Status and Evolution of ATLAS Workload Management System PanDA

Kaushik De
Univ. of Texas at Arlington

GRID 2012, Dubna
July 17, 2012

Outline

- Overview
- PanDA design
- PanDA performance
- Recent Improvements
- Future Plans

Why PanDA

- The ATLAS experiment at the LHC is a large and complicated international science project
  - Ambitious scientific program – Higgs, SUSY...
  - Computing is critical to success of scientific program
  - Hundreds of petabytes of data are distributed world-wide to hundreds of WLCG computing centers
  - Thousands of physicists analyze the data
- PanDA goal:
  - An automated yet flexible workload management system (WMS) which can optimally make distributed resources accessible to all users

A Brief History

- PanDA project was started in Fall 2005
  - Originally developed in US for managed production
  - Successfully used on OSG (Open Science Grid) resources in the US in 2006
- Extended to user analysis in 2006
  - Simple interface for submitting user work to grid
- Deployed on EGEE resources in 2007
- Adopted as the ATLAS wide WMS in 2008, before first LHC data in 2009

References

- https://twiki.cern.ch/twiki/bin/viewauth/Atlas/PanDA
- http://panda.cern.ch:25880/server/pandamon/query
- Recent Improvements in the ATLAS PanDA Pilot, P. Nilsson, CHEP 2012, United States, May 2012
- PD2P : PanDA Dynamic Data Placement for ATLAS, T. Maeno, CHEP 2012, United States, May 2012
- Evolution of the ATLAS PanDA Production and Distributed Analysis System, T. Maeno, CHEP 2012, United States, May 2012

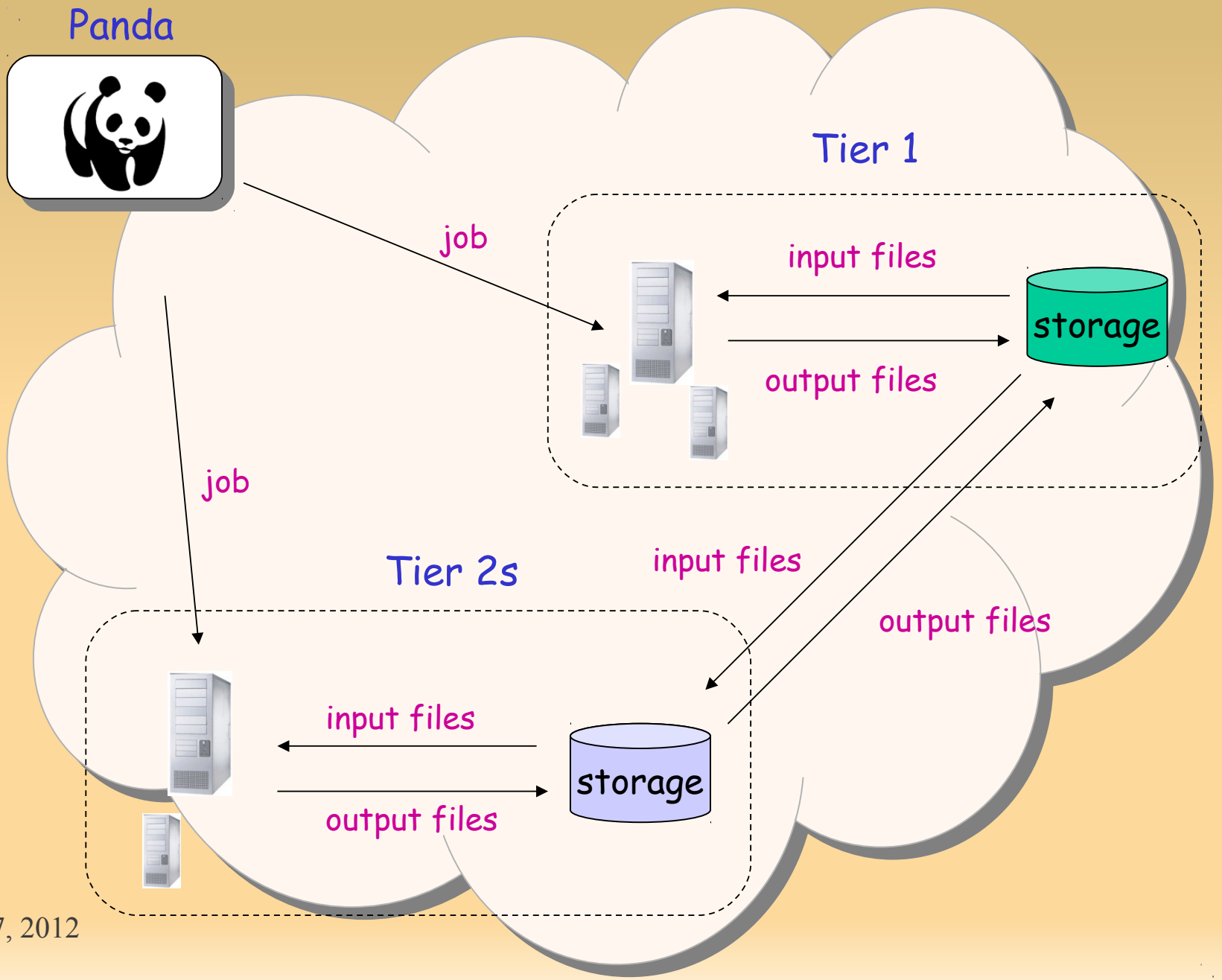
PanDA Philosophy

- PanDA WMS design goals:
  - Achieve high level of automation to reduce operational effort
  - Flexibility in adapting to evolving hardware and network configurations (i.e. computing model)
  - Support diverse and changing middleware
  - Insulate user from hardware, middleware, and all other complexities of the underlying system
  - Unified system for production and user analysis
  - Incremental and adaptive software development

PanDA Basics

- Key features of PanDA
  - Pilot based job execution system
    - ATLAS work is sent only after execution begins on CE
    - Minimize latency, reduce error rates
  - Central job queue
    - Unified treatment of distributed resources
    - SQL DB keeps state - critical component
  - Automatic error handling and recovery
  - Extensive monitoring
  - Modular design

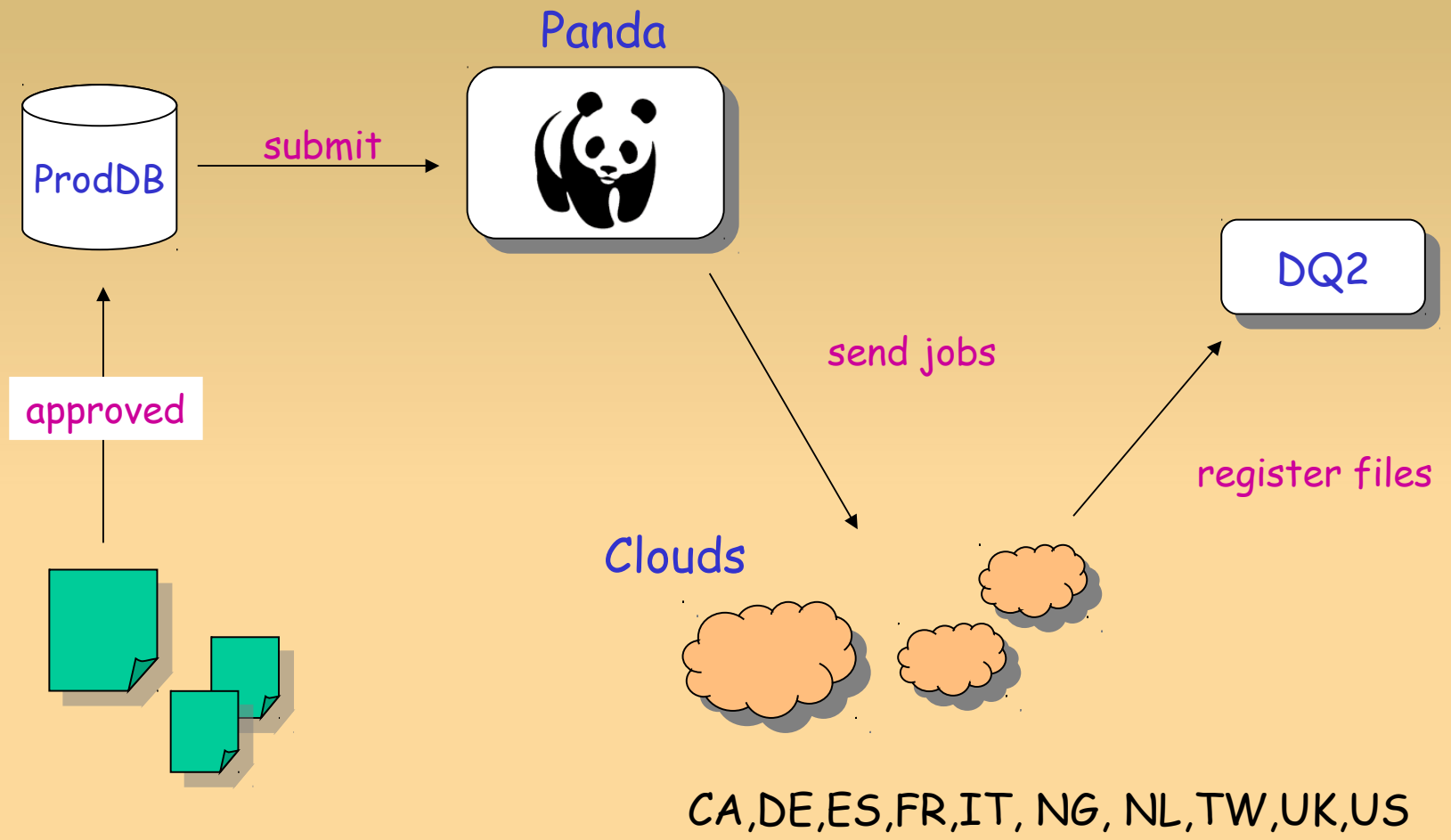
Simplified View



PanDA Components

- PanDA server
- Database back-end
- PanDA pilot system
  - Job wrapper
  - Pilot factory
- Brokerage
- Dispatcher
- Information system
- Monitoring systems

PanDA Design



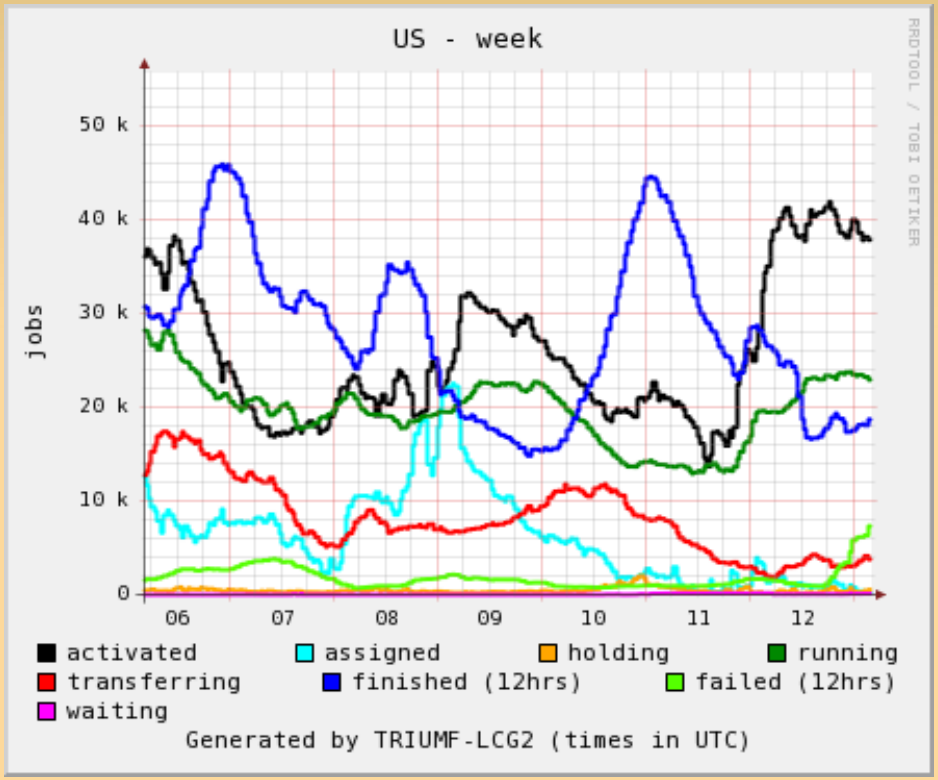
- HTTP/S RESTful communication (curl+grid proxy+python)
- GSI authentication via mod_gridsite
- Workflow is maximally asynchronous

What is a Job

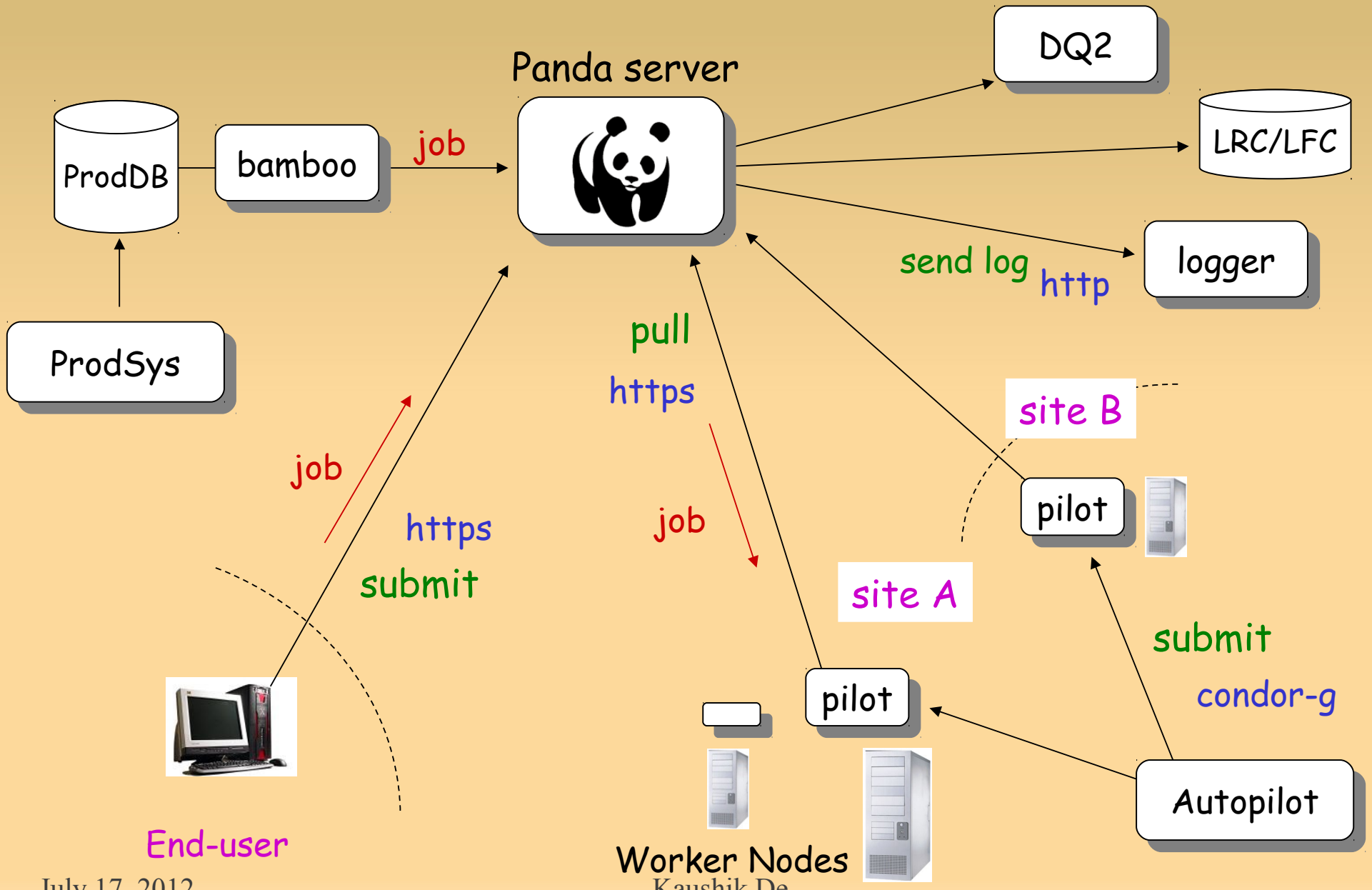
- Basic unit of work is a job:
  - Executed on a CPU resource/slot
  - May have inputs
  - Produces outputs
- ProdSys – layer above PanDA to create jobs from ATLAS physics 'tasks'
- User analysis work divided into jobs by PanDA
- Pilot may run multiple jobs on request

Job States

- Panda jobs go through a succession of steps tracked in DB
  - Defined
  - Assigned
  - Activated
  - Running
  - Holding
  - Transferring
  - Finished/failed



PanDA Interfaces



What is a Pilot Job

- Lightweight execution environment to prepare CE, request actual payload, execute payload, and clean up
- Handles data stage-in and stage-out between worker node disk and local SE
- Pilot jobs started by Job Scheduler(s); actual ATLAS job (payload) is scheduled when CPU becomes available, leading to low latency
- Monitoring thread, job recovery, experiment specific setup and post processing...

Workload Management

Production managers

define

task/job repository (Production DB)

submitter (bamboo)

production job

https

PanDA server

Data Management System (DQ2)

https

Local Replica Catalog (LFC)

https

Logging System

pull

https

job

analysis job

https

submit

End-user

EGEE/EGI

pilot

Worker Nodes

https

OSG

pilot

https

NDGF

pilot

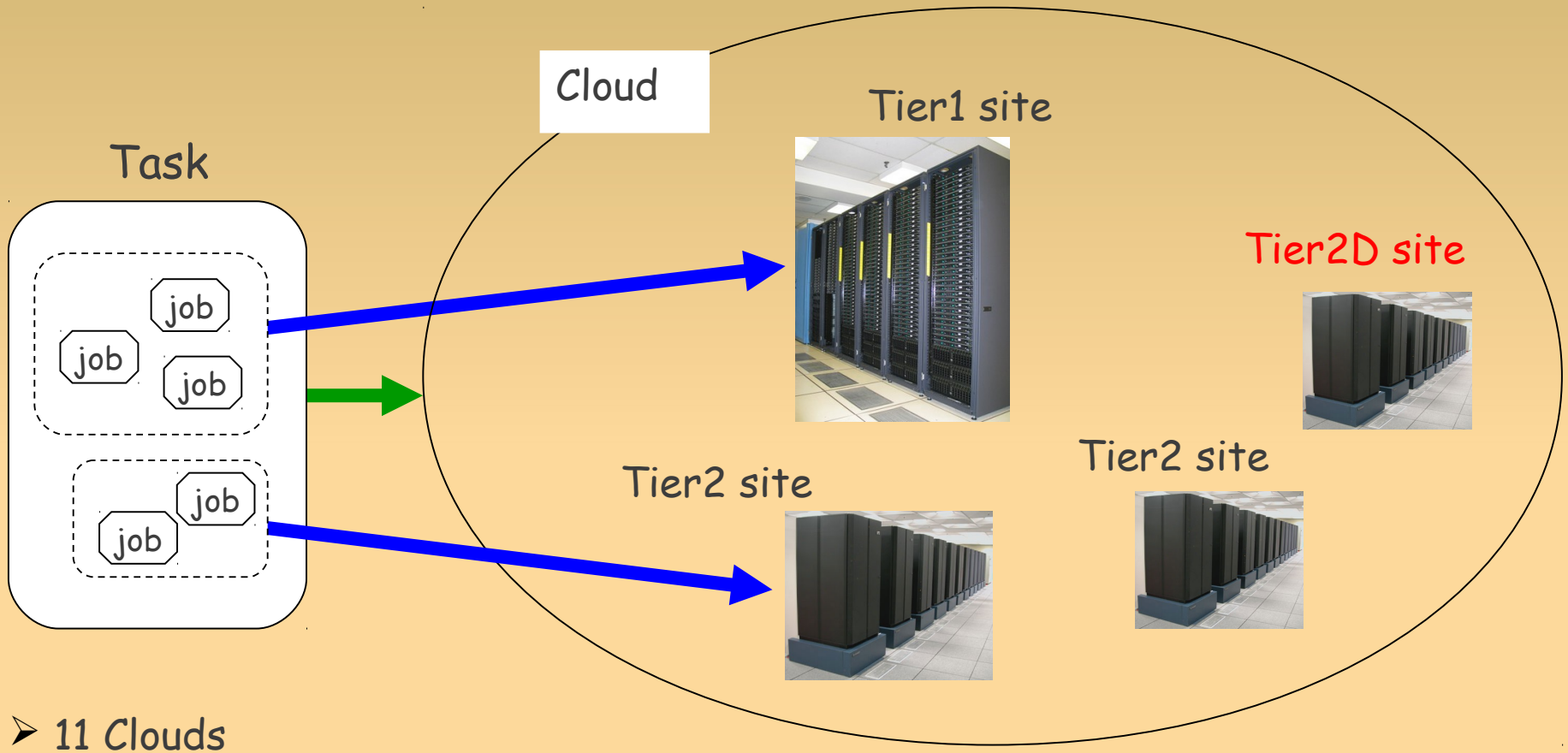
arc

ARC Interface (aCT)

condor-g

pilot scheduler (autopyfactory)

ATLAS Computing Model



- 11 Clouds
  10 T1s + 1 T0 (CERN)
  Cloud = T1 + T2s + T2Ds (except CERN)
  T2D = multi-cloud T2 sites
- 2-16 T2s in each Cloud

Task → Cloud
  Task brokerage
Jobs → Sites
  Job brokerage

Task Brokerage

- Matchmaking per cloud is based on:
  - Free disk space in T1 SE, MoU share of T1
  - Availability of input dataset (a set of files)
  - The amount of CPU resources = the number of running jobs in the cloud (static information system is not used)
  - Downtime at T1
  - Already queued tasks with equal or higher priorities
  - High priority task can jump over low priority tasks

Job Brokerage

- Brokerage policies define job assignment to sites
  - IO intensive or TAPE read -> T1
  - CPU intensive -> T1+T2s
  - Flexible: clouds may allow IO heavy jobs at T2s with low weight
- Matchmaking per site in a cloud
  - Software availability
  - Free disk space in SE, Scratch disk size on Worker Node (WN), Memory size on WN
  - Occupancy = the number of running jobs / the number of queued jobs, and downtime
  - Locality (cache hits) of input files

Job Dispatcher

- High performance/high throughput module
- Send matching job to CE upon pilot request
  - REST non-blocking communication
  - Different from brokerage, which is asynchronous
- Matching of jobs based on
  - Data locality
  - Memory and disk space
- Highest priority job is dispatched

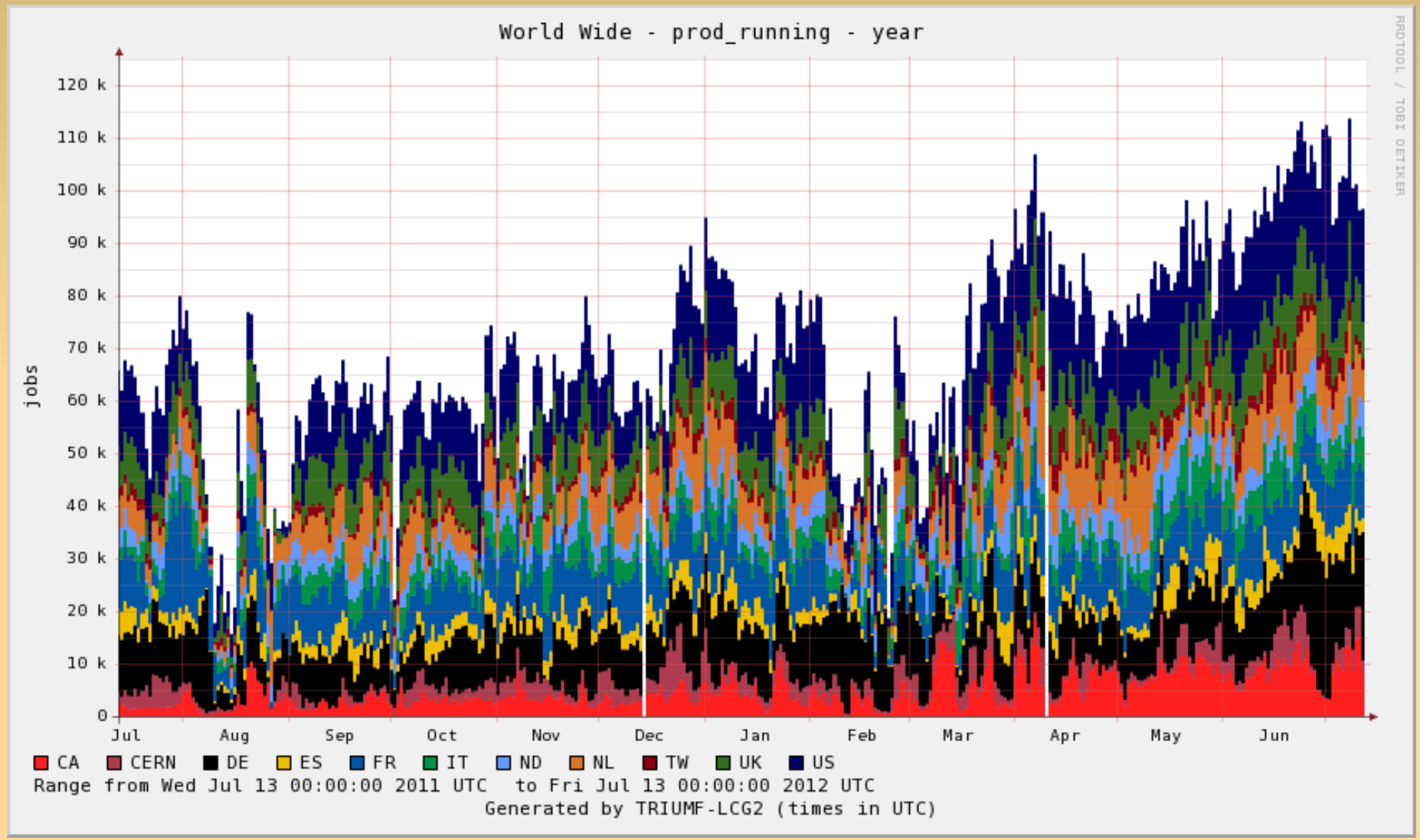
Data Management

- LFC file catalog
- Asynchronous file transfers by ATLAS DDM (DQ2)
  - Dispatch of input files from T1 SE to T2 SE, pre-staging of input files on T1 TAPE SE, aggregation of output files to T1 SE from T2 SE
  - CE CPU is not wasted waiting for transfers – pilot job starts only after input files ready, and ends after output is put on local SE
- Reusing input/output files as caches
  - Cache lifetime defined per cloud, job brokerage takes cache hits into account
- HTTPS message exchange between PanDA and DDM

Example of Flexibility

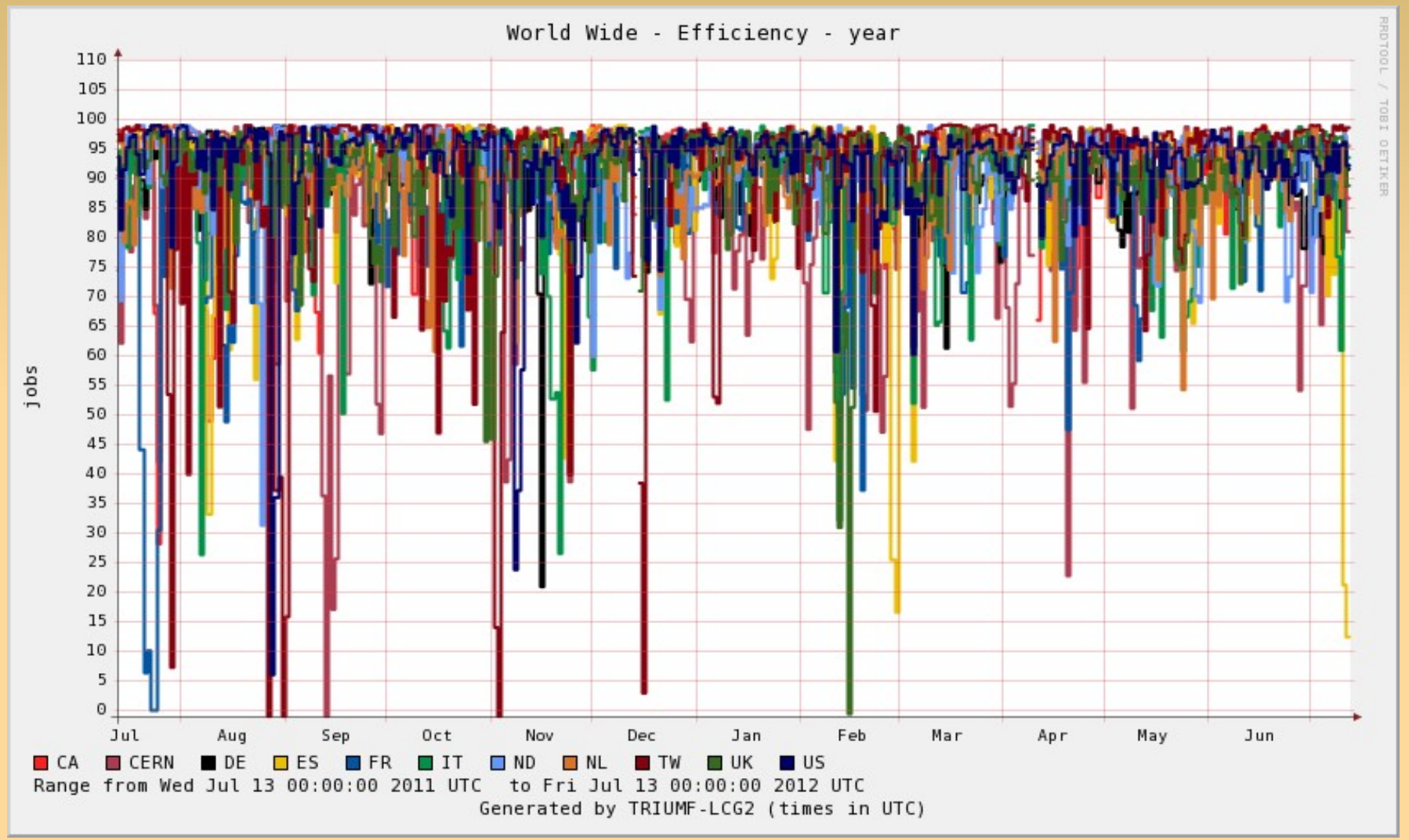
- PanDA supports multiple DDM solutions
  - Caching without LFC lookup
  - Pandamover file transfer (using chained Panda jobs)
  - Direct access if requested (by task or site)
  - Customizable lsm (local site mover)
  - Multiple default site movers are available
  - Flexible dataset sizing/containers for scalability

Performance - Production



Average number of concurrently running jobs per day

Cloud Efficiency



Average efficiency >95% - mostly site & application errors remain

User Analysis in PanDA

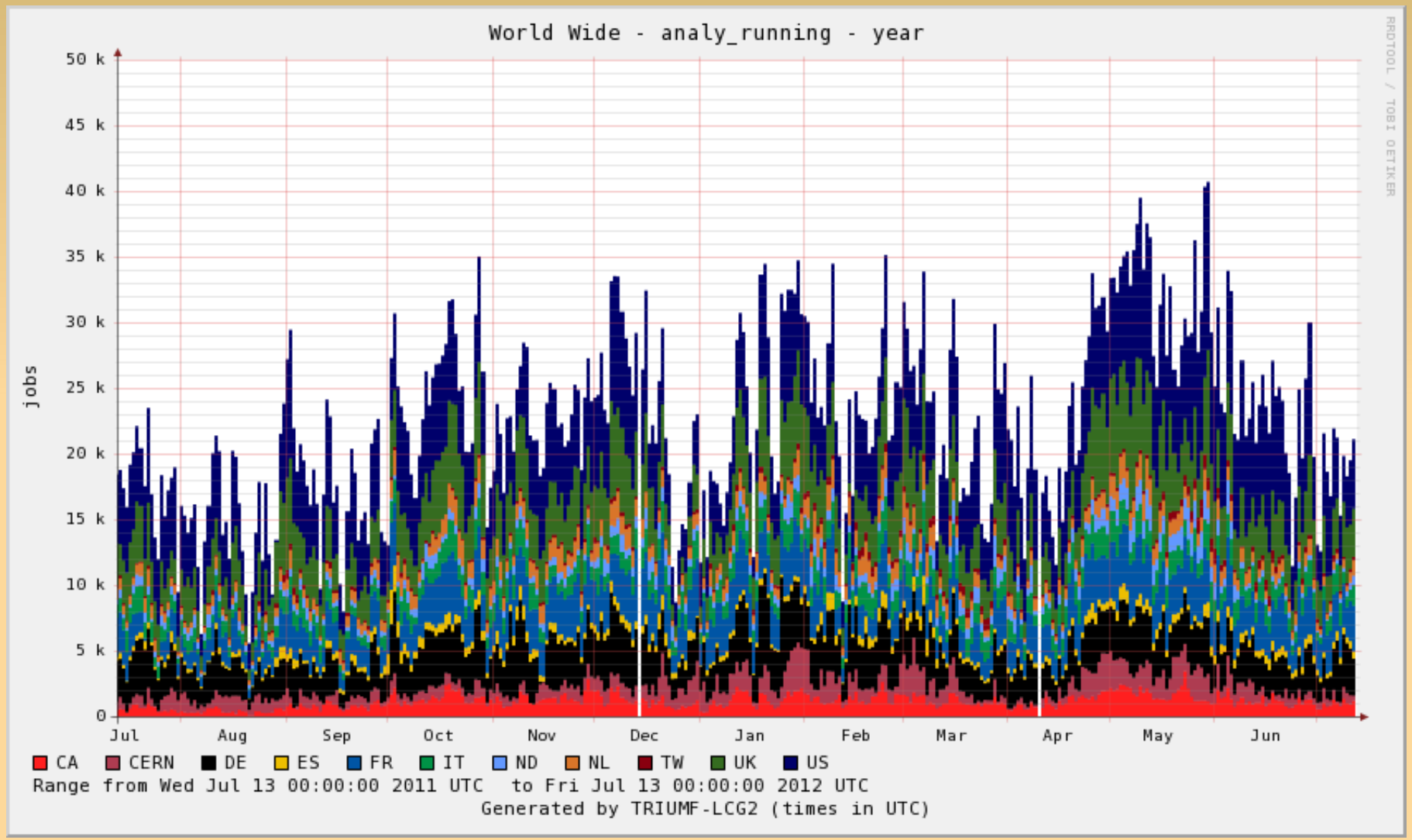
- Flexibility in job definition
  - Customization of source files
  - Adding new algorithms to application (athena) or arbitrary executables
- Fast turnaround for iteration
  - The user submits a user task (jobset) that is converted to many jobs for parallel execution
  - Supports IO intensive workflows
- Jobs go to data
  - No input file dispatch, no output file aggregation from multiple jobs
  - Data Transfer Request Interface (DaTRi) or PD2P (Dynamic Data Placement) options
- Dataset container (a set of datasets) as input and output
- Calculates priority and quotas per user or per working group

User Analysis Work Flow

Analysis Brokerage

- Works with jobsubset
  - A jobset may be split to be brokered to multiple sites
- Matchmaking per site without cloud-boundaries
  - Scratch disk size on WN, Memory size on WN
  - Software availability, Downtime
  - Occupancy = the number of running jobs / the number of queued jobs
  - Availability of input datasets

Analysis Performance



Average number of concurrently running jobs per day

PD2P – Recent Development

- PD2P = PanDA Dynamic Data Placement
- PD2P used to distribute data for user analysis
  - For production PanDA schedules all data flows, but initial computing model for user analysis was static distribution – PanDA sent jobs to data
  - Soon after LHC data started, we implemented PD2P
- Asynchronous usage based data placement
  - Repeated use of data → additional copies
  - Backlog in processing → additional copies
  - Rebrokerage of queued jobs use new data location
  - Deletion service removes less used data

Cloud Computing and PanDA

- ATLAS cloud computing group set up few years ago to exploit virtualization and clouds
  - PanDA queues in clouds – additional resources
  - Tier 3 in clouds – good for small institutes
- Excellent progress so far
  - Helix Nebula for MC production (CloudSigma, T-Systems and ATOS – all used)
  - Futuregrid (U Chicago), Synnefo cloud (U Vic)
- Personal PanDA analysis queues being set up

Big Data Plans

- AMS experiment testing PanDA
- CMS experiment evaluating PanDA for user analysis
- Other common projects under discussion
- US DoE recently selected “Next Generation Workload Management and Analysis System for Big Data” for ASCR funding
  - Integrate PanDA with dynamic network provisioning
  - Develop and release core PanDA package

Many Other Evolutions

- Federated storage (FAX)
  - Step by step plan to integrate into PanDA
  - First steps already successful
  - Different than current data management model
- JEDI – dynamic job definition
  - Higher level service to automatically define jobs from physics tasks
  - New level of brokerage
  - Better resource matching – especially MP jobs
- ……

PanDA Team

- K. De & T. Wenaus – coordinators
- T. Maeno – lead developer
- P. Nilsson – pilot developer
- Many other US ATLAS developers, CERN IT team, OSG support, Dubna collaborators, and other ATLAS contributions