Monitoring for analysis jobs and computing infrastructure

Jordi Nadal

II. Institute of Physics
Georg-August University Göttingen

03/12/2013

Outline

- Motivation
- Monitoring for cluster infrastructure:
    - **HappyFace**
    - **Smart Monitoring System**
    - **Visualisation of Big Data**
- Monitoring for the user analysis:
    - **Job Execution Monitoring**

Motivation

- The importance of monitoring is growing due to:
    - Large computing facilities consist of heterogeneous hardware and provides a plethora of services
    - The status check of hardware and services as well as proper action taking are the main task for administrators
    - Increasing complexity of the computing systems

The HappyFace project

- The **HappyFace** project is a meta-monitoring framework that aggregates, processes and stores remote and local site information from different monitoring sources.

- Joint collaboration within the DE cloud in terms of the module development. The new core has been developed at KIT (CMS) by Gregor Vollmer and will be further developed by the Georg-August-Universtität Göttingen (ATLAS).

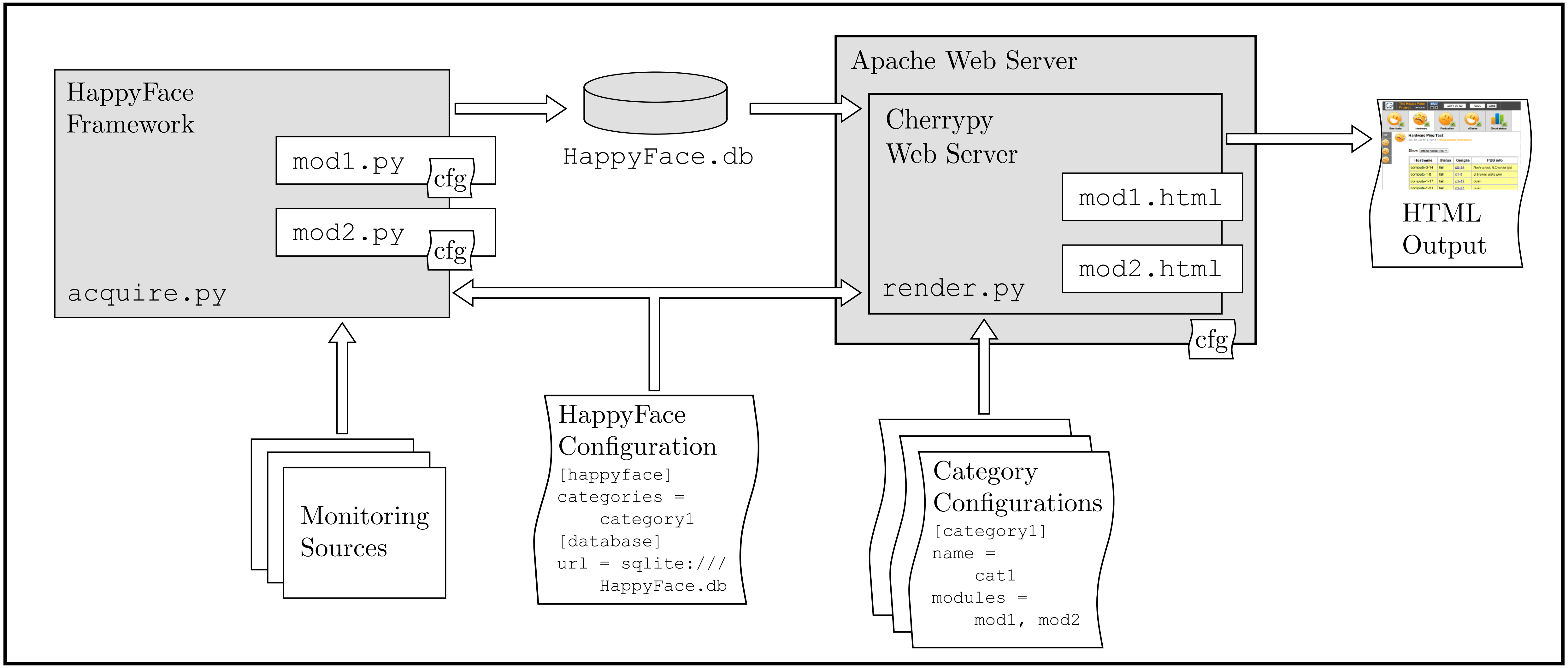
General requirements

- Fulfilling all requirements for any monitoring system:
  - **Scalability** - it does neither depend on the size nor on the possible increase/grow of the computing infrastructure. HP is used for the CMS Tier-1 (DE) and Tier-2 centres
  - **Extensibility** - invariant to the hardware or functional extension of the computing infrastructure
  - **Data-delivery models** - a monitoring system provides a constant stream of data. HP does this every 15 minutes.
  - **Portability** - availability to aggregate monitoring data independent of environment or platforms
  - **Security** - access control and authentication

Extra requirements

- As a meta-monitoring framework, HappyFace is also required to have:
  - **Single access point**
  - **Up-to-date monitoring information**
  - **History functionality**
  - **Fast accessibility**
  - **Comfortable usage**
  - **Modular structure**

Basic workflow



HappyFace Framework

mod1.py cfg

mod2.py cfg

acquire.py

HappyFace.db

Apache Web Server

Cherrypy Web Server

mod1.html

mod2.html

render.py

cfg

HTML Output

Monitoring Sources

HappyFace Configuration

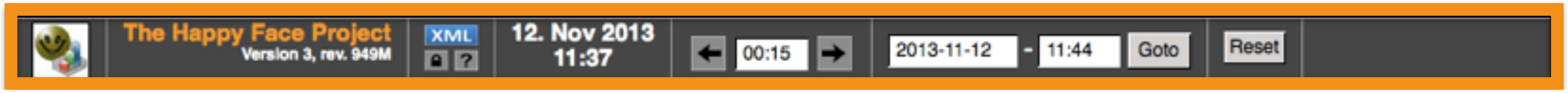
```
[happyface]
categories =
     category1
[database]
url = sqlite:///
     HappyFace.db
```

Category Configurations

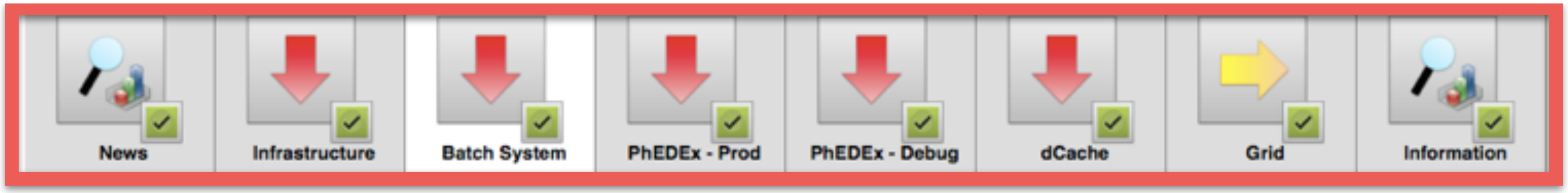
```
[category1]
name =
     cat1
modules =
     mod1, mod2
```

Happy Face Interface

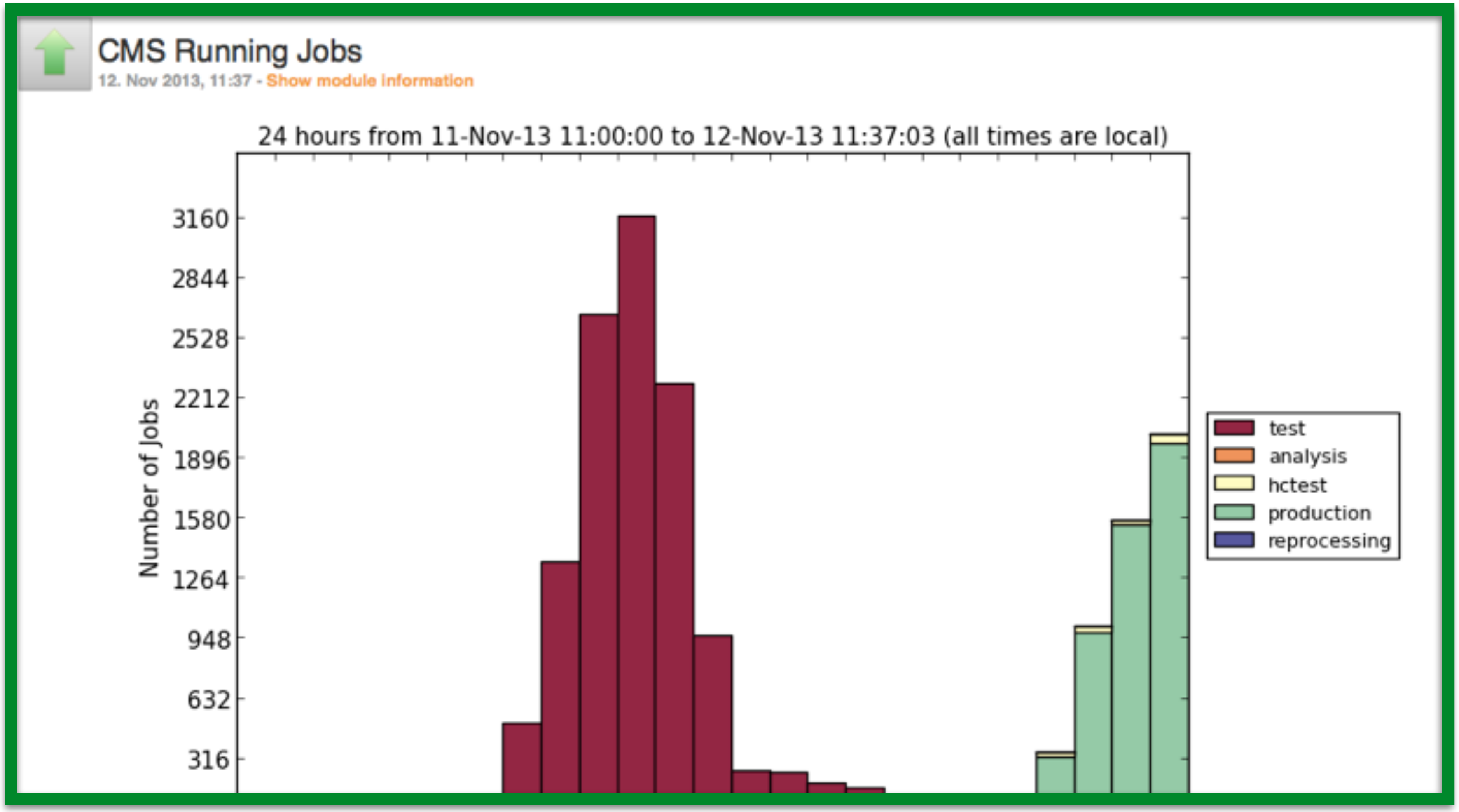
History navigation functionality



Category navigation bar



Fast navigation through different modules inside a category


CMS Running Jobs
12. Nov 2013, 11:37 - Show module information
24 hours from 11-Nov-13 11:00:00 to 12-Nov-13 11:37:03 (all times are local)
Number of Jobs
3160
2844
2528
2212
1896
1580
1264
948
632
316
test
analysis
hctest
production
reprocessing

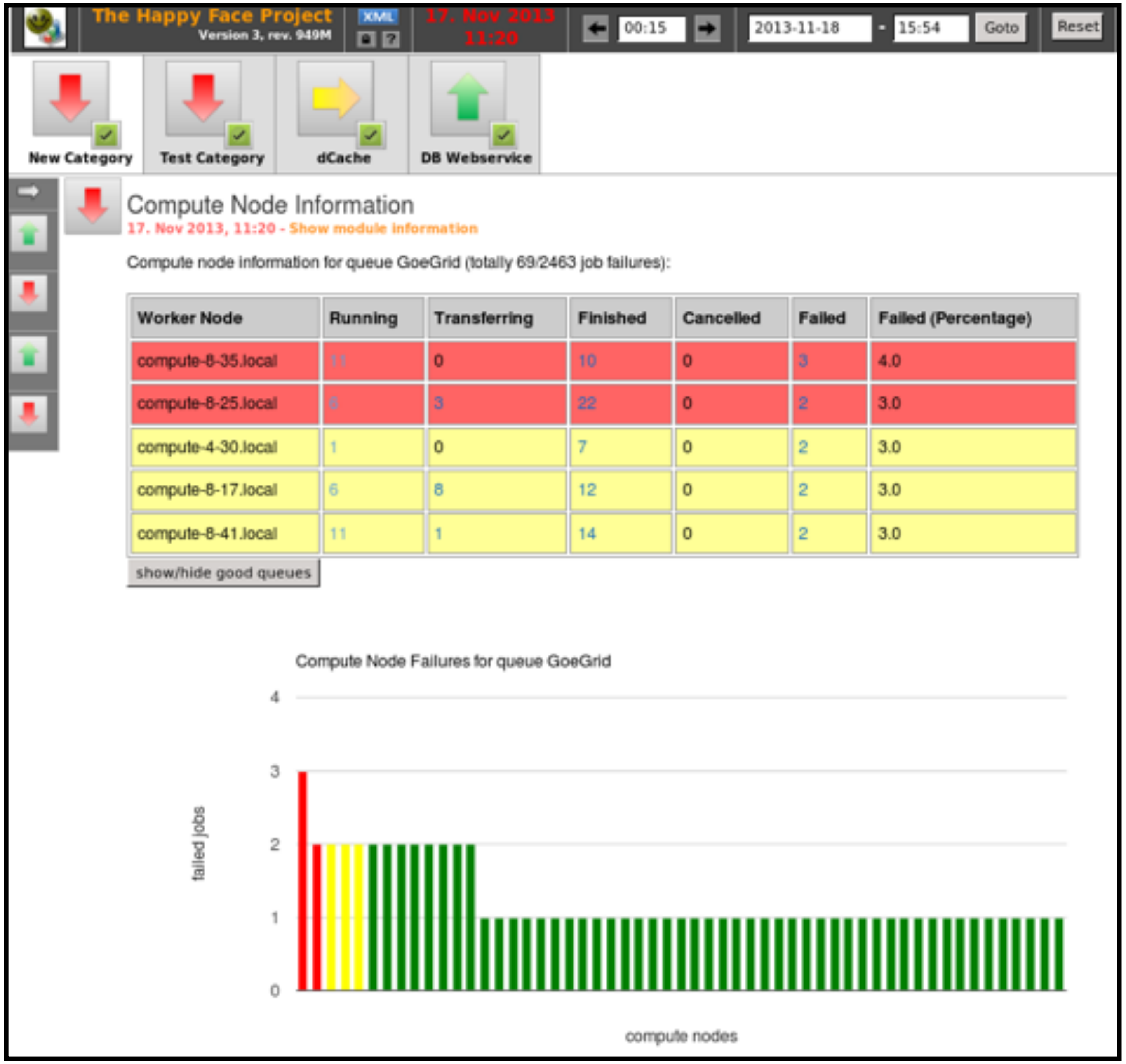
Individual module content

Some modules developed

- Apel Accounting
- GStat
- Panda
- HammerCloud Functional Tests
- Analysis Ganga Jobs
- Compute Node Information
- DDM Dashboard
- DDM Deletion
- Nagios
- SAM Tests
- Ganglia
- dCache Dataset Restore Monitor
- dCache Pool Information
- …

Modules

- Apel Accounting
- GStat
- Panda
- HammerCloud Functional Tests
- Analysis Ganga Jobs
- **Compute Node Information**
- DDM Dashboard
- DDM Deletion
- Nagios
- SAM Tests
- Ganglia
- dCache Dataset Restore Monitor
- dCache Pool Information
- …



Compute Node Information

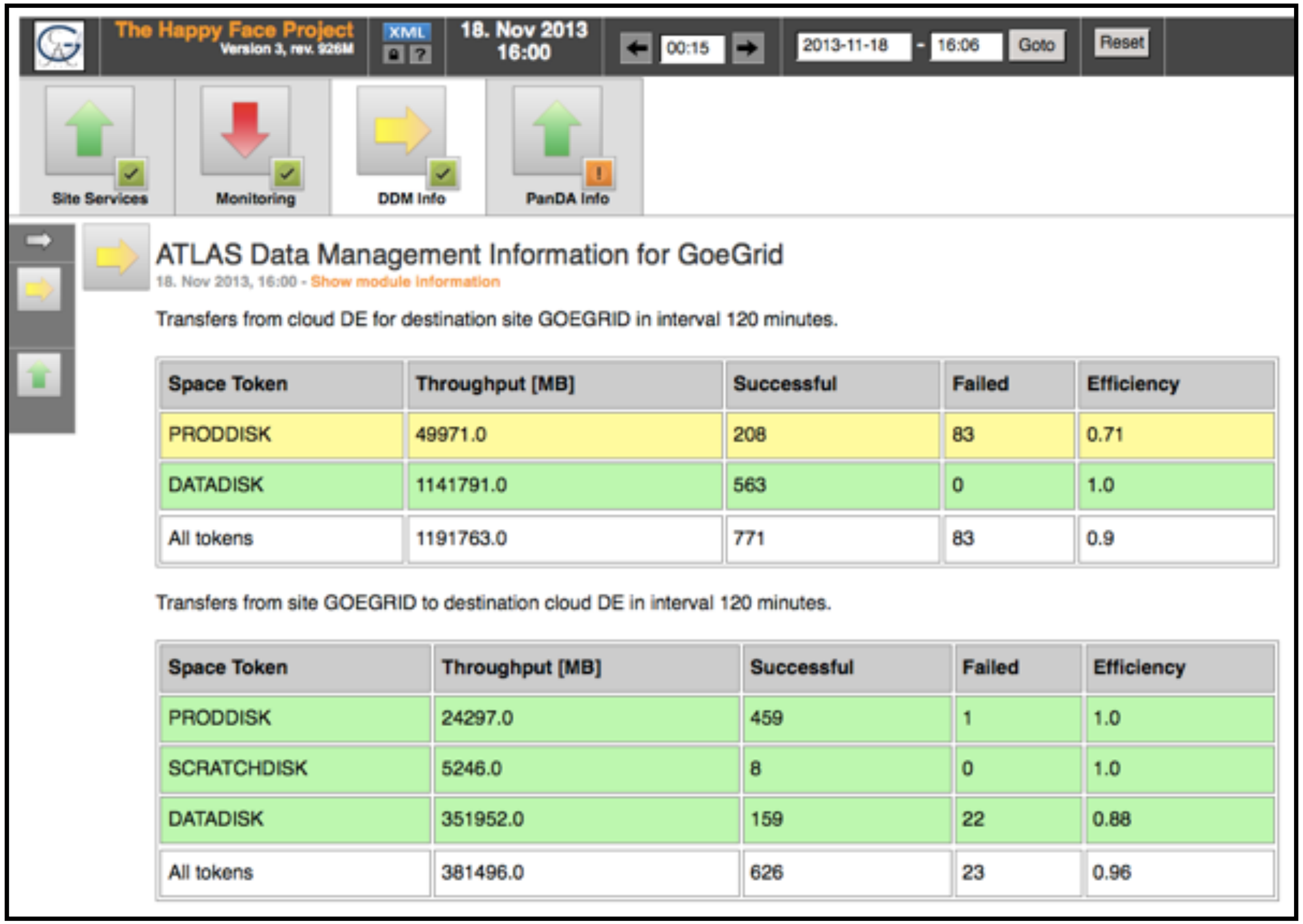
17. Nov 2013, 11:20 - Show module information

Compute node information for queue GoeGrid (totally 69/2463 job failures):

Worker Node	Running	Transferring	Finished	Cancelled	Failed	Failed (Percentage)
compute-8-35.local	11	0	10	0	3	4.0
compute-8-25.local	6	3	22	0	2	3.0
compute-4-30.local	1	0	7	0	2	3.0
compute-8-17.local	6	8	12	0	2	3.0
compute-8-41.local	11	1	14	0	2	3.0

Modules

- Apel Accounting
- GStat
- Panda
- HammerCloud Functional Tests
- Analysis Ganga Jobs
- Compute Node Information
- **DDM Dashboard**
- DDM Deletion
- Nagios
- SAM Tests
- Ganglia
- dCache Dataset Restore Monitor
- dCache Pool Information
- …



ATLAS Data Management Information for GoeGrid

18. Nov 2013, 16:00 - Show module information

Transfers from cloud DE for destination site GOEGRID in interval 120 minutes.

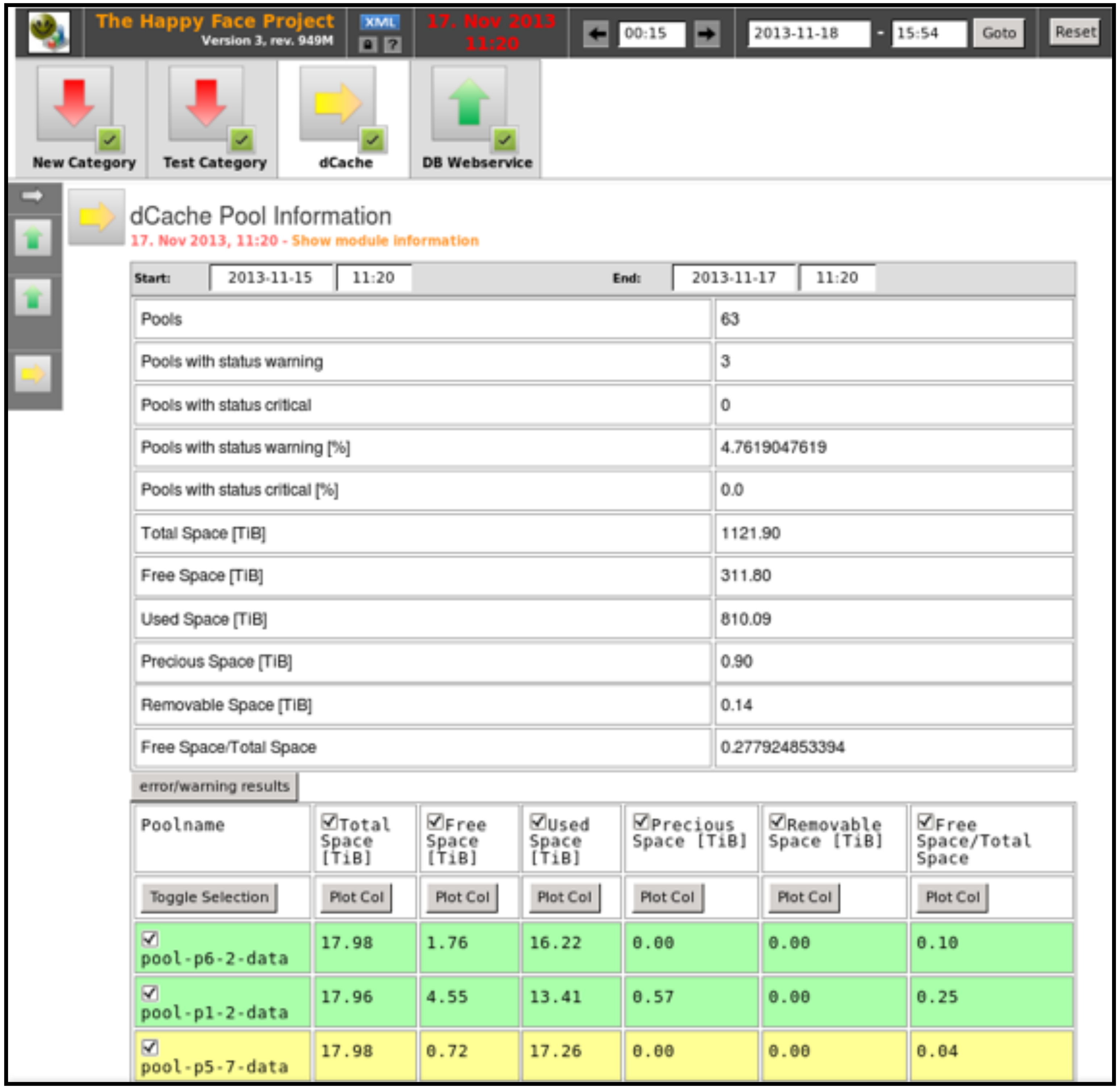
Space Token	Throughput [MB]	Successful	Failed	Efficiency
PRODDISK	49971.0	208	83	0.71
DATADISK	1141791.0	563	0	1.0
All tokens	1191763.0	771	83	0.9

Transfers from site GOEGRID to destination cloud DE in interval 120 minutes.

Space Token	Throughput [MB]	Successful	Failed	Efficiency
PRODDISK	24297.0	459	1	1.0
SCRATCHDISK	5246.0	8	0	1.0
DATADISK	351952.0	159	22	0.88
All tokens	381496.0	626	23	0.96

Modules

- Apel Accounting
- GStat
- Panda
- HammerCloud Functional Tests
- Analysis Ganga Jobs
- Compute Node Information
- DDM Dashboard
- DDM Deletion
- Nagios
- SAM Tests
- Ganglia
- dCache Dataset Restore Monitor
- **dCache Pool Information**
- …



dCache Pool Information

17. Nov 2013, 11:20 - Show module information

Start: 2013-11-15 11:20 End: 2013-11-17 11:20

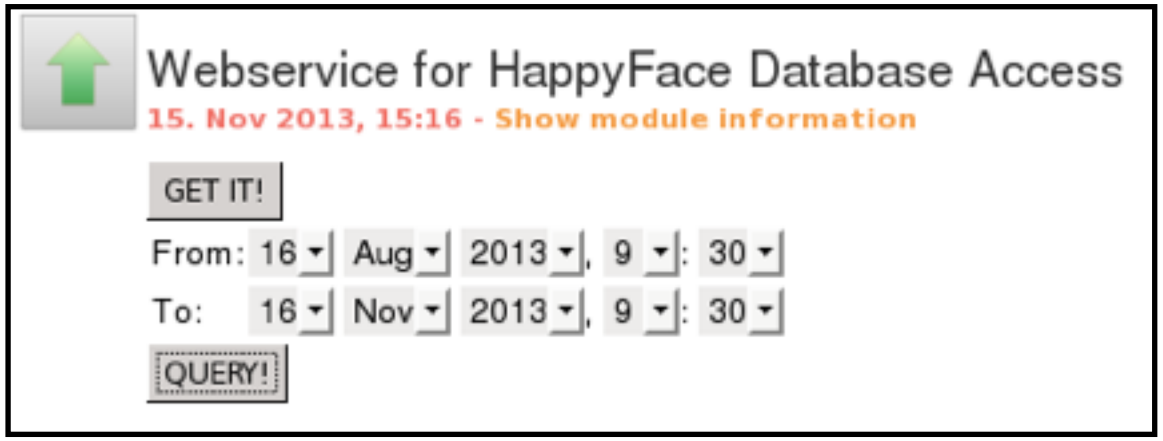
Pools	63
Pools with status warning	3
Pools with status critical	0
Pools with status warning [%]	4.7619047619
Pools with status critical [%]	0.0
Total Space [TiB]	1121.90
Free Space [TiB]	311.80
Used Space [TiB]	810.09
Precious Space [TiB]	0.90
Removable Space [TiB]	0.14
Free Space/Total Space	0.277924853394

error/warning results

Poolname	☑Total Space [TiB]	☑Free Space [TiB]	☑Used Space [TiB]	☑Precious Space [TiB]	☑Removable Space [TiB]	☑Free Space/Total Space
Toggle Selection	Plot Col	Plot Col	Plot Col	Plot Col	Plot Col	Plot Col
☑ pool-p6-2-data	17.98	1.76	16.22	0.00	0.00	0.10
☑ pool-p1-2-data	17.96	4.55	13.41	0.57	0.00	0.25
☑ pool-p5-7-data	17.98	0.72	17.26	0.00	0.00	0.04

Web Services

- Providing access to the monitoring data aggregated in the HappyFace database:
- HappyFace module for direct database access
- REST-ful web service for easy, non-standardised access (JSON output)
- W3C-compliant WSDL/SOAP-based web service for database access
  - WSDL generator
  - WSDL file
  - Python client and server stubs
  - Python client and server implementation



Webservice for HappyFace Database Access

15. Nov 2013, 15:16 - Show module information

From: 16 Aug 2013, 9 : 30

To: 16 Nov 2013, 9 : 30

Conclusions

- HappyFace monitoring tool covers the full spectrum of hardware, software, and services gathering information from remote and local site information

- Its design makes HappyFace a flexible, easy configurable and a reliable tool to monitor any computing site

- Ongoing tasks:
    - New modules development; PBS and CreamCE
    - OGSA web-service implementation from internal information resources
    - RPM & YUM repository

Final remarks HappyFace project

- We have a powerful meta-monitoring framework able to aggregate, process, and store monitoring data from difference sources

- Also, it supplies a web service which it is able to provide the desired data in different format and time frames.

Final remarks HappyFace project

- We have a powerful meta-monitoring framework able to aggregate, process, and store monitoring data from difference sources

- Also, it supplies a web service which it is able to provide the desired data in different format and time frames.

Let’s analyse all this data!

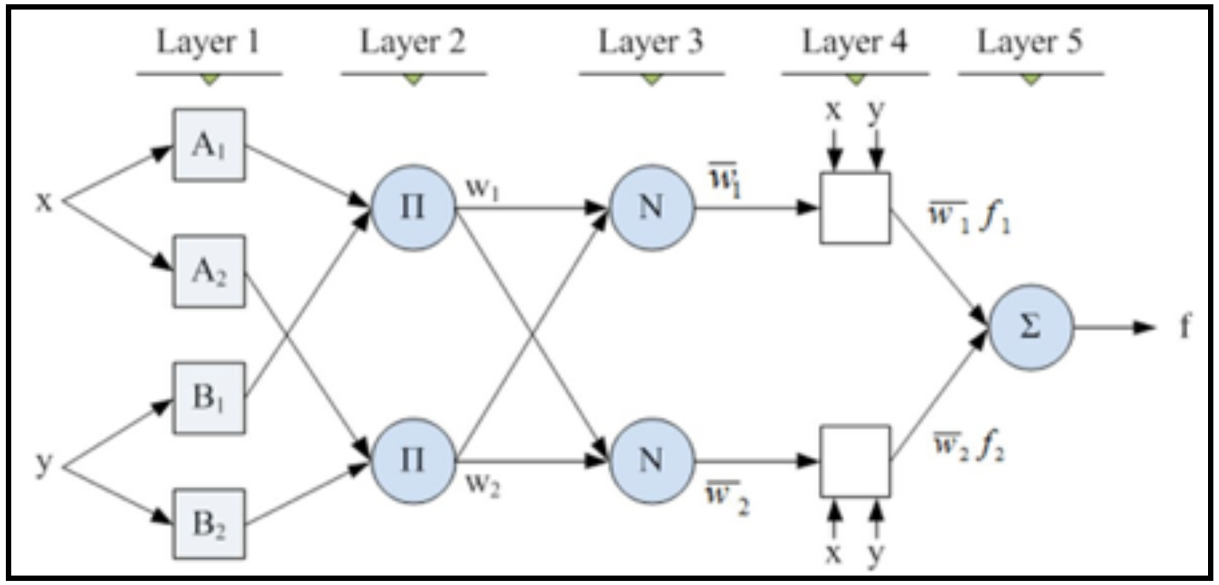
Smart Monitoring System

- Analyses the site's monitoring data looking for the failure patterns
- Performs a failure root cause analysis
- According to the detected failure patterns the system is able to provide short-term failure predictions
- The working framework should be able to handle the linguistic terms from monitoring data (Ok, Warning, Failed) and the ability to learn from a training data

ANFIS: Adaptive Neuro-Fuzzy Inference Systems

- ANFIS is a Feed-Forward Neural Network with the activations functions from Fuzzy Inference System (Takagi-Sugeno)

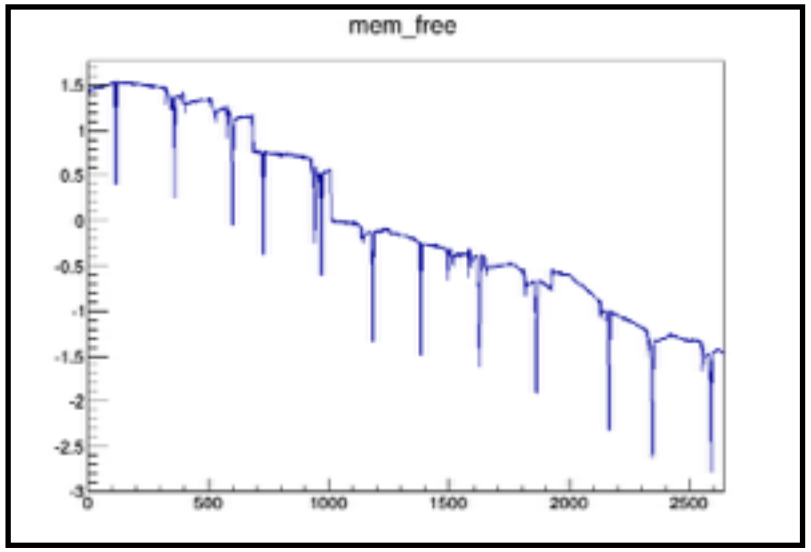
- Since ANFIS integrates both neural networks and fuzzy logic principles, it has potential to capture the benefits of both in a single framework

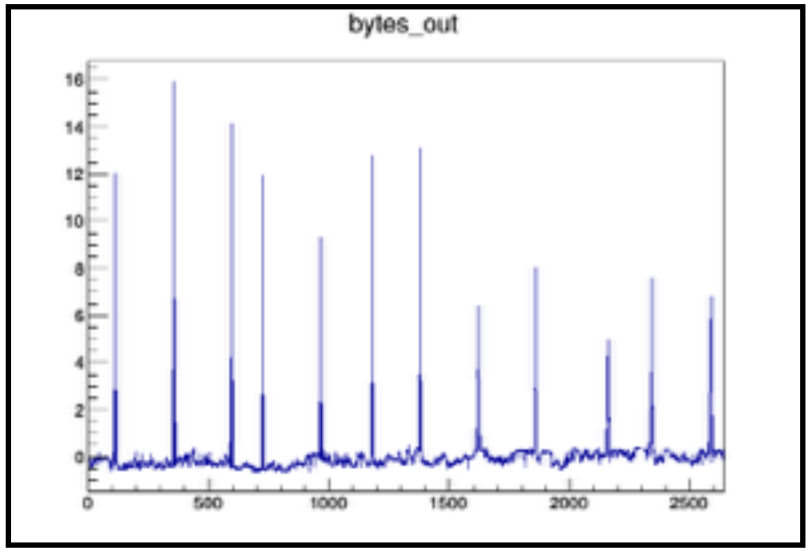


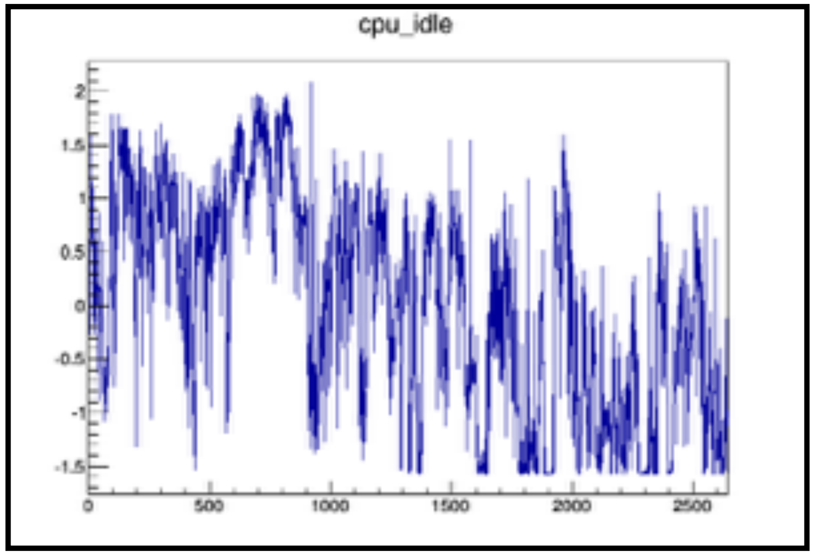
- The inference system is based on IF-THEN type rules, which can be adapted due to the neural network learning capability and hence approximate any non-linear function

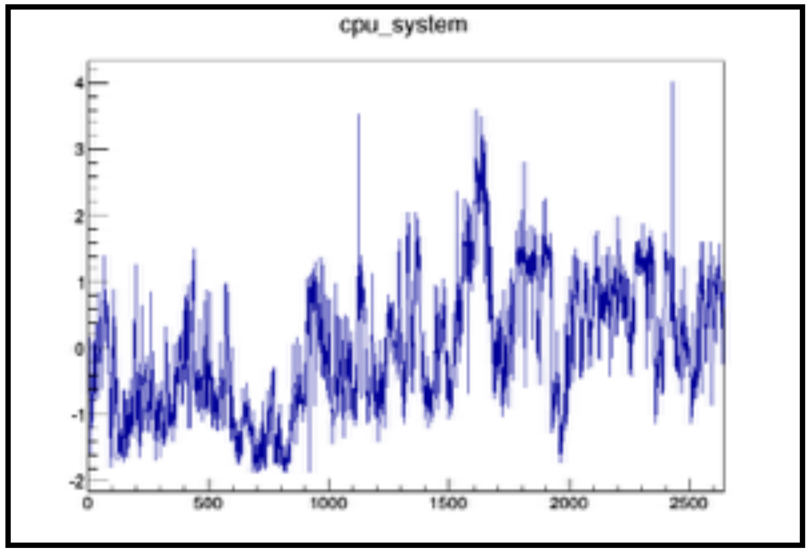
Smart Monitoring System

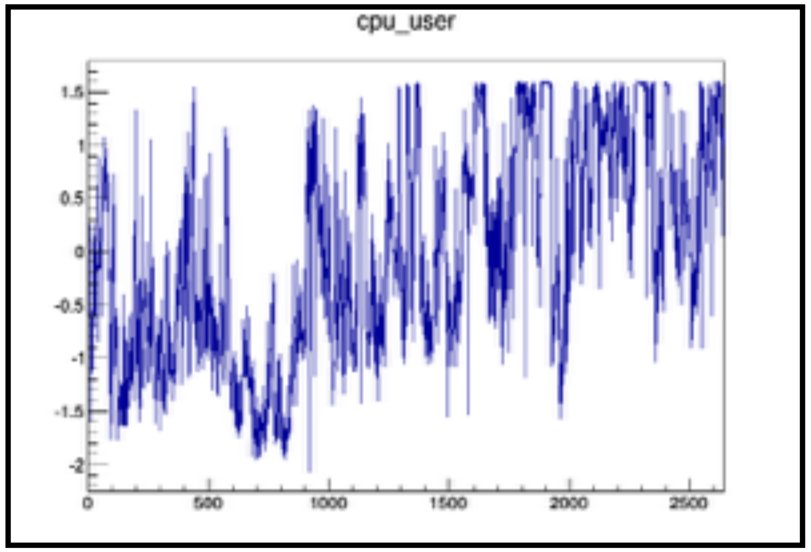
- Each of the sub-services guarantees availability of the Grid services.
- Each sub-service consumes certain amount of computing resources like:
  - Memory - *mem - buffers, cached, free, shared, etc*
  - CPU - *idle, wio, nice, user, running, total, etc*
  - Network - *pkts in, pkts out, tcp established, tcp listen tcp timewait, etc*

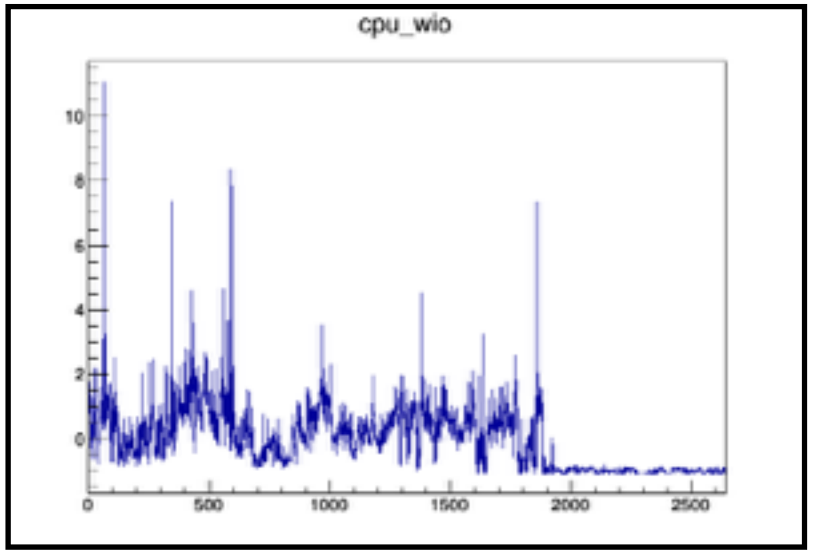






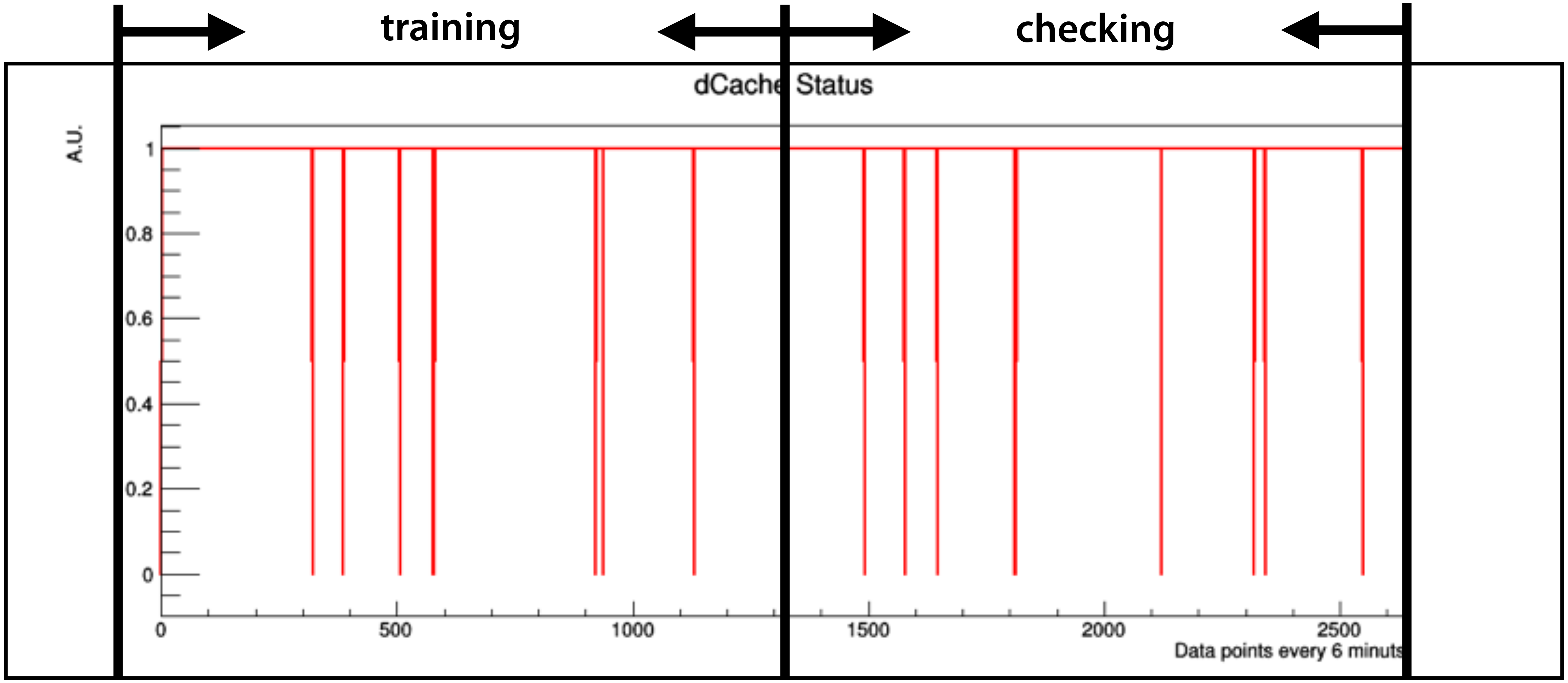






Case Study - dCache Storage

- dCache storage system relies on the Chimera server to host metadata information for all stored files.
- All information about the Chimera can be traced down from Chimera log files.
- Chimera monitoring took 23 days (every 6 minutes) in which 8 registered failures were observed.



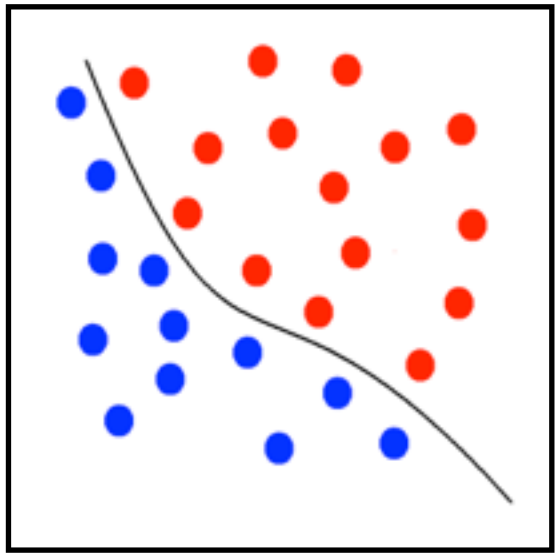
Case Study - dCache Storage

- The amount of data was not enough for training the neural network
- Two steps preprocessing is needed:

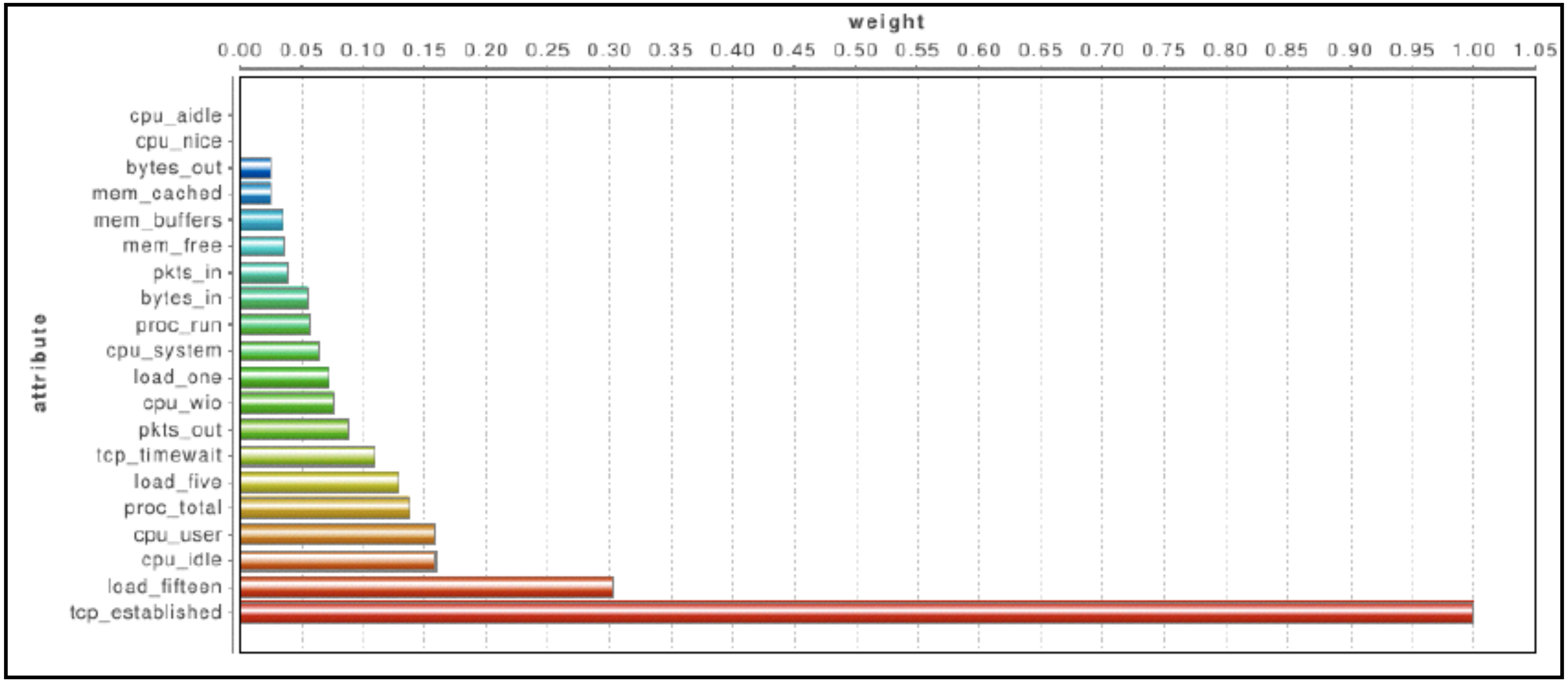
  - Z-normalisation:

$$x_i^{'} = \frac{x_i - \mu}{\sigma}, \; i \in \mathbb{N}$$

  - Support Vector Machine (SVM) is used to extract the most important features

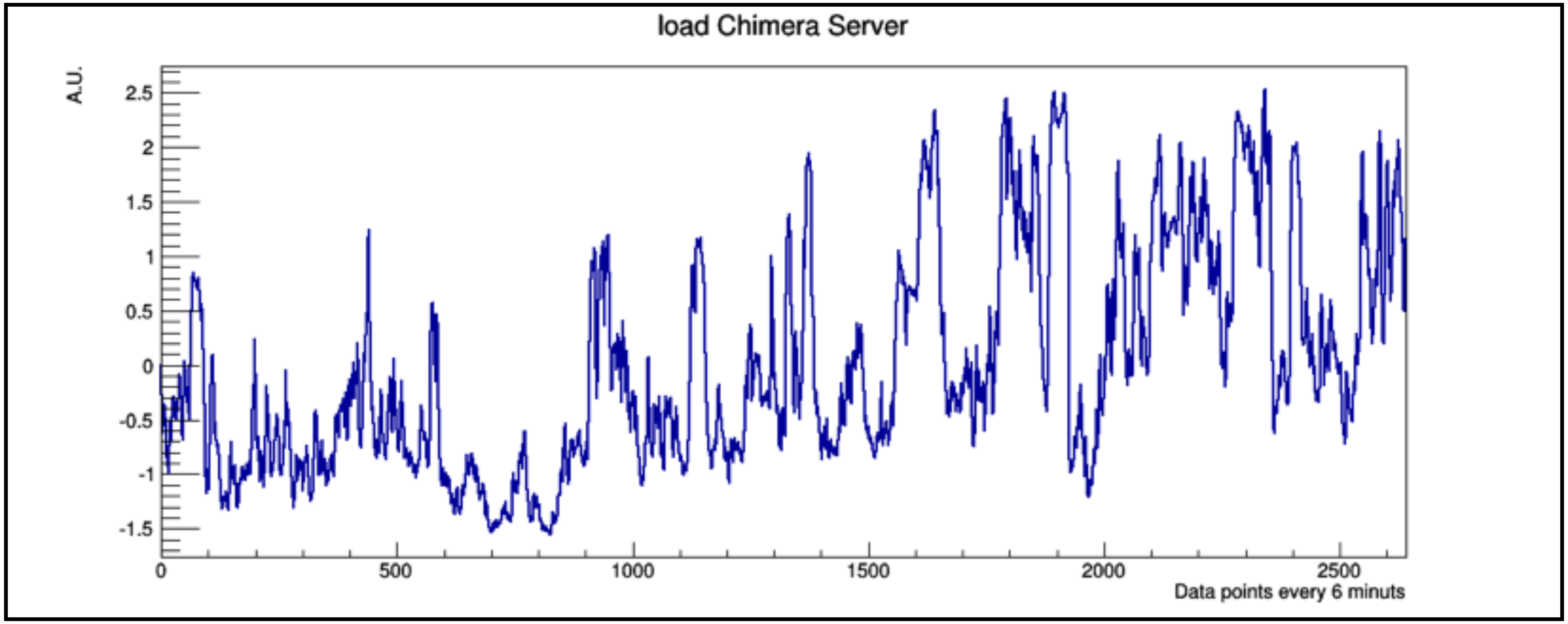


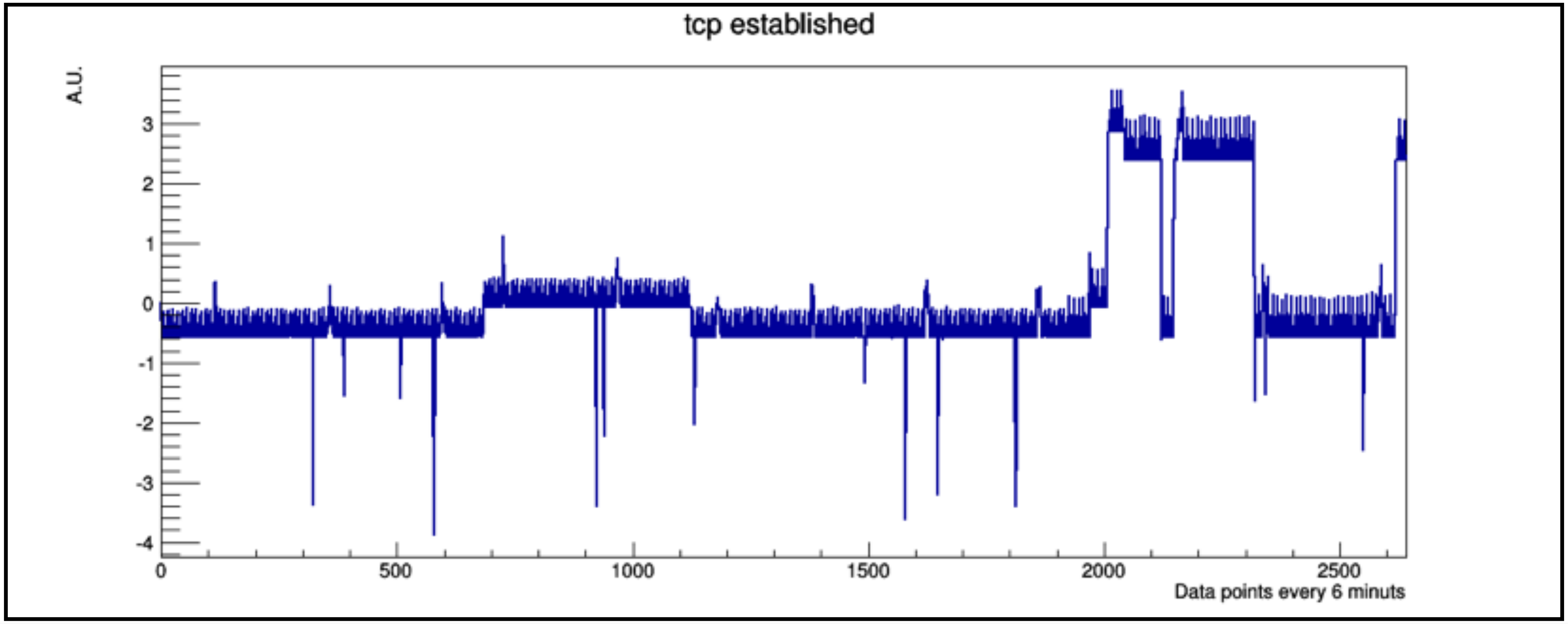
Case Study - SVM Output



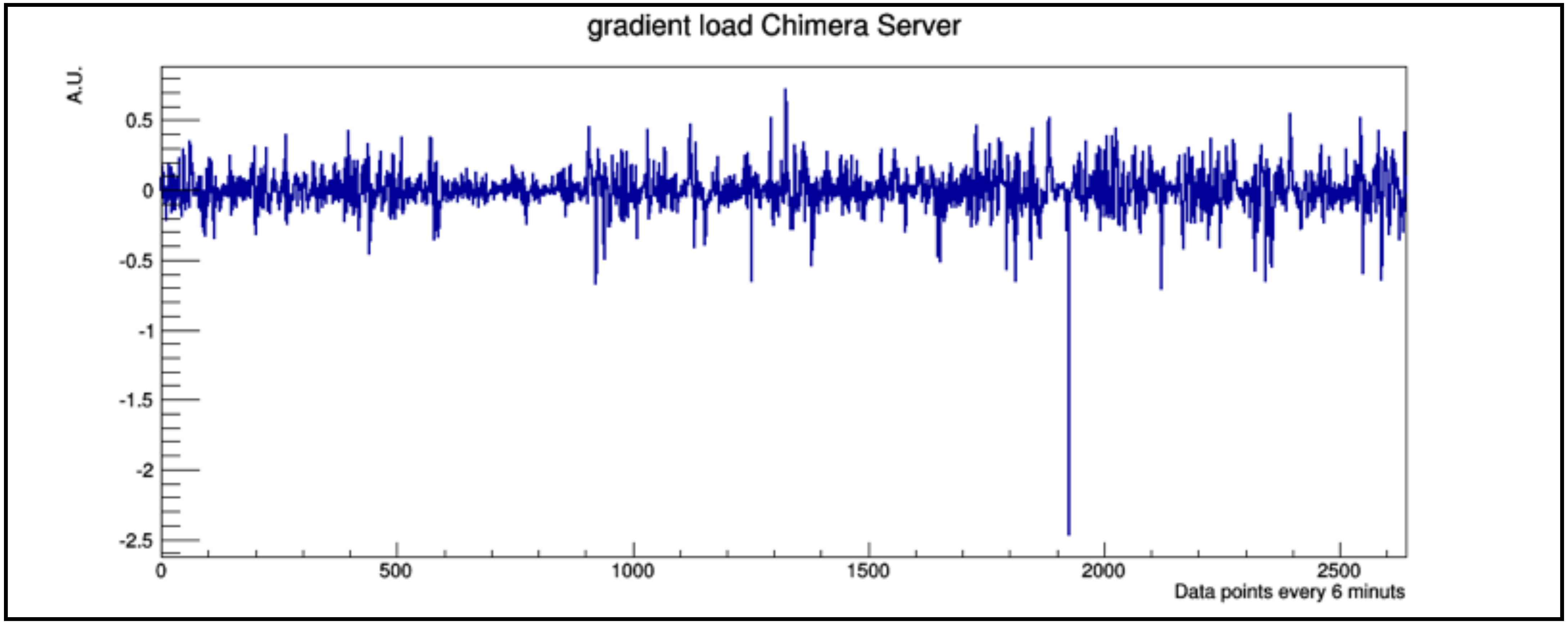
- Most important attributes are **load_fifteen** (load Chimera Server) and **tcp_established** (tcp connections)

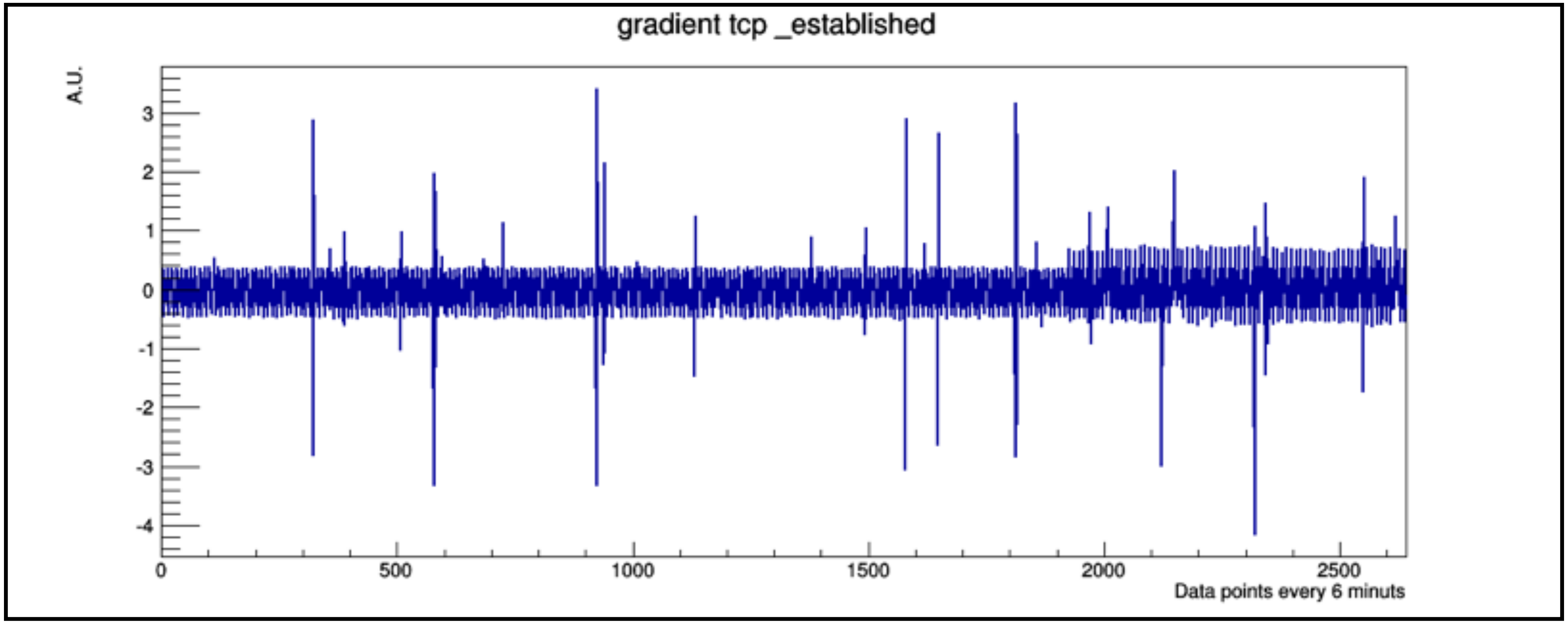
Main attributes





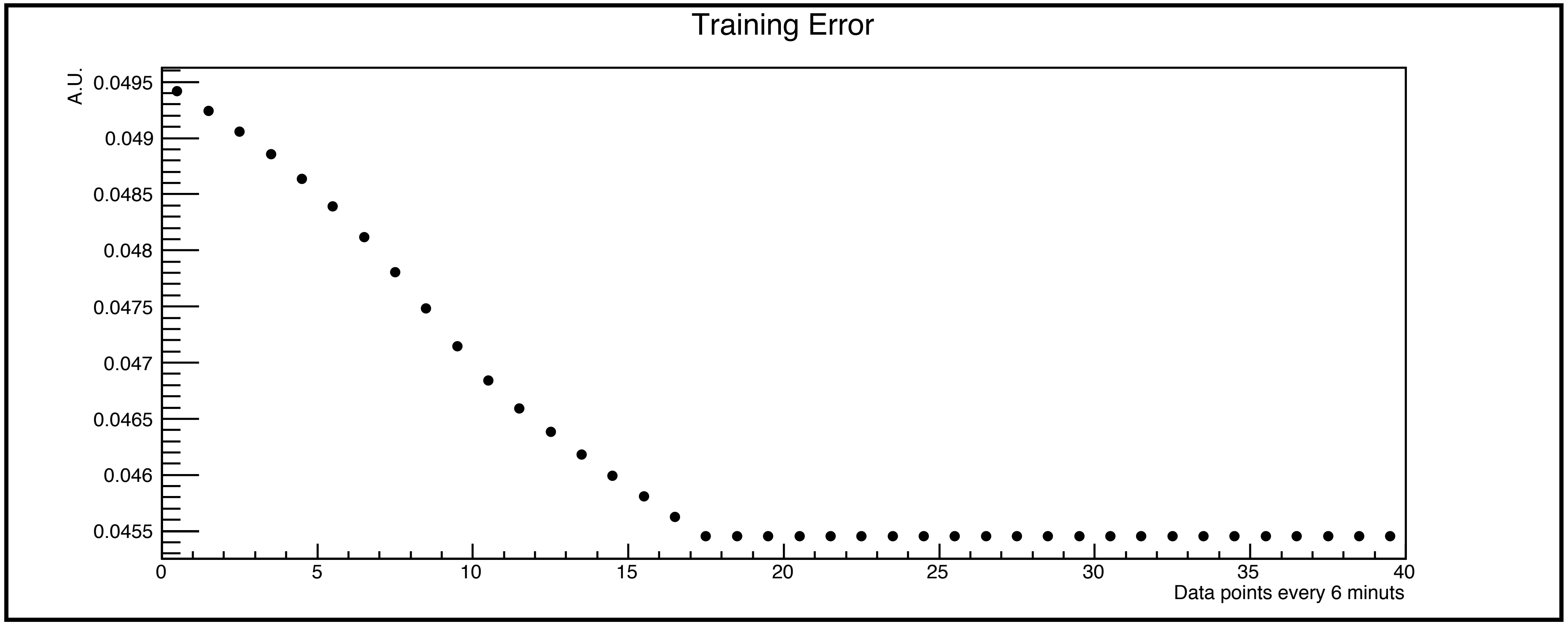
Main gradient attributes





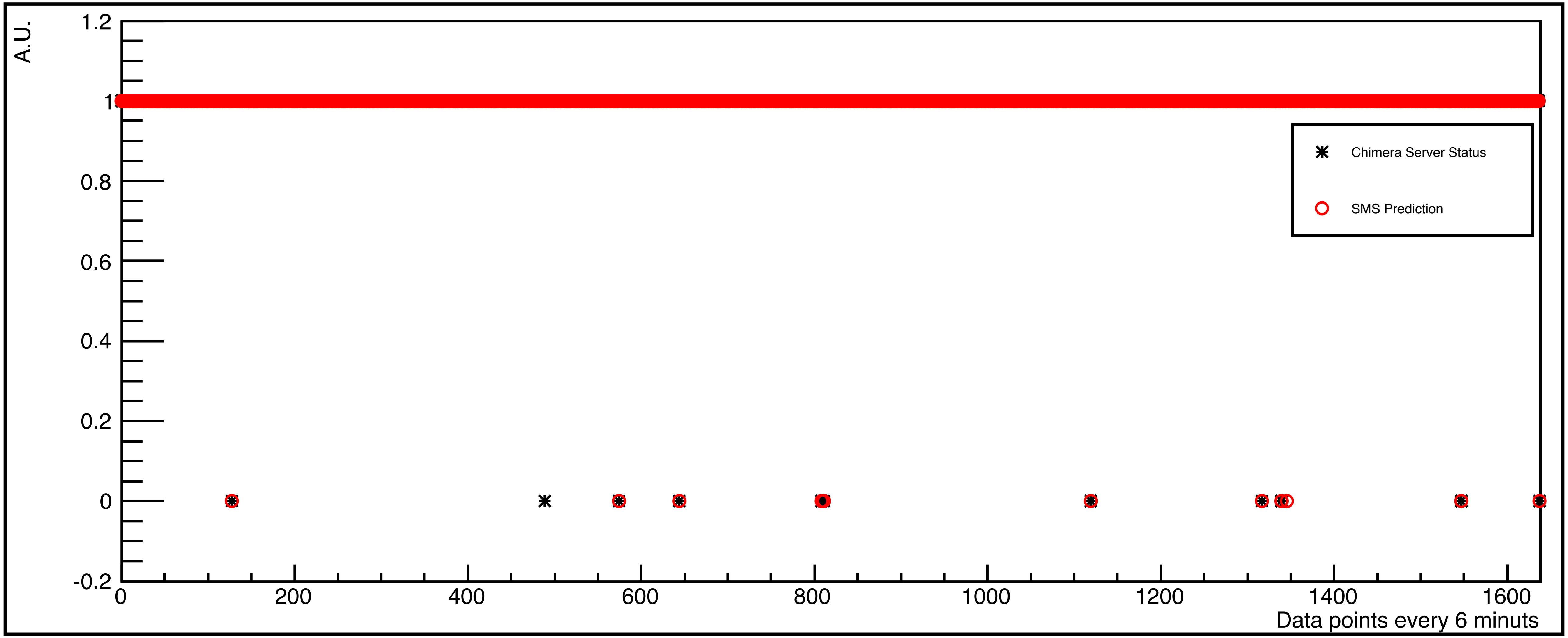
Training dataset and error

- For the ANFIS training 40 epochs were defined
  - The minimum training error has been reached at the 20th epoch
- MATLAB is taking care not to over-training the NN



Results

- A dCache failure is consider for all predicted values below 0.8
- Good agreement with dCache status is achieve by the system!



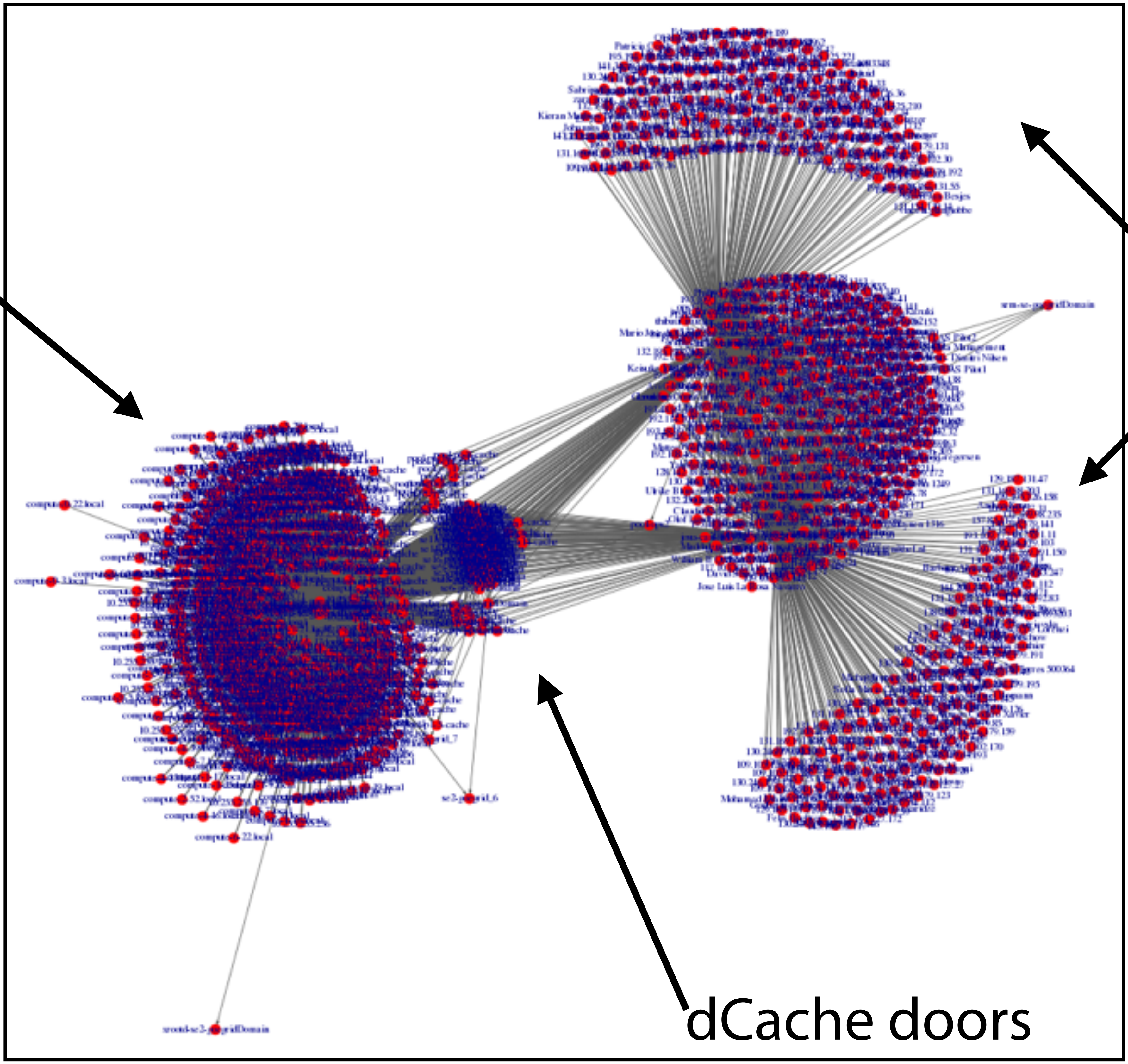
Conclusions

- Results are promising and give additional motivation to extend this project
- Data is continuously collected for further analysis and not only for dCache
- The final goal would be to provide a standalone package able to process monitoring information and provide a forecast and analysis of the failures with a certain credibility.

Visualisation of Big Data

- The project is in an initial state fase and still some aspects are up on the air
- The plan of action:
- Due to the huge volume of the dCache log (billing) we want to get a sample of this data by using bootstrap re-sampling technique
- When the sample is validated and really represents the original dataset some interesting features can be extracted by:
  - Real-time plots
  - Statistical analysis
  - Visualisation of graph paths

Visualisation of Big Data



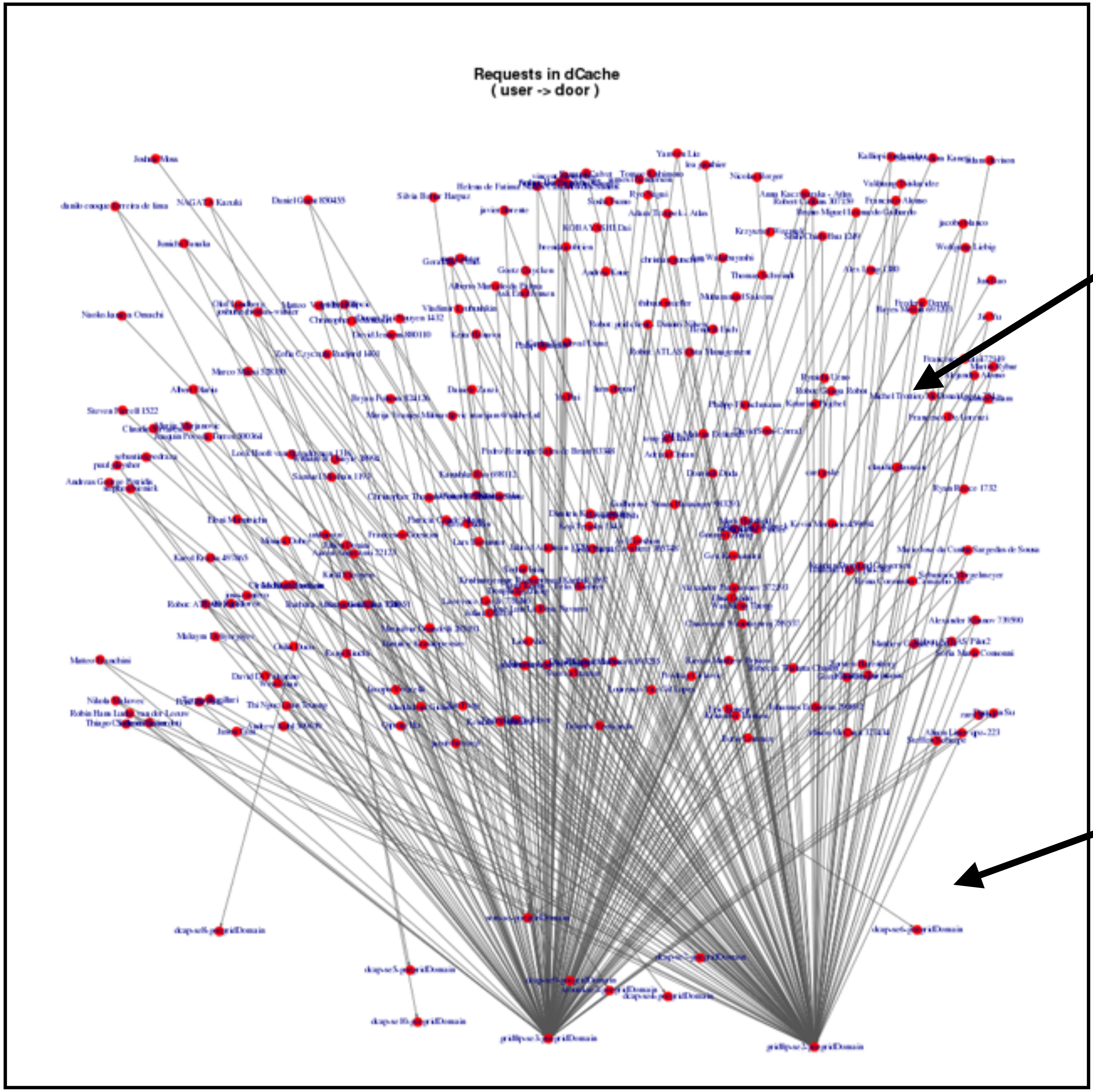
Compute Nodes

Users

dCache doors

Visualisation of Big Data

Visualisation of Big Data



Users

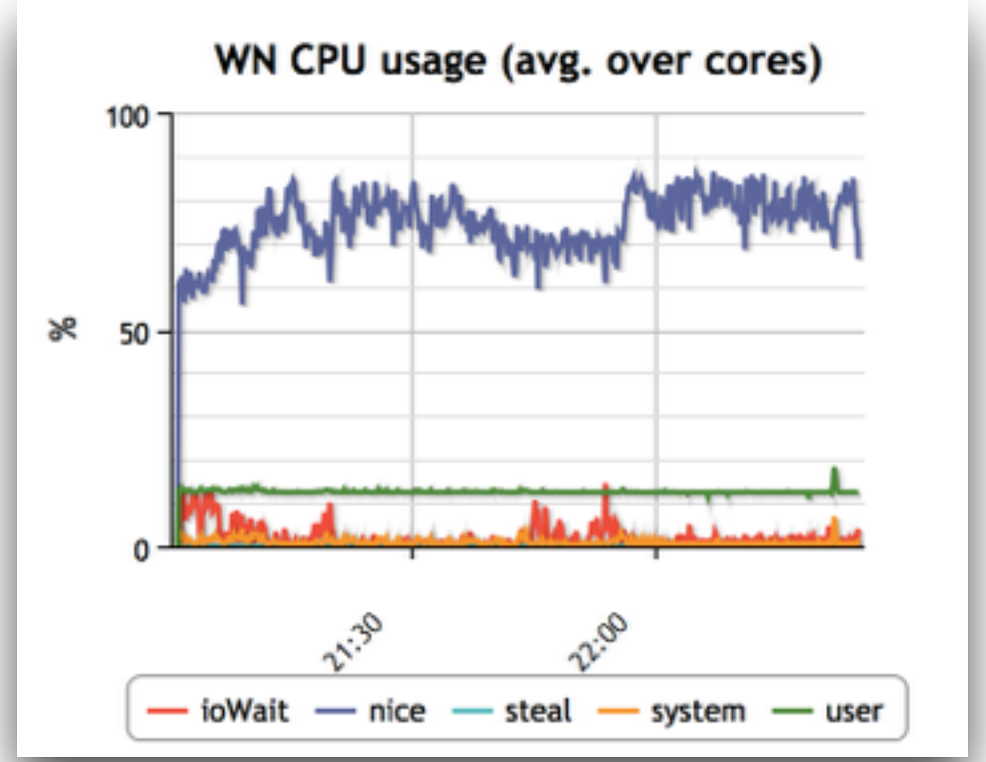
dCache doors

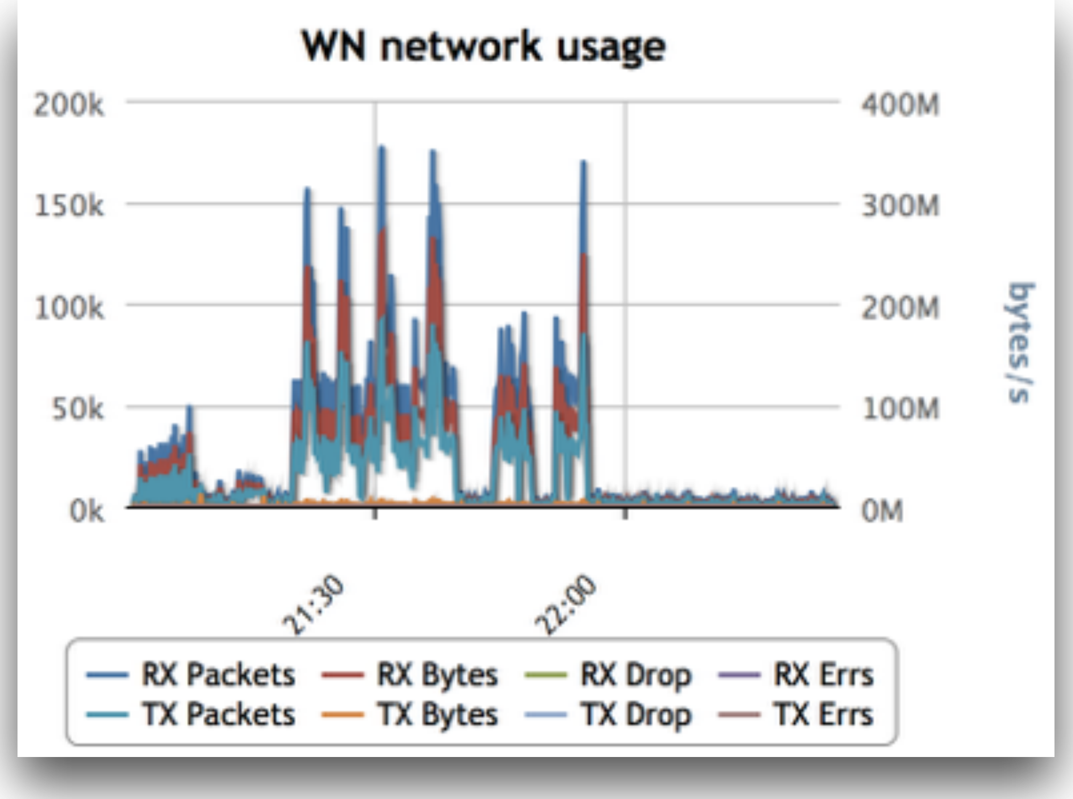
Outlook

- Monitoring for cluster infrastructure:
  - **HappyFace**
  - **SmartMonitoringSystem**
  - **Visualisation of Big Data**
- Monitoring for the user analysis:
  - **Job Execution Monitoring**

Job Execution Monitoring

- The **JEM** (Job Execution Monitor) is a customisable job-centric monitoring system running in user space.
  - A **system monitor** runs in parallel to the user job measuring parameters like cpu load, network traffic, free RAM, free disk space on several filesystems, etc.





Job Execution Monitoring

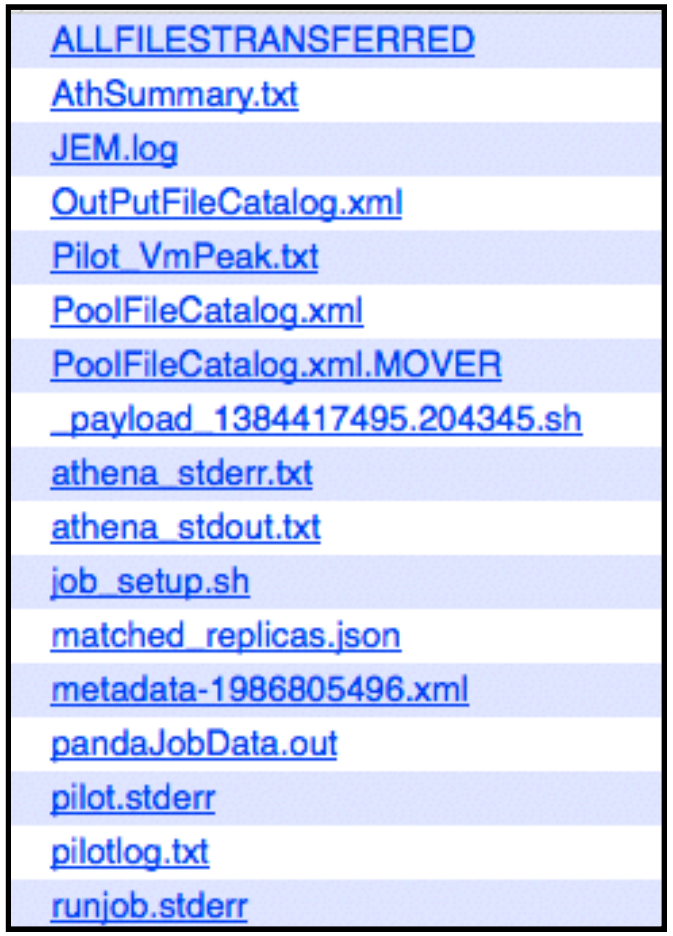
- The **script monitor** analyses the user's job script giving feedback to the user about its current status. In case of failures a variety of debug information is provided.

- The **file watcher**, which monitors files for changes and provide the contents in real time to the user.

- The **process watcher** monitors the child process tree of the user job, looking for starting and exiting processes specified by the user.

- The **remote debugging** facility to deeply monitor execution progress inside user libraries helping to spot user algorithm crashes and memory leaks.

- All extra information is embedded in the job log files.



```
ALLFILESTRANSFERRED
AthSummary.txt
JEM.log
OutPutFileCatalog.xml
Pilot_VmPeak.txt
PoolFileCatalog.xml
PoolFileCatalog.xml.MOVER
_payload_1384417495.204345.sh
athena_stderr.txt
athena_stdout.txt
job_setup.sh
matched_replicas.json
metadata-1986805496.xml
pandaJobData.out
pilot.stderr
pilotlog.txt
runjob.stderr
```

Job Execution Monitoring

```
--enableJEM --configJEM '+debug;+ver=dev;
+live;+watch=athena_stdout.txt;+livewatch'
```

- Useful links:
  - https://twiki.cern.ch/twiki/bin/viewauth/AtlasComputing/JobExecutionMonitor
  - http://jem.physik.uni-wuppertal.de/JEM/
  - http://jem.physik.uni-wuppertal.de/JEM/jobid/PanDA.1637324001
- Contact Information:
  - volkmer@physik.uni-wuppertal.de
  - jem@uni-wuppertal.de

Conclusions

- JEM project is mainly focus on MC validation task
- JEM is a powerful tool if you want to debug/monitor/check you analysis jobs on the grid.
- It is fully integrated in the ATLAS software, easy to use and provide results as soon as the job ends.

The End