Towards a more Robust and Efficient Failure Prediction in Cloud Systems

Master Thesis

Ignacy Skrzeczek (0116408)
Graduate School of Natural Sciences (GSNS)
Utrecht University

Dr. Nishant Saurabh
First supervisor

Dr. A.A.A. (Hakim) Qahtan
Second supervisor

Diogo Landau, MSc
Daily supervisor

19 November 2024

Abstract

Cloud systems have become crucial computing infrastructure, delivering services to diverse applications. The data center which is home to this infrastructure is prone to failure, due to dynamic nature, complexity and scale. This paper explores failure prediction systems which help to offset damage caused by such a failure. With help of machine learning techniques combined with low level monitoring data we propose a predictive solution that identifies failed jobs based on granular hardware performance data. We compare various machine learning models and test potentially novel features to find the best solution in terms of precision, accuracy, f1-score and recall. We examine how GPU metrics contribute to a prediction of jobs which make use of them. Better failure detection leads to superior failure management, reduces resource wastage and provides better service reliability.

Contents

List of Figures

Chapter 1

Introduction

Data centers are at the heart of modern IT applications. They enable both cloud-based and on-premise solutions across various scales. Emerging resource-intensive workloads (e.g. Big Data and AI-based) test the limits of existing computing facilities, already operating at an immense scale. Performance of many of such application workloads rely not only on higher throughput but also lower response time, high fault tolerance and scalability. What makes things even more obscure are the many layers of computing systems that deploy and execute diverse application workloads. While comparing high-level application key performance indicators (KPIs) gives us an idea about overall performance, uncovering some operational inefficiencies can only be achieved with fine-grained analysis of low-level metrics. Available data about low level metrics is scarce. Commercial data centers typically do not share their data, presumably for competitive and security reasons. However, scientific infrastructure such as Lisa[61] and CloudLab[15] can provide insight into data center operations.

Recently, some of the recent research has focused on understanding and analyzing operational efficiency of data centers. Ousterhout (2018)[48] delved into common pitfalls like making use of superficial measurements (looking at a CPU as a whole instead of at separate cores and threads) or confirmation bias (when measuring performance you are hoping for a certain outcome) encountered when measuring or instrumenting performance. Ousterhout further claims that the data center measurements should always run one level deeper to better understand the underlying factors behind performance. Versilius et al.(2022)[61] conducted a study utilizing a fine-grained Lisa dataset and documented a data center's performance over an 8-month period at 15-second intervals. They highlight differences between ML dedicated infrastructure and standard infrastructure (with standard infrastructure not having a dedicated GPU). ML nodes are more likely to fail their jobs and are more power-hungry. According to them, longer jobs in general have higher failure rate and this results in a waste of a significant

amount of resources, time and energy.

Another study by Maricq et al.(2018)[39] highlighted hardware performance variation using CloudLab infrastructure dataset. One of the main takeaways of their work is that the coefficient of variance of hardware performance of up to 10% can be attributed to hardware variability. They advise, whenever possible, to choose lower variance hardware when looking for statistically meaningful conclusions. Such conclusions refer to outlier servers with metrics such as disk or CPU usage which stand out among other servers.

The aforementioned research does not exhaust the potential of all datasets. More intensive researches can shed light on the job failures in the cloud. Perhaps some of those failures could be attributed to anomalies like a fail-slow hardware causing a timeout. These performance anomalies could help us to understand what contributed to the failure in job completion. Servers that executed jobs which encountered failure could also be highlighted, and fine-grained details from them can be extracted, such as number of jobs being executed on that machine at the time of failure or usage of particular resources (GPU, CPU, Memory, Disk). The subsequent submissions and re-submissions of jobs and how they act in unison has not been considered as well. The previous research [50][39][35] makes use of a very narrow set of features (CPU, Disk, Memory usage). They have proven to be good predictors of failure and performance anomaly but we believe there is room for improvement. For example many data-centers are adding GPUs to their pool of resources, and the GPU usage could support failure prediction.

In this thesis, I aim to apply machine learning, feature selection algorithms and non-parametric statistics to obtain a better understanding of the above-mentioned phenomena. The data exploration will allow us to narrow down the objectives for the datacenter operational analysis. Non-parametric statistics will attempt to draw more global conclusions about the observations made in cloud infrastructure and subsequent performance anomalies. Feature selection algorithms will help to evaluate metrics that support predicting job failures. Machine learning classification methods will attempt to validate the use of new features. To achieve the thesis goals, we will look for datasets that would validate our ideas. Execution failures have a negative impact on the services provided by a data center; this thesis will try to come up with new metrics that are good predictors of a failure.

1.1 Problem Statement

Unpredictable behaviour caused by performance anomalies can be a cause of serious problems for both cloud user and provider. They make cloud

computing among other less reliable and more costly. Limiting anomaly impact would improve many services that make use of the cloud. They are often responsible for job failures in the cloud environment. Finding metrics that represent performance anomalies would also support failure prediction. There are many approaches to tackle the problem of performance anomaly detection. The first problem is in which hardware-level metrics could enable perform efficient anomaly detection, and how to find them. Some of these metrics could become new meaningful predictors of future job failures. The second problem is what statistical, machine learning or deep learning method to choose for those metrics to gain the most accurate failure prediction. There seems to be no consensus about this, and this work will not definitively answer those questions. Instead, we will focus on one promising approach and apply it in the context of a real-world dataset.

1.2 Research questions

In the context of problem statement discussed above, this thesis aims to answer the following main research question (MRQ):

MRQ: How can we design and develop an efficient failure prediction technique for cloud-native applications and platforms ?

To efficiently answer the main research question we further divided **MRQ** into three research questions (**RQ**) and sub-questions (**SRQ**).

RQ1: What are the state-of-the-art (SOTA) failure prediction techniques in the cloud?

SRQ1.1: What features are commonly used by SOTA cloud failure prediction techniques and what are their limitations?

SRQ1.2: What datasets were previously used to test failure prediction techniques in the cloud, using the features identified in SRQ1.2?

To answer RQ1 we will embark on a literature search for novel and traditional approaches to predict failure in the cloud. We will look at among others, statistical, deep learning and machine learning approaches. For SRQ1.1 the preliminary analysis displayed limitations of current failure prediction models in the cloud. Jassas and Qusay H. Mahmoud [35] used SelectKBest, Feature Importance, Recursive Feature Elimination to choose the best features for cloud failure prediction model. They chose: job ID, task index, machine ID, RAM usage, CPU usage, disk usage, priority, and scheduling class as possible predictors. The work of Ren et al. [50] also selected CPU, memory and disk usage as features but added number of spawned containers and a number of co-located workloads on the machine as additional predictors. We believe that there are more features that could be used to enhance failure prediction in the cloud. For SRQ1.2 we

will look for performance related datasets that offer fine-grained insights into the operations of a data-center. Ideally we would like to find a dataset offering identical metrics to those we decided to focus on. This would allow us for a better validation of the chosen approach. If such data is not found, we will focus on a single dataset.

RQ2: How can we identify relevant features for cloud failure prediction that overcomes limitations of the features identified by the SOTA techniques?

SRQ2.1: Which feature selection methods, are appropriate for identifying features relevant for cloud failures based on the datasets found in SRQ1.3?

SRQ2.2: What data-granularity (based on timestamp) is necessary to accurately represent a feature for cloud failure prediction ?

To answer RQ2 we will try to find possible reasons behind job failures and features that correspond with them. We believe that literature about performance anomaly will offer us some guidance. Discovered features will have to be available in one of the datasets found in SRQ1.2. SRQ2.2 will be about determining how large can the intervals of time be, between different data-points to both efficiently and accurately represent the features. It could be that different features require various amount of time intervals for accurate representations.

RQ3: How do the new features contribute to the accuracy of a failure cloud prediction model ?

SRQ3.1: What are the outcomes of applying different ML classification techniques using the features identified in RQ2 ?

Answering RQ3 will help us to determine whether we have been successful in our search for new features used for failure prediction in the cloud. SRQ3.1 will apply various ML classification techniques based on a model created from features discovered in RQ2. We will compare those classifiers in terms of time, accuracy, precision, ROC curve to select the best one for a given dataset. If the features discovered in SRQ2.1 appear in multiple datasets used for failure prediction we could go one step further. Testing our failure prediction technique on multiple datasets would give it a stronger validation.

1.3 Literature Review Procedure

For the purpose of literature review procedure we decided on backward and forward snowballing, to find relevant advice, frameworks and studies. A lot of the papers in the field of performance anomaly detection is relatively new, especially those that make use of deep-learning methods. We looked for them by filtering google scholar by the year of publishing.

The papers that had the biggest impact on the literature review procedure were two taxonomies by Tanja Hagemann and Katerina Katsarou(2020)[28] and Moghddam et.al(2019)[45]. They were both found thanks to google scholar's search engine. Those papers collect relevant sources on performance anomaly detection, and offer a structure with which one can look at other sources. Research on the LISA dataset conducted by Verluis et al.(2023)[61] was the most recent paper found. This made it a good source for finding other relatively recent papers. Most of the papers included in the bibliography have a significant number of citations, that is above 50. There were two problems encountered while looking for sources. Some papers focused on attack like DDoS as a source of an anomaly. While this can lead to performance anomaly, and the detection mechanisms used are the same, most dataset do not support this type of research. The second problem was connected to the application layer, a lot of papers make some use of this layer either to minimise its impact or to look at the problem from the application perspective. Many dataset offers very little in terms of application information, look at the performance from a different perspective and collect different metrics. This makes research replication and comparison difficult. Literature review gave us a better overview of available methods. It also created some confusion due to the abundance of possible approaches, making the decision making process complicated.

1.4 Research method

For RQ1 we will look for traditional and novel approaches that successfully predict failure in the cloud. Among them, statistical, deep learning and machine learning approaches. For SRQ1.1 We will look at various features used in previous research to predict failure in the cloud environment. This will be a focus point of our literature review process. For SRQ1.2 We will look for and compare all recent datasets used for the purpose of failure prediction in the cloud. We will look at their similarities, how they differ and what features they offer.

For RQ2 we will delve into the reasons behind failure occurrence in the cloud. Problems such as bad scheduling, noisy neighbor problem, resource contention or insufficient resources. We will try to come up with a way to explain them in terms of features offered by the datasets found in SRQ1.2. For SRQ 2.1 Feature selection algorithms like SelectKBest[1], Feature Importance[66] or Recursive Feature Elimination[11] could help to determine whether the new features will aid the failure prediction. If it turns out that none of the new features are valid predictors, we will conduct failure prediction on tried and tested features. For SRQ2.2 selecting the right data granularity to correctly represent but not provide too much information

will be a challenge. To overcome it we plan to make us of multiple different granularity settings creating models with different characteristics. If the newly selected features allow for it, the failure detection could be tested on a different dataset.

For RQ3 we will train multiple a failure detection machine-learning models with the features chosen in SRQ 2.1. The models will be different in terms of data-granularity selected in SRQ2.2. For SRQ3.1 multiple different Machine Learning classifiers will be tested. These would include: Decision Tree[56], Random Forest[5], Naive Bayes[51], Gradient Boosting[18], XGB Boosting[8] and K-nearest-neighbour[10] and iForest[38]. They will be compared in terms of accuracy, precision and training time to highlight the classifier which works best in the context of this dataset.

1.5 Threats to validity

1. There number of studies about hardware performance measured over a long period of time is limited. Often there are many papers based on just one dataset, without the possibility to replicate the results on different data. The data either does not exist, is not available, or has different metrics. Moreover, the hardware components of the data-center change ever couple of years with totally new equipment that could produce different results. Studies from early 2010s that show hard-drive as the most anomaly prone hardware [6] could no longer hold true. All of this means that we could have serious issues with replicating our results on different datasets.

2. The inherent limitations of datasets obscure our understanding of reasons behind job-failure. In one of the datasets we came across an issue, when multiple jobs were being executed on a single machine, we did not always know how many resources are being used by any particular job. We only knew the total resource usage of that machine.

3. In the work of Marciq et al.2018[39] information about the application helps them to understand the variance that is caused by it and take it into account when measuring hardware performance. Most of the dataset do not support us with enough information to do that. This means that sometimes a failure will happen for application-specific reasons and we will not know that it did.

1.6 Timeline and milestones

The thesis project began with initial literature review on performance anomaly detection. Datasets containing metrics used in relevant research were discovered and explored. The Ethics and Privacy Quick Scan of the Utrecht University Research Institute of Information and Computing Sciences was conducted. This research was classified as low-risk with no ethics review or privacy assessment required. The next stage of the project will include applying Machine Learning approach to the dataset, after identifying key variables. The following timeline is proposed:

1. Preparing the datasets, including data wrangling and removing information that is irrelevant form the perspective of this research. The current size of the data, being over 126GB makes it cumbersome and removing some of it would support further analysis. [2 weeks]

2. The datasets proposed for this research offers many metrics and possible ways to gather meaningful insights. Most of these metrics need to be excluded, to avoid over-fitting and limit the computing power requirement. Techniques like best-first feature selection might aid the search for the most statistically powerful metrics. At this point we will have to decide how much data is necessary to represent the features. [3 weeks]

3. Applying the machine learning methods to answer the research questions and evaluating the results. [3 weeks]

4. Visualizing the results, will support the conclusions reached in the previous step.[1 week]

5. Writing the thesis, ideally it will be conducted during the whole duration of the project. This stage will also include making final refinements to the previous sections. [1 month]

Chapter 2

Backgroud

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.[46] Cloud computing is the backbone of modern society. The word cloud itself implies something see-through, abstract and available anywhere. The reality of the cloud is more complicated, it is a network of interlinked data centers where geography and political borders matter. It is a tremendously lucrative business, where scale and know-how matter. Cloud is at it's heart a business of borrowing computers, making an existing in data-center hardware an abstraction to the user. Since it's inception the offerings of cloud providers have been growing. These solutions are often labeled by the amount of control they give to the user through three service models, i.e. Software as a Service(SaaS), Platform as a Service(PaaS), Infrastructure as a Service(IaaS). To ensure the quality of service, a Service Level Agreement(SLA) is signed between user and the cloud provider. The specifics of SLA depend on the type of services the user is trying to access, where SLAs can include information about availability, data consistency, response time, security, payment, usage policy, backup and disaster recovery. One of the key reasons for occurrence of SLA violations are anomalies.

A performance anomaly in the cloud is a deviation from application or infrastructure normal performance. A Virtual Machine in a cloud center that is under utilizing its resources could produce an anomaly. This work will focus on performance anomalies, those are anomalies that impact quality of service. We will attempt to tie the performance anomaly to hardware resources at the exact moment of occurrence. We hope that approach will help us find new unique features for failure prediction in the cloud. Understanding different types of performance anomalies will support finding different features that support prediction of job-failures. In the business context failures can lead to a violation of the SLA usually leading to some

sort of monetary compensation. If a cloud provider knows that the job has a high chance of failure, he may attempt to prevent it with one of many failure mitigation policies.

2.1 Anomaly as a needle in a haystack

The size of the modern cloud infrastructure is vast and continually expanding. The amount of information about the resources resembles a haystack while the anomalies can be perceived as needles, almost impossible to find. The needle in the haystack metaphor highlights how difficult it is to find the source of poor performance within a complex system such as Cloud. The title of this subsection was inspired by a book Systems Performance: Enterprise and the Cloud by Brendan Gregg[23]. His many works propose a different approaches to troubleshooting and analyzing performance of distributed systems. Utilization, Saturation and Errors(USE)[24] is one such approach used to gain performance insights about a system. The idea behind the method is quite simple, for any given resource look for errors, utilization and saturation to better understand performance bottlenecks.

The traditional metrics such as RAM and CPU usage can be difficult to comprehend the context of such complex systems due to amount of available information in time. In one of his other works Gregg advocates for more data visualization-centered approach usage of heat maps and flame graphs[22]. All of these proposed solutions highlight that our understanding of performance data of a distributed systems is limited. This is especially true in the vastness of a cloud environment.

There are several reasons why performance and anomaly analysis in cloud has proven difficult. The secretive nature of cloud providers limits the information that are available to us. Heterogeneity of service, the multitude of offerings in the cloud makes anomalies context dependent. Multitenancy, the sharing of the same hardware between different users complicates anomaly detection. Virtualization, creating additional layers of abstraction between hardware and software can mean that the infrastructure anomalies do not manifest themselves in the virtualized environment and vice versa. The last reason, is the scale of the available data and the scaling offered by the cloud means that anomalies can sometimes be captured for a very short period of time. Detecting anomalies can lead to the development and implementation of mitigation policies or their removal. There are tangible benefits of reducing the impact of the anomalies. It helps the provider maintain the SLA and makes the data-center operations more energy efficient.

2.2 Types of anomalies

A survey conducted by Chandola et al.(2009)[7], anomalies are differentiated by types. There are point anomalies, where individual points of data are considered anomalies in the context of the rest of data. The above mentioned work mentions that this is the focus of majority of anomaly related research. Contextual anomalies, sometimes called conditional anomaly means that data is considered anomaly within a certain context. This context is defined by establishing contextual and behavioural attributes. Contextual attribute takes into consideration the neighbourhood of the data-point for example a month in which the temperature measurement took place. Behavioural are more broad and lack the specific context, for example average temperature on planet earth irrespective of the time and place. A collective anomaly is a group of data-points that by themselves might not be considered anomaly but as a collective present unusual behaviour. Chandola et al.(2009) [7] note that while point anomalies can occur in any dataset, contextual anomalies require context information and collective anomalies require data instances to be related.

2.3 Performance anomalies

Performance anomalies, are anomalies that impact the performance of a given task. These anomalies can manifest themselves at every layer that is involved in delivering the solution. The source of the anomaly could be hardware, application code, OS, the cloud platform's management of resources or some combination of those. The source of the issue is not always obvious due to many above-mentioned layers involved and dynamic nature of the problem. This is apparent with works such as that of Dean et al.(2015)[12] proposing a diagnostic tool that distinguishes between external and internal sources of anomalies.

There are many possible sources of performance anomalies, they are explored by Moghddam et.al(2019). Hardware malfunction is one of them, this means that a piece of hardware is behaving out of the ordinary and should be repaired or replaced. A resource shortage could lead to performance degradation. A system has not been provided with enough resources, due to resources being unavailable or business policy. This often leads to unexpected and unreliable behaviour. A bottleneck problem is somewhat similar to resource shortage but it is more specific. There is a single source of performance issues, for example a CPU hungry application not receiving enough processing power while all other resource requirements are met. The consequence is once again performance degradation, and unreliable and unexpectable behaviour. Resource contention could be behind some performance anomalies. Multiple users sharing physical hardware can lead

to unexpected behaviour. This can lead among others to increased latency or a performance degradation. Finally the source of a performance anomaly can be an external attack. An attack against the cloud service can disrupt part or the service with consequence to performance, availability or data integrity.

The performance anomalies can be further broken down by their behaviour pattern into fail-stops and fail-slows. A fail-stop anomaly is a task has been terminated prematurely, such failures are easier to recognize due to their binary nature. A fail-slow anomaly is more difficult to tackle, the task has not been terminated but the time of execution is significantly extended. To combat this applications are written to turn fail-slow, another option is to set an execution time limit after which the job automatically fails.

This work will focus on performance related anomalies, specifically under-performing hardware and resource contention. Any given piece of hardware can encounter anomalies that are unique to its purpose and model. For example due to malfunctioning fans the CPU could be overheating, another reason could be simple performance degradation. The CPU could also be slowing down while not being faulty. There are many interdependencies between different pieces of hardware, lack of RAM memory could be slowing otherwise healthy CPU down [65]. This further complicates localisation of the actual source of the issue. This research will attempt to uncover some of the characteristics of underperforming and shared hardware in a cloud environment.

2.4 Monitoring and data collection

Monitoring is a safety-net of a system, it allows to evaluate systems performance and react to the changing circumstances that could undermine the user experience. By collecting various system traces, key performance indicators, workload metrics we can monitor the health of our system. The collected metrics can be anything from downtime, % of memory used or number of requests over time. Some monitoring tools are built into the cloud, Microsoft Azure Monitor[43] is an example of such a service. There are also open-source tools like Prometheus that allow for monitoring of distributed systems. Tools like Microsoft Azure Monitor and Prometheus provide us with system-level metrics, if configured they can also offer insights into application-level and network-level insights. There is a lot of customization involved, from creating a dashboard that focuses on metrics that are of interest to the user, to choosing a time-interval at which the data is collected. Both cloud-provider and open-source monitoring solutions can alert the user based on rules that the user has defined. For example if a

number of requests reaches a certain threshold the user receives an email informing him about that event. Monitoring can happen at multiple layers of the cloud. Applications, infrastructure, networking and the data-center all offer separate insights into the performance of the cloud and generate their own data. This data can be combined to give us a deeper understanding of the impact of anomalies on cloud performance.

2.5 Cloud Performance Datasets

Although all cloud providers monitor and collect data about the performance of their data centers, they are not keen on sharing it. Works like that of Chen et al.(2014)[9] about failures in the Google cluster, and based on a Google dataset are a rarity. One possible explanation is cloud providers trying to keep a competitive advantage, over their adversaries. Scientific community has access to educational data-centers created for the purpose of cloud research.

In United States an example of such infrastructure is CloudLab operational since 2014, during the first four years of its existence it served nearly 4000 users to run over 79000 experiments on 2250 servers [15]. CloudLab infrastructure is behind generating a large dataset based on 835 servers that were collecting fine-grained performance data over a period of 10 months [39]. The research tried to capture hardware performance variability, they tested all hardware configurations for normality and checked variance between different components. Multiple benchmarks were used to evaluate the servers. They admit that the method applied is not perfect and the application layer could still explain some of the variability.

An example of this kind of infrastructure and the main focus of these thesis is the Lisa dataset. Lisa itself just like CloudLab is a data-center that was setup for the purpose of research. The infrastructure is located in the Netherlands, where scientists from multiple different fields made use of it while it was collecting performance data. Their identity and the type of experiments and applications they run is not disclosed. This obscures the application layer, the insights gathered include hardware, network and data-center level. It is worth noting that the metrics on Lisa and CloudLab are different making a direct comparison complicated if not impossible.

2.6 Lisa datasets

The Lisa data center is composed of 349 nodes that exist within 20 racks. There are two different type of racks, 4 machine learning rack in which every node has both a CPU and GPU. All the other racks consist of node with only CPUs. The machine learning racks can have up to 7 nodes

while the generic ones up to 32. Some of these nodes are used for entry, administration or compilation. They do not however, appear in the dataset. [61].

Over 800 distinct users made use of the data center during the monitoring. The jobs are related to the fields like bio-informatics, physics, computer science, chemistry and machine learning.[61] A job can only have one user submitting it, with a unique UserID and a GroupID that users can share. Multiple monitoring tools have been employed to gather over 350 different metrics. SLURM monitored jobs and usage of nodes. Prometheus gathered all the fine grained performance data with several additional libraries from Intel and Nvidia for additional GPU and CPU metrics.[61]

Surf-dataset has the approximate size of 126GB, stored in an efficient Apache Parquet format. It contains more than 300 fine grained metrics containing information about GPUs, CPUs, memory, network or power usage and more of the monitored machines. All of the metrics were labeled with the exact time of their capture. Time is presented as unix time stamp, with 1577660400 being the earliest timestamp and translating to Monday December 30 2019 00:00:00 CET. This data was captured with 15 second intervals meaning the next timestamp is 1577660400 + 15. The last available timestamp is 1596837585 translating to Friday August 07 2020 23:59:45 GMT CET. A lot of the metrics were not being gathered throughout all of this time, but all of them can be encapsulated between 1577660400-1596837585. This would be in total about 19177185 seconds almost 222 days, but on a rare occasion data center would stop capturing metrics. More information about this dataset can be found in the work of Versilius et al.(2023)[61].

Job-dataset has the approximate size of 601MB, stored in a csv format. It contains information about jobs that have been submitted to the data center. Columns like JobID, GroupID and UserID help to distinguish who submitted a job. Submit, Start and End establishes when a job has been submitted, when it started and when it finished. The time format is entirely different from the surf-data, 2019-12-27T09:09:30 is an example of a time given in the job-dataset. State determines whether the job has been completed successfully or an issue has occurred. Those other failed states are Cancelled, Failed, Out of memory, Node Fail and Timeout. Node-List allows us to identify what to what node a certain job was assigned. A job with r11n14 has been assigned, to the eleventh rack, and within it fourteenth node. A job can have more than one node assigned to it. Users can set a timelimit, that can be no longer than 5 days, if the job exceeds the timelimit it is stopped and gains a timeout status. Other columns are mostly redundant information, available for the sake of convenience.

To match the two datasets, time format in one of them has to be converted. Since the surf-data is so much larger, it is more convenient to

convert the job data time into unix-time format and not the other way around. A lot of consideration involves partitioning the data into chunks that are large enough to approximate the behavioral pattern of the machines but small enough to account for computing limitations and model creation.

One of the limitations of the Lisa dataset stems out of the separation of jobs and surf-data. A single machine can execute multiple jobs simultaneously, but in those cases we cannot tell how many resources does a particular job take up. The surfs dataset has information about the overall resource usage of a machine but not of a particular job. Information about collocated workloads a machine is executing could still prove to be a meaningful predictor of a failure, and the datasets offers information about that. Unlike some other datasets[53] Lisa and CloudLab do not have a predefined number of anomalies. Knowing exactly how many anomalies exist within a dataset and where they are would allow for a much deeper analysis of performance anomaly discovery approaches.

2.7 Performance Analysis

The amount of collected data often means that monitoring itself is insufficient to find performance anomalies. It can tell us that something out of the ordinary is happening, but what exactly and why can be more difficult to explain. To obtain insight from the performance monitoring data, several analysis techniques can be employed, as discussed next.

One of the common method is using Signature Based Analysis. It is an approach that first captures a behavior pattern of a application. Whenever performance metric in the application environment or the application itself changes, it can be compared to the original behavior pattern. This approach was proposed and tried by Mi et al.(2008)[42]. They measured latency of different combination of transactions. These different values were then turned into signatures, that allowed for performance comparison between different versions of the application. It is possible to create a library of different signatures that represent abnormal behaviors. Later these anomalies can be classified and detected based on their signature. The main limitations of this approach is that it is difficult to capture all abnormal behavior and that this approach requires tailoring signatures for different applications. This is because good performance depends on the specific context of the system. For example important metric could be throughput or latency both need different signatures to detect performance degradation.

Threshold Based Approach is oriented about reaching certain pre-defined thresholds. Performance anomalies often significantly increase or decrease

the resource usage, when this happens an action can be triggered. Approach like this, while very basic can offer a baseline that we can compare other solutions with. It can be combined with other types of approaches. This was the case with the work of Hong et al.(2015)[30] where they propose a solution for automatic detection of anomalies in a cloud computing system. While the solution is based on a machine learning technique called Hidden Markov Models (HMM) it also makes use of threshold-based approach. Virtual Machines monitor their own performance and send metrics to the host machine only when a certain threshold is reached. This threshold for sending samples can dynamically adjust depending on performance metrics. Combination of the HMM and threshold based approach allows to make use of both historic as well as present data to determine an anomaly. According to researchers, making use of just threshold based approach would have a higher rate of False Positives.

Control Theory is an approach grounded in mathematics, it analyses and corrects the system based on certain rules. These corrections can happen within an open loop where the changes are made based on the input, or a closed loop where changes are only proposed but not executed. In case of the closed loop controller oversees the whole process, and acts on proposed decisions. Control Theory is applied in cloud resource management for example to reduce the cost of the deployments. The work of Al-Shishtawy et al.(2013)[55] proposes ElastMan elasticity controller for Cloud-based key-value stores. The stated goal of ElastMan is to achieve Service Level Objectives (SLO) at a minimal cost by resizing service based on workload. The system uses both feedforward and feedback control. In feedback control, the current metrics are compared with specified desired metrics. Feedforward control makes use of a model that takes into account current system metrics to anticipate future. In the case of ElastMan feedforward control attepmts to anticipate future workload while feedback control corrects modeling errors. The model behind feedforward controler is binary classifier built using logistic regression[59].

Statistical Approaches or in other words probability based methods are perhaps the most varied of all the approaches. This makes them somewhat difficult to generalize. In the context of this research time series analysis is one approach that seems fitting, the data is often collected with timestamps that inform us about when an event has happened. It can help us to understand various patterns and trends within a dataset. Guigo et al.(2017)[27] made us of this approach for the purpose of anomaly detection. Another popular approach is making use of the regression analysis[16] to find correlation between different metrics. It makes use of the interdependencies between different pieces of hardware. The already mentioned in Threshold Based Approach Markov model[30] is also considered to be a statistical approach. This section does not exhaust the available statistical meth-

ods used for performance anomaly analysis, there are also approaches that make use of Bayesian statistics, logistical regression, principal component analysis and many more.

Machine Learning is different from statistical approaches in that they are not based on deterministic rules but instead devise the rules themselves from the dataset. Random forest[14] approach is one of the most commonly used . It splits the data into decision trees that trains its own model to make a prediction, at the end those predictions are merged through average or voting to find the most common outcome. Another popular machine learning approach that is more commonly used in intrusion detection then anomaly detection are decision trees. Decision trees splits possible decisions into branches, partitioning data based on the decision taken. The end of a branch is a possible prediction, this approach has been utilized for performance anomaly detection by Tuncer et al.(2017)[60]. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a clustering algorithm that groups data based on density. Density in the context of DBSCAN is the number of data points within a certain radius. Data points that don't belong to a radius are considered noise. PerfInsight[64] is an example of a performance anomaly detection system based on DBSCAN. There are many more methods used for performance anomaly detection that are cosidered Machine Learning based, Support Vector Machines, k-means, Fuzzy clustering are some of the examples.

Deep Learning methods while being very similar to Machine Learning methods make use of neural networks. These are trained to understand patterns within the given data, and are able to predict output data based on a given input. Multi-layer perceptron is one of the most popular methods, making use of neural networks to understand even a complex pattern. Ahmad Alnafessah and Giuliano Casale made use of this method to find performance anomalies in Apache Spark[3], a distributed computing system. Autoencoder approach tries to reconstruct a given dataset and put back into it's original form after a compression. Neural network is being trained to minimize the difference between the two data representations before and after encoding. Mohammad Islam and Andriy Miranskyy tried to detect anomalies within the cloud environment by making use of Gated-Recurrent-Unit-based autoencoder. Another method called Self organizing map, creates and trains a multidimensional map. Some regions of the map are associated with normal and some with abnormal performance. Stephanakis et al.(2019)[57] applied this method in the field of cloud anomaly detection. This does not exhaust the Deep Learning methods that were applied to performance anomaly detection Recurrent neural network, Convolutional neural network or Adaptive neuro-fuzzy inference system. This family of methods is perhaps the most fashionable currently.

2.8 Cloud Mitigation techniques

Cloud providers adopt policies that allow them to mitigate performance related issues in the cloud that might be caused by anomalies. In this section, we briefly discuss some of the mitigation techniques.

Overprovisioning[54] provides more resources then is actually needed for a given task. This is especially effective when a heavy load is expected, ensuring higher reliability and latency. The obvious downside to this approach is resource wastage, and increased costs. Another technique called Horizontal Scaling[44] refers to policy of adding or removing additional servers, depending on the current needs of a workload. The issue with horizontal scaling, is that it increases all available resources without specifically addressing the bottleneck. This could lead to serious resource and cost wastage. In contrast, Vertical Scaling[44] is a policy of adding or removing resources from a server, depending on the current needs of a workload. The scalability that can be achieved with just one machine is limited, and there is only single point of failure.

Load Distribution[21] ensures that no single server becomes overloaded, by allocating workload according to server's capabilities. The downside to this policy is that the load balancer can become a single point of failure and additional latency. On the contrary, Load Shedding[17] is a policy that drops or delays some of the incoming requests to maintain the overall performance of a distributed system. This comes at the cost of reduced availability, and loss of some data making the system in general less predictable.

Another common approach is using VM Migration[47], a process of transferring VMs between servers. A migration can be live, promising no disruption to the application. This is achieved by maintaining the previous infrastructure running, until the process is complete. A cold migration does not maintain the application running. A migration could be beneficial if for example the target of the migration is less utilized or the current server needs to be taken down. This process impacts the performance of both of the servers, making use of additional resources during the time of migration.

All of the aforementioned measures come with some kind of trade-offs, they might help mitigate the effects of anomalies but it comes at a cost. This cost can be reduced performance, increased costs, lower availability, loss of data etc. Anomaly detection promises to remove some of the unpredictability by addressing specific issues directly, rather then finding workarounds. Once the source of anomaly is detected, measure can be taken that fix or cover the issue. Those can be both software as well as hardware improvements. In the context of the Lisa dataset we do not know which if any of these policies were used. This could contribute to a higher failure rate and limits the scope of our research.

Chapter 3

Related work

This section discusses performance anomaly detection based related works. We used snowballing approach with the main starting points being the works of Versluis et al.(2022) [61] and taxonomies by Tanja Hagemann and Katerina Katsarou(2020) [28] and Moghddam et.al(2019) [45]. Together they provide the basis of this research, and show that the subject is relevant.

Versluis et al.[61] describe the data center and it's operations over a period of 8 months in the Lisa dataset. They elaborate on the structure of the data center, characterizing hardware resources that were measured. Over 300 different types of measurements were taken, various CPU and GPU related metrics such as their temperature, or the energy consumed. The jobs that the machines worked on are described, with information such as average completion time or job submission dates. They looked for differences between Machine Learning (ML) and traditional racks for example by comparing the number of unsuccessful jobs. The authors looked for relationships between metrics by applying Spearman[2] and Kendall[19] statistical measures. One of the correlations that the authors highlight is that server swap memory, network receive fifo, TCP open socket all correlate with GPU temperature. They explain that the data they worked with often does not meet the normality assumption, making a case for use of non-parametric statistics.

Mohammad S. Jassas* and Qusay H. Mahmoud[35] analyses job failure and prediction models in the cloud by applying machine learning techniques. The prediction model was created with help of multiple feature selection techniques (SelectKBest, Feature Importance, Recursive Feature Elimination (RFE)). The selected features for the task of predicting job failures included: job ID, task index, machine ID, RAM, CPU, disks space, priority, and scheduling class. Next step looked for the best performing ML classification algorithm considering Random Forest (RF), Decision Trees (DTs), K-Nearest Neighbours (KNN), Quadratic Discriminant Analysis (QDA), Gradient Boosting, XGBoost and Naive Bayes (NB). They tested

their approach on three different datasets: Google cluster, Mustang, and Trinity applying different combination of feature selection technique and classification algorithm. For every dataset they measured training time, testing time and accuracy. In every dataset RF and DT achieved the highest accuracy. RF had the best accuracy and recall while DT had a significantly shorter training time suggesting a more scalable solution.

A systematic review conducted by Tanja Hagemann and Katerina Katsarou(2020)[28] looks into anomaly detection in cloud environments. They reviewed over 215 publications from the 2010s, and divided methodologies into three distinct categories, machine learning, deep learning, statistical approach. Some of the approaches follow none of these methodologies, and have their own separate category. They give multiple examples of methods and scientific research that made us of them in each category. They mention that the main challenges when detecting anomalies in the cloud environment are heterogeneity of services, multi-tenancy, virtualization and dynamic scaling. They believe that application of methods such as reinforcement learning, generative adversarial networks, attention mechanisms, graph-based approaches, active learning, adversarial learning and explainable AI are interesting directions of further research. The increasing popularity of using neural-network based performance anomaly detection methods is apparent in the work.

Ghiasvand et al.(2019)[20] propose a novel mechanism that uses vicinity-based statistical anomaly detection to detect node anomalies. This vicinity could be understood as similarity between different nodes. This resemblance takes into account variables such as hardware architecture, resource allocation, physical location, and time of failures. They tested how good these metrics were at anomaly detection. None of the variables were discounted, but resource allocation did not support identification of anomalies with their data. The failure detection mechanism was tested and it achieved precision of 62% and a recall of 89%.

Resource degradation can lead to the violation of SLA in the cloud environment. A taxonomy by Moghddam et.al(2019)[45] looks at how workloads-driven and anomalies-driven performance management address the challenge of efficient cloud resource management. A system that makes use of reinforcement learning and workload prediction to scale the VMs with SLA in mind is given as an example of workload driven performance management. A different approach making use of naive Bayes classifier to identify the anomalies and come up with fixes is an example of a anomaly-driven performance management. The researchers note that cloud providers have many tools at their disposal such as over-provisioning, auto-scaling or load shedding to mitigate performance issues. The paper describes downsides that those mitigation policies come with, such as increased costs or sacrificing quality of service. The authors describe how different data sources

from various layers of the cloud are distinguished and explain how to extract insight from them.

Maricq et al.(2018) [39] work based on Cloudlab infrastructure[15] tries to improve our understanding of reasons behind hardware performance variability. They wanted to understand how much does a performance of the same hardware running the same software repeatedly varies. They designed a testing framework that was applied to create a dataset in the CloudLab environment. Their dataset includes information about the OS as well as hardware layer helping to mitigate variance caused by OS. We know that every machine was running on Ubuntu 16.04 with Linux kernel release 4.4.0-75-generic. Data was gathered over a period of 10 months based on 835 servers. The dataset was split into three parts based on the tested hardware: CPU, Memory and Disk. To compare the performance of different servers they use non-parametric Mann Whitney U[41] test and MMD test[25] to devise a method for discovery of outlier servers, that give unrepresentative results. The researchers estimated that a coefficient of variance of up to 10% can be attributed to hardware variability, while higher values indicate a problem with measuring. They mention that one of the limiting factors of research conducted in the cloud is the so called noisy neighbour problem. This means that the workloads run by one tenant can affect other tenants in a shared environment.

Ren et al.(2018)[50] looked into anomalies in co-located datacenter workloads. The research was conducted based on Alibaba dataset. Alibaba is among others a large scale cloud provider in China. They applied Dynamic Time Warping (DTW)[52] to understand how similar the nodes are. The DTW algorithm measures similarity between two sequences that might have happened at a different time or with a different speed. For the anomaly analysis and diagnosis they made us of Machine Learning approach called Isolation Forest or iForest. Isolation Forest is an approach tailored for anomaly detection. It is a decision tree that makes use of anomaly score, the idea behind it is that the anomalies are unique and few in numbers. The authors discovered that performance disparity between different machines in the cluster was substantial. System failures, unreasonable scheduling and workload imbalance were mentioned to be the main causes of performance anomalies.

Some of the recent research also focused on understanding and analyzing operational efficiency of data centers. Ousterhout (2018)[48] delved into common pitfalls like making use of superficial measurements or confirmation bias encountered when measuring or instrumenting performance. Measurements such as server request are crucial but do not answer an important question, what is the key bottleneck. Ousterhout further claims that the data center measurements should always run one level deeper to better understand the underlying factors behind performance. He gives an

example of a latency measurement for remote procedure calls, one could measure deeper by breaking down that latency determining how much time is spent where. This work of his serves as a guideline, of how to make this type of research successful.

Salman et al.(2017)[53] applies Random Forest and Linear Regression for the purpose of anomaly detection in the cloud. They found a UNSW dataset that included nine different types of attacks that often occur in the cloud. They decided to exclude some of the 49 features, by using best-first feature selection technique[33]. The algorithm looks for the features that are best at a given task, this was done to reduce complexity and overfitting. They used linear regression and random forest to see how effective those methods would be at finding the attacks within the dataset. Linear regression estimates the most fitting linear function for a given dependent variable with the information from independent variables. Random forest divides the data into decision trees that trains each tree to make a prediction, at the end those predictions are taken into account through average or voting. Random forest has outperformed linear regression at all the metrics (overall error, precision, recall) in anomaly detection. Overall the detection rate had accuracy of about 99%. Classification proved more challenging with only 80% accuracy, researchers explain this by similar patterns between different types of attacks.

Ibidunmoye et al.(2017)[32] propose a system for black box anomaly detection. The proposed solution analyses data center node performance, based on unsupervised prediction mechanism. It looks at the metrics of some of the components of a node, establishing what is a normal behaviour based on a VAR model. If metrics deviate from the model they are considered anomalies. They evaluated the system on dataset based on 100 production-level virtual machines as well as a testbed. They also devised a different approach to compare it with VAR model. The proposed VAR solution successfully detected on average 88% anomalies explaining 62% SLA violations and outperformed the alternative.

Another work by Ibidunmoye et al.(2015)[31] distinguishes performance anomalies into latency and throughput related. They provide a description of various types of anomalies, such as point, collective, contextual and pattern. They describe bottlenecks by types such as, resource saturation, resource contention and behaviours such as single, multiple and shifting. There are few examples of application-related issues like memory leak and CPU hogging. They give examples of statistical and machine learning methods used to discover anomalies like ANOVA or Decision Tree and give examples of research that used them for performance anomaly discovery.

Dean et al.(2015)[12] explains why it can be difficult to point to the source of an anomaly happening in the cloud. The origins of an issue can be traced to hardware, application, or improper resource allocation. To

address this issue they designed a PrefCompass, a diagnostics tool capable of pinning down the source of a fault in a IAAS cloud. It is based on a kernel-tracing tool to extract low level information. The information is then analysed, with metrics that are more than two standard deviations apart from the mean, being considered anomalies. This system has been tested with some popular software such as Hadoop, Cassandra or MYSQL successfully identifying both internal and external issues.

Chen et al.(2014)[9] analysed failed jobs on Google cloud cluster. Researchers highlight the amount of computing resources being wasted on failed jobs. For example a job that had either high or low priority had a three time higher chance of failure. At least 34,8% of all jobs have a considerably different resource usage depending on whether they succeeded or failed. Techniques such as k-means clustering or Mann–Whitney U test to support failure analysis were employed. They present information about the failed jobs. Researchers speculate that, by limiting the amount of possible job re-submissions they could limit the waste of resources.

Do et al.(2013)[13] describe effects of limpware, that is hardware that is performing well below its specification on resources running in the cloud. A benchmark was designed, being comprised of data intensive protocols, load stress, fault injection and general white-box metrics. They evaluated fault tolerance, on platforms such as Hadoop, Cassandra or HBase. They elaborated on how one failing component can cascade into an effective fail-slow. The predominant issues with disks that lead to slowdown failure were weak head, vibration, firmware bugs and bad sector remapping. When it comes to network infrastructure, the most common problems were broken module or adapter, corrupt packets, network driver bugs, power fluctuation. To address limplock problems the authors advocate for converting fail-slow to fail-stop, and to design application that avoid cascading.

Guan et al. (2013)[26] designed adaptive anomaly identification system in cloud data center. This solution makes use of principal component analysis, performing a linear transformation and mapping a set of data points onto a new axis. A prototype of this solution was tried on various hardware components of a Google data centers, outperforming similar systems. This difference in performance was displayed as a set of ROC curves. They mention several reasons why spotting anomalies within cloud infrastructure can be challenging. Those are dynamics of runtime cloud states, heterogeneity of configuration, non-linearity of failure occurrences and the sheer volume of data.

Javadi et al.(2013)[36] proposes a public Failure Trace Archive (FTA) for all kinds of data produced by distributed systems. With the archive they designed a toolbox and a simulator that allows for a comparative analysis between systems. This toolbox has been implemented in Matlab, and offers features such as MySQL for querying or DataTable. All of the

datasets included in FTA come from the 2000s, with the last update on the website of the archive dating back to 2020. The authors discussed how failures may be classified differently between the available datasets. It makes drawing robust conclusions and replicating them between datasets difficult.

A survey by Birke et al.(2012)[6] looks into performance of several cloud data centers with a particular focus on hardware over a period of two years. They compare workloads and resource utilization. Data visualization techniques are employed, to display trends in resource usage. They focused particularly on resources such as cpu, memory and storage. An economic analysis is introduced comparing supply and demand for types of hardware. This is done to avoid potential resource bottlenecks. By showcasing, that the demand for certain resources is increasing, their work intended to help with capacity planning of future data centers.

Research conducted by Vishwanath et al. (2010)[62] characterizes cloud computing hardware reliability. They looked at the data about inventory of machines, and the stored information about machine repairs from a datacenter. They discovered that reliability of machines on a server that has already experienced a failure is completely different from the ones that did not. The machines that already experienced a failure were more prone to further failures. Researchers observed that HDD disks were the component that were most unreliable, and most often in need of a replacement. They calculated that on average over a span of a year any given server had a chance of 8% to experience a hardware failure.

3.1 Limitations of the related works

During the literature review, we came across several issues with cloud related research. Perhaps the biggest issue stems from the secretive nature of cloud providers. We came across only two datasets provided by large-scale cloud providers Alibaba[50] and Google[26]. The Google cloud trace is at this point over 10 years old, and the infrastructure that produced it is at this point ancient. The datasets created by the scientific community like LISA[61] or CloudLab [15] have their own limitations. A fundamental difference between commercial cloud and these scientific test-beds is the visibility (monitoring, modifying, instrumenting) of raw hardware resources for every user. Some features of a commercial cloud are missing. For example, there is little insight into cloud-mitigation techniques which are at the core of a lot of successful cloud solutions. There is no up or out scaling involved, which in the context of this research could help shine some light on job failures.

Another problem is that the research based on these scarce datasets is

difficult to replicate, some variables might exist within one dataset and be absent in the other. The CloudLab offers information exclusively about CPU, Memory and Disks while Lisa also has power usage or start and end of job executions among many others. Failure prediction based on Alibaba trace made us of the number of containers spawned on a machine for the purpose of failure prediction, but this information is not available with LISA.

The most important finding, which will be the focus point of this research are the features selected for failure prediction. In most works [35],[50] making use of Machine Learning for failure prediction in the cloud the features selected for the model were CPU, Memory and Disks usage. While all of these features are successful at failure predictors, we believe that there is space to find such features. For example, today more and more data-centers are making use of GPUs and perhaps some of their resource metrics could enhance future failure prediction model.

Chapter 4

Architecture

Failed jobs contribute to resources wastage of a data-center. This is especially true for tasks involving a GPU which often take a long time and have a higher power consumption [61]. Previous research concerning the failure detection systems often does not take into account the prevalence of a GPU in a modern data-center. It uses a relatively narrow set of features to attempt a failure prediction. These features sometimes fail to capture various performance anomalies that can impact job success. The proposed architecture will support addressing these concerns, as it was conceived for that purpose.

Figure 4.1 describes high level architecture of a failure prediction solution. Metric data essential for the solution is gathered during performance monitoring which is outside of the scope of this thesis. This is the first indispensable component of the architecture. The historical or real time data captured in performance monitoring is the second module described in the figure.

Next block encapsulates feature separation. Captured features need to be separated, transformed, ordered or combined. What is needed depends on how the performance metrics are arranged and how the data should be structured for the purpose of Feature Selection and Predictive Modeling.

The next component describes various granularity settings which can be selected. One of the ways to achieve coarse granularity is by increasing time intervals between different data points. This would help create a solution where scalability is more important than a very high accuracy of prediction.

The 'Feature selection' block describes the application of techniques which help find the best features for our predictive solution. Insights about the impact of performance anomalies on the low level hardware metrics support the selection of candidate feature groups. This can entail application of feature selection methods such as Recursive Feature Elimination[11] or Select K-Best[1]. The main benefit of feature selection is that it can remove

Figure 4.1: High-level architecture overview

the features which do not contribute to a successful prediction, leading to a more compact solution. Moreover it can help to tackle the problem of overfitting.

The final architecture component which concerns this thesis is called predictive modeling. Those models can be Deep Learning or Machine Learning classification among others. This stage concers how these models compare to each other and what are the trade-offs between them. For example, we can compare various Machine Learning classifiers such as Random Forest[5], Gradient Boosting[18] or Logistic Regression[37] and look for the one which satisfies the key performance indicators of any particular solution. These indicators can be among others accuracy, precision and computing time.

The final block of the architecture figure is called action and encapsulates what happens when a job failure is predicted. Mitigating actions such as load distribution [21], horizontal and vertical scaling[44] or job cancellation can be applied to prevent wasting of valuable resources. How to choose an appropriate action for a given solution is outside of the scope of this thesis.

Chapter 5

Method

The aim of this master thesis was to look into other possible predictors of job failure in the context of a large scale cloud-like data center. To test the relevance of those features a recent low-level metric dataset originating from a data-center has been found. A Machine-Learning failure detection model was built based on features from the above-mentioned data center.

This chapter explains how data had to be transformed to perform Machine Learning feature selection and classification. There were more than 300 possible features to choose from in the dataset. To select the feature sets used for the Machine-Learning from a review into the state of the art has been conducted. This was supported by feature selection techniques such as SelectKBest[1] or Recursive Feature Elimination(RFE)[11] which helped to create different settings for the Machine Learning model. Different granularity settings were applied to highlight the trade-off between accurate prediction and scalability of the solution. Multiple Machine Learning classifiers like Random Forest[5] or k-Nearest Neighbor[10] were trained on different combinations of features, to look for a stand-out solution in the context of the dataset.

5.1 Feature separation

To conduct a Machine Learning job failure prediction we needed information about the jobs and low-level machine metrics captured during the job execution. We chose a dataset originating from Lisa data center which met all above mentioned requirements. Lisa datasets had the job data and metric data separated. Job data included information about the job start-time and end-time and the rack and node number which were responsible for execution. Metric data included information about various metrics captured in 15 second intervals, and which machines were responsible for execution.

Figure 5.1 and 5.2 describe initial data transformations. The datasets

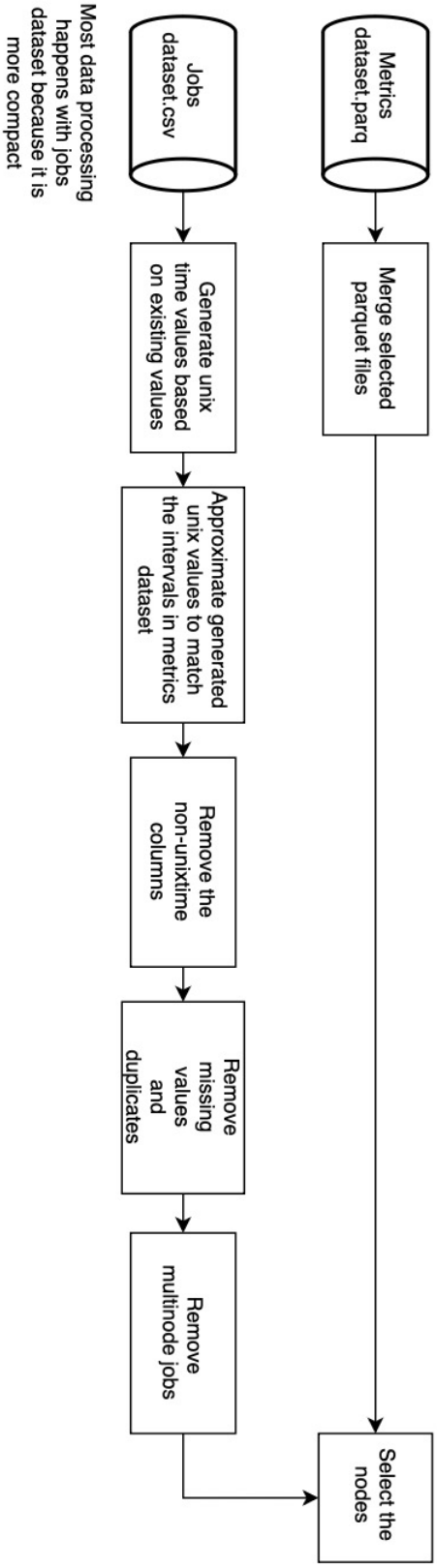

Figure 5.1: Data transformation 1

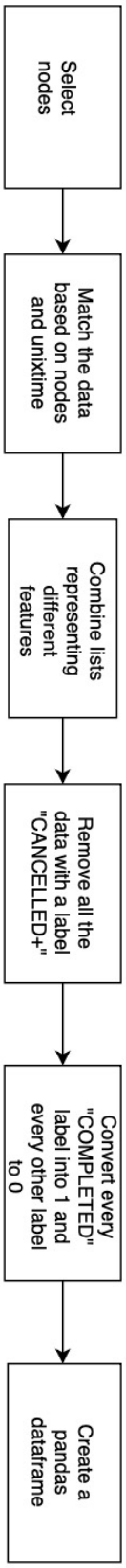

Figure 5.2: Data transformation 2

used made use of different time formats. To align the job data with the metric data the start and end times had to be transformed into unix time with a support of a Python library. Multi-node jobs were removed to reduce the initial complexity. Nodes supported by a GPU were separated from regular jobs by filtering for GPU specific job partition. Disruptive jobs with missing values and canceled jobs were removed. After these operations the amount of jobs went from the initial number of 1857495 to 37449 this is about 2%. The main reason for this large loss of data is that the job dataset starts before the metrics dataset and we can only match the part where the time of the two datasets align. Another source of data loss is that jobs that took less than 15 seconds could not be approximated and had to be removed. Canceled jobs are not always jobs that would fail and proved to be very difficult to predict, a decision was made to remove them. Finally the dataset was updated at a later date with adjusted node timestamp lags. At that time work has progressed too far to make use of the update. Perhaps it was possible to retain more data, but 37449 events seemed sufficient for the task.

5.2 Grouping and structuring of performance metrics

The metrics dataset had 328 unique features, but some of them were captured after the job data stopped being registered and thus could not be used. The features can be grouped into the following categories:

- Memory metrics
- Disk metrics
- Network metrics
- File system metrics
- GPU metrics
- CPU metrics
- System metrics
- Load metrics

During initial testing we found out that selecting features would prove to be a challenge. With 328 features the number of combinations that could be tested with feature selection techniques was overwhelming. Due

to insufficient computing power, it would be impractical to use feature selection techniques alone.

We decided for a hybrid approach where we would narrow down the feature pool with the help of research insights. During the process of literature review we discovered that SoTA features used for failure detection were disk, memory and CPU usage[35]. One of the limitations of the dataset we used that became apparent was that the CPU usage data was missing, limiting us to information about disk and memory. There were multiple disk and memory related metrics and we decided to include all of them.

Research about anomaly detection [14] pointed us in a direction of network metrics and we decided to include some(which) of them. Those were features such as node forks, node netstat tcp in errs.

For the ML jobs we would test all of the GPU metrics in combination with the state of the art to see which one performed best. The data about Graphic cards was nested as any given node had access to at least 4 GPU's. The available features were:

- nvidia gpu temperature celsius
- nvidia gpu fanspeed
- nvidia gpu power usage milliwatts
- nvidia gpu memory used bytes

To make it compatible with the rest of the metric used for prediction we only make use of the highest value among all of the cards at a particular timestamp moment. All of the above mentioned GPU features supported this information reduction.

5.3 Granularity setting

Lisa dataset captures metrics in 15 second intervals. In many use-cases this might prove to be too much data to be processed. One of the ways to make the solution more scalable was to reduce data input to every n intervals. This however, came at a significant cost to the prediction accuracy. Another approach aggregated the metrics for every instance into maximum value, minimum value, standard deviation, and mean value. This effectively diluted every feature into four separate ones. With this approach feature selection looks at the aggregated values instead of features themselves. It could for example select feature1-min and feature1-std to be relevant but exclude feature1-max and feature1-mean. While this did impact precision and accuracy, the decrease was relatively small.

5.4 Feature Selection Techniques

Feature selection proved to be difficult in the context of Lisa dataset. This was due to flattening of the features. This made feature selection techniques like Recursive Feature Elimination[11], Random Forest[5] feature importance, unreliable. They would credit the first feature in the data frame with inflated importance. To overcome this issue a model agnostic permutation importance[4] was applied. It mixes feature values to see whether the model performance suffers from it. The assumption behind this is that, if a feature is a good predictor shuffling values should degrade model's performance. The downside of this approach is that it is computationally heavy task in the context of a large dataset. Even after applying permutation importance, the results were fragile. Random Forest and RFE selection techniques like were applied to discover best performing features.

Some feature metrics had a different data-structure, with information being nested. This was the case for GPU metrics where every GPU under a node had his own metrics being captured. For every such case there was a need to transform the data into something consumable by the Machine Learning pipeline. For the GPU metrics we created a list based of the maximal value among the graphic cards working on a job at a given time. This was time consuming, we did this because we felt that GPU metrics were critical to this research, but when other metrics required a unique transformation, they were skipped.

Given a sufficient computing power we could find the optimal set of features for failure prediction. This proved not feasible with available computing power, especially since permutation importance made the task more difficult. Instead we used permutation importance and feature correlation ad hoc to look which features carry similar information about performance. During this process we discovered that for standard jobs Node Memory Mapped can be replaced by node netstat tcp inerrs and outperforms it.

5.5 Machine Learning classification

Figure 5.3 describes the data transformations leading to the Machine Learning classification. To be able to perform Machine Learning based failure prediction we have to define what a failure and success are and turn it into a binary 0 and 1. To determine this we looked at the column Status in the jobs dataset. Initially everything with a status 'COMPLETED' was a 1 and everything else a 0. During the initial testing it became apparent that jobs with status "CANCELLED+" are difficult to correctly predict. Their cancellation can have nothing to do with underlying hardware and be dictated by a different factor. A decision was made to exclude them for the dataset. The initial pandas dataframe has the following structure

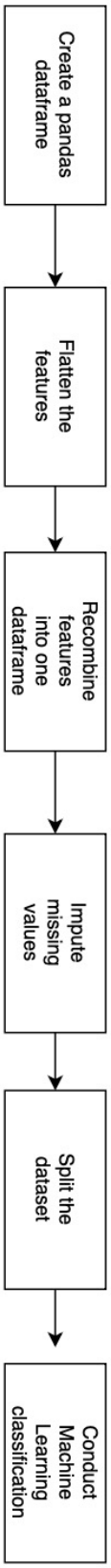

Figure 5.3: Machine Learning Classification

[label, features1, features2, features3]. Label contains a 1 or 0 determining whether a job has been completed or not. Feature 1-3 contains a list of metrics with 15 second intervals from a machine which executed a given job.

To ensure compatibility with Machine Learning models the features within a dataframe had to be flattened. Flattening transforms information which was nested in the lists into a one dimensional format. Features were separately flattened and reassembled into a dataframe. This came at a cost to Machine Learning Feature Importance which became difficult to interpret because of this operation. Some of the Machine Learning classifiers require the data to be complete and include only numbers. This was not always the case with the metrics data, and some of the data had to be imputed. The values were imputed based on 'mean' strategy. This meant that if a value was missing, the mean value of the list would be inserted in its place. 'Median' and 'Most frequent' strategies were also tried but the difference between them did not seem to impact Machine Learning Classification. For the purpose of Machine Learning data is split into a training and testing sets. The training set contains 80% of the feature while the testing set has 20%. This proportion is considered to be common, no tests involving different splits were conducted. The next step defines Machine Learning classification models these included at various points Random Forest[5], Decision Tree[56], SVM[34], Gradient Boosting[18], K-Nearest Neighbor[10], AdaBoost[63] and Logistic Regression[37]. To offset the imbalance between the number of completed jobs and failed jobs balanced class weight were applied. This method is based on comparing the ratio of 0's to 1's in the dataset, this value is then used to offset the class imbalance. Not every Machine Learning classifier supports balancing the class weight.

$$w_c = \frac{n}{k \cdot n_c}$$

where:

- w_c is the weight assigned to class c,
- n is the total number of samples,
- k is the number of classes,
- n_c is the number of samples belonging to class c.

Chapter 6

Results

This chapter will contain the results of tests conducted on many failure predicting models. It looks into the impact of using different combinations of features and how they influence failure prediction. It delves into the differences between standard node and GPU node and how the latter is more difficult to correctly predict in the context of this dataset. A section of this chapter is dedicated to comparing Machine-Learning classifiers in terms of their accuracy, precision, recall and F1-Score. Different granularity settings are compared, to understand how they affect the models.

All of the below mentioned experiments were executed locally on an M1 MacBook Pro with 16GB of ram and 512GB of memory, operating on MacOS 14.6.1. The data was stored on the disk, the format of the data was parquet for metrics and jobs as a csv. The tests were conducted on Python 3.11.3 with help of libraries such as Pandas[40], Numpy[29], Pyarrow, Datetime, Sklearn[49] and Matplotlib[58].

6.1 Results for feature groups

The following tests were conducted on different sets of compute nodes. Table 6.1 shows which nodes were used and how many jobs there were in total.

Figure 6.1 describes machine learning classification conducted on nodes without access to GPU, without applying aggregation. Random Forest and Histogram based Gradient Boosting classifiers were applied. They were trained on sets of 2-3 features. Every classification included the same set of 352 successful and 151 unsuccessful jobs. Knowing that the state of the art solutions make use of Memory and disk usage, multiple tests on various disk and Memory features were attempted. In terms of accuracy and precision Node Memory Mapped has consistently outperformed metrics like Node Memory Active, Node Memory Inactive, Node Memory Free and Node Memory Cached. Feature importance showcased that for these

Nodes used for GPU jobs(99)	**Nodes used for standard jobs(503)**	**Nodes used for ML classifiers comparison(920)**
r31n4	r10n23	r10n23
r30n7	r13n22	r13n22
r33n4	r10n3	r10n3
r33n5	r11n5	r11n5
r30n1	r26n26	r26n26
r30n6	r14n26	r14n26
r31n1	r14n8	r14n8
r31n2	r11n31	r11n31
r31n3	r25n32	r25n32
r32n1	r27n7	r27n7
r32n2	r14n25	r14n25
r32n3	r10n25	r10n25
r32n4		r13n1
r32n5		r12n8
r32n6		r13n28
r32n7		r12n8
r33n3		r14n29
r33n2		r14n27
r31n5		

Table 6.1: Nodes used for tests and a total number of jobs among these nodes

jobs Node Memory Active could be replaced by Node Netstat Tcp InErrs. Again in terms of accuracy and precision Node disk Bytes Written have outperformed Node disk Bytes Read. Node forks turned out to be the best supporting feature, adding a little bit of extra accuracy to almost every model. The best solution with Random Forest and Histogram based Gradient Boosting accuracy of 0.96 consists of Node Netstat TCP InErrs, Node disk Bytes Written and node forks.

Figure 6.2 describes machine learning classification conducted on nodes without access to GPU, when aggregation was applied. This aggregation transformed every list of feature metrics of a job into standard deviation, mean value, maximal value and minimal value of the list metrics. Random Forest and Histogram based Gradient Boosting classifiers were applied. They were trained on many sets of 2 to 3 features. Every classification included the same set of 352 successful and 151 unsuccessful jobs. The main takeaway from these tests was that features which were the best for non-aggregate model did not perform best with aggregated features model. There is a visible decrease in accuracy when compared with non-aggregate

RF	HIST_GR	Feature1	Feature2	Feature3
0.92	0.93	Node_Memory Mapped	Node_Disk Bytes Written	Node_disk_bytes_read
0.94	0.95	Node_Memory Mapped	Node_Disk Bytes Written	node_forks
0.93	0.94	Node_Memory Mapped	Node_Disk Bytes Written	surfsara_power_usage
0.93	0.94	Node_Memory Mapped	Node_Disk Bytes Written	node_network_receive_errs
0.93	0.94	Node_Memory Mapped	Node_Disk Bytes Written	node_intr
0.93	0.94	Node_Memory Mapped	Node_Disk Bytes Written	SKIPPED
0.92	0.93	Node_Memory Mapped	node_disk_bytes_read	SKIPPED
0.92	0.92	Node_Memory_Active	Node_Disk Bytes Written	SKIPPED
0.92	0.94	node_memory_Cached	Node_Disk Bytes Written	SKIPPED
0.92	0.92	node_memory_Inactive	Node_Disk Bytes Written	SKIPPED
0.92	0.92	node_memory_MemTotal%20-%20node_memo	Node_Disk_Bytes_Written	SKIPPED
0.82	0.82	node_network_transmit_errs	node_memory_Dirty	SKIPPED
0.92	0.93	node_memory_MemFree	Node_Disk Bytes Written	SKIPPED
0.93	0.94	Node_Memory Mapped	Node_Disk Bytes Written	node_memory_HardwareCorrupted
0.94	0.94	Node_Memory Mapped	Node_Disk Bytes Written	node_memory_MemTotal%20-%20node_memory_M
0.95	0.95	node_load1	node_disk_bytes_written	node_forks
0.94	0.95	node_load1	node_disk_bytes_written	node_netstat_Tcp_InErrs
0.95	0.96	surfsara_ambient_temp	node_disk_bytes_written	node_netstat_Tcp_InErrs
0.95	0.96	surfsara_ambient_temp	node_disk_bytes_written	node_forks
0.94	0.94	surfsara_ambient_temp	node_netstat_Tcp_InErrs	node_forks
0.95	0.96	node_disk_bytes_written	node_netstat_Tcp_InErrs	node_forks
0.95	0.96	node_disk_bytes_written	node_netstat_Tcp_InErrs	node_procs_blocked
0.94	0.95	node_disk_bytes_written	node_netstat_Tcp_InErrs	node_procs_running
0.95	0.96	node_disk_bytes_written	node_netstat_Tcp_InErrs	node_memory_HardwareCorrupted
0.95	0.96	node_disk_bytes_written	node_netstat_Tcp_InErrs	node_network_transmit_errs
0.95	0.95	Node_Memory Mapped	Node_Disk Bytes Written	node_forks
0.96	0.96	node_forks	node_disk_bytes_written	node_netstat_Tcp_InErrs
0.95	0.96	node_netstat_Tcp_InErrs	node_disk_bytes_written	SKIPPED
0.92	0.93	node_netstat_Tcp_InErrs	Node_Memory Mapped	SKIPPED

Figure 6.1: Tests for standard jobs with no aggregation

RF	HIST_GR	Feature1	Feature2	Feature3
0.93	0.91	node_netstat_Tcp_InErrs	node_disk_bytes_written	node_forks
0.94	0.93	node_memory_Mapped	node_disk_bytes_written	node_forks
0.94	0.93	node_memory_Inactive	node_disk_bytes_written	node_forks
0.93	0.91	node_memory_Mapped	node_disk_bytes_written	node_netstat_Tcp_InErrs

Figure 6.2: Tests for standard jobs with aggregation

models, however it was not substantial.

Figure 6.3 describes machine learning classification conducted on nodes with access to GPU without applying aggregation. Random Forest and Histogram based Gradient Boosting classifiers were applied. They were trained on 2-3 feature sets. Every classification included the same set of 68 successful and 31 unsuccessful jobs. The precision values are lower for the jobs involving a GPU, they seem to be more difficult to accurately predict. For the non-GPU jobs metrics precision values were around 0.92 while for GPU jobs the value are closer to 0.84. We did not notice a meaningful difference between the classifiers, sometimes Random Forest performs better and at other times Histogram based Gradient Boosting performs better.

It is difficult to draw robust conclusions because of the low number of data supporting the conclusions. It seems that the GPU metrics improved the model, and it is possible that temperature of GPU is the best indicator of failure. Final observations for these jobs is that node netstat tcp in errs performs worse in comparison with node memory mapped while for the normal jobs the opposite was true.

RF	HIST_GR	Feature1	Feature2	Feature3
0.85	0.81	node_netstat_Tcp_InErrs	node_disk_bytes_written	node_forks
0.86	0.86	node_forks	node_disk_bytes_written	nvidia_gpu_temperature_celsius
0.82	0.82	node_netstat_Tcp_InErrs	node_disk_bytes_written	nvidia_gpu_temperature_celsius
0.81	0.81	node_netstat_Tcp_InErrs	node_disk_bytes_written	nvidia_gpu_fanspeed
0.83	0.86	node_forks	node_disk_bytes_written	nvidia_gpu_fanspeed
0.86	0.82	node_forks	node_disk_bytes_written	nvidia_gpu_power_usage_milliwatts
0.87	0.89	node_forks	Node_Memory Mapped	nvidia_gpu_temperature_celsius
0.85	0.8	Node_Memory Mapped	Node_Disk Bytes Written	SKIPPED
0.85	0.8	Node_Memory Mapped	Node_Disk Bytes Written	SKIPPED
0.84	0.88	Node_Memory Mapped	Node_Disk Bytes Written	nvidia_gpu_temperature_celsius
0.85	0.86	Node_Memory Mapped	Node_Disk Bytes Written	nvidia_gpu_power_usage_milliwatts
0.87	0.85	Node_Memory Mapped	Node_Disk Bytes Written	nvidia_gpu_fanspeed
0.84	0.85	Node_Memory Mapped	Node_Disk Bytes Written	SKIPPED
0.83	0.83	Node_Memory Mapped	Node_Disk Bytes Written	nvidia_gpu_memory_used_bytes
0.85	0.8	Node_Memory Mapped	Node_Disk Bytes Written	SKIPPED
0.84	0.88	Node_Memory Mapped	Node_Disk Bytes Written	nvidia_gpu_temperature_celsius
0.85	0.86	Node_Memory Mapped	Node_Disk Bytes Written	nvidia_gpu_power_usage_milliwatts
0.87	0.85	Node_Memory Mapped	Node_Disk Bytes Written	nvidia_gpu_fanspeed
0.84	0.85	Node_Memory Mapped	Node_Disk Bytes Written	SKIPPED
0.83	0.83	Node_Memory Mapped	Node_Disk Bytes Written	nvidia_gpu_memory_used_bytes

Figure 6.3: Tests for GPU jobs with no aggregation

6.2 Results for Machine Learning Classifiers

Model	**Accuracy**	**Class**	**Precision**	**Recall**	**F1-Score**
AdaBoost	0.91	0	0.89	0.78	0.83
		1	0.92	0.96	0.94
Logistic Regression	0.80	0	0.67	0.55	0.61
		1	0.84	0.90	0.87
K-Nearest Neighbors	0.92	0	0.91	0.80	0.85
		1	0.93	0.97	0.95
Decision Tree	0.93	0	0.90	0.85	0.87
		1	0.94	0.96	0.95
Random Forest	0.95	0	0.94	0.87	0.90
		1	0.95	0.98	0.97
Hist Gradient Boosting	0.95	0	0.95	0.87	0.91
		1	0.95	0.98	0.97
Gradient Boosting	0.94	0	0.94	0.82	0.88
		1	0.94	0.98	0.96

Table 6.2: Comparison of Classification Metrics for Different Models

The comparison of various machine learning classifiers involved the standout solution. This included node forks, node disk bytes written and node netstat tcp in errs which in previous tests had 0.96 Random Forest and Histogram Gradient Boosting accuracy. To validate the results more jobs were added, totaling 667 successful jobs and 253 failed ones. Table 6.1 describes the results of various machine learning classifiers. It displays accuracy, precision, recall and F1-score distinguishing the results between the classes. Figure 6.4 contextualizes these results, showing that the differences between classifiers are slight. The results for Histogram Gradient

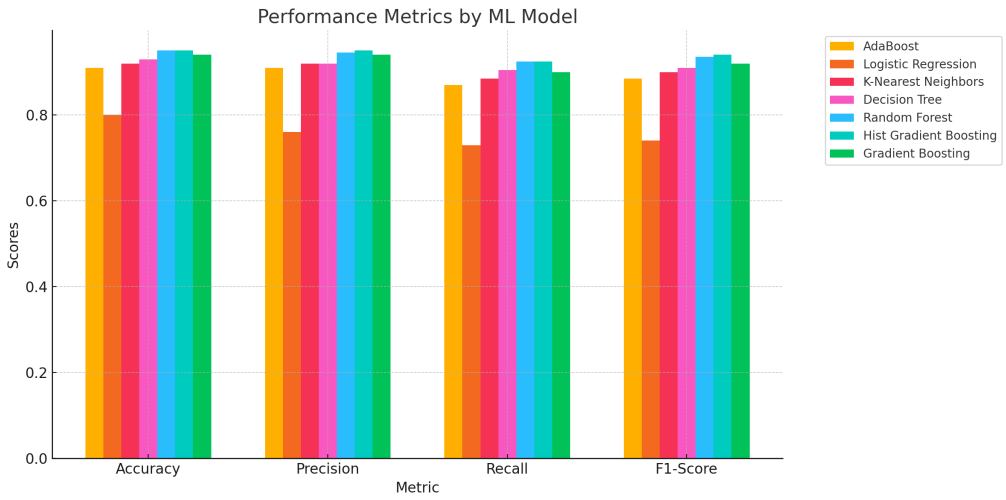

Figure 6.4: Visualization of results for different ML classifiers

Boost[18] and Random Forest[5] are the best among all classifiers and are similar to previous results on the same metrics with fewer nodes. Every Machine Learning classifier is better at predicting class '1'(successful job) rather than class '0'(failed job). Other applied classifiers included K-nearest neighbours[10], Decision Tree[56], Logistic Regression[37], AdaBoost[63] and Gradient Boosting[18]. Figure 6.5 shows the ROC(Receiver Operating Characteristic) curve created with Random Forest classifier. The Area under the Curve(AUC) can range from 0 to 1 where 1 is a perfect model capable of always distinguishing between the classes and a 0.5 suggesting a random chance. The AUC of 0.98 suggests a very capable model.

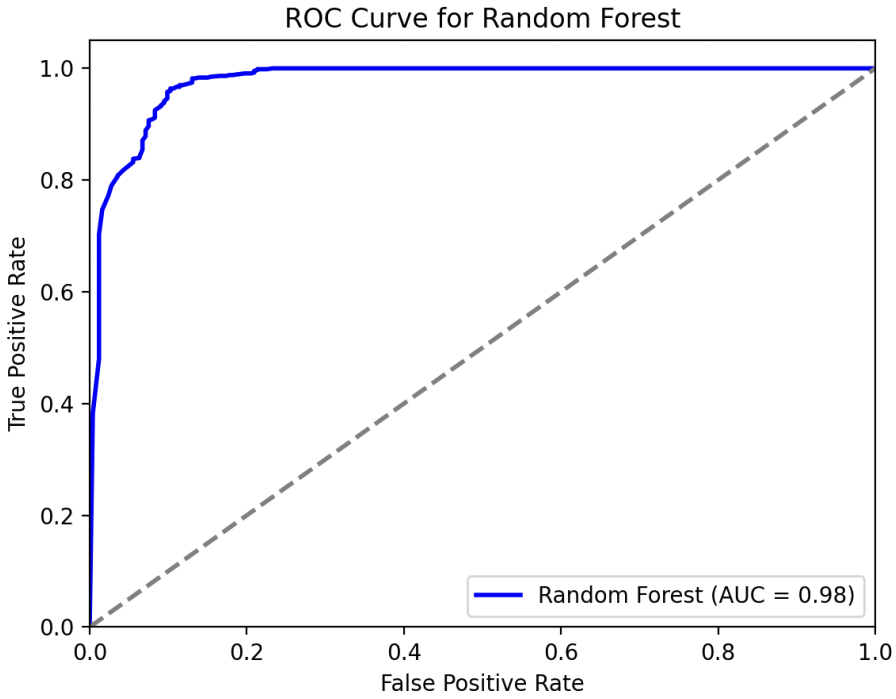

Figure 6.5: ROC curve for the standout solution

Chapter 7

Discussion and conclusion

This chapter gives answer to the research questions which were posed at the start of this work. It explores the limitations, discussing the reasons why the scope of this thesis was reduced. Finally it delves into future works, the directions researchers can take this topic in the future.

To answer RQ1 we conducted a literature review. With help of taxonomies by Tanja Hagemann and Katerina Katsarou [28] and Moghaddam et al.(2019)[45] we have established that those were currently the most popular performance analysis methods.

- Signature Based
- Threshold Based
- Statistical
- Machine Learning
- Deep Learning

Most of those methods offer multiple solutions for failure prediction purpose. Signature based approaches are about capturing certain patters(signatures) and later recognizing them when they repeat. An example of a solution which makes use of Signature Based approach is the work of Mi et al.(2008)[42]. Threshold based approaches are often used to support other methods, this was the case in the work of Hong et al.(2015)[30] where it supported a Machine Learning method. They trigger certain action or analysis based on a predefined threshold which can be static or dynamic depending on specified conditions. Statistics based approaches like Regression Analysis[33], or Hidden Markov Model[30] offer another avenue to explore in terms of failure prediction. Neural network based Deep Learning methods are gaining popularity over the last few years. Self-organizing map [57] or multi-layer perceptron [3] have been used for performance analysis purpose. Finally there are Machine Learning methods, the focus point of

this work. Mohammad S. Jassas and Qusay H. Mahmoud[35] made use of multiple Machine Learning techniques for job-failure prediction purpose. Those were Decision Trees, Random Forest, XGBoost, Gradient Boost, K-Nearest-Neighbour and Naive Bayes algorithm.

To answer SRQ1.1 we made use of snowballing as a literature review method to identify and gather relevant studies. In the process of literature review we established that CPU, Disk and Memory usage are the features most commonly used for failure prediction in the cloud. This was the case in the works of Mohammad S. Jassas and Qusay H. Mahmoud(2022)[35], Marciq et al.(2018)[39], Vishwanath et al.(2010) [62], Birke et al.(2012)[6], Rui Ren et al.(2018)[50] and Chen et al.(2014)[9]. None of the proposed solutions perfected failure detection, there is still room for improvement when it comes to predicting failure. Another serious limitation is that the jobs which are predicted in aforementioned work do not make use of capabilities of a GPU. The work of [61] highlights that workload including a GPU could be more prone to failure with a large amount of resources being wasted.

To answer SRQ1.2 we made use of snowballing as literature review method to identify and gather relevant studies. Work o Jassas and Mahoud(2022)[35] made use of Google cluster, Mustang and Trinity datasets. Work of Chen et al.(2014)[9] also made use of Google cloud cluster for the purpose of failure prediction. The work of Vishwanath et al.(2010)[62] makes use of multiple data sources from a Microsoft data-center. Rui Ren et al.(2018)[50] data was sourced from a Chinese cloud provider Alibaba.

To answer RQ2 we found a dataset supporting this research which was not previously used for failure detection purposes. It contains more than 300 hundred of low level hardware metrics and a corresponding jobs dataset. This allowed us to make use of state of the art features as well as features which might overcome the limitations of state of the art. To establish which features were relevant we made use of Machine Learning for failure prediction. Relevant features had to perform well in ML classification.

To answer SRQ2.1 we applied various feature selection methods like RFE[11] or RF[5] on some of the features from the dataset. We encountered difficulties, feature importance seemed to depend on the order in which the feature importance was estimated. We suspected that the issue might be tied to auto-correlation but that turned out not to be the case. At the heart of the issue was the flattening of the metric data, which seemed to distort the feature importance. To overcome it we had to apply permutation importance[4]. This solved the initial problem, but made feature selection resource intensive. Because of that we used it only in an ad-hoc fashion, our most important discovery was that node netstat tcp was a relevant feature and could replace node memory mapped. Due to these difficulties we did not answer SRQ2.1.

To answer SRQ2.2 a data transformation was performed which aggregated long lists into just four values. Those values were standard deviation, mean, minimum and maximum. Machine Learning classification based on transformed data turned out to be close but slightly worse in terms of accuracy. The new aggregated metrics did not reflect the results of their non-aggregate counterparts, the best aggregate solution could not be the best non-aggregate solution. It seems that finer data granularity does lead to a more precise model, but a good-enough model can be created with relatively small amount of data being fed into the model.

To answer RQ3 we applied various Machine Learning classification techniques on different sets of features. It is noteworthy that a lot of features were successful at detecting failure. For all standard jobs the accuracy does not drop below 0.90 regardless of the selected features. Some of this can be explained by class 2:1 imbalance, which was being alleviated by giving a correct prediction of a failed job more important. A different explanation is that a higher usage of any given feature often signals general high resource usage which is often indicative of failure. Another reason could be that longer running jobs have a higher failure chance [61] and every feature carries information about the execution time. Some unusual features seemed to contribute to the improvement in models accuracy, those would be node forks, various gpu metrics, node netstat tcp in errs seemed to find failures which were previously misclassified. Another insight was that, in the context of this dataset Node Memory Mapped outperformed other memory metrics such as Node Memory Cached, Node Memory Active or Node Memory Inactive. For the disk metrics Node Disk Bytes Written outperformed Node Disk Bytes Read. Finally, the jobs involving a GPU component proved to be much more difficult do accurately predict. No combination of features reached an accuracy of 0.90 and instead found themselves in the range of 0.81-0.89. The graphic card metrics seemed to meaningfully contribute to a good model. All of this should be taken with a grain of salt because of low sample-size for GPU jobs.

To answer SRQ3.1 we applied various Machine Learning classification techniques on different sets of features. For AdaBoost, Logistic Regression, K-Nearest Neighbour, Decision Tree, Random Forest, Histogram based Gradient Boosting and Gradient Boosting we looked at Accuracy, Precision, Recall and F1 Score. We compared all of them on the standout solution which achieved the best results for Random Forest and Histogram based Gradient Boosting. All of them preformed remarkably similarly with the exception of Logistic Regression. Figure 6.4 displays the results.

7.1 Conclusion

Since the models trained here make use of unique features, it is impossible to apply it in a context of another dataset which does not offer the same features. We can speculate about how varied the result would be. Research by Jassas [35] applied various Machine Learning classifiers to predict failure in 3 different datasets. In Google Trace a Random Forest classifier making use of Select-K-Best feature importance achieved an precision of 0.98 while in a Trinity dataset with the same setup precision dropped to 0.72. This give us an idea about how dissimilar the results can be if we try a solution on another data.

One of the reasons to apply job failure prediction is to reduce the waste of data center resources. Data center capacities are growing to support increasing demand of the AI sector. Building data centers is expensive, and it requires a sufficient power-supply to operate. Predicting anomalies and failures before they happen, and acting on this knowledge offers increased data center capacity without incurring a financial cost. We believe that better failure detection systems helps to tackle capacity challenges modern data centers are facing.

Perhaps Machine Learning based failure detection will not prove to be the best way to tackle this problem, but the features which were the focus point of this thesis could be used in another performance analysis methods and contribute to them.

In this thesis we looked into various failure predicting systems. We delved into their limitations and constraints like using a limited set of features or difficulty with reproducing results across the datasets. We proposed our own machine learning solution based on a low level hardware metrics dataset. We experimented with various combinations of features, and machine learning models and compared them in terms of accuracy, precision, f1-score and recall. Our data was transformed to allow for various granularity settings and separation of jobs which involved GPU's to allow for separate testing. In the end we visualized the results.

7.2 Limitations

Lisa datasets were updated while this thesis was far progressed, fixing some of the problems with node timestamp lags. This happened when the thesis progressed quite far and it did not make use of the update. This could have helped with the large numbers of jobs lost during data cleaning.

One of the biggest limitations are the missing metrics for the CPU in the Lisa dataset. They are collected but not at the same time as job data. This is important because most of the Machine Learning failure prediction systems make us of CPU related information. The current solution can

accommodate another feature, it is likely that adding CPU metrics would slightly alter evaluation of some of the metrics. To overcome this, solution would need to be thoroughly tested on a dataset with access to CPU and other metrics being available.

Solution as is, only uses feature importance in an ad-hoc fashion, to compare features which seemed to perform well and understand which of them work well together. Given a sufficient computing power and adapting nested features, an optimal solution for this dataset could have been found. Due to the scale of the dataset it was not possible.

Another limitation was a limited number of Machine Learning jobs which involved a GPU. These jobs were the most likely to be cancelled before concluding. Cancelled jobs were excluded from the analysis as they were difficult to predict accurately. After the data cleaning, the number of jobs viable for the analysis was around a hundred.

Reproducibility of the results can be a challenge because the metrics captured in the data center are often different. For example, Lisa Dataset did not contain information about the number of spawned containers. This metrics was used for Machine-Learning prediction in the work of Ren et al[50]. Versluis et al[61] mention in the title of their paper points out another issue, the scientific community needs more datasets describing low-level performance of a data center. The dataset are both scarce and have substantial differences between one another.

7.3 Future Works

Machine Learning failure prediction proposed in this thesis defines failure as anything which is not a successful completion of the job while excluding canceled jobs. This is represented in transforming all successful state labels in the jobs dataset into 1's and other states into 0's. An alternative approach would look into individual states like 'OUT OF MEMORY','TIMEOUT','NODE FAIL','FAILED' and show how features based on low level performance metrics predict those states. One of the potential research questions could be, 'What failed state is the most difficult to predict?'

Bibliography

[1] David V Akman et al. “k-best feature selection and ranking via stochastic approximation”. In: *Expert Systems with Applications* 213 (2023), p. 118864.

[2] M.G. Akritas. “Spearman’s Rank Correlation Coefficient”. In: *Handbook of Statistics, Volume 30*. Ed. by J.M. Bernardo et al. Elsevier, 2011, pp. 383–401.

[3] Ahmad Alnafessah and Giuliano Casale. “Artificial neural networks based techniques for anomaly detection in Apache Spark”. In: *Cluster Computing* 23.2 (2020), pp. 1345–1360.

[4] André Altmann et al. “Permutation importance: a corrected feature importance measure”. In: *Bioinformatics* 26.10 (2010), pp. 1340–1347.

[5] Gérard Biau and Erwan Scornet. “A random forest guided tour”. In: *Test* 25 (2016), pp. 197–227.

[6] Robert Birke, Lydia Y Chen, and Evgenia Smirni. “Data centers in the cloud: A large scale performance study”. In: *2012 IEEE Fifth international conference on cloud computing*. IEEE. 2012, pp. 336–343.

[7] Varun Chandola, Arindam Banerjee, and Vipin Kumar. “Anomaly detection: A survey”. In: *ACM computing surveys (CSUR)* 41.3 (2009), pp. 1–58.

[8] Tianqi Chen and Carlos Guestrin. “Xgboost: A scalable tree boosting system”. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[9] Xin Chen, Charng-Da Lu, and Karthik Pattabiraman. “Failure analysis of jobs in compute clouds: A google cluster case study”. In: *2014 IEEE 25th International Symposium on Software Reliability Engineering*. IEEE. 2014, pp. 167–177.

[10] Padraig Cunningham and Sarah Jane Delany. "K-nearest neighbour classifiers-a tutorial". In: *ACM computing surveys (CSUR)* 54.6 (2021), pp. 1–25.

[11] Burcu F Darst, Kristen C Malecki, and Corinne D Engelman. "Using recursive feature elimination in random forest to account for correlated variables in high dimensional data". In: *BMC genetics* 19 (2018), pp. 1–6.

[12] Daniel J Dean et al. "Perfcompass: Online performance anomaly fault localization and inference in infrastructure-as-a-service clouds". In: *IEEE Transactions on Parallel and Distributed Systems* 27.6 (2015), pp. 1742–1755.

[13] Thanh Do et al. "Limplock: Understanding the impact of limpware on scale-out cloud systems". In: *Proceedings of the 4th annual Symposium on Cloud Computing*. 2013, pp. 1–14.

[14] Qingfeng Du, Tiandi Xie, and Yu He. "Anomaly detection and diagnosis for container-based microservices with performance monitoring". In: *Algorithms and Architectures for Parallel Processing: 18th International Conference, ICA3PP 2018, Guangzhou, China, November 15-17, 2018, Proceedings, Part IV 18*. Springer. 2018, pp. 560–572.

[15] Dmitry Duplyakin et al. "The design and operation of {CloudLab}". In: *2019 USENIX annual technical conference (USENIX ATC 19)*. 2019, pp. 1–14.

[16] Mostafa Farshchi et al. "Metric selection and anomaly detection for cloud operations using log and metric correlation analysis". In: *Journal of Systems and Software* 137 (2018), pp. 531–549.

[17] Daniel Fireman et al. "Improving tail latency of stateful cloud services via gc control and load shedding". In: *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE. 2018, pp. 121–128.

[18] Jerome H Friedman. "Stochastic gradient boosting". In: *Computational statistics & data analysis* 38.4 (2002), pp. 367–378.

[19] C. Genest and J. Nešlehov'a. "Kendall's Rank Correlation Coefficient". In: *Handbook of Statistics, Volume 30*. Ed. by J.M. Bernardo et al. Elsevier, 2011, pp. 403–424.

[20] Siavash Ghiasvand and Florina M Ciorba. "Anomaly detection in high performance computers: A vicinity perspective". In: *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*. IEEE. 2019, pp. 112–120.

[21] Einollah Jafarnejad Ghomi, Amir Masoud Rahmani, and Nooruldeen Nasih Qader. "Load-balancing algorithms in cloud computing: A survey". In: *Journal of Network and Computer Applications* 88 (2017), pp. 50–71.

[22] Brendan Gregg. "Blazing performance with flame graphs". In: *27th Large Installation System Administration Conference*. USENIX Association. 2013.

[23] Brendan Gregg. *Systems Performance: Enterprise and the Cloud*. Prentice Hall, 2013. ISBN: 978-0-13-339017-4.

[24] Brendan Gregg. "Thinking methodically about performance". In: *Communications of the ACM* 56.2 (2013), pp. 45–51.

[25] Arthur Gretton et al. "A kernel two-sample test". In: *The Journal of Machine Learning Research* 13.1 (2012), pp. 723–773.

[26] Qiang Guan and Song Fu. "Adaptive anomaly identification by exploring metric subspace in cloud computing infrastructures". In: *2013 IEEE 32nd International Symposium on Reliable Distributed Systems*. IEEE. 2013, pp. 205–214.

[27] Fabio Guigou, Pierre Collet, and Pierre Parrend. "Anomaly detection and motif discovery in symbolic representations of time series". In: *arXiv preprint arXiv:1704.05325* (2017).

[28] Tanja Hagemann and Katerina Katsarou. "A systematic review on anomaly detection for cloud computing environments". In: *Proceedings of the 2020 3rd Artificial Intelligence and Cloud Computing Conference*. 2020, pp. 83–96.

[29] Charles R Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (2020), pp. 357–362.

[30] Bin Hong et al. "DAC-Hmm: detecting anomaly in cloud systems with hidden Markov models". In: *Concurrency and Computation: Practice and Experience* 27.18 (2015), pp. 5749–5764.

[31] Olumuyiwa Ibidunmoye, Francisco Hernández-Rodriguez, and Erik Elmroth. "Performance anomaly detection and bottleneck identification". In: *ACM Computing Surveys (CSUR)* 48.1 (2015), pp. 1–35.

[32] Olumuyiwa Ibidunmoye, Ewnetu Bayuh Lakew, and Erik Elmroth. "A black-box approach for detecting systems anomalies in virtualized environments". In: *2017 International Conference on Cloud and Autonomic Computing (ICCAC)*. IEEE. 2017, pp. 22–33.

[33] Anil Jain and Douglas Zongker. "Feature selection: Evaluation, application, and small sample performance". In: *IEEE transactions on pattern analysis and machine intelligence* 19.2 (1997), pp. 153–158.

[34] Vikramaditya Jakkula. “Tutorial on support vector machine (svm)”. In: *School of EECS, Washington State University* 37.2.5 (2006), p. 3.

[35] Mohammad S Jassas and Qusay H Mahmoud. “Analysis of job failure and prediction model for cloud computing using machine learning”. In: *Sensors* 22.5 (2022), p. 2035.

[36] Bahman Javadi et al. “The Failure Trace Archive: Enabling the comparison of failure measurements and models of distributed systems”. In: *Journal of Parallel and Distributed Computing* 73.8 (2013), pp. 1208–1223.

[37] Michael P LaValley. “Logistic regression”. In: *Circulation* 117.18 (2008), pp. 2395–2399.

[38] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. “Isolation forest”. In: *2008 eighth ieee international conference on data mining*. IEEE. 2008, pp. 413–422.

[39] Aleksander Maricq et al. “Taming performance variability”. In: *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. 2018, pp. 409–425.

[40] Wes McKinney et al. “pandas: a foundational Python library for data analysis and statistics”. In: *Python for high performance and scientific computing* 14.9 (2011), pp. 1–9.

[41] Patrick E McKnight and Julius Najab. “Mann-Whitney U Test”. In: *The Corsini encyclopedia of psychology* (2010), pp. 1–1.

[42] Ningfang Mi et al. “Analysis of application performance and its change via representative application signatures”. In: *NOMS 2008-2008 IEEE Network Operations and Management Symposium*. IEEE. 2008, pp. 216–223.

[43] Microsoft. *Azure Monitor overview*. World Wide Web. Accessed Month Day, Year. Accessed Year. URL: `https://learn.microsoft.com/en-us/azure/azure-monitor/overview`.

[44] Victor Millnert and Johan Eker. “HoloScale: horizontal and vertical scaling of cloud resources”. In: *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*. IEEE. 2020, pp. 196–205.

[45] Sara Kardani Moghaddam, Rajkumar Buyya, and Kotagiri Ramamohanarao. “Performance-aware management of cloud resources: a taxonomy and future directions”. In: *ACM Computing Surveys (CSUR)* 52.4 (2019), pp. 1–37.

[46] National Institute of Standards and Technology (NIST). *The NIST Definition of Cloud Computing*. `https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf`. Accessed: [insert date of access]. Sept. 2011.

[47] Mostafa Noshy, Abdelhameed Ibrahim, and Hesham Arafat Ali. "Optimization of live virtual machine migration in cloud computing: A survey and future directions". In: *Journal of Network and Computer Applications* 110 (2018), pp. 1–10.

[48] John Ousterhout. "Always measure one level deeper". In: *Communications of the ACM* 61.7 (2018), pp. 74–83.

[49] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[50] Rui Ren et al. "Anomaly analysis for co-located datacenter workloads in the alibaba cluster". In: *arXiv preprint arXiv:1811.06901* (2018).

[51] Irina Rish et al. "An empirical study of the naive Bayes classifier". In: *IJCAI 2001 workshop on empirical methods in artificial intelligence*. Vol. 3. 22. Citeseer. 2001, pp. 41–46.

[52] Hiroaki Sakoe and Seibi Chiba. "Dynamic programming algorithm optimization for spoken word recognition". In: *IEEE transactions on acoustics, speech, and signal processing* 26.1 (1978), pp. 43–49.

[53] Tara Salman et al. "Machine learning for anomaly detection and categorization in multi-cloud environments". In: *2017 IEEE 4th international conference on cyber security and cloud computing (CSCloud)*. IEEE. 2017, pp. 97–103.

[54] Haiying Shen and Liuhua Chen. "Resource demand misalignment: An important factor to consider for reducing resource over-provisioning in cloud datacenters". In: *IEEE/ACM Transactions on Networking* 26.3 (2018), pp. 1207–1221.

[55] Ahmad Al-Shishtawy and Vladimir Vlassov. "Elastman: elasticity manager for elastic key-value stores in the cloud". In: *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*. 2013, pp. 1–10.

[56] Yan-Yan Song and LU Ying. "Decision tree methods: applications for classification and prediction". In: *Shanghai archives of psychiatry* 27.2 (2015), p. 130.

[57] Ioannis M Stephanakis et al. "Hybrid self-organizing feature map (SOM) for anomaly detection in cloud infrastructures using granular clustering based upon value-difference metrics". In: *Information Sciences* 494 (2019), pp. 247–277.

[58] Sandro Tosi. *Matplotlib for Python developers*. Packt Publishing Ltd, 2009.

[59] Beth Trushkowsky et al. "The {SCADS} Director: Scaling a Distributed Storage System Under Stringent Performance Requirements". In: *9th USENIX Conference on File and Storage Technologies (FAST 11)*. 2011.

[60] Ozan Tuncer et al. "Diagnosing performance variations in HPC applications using machine learning". In: *High Performance Computing: 32nd International Conference, ISC High Performance 2017, Frankfurt, Germany, June 18–22, 2017, Proceedings 32*. Springer. 2017, pp. 355–373.

[61] Laurens Versluis et al. "Less is not more: We need rich datasets to explore". In: *Future Generation Computer Systems* 142 (2023), pp. 117–130.

[62] Kashi Venkatesh Vishwanath and Nachiappan Nagappan. "Characterizing cloud computing hardware reliability". In: *Proceedings of the 1st ACM symposium on Cloud computing*. 2010, pp. 193–204.

[63] Cao Ying et al. "Advance and prospects of AdaBoost algorithm". In: *Acta Automatica Sinica* 39.6 (2013), pp. 745–758.

[64] Xiao Zhang, Fanjing Meng, and Jingmin Xu. "Perfinsight: A robust clustering-based abnormal behavior detection system for large-scale cloud". In: *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE. 2018, pp. 896–899.

[65] Xiaodong Zhang, Yanxia Qu, and Li Xiao. "Improving distributed workload performance by sharing both CPU and memory resources". In: *Proceedings 20th IEEE International Conference on Distributed Computing Systems*. IEEE. 2000, pp. 233–241.

[66] Alexander Zien et al. "The feature importance ranking measure". In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2009, Bled, Slovenia, September 7-11, 2009, Proceedings, Part II 20*. Springer. 2009, pp. 694–709.