Bachelor Informatica

University of Amsterdam

Characterizing access patterns from ftp logs: a case study on Euro-Argo research infrastructure

Ewoud Bouman

September 15, 2018

Supervisor(s): dr. Zhiming Zhao, dr. ir. Arie Taal, dr. Spiros Koulouzis

Signed: Signees

This thesis aims to characterise the access patterns and the performance of a data infrastructures from its FTP log files. The research uses log files produced by a data infrastructure in a environmental science domain called Euro-Argo. First an exploratory study of the Euro-Argo log files is presented with the goal of determining how the data infrastructure is used and how much information can be extracted from the log files. Based on these results we determine how these insights can be used to improve future service quality of a data infrastructure. We tried to forecast the number of download requests to the server based on the historical patterns extracted from the log files. The forecasts can help with the efficient allocation of the available resources to improve the offered service level. In the thesis, we conclude that log file analysis can indeed improve the service quality offered by the Euro-Argo data infrastructure.
The research is partially supported by EU H2020 ENVRIPLUS project (654182), VRE4EIC (676247) and Euro-Argo research infrastructure.

Contents

CHAPTER 1

Introduction

Analysing log files to characterize service usage or improve quality metrics is not a new discipline. The derived metrics and characterisations can be used to predict and provision for the future. However, there is not much information available about the use cases of log analysis within small but highly specialized domains. This thesis will examine how the analysis of log files can be applied to such a specific domain.

The Euro-Argo research infrastructure is the European contribution to the global argo program. Currently there are more than 3500 autonomous float instruments deployed globally by Euro-Argo to measure temperature, salinity or other properties of the earth's oceans. The collected raw data is processed and then made available via its data portal, and can be accessed by user communities from a FTP server.

The quality of the services offered to the research community determines the effectiveness of how they can use the data for their research. This means fast and easy access to their data on a reliable schedule. To guarantee this the Euro-Argo data infrastructure needs to allocate sufficient resources for the storage of data, execution of service requests and bandwidth for down and uploads. To correctly allocate sufficient resources an understanding is necessary about the workload patterns expected in the future.
This knowledge is currently lacking, log analysis might change this.

Successfully applying log analysis requires a diverse skill set. It requires general knowledge about data science and analytics. Domain specific knowledge is necessary to understand what the possible events in the log files represent.
Because scientific data infrastructures are aimed at experts in their domain it can be difficult for an outsider to identify and understand the possible patterns and events hidden within the log files.
The high entry barrier combined with a niche target audience can make the idea of analysis seem uninteresting.
This is a shame because not only does this leave possible infrastructure optimizations ignored but there may also be characteristics hidden in the log files that can lead to new insights about how the data itself is used once released.
This thesis examines how the log files produced by a FTP server can be used for optimization purposes. Extra focus is given to the exploration of the log files to determine the amount of knowledge that can be extracted from the logs to better understand how the Euro-Argo data infrastructure is used.

1.1 Research question

Over a period of 18 months the operation history of the Euro-Argo data infrastructure has been collected in the log files. The research question addressed in this thesis about these log files is:

How to optimise the accessibility and performance of a data infrastructure like Euro-Argo using the FTP log files?

To better answer the research question, we identified two sub-questions in this thesis:

1. *What characteristics and access patterns of a data infrastructure can we extract from its FTP logs?*
2. *What optimisation strategies can be proposed based on the access log files?*

CHAPTER 2

Related work

This chapter provides an overview of the literature examined for this thesis.
The first part reviews literature relevant to the field of log analysis. This is followed by a more focused review of the analysis of log files produced by FTP servers. The final section examines use cases where log files served as the source of information for prediction strategies.

Oliner, Ganapathi and Xu [18] analysed the current state of log analysis. While most log files are intended to facilitate debugging operations these days log files are also used to optimise resource provisioning.
Log files can reveal how resources in a process are used. This can help with understanding and improving the performance of a process.
But using log files to optimise processes can be a challenge. Analytical and statistical models can mine large amounts of data but this does not usually provide clear and actionable insights. The interpretation of the information generated by the mining of log files is a subjective process usually performed by humans. To assist with the task of interpretation visualization techniques where proposed.

Jansen [15] proposed a general methodology for the analysis of log files consisting of three stages, collection, preparation and analysis.
The collection stage is the process of collecting the files for a given period from the log files where the research question defines what information is necessary to collect.
The preparation stage prepares the log files for analysis. Log files need to be cleaned by removing the corrupted entries in the data and parsed into a correct representation of the content. The analysis stage is the process of analysing the prepared data.
The proposed methodology provides structure for the analyst that can be combined with other methods.

Several papers have been written about the analysis of FTP log files within a specialized domain:

For the EROS data repository Zong et al. [21] analysed the optimization possibilities for their FTP server and storage environment. The authors analysed a log file containing 3 years of download requests.
They concluded that conventional cache optimization techniques like market basket analysis prefetching or classification based replacements solution fail because of a large variety of users and files.
By characterizing the user behaviour and system workload two custom prefetching algorithms where proposed. In combination with a least recently used caching strategy the authors achieved a six percent cache improvement.
For the UNAVCO geoscience institute Hodgkinson and Rezgui [12] analysed the FTP access logs to identify trends in GPS-station data usage. By applying a map reduce approach they noticed that users had increased interests in data from specific historical periods and geographical lo-

cations. The authors also observed a select group of heavy data users which they wanted to target in the future for possible research collaborations. And infrastructure improvements could be achieved by allocating extra resources on periods favoured by researchers for their visits to the server.

The CPTEC (Brazilian center of weather forecast and climate studies) performed an analysis of their FTP log files using the ELK (Elasticsearch, Logstash and Kibana) stack. With the ELK stack the authors [16] achieved several results. They characterized common patterns with problematic file transfers and file deletions. The information contained in the logs also provided a better understanding of how their data was requested by their users.

The following papers address how log files can be used for prediction strategies.

Rodrigo, Enayat and Rajkumar [3] applied a time series analysis with a ARIMA model on the web traffic of an HTTP server. Using a three week training set they forecast the web traffic per hour in the fourth week. The results showed a accuracy of up to 91%. For their purpose this made the model usable for the creation of a more efficient resource utilization strategy.

Vazquez, Krishnan, and John [20] implemented an automated resource scaling algorithm for dynamic resource provisioning.
By forecasting future resource demand they pro actively allocated the necessary resources for a cloud computing environment.
They compared several forecasting models for their ability to provide accurate forecasts over different periods of time.

CHAPTER 3

Data introduction

In this chapter, we will first introduce the data server of Euro-Argo, and then use Euro-Argo as an example to introduce the typical structure of a data access log.

This will give the reader a better understanding of the subsequent chapters.

3.1 Euro-Argo FTP server

The Euro-Argo FTP server is part of a global Argo data center (GDAC) and provides the master copy of the Argo data set.
The data set archives the measurements collected by the Argo floats. Floats are autonomous platforms deployed in the oceans for the monitoring of the environment.
Each float is associated with 4 different types of files [4] contained in the data set:

- Trajectory files: A single file containing the complete trajectory history of an individual float and the corresponding measurements.
- Technical files: A single file with technical status of an individual float.
- Metadata files: A single file with general information of an individual float.
- Profile files: A set of files containing the measurements acquired over a period of time. Each profile contains a single period of an individual float.

The FTP server provides the Argo data in three different formats:

- Individual format: The files are provided individually. Each float is contained in their own subdirectory on the server. This subdirectory contains all 4 types of information.
- Geographical format: A single file generated per day per ocean containing all the float profiles acquired that day.
- Latest data: A single file generated per day containing all the float profiles received that day.

Other files available on the server include index files for FTP services, checksum files and monthly archives.

3.2 The log files

The log contains the traces collected by the FTP server. For each individual request received by the server a single line is written to the log file after the completion of the request. Each line contains 14 different fields. Table 3.1 presents the structure of the log file [19] accompanied by a random, and anonymized, log entry extracted from the Euro-Argo log files.

Table 3.1: Log file structure of the Euro-Argo FTP server

Field	**Description**	**Example**
Current-time	Local server time of the finished request	Sat Apr 1 00:00:00 2017
Transfer-time	Total time of the file transfer rounded to seconds	1
Remote-host	IP address of the remote-host	123.123.123.123
File-size	Total size of the finished transfer in bytes	24344
File-name	Name of the transferred file	/ifremer/argo/dac/coriolis /3900521/profiles /D3900521_335.nc
Transfer-type	Flag indicating the transfer type (a:ascii or b:binary)	a
Special-action-flag (*)	Flag indicating if a file compression was performed (c:compressed, u:uncompressed, t:tar'ed, _:no action)	_
Direction (*)	The request direction of the transfer (i:incoming,o:outgoing or d:deletion)	o
Access-mode (*)	Method by which the user is logged in (a:anonymous, g:guest, r:real)	a
Username	Name of the remote user	flashfxp-user@flashfxp.com
Service-name (*)	Name of the transfer service	FTP
Authentication-method (*)	Method of service authentication (0:None or 1:RFC931)	0
Authenticated-user-id (*)	User id returned by the authentication-method, a * star is used if not available	*
Completion-status	Flag indicating the final status of the transfer (c:complete,i:incomplete)	c

From the 14 fields 6 are of little value for the analysis because they contain identical values. These fields are marked with a (*) next to their field name in the Table 3.1. The 6 fields share the same value for all entries because of the following reasons:

- Special-action-flag: the server does not perform actions.
- Direction: users are only allowed to download files
- Access-mode: the server only accepts anonymous access.
- Service-name: only the FTP service is enabled.
- Authentication-method: there is no authentication method available on the server.
- Authenticated-user-id: there is no authentication method available on the server.

CHAPTER 4

A hypothesis driven approach

This chapter describes the methodology used in this thesis.
The first section describes the approach used to answer the first sub-question. A description is given of the process structuring the log file analysis.
The second section contains the methodology behind a proposed optimisation strategy based on the contents of the Euro-Argo log files.

4.1 Hypothesis driven approach

The answer to the research question was motivated by 2 sub-questions. The purpose of the first sub-question was to identify patterns and characteristics that could be used for optimisation strategies.
The approach to answer the sub-question was influenced by 2 factors. First there was no prior knowledge available about the contents of the log files. This meant that it was not clear where to start looking in the log files for possible patterns.
And the second factor was that the literature study did not reveal cases where detailed optimisation strategies where proposed using only log files produced by a FTP server.

To deal with the lack of knowledge and unknown expectations a structured approach was necessary to guide the process of answering the first sub-question.
A hypothesis driven approach was formed that made it possible to track, evaluate and communicate the progress of the knowledge gathering while searching for possible answers.

The hypothesis driven approach consisted of 4 steps. In theory these steps could be performed in a sequential order but in practice the steps where performed in an iterative nature. Each step could lead to to new insights, results or questions making it necessary to revisit the previous steps to reflect the new found knowledge.

Multiple iterations starting with different hypothesis where necessary before a number of patterns and characteristics where discovered that could possibly lead to an answer for the first sub-question.

The 4 steps of the hypothesis driven approach where:

- **Raise hypothesis**

The first step was to raise hypothesis about the possible unexplored information in the log files. The hypothesis represented guesses about possible patterns in the log files formed by previously acquired knowledge, assumptions and curiosity. Rejecting or accepting the hypothesis was not the goal because it was very unlikely to stumble upon a definite answer. Instead the hypothesis where revised or replaced during the 4 step process.

The hypothesis forced to keep a perspective during the exploration of the data. This was necessary because without a clearly defined direction it was easy to lose sight of the steps necessary to arrive at a final answer.

- **Prepare data**

With the hypothesis as the guideline, the next step was to prepare the data for exploration. The preparation step included multiple actions.
First the relevant data had to be selected. The available data had to be evaluated in terms of quality, quantity and relevancy.Depending on the hypothesis further processing of the data was required such as enriching the data with geographical information or aggregating a series of events over a time interval.
Finally the data had to be prepared into a format suitable for efficient exploration.

- **Explore data**

The exploration step followed a semi-structured approach using the hypothesis as a starting position. Using data visualizations techniques the data was explored to evaluate if the data could support what the hypothesis stated. During this process new knowledge and insights of the data set was acquired.

- **Interpret insights**

The final step was to reflect on the knowledge extracted during the exploration stage. Based on the newly acquired knowledge and insights the hypothesis was either revised or dropped to explore a new hypothesis until a clear answer to the research question was formed.

4.2 Forecast experiment

This section demonstrates a possible optimisation strategy for the Euro-Argo data infrastructure using the FTP log files.
A forecasting experiment was performed to determine the feasibility of forecasting future events based on the observations contained in the data set.

For the experiment the number of download requests received by the FTP server where aggregated at equally spaced points in time. These values where then used to forecast the number of requests in the future.
Three different forecast models where compared to determine the most effective forecasting strategy.

4.2.1 Accuracy measure

To evaluate and compare the forecasts produced by the 3 models a measure of accuracy was necessary. Let $\hat{y}_t$ be the forecast value and y_t the observed value at time t then the forecast error at time t can be defined as

$$\epsilon_t = \hat{y}_t - y_t \tag{4.1}$$

To asses the accuracy over a number of forecasts the MAPE score was used.

$$\text{MAPE} = \frac{100}{n} \sum_{t=1}^{n} \frac{|y_t - \hat{y}_t|}{y_t} \tag{4.2}$$

The MAPE score represents the percentage error between the actual observed value and the forecasts produced by the forecast model. The score can be used to compare forecast performance of models on different data sets.
A lower score represents a better result.

4.2.2 Forecast models

The experiment used three different forecasting models. A naive, mean and ARIMA model

Naive model

The naive model forecasts all h future values equal to the last observed value y_t.

$$\hat{y}_{t+h} = y_t \tag{4.3}$$

The results produced by this model served as a standard of comparison for the other 2 models.

Mean model

The mean model forecasts all h future values as the average of the n most recent observations

$$\hat{y}_{t+h} = (y_1 + \cdots + y_t)/n \tag{4.4}$$

where $(y_1 + \cdots + y_t)$ represents the n most recent observations.

ARIMA method

ARIMA models are a class of linear models fitted to time series data. These models can be used to understand the data or to forecast future variables in the time series.
The acronym ARIMA stands for **A**uto**R**egressive **I**ntegrated **M**oving **A**verage representing the 3 methods the model consists of.

The Autoregressive (AR) component forecasts a variable as a linear combination of the historic values of the variable. The order p determines the number of historic, lagged, values to be used.
The AR(p) model can is expressed as

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \epsilon_t \tag{4.5}$$

Here y_t is the response variable at period t,
$y_{t-1}..y_{t-p}$ are the values at different time lags,
$\phi_1..\phi_p$ are the coefficients of the model,
ϵ_t represents the white noise at period t and c is a constant.

The Moving Average (MA) component uses the moving average over the past forecast errors to represent the error of the model. The order q specifies the number of past errors to be included in the model.

$$y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q} \tag{4.6}$$

with ϵ_t white noise and $\theta_1, ..\theta_q$ the coefficients of the model.

The Integrated part represents the order of differencing applied on the time series. Differencing is the act of computing the differences between consecutive values.
This is done to remove the dependence on time, such as trends ans seasonality, within the time series.
The order d specifies the number of times the series need to be differenced.
For example differencing with $d = 1$ equals:

$$y'_t = y_t - y_{t-1} \tag{4.7}$$

with the $'$ indicating the number of differences applied.
Differencing with $d = 2$ equals:

$$y''_t = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) \tag{4.8}$$

Combined the 3 methods form an ARIMA(p,d,q) model. After differencing d times this model can be written as:

$$y_t = c + \phi_1 y_{t-1} + \cdots + \phi_p y_{t-p} + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q} + \epsilon_t \tag{4.9}$$

This equation can be rewritten [13] to produce forecasts $\hat{y}_{t+h}$ as:

$$\hat{y}_{t+h} = c + \phi_1 y_{t+h-1} + \cdots + \phi_p y_{t-p+h} + \theta_1 \epsilon_{t+h-1} + \cdots + \theta_q \epsilon_{t-q+h} \tag{4.10}$$

where:

- All the future observations y_{t+h-n} will be replaced by their forecasts $\hat{y}_{t+h-n}$.
- All future errors ϵ_{t+h} become zero.
- All past errors ϵ_{t+h-n} not known at time t become zero.

The *auto.ARIMA* function from the R Forecast [14] package was used to automate the selection of the optimal (p, d, q) parameters.

4.2.3 Training and the test set.

For the evaluation of the accuracy of the three forecasting models a rolling forecasting procedure was used [13].
The models where trained on a training set and the accuracy of the models where evaluated on the test set. Using a data set with N observations, a training length of k and a forecast horizon of h the procedure works as follows:

1. Select the first k observations from the data set as the training set for the forecasting model.
2. Select the observations from $k + 1$ to $k + h$ from the data set as the test set.
3. Produce h forecasts with the forecasting model and calculate the MAPE score for each forecast using the observations from the test set.
4. Remove the first observation from the data set and repeat the previous steps until the data set contains less than $k + h$ observations.

Afterwards calculate the average MAPE score for each forecast horizon.
With the rolling forecasting procedure it was possible to evaluate the accuracy of the 3 models over multiple time periods.

CHAPTER 5

Toolchain design

This chapter introduces the toolchains used in this thesis. The first section contains the toolchain for the log file analysis. A description is given of the design and the implementation.
The second section gives an overview of the toolchain used in the forecast experiment.

5.1 Analysis toolchain

This section will give an overview of the toolchain implemented to analyse the FTP log files produced by the Euro-Argo data infrastructure. The goal of the toolchain was to offer an environment in which the 4 step knowledge discovery process could be performed in an efficient manner.

5.1.1 Design

While locally archived FTP log files served as the main source of data for this thesis the toolchain was not designed around a single source of information or environment.
Instead it followed a flexible and modular approach. Flexible as in supporting a range of log files such as web server logs, csv files or user defined formats. The idea behind this was that if in the future new log files became available the toolchain would not require major revisions incorporating them.
Modular meant that it should be possible to replace parts of the toolchain without affecting the overall workflow of the data. This also made it possible to reuse parts of the toolchain for different goals such as performing experiments for the second sub-question.
The toolchain was designed around 5 different stages:

1. Data gathering. The selection and gathering of unprocessed entries in the log files.
2. Data processing. Gathered data needed to be formatted into a format suitable for storage. This step also included the removal of corrupt data and the enrichment of data based on what was available in the log files.
3. Data storage. The processed data need to be stored in an environment where it could be queried for exploratory actions.
4. Data analysis. This represented an optional stage. Depending on the hypothesis the processed data required further processing before the data could be explored.
5. Data visualization. The visualization stage presented the data in a graphical format. Visualization techniques made to possible to explore new insights or observations and evaluate if its was worth further analysis. This was important because preparing data for further analysis requires substantial amounts of time and effort, especially if one is looking for something that is not there.

5.1.2 Toolchain implementation

For the implementation of the toolchain the **ELK** stack [1] (**E**lasticsearch, **L**ogstash, and **K**ibana) was chosen to serve as the base supplemented by several tools for further analysis. The ELK stack is an open-source log analytics platform consisting of multiple individual components.

Figure 5.1: Workflow of the analysis toolchain.

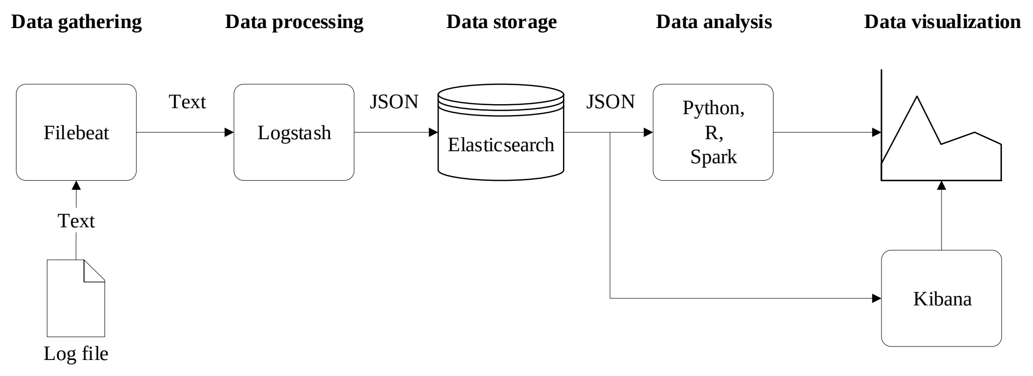


Figure 5.1 shows the flow of the data in the implemented toolchain. The individual stages where implemented as followed:

Data gathering

The data gathering stage was handled by Filebeat [8]. Filebeat is a log forwarder that monitors an individual log file for new or unprocessed events.
These events are automatically extracted and shipped to the processing stage.

Data processing

The data processing was handled by logstash [10]. Logstash is a data processing pipeline that can process data from multiple input sources simultaneously. Logstash performed 3 actions in the processing stage.

The first action was the collection of the events gathered by the gathering stage.
After receiving the event data the contents where processed. For each log format or input source a conditional filter based on grok, a regex library, was applied on the input data.
This transformed the data into a predefined structure. The structured data was enriched using external Logstash libraries.
Examples of this enrichment included the deciphering of geo information from IP addressess or the anonymization of personal information.
Then the processed data was forwarded to a storage solution.

An example of a processed event from the log files is as follows:

Listing 5.1: Random log event

```
Mon Jan 1 23:59:59 2017 1 192.168.123.123 19052 /ifremer/argo/dac/aoml/2901366/
    ↪ profiles/R2901366_305.nc b _ o a username@domain.com ftp 0 * c
```

Listing 5.1 contains a random event retrieved from the Euro-Argo log files.

Listing 5.2: Processed log event

```
{
    "timestamp": "2017-01-01T23:59:59.000Z",
    "remote_host": "xxx.xxx.xxx.xxx",
    "username": "anon@anon.com",
    "geoip": {
        "region_name": "Shanghai",
        "country_name": "China",
        "city_name": "Shanghai",
        "country_code3": "CN",
        "region_code": "31",
        "latitude": xxx.xxxx,
        "longitude": xxx.xxxx,
        "continent_code": "AS",
        "country_code2": "CN",
        "timezone": "Asia/Shanghai"
    },
    "file_name": "R2901366_305.nc",
    "file_size": 19052,
    "file_ext": ".nc",
    "ftp_dir": "dac",
    "dac_center": "aoml",
    "float_number": 2901366,
    "float_data": "profiles",
    "float_cycle": 305,
    "float_descend": "False",
    "float_merged": "False"
    "float_time_mode": "R",
    "completion-status": 1,
    "transfer_time": 1,
    "transfer_type": "b",
    "special_action-flag": "_",
    "direction": "o",
    "access_mode": "a",
    "time_year": 2017,
    "time_month": "Jan",
    "time_day": "Mon",
    "time_clock": "23:59:59"
}
```

The final result of the processing of the random event is shown by listing 5.2.
In the processing stage we aimed to extract as much information as possible from the log files. This was done to prepare as much data as possible for the hypothesis driven approach minimizing the time invested on the data preparation step.

Data storage

The processed data was loaded into a storage environment. This made it possible to perform query operations on the data using external tools. Storage was provided by a Elasticsearch storage engine. Elasticsearch is designed for near real-time search and analytics operations on the stored data. It is a document oriented storage engine allowing for a more flexible storage approach compared to relational databases. Schemas are allowed to change making it possible to adapt to changes in the log file formats without having to rebuild the whole database.
As an alternative it is possible to use a mongodb [17] storage solution. This is a NoSQL database focused on storage, not on search and analytics operations. Kibana does not work with Mongodb. The advantage of Mongodb is that it requires less resources compared to Elasticsearch.

Data analysis

The processing stage prepared the data into a form suitable for basic exploration.
But to produce more deeper analytics further processing of the data was required. The analysis stage queries the storage environment to further process the data.
The toolchain provides access to the storage environment for the following frameworks and environments:

- Apache spark. The Elasticsearch Hadoop plugin [5] makes it possible for Apache spark to interact directly with Elasticsearch. Spark can perform complex data operations on large data sets.
- Jupyter. Using the Elasticsearch-py [6] plugin it is possible to connect to the Elasticsearch environment directly from within a Jupyter notebook. This makes it possible to perform more advanced data operations.Combined with Pyspark [2], the Spark Python API, it is possible to perform in-memory data analysis in a Jupyter notebook using Spark and Elasticsearch.
- R. The Elasticsearchr plugin [7] provides an R interface to Elasticsearch making it possible to interact directly with the data stored in the Elasticsearch environment.

Data visualization

The basic visualizations are handled by Kibana [9], a web based visualization and management tool for Elasticsearch.
Kibana can query Elasticsearch directly and produce graphs such as scatter plots, bar charts and heat maps. This made it possible to efficiently explore insights and questions allowing the user to evaluate if the hypothesis was worth further exploration.

But Kibana was limited in the number of queries it supported and it could not interface with the data analysis tools.

The Jupyter and R notebook environments offered more data exploration and visualization possibilities with tools such as matplotlib and ggplot. These tools came with more freedom compared to Kibana but required more work to produce visualizations.

5.2 Forecast toolchain

This section describes the toolchain used to implement the forecasting experiment. The forecasts represented the number of events in a future period and where based on what was observed in the log files.
The purpose of the toolchain was to produce the forecasts automatically on a time-based schedule.

5.2.1 Forecast implementation

The toolchain consisted of 2 parts, the data processing stage and the forecasting stage.

Figure 5.2: Workflow of the forecast toolchain.

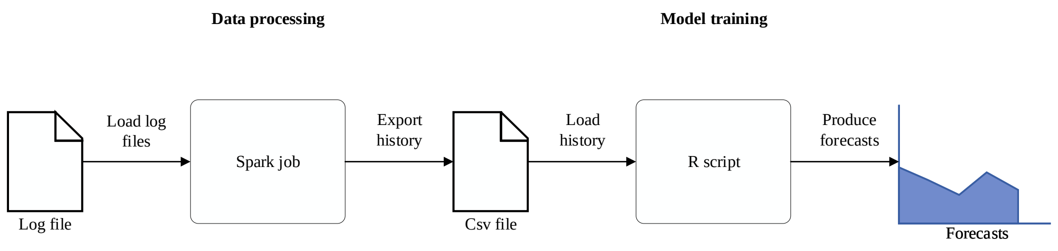


Figure A.1 shows the logic of the toolchain. A cronjob launches the spark job for the data processing. On completion an R script is executed to produce the forecasts.
Data processing was necessary to represent the individual events in the log files as a series of data points indexed by time. These data points represented the access history to the Euro-Argo data infrastructure.

Data processing

The log files where loaded into a Apache Spark dataframe. For each individual event the timestamp was extracted.
Events where then grouped by non-overlapping time intervals based on their timestamp value using a tumbling window operation.

Listing 5.3: Raw log data

```
Mon Jan 1 00:05:12 2017 1 123.4...
Mon Jan 1 00:24:33 2017 4 192.3...
Mon Jan 1 00:54:47 2017 6 888.2...
...
Mon Jan 1 23:52:22 2017 1 102.6...
Mon Jan 1 23:54:22 2017 2 135.2...
```

Listing 5.4: Grouped events

```
2017-01-01 01:00,3 events
2017-01-01 02:00,5 events
2017-01-01 03:00,8 events
...
2017-01-01 23:00,3 events
2017-01-01 24:00,2 events
```

For example, listing 5.3 shows the unprocessed log events and listing 5.4 the counted events after applying the tumbing window operation.
By counting the number of events per time interval a historical sequence of events was extracted from the log files.

Forecasting traffic

The results produced by the spark operation where loaded into an R time series object. Using the R Forecast [14] package a forecast model was fitted on the extracted historical data. This model was used to forecast n periods into the future.

CHAPTER 6

Knowledge extraction from logs

This Chapter presents the analysis of the log files. The first part contains the exploration for patterns in the log files with the goal to answer the first research sub-question: Then the extracted patterns are evaluated for usage in optimisation possibilities to answer the second research sub-question.

6.1 Log analysis

For this analysis the goal is to find patterns and characteristics hidden in the log files. The results will help answer the first sub-research question:

- *What characteristics and access patterns of a data infrastructure can we extract from its FTP logs?*

Multiple box plots are show in this section, these plots use the following convention:

- The whiskers represent the lowest point within 1.5 IQR of the lower quartile and the highest point within 1.5 IQR of the upper quartile.
- Outliers are shown as a black lozenge.

6.1.1 User analysis

To understand how the Euro-Argo data infrastructure was used by the users we start by exploring user patterns and characteristics. A better understanding of how users are using the data infrastructure can help with the interpretation and exploration of others patterns in the data set.
This led to the first hypothesis we explored:

Hypothesis 1

- Hypothesis 1: Users do not revisit the Euro-Argo data infrastructure.

The hypothesis was based on the assumption that most people needed the data for incidental reasons and had no interest of revisiting the data structure for possible updates.

This hypothesis exposed the assumption that it was possible to identify individual users. The FTP log files did not contain reliable user identification values.

The first step then was to determine how a user could be defined in terms of the log files. Two fields in the access log contained possible user traces, the remote-host and the user-name field.

The remote-host field contained the IP address of the user active on the FTP server. However, there are two reasons why a IP address was not suitable for the identification of an individual user:

- Multiple users can share a single IP address because of NAT and proxy protocols.
- A single user may have multiple IP addresses because of dynamic addressing, ASN networks and by switching locations

Identifying users using the username field had a different issue. The Euro-Argo FTP server operates as an anonymous archive site. This means that users don't need to register or identify themselves before downloading files. Each user identifies automatically as anonymous.
While operating in anonymous mode the username field contains the string input of the password field. As a courtesy [11] users can provide their e-mail in the password field for identification. Most FTP clients provide a default identifier if the user does not provide any input in the password field.

Table 6.1: Top 10 usernames seen in the access logs. Usernames with a (*) are anonymized for privacy reasons

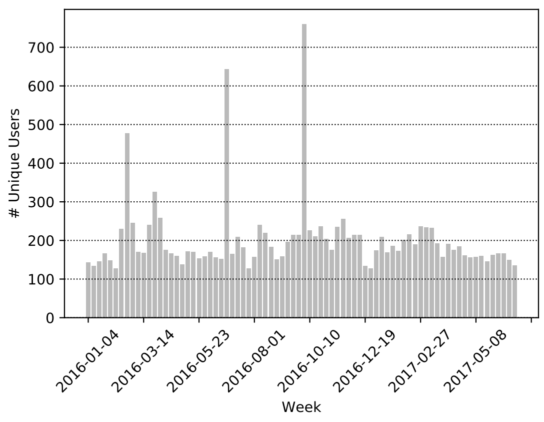
Username	IP addresses	Countries of origin	Total requests	Percentage requests
-wget@	275	36	29987186	26.9
_____@noaa.gov (*)	2	1	19602836	17.6
anon@localhost	182	28	12546398	11.3
flashfxp-user@flashfxp.com	69	2	11698617	10.5
lftp@	16	6	10453675	9.4
anonymous@anonymous.com	20	3	6000675	5.4
__@__.fr (*)	1	1	4355723	3.9
?	63	11	3568743	3.2
joe@	1	1	2384212	2.1
argo@__.edu (*)	1	1	1907693	1.71

Table 6.1 lists the 10 most popular usernames in the Euro-Argo data set. These 10 usernames where responsible for more than 90% of the download requests in total. The Table lists for each username the amount of unique IP addresses and the origin countries associated with the IP addresses. While some usernames had a clear relation between IP address and/or origin most usernames had not. This pattern was similar for all the usernames, 204 in total, in the data set.

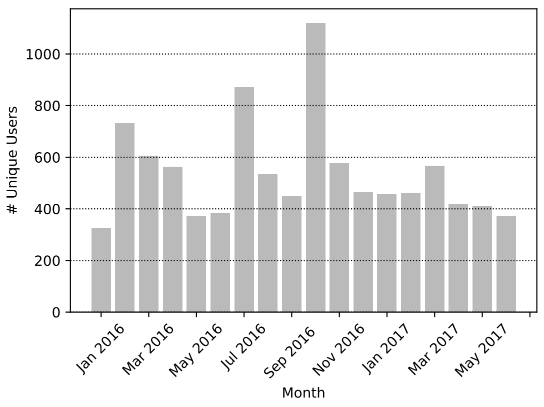
A combination of these 2 fields to define a user did not solve the issue of the many-to-many relationships found with the IP address field or the inconsistent optional user identification.
Based on the information provided by the access logs, it was clear that each of the possible identifiers came with their own set of complications.
To simplify further exploration, we considered each unique IP address as an individual user.
This was the simplest solution available that did not depend on the optional action of a user on the FTP server.

Figure 6.1: The number of unique users per time period

(a) Weekly total unique users



(b) Monthly total unique users



By assuming that a single IP address corresponded with an individual users it was possible to extract a possible visitors pattern.
Figure 6.1 shows the number of unique visitors over a weekly and monthly time period. While there was no reference available the charts did depict a relative stable pattern of unique users visiting the FTP server over the course of 18 months.

Hypothesis 2

By assuming that an individual user corresponded with a single IP address and that users where revisiting the FTP server, we next explored how a typical visit to the FTP server might have looked like.

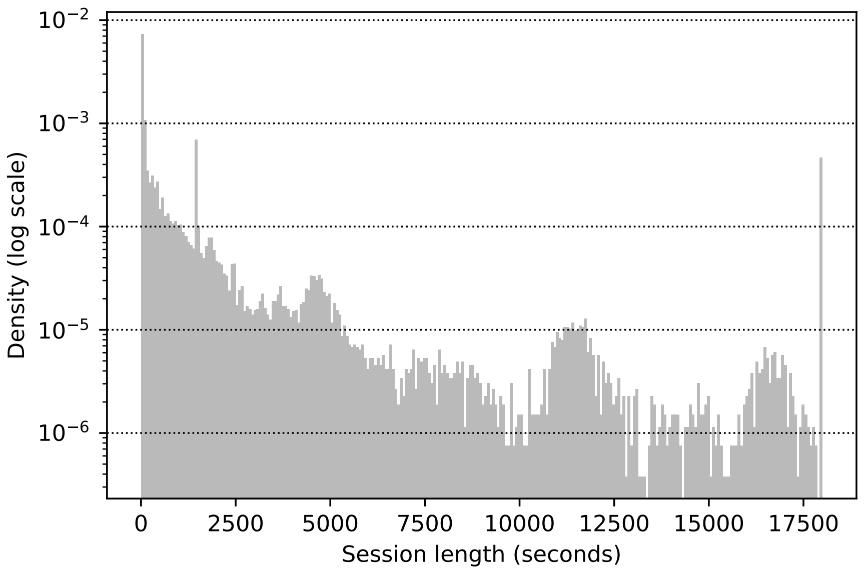
- hypothesis 2: Users do not share similar usage patterns.

The FTP server offers files in multiple formats. Here the assumption was that the users have different usage preferences.

By identifying the actions an individual user performed over a time period, a session, it was possible to extract possible usage patterns. But the log files did not contain session identifiers.

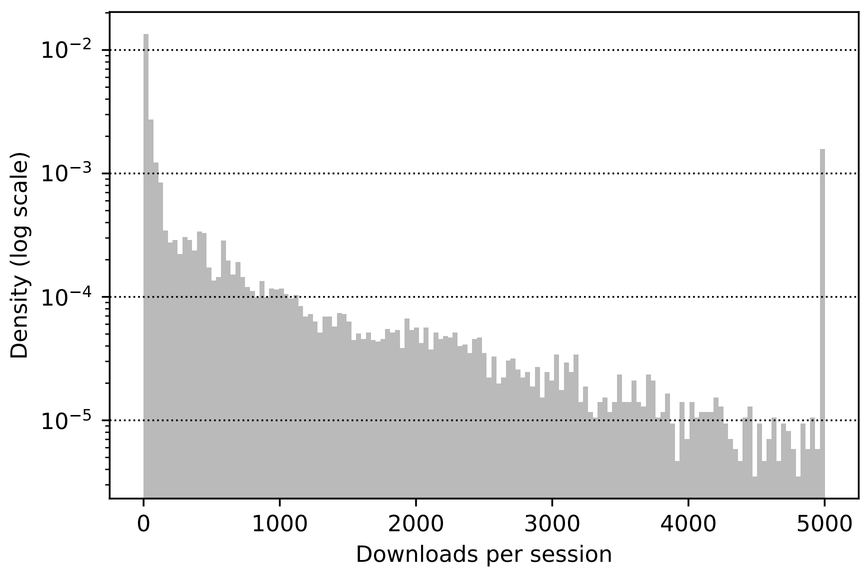
To identify and reconstruct the user sessions a set of criteria was used to define the concept of a session based on the log files. A session is a chain of requests from a single user within a time period of 30 minutes between consecutive completed download requests. The time of the first action was the time stamp in the log file minus the transfer time. With these definitions it was possible to reconstruct how the user sessions may have looked like.

Figure 6.2: Histogram of the length in seconds of each session.



The histogram of Figure 6.2 lists the session times in seconds. While most sessions lasted only a few seconds there was a large peak on the right containing the sessions with a length longer then 120 hours in time.

Figure 6.3: Histogram of the number of downloads in a session.



The histogram of Figure 6.3 shows the number per downloads in a session and has a similar pattern of extremes. Here most users requested a small number of files and a small number of users requested a large number of files.

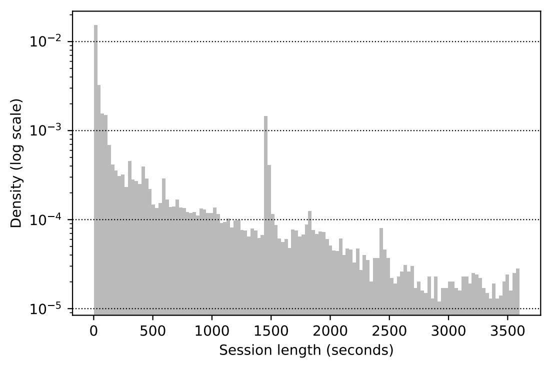
The observed extremes could mean that the set of criteria used to reconstruct the sessions from the access logs where wrong. But a shorter interval of 15 minutes showed similar results. It could also mean the presence of scripted behaviour where a user was downloading files on a short and fixed interval. Or that users where downloading large data sets on a slow connection. But it could also depict perfect normal usage on the FTP server.

Because of the extremes observed in terms of session time and the number of downloads it was difficult to reason about the characteristics of a typical session.

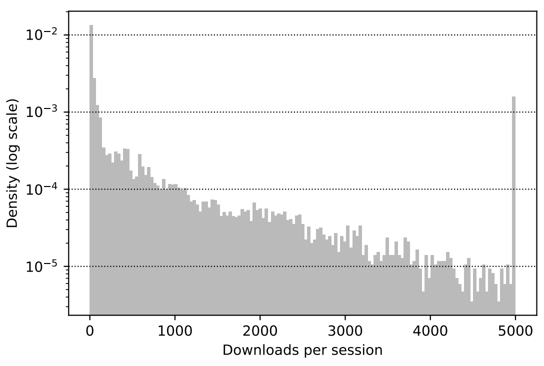
By removing all the sessions with a length longer than 60 minutes a new set of sessions was created containing 97% of the original requests.

Figure 6.4: Histograms of all sessions under 3600 seconds

(a) Session length



(b) Downloads per session



In terms of session lengths it was observed in Figure 6.4a that most sessions where very short. More than half of the sessions where under 60 seconds.

The spike around 1500 seconds is the result of a single user. This user made a repeating number of requests with an interval of 20 minutes, this is likely a scripted set of requests.
The number of downloads per session listed by Figure 6.4b changed substantially. There was a strong pattern where users downloaded a small number of files in a single session. More than half of the total amount of sessions downloaded less than three files in a single session.
This could mean that users visited the FTP server with a clear understanding of what they wanted to download.

The session reconstruction showed that it was possible to observe and reason about possible user behaviour in simple terms of time spent and files requested during a time period. But it was not possible to identify an characterise types of users based on this information.

Hypothesis 3

Rather than looking for the usage patterns of all users, the next approach was to only look at the usage patterns of the most active users.

- Hypothesis 3: Users new to the FTP server are more active compared to the other others.

The motivation for this hypothesis was that new users download substantial parts of the Euro-Argo data set before before they are up to date with their data set and turn into more average active users.

By looking at the top users per month in terms of total data traffic (volume) and requests over the span of 18 months a clear pattern was revealed.

Figure 6.5: Histogram of the top users in terms of volume and requests. The x axis represents the total number of appearances of an individual user in the monthly rankings.

(a) Top in volume

(b) Top in requests

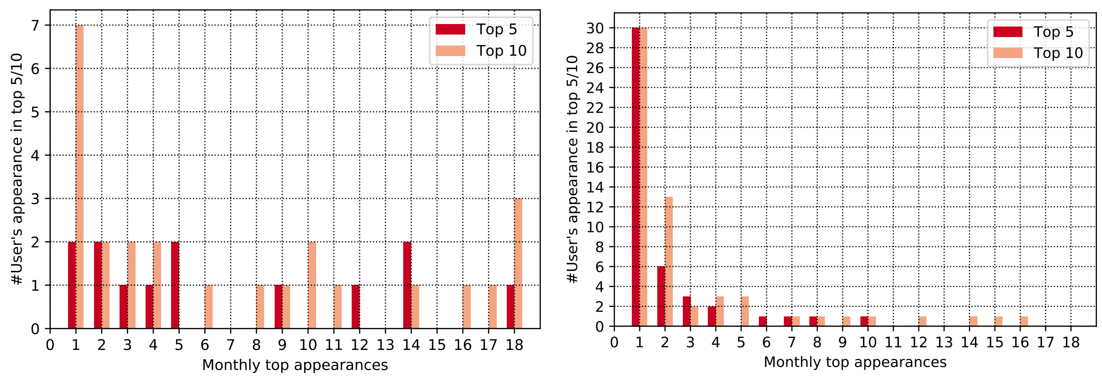


Figure 6.5 shows the amount of times a user appears in the top 5 and/or top 10 ranking per month. The volume ranking contained the top users per month in terms of traffic volume. And the request ranking contained the top users in the number of successful (completed downloads) requests.

The following was observed in Figure 6.5a in terms of volume:
There was not much rotation between users in the top 5. There where only a total of 13 different users where 4 of them appeared more than 11 times. This showed that a small group of users where heavily active over a long and consistent period of time.
The users who only appeared once or twice where not new, one appeared first in January 2017 and the other three have been active since the start of the log files.
By increasing the ranking to 10 users a total of 25 different users appeared. While almost doubling the total amount of users, 5 of the new users only appeared once. Instead what happened was that there was even less rotation among the users due to the loss of competition because now 5 of them appeared 16 times or more.

For the top users in number of requests the following was observed in Figure 6.5b:
The where 45 users in the top 5 results with only a single user with more than 4 appearances. And the top 10 results contained 59 different users with only a single user appearing more than 6 times. It is possible that this is a characteristic for a new or temporary user on the server.

Another observation was that there was a noticeable overlap between the top volume and top requests users. 10 of the 13 users from the top 5 in volume also appeared in the top 10 request rankings. And for the top 10 volume ranking 17 of the 25 users also appeared in the top 10 request rankings.

This was because the top users in volume where mostly interested in individual float files. For example 6 out of the 8 users with more than 3 appearances in the volume top 5 only downloaded float files, 1 downloaded a combination of float and geo files and 1 only geo files.

While it is possible that these users where synchronizing these files for archiving reasons, it was not possible to prove the intent of a user with the information available.

6.1.2 File patterns

This section analysed the possible impact the available files could have on the traffic observed on the FTP server.

File size distribution

To understand how the available files on the FTP server could affect traffic, we started by analysing the file size distribution of the unique files in the log files. The following hypothesis served as a starting position:

- Hypothesis 4: The files stored on the FTP server follows an even distribution of file sizes.

The log files where not very suited to explore this hypothesis. Because the log files only contained the file names of requested files there was no knowledge of the underlying file system. For this analysis each unique file name was considered as a separate file.
This can give a potential distorted view of the reality because files could be renamed to reflect post-processing operations.

Figure 6.6: File size histogram of the 47 million unique files seen in the log files

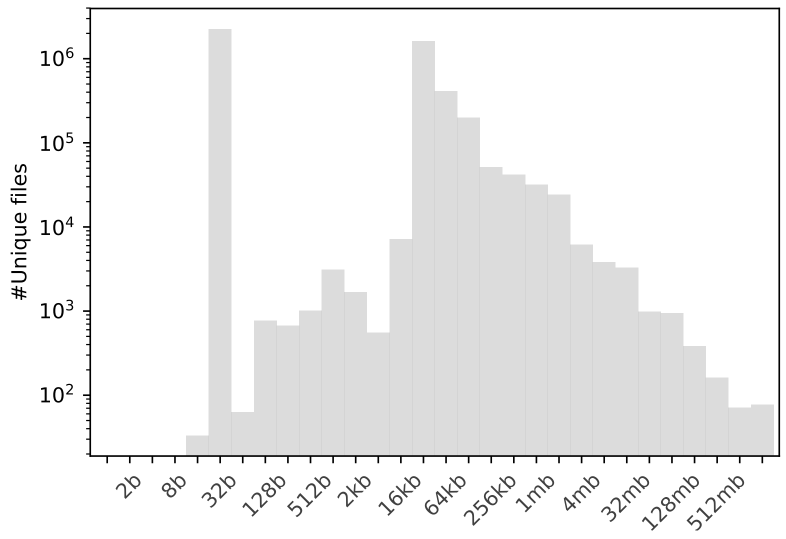
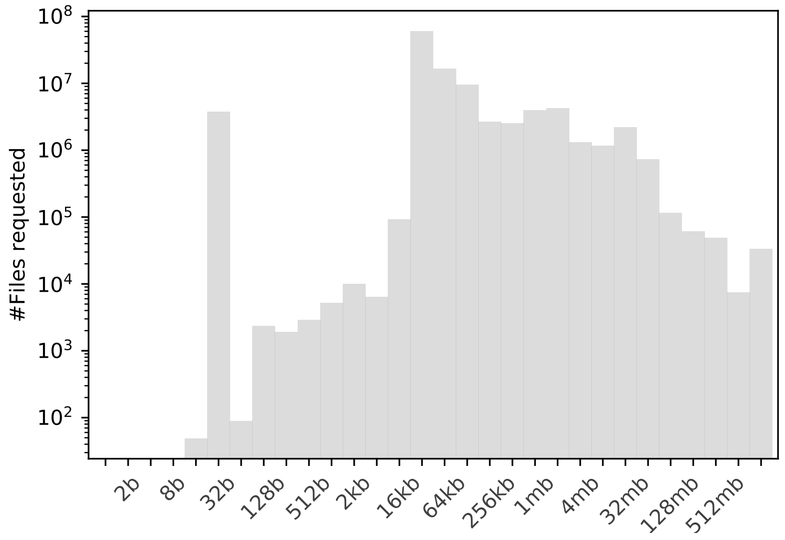


Figure 6.6 shows the presence of a large amount of small files. The files grouped under the 32byte bin are md5 hash files with a size around 50bytes. Md5 files are generated for most files on the server and can be used to verify data integrity, in total almost half of the files on the server are md5 files.
The 32kb bin contains all the individual float files. Together these 2 type of files represented more than 97% of the separate files reconstructed from the log files

Figure 6.7: File size distribution of all download requests



In Figure 6.7 the histogram shows the file distribution of each unique file with the total request number as its associated weight. This depicted very clearly how the individual float files, contained in the 32kb bin, shaped the overall traffic observed on the server.

Hypothesis 5

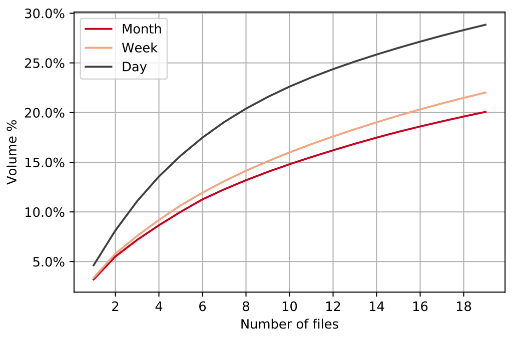
With the knowledge of the distribution of the file sizes the following analysis focused on the impact individual files could have on the server traffic. This raised the next hypothesis:

- Hypothesis 5: Due to the large number of files on the FTP server, individual files don't have a measurable impact on the traffic.

By selecting the n files responsible for the largest amount of data traffic generated over a specific time period the several patterns where observed.

Figure 6.8: Figure a shows the impact individual files can have on the data traffic,Figure b shows the corresponding file request traffic of these files.

(a) Average file volume traffic



(b) Average file request traffic

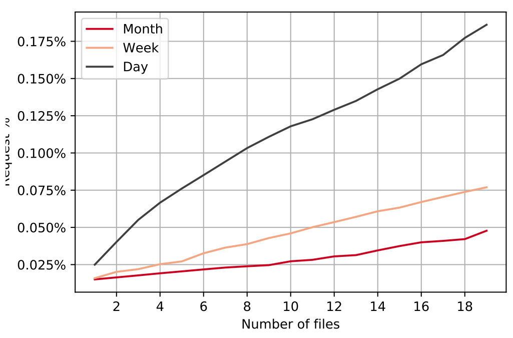


Figure 6.8a lists the average amount of data volume generated by the top n files per time frame over a period of a day, a week and a month. Keeping in mind that there where more then 47 million different files in the log files it was noticeable how a small number of files could generate a large amount of the total data volume.

All three periods showed a pattern where a small number of files could generate a substantial amount of the total data volume over a single period. The weekly and monthly periods showed a similar trend of the impact the number of files had on the volume.
For the daily period the impact of the individual files was larger, in particular for the first number of files in the top 20.

Figure 6.8b lists the average amount of requests for the top files.
The request percentages are much lower for these files compared to the volume percentages. A possible explanation for this was that the selected files where much larger in size compared to the other files.
The daily request traffic is much higher in percentage compared to the other 2. This showed a possible pattern of short-lived file popularity, a short period with a large number of requests for a file followed by a loss of interest by the users. The shorter time period, such as a day, kept rotating the popular files in the top n rankings explaining why the daily top numbers where higher compared to the other 2 periods .
The variety in sizes for bigger files could explain why a doubling in request percentage did not corresponded with a doubling in volume percentage compared to the other periods.

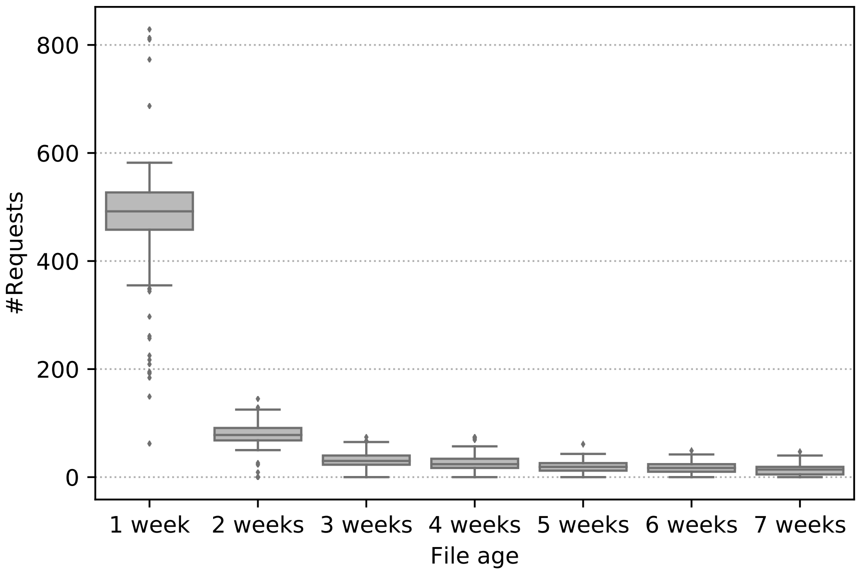
Hypothesis 6

Files observed in the previous hypothesis came from 3 different sources on the FTP server. The first source, the index files found in the root of the FTP server, contributed a small number of files. Index files contain meta information about the file content available on the server.
The second source of files where merged profiles of individual float files. The third source where files from the geographical source.
To explore why some files where so much more popular than others, we analysed the following hypothesis:

- Hypothesis 6: File popularity is linked with the age of the file. New files receive the most requests.

Here we only analysed files from the geographical source because for the other sources it was not possible to reliably track individual file usage over time using the log files.

Figure 6.9: Impact of file age on popularity, expressed as the number of request per weekly period



The files from the geographical source appeared relatively popular within the first week of distribution as shown by Figure 6.9.
A download request rate around 500 for a file in a single week is a high number for the FTP server. This result does question the assumption of a single user equalling a single IP address in the user analysis part. There it was observed that there where on average 200 individual users visiting the FTP server.
The download request rate drops substantially after the first week and keeps getting nearer to zero week after week.

This showed that file popularity, at least for files from the geographical source, is determined by the age of the file.

6.1.3 Traffic analysis

This section analysis the log file from the traffic perspective. The analysis is based on the traffic to the server between the period of 2017-01-01 and 2017-06-30. Times shown are in Coordinated Universal Time.
A subset of the available data set was chosen because in practice it was not possible to detect patterns over longer periods of time, instead it added only noise to the knowledge gathering process.
The introduced noise made it more difficult to identify or interpret possible patterns and characterisations over longer periods of time.
Daily traffic

The analysis starts with the aggregated traffic in terms of requests and data observed on the daily level over the course of 6 months.

- Hypothesis 7: The traffic in terms of requests and data are directly related to another.

The idea behind this hypothesis was that more request traffic correlates with more data traffic and vice versa.

Figure 6.10: The shape of the daily traffic during the first six months of January 2017

(a) The request traffic

(b) The data traffic

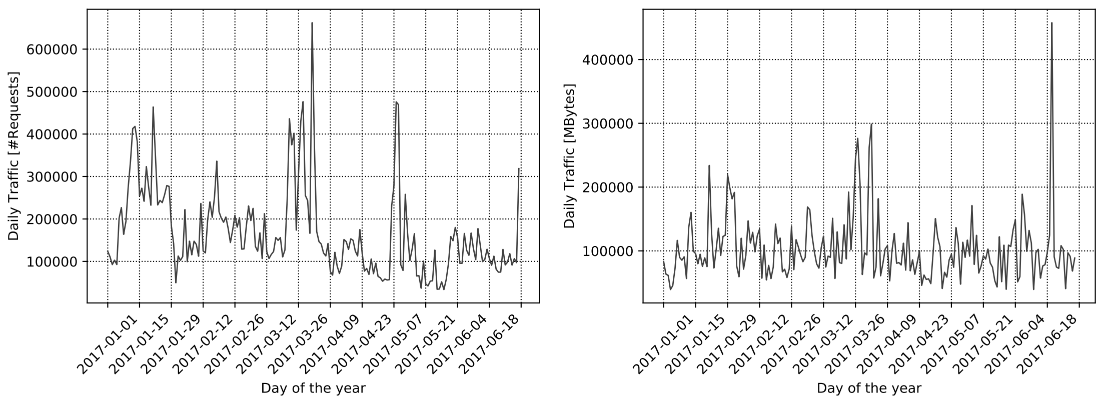
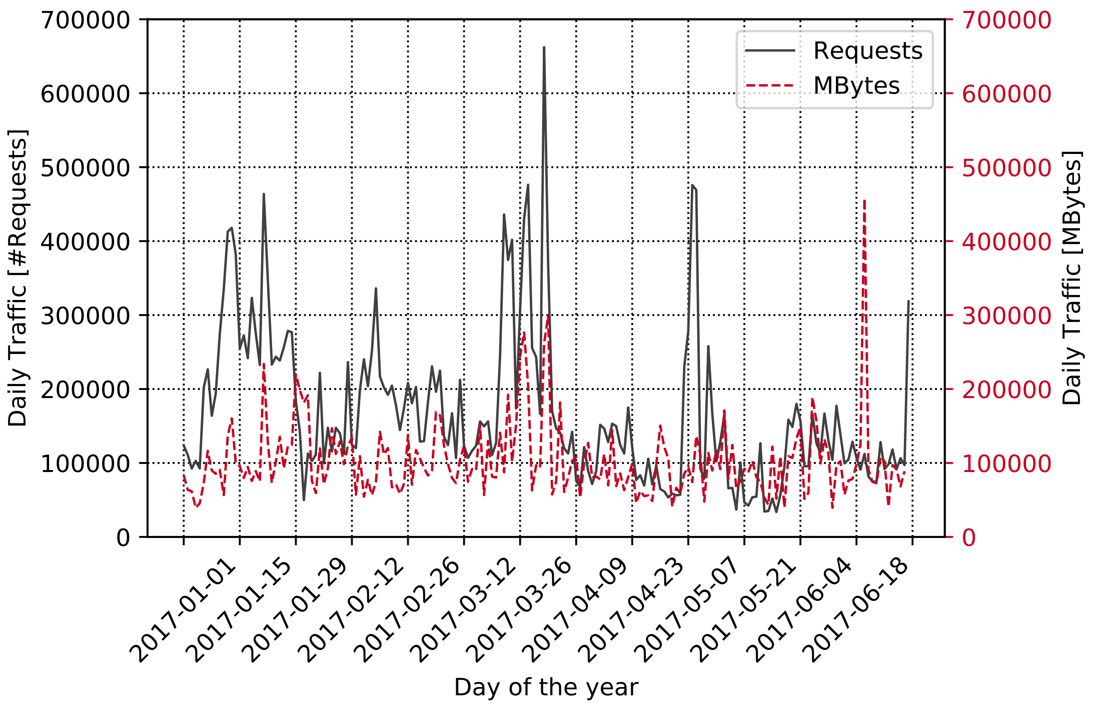


Figure 6.10a showed no clear pattern in terms of request traffic. It was not possible to observe a pattern of a working week or to detect changing traffic patterns over time. For the data traffic in in Figure 6.10b these 2 patterns where not observed either.
But it was noticeable that the request and data traffic did not had to show similar behaviour on the daily level.

Figure 6.11: Comparison of the data and request traffic



A change in activity with the request traffic did not have to correspond with a similar change in data traffic and vice versa. This is shown in Figure 6.10. In general there where 3 different types of responses observed between the request and data traffic

- A similar response where the changes in traffic correspondent with each other but not necessarily on the same scale. This is clear to see around the second week of January where both the data and request traffic change over time.
- Inverse response where the increase in traffic in one correspondents with the decrease in traffic of the other. For example near the third week of January there is a steep decline in request traffic while the data traffic shows growth.
- No response in change. Here a change in one does not lead to a noticeable change in the other. A clear example is between 4 and 18 June. There is a maximum observed in the data traffic but the request traffic remains unchanged near the lower end of the graph

These 3 differences between request and data traffic showed that it was not possible to conclude that there is a clear correlation between the 2.
A possible explanation lays in the fact that most files on the server where small in file size. The request traffic considered each request as equal but the data traffic was determined by the size of the requested file. Here a small number of requests for large files could heavily influence the data traffic.
This meant that it was not possible to identify a clear correlation between request and data traffic.

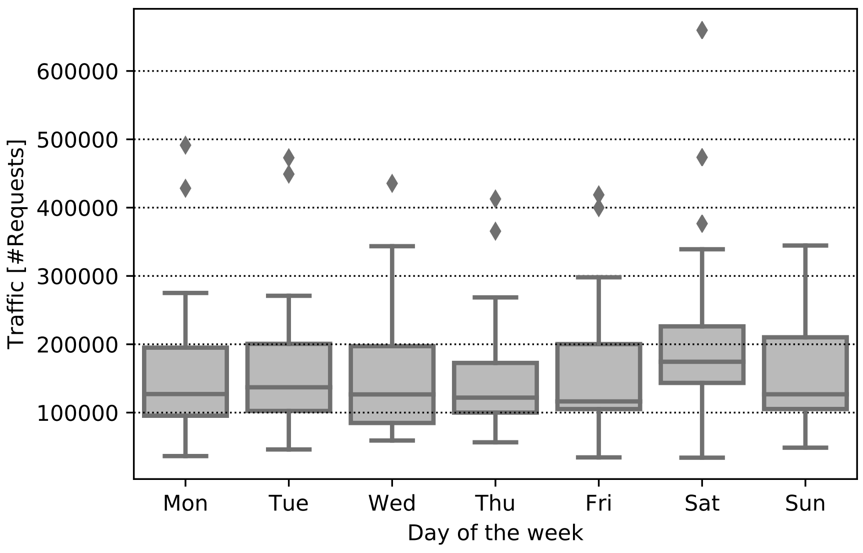
Hypothesis 8

To further explore for possible effects of time on the traffic the next analysis focused on the possible impact the day of the week can have on the server traffic.

- Hypothesis 8: There is a decrease in traffic to the FTP server on the weekends.

Here the assumption was that users visited the FTP server because of their profession. When they are not working in the weekends they have less reason to visit the FTP server. **Request traffic**

Figure 6.12: The observed request traffic during the first 6 months of 2017. Along the x-axis the traffic collected on each day of the week is given.



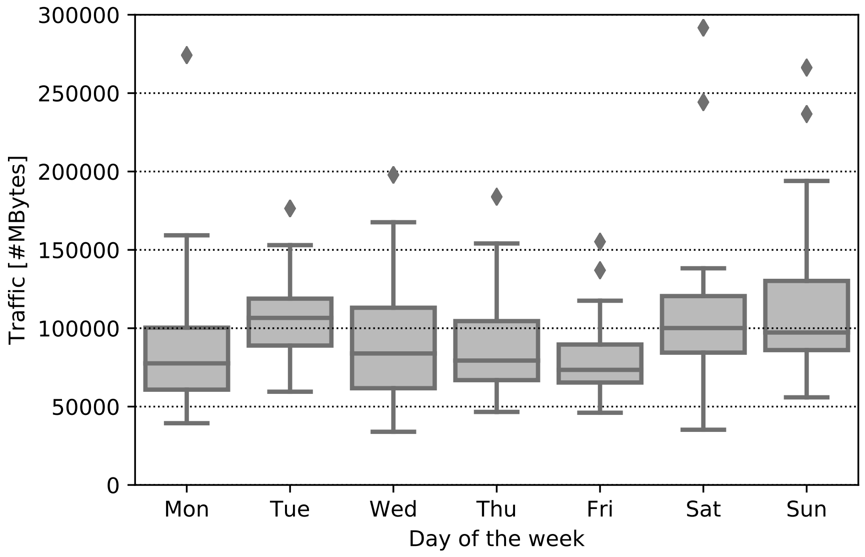
The box plot in Figure 6.12 contains 181 data points representing the total amount of requests received on a day of the week.
Several points of interest where observed:

- the box for Saturday has a median around 180000 while the other days have the median centred around 120000.
- Saturday and Tuesday have a relative symmetrical distribution of requests (excluding the outliers) compared to the other days who all more skewed to the right.
- The IQR of the Saturday and Thursday box is relatively short compared to the rest of the days who are similar in IQR and position.

Based on these observations it was not clear to determine whether the points of interest where influenced by a day of week effect such as a work week or a weekend.

Data traffic

Figure 6.13: The observed data traffic during the first 6 months of 2017. Along the x-axis the traffic collected on each day of the week is given.



For the data traffic shown by Figure 6.13 the following was observed:

- The 5 week days do not share clear similarities based on the boxes, medians and whiskers.
- On Saturday and Sunday the median is centred at a similar height. But on Sunday there is more variability in the box length and in the lower whisker.
- Between Tuesday and Friday the traffic shows a slight downwards trend.

There was no strong pattern observed in the data traffic.

Regional request traffic

To detect possible regional effects, the weekly traffic was analysed from the perspective of the 3 major continents. These 3 continents, North America, Europe and Asia, are together responsible for more than 90% of the total request and data traffic.

Figure 6.14: The request traffic from the 3 dominant continents during the first 6 months of 2017.

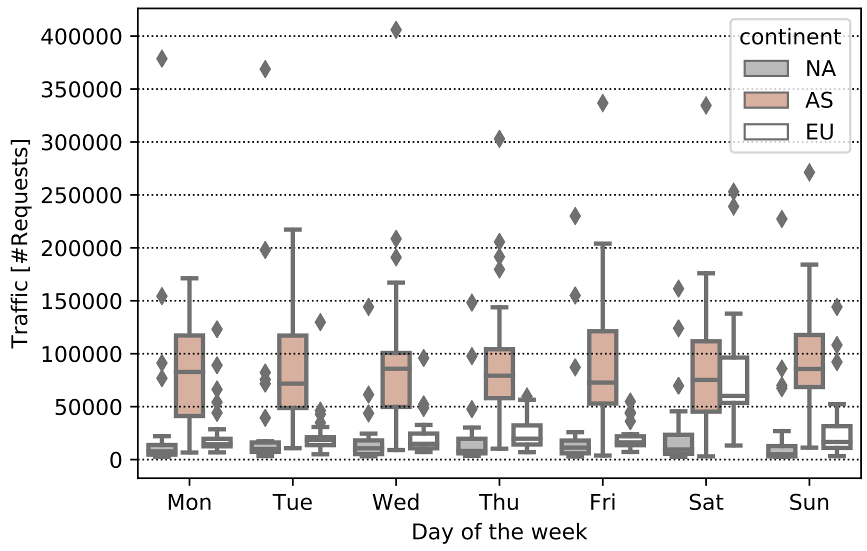


Figure 6.14 shows the box plot of the 3 dominant continents, from this the following was observed:

- On Saturday there was a huge increase of traffic originating from Europe. This is the only day of the week where the boxes of Europe and Asia where on a similar level. Combined with the height of the North American box this explained the high median value on Saturday in the previous request traffic Figure 6.12.
- On Wednesday the Asian box shows a distribution with a skew to the left, something that did not happen on the other days.
- A more general observation is the fact that while the median of the Asian box is centred around the same value the lower and upper quartiles showed much variation over the 7 days.
- North-America showed little variation over the 7 days compared to the other 2 continents.

These observations show that there are some regional and weekly patterns in the request traffic:

- Asia generated the most request traffic on each day of the week.
- Europe had a major increase in requests on Saturday.
- North-America is evenly spread over the week compared to the other continents.

Regional data traffic

Figure 6.15: The data traffic from the 3 dominant continents during the first 6 months of 2017.

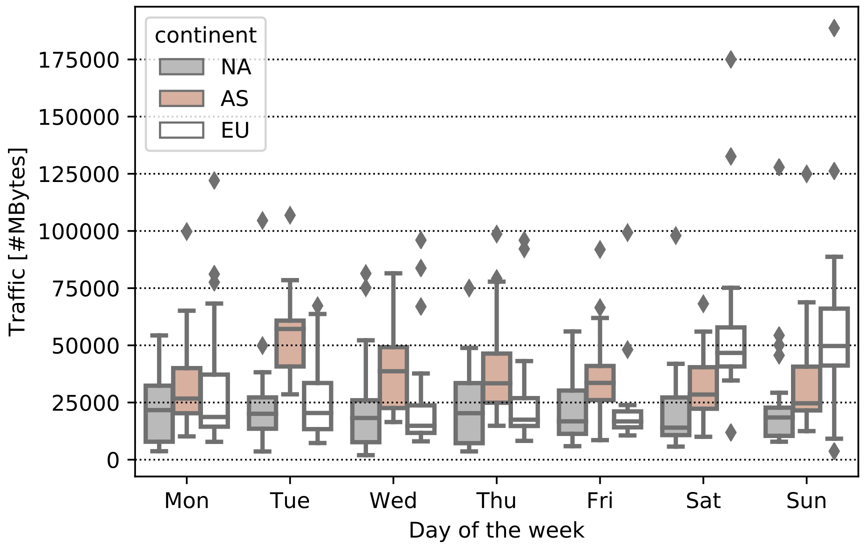


Figure 6.15 shows the data traffic from the 3 dominant continents:

- In the weekends Europe was very active with a gradual decline during the days after the weekend.
- Asia was active during all days and had a noticeable increase in MBytes on Tuesday.
- North America did not show a clear pattern.

The results of this analysis showed that there are patterns in the traffic influenced by time or region.

Hypothesis 9

To further analyse the impact of time and region we explored the server traffic per hour of day. We raised the following hypothesis:

- Hypothesis 9: There is an increase in traffic to the server during working hours.

Here the assumption was that users mainly visited the FTP server during working hours.

Request traffic

Figure 6.16: The observed request traffic during the first 6 months of 2017. Along the x-axis the traffic collected per hour of the day is given. The plot contains around 4200 observations.

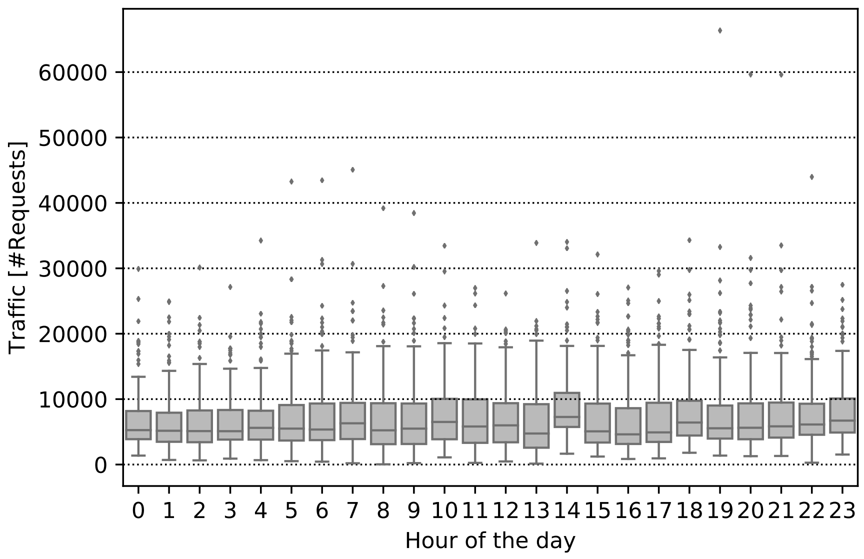


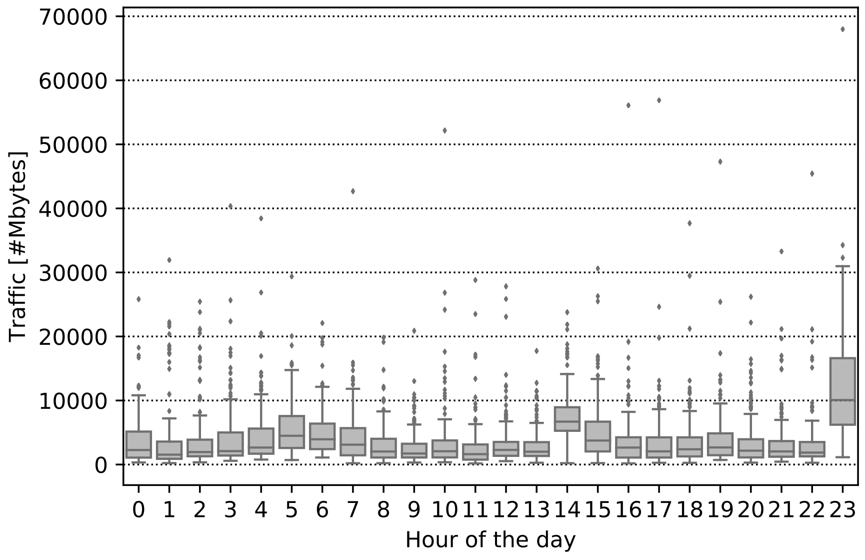
Figure 6.16 shows the number of requests received grouped by hour, the following was observed:

- All the median values are around the same center with the exception of the box at 14:00 hour.
- All the median values shared a similar center (with the exception of the box at 14:00 hour).
- The boxes during the first 5 hours of the day all shared a similar distribution of requests.
- During the rest of the hours the boxes showed little variation in IQR size and height (the box at 14:00 excluded).

At the hourly level the request traffic appeared relatively stable. There was no clear pattern of peak times in the traffic during office hours or downtime during the night.

Data traffic

Figure 6.17: The observed data traffic during the first 6 months of 2017. Along the x-axis the traffic collected per hour of the day is given.



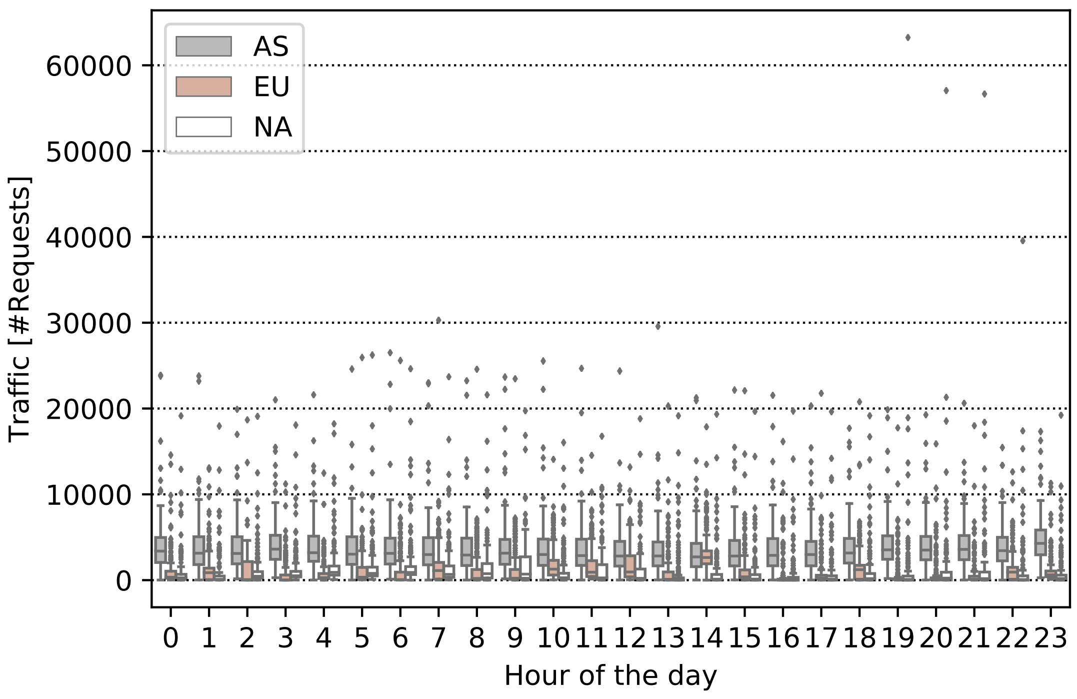
Looking at the data traffic in Figure 6.17 the following was observed:

- During the last hour of the day there was a big increase in IQR size and height compared to the other hours.
- The other boxes showed more variety in data distribution compared to the request traffic.
- The box at 14:00 hour showed similar behaviour when compared to the request traffic.

From the overall traffic perspective it was difficult to interpret why the request traffic appeared stable during the day whereas the data traffic showed more variation.

Regional request traffic

Figure 6.18: The observed request traffic from the 3 dominant continents during the first 6 months of 2017. Along the x-axis the traffic collected per hour of each continent is given.

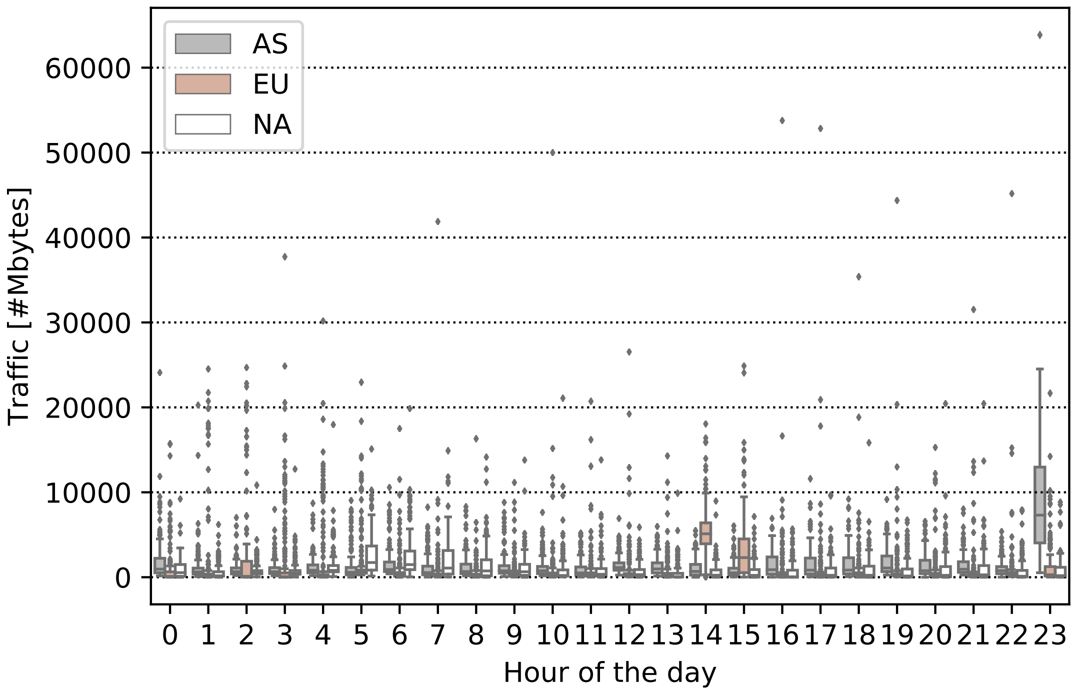


From Figure 6.18 the following was observed:

- Traffic from Asia showed very little variation during the day. Requests are coming continuously irrespective of the time of the day.
- Europe had a small increase in traffic between 02:00 and 12:00.
- North-America showed a subtle increase in traffic during the first 12 hours of the day.

Regional data traffic

Figure 6.19: The observed data traffic from the 3 dominant continents during the first 6 months of 2017. Along the x-axis the traffic collected per hour of each continent is given.



The data traffic in Figure 6.19 did show some increased activity during certain hours of the day:

- Asia had a large peak at 23:00.
- Europe showed an increase of data traffic around 14:00.
- North-America had a slight increase between 05:00 and 08:00.

Based on the request and data traffic observations it was not straightforward to characterise traffic behaviour that depends on the time of the day.

The request traffic remained stable during the day showing little effect of local working hours and day and night patterns.
On the continent level there where some signs of users preferring certain hours of the day but those where very weak.

In particular traffic from Asia did not seem to be influenced by local working hours or day and night patterns.
This may suggest the presence of automated processes generating most of the requests but this is impossible to proof.

The data traffic showed more variation during the hours of the day.
A possible explanation might be that non-automated users preferred a specific type of file much larger in size such as the files representing a collection of measurements. A small increase in requests, such as 500, does not lead to a noticeable difference in request traffic but can lead to a much higher outcome in data traffic.

6.2 Optimisation possibilities

The knowledge extracted by the log analysis was used for the evaluation of the second sub-question:

- *What optimisation strategies can be proposed based on the access log files?*

The analysis and exploration in the previous section extracted multiple patterns and characteristics. From these we identified two that could possibly be used in a optimisation strategy.

The file hypothesis highlighted how a select number of files could drive traffic in terms of volume and requests. This information can possibly be used for a caching strategy where popular files are loaded into a cache or old files are archived on a cheaper storage medium.
But the problem is that the ftp log files cannot reflect what happens with the files on the file system level. This can lead to a mismatch to what is observed in the ftp log files and what actually happens on the server. Designing a caching strategy based on the log files alone cannot function as a reliable approach.

The second pattern observed was with the hourly request traffic.
The hourly traffic appeared stable during the day over the course of 6 months.. A possible optimisation based on this pattern is to forecast the traffic to the Euro-Argo data infrastructure in the near-future. The forecast traffic could be used to dynamically provision the resources necessary to handle the future traffic.

This can be used to provide a better and more efficient service level to the users of the Euro-Argo data infrastructure.

A big advantage of this strategy is that it only depends on the timestamp field on the log files. There are no assumptions or interpretations necessary for the extraction of the daily traffic patterns based on the timestamp.

CHAPTER 7

Forecasting service performance using logs

This chapter presents the experiment conducted to determine the effectiveness of multiple forecasting models to forecast the future traffic to the data infrastructure. Several forecasting models where compared under multiple scenario's to measure their ability to produce accurate forecasts.

7.1 Forecast experiment

The analysis in the previous chapter showed that the hourly request traffic contained patterns that potentially could be used within a optimisation strategy. This experiment evaluates if the observed traffic is a suitable source of information that can be used to produce reliable forecasts. Based on the observed traffic the experiment will attempt to forecast the number of download requests in the future received by the data infrastructure.

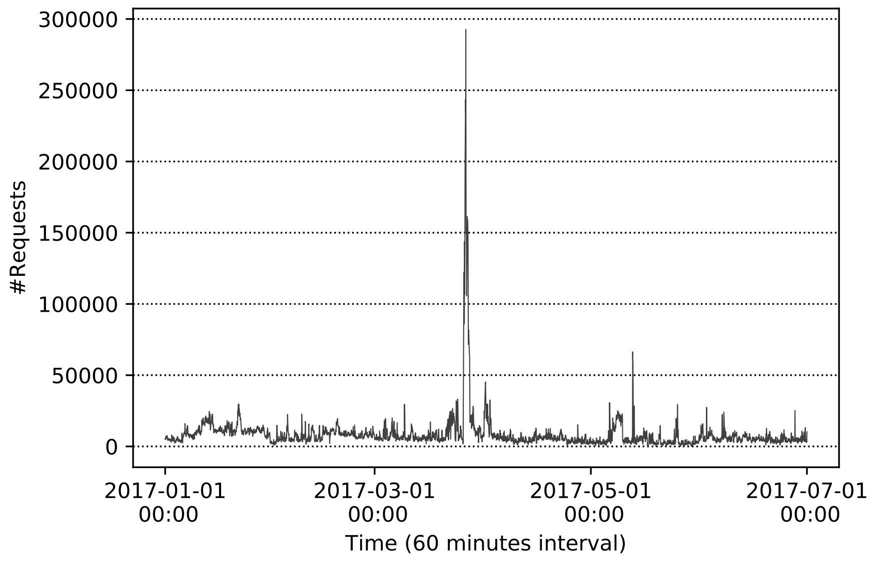
To measure the effectiveness of the forecasting approach 3 different models where evaluated, these where:

- Naive model
- Mean model
- ARIMA model

7.1.1 Data set

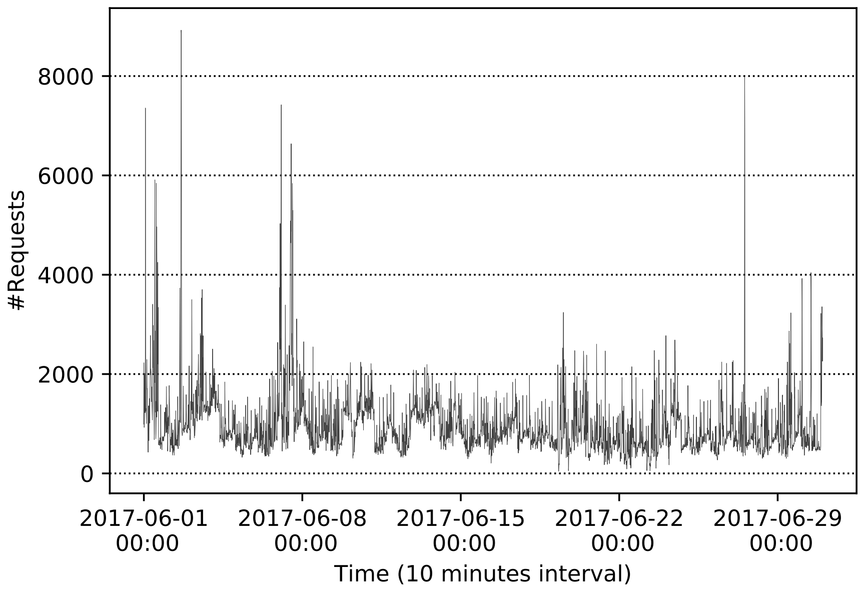
The experiment used two different data sets both containing a sequence of observations at equally spaced points in time, a time series data set.

Figure 7.1: Data set for the first experiment. The data set contains the total number of requests per 60 minutes for the first 6 months of 2017.



The first data set, Figure 7.1, contained the number of requests per hour collected over the period from 2017-01-01 until 2017-07-01.

Figure 7.2: Data set for the second experiment. The set contains the total number of requests per 10 minutes for the month of June in 2017.



The second data set, Figure 7.2, contained the number of requests per 10 minutes collected over the period from 2017-06-01 until 2017-07-01. We included this data set to observe if an even smaller time interval could produce better forecasts.

7.2 Accuracy

For the experiment 2 different tests where performed.
The first test measured the accuracy of each model to produce a number of h forecasts into the future, the horizon, starting from a forecast horizon of $h = 1$ up to a horizon of $h = 24$. Each forecast will be compared to the actual observed value contained in the testing set.

The second test measured the impact of the length of the training set. While it is possible to use all available data points for the training of the forecast model this does not guarantee the most accurate results.
The time patterns represented by the data points can change over time. For example the traffic to the FTP server may change due to the increase of available files, new users discovering the data infrastructure or old users changing their download behaviour.
For this experiment we where interested in forecasting the expected traffic in the near future. Only including the last 3 weeks of observed traffic instead of the last 18 months might produce more accurate forecasts. To determine the optimal training set length all forecasts where produced multiple times, each with a different training set length.

7.2.1 First data set

This section presents the results for the forecast experiment on the 60 minutes interval data set.

Table 7.1: Average MAPE (%) scores of the rolling forecasting procedure with a training length of 24 and 48 data points.

Training length	**Forecast horizon (h)**	**ARIMA**	**Mean**	**Naive**
24	1	46.14	49.25	26.13
	2	46.11	50.25	30.62
	4	49.32	52.01	36.40
	6	220.18	53.49	41.15
	8	222.74	54.83	44.6
	12	226.44	57.07	49.51
	24	239.23	61.93	58.71
	Average	150.02	54.11	41.01
48	1	27.79	57.70	26.18
	2	32.01	58.27	30.69
	4	36.84	59.30	36.49
	6	40.62	60.20	41.26
	8	43.24	61.04	44.77
	12	47.00	62.56	49.66
	24	55.11	66.27	58.88
	Average	40.37	60.76	41.13

The accuracy results produced by the 3 models trained on a training set with a length of 24 and 48 are shown in Table 7.1. Starting with the training length of 24 the following was observed. With a forecast horizon of $h = 1$ the mean model scored the worst with score of 49.25% closely followed by the ARIMA model with 46.14%.
The naive model gained a score of 26.13%, this despite being the most basic forecasting model.

By increasing the forecast horizon the ARIMA model gained a big accuracy loss with a forecast horizon longer than 5. This was the only time that a model saw such a big accuracy drop in the experiment.

The naive model achieved the best accuracy score at each forecast horizon. The mean model did got close on the $h = 24$ horizon.
Despite the bad start the mean model scaled better under increasing forecast horizons lengths compared to the other models and at $h = 24$ the mean came close to the performance of the naive model.

Next we increased the training length to include the 48 most recent data points.

Here the naive model showed a minor accuracy loss at each forecast horizon.
The mean model did not improve either, all forecasts produced worse accuracy results compared to the results of the previous training length.
But the ARIMA model showed great improvement, the average error score dropped from 150.02% to 40.37%. With a short forecast horizon, below $h = 6$, the naive model had a slight accuracy advantage but above that the ARIMA model outperformed the naive model.

The increase from 24 to 48 hourly data points led to a big improvement for the ARIMA model. But for the mean model the extra data points available available caused an increase in the error score.

Table 7.2: Average MAPE (%) scores of the rolling forecasting procedure with a training length of 72 and 168 data points.

Training length	**Forecast horizon (h)**	**ARIMA**	**Mean**	**Naive**
72	1	26.97	62.93	26.23
	2	30.96	63.35	30.74
	4	35.82	64.11	36.54
	6	39.33	64.79	41.32
	8	41.72	65.42	44.84
	12	45.57	66.56	49.73
	24	52.86	69.63	59.01
	Average	39.03	65.26	41.20
168	1	26.02	74.69	26.46
	2	29.90	74.89	30.99
	4	34.81	75.28	36.83
	6	38.36	75.65	41.67
	8	41.03	76.01	45.24
	12	44.84	76.72	50.21
	24	52.78	78.71	59.64
	Average	38.25	75.99	41.58

Table 7.2 shows the results for the training lengths of 72 and 168.
The increase in data points gave the ARIMA model a small increase in accuracy at each forecast horizon.
On the forecast horizons larger than 2 the ARIMA model outperformed the naive model.
The mean model continued to lower in accuracy for each forecast horizon h.

Finally we included 168 data points in the training set, representing 7 days of hourly data.
The ARIMA model gained another small increase in accuracy due to the increase of data points.
With the 168 training length the ARIMA model outperformed all other models at each forecast horizon.

The naive model suffered a minor decrease in accuracy. The mean model was again the worst performer.

The two best MAPE scores where produced under different conditions.
The best MAPE score, 26.02% was produced by the ARIMA model using a training length of 168 and $h = 1$. But the second best score, 26.013% was produced by the NAIVE model with a training length of 24 and $h = 1$.
For short forecast horizons the accuracy of the naive model is comparable to an ARIMA model.

7.2.2 Second data set

This section presents the results for the forecast experiment on the 10 minutes data set.

Table 7.3: Average MAPE (%) scores of the rolling forecasting procedure with a training length of 24 and 48 data points.

Training length	**Forecast horizon (h)**	**ARIMA**	**Mean**	**Naive**
24	1	26.11	38.71	26.38
	2	29.38	39.33	31.29
	4	32.83	40.28	36.57
	6	34.73	41.00	39.57
	8	35.96	41.59	41.57
	12	37.95	42.64	43.84
	24	41.86	44.91	48.23
	Average	34.12	41.21	38.21
48	1	24.90	42.11	26.38
	2	28.12	42.45	31.29
	4	31.40	42.97	36.57
	6	33.26	43.36	39.59
	8	34.81	43.67	41.60
	12	37.03	44.15	43.87
	24	41.37	45.19	48.27
	Average	32.98	43.41	38.23

Table 7.3 shows the results for the 3 models trained on the data set with 24 and 48 data points.
Here the naive model had a similar ∼26% accuracy score for $h = 1$ compared to the 60 minutes data set. The mean model was the least accurate of the 3 models on the lower forecast horizons. The ARIMA model produced the most accurate forecasts of the 3 models on all the forecast horizons. Here the short training length did not degrade the accuracy.

By increasing the training length to 48 data points the ARIMA model gained accuracy improvements at all forecast horizons.
The naive model showed no change in accuracy. The mean model was the worst of the 3 models except at $h = 24$ where it outperformed the naive model.

Table 7.4: Average MAPE (%) scores of the rolling forecasting procedure with a training length of 72 and 288 data points.

Training length	**Forecast horizon (h)**	**ARIMA**	**Mean**	**Naive**
72	1	24.43	42.51	26.33
	2	27.49	42.72	31.22
	4	30.64	43.04	36.43
	6	32.45	43.27	39.37
	8	33.68	43.47	41.32
	12	35.56	43.84	43.64
	24	39.08	44.90	48.04
	Average	31.90	43.39	38.05
288	1	23.30	48.71	26.16
	2	26.15	48.78	30.89
	4	28.92	48.90	35.94
	6	30.44	49.01	38.83
	8	31.61	49.12	40.75
	12	33.10	49.30	43.01
	24	35.76	49.78	47.25
	Average	29.90	49.09	37.55

Table 7.4 shows the results for the 3 models trained on the data set with 72 and 288 data points.
A training length of 72 points represented 12 hours of observations. The ARIMA model was the most accurate for all 24 forecast horizons.
The mean model showed little variation of accuracy for the different horizon h values, the model performed relatively good on the longer forecast horizons compared to the naive model.
The naive model showed no performance differences.

The training length of 288 points represented 48 hours of observations.
The ARIMA model earned a MAPE score below 24% with a forecast horizon with $h = 1$ which was the lowest score observed.
The naive model gained a small increase in accuracy at each forecast horizon. The mean model produced the worst forecasts but here the variation between the MAPE scores for different forecast horizons was even smaller compared to what was observed at the previous training length.

With this data set the ARIMA model performs the best out of the 3 models. The model produces the best short-term and long-term forecasts at each training length. To get the most performance out of the ARIMA model a longer training length is required.
A naive model can be an alternative for short training lengths with a short forecast horizon.

CHAPTER 8

Discussion

This chapter discusses the results and conclusions of the analysis and experiments performed on the log files. The first section contains the discussion for the analysis and the second section the forecast experiments.

8.1 Analysis

The log file analysis was guided by a hypothesis driven approach using multiple perspectives.

From the file perspective the observations where clear and mostly satisfying. Despite the initial concerns that the log files could not reflect file operations such as file deletions, creations or rename actions depending on the file names alone produced interesting results.
Examples included that most files only have a short period of user interest or that a small number of files could have a observable impact in terms of data volume.

But these results where more useful for analysis purposes then for optimisation strategies. While possible optimisations can be simple in terms of execution, such as caching files with a file size over n megabytes for the first week after distribution. But without access to the underlying file system, the log files cannot depict the actual file usage of the data infrastructure making them unreliable as the only source of information for caching optimisations.

The traffic analysis delivered visible results as well.
Here we showed that there where periodical and regional patterns in the traffic history from the data infrastructure. It was, however, impossible to interpret what the driving factors might be behind these patterns using the information available.
That is why the possible answers for the traffic analysis consisted mostly out of observations and not out of reasons or motivations. The logs cannot reveal for example why there was an outlier in the traffic observed at 14:32 originating from Asia.

And the analysis failed to explain if the traffic contained trends that changed over time. To spot such trends more data is needed, at least 2 years, to compare different periods with each other.
And even then this could be difficult because single IP addresses, assumed to be individual users, had a big impact on the overall observed traffic. This leads to a distorted and/or subjective view about what normal traffic is and how it should look like over a period of time.

But the traffic was chosen as the target for an optimisation strategy in the experiment. The fact that the observations contained clear patterns that presented themselves without interpretation made it a suitable target for experimentation.

The user analysis proved to be the most difficult part. Basic questions such as "What is a user?" had no clear answer, merely interpretations.
This made it difficult to reason about usage characteristics and patterns given that there was no clear understanding who and/or what is visiting the data infrastructure.This is not a problem that is solvable with the current FTP infrastructure, the log files are not designed for that purpose.

But treating unique IP addresses as individuals users did produce results. There was only a small number of users active on the data infrastructure and most of the traffic was generated by a small subset of those users, both with little change over time.

But while interesting, the user patterns depend on assumptions to produce results. This made them unsuitable for optimisation possibilities.

8.2 Forecast Experiment

This experiment was chosen because it made use of the best available information in the log files, the timestamps field. The information provided provided by this field was consistent in quality and required no interpretation or assumptions to understand the content.
The experiment measured the impact of the training set length on the accuracy results produced by the 3 models and the capability to forecast values h points in the future.

In practice the impact of the training length depended on the forecast model and the dataset. For the 60 minutes data set a sufficient number of data points was necessary for the ARIMA model to forecast h points in the future, more than that only created minor improvements.
The mean model however performed better on both data sets with shorter training lengths. But in general the results for the ARIMA and naive models with a short forecast horizon where acceptable. Most MAPE scores where under 30% which can make them usable for a forecasting strategy.

The question is than which model to chose. For very short forecast horizons the naive model is sufficient.While being almost as accurate as the ARIMA model the implementation of it is the easiest of the three and it does not require much resources or knowledge for the successful execution of the model.
For longer forecast horizons the ARIMA model is the best candidate but even then the naive model is not a terrible alternative for some use cases as shown by the results of the experiment.

But the optimisations based on the forecasts are limited.
With the log files it was not possible to detect recurring issues or bottlenecks affecting the data infrastructure. This made it impossible to proof or measure how a proactive resource scaling approach could improve the accessibility or performance of the Euro-Argo data infrastructure.

CHAPTER 9

Conclusion

This study has shown that despite a limited amount of knowledge and information log files can provide characteristics and workload patterns if one asks the right questions.
It was possible to see patterns and characteristics in the Euro-Argo log files from multiple perspectives, the user, the traffic and the files on the FTP server. Very noticeable was how a select number of users or files could have a big impact on the overall observed workload reconstructed from the log files.

It was, however, less clear how these characteristics could be used for optimisation strategies. While insightful the knowledge usually led to more questions or a better understanding of the limitations of the log file.

The clearest pattern available, the hourly traffic to the FTP server over time, proved itself as a reliable source for optimisation possibilities. Using the historical traffic patterns we showed that it was possible to forecast the traffic to the FTP server.

The research question for this thesis was how the Euro-Argo data infrastructure could be optimized based on its operation history reflected by the content of the access log files.
The answer to this question is that the FTP log files are not sufficient to be used as the primary source of information. The log files need to provide both an accurate depiction of the historical workload and function as the source for optimisation strategies.

In reality the FTP log files struggled with both tasks independently. The only exception was with the the hourly traffic as shown by our experiment.
FTP log files are rather limited and can only provide a limited number of reliable metrics and observations. Unless combined with other logs such as firewall, HTTP server or file system logs, FTP log files mostly lead to more questions.

Bibliography

[1] elasticsearch. `https://www.elastic.co/`. Accessed: 2080-04-30.

[2] Apache. Apache spark python api. `https://pypi.org/project/pyspark/`, 2018. Accessed: 2018-06-05.

[3] Rodrigo N. Calheiros, Enayat Masoumi, Rajiv Ranjan, and Rajkumar Buyya. Workload prediction using arima model and its impact on cloud applications' qos. *IEEE Trans. Cloud Computing*, 3(4):449–458, 2015.

[4] Thierry Carval, Robert Keeley, Yasushi Takatsuki, Takashi Yoshida, Claudia Schmid, Roger Goldsmith, Annie Wong, Ann Thresher, Anh Tran, Stephen Loch, and Rebecca Mccreadie. *Argo user's manual V3.2*, sep 2017. doi:10.13155/29825.

[5] elastic. elasticsearch-hadoop. `https://github.com/elastic/elasticsearch-hadoop`, 2018. Accessed: 2018-06-06.

[6] elastic. elasticsearch-py. `https://github.com/elastic/elasticsearch-py`, 2018. Accessed: 2018-06-06.

[7] elastic. elasticsearchr. `https://cran.r-project.org/web/packages/elasticsearchr/`, 2018. Accessed: 2018-06-06.

[8] elastic. filebeat. `https://github.com/elastic/beats`, 2018. Accessed: 2018-06-06.

[9] elastic. kibana. `https://github.com/elastic/kibana`, 2018. Accessed: 2018-06-06.

[10] elastic. Logstash. `https://github.com/elastic/logstash`, 2018. Accessed: 2018-06-06.

[11] Network Working Group. How to use anonymous ftp. `https://tools.ietf.org/html/rfc1635`, 1994. Accessed: 2018-05-26.

[12] Kathleen Hodgkinson and Abdelmounaam Rezgui. Safal: A mapreduce spatio-temporal analyzer for unavco ftp logs. In *CSE*, pages 1083–1090. IEEE, 2013.

[13] R.J. Hyndman and G. Athanasopoulos. (2013) forecasting: principles and practice. otexts: Melbourne, australia. `http://otexts.org/fpp/`. (Accessed on 05/17/2018).

[14] Rob Hyndman, Christoph Bergmeir, Gabriel Caceres, Leanne Chhay, Mitchell O'Hara-Wild, Fotios Petropoulos, Slava Razbash, Earo Wang, and Farah Yasmeen. *forecast: Forecasting functions for time series and linear models*, 2018. R package version 8.3.

[15] Bernard J Jansen. The methodology of search log analysis. pages 100–123, 2009.

[16] I Koga and E Almeida. File transfer log analysis: A meteorological data 2 center case study. 7:13, 01 2016.

[17] mongodb. mongodb. `https://www.mongodb.com/`, 2018. Accessed: 2018-06-06.

[18] Adam Oliner, Archana Ganapathi, and Wei Xu. Advances and challenges in log analysis. *Queue*, 9(12):30:30–30:40, December 2011.

[19] ProFTPD. xferlog - proftpd server logfile. `http://www.castaglia.org/proftpd/doc/xferlog.html`, 2000. Accessed: 2018-05-26.

[20] Carlos Vazquez, Ram Krishnan, and Eugene John. Time series forecasting of cloud data center workloads for dynamic resource provisioning. *JoWUA*, 6(3):87–110, 2015.

[21] Ziliang Zong, Ribel Fares, Brian Romoser, and Joal Wood. Faststor: improving the performance of a large scale hybrid storage system via caching and prefetching. *Cluster Computing*, 17(2):593–604, 2014.

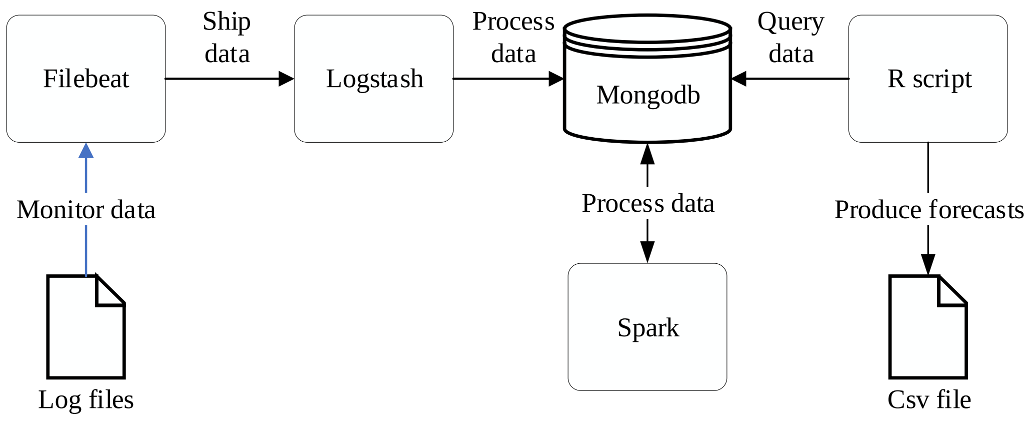
APPENDIX A

Appendix

A.1 Forecast toolchain

The proposed forecast toolchain in section 5.2 was basic in setup.
Currently the log files need to be processed by a spark job each time a new forecast model is trained.
If the forecast toolchain will see actual usage it is advised to store the processed results in a database. Then spark can query the database and only process the events that are new before forwarding the data to the forecast package.
For such an implementation it is possible to reuse the first 3 stages of the analysis toolchain.

Figure A.1: Workflow of the improved forecast toolchain.



yoyo $\overline{h}$ and $\overline{h}$ and h and h.

A.2 Forecast Performance

A.3 Forecast Performance old

first data set

Training length	**Forecast horizon (h)**	**Naive**	**Mean**	**ARIMA**
	1			
	2			
	3			
	6			
	12			
	24			
	$\overline{h}$			
	1			
	2			
	3			
	6			
	12			
	24			
	$\overline{h}$			
	1			
	2			
	3			
	6			
	12			
	24			
	$\overline{h}$			

Training length	Forecast horizon	ARIMA	Mean	Naive
	1	0.06448563	0.0002539299	0.003133817
	2		0.0002498992	0.003027005
	4		0.0002458686	0.00305522
24	6		0.0002317614	0.003047158
	8		0.0002680371	0.00312374
	12		0.0002599758	0.003170093
	24	0.06022879	0.0002438533	0.003075373

Training length	Forecast horizon	ARIMA	Mean	Naive
	1	0.07355225	0.0002612394	0.003106521
	2		0.0002713649	0.003199676
	4		0.0002490887	0.003153098
48	6		0.0003199676	0.003153098
	8		0.0002247874	0.003136898
	12		0.000236938	0.00309437
	24	0.07342042	0.0002531389	0.003019441

second data set

Training length	Forecast horizon	ARIMA	Mean	Naive
	1	0.06448563	0.0002287893	0.003286463
	2		0.0002430887	0.003303146
	4		0.0002192564	0.003360343
24	6		0.0002645377	0.003088656
	8		0.0002550048	0.003279314
	12		0.0002836034	0.003305529
	24	0.06022879	0.0002859867	0.003393708

Training length	Forecast horizon	ARIMA	Mean	Naive
	1	0.07355225	0.0002708533	0.003350911
	2		0.0002516779	0.003264621
	4		0.0002396932	0.003324545
48	6		0.0002396932	0.003326942
	8		0.0002780441	0.003355705
	12		0.000246884	0.003331735
	24	0.07342042	0.0002612656	0.003398849

Training length	Forecast horizon	ARIMA	Mean	Naive
	1	0.09390309	0.0002675988	0.003379942
	2		0.0002603664	0.003423337
	4		0.0002941176	0.003300386
72	6		0.000277242	0.00334378
	8		0.0002917068	0.003353423
	12		0.0002965284	0.003288332
	24	0.09502411	0.000277242	0.003305207

Training length	Forecast horizon	ARIMA	Mean	Naive
	1	0.1400432	0.0002797558	0.0002287893
	2		0.0002263479	0.0002430887
	4		0.0002644964	0.0002192564
288	6		0.0002822991	0.0002645377
	8		0.0002721261	0.0002550048
	12		0.0002568667	0.0002836034
	24	0.1403357	0.000249237	0.0002859867