Grid Monitoring Projects

Each project funded by the HGF-A with 0.5 FTE

Poster Session, Mid-Term Review 30.11&1.12.2009

PHYSICS AT THE TERA SCALE
Helmholtz Alliance

Getting all required information for a grid site is complicated

Monitoring information is not clearly arranged, there are:

- Many sources of valuable information
- Different information displays provided by different technologies

Totality of all monitoring systems is uncomfortable to use, you have to:

- Manage many browser tabs / windows
- Change the settings of the web interfaces (time range, site, …)
- Long waiting time until page opens up, often more than 30 seconds

Consequences:

- Unnecessary increase of administration effort for a grid site
- Difficult to identify correlations
- Nearly impossible to get a quick overview on a site's status for non experts, especially if several services at different sites are involved

Idea to ease administration: Meta-monitoring

Such a framework should:

- Collect and process all important monitoring information
- Present the current status of a grid site and its services
- Display simple rating / warning system (smiley faces, arrows, …)

Design properties:

- Framework has a modular layout: There is a static core that provides the basic functionality for the dedicated tests. The individual tests can be plugged in.
- Decoupling of collecting the information and the actual visualisation
- All information is accessible via a single website, including a history
- Visualisation should provide a smart and quick overview on the monitored service which also allows to identify correlations

The HappyFace Project provides such a smart summary of existing information

Site Specific Monitoring

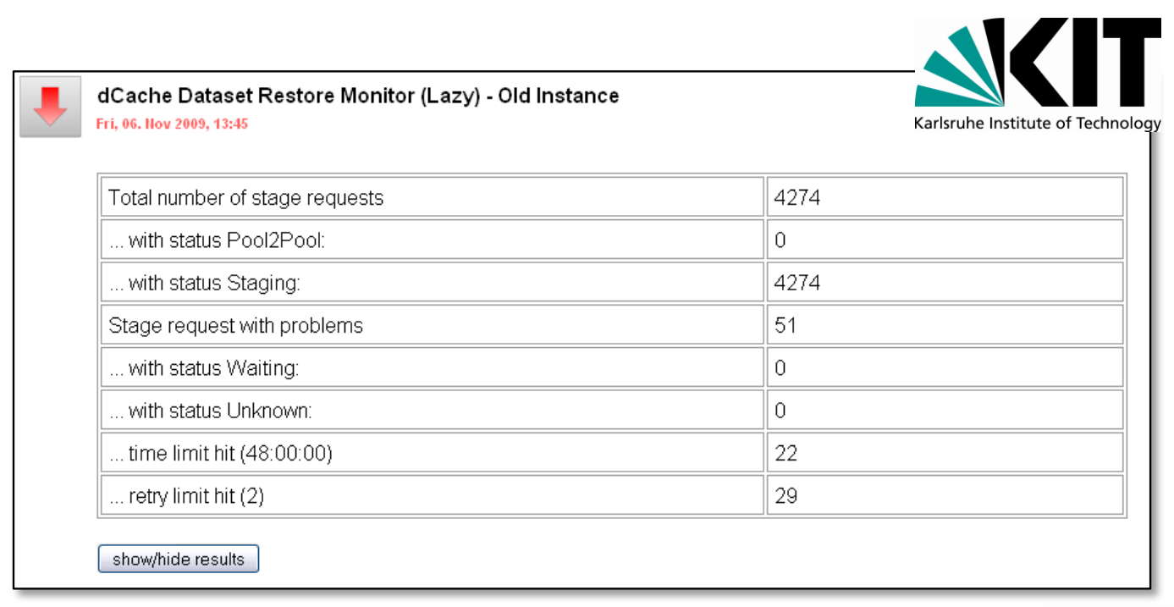
The HappyFace Project

of Multiple Information Systems

Selected modules

dCache Dataset Restore (Lazy)

- Processing of the dCache Dataset Restore Monitor web page
- Possibility to define thresholds of staging requests with problems
  - Time limit hit
  - Retry limit hit
  - Status waiting



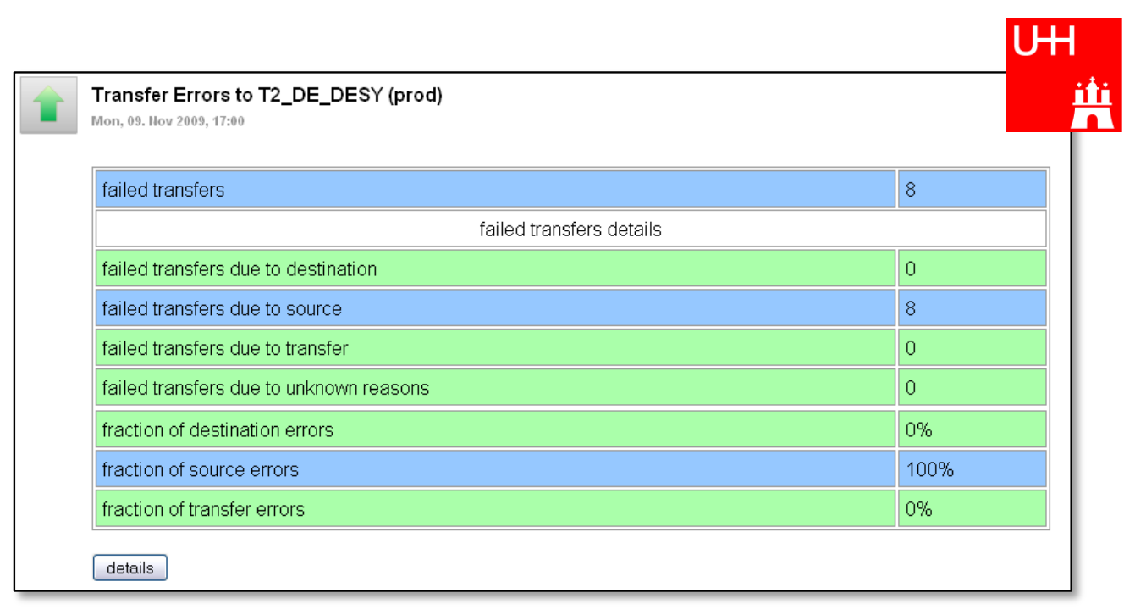
dCache Dataset Restore Monitor (Lazy) - Old Instance
Fri, 06. Nov 2009, 13:45

Total number of stage requests	4274
... with status Pool2Pool:	0
... with status Staging:	4274
Stage request with problems	51
... with status Waiting:	0
... with status Unknown:	0
... time limit hit (48:00:00)	22
... retry limit hit (2)	29

show/hide results

CMS PhEDEx Transfer Errors

- Parses the XML provided by the PhEDEx server
- Module distinguishes between source, destination, transfer and unknown error types
- Detailed information provided as sub table
- Error/warning thresholds are fully configurable



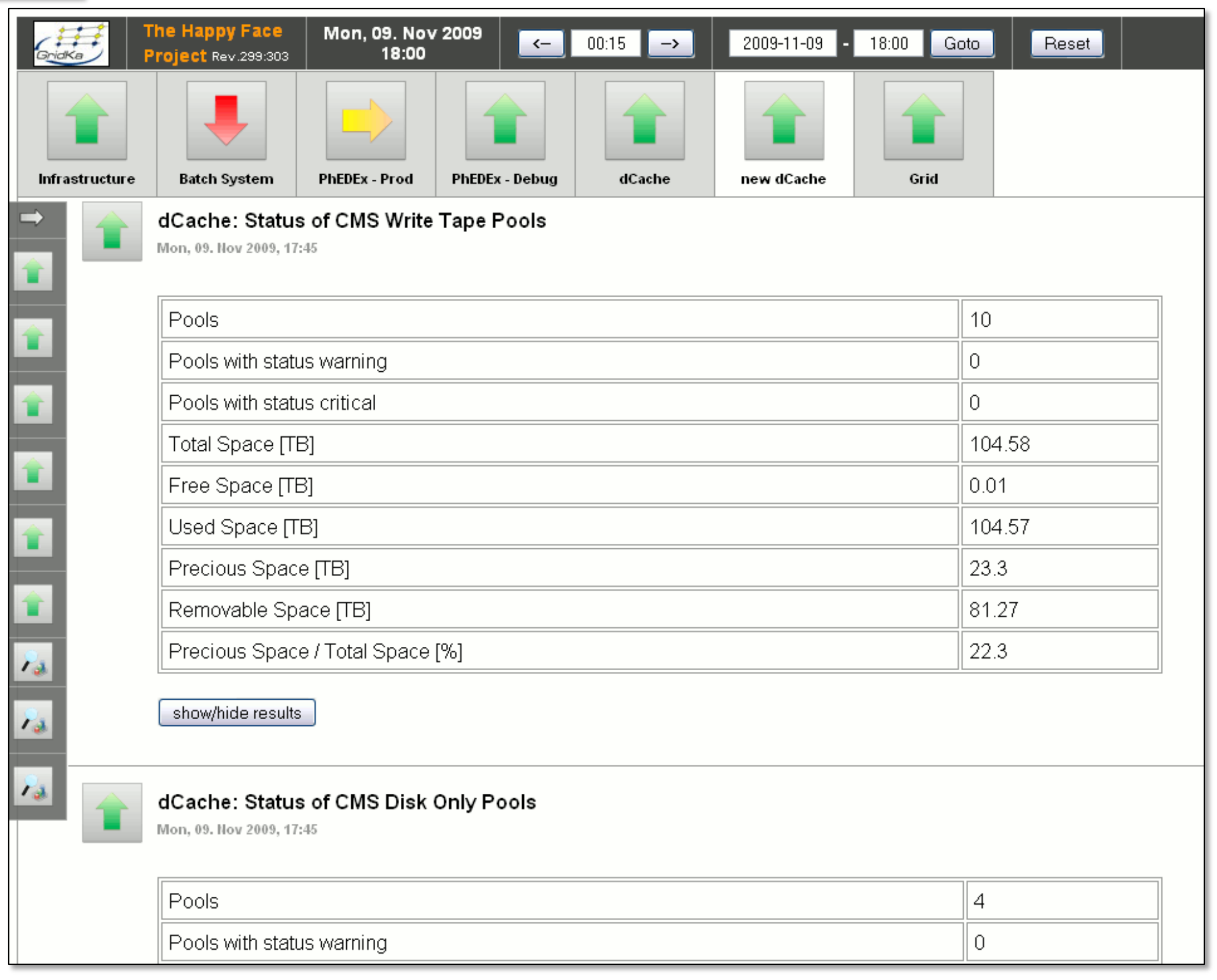
Transfer Errors to T2_DE_DESY (prod)
Mon, 09. Nov 2009, 17:00

failed transfers	8
failed transfers details	
failed transfers due to destination	0
failed transfers due to source	8
failed transfers due to transfer	0
failed transfers due to unknown reasons	0
fraction of destination errors	0%
fraction of source errors	100%
fraction of transfer errors	0%

details

Architecture

- The HappyFace Core provides all basic functionality needed by all tests and organises the test execution
- Each test is represented by a module, which can be plugged in
- Each module can be activated/arranged in the global configuration
- Core and all modules available on a central subversion repository
- Development of the modules in Aachen, Goettingen, Hamburg and Karlsruhe
- HappyFace used for the monitoring at 5 ATLAS/CMS sites



dCache: Status of CMS Write Tape Pools
Mon, 09. Nov 2009, 17:45

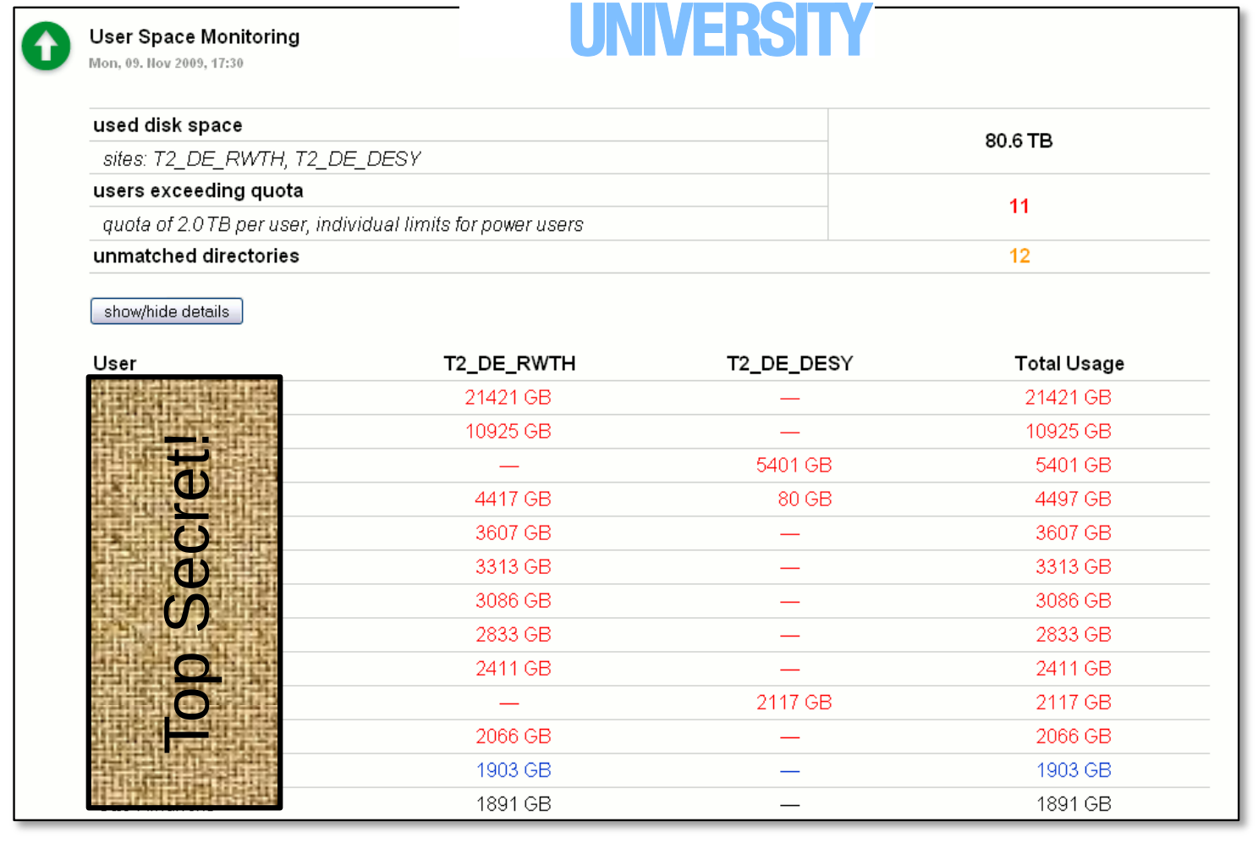
Pools	10
Pools with status warning	0
Pools with status critical	0
Total Space [TB]	104.58
Free Space [TB]	0.01
Used Space [TB]	104.57
Precious Space [TB]	23.3
Removable Space [TB]	81.27
Precious Space / Total Space [%]	22.3

show/hide results

dCache: Status of CMS Disk Only Pools
Mon, 09. Nov 2009, 17:45

Pools	4
Pools with status warning	0

The Karlsruhe HappyFace Instance



User Space Monitoring
Mon, 09. Nov 2009, 17:30

used disk space sites: T2_DE_RWTH, T2_DE_DESY	80.6 TB
users exceeding quota quota of 2.0 TB per user, individual limits for power users	11
unmatched directories	12

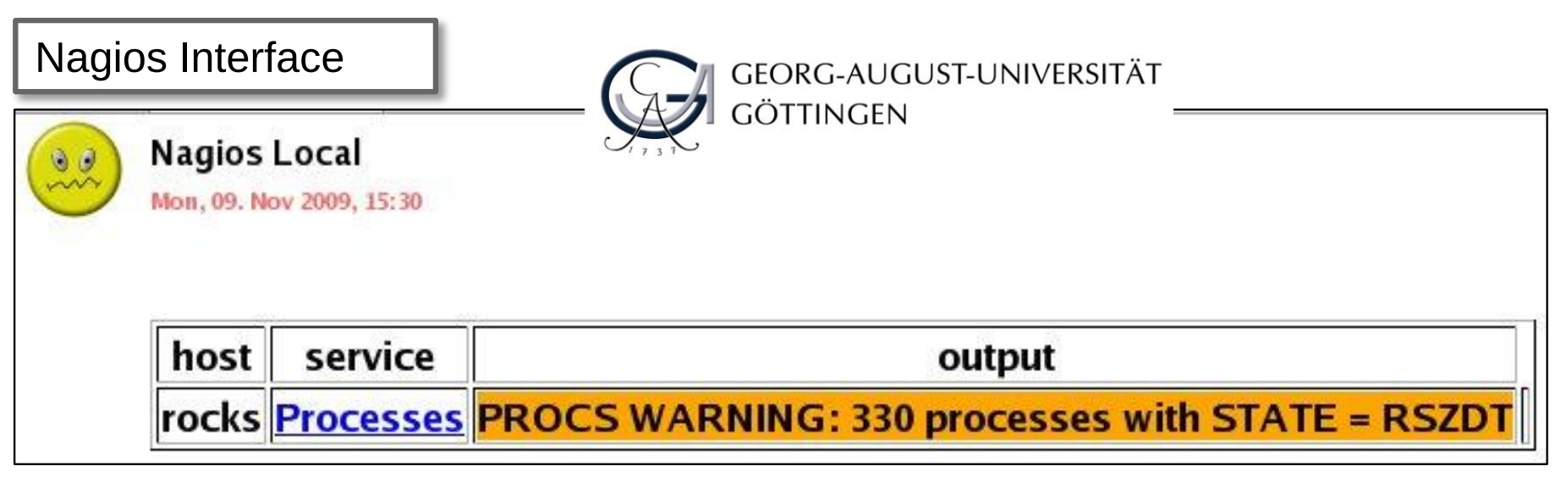
show/hide details

User	T2_DE_RWTH	T2_DE_DESY	Total Usage
Top Secret!	21421 GB	—	21421 GB
	10925 GB	—	10925 GB
	—	5401 GB	5401 GB
	4417 GB	80 GB	4497 GB
	3607 GB	—	3607 GB
	3313 GB	—	3313 GB
	3086 GB	—	3086 GB
	2833 GB	—	2833 GB
	2411 GB	—	2411 GB
	—	2117 GB	2117 GB
	2066 GB	—	2066 GB
	1903 GB	—	1903 GB
	1891 GB	—	1891 GB

T2 User Space Monitoring

- Information about used disk space per user exported via xml
- The HappyFace module reads in and processes these xml files per site
- Plan: Provide certificate based access to this information

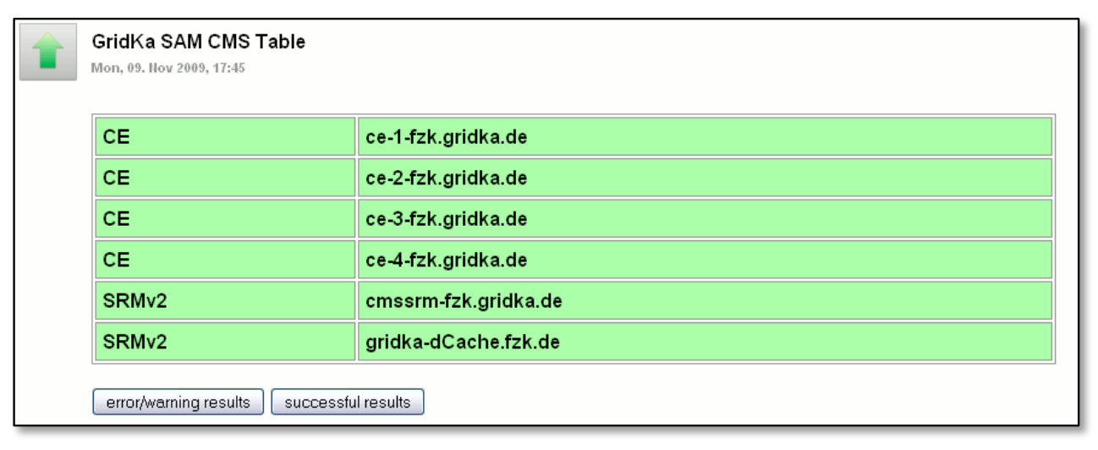
Nagios Interface



Nagios Local
Mon, 09. Nov 2009, 15:30

host	service	output
rocks	Processes	PROCS WARNING: 330 processes with STATE = RSZDT

- Summarizes warnings and error messages of Nagios monitoring
- Combines advantages of Nagios (lots of modules, including modules from EGEE) and HappyFace (lightweight, clear)
- Communication via ssh



GridKa SAM CMS Table
Mon, 09. Nov 2009, 17:45

CE	ce-1-fzk.gridka.de
CE	ce-2-fzk.gridka.de
CE	ce-3-fzk.gridka.de
CE	ce-4-fzk.gridka.de
SRMv2	cmssrm-fzk.gridka.de
SRMv2	gridka-dCache.fzk.de

error/warning results | successful results

SAM Test Results

- Summary of the SAM tests for a site
- Supports experiment specific and ops test
- Sub tables for summary of test results

Job Execution Monitor

Wuppertal

User centric real-time grid job monitoring for the WLCG

*To deal with failures of user jobs on the LHC Computing Grid (LCG), the University of Wuppertal has developed the **Job Execution Monitor** (JEM), providing new possibilities to find problems in largely distributed computing grids and to allow for analyzing these problems in **nearly real-time**. JEM was developed primarily to aid physicists using the software framework **Athena** to simulate and analyse collision event data of the ATLAS detector, one of the four big experiments at the Large Hadron Collider (LHC) at CERN, Geneva.*

JEM monitoring functionality

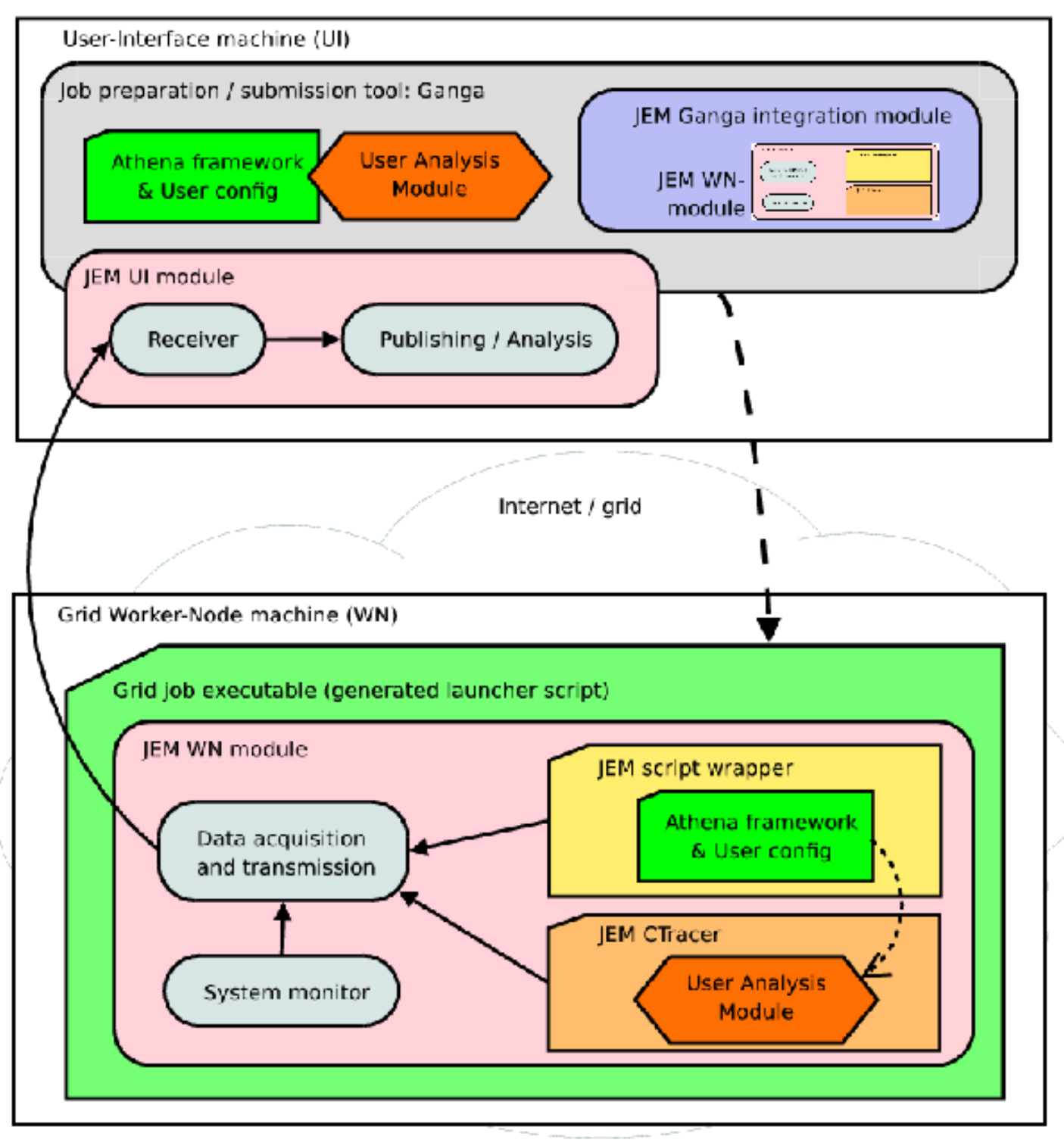
Data gathered by the Job Execution Monitor includes

- Detailled execution flow in **shell**- and **python scripts**, as well as in **compiled binary modules** by means of a novel technology named **C-Tracer**
- Periodic measurement of worker node system metrics like **CPU**- and **RAM-Usage**, network traffic, disk usage, etc.
- Monitoring of **job progress**: Job start-, end- and „next physics event"-notifications
- Real-time **peeking** in the job's (stdout/-err) output

Using JEM, the user is able to trace problems in Athenas environment setup, helper functions and in the physics analysis itself **during the job's execution**. For the ease of use, JEM has recently been transparently integrated into ATLAS' grid job submittage and management-tool, **Ganga**.

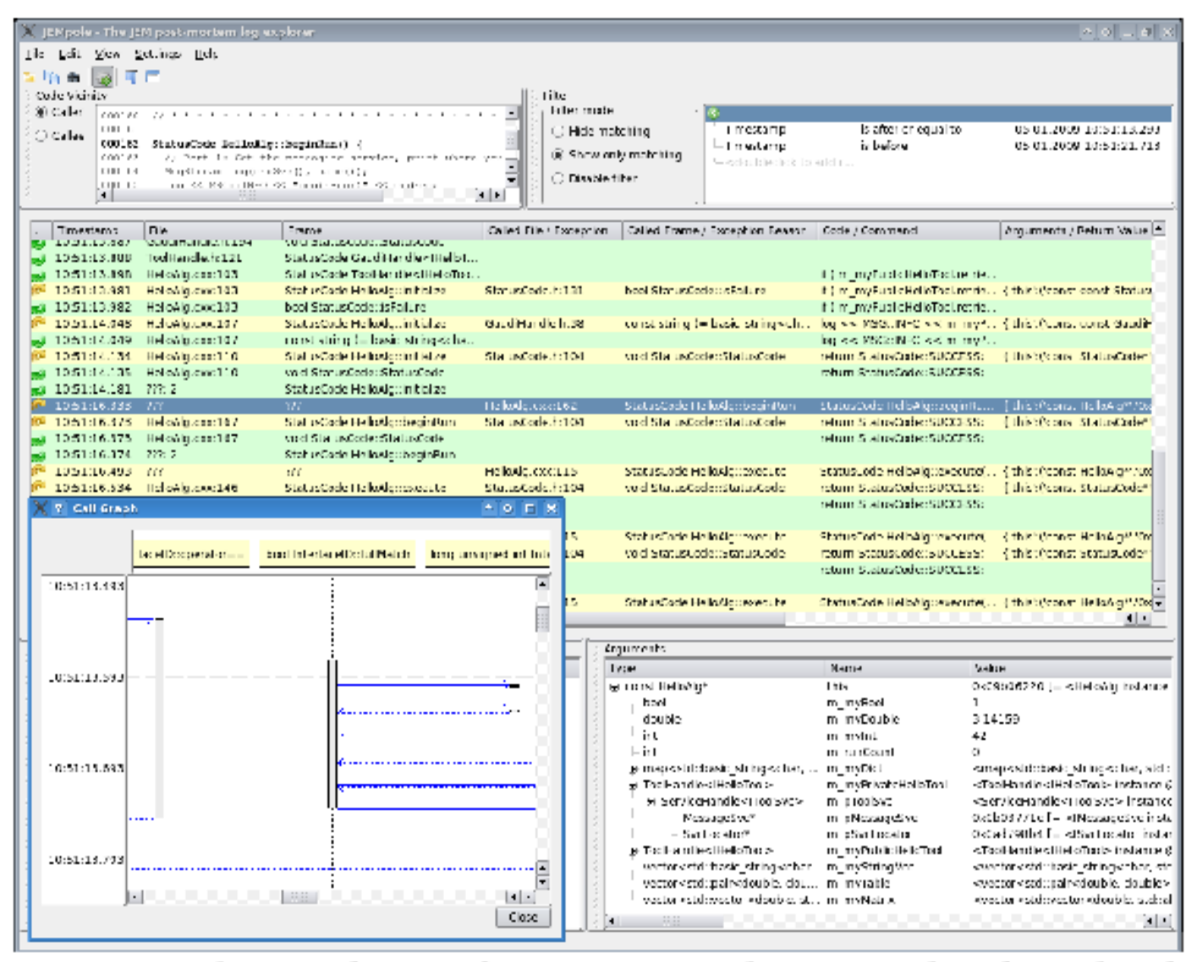
Architecture of the monitoring software

The Job Execution Monitor is a distributed application, implemented in python, consisting of two main parts. One part is run in user space on the grid UI machine (the machine used to submit jobs to the grid) and the other part is submitted alongside the user jobs to grid computing elements (CEs).



User-Interface machine (UI)
Job preparation / submission tool: Ganga
Athena framework & User config
User Analysis Module
JEM Ganga integration module
JEM WN-module
JEM UI module
Receiver
Publishing / Analysis
Internet / grid
Grid Worker-Node machine (WN)
Grid job executable (generated launcher script)
JEM WN module
Data acquisition and transmission
System monitor
JEM script wrapper
Athena framework & User config
JEM CTracer
User Analysis Module

Submitting a monitored Athena job. JEMs worker node module is run instead of the Athena-launcher, starting its services, and then spawning the user analysis algorithm. At the same time, on the grid user interface, the receiving module of JEM is run, that presents the data to the user.



To aid the user in the analysis of monitoring data, we developed a log viewer application providing useful functions like color-coding the data, sorting and filtering, browsing of user job memory, call- and dependency-graphs, etc

All monitoring data gathered during the job run is transmitted nearly in **real-time** to the UI machine to allow for **direct analysis**, as well as for **post-mortem job failure analysis** even if all of the job's output was discarded by the grid middleware as a result of the failure.

Using JEM, **valuable grid resource usage** can be optimized:

- By aborting, fixing and re-submitting **faulty jobs** as soon as an error is discovered, as opposed to after the job finished execution (possibly after hours of wasted CPU time)
- By finding the reason for jobs **hanging / never completing**
- By discovering the cause of job **crashes**.

HELMHOLTZ ASSOCIATION | RWTH AACHEN UNIVERSITY | GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN | UHH | KIT Karlsruhe Institute of Technology | BERGISCHE UNIVERSITÄT WUPPERTAL