57",
   "primary_key": "site.name"
  }
}
```

site=T1_* | grep site.name

Showing 1-10 of 16 rows

DAS key: site
CMSName: T1_US_FNAL_MSS
Record: 4be81f2687eed75f5f000032, show, hide

```
{
 "das_id": [4be81e7187eed75f5f000002],
 "_id": 4be81f2687eed75f5f000032,
 "site": [
   {
    "name": "T1_US_FNAL_MSS"
   },
   {
    "name": "T1_US_FNAL_MSS"
   }],
 "cache_id": [4be81f2687eed75f5f000030, 4be81f2587eed75f5f00000f],
 "das": {
    "expire": "1273506945.57",
    "primary_key": "site.name"
   }
}
```

site=T1_* | sum(site.src_node_files)

Showing 1-10 of 16 rows

function: sum
result: 1808184.0
key: site.src_node_files
Record: 0, show, hide

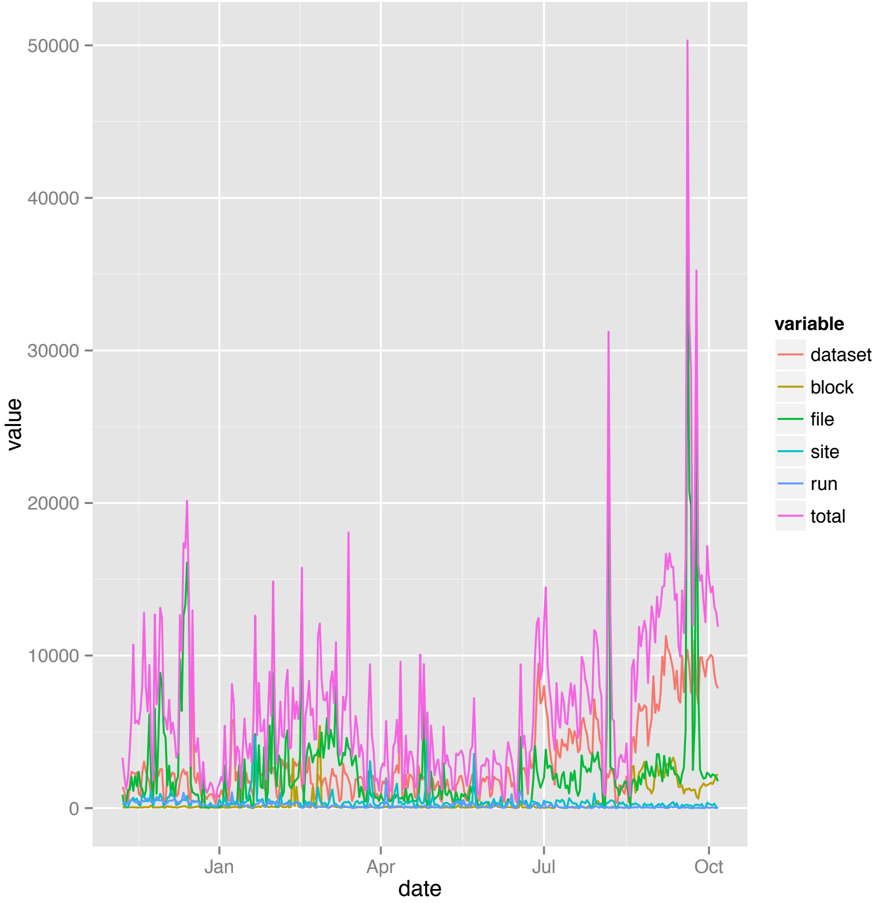
```
{
 "function": "sum",
 "_id": 0,
 "result": "1808184.0",
 "key": "site.src_node_files"
}
```

Free text-based query language, filters, aggregators

- DAS interactive webpage fully replaced DBS webpage as main user entry point for data discovery in 2011

DAS performance

- Input queries/day on DAS during 2013
- Each query produces O(10-1000) results
- On average O(10M) results entering and served from DAS cache every day

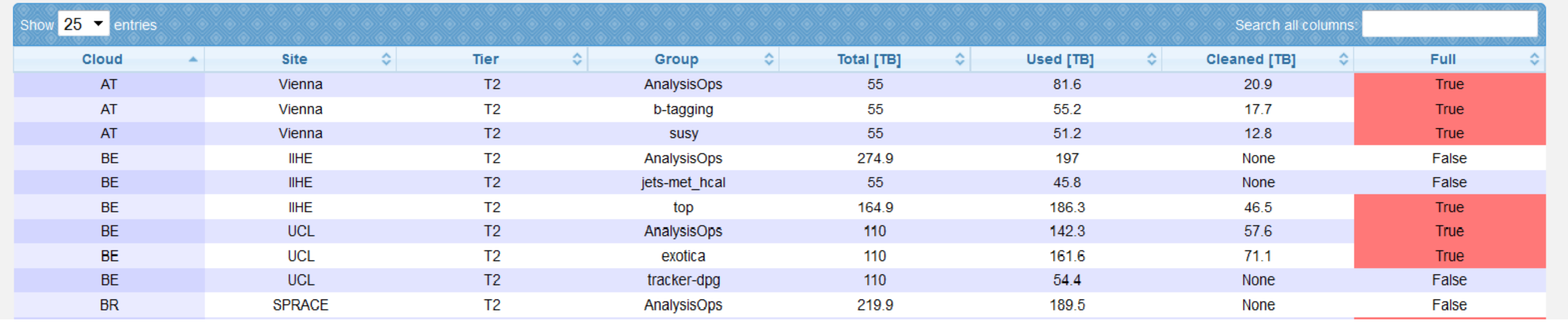


Additional services

- External services can be easily interfaced to CMS DM components through their web services

CMS POPULARITY & VICTOR

CERN IT Experiment Support



Cloud	Site	Tier	Group	Total [TB]	Used [TB]	Cleaned [TB]	Full
AT	Vienna	T2	AnalysisOps	55	81.6	20.9	True
AT	Vienna	T2	b-tagging	55	55.2	17.7	True
AT	Vienna	T2	susy	55	51.2	12.8	True
BE	IIHE	T2	AnalysisOps	274.9	197	None	False
BE	IIHE	T2	jets-met_hcal	55	45.8	None	False
BE	IIHE	T2	top	164.9	186.3	46.5	True
BE	UCL	T2	AnalysisOps	110	142.3	57.6	True
BE	UCL	T2	exotica	110	161.6	71.1	True
BE	UCL	T2	tracker-dpg	110	54.4	None	False
BR	SPRACE	T2	AnalysisOps	219.9	189.5	None	False

- **Victor** data cleaning service deployed in 2011
  - Interfaced with PhEDEx and dataset popularity service to identify unused replicas that can be cleaned up
- Next? Dynamic data placement service to trigger replication in PhEDEx of “hot” (popular) datasets

Summary

- CMS Data Management based on independent core components individually optimized to ensure scaling
- User interface provided by an Aggregation Service that integrates information from the underlying services
- Using a common CMSWEB web service framework allows
  - independent development and evolution of underlying services
  - simplified integration and regression testing when rolling out new service versions
  - building external services that integrate information from several sources in a clean manner