

ALMA MATER STUDIORUM · UNIVERSITÁ DI BOLOGNA

SCHOOL OF SCIENCE

Department of Physics and Astronomy (DIFA)

Master Degree in Applied Physics

**Anomaly Detection prototype
for log-based Predictive Maintenance
at INFN-CNAF Tier-1**

Supervisor:

Prof. Daniele Bonacorsi

Presented by:

Francesco Minarini

Co-Supervisors:

Dr. Leticia Decker de Sousa

Dr. Luca Giommi

Academic Year 2018-2019

*If you torture the data long
enough, it will confess*

–**Ronald Coase**

*It's tough to make predictions,
especially about the future*

–**Danish proverb**

Abstract

Separare nettamente l'evoluzione della fisica delle alte energie dalle risorse computazionali da essa richieste é, oggi, impossibile. LHC, ogni anno, produce infatti dozzine di PetaBytes di dati, la cui gestione richiede il coordinamento di *storage*, risorse di calcolo e *networks* ad alte prestazioni. Come conseguenza del passaggio a HL-LHC, é previsto un significativo aumento nel volume di dati. Ciò implica la necessità di un parallelo *upgrade* delle risorse computazionali. In tal senso, la *HEP Software Foundation* ha pubblicato un documento ufficiale in cui vengono definite le tappe di sviluppo delle piattaforme *software* e *hardware* necessarie a rendere le infrastrutture di calcolo adeguate alla fase ad Alta Luminosità. INFN-CNAF, centro di calcolo nazionale dell'INFN, contribuendo e partecipando all'aggiornamento collettivo, ha impostato uno studio preparatorio con l'obiettivo di definire un paradigma di manutenzione predittiva automatizzata basata su log file dei servizi ospitati al centro. L'obiettivo di lungo termine consiste sia in un miglioramento delle performance sia nella costruzione di un'infrastruttura di calcolo altamente efficiente e automatizzata.

La tesi contribuisce a tale studio sviluppando un prototipo originale in grado di identificare, in modo *unsupervised*, intervalli temporali critici nei log file giornalieri di un dato servizio relativi ad un dato giorno e di esplorarli mediante tecniche di *Text Processing*, estraendo così contenuto informativo di alto livello.

Abstract

Splitting the evolution of HEP from the one of computational resources needed to perform analyses is, nowadays, not possible. Each year, in fact, LHC produces dozens of PetaBytes of data (e.g. collision data, particle simulation, metadata etc.) that need orchestrated computing resources for storage, computational power and high throughput networks to connect centers. As a consequence of the LHC upgrade, the Luminosity of the experiment will increase by a factor of 10 over its originally designed value, entailing a non negligible technical challenge at computing centers: it is expected, in fact, an uprising in the amount of data produced and processed by the experiment. With this in mind, the HEP Software Foundation took action and released a road-map document describing the actions needed to prepare the computational infrastructure to support the upgrade. As a part of this collective effort, involving all computing centres of the Grid, INFN-CNAF has set a preliminary study towards the development of data+AI driven maintenance paradigm. As a contribution to this preparatory study, this master thesis presents an original software prototype that has been developed to handle the task of identifying critical activity time windows of a specific service (StoRM). Moreover, the prototype explores the viability of a content extraction via Text Processing techniques, applying such strategies to messages belonging to “anomalous” time windows.

Contents

Introduction	5
1 HEP Computing and Analytics	8
1.1 The Road to HL-LHC	9
1.1.1 HL-LHC Computing	10
1.2 Statistical/Machine Learning at LHC	11
1.3 Intelligence Operations at LHC	11
2 LHC Computing Grid	13
2.1 The WLCG project	14
2.2 The INFN-CNAF Tier-1 Facility	17
3 Maintenance paradigm at INFN-CNAF	19
3.1 Definition of “System Maintenance”	19
3.2 Maintenance criteria	20
3.2.1 Reactive Maintenance	20
3.2.2 Preventive Maintenance	21
3.2.3 Predictive Maintenance	22
3.2.4 Advanced Maintenance	23
3.3 Log-Based Maintenance	24
3.3.1 The Log Bucket	25
3.3.2 StoRM Log Files	26
3.3.3 StoRM Front-End	27
3.3.4 StoRM Back-End	28

3.4	Aim of this work	29
3.4.1	Code formal annotations	29
4	Methodology description	31
4.1	System's behaviour extraction.	31
4.1.1	Log Volatility	32
4.2	Unsupervised Learning	33
4.3	Essential concepts of Anomaly Detection	34
4.4	Support Vector Machines	37
4.4.1	One-Class SVMs	43
4.5	TFIDF Information Retrieval	44
5	Analysis results	48
5.1	Data preprocessing	48
5.2	Data analysis	49
5.2.1	Results on StoRM Front-End service	49
5.2.2	Results on StoRM Back-End service	57
5.3	Content extraction through Text Processing	63
5.3.1	Methodology and preliminary results	63
6	Conclusions and next steps	66
6.1	Next Steps	67
	Bibliography	77

Introduction

Splitting the evolution of HEP from the one of computational resources needed to perform analyses is, nowadays, not possible. Each year, in fact, LHC produces dozens of PetaBytes of data (e.g. collision data, particle simulation, metadata etc.) that need orchestrated computing resources for storage¹, computational power² and high throughput networks to connect centres and allow data transfers³. These ambitious requirements have been met by the WLCG project, an infrastructure that has always played a crucial role in LHC discoveries. The request for resources, nonetheless, is deemed to raise in the near future, after Long Shutdown 2. The LHC upgrade will, in fact, increase the Luminosity of the experiment (see Chapter 1 for details) by a factor of 10 over its originally designed value, an unprecedented result that will entail a non-negligible technical challenge, starting from Run-3 and Run-4. It is expected, as a consequence of the Luminosity increment, a significant uprising in the amount of data produced by the experiment as well as in the processing workload over Grid members, the Tiers. With this in mind, the HEP Software Foundation took action and released a road-map document [2] describing the actions needed to prepare the computational infrastructure to support the upgrade. As a result, an Operational Intelligence group was established with a view to improving, via Analytics and Machine Learning tools, the usage of present-day facilities and their adequacy in view of HL-LHC experiments.

As a part of this collective effort, involving all computing centres of the Grid, INFN-CNAF has set a preliminary study towards the development of an AI-driven

¹order of magnitude required: ExaBytes

²order of magnitude required: 10^5 high-end CPUs

³order of magnitude: 60 GB/s transfer

maintenance paradigm [3] based on log files produced by specific computational services. The foresight is, in the reasonably near future, to build up an automatised maintenance paradigm, capable of predicting when and why a service might fail, thus improving the efficiency and reliability of INFN-CNAF computing center.

As a contribution to this preparatory study, this master thesis presents an original lightweight software prototype that has been developed to handle the task of identifying critical time windows of a service (StoRM). Moreover, the prototype explores the viability of a content extraction via Text Processing techniques, applying such strategies to messages belonging to “anomalous” time windows.

Chapter 1 provides an overview of the upgrade schedule at LHC, describing HSF Community White Paper implications over the future of HEP computation. Intelligence Operations and Machine Learning approaches at LHC are, as well, discussed.

Chapter 2 provides a description of the WLCG project, with particular attention to the INFN-CNAF Tier 1 facility, which this work focuses on.

Chapter 3 defines the concept of “System Maintenance” describing its common declinations and introduces the log-based maintenance project, one of the CNAF main lines of research.

Chapter 4 Describes the theoretical methodology of the prototype developed. Essential theoretical concepts of used algorithms are given as well

Chapter 5 presents selected results of the prototype application over available StoRM log files. The discussion will highlight each step of the analysis. Finally, few remarks related to the exploration of a Text Processing tool over log messages will be given.

Chapter 6 draws conclusions from the application of the prototype and dis-

cusses future developments.

Chapter 1

HEP Computing and Analytics

This chapter sheds some light on the most interesting aspects of the major upgrade of LHC and its evolution towards unprecedented collision energy and discovery-potential. Although this Master Thesis will focus exclusively on computational and analytics-related aspects, it is nonetheless appropriate to briefly describe a deeply connected topic: the hardware upgrade LHC is bearing. In order to improve understanding of the following, a technical definition of “Luminosity” is given.

Definition 1.0.1. (*Luminosity*) *In scattering theory and accelerator physics, luminosity (\mathcal{L}) is the ratio of the number of events detected (dN) in a certain time (dt) to the interaction cross-section (σ) [1].*

$$\mathcal{L} = \frac{dN}{\sigma dt}$$

Under a computational point of view, the higher the luminosity the larger the amount of data that can be extracted from the experiment and, as a consequence, the higher the potential for useful discoveries and measurements. To give an idea of the magnitude of the foresaid amount of data, one can observe that at CMS, with current working Luminosity, the trigger performs a data selection and transfer equal to 1 GB/s [4], enough to fill the memory of the average retail laptop (\approx 500 GB HDD) in \approx 8 minutes.

1.1 The Road to HL-LHC

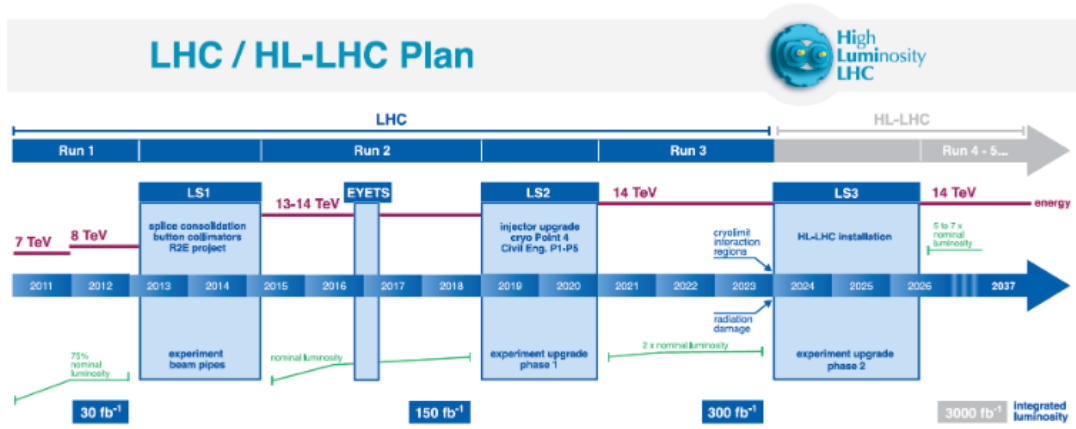


Figure 1.1: *Pictorial representation of the expected LHC evolution towards High Luminosity run.*

The HL-LHC project aims to crank up the performance of the LHC in order to increase the potential for discoveries after 2025. The objective is to increase luminosity by a factor of 10 beyond the LHC design value¹.

The first part of the project started back in 2011 and was partly financed by the European Commission's seventh framework programme (FP7). This first phase brought together many laboratories from CERN Member States, as well as from Russia, Japan and the US. The design study came to a close on October 2015 with the publication of a technical design report, followed by the start of the project construction phase at CERN and in supporting-industries. The civil-engineering work started on April 2018.

LHC is currently in its Long Shutdown 2, which is scheduled to last until 2020 included. During this Shutdown, experts will upgrade Phase-1 Injectors, aiming at the expected 14 TeV Centre-Of-Mass energy and increasing Luminosity of a factor of 3. Phase-2 will then bring peak Luminosity up to $\mathcal{L} = 7 \times 10^{34} \text{cm}^{-2} \text{s}^{-1}$. An integrated Luminosity of 300fb^{-1} per year will be achieved thanks to new-gen 16 T focusing magnets and crab-cavities. In the final scenario, a mean value of 300

¹ $\mathcal{L}_{design} = 10^{33} \text{cm}^{-2} \text{s}^{-1}$

interaction per bunch is expected to be reached [5]

The project is led by CERN with the support of an international collaboration of 29 institutions in 13 countries, including the United States, Japan and Canada, with a material budget for the accelerator of 950 million Swiss francs for 2015-2026 term.

1.1.1 HL-LHC Computing

Since such an upgrade will imply the production of a huge amount of previously unseen data and, as a consequence, facilities for storage as well as computing systems will require a parallel improvement. In fact, even though often mistakenly underestimated, only a properly structured data analysis segment will allow the whole scientific community to take full advantage of this huge upgrade currently in progress.

Having set this as a goal, the HEP Software Foundation (HSF) released a Community White Paper (CWP) [2] that paves the expected road-map for software and computing sector adjustment towards HL-LHC. Following the priorities of the CWP, the most pressing steps are:

- optimisation of software scalability and computational efficiency (as well as hardware usage) in function of the future amount of data,
- activation of fruitful new Analytics approaches implementation,
- assurance of software long-term sustainability,
- assurance of appropriate career recognition to physicists that specialise in the Analytics and software development sector.

As quickly outlined , a lot of work has been done and even more is expected to be performed in the near future under both hardware and software aspects. This Master Thesis work will mostly focus on two topics of this enormous collective effort: Analytics and Statistical/Machine Learning rising as promising methods for physics data analysis.

1.2 Statistical/Machine Learning at LHC

Definition 1.2.1. (*Machine Learning*) *Machine Learning is a branch of Artificial Intelligence that applies statistical methods to perform tasks of growing complexity without a specific user instruction, relying only on recognition of recurrent data structures, often referred to as “patterns” [6].*

Through the years, Machine Learning has proven to be a very performing and flexible tool in data analysis, to the point that it changed the way reality is observed. A simple example can be the growing development of the Self-Driving Cars, whose creation has been a direct consequence of Machine Learning proving to be able to perform better traffic handling decision as well as a safer driving conduct.

Also the scientific world has benefitted from the application of Machine Learning. For instance, Machine Learning techniques have found a fertile ground also in the domain of computational optimization, allowing the community to take advantage of data-driven efficiency improvements and perform faster and with higher reliability (e.g. [9]) than traditional approaches without a significative expense in terms of hardware overhauls.

This latter is rapidly growing its importance in the domain of LHC computation, as the High Luminosity run will necessarily bring huge challenges to the table of data analysis.

Considering the foresaid use case, the focus is identifying the state of the art and the new challenges in this domain.

1.3 Intelligence Operations at LHC

In order to fully exploit the hardware upgrade of LHC, the scientific community of HEP physics will also have to take action in the domain of software development, optimization and maintenance.

The first step towards an highly improved computational efficiency at computing centres has already been done, for example, by recollecting log data coming from services and making it available for further analyses with Big Data inspired meth-

ods. Nonetheless, the ambitious goal of handling the data flux of HL-LHC can be reached only after a huge collective effort: as a consequence, recently (Spring 2019), an official OpInt team has been kicked off to focus on this topic. Its goal is to reach a deeper (actionable) understanding of a variety of computing operations: among these goals, one is to study fruitful applications of this knowledge, for instance tailoring Machine Learning algorithms to improve and increase the level of automation of analytical pipelines.

Some of these applications, coming from both previous existent studies and recent OpInt powered discoveries have proven their value in the following use-cases:

- ML methods helped avoiding network congestion by predicting data transfer patterns, [7]
- time-series analysis helped improving the estimate of on-CPU time of applications, [8]
- anomaly Detection helped preventing system failures. [9]

The result of this effort will be a complete change of paradigm when it comes to computing maintenance: from the simple Breakdown paradigm Today, to the expected Predictive one Tomorrow. These terms will be explained in depth in chapter 3.

Chapter 2

LHC Computing Grid

The term “Computational/Computing Grid” refers to an organisation of dislocated computer resources linked together via high performance networks to reach a pre-defined common goal. In the domain of physics, this goal is to provide to each member of the community the correct amount of resources to efficiently and independently process physical data.

The WLCG is based on an ad-hoc software called “middleware” that enables actual experiments’ applications to access and use the deployed hardware. The building blocks of such middleware, in terms of Grid components, are briefly presented in the following [4]:

Computing Element (CE) manages the user’s requests for computational power at a Grid site. This power is provided by clusters of computers organized in farms and managed by software tools. The CE manages jobs submitted by users as well as interactions with the services of the Grid.

Working Node represents the physical site where the computation is actually performed. Code scripts can be used to configure the environment properly.

Storage Element is responsible for data keeping and data-access granting. Depending on the importance of data, it gets saved on two different media: Tapes and Disks. The former guarantees a long-term secure storage, whereas the latter

is more indicated for data that will be frequently requested for analyses. The protocol SRM (Storage Resource Manager) offers a common interface to access data remotely. Main types of stored data include e.g. raw data from the detector, data produced by users' analyses and Monte Carlo simulations.

User Interface is the machine on which a user interacts with the Grid; through the UI, any user can access remote computing resources.

Central Services help the user access computing resources. Some examples are data catalogues, information systems, workload management systems and data transfer solutions.

As far as LHC computing is concerned, these solutions are offered by the WLCG project.

2.1 The WLCG project

The WLCG (Worldwide LHC Computing Grid) is a global collaboration counting more than 170 infrastructures distributed over 42 countries, linking up national and international grid infrastructures. Leaning on Grid projects as EGI¹ and OSG², the WLCG mission is to provide a common middleware on top of which each experiment can load its own applications and run them on computing facilities, these latter providing both computational power (e.g. CPU Cores) and storage supports.

¹European Grid Infrastructure

²Open Science Grid

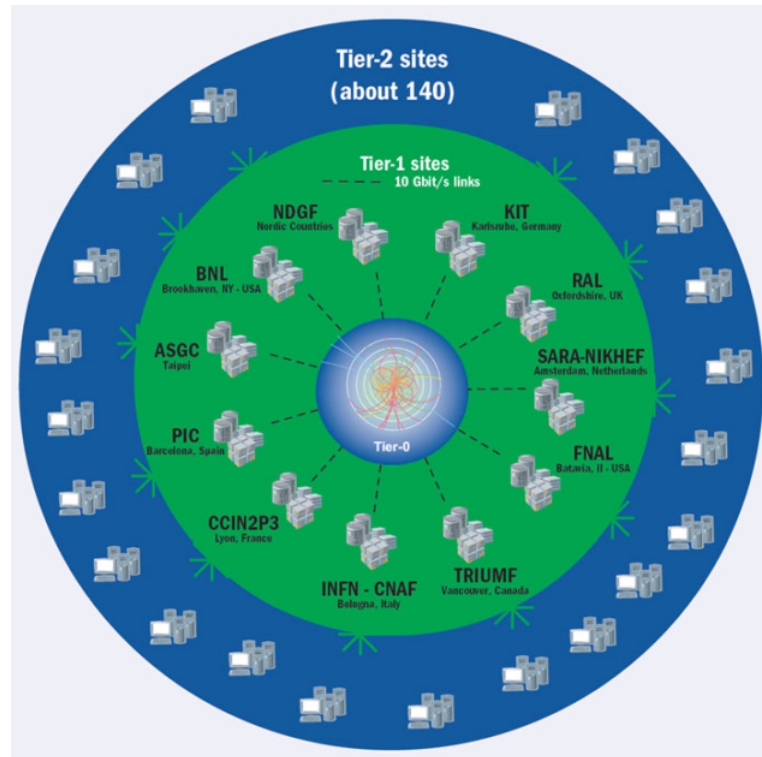


Figure 2.1: *Pictorial representation of the Tier Hierarchy with focus on computing centres and respective locations.*

The whole WLCG infrastructure can be reduced to a four-step hierarchy [10] of highly specialised computing centers (called “Tiers”). In order of duty-importance and tasks each Tier-level performs:

Tier-0

- Collection of RAW data for a first reconstruction.
- Distribution of RAW data and reconstruction output to Tier-1.
- Data reprocessing when LHC is not acquiring new data.

Tier-1

- Large-scale centralized data re-processing.

- Storage for RAW and RECO data.
- Storage for a fraction of simulated Tier-2 data.

Tier-2

- Data transfer from/to Tier-1
- Distributed data analysis.
- Monte Carlo Simulations.

Tier-3

- Small flexible computing resource.

despite not rigidly defined in all LHC experiments at the same level, both a small computing farm or even a personal laptop might act as Tier-3. Nonetheless, they represent ultimately a good resource for the local community of physics end-users and analysts

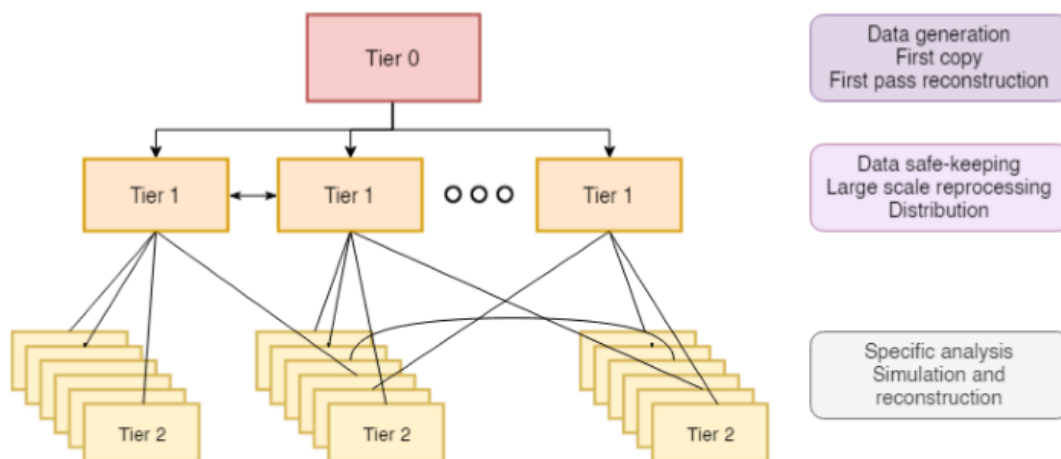


Figure 2.2: Pictorial representation of the Tier Hierarchy with focus on inter-facility relations as defined by the MONARC project.

For the sake of completeness, it is worth pointing out that although their importance is growing nowadays, Tier-3 facilities have not been formally included in the Memorandum Of Understanding (MoU) of WLCG (see Figure 2.2); as a consequence, unlike other Tiers, their availability is not checked on a regular basis and their reliability may thus not be stable in time.

2.2 The INFN-CNAF Tier-1 Facility

Of the 13 LHC Tier-1 facilities spread around the globe, one of them refers to the National Institute for Nuclear Physics (INFN), the Italian research agency for nuclear and subnuclear physics related activities. In particular, the Tier-1 is bound to the National Centre of Research and Development in Information Technology [11] (INFN-CNAF), an agency that has participated in the development of a common middleware ever since the foundation of WLCG project and that, since 2003, hosts the Italian Tier-1 facility.

CNAF Tier-1 operations can be summed up into three main groups:

Farming accounts for all computational services of the Tier.

The computing farm infrastructure acts as the service underlying the workload management system of all experiments connected with the processing resources at CNAF, allowing jobs to directly access data at CNAF, since the disk storage system is organized in file-system mounted straight up the Worker Node.

Each experiment has at least a dedicated queue, and the computing resources are centrally managed by a unique batch system. On average, a total of the order of 100k batch jobs are executed every day at CNAF, with resources availability of 24/7.

Storage for both storage itself and data transfer services. As foresaid at the beginning of the chapter, tapes are devoted to data that needs to be long-term secured, whereas disks, mostly SATAs, are used to store frequently accessed data. Few SSDs are implemented for those applications that require I/O intensive operations.

The storage is managed by StoRM [14], a Storage Resource Manager solution developed at CNAF, based on a home-made integration of IBM General Parallel File System (GPFS) [12] for disk access with the IBM Tivoli Storage Manager (TSM) [13] for tapes.

The amount of data currently stored and being processed at CNAF is in the order of tens of Petabyte, with a breakdown - at the time of this thesis - of about 23 PB of data on disk, and 48 PB of data on tape. This is expected to grow massively in the following years [11].

Networking accounts for network connection and security. Two main networks must be distinguished: WAN (Wide Area Network) and LAN (Local Area Network). WAN is provided by GARR (Italian National Research and Educational Network) and is the network CNAF connects to with various levels of redundancy and performance. LAN connections are based upon the Ethernet standard and empowered with a specific design to attend the requirements of the centre.

The requirements of heavy CPU-intensive data analyses led to providing a data access bandwidth of the order of hundreds Gigabit per second, and to implementing a Datacenter Interconnection with CINECA at a rate of 400 Gbps. [11]

In order to improve the quality of CNAF middleware in anticipation of the computational challenge brought by HL-LHC, the INFN-CNAF is endowing Tier-1 operations with modern data-driven diagnostic infrastructures, the goal being to monitor and support the processing efficiency.

This Thesis takes part to this major effort developing a prototype of “intelligent” maintenance-oriented information extraction from Log Files.

Next chapter will discuss the meaning and impact of “Maintenance” techniques in the data centre domain and discuss further the log files structure.

Chapter 3

Maintenance paradigm at INFN-CNAF

As previously described in chapter 2, the main CNAF's guideline is to upgrade the computational and storage resources to attend the workload jump. Thereby, there is a strong necessity to create a service infrastructure in order to provide system monitoring and system maintenance supports. In the following, an overview of System Maintenances (in the context of computing science) is presented; it is, thereafter, presented also the CNAF's specific use case.

3.1 Definition of “System Maintenance”

To define “System Maintenance” in the computational services context, it is necessary to answer the following question:

What defines a service as “properly working”?

First of all, the premise of any service is to perform tasks. Based on it, a mandatory system's characteristic is “Reliability”, meaning the system always performs as correctly as expected. Secondly, any service needs to perform the aforementioned task with (at least) minimum acceptable performance. Hence, another critical point is “Efficiency”.

In the context of this thesis, “System Maintenance” is defined as:

Definition 3.1.1. (*System Maintenance*) “*System Maintenance*” defines the combination of all actions, required in the life span of a system, meant to keep it efficient and reliable during its lifetime [15]

There is a huge variety of criteria that can be used to address the problem. The choice is mostly based on cost: that is, the computational complexity of algorithms and resources consumption at run-time.

3.2 Maintenance criteria

To distinguish maintenance processes following the above indicated factors, industrial standards have never been established. The absence of clear standards may generate some potential misunderstandings and make the most appropriate maintenance criterion adoption difficult.

Trojan et al. [16] propose a structure composed by four different approaches arranged in increasing complexity levels:

- *Reactive Maintenance,*
- *Preventive Maintenance,*
- *Predictive Maintenance,*
- *Advanced Maintenance.*

To highlight the differences among criteria, they are briefly described in the following.

3.2.1 Reactive Maintenance

Reactive Maintenance, also known as Breakdown Maintenance, refers to the ensemble of procedures deployed once the failure has already occurred, in order to restore the pristine behaviour. Since the procedure is done after the breakdown, this criterion can only aim to reduce to zero the impact of the rupture on the system [17].

As a classical example of this procedure, one can think of the mechanic intervention after a car engine break. His duty, in fact, is to allow the vehicle to get back to work as fast as possible.

The cost of this procedure is based on a loss control policy: *if* an intervention is needed, then its cost is accounted for, otherwise there is no cost.

The main disadvantage of this procedure depends on the intervention being performed once the breakdown has happened. This means, in first place, that any delay in the repairs impacts negatively the system. Then, as a side effect, human resources may be redirected to stop the performance leak, leading to a cascating effect on other sectors of the system.

3.2.2 Preventive Maintenance

Preventive Maintenance refers to the collection of procedures performed in order to lessen the likelihood of a system failure.

Preventive Maintenance actions involve both periodic overhauls of equipment, as well as mass-replacement of fault-prone parts (Planned Preventive Maintenance). It may represent, at times, a non-negligible and avoidable cost (e.g. this happens if a machinery gets substituted because of age, but it would, nonetheless, continue to work properly). Lately, thanks to the ever-growing branch of data-collecting technologies, Preventive Maintenance procedures can act in a more targeted manner since it is easier to evaluate deeply the wear of components using sensors and metadata analysis (Condition-based Maintenance), thus reducing the frequency of overhauls [18].



Figure 3.1: *CJ-130J Hercules propellers cleaning from salt and moisture is an example of preventive maintenance.*

In short: Preventive Maintenance techniques stem from the idea of achieving the “nothing breaks down” standard, whereas reactive methods are activated once the rupture has happened. As a consequence, downtimes of the system, in a preventive context, can be limited to an intervention scheduling in the best case, which represents a good improvement with respect to reactive paradigms. On the other hand, as a drawback, preventive methods tend to happen in overly expensive bursts, often concomitant with major replacements.

3.2.3 Predictive Maintenance

The natural evolution of Preventive methods is Predictive Maintenance, which is designed to determine the status of running services and predict events of interest as soon as possible on them.

Predictive Maintenance is achieved complementing diagnostic data of Preventive methods with Analytics based on historical trends and/or recurrent patterns. The continuous system monitoring and prediction maintenance helps preventing rupture events, thus decreasing the chance of breakdowns [19].

In the following, examples of effective Predictive Maintenance application are given:

Predictive Maintenance examples

- [20] presents disk-sensor based methods to prevent HDD failures.

- [21] presents a Time-Series based method to forecast the computational burden over the Wikimedia Grid.
- [22] shows a method to prevent catastrophic data centre failures.

the main drawback of the predictive criterion is represented by the need of specialists to analyse results as well as the high computational complexity.

3.2.4 Advanced Maintenance

This approach sums up the whole collection of modern paradigms that stem from the idea of both forecasting failures and diagnose their origins (that's why they are called "Advanced"). Thanks to the ever-growing branches of Machine Learning and Big Data Analytics, Advanced procedures can continuously improve the quality of their predictions and actions, thus reaching (ideally) maximum efficiency.

With respect to the predictive criterion, this collection of AI-engined methods is able to understand *when* and *why* failures will occur, thus both supporting (with dashboards and alerts see Figure 3.2) decisions and taking actions. For instance, while a predictive method would only point a potential failure, leaving to a human operator the final decision, an Advanced one would also start a supportive pipeline when needed. As a consequence, Advanced methods would represent a huge improvement in any maintenance approach, reducing human intervention and decreasing downtimes, with respect to previous criteria, even more. As a drawback, the Advanced infrastructure development and deployment is clearly time-consuming and requires state-of-the-art methods to perform properly.

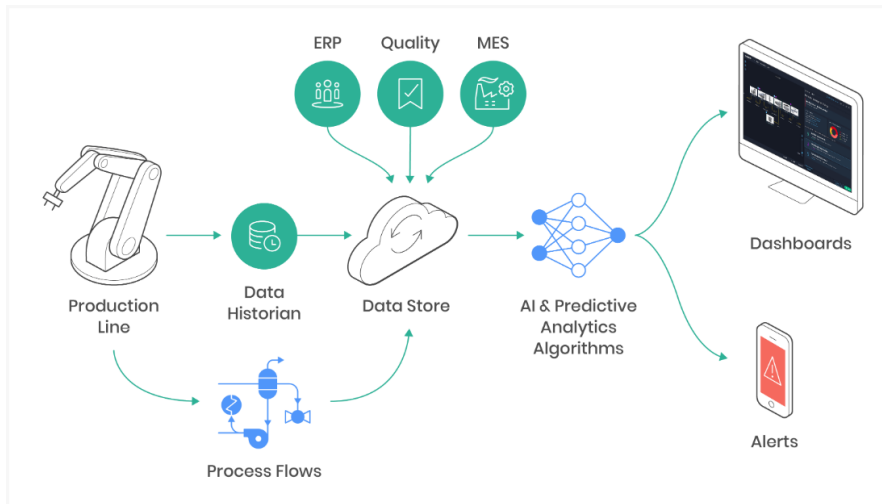


Figure 3.2: *Predictive/Advanced maintenance infographic.*

Based on previous overview, it is worth focusing on the main research of CNAF aiming to improve its Quality of Service and getting ready for HL-LHC necessities.

3.3 Log-Based Maintenance

Most of the machines at CNAF is continuously in order to extract useful metrics (e.g, Computational Load average, CPU-consumption, number of requests to specific service etc.) and create updated relational databases. In addition, every machine/service produces one/multiple log files containing typically textual data, such as the type of request tackled and its state, the Bash-command used and so on.

Log-files are often non-structured data and machine/service-specific, which makes their collection and processing quite difficult. Moreover, even when the type of log files seems to follow a “common standard”, its generation may change depending on who has written that log file.

Currently, CNAF’s data collection relies upon the Rocket-fast system for Log processing (Rsyslog [24]), an open source software, based on syslog, which is a common standard for logging and forwarding messages in a network. At the same time, to complete the information available, various machines’ metrics

are stored in a database called InfluxDB, which is a Time-Series database used to store large amount of timestamped data in several tables, one for each use-case metric [23]. Every database table is indexed by timestamps and contains columns corresponding to metrics of a given machine hostname. Various tags can be used to group different machines of the same service or experiment.

3.3.1 The Log Bucket

In April 2019, CNAF started developing a single repository (“bucket”) for the data centre, aiming to providing a storage-safe location and a simpler access point to log data for analysts, in view of future improvement to the maintenance process. Nowadays, Data is collected using Rsyslog¹ [24] to redirect the flow of log files to a common folder destination as described in figure 3.3:

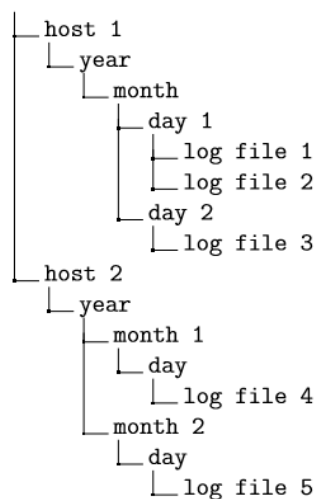


Figure 3.3: *Structure of Rsyslog collective bucket.*

Rsyslog offers high-performance, great security features and a modular design as being able to accept inputs from a wide variety of sources, transform them, and output to the results to diverse destinations. Rsyslog can deliver over one million messages per second to local destinations when limited processing is applied.

¹Rocket-fast System for log-processing

Log files are accessible via Network File System (NFS), allowing the access to files via network with performance similar to local operations.

In the following, the StoRM service, whose logs were studied by this thesis prototype, will be quickly described.

3.3.2 StoRM Log Files

StoRM [14] is the storage management service adopted by the INFN-CNAF Tier-1 and developed in the context of WLCG project with the aim of providing high performing file systems to manage storage capabilities distribution. In particular, it is used by HEP experiments, including e.g. ATLAS. With each experiment implementing differently the StoRM service. Its performance relies on the Storage Resource Manager (SRM) protocol, that divides operations in synchronous and asynchronous, where the former indicates typically Namespace operations (srmls, srmMkdir and so on), while the latter indicates specific functionalities, such as srmPrepareToPut, srmPrepareToGet and so on. Its structure can be split into two main stateless components: Front-End (FE) and Back-End (BE). To give an idea of the background of StoRM, a simple StoRM service schema is proposed in figure 3.4.

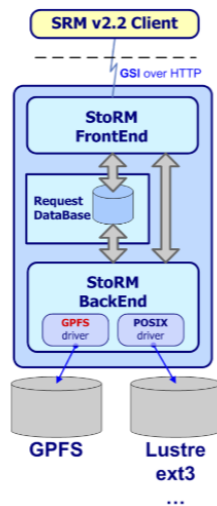


Figure 3.4: *Schema of StoRM with one FE and one BE component.*

In the following, the case of the ATLAS experiment is chosen for a first exploration. There is no specific reason for choosing ATLAS over other experiments: the pipeline, in fact, was designed to be as flexible and “experiment-independent” as possible.

Next paragraphs shed some light on the nature of logs coming from FE and BE.

3.3.3 StoRM Front-End

This part of the StoRM structure provides the final user the access to the SRM web service interface, manages user credentials and authentication processes, stores SRM request data and retrieves the status of ongoing requests.

Whenever a new SRM request has to be managed, FE logs a new line associated to that specific request, including the token that links the request to BE operations. In order to manage the wide amount of requests², 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 such log file is shown in Figure 3.5:

```
1543979711.194,2018-12-05 03:15:11.194000,38,INFO,573e4750-0ec1-430a-8d31-8a8e55ec2b81,Result for request 'BOL status' is 'SRM_REQUEST_INPROGRESS'
1543979711.209,2018-12-05 03:15:11.209000,62,INFO,e3ef6b54-3848-486c-924a-082a821a713e,Result for request 'Release files' is 'SRM_SUCCESS'
1543979712.381,2018-12-05 03:15:12.381000,14,INFO,fe037e9b-0691-4fd1-8bd7-9ba294fb8dc5,process_request : Connection from ::ffff:192.12.15.92
```

Figure 3.5: *Sample of log messages structure coming from StoRm-FE*

each lines contains:

- *Datetime*
- *Thread* that manages the request
- *Type* of message (INFO, WARN, DEBUG, ERROR, NONE)
- *Request-ID*
- *content of the message*

²order of few dozens per second on the average day

3.3.4 StoRM Back-End

Since it runs all synchronous and asynchronous SRM functionalities (requests for files, memory spaces and so on) and allows interaction with other Grid elements, the Back-End can be considered the core of StoRM service. Typical BE log files (named *storm-backend.log*) entries include the operator that has requested the action(DN), the involved files locations (SURL) and the result of the operation. Moreover, the verbosity level of BE log files relies on the log-back framework, that allows it to be set up appropriately with respect to the use-case. In conclusion, BE logging activity includes also a *storm-backend-metrics.log* (containing type, number of operations in the last minute, operations from start up and average duration of operations) and a *heartbeat.log* file that contains information on the number requests processed by the system from its startup, adding new information at each “beat”. An example of log messages contained in a *storm-backend.log* is shown in figure 3.6:

```
1559087993.135,2019-05-28 23:59:53.135000,
INFO,xmlrpc-954234,srmls: user </DC=org/DC=open-sciencegrid/OU=Open Science Grid/OU=Services/CN=dynamo.mit.edu>
Request for [SURL: [srm://storm-fe-cms.cr.cnaf.infn.it/cmsdisk/store/data/Run2018E/Commissioning/RAW/v1/000/325/511/]]
successfully done with: [status: SRM_SUCCESS: All requests successfully completed]
1559087993.169,2019-05-28 23:59:53.169000,
INFO,xmlrpc-954192,srmls: user </DC=org/DC=open-sciencegrid/OU=Open Science Grid/OU=Services/CN=dynamo.mit.edu>
Request for [SURL: [srm://storm-fe-cms.cr.cnaf.infn.it/cmsdisk/store/data/Run2016H/ZeroBiasPixelHVScan3/MINIAD/PromptReco-v2/000/283/553/00000]]
successfully done with: [status: SRM_SUCCESS: All requests successfully completed]
1559087993.197,2019-05-28 23:59:53.197000,
INFO,xmlrpc-954233,srmls: user </DC=org/DC=open-sciencegrid/OU=Open Science Grid/OU=Services/CN=dynamo.mit.edu>
Request for [SURL: [srm://storm-fe-cms.cr.cnaf.infn.it/cmsdisk/store/data/Run2016H/ZeroBiasPixelHVScan3/MINIAD/PromptReco-v2/000/283/551/00000]]
successfully done with: [status: SRM_SUCCESS: All requests successfully completed]
```

Figure 3.6: *Sample of log messages structure coming from StoRM-BE*

3.4 Aim of this work

As aforementioned, CNAF is implementing an “Intelligent” Maintenance on its site, aiming to both improve present-day performance and prepare for the upcoming HL-LHC challenge.

Currently, most of the CNAF efforts are focused on the log files produced by services running at CNAF Tier-1, with particular attention, since April 2019, to the StoRM storage service, whose log files were made available as expendable study-cases.

This work, continuing the recent research line [3], [42], develops an original software prototype, endowed with a modular approach, for Anomaly Detection on log files data. The goal of this software is ultimately to find, with good performance, interesting insights about the computational behaviour of the service, with the intention of paving the way for further technical studies as well as future development of a full Predictive Maintenance paradigm at CNAF Tier-1.

3.4.1 Code formal annotations

The analysis was performed using the most recent version of Python, namely 3.7, as well as the most updated packages, for optimal code efficiency and support. The code is distributed under GNU GPLv2 licence and is freely available on <https://github.com/FrancescoMinarini>.

The code was tested over two different “retail” machines:

- HP Spectre x360 2018 Intel i5 8th-gen 4-Core 8-Thread CPU, 8 GB RAM, Windows 10 + Windows Subsystem for Linux.
- Asus K501UX 2015 intel i7 6th-gen 2-Core 4-Thread CPU, 12 GB RAM, Ubuntu 19.04

The execution of the software requires less than a single hour when deployed on the average log file. This is a good result in comparison with previous results, that scaled up to 8 hours [42]. As of this thesis, the correct functioning of the

code is verified on both architectures.

Next chapter will highlight the theoretical background of the prototype that was developed, explaining the *ansatz* on which work was built on and reporting essential aspects of the algorithms involved in the structure of the prototype.

Next chapter will present essential concepts of the developed prototype, explaining the supposition on which work was built on and highlighting useful remarks of the algorithms involved.

Chapter 4

Methodology description

4.1 System’s behaviour extraction.

Before building a Maintenance approach, it is necessary to characterise the system behaviour, in order to identify aspects of interest related to a given phenomenon and ground the analysis on them. As previously outlined, log files are generated according to a specific action on the system. That is, it is possible to extract information regards to its behaviour analysing its logging activity.

The required “logging activity” definition, in the context of this thesis, is presented:

Definition 4.1.1. (*logging activity*) is the writing of a line on a corresponding log file according to a given service

Given this, the following supposition is assumed:

Any running service undergoing a potentially irregular workload tends to express its execution with a peculiar shape of its logging activity: this may happen either in form of activity drops/uprisings or plateaux-like shapes.

Following this line of thought, a naive method to deal with the identification of interesting behaviour phases, such as anomalous ones, would be simply extracting the amount of log entries the service produced in a certain time-window and (visually) isolate criticalities.

This approach, conversely, results fruitful only in case of simple detections; that is, by-eye-detectable steep drops/uprisings and major trends. In other more subtle cases, this approach will inevitably misread, if not ignore totally, the whole variety of interesting phenomena, such as, for instance, anomaly precursors clusters and deeper patterns (see Appendix B).

4.1.1 Log Volatility

The method is based on an observable capable of intercepting changes in logging activity and highlight them. A promising candidate for this role is a well-known and extensively used metric belonging to the “Econophysics” domain, where it is used to detect swings in Stock Market data: the “Volatility” [25].

Definition 4.1.2. (*Volatility*) *Volatility is a measure of change through time that quantifies behaviour swings of a determined physical quantity: Higher Volatility values indicate lack of stability.*

Mathematically, it’s the standard deviation of the physical quantity over a time window and it is bounded to the interval $[0, +\infty]$

The metric behaviour is quite simple: it peaks whenever a consistent change happens in the dataset. It stays low if data is not appreciably changing over time. Figure 4.1 helps grasping some intuition about the Volatility.



Figure 4.1: *Simplified representation of Volatility (green line) when evaluated on a ramp-up line (top) and on a ramp-down one (bottom).*

As shown in Figure 4.1, Volatility cannot distinguish between “Ups-and-Downs”, which is not to be seen as a defect: for our purpose, in fact, the tool has to

highlight just the changes of behaviour.

Once Volatility data is produced from log-activity, it is reasonable to think that potentially critical states of the system would be expressed in form of anomalous volatility points with respect to the other collected ones. At the same time, time windows could be marked as “anomalous” depending on the persistence of anomalies (see chapter 5 for details). Moreover, The Volatility, as it was defined, is extremely specialised in capturing changes in trends, which makes it capable of capturing potential precursors and similar phenomena. Therefore, it is required the deployment of a pipeline able to determine whether a point should be considered as regular or not is required. Luckily enough, this task can take advantage of a well developed branch of statistical unsupervised learning algorithms called “Anomaly Detection”.

4.2 Unsupervised Learning

Unsupervised Learning is defined as follows [26]:

Definition 4.2.1. (*Unsupervised Learning*) *Unsupervised learning is a type of “learning” whose goal is to find patterns without having any kind of pre-formed label to rely on.*

The application of an Unsupervised algorithm is often more challenging than a Supervised one for a quite simple reason: in a Supervised environment it is possible to *back-check* our work, assessing accuracy and interpretability at the same time. As a consequence, Unsupervised techniques are used for exploratory tasks, where finding patterns may be helpful for further right-on-target analyses. Since labeling data is a heavily time-consuming practice in pattern recognition, unsupervised learning is widely used as problem solving resource, Unsupervised methods are nowadays growing their importance in the research scenario.

As an illustration of their successful diffusion, some possible application are:

- Medical research [27]: unsupervised methods are used to discover subgroups inside pathology classes or to segment tissues in PET images.

- Digital marketing [28]: by clustering customer information and web behaviour, one can recommend better services (“who bought this also purchased...”) and improve publicity visualisation across sites (“Put ads where customers click the most”).

4.3 Essential concepts of Anomaly Detection

Firstly, let’s define what is a “statistical anomaly” in the context of this thesis.

Definition 4.3.1. *Let $x_1 \dots x_n$ be some observation that follow a certain distribution F . A statistical anomaly is an observation that appears not to come from F . [29]*

For the sake of completeness, it is worth pointing out that an anomaly, in general, can happen in several ways. The most common three are listed in the following [30]:

Point-like anomaly: a data point may be anomalous if its value happens to be significantly too far from the rest of data. Figure 4.2 shows a simple case:

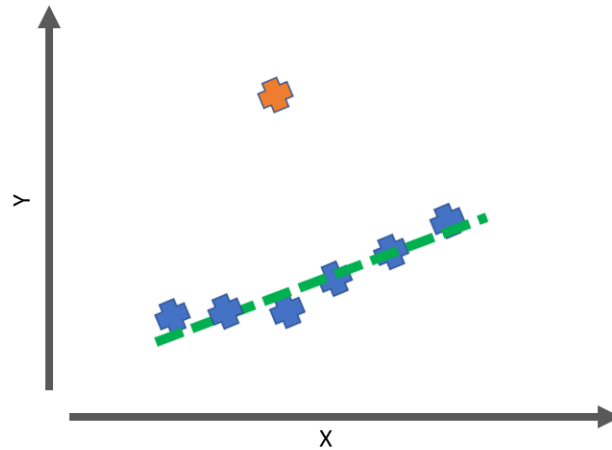


Figure 4.2: *Simple representation of a point-like anomaly.*

Contextual anomaly: The anomaly is context-specific. That is, a quality that generally does not, in general, imply an anomalous behaviour, but could be in a specific environment (see Figure 4.3).



Figure 4.3: *Simple representation of a contextual anomaly.*

Collective anomaly: In this case, the anomaly is not carried by the single value/behaviour but by the co-occurrence of multiple similar values/behaviours. Figure 4.4 shows an example taken from a personal project¹.

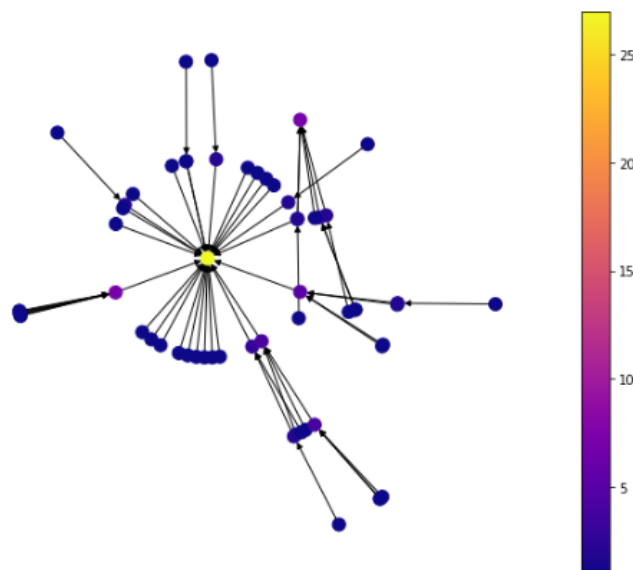


Figure 4.4: *Network reconstruction, color-coded in terms of degree centrality, of a fraud (Ponzi Scheme). Centralization of arrows is a case of collective anomaly.*

¹Personal didactic project for Complex Networks course

Definition 4.3.2. *Anomaly Detection (AD), also frequently referred to as “Outlier Detection (OD)”, consists in the identification of items, events or observations which raise statistical suspicions by being significantly different from the majority of data. [31]*

The discipline of detecting anomalies has a quite relevant importance since, by providing rich and easily interpretable actionable information about the nature of data points, it may lay the groundwork for a broad variety of applications, spacing from banking fraud detection to a contribution in scientific discoveries/validation of discoveries.

The typical AD algorithm consists in the following pipeline:

- Build a set (Profile) P of instances following a Gaussian distribution.
- Pass instances through P .
- Mark as “anomalous” every instance that does not statistically conform to P .

Throughout the years, this line of thought generated a plethora of viable algorithms based on well known statistical learning methods such as, for instance, clustering-based AD algorithms. However, although the availability of a deeply tested toolbox of algorithms constitutes an undoubtful advantage, most of what could be called “classical” approaches may suffer from at least two potentially insidious drawbacks, briefly discussed in the following.

Profiling vs Detection quality Trade-off: Since “classical” algorithms are optimized for profiling Gaussian -at least in good approximation- distributed instances, and find anomalies as a consequent step, some data might lead to obtaining too many *false-positives* and/or *false-negatives* that both ruin the accuracy of the model and make it harder to interpret the output of the detection. This frequently happens when one forces these methods over non-normal data.

Curse of Dimensionality: Since classical algorithms tend to have high requirements in terms of computational complexity when dimensions of the feature space rise up, as distribution densities have to be extracted, classical algorithms viability is constrained to lower-dimensional problems.

For this thesis work, an unsupervised method inspired by a well-known supervised algorithm was chosen: One-Class Support Vector Machines (OCSVM). In the following, basic intuition about Support Vector Machines (SVM) based on [32] are given.

4.4 Support Vector Machines

Let's take into account a classic 2-class linear classification task. The goal of our classifier is to structure a model $f(x)$ that will correctly assign the right class to each point . Its performance is typically achieved evaluating the "Empirical Risk" (that can be calculated for instance as the mean fraction of misclassified points over the total). [32]

Focusing exclusively on optimising the Empirical error may potentially lead to two drawbacks.

Firstly [32], one should take into account a huge amount of data points to ensure a proper convergence of Empirical Risk to the Expected Risk (not known a-priori). (Law of large numbers).

Secondly [32], an excessive optimisation of the Empirical Risk may assist the adoption of overfitting models; as to say: "models that over-train their decision function to exactly match the training data."

While the first observation may be dealt with with proper data collecting techniques, the second issue is clearly more insidious and subtle and may affect models without any evident symptom.

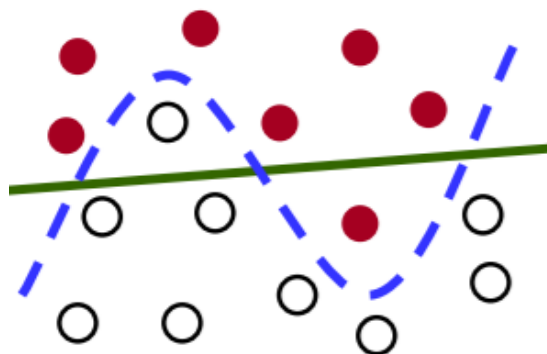


Figure 4.5: *Simplified representation of a classification task and a classification model.* [32]

The green line in Figure 4.5 above represents the adoption of a linear model. As one can see, although committing few errors, it seems able to capture the “spirit” of data. As a consequence, a good performance can be expected while generalizing to unseen examples. On the other hand, the dashed line seems to memorise the position of any single training data point. Hence, when generalizing, poor performance might be expected.

Figure 4.5 intuitively shows, hence, that a good performing model should strike the balance between:

- Complexity (in terms of mathematical refinement of the model function).
- Accuracy (in terms of mis-classification ratio).

To sum up: the best performing model tends frequently to be the simplest one (accordingly to the complexity of the task, of course). This Occam’s razor inspired intuition happens to be fully proved. In the following, the most interesting ideas of the theory (namely Vapnik-Chervonenkis theory) that allows to fully verify our intuitions will be showed.

Vapnik and Chervonenkis [33] translated the intuition of model complexity introducing the Vapnik-Chervonenkis (VC) Dimension.

Definition 4.4.1. (VC Dimension) *The VC Dimension \mathcal{H} of a function/class of functions is a measure of complexity and can be evaluated via the number of data samples that function/class of functions explains.*

further details of this measure are beyond the scope of this thesis. They can be found, however, in [32], [33]. Thereafter, it was proved that the Expected risk (not-known *a-priori*), given a model f , could be bound by the following expression named “*Structural risk*”:

$$R_{\text{expected}}(f) \leq R_{\text{empirical}}(f) + \underbrace{\sqrt{\frac{\mathcal{H} \cdot (\log(2N/\mathcal{H}) + 1) - \log(\epsilon/4)}{N}}}_{\text{VC confidence}} \quad (4.1)$$

where:

- N is the number of samples in the dataset
- \mathcal{H} is the VC Dimension

with constant ϵ small but different from zero.

the empirical risk has been already described, so focus is on interpreting the VC confidence. As can be seen, as the ratio of N/\mathcal{H} gets larger, the VC confidence tends to zero, which is consistent with the idea that for large values of N , empirical risk minimization is sufficient.

The idea of Vapnik and Chervonenkis is then to optimize this bound, which should lead to a balance between model complexity (VC Dimension) and Empirical risk. This direct approach, nonetheless, proved not to be good enough in a variety of typical situations, as equation 4.1 can lead to a non-linear optimization problem [32].

Cherkassky and Mulier [34] and Muller et al. [35] showed that, for a linear classification problem, the hyperplane leaving the highest margin between the classes led to the best results.

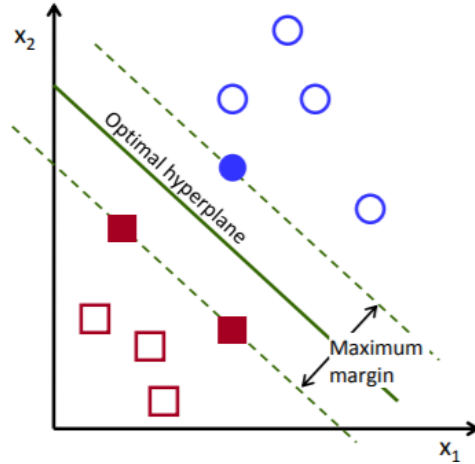


Figure 4.6: A simple pictorial representation of the maximum margin model. [32]

Vapnik et al [36], then, showed that the VC Dimension of a model could be bound by the following expression:

$$\mathcal{H} \leq \min\left(\frac{R^2}{m^2}, D\right) + 1 \quad (4.2)$$

Where:

- R is the radius of the sphere containing all examples.
- m is the margin left by the hyperplane.
- D is the dimension of the feature space.

Therefore, maximizing the margin means minimizing the VC Dimension. Moreover, as the Optimal hyperplane is a perfect separator (see Figure 4.6), maximizing the margin will also minimize the upper bound on the expected risk.

The whole problem now reduces to estimating the margin and maximize it. As for the margin, it is a simple trigonometry result (distance generical point x vs plane(w, b) in 3D space):

$$d(x, (w, b)) = \frac{|w^T x + b|}{\|w\|} \quad (4.3)$$

with freedom to choose the canonical hyperplane for simplicity: $|w^T x + b| = 1$.

Thus, the margin can be written as:

$$m = \frac{2}{\|w\|} \quad (4.4)$$

To maximize the margin, one can formulate the following optimization problem[32]:

- **minimize** $J(w) = \frac{1}{2}\|w\|^2$ where the quadratic function is chosen in order to avoid local minima.
- **subject to the condition:** $y_i(w^T x_i + b) \geq 1 \quad \forall i$

It is possible to find its solution via the Karush-Kuhn-Tucker conditions [37], [38]. (We will not delve into the mathematical details of the full proof as it exceeds the scope of this thesis). The outcome is nonetheless interesting as it states that for every point [32]:

$$\alpha_i [y_i(w^T x_i + b) - 1] = 0 \quad (4.5)$$

So, apart from the trivial case of $\alpha_i = 0$, one can see that data points (which are vectors in the feature space) should then lie over the two supporting hyperplanes that build the optimum margin, which has a deep consequence: the result of the optimization shows that the ideal hyperplane can be built focusing only on support vectors. A state-of-the-art classification can be performed by assessing correctly the margin and the support vectors, which allows SVMs to have low computational power absorption.

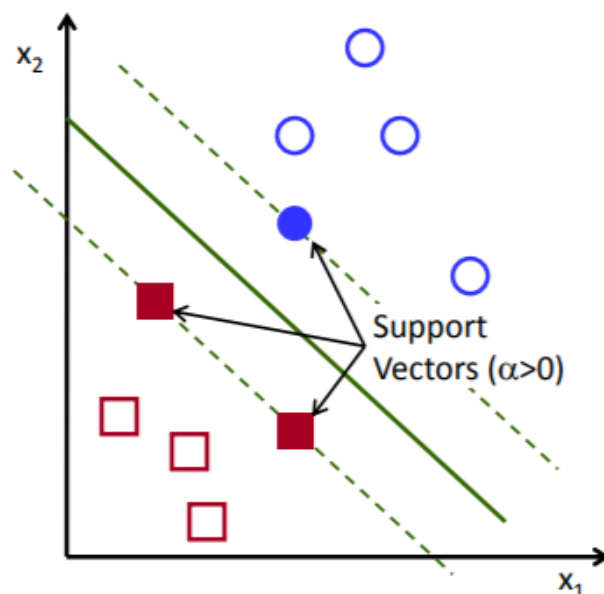


Figure 4.7: *Graphical example of Support Vectors.* [39]

By now SVMs, in the case of linearly separable classes, were discussed. Nonetheless, SVMs can be a viable choice also for non-linear problem, as they can take advantage of some interesting mathematical results it is worth noting in the following.

A well-known theorem of data analysis (Cover's Theorem [39]) suggests that hard learning problems (including non-linear ones) may find a simpler solution, typically linear, if properly cast to a higher (but still under control) dimensions. This operation, in the domain of SVMs, is called "Kernel trick", which consists in a mapping of points to a different domain performed through a set of functions (typically Radial Basis Functions (RBF), but it's not a mandatory choice) that allow a simpler approach to data.

Having developed some intuition about the class of Support Vector Machines, this thesis focuses now on seeing how their behaviour can be adapted to work with only one class in an unsupervised environment.

4.4.1 One-Class SVMs

In this thesis, the Schölkopf idea of One-Class SVM. A brief description is presented in the following.

In an AD paradigm, our data is split, ultimately, in two expected classes: regularities and anomalies. OCSVM, having to deal with an Unsupervised splitting, considers a mapping of data that separates data itself from the origin of the feature space. At this point, the OCSVM algorithm kernel-tricks input data into a higher dimensional feature space and iteratively finds the maximal margin hyperplane which best separates the training data from the origin.

In the following picture, the graphical result of a standard OCSVM application, taken from its official documentation, is showed in figure4.8 [40].

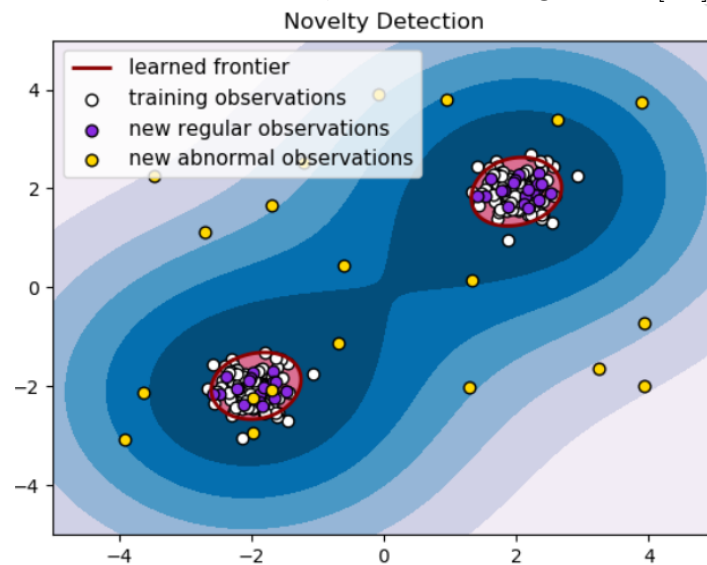


Figure 4.8: *Pictorial representation of the application of a OCSVM with RBF kernel trick.*

Since the prototype tries to take advantage of the implicit information contained inside log messages collected in log files, it is appropriate to describe the Natural Language Processing (NLP) method that was proposed as a mean of textual information extraction.

4.5 TFIDF Information Retrieval

Definition 4.5.1. (*Information Retrieval*) *Information Retrieval (IR) describes the ensemble of content-mining techniques involved in searching for specific information over a collection of textual documents and/or other medias.*

To mine out information about system failures and/or lags, the implementation of a IR technique able to reach out for those informations is thus required. Text-Frequency Inverse Document Frequency analysis (TFIDF) was then chosen.

Definition 4.5.2. (*TFIDF*) *TFIDF is a measure whose goal is highlighting the importance of a word/Ngram to a document in a collection/corpus of other documents. Its value increases proportionally to the number of times a word/Ngram appears in the corpus and inversely proportionally to the number of documents that contain the word.*

Definition 4.5.3. (*Ngram*) *An Ngram is a contiguous sequence of N items from a given sample of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application.*

For instance, splitting in 2-grams a simple sentence such as “*This font is black*” leads to:

- *This font*
- *font is*
- *is black.*

In the following, we will focus, for the sake of simplicity, on 1grams (words).

TFIDF value can be calculated from two separate terms: Term Frequency (TF) and Inverse Document Frequency (IDF).

TF extraction is really straight-forward:

$$TF_{i,j} = \frac{n_{i,j}}{N_j} \quad (4.6)$$

where:

- $n_{i,j}$ indicates the occurrence of item i in document j .
- N_j indicates the dimension of document j .

So, as an example, consider a document of 1000 words, where the word “physics” appears 32 times. Then, $TF_{physics} = 32/1000 = 0.032$.

IDF extraction is defined as:

$$IDF_{t,D} = \log \left[\frac{N}{1 + |d \in D : t \in d|} \right] \quad (4.7)$$

Where:

- N indicates the number of documents available in the collection.
- $1 + |d \in D : t \in d|$ indicates the number of documents where item t is readable. The +1 accounts for avoiding the risk of divide-by-zero errors.

So, for example, consider a collection on 1000 documents with the word “physics” appearing in 10 of them. Then $IDF_{physics,1000} \approx 2$.

Finally, TFIDF can be extracted by multiplying TF and IDF. To conclude the example, the TFIDF value for the word “physics” in our simple example is $TFIDF_{physics} = 2 \cdot 0.032 = 0.064$.

Figure 4.9 shows an example of how a generic output of TFIDF may appear.

As one can see, TFIDF calculation allows to change the encoding of our IR analysis from the context of pure language to a more numerical one. At the same time, evaluating words under an “importance” basis, allows us to mine out only the most interesting locutions, leaving out very common ones.

	Document 1	Document 2	Document 3	Document 4	Document 5	Document 6	Document 7	Document 8
Term(s) 1	10	0	1	0	0	0	0	2
Term(s) 2	0	2	0	0	0	18	0	2
Term(s) 3	0	0	0	0	0	0	0	2
Term(s) 4	6	0	0	4	6	0	0	0
Term(s) 5	0	0	0	0	0	0	0	2
Term(s) 6	0	0	1	0	0	1	0	0
Term(s) 7	0	1	8	0	0	0	0	0
Term(s) 8	0	0	0	0	0	3	0	0

← Word Vector
(Passage Vector)

Figure 4.9: *TFIDF matrix of a generical corpus of documents.*

The following picture shows how words would rank in a fictional TFIDF value-Occurrence space.

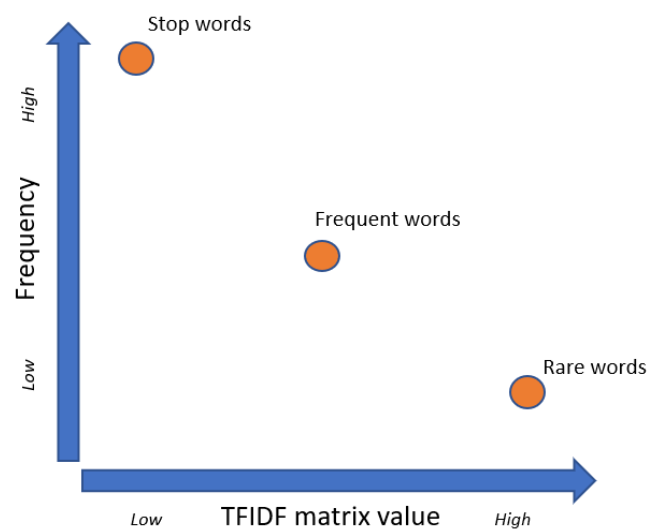


Figure 4.10: *Representation of words positioning in a TFIDFscore-Occurrence space .*

Following chapter will implement all the algorithms described in this one, discussing thoroughly any choice made to perform the analysis as well as each output step of the pipeline.

Chapter 5

Analysis results

In this chapter, the application of the prototype discussed in Chapter 4 to the available logging data from CNAF systems is presented. Some insights from such analysis are highlighted in detail.

5.1 Data preprocessing

Before proceeding with the analysis, the following operations of data whitening are performed:

- **Handling missing data.** Log data are treated by checking occurrences of “NaN values” and erroneous entries among all log messages in the dataset. This is performed as a primary step in data preparation.
- **Data re-sampling.** A rounding up of log message timestamps to the nearest 5 minutes is performed. The reason for this choice is related to the fact that the input log files may count up to 3-4 million messages per day, and they are collected with a granularity to the level of the millisecond, potentially leading to high load on computational resources in the analysis phase. Moreover, as outlined in Chapter 4, there is no interest in a point-like view of log files, focusing on each of the specific log entries, but more in overall changes of

behaviour of the service that produces such log files.

Several different time-windows were analysed: the only detected difference between each resulting outcome was the higher/lesser smoothing effect over patterns. Rounding up to 5 minutes of logging activity proved to yield a resampling of all messages in time windows without any loss of generality and richness of the information therein.

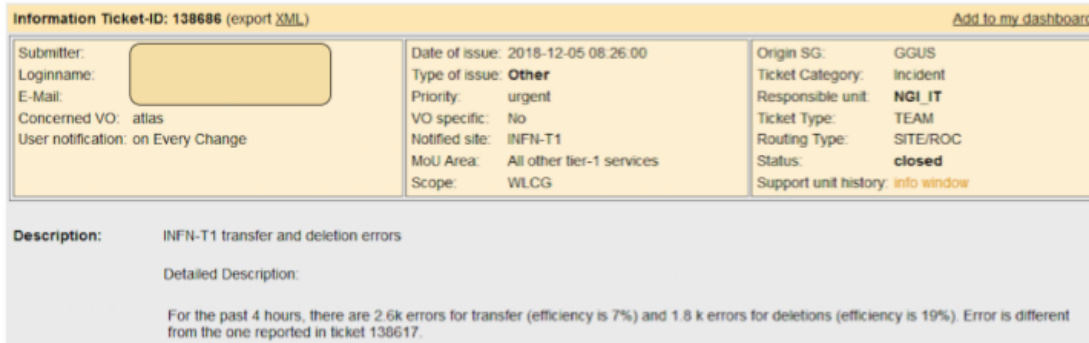
- **Data scaling.** The volatility value are scaled in the $[0,1]$ interval. This step is performed after the calculation of the volatility (see Chapter 4 for details) and before applying the unsupervised method. The motivation behind this step is to avoid a scale-factor bias towards higher volatility values.

5.2 Data analysis

The deployment of the prototype on StoRM Front-End data will be described in the next section, whereas the application on Back-End data will be described in Section 5.2.2).

5.2.1 Results on StoRM Front-End service

The input data under study has been collected in the time-period ranging from December 2nd to December 8th, 2018. The reason for this choice is related to the fact that a “ticket” (a notification of a technical problem by a user via official reporting systems, like GGUS [41] in this case) had been opened, thus indicating an anomaly whose cause was not understood, and assigned to the attention of the proper CNAF StoRM team.



Information Ticket-ID: 138686 (export XML)		Add to my dashboard	
Submitter:	<input type="text"/>	Date of issue:	2018-12-05 08:26:00
Loginname:	<input type="text"/>	Type of issue:	Other
E-Mail:	<input type="text"/>	Priority:	urgent
Concerned VO:	atlas	VO specific:	No
User notification:	on Every Change	Notified site:	INFN-T1
		MoU Area:	All other tier-1 services
		Scope:	WLCG
		Origin SG:	GGUS
		Ticket Category:	Incident
		Responsible unit:	NGI_IT
		Ticket Type:	TEAM
		Routing Type:	SITE/ROC
		Status:	closed
		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 5.1: A GGUS ticket reporting an anomaly in the behaviour of the StoRM Front-End service at INFN-CNAF (more details in the text). User information have been anonymized.

As explained in [3], the information in the ticket suggests to further investigate the days ranging from December 5th to December 8th and tag this as a “critical” time-window, while days ranging from December, 2nd to December 4th are tagged as a “regular” time window. Starting from these data, the prototype developed in this thesis is hence expected to be able to detect an anomaly in the time window tagged as “critical”.

The number of log entries (“log count” in the following) is extracted from the data on December 2nd - the regular sample - and from the data on December 7th - the critical sample. The log count for the regular sample is shown in Figure 5.2.

The comparison of 5.2 and 5.3 shows that the latter displays evident peaks in log count, twice as high as in the former. Due to the absence of evident peaks, the regular time window is harder to interpret, as dominated by various oscillations that may well not be so relevant at all. The identification of a proper observable able to spot changes of behaviours in log count emerges as a need, and such need has been addressed by using the Volatility variable. The Volatility was evaluated via a 6-entry rolling window sliding over log message counts. Due to the 5 minutes resampling of the timestamps, Volatility is hence evaluated over a 30 minutes time window. In 5.4 the Volatility for 4 reference days is shown, with box plot that help to highlight percentile compositions for each.



Figure 5.2: Log count from CNAF StoRM Front-End on a regular day (see text for details).



Figure 5.3: Log count from CNAF Storm Front-End on a critical day (see text for details).

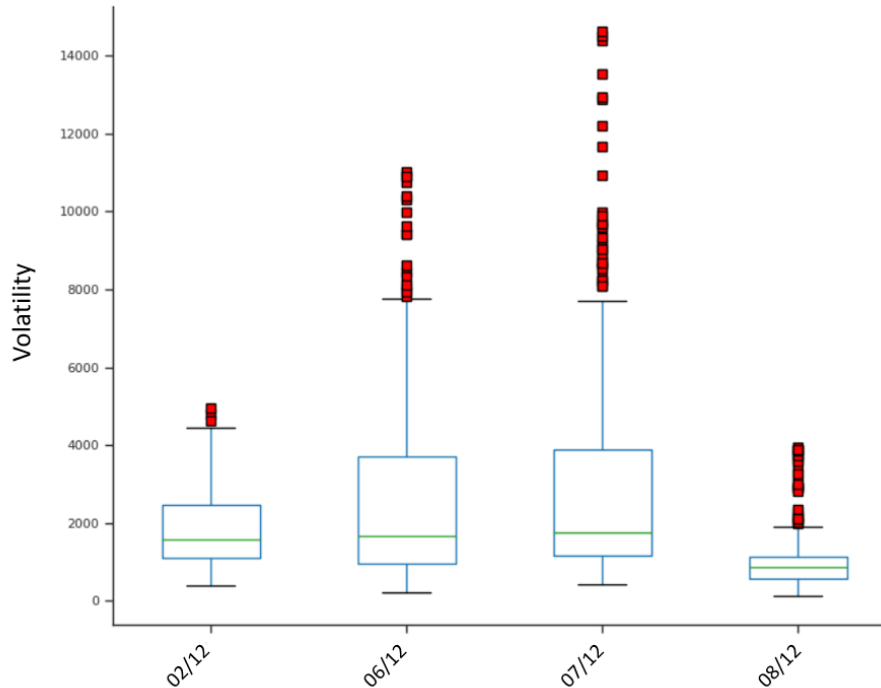


Figure 5.4: *Volatility on CNAF StoRM Front-End logs, on 4 reference days.*

On December, 2nd the Volatility presents just very few outliers, which is compatible with it being a “regular” day. On December, 6th-7th the Volatility shows a large number of outliers, which becomes even more evident on December, 7th; on this day, they reach the maximum value across all examined days, and also reveal a peculiar structure in one tail, compatible with a “critical” day. This shows that the Volatility is indeed a promising variable to disentangle between “regular” and “critical” days.

As a consequence, December, 2nd and December, 7th can be chosen as a reference for “regular” and “critical” days respectively. As such, the unsupervised anomaly detection algorithm described in detail in Chapter 4 has been applied to both, as discussed in the following.

The unsupervised algorithm (OCSVM) was implemented with RBF kernel and applied to this case to perform an anomaly detection task. Results are shown for

December, 2nd in Figure 5.5 and for December, 7th in Figure 5.6

In the top plot of Figures 5.5 and 5.6, the Volatility values are color-coded after the application of the algorithm, as follows: red dots represent detected anomalies, whereas green dots represent regularities.

On the “regular” day, the detected anomalies are a minority among the dots, and when they occur they persist for relatively short time intervals: most of the dots on that “regular” day - mainly visible in its first half - are green, thus indicating that that day was behaving regularly most of the time, apart from few and short oscillations. On the other hand, on a “critical” day, anomalies that occur tend to peak more and be more persistent over time. Another interesting observation is that there are quite some almost-zero Volatility intervals (one of which is very persistent): these might also be indication of a problem, e.g. a “frozen” service, thus being properly tagged by the algorithm as anomalies as well. A similar analysis was performed also on other days shown in Figures 6.1, 6.2 for which data was available and tagged as “regular” vs “critical”: results of these analyses lay along the lines of the aforementioned discussion, and the corresponding figures can be found in Appendix A. Despite the limited amount of tagged days on which to run the OCSVM method, the method turns out to be reasonably validated as a tool to spot anomalies in the log count volatility.

The application of the same prototype to StoRM Back-End logging data is described in the following section.

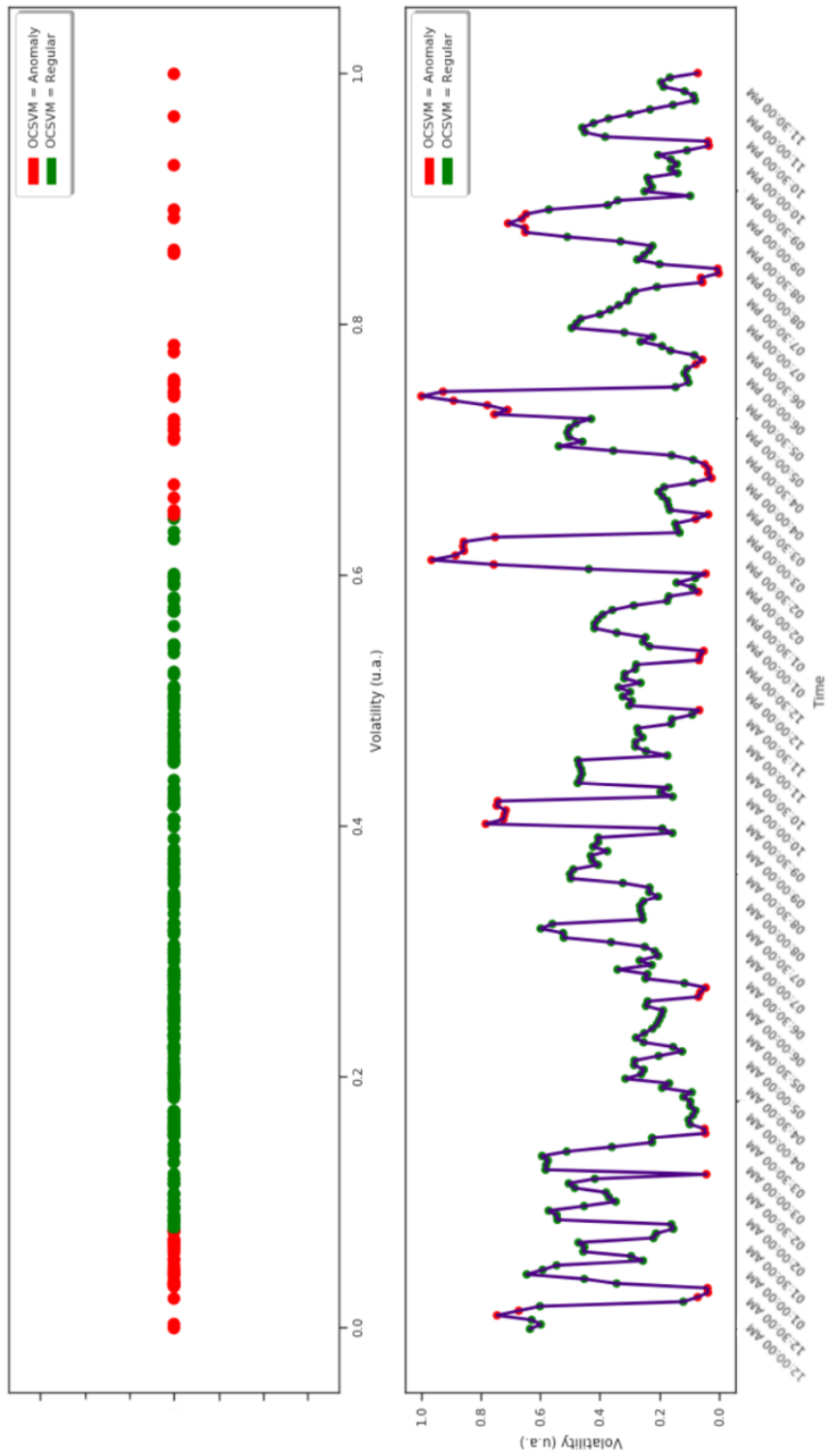


Figure 5.5: (top) Volatility values, color-coded depending on the label predicted by OCSVM for a “regular” day: red for anomalies and green for regularities. (bottom) Volatility unrolled as a time-series, with anomalies highlighted with the same colour codes.

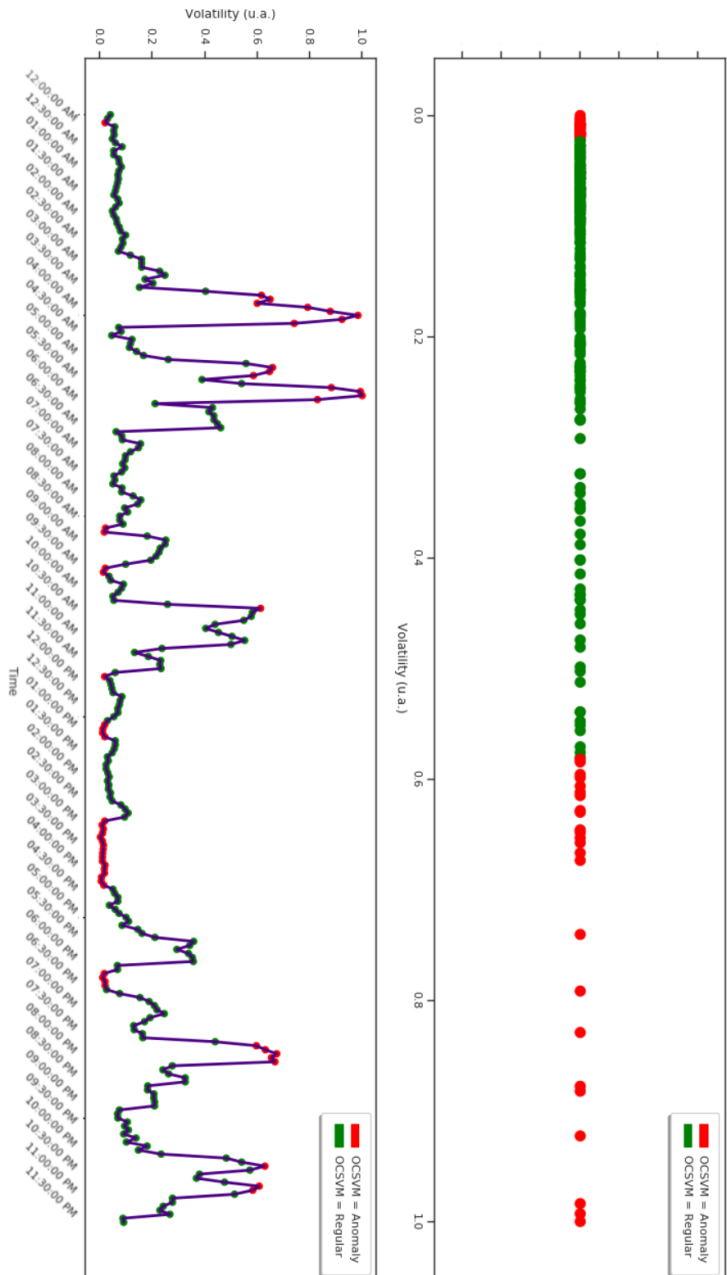


Figure 5.6: (top) Volatility values, color-coded depending on the label predicted by OCSVM for a “critical” day: red for anomalies and green for regularities. (bottom) Volatility unrolled as a time-series, with anomalies highlighted with the same colour codes.

5.2.2 Results on StoRM Back-End service

The Back-End analysis focuses on a specific day, i.e. May, 23rd 2019. This choice is motivated by previous analyses [42] that identified this as a potentially problematic day. While the CNAF operators could not offer expert insight on what actually might have happened on that specific day, evidence shows that the size of the StoRM Back-End log on that day was larger than other daily logs by a factor almost 3, as shown in Figure 5.7.

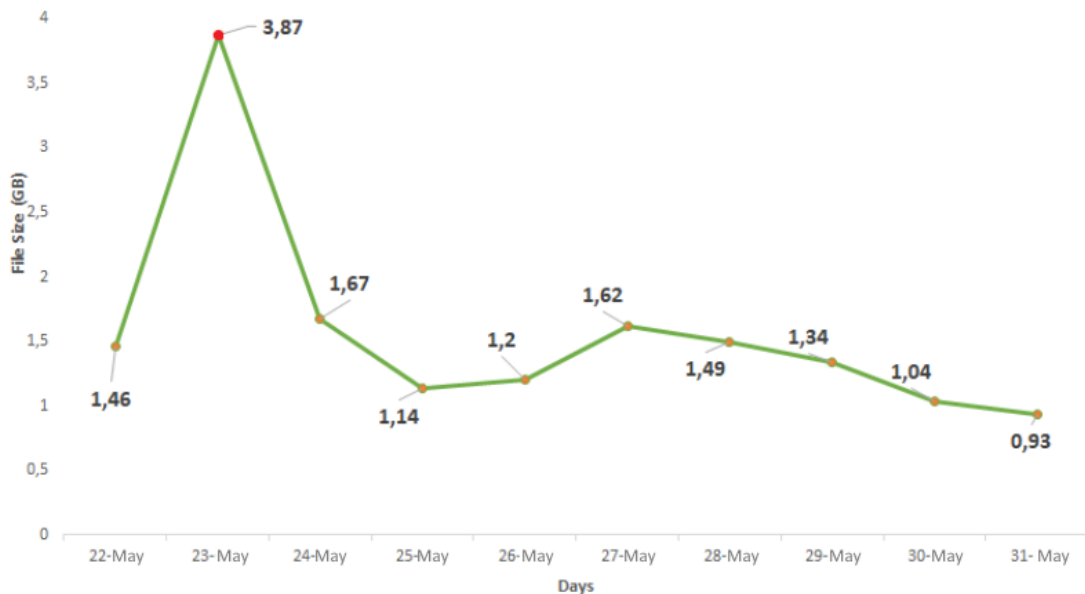


Figure 5.7: *Size of StoRM Front-End logs throughout the analysed period. May, 23rd 2019, the day discussed in the text, presents an anomaly.*

Despite the evident anomaly, this cannot be used to infer that this day was for sure a “critical” one for StoRM operations: indeed, the file size only depends on the amount, and not the kind, of messages produced and logged on the file itself. As an example, if many researchers would need a large set of files for a given data analysis at the very same time, this would appear in the logging as a much larger number of e.g. authorisation requests, SRM Prepare-ToGet entries, etc. Despite its load, if that amount gets correctly managed and carried out by StoRM services, one would end up with a huge log file for that day, while no major operation disruption or system malfunction asso-

ciated with it. Yet, May, 23rd stand as an interesting day to explore in more depth.

Following the same data preparation steps outlined before, the log count (i.e. the number of log messages) was extracted also for May, 23rd, and it is shown in Figure 5.8.

During morning hours, with the exception of few spikes and bumps, the activity seems to have an oscillatory behaviour. Between (indicatively) 3:00:00 PM and 4:30:00 PM there is a high peak of activity followed by a steep drop. In late evening, between (indicatively) 6:00:00 and 10:00:00 PM, activity bumps up again, despite never again at the same level as earlier in the day.

This, although not allowing any assumption on the nature of the behaviour of the StoRM service yet, it indicates few specific time windows that would benefit from a deeper analysis. As in the Front-End case study, the emerging need of a more quantitative observable able to spot changes is addressed by the Volatility. The Volatility for May 23rd, 2019 is shown in Figure 5.9.

It is hence of interest to compare Figure 5.8 (log count) with Figure 5.9 (Volatility). The morning activity in the Volatility plot actually shows few bumps that were somehow previously partially hidden. This confirms once again the usefulness of the Visibility as a variable to enhance the expressivity of log count information. Moreover, a trend of bumps, starting at (approximately) 12:00:00 PM and allegedly preempting the main peak has been very clearly highlighted in the Volatility plot, way more evidently (and less oscillating) than in the log count plot. Additionally, the large steep drop seen in the log count plot was also correctly detected in the Volatility plot, in terms of a very large peak. Its impact seems to be limited to the interval between 15:00:00 PM and 16:00:00 PM, an indication that comes in a clear manner only from the Volatility plot. The late evening bump, between (approximately) 06:00:00 PM and 10:00:00 PM, appears to be highlighted mostly in its ending part (between 21:30:00 PM and 22:00:00 PM) in the Volatility plot. In general, the Volatility plot offers a much better method to visualise changes

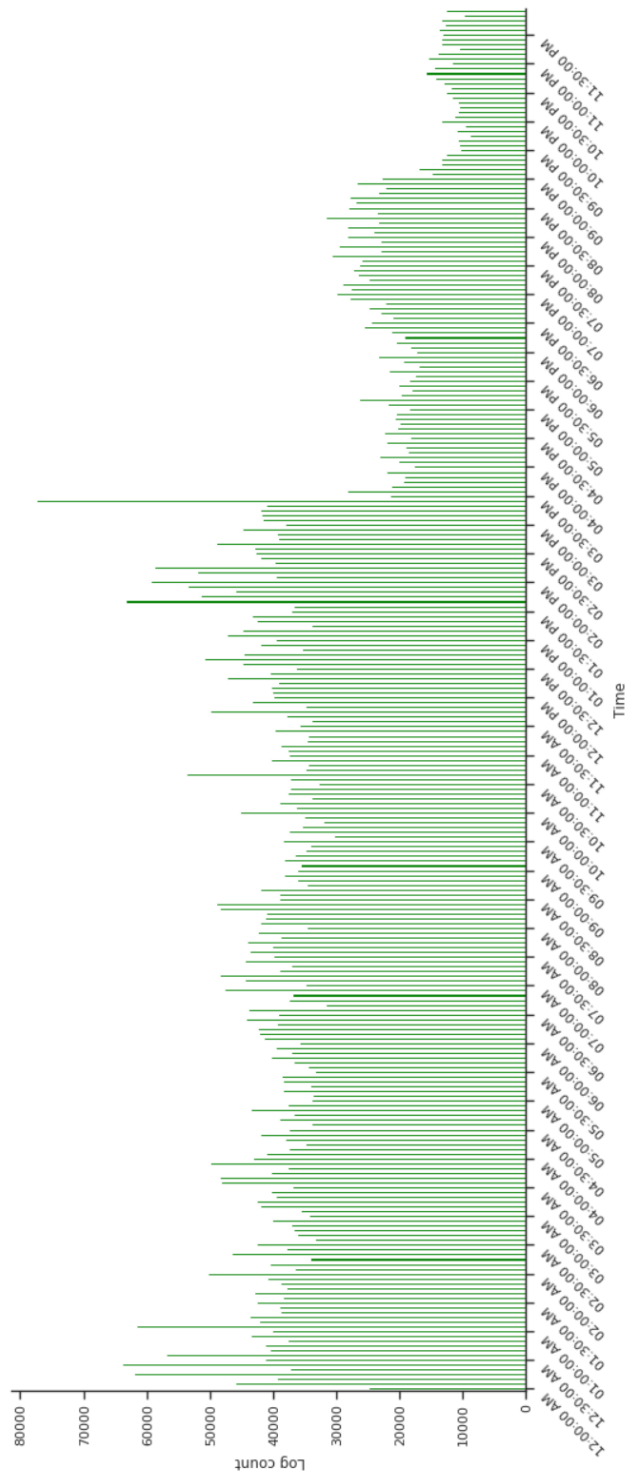


Figure 5.8: Log count from CNAF StORM Front-End on May 23rd, 2019 (critical day, see text for details),

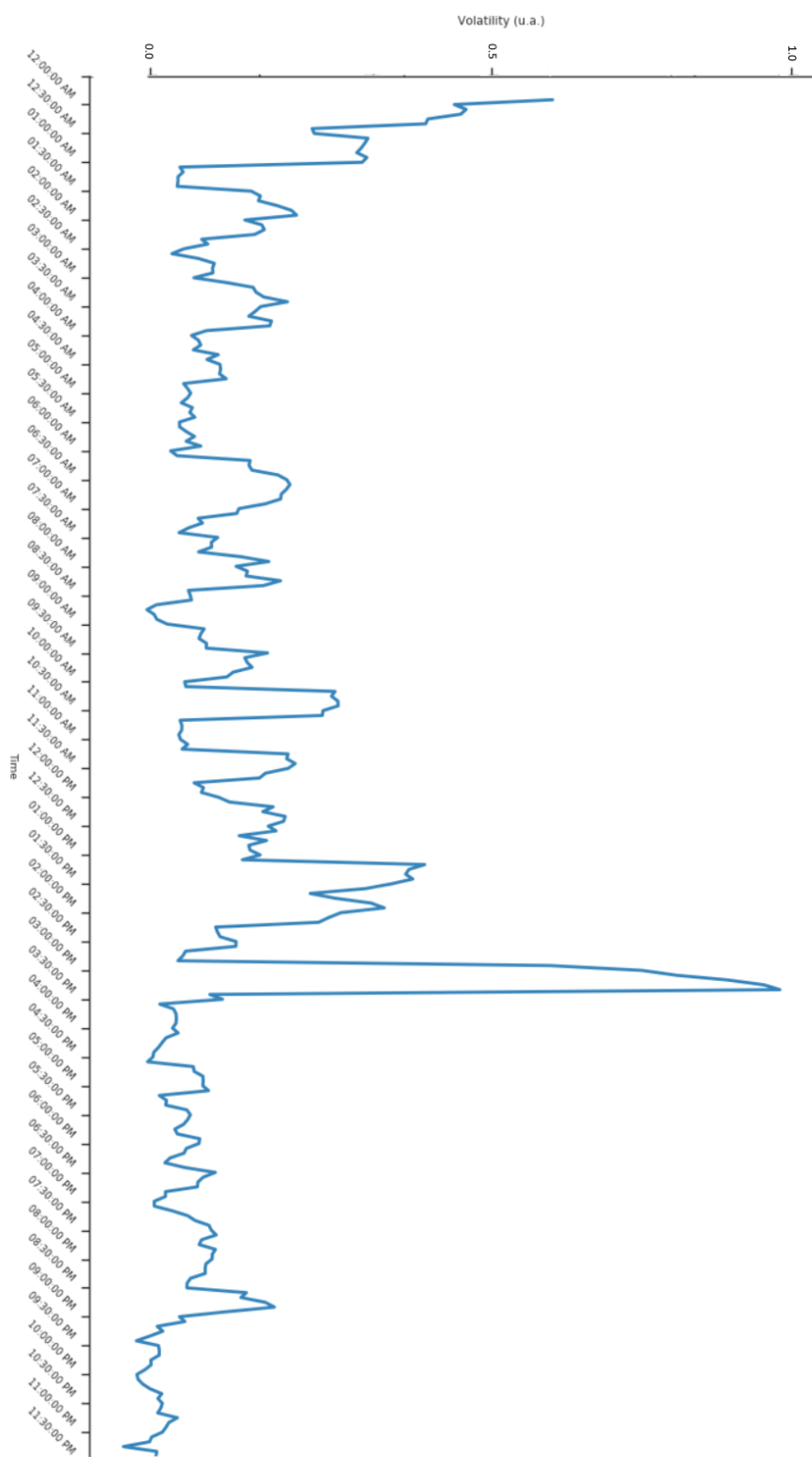


Figure 5.9: Volatility, unrolled as Time-Series, extracted from CNAF Storm Back-End log on May 23rd, 2019.

of behaviour (see also Appendix B) with respect to other methods, thus making its use an advantage with respect to other options. In fact, by highlighting very specific time windows, based on Volatility quantitative behaviour, the analysis can focus over a selected amount of messages to be analysed, instead of requiring the analysis of the entirety of the logs of that given day. On average, over the days analysed in this thesis, this reduces the burden of at least a factor of 10 for every day of logging, which in turn helps the application of any further CPU-intensive algorithm, i.e. Analytics tools would eventually perform faster. This brings up an aspect of actual effectiveness of this method in view of its integration in the overall CNAF strategy on predictive maintenance: the ability to run e.g. text processing CPU-consuming algorithms only on the segments of the log files that are identified as interesting by the anomaly detection algorithm. This is discussed in more depth in Section 5.3.

The OCSVM algorithm, as before, can also be applied to the logs from this particular day. The top plot of Figure 5.10 represents Volatility values color-coded after the application of the algorithm (as before, red points represent detected anomalies whereas green ones represent regularities). The plot shows that the algorithm highlights major peaks as well as almost-zero Volatility values.

Not surprisingly, the same method applied to this day produces results as in figure 5.10 that are interesting for various reasons.

Firstly, the time-series view as from the bottom plots of 5.6 and 5.10 may suggest that the distinction between red and green dots could be as effectively done by applying static threshold (e.g. Volatility greater than a given value). This would be an over-simplification: first of all, one would need 2 thresholds, one for high values and one for low values; additionally, 5.10 shows that these thresholds would not work as good, given there are periods in which red and green dots were not so easy to be predicted as such. In other words, a decision boundary here is not a trivial linear threshold (as one naively one could do via a purely visual inspection), instead it appears as an elliptic decision boundary. This underlines

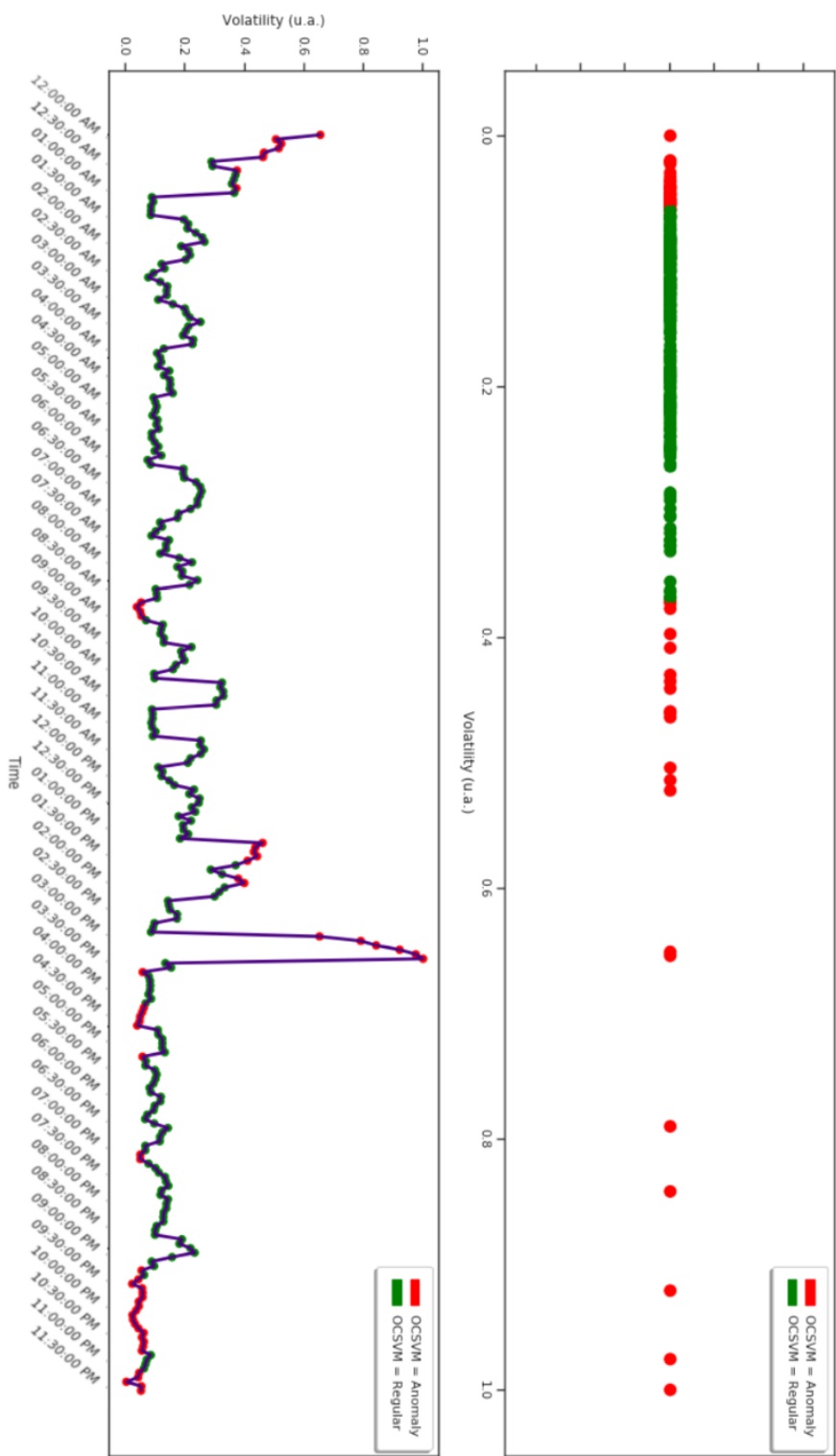


Figure 5.10: (top) Volatility values color-coded depending on the label predicted by OCSVM: red for anomalies and green for regularities. (bottom) Volatility points highlighted with the same colour code.

the importance of implementing an algorithm like this one, instead of focusing on a just visual identification of thresholds.

Secondly, the algorithm is good at identifying peaks and drops that deserve attention. The peaks highlighted by the algorithm are a subset of the peaks assumed as interesting in the previous step: this represents a good step forward in the analysis, as the indications get clearer and clearer. At the same time, the algorithm also highlights as anomalies very low values of Volatility that are stable for long time, resulting in a low-Volatility plateau: this might happen when the activity of the service may drop to zero (due to a major breakdown) or get stuck in a loop that produces regularly the same logging.

Thirdly, with respect to Figure 5.6, Figure 5.10 allows to reduce even further the number of log messages to be analyzed in order to extract textual information about what may have caused the issue. This is discussed in more depth in Section 5.3.

5.3 Content extraction through Text Processing

As a final component of the prototype developed in this thesis work, text processing techniques have been applied to the segments of the log messages identified as anomalies in order to extract content to be connected with possible sources of such anomaly. This step could in principle be implemented with sophisticated techniques from Natural Language Processing (NLP). This will be discussed in the conclusion remarks, as it goes beyond the scope of this thesis: still, an attempt towards this direction was tried out, and it is presented and briefly discussed in the following.

5.3.1 Methodology and preliminary results

Once the relevant time window(s) from a given daily log are identified by the OCSVM algorithm, the entire text in the log entries corresponding to the selected time window can be extracted and analysed. To do so, information retrieval was

done via TFIDF (see Chapter 4) and following steps are briefly listed in the following.

In this specific analysis, the focus is on May 23rd, 2019, precisely on the time range from 3:00:00 PM and 4:00:00 PM, i.e. log data corresponding to the peak in Figure 5.9

The text processing analysis follows this pipeline:

- **Anonymisation.** Proper filtering via regular expressions is applied in order to anonymise all user-related entries. These include e.g. IDs, e-mail contacts, name and surname as from the Grid certificate's Distinguished Name (DN), user affiliation, etc). All these details would be anyway under privacy regulations, but are anyway parts of the log text that are completely unnecessary to the test at stake.
- **TFIDF Word Tokenisation.** The text from log message is then tokenised in tuples of words with 2 (diads) or 3 (triads) items each. This steps could be done in more sophisticated ways, while this one is still a simple and efficient approach in order to grasp the “sense” of the message.
- **TFIDF Information Retrieval.** Tokenised messages were then used as collection of documents onto which TFIDF could be deployed. This step requires TFIDF fine-tuning setting, in particular *i)* automatic exclusion of English stop-words; *ii)* ignoring diads/triads with a frequency strictly greater to 80%.

The outcome of this entire process for the mentioned log file highlighted the presence of some most relevant key-tuples, among which for example:

- “*SRM-FAILURE REQUESTS FAILED*”;
- “*SURL SURL FAILED*”¹;
- “*FAILED STATUS SRM-FAILURE*”.

¹The repetition of “SURL” here is just a consequence of the regex filtering.

The SRM documentation (and the StoRM experts at CNAF) confirms that all these messages are related to operations expected to succeed but actually failing on the SRM interface, namely the first and the third are symptoms of a failed SRM request (e.g. prepare to access a file, actually access a file, prepare to write a file, actually write a file), while the second is a typical signature of a transfer failure from a source “SURL” (Storage URL on the WLCG Grid) to the destination SURL. In a nutshell, it can be inferred that in the time window tagged as anomalous via the OCSVM algorithm, the text content indeed indicates a possible source of the anomaly in the failure of SRM-level interactions among user requests and CNAF storage, that might have resulted in massive logging activity.

In conclusion, the analysis done based on this prototype seems along the original requirements and indicates the possibility of allowing any CNAF operator to dig deeper into what happened in critical time windows, and consequently acquire knowledge on it. This step is fundamental in the evolution of a CNAF predictive maintenance paradigm.

Chapter 6

Conclusions and next steps

This thesis outlines my contribution to the work done in view of the future deployment of a log-based Predictive Maintenance solution at INFN-CNAF Tier-1. The original contribution of this work is an original lightweight software prototype designed to identify critical time windows of the StoRM service. Moreover, the prototype was designed to be extensible to other services in which the criticality is related to the production rate of log entries. Another addressed point is the potential of content extraction via Text Processing techniques.

After defining and preprocessing the observable, namely the Volatility, an Anomaly Detection algorithm was deployed to detect automatically anomalous time windows of a service taken as example (StoRM storage service in this case) inside analysed days. Finally, using a Text Processing tool, namely TFIDF, some actionable content was extracted from the aforementioned anomalous time windows.

The pivotal results were:

- The Unsupervised approach has been validated and ranked as effective in the proposed task. This work confirms previous results ([3], [41]) developing an original and independent prototype based on the volatility metric revealing critical workloads. The prototype is currently part of the analytical tools

that support CNAF's maintenance infrastructure.

- The run-time of the algorithm over a off-the-shelf business laptop¹ can be considered good, as time consumption of the entire prototype action is well below a single hour over average log file sizes from StoRM (≈ 1 GB), which represents a step forward with respect to previous works.

6.1 Next Steps

Future directions of the work from this thesis involve: Text Processing of log messages and prototype's computational speed further improvement.

As regards the former, the speeding-up of message processing laid the groundwork for further and more rigorous Text Processing studies: as an example, collaborators of this work are already exploring the development of an original Sentiment Analysis strategy over log messages to detect recurrent speech templates over time.

Regarding the domain of computational speed some improvement can be obtained exploring the evergrowing domain of external computing resources (e.g. Graphical Processing Units (GPUs), Artificial Processing Units (APUs) and Tensor Processing Units (TPUs), etc.) and adapting code to run over it. Moreover, adapting the approach showed in this thesis to run over advanced Analytics platforms (such as Apache, Openstack, etc.) may extend the potential of the prototype even further.

All the aforementioned next steps, some of which have explored in a preliminary manner (e.g. Text Processing) stand as strong suggestions for future directions of development. INFN-CNAF is strongly interested in advancing on projects and is currently focusing on increasing the amount of different log files (from different services) in order to explore e.g. correlations, metrics etc.

To encourage further developments, the entire prototype's code will be released with GNU GPLv2 Free Software Licence

¹HP Spectre x360 2018 Intel i5 8th-gen 4-Core 8-Thread CPU, 8 GB RAM, Windows 10 + Windows Subsystem for Linux

(<https://github.com/FrancescoMinarini>), being an initial help to support the evolution of a Predictive Maintenance architecture at INFN-CNAF Tier-1.

Appendix A

This appendix shows, for the sake of completeness, more charts obtained during the Front-End analysis (see Chapter 5). Figures 6.1 and 6.3 refer to a day marked as “critical” (December 6th). Figures 6.2 and 6.4 refer to a day marked as “regular” (December 8th).



Figure 6.1: Log count from CNAF StoRM Front-End on a critical day (see text).

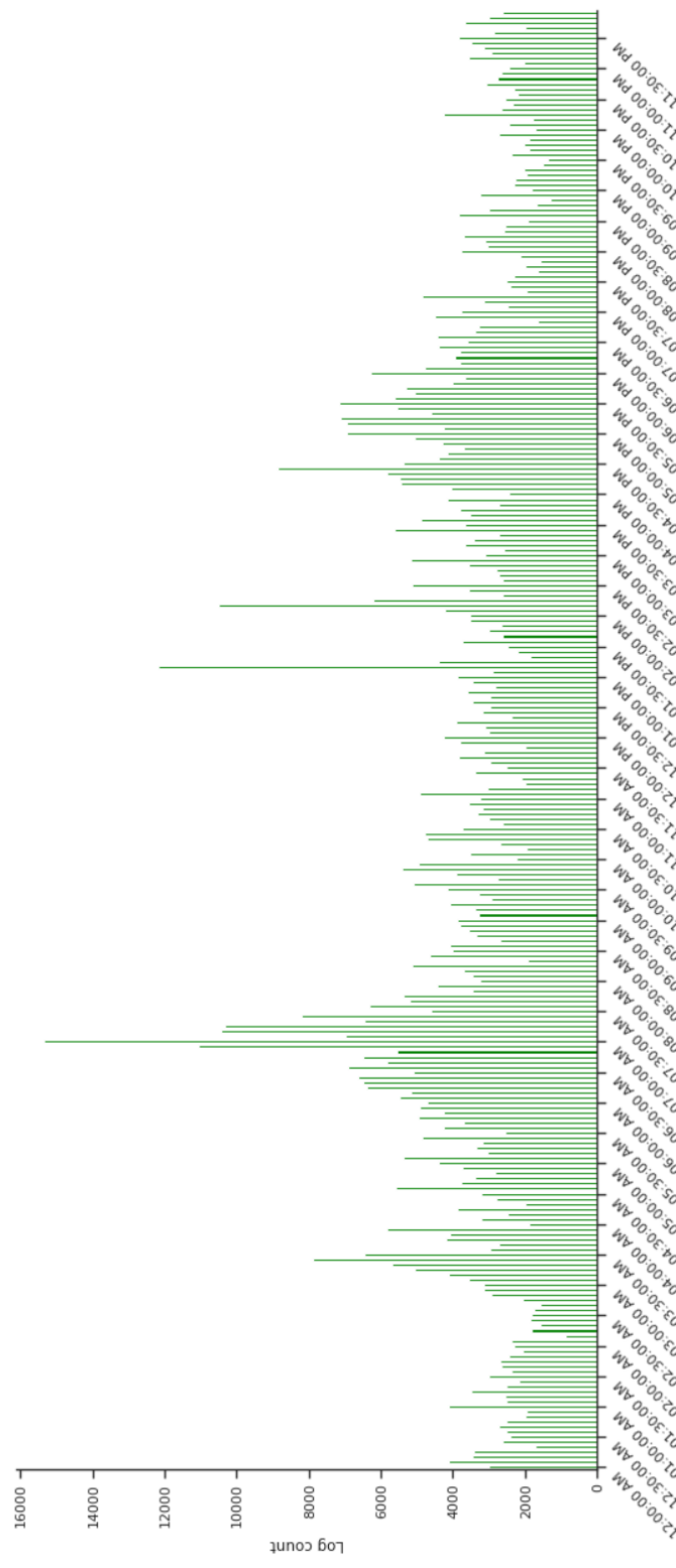


Figure 6.2: Log count from CNAF StoRM Front-End on a regular day (see text).

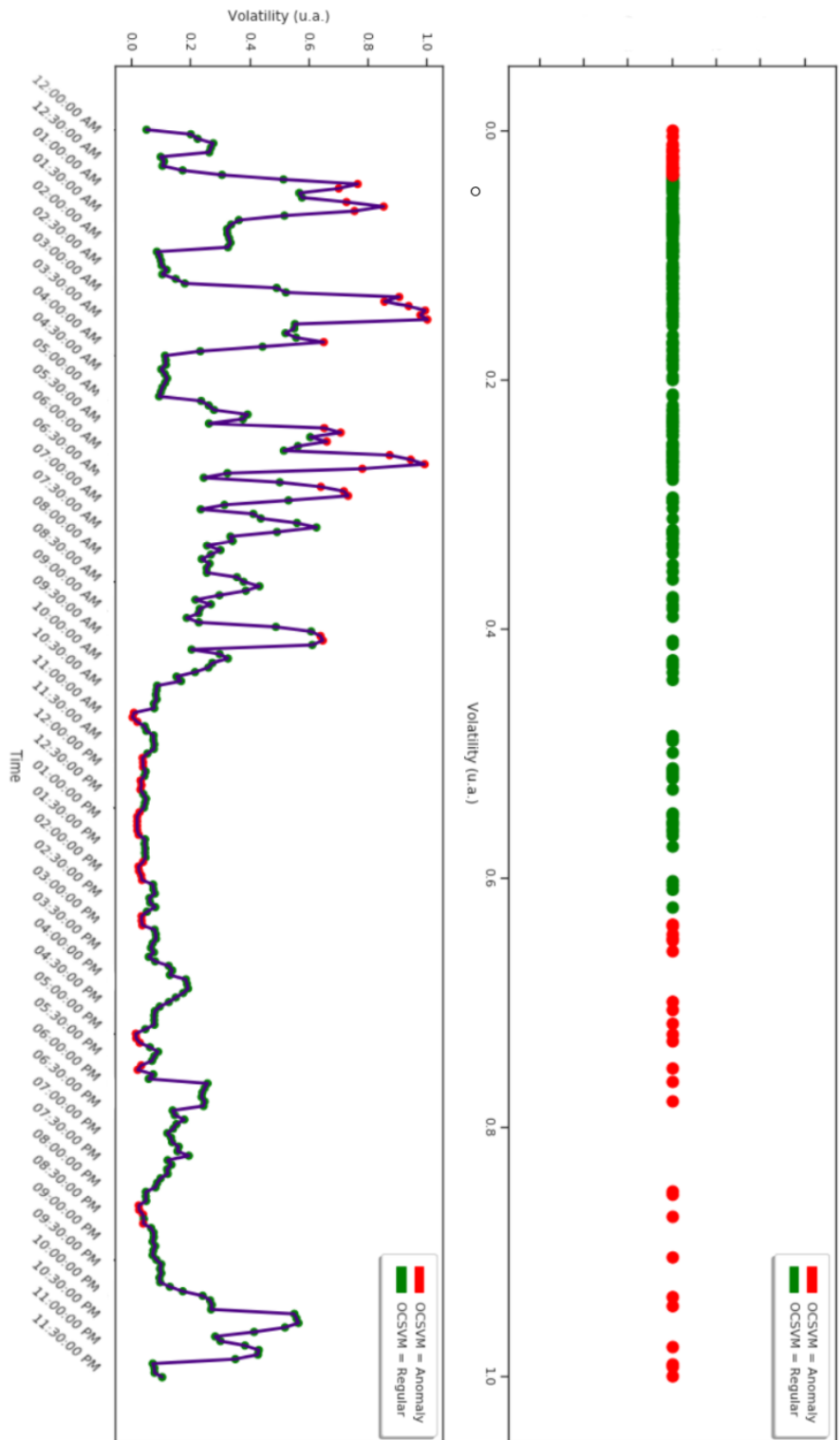


Figure 6.3: (top) Volatility values color-coded depending on the label predicted by OCSVM: red for anomalies and green for regularities. (bottom) Volatility points highlighted with the same colour code.

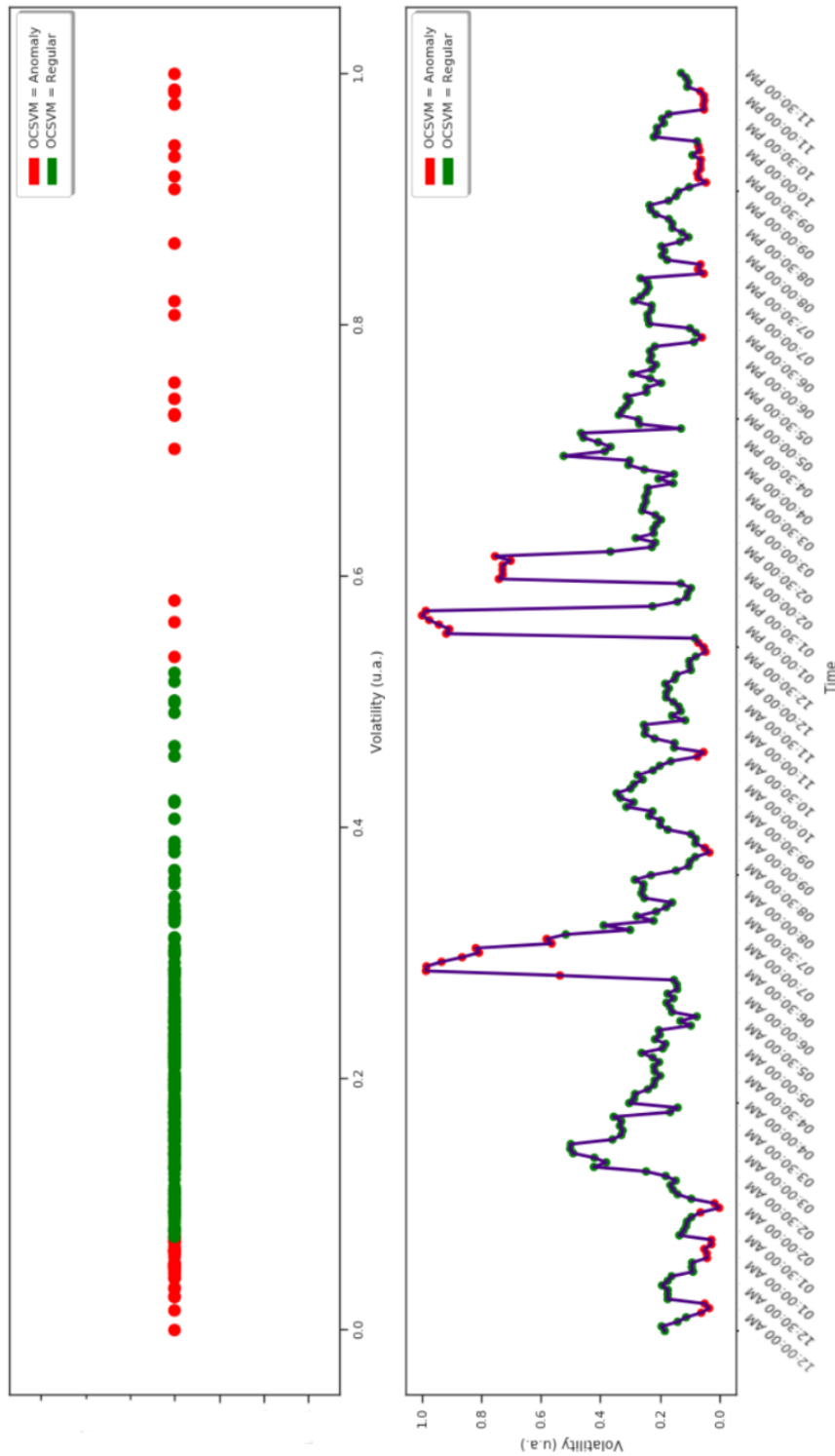


Figure 6.4: (top) Volatility values color-coded depending on the label predicted by OCSVM: red for anomalies and green for regularities. (bottom) Volatility points highlighted with the same colour code.

Appendix B

This appendix shows a further example obtained during Back-End analysis (see Chapter 5). Figure 6.5 shows the log count extracted from May 27th and Figure 6.6 shows the volatility chart related, where volatility is able to capture also a growing trend. This is a further confirmation of its flexibility.

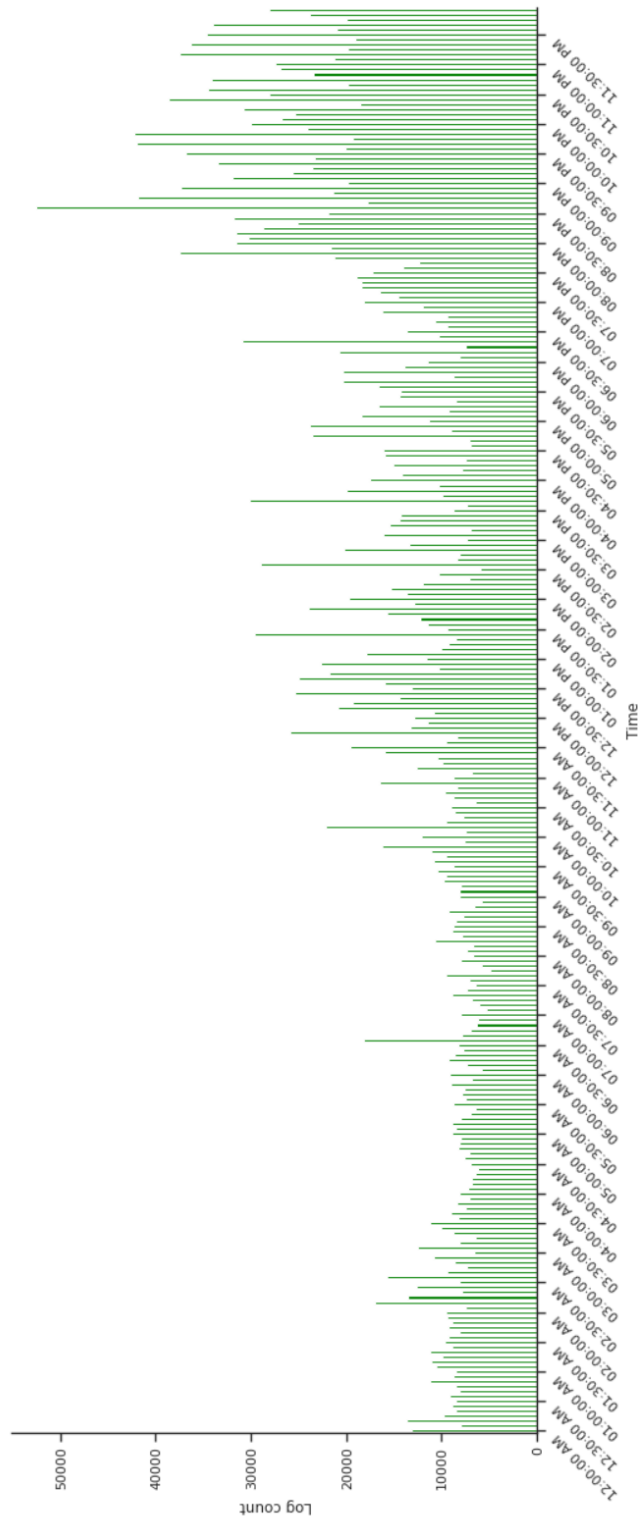


Figure 6.5: Log count on CNAF StoRM Back-End log file (see text).

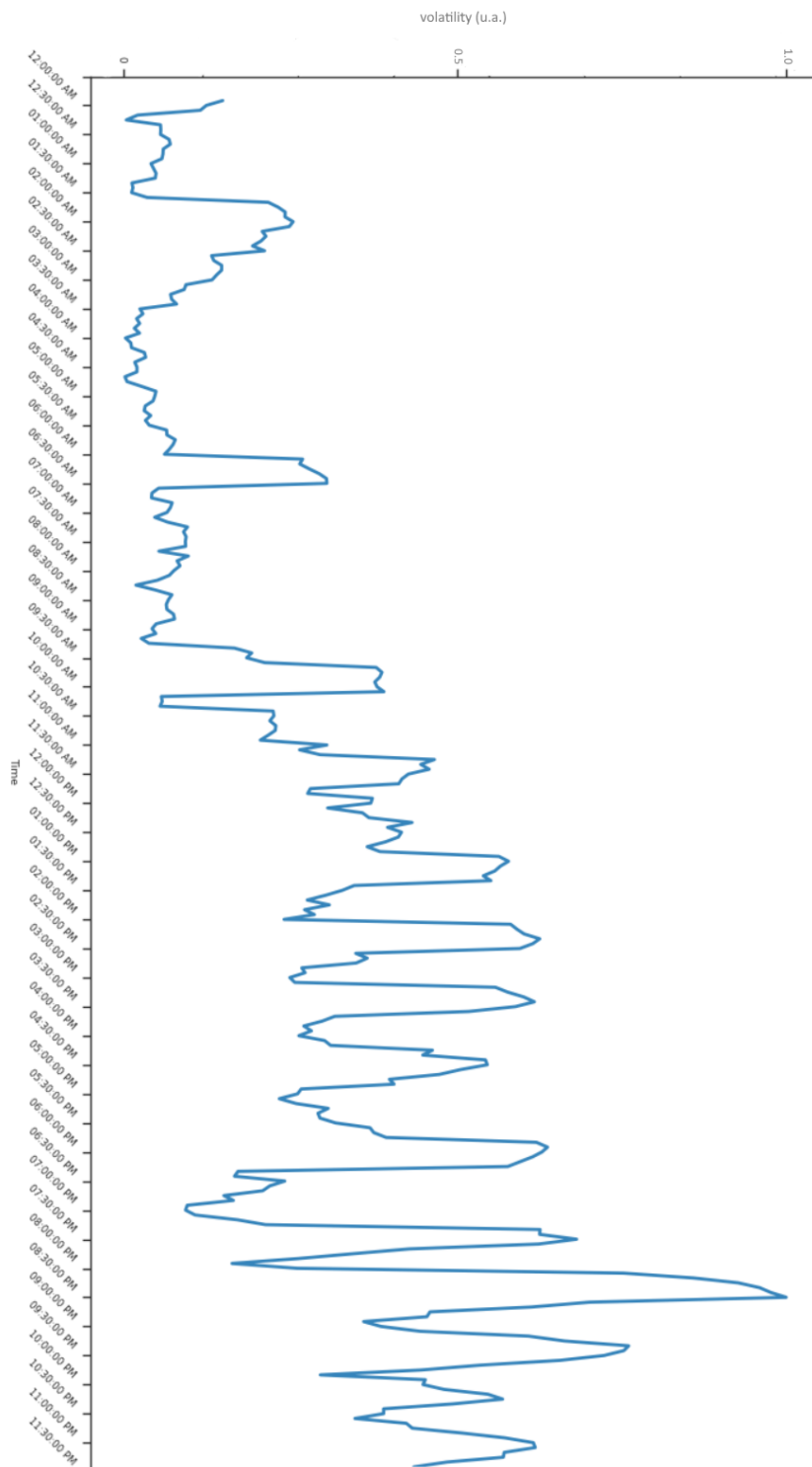


Figure 6.6: Volatility on CNAF StoRM Back-End log file (see text) unrolled as a time-series.

Bibliography

- [1] Herr, W; Muratori, B: “*Concept of Luminosity*”, 2006, DOI: doi:10.5170/CERN-2006-002
- [2] HEP Software Foundation: “*A Roadmap for HEP Software and Computing RD for the 2020s*”, December 15, 2017, DOI:”10.1007/s41781-018-0018-8”
- [3] Giommi, L et al(): “*Towards Predictive Maintenance with Machine Learning at the INFN-CNAF computing centre*”, International Symposium on Grids Clouds 2019, ISGC2019, 31st March - 5th April, 2019.
- [4] CERN: “*CMS The computing Project: technical design report*”, 2005, CERN-LHCC-2005-023
- [5] HL-LHC Project:
<https://home.cern/science/accelerators/high-luminosity-lhc>
- [6] Bishop, C. M.: “*Pattern Recognition and Machine Learning*”, Springer, 2006, ISBN 978-0-387-31073-2
- [7] Najm, I. Khalaf, Alaa Lloret, Jaime Bosch, Ignacio: *Machine Learning Prediction Approach to Enhance Congestion Control in 5G IoT Environment*. 2006, DOI: 10.3390/electronics8060607
- [8] Janardhanan, Deepak; Barrett, Enda: *CPU Workload forecasting of Machines in Data Centers using LSTM Recurrent Neural Networks and ARIMA Models*, 2018, DOI: 10.23919/ICITST.2017.8356346

-
- [9] Jin, Shi; Chakrabarty, Krishnendu: *Anomaly-Detection-Based Failure Prediction in a Core Router System*, 2016.
- [10] The MONARC project: <https://monarc.web.cern.ch/MONARC/>
- [11] CNAF: <https://www.cnaf.infn.it/>
- [12] IBM General Parallel File System: ibm.com/support/knowledgecenter/
- [13] Tivoli Storage Manager: ibm.com/support/
- [14] The StoRM project: <https://italiangrid.github.io/storm/index.html>
- [15] UNI EN 13306, 2003
- [16] Trojan, Flavio; and Rui Marcal, F.M: *Proposal of Maintenance-types Classification to Clarify Maintenance Concepts in Production and Operations Management*, July 2017, DOI:10.15341/jbe(2155- 7950)/07.08.2017/005
- [17] IEC60050, Chapter 191, *Dependability and quality of service*.
- [18] Hinds, Michael: “*Preventive Maintenance*”, NY Times.
- [19] Mobley, R, Keith: “*An introduction to Predictive Maintenance*”, Butterworth-Heinemann. ISBN 978-0-7506-7531-4, 2002
- [20] Chuan-Jun, Su; Quan Yon, Jorge A: *Big Data preventive Maintenance for Hard Disk Failure Detection*, 2018, pp. 471:481. DOI:10.18178/ijiet.2018.8.7.1085
- [21] Mazumdar, Somnath; Kumar Anoop S.: *Forecasting Data Center Re-source Usage: An Experimental Comparison with Time-Series Methods* .In: *International Conference on Soft Computing and Pattern Recognition*. Springer, pp. 151â165, 2016.
- [22] You-Luen Lee et al: *DC-Prophet: Predicting Catastrophic Machine Failures in Data Centers*. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2017

- [23] InfluxDB v1.7 <https://docs.influxdata.com/influxdb/v1.7/>
- [24] RSYSLOG <https://www.rsyslog.com/>
- [25] Micciché, Salvatore; Bonanno, Giovanni; Lillo, Fabrizio; Mantegna, Rosario N.: *Volatility in Financial Markets: Stochastic Models and Empirical Results* In: *Physica A*, 314, pp 756-761, 2002, DOI: [https://doi.org/10.1016/S0378-4371\(02\)01187-1](https://doi.org/10.1016/S0378-4371(02)01187-1)Get
- [26] Hinton, Jeffrey; Sejnowski, Terrence: *Unsupervised Learning: Foundations of neural computation*, MIT Press, 1999, ISBN 978-0262581684.
- [27] Filipovych, Roman; Resnick, Susan M.; Davatzikos, Christos: *Semi-supervised Cluster Analysis of Imaging Data*, In: *NeuroImage* 54(3), 2011, DOI:10.1016/j.neuroimage.2010.09.074
- [28] Punj, Girish; Stewart, David W.: *Cluster analysis in digital marketing: Review and suggestions of application.*, 1983
- [29] Wilks, D, S: *“Statistical Methods in the Atmospheric science”*, Academic Press, 1995
- [30] Kaur, Sarbjeet; Kaur, Prabhjot: *“Review of different types of anomalies and Anomaly detection techniques in Social Networks based on graphs”*, in : *International Journal of Computer Trends and Technology (IJCTT) â Volume 47 Number 2* , 2017
- [31] Zimek, Arthur; Schubert, Erich: *“Outlier Detection”*, In: *Encyclopedia of DataBase Systems*, pp 1-5, Springer, 2017, DOI: <https://doi.org/10.1007/978-1-4899-7993-30719-1>
- [32] Gutierrez Osuna, Ricardo: *Pattern Recognition Course* <https://psi.engr.tamu.edu/courses/>
- [33] Vapnik, N.V; Chervonenkis, A.Ya: *On the uniform convergence of relative frequencies of events to their probability*, In: *Theory of Probability and its applications*, 1971.

-
- [34] Cherkassky, V. and Mulier, F: *Learning from Data: Concepts, Theory, and Methods*.. In: *Wiley Interscience*, 1998.
- [35] Muller et al: *An Introduction to Kernel-Based Learning algorithms*. In: *IEEE Transactions on Neural Networks* 12(2):181-201, February, 2001.
- [36] Vapnik, Vladimir, N et al: *A Training algorithm for optimal margin classifiers*, 1992.
- [37] Karush, W: “*Minima of functions of several variables with inequalities as side constraints*”, Master’s thesis dissertation at University of Chicago, 1939.
- [38] Kuhn, Harold W; Tucker, Albert W: “*Nonlinear Programming*”, In: *proceeding of second Berkeley Symposium*, pp 481-492, 1951.
- [39] Cover, T.M: *Geometrical and Statistical properties of systems of linear inequalities with applications in pattern recognition*, In: *IEEE Transactions on Electronic Computers*, 1965.
- [40] The SciKit Project: <https://scikit-learn.org/>.
- [41] Global Grid User Support: <https://wiki.egi.eu/wiki/GGUS:MainPage>
- [42] Rossi Tisbeni S: *Big Data Analytics towards Predictive Maintenance at INFN-CNAF*, *Master Thesis*, 2019.

List of Figures

1.1	<i>Pictorial representation of the expected LHC evolution towards High Luminosity run.</i>	9
2.1	<i>Pictorial representation of the Tier Hierarchy with focus on computing centres and respective locations.</i>	15
2.2	<i>Pictorial representation of the Tier Hierarchy with focus on inter-facility relations as defined by the MONARC project.</i>	16
3.1	<i>CJ-130J Hercules propellers cleaning from salt and moisture is an example of preventive maintenance.</i>	22
3.2	<i>Predictive/Advanced maintenance infographic.</i>	24
3.3	<i>Structure of Rsyslog collective bucket.</i>	25
3.4	<i>Schema of StoRM with one FE and one BE component.</i>	26
3.5	<i>Sample of log messages structure coming from StoRm-FE</i>	27
3.6	<i>Sample of log messages structure coming from StoRM-BE</i>	28
4.1	<i>Simplified representation of Volatility (green line) when evaluated on a ramp-up line (top) and on a ramp-down one (bottom).</i>	32
4.2	<i>Simple representation of a point-like anomaly.</i>	34
4.3	<i>Simple representation of a contextual anomaly.</i>	35
4.4	<i>Network reconstruction, color-coded in terms of degree centrality, of a fraud (Ponzi Scheme). Centralization of arrows is a case of collective anomaly.</i>	35

4.5	<i>Simplified representation of a classification task and a classification model. [32]</i>	38
4.6	<i>A simple pictorial representation of the maximum margin model. [32]</i>	40
4.7	<i>Graphical example of Support Vectors. [39]</i>	42
4.8	<i>Pictorial representation of the application of a OCSVM with RBF kernel trick.</i>	43
4.9	<i>TFIDF matrix of a generical corpus of documents.</i>	46
4.10	<i>Representation of words positioning in a TFIDFscore-Occurrence space</i>	46
5.1	<i>A GGUS ticket reporting an anomaly in the behaviour of the StoRM Front-End service at INFN-CNAF (more details in the text). User information have been anonymized.</i>	50
5.2	<i>Log count from CNAF StoRM Front-End on a regular day (see text for details).</i>	51
5.3	<i>Log count from CNAF StoRM Front-End on a critical day (see text for details).</i>	52
5.4	<i>Volatility on CNAF StoRM Front-End logs, on 4 reference days.</i>	53
5.5	<i>(top) Volatility values, color-coded depending on the label predicted by OCSVM for a “regular” day: red for anomalies and green for regularities. (bottom) Volatility unrolled as a time-series, with anomalies highlighted with the same colour codes.</i>	55
5.6	<i>(top) Volatility values, color-coded depending on the label predicted by OCSVM for a “critical” day: red for anomalies and green for regularities. (bottom) Volatility unrolled as a time-series, with anomalies highlighted with the same colour codes.</i>	56
5.7	<i>Size of StoRM Front-End logs throughout the analysed period. May, 23rd 2019, the day discussed in the text, presents an anomaly.</i>	57
5.8	<i>Log count from CNAF StoRM Front-End on May 23rd, 2019 (critical day, see text for details),</i>	59
5.9	<i>Volatility, unrolled as Time-Series, extracted from CNAF StoRM Back-End log on May 23rd, 2019.</i>	60

5.10	<i>(top) Volatility values color-coded depending on the label predicted by OCSVM: red for anomalies and green for regularities. (bottom) Volatility points highlighted with the same colour code.</i>	62
6.1	<i>Log count from CNAF StoRM Front-End on a critical day (see text).</i>	70
6.2	<i>Log count from CNAF StoRM Front-End on a regular day (see text).</i>	71
6.3	<i>(top) Volatility values color-coded depending on the label predicted by OCSVM: red for anomalies and green for regularities. (bottom) Volatility points highlighted with the same colour code.</i>	72
6.4	<i>(top) Volatility values color-coded depending on the label predicted by OCSVM: red for anomalies and green for regularities. (bottom) Volatility points highlighted with the same colour code.</i>	73
6.5	<i>Log count on CNAF StoRM Back-End log file (see text).</i>	75
6.6	<i>Volatility on CNAF StoRM Back-End log file (see text) unrolled as a time-series.</i>	76

List of Acronyms

HEP High Energy Physics

LHC Large Hadron Collider

WLCG Worldwide LHC Computing Grid

INFN National Institute of Nuclear Physics

CNAF National center for research and development of computer and telematic technologies

StoRM Storage Resource Manager (service)

CMS Compact Muon Solenoid

GB GigaByte

MB MegaByte

HDD Hard Disk Drive

CERN Conseil européen pour la recherche nucléaire

HSF HEP Software Foundation

CWP Community White Paper

OpInt Operation Intelligence

ML Machine Learning

CPU Central Processing Unit

CE Computing Element

SRM Storage Resource Manager (protocol)

UI User interface

EGI European Grid Infrastructure

OSG Open Science Grid

MoU Memorandum of Understanding

SATA Serial Advanced Technology Attachment

SSD Solid State Disk

I/O Input/Output

GPFS IBM General Parallel File System

TSM IBM Tivoli Storage Manager

WAN Wide Area Network

LAN Local Area Network

GARR Italian National Research and Educational Network

CINECA InterAcademic Union of North-Eastern Italy for Automated Calculus

GBps GigaBytes per second

AI Artificial Intelligence

Rsyslog Rocket Fast System for Log processing

NFS Network File System

FE Front-End

BE Back-End

AD Anomaly Detection

OD Outlier Detection

OCSVM One-Class Support Vector Machine

SVM Support Vector Machine

RBF Radial Basis Functions

IR Information Retrieval

TFIDF Term Frequency Inverse Document Frequency

NLP Natural Language Processing

GPU Graphics Processing Unit

APU Accelerated Processing Unit