Graz University of Technology

Master Thesis

Cloud Storage Performance Analysis

A Global Benchmark

by
Johannes Simon Innerbichler, 0931506

Supervisor:
Franz Pernkopf

Assessor:
Franz Pernkopf

Graz, April 5, 2016

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

date	(signature)

Eidesstattliche Erklärung[1]

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am	(Unterschrift)

[1] Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Abstract

Public cloud storage services are nowadays a data intensive domain and already producing a dominant share of the Internet traffic world wide. Huge files are synchronized between clients and different data centers of each storage provider. Little is known about the performance of each individual service. Execution times of upload and download operations may vary over time. According to current research the performance depends on daytime and other factors, e.g. distances to the nearest data center.

In this thesis, a global distributed cluster of servers perform periodic interactions with three of the most used public cloud storage provider, i.e. Dropbox, Google Drive, and OneDrive. Collected data is used to obtain insights in the behavior of these services. Techniques coming from machine learning and statistical analysis are applied in order to infer knowledge about the geographic performance over the daytime for each single provider. Different algorithms are compared with each other in order to find the optimal regression model. Performances of each storage is evaluated in order to find the best service in a given situation.

Key words: cloud storage, global regression, geographical modeling, dropbox, google drive, onedrive

Acknowledgements

Firstly, I would thank my advisor, Franz Pernkopf, for his great work in advising me. During several phases of this thesis I received an outstanding help. Every question and problem was discussed in a very patient and competent way in order to find an appropriate solution.

Without my girlfriend Katharina this thesis would not have been possible. The encouragement I received from her made me solve the hardest problems.

Last, but not least, I want to say thank you to my parents, who supported me mentally during my studies and this thesis.

Johannes Simon Innerbichler

Contents

List of Figures

List of Tables

1 Introduction and Motivation

Consumer cloud storage is getting more popular over recent years. It enables private people and businesses to store data in the public cloud in order to back it up and make it available from everywhere at any time. Since personal cloud storage services synchronize a local folder into the cloud it is a data-intensive domain and is already producing a significant share of the Internet traffic nowadays. According to Cisco [2] 1136 million users of consumer cloud storage produced 14 exabytes of traffic in the year 2014. They forecast traffic of 39 exabytes in 2019. Large data centers state the physical environment for storing this extensive amount of data. Providers of cloud storage are responsible for keeping data available and have to make sure it stays accessible 24 hours a day. This performance can only be achieved by replicating data on several globally distributed data centers. S. Rhea et al. [3] defines cloud storage as a pool of distributed resources acting as one, which is highly fault-tolerant by using redundancy and distributing data, and are highly persistent through the creation of versioned copies.

The number of cloud storage provider is still increasing tremendously. By the end of 2015 more than 70 different storage solutions were available. One may ask what differentiates public cloud storage provider from each other, apart from the pricing and provided features, e.g. sharing functionality or collaboration features. Every single service has different distributed data centers and distinct algorithms running in the background. Several research topics analyzed different providers regarding performance issues. Idilio Drago et al. [4] benchmarked various cloud storage services with respect to different file types and evaluated features, such as delta encoding and file compression before uploading. They used a fixed global location of their test framework and assumed that performance is independent of external parameters, such as time of the day, global location, and date of benchmark. They are already mentioning that their results may vary at different locations, but proposed it as future work. Additional research deals with the performance using different sets of file types [5]. Sets may be composed of files which are easy to compress (sparse) and files where compression does not affect the overall file size. In this case execution times for up- and downloads varied for different set of files and providers. In [6], Dropbox, the most common cloud storage provider, is tested. Their in depth evaluation focused on used protocols and the impact of different data sets. They introduced the Round Trip Time (RTT), which is influenced by the distance of the storage server. Therefore, reached bandwidths depend on the geographical location of the client. A first global benchmark was executed by [7] on Amazon EC3, which is an online storage provider. It was selected, because it is used by the most common cloud storage provider, i.e. Dropbox. Several operations were performed at different geographical regions and the results compared. According to their results the effective bandwidth depends on the location of the client.

In this project mentioned benchmarks are extended to public cloud storage providers, i.e. performance of three of the most common providers (Dropbox, Google Drive, and OneDrive) are evaluated with respect to the geographical location of the client and other parameters, such as different sets of files, time of the day, day of the week, and month. The following sets of files were chosen: one 1MB file, one 10MB file, one 100MB file, one 100MB sparse file. Each file set is uploaded and downloaded repeatedly to a certain provider and the performance is measured. Geographically distributed measurement servers were setup, which are performing file operations on a regular basis. Collected data is stored in a central database, evaluated using statistical techniques (i.e. appropriate outlier detection) and different regression techniques are applied. In the first phase the influence of external parameters (i.e. month, weekday, and time of the day) is determined at two locations (London and Singapore). The second phase extends inferred knowledge to a global scale. Applied regression models are multiple linear regression, fitting of Gaussian shaped functions, and support vector machine regression (SVR). At last, a k-nearest-neighbors classification approach was evaluated.

Knowledge about statistical methods was mainly gathered from [8] and their corresponding online course provided by Stanford Online. They used the statistical language R in order to perform multiple linear regression and evaluation of the fitted function. Due to existing knowledge of Matlab, it was chosen to be the environment for data analysis in this project. Bishop [1] provided material for fitting Gaussian models and applying Support Vector Machine Regression. It is an excellent book about fundamentals in machine learning.

1.1 Motivation

This thesis was executed in partnership with CrossCloud GmbH. Their product is called CrossCloud, which is an environment for multiple Cloud Storage Services like Dropbox or Google Drive that lets users easily use multiple cloud storage accounts and collaborate with others across different cloud storage services. The environment consists of an application for desktop operating systems (Windows, Linux, and MacOS), a mobile application (iOS, Android) and a web application. The user installs the client applications on all devices. It can aggregate all available cloud storage accounts in a way where CrossCloud is the only application needed by the user.

Many users of cloud storage have multiple accounts at different cloud storage providers, but are not able to use all space available. In addition, many users have to use multiple services in order to collaborate with different groups of people over different services, but are not able to do this efficiently. This is caused by the fact that the usage of multiple services implies the installation of different client applications on all devices as well as the adaptation to various ways of how cloud storage works. When paying attention to security, it is further complicated to enforce a security policy for files stored in the cloud across different services since this is dependent on the support of security features of the different providers. All these facts make it hard for users to use different services and therefore make them dependent of certain providers they are familiar with.

CrossCloud is now an environment, designed to support multiple cloud storage services. It's components can connect to different cloud storage providers in the background and use their public API to sync data across services. Users add all their accounts to CrossCloud (i.e. allow CrossCloud to connect to their accounts). At the example of the desktop application, the user only interacts with the filesystem (a dedicated CrossCloud folder). CrossCloud detects if a synchronization is necessary and syncs the data to different services. By default, CrossCloud distributes the user's files to different services so that the aggregated space of all accounts can be used. Finding the optimal storage for each individual file can be improved by gaining insights in data usage and behavioral knowledge of each storage with respect to performance. Figure 1.1 depicts an overview of the CrossCloud's architecture. Users can specify synchronization

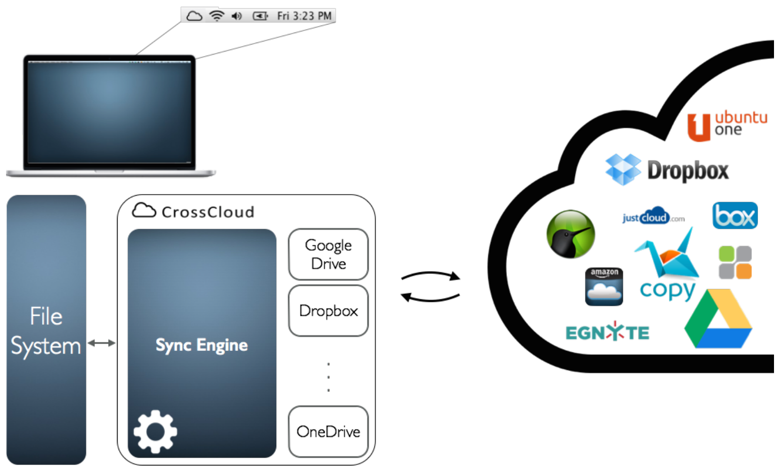


Figure 1.1: The basic structure of CrossCloud's client application.

rules (e.g. sync all contents of a specific folder to Dropbox) to customize the behavior of the synchronization or tell CrossCloud to keep a copy of the data on all accounts. CrossCloud follows a smart management approach, which means that the user does not necessarily have to know where his or her data is stored. It decides on its own where to place the files and gives users access to the data in any situation (desktop application, mobile application and web application). Users can easily share files across services by simply telling CrossCloud to sync and share files or folders on a specific service. The user always interacts with the file system (or app) on the target platform which ensures high usability. Further, CrossCloud allows users to apply client-side encryption to the data independent from where the data is stored and if the specific provider supports encryption.

Goal of the project is do perform a global benchmark of the three most used cloud storage provider i.e. Google Drive, Dropbox and OneDrive. Learned insights are used to determine the optimal provider for a given situation (i.e. file size, file type, time of day, etc.). A prototype implementation of the measurement is set up in order to show that performance depends on the mentioned external parameters. The setup has to be easily extendable by additional benchmarks, cloud service providers and parameters. Inferred knowledge can later be used in CrossCloud for a improved decision making algorithm.

1.2 Outline

Chapter 2 deals with acquisition of performance related data. After showing data center locations of each provider, the globally distributed setup of measurement servers is introduced. Insights in the used architecture are provided and details about the used technologies are reviewed. The succeeding Chapter 3 deals with modeling execution times at fixed locations (i.e. London and Singapore). Influences of the external factors, e.g. daytime, day of the week, and month are evaluated. The first part introduces statistical methods used for evaluation followed by utilization of this methods in order to determine an optimal predictive model. Chapter 4 introduces different approaches for modeling patterns for global performance. Firstly information about data preprocessing is provided, succeeded by theoretical explanation of the used regression algorithms. Section 4.2.1 introduces the simplest applied regression, i.e. Multiple Linear Regression. Fitting of 2-dimensional Gaussian functions with fixed centers is explained in Section 4.2.2. Applying Support Vector Regression to collected data is discussed in the subsequent Section 4.2.3. The last technique used k-nearest-neighbor (KNN) methods in order to select optimal services given certain parameters. Modeling performance and gained insights are the topic of the last section in the given chapter. The last Chapter presents future work and how this project will prospectively continued.

2 Data Acquisition

In order to perform a global benchmark, globally distributed servers were rented, from which each cloud storage is benchmarked from. Therefore, it was vital to choose the location of the server manually. Single servers interact with every cloud storage provider on a regular basis using different sets of benchmark files. Benchmarks differ from each other by the file type and size. Files for benchmark execution are listed in Table 2.1. Every benchmark is performing an upload operation using the corresponding files, preceded by a download. Time spans for each operation are measured and stored in a database.

Benchmark Type	**Description**
ONE_LARGE_SIZE_FILE	single 100 MB dense text file
ONE_MEDIUM_SIZE_FILE	single 10 MB dense text file
ONE_SMALL_SIZE_FILE	single 1 MB dense text file
ONE_SPARSE_FILE	single 100 MB sparse text file

Table 2.1: Different file sets used for benchmarks.

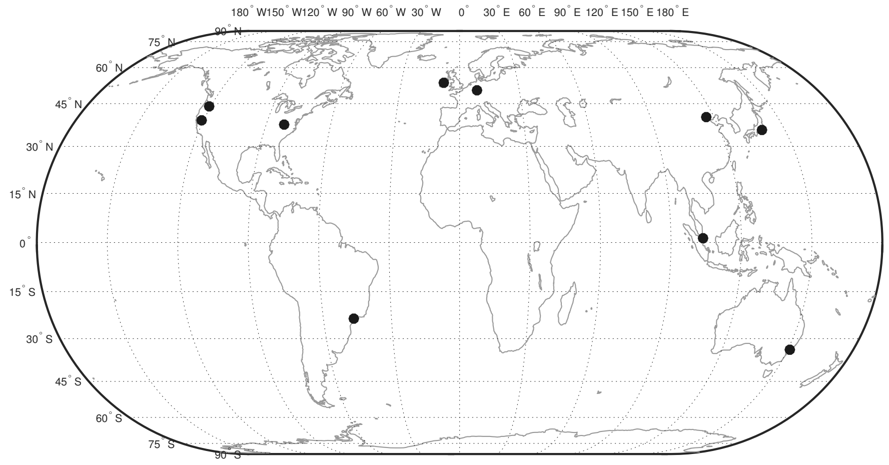
Random text files are generated densely, i.e. file compression techniques do not affect the total file size. In the case of sparse files compression has an impact on the size of the file. If execution times for sparse and dense files differ, it can be assumed, that files are compressed before being uploaded or downloaded.

2.1 Data Center Locations

In order to store vast amounts of data in the most reliable way, multiple data centers of each service provider are distributed over the globe. Knowledge about regions of used data centers is publicly available in the Internet. It is assumed that client locations near to data centers provide a better performance. In general, each service provider decides to which data center the client application connects. This decision is mainly made by additional servers of which the exact location is unknown. It is assumed that this part of execution time can be neglected for larger files and affects benchmarks with smaller files exclusively. An in depth evaluation of generated overhead is done for Dropbox in [6].

2.1.1 Dropbox

Dropbox uses Amazon EC3 data centers for their services. Public information is available online at `http://aws.amazon.com/about-aws/global-infrastructure/`. Figure 2.1 shows a geographical map of stated data centers. It can be seen that there is a focus on North America, Europa, and East Asia. A single server is located in South America and Australia. No data centers are located in Africa.

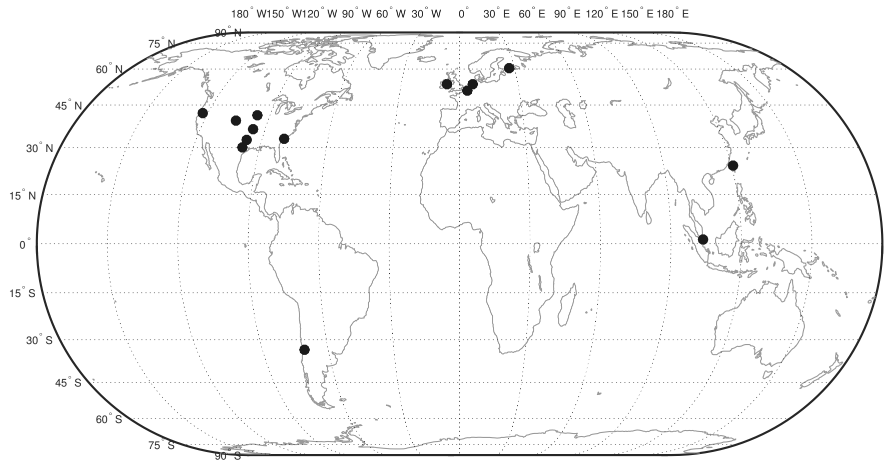


180° W 150° W 120° W 90° W 60° W 30° W 0° 30° E 60° E 90° E 120° E 150° E 180° E
90° N
75° N
60° N
45° N
30° N
15° N
0°
15° S
30° S
45° S
60° S
75° S
90° S

Figure 2.1: Data center locations of Amazon EC3 used by Dropbox

2.1.2 Google Drive

Google invests over two billion US Dollar a year for expanding and improving their infrastructure. Information about Googles data center locations can be found at `http://www.google.com/about/data centers/inside/locations/`. They are shown in Figure 2.2. In general Google provides a large number of data centers in North America and Europe. It focuses on the same regions as Dropbox, but excludes Australia. It can be assumed that a slower bandwidth is achieved in Australia. Again no data center was placed in Africa.



180° W 150° W 120° W 90° W 60° W 30° W 0° 30° E 60° E 90° E 120° E 150° E 180° E
90° N
75° N
60° N
45° N
30° N
15° N
0°
15° S
30° S
45° S
60° S
75° S
90° S

Figure 2.2: Data center locations of Google used by Google Drive.

2.1.3 OneDrive

Little is known about Microsoft's data centers. According to [9] over 100 data centers are placed in a global network containing more than one million servers. A selected few locations are mentioned in the fact sheet. They are shown in Figure 2.3. Again they focus on USA and Europe. Microsoft dedicated a data center to South America, Australia, and Asia (China) each. Up until now over 15 billion US Dollar have been invested in the infrastructure in order to deliver a reliable and scalable service.

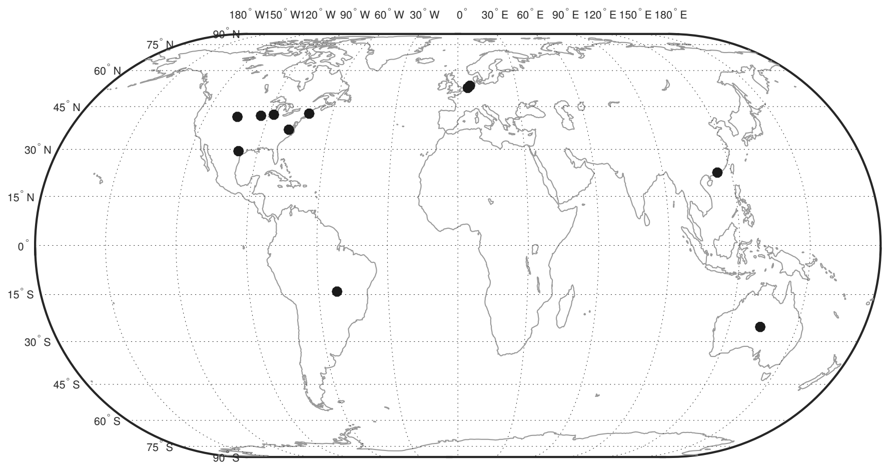

180° W150° W120° W 90° W 60° W 30° W 0° 30° E 60° E 90° E 120° E 150° E 180° E
90° N
75° N
60° N
45° N
30° N
15° N
0°
15° S
30° S
45° S
60° S
75° S
90° S

Figure 2.3: Data center locations of Microsoft used by OneDrive.

2.2 Globally Distributed Measurement Servers

In order to perform a global benchmark, files have to be exchanged with the three mentioned cloud storages (i.e. Dropbox, Google Drive, OneDrive) on different geographical locations. Globally distributed computation power with an appropriate bandwidth is necessary in order to perform this benchmark. Platform as a service (short PaaS) provider serve the framework for running such applications in the Internet or Cloud, respectively. A PaaS provider hosts hardware and software on its own infrastructure. Unfortunately this model does not focus on the location of specific hardware where deployed applications are executed. The second potential model is IaaS, i.e. Infrastructure as a service. It is a form of cloud computing that provides visualized computing resources in the Internet. Third-party providers are hosting hardware, software, and other infrastructure. Certain vendors offer a selection of server locations, at which execution power is provided. Visualized hardware can be rented on fixed locations in form of virtual machines (short VM). Other important properties of hosted machines are pricing, monthly traffic limit, and appropriate bandwidth. Potential IaaS provider were Vultr [10], Digital Ocean [11], and Amazon Web Services (short AWS) [12].

In case of Vultr computation power in the cloud is provided at dedicated locations (e.g. Frankfurt, Amsterdam, Paris, London, Tokyo, New York, Georgia, Miami, Florida, Sydney, Australia, etc.). A virtual machine with 1000GB monthly traffic costs 10€ a month. Their bandwidth of 480MB/sec. is appropriate for the given context. Digital Ocean is providing virtual machines hosted on fast solid state drives. Users can choose servers in New York, Amsterdam, San Francisco, Singapore, Frankfurt, and Toronto. All servers come with 1GB/sec. network interfaces. Pricing start with 1TB of bandwidth per month. The cheapest plan fulfills all requirements and starts at 5€ per month. Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable computation capacity in the cloud. Servers can be hosted in North America, Brazil, Europe, and Asia Pacific. For 4€ a month users can acquire traffic of 10GB. No information about connection speed can be found. Other hosting services were either too expensive or are indeterminate about the exact server location. The mentioned three providers constitute the necessary hardware for executing the proposed analysis.

Due to the data intensive domain a lot of reoccurring traffic is produced. After the first month of execution Amazon complained about potential distributed denial-of-service attacks originating at the hosted virtual machines. This lead to blockage of this service, which made it unusable for further benchmark execution.

2.2.1 Cluster of Virtual Machines

Digital Ocean and Vultr were used for hosting virtual machines. The number of total hosted servers is 19. Table 2.2 lists all measurement server. Nine of them are placed in the USA, seven in Central Europe, and two in Asia. A single machine is hosted in Australia. A central database was setup in Amsterdam with endpoints on a dedicated host. A detailed insight about the central database server architecture can be found in Section 2.3.

Server Name	Location	Host	Server Name	Location	Host
VMamsterdam	Amsterdam	Digital Ocean	VMfrankfurtvultr	Frankfurt	Vultr
VMsanfrancisco	San Francisco	Digital Ocean	VMjapanvultr	Tokyo	Vultr
VMlondon	London	Digital Ocean	VMatlantavultr	Atlanta	Vultr
VMnewyork	New York	Digital Ocean	VMlondonvultr	London	Vultr
VMsiliconvalley	Silicon Valley	Digital Ocean	VMlosangelesvultr	Los Angeles	Vultr
VMfrankfurt	Frankfurt	Digital Ocean	VMseattlevultr	Seattle	Vultr
VMsingapore	Singapore	Digital Ocean	VMmiamivultr	Miami	Vultr
VMdallasvultr	Dallas	Vultr	VMchicagovultr	Chicago	Vultr
VMsydneyvultr	Sydney	Vultr	VMnewyorkvultr	New York	Vultr
VMamsterdamvultr	Amsterdam	Vultr	VMparisvultr	Paris	Vultr
VMnewyorkvultr	New York	Vultr			

Table 2.2: Hosted servers and their geographical location.

Figure 2.4 depicts global locations of all measurement servers, which are used for executing benchmark tasks. A higher number of machines are available in North America and Europe. This results in a higher granularity of measurements at that regions. Cloud storage provider are focusing on that regions as well, which redounds to the benefit of this analysis. Due to the limitations of accessible virtual machines in South

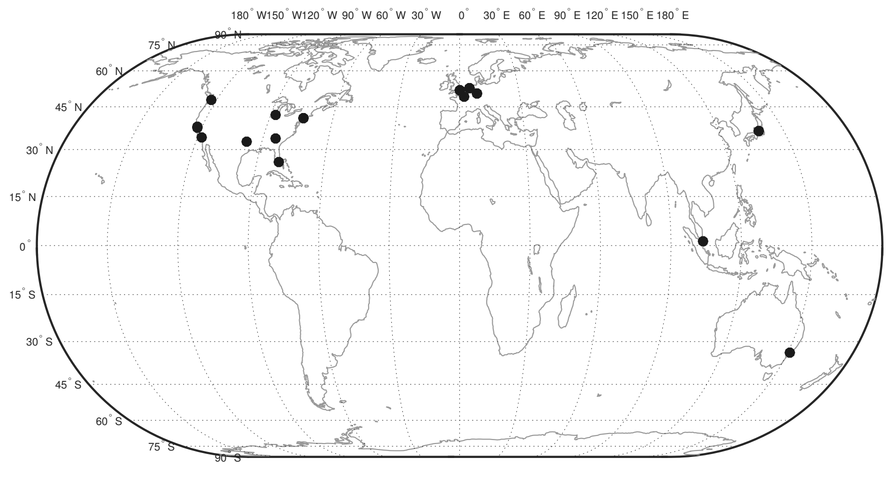

180° W150° W120° W 90° W 60° W 30° W 0° 30° E 60° E 90° E 120° E 150° E 180° E
90° N
75° N
60° N
45° N
30° N
15° N
0°
15° S
30° S
45° S
60° S
75° S
90° S

Figure 2.4: Geographical overview of hosted servers used for measurements.

America, Africa, and Central Asia little insight can be gained in these areas.

2.2.2 Applied Technologies

When maintaining this number of servers, administrative tasks have to be done in an automatized manner. Several technologies exist for application deployment and systems administration on a broader scale. In general Linux was chosen as host operating system for executing web applications. This decision was made, because it is a de facto standard for web servers. This leads to the fact that a lot of free and open source tools are available for this domain. Due to existing knowledge of Ubuntu, it was selected as Linux derivative. Deployment of the application and rolling out software updates was done using Docker [13]. Docker is an open source project initiated by Solomon Hykes. Deployment of applications can easily be done inside software containers, which provide an additional layer of abstraction and isolation to the host operation system. Once Docker is installed on all servers, it allows containers to be independently run within a single Linux instance, avoiding the effort of starting and maintaining additional virtual machines. In general Docker differs between container and software images. A container is a stripped-to-basic representation of a Linux operating system. Docker images are loaded and executed inside a container, without affecting other containers or the host system. Every new version of the benchmarking application is wrapped inside a Docker image and deployed to containers on all rented machines. In case of Digital Ocean, virtual machines with pre-installed versions of Docker can be set up. Vultr does not offer this feature. Therefore, Docker has to be installed on every instance after initialization of the virtual machine manually. Python in combination with Fabric [14] was used for executing administrational tasks on multiple servers. Fabric is a Python library for executing system administration tasks using the SSH network protocol. A set of basic tools is provided in order to execute local or remote shell commands. In this project fabric was used to install Docker on virtual machines, administrate operating systems, and to deploy new versions of the benchmarking software.

2.2.3 Distribution of Benchmarking Software

In order to perform benchmarks on each location, benchmarking software has to be deployed and executed on all virtual machines. As already mentioned, Docker was used to realize this task. A Docker image with proper system settings and installed packages were created. In general, new images are created by deriving from an existing image. For storing newly created images, a public repository called DockerHub is officially provided, where images can be publicly stored and retrieved. Figure 2.5 depicts a rough overview of the deployment process. First an image with the identifier `jinnerbichler/cloudmapbase` was created. This image is based on the official Ubuntu 14.04 image. It represents the base image for further images. Since software was developed in Java (further details can be found in Section 2.3.1), the Java Virtual Machine (JVM) was used as platform for executing benchmark execution. Therefore, Java runtimes have to be installed in all executing Docker containers. After creating an appropriate directory structure for the project and setting the timezone to UTC, this image was uploaded to Docker Hub. All this setup had only been done once and was therefore executed manually. Up to now no executable benchmark software is present on the base image. A separate image with the identifier `jinnerbichler/cloudmap` was created containing the executable software. This image is based on `jinnerbichler/cloudmapbase`. For each software update a new version of the Docker image is created and updated on Docker Hub. Since this process is repeated for every new update of the benchmarking software is needed to be done in an automized manner. Docker can build images automatically by reading instructions from a Dockerfile. A Dockerfile is a text document that contains all commands needed for assembling a new image. In this case a simple Dockerfile is sufficient for creating a proper image. First the executable file is copied in the previously created directory structure. Then software settings are added to the image in form of an SQLite database. Further information about this database can be found in Section 2.3.1 The developed

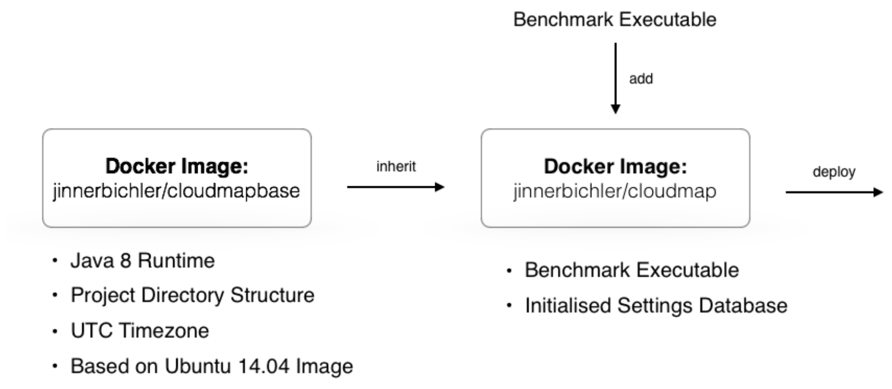


Figure 2.5: Procedure for deploying new benchmarking software using Docker images.

Docker file can be found in the Appendix (see Section 6.1).

After updating the image `jinnerbichler/cloudmap` on the public repository, each single virtual machine has to be informed about a new version of the image. As already mentioned this procedure was automized using the Fabric library. Using Fabric shell commands can be executed on all hosted instances. First the executable is build locally, a new version of the Docker image created and deployed to Docker Hub. Finally the image was updated on the virtual machines and the existing container updated. Implemented Python source code can be found in the Appendix (see Section 6.2).

2.3 Measurement Architecture

In order to perform benchmarks on a global scale, the entire architecture must be designed to be easily extendible by additional measurement servers. Therefore, each measurement server (from now on called client) has to hold as little information as possible. This makes it possible to set up new clients with little effort. The entire architecture is controlled by a centralized server (from now on called main server). Files used for benchmarking are administrated by this server. Every result of a benchmark is reported to the main server, which stores the data persistently in a central database. Figure 2.6 depicts an overview of the applied architecture, where N clients are bidirectional communicating with the main server.

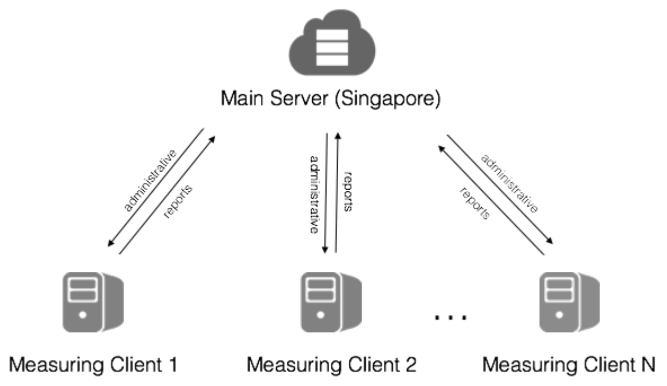


Figure 2.6: Applied client/server architecture implemented for benchmark execution.

Each newly added client requests necessary data used for initialization from the main server, i.e. files used for benchmarking (see Table 2.1) or information used for authenticating to each cloud service. Each client has a static and dedicated Internet

Protocol address (from now on IP address). This static IP address can directly be linked to a geographical location. It is added as metadata to each reported benchmark. The main server uses this information, assigns the report to a single client and geographical location respectively.

All three cloud storage provider support the OAuth 2.0 authentication standard [15] with varying optional feature implementations. This authentication mechanisms exchanges token sets (i.e. access token and refresh token) with the authenticating party, which may be valid a limited time. The access token is used to obtain authorized access to certain resources, which is access to storage of a certain service provider. After expiration of the access token it can be updated using a refresh token. The main idea behind OAuth is to decouple the authorization server and the resource server, in order to increase the level of security. Therefore, no user credentials need to be stored in order to get long term access to protected resources. The documentation states that it is possible to obtain a static access token, which is not time-limited. This means that no refresh is necessary to gain long term access, i.e. no refresh token is exchanged. This adaption of the standard is heavily used by Dropbox. After verifying the user credentials, a single access token is provided, which is valid for an unlimited time. In case of Google Drive and OneDrive the access token is valid for 3600 seconds and needs to be refreshed using a static refresh token. In order to set up a new client, the token sets have to be initialized and store persistently on the client's storage.

2.3.1 Applied Technologies

The main server is implemented to be a public representational state transfer service (REST), which can easily be accessed over the Internet using standard HTTP-based requests. This application service was implemented using the Spring Framework Version 3.1 [16]. It is a web-application framework for the Java platform licensed under Apache License 2.0. A domain specific RESTful API was implemented in order to provide functionality for reporting benchmarking and administrative tasks. Many integrations of persistent database technologies are available for this framework. Apache Tomcat was used as application web server, which provides a "pure Java" HTTP web server environment for Java code to run.

In this project the MongoDB database technology [17] Version 3.0 was used for storing reported results persistently. MongoDB is a cross-platform document-oriented database and classified as a NoSQL database, meaning that no static schema has to be setup in order to store data. Each entry is stored in form of an JSON-like document with dynamic schemas. This dynamic behavior makes adaption and extension of the database easy, which is extremely helpful during the conceptual phase. MongoDB is licensed under GNU AGPL v3.0.

To use the same language and technology on both, the main server and on each client, the Java Virtual Machine (JVM) was the basis for client-side development. All three benchmarked services provide SDKs for this platform, which include OAuth authentication and data-exchange functionality. As already mentioned, authentication data has to be stored persistently on each client. In order to solve this task, SQLite [18] was used. SQLite is a relational database management system and licensed under Public Domain. In contrast to other databases, SQLite is not a client-server database engine. Data can be directly stored and accessed over a local file.

FreeGeoIp [19] was used to link IP addresses to geographical locations. It provides location information inform of latitudes and longitudes. In general FreeGeoIp provides a public HTTP API with limited queries per hours. Additional requests can be made by hosting it on a dedicated server. In this case the location database is periodically updated in order to guarantee most recent information. Therefore, this service was installed and hosted on the main server.

2.3.2 Client Architecture

As already mentioned the client application was implemented in Java and runs in the JVM. Therefore, object oriented paradigms can be applied. Figure 2.7 shows a rough overview of the software architecture. In general, it can be said that each module is represented by its own class. The core of the application is the `Benchmark Engine`, which is in charge of periodic execution of benchmarks and reporting the results to the main server. Each execution is scheduled to take place n minutes after the previous run, whereas $P(X = n) = \mathcal{U}(0, 30)$. This was necessary to prevent periodic executions on same day times.

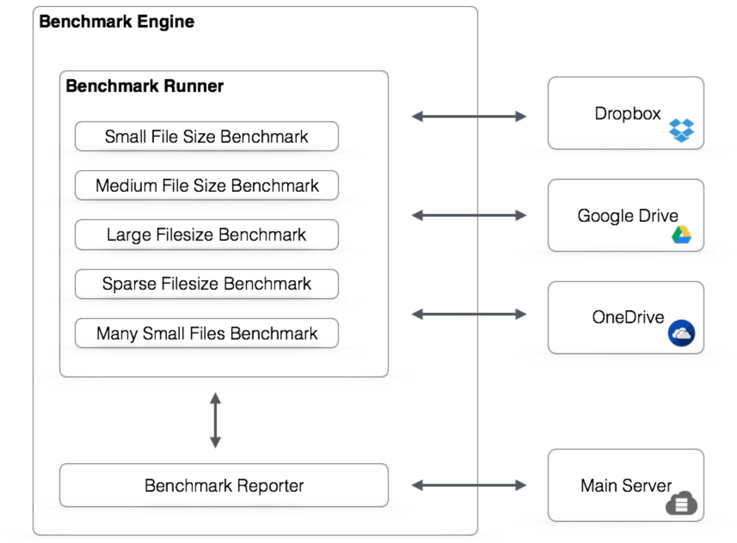


Figure 2.7: Architecture of benchmarking client.

Execution of benchmark is done by the `Benchmark Runner`. For each of the four set of files a dedicated benchmark was implemented, i.e. `Small File Size Benchmark`, `Medium File Size Benchmark`, `Large File Size Benchmark`, and `Sparse File Size Benchmark`. The `Benchmark Runner` executes each single benchmark sequentially on each cloud service. Afterwards results are reported to the main server. A more detailed diagram of implemented classes can be found in the Appendix (see Figure 6.1).

2.3.3 Main Server Architecture

Figure 6 gives an overview of the implemented structure on the main server. At Spring Framework modules called *Controller*, which provides a structured entry for incoming HTTP requests. The two most important controller are the `Benchmark Result Controller` and `Benchmark File Controller`.

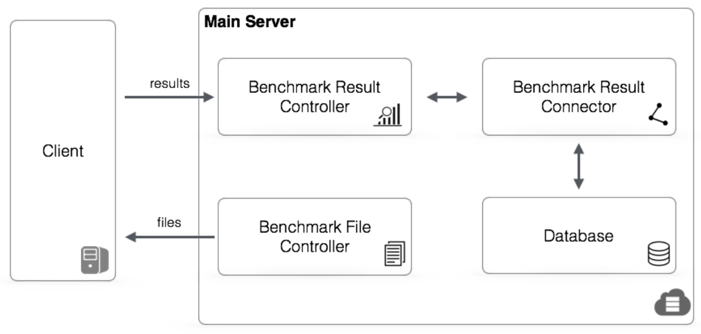

Main Server
Client
results
files
Benchmark Result Controller
Benchmark Result Connector
Benchmark File Controller
Database

Figure 2.8: Architecture of central main server.

The `Benchmark File Controller` allows each client to download necessary files. This is mainly during the initialization of each individual client. Results of executed benchmarks are reported to the `Benchmark Result Controller`, which processes the result and passes it further to the `Benchmark Result Connector`. This module implements a connection to the database and stores the data persistently.

2.3.4 Collected Data Sets

The overall amount of measured data points for each measurement server can be found in Table 2.3. Due to the fact that `VMlondon` and `VMsingapore` were used for evaluating local dependencies, more samples are available in London and Singapore.

Server Name	Location	# Samples	Server Name	Location	# Samples
VMamsterdam	Amsterdam	175,792	VMfrankfurtvultr	Frankfurt	118,784
VMsanfrancisco	San Francisco	205,472	VMjapanvultr	Tokyo	85,360
VMlondon	London	1,463,523	VMatlantavultr	Atlanta	170,432
VMnewyork	New York	190,944	VMlondonvultr	London	90,240
VMsiliconvalley	Silicon Valley	136,752	VMlosangelesvultr	Los Angeles	140,528
VMfrankfurt	Frankfurt	170,624	VMseattlevultr	Seattle	142,912
VMsingapore	Singapore	1,378,084	VMmiamivultr	Miami	188,288
VMdallasvultr	Dallas	164,768	VMchicagovultr	Chicago	94,144
VMsydneyvultr	Sydney	138,896	VMnewyorkvultr	New York	112,960
VMamsterdamvultr	Amsterdam	77,648	VMparisvultr	Paris	133,840
VMnewyorkvultr	New York	112,960			

Table 2.3: Hosted servers and their amount of measured data points.

For modelling local dependencies 1,664,712 data samples were used. At the end of the benchmark the global analysis was executed with 4,552,256 data samples.

3 Local Modeling Approaches

As already mentioned in the first phase, execution times are modeled at two locations in order to determine the influence of external parameters omitting the geographic location at this time. Measures from London and Singapore were used for this approach. The central main server was placed in Amsterdam. Modelling is done by excluding outliers in the first step, which may influence regression negatively. The last section deals with multiple linear regression and statistical methods coming with this kind of regression. It provides details about this approach, which allows to weight the influence of each external parameter. If execution times are independent of certain predictors, they will be neglected in the global model. Due to simplicity reasons, solely Dropbox and Google Drive are compared with each other. OneDrive will be added later when modeling global performance.

3.1 Data Preprocessing

Due to some network flaws and other factors (e.g. measurement errors) certain measured execution times may behave abnormally. In some cases large files are be dissembled into smaller chunks (e.g. 1kB chunks). Each chunk is uploaded/downloaded separately, whereas the client has to handle error cases. Due to the fact that HTTP requests are interchanged, timeouts may occur and have to be handled. These timeouts are occurring randomly and are not part of the normal behavior. This leads to observations that are distant to other data points. In statistics those data points are called outliers. They do not represent normal behavior. Outliers need to be detected and excluded from the obtained data set. In Figure 3.1 a file, with a size of 100MB, was uploaded on multiple daytimes to Google Drive. Looking at the collected execution times it can be seen that prominent outlier occurred. Due to the fact that measured execution times are outstanding by taking up to 10 times longer than the average, they can be easily detected using the percentile measure. Percentiles are mainly used in statistics and are indicating the value below a given percentage of observations in a group of observations fall. For example, the 25 percentile of a data set is the value, where 25 percent of the data can be found below this value. In this project outliers are detected by calculating the 99 percentile of each set of execution times and classifying values above this percentile as outliers. Figure 3.1 depicts detected outliers using the percentile measure.

This approach works well if outliers exists and are very distinct from the rest of the data. If a few outliers are close to the main data points (i.e. data points not classified as outliers) wrongly detected outlier can be observed. Due to the fact that outliers in the obtained data sets fulfil the first assumption, it was chosen as mechanism for outlier

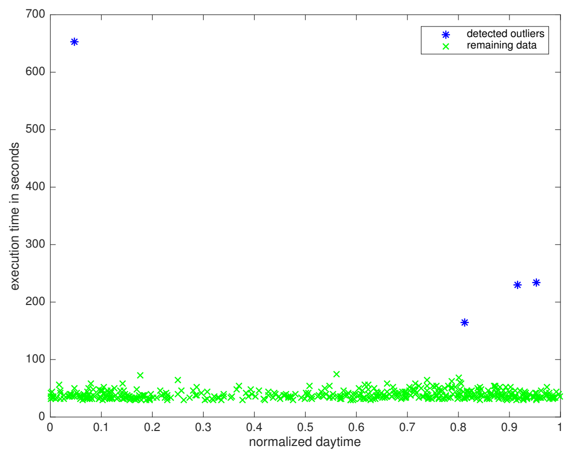


Figure 3.1: Detected outliers (blue) and remaining data (green) using a 99 percentile measure. Applied to collected data for uploaded files with size of 100MB to Google Drive.

detection.

3.2 Modeling Approaches

Simple linear regression deals in the simplest case with a singe scalar predictor variable x and a single scalar response variable (e.g $y = bx^2 + \epsilon$). By extending to multiple predictor variables in form of a vector $\mathbf{x}$ this becomes multiple linear regression (e.g. $y = \beta\mathbf{x} + \epsilon$), which is also known as multivariate linear regression. In the first part of this section it is shown how linear regression can be used to determine the influence of external parameter, i.e. daytime, weekday, and month. The second part deals with the selection of parameters for further global modeling.

3.2.1 Influence of External Parameters

In the first phase multiple linear regression is mainly used to answer tree questions:

1. Is there a relationship between the reached bandwidths and factors, such as time of the day, weekday, and current month? How strong is the relationship mentioned above?
2. What is the quality of the fitted model?
3. What is the influence of a single factor in the model?

In order to determine a dependency between reached bandwidths and parameters, e.g. time of the day, weekday, and month, multiple linear regression is used. It is a very simple but powerful approach of supervised learning. In general linear regression is a statistical technique used for modeling a linear relationship between independent variables $\mathbf{x} = [x_1, \ldots, x_p]$ and the dependent variable y. If simple linear regression is used only a single dependent variable is present. The relationship between the response y and the input variables $\mathbf{x}$ is modeled linearly, i.e. $y = \beta_0 + \beta_1 x + \epsilon$, whereas ϵ states the error term. Linear regression finds the optimal value of β_0 and β_1, which leads to a minimal mean squared error on training data.

In case of multiple linear regression p predictors are used to model y, whereas each predictor gets a separate weight term. This results in the following model, i.e

$$\hat{y} = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p. \tag{3.1}$$

Here $\hat{y}$ states an estimation for the real value of y given $x_1, x_2, \ldots, and\ x_p$. Therefore, multiple coefficients β_j have to be learned in order to minimize the sum of squared residuals of n training data samples

$$RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \tag{3.2}$$

Target values of the training data are represented in y_i, i.e. measured upload/download execution times. Predicted values in the model are stated as $\hat{y}_i$. Estimated values for the coefficients β_0, β_1, ..., β_p are computed using the statistical software package provided by Matlab and the method `fitlm`. In the following statistical techniques are briefly introduced to answer the mentioned questions above.

Is there a relationship between predictors and response? This question is answered by performing hypothesis tests on the computed model. Here we test the *null hypothesis* of

$$H_0\text{: } \beta_1 = \beta_2 = ... = \beta_p = 0 \tag{3.3}$$

against the *alternative hypothesis*

$$H_a\text{: at least one coefficient } \beta_j \text{ is non-zero.} \tag{3.4}$$

This test takes the absolute value of the coefficients into account. Due to the fact that valid coefficients might be close to zero this test is properly done by calculating the F-statistic,

$$F = \frac{(TSS - RSS)/p}{RSS/(n-p-1)}, \tag{3.5}$$

whereas $TSS = \sum(y_i - \overline{y})^2$ and $\overline{y}$ is the mean of all target values y in the training data. TSS measures the total variance in y. If there is no relationship F takes values close to 1. Otherwise, we expect F to be greater that 1, if a strong relationship exists.

What is the accuracy of the model? The question after choosing the alternative hypothesis concerns the quality of the model. Usually two quantities describe the accuracy of a fit: the `residual standard error` (RSE) and the R^2 statistics. The RSE is defined as follows, i.e.

$$RSE = \sqrt{\frac{1}{n-2}RSS}, \tag{3.6}$$

whereas RSS is defined in Equation 3.2. In general it can be seen as a lack of fit to the model. A small value means that the model fits well. It provides an absolute measurement of the quality and is measured in units of y. However, small values of y result in a small RSE. Therefore, it is not always clear what states a good value of the RSE. The R^2 statistics provides a relative solution, because it is normalized to be between 0 and 1. Therefore, it is independent of the scale of y. The following term

$$R^2 = \frac{TSS - RSS}{TSS} \tag{3.7}$$

is used to calculate the statistics. RSS measures the amount of variability that remains unexplained after performing regression. Therefore, $RSS - TSS$ measures the amount of variability that is explained by the model. R^2 measures the part of variability of y that can be explained by x. Values close to 1 indicate a good model.

How important is a single factor for the overall fit? In order to measure the influence of a single predictor the standard error $SE(\beta_j)$ is computed for every factor. Equation

$$SE(\beta_0)^2 = \sigma^2 \left[\frac{1}{n} + \frac{\overline{x}^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2} \right] \tag{3.8}$$

and

$$SE(\beta_j)^2 = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i^j - \overline{x}^j)^2} \tag{3.9}$$

were used to calculate the SE for each β_i. σ^2 states the variance of the data. This values can be used to compute the confidence intervals. A 95% confidence interval is the range of values, where with a probability of 95% the true value is within that range. In case of linear regression the confidence interval for each predictor can be obtained by

$$\beta_j \pm 2 \cdot SE(\beta_j). \tag{3.10}$$

To get a measure of importance the t-statistic is computed in Equation 3.11, which measures the amount of deviation that β_j is away from 0. This value follows a t-distribution.

$$t_j = \frac{\beta_j}{SE(\beta_j)} \tag{3.11}$$

The p-value is the probability of observing any value that equals t_j or is larger, assuming that β_j is zero. A small p-value indicates that it is likely that a connection between the predictor and the response exists. In practice p-values smaller than 5% reject the null hypothesis for the given predictor.

3.2.2 Applied Linear Model

As mentioned before our predictor variables consists of time of the day, weekday, and current month. Day times are normalized to be between 0 and 1. Therefore, this predictor states a continuous variable. It is assumed that execution times change non-linear over time and are modeled with higher order terms up to the order of three.

In case of weekdays and months only discrete values are assigned, i.e values from 1 (Monday) to 7 (Sunday) and 1 (January) to 12 (December), respectively. For each day and month binary variables are introduced, which have the following meaning:

$$x_{wd,i} = \begin{cases} 1 & \text{if the } i^{th} \text{ day of the week is modeled} \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

and

$$x_{m,j} = \begin{cases} 1 & \text{if the } j^{th} \text{ month is modeled} \\ 0 & \text{otherwise} \end{cases} \tag{3.13}$$

Each variable gets a dedicated parameter $\beta_{wd,i}$ or $\beta_{m,j}$. Therefore, the overall regression function for modeling execution times is modeled as

$$y_{ex} = \beta_0 + \beta_{day,1} \cdot x_{day} + \beta_{day,2} \cdot x_{day}^2 + \beta_{day,3} \cdot x_{day}^3 + \sum_{i=2}^{7} \beta_{wd,i} \cdot x_{wd,i} + \sum_{j=2}^{12} \beta_{m,j} \cdot x_{m,j}, \quad (3.14)$$

where $x_{wd,i} \in \{0,1\}$ and $x_{m,i} \in \{0,1\}$. It is important to note that both sums start at 2, meaning that $\beta_0 + \beta_{day,1} \cdot x_{day} + \beta_{day,2} \cdot x_{day}^2 + \beta_{day,3} \cdot x_{day}^3$ models execution times on Monday in January, which states the reference point. $\beta_{wd,i}$ and $\beta_{m,j}$ hold information about the average difference per day respectively month to the reference day. The overall number of parameters to be learned is 21.

3.3 Evaluation of Local Regression

For each benchmark test and cloud service file set (see Table 2.1) are uploaded to the cloud storage, preceded by a download. For each benchmark and location (London and Singapore) a separate model has to be fit to the collected data. The most interesting model occurred for uploading large files. Since the benchmarks behave similar, this case is investigated further. In the following sections the model for the benchmark is discussed with further details for Dropbox and Google Drive in a separated manner. Afterwards both providers are compared against each other.

3.3.1 Model for Uploading large Files to Dropbox

Measured execution times and fitted model of measurements in London and Singapore can be seen in Figure 3.2. The red boundaries depict the corresponding confidence interval (see Equation 3.10). A observation is that after noon (normalized: > 0.5) larger execution times are observed in London and the other way around in Singapore.

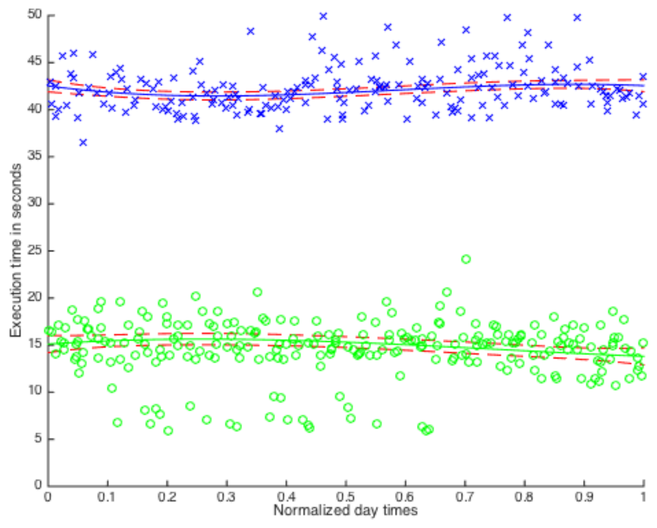


Figure 3.2: Fitted model for uploading large files to Dropbox in London (blue) and Singapore (green).

A first check is made by fitting a straight line with slope of zero into the area between the upper and lower confidence bound. In both cases it is not possible. Another observations is that operations in Singapore are almost three times faster than in London. Estimated model parameters, SE, and p-values are listed in Table 3.1. Only the first

three months are evaluated in order to determine the influence of certain months.
In case of modeling the cubic function of the day times in London all three parameters got a p-value of less than 5%, meaning the null-hypothesis can be rejected in this case. This proofs that there is a relationship between the time of the day and measured execution time for large files uploaded to Dropbox. The difference between days of the week are modeled with parameter $\beta_{wd,j}$, where $j > 2$. In this case Monday is used as reference point, meaning that $\beta_{wd,2}$ states the average difference between execution times on Monday and Tuesday, which is 0.61 seconds. This little difference results in a rather large p-value of 0.014. The largest difference can be found on Saturday with almost one second of variation. If we take a look at the parameter dedicated to modeling the month little dependency exists, which results in rather large p-values. This means that the dependency of the month is questionable and needs further evaluation.

Having a look at the estimated parameters for Singapore worse p-values are computed. No coefficients trained for modeling the day times have a p-value less than 5%. So the null hypothesis cannot be rejected for these parameters. On the other hand, very small p-values occur for days of the week and months, meaning that there is a relationship between weekday and month and the reached execution time.

	London			Singapore		
Coefficient	**Value**	**SE**	**p-value**	**Value**	**SE**	**p-value**
β_0	42.799	0.36684	approx. 0	18.716	0.53617	$2.6263 \cdot 10^{-212}$
$\beta_{day,1}$	-8.4859	2.3399	0.0002956	4.3606	3.3424	0.19216
$\beta_{day,2}$	19.796	5.4323	0.00027642	-10.135	7.7999	0.19395
$\beta_{day,3}$	-11.289	3.5662	0.0015751	4.4731	5.1383	0.38411
$\beta_{wd,2}$	0.61064	0.24753	0.013723	-0.51775	0.35875	0.14911
$\beta_{wd,3}$	0.18179	0.259	0.48285	-1.3313	0.38086	0.00048251
$\beta_{wd,4}$	0.47006	0.26498	0.076252	-1.1065	0.38088	0.003708
$\beta_{wd,5}$	0.88268	0.26492	0.00088145	0.40142	0.38131	0.29258
$\beta_{wd,6}$	0.95631	0.26171	0.00026589	0.38497	0.38041	0.31165
$\beta_{wd,7}$	0.71557	0.26104	0.0061851	0.27391	0.37323	0.46309
$\beta_{m,2}$	-0.27066	0.18475	0.1431	-3.6078	0.28283	5.4432e-36
$\beta_{m,3}$	0.80021	0.35642	0.024889	-2.3298	0.53181	1.2382e-05

Table 3.1: Estimated model parameters for uploading large files to Dropbox.

Table 3.2 showed quantities concerning the quality of the fit. Single quantities were explained earlier in this section.

Quantity	**London**	**Singapore**
Residual Standard Error	2.80	4.56
R^2	0.0941	0.2680
F-statistic	17.1	72.4

Table 3.2: Quality of model for uploading large files to Dropbox.

A standard residual error (RSE) of 2.80 was calculated for London, meaning that on average the response deviates 2.80 seconds from the regression line. A worse RSE is calculated for Singapore. Mainly because the data set has a larger TSS, which is hard to model with low complexity. A relatively small value of R^2 indicates that there is a relationship between the predictors, but it is a weak one. It matches with the observations that a straight line nearly matches into the confidence interval. The F-statistic values of 17.1 and 72.4 were computed for London and Singapore, respectively. It is larger that one, so the null hypothesis can be reject twice based on these quantities. On the other hand it is not large enough to classify the models as good ones. In general measurements of Singapore are modeled with a higher accuracy that the model of London, which was modeled with a residual standard error of 4.56F.

One reason for the short coming of the model for the execution times in London may be the fact that the relationship between weekday and month is not strong enough. The next step is to remove this relationship in the model and compare the results.

Adapting the model

Now the parameters for weekdays and months are removed and the model is learned again. This time the day or month on which the measurement was executed is not relevant anymore. The quality and accuracy of the fitted model is afterwards compared to the previous one. The following equation shows the new model

$$t'_{ex} = \beta_0 + \beta_{day,1} \cdot x_{day} + \beta_{day,2} \cdot x_{day}^2 + \beta_{day,3} \cdot x_{day}^3. \tag{3.15}$$

Measured data points and the fitted model are depicted in Figures 3.3 to 3.6. Execution times measured in London have a mean of 43.42 seconds and a variance of 19.51. At a first glance execution times in Singapore seem to have less deviation, but

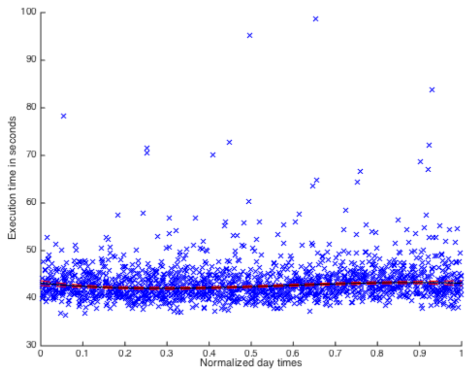


Figure 3.3: Adapted model and measured data points for uploading large files to Dropbox in London.

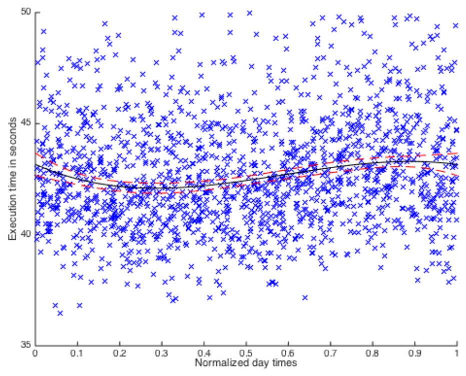


Figure 3.4: Adapted model and measured data points for uploading large files to Dropbox in London (zoomed).

outliers are larger and therefore have a bigger impact. Outliers are not yet handled, because too little measurements are available right now to classify them as discardable measurements. Execution times in Singapore have a mean of 17.48 seconds and a variance of 284.31. Ignoring the outliers for now, it can be said that execution times in Singapore increase beginning with noon. Many outliers are gathered at times around 130 seconds. It can be assumed that there is a weak accumulation point at this region. Further investigation will be done in the future after collecting more data.

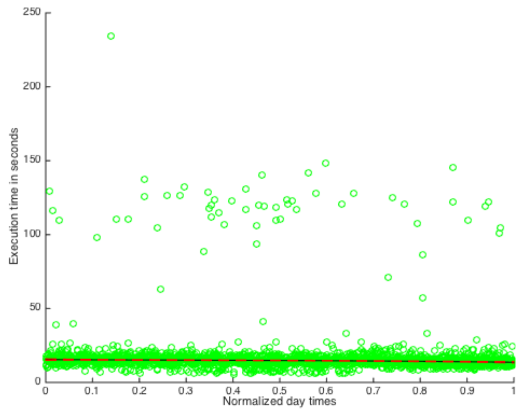


Figure 3.5: Adapted model and measured data points for uploading large files to Dropbox in Singapore.

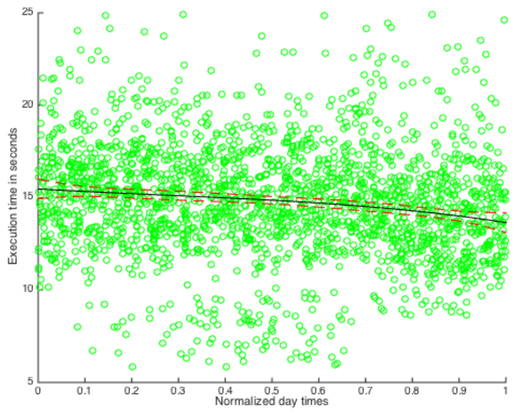


Figure 3.6: Adapted model and measured data points for uploading large files to Dropbox in Singapore (zoomed).

Having a look at the learned parameters it can be seen that the 3^{rd} order factor is more relevant in the model for London. In case of modeling execution times in Singapore a linear model is sufficient. This is reflected in large p-values close to one. Estimated coefficients are listed in Table 3.3.

	London			Singapore		
Coefficient	**Estimate**	**SE**	**p-value**	**Estimate**	**SE**	**p-value**
β_0	43.168	0.25795	approx. 0	15.472	0.27143	approx. 0
$\beta_{day,1}$	-8.4079	2.2298	0.0001684	-1.471	2.3613	0.53337
$\beta_{day,2}$	19.414	5.1747	0.00018157	1.3516	5.51	0.80625
$\beta_{day,3}$	-11.011	3.3964	0.0012096	-1.6979	3.6299	0.64002

Table 3.3: Estimated model parameters using the adapted model for uploading large files to Dropbox.

When it comes to determining the overall quality of the fit, better results occur than modeling the day of the week and months. Corresponding quantities can be found in Table 3.4. Especially for Singapore a more accurate model is learned with a rather good R^2 of 0.422 and a larger value for F of 532. In case of London the precision is worse than in Singapore. The value for R^2 reflects that less variability is explained by this model. Compared to the model which takes additional parameters into account leads to better results (cf. Table 3.2). Therefore, it can be assumed that only a weak relationship between weekdays and months exists.

Quantity	**London**	**Singapore**
Residual Standard Error	2.67	3.23
R^2	0.095	0.422
F-statistic	59.5	532.0

Table 3.4: Quality of adapted model for uploading large files to Dropbox.

3.3.2 Model for Uploading large Files to Google Drive

For Google Drive the model without weekdays and months leads to better fitting results. Therefore, the second model is used initially. Looking at Figure 3.7 and 3.8 it can be seen that beginning with noon execution times are increasing. An additional observation is that both curves have a similar shape. When comparing β_0 of both locations in Table 3.4 execution times in London are almost twice as fast as in Singapore.

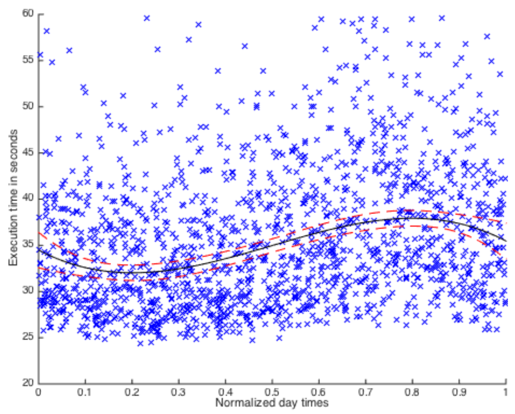


Figure 3.7: Adapted model and measured data points for uploading large files to Google Drive in London (zoomed).

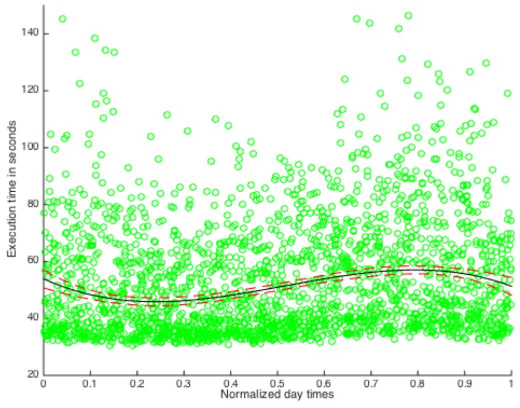


Figure 3.8: Adapted model and measured data points for uploading large files to Google Drive in Singapore (zoomed).

Computed p-values are much smaller than for Dropbox. Therefore, the null hypothesis can be discarded. This means there is a strong relationship between the time of the day and the execution times. Additionally, it can be said that all four parameters are strongly involved in the model, because no p-value exceeds 5%.

	London			Singapore		
Coefficient	**Estimate**	**SE**	**p-value**	**Estimate**	**SE**	**p-value**
β_0	34.486	0.97529	$5.8947 \cdot 10^{-206}$	53.904	1.6049	$3.6249 \cdot 10^{-200}$
$\beta_{day,1}$	-26.648	8.4801	0.0017041	-74.482	13.887	$9.0092 \cdot 10^{-08}$
$\beta_{day,2}$	82.928	19.78	$2.8994 \cdot 10^{-5}$	202.53	32.332	$4.493 \cdot 10^{-10}$
$\beta_{day,3}$	-55.297	13.032	$2.3203 \cdot 10^{-5}$	-130.64	21.28	$9.8156 \cdot 10^{-10}$

Table 3.5: Estimated model parameters using the adapted model for uploading large files to Google Drive.

The quality of the model is reflected in a good value for R^2 and F-statistic. A large RSE is explained by a high value of the total sum of squares (TSS). The initial variability is very high and can not be explained entirely by the trained model. Both model fit the data well.

Quantity	**London**	**Singapore**
Residual Standard Error	10.2	18.9
R^2	0.213	0.175
F-statistic	154.0	157.0

Table 3.6: Quality of adapted model for uploading large files to Google Drive.

3.3.3 Comparison of Cloud Storage Provider

In the previous sections every cloud service was investigated on its own. In this section the two cloud service providers are compared for London and Singapore. In case of London (see Figure 3.9) Dropbox is on average about 10 seconds faster during the entire day and the performance does not vary that much. Dropbox would be the optimal provider if the client is located in London. On the other hand, the average difference in Singapore is much higher. Dropbox is almost 40 seconds slower than Google Drive. Here Google Drive is the optimal provider, i.e it is roughly 3 times faster. Furthermore, there is a stronger dependency on the day time.

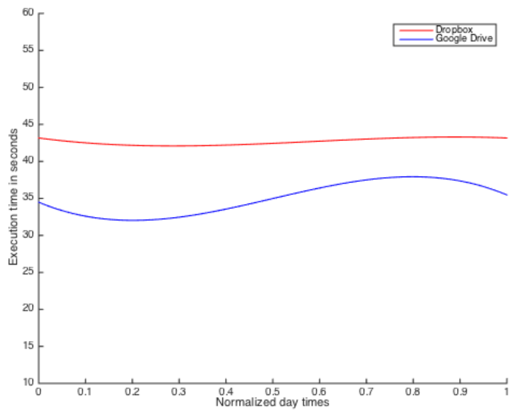


Figure 3.9: Comparison of execution times for Dropbox and Google Drive in London.

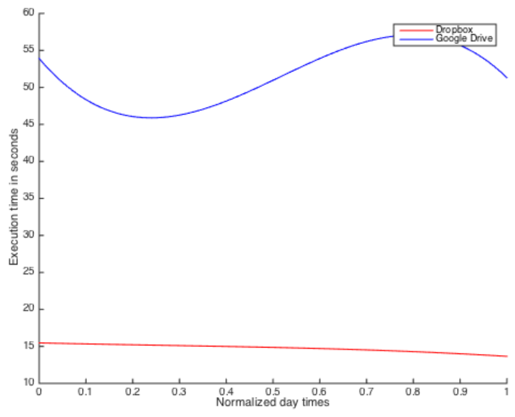


Figure 3.10: Comparison of execution times for Dropbox and Google Drive in Singapore.

4 Global Modeling Approaches

After determining a local model regression was extended to a global scale by adding observations from several distributed measurement servers. The goal is to create a model of execution time for each cloud service provider in order to predict future performance. Predicted execution times can be used to classify the optimal storage provider with given location and daytime. In the first step the amount of data is reduced in order to speed up computation time. This is done by reducing the amount of possible daytimes to 24, whereas each hour of the day an averaged value is assigned.

In the second step global regression is applied. The input space of the geographic model consists of latitude and longitude, which are an angular measurement do define specific points on the Earth's surface. Latitudes (ϕ) are geographic coordinates that specifies the north-south position of a point on the Earth's surface. On the other hand longitudes (λ) are the east-west position of a point on the Earth's surface. Ranges of each type of coordinate are defined as follows

$$\begin{aligned} \text{latitude:} \quad & \phi & -90^\circ \le \phi < 90^\circ \\ \text{longitude:} \quad & \lambda & -180^\circ \le \lambda < 180^\circ. \end{aligned}$$

For every hour of the day a separate model is computed. The overall input space $\mathbf{x}$ for predicting results is latitude, longitude, hour of the day. The goal of the following regression tasks is to find a mapping for each cloud service, file type and operation, which maps given geographical coordinates and hour of the day on execution time of certain operations.

Several regression methods are applied to measured data sets and optimal techniques chosen by evaluating regression performance of each method. Advantages and disadvantages are discussed with respect to prediction performance. The preceding Chapter applies multiple linear regression to collected data and states the simplest and most intuitive technique for the given domain. Later on, fitting of two dimensional Gaussian functions with fixed centers is explained, followed by support vector machine (SVM) regression. SVMs are supervised learning models that analyze data used for classification and regression analysis. Several approaches for applying SVMs in order to solve regression problems exists. This subcategory of kernel methods is used for building a predictive model for execution times. Finally the K-nearest neighbor classifier is used in order to find the optimal provider.

4.1 Data Preprocessing

Again, outliers were removed using mechanisms explained in Section 3.1. In order to reduce training data by a reasonable amount a moving average approach was used. The data set is analyzed and reduced by taking the average of different subsets of the data. Execution times over the day are split into 24 subsets, whereas subsets have a fixed size and are non-intersecting. Therefore, each subset represents the mean execution time within a certain hour of the day. This is done after excluding outliers (see Section 3.1). Applied reduction on data can be found in Figure 4.1. It can be seen that execution

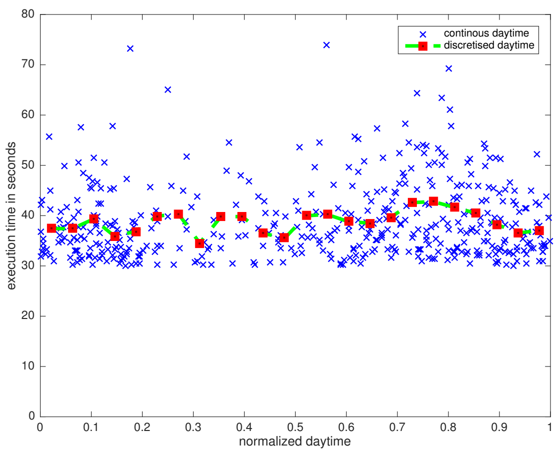


Figure 4.1: 24 discretized daytimes by taking the mean execution time of each hour. Applied to collected data for uploaded file with size of 100MB to OneDrive.

times do not change much between consecutive hours of the day. Moving average smoothing is not influencing the overall model heavily, whereas reducing computation time. Measured execution times y_h^* after discretization are calculated as follows

$$y_h^* = \frac{\sum\limits_{i=1}^{n} \gamma_{h,i} \cdot y_i}{\sum\limits_{i=1}^{n} \gamma_{h,i}} \qquad h \in \{1, 2, \dots, 24\}, \quad \gamma_{h,i} \in \{0, 1\} \tag{4.1}$$

whereas

$$\gamma_{h,i} = \begin{cases} 1, & \text{if } x_i \geq \frac{h-1}{24} \text{ and } x_i < \frac{h}{24}, \\ 0, & \text{otherwise.} \end{cases} \tag{4.2}$$

4.2 Modeling Approaches

Several regression analysis approaches were taken in order to find an optimal model for extrapolating future performance. Each created regression function works differently and provides varying predictive performance. Performances of each technique are compared with each other and the best method that maps input $\mathbf{x}$ to target y chosen for further prediction. The general goal is to find a function

$$\Psi : \mathbf{x} \to y, \tag{4.3}$$

which minimizes regression error. The variable $\mathbf{x}$ states a tuple $\langle \phi, \lambda, h \rangle$ containing geographical coordinates (ϕ, λ) and the hour of the day h. Values of latitude variables are defined as

$$\{\phi \in \mathbb{R} \mid -90 \leq \phi < 90\} \tag{4.4}$$

and longitude variables as

$$\{\lambda \in \mathbb{R} \mid -180 \leq \lambda < 180\}. \tag{4.5}$$

Hours of the day range from 1 to 24, therefore

$$\{h \in \mathbb{N} \mid 1 \leq h \leq 24\}. \tag{4.6}$$

The output domain Y consists of modeled execution time in seconds ($y \in \mathbb{R}$). As already mentioned, the function Ψ predicts execution times for each benchmarked cloud service separately, which is used to determine the optimal storage provider with given $\langle \phi, \lambda, h \rangle$ by choosing the one with the fastest execution time.

4.2.1 Multiple Linear Regression

In order to build a global model using multiple linear regression a similar approach as in Section 3.2 is taken. Latitudes and longitudes are modeled up to the power of N_p. The used regression function for learning is

$$y_h = \Psi_{mlr}(\phi, \lambda, h) = f_h(\phi, \lambda) = \beta_0 + \sum_{i=1}^{N_p} \beta_{\phi,i,h} \cdot \phi^i + \sum_{i=1}^{N_p} \beta_{\lambda,i,h} \cdot \lambda^i, \tag{4.7}$$

where we trained with samples from y_h^*. Regression is applied separately for all 24 hours of a day h and each cloud service provider. The optimal degree of polynomials N_p is determined by applying regressions to a limited amount of data points with different values for N_p and chose the value with the best regression performance on a validation set.

4.2.2 Fitting Gaussian Functions

It is assumed that performance increases (i.e. low execution times) the closer clients are located to the closest data center (as mentioned in [6]). Decreasing performance can be modeled by the natural exponential function. This assumption led to the idea of fitting two dimensional Gaussian functions with fixed location of the center (mean) to collected data sets. A separate Gaussian function is dedicated to each data center, whereas the center of the according Gaussian function is fixed at the geographical location of the data center. The following equations describe the used modeling function

$$\begin{aligned} y_h &= \Psi_{gc}(\phi, \lambda, h) = f_h(\phi, \lambda) \\ &= \sum_{i=1}^{N_s} A_{i,h} \exp\left(-\Big(a_{i,h}(\phi - \phi_{dc,i})^2 - 2b_{i,h}(\phi - \phi_{dc,i})(\lambda - \lambda_{dc,i}) + c_{i,h}(\lambda - \lambda_{dc,i})^2\Big)\right), \end{aligned} \tag{4.8}$$

where N_s is the number of data centers of a given cloud storage provider. Parameters $a_{i,h}$ to $c_{i,h}$ are defined as

$$a_{i,h} = \frac{\cos^2(\theta_{i,h})}{2\sigma^2_{\phi,i,h}} + \frac{\sin^2(\theta_{i,h})}{2\sigma^2_{\lambda,i,h}}, \tag{4.9}$$

$$b_{i,h} = -\frac{\sin(2\theta_{i,h})}{4\sigma^2_{\phi,i,h}} + \frac{\sin(2\theta_{i,h})}{4\sigma^2_{\lambda,i,h}} \tag{4.10}$$

and

$$c_{i,h} = \frac{\sin^2(\theta_{i,h})}{2\sigma^2_{\phi,i,h}} + \frac{\cos^2(\theta_{i,h})}{2\sigma^2_{\lambda,i,h}}. \tag{4.11}$$

They describe variance in direction of the latitude and longitude and orientation of the Gaussian function. The amplitude $A_{i,h}$ is the learned execution time at the data center, which location are fixed at $(\lambda_{dc,i},\ \phi_{dc,i})$. Impacts of single parameters are shown in Figure 4.2 and Figure 4.3. Both examples use a fictive data center and location (0,0).

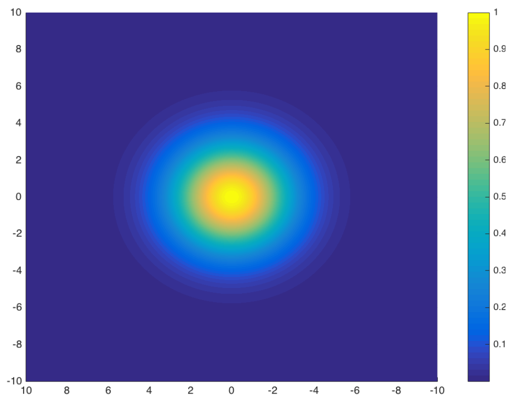


Figure 4.2: Example of Gaussian function with identical variance and no rotation ($A_{i,h} = 1$).

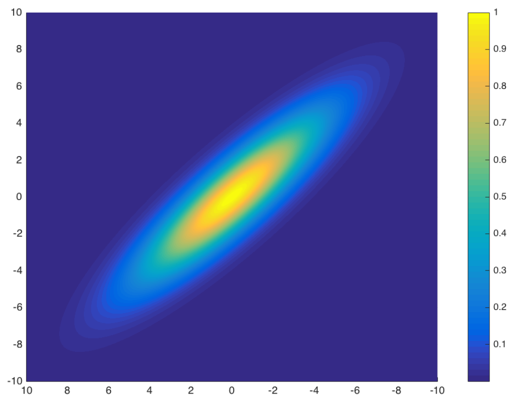


Figure 4.3: Example of Gaussian function with $\frac{\sigma_{\phi,i,h}}{\sigma_{\lambda,i,h}} = \frac{1}{4}$ and $\theta_{i,h} = \frac{\pi}{4}$ ($A_{i,h} = 1$).

Each model for Dropbox, Google Drive, and OneDrive is the sum of N_s Gaussian functions, whereas each function is defined by four parameters $A_{i,h}$, $\sigma_{\phi,i,h}$, $\sigma_{\lambda,i,h}$, and $\theta_{i,h}$. In case of intersecting curves, the lower value is selected. This results in an overall amount of $4N_s$ of parameters. A function of each data center has to be learned separately. Measured data in the close surrounding (i.e. measurement servers within a radius of 50 coordinate units) are used to learn the parameters. The resulting non-linear least squares problem is solved by using the Levenberg-Marquadt algorithm. This algorithm was available in the Matlab framework and implemented in the function `lsqcurvefit`.

In some regions not sufficient measurement servers were available in order to solve the least square problem. It is assumed that connections to the data center behave globally similar. Parameters of Gaussian functions of data center where not enough data points were available are estimated to be the mean of all other learned parameters.

4.2.3 Support Vector Machine Regression

A more powerful regression tool is support vector machine regression (SVR). In the field of machine learning support vector machines are models learned by supervised learning algorithms. It is mainly used for classification and regression analysis. SVM has been developed mainly from Cortes and Vapnik under the name support vector networks [20] and was mainly aimed to solve two-group classification problems. One year after their publication they applied SVM to solve regression problems [21]. The first part of this Section introduces SVM in general preceded by describing how SVM solves the given multidimensional regression problem. Knowledge was mainly gathered from Chapter 7 of [1], from which illustrations in the following Sections are originated.

Introduction to Support Vector Machines

The main idea of support vector machines is solving a two-class linear classification problem by usage of the following equation

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b \tag{4.12}$$

where $\phi(\mathbf{x})$ is a feature-space transformation and b is an explicit bias parameter. Parameters $\mathbf{w}$ and b are trained with N input vectors $\mathbf{x}_1, \cdots, \mathbf{x}_N$ with corresponding target values $t_1, \cdots, t_N$ where $t_n \in \{-1, 1\}$. New data points are classified by taking the sign of $y(\mathbf{x})$ into account. If the data set can be linear separated, there exists at least one set of parameters $\mathbf{w}$ and b for which $y(\mathbf{x}_n) > 0$ if $t_n = +1$ and $y(\mathbf{x}_n) < 0$ if $t_n = -1$. Hence,

$$t_n y(\mathbf{x}_n) > 0 \tag{4.13}$$

is true for all points in the training set. Of course many parameters exist for which Equation 4.13 holds. SVM finds the best set of parameters by introducing the concept of *margin*, which is the smallest distance between the decision boundary and any sample from the training set. Optimal parameters maximize the margin. Figure 4.4 depicts this concept and Figure 4.5 shows an example for the optimal decision boundary. The

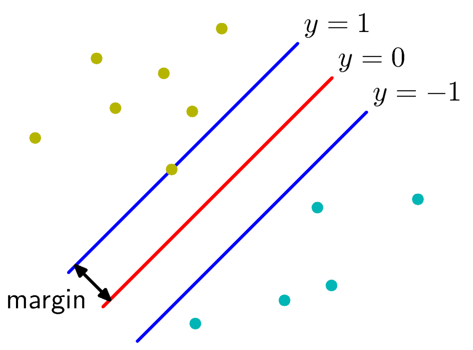


Figure 4.4: Margin is defined as perpendicular distance between the decision boundary (red line) and the nearest data point in the training data set. Figure originated from [1].

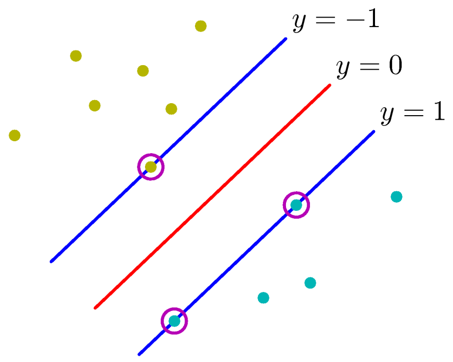


Figure 4.5: Optimal decision boundary (red line) and support vectors (circles). Figure originated from [1].

decision boundary is mathematically a hyperplane where point on the plane fulfil $y(\mathbf{x}) = 0$. According to [1] this hyperplane is defined by parameters which maximize the perpendicular distance to the closest training point $\mathbf{x}_n$. These parameters can be found by solving

$$\arg\max_{\mathbf{w},b} \left\{ \frac{1}{\|\mathbf{w}\|} \min_{n} \left[t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b) \right] \right\}. \tag{4.14}$$

Division by the norm of $\mathbf{w}$ in necessary to penalize complex models and therefore avoid overfitting. After introducing Lagrange multipliers $\mathbf{a} = (a_1, \cdots, a_N)^T$ a *dual* *representation* of the maximum margin can be found by maximizing

$$\widetilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m), \tag{4.15}$$

with the following constraints

$$a_n \geq 0, \tag{4.16}$$

and

$$\sum_{n=1}^{N} a_n t_n = 0, \qquad\qquad n = 1, \cdots N. \tag{4.17}$$

In the *dual representation* the kernel is defined by $k(x, x') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$. After solving this quadratic problem with constraints the following function results

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n a_m k(\mathbf{x}, \mathbf{x}_n) + b. \tag{4.18}$$

Further information about the derivation can be found in Chapter 7 of [1].

Non-linearly separable data points So far training data was linearly separable in the feature space $\phi(\mathbf{x})$. In that case the support vector machine will find the optimal separation plane. On the other hand, it will fail if linear separation is impossible. In order to circumvent this behavior [20] introduced so called *slack variables* $\xi_n \geq 0$ for each training data point $\mathbf{x}_n$. They penalize points, which are on the wrong side of the margin linearly with the distance. *Slack variables* are defined as $\xi_n = 0$ for points that are on or inside the correct margin boundary and $\xi_n = |t_n - y(\mathbf{x}_n)|$ for points on the other side of the decision boundary. Data points on the decision boundary $y(\mathbf{x}_n) = 0$ will have $\xi_n = 1$. Points with $\xi_n > 1$ are classified wrongly. Figure 4.6 illustrates the behavior of ξ_n.

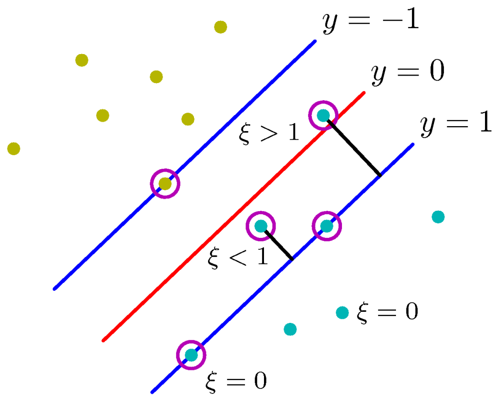


Figure 4.6: Definition of slack variables. Figure originated from [1].

This additional freedom leads to *soft margin* classification constraints

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n, \;\; \forall \;\; n = 1, \ldots N. \tag{4.19}$$

For this case the following function must be minimized in order to apply a *soft margin*

$$C \sum_{n=1}^{N} \xi_n + \frac{1}{2}\|\mathbf{w}\|^2, \tag{4.20}$$

where $C > 0$ controls the balance between penalty coming from the *slack variable* and penalty from the margin. Solving this problem leads to the same *dual representation* as in Equation 4.15, but with the following constraints

$$0 \geq a_n \geq C, \tag{4.21}$$

and

$$\sum_{n=1}^{N} a_n t_n = 0, \qquad n = 1, \cdots N. \tag{4.22}$$

The constraint in Equation 4.21 is known as *box constraints*.

Support Vector Machines for Regression

In order to solve regression problems using SVMs [20], the quadratic error function is replaced with an ϵ*-sensitive error function*,

$$E(\mathbf{w}) = C \sum_{n=1}^{N} E_\epsilon\Big(y(\mathbf{x_n}) - t_n\Big) + \frac{1}{2}\|\mathbf{w}\|^2, \tag{4.23}$$

whereas

$$E_\epsilon(y(\mathbf{x}) - t) = \begin{cases} 0, & \text{if } |y(\mathbf{x}) - t| < \epsilon, \\ |y(\mathbf{x}) - t| - \epsilon, & \text{otherwise.} \end{cases} \tag{4.24}$$

In case of regression $y(\mathbf{x_n}) \in \mathbb{R}$ and $t_n \in \mathbb{R}$ are real values. The new error function E_ϵ is zero if the absolute distance between the prediction and the target is smaller than ϵ, where $\epsilon > 0$. Equation 4.24 states a simple error function, where the error increases linear with the distance. An example plot of the introduced error function is shown in Figure 4.7. In order to solve this optimization problem using Lagrange multiplier,

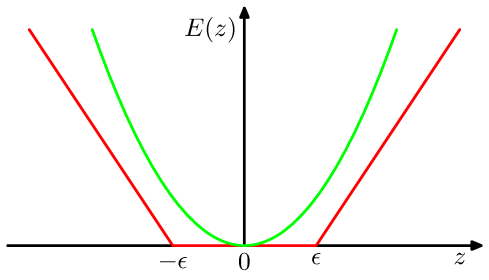


Figure 4.7: Example plot of linear ϵ-sensitive error function in comparison with quadratic error function. Figure originated from [1].

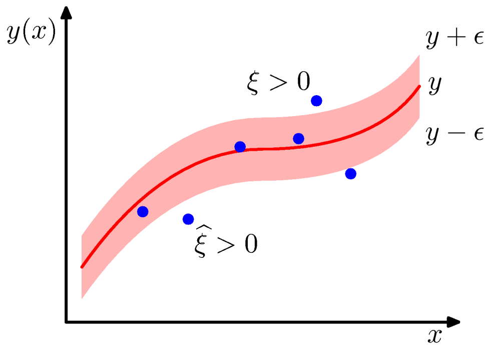


Figure 4.8: Applied SVM regression with ϵ-tube and slack variables. Figure originated from [1].

slack variables are used. Two *slack variables* $\xi_n \geq 0$ and $\hat{\xi_n} \geq 0$ are needed for each data point $\mathbf{x}_n$. The first *slack variable* ξ corresponds to points above the ϵ-tube, where $t_n > y(\mathbf{x}_n) + \epsilon$ holds. The other variable $\hat{\xi}$ is dedicated to variables underneath the ϵ-tube, where $t_n < y(\mathbf{x}_n) - \epsilon$ is true. If $\xi > 0$ then $\hat{\xi} = 0$ and if $\hat{\xi} > 0$ then $\xi = 0$. Points inside the tube fulfill the condition $y(\mathbf{x}_n) - \epsilon \leq t_n \leq y(\mathbf{x}_n) + \epsilon$. Figure 4.8 illustrates the mentioned conditions graphically. Similar to SVMs with soft margin, this allows data points to be outside of the ϵ-tube, whereas the following conditions must hold

$$t_n \geq y(\mathbf{x}_n) + \epsilon + \xi_n \tag{4.25}$$

$$t_n \leq y(\mathbf{x}_n) - \epsilon - \hat{\xi}_n. \tag{4.26}$$

Using *slack variables* the following error function for support vector regression results in

$$C \sum_{n=1}^{N} (\xi_n + \hat{\xi}_n) + \frac{1}{2}\|\mathbf{w}\|^2. \tag{4.27}$$

Again, by applying Lagrangian multipliers (see Chapter 7 of [1] for further derivation) the *dual representation* of the problem is formulated as

$$\begin{aligned} \widetilde{L}(\mathbf{a}, \hat{\mathbf{a}}) = -\frac{1}{2} \sum_{n=1}^{N} \sum_{m=1} N(a_n + \hat{a}_n)(a_m + \hat{a}_m) k(\mathbf{x}_n, \mathbf{x}_m) - \epsilon \sum_{n=1}^{N} (a_n + \hat{a}) \\ + \sum_{n=1}^{N} (a_n - \hat{a}) t_n, \end{aligned} \tag{4.28}$$

where $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})\phi(\mathbf{x}')$ is the kernel function. Solving this Lagrange problem with the following constraints

$$\sum_{n=1}^{N} (a_n - \hat{a}_n) = 0, \tag{4.29}$$

$$0 \leq a_n \leq C, \tag{4.30}$$

$$0 \leq \hat{a}_n \leq C, \tag{4.31}$$

leads to the regression function

$$y(\mathbf{x}) = \sum_{n=1}^{N} (a_n - \hat{a}_n) k(\mathbf{x}, \mathbf{x}_n) + b. \tag{4.32}$$

The bias parameter b is computed by averaging over all training examples

$$b = \frac{1}{N} \sum_{n=1}^{N} \Big(t_n - \epsilon - \sum_{m=1}^{N} (a_m - \hat{a}_m) k(\mathbf{x}_n, \mathbf{x}_m) \Big). \tag{4.33}$$

In SVM for regression only support vectors are contributing to the predictive model, i.e. $a_n \neq 0$ and $\hat{a}_n \neq 0$. Therefore, all points on the boundary of the ϵ-tube or outside are support vectors. Points inside the ϵ-tube are not contributing to the overall model, because $a_n = 0$ and $\hat{a}_n = 0$. The type of kernel function $k(\mathbf{x}, \mathbf{x}')$ is a design decision and has to be chosen based a priori information of the domain. In order to train a global model a Gaussian kernel

$$k_g(\mathbf{x}, \mathbf{x}') = exp\left(\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right) \tag{4.34}$$

was used, where σ is the width of the kernel. In order to model execution time over the day a separate parameter $\mathbf{a}_h$, $\hat{\mathbf{a}}_h$, and b_h were learned for each hour h and service. Regression is performed by computing

$$y_h = \Psi_{svm}(\phi, \lambda, h) = f_h(\phi, \lambda) = \sum_{n=1}^{N_s}(a_{n,h} - \hat{a}_{n,h})k_g\left(\begin{bmatrix}\phi\\ \lambda\end{bmatrix}, \begin{bmatrix}\phi_n\\ \lambda_n\end{bmatrix}\right) + b_h \tag{4.35}$$

for a given location $\begin{bmatrix}\phi & \lambda\end{bmatrix}^T$. In order to integrate data center locations into to the model, artificial data points with low execution times may be resampled at locations of each data center.

4.2.4 K-Nearest-Neighbor Classification

A different approach for finding the optimal provider without building global regression models is by simply taking executions time of the k closest measurement servers into account. This is done by performing a k-nearest neighbor search around a given location. If $k > 1$ more than one execution time is included in the algorithm. Measurements closer to the given location are modeled to be more important. This is done by weighting times depending on the distance and summing up all k normalized execution times (see Figure 4.36). Normalization increases linearly with distance to the measurement server. Figure 4.9 illustrates this behavior.

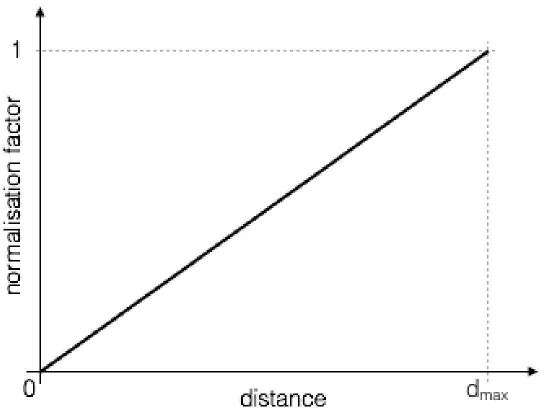


Figure 4.9: Factor for linearly normalizing execution times by taken the distance into account.

Based on decision values ρ_s for each cloud service the optimal provider is chosen, whereas

$$\rho_{h,s} = \sum_{i=1}^{k} \frac{d_i}{d_{max}} t_{h,i,s}. \tag{4.36}$$

Distance between the i-th measurement server, the test location is stored in d_i and h states the hour of the day. The distance of the most distant server is d_{max} and $t_{i,s}$ states measured execution time of service s at the given server. The provider with the smallest decision values is chosen to be optimal (see Equation 4.37).

$$\rho_h^* = \underset{s}{\operatorname{argmin}}(\rho_{h,s}) \tag{4.37}$$

The best value of k is determined by applying leave-one-out crossvalidation, whereas up to 10 neighbors are evaluated.

4.3 Evaluation of Global Regression

In this section performances of different regression and classification techniques are discussed and compared with each other. Performance of each model is assessed by applying leave-k-out-cross-validation, omitting a single measurement server from the training data. The removed server represents test data. Trained models are used to determine the best service at the omitted client's location. The optimal service found by the model is compared with the optimal server obtained by measurements (test data). As stated in Section 4.1 a moving average approach was taken in order to reduce the amount of training data points for each client. This leads to 24 data points (N_{test}) for each service and client and the number of total data points $N_d = 24 \cdot N_s$, whereas N_s is the number of measurements clients of service s. Training data ($N_{train} = N_d - N_{test}$) is applied to each modeling technique and performances are compared with each other. As error function the mean absolute percentage error (MAPE), i.e.,

$$\text{MAPE} = \frac{1}{24 N_{test}} \sum_{i=1}^{N_{test}} \sum_{h=1}^{24} \left| \frac{y_{i,h} - f_{i,h}}{f_{i,h}} \right| \tag{4.38}$$

is used to measure regression errors, where $y_{i,h}$ is the model prediction and $f_{i,h}$ is the actual value at hour h. It provides information of the error relative to the actual measurement. This relative error function was necessary, because files with different sizes were uploaded and downloaded with different averaged execution times. The MAPE divides the error by the actual value, which makes the error comparable and independent of the value range of execution times. Due to the fact that modeled execution times for each type of file only differ by a linear scaling factor, figures and performances are explicitly shown for downloading a large file (100MB) from each provider. The last section deals with comparison of execution times for different types of files and operations, i.e. uploads and downloads.

4.3.1 Evaluation of Multiple Linear Regression

In case of multiple linear regression a design parameter is the order of the applied polynomials N_p (see Equation 4.7). In order to determine N_p, 24 random samples are selected (out of N_d overall data samples) to be the test set. Models are computed using orders up to 8 and the performance assessed by using the set of test data points. Performance is measured by comparing the fastest service of all three models s_m with the fastest service in the test set s_t. The classification error rate (CE) quantifies the performance and is computed as follows

$$\text{CE} = \frac{1}{24} \sum_{i=1}^{24} \text{ce}(s_{m,i}, s_{t,i}), \tag{4.39}$$

where

$$\mathrm{ce}(s_m, s_t) = \begin{cases} 0 & \text{if } s_m = s_t \\ 1 & \text{otherwise} \end{cases} . \tag{4.40}$$

Figure 4.10 shows the reached classification error for each order N_p. It can be seen

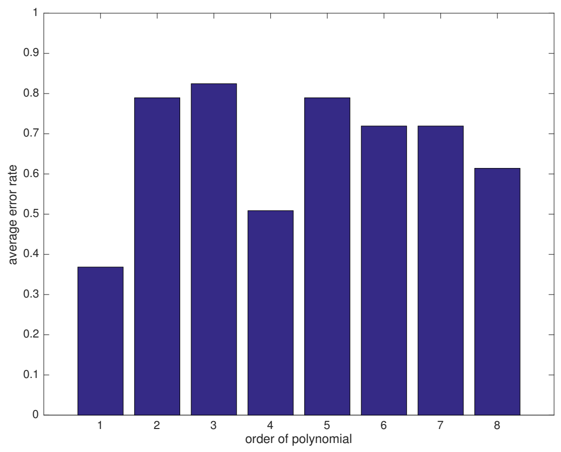


Figure 4.10: Classification error rate after applying different orders N_p of the used polynomial.

that an order of 1 leads to the smallest classification error rate of approximately 38% wrongly matched services. The main reason lies in the nature of polynomial curve fitting. In order to achieve a minimal training error with polynomials up to the 8th degree, values other than the training set are not modeled well enough. This leads to values smaller than zero in some regions, which again leads to this overfitted behavior. Linear modeling of geographical execution times shows the smallest generalization error. The following Figures depict modeled execution times for the first hour of the day.

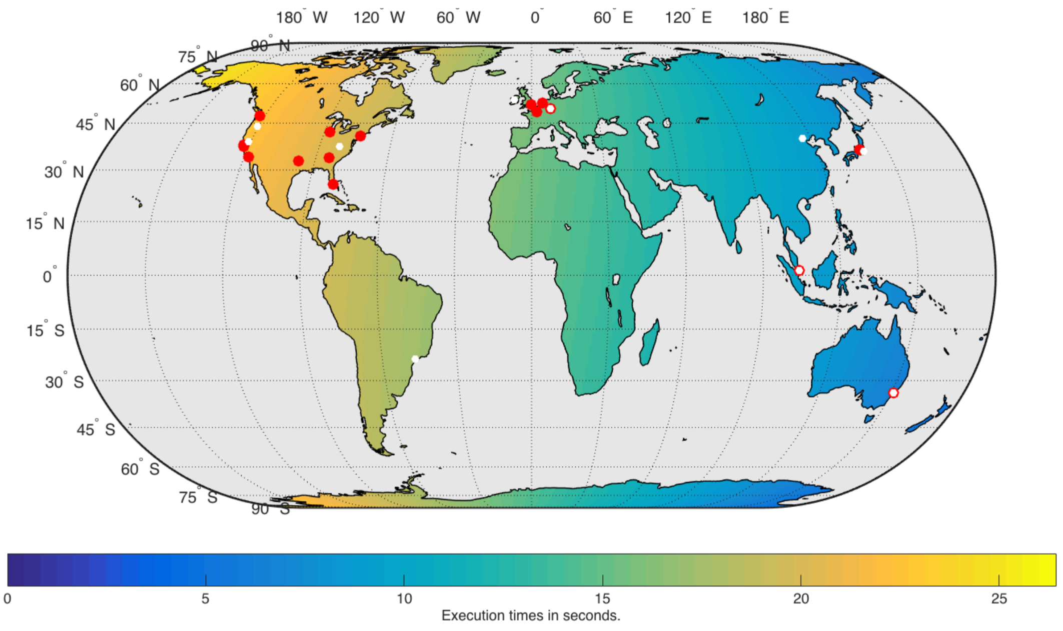


Figure 4.11: Multiple linear regression model of execution times for downloading a large file from Dropbox ($N_p = 1, h = 1, MAPE = 0.31\%$). Red dots indicate clients used for measurement. White dots indicate data centers of Dropbox.

Figure 4.11 to 4.13 depict trained model for downloading a 100MB file for each cloud service. The overall model is flat and is strictly monotonically decreasing or increasing in direction of each coordinate λ and ϕ. In case of Dropbox (Figure 4.11) modeled execution times are worse in the western hemisphere. Figure 4.12 shows the global

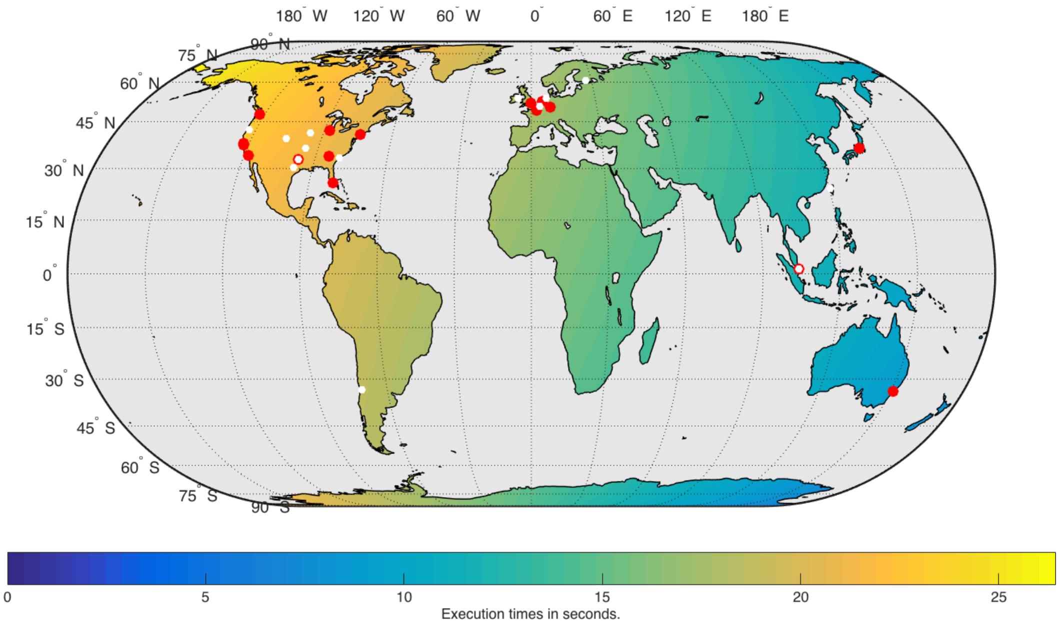


Figure 4.12: Multiple linear regression model of execution times for downloading a large file from Google Drive ($N_p = 1, h = 1, MAPE = 0.40\%$). Red dots indicate clients used for measurement. White dots indicate data centers of Google Drive.

regression for downloading a file from Google Drive. Execution times for OneDrive are shown in Figure 4.13. OneDrive performs better than the other two provides by having execution times between 20 and 40 seconds. Table 4.1 lists the parameters of the learned multiple linear regression model. In can be seen that the bias parameter of OneDrive β_0 is smaller than the bias parameters of Dropbox and Google Drive, which indicates a fast average of execution times. All parameters $\beta_{\phi,1,1}$ are below zero, meaning that performance is lower in the northern hemisphere and decreases to the South. On the other hand, $\beta_{\lambda,1,1}$ is above zero for each service. This indicates higher execution times in Europa than in North America.

Service	β_0	$\beta_{\phi,1,1}$	$\beta_{\lambda,1,1}$
Dropbox	15.121	-0.053	0.091
Google Drive	16.403	-0.042	0.021
OneDrive	8.774	-0.052	0.023

Table 4.1: Learned parameter after fitting Equation 4.7 with $N_p = 1$ and the first hour of the day ($h = 1$).

In case of downloading large files OneDrive is the best provider after modeling with multiple linear regression. Figure 4.14 depicts the best service globally. Due to the simplicity of the model ($N_p = 1$) no nonlinear global variation in execution times was considered. Therefore, in certain regions other cloud services may perform better if regression is done with nonlinear terms.

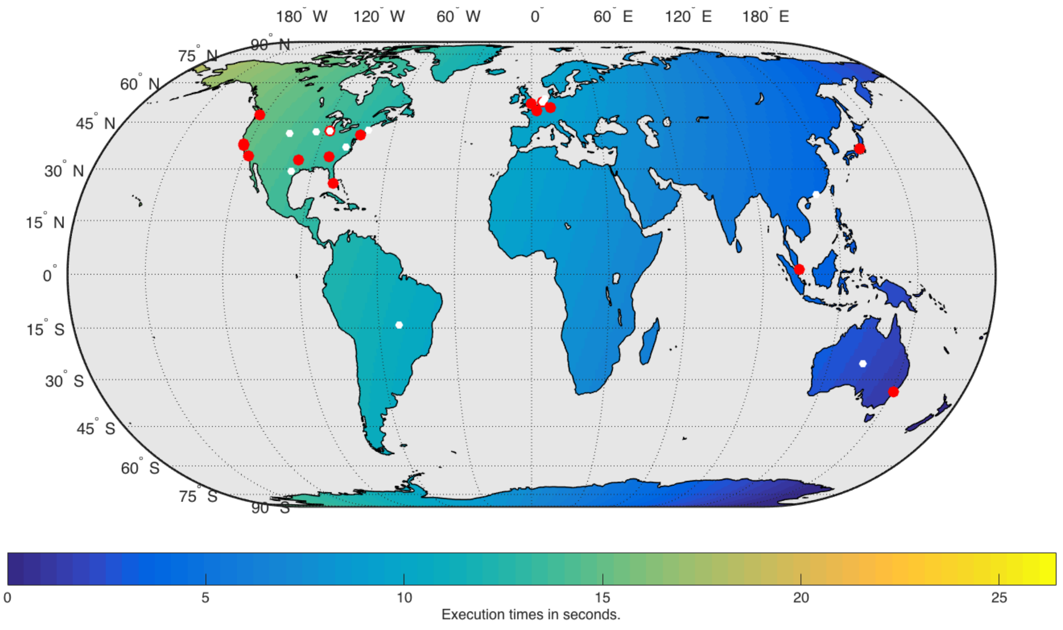


Figure 4.13: Multiple linear regression model of execution times for downloading a large file from OneDrive ($N_p = 1, h = 1, MAPE = 1.30\%$). Red dots indicate clients used for measurement. White dots indicate data centers of OneDrive.

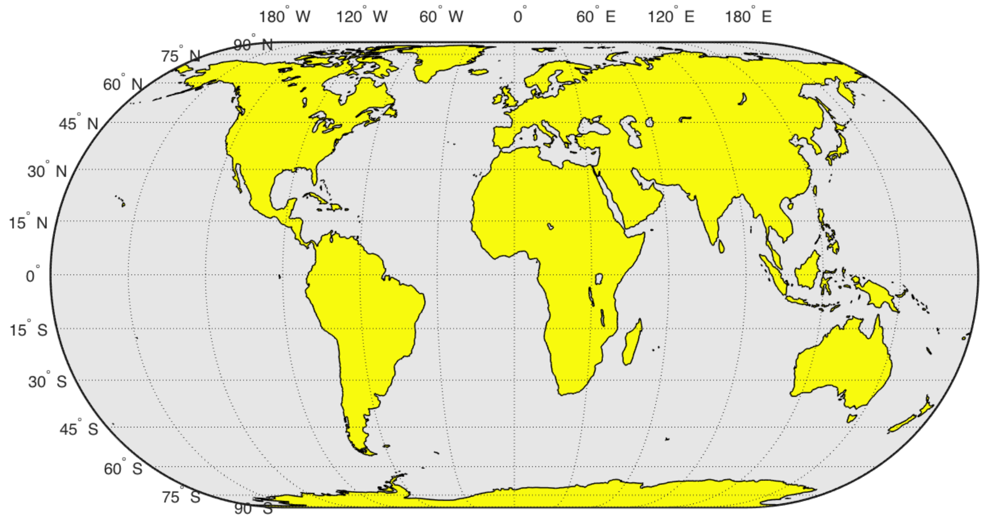


Figure 4.14: Classification of the fastest cloud storages for downloading large files ($N_p = 1, h = 1$). Blue areas indicate Dropbox, green areas indicate Google Drive, and yellow areas indicate OneDrive.

4.3.2 Evaluation of Gaussian Function Fitting

Each known data center a separate Gaussian function is dedicated. This approach was based on the assumption that performance of services decrease with increasing distance to the next data center. Figure 4.15 shows the global model of execution times for Dropbox. The more data centers within a certain area are, the lower the execution times are. This approach leads to higher performance in USA and Europe and large execution times in Africa and West Asia. Modeled execution times for Google

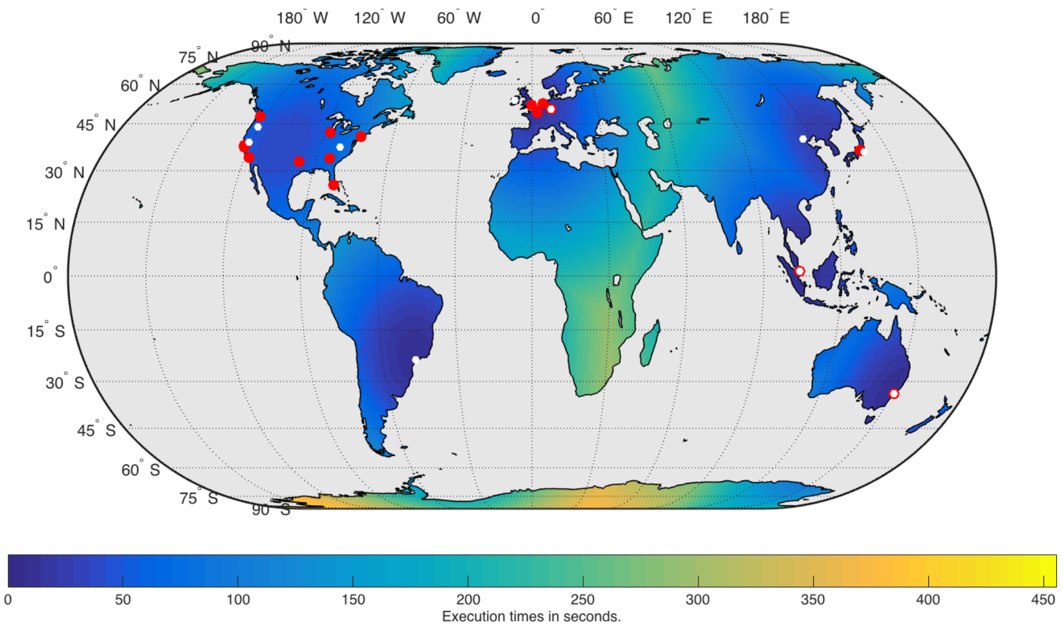


Figure 4.15: Model with fitted Gaussian functions to execution times for downloading a large file from Dropbox ($h = 1, MAPE = 2.16\%$). Red dots indicate clients used for measurement. White dots indicate data centers of Dropbox.

Drive are shown in Figure 4.16. Again better execution times can be found in Europe and USA. In comparison to Dropbox a better overall performance is trained.

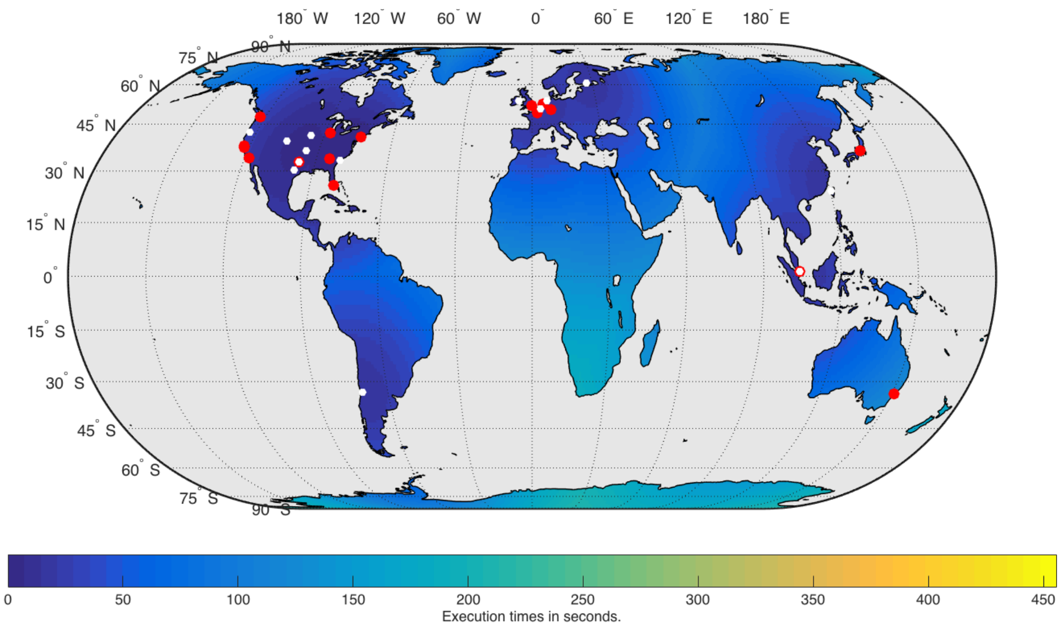


Figure 4.16: Model with fitted Gaussian functions to execution times for downloading a large file from Google Drive ($h = 1, MAPE = 0.55\%$). Red dots indicate clients used for measurement. White dots indicate data centers of Google Drive.

Microsoft provides data centers in South America and Australia as shown in Figure 4.17. Additional data centers lead to smaller execution times in certain regions. In can be seen that performance at eastern USA is higher than on the West coast.

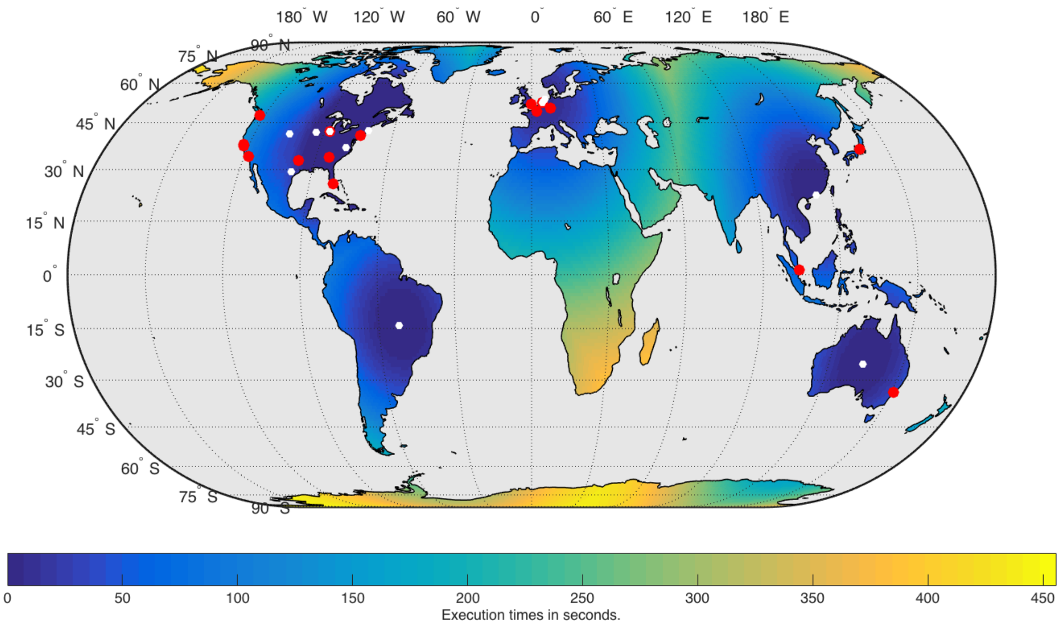


Figure 4.17: Model with fitted Gaussian functions to execution times for downloading a large file from OneDrive ($h = 1, MAPE = 8.79\%$). Red dots indicate clients used for measurement. White dots indicate data centers of OneDrive.

Learned and averaged parameters can be found in Table 4.2. Google Drive has the smallest amplitude A_i, which indicates fast operations. All models have standard deviation around 49 in both directions, which means that decrease of performance only slightly correlates with the direction in which distance increases.

Service	$mean(A_i)$	$mean(\theta)$	$mean(\sigma_\phi)$	$mean(\sigma_\lambda)$
Dropbox	88.39	-80.89	49.63	48.93
Google Drive	56.15	50.40	50.40	49.73
OneDrive	111.47	-25.05	43.11	43.52

Table 4.2: Learned parameters after fitting Gaussian functions to execution times for downloading large files during the first hour of the day ($h = 1$).

Using Gaussian curve fitting for determination of the fastest service, leads to a global classification map shown in Figure 4.18. Due to additional data centers in South America and Africa OneDrive has the best overall performance and is the fastest service except in West Asian and East Africa.

4.3.3 Evaluation of Support Vector Regression

SVM regression provides the highest degree of freedom, because it is not bound to polynomial functions or any data center location. Furthermore, evaluation was executed while neglecting artificial resampling of data points at locations of data centers, which led to poor regression performance. SVM regression led to the smallest MAPE for the entire fit for each service provider. Figure 4.19 shows a global map of execution times modeled for Dropbox. A slightly better performance is observed in Europe and USA,

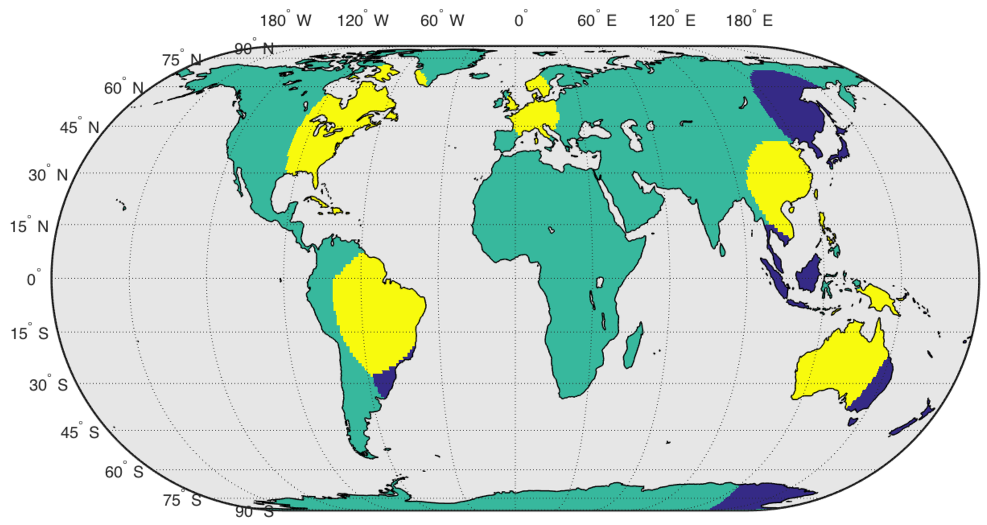


Figure 4.18: Classification of the fastest cloud storages for downloading large files ($h = 1$). Blue areas indicate Dropbox, green areas indicate Google Drive, and yellow areas indicate OneDrive.

which is reasonable due to the high density of data centers. In case of Australia and South Asia, a worse performance was modeled. In general no high global variation was learned.

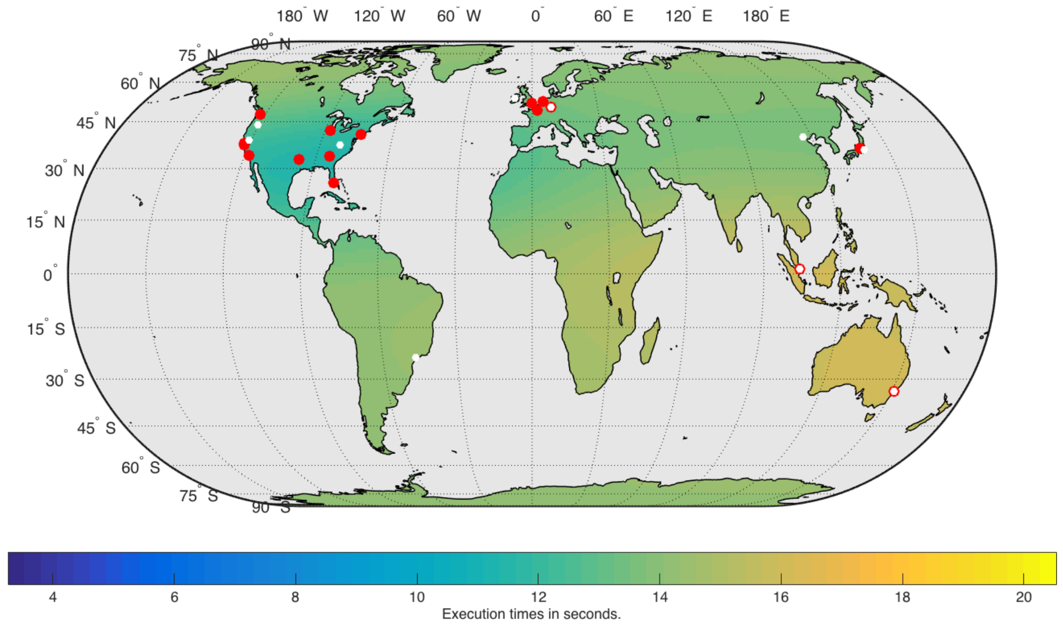


Figure 4.19: SVM regression model for execution times for downloading a large file from Dropbox ($h = 1, MAPE = 0.16\%$). Red dots indicate clients used for measurement. White dots indicate data centers of Dropbox.

The global map of Google Drive has again better performances in Europe and USA and worse execution times in Australia. The average execution time is worse than for Dropbox. In asian regions, execution times are rather constant around 18 seconds.

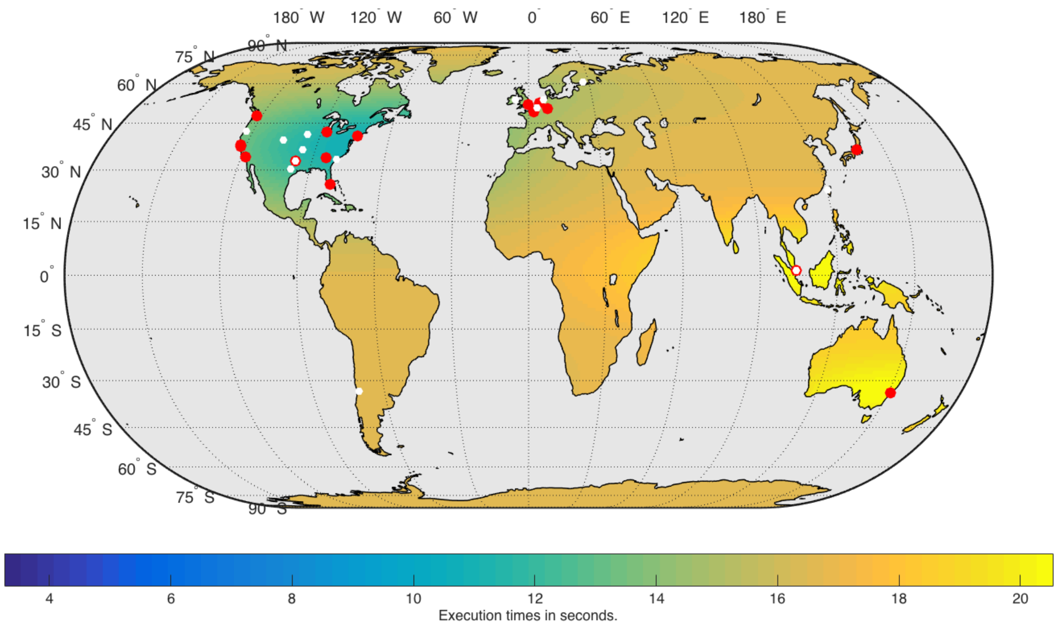


Figure 4.20: SVM regression model for execution times for downloading a large file from Google Drive ($h = 1, MAPE = 0.15\%$). Red dots indicate clients used for measurement. White dots indicate data centers of Google Drive.

OneDrive shows the best overall global performance (see Figure 4.21). This is achieved due to additional data centers in South America and Australia. The average time for uploading a large file is around 10 seconds.

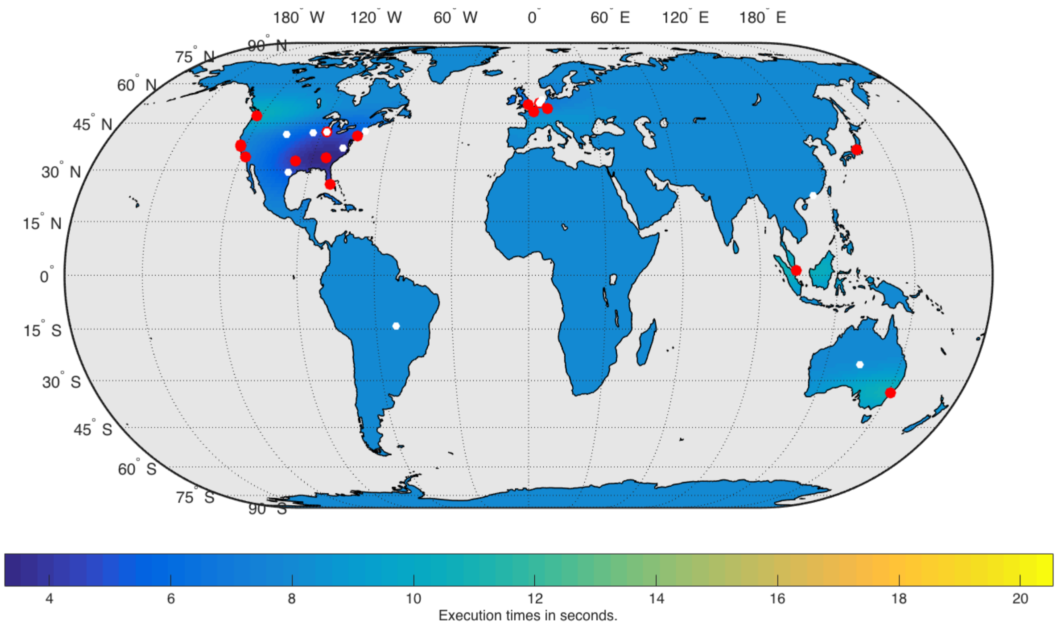


Figure 4.21: SVM regression model for execution times for downloading a large file from OneDrive ($h = 1, MAPE = 0.37\%$). Red dots indicate clients used for measurement. White dots indicate data centers of OneDrive.

Similar to other modeling techniques OneDrive has the best global map of execution time and is classified as the best service around the world.

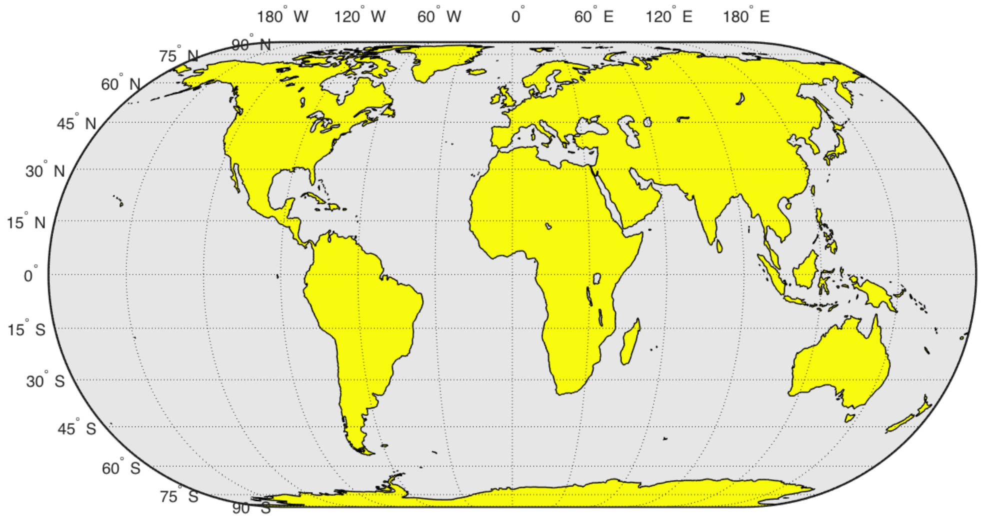


Figure 4.22: Classification of the fastest cloud storages for downloading large files ($h = 1$). Blue areas indicate Dropbox, green areas indicate Google Drive, and yellow areas indicate OneDrive.

4.3.4 Evaluation of K-NN classification

In order to determine the optimal number of neighbors k_{opt}, 24 random samples are selected out of N_d samples. The best service is determined by applying K-NN to selected samples with different number of neighbors. The best found providers are compared with the fastest provider determined by measurements and the error rate is calculated. This was done 10 times and error rates are averaged. Figure 4.23 depicts error rates for different number of neighbors. It can be seen that the lowest error rate of approximately 30% was achieved with $k = 1$ and $k = 6$. Due to faster computation time the optimal number of neighbors was chosen to be 1 ($k_{opt} = 1$). Hence, K-NN classification is done by searching the closest measurement client and selecting the fastest measured service. Figure 4.24 shows a global classification map of the best provider using K-NN classification. It can be seen that in North America the best provider is Google Drive. In other regions OneDrive is the best provider.

4.3.5 Model Comparison

The different modeling techniques were applied to measured data in order to predict the optimal provider of a certain geographic location. All models have different attributes and use different approaches to classify the service with the fastest execution time. Table 4.3 compares results of different models for uploading a large file with respect to classification error rates on test data and the mean absolute percentage error of regression models. In case of K-NN classification no MAPE is listed, because no regression model was created.

The worst error rates resulted after fitting Gaussian functions and k-nearest neighbor classification. Both approaches led to an error rate of around 50%, meaning that

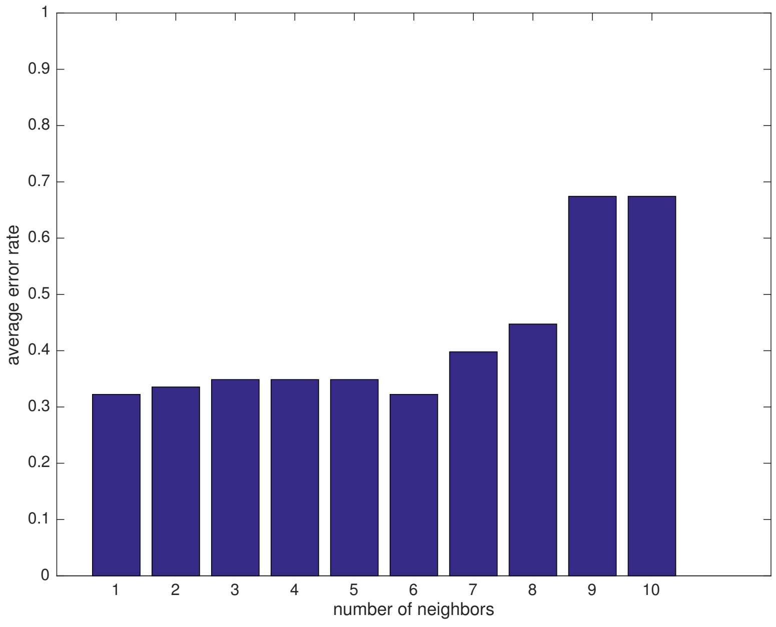


Figure 4.23: Averaged error rate for different numbers of neighbors k

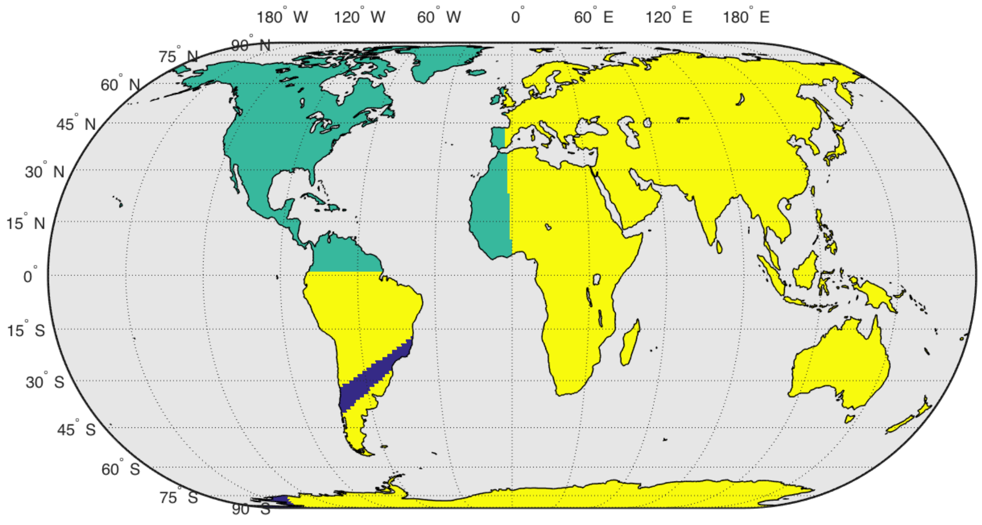


Figure 4.24: Classification of the best service for downloading a large file ($k = 1$). Blue areas indicate Dropbox, green areas indicate Google Drive, and yellow areas indicate OneDrive.

optimal services on every second location were matched wrongly. Building a global regression model by placing Gaussian functions at data center locations resulted in the highest mean absolute percentage errors. In case of OneDrive the MAPE is 8.79, meaning that on average absolute modeled execution time differ by factor of 8 from measured execution times. Additionally, the approach is computationally the most expensive one among all four techniques.

Multiple linear regression led to a rather low classification error of 15.6%. Mean absolute percentage errors between 0.31 % and 1.30% were reached, which states the second best regression model for uploading large files.

	Error rate [%]	MAPE		
		Dropbox	Google Drive	OneDrive
Multiple Linear Regression	15.57	0.31	0.40	1.30
Gaussian Curve	47.37	2.16	0.55	8.79
Support Vector Regression	11.84	0.16	0.15	0.37
K-nearest-neighbor	54.82	-	-	-

Table 4.3: Classification error rates and MAPE for different modeling techniques for downloading a large file for each service.

The smallest classification error of under 12% was achieved with support vector regression. Additionally, the smallest MAPEs were achieved using SVM for regression. On average the modeled values differ around 40% to actual measured values. The highest errors occur for the models of OneDrive, meaning that this cloud service is hard to predict.

4.3.6 Evaluating Different Filesets

In the previous section modeled execution times of uploading large files were investigated. In this section additional upload and download operations for other file types (see Table 2.1) are investigated. Table 4.4 shows classification errors and mean absolute percentage errors for all filetypes and modeling methods. In general, Gaussian curve fitting results in high classification error and MAPE. Therefore, the assumption that distances to the closest data center plays a vital role in predicting execution times does not hold.

In case of determining the optimal provider using the K-NN classification small classification errors result for upload operations. In contrast to upload operations, relative high errors occur if optimal services for download operations are classified. In can be assumed that uploads are performed to the closest data center. Due to replication of files to multiple data centers it cannot be assured that files are downloaded from the closest data center. The regression model with the smallest MAPE is the SVM regression model. For every service, operation and file type the best MAPE was achieved by SVM regression. Best classification rates were accomplished with SVM regression. Multiple linear regression states the simplest regression model. Due to the accurate model rather small mean classification errors and absolute percentage errors occurred.

In general it can be said that models for upload operations have a smaller error rate and can be modeled with higher accuracy. This is explained by the fact that upload operations are always executed on a single server. In case of download operations the corresponding file might be stored on multiple data centers. Each cloud service selects the optimal hosting data center differently, which leads to distinctive behavior. Selections are often based on different influences, e.g. data center workload

Filetype	Method	Error rate [%]	MAPE		
			Dropbox	G. Drive	OneDrive
UPLOAD ONE SMALL SIZE FILE	Multiple Linear Regression	24.56	0.72	0.79	1.23
	Gaussian Curve	80.48	5.03	105.32	8.46
	Support Vector Regression	3.29	0.13	0.14	0.16
	K-nearest-neighbours	4.17	-	-	-
UPLOAD ONE MEDIUM SIZE FILE	Multiple Linear Regression	4.17	0.54	0.42	0.72
	Gaussian Curve	64.91	1.59	23.07	3.43
	Support Vector Regression	2.19	0.15	0.13	0.13
	K-nearest-neighbours	3.07	-	-	-
UPLOAD ONE LARGE SIZE FILE	Multiple Linear Regression	9.87	0.46	0.26	1.40
	Gaussian Curve	7.46	0.33	0.13	0.91
	Support Vector Regression	6.80	0.17	0.11	0.32
	K-nearest-neighbours	7.68	-	-	-
UPLOAD ONE SPARSE FILE	Multiple Linear Regression	14.04	0.60	0.28	1.52
	Gaussian Curve	5.04	0.40	0.13	0.80
	Support Vector Regression	5.04	0.26	0.11	0.26
	K-nearest-neighbours	5.92	-	-	-
DOWNLOAD ONE SMALL SIZE FILE	Multiple Linear Regression	59.87	0.80	1.12	1.84
	Gaussian Curve	71.27	9.42	99.50	15.61
	Support Vector Regression	14.91	0.15	0.16	0.22
	K-nearest-neighbours	37.72	-	-	-
DOWNLOAD ONE MEDIUM SIZE FILE	Multiple Linear Regression	50.88	0.63	0.98	1.87
	Gaussian Curve	81.14	2.73	92.70	4.95
	Support Vector Regression	22.59	0.16	0.17	0.31
	K-nearest-neighbours	53.51	-	-	-
DOWNLOAD ONE LARGE SIZE FILE	Multiple Linear Regression	15.57	0.31	0.40	1.30
	Gaussian Curve	47.37	2.16	0.55	8.79
	Support Vector Regression	11.84	0.16	0.15	0.37
	K-nearest-neighbours	54.82	-	-	-
DOWNLOAD ONE SPARSE FILE	Multiple Linear Regression	17.54	0.30	0.44	1.50
	Gaussian Curve	48.46	1.99	0.56	7.67
	Support Vector Regression	11.84	0.15	0.15	0.38
	K-nearest-neighbours	54.82	-	-	-

Table 4.4: Error rates and mean absolute error for different modeling techniques and filetypes.

5 Conclusion

The thesis was set out to explore behavior of the three most common cloud storage services with respect to time and location and has applied several modeling techniques in order to gain insight for each service. Used modeling techniques made it possible to determine the best provider for every location on earth and each hour of the day. In addition to classification rates, computational performance is an important factor for further implementations of performed benchmarks. Execution times over the day do not vary enough to justify complex modeling techniques, which are computational expensive. Simple models, i.e. multiple linear regression or SVM regression, provide sufficient modeling capabilities for building a powerful global model. The assumption that better execution times at locations close to data centers cannot be validated. Placing Gaussian functions at data center locations did not result in good models. An increasing number of proprietary networks are spanned over the globe. Amazon for instance, has many data centers placed on the earth in combination with high speed networks, which are exclusively used by their services. Many undocumented entry points to these networks exists. As soon as a client is located close to an entry point, the distance to the next data center does not play a role anymore. In some cases the fastest service at the closest measurement location provides enough information to determine the service with the lowest execution time. Especially upload operations can be modeled, and therefore predicted, in a very thorough manner. One reason for this behavior is the fact that data is always uploaded to a single data centers. Decision making algorithms are simpler, because data is transferred from to client to a single data center. Once the file is in the cloud it is replicated to multiple data centers in order to guarantee a maximal uptime and provide backups in case of data loss. Complex algorithms are shifting data of the user between multiple data centers of a single provider in the background. At the time the user wants to download the file again at a given location, data might no be stored on the optimal data center, which leads to a higher variation of download execution times. This behavior makes is difficult to extract patterns in form of statistical models. Therefore, it states a vast challenge to predict execution times of download operations, because sophisticated replication algorithm must be considered in the model. In most of the cases a two dimensional plane is sufficient to model global execution times with a relatively small MAPE of a certain provider at a given hour of the day. This infers strictly monotonic linearly increasing or decreasing performance alongside the longitude and latitude axes. The multiple linear regression model does not allow nonlinear variations. Support vector regression states the overall best modeling technique for both regression of global execution times and classification of the most performant provider. It allows nonlinearities in the model, which are necessary to model performance with minimum error. It can be said that all cloud storage providers heavily focus on Northern America and Europe. It is justified by better performance in those regions and a higher density of data centers. For instance, Google Drive provides 9 data centers within the United States of America. Rather fast execution times were observed in East Asia, which states an additional economic region.

5.1 Future Work

Finding the best cloud storage provider at certain daytimes and geographic location can be useful in several use cases. Especially in the field of file-based collaboration performance plays a vital role. If two or more persons are working on the same file, each others changes should be transmitted as fast as possible in order to avoid conflicts. In addition, cloud storage is ofter used to have access to data in the cloud from everywhere. This leads to a situation, where a file has to be download before it is firstly used. In case of many files slow download operations decreases productivity.

Gained insights will be used in a commercial product called CrossCloud, which aggregates many storages into one large pool of data which is constantly synchronized with a local folder. By default, CrossCloud decides on its own where a certain file or folder is uploaded to. This is currently done by selecting the storage with the least used storage in order to guarantee equal distribution of used storages. In the near future, the behavior of the user with respect to file handling is learned, which will make it possible to predict usage of certain files at specific location, days, and daytimes. Once this prediction is combined with knowledge gained in this thesis, it is possible to store file on storages in such a way that the optimal performances for next file usage can be guaranteed. This increases productivity and optimizes user experience in case of file-based collaboration. In addition to data collected from global distributed measurement server, each synchronization action of the user (i.e. upload and download) can be tracked. In combination with the location of the user it can be used to iteratively improve global performance models, which makes predictions more accurate. In the near future, benchmarks will be extended to support additional cloud storage provider, such as Box.com or any WebDAV storage.

6 Appendix

6.1 Docker File

```
FROM jinnerbichler/cloudmapbase
COPY target/cloudmap.jar /cloudmap/
COPY init/prefs.db /cloudmap/
```

Listing 6.1: Docker file for creating Docker image with benchmark application.

6.2 System Administration and Deployment Script

```
from fabric.api import *
from fabric.network import ssh

# define server
server = {}

# digital ocean
server['45.55.74.113'] = 'CloudMapNewYork'
server['188.166.32.70'] = 'CloudMapAmsterdam'
server['104.236.180.35'] = 'CloudMapSanFrancisco'
server['128.199.254.116'] = 'CloudMapSingapore'
server['178.62.87.245'] = 'CloudMapLondon'
server['46.101.156.131'] = 'CloudMapFrankfurt'

# vultr
server['45.32.246.252'] = 'CloudMapSydneyVultr'
server['45.32.236.11'] = 'CloudMapAmsterdamVultr'
server['104.238.158.30'] = 'CloudMapFrankfurtVultr'
server['104.238.170.161'] = 'CloudMapLondonVultr'
server['104.238.191.168'] = 'CloudMapParisVultr'
server['108.61.215.249'] = 'CloudMapAtlantaVultr'
server['107.191.51.206'] = 'CloudMapChicagoVultr'
server['107.191.45.63'] = 'CloudMapDallasVultr'
server['45.32.69.67'] = 'CloudMapLosAngelesVultr'
server['45.63.104.28'] = 'CloudMapMiamiVultr'
server['45.63.14.129'] = 'CloudMapNewYorkVultr'
server['104.238.156.184'] = 'CloudMapSeattleVultr'
server['104.207.150.153'] = 'CloudMapSiliconValleyVultr'
server['45.63.120.118'] = 'CloudMapJapanVultr'

deploy_counter = 1;

# set fabric config
env.hosts = server.keys()
env.user = "root"

INIT_DIR = "init"
TARGET_DIR = "target"
JAR_NAME = "cloudmap.jar"
PREFS_FILE = "prefs.db"
CONTAINER_NAME = "cloudmap_container"

def deploy():

    build()
    deploy_docker()

    host = env.host
    if host is '54.94.173.50':
        host = 'ubuntu@54.94.173.50'

    print("######################################################################################")
    print("################ DEPLOYING TO %s (%s) - Deploying %d/%d" % (server[host], host,
                                                                deploy_counter, len(server)))
    print("################################################################################")

    # set timezone
    run('timedatectl set-timezone UTC')

    deploy_container()

    global deploy_counter
    deploy_counter += 1

@runs_once
def build():
    with settings(warn_only=True):
        local("rm -rf %s" % TARGET_DIR)
    local("mvn -q clean compile assembly:single")

@runs_once
def deploy_docker():
    # start docker locally
    run('eval "$(docker-machine env default)"')
    create_docker()
    push_docker()

@runs_once
def create_docker():
    local("docker build -t jinnerbichler/cloudmap:latest .")

@runs_once
def push_docker():
    local("docker push jinnerbichler/cloudmap")

def deploy_container():
    # stop and remove all previous container
    with settings(warn_only=True):
        run("docker stop $(docker ps -a -q)")
    with settings(warn_only=True):
        run("docker rm $(docker ps -a -q)")

    run('docker pull jinnerbichler/cloudmap')
    run('docker run -d --name %s jinnerbichler/cloudmap bash -c "cd cloudmap && java -jar %s"' %
          \
              (CONTAINER_NAME, JAR_NAME))

    # sanity check
    run('date +"%T"')
    run("docker ps")
    run("docker top %s" % CONTAINER_NAME)

def auth():
    local("mvn clean compile assembly:single")
    local("mkdir -p " + INIT_DIR)
    local("cp " + TARGET_DIR + "/" + JAR_NAME + " " + INIT_DIR)
    local("echo 'Authenticating Google Drive'")
    local("cd " + INIT_DIR + "&& java -jar " + JAR_NAME + " gd_auth")
    local("echo 'Authenticating One Drive'")
    local("cd " + INIT_DIR + "&& java -jar " + JAR_NAME + " od_auth")

def get_logs():
    with settings(warn_only=True):
        log_dir = "remote_logs"
        local("mkdir -p %s" % log_dir)
        log = run("docker exec cloudmap_container tail -n 1000 /cloudmap/logs/cloudmap.log")

        log_path = log_dir + "/" + server[env.host] + ".log"
        with open(log_path, "w") as text_file:
            text_file.write(log)
            print("stored log of %s to %s" % (server[env.host], log_path))

def install_docker_if_neccessary():
    found_string = run("echo $(type docker)")
    if "not" in found_string:
        run("sudo apt-get update")
        run("sudo apt-get install curl")
        run("sudo curl -sSL https://get.docker.com/ | sh")
```

Listing 6.2: Fabric configuration file for administration and deploying software updates.

6.3 Class Diagram of Benchmark Client

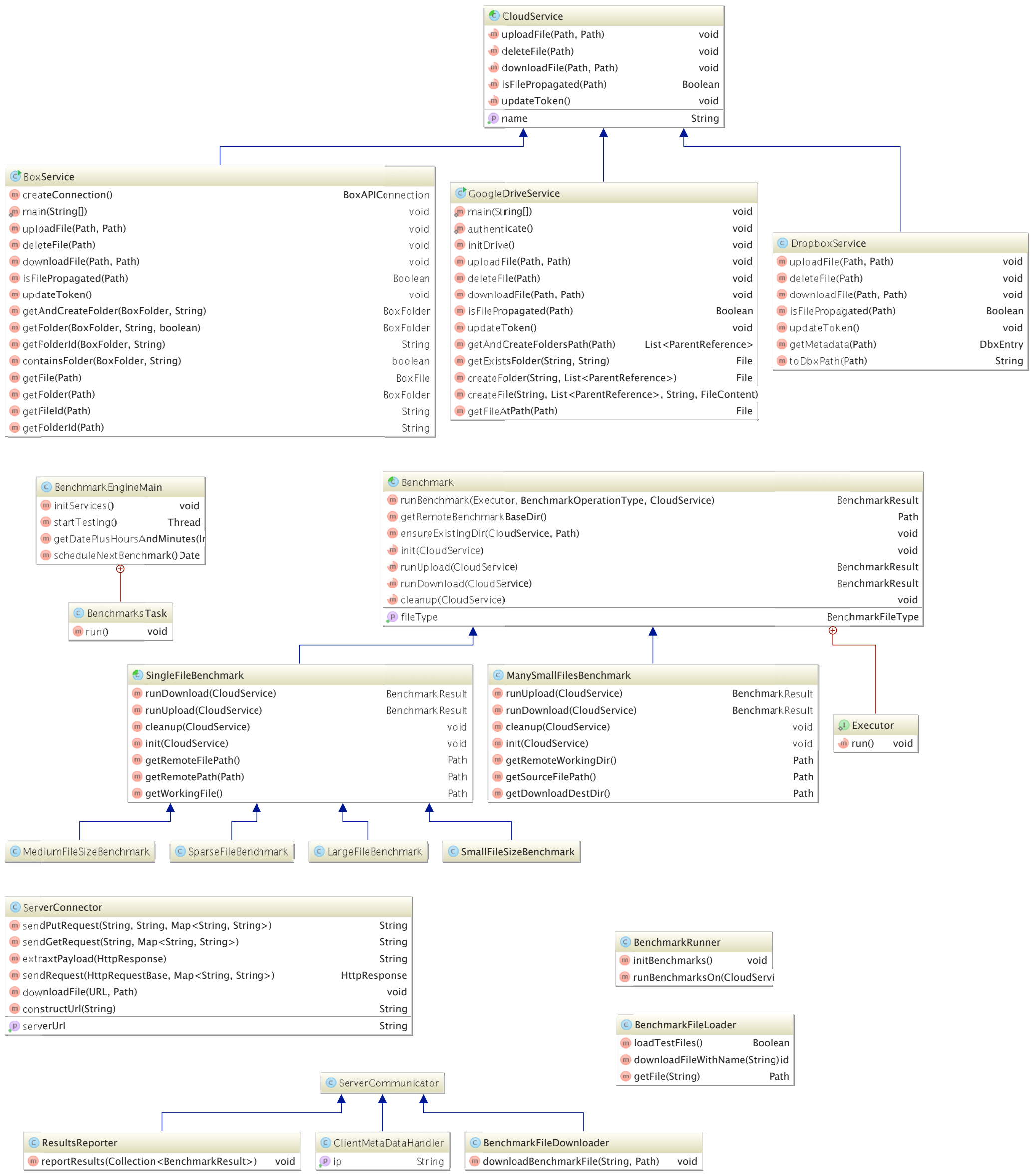

CloudService
uploadFile(Path, Path) void
deleteFile(Path) void
downloadFile(Path, Path) void
isFilePropagated(Path) Boolean
updateToken() void
name String
BoxService
createConnection() BoxAPIConnection
main(String[]) void
uploadFile(Path, Path) void
deleteFile(Path) void
downloadFile(Path, Path) void
isFilePropagated(Path) Boolean
updateToken() void
getAndCreateFolder(BoxFolder, String) BoxFolder
getFolder(BoxFolder, String, boolean) BoxFolder
getFolderId(BoxFolder, String) String
containsFolder(BoxFolder, String) boolean
getFile(Path) BoxFile
getFolder(Path) BoxFolder
getFileId(Path) String
getFolderId(Path) String
GoogleDriveService
main(String[]) void
authenticate() void
initDrive() void
uploadFile(Path, Path) void
deleteFile(Path) void
downloadFile(Path, Path) void
isFilePropagated(Path) Boolean
updateToken() void
getAndCreateFoldersPath(Path) List<ParentReference>
getExistsFolder(String, String) File
createFolder(String, List<ParentReference>) File
createFile(String, List<ParentReference>, String, FileContent)
getFileAtPath(Path) File
DropboxService
uploadFile(Path, Path) void
deleteFile(Path) void
downloadFile(Path, Path) void
isFilePropagated(Path) Boolean
updateToken() void
getMetadata(Path) DbxEntry
toDbxPath(Path) String
BenchmarkEngineMain
initServices() void
startTesting() Thread
getDatePlusHoursAndMinutes(Ir
scheduleNextBenchmark() Date
BenchmarksTask
run() void
Benchmark
runBenchmark(Executor, BenchmarkOperationType, CloudService) BenchmarkResult
getRemoteBenchmarkBaseDir() Path
ensureExistingDir(CloudService, Path) void
init(CloudService) void
runUpload(CloudService) BenchmarkResult
runDownload(CloudService) BenchmarkResult
cleanup(CloudService) void
fileType BenchmarkFileType
SingleFileBenchmark
runDownload(CloudService) BenchmarkResult
runUpload(CloudService) BenchmarkResult
cleanup(CloudService) void
init(CloudService) void
getRemoteFilePath() Path
getRemotePath(Path) Path
getWorkingFile() Path
ManySmallFilesBenchmark
runUpload(CloudService) BenchmarkResult
runDownload(CloudService) BenchmarkResult
cleanup(CloudService) void
init(CloudService) void
getRemoteWorkingDir() Path
getSourceFilePath() Path
getDownloadDestDir() Path
Executor
run() void
MediumFileSizeBenchmark
SparseFileBenchmark
LargeFileBenchmark
SmallFileSizeBenchmark
ServerConnector
sendPutRequest(String, String, Map<String, String>) String
sendGetRequest(String, Map<String, String>) String
extraxtPayload(HttpResponse) String
sendRequest(HttpRequestBase, Map<String, String>) HttpResponse
downloadFile(URL, Path) void
constructUrl(String) String
serverUrl String
BenchmarkRunner
initBenchmarks() void
runBenchmarksOn(CloudServi
BenchmarkFileLoader
loadTestFiles() Boolean
downloadFileWithName(String) id
getFile(String) Path
ServerCommunicator
ResultsReporter
reportResults(Collection<BenchmarkResult>) void
ClientMetaDataHandler
ip String
BenchmarkFileDownloader
downloadBenchmarkFile(String, Path) void

Figure 6.1: Class diagram of benchmark client.

6.4 Class Diagram of Main Server

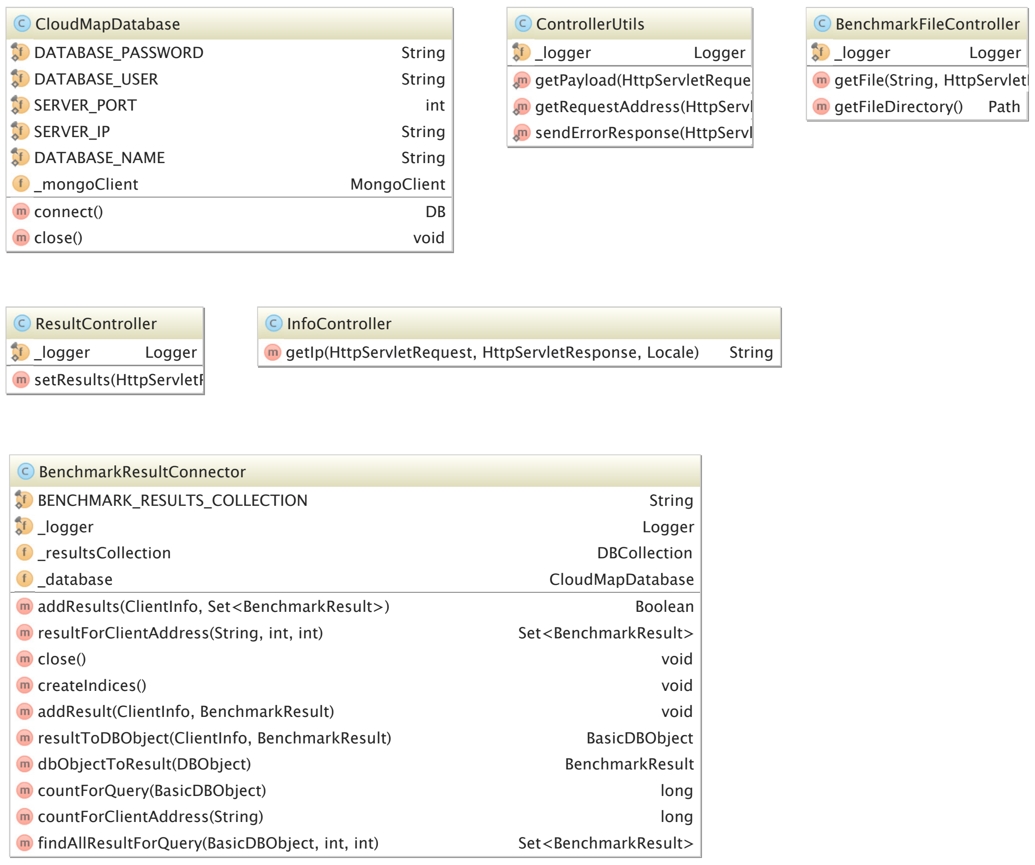

CloudMapDatabase
DATABASE_PASSWORD String
DATABASE_USER String
SERVER_PORT int
SERVER_IP String
DATABASE_NAME String
_mongoClient MongoClient
connect() DB
close() void
ControllerUtils
_logger Logger
getPayload(HttpServletReque
getRequestAddress(HttpServl
sendErrorResponse(HttpServl
BenchmarkFileController
_logger Logger
getFile(String, HttpServletI
getFileDirectory() Path
ResultController
_logger Logger
setResults(HttpServletF
InfoController
getIp(HttpServletRequest, HttpServletResponse, Locale) String
BenchmarkResultConnector
BENCHMARK_RESULTS_COLLECTION String
_logger Logger
_resultsCollection DBCollection
_database CloudMapDatabase
addResults(ClientInfo, Set<BenchmarkResult>) Boolean
resultForClientAddress(String, int, int) Set<BenchmarkResult>
close() void
createIndices() void
addResult(ClientInfo, BenchmarkResult) void
resultToDBObject(ClientInfo, BenchmarkResult) BasicDBObject
dbObjectToResult(DBObject) BenchmarkResult
countForQuery(BasicDBObject) long
countForClientAddress(String) long
findAllResultForQuery(BasicDBObject, int, int) Set<BenchmarkResult>

Figure 6.2: Class diagram of main server.

Bibliography

[1] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[2] I. Cisco Systems. (2015) Cisco global cloud index: Forecast and methodology, 2014–2019 white paper. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.html

[3] S. Rhea, C. Wells, P. Eaton, D. Geels, B. Zhao, H. Weatherspoon, and J. Kubiatowicz, "Maintenance-free global data storage," *Internet Computing, IEEE*, vol. 5, no. 5, pp. 40–49, Sep 2001.

[4] I. Drago, E. Bocchi, M. Mellia, H. Slatman, and A. Pras, "Benchmarking personal cloud storage," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13. New York, NY, USA: ACM, 2013, pp. 205–212. [Online]. Available: http://doi.acm.org/10.1145/2504730.2504762

[5] W. Hu, T. Yang, and J. N. Matthews, "The good, the bad and the ugly of consumer cloud storage," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 3, pp. 110–115, Aug. 2010. [Online]. Available: http://doi.acm.org/10.1145/1842733.1842751

[6] I. Drago, M. Mellia, M. M. Munafo, A. Sperotto, R. Sadre, and A. Pras, "Inside dropbox: Understanding personal cloud storage services," in *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, ser. IMC '12. New York, NY, USA: ACM, 2012, pp. 481–494. [Online]. Available: http://doi.acm.org/10.1145/2398776.2398827

[7] A. Bergen, Y. Coady, and R. McGeer, "Client bandwidth: The forgotten metric of online storage providers," in *Communications, Computers and Signal Processing (PacRim), 2011 IEEE Pacific Rim Conference on*, Aug 2011, pp. 543–548.

[8] T. H. Gareth James, Daniela Witten and R. Tibshirani, *An Introduction to Statistical Learning with Applications in R*. Springer, 2014.

[9] M. Corporation. (2016) Microsoft data centers. [Online]. Available: https://www.microsoft.com/datacenters/

[10] V. H. LLC. (2016) Ssd vps servers, cloud servers and cloud. [Online]. Available: https://www.vultr.com/

[11] D. Inc. (2016) Simple cloud infrastructure for developers. [Online]. Available: https://www.digitalocean.com/

[12] I. Amazon Web Services. (2016) Amazon webs services (aws) - cloud computing service. [Online]. Available: https://aws.amazon.com/

[13] D. Inc. (2015) Docker - build, ship and run. [Online]. Available: https://www.docker.com/

[14] J. Forcier. (2016) Fabric. [Online]. Available: http://www.fabfile.org/

[15] IETF. (2016) Oauth 2.0 rfc6749. [Online]. Available: http://tools.ietf.org/html/rfc6749

[16] P. Software. (2016) Spring framework. [Online]. Available: http://projects.spring.io/spring-framework/

[17] I. MongoDB. (2016) Mongodb for giant ideas. [Online]. Available: https://www.mongodb.org/

[18] (2016) Sqlite. [Online]. Available: https://www.sqlite.org/

[19] A. Fiori. (2016) freegeoip.net. [Online]. Available: https://freegeoip.net/

[20] V. N. Vapnik, *The Nature of Statistical Learning Theory.* New York, NY, USA: Springer-Verlag New York, Inc., 1995.

[21] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, “Support vector regression machines,” in *Advances in Neural Information Processing Systems 9*, M. Mozer, M. Jordan, and T. Petsche, Eds. MIT Press, 1997, pp. 155–161. [Online]. Available: http://papers.nips.cc/paper/1238-support-vector-regression-machines.pdf