Danila Oleynik (on behalf of BigPanDA team)

BigPanDA: Next Generation Workload Management System for Big Data

Outline

- Introduction
  - PanDA in ATLAS
  - ASCR project
- BigPanDA 2014 achievements
- Summary

PanDA Workload management system

- PanDA project was started in Fall 2005
  - Production and Data Analysis system for ATLAS
  - Goal: An automated yet flexible workload management system (WMS) which can optimally make distributed resources accessible to all users
  - Originally developed in US for US physicists
- Adopted as the ATLAS wide WMS in 2008 (first LHC data in 2009) for all computing applications
  - The ATLAS experiment at the LHC - Big Data Experiment
  - ATLAS DDM manages ~150 PB of data, distributed world-wide at ~130 WLCG computing centers
  - Expected rate of data influx into ATLAS Grid ~40 PB of data per year
  - Thousands of physicists from ~40 countries analyze the data

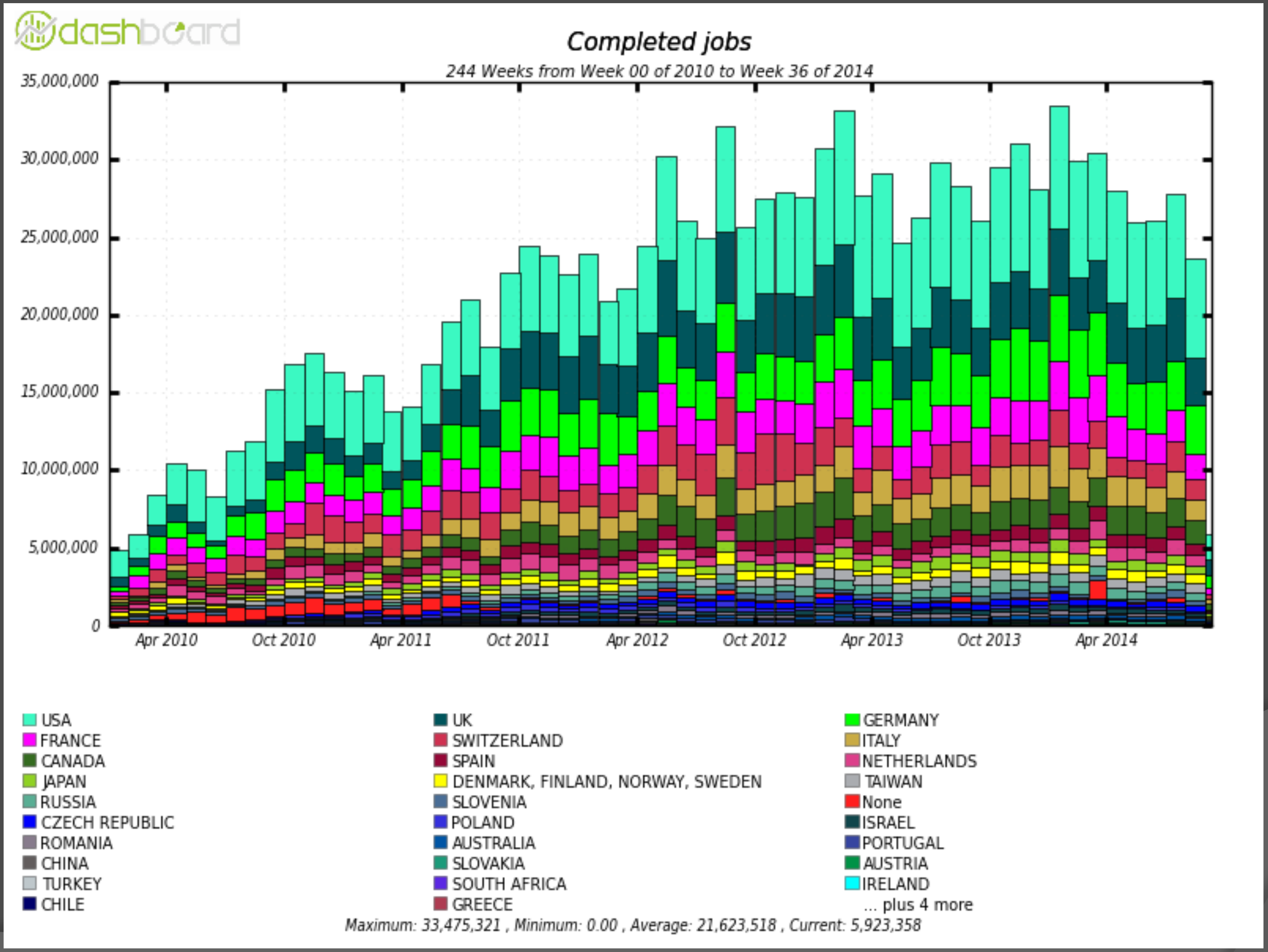
PanDA WMS design goals

- Achieve high level of automation to reduce operational effort
- Flexibility in adapting to evolving hardware and network configurations
- Support diverse and changing middleware
- Insulate user from hardware, middleware, and all other complexities of the underlying system
- Unified system for central MC production, data processing and user data analysis
- Incremental and adaptive software development

Key Features of PanDA

- Workflow is maximally asynchronous
- Pilot based job execution system
  - Condor based pilot factory
  - Payload 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
- HTTP/S RESTful communications
- GSI authentication
- Use of Open Source components

Jobs completed by PanDA per month for ATLAS since 2010



PanDA’s Success

- The system was developed by US ATLAS for US ATLAS
- Adopted by ATLAS Worldwide as Production and Analysis system
- It was proven that PanDA can manage in excess of 1 million jobs per day
- Adopted to evolution in ATLAS computing model
- PanDA is exascale now: 1.2 Exabytes of data processed by PanDA in 2013

Next Generation "Big PanDA"

- ASCR and HEP funded project "Next Generation Workload Management and Analysis System for Big Data". Started in September 2012.
- Generalization of PanDA for HEP and other data-intensive sciences, and a wider exascale community.
- Project participants from ANL, BNL, UT Arlington
- Alexei Klimentov – Lead PI
- **WP1 (Factorizing the core):** Factorizing the core components of PanDA to enable adoption by a wide range of exascale scientific communities
- **WP2 (Extending the scope):** Evolving PanDA to support extreme scale computing clouds and Leadership Computing Facilities
- **WP3 (Leveraging intelligent networks)**: Integrating network services and real-time data access to the PanDA workflow
- **WP4 (Usability and monitoring):** Real time monitoring and visualization package for PanDA

Next Generation “Big PanDA”. Collaborations

- Through ASCR project, PanDA has moved well beyond ATLAS
- Collaboration between ATLAS, ALICE, nEDM experiments for efficient usage of opportunistic resources, especially HPC and LCF
- LSST and AMS uses PanDA for distributed data processing
- Other communities getting involved

WP1 Factorizing the core

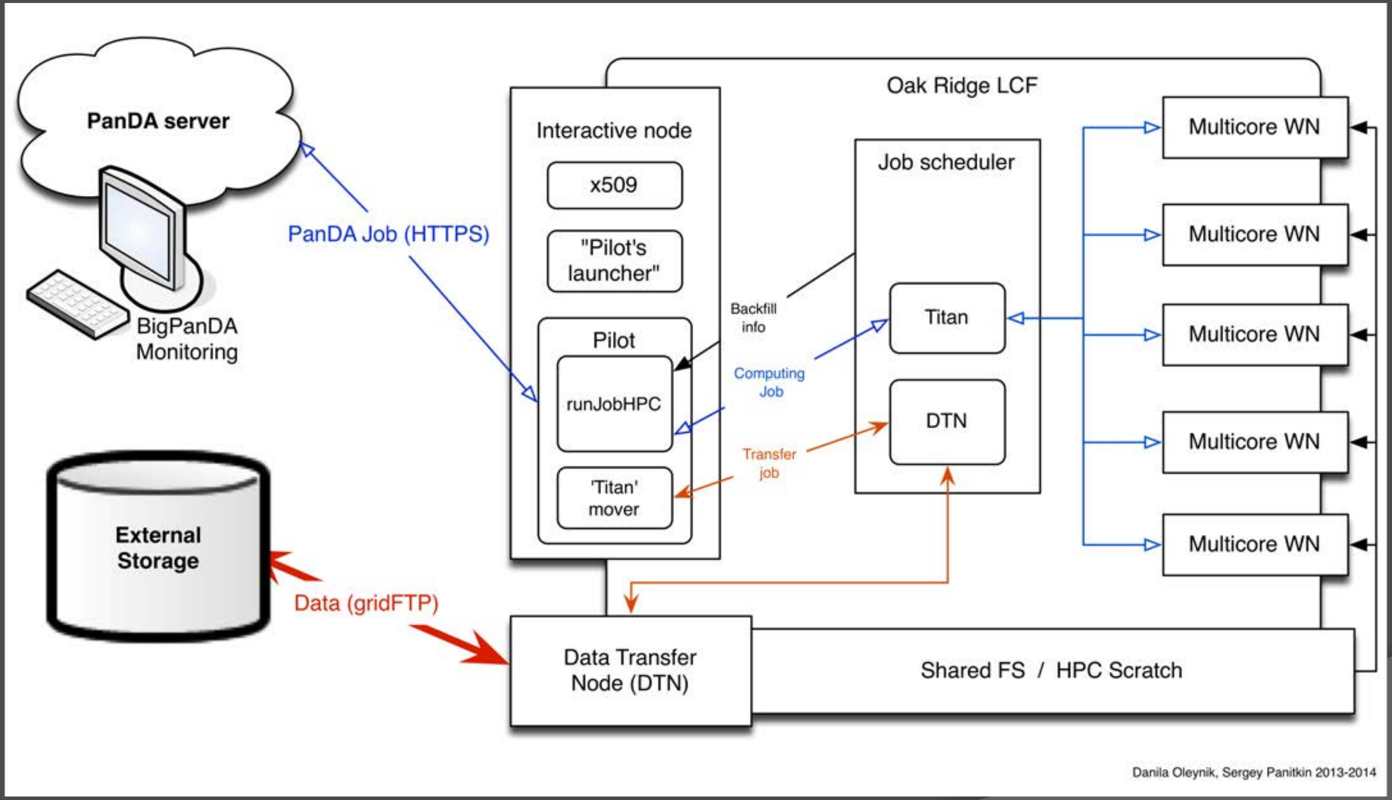
- New Code repository.
  - Migration from CERN SVN to GitHub
  - New build system
  - Distribution through RPMs
- PanDA server improvements.
  - Split core and experiment specific packages
  - Installed dedicated instances for different facilities (now running 3 instances)
- PanDA pilot improvements.
  - Core pilot has been refactored to a generic (VO independent) version;
  - VO specifics are handled as plug-ins;
  - Execution backends are handled as plug-ins
- Multiple database backends.
  - Oracle database backend (ATLAS, AMS)
  - mySQL (running on EC2 PanDA server) – LSST, ALICE
- New PanDA monitoring.
  - Redesigned
  - Refactored to ensure modularity
- New PanDA instance with MySQL backend deployed in Amazon EC2
  - Instance tuned for multi-VO support

WP2 Extending the scope. PanDA @ ORNL LCF.

- Special features of running at LCF
  - Parallel file system shared between nodes.
  - Worker nodes have extremely limited connectivity
  - One-Time Password Authentication
  - Internal job management tool: PBS/TORQUE
  - One job occupy minimum one node (16 cores)
  - Limitation of number of jobs in scheduler for one user
  - Using special data transfer nodes (high speed stage in/out)
- These features developed through ASCR-BigPanDA, going beyond GRID's and clouds

WP2 PanDA architecture for Titan

- Pilot(s) executes on HPC interactive node
- Pilot interact with local job scheduler to manage job
- Data, produced on HPC automatically moves to external storage



Started works for integration with NERSC with using same architecture

WP2 Backfill on HPC

- A lot of applications can effectively use a single core
- HPC is full, means that the system have allocated all the cycles it is able to deliver
  - It is probably not all cycles available
  - Just as there is room for sand in the jar of rocks, there's
- room for "simple" jobs on even a "full" HPC

WP2 PanDA pilot with backfill mode

- PanDA pilot algorithm has been adopted to use backfill information and to submit jobs with optimal parameters
  - Pilot uses SAGA API from complimentary ASCR project
- Special middleware which allows to execute bunch of single node/single threaded workloads transparently, as MPI multi-node workloads, was implemented
- During functional test of Pilot on Titan with realistic payloads (ATLAS and ALICE), efficiency of usage of supercomputer was increased by 2,2%
- We face very small waiting time (avg 70 sec.) for jobs, which ran in backfill mode with optimal parameters

WP3 Leveraging intelligent networks

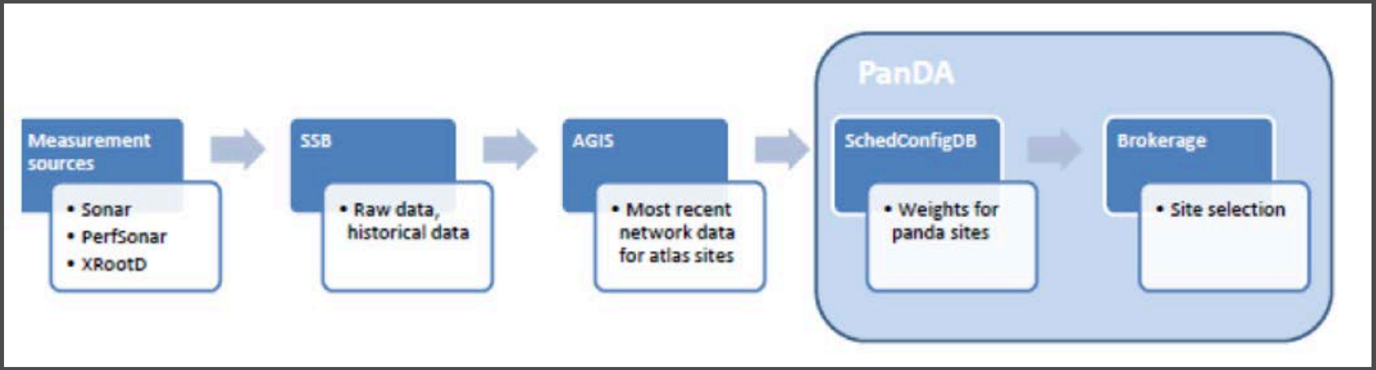
- Why WMS should care about networking?
  - Distributed workload management systems need to transfer data (or use direct access) both for input and output
  - Data transfer/access is done asynchronously
  - Data transfer/access systems can be optimized for network performance – PanDA use these enhancements
  - But network information can also be used directly in workflow management in PanDA at a higher level
- Goal for PanDA
  - Direct integration of networking with PanDA workflow – never attempted before for large scale automated WMS systems
- Main PanDA use cases
  - Use network information for cloud selection
  - Use network information for job assignment
  - Use network information for site selection

WP3 PanDA and Networking. Concept.

- PanDA as workload manager
  - PanDA automatically chooses job execution site
    - Multi-level decision tree – task brokerage, job brokerage, dispatcher
    - Also manages predictive future workflows – at task definition, PD2P (PanDA Dynamic Data Placement)
  - Site selection is based on processing and storage requirements
    - Can we use network information in this decision?
    - Can we go even further – network provisioning?
  - Further – network knowledge used for all phases of job cycle?
- Network as resource
  - Optimal site selection should take network capability into account
    - We do this already – but indirectly using job completion metrics
  - Network as a resource should be managed (i.e. provisioning)
    - We also do this crudely – mostly through timeouts, self throttling

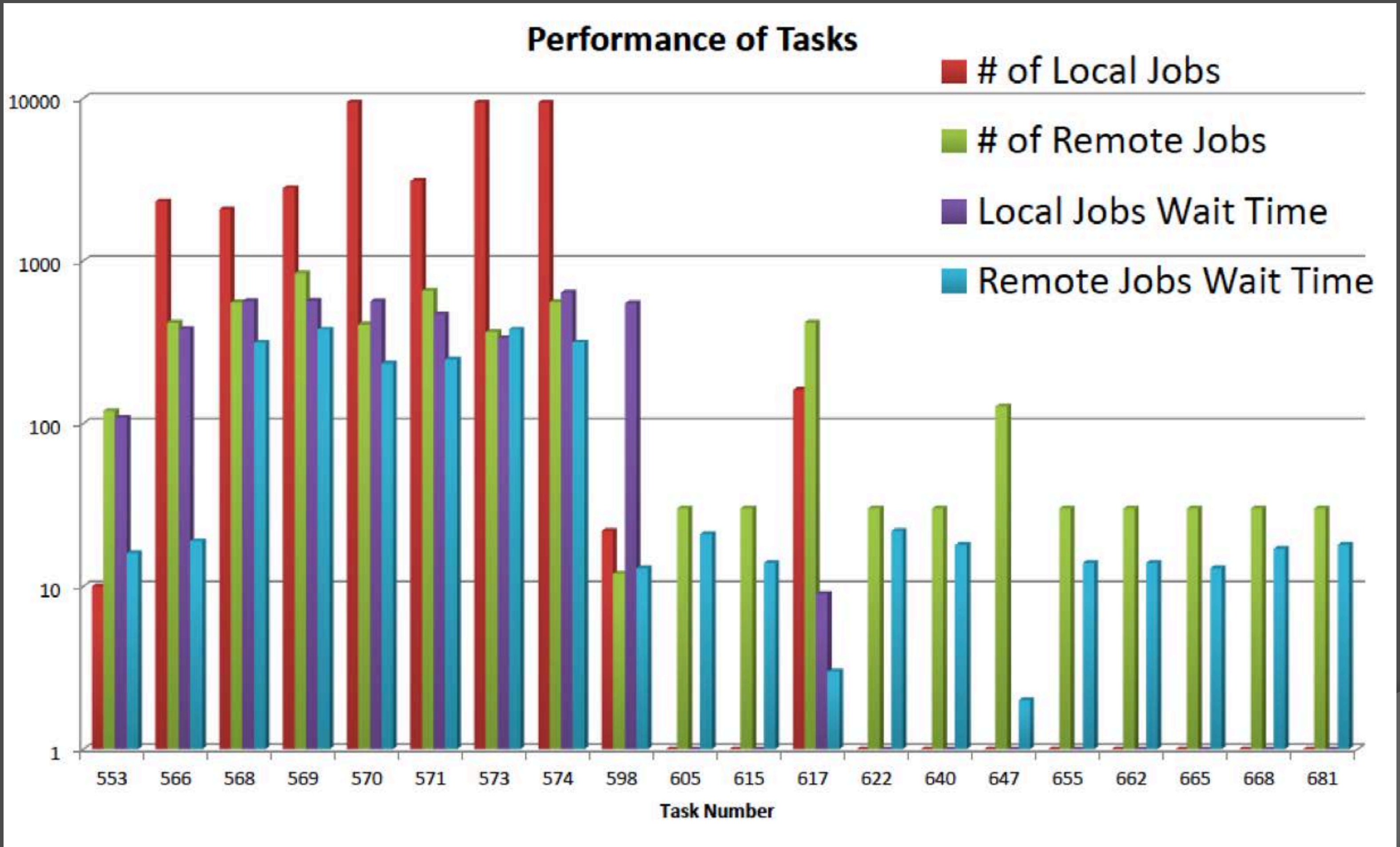
WP3 Network Metrics Implementation.

- PanDA WMS was extended with special components:
  - Collecting network metrics from different providers.
  - Organized collected data and arrange it in proper format
- Brokerage takes these data into account for distribution of jobs to sites



- Initial testing showed significant reduction of jobs waiting time

WP3 Reduction of waiting time

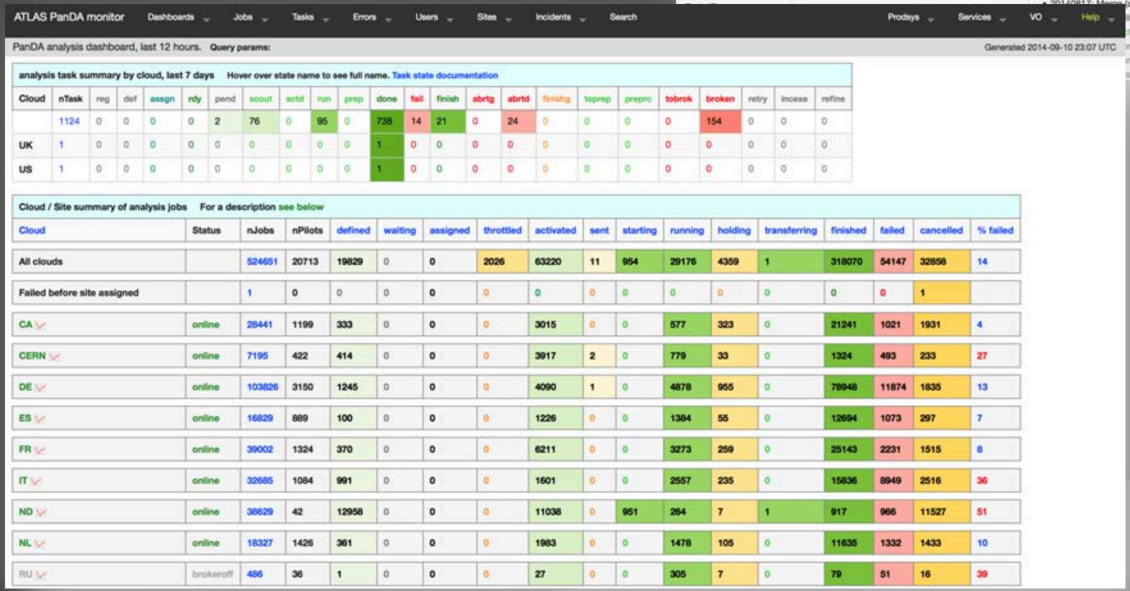


WP4 Usability and monitoring

- Complete analysis of previous implementation of PanDA monitoring, showed that this component should be designed from scratch
- New PanDA monitoring web application was developed based on Django framework
  - Allows rapid development and easy extension

WP4 Usability and monitoring.
New PanDA monitoring

- http://bigpanda.cern.ch
- http://pandawms.org



analysis task summary by cloud, last 7 days Hover over state name to see full name. Task state documentation

Cloud	nTask	reg	def	assgn	rdy	pend	scout	sctd	run	prep	done	fail	finish	abrtg	abrtd	finishg	toprep	preprc	tobrok	broken	retry	incexe	refine
	1124	0	0	0	0	2	76	0	95	0	738	14	21	0	24	0	0	0	0	154	0	0	0
UK	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
US	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Cloud / Site summary of analysis jobs For a description see below

Cloud	Status	nJobs	nPilots	defined	waiting	assigned	throttled	activated	sent	starting	running	holding	transferring	finished	failed	cancelled	% failed
All clouds		524651	20713	19829	0	0	2026	63220	11	954	29176	4359	1	318070	54147	32858	14
Failed before site assigned		1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
CA	online	28441	1199	333	0	0	0	3015	0	0	577	323	0	21241	1021	1931	4
CERN	online	7195	422	414	0	0	0	3917	2	0	779	33	0	1324	493	233	27
DE	online	103826	3150	1245	0	0	0	4090	1	0	4878	955	0	78948	11874	1835	13
ES	online	16829	889	100	0	0	0	1226	0	0	1384	55	0	12694	1073	297	7
FR	online	39002	1324	370	0	0	0	6211	0	0	3273	259	0	25143	2231	1515	8
IT	online	32685	1084	991	0	0	0	1601	0	0	2557	235	0	15836	8949	2516	36
ND	online	38629	42	12958	0	0	0	11038	0	951	264	7	1	917	966	11527	51
NL	online	18327	1426	361	0	0	0	1983	0	0	1478	105	0	11635	1332	1433	10
RU	brokeroff	486	36	1	0	0	0	27	0	0	305	7	0	79	51	16	39

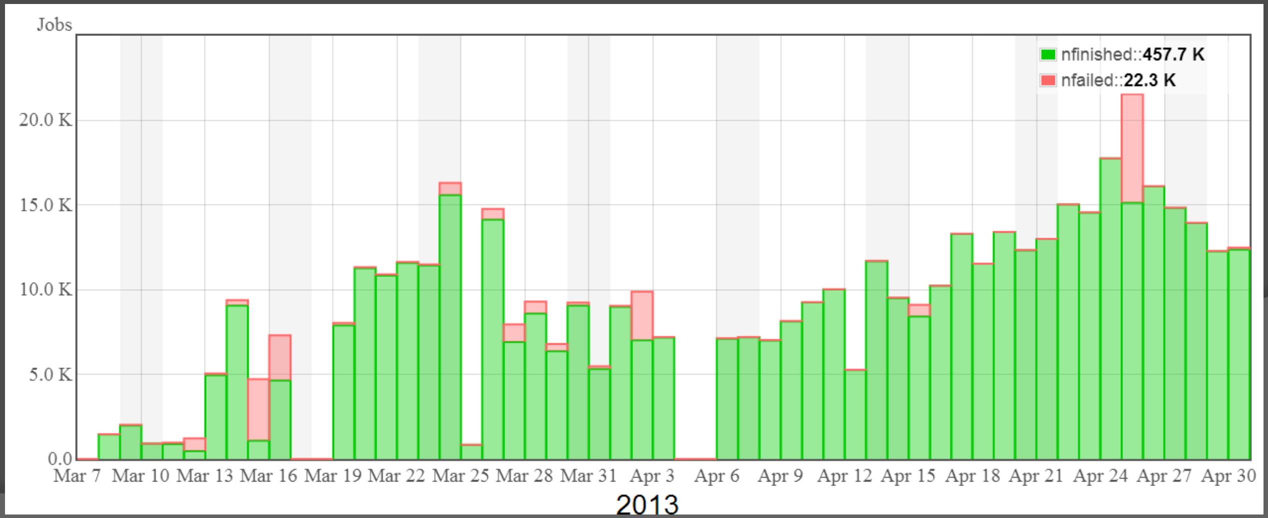
Summary

Significant progress in all work packages. The project is on schedule

- WP1 Factorizing the core:
  - New GitHub repository is in place. New build system.
  - New PanDA server distribution and installation mechanism via rpm
  - Code refactoring well underway for all components
  - Dedicated PanDA server instance running on EC2 with MySQL backend
- WP2 Extending the scope:
  - Successful integration of PanDA with Titan at OLCF
  - Multiple ATLAS and ALICE workloads ported to Titan
  - Integration of unused resources harvesting capability ("backfill") in PanDA pilot
  - Port of the setup developed on Titan started at NERSC
- WP3 Leveraging intelligent networks:
  - First ever usage of network metrics in distributed workload management system
  - Significant reduction of waiting time for jobs
- WP4 Usability and monitoring:
  - New PanDA monitoring

Backup. PanDA @ GCE

- In 2013 PanDA was successfully tested with Google Compute Engine
  - We ran for about 8 weeks
  - Very stable running on the Cloud side. GCE was rock solid.
  - We ran computationally intensive jobs
  - Physics event generators, Fast detector simulation, Full detector simulation
  - Completed 458,000 jobs, generated and processed about 214M events



Backup. PanDA@OLCF. Functional testing results (June-July)

- In May 2014 we ran first 24 hour continuous job submission test via PanDA@EC2 with pilot in backfill mode, with MPI wrappers for two workloads from ATLAS and ALICE
  - Stable operations
  - ~22k core hours collected in 24 hours
  - Observed encouragingly short job wait time on Titan ~4 minutes
- Ran second 24 hour test in July 2014, with pilot modifications that were based on information obtained from the first test
  - Limit on number of nodes was removed in pilot
  - Job wait time limit introduced – 5 minutes. Simple "kill job"
  - ~145k core hours collected
  - Average wait time ~70 sec
  - Observed IO related effects that need to be understood better

Backup. PanDA@OLCF. Functional testing results (August)

- Testing algorithm for internal rescheduling of payload in pilot
  - Pilot gets free resource information from Titan's resource manager
  - Forms job parameters according to free resources and queue policies
  - Submits job to PBS
  - If job exceeds wait time limit, pilot cancels the job and repeats the cycle
- Wait time limit for a job in PBS was set to 2 minutes
- Ran continuously for ~10 hours
- Highly CPU bound payload to avoid IO issues
- Were able to collect ~ 200,000 core hours
- Max number of nodes per job – 5835 (93360 cores)
  - Close to entire Grid in size!
- Used ~2.3% of all Titan core hours or ~14.4% of free core hours