Towards Predictive Maintenance with Machine Learning at the INFN-CNAF computing centre

L. Giommi[*]
University of Bologna, Italy
E-mail: luca.giommi3@unibo.it

D. Bonacorsi, T. Diotalevi, S. Rossi Tisbeni, L. Rinaldi

University of Bologna, Italy

L. Morganti, A. Falabella, E. Ronchieri, A. Ceccanti, B. Martelli

INFN-CNAF, Italy

The INFN-CNAF computing center, one of the Worldwide LHC Computing Grid Tier-1 sites, is serving a large set of scientific communities, in High Energy Physics and beyond. In order to increase efficiency and to remain competitive in the long run, CNAF is launching various activities aiming at implementing a global predictive maintenance solution for the site.
This requires a site-wide effort in collecting, cleaning and structuring all possibly useful data coming from log files of the various Tier-1 services and systems, as a necessary step prior to designing machine learning based approaches for predictive maintenance.
Among the Tier-1 services, efficient storage systems are one of the key ingredients of Tier-1 operations. CNAF uses the StoRM service as a Grid Storage Resource Manager solution: its operations are logged in a very complex manner, as the log content is deeply unstructured and hard to be exploited for analytics purposes. Despite such difficulty, the StoRM logs are a precious source of information for operators (e. g. real-time monitoring and anomaly detection), for developers (e. g. debugging, service stability, code improvements) and for site managers (service optimization, storage usage efficiency, time and money saving ways to spot and prevent unwanted behaviors).
Based on previous experiences on Big Data Analytics and Machine/Deep learning in the CMS experiment, this work describes how the StoRM logs can be handled and parsed to extract the relevant information, how such log handling can be designed to work automatically, how to define and implement metrics to tag critical states of the service, how to correlate StoRM events with external services events, and ultimately how to contribute to the future CNAF-wide predictive maintenance system.

Initial results in this activity are presented and discussed. Furthermore, a mention to ongoing complementary work at the CNAF center is also mentioned.

International Symposium on Grids & Clouds 2019, ISGC2019
31st March - 5th April, 2019
Academia Sinica, Taipei, Taiwan

[*]Speaker.

1. Introduction

The INFN-CNAF [1] computing center, one of the Worldwide LHC Computing Grid Tier-1 sites, uses StoRM (STOrage Resource Manager) [2] as a Grid SRM solution. In order to increase efficiency and to remain competitive in the long run, CNAF is launching various activities aiming at implementing a global predictive maintenance solution for the site. Because efficient storage systems are one of the key ingredients of Tier-1 operations, CNAF decided that a starting point of such an ambitious project could be the investigation of the StoRM service.

The state and the operations records of this service (and of the services associated to it) are stored in several log files and in a complex manner, as the log content is deeply unstructured and hard to be exploited for analytics purposes. The first step of this work is to handle and parse the log files to extract relevant pieces of information and design it to work automatically. In order to predict anomalies, it is crucial to define both a problematic period (with some anomalies in the system) and a normal one (to be used as benchmark). In this way it is possible to compare the behavior of the service in both periods and define metrics to tag critical states. This operation can be done with the help of Machine Learning (ML) techniques. A predictive model is built through specific training data, therefore this model would be able to primarily detect problems which are similar to the specific one. Despite this, a general behavior of the system in critical situations can be defined starting from this specific problem and generalized to a broader scenario. In order to verify this statement and to create an effective predictive system (that can be exploited in the perspective of predictive maintenance), it is necessary to define other different critical states detected in the past, to collect all the useful data logs and to create for each one of them a model that will be able to detect similar problems in the future.

In particular, this work describes the steps followed in the study of the log files coming from different sources referred to a particular period of time (specifically from December 1st to December 8th, 2018) and how such information could be correlated and used to create ML models for anomalies prediction.

2. Storage Resource Managers

Storage Resource Managers (SRMs) are middleware services whose function is to provide dynamic space allocation and file management of shared storage resources geographically distributed [3]. SRMs can have files stored in several physical locations and can also bring them from tape to disk for direct access by a client [4]. Once the file is available for I/O, a TURL (transfer URL) is returned for a temporary access to the file controlled by the pinning lifetime. A similar capability exists when a client wishes to put a new file into SRM. SRMs do not perform file transfers, but can invoke middleware components to do them, as GridFTP.

The SRM interface provides asynchronous and synchronous methods. Asynchronous methods return a token associated to a request, and the corresponding action is performed by the SRM service asynchronously (such as the srmPrepareToPut method for preparing the SRM to accept new data and the srmPrepareToGet method to get read access to the desired data resource). The client can retrieve the status of the request at any time by addressing it through the specific token. This is the case for data access functionalities. Instead, synchronous requests return at completion giving back the control to the client (e.g. blocking calls). This is the case of directory and file management (e.g. srmLs, srmMkdir and srmRmdir) and space management functions (e.g. srmReserveSpace).

3. The StoRM service

Since 2003, CNAF has been hosting the Italian Tier-1 data center for the High-Energy Physics (HEP) experiments at the Large Hadron Collider in Geneva (and more, non-HEP experiments), providing the resources, support and services needed for data storage, data distribution, data processing, data analysis and Monte Carlo production. CNAF has been involved in the development of a large set of software products. In the area of middleware for distributed systems CNAF supports, maintains and further evolves widely-used products, such as StoRM [5].

StoRM is a storage manager service for generic disk-based storage systems, compliant with the standard SRM interface version 2.2. StoRM is the SRM solution adopted by the Tier-1 hosted at CNAF (and by other Tier-2s), and it has been developed with the specific aim of providing high performing parallel file systems like GPFS and Lustre, but also other standard POSIX file systems, through a SRM interface (currently CNAF uses GPFS as parallel file system). In a cluster configuration, the file system allows large numbers of disk attached to multiple storage servers to be configured as a single file system. Since it generalizes the file system access to the Grid, StoRM also takes advantage from the file system security mechanism to ensure an authenticated and authorized access to the data.

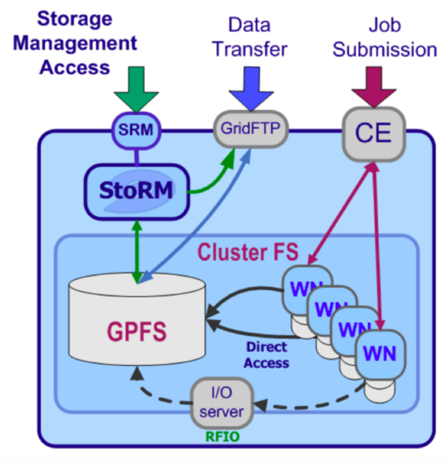
A grid site that provides an efficient, secure and reliable way to access data through standard POSIX calls [3, 6] is shown in Figure 1a. A job running on a worker node belonging to the GPFS NSD (Network Shared Disk) has direct access to the desired file into the Storage Element. In this scenario StoRM assures that the user data will be available for all the time the access operation goes on.

StoRM has a multilayer architecture made by two stateless components, called *Frontend* and *Backend*, and one database [7, 8]. The Frontend exposes the SRM web service interface, manages user authentication, stores SRM requests data into a database, retrieves the status of ongoing requests, and interacts with the Backend. The Backend is the core of StoRM service since it executes all synchronous and asynchronous SRM functionalities. It processes the SRM requests managing files and space, it enforces authorization permissions and it can interact with other Grid services. Moreover the Backend is able to use advanced functionalities provided by the file system to accomplish space reservation requests. A simple StoRM Service Architecture schema is shown in Figure 1b.

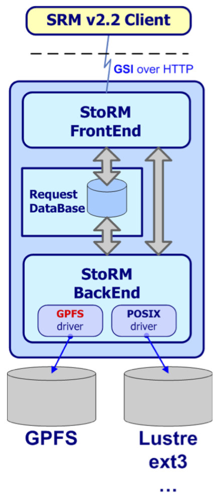
3.1 Logging

The logging activity represents an important functionality of both StoRM components, and of the services linked to them. They have multiple ways of logging, or rather there are different kind of files in which specific information is stored [9].

The information related to the services can be obtained by the log files produced by the services themselves or through the ELK (Elasticsearch, Logstash and Kibana) stack suite [10], thanks to a complementary infrastructure work presented in this conference [11].



(a)



(b)

Figure 1: Pictorial view of storage management with GPFS and StoRM is shown in 1a (where WN stands for Worker Node and CE for Computing Element). A simple StoRM service architecture schema, with one Backend and one Frontend, is shown in 1b.

The StoRM service at CNAF is used by HEP experiments, including e.g. ATLAS. Each experiment has a different implementation of the StoRM service and in some cases the structure and rules of logging are different. In the following, the case of the ATLAS experiment is chosen for a first exploration. There is no particular reason for choosing the ATLAS case: the followed steps can be applied also for the other experiments, except for the parsing part that has to be modified due to the fact that the rules of logging are different.

3.1.1 Frontend logging

The Frontend stores information about the service status and about the SRM requests received and managed by the process. In order to manage the considerable amount of requests (of the order of a few dozen per second in a normal day), two Frontend services have been deployed for the ATLAS experiment on two different servers (storm-fe-atlas-07 and storm-atlas) logging two different files, named *storm-frontend-server.log*. An example of the *storm-frontend-server.log* file content is shown in Figure 2.

```
12/01 03:48:11.701 Thread 41 - INFO [41eec2e2-80a7-4c1b-82b4-d42ff57f0b7e]: Result for request
 'PTP status' is 'SRM_REQUEST_INPROGRESS'
12/01 03:48:11.717 Thread 13 - INFO [1b3a9db9-1325-467f-9b8b-68347dbb6ad3]: process_request :
Connection from 2001:1470:ff80:12:8e23:c32e:495d:3846
12/01 03:48:11.849 Thread 13 - INFO [1b3a9db9-1325-467f-9b8b-68347dbb6ad3]: Request 'BOL statu
s' from Client IP='2001:1470:ff80:12:8e23:c32e:495d:3846' Client DN='/DC=ch/DC=cern/OU=Organic
Units/OU=Users/CN=atlact1/CN=555105/CN=Robot: ATLAS aCT 1' # Requested token '17b26868-7752-4e3
0-bcdb-c05d1f72c1b9'
12/01 03:48:11.852 Thread 13 - INFO [1b3a9db9-1325-467f-9b8b-68347dbb6ad3]: Result for request
 'BOL status' is 'SRM_REQUEST_INPROGRESS'
```

Figure 2: Example of the *storm-frontend-server.log* file content.

Each line contains the 'datetime', the 'thread' that manages the request, the 'type' of the message (that can be INFO, WARN, ERROR and NONE), the ‘request-id’ and the actual content of the message.

Another service linked to the Frontend is *monitoring.log*. It provides information about the operations executed in a certain amount of time, called Monitoring Round, that is 1 minute by default.

3.1.2 Backend logging

The Backend log files provide information on the execution process of all SRM requests. StoRM Backend log files are the followings:

1. *storm-backend.log*, the main log file where each single request and errors are logged;

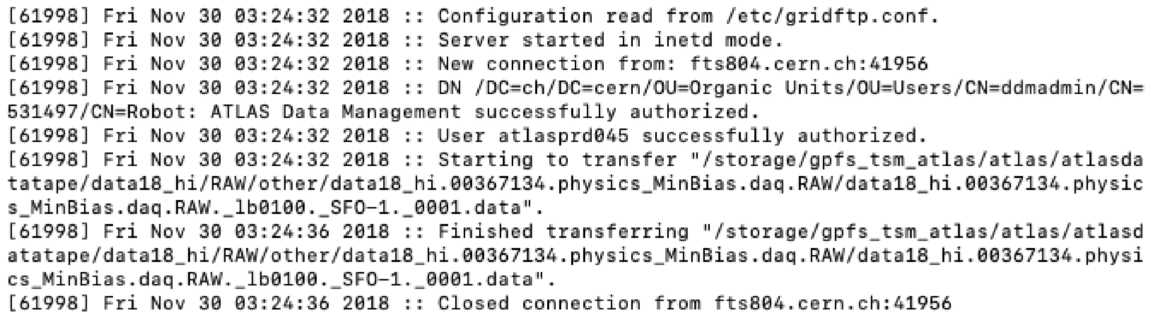
2. *heartbeat.log*, an aggregated log that shows the number of synchronous and asynchronous requests occurred from startup and on last minute;

3. *storm-backend-metrics.log*, a finer grained monitoring of incoming synchronous requests, containing metrics for individual types of synchronous requests.

The Backend service stays in the the storm-atlas machine, the same in which one of the two Frontend services is located.

3.2 GridFTP

StoRM involves the GridFTP middleware component [12] to perform file transfer operations. In the current deployment at the INFN-CNAF site, StoRM uses for the ATLAS experiment two GridFTP servers, called ds-808 and ds-908. The information about GridFTP processes is stored in two log files, *storm-globus-gridftp.log* and *storm-gridftp-session.log*. They have a similar content, and for our purposes the second will be explored and used. It contains all the information regarding transfers, namely the hostname opening the connection, the DN (Distinguished Name) and the alias of the user, the TURL of the transferred file. An example of the *storm-gridftp-session.log* file content is shown in Figure 3.

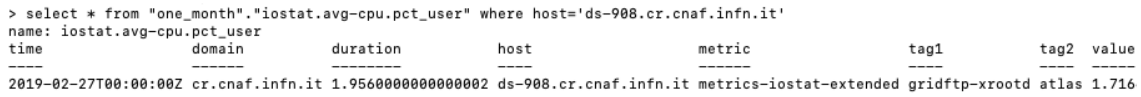


```
[61998] Fri Nov 30 03:24:32 2018 :: Configuration read from /etc/gridftp.conf.
[61998] Fri Nov 30 03:24:32 2018 :: Server started in inetd mode.
[61998] Fri Nov 30 03:24:32 2018 :: New connection from: fts804.cern.ch:41956
[61998] Fri Nov 30 03:24:32 2018 :: DN /DC=ch/DC=cern/OU=Organic Units/OU=Users/CN=ddmadmin/CN=
531497/CN=Robot: ATLAS Data Management successfully authorized.
[61998] Fri Nov 30 03:24:32 2018 :: User atlasprd045 successfully authorized.
[61998] Fri Nov 30 03:24:32 2018 :: Starting to transfer "/storage/gpfs_tsm_atlas/atlas/atlasda
tatape/data18_hi/RAW/other/data18_hi.00367134.physics_MinBias.daq.RAW/data18_hi.00367134.physic
s_MinBias.daq.RAW._lb0100._SFO-1._0001.data".
[61998] Fri Nov 30 03:24:36 2018 :: Finished transferring "/storage/gpfs_tsm_atlas/atlas/atlasd
atatape/data18_hi/RAW/other/data18_hi.00367134.physics_MinBias.daq.RAW/data18_hi.00367134.physi
cs_MinBias.daq.RAW._lb0100._SFO-1._0001.data".
[61998] Fri Nov 30 03:24:36 2018 :: Closed connection from fts804.cern.ch:41956
```

Figure 3: Example of the storm-gridftp-session.log file content.

3.3 CNAF monitoring system

The monitoring infrastructure at CNAF is based on InfluxDB [13] as time series database to store data gathered from various sensors. InfluxDB is targeted at use cases for DevOps, metrics, sensor data, and real-time analytics. Some of the key features that InfluxDB currently supports are: SQL like query language, HTTP(S) API, storing of billions of data points, database managed retention policies for data. An example of a query is shown in Figure 4. When a query is done to the database, the retention policy (RP) has to be specified: this can be 1 week, 1 month, 6 months, 1 year. The first two RPs have a finer mode of logging, 1 row at each minute and 1 row at each 15 minutes respectively.



```
> select * from "one_month"."iostat.avg-cpu.pct_user" where host='ds-908.cr.cnaf.infn.it'
name: iostat.avg-cpu.pct_user
time                 domain          duration           host                   metric                    tag1           tag2  value
----                 ------          --------           ----                   ------                    ----           ----  -----
2019-02-27T00:00:00Z cr.cnaf.infn.it 1.9560000000000002 ds-908.cr.cnaf.infn.it metrics-iostat-extended gridftp-xrootd atlas 1.716
```

Figure 4: Example of a query to InfluxDB.

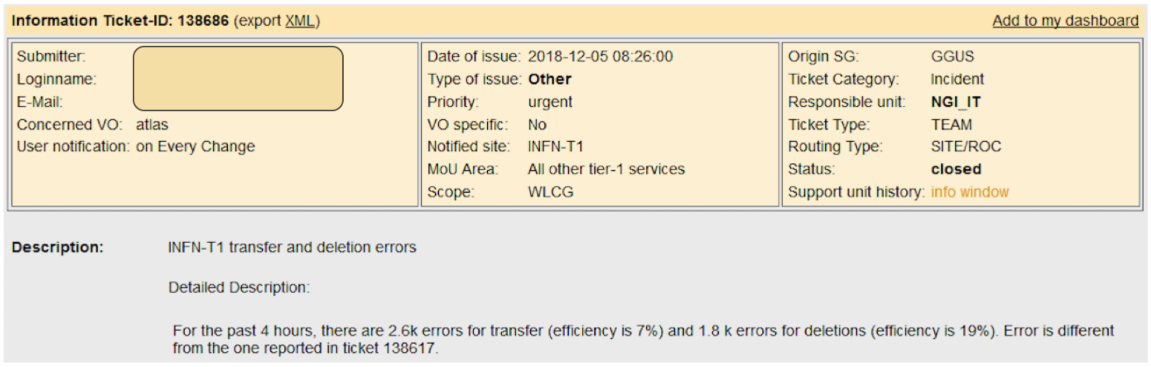
4. Machine Learning approaches on CNAF logs

As previously mentioned, CNAF aims at building a predictive maintenance system using data stored in log files. There are many log sources and each one has a different structure with a different content. A big effort has been devoted to parse all these information, creating different classes for each source that take as input a log file and give as output a csv file. Therefore the stored information (features) is large, and it is not easy to use them for a unique prediction system. As a preliminary approach, it has been preferred to start with the sources individually taken, investigating the behavior of the features contained inside each log file. Then ML techniques have been used in order to create a model for each of them that would be able to predict anomalies. The possible sources to be used for this purpose are: metrics suitably taken from InfluxDB, storm-frontend-server.log files (for both storm-fe-atlas-07 and storm-atlas), monitoring.log, storm-backend.log, heartbeat.log, storm-backend-metrics.log, storm-gridftp-session.log files (one for ds-808 and one for ds-908). Currently subset of these sources have been used, however including all of them in order to have as much information as possible for a better predictive system will be our next move.

4.1 Choice of the "critical period"

An important aspect is the definition of a period of interest in which an anomaly of the system is detected. In this way it is possible to compare the features in this period respect to a normal one, so to study differences in behavior and correlations. In this ML techniques may help to define which features are more important for a discrimination between the two different periods and may provide a way to forecast an anomaly similar to that detected in the problematic one.

In this work, the days ranging from December 5th to December 8th have been selected as the problematic period, while the days between December 1st to 4th of have been selected as the normal one. These days have been selected because an anomaly was detected in the morning of the December 5th, as reported by a GGUS ticket (see Figure 5). After a first investigation, the problem seemed to be related to the storage access from the farm worker-nodes. Such access, for the ATLAS experiment, is related to jobs of two kinds: *Pilot*, which are jobs coming from the farm, and *Data Management*, which are jobs coming from the Grid. In order to disentangle the problem, starting from the afternoon of December 7th no more jobs from the farm (of type *Pilot*) were accepted until December 10th. On December 13th the situation improved and two issues were detected: a wrong configuration of the file system and a wrong configuration of the ATLAS Production and Analysis system (PANDA) queues. Moreover, a new GridFTP server was added in order to manage rate of requests.


Information Ticket-ID: 138686 (export XML) Add to my dashboard

Submitter:	[redacted]	Date of issue:	2018-12-05 08:26:00	Origin SG:	GGUS
Loginname:		Type of issue:	**Other**	Ticket Category:	Incident
E-Mail:		Priority:	urgent	Responsible unit:	**NGI_IT**
Concerned VO:	atlas	VO specific:	No	Ticket Type:	TEAM
User notification:	on Every Change	Notified site:	INFN-T1	Routing Type:	SITE/ROC
		MoU Area:	All other tier-1 services	Status:	**closed**
		Scope:	WLCG	Support unit history:	info window

Description: INFN-T1 transfer and deletion errors

Detailed Description:

For the past 4 hours, there are 2.6k errors for transfer (efficiency is 7%) and 1.8 k errors for deletions (efficiency is 19%). Error is different from the one reported in ticket 138617.

Figure 5: Part of the content of the ticket notifying the anomaly. Details on the submitter has been blanked out.

For what concern the improper configuration of the filesystem, it is possible to check which is the maximum limit of memory in the GPFS cluster, how much memory has been assigned and the quota that remains in doubt. The sum of the two contributions should not overcome the maximum limit, but a StoRM process can consider that there is free space even if the sum overcomes the limit. In this case, StoRM tells to GridFTP that there is free space even if it is not possible to write on the file system, hence the transfers fail. The quota disk of GPFS is almost 30 PB, and the doubt quota disk was of the order of 200-300 TB during the problematic days, whereas in a normal day it is of the order of 1 TB. In this situation, the sum of the assigned memory plus the doubt quota was almost, or overcoming, the limit quota. This problem was detected and solved in the evening of the December 9th.

The storage-farm access modalities are defined within the ATLAS Workload Management System [14] and the ATLAS Grid Information System (AGIS [15]), where several "copytool" can be used to optimize the storage access at a given site. For the INFN-T1, data on storage are read-accessed from the farm worker nodes via POSIX access (since the GPFS file system is exported and mounted on all the worker nodes). This default (primary) access mode is managed by the "storm copytool", written ad hoc for sites having StoRM for storage management. An alternative access could be done copying the input data on the WN with wrapped gfal utilities (the copy request passes through SRM-GridFTP). This procedure is managed by the "rucio copytool". Writing action instead has to be always authenticated so the *Pilot* certification is used to allow to write on the storage passing through StoRM, and the "rucio copytool" is always used. The problem about the wrong configuration of the queues was due to the fact that "storm" (POSIX) access was not set as primary and the "rucio copytool" was selected, this causing an abnormal increase of access through StoRM-GridFTP and overload of the system.

As detailed, in the first days of December, unfortunately there were two problems of different nature and also several interventions made by the experts in the meanwhile before solving the critical situation. There is no certainty that these two facts were the cause of the problem and there is no certainty that the period considered as "normal" did not show any problem; on the contrary, despite the two issues were discovered in the critical period, it is very likely that they existed also in the previous days. The only thing we know is that in the normal period the system seemed to be robust against these facts, without showing any critical state. So the idea is to compare the problematic period to the immediately previous one using ML techniques, in order to try to assess whether there is a real difference between the two periods and whether there is a chance to predict the anomalies in advance.

4.2 Procedure

In the case of the *storm-frontend-server.log* file, considering the sum of logs from the two different hosts storm-fe-atlas-07 and storm-atlas, the rate of logging is high (almost 50 Hz, so 50 rows per second) and consequently the files are not very handy given their size (one day logging is of the order of 1 GB). Therefore, a possible way to manage these files was to try to summarize their content in only one row every 15 minutes. In this way for example we can say how many PrepareToPut (PTP) or PrepareToGet (PTG) were done in some fixed 15 minutes only summing the times that these terms appeared as "request" in the log message. The duration of 15 minutes is arbitrary but justified: data in InfluxDB used for this work have a retention policy of 1 month, so there is 1 row every 15 minutes. From a point of view of comparison between different sources, we need to standardize the data and thus 15 minutes seems a reasonable choice. Moreover, 15 minutes are also enough to eventually provide a warning in case an anomaly is about to come.

The steps followed for each source are:

- parse log files, converting them in the csv form;
- study the content of the messages and then decide if they contains sub-information that must become new features for the ML techniques;
- summarize the content in one row at each 15 minutes;
- check the correlation of features;
- use ML algorithms for the classification problem at issue, where logs from "good" days are assigned to the "0" class and logs from "bad" days are assigned to the "1" class;
- define the most important features for the discrimination between the two days and use them in the building of the best models.

5. Results

At the beginning of this work, log files from the single sources have been individually taken and only later the information has been combined to check for correlation.

5.1 InfluxDB metrics

Many metrics can be extracted from the Influx database. In Table 1 the ones selected are shown together with a description.

Metric	**Description**
gpfs_atlas.*	* (read, write) reading and writing speed from the file system for the two GridFTP machines measured in bytes per second
interface.bond0.*xBytes	bytes * (r,t) received and transferred on the net interface bond0
interface.bond0.*xDrops	packets lost in * (r,t) reading and writing on the net interface bond0 measured in bytes
interface.bond0.*xErrors	* (r,t) reading and writing errors on the net interface bond0 measured in bytes
iostat.avg-cpu.pct_*	percentage of time where the cpu is * (idle, iowait, nice, user, system)
load_avg.five_*	average over 5 minutes of the CPU load average for the two GridFTP machines and the two Frontend services
storm.async_*_*_storm-atlas	average number of * (ptg, ptp), the average of those that fails, of those that are successfully ended, average in duration * (n, fail, ok, time) in the machine storm-atlas
storm.sync_storm-atlas	average number of synchronous operations for the storm user in the machine storm-atlas
user_percent.*.*	* (cpu, mem) CPU time, memory used by the storm process in the machine * (storm-atlas, storm-fe-atlas-07)
perc_mem_free_*	percentage of free memory of the machines where the two GridFTP and the two Frontend services are located

Table 1: Description of the metrics extracted from InfluxDB for the present work (they are 47 in total).

The correlation matrix of the more meaningful metrics (among the 47 total metrics), considering only “bad” days and with the absolute value of the correlation coefficients greater than 0.6, is shown in Figure 6 .

A dozen of ML algorithms has been used (including AdaBoost classifier “AB”, XGBoost “XGB”, RandomForest “RF” and Multi-layer Perceptron classifier “MLP”), all of them taken from the scikit-learn library [17]. No ML frameworks (such as Tensorflow or Theano) have been used and no hyperparameter tuning has been performed, since the performances of the models obtained are enough for our limited purpose. Indeed, for all of them the accuracy is close to 100% (see Figure 7) and the area under ROC curve is close to 1.0.

This result shows that every algorithm is able to predict almost always that an input row with target “0” comes from a good day and one with target “1” comes from a bad day. It means that the

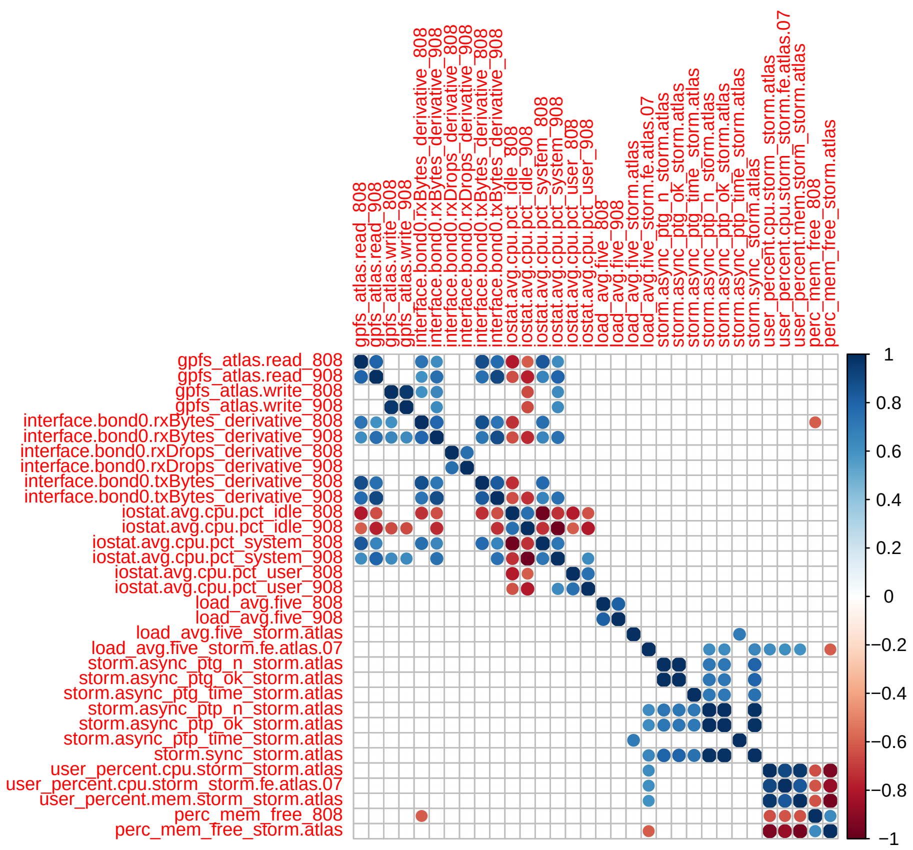


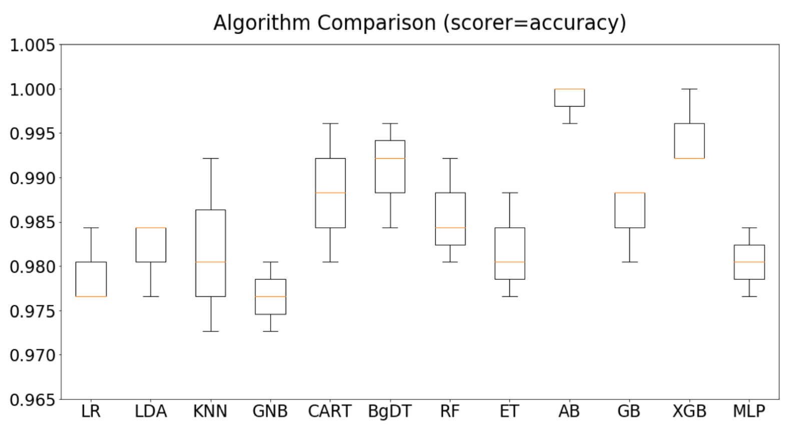
Figure 6: Correlation matrix of the more interesting InfluxDB metrics considering only "bad" days, with the absolute value of the correlation coefficients greater than 0.6.

metrics used have an intrinsic behavior which is very different in general between the two different kind of days.

A possible improvement of this analysis can be the definition of the more relevant features for the discrimination between the two kind of days and used in model construction. In literature different techniques are proposed for what is called "feature selection" [16].

- Statistical tests can be used to select those features that have the strongest relationship with the output variable. The scikit-learn library provides the SelectKBest class that can be used with a suite of different statistical tests to select a specific number of features. One of them is the chi squared (chi^2) statistical test [18]. In this method the chi-squared stats is computed between each non-negative feature and class. This score can be used to select the number of features with the highest values for the test chi-squared statistic. The chi-square test measures dependence between stochastic variables, so this function "weeds out" the features that are the most likely to be independent of class and therefore irrelevant for classification.

- The Recursive Feature Elimination (or RFE) [19] works by recursively removing attributes and building a model on those that remain. It uses the model accuracy to identify which



LR: LogisticRegression
LDA: LinearDiscriminantAnalysis
KNN: KNeighborsClassifier
GNB: GaussianNB
CART: DecisionTreeClassifier
BgDT: BaggingClassifier
RF: RandomForestClassifier
ET: ExtraTreesClassifier
AB: AdaBoostClassifier
GB: GradientBoostingClassifier
XGB: XGBoostClassifier
MLP: MultiLayerPerceptronClassifier

Figure 7: Accuracy scores for different ML algorithms using InfluxDB metrics.

attributes (and combination of attributes) contribute the most to predict the target attribute.

- Principal Component Analysis (or PCA) [20, 21] uses linear algebra to transform the dataset into a compressed form. The PCA procedure produces eigenvectors-eigenvalues pairs where an eigenvalue tells us how much variance there is in the data in the direction defined by the eigenvector. The higher each component of the eigenvector is, the more that component is important in the definition of the direction given by that eigenvector. The eigenvector with the highest eigenvalue is therefore the principal component.

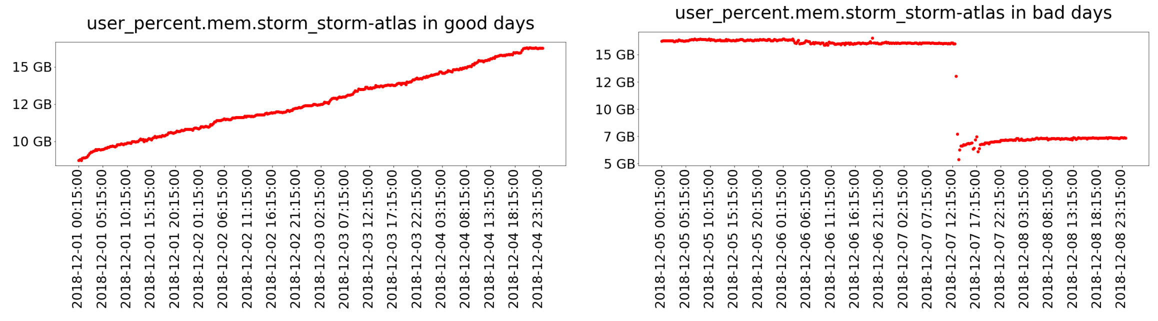
- A benefit of using ensembles of decision tree methods like gradient boosting (i.e. XGBoost) or adaptive boosting (i.e. AdaBoost) is that they can automatically provide estimates of feature importance from a trained predictive model. The importance of a feature is the increase in the prediction error of the model after we permuted the features values [22]. Generally, importance provides a score that indicates how useful or valuable each feature was in the construction of the boosted decision trees within the model. The more an attribute is used to make key decisions with decision trees, the higher its relative importance.

As already said the algorithms used give almost the same result in terms of accuracy (and also in terms of area under ROC curve). For this reason, in the case of RFE and feature importance, only the AdaBoost algorithm was considered as the more performing. In order to make feature selection from these four different methods, the need arises to create only one summary way to give a final ranking of the features. The approach adopted is the following. For each one of the four methods there is a feature ranking and, as if it was a sport competition, a score is assigned from 1st to 10th position giving the following points (starting from the first): 25, 18, 15, 12, 10, 8, 6, 4, 2, 1. At the end the final ranking is produced summing up the intermediate scoring. This procedure gives the feature ranking shown in Table 2.

Here the *user_percent.mem.storm_storm-atlas* metric comes out in the first position with a great detachment from the second metric so it can be considered as the main feature to infer that in the whole system there is a discrepancy in values between good days and bad days . A comparison between these two different behaviors is shown in Figure 8.

	Metric	**Scoring**
1	user_percent.mem.storm_storm-atlas	75
2	user_percent.cpu.storm_storm-atlas	37
3	perc_mem_free_storm-fe-atlas-07	36
4	perc_mem_free_808	31
5	interface.bond0.txBytes_derivative_808	25

Table 2: Final scoring of the 5 best metrics obtained by the described feature selection procedure. See in the text for more details.



(a) (b)

Figure 8: Comparison of the memory usage from the storm user in the storm-atlas machine between good days (8a) and bad days (8b).

5.2 *storm-frontend-server.log*

For the *storm-frontend-server.log* files, the approach used is the same shown for Influx metrics, but this time the files contain more complex messages. The main elements that may appear in a row of a log file, which become the columns of the parsed csv files, are: ‘datetime’, ‘thread’, ‘type’, ‘request-id’,‘ DN’, ‘requested token’, ‘num_surl’, ‘IP’, ‘request’ and ‘result’. The ‘thread’ manages the request. The ‘type’ can be INFO, WARN, ERROR, NONE in increasing order of the critical issue of the message. An INFO message contains general information about the system, NONE type is related to the configuration of the service. ‘Request-id’ is the identifier of the user connecting to the service, thus the progress in time of the actions and requests made by the user can be seen by grepping for the request-id in the log file. ‘DN’ is the distinguished name related to the user. The ‘requested token’ is the identifier related to the request of the user; through it, the status of the request can be monitored in the *storm-backend.log* file and allows the Backend to keep track all the requests. The ‘num_surl’ is the number of site URLs related to the files on which a request is done. ‘IP’ is the IP address related to the user that has connected to the service. In the end, the last elements that may appear are ‘result’ and ‘request’. For each of them there are many possibilities that will not be explained here.

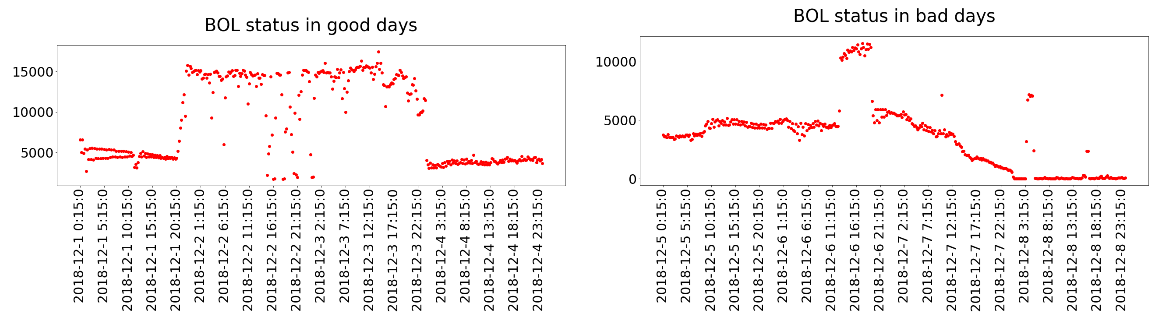
As said the rate of logging in the *storm-frontend-server.log* files is very high and thus we decided to aggregate the behavior of the service at each 15 minutes. For this purpose, the csv file coming from the parsing operation has to be changed, where its columns will be the ones previously explained plus one column for each kind of ‘type’, ‘request’ and ‘result’ present in the log files. Some columns will be deleted because not useful anymore (‘thread’ and ‘request-id’), and other will be changed (‘DN’, ‘IP’ and ‘requested token’). These last ones will not contain anymore the specific element of string type but a ‘1’ when a DN, an IP or a requested token is inside the message, otherwise a ‘0’. Finally the ‘one hot encoding’ procedure can be made: the result is that each message of the log will be converted in zeros and ones, where the ones are related to those column containing part of that message. At the end, lines aggregating the statistics of the previous 15 minutes are written in the final csv.

The features produced for this file are 53. The ML techniques have been used and, as in the case of InfluxDB, models with high accuracy (over 0.93) and high area under ROC curve (over 0.96) have been produced. Also here AdaBoost algorithm resulted as the most performing. Then, the features selection procedure previously shown has been applied, obtaining the results shown in Table 3.

	Metric	**Scoring**
1	BOL status	58
2	Abort request	55
3	num_surl	55
4	Rm	30
5	SRM_INTERNAL_ERROR	25

Table 3: Final scoring of the 5 best features from the *storm-frontend-server.log* files obtained by the feature selection procedure.

Here the *BOL status* (Bring On Line), *Abort request* and *num_surl* features are in the first three positions and they can be considered as the main features to be used to infer that in the whole system there is a discrepancy in values between good days and bad days. A comparison between these two different behaviors for the first feature is shown in Figure 9.



(a) (b)

Figure 9: Comparison of the number of BOL status requests between good days (9a) and bad days (9b).

5.3 *storm-gridftp-session.log*

This log file contains all the information regarding data transfers. In the parsing action, many columns have been added into the csv file (where the csv file contains a row for each transfer), that are: the kind of user (atlas, pilatlas, atlasprd), the storage area of the transferred file location (atlasdatadisk, atlasdatatape, atlasscratchdisk, atlasmctape), the different kinds of failures, the different providers related to main IPs of who connects and the duration of the transfer. Each time a transfer has one of these information in the log file, a "1" is added to the corresponding column. A "0" is added otherwise. In order to sum up the behavior of the system every 15 minutes, all the columns of the resulting csv file will have the sum of the content in the last 15 minutes. About the "duration" column, it has been substituted by the mean, the maximum, the 50th percentile and the 95th percentile of the transfers duration of the 15 minutes, in addition to the number of transfers.

The features produced for this file are 32. The previously described ML algorithms have been used and all of them gave good results, with an accuracy over 0.94 and an area under ROC curve over 0.96. The AdaBoost classifier produced the best score for both the two metrics. Same result is for both ds-808 and ds-908 GridFTP servers. Then, the features selection procedure previously shown has been applied, obtaining the results shown in Table 4 for the ds-908 GridFTP server.

	Metric	**Scoring**
1	abort	62
2	disk_area_atlasdatatape	45
3	duration_mean	39
4	DN_ADM	37
5	globus_xio: System error in send	30

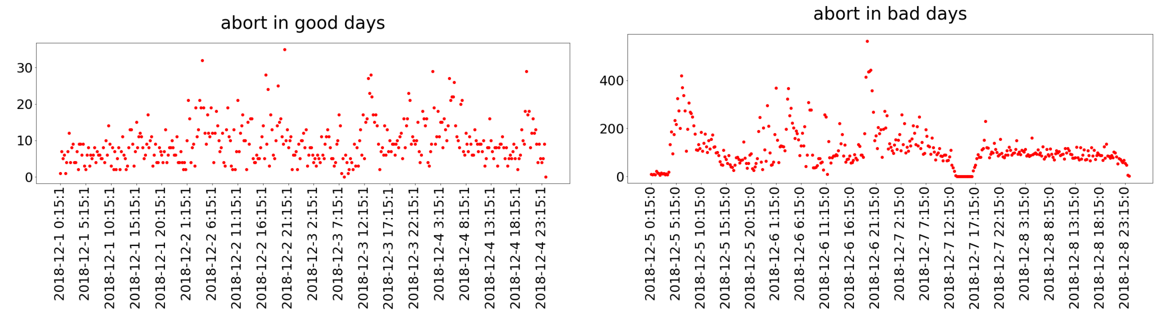
Table 4: Final scoring of the 5 best features obtained by the feature selection procedure for the ds-908 GridFTP server.

The *abort* feature is in the first position in the feature selection procedure, as shown in Table 4, and it can be considered as the main feature to be used to deduce that in the whole system there is a discrepancy in values between good days and bad days. A comparison between these two different behaviors for this feature is shown in Figure 10.

After the exploration of these three different sources (InfluxDB, *storm-frontend-server.log*, *storm-gridftp-session.log*) individually taken, then the correlation matrix of the features taken from all these sources is computed and it is of great interest for the technicians because it provides a way to compare elements coming from different sources with a different content and maybe find unexpected correlations. Here it is not shown because such a detailed analysis is not the scope of this paper which instead focuses on the more general trends.

6. Conclusions

One of the future goals of CNAF is the implementation of a global predictive maintenance solution for the site. Since efficient storage systems are an essential component of Tier-1 operations,



(a) (b)

Figure 10: Comparison of the number of abort requests between good days (10a) and bad days (10b).

a good starting point has been identified in the StoRM service. The present work is based on data coming from this service (and from those related to it) in the form of log files. The log sources are different and with different information content. A considerable effort has been spent to extrapolate a usable content from log files that have an inhomogeneous structure. Surely a standardized way of keeping logging, with defined and clear rules reducing the variability at minimum (i.e. by using a simple uniform date format for the different sources), would be an essential ingredient for a good automated system of parsing, but unfortunately at the moment this does not exist at CNAF. Clearly this is a fundamental part for a center aiming a collecting information from many services in a structured form so to control the behavior of the global system in a different way with respect to the actual monitoring of the machines and the manual checks made by technicians every time a problem occurs.

After having extracted these information, the aim was to use them for the building of ML models that were able to predict when an anomaly occurs in the system. This paper describes how the information has been used to show correlation of features, to show which of those features can be considered as the main ones to define the critical state of the system, and to build ML models that are able to say which is the probability that the system is in a critical state. It means that a built ML model tells us which is the health status of the system, every 15 minutes, in an automated way. Regarding the second point, the features found for the log files used are: *user_percent.mem.storm_storm-atlas* for the Influx Database, *BOL status*, *Abort request* and *num_surl* for the *storm-frontend-server.log* files, *abort* for the *storm-gridftp-session.log* file (related to the ds-908 GridFTP server).

This work has been performed considering data of a particular period but, despite the specific critical situation, the procedure is general and usable for any kind of situation. Moreover the obtained result can be partly exploited for each problem (at least similar ones) because the used features describe the overall system behavior, and therefore whenever a problem occurs the system shows a peculiar behavior that can be generalized in some way.

The work performed is preliminary and one of the reason is that not all the available log sources have been used. In fact, the *monitoring.log*, *storm-backend.log*, *heartbeat.log* and *storm-backend-metrics.log* files have not been used at the moment and they can surely give a whole description and a whole knowledge of the system for the ML purpose. Other obvious improvements could be achieved collecting data for more critical period in order to test the ML model produced and eventually create other models for different cases.

References

[1] *CNAF*. https://www.cnaf.infn.it/en/institute/

[2] *StoRM*. http://italiangrid.github.io/storm/

[3] E. Corso, S. Cozzini, A. Forti et al., *StoRM: A SRM solution on disk based storage system*, in Proceedings of the Cracow Grid Workshop 2006 (CGW2006), Cracow, Poland, 2006.

[4] A. Carbone, L. dell'Agnello, A. Ghiselli et al., *Performance Studies of the StoRM Storage Resource Manager*. 3rd IEEE International Conference on e-Science and Grid computing (eScience2007), Bangalore, India, December 2007, pp. 423-430.

[5] L. Magnoni, R. Zappi and A. Ghiselli, *StoRM: A flexible solution for Storage Resource Manager in grid*. IEEE Nuclear Science Symposium Conference Record (2008), pp. 1971-1978.

[6] R. Zappi, *Storage Resource Managers: Interface and SRM services*. Presentation at the INFN Training School for Grid User, November 2007.

[7] R. Zappi, E. Ronchieri, A. Forti et al., *An Efficient Grid Data Access with StoRM*. Data Driven e-Science, Springer New York (2011), pp. 239-250.

[8] *StoRM: a Manager for Storage Resource in Grid*. http://italiangrid.github.io/storm/documentation/functional-description/1.11.2/

[9] *StoRM System Administration Guide*. http://italiangrid.github.io/storm/documentation/sysadmin-guide/1.11.14/

[10] *What is the ELK Stack?*. https://www.elastic.co/elk-stack

[11] T. Diotalevi et al., *Collection and harmonization of system logs and prototypal Analytics services with the Elastic (ELK) suite at the INFN-CNAF computing centre*. In this conference.

[12] *GridFTP*. http://toolkit.globus.org/toolkit/docs/latest-stable/gridftp/

[13] S. Bovina, D. Michelotto, G.Misurelli, *CNAF Monitoring system*. CNAF Annual Report (2015), pp. 111-114.

[14] *Overview of ATLAS PanDA Workload Management*. J. Phys.: Conf. Ser. 331 072024

[15] *AGIS: Integration of new technologies used in ATLAS Distributed Computing*, J. Phys.: Conf. Ser. 898 092023

[16] *Feature Selection For Machine Learning in Python*. https://machinelearningmastery.com/feature-selection-machine-learning-python/

[17] *scikit-learn*. https://scikit-learn.org/stable/

[18] *sklearn.feature_selection.chi2*. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html

[19] *sklearn.feature_selection.RFE*. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html

[20] I. T. Jolliffe, *Principal Component Analysis*. Springer-Verlag, New York (2002).

[21] M. Arnaz and G. Robert, *PCA-Based Feature Selection Scheme for Machine Defect Classification*. IEEE Transactions on Instrumentation and Measurement (2005). 53(6), pp. 1517 - 1525.

[22] *Feature Importance*. https://christophm.github.io/interpretable-ml-book/feature-importance.html