CentraleSupélec

IT4BI

BNP PARIBAS
La banque d'un monde qui change

Decision Support and Business Intelligence

Information Technologies for Business Intelligence

Master Thesis

Ronald Mitterand NYAMI TCHOUMBUN

REAL-TIME LOG ANALYTICS

prepared at BNP Paribas

Defended on November 4/5, 2015

Advisor :	Sébastien COISNE	-	BNP Paribas
Advisor :	Gianluca QUERCINI	-	CentraleSupelec
Supervisor :	Nacéra BENNACER	-	CentraleSupelec

Acknowledgments

This thesis was made possible thanks to the patience and constant support of my professional and academic advisers; Germain Chaton, Sébastien Coisne, my professors Gianluca Quercini and Nacéra Bennacer, may you please accept my expression of deepest gratitude.

This thesis was made possible within the context of my internship at BNP Paribas.

Thank you mother, my dearest mother Tomeko Odette. You always know how to create in me this desire to forge ahead. Thank you Alain Gatchou, I will forever be grateful to you . Thank you Tepong Armelle, thank you Tomeko Ronald, thank you my brother and thank you my sisters, don't stop believing in me, it gives me faith, hope, and courage.

I especially dedicate this thesis to the memory of my late, beloved father Tchoumbun Daniel. "Dad, you will forever remain my source of strength and inspiration."

Contents

Chapter 1
Introduction

Globally speaking, the analysis of log data has revealed itself to be an important task for companies willing to improve the performance of their computing system. Such system monitoring and supervision are important prerequisites to an efficient performance management.
This thesis work specifically focuses on Next log analysis in real time, where Next can be described as an innovative web Content Management System (CMS) installed on top of the distributed computing system of a banking institution.
The company behind Next is a major global bank. The group is active in 75 countries in the world with nearly 185.000 employees, more than 76% in Europe, it has key positions in its two main areas of activity: 73% Retail Banking and Services (RBS), 27% Corporate Institutional Banking (CIB). In Europe, the group with four domestic markets in Belgium, France, Italy and Luxembourg is the leader in consumer finance. The group is also developing its integrated retail banking model in countries around the Mediterranean basin, in Turkey, in Eastern Europe with a large network in the Western United States. Its Investment Solutions and Corporate and Investment Banking businesses are among the leaders in Europe and it boasts a solid foothold in the Americas as well as a robust and rapidly-growing presence in Asia-Pacific. All this to give an idea on the large amount of log data to be processed daily as generated by their distributed computing system carrying all websites activities and transactions. More information related to the company latest description are available in [Paribas 2013].
In this leading banking institution of the Eurozone, there are millions of customers simultaneously connecting and performing billions of transactions every day. In the institution's performance management department, Next is depicted as their Content Management System with a distributed and a layered architecture. Next has several server farms located in different sites and in each site there are several server's parks.
Next underlying distributed hardware generates petabytes of data about events through messages stored in their system log files. As the system log file size is huge, there is a need of suitable techniques and dedicated algorithms with reliable storage system to process them. There is a real need to capture messages behind each event since they carry important knowledge about the diagnosticity of the system.
In that context, these researches are particularly carried out on logs generated by the servers carrying the banking institution's websites with contained applications. These studies propose reasonable solutions to issues related to the monitoring in

real time of the distributed architecture through implementation of a Next Log Analysis Algorithm in Real Time (Nlart) for Anomaly detection.
The Next server's failure issues are similar to those already encountered in modern distributed computing systems where it is essential to be able to detect server's failures as soon as they occur or possibly predict them in order to react in time and quickly apply patches and corrections. In such distributed computing systems, the failure detection ability is a prerequisite in order to ensure performance, security, reliability, scalability and availability optimizations.
This thesis work thus focuses on a way to leverage Next system oriented toward performance management and error monitoring performing analytics on the server's log data in order to enhance at a macro level end user experience online while providing important auditing information for the business side.
The main objective being to develop solutions allowing to monitor the underlying distributed architecture of servers detecting their failures in real time, this our thesis work proposes an algorithm that processes the large amount of generated log data tracking for anomalies that may occur followed by the integration of generated results into a human machine interface allowing to easily and visually browse the results.
The proposed algorithm is tracking for server's profiles changes. It detect anomalous servers based on those changes to build groups of normal and anomalous clusters. The behaviour and characteristics of server's are all extracted from generated logs. The algorithm is running an anomaly detection paradigm by extending a clustering algorithm already available in the literature as it will extensively be described later. The design of the algorithm mainly relies on system workload attributes since it takes into consideration the fact that an anomaly is related to a software failure, an application failure, a system overload or a human mistake. The algorithm design also relies on domain expertise and knowledge since it makes use of threshold values defined according to the server's functional properties.
In these researches, input data are streams of logs and output are statistical models allowing visual analytic. The approach is to be applied in an unsupervised and online manner but for testing and research the learning will be made with labelled training dataset.
This work is organized into several chapters. *Chapter 2* is about the previous related work, *Chapter 3* is about the research purpose together with the proposed solution, it depicts the context architecture and the data structure. *Chapter 4* is about a detailed description of the research algorithm and its implementation, it describes the experiments conducted followed by their evaluation and the interpretation of observed results. *Chapter 5* is about the conclusion together with recommendations for possible improvements and future perspectives.

Chapter 2

Related work

In this section, we are reporting on existing approaches in the field of real time anomaly detection from server's log files in distributed computing systems.
The main objective is to converge as much as possible toward reasonable solutions among existing ones in the research field. To do so, the research studies are concentrating at first on the different types of data mining techniques used for anomaly detections on time series data [Gupta 2014]. The studies are particularly paying attention to researches on the monitoring of server's from generated log data [Xu 2009a] [Xu 2009b] [Sequeira 2002] [Hill 2007] [Yang 2008] [Yamanishi 2005] [Ziv 1988] [Shahid 2012].
Once this is done, the work then focus on available studies of real time anomaly detection techniques through clustering of log data [Chen 2002] [Chen 2004] [Vaarandi 2003] [Münz 2007] [Liu 2010] [Liu 2011] [Burbeck 2005]. The research then finally shows in the Table 2.1 a comparative description of the different approaches as depicted more in detail hereafter.

2.1 Generalities on anomaly detection techniques for temporal data.

Manish Gupta et al. in [Gupta 2014] have an interesting survey on outlier detection techniques for temporal data. Their paper provides a thorough study about system diagnosis considering the detection of anomalies as a process able to provide warning and status information about system failure. They also provide a classification for temporal data in an interesting manner. They propose several types of temporal data distinguishing them into time series, streams, distributed, spatiotemporal and network data. For streams, they suggest to consider discriminative approaches based on the definition of a similarity function that takes into account the similarity measure between two sequences of temporal data.

The amount of current researches and applications in the field is increasingly huge and in order to well define our area of interest, we need first to understand certain aspects of real time anomaly detection from server's log data.
According to the detection time we can distinguish offline (batch) and online (real time) detection [Gupta 2014]. In the context of monitoring of a distributed computing system as it is the case for us, a real time anomaly detection technique

Research title	Data stream type	Class	Operational Mode
[Xu 2009a]	free text logs	frequent pattern mining and principal component analysis	Batch
[Xu 2009b]	free text logs	frequent pattern mining and principal component analysis	Batch and Real time
[Gupta 2014]	logs	Clustering , Neural Network , Bayesian Network	Batch and Real time
[Sequeira 2002]	logs	Dynamic clustering	Real time
[Hill 2007]	logs	Dynamic Bayesian network	Real Time
[Yang 2008]	wireless sensor logs	QS SVM	Real Time
[Yamanishi 2005]	syslog	Dynamic Hidden Markov Models	Real Time
[Ziv 1988]	logs	Classification	Real time
[Shahid 2012]	logs	Classification and QS SVM	Real Time
[Chen 2002]	logs	probabilistic context free grammars and pinpoint	Real time
[Chen 2004]	logs	probabilistic context free grammars path	Real time
[Vaarandi 2003]	logs	frequent pattern mining and clustering	Real time
[Münz 2007]	Network Traffic logs	Clustering k-means	Real Time
[Liu 2010]	logs	Map Reduce Based mining	Real time
[Liu 2011]	logs	Map Reduce Based Clustering	Real time
[Burbeck 2005]	Server's log	fast Incremental Clustering with BIRCH	Batch
Our approach	Server's log	Clustering with Clustream	Real time

Table 2.1: Comparative summary.

is a necessary approach because it means early detection of system failure, software application failure, incorrect setup (human mistake) or system overloading [Pertet 2005].
Anomaly detection techniques generally depend on the type of targeted anomalous data that can either be a point anomaly or a statistical anomaly [Gupta 2014]. The first one relates to methods designed to detect simple point anomalies that is individual data instance anomalous relatively to the rest of the data points, while the second one is more about the distribution of the data. Considering the data complexity in a large and distributed computing system, it is more suitable for an anomaly detection process, as it is the case for us, to take into account contextual information [Gupta 2014]. We can also distinguish in their review several class based approaches such as classification, clustering, Neural Network and Bayesian Network.

2.2 Online anomaly detection techniques for log stream data.

Wei Xu et al. in [Xu 2009a] propose anomaly detection through frequent pattern mining and principal component analysis. In an online and supervised setting, their approach allows the automatic monitoring of console logs in order to automatically detect anomalous execution traces. They propose a combination of frequent pattern mining and Principal Component Analysis for failure detection that pre-processes the logs and generates a labelled dataset that is used for an evaluation in a next step. The pattern mining step filters normal events based on the assumption that they represent the majority of all events in logs while the principal component modelling step allows to decide if non-matching data sequences are anomalous.

They experimented with multiple logs files generated by a Hadoop production environment running on top of a cluster of 203 machines. They found results that are better in detection accuracy than their previous offline research in [Xu 2009b] with very low detection latency and high accuracy. They achieved in the online experiments 86% Precision and 100% Recall.

Sequeira and Zaki in [Sequeira 2002] use dynamically maintained cluster models to detect outliers in data streams. Their research makes use of a normalized length of a longest common subsequence as sequence similarity measure to dynamically perform clustering. This longest common subsequence is then used to compute a sequence rating that indicates anomalies when falling under some set of criteria with predefined small thresholds. They justify the choice of dynamic clustering over other approaches like K-means by the fact that it is not subject to random initialization, thus has a better accuracy and does not need predefined number of clusters nor previously imposed number of iterations. They experiment on a collection of 150 sessions of user audit data by dynamically generating clusters on demand and they observe good results with 80% detection rate and 15% false positive rate. ADMIT (Anomaly-based Data Mining for Intrusions) that they develop in their research shows a short training time and a good scalability for real time applications.

Hill et al. in [Hill 2007] propose an approach based on Dynamic Bayesian network. The network structure adapt as new state variables arrive. They propose two contributions for detecting anomalies. In the first contribution they use a hidden Markov model with Kalman filtering sequentially deducing the posterior probabilities of new arriving variables. They make use of the posterior distribution to build a Bayesian credible interval for the most recent set of log data. Any log data that fall outside of the Bayesian credible interval can be classified as anomalous. In the second contribution, they use a Dynamic Bayesian network with 2 layers where the status of each log data is also modelled as a hidden state variable over the time. Then they use the most likely value given the posterior distribution of the hidden state variable to classify the incoming log data as normal or anomalous.

Zhang et al. in [Yang 2008] describe outlier detection techniques on wireless sensor network. They show that main challenges for outlier detection in many distributed settings are related to resource constraints such as memory, computational capacity, bandwidth, online processing cost of distributed streaming data, dynamicity of the network topology, frequent communication failures, need for scalability or category of outliers (errors, events, malicious attacks and so on.). They run their experiments to find top distance based outliers making use of the global data on a network of sensors, each sensor being considered a node associated to stream of incoming nodes. In their research, the anomaly detection

is consisting of a process where each node is computing a local radius by running locally an extension of the one class support vector machine known as Quarter Sphere Support Vector Machine (QS SVM). The computed radius is then sent to a parent node in the network that combines it to its own radius in order to provide a global radius. The global radius itself is then finally sent back to the children to detect anomalies. Using a statistical-based outlier detection techniques as their temporal outlier detection, their work is reporting a reduced communication overhead contrasting with a low accuracy in the detection process.

[Yamanishi 2005] proposes dynamic syslog mining in order to detect failure symptoms and to discover sequential alarm patterns among computer devices. Syslog can generally be considered as a standard for message logging based on the separation between the software that generates messages, the system that stores those messages and the software that reports and analyses them. Yamanishi et al. key ideas of dynamic syslog mining are to represent syslog behaviour using a mixture of Hidden Markov Models to adaptively learn the model in an online discounting learning algorithm followed by a dynamic selection of the optimal number of mixture components. They make use of universal statistical tests with dynamically optimized thresholds to give their anomaly scores. The definition of anomaly scores they use is also known as universal test statistic and is in fact already developed by Ziv in [Ziv 1988]. This scoring is the combination of Shannon information and event compression efficiency. The concept of anomaly detection in this case states that if the Shannon information of two events are equals, then the event with smaller compression rate (higher regularity) would result in a larger anomaly score.

In [Gunte 2007], the authors develop interesting anomaly detection algorithms. The input to their anomaly detector is a time-series of values from a log summarizer. Each value represents the performance of a stream of disk IO events. They make use of two families of anomaly detection algorithms, the mean N standard deviations and the cumulative distribution function. So they make researches on two different approaches: Static, which uses the first N values in a run and does not change over time and Heuristic, which updates the historical data set with the current value if it is not an anomaly. The benefit in this latter approach being to prevent extreme values to be integrated while still allowing the input data set to evolve. They also add to their work the option of smoothing using the exponential weighted moving average function in order to dampen the impact of outliers.

Path based analysis as in [Chen 2002] by Chen et al. proposes also a clustering and probabilistic method to process context free grammars [Chen 2004] and analyses execution paths in server systems in a manual manner. They experiment on large, complex, distributed computing systems. They successfully detect and

diagnose failures by running a Java based framework allowing to detect anomalies without any prior knowledge of the application components by monitoring the traffic and performing log data analysis; the framework is known as the pinpoint approach.

Vaarandi in [Vaarandi 2003] proposed a novel clustering algorithm allowing the detection of frequent patterns from log files. His contribution is aiming at building log file profiles and at identifying anomalous log file lines. His algorithm is specifically clustering based and builds clusters of frequent words from event logs. He is able to detect anomalies by first passing over the data and building a data summary collection of all frequent words and then building a cluster candidate table by in a second pass over the log file line by line. When one or more frequent words have been detected in a line, a cluster candidate is formed using these frequent words as its attributes. If the cluster candidate is not present in the candidate table, it will be inserted with a support value equal to 1, otherwise its support value will be increased. After all cluster candidates have been built, the support values of the cluster candidates are examined and the clusters candidates that are frequently used are selected by the algorithm as normal clusters while the other one are considered anomalous.

Traffic anomaly detection as depicted by Gerhard Munz in [Münz 2007] consists in a flow based anomaly detection through the K-means clustering algorithm. Training log data set generated from traffic are divided into groups of normal and anomalous traffic. They are able to detect anomalous traffic by computing online the distance from the cluster's centroid to the incoming log data entry. Advantage of their approach resides in the fact that all the task of data Pre-processing such as duplicate removal, independence checking, normalization, missing data fixing, all the task of data cleaning and data filtering are made easier and they can avoid incoherencies in rules and patterns extracted.

System anomaly detection in distributed systems through MapReduce-Based log analysis as depicted by Yan Liu in [Liu 2010] shows how computing tasks are assigned to multiple machines in parallel in order to improve the processing speed. They demonstrate in these researches and with good results that distributing the task over several machines is a serious candidate solution to real time log analysis optimization.
K. Burbeck et al. use fast Incremental Clustering in [Burbeck 2005] to classify data into categories. The normal or anomalous categories are not defined in advance since they use an unsupervised learning approach to plot data's features in an n-dimensional space. The ADWICE (Anomaly Detection With real-time Incremental Clustering) software that is developed at the University of Linkopings extends BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) as already developed by [Zhang 1996] to implement fast, scalable and adaptive

anomaly detection. Their experimental evaluation with real network data and 1999 Kdd Cup dataset show a detection rate of 95 % and a false positives rate of 2.8 %. Their type of training is often subject to production of non-intuitive groupings.

In summary and according to the mining framework one can operate in either supervised, semi-supervised or unsupervised modes. Supervised approach needs as input a labelled training data allowing the anomaly detection engine to compare the ground truth to the new data to be classified, semi-supervised approaches are trained on both labelled and unlabelled dataset needing in some cases only one of the classes to be labelled. Unsupervised modes does not need the training set to be labelled and anomaly detection techniques in this case allow for more flexibility and less human pre-processing steps.

These observations translate into the need to build an anomaly detection system able to notice on the flow the different signs of failure in our large distributed computing systems. A detector able to find unusual variations in the parameters describing system status, identifying changes in system behaviour that indicate a network and/or devices security failure, detecting gap from a usual pattern that provides early warnings of crash. Our approach thus considers an effective real time anomaly detection process that requires continuous learning. The approach presented in this thesis research can be classified in the field of unsupervised online learning. It uses CLUSTREAM as developed by Aggarwal et al. in citeaggarwal2007clustering and extends it to cluster in real time server's logs into categories of normal and anomalous data.

Clustream algorithm computes intermediate cluster statistics as microclusters classifying each new input data stream point as suitable to be inserted in a previously existing microcluster or not. The algorithm creates a new microcluster for isolated anomaly and outliers. Part of the processs in Clustream algorithm can be used in anomaly detection by tracking for newly created microclusters. So by computing the distance between the input data stream point and its closest microcluster, normalized by the cluster extent It is even possible to compute an anomaly score. This distance could be used to show that point far from a small microcluster, when having small extent could be representing a larger anomaly than point that are far from a large microcluster.
K. Burbeck et al. used BIRCH to develop ADWICE in their fast Incremental Clustering algorithm [Burbeck 2005], since Clustream is extended from BIRCH to allow evolving data stream clustering both approaches may show some similarities.

Chapter 3

Proposed Solution

This chapter at first describes the context as briefly introduced earlier, it then goes through a deep presentation of the problem statement and finally describes in details the proposed algorithm.

3.1 Context

In order to better understand our choices in the algorithm design and the derived solution proposed there is a need to specify the context. This section at first goes through a detailed description of the topological structure of Next. It then gives the description of log data structures in the case of Next as it will be used as input to the anomaly detection algorithm. It finally gives the deduced related abstraction done in order to formalize the problem followed by all important term definitions.

Next is running on top of a well-defined technical architecture. Its production environment is divided into many servers farms located in different sites containing several server's parks. A site can be depicted as a distributed physical architecture as shown in Figure 3.1 with load balancing levels allowing a boosted redundancy.

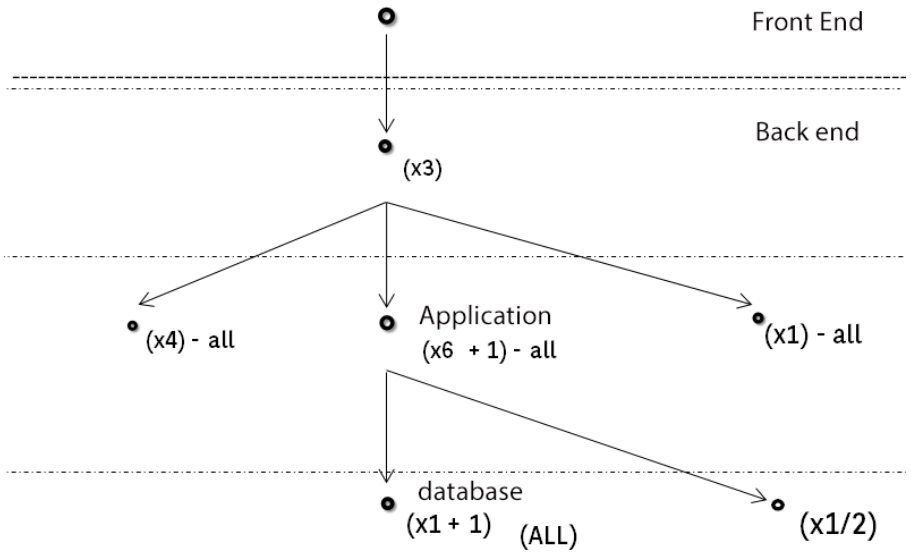

Figure 3.1: Lattice model.

On this site there are several parks. The *x1 +1* means there is one server installed at the database layer with one server working as a backup. *x1/2* here means that the database server is also linked to another site and thus aggregated for multiple sites. *x6+1* at the application layer refers to the fact that there are 6 servers installed plus one working as backup. It corresponds to a single server park. There are other parks having 4 and 1 servers installed and they are depicted as *x4* and *x1*.
A site is later on divided into several server parks, the level that is to be considered in the Nlart clustering paradigm.

Next architecture can logically be represented as a lattice where servers are organized into parents and children. Each node can be one functional module. It can be a database module or an application module as described more in details in the following section. The database module for instance can have several instances and eventually one or more backups, identically the application layer can have several instances and eventually one or more backups depending on the needs and physical configurations.

It is interesting to assume the distinction between servers and server's instances where an instance relates to a snapshot of a server at a given timestamp. This definition enable to differentiate among server's instances those that are anomalous from those that are not.

The monitoring structure is built following the particular structure described by Lavinia et al. in [Lavinia 2010]. The communication protocol in that case consists of nodes through which the information flow is spread from one server node to the other. The logging is executed following that parents and children framework policy. For detection at local level, there is defined a hierarchical structure of servers along which traffic is channelled for real time analytics, see the Figure 3.2.

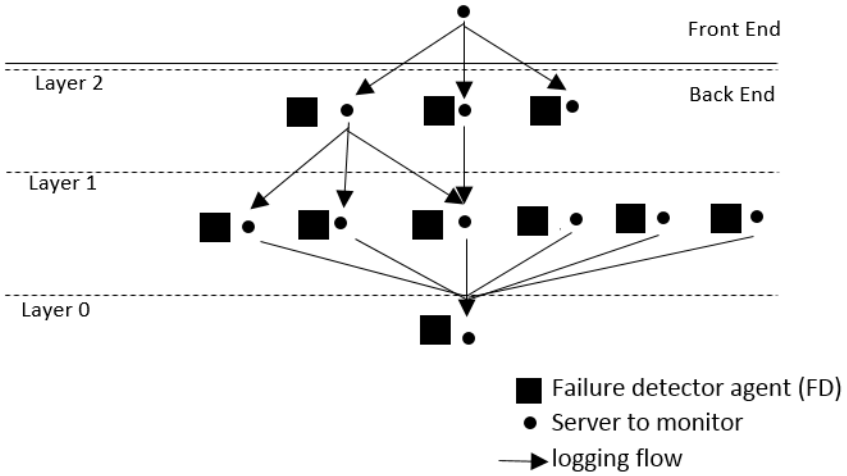

Figure 3.2: Hierarchical protocols.[Lavinia 2010]

The information flow within such a hierarchical structure allows optimized real time diagnosticity for involved servers through Failure Detector agents. The servers are carrying all website transactions and services. The different operations they serve generate Log data that are collected on the flow by those Failure Detector (FD) agents installed on top of each of them in a distributed manner. The logging protocol is shown for each server to be monitored in top down flow. Operations are processed from the front end where user requests are issued to the back end. Logging flow thus goes from load balancer *Layer 2* passing through the application *Layer 1* to arrive to the database *Layer 0*.

The following parameters are used in our research as they are extracted in real time from the servers by those Failure Detector agents:

- the Number of active sessions on the servers. it is a numeric attribute and the header in the log file refers to it as context1, notation also used in the following sections for simplicity.
- the Number of different users categories. These numbers related to user's different roles, they are categorical and the log file header refers to them as category.
- the Number of expired sessions after a timeout, it is a numerical attribute and it is referenced as context2 in the log file.
- the Number of disconnected sessions, it is numerical and referenced as context3 in the log file.
- the Number of expired and cleaned session by the garbage collector, it is a numerical attribute and it is referenced as context4 in the log file.
- the Number of disconnected sessions and purged by the cache (garbage collector), it is numerical attribute and it might show less impact in the analysis according to expert's recommendations. It is referenced as context5 in the log file.
- the Server response time, it is a numerical attribute and it is referenced as responseTime in the log file.
- the number of Session type referenced as sessionType and representing the type of the session in the log file.
- the Workload carried by the Park of servers in the site referenced as workload in the log file.
- the Logical representation of the server instance that are referenced as serverChild, serverParent or IP address depending on the position in the layered architecture in the log file.

- the time of arrival of a new event in the log which is represented by timestamp in the log file.

Those features are used as attributes to characterize server's instances. Server's system status is monitored from several thousand log stream sources with a velocity rate of million lines per minute.

Taking for example the log streams of two layers in one park: database log stream and the application log stream, their data structure and format is the same as those in Apache based servers with specificities of IBM WebSphere Application Server (WAS). The Table 3.1 shows such sample log entries.

timestamp	serverParent	serverChild	category	context1	context2	context3	context4
2015-04-29 00.00.24	serverA	ALL	ALL	32074	827	1421	1221
2015-04-29 00.00.24	serverA	serverA1	10	10040	110	611	1000
2015-04-29 00.00.24	serverA	serverA2	20	10534	112	700	110
2015-04-29 00.00.24	serverA	serverA3	30	11500	605	110	111
2015-04-29 00.10.24	serverA	ALL	10	31039	375	1798	3309
2015-04-29 00.10.24	serverA	serverA1	10	10104	115	589	1100
2015-04-29 00.10.24	serverA	serverA2	10	10500	130	605	1107
2015-04-29 00.10.24	serverA	serverA3	10	10435	130	604	1102
2015-04-29 00.00.24	serverB	serverB1	10	10534	104	1000	1103
2015-04-29 00.00.24	serverB	serverB2	20	10574	105	1001	1107
2015-04-29 00.00.24	serverB	serverB3	30	10834	104	989	1507

Table 3.1: sample lines of Logs

The database logs entries are time stamped values that are generated by specific FD agents as installed in the related node. *ALL* appears each time the parent node aggregates for all children nodes. The children nodes in this case are represented by the serverChild attribute. The FD also records for each server serverParent and serverChild the relative context1, context2, context3, context4 and category attribute.

From such extracted log messages, it is possible to parse out timestamp, serverParent, serverChild and numerical attributes used as input to Nlart for anomaly detection. The attribute serverParent represents the node in the lowest level in the lattice; it is the database layer. Choosing such structure is motivated by the fact that all user's requests processed by the servers in the application layer are forwarded to the database layer where they can be aggregated. The attribute serverChild represents the application layer servers carrying all website applications and services while category represents the different categories of users available with users and context related measures named $context_i$ as previously described.

3.2 Problem statement

Considering the previously described architecture and data structure, there is a need to define a suitable framework policy of anomaly detection for system failures or break down avoidance.
In such context the problem relies in the difficulties to track for system's failures or break down since servers are distributed and layered along the computing system. As mentioned in [Burbeck 2005] researches, a good solution to such issues relies in tracking for changes and unusual events in the logged attributes. In the case of Next and considering dealing with streaming data the logged attributes are characterizing servers at any given timestamp and the unusual behaviours are relating to local server's failures. The solution to the need of anomaly detection in Next's servers thus consists in tracking for server's profile change based on the aggregated logs. The solution consists in an algorithm running in real time.

To better understand the problem we can consider the Figure 3.3 as representing a sample Park structure with two layers Layer 1 and Layer 2. They can reference for example application and database servers.

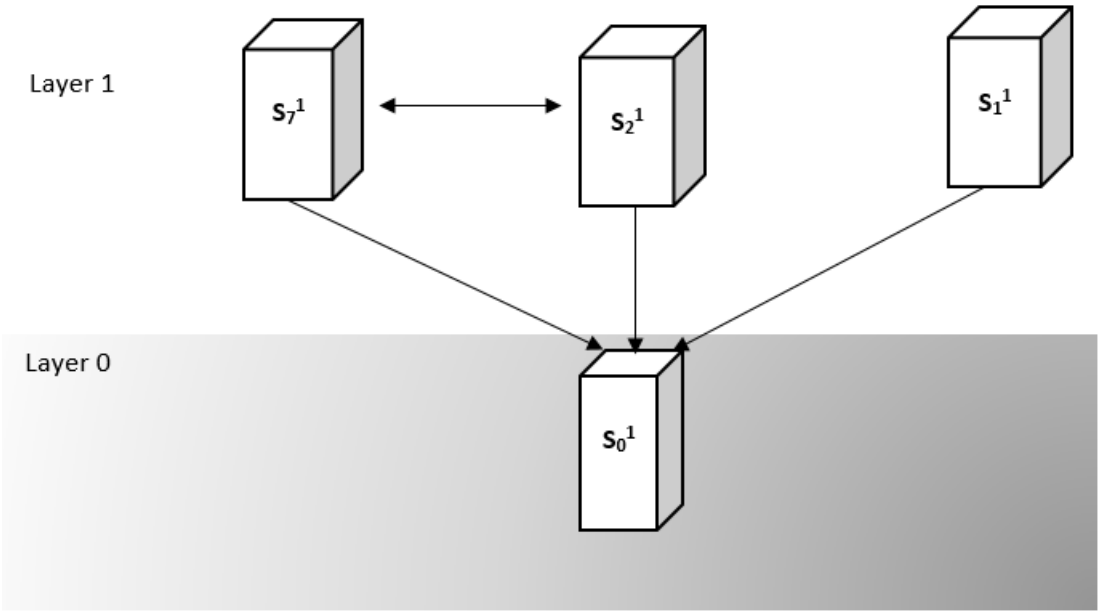

Figure 3.3: Sample Park structure with two layers

Once users are connecting to the websites and are issuing requests, all servers are processing requests in a continuous flow. At the application level, requests can be processed and forwarded to the database level in case there is a need to connect to that database. As a consequence the attribute accounting for the number of user requests processed (context1) can be aggregated at the database level to

show the total number of users that successfully connected to their account on the website and which requests were forwarded on that Park and in that Site by the load balancer.
Possible dysfunctions in this case may rely in an incoherent gap event occurring between the number of user requests on the application layer and the one observed at the database layer but this type of unusual behaviour can easily be detected by suitable statistics.
The complexity arises when the gap event is occurring within the servers of the same layer independently from the park or the site. In this latter case, it is always difficult to classify the gap in the server instance behaviour as a normal or an anomalous event.
So considering servers S_i^m and attributes C_N with m the Park identity, i the layer identity and N the number of attributes, we assume that the following heuristics hold: If there occurs a failure on server S_i^m, it will translate in an incoherence in the observed value of attributes C_N of that server. If grouped by attributes C_N and considering the experts predefined settings, all servers S_i^m are expected to behave the same way within their Park; they can be grouped in the same cluster and they must evolve keeping similar profiles. Any server S_i^m that is showing a different profile from the previous state is reflecting the occurrence of an anomalous event. The grouping based on Park assignment here suppose experts predefined settings, this later meaning that the configuration are made in ISO mode with same physical and logical configurations for all servers within the same group.

Recalling that an anomalous event is an unusual variation in at least one of the attributes characterizing the servers, there is a need to formally define those attributes. At layer Layer1 in Park1, similarly to Layer1 in other Parks, each server can be represented by the following parameters:
$S_1^1 =< C_1, C_2...C_N >$
$S_2^1 =< C_1, C_2...C_N >$
$S_3^1 =< C_1, C_2...C_N >$
At this layer, C_1 to C_N account for the different attributes related to servers functional properties.
In the Figure 3.3 there are 7 servers installed in the Layer1 $< S_1^1, S_2^1...S_7^1 >$. C_1 is the number of users that successfully connected to the website and which requests are successfully processed on the servers where it is computed. C_2 is the context2 and C_N is the contextN.
At Layer0 in Park1, similarly to all Layer1 in other Parks, each server can be represented by the following parameters:
$S_0^0 =< C_1, C_2...C_N >$. At this layer Layer0, the server S_0^0 aggregates attributes of Layer1 together with attributes of its own. C_1 to C_N account for those aggregates. The layer allows to have a global view of the park Park1 behaviour.

The process of server failure detection through Nlart anomaly detection while considering the Figure 3.4 as an extract from logs consists in producing for those servers a snapshop of context1, context2, contextN attributes at successive time periods, example of T_0, T_1 and T_N. Considering all attribute parameters as a vector Point P in the data stream such that $P =< S, T >$ with S a multi-dimensional record of all servers feature and T the corresponding timestamp, if any update in the vector $< C_1, C_2, C_N, T >$ translates into the creation of a new cluster, there is a suspicion of an anomaly. The example in Figure 3.4 considers only three attributes context1 as C_1, context2 as C_2, context3 as C_3 and only three servers from the sample park Park1 as previously described. Figure 3.4 shows the corresponding log data for application servers S_1^1, S_2^1, S_3^1 and database server S_1^0 at time T_0.

T_0	S_1^1	S_2^1	S_3^1	S_1^0
C_1	3	3	3	9
C_2	2	2	2	6
C_3	3	3	3	9

Figure 3.4: Park1 Log entries at T_0 for servers S_1^1, S_2^1 , S_3^1 and server S_1^0

At time T_1, a new vector point arrives and the log data file is updated as shown in the Figure 3.5.

T_1	S_1^1	S_2^1	S_3^1	S_1^0
C_1	3	3	4	10
C_2	2	2	2	6
C_3	3	3	3	9

Figure 3.5: Park1 Log entries at T_1 for servers S_1^1, S_2^1 , S_3^1 and server S_1^0

At time T_2, the update as a new vector point arrives shows the new log data as in the Figure 3.6.

T_2	S_1^1	S_2^1	S_3^1	S_1^0
C_1	5	5	1	11
C_2	2	2	1	5
C_3	3	3	1	7

Figure 3.6: Park1 Log entries at T_2 for servers S_1^1, S_2^1 , S_3^1 and server S_1^0

Evolution of the clustering and anomaly detection process as new log entries are arriving can be seen in the Figure 3.7 where new vector points arriving at T_0, T_1 and T_2 update the macroclusters profiles.

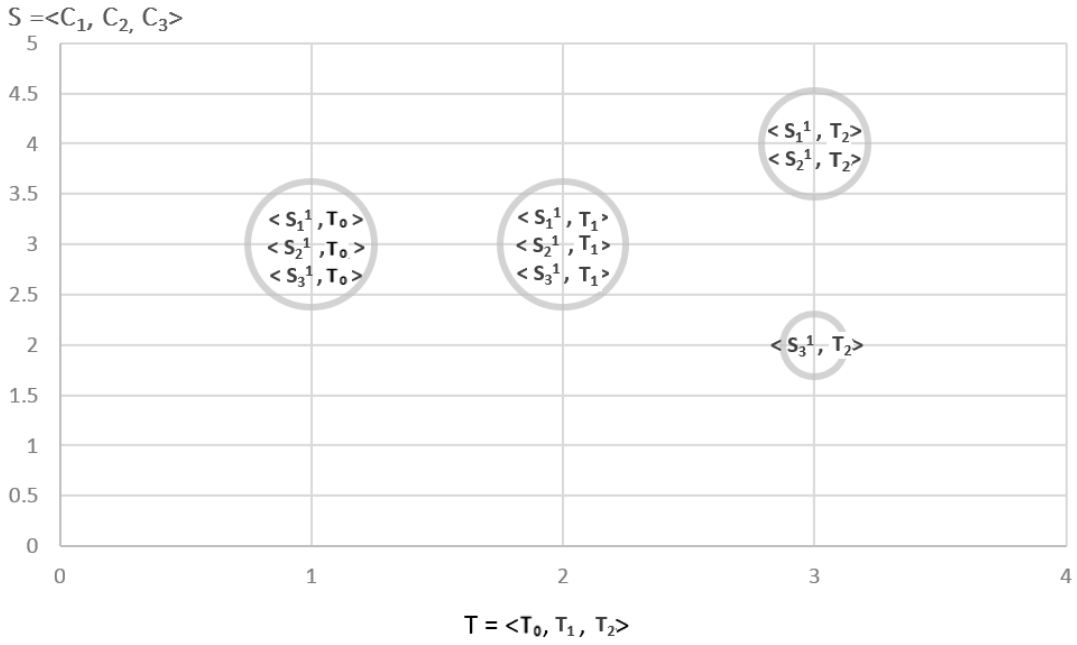

Figure 3.7: Macroclusters

At a higher level the process is thus consisting in tracking the changes in the cluster's profile as shown in the Figure 3.7.

3.3 Algorithm design

Next Log Analytics in Real Time (NLART) algorithm design needs to take into consideration all the advantages of Clustream clustering and personalize it in order to add efficient anomaly detection capabilities. In the first part this section goes through clustering definition and setup and then gives a deep understanding of Clustream algorithm later on extended in order to allow anomaly detection.

To better understand the clustering approach there is a need to specifically state some important definitions.

- Overview of window model.
  Windows in temporal data are models used to process only a slice of data at a time, they allow to avoid negative effects of concept drift [Aggarwal 2013] and they can be distinguished into:
  - Fixed Sliding Windows: Easy to implement.
  - Landmark Windows: divided into several basic windows.
  - Damped Windows: Assigns weights to data points to lessen the influence of older points and the one used in this work.

- Adaptive Windows: Extension of Fixed Sliding Windows, maintains optimal width throughout streaming process.

Sliding Windows provide a way of limiting the analysed data stream tuples to the most recent instances, the technique is deterministic, as it does not involve any random selections and prevents stale data from influencing statistics. They are used to approximate data stream query answers and maintains recent statistics. They are often called buckets and they allow to build features that are additive.

- Distance measure
  Dis/Similarity between points are computed through several types of distance measures but our approach only processes Euclidian distance. Other types of applicable distance measures include:

  - Euclidian distance that applies to any n-dimensional vector. It is used to compute distance between new point and cluster centres on one hand and to compute distance between cluster's centres on the other one.

  - There exist also in the literature other distance measures such as Minkowski (a generalization of both the Euclidean distance and the Manhattan distance), Caneberra (a numerical measure of the distance between pairs of points in a vector space and a weighted version of Manhattan distance) or Mahalanobis(a measure of the distance between a point and a distribution).

- Types of clusters
  The algorithm shown in this research processes spherical clusters.

  - Spherical Gaussian. There are some test that are executed to assert for Gaussian population in our log server dataset.

  - Arbitrary shapes (i.e. non spherical).

- Pyramidal Time Frame
  Pattern used to store information at snapshots in time.

- Cluster Centroid, Radius and Diameter
  Given N n-dimensional data points in a cluster and $i = 1, 2...N$
  Given the three terms Centroid, Radius and Diameter where the Centroid is the clusters mean, the Radius is the average distance from the objects within the cluster to their centroid and the Diameter is the average pair-wise distance within the cluster.
  Radius can be seen as root mean squared RMS deviation of the data points

from the centroid.
The algorithm maintains statistical summary information in the form of linear sum of the N data points LS and square sum of the N data points SS with respect to the elements of the $n-dimensional$ data points. This allows the algorithm to calculate the RMS deviation only making one pass at all the data points.
For example, supposing to have three data points: (1,1,2), (2,2,2), and (1,2,3). The linear sum is (4, 5, 7), the linear sum of the squares is (6, 9, 17) and N is 3

Clustream is an algorithm that allows to analyse streams in two steps.
The online step summarizes the data stream into micro clusters and stores them periodically in a structure in order to allow in the offline step an efficient search for clusters at different times or time scales.
Clustream is based on BIRCH algorithm [Aggarwal 2007] except from the fact that it does not implements the Tree structure to maintain its microclusters while allowing online learning.
BIRCH itself as proposed by [Zhang 1996] stands for balanced iterative reducing and clustering using hierarchies, it defines the Cluster Features CF such that CF is $3-tuple[N, LS, SS]$. CF structure is summarizing the statistical information required to compute the Centroid, Radius and Diameter of a cluster.
For example, suppose you have three $2-dimensional$ *data points: (1,2), (3,4) and (5,6) in a cluster* C_1.
The CF *of* C_1 *is [3,(1+3+5,2+4+6),(1+9+25,4+16+36)] = [3,(9,12),(35,56)].*
The interesting properties relies in the fat that CF_s are additive and substractive.
For example, for two given clusters, C_1 *and* C_2 *with their respective cluster features,* CF_1 *and* CF_2*, the* CF *generated from merging* C_1 *and* C_2 *is* $CF_1 + CF_2$.
Thus considering a stream as an unbounded sequence of pairs (S, t), where S is a structured vector representing a server with it attributes and $t\ \varepsilon\ T$ is the timestamp that specifies the arrival time of vector S on the stream.
Clustream microcluster extends the BIRCH CF structure by adding two temporal parameters to the CF.
The first parameter is the $n-dimensional$ linear sum of the timestamps ST of all the data points whose statistical summary information is contained in the CF.
The second parameter is the $n-dimensional$ square sum of the timestamps SST.
The triplet CF is extended to $5-tuple\ CFT = [N, LS, SS, LST, SST]$.

From the CFT, it is possible to compute the Centroid $\overrightarrow{S_c}$,the Radius r and the Diameter d of the cluster. Storing all the clusters points not needed any more. The centroid vector that can be considered as the middle of the cluster is computed as:
$\overrightarrow{x_c} = \frac{1}{n}\sum_i^n \overrightarrow{S_i}$ where $\overrightarrow{S_i}$ is the $i-th$ vector in the cluster and n the number of vectors in the cluster. The average distance from vectors to the centre of their

cluster is: $r = \sqrt{\frac{1}{n}\sum_i^n(\overrightarrow{S_c} - \overrightarrow{S_i})^2}$ and the average pairwise distance within a cluster is: $d = \sqrt{\frac{1}{n(n-1)}\sum_j^n\sum_i^n(\overrightarrow{S_j} - \overrightarrow{S_i})^2}$.
These parameters are computed online with a single pass using the CFT_s.
The ST and SST are parameters used to compute the temporal standard deviation of the cluster as follows: $d = \sqrt{\frac{1}{(n-1)}\sum_j^n(\overrightarrow{SST} - (\overrightarrow{LST})^2)}$

This summary information as stored in the microclusters are used during the offline step which depends on inputs like the time horizon or the clustering granularity.

Clustream uses a pyramidal time frame to manage efficiently the balance between the storage availability and the ability to recall summary statistics from different time horizons [Aggarwal 2007].
It is also important to notice that Clustream concept of the microcluster maintenance algorithm derives ideas from the $k - means$ and $k - NearestNeighbour$ algorithms.
The initialization phase in the microclusters creation uses an offline process. In that phase they store the first points on disk and use a standard k-means clustering algorithm. Once the initial microclusters are created, the new points are incrementally inserted in a in CFT Tree structure similarly to a $B+_{tree}$. This incremental insertion consists in finding for each new arriving vector the closest leaf entry, to add that vector to that leaf entry and to update the CFT.
If the diameter becomes higher than the maximum allowed diameter then the leaf is split. If the maximum number of leaf nodes is found then the parent is split. They define in the previously defined phases two important parameters for a CFT Tree structure: the first one known as the Branching factor defines the maximum number of children and the second one represents the maximum diameter of subclusters stored at the leaf nodes.

In this research, Cluster feature considers a $7 - tuple$ structure $CF = [N, LS, SS, LST, SST, LSCCT, SSCCT]$ where $LSCCT$ stands for linear sum of cluster creation time and $SSCCT$ the square sum of the cluster creation time. They are used to compute feature vectors for each cluster. $LSCCT$ and $SSCCT$ parameters are used for the computation of the temporal standard deviation related to Clusters creation time in order to keep track of recent updates that are inferring profile changes. $d = \sqrt{\frac{1}{(n-1)}\sum_j^n(\overrightarrow{SSCCT} - (\overrightarrow{LSCCT})^2)}$. Those new features allow to easily track for changes in the distribution of clusters and their creation time thus enabling the possibility to track for changes in the number of microclusters created.
The temporal standard deviation d computed from $LSCCT$ and $SSCCT$ are integrated in the extension of Clustream as it is shown in the Figure 3.8.

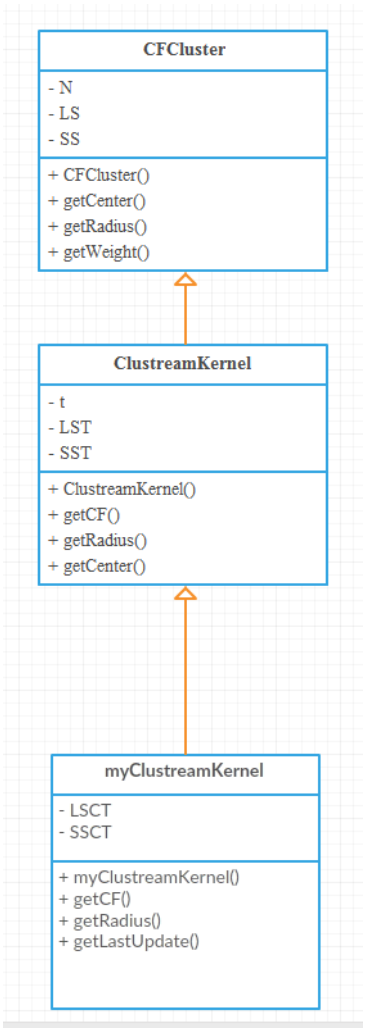

Figure 3.8: Extending Micro Cluster Kernel

Where it is possible to distinguish the different parameters of interest allowing to build a list of anomalous points.
‘ Considering the example of several three-dimensional vectors S_1(2,5,7), S_2(4,3,5), S_3(3,2,0) to S_7 arriving at t_1, t_2, t_3 in computing CF_1 :

N=3 clusters (initialization)
LS=(9,10,12,...)
SS=(29,38,74,...)
Center1 = (9/3,10/3,12/3,...)
LST=(t1+t2+t3)
SST=($t1^2 + t2^2 + t3^2$)

Notice t_1 is in the format *2015-03-24T10:30:30* that is to be converted to decimal epoch *1427193030* before operations. Focusing on the new parameters, they are computed on the flow during an intermediate micro clustering phase.
The process can be seen on the Figure 3.9.

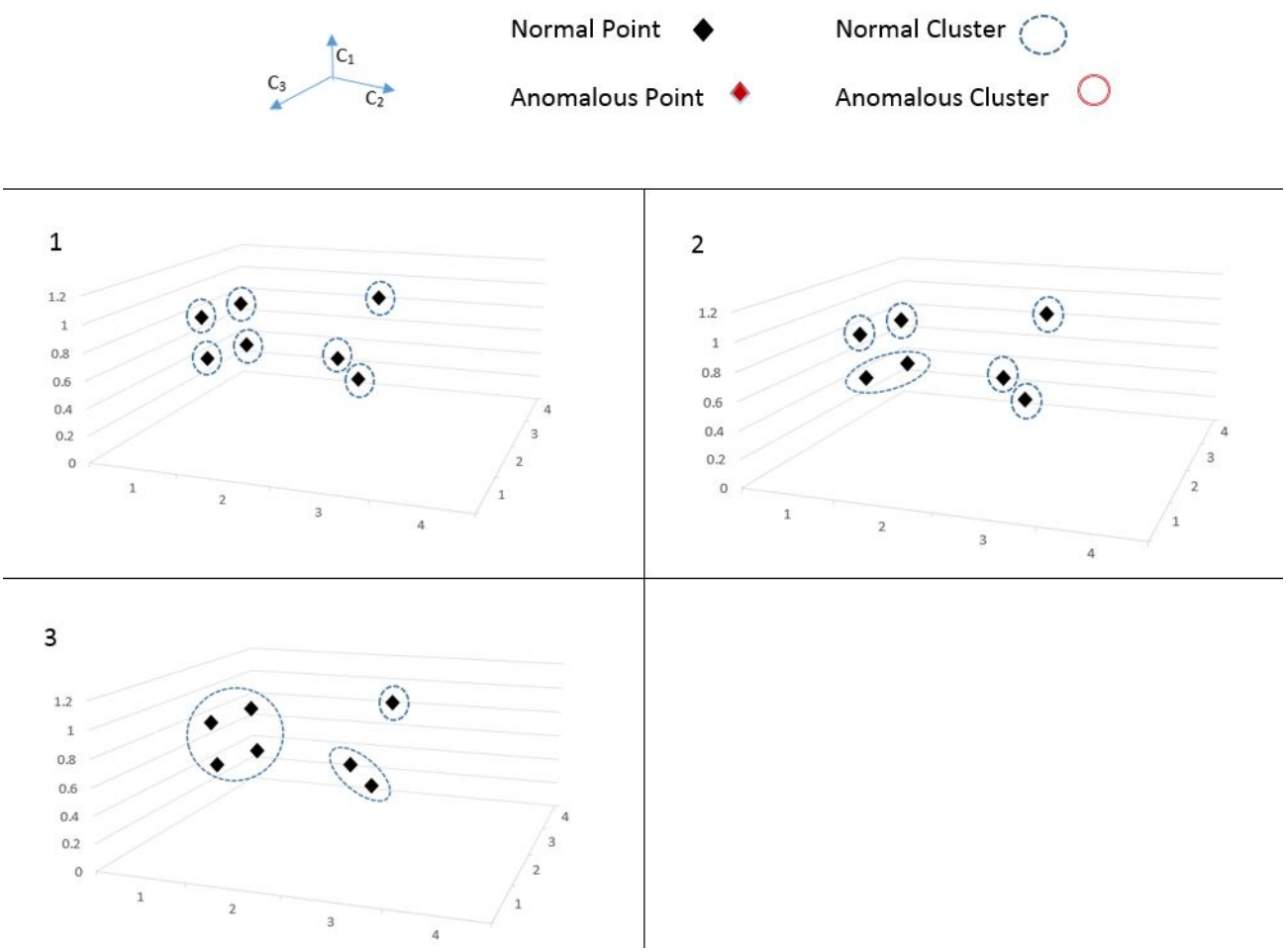

Figure 3.9: step 1

At steps 1, 2 and 3, the process goes from six Cluster's creation times to three cluster's creation times.
For the 3 clusters, the CFV are respectively:

<n=4, LS, SS, LST, SST, LSCCT=(t_1), SSCCT=(T_1)>
<n=2, LS, SS, LST, SST, LSCCT=(t_2), SSCCT=(T_2)>
<n=1, LS, SS, LST, SST, LSCCT=(t_3), SSCCT=(T_3)>

Then at steps 4 and 5 a new point arrives as shown on Figure 3.10

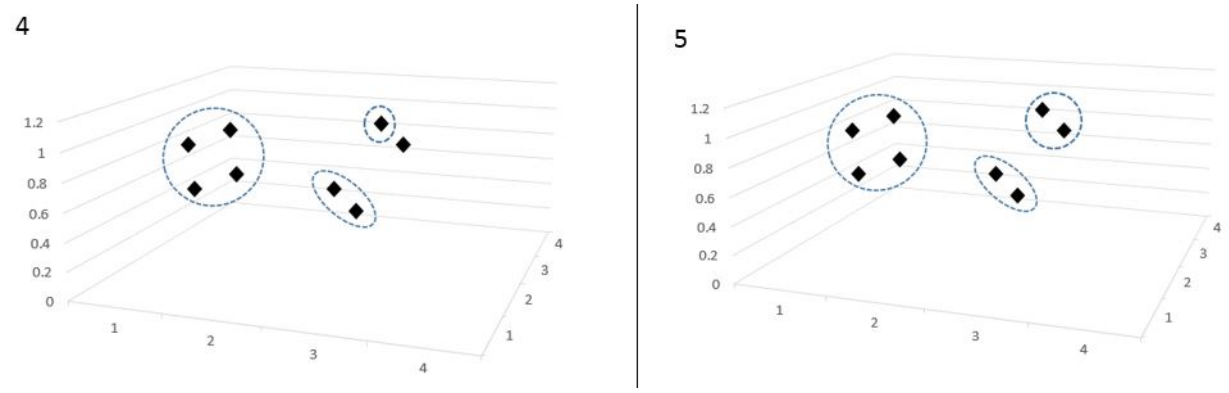

Figure 3.10: step 2

The point is merged within a previously existing cluster and the Cluster's creation times are not updated. For the 3 clusters, the CFV are respectively:

<n=4, LS, SS, LST, SST, LSCCT , SSCCT >
<n=2, LS, SS, LST, SST, LSCCT , SSCCT >
<n=2, LS′, SS′, LST′, SST′, LSCCT, SSCCT>

The number of points in the last microcluster has increased. LSCCT and SSCCT remain unchanged while all other features are updated.

Then at steps 6 and 7, a new point arrives as shown on Figure 3.11.

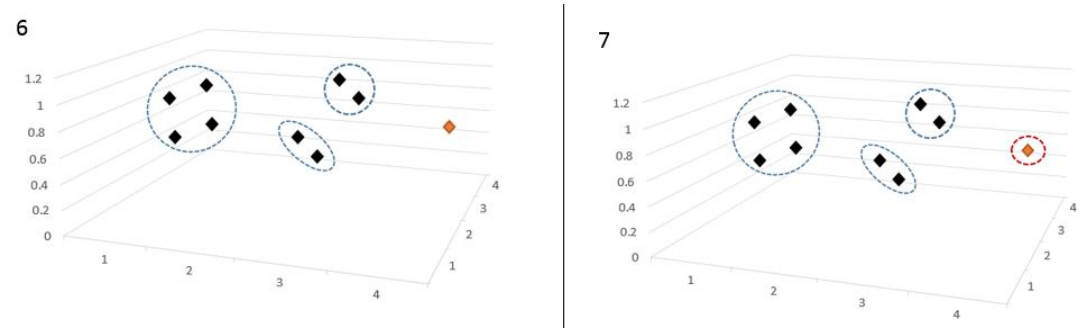

Figure 3.11: step 3

The point is not merged within a previously existing cluster and the Cluster's creation times are not updated for them. There is available room in memory and there is creation of a new microcluster. For the 4 clusters, the CFV are respectively:

<n=4, LS, SS, LST, SST, LSCCT , SSCCT >
<n=2, LS, SS, LST, SST, LSCCT , SSCCT >
<n=2, LS′, SS′, LST′, SST′, LSCCT, SSCCT>
<n=1, LS″, SS″, LST″, SST″, LSCCT″, SSCCT″>

There is a newly created cluster after arrival of the new log entry. This change which is implying a anomalous behaviour, will be reflected by a change at the cluster at the macroclustering level as shown on Figure 3.7.

NLART algorithm can be formalized as follows :

Algorithm 1 NLART algorithm

1: **procedure** NLART–ANOMALY DETECTION(LOG STREAM: L, INITIAL MICROCLUSTERS NUMBER: q, TIME HORIZON: t_h)
2: let $M_{Boundary}$ the maximum cluster boundary.
3: let t_x the current cluster creation time.
4: **for** each node data point $P \in L$ **do**
5: read data point $P(S, T_S)$
6: find closest microcluster
7: compute point to cluster boundary $C_{Boundary}$
8: if $C_{Boundary} \leq M_{Boundary}$
9: ... add_merge P into microcluster
10: ... update microcluster: centre(T_S) = f(t_h,t_x)
11: else
12: ... create new microcluster : centre(T_S) = f(t_h,t_x)
13: ... update cluster anomaly list
14: end if
15: if K-means request arrives
16: update macrocluster: centre(microclusterList, AnomalyList)
17: end if
18: **end for**
19: **end procedure**

The maximum boundary is just as defined in the Clustream framework and it allows to know about the acceptable boundary a cluster is allowed to have in order to absorb new points. The *step 4* to *step 7* iterate through microcluster List to find the closest one. The *step 8* to *step 10* apply the previously described micro clustering phase. Clustering process comes next to a normalization process on all involved dimensions. The algorithm integrates anomaly detection between *step 11* and *step 14*. The algorithm at the final step partitions data space into a cluster list space which subsequently is used to store the statistical information of the evolving data stream. Given point $P(S, t_s)$ such that vector S is the multi-dimensional record of server features and t_s is the corresponding timestamp.
The implementation of the pyramidal time frame as it is the case for Clustream in Nlart algorithm is not straightforward since there is no need to balance accuracy with storage capability. It is not essential in Next distributed computing system considering its boosted and redundant architecture with very large memory settings

available for the experimental setup but as it is the case in general for real applications it is good practice to always implement suitable memory management policy.

The Cluster Feature Vectors (CFV) are processed through microclusters characteristics and additivity properties.
In each snapshot, the number of microclusters is initialized in *step*1.
At each new vector arrival the Join, Merge Delete operations are carried out.
Add operation works by finding the nearest Microclusters through Root Mean Square and Euclidian distance computation, explaining as shown before why the clustering depends on K-nearest Neighbour.
Delete operation find oldest average timestamps of last points in CFV while Merge operation find the closest two clusters. In this step, the anomaly detection operation find the Clusters with points not belonging to previously existing clusters. The full process then groups servers by computing intermediate statistics referred to as microclusters and tracks for changes in the clusters.

In summary, the general process can be divided into three main steps.
- *Step I*
The algorithm computes intermediate cluster statistics referred to as microclusters. Whenever a new point P arrives there are three possibilities:
-P is classified as belonging to existing cluster.
-P is labelled as an outlier.
-P is representing an evolution in the stream characteristics.
- *Step II*
In both second and third cases of the first step, the new data point$P$ is not associated to any of the existing microclusters and it is requiring the creation of a new microcluster m. By adding parameters that keeps track of the variation in the number n of clusters and their creation time (LSCCT, SSCCT), it is possible to track for anomalous event occurrences and by saving snapshots of their characteristics it is possible to keep tracks of anomalies.
- *Step III*
Macroclustering into normal and anomalous groups.
During this step the microclusters are considered as pseudo points and the seeds are not random.

The approach presents several advantages since it combines positive aspects from available methods in the literature reviewed so far. It is working in a totally unsupervised manner as it does not require labelled instances in order to build the detection model. It extends a robust Clustream clustering algorithm, enabling the avoidance of usual clustering issues such as sensitivity to outliers and multiple pass on the data set. Clustream added value is the ability to achieve better efficiency, increased scalability with a reasonable power of evolution and change detection. Its intrinsic tilted framework allows dynamicity putting more weight to the most recent data, the microclustering part ensure better quality that other methods such as k-means or k-medians used alone. All these allowing the possibility to automatically

build sets of normal and anomalous events in real time.

These advantages also give the possibility to apply the algorithm to other monitoring system without heavy nor complex modification.

Chapter 4

Experiments, results and discussion

This chapter covers all the experiments conducted so far together with the observed results, their inherent analysis and the discussion for possible improvement.

4.1 Experiments and results

The experiments are done with the goal of assessing server performance tracking for anomaly and unusual behaviour. In order to internally evaluate the proposed framework, algorithm is tested against enterprise level resources. Both the clustering and anomaly detection qualities are evaluated. The algorithm is analysed in different environment configuration in terms of initial number of microclusters q,t_h and max-boundary M.

The expectations being to increase detection accuracy of server's failure or malfunctions, the experimental setup is designed as shown in the Figure 4.1.

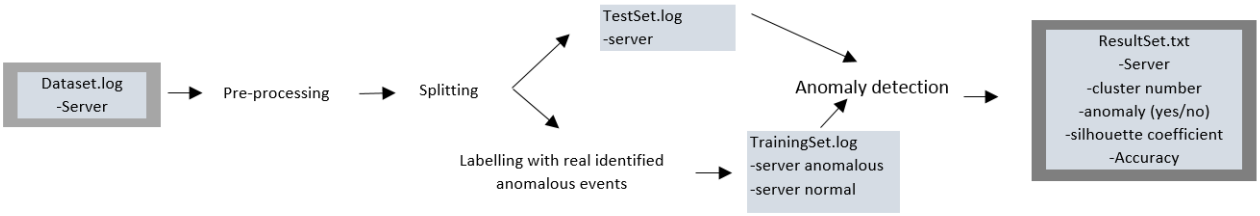

Figure 4.1: Experimental setup

It is possible to see on the figure how the input log data are pre-processed and separated into training and test set. The model is built from the training and applied to the test set. The output are the resulting Clusters, micro clusters centers and the evaluation metrics that is the silhouette coefficient for the Clustering quality and ROC Area Under the Curve with respect to True Positive Rate and False Positive Rate for the anomaly detection accuracy itself.

The improvements are then tested on the algorithm that is externally evaluated. From application server's extracted features as described on the previous Figure 3.1, the task consists in the following:

a - Algorithm Quality on Real Dataset
Test the algorithm quality and error stability when analysing incrementally Next streams of logs from 24 Hours run to 1 week.

b - General Anomaly detection Behaviour
Test the algorithm against previous techniques and Comparison of the different results.

4.1.1 Algorithm Quality on Real Dataset

For this experiment, the anomaly detection quality or accuracy with Area Under the Curve (ROCAUC) between multiple runs of Next log streams is measured. The analysis done on the error stability is performed through the clustering quality with silhouette coefficient assessing in a combine way the intra cluster homogeneity and the inter cluster separation.

There are 300.000 events in the one week dataset for one Server Park that are used for the experiment. It contains almost 77.000 records for each server's run. Experiments are performed each with different random initial centroids in the macro clustering phase. The test runs on different initial microclusters number q, time horizon t_h and max-boundary M. Each test executes the following steps:

1. Continue until no more line in the log stream files.
2. Select the next stream line.
3. Select intermediate centroids (If the first run, then use the initial centroids).
4. Execute the micro-clustering program.
5. Execute the anomaly detection program using CFV and RMS Deviation on Cluster Creation Time.
6. Execute the macro-clustering program
7. Store the intermediate Clusters generated.
8. Compute the accuracy Detection Rate(TPR) vs. False Positives Rate (FPR) when anomaly ground truth provided.

The flow of operations follows the path depicted in the Figure 4.1 where input log dataset are pre-processed. During this pre-processing step the input dataset is tested for dependence.
The Figure 4.2 shows the correlation matrix of the one week run log data.

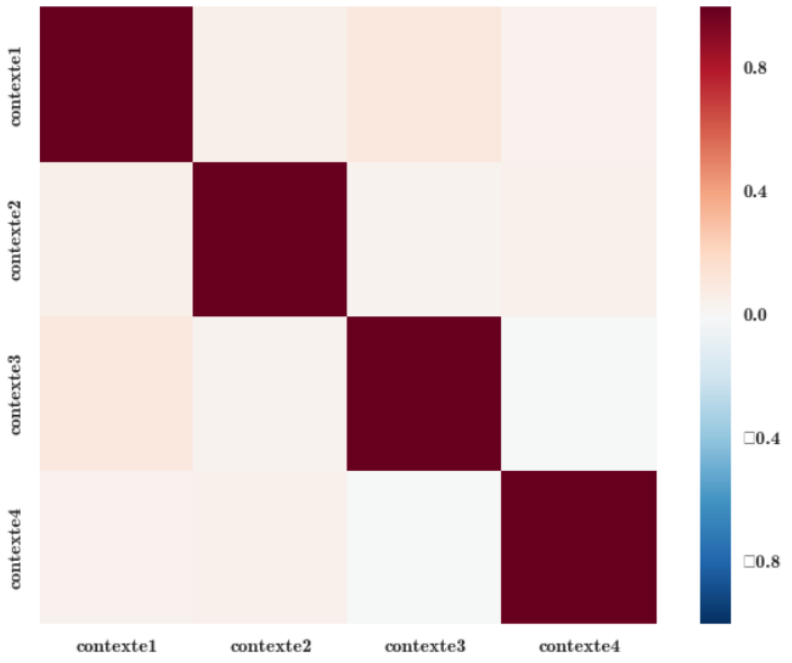

Figure 4.2: Correlation Matrix

It is possible to see that the four dimensions involved show pairwise acceptable correlation coefficients.
From that figure, the server's features correlation coefficient are ranging from 0.4 to 0.9 with a majority in the lower bound. This operation is a necessary step. It allows to assess how correlated server's features are. Highly correlated values for instance implies the fact that involved features evolve the same way and are strongly dependent which is not interesting for the algorithm that is intrinsically seeking for dissimilarity in cluster's profile to raise an alert of anomaly. The kolmogorov smirnov test on random log data sample is also conducted against normal distribution. Considering the null hypothesis H_0: The population distribution is normal, the test shows a p-value oscillating around 0.074 higher than the cut-off level that is configured in our experiment at 0.05; it thus shows that there is still an 7.4% chance that the null hypothesis is true. On the other hand being in the contest of tracking for outliers it is somehow more interesting to rather consider the correlation matrix test.

After pre-processing, both training and test files are separated; the first one with labelled classes and the other one without. In the training log file anomalous events represent the positive class (labelled with 1) while normal events represent the negative class (labbeled with 0). The Nlart algorithm is trained on the log files for increasing values of the input parameters, we start all the test initializing q to the number of points in the user define time windows. t_h is initialized at 0 as we know it is always a positive number. The training phase builds an output model

that distinguishes between the positive and the negative class in the input. Each line in the test data is then fed to the models which decides if the server belongs or not to a class. Intermediate Cluster quality is also computed through Silhouette coefficient. Silhouette Coefficient as previously said is used to compute the separation distance between the produced clusters, its analysis allows then to assess the suitable values for input parameters.

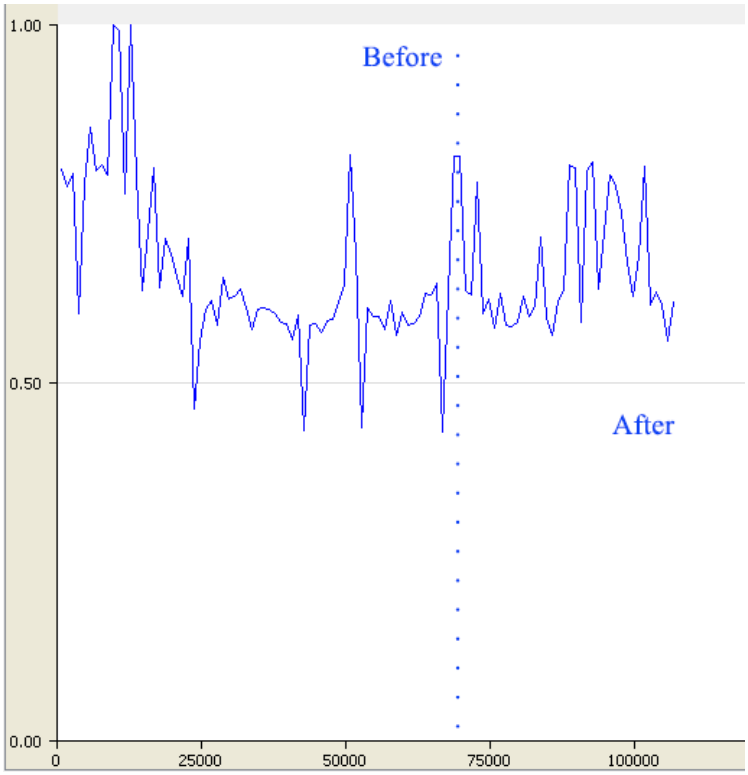

Figure 4.3: Silhouette Coefficient

Figure 4.3 presents the Silhouette coefficient evolution according to different values of those parameters during the micro and the macro clustering phase. It shows a relative stability for values of M around 10.000 while it is less sensitive to changes in q and t_h. Evolution of the Silhouette Coefficient with respect to such value of M versus timestamp shows a variation between 0.5 and 1.0 which is acceptable.

Since microclusters in Nlart algorithm summarize log stream data information, the anomaly detection result depends on clustering results and the evaluation of the stability of the silhouette coefficient was used as guidance for the choice of the most suitable values for input parameters. Figure 4.6, Figure 4.5 and Figure 4.4 present the accuracy evolution.

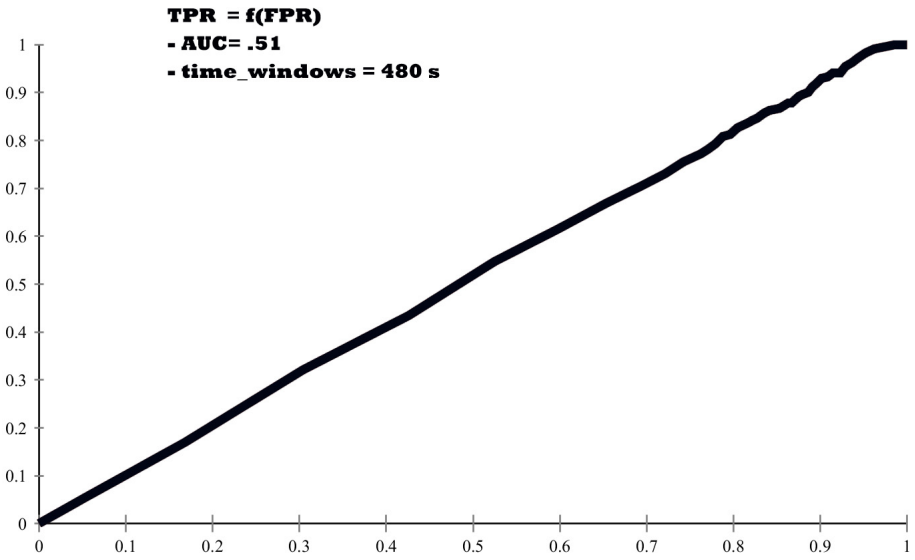

Figure 4.4: ROC AUC: TPR with respect to FPR.

This test assume fixed values for M and t_h while q is changing. The algorithm seems to show les sensitivity to the decreasing values of q. This is somehow explained by the fact that deciding the time impose a fixed number of initial points and decreasing it has a bad effect on the accuracy. Increasing it too has no effect. The curve is almost on the diagonal and the process needs improvements.

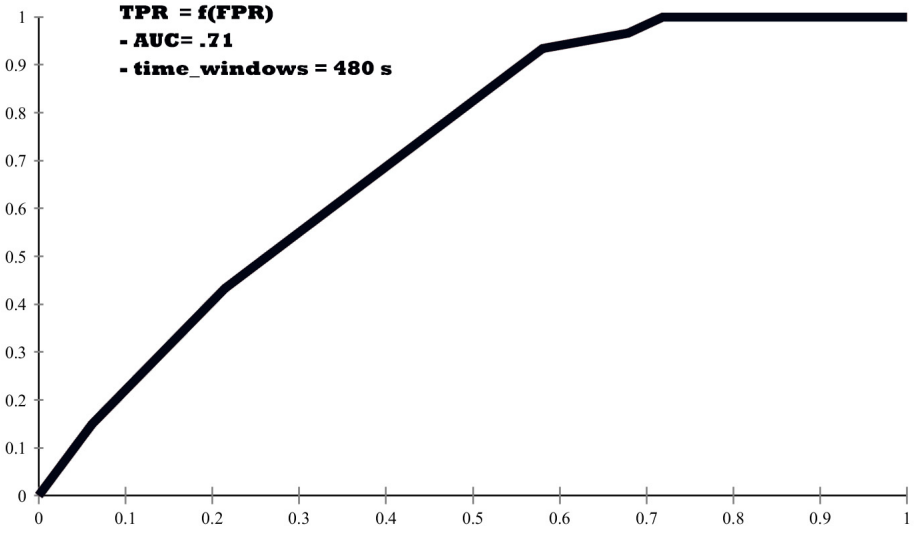

Figure 4.5: ROC AUC: TPR with respec to FPR

This test assumes fixed values for q and t_h while M is changing. The accuracy shows a sensitivity to the increasing values of M. The maximum accuracy observed is when it is between 8.000 and 11.000. For values exceeding that range the True Positive Rate is too low and this can be explained by the presence of too much overlapping microclusters. The curve in this figure is on top of the diagonal which is acceptable.

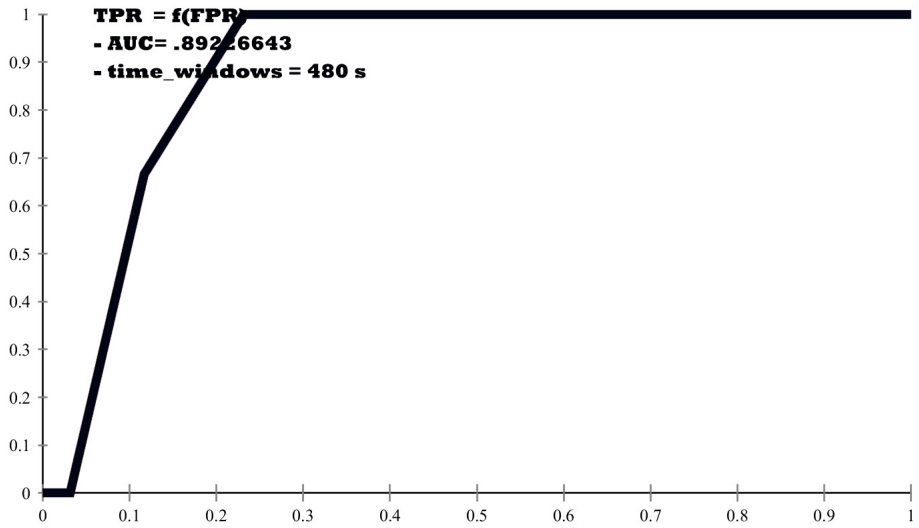

Figure 4.6: ROC AUC: TPR with respec to FPR.

This test is assuming fixed values for q and M while t_h is changing. It is possible to see the high accuracy. It is highly changing for increasing values of this parameter. By setting the time windows to 8 minutes we observed a better accuracy than for previous experiments. We realized that there is no fixed optimum t_h a priori since we were unable to really characterize the accuracy with given fixed values. It was always varying along the online learning which can be explained by the fact that anomalous event being by definition supposed to be rare they are occurring at a periodic and continuous rate in the dataset. This latter observation suggests that optimal values for this parameters are really application specific. The curve in this figure is on top of the diagonal which is good in the case of Nlart.

4.1.2 General Anomaly detection Behaviour

For this experiment, comparison of the number of detected anomalies before and after applying Nlart algorithm are done on the Next server's log.
The input dataset is made of fifteen days Log samples of real time supervision of the distributed architecture.

The experimental setup is executed just as depicted in the Figure 4.7 where all the training and testing step are removed in order to run in a totally unsupervised manner. Important settings and thresholds are derived from the previous testing and the results are shown in the Figure 4.8 where on the first time windows there occurs that the serverChild1 is labelled as anomalous while the other one are not.

Figure 4.7: Experimental setup for external evaluation.

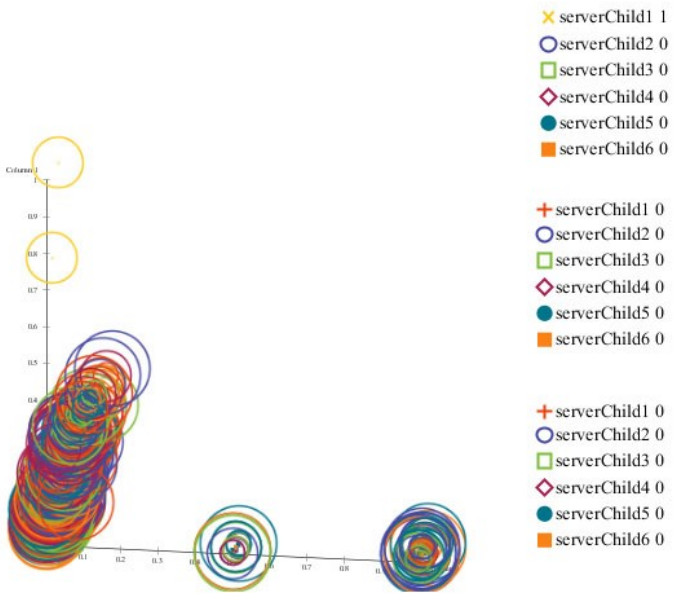

Figure 4.8: Clusters with respect snapshots of time

In Figure 4.8 it is possible to see that two serverChild1 instances are anomalous. The observation in the results as executed by domain experts in a global view suggests that the Nlart algorithm proposed allows to detect anomalies in a proportion of 3:1 of unusual events with respect to previous applied techniques.

4.2 Discussion

To assess the anomaly detection performance of Nlart, analysis are done on the impact of the different input parameters on the accuracy of the algorithm through the ROC curve.
From observed results, we can see that the algorithm keeps high detection rate with

good ROC AUC values around 0.8 when clustering phase achieve good performance that is using M values around 10.000. The different values of Silhouettes coefficient has allowed us to monitor the impact of those parameters on the algorithm. We can see that clustering quality is acceptable given the evolution of the observed coefficients. Our experiments realized good detection quality around .89 with acceptable false positive rate and given the fact that it achieved results similar to the one observed in the researches conducted in ADWICE by [Burbeck 2005] as described in Chapter 2.

Chapter 5

Conclusion and Perspectives

This work proposes a reasonable solution to early detection of servers failure in scale distributed computing environments based on real-time analytics of logs. The proposed solution is a naive based implementation of real time clustering for stream log data. It showcases models to diagnose a large and distributed computing systems, namely Next, in real time implementing control monitors. It experiments a novel application based on machine learning Clustreams methodology extending it to Nlart (Next log Analysis in Real Time). It describes Anomaly Detection based on unlabelled streaming data: given theoretical thresholds, learning from false reports, tracking anomalies in the server's logs to detect malfunction or failures in order to offers better services to end user on the BNP Paribas Banking institution website. The essential idea behind the extended model presented is to perform effective data summarization, to effectively perform tasks such as streaming data clustering and then anomaly detection. The technique therefore provides a framework upon which many other data mining tasks can be built. The thesis work starts with global presentation of the Next computing system and a quick understanding of the environment. When a new data point consisting of a server's features and timestamp arrives, its similarity to each cluster group of already available servers is computed. For the case study, Euclidian similarity measure between server instances is used, there is computed a temporal standard deviation on the cluster creation time.

Then, the research goes through the extraction of the logs from the website servers. The usage of those log data as input to the presented next log analytics algorithm in real time for anomaly detection. The enhancement of the algorithms and finally the refinement of the concept after experiments follows and analysis of the results and iteration through studies is done. The work also consists of several iterates through implementations of tools and custom scripts fed by related works in previous studies. It could be interesting to try to achieve better results in future works by using distance measures other than Euclidian distance. In streaming data intrinsically depending on the time dimension, it is clear Euclidean distance between two time dependents is subject to incorrect results when there are distortion in the time axis. [Keogh 2005] proposes a way to deal with this by using a dynamic time warping. In a naÃ¯ve approach we have assumed that there were no distortions. The anomaly detection through Nlart correctly determined that the dataset contained anomalies with acceptable Accuracy. Additionally, the Human Machine Interface is able to show in real time where and when the error occurs to allow the domain expert to understand exactly why the Next Computing system

server involved is reporting an anomaly.

Future possible improvements may also consist in testing and comparing other available algorithms in the field of log analysis in general or more particularly to test in an on-line single flow of operations integration of the extended K-means anomaly detection.

Bibliography

[Aggarwal 2007] Charu C Aggarwal, Jiawei Han, Jianyong Wang and S Yu Philip. *On clustering massive data streams: a summarization paradigm.* In Data Streams, pages 9–38. Springer, 2007. (Cited on pages 18 and 19.)

[Aggarwal 2013] Charu C Aggarwal. *A Survey of Stream Clustering Algorithms.*, 2013. (Cited on page 16.)

[Burbeck 2005] Kalle Burbeck and Simin Nadjm-Tehrani. *ADWICE–anomaly detection with real-time incremental clustering.* In Information Security and Cryptology–ICISC 2004, pages 407–424. Springer, 2005. (Cited on pages 3, 4, 7, 8, 13 and 34.)

[Chen 2002] Mike Y Chen, Emre Kiciman, Eugene Fratkin, Armando Fox and Eric Brewer. *Pinpoint: Problem determination in large, dynamic internet services.* In Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on, pages 595–604. IEEE, 2002. (Cited on pages 3, 4 and 6.)

[Chen 2004] Yen-Yang Michael Chen, Anthony Accardi, Emre Kiciman, David A Patterson, Armando Fox and Eric A Brewer. *Path-based failure and evolution management.* PhD thesis, University of California, Berkeley, 2004. (Cited on pages 3, 4 and 6.)

[Gunte 2007] Dan Gunte, Brian L Tierney, Aaron Brown, Martin Swany, John Bresnahan and Jennifer M Schopf. *Log summarization and anomaly detection for troubleshooting distributed systems.* In Grid Computing, 2007 8th IEEE/ACM International Conference on, pages 226–234. IEEE, 2007. (Cited on page 6.)

[Gupta 2014] Manish Gupta, Jing Gao, Charu Aggarwal and Jiawei Han. *Outlier detection for temporal data.* Synthesis Lectures on Data Mining and Knowledge Discovery, vol. 5, no. 1, pages 1–129, 2014. (Cited on pages 3 and 4.)

[Hill 2007] David J Hill, Barbara S Minsker and Eyal Amir. *Real-time Bayesian anomaly detection for environmental sensor data.* In Proceedings of the Congress-International Association for Hydraulic Research, volume 32, page 503. Citeseer, 2007. (Cited on pages 3, 4 and 5.)

[Keogh 2005] Eamonn Keogh and Jessica Lin. *Clustering of time-series subsequences is meaningless: implications for previous and future research.* Knowledge and information systems, vol. 8, no. 2, pages 154–177, 2005. (Cited on page 35.)

[Lavinia 2010] Andrei Lavinia, Ciprian Dobre, Florin Pop and Valentin Cristea. *A failure detection system for large scale distributed systems.* In Complex, Intelligent and Software Intensive Systems (CISIS), 2010 International Conference on, pages 482–489. IEEE, 2010. (Cited on page 10.)

[Liu 2010] Yan Liu, Wei Pan, Ning Cao and Guangwei Qiao. *System anomaly detection in distributed systems through MapReduce-Based log analysis.* In Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on, volume 6, pages V6–410. IEEE, 2010. (Cited on pages 3, 4 and 7.)

[Liu 2011] Weiguo Liu and Jia OuYang. *Clustering algorithm for high dimensional data stream over sliding windows.* In Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on, pages 1537–1542. IEEE, 2011. (Cited on pages 3 and 4.)

[Münz 2007] Gerhard Münz, Sa Li and Georg Carle. *Traffic anomaly detection using k-means clustering.* In GI/ITG Workshop MMBnet, 2007. (Cited on pages 3, 4 and 7.)

[Paribas 2013] BNP Paribas. *2013 Annual Report*, 2013. (Cited on page 1.)

[Pertet 2005] Soila Pertet and Priya Narasimhan. *Causes of failure in web applications (cmu-pdl-05-109).* Parallel Data Laboratory, page 48, 2005. (Cited on page 3.)

[Sequeira 2002] Karlton Sequeira and Mohammed Zaki. *ADMIT: anomaly-based data mining for intrusions.* In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 386–395. ACM, 2002. (Cited on pages 3, 4 and 5.)

[Shahid 2012] Nauman Shahid, Ijaz Haider Naqvi and Saad Bin Qaisar. *Quarter-sphere SVM: attribute and spatio-temporal correlations based outlier & event detection in wireless sensor networks.* In Wireless Communications and Networking Conference (WCNC), 2012 IEEE, pages 2048–2053. IEEE, 2012. (Cited on pages 3 and 4.)

[Vaarandi 2003] Risto Vaarandi*et al. A data clustering algorithm for mining patterns from event logs.* In Proceedings of the 2003 IEEE Workshop on IP Operations and Management (IPOM), pages 119–126, 2003. (Cited on pages 3, 4 and 6.)

[Xu 2009a] Wei Xu, Ling Huang, Armando Fox, David Patterson and Michael Jordan. *Online system problem detection by mining patterns of console logs.* In Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on, pages 588–597. IEEE, 2009. (Cited on pages 3 and 4.)

[Xu 2009b] Wei Xu, Ling Huang, Armando Fox, David Patterson and Michael I Jordan. *Detecting large-scale system problems by mining console logs.* In Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, pages 117–132. ACM, 2009. (Cited on pages 3 and 4.)

[Yamanishi 2005] Kenji Yamanishi and Yuko Maruyama. *Dynamic syslog mining for network failure monitoring.* In Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, pages 499–508. ACM, 2005. (Cited on pages 3, 4 and 6.)

[Yang 2008] Zhang Yang, Nirvana Meratnia and Paul Havinga. *An online outlier detection technique for wireless sensor networks using unsupervised quarter-sphere support vector machine.* In Intelligent Sensors, Sensor Networks and Information Processing, 2008. ISSNIP 2008. International Conference on, pages 151–156. IEEE, 2008. (Cited on pages 3, 4 and 5.)

[Zhang 1996] Tian Zhang, Raghu Ramakrishnan and Miron Livny. *BIRCH: an efficient data clustering method for very large databases.* In ACM SIGMOD Record, volume 25, pages 103–114. ACM, 1996. (Cited on pages 7 and 18.)

[Ziv 1988] Jacob Ziv. *On classification with empirically observed statistics and universal data compression.* Information Theory, IEEE Transactions on, vol. 34, no. 2, pages 278–286, 1988. (Cited on pages 3, 4 and 6.)

Abstract: Machine learning allows to extract interesting insights from large amounts of data for specific needs and usage. Performance, security, reliability, scalability and availability optimization are representing some of those important needs. In the context of real time servers monitoring, generated log data can be used to detect failures or break down in the system. This work describes how to perform real time log analysis in a banking distributed computing system called Next in order to detect and track those anomalies.

This thesis work proposes a Next Log Analysis Algorithm (Nlart) in Real Time for anomaly detection. In our researches, we showcase a framework that is based on a custom clustering algorithm. Log data containing records of server's features are clustered into groups of normal and anomalous entities. The corresponding data are then tagged based on the resulting group distribution, the tag is then used to raise alerts of anomaly. The implementation of Nlart in the production environment of Next shows an increase in the detection of anomalies with a ratio of 3:1 with respect to previous applied techniques.

Throughout our studies, we provide a detailed description of the algorithm, the model designed and the complete anomaly detection process. We then finally present experimental results followed by the interpretation of observed results together with the conclusion and future possible improvements.

Keywords: real time log analysis, anomaly detection, clustering.