Networking and Workload Management

Kaushik De

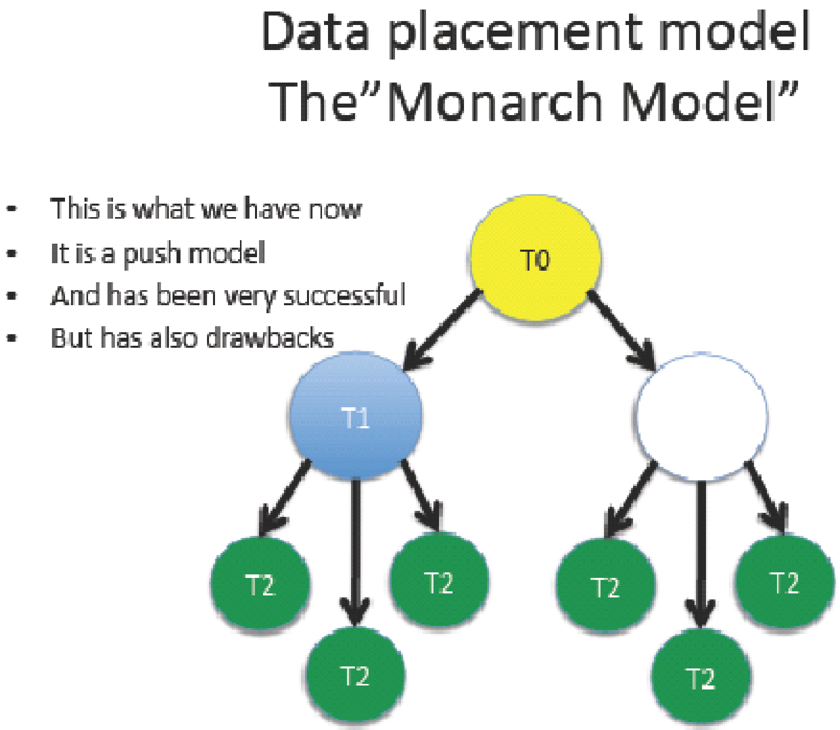
Univ. of Texas at Arlington

LHCONE P2P Workshop, CERN
December 13, 2012

Introduction

- Workload Management Systems (WMS) are crucial to the success of the LHC, and other high data volume sciences
- Distributed computing infrastructure, like the WLCG, especially require sophisticated WMS
- These WMS implicitly and explicitly depend on networking
  - Need agile WMS design to work with evolving network capabilities
  - But networking is used as a black box – not as a managed resource
  - Is this inevitable – or can we evolve to a better model?
- In this talk:
  - Show some recent WMS changes motivated by networking
  - I will describe how networking is used in a WMS
  - But focus on the potential for future improvements
- I will use PanDA WMS as example – since I know it best
  - I hope to provoke discussion and exchange of ideas

ATLAS Computing Model – Ancient History



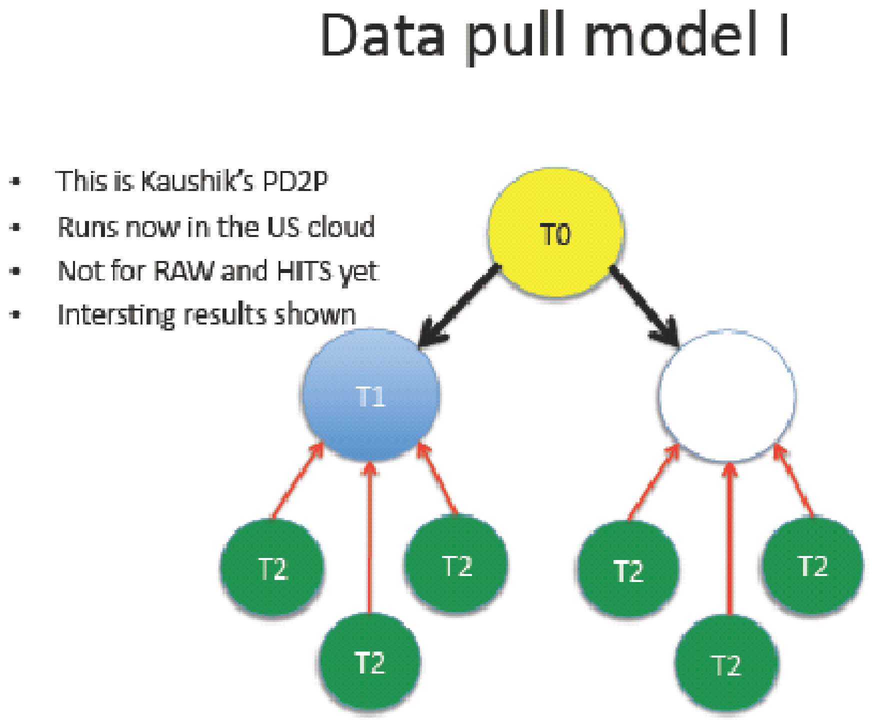
Data placement model
The"Monarch Model"

- This is what we have now
- It is a push model
- And has been very successful
- But has also drawbacks

- The original computing model for ATLAS was a push model, based on network capabilities 10 years ago
- PanDA WMS was designed to work with this rigid data placement model
- Within few months of LHC startup, we realized it was necessary to change
- PanDA quickly evolved, thanks to flexible design

ATLAS Pulled by Networking

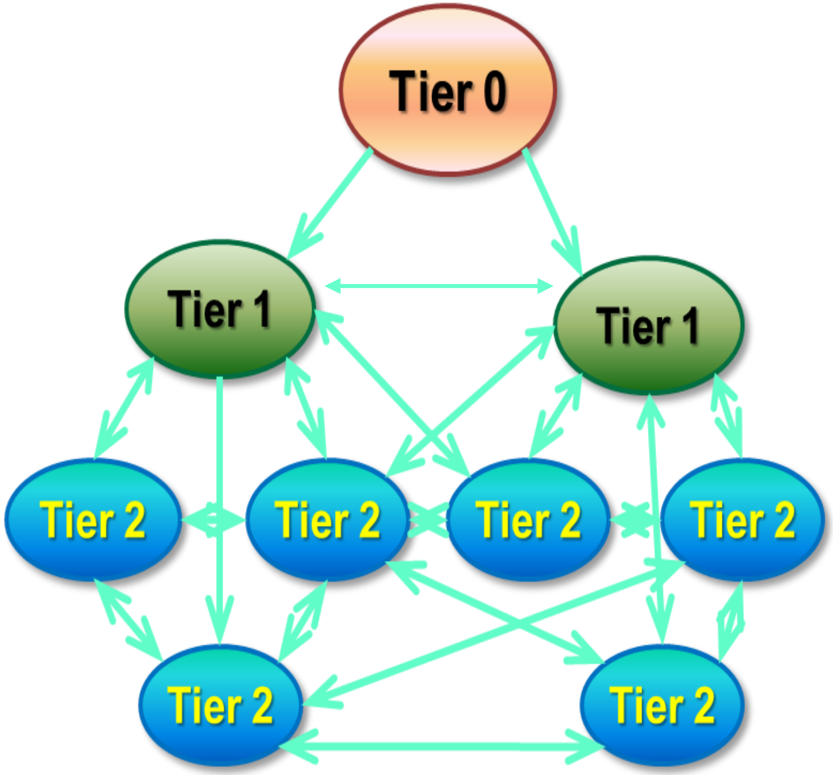
- By middle of 2011, Kors showed the changed computing model implemented in PanDA
- We quickly went from push model to pull model
- But some pushing came back in 2012 by physics analysis groups
- Now we have push-pull model for user analysis
- Flexibility wins again



Data pull model I

- This is Kaushik's PD2P
- Runs now in the US cloud
- Not for RAW and HITS yet
- Intersting results shown

Computing Cloud Boundaries



- PanDA continued to evolve as networking transcended national boundaries
- Michael showed this picture on Monday
- Many Tier 2’s participate in multi-cloud production
- No longer forced into geographical isolation
- ATLAS benefits from faster data processing

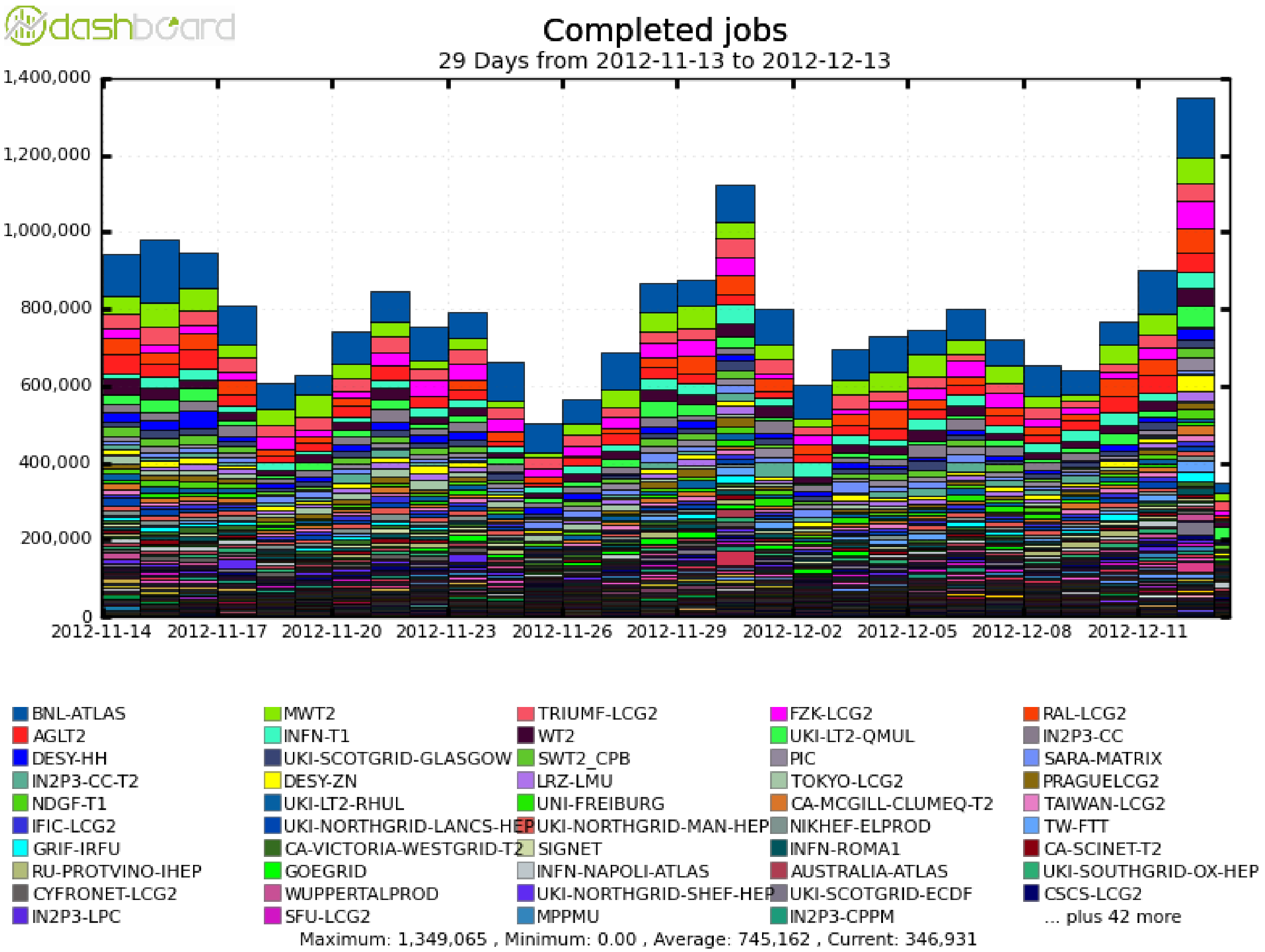
PanDA Introduction

- What is PanDA:
  - An automated, flexible workload management system which can optimally make distributed resources accessible to all users
- Designed for the ATLAS experiment at the LHC
  - Hundreds of petabytes of data are distributed world-wide to over one hundred WLCG computing centers
  - Hundred thousand production jobs run continuously on these resources
  - Thousands of physicists analyze the globally distributed data
  - PanDA is now being tested by AMS and CMS experiments

References

- https://twiki.cern.ch/twiki/bin/viewauth/Atlas/PanDA
- http://www.usatlas.bnl.gov/twiki/bin/view/PanDA/WebHome
- 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 Scale



Number of Analysis Users: (unique)

Users in the last 3 days : **458;** 7: **623;** 30: **941;** 90: **1240;** 180: **1547;**

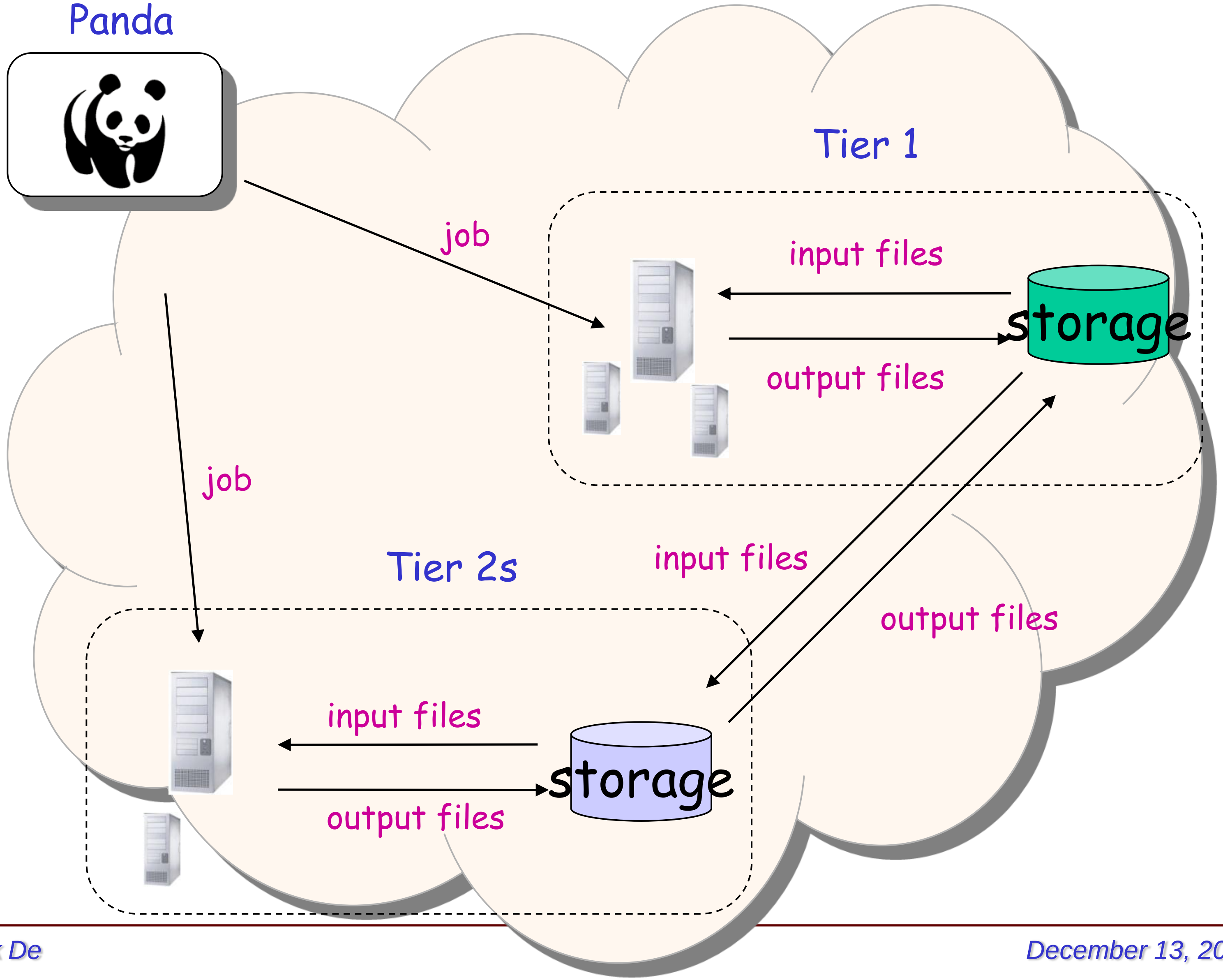
PanDA Philosophy

- PanDA WMS design goals:
  - Achieve high level of automation to reduce operational effort
  - Flexibility in adapting to evolving hardware and network capabilities
  - Support diverse and changing middleware
  - Insulate user from hardware, network, middleware, and all other complexities of the underlying system
  - Unified system for organized production and user analysis
  - Incremental and adaptive software development
- PanDA and DDM
  - PanDA uses a independent and asynchronous Distributed Data Management system (DDM) called DQ2 in ATLAS
  - DDM is tightly coupled to networking – will not address here

PanDA Basics

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

PanDA Workflow for Production

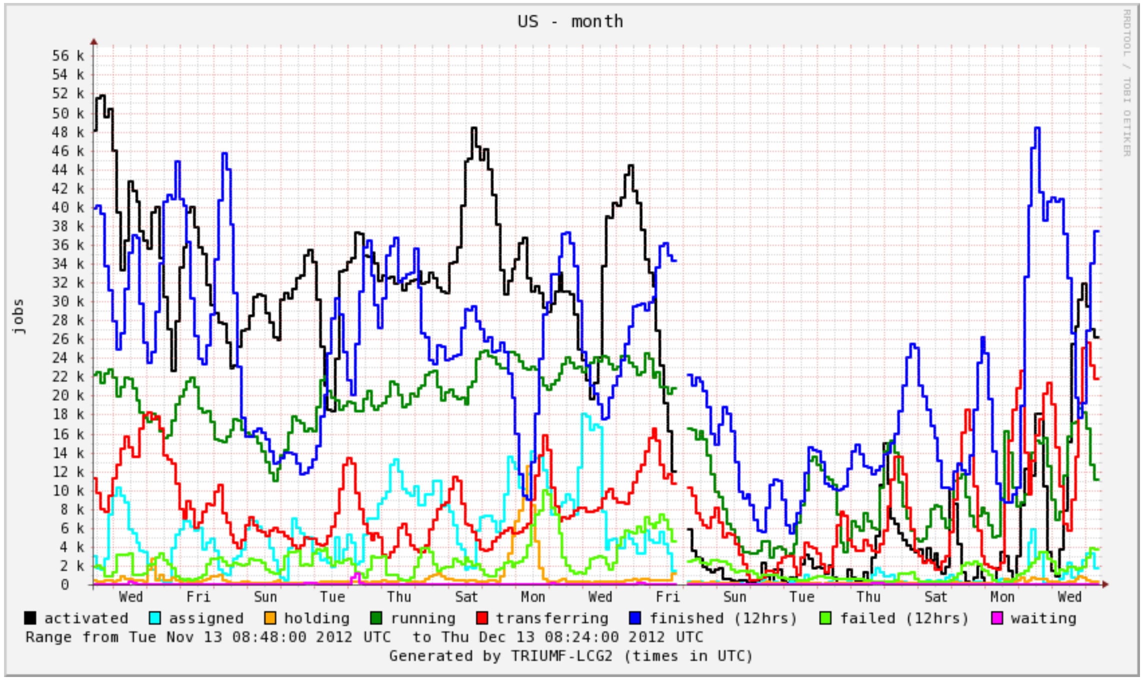


What is a Job

- In PanDA, the 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 is 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



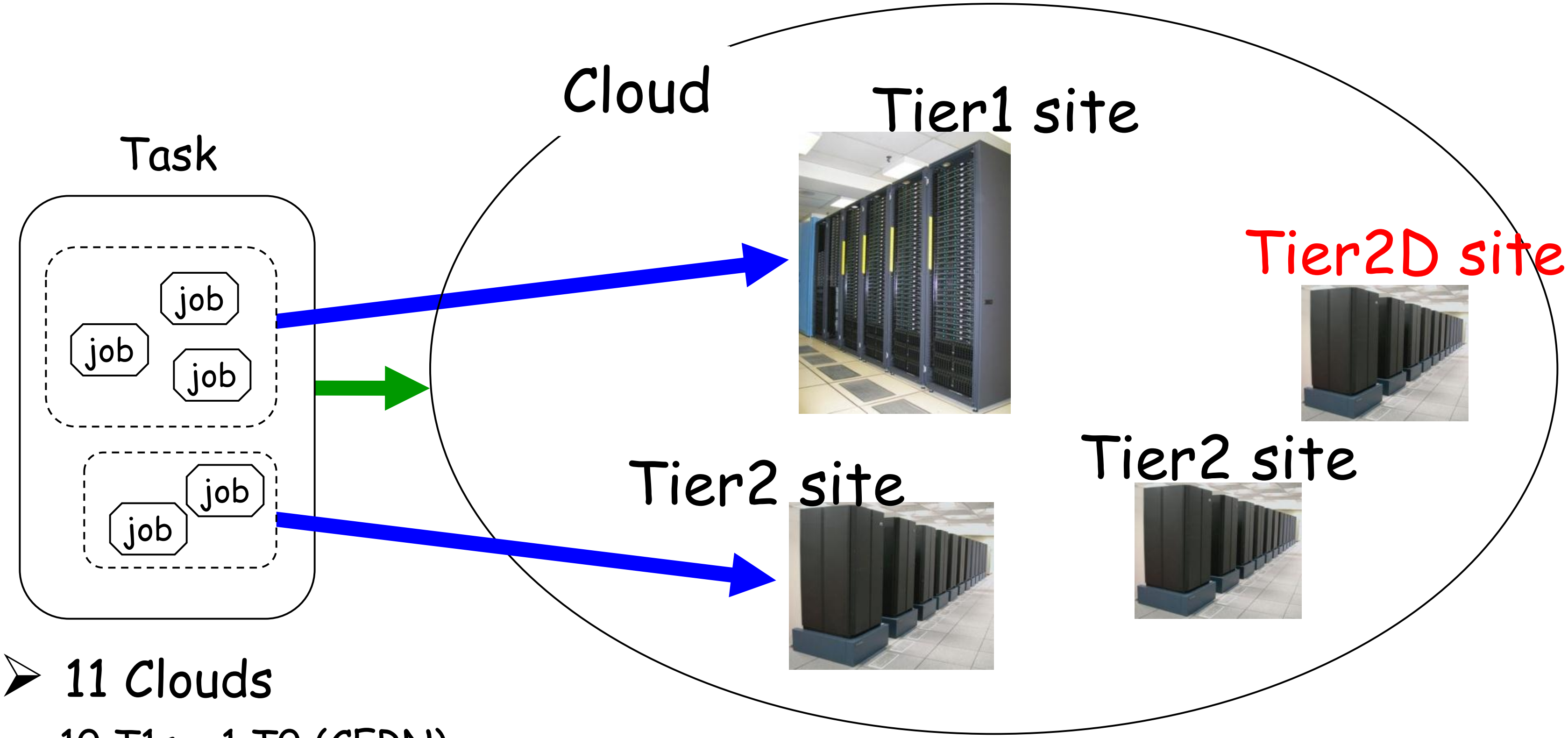
Assigned Jobs

- Assigned -> Activated workflow
  - Group of jobs are assigned to a site by PanDA brokerage
  - For missing input files, data transfer is requested asynchronously
  - PanDA waits for “transfer completed” callback from DDM system to activate jobs for execution
  - Network data transfer plays crucial role in this workflow
- Can network technology help assigned->activated transition?
  - Can we use network provisioning in this step?
  - Jobs are reassigned if transfer times out (fixed duration) – can knowledge of network status help reduce the timeout?
  - Can modification of network path help?

Transferring Jobs

- Transferring state
  - After job execution is completed, asynchronous data transfer is requested from DDM
  - Callback is required for successful job completion
- How can network technology help?
  - Similar questions as assigned state
  - Very long timeout delays completion – can network status info help
  - Can we balance CPU resource vs Network resource
  - At what point can we give up on transfer and rerun the job?

ATLAS Computing Model



Task

job

job

job

job

job

Cloud

Tier1 site

Tier2D site

Tier2 site

Tier2 site

- 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
- Can knowledge of network help
  - Can we consider availability of network as a resource, like we consider storage and CPU resources?
  - What kind of information is useful?
  - Can we consider similar (highlighted)factors for networking?

Job Brokerage

- Brokerage policies define job assignment to sites
  - IO intensive or TAPE read -> prefer 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

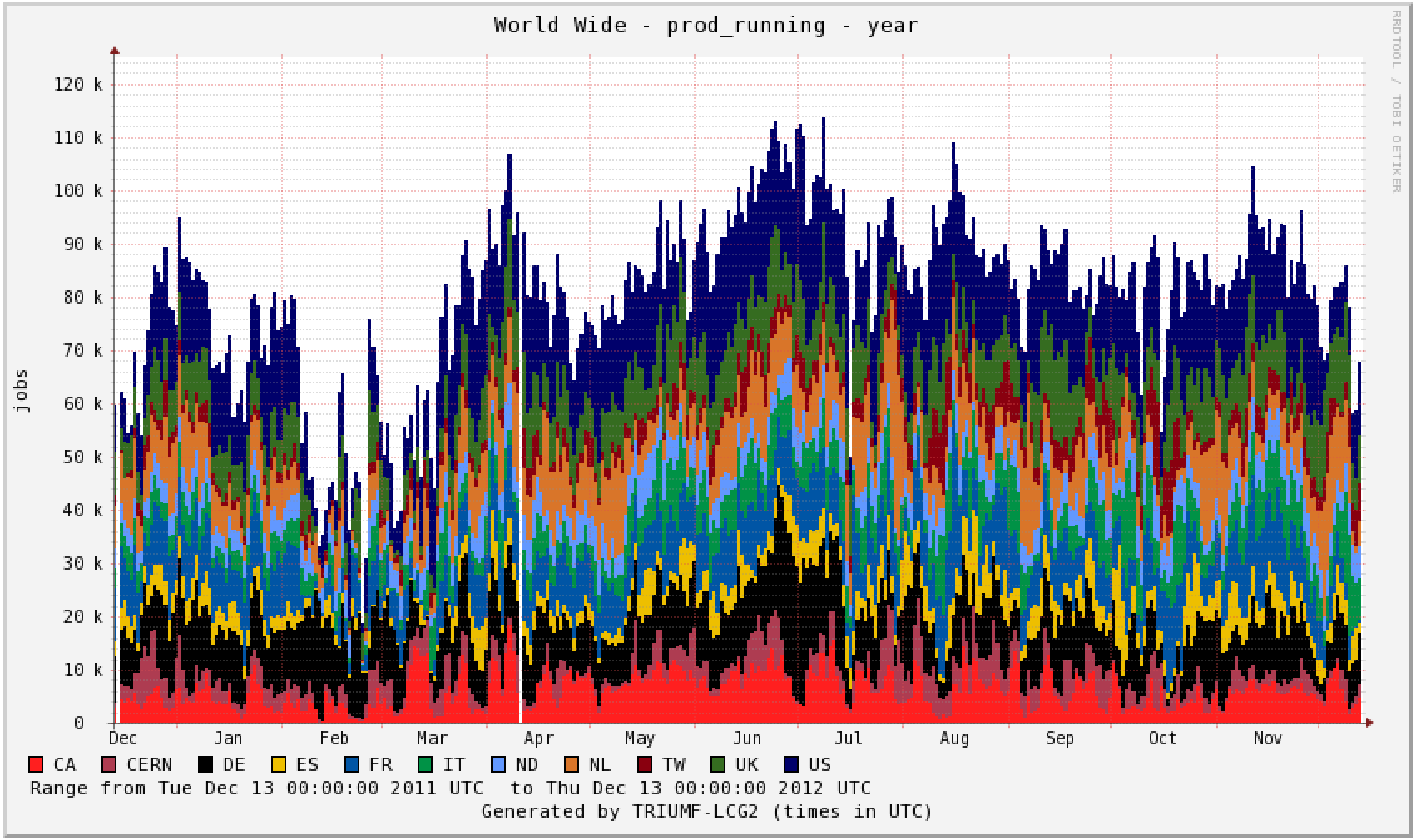
Networking as a Resource

- Job brokerage is critical function of WMS
  - We currently consider storage and CPU resources only in brokerage
  - Networking is assumed – why not treat as a resource to be brokered
  - First, we should try to use network information for site selection
  - Second, can we use provisioning to improve workflow
  - Third, can we improve/modify paths dynamically
  - Plenty of opportunity for future work

Job Dispatcher

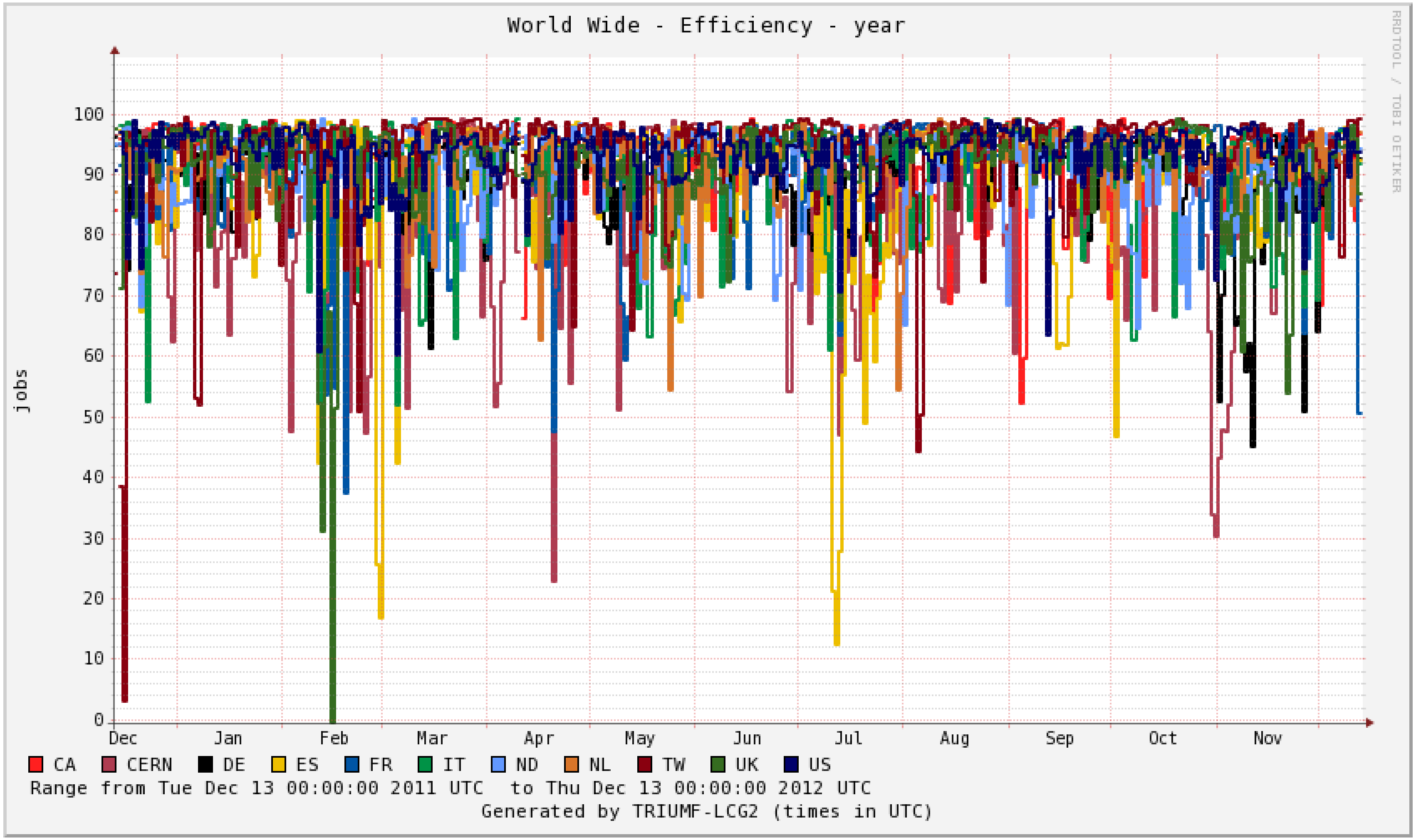
- High performance/high throughput module
- Send matching job to CE upon pilot request
  - REST non-blocking communication
  - Different from brokerage, which is asynchronous
- Matching of jobs based on
  - Data locality
  - Memory and disk space
- Highest priority job is dispatched
- At this point networking is not as important
  - Is this true – we still have to transfer output
  - Can we initiate provisioning?

Performance - Production



Average number of concurrently running jobs per day

Cloud Efficiency

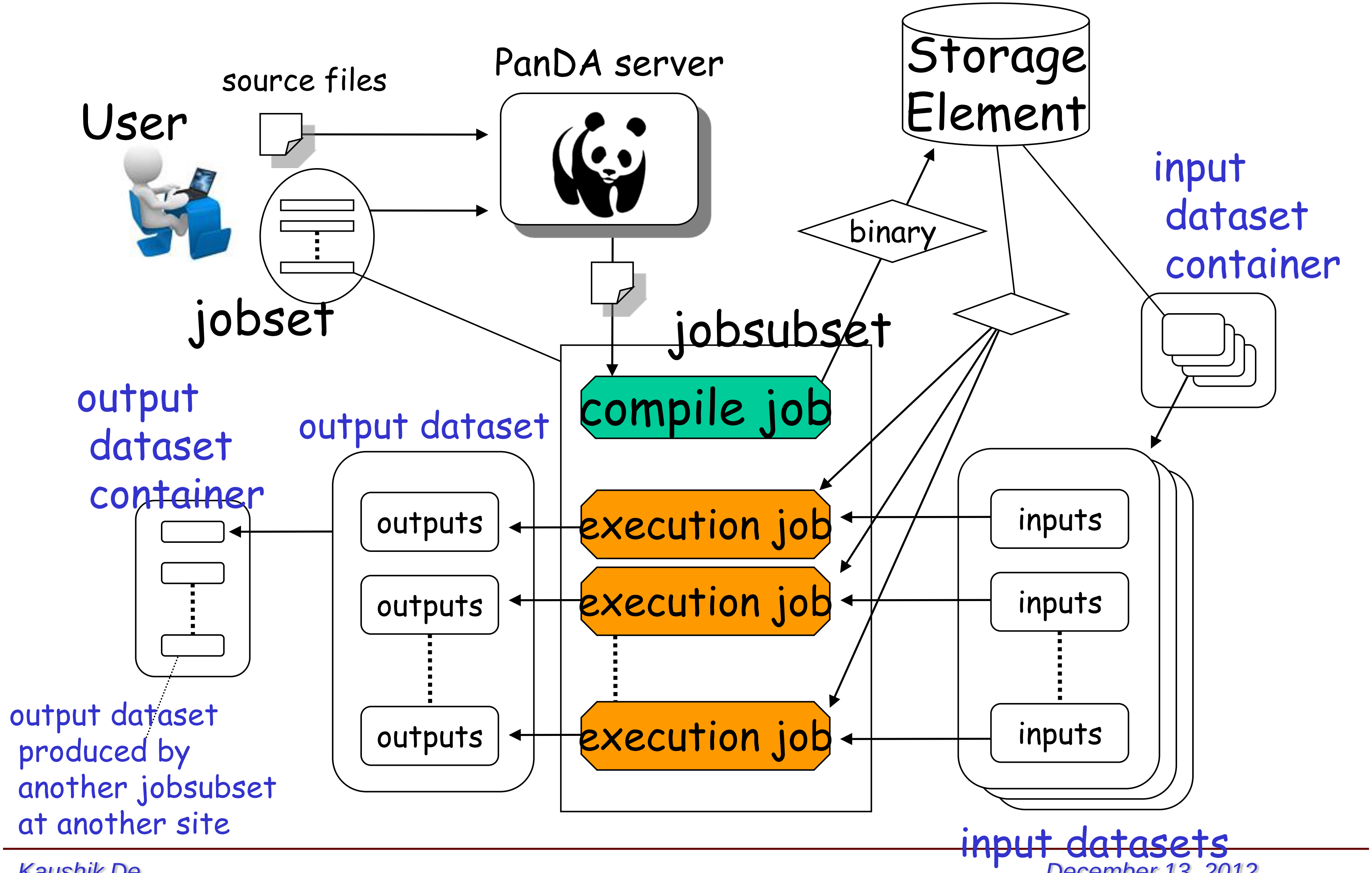


Average efficiency >95% - mostly site & application errors remain

User Analysis in PanDA

- Flexibility in job definition
  - Customization of source by user
  - 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 (can be requested)
  - 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

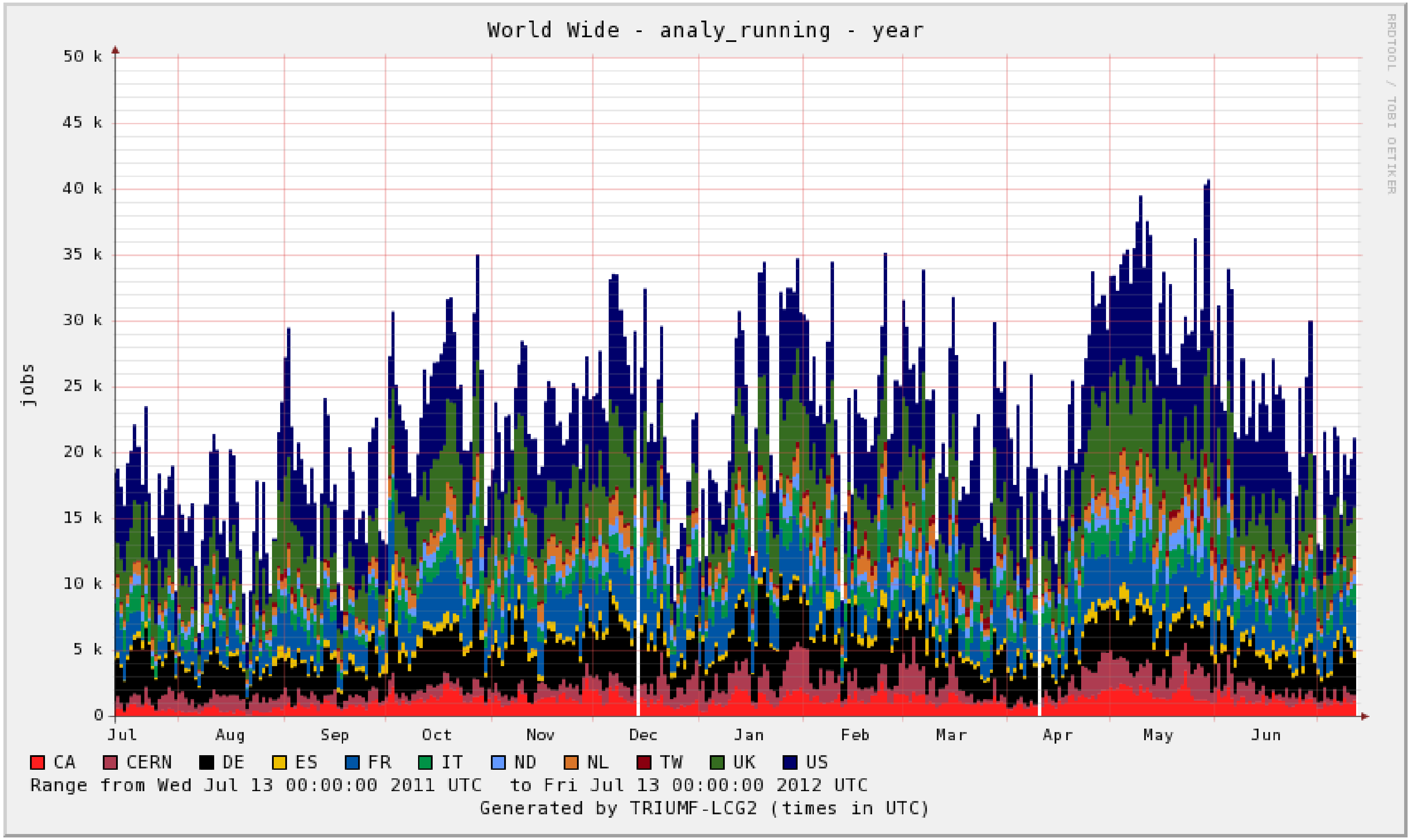
User Analysis Work Flow



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
- Network is not considered in matchmaking
  - Since job goes to data
  - But is this model good enough?
  - Will this be true when we access data over WAN?

Analysis Performance



Average number of concurrently running jobs per day

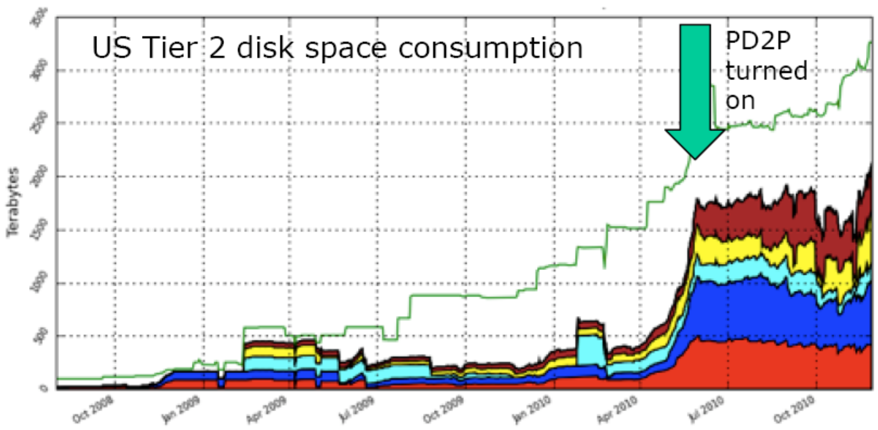
PD2P – How LHC Model Changed

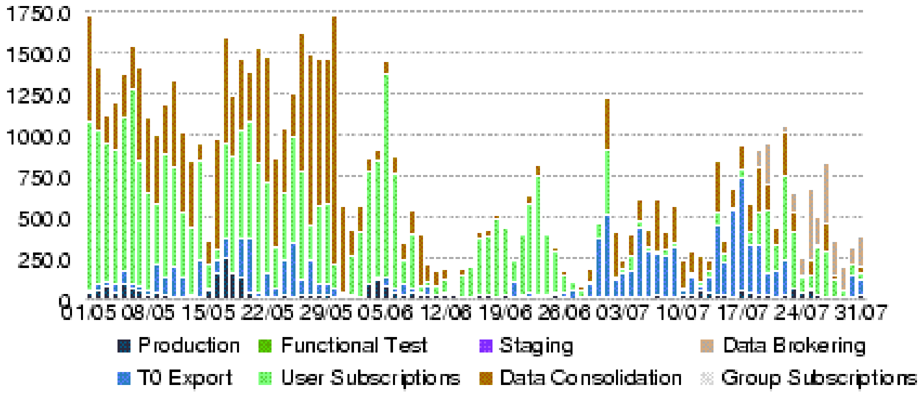
- PD2P = PanDA Dynamic Data Placement
- PD2P used to distribute data for user analysis
  - For production PanDA schedules all data flows
  - Initial ATLAS computing model assumed pre-placed data distribution for user analysis – PanDA sent jobs to data
  - Soon after LHC data started, we implemented PD2P
- Asynchronous usage based data placement
  - Repeated use of data → make additional copies
  - Backlog in processing → make additional copies
  - Rebrokerage of queued jobs → use new data location
  - Deletion service removes less used data
  - Basically, T2 (and now T1) storage used as cache for user analysis

Network vs Storage

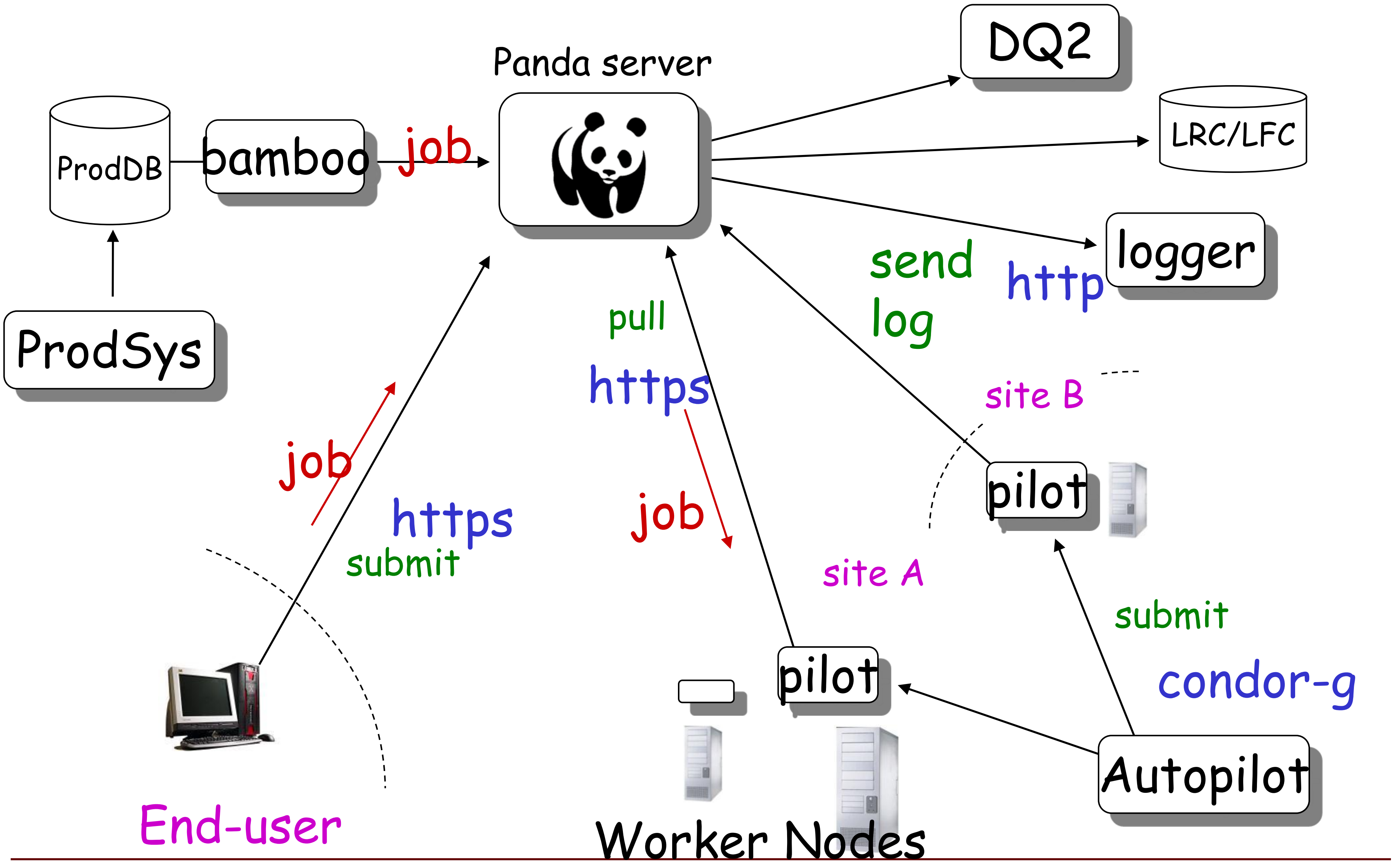
❑ PD2P fundamentally changed computing model

- Decline in storage use
- Change in network usage pattern
- Shown by Michael in ATLAS meeting on Monday





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...
- With Federated Storage deployment network becomes an important resource

FAX Integration with PanDA

- Federated Data Storage will be described in other talks
  - For this talk I assume sites have se tup global redirector to access data over WAN
- We have developed detailed plans for integrating FAX with PanDA
  - Networking plays an important role
  - This time we are paying attention to networking from the beginning

FAX for Fault Tolerance

- Phase I goal
  - If input file cannot be transferred/accessed from local SE, PanDA pilot currently fails the job after a few retries
  - We plan to use Federated storage for these (rare) cases
  - Start with file staging/transfers using FAX
    - Implemented in recent release of pilot, works fine at two test sites
    - Next step – wider scale testing at production/DA sites
- Phase 2
  - Once file transfers work well, try FAX Direct Access
- Phase 3
  - Try FAX for transfer of output files, if default destination fails
- This is now under testing at many sites

FAX for Managed Production

- Managed production has well defined workflow
  - PanDA schedules all input/output file transfers through DQ2
  - DQ2 provides dataset level callback when transfers are completed
- FAX can provide alternate transport mechanism
  - Transfers handled by FAX
  - Dataset level callback provided by FAX
  - Dataset discovery/registration handled by DQ2
- File level callback
  - Recent development – use activeMQ for file level callbacks
  - On best effort basis for scalability – dataset callbacks still used
  - FAX can use same mechanism
- Work in progress

FAX for Distributed Analysis

- Most challenging and most rewarding
  - Currently, DA jobs are brokered to sites which have input datasets
  - This may limit and slow the execution of DA jobs
  - Use FAX to relax constraint on locality of data
- Use cost metric generated with Hammercloud tests
  - Provides ‘typical cost’ of data transfer between two sites
- Brokerage will use ‘nearby’ sites
  - Calculate weight based on usual brokerage criteria (availability of CPU…) plus network transfer cost
  - Jobs will be dispatched to site with best weight – not necessarily the site with local data or available CPU’s
- Cost metric already available – soon to be tested in brokerage

Summary

- In the past WMS assumed:
  - Network is available and ubiquitous
  - As long as we implement timeouts, workflow will progress smoothly
  - Computing models can tell us how to design workflows
- What we learned from the LHC:
  - Flexibility in WMS design is more important than computing model
  - Network evolution drives WMS evolution
  - We should start thinking about Network as resource
  - WMS should use network information actively to optimize workflow
  - Resource provisioning could be important for the future
- The future:
  - **A**dvanced **N**etwork **S**ervices for **E**xperiments (ANSE), NSF funded (Caltech, Michigan, Vanderbilt and U Texas Arlington)
  - Next Generation Workload Management and Analysis System for Big Data, PANDA integration with networking, DOE funded (BNL, U Texas Arlington)