Improving data processing performance with Spark

Utilizing and optimizing Apache Spark to improve the data processing performance of the WifiSpeed.

Aug 30, 2022

Tacettin Emre Bök

28091

Internship Start-End Dates: 30.06.22-30.08.22

Company: QuantWifi

Supervisor: Ali Sayinta

Submission Date:

Faculty of Engineering and Natural Sciences

Sabancı University

Computer Science and Engineering

# ABSTRACT

QuantWifi is a company that provides Wi-Fi performance solutions to by measuring Wi-Fi performance between gateway and associated Wi-Fi clients. The solution of QuantWifi makes decisions on link quality based on performance data in correlation to Wi-Fi QoS metrics. To do this, QuantWifi collects a data from routers and processes them to apply machine learning algorithms. This internship project aims to optimize the data processing jobs of QuantWifi’s project “WifiSpeed” by utilizing suitable methods for Apache Spark. Firstly, related literature about this optimization subject is reviewed. By doing this, a list of methods and techniques to do data processing faster found. In the next step, these methods are tested to see if they are appropriate to apply on real jobs of WifiSpeed. After testing these results and analysing them, the appropriate ones are applied to the real jobs of WifiSpeed. Some of these methods are data filtering, caching, right sizing of partitions, using broadcast joins etc. In the analysis of this project, it can be said that the methods that are found was beneficial to use and they increased the speed and efficiency of WifiSpeed’s data processing. But it is concluded and that there is a lot of way to go in terms of Apache Spark and using it to optimize data processing of WifiSpeed. In the light of the positive findings and improvements in this project, it is recommended for the company to enhance their interest about Spark as it is very promising framework to improve their data processing.

Table of Contents

[ABSTRACT 2](#_Toc114871626)

[1.INTRODUCTION 5](#_Toc114871627)

[2. COMPANY INFORMATION 6](#_Toc114871628)

[2.1 Contact 6](#_Toc114871629)

[2.2 Company Information 6](#_Toc114871630)

[2.3 Organization 7](#_Toc114871631)

[3. PROJECT BACKGROUND 8](#_Toc114871632)

[3.1 Department Information 8](#_Toc114871633)

[3.2 Initial State of the WifiSpeed’s Data Processing 8](#_Toc114871634)

[3.3 Motivation and Problem Definition of the Project 9](#_Toc114871635)

[3.4 Related Literature 10](#_Toc114871636)

[3.4.1 What is Spark? 10](#_Toc114871637)

[3.4.2 In memory processing and parallelism 10](#_Toc114871638)

[3.4.3 How to Optimize Spark Jobs? 11](#_Toc114871639)

[4.INTERNSHIP PROJECT 13](#_Toc114871640)

[4.1 Project Objective 13](#_Toc114871641)

[4.2 My Responsibilities 14](#_Toc114871642)

[4.3 Methodology and Tools 14](#_Toc114871643)

[4.4 Expected Outcome and Deliverables 15](#_Toc114871644)

[4.5 Details 16](#_Toc114871645)

[4.5.1 Understanding the Spark UI 16](#_Toc114871646)

[4.5.2 Data Filtering (Lazy Loading) 18](#_Toc114871647)

[4.5.3 Managing the Number and the Size of Partitions 18](#_Toc114871648)

[4.5.4 Caching 21](#_Toc114871649)

[4.5.5 Broadcast Join 22](#_Toc114871650)

[4.6 Results 22](#_Toc114871651)

[5.INTERNSHIP EXPERIENCE 23](#_Toc114871652)

[5.1 Learning 23](#_Toc114871653)

[5.2 Relation to Undergraduate Education 24](#_Toc114871654)

[5.3 Difficulties 24](#_Toc114871655)

[5.3.1 Understanding a New Framework 24](#_Toc114871656)

[5.3.2 Adapting to an Ongoing Project 25](#_Toc114871657)

[5.4 A Typical Day 25](#_Toc114871658)

[6. CONCLUSIONS 26](#_Toc114871659)

[7. RECOMMENDATIONS 27](#_Toc114871660)

[REFERENCES 28](#_Toc114871661)

[© TemplateLab.com](https://templatelab.com/)

.

# 1.INTRODUCTION

The WifiSpeed project of the QuantWifi identifies and resolves Wi-Fi issues correctly and efficiently by making decisions according to measurements that are made by their software continuously about Wi-Fi performance of a router. To elaborate more, the agent collects the device’s performance data, and this data is stored. With the help of this big data, solving Wi-Fi related issues becomes easier. As it can be guessed, fast and efficient data processing is a particularly important part of WifiSpeed as the amount of data obtained from routers is more than ten billion rows and processing this data faster will provide better user experience to the customers of WifiSpeed. To achieve this, I optimized the already existing algorithms and queries about data processing by using Apache Spark which is a distributed processing system used for big data processing. Utilizing Spark enhanced the data processing performance because it offers parallelism, in-memory processing, fault tolerance, lazy evaluation etc. In the upcoming chapters of this report, firstly, background of the project such as related literature and initial state of the data processing of WifiSpeed will be explained. After that, my project, which is utilizing and optimizing data processing of WifiSpeed with Apache Spark, will be elucidated in detail. Lastly, my individual experience in this internship project will be analysed and discussed.

# 2. COMPANY INFORMATION

## 2.1 Contact

QuantWifi Bilişim ve Ticaret Anonim Şirketi

Sanayi Mah. Teknopark Blv. NO:1/4C İç Kapı No: Z08 PENDİK/İSTANBUL TÜRKİYE

+905324202700

[Web Page](https://quantwifi.com)

## 2.2 Company Information

QuantWifi founded in 2017 by Ali Sayinta who is still the CEO of the company. The company focuses into consumer’s home networks and offers solutions to telecommunication industry with their product: WifiSpeed. In 2021, Turk Telekom, which is one of the leading telecommunication companies in Turkey, became a customer of WifiSpeed. In addition, Turk Telekom Ventures invented to QuantWifi. Moreover, in 2020, QuantWifi and Gamgee announced their strategic partnership. Gamgee is a company from Netherlands, and they provide hardware independent Wi-Fi solutions. Currently, the company has one office in Teknopark İstanbul and six employees (eight with interns).

## 2.3 Organization

QuantWifi’s main project WifiSpeed has four departments. These are: agent development, data warehouse, data science and Web UI. In agent development, embedded software engineers are working to develop and improve WifiSpeed’s agent. In data warehouse department, company stores and processes their data. Data scientists are analysing and applying machine learning algorithms to the data that is in the data warehouse. Lastly, In Web UI department, Caner Köroğlu, who is a full stack software engineer, creates useful and easy-to-use interfaces for the company’s customers. Figure 1 shows the structure of the company.

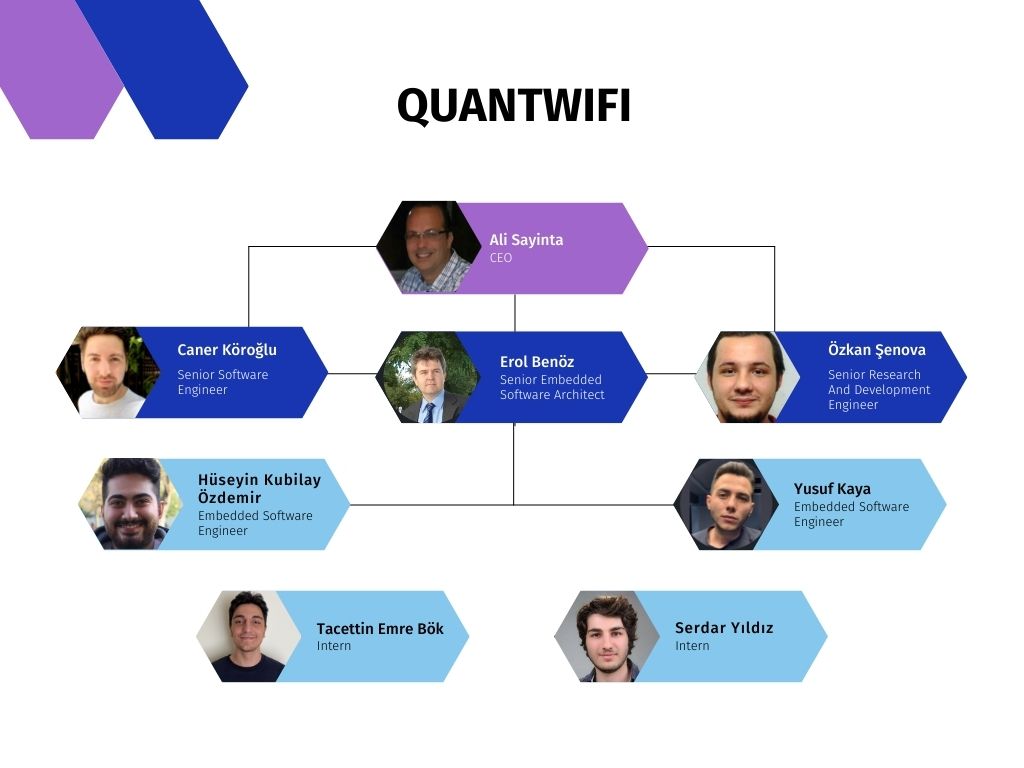


Figure 1

# 3. PROJECT BACKGROUND

## 3.1 Department Information

As it mentioned before, QuantWifi is still a growing company considered as a start-up. So, the departments are not very clear at this stage of the company’s journey. The company have department for data warehouse and data analytics, but this project was not included in any of these. For this project, I worked very closely with Mr. Ali ([ali.sayinta@quantwifi.com](mailto:ali.sayinta@quantwifi.com)) who is the CEO of the company. Although he is the CEO, he was more than interested to all the people working on the project WifiSpeed. He was curious about Spark, and he was willing to implement its parallelism manner to their processes with data. So, we worked together to understand, implement Spark in order to optimize the data processing jobs of the WifiSpeed. Other than Mr. Ali, I worked with Mr. Özkan (ozkan.senova@quantwifi.com) who is a Senior Research and Development Engineer time to time because we were trying to implement a cluster with multiple machines for Spark. He initiated the nodes from AWS and taught me how to use and monitor them.

## 3.2 Initial State of the WifiSpeed’s Data Processing

Currently, company uses an open-source column-oriented Database Management System called ClickHouse to store their data. In addition, this DBMS has an engine to execute SQL queries and process the data. ClickHouse executes these queries surprisingly fast because it knows the structure and semantics of the data. Other than that, the analytics jobs are handled in Python. Most of these Python processes are related to machine learning part of the project and some of them are for driving company’s regular reports (hourly, daily). At the beginning, some of these codes were written using PySpark library which is an interface for Apache Spark in Python. But the algorithms and processes in those codes are not functioning as desired. That is because these codes were written in a sequential logic, and it is expected that Spark will handle the parallelization and make the jobs faster, but this is not the case. There is a big difference between processing the data with regular Python (pandas, NumPy etc.) and using Spark. To implement Spark, the logic and functioning of it should be learned very well. On the other hand, other machine learning algorithms were working very well as they are implemented with regular Python. Company plans to implement Spark to their regular hourly, daily report processes and if this results in success, they will expand this project’s range and implement Spark to the WifiSpeed’s machine learning algorithms.

## 3.3 Motivation and Problem Definition of the Project

As it discussed before, data processing and machine learning are very crucial aspects of WifiSpeed. The data of 10-15 million routers’ performance is retrieved every five minutes and this big data is used to provide better customer service. In the current state of the WifiSpeed’s data processing, codes are taking a lot of time to execute, and the resources are not used efficiently, this is mainly stems from the size of the data and the absence of a distributed data processing system. Regarding these symptoms, company sees a possibility of improvement in an open-source distributed processing system called Apache Spark. In a big data situation like this, Apache Spark, which is an open-source, distributed processing system used for big data workload, comes very handy as it is one of the most efficient big data processing tools today. In addition, big companies like Amazon, Netflix, Yahoo and eBay uses Apache Spark, this is also increasing the reliability of the tool. To conclude, current problem of the WifiSpeed’s data processing is that most of the data processing jobs are very slow and inefficient in terms of resources. This is inevitable in the current state of the WifiSpeed data processing because the data is big and trying to process this data locally on one computer will take a lot of time and resources.

## 3.4 Related Literature

Spark is in demand and very popular as a cause of the rise of big data. Because of this, there are a lot of sources both academic and industrial. Resources that are benefited in this project will be indicated in sub-headings.

### 3.4.1 What is Spark?

According to Spark’s official website, Spark is a distributed analytics engine for large scale data processing. In addition, as Salman Salloum stated, Spark is a rapidly evolving open-source project, and it has well-designed and easy to use APIs in Scala, Java, Python and R. Moreover, it has additional tools like Spark SQL for SQL, MLlib for machine learning, GraphX for graph processing, and structured streaming for stream processing. In more detail, Spark is a distributed cluster computing framework which offers advanced in-memory programming and parallelism with its upper-level libraries for scalable big data processing and machine learning.

### 3.4.2 In memory processing and parallelism

Graphical user interface, diagram, application

Description automatically generatedIt is very important to understand Spark’s in memory processing and parallelism before trying to work with Spark. Below in Figure 2, you can see the architecture of Spark.

Figure 2

According to Lei Gu and Huan Li, Spark is designed to work with clusters consisting multiple machines. Furthermore, Alexey Grishchenko explains that one of these machines is “Master” which has Driver Program in it, this node also called Driver Node. Other nodes are called “Worker Node” and they have executors inside. Basically, the idea is that the data is partitioned and distributed among workers and operations on this data is occurring in parallel and in memory of these nodes. For example, let’s say there is a data with 100 rows, Spark partitions this data into 4 worker nodes, each node has 25 rows. When count is called from master node on this dataset, each worker counts the number of rows in their partition and these numbers are summed up by “Cluster Manager” and the result returns to the master node. This was a very basic example about how parallelism works in Spark. Spark is 100 times faster in big data analytics than traditional approaches because of these two principles that it provides.

### 3.4.3 How to Optimize Spark Jobs?

As Salloum explains, Spark has two types of processes to perform on datasets, these are transformations and actions. Spark jobs are created when an action is called. This procedure is called “Lazy Evaluation” in Spark. Lazy evaluation means that Spark does not trigger a job when series of transformations are called on an object. All the transformations called on the object is serialized in DAG (Directed Acyclic Graph) and all of them are done when a specific action is called on the object. When it comes to optimizing, it is known that there is no upper limit. In this project, some practical techniques are used to make Spark work faster and efficiently. The start point of this optimization project was understanding the Spark User Interface. Even though it sounds like an easy task, it is a very important part of understanding Spark and optimizing it. Spark UI has jobs, stages, storage, environment, executors and SQL tabs. A book called Learning Spark by Jules S. Damji and Matei Zaharia (developer of Spark) helped a lot about understanding the Spark UI and other fundamentals of Spark. All these tabs are very useful while optimizing Spark jobs. In more practical optimizing, Daniel Tomes was the source that we benefited a lot. Trying to understand and apply his principles and techniques, we were able to see serious results in the project. Descriptive names of some of these techniques are Data Filtering (Lazy loading), Right sizing (partitions and shuffles), Dynamically changing shuffle partitions and normal partitions, join optimization, avoiding UDFs (user defined functions). Other than Daniel Tomes, Simon Grah’s article which named “6 recommendations for optimizing a Spark job” helped a lot with techniques such as maximizing parallelism, broadcast join, caching, observing executors’ memories. Also, we found some additional configuration settings like Spark Adaptive Query Execution (AQE) which showed great results.

# 4.INTERNSHIP PROJECT

## 4.1 Project Objective

As mentioned earlier, the objective of this project is making the WifiSpeed’s data processing jobs faster and more efficient by utilizing Apache Spark. For the end results, company expects faster algorithms and sufficient use of the Spark framework. For this purpose, different techniques and methods are found from variety of sources, and they are implemented. Scope of this project only includes optimizing existing jobs and utilizing Apache Spark. There will not be any new jobs created in this project. Sometimes, structure of the code should be changed in order to achieve optimization, but this restructuring process will not be considered as a new code. To achieve faster data processing jobs, the most crucial part is understanding and correctly using the Spark’s ability to execute multiple tasks concurrently at the same time. Number of cores in the cluster of the Spark application determines how many concurrent tasks can we execute at the same time. In other words, each core can do one task independently from each other. As it can be guessed, if it can be applied properly, Spark will make jobs faster than it could be imagined. So, this project will address the slow data processing problem with Spark’s parallelism. Distributing the workload of a job to multiple cores will make the data processing faster and more efficient.

## 4.2 My Responsibilities

In this project, we worked closely with Mr. Ali who is the CEO of the QuantWifi. He was also working with the data analysts and engineers to come up with algorithms to process the data faster. Also, they were working on machine learning too. My responsibility in this project was optimizing the jobs that created by Mr. Ali and the data team. In detail, first responsibility of mine was to do research about Apache Spark and inform Mr. Ali about my recent findings every day. After that, we met with Mr. Ali at least twice a week and we considered the new findings. Also, Mr. Ali was working on Spark. So, these weekly meetings of us were an information exchange between us. After I did my research about Spark completely, my next and last responsibility was to work more closely with Mr. Ali and apply the techniques and methods that we found with him. In order to work on the actual data and implement the things that I found, I had to work with Mr. Ali on his computer because the data of Turk Telekom was confidential and only accessible with special VPN of them, and this VPN access was not provided to interns.

## 4.3 Methodology and Tools

Methodology of this project is to make data processing jobs distributed and execute them in parallel. To achieve this properly, one of the methods that is used is adjusting the sizing of partitions on each node of the cluster. This adjustment is crucial in a distributed processing system like Spark because the size of the partition should not be too much for a core as this will create less partitions (as the size of the data’s partitions increases, number of partitions will be less) and this will result in slow processing and longer execution time. Also, the size of the partition should not be too low because this will result in too many partitions and the number of partitions should not be more than the number of cores. Ideally, number of partitions should be 2-3 times more than the number of cores. Another method that is used is caching. Caching means storing a data frame in memory. Even though Spark is an in-memory data processing engine, the data is not always stored in memory. When a data frame is used repeatedly in the same job, it needs to be cached as retrieving data from memory is much faster than retrieving data from disk. This method is simple but very effective. Appropriate caching led to much faster jobs in this project. Some of the methods are listed and shortly explained above, more detailed explanation of all optimization methods and techniques will be provided in the details part.

## 4.4 Expected Outcome and Deliverables

The objective of this project is to execute the WifiSpeed’s data processing jobs faster and more efficient by using Spark. As an outcome, it is expected to see the improvement in the efficiency and the speed of the WifiSpeed’s data processing jobs. To achieve this, it is planned to use methods that are found by doing detailed research about the subject. Some of these methods are relatively simple to implement, these methods are mostly about the configuration settings of Spark. Some of these simple methods are enabling AQE (Adaptive Query Execution), properly allocating resources, data filtering, caching etc. On the other hand, some of the methods are hard to implement and apply, some of these hard methods are right sizing of the partitions, bucketing and restructuring the codes in a way that it can be executed in parallel. Using all the appliable methods, it is desired to optimize the data processing as much as it can be optimized. Result of the project will be easily seen from the speed improvement of jobs. It is planned to deliver both speed and efficient resource usage.

## 4.5 Details

In this section of the report, the Spark methods and techniques that are used to optimize data processing jobs will be explained in detail. At the initial stage of WifiSpeed’s data processing works, Spark is rarely used, and the company did not put effort in improving their data processing by using Spark.

### 4.5.1 Understanding the Spark UI

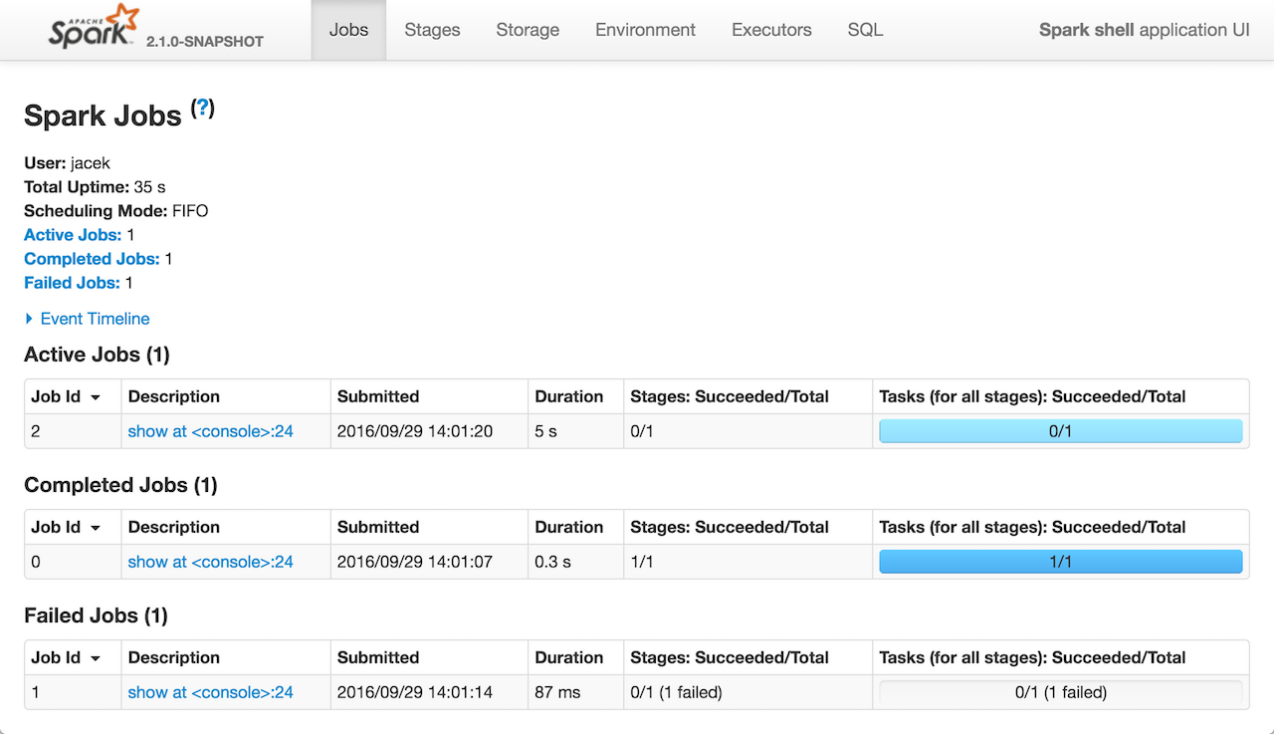
 Although understanding the Spark UI is not a direct way to optimize Spark jobs, it is very essential for optimizing. If Apache Spark is correctly working on the system, Spark UI can be accessed from [***http://localhost:4040/***](http://localhost:4040/)***.*** Spark UI has 6 different tabs, these are jobs, stages, storage, environment, executors and SQL. Below in Figure 3, you can see a screenshot of the Spark UI.

Figure 3

As it is explained shortly in previous parts of this report Spark jobs consist of one or more stages, and each stage consists of multiple tasks. Each of these tasks are completed by an available core in your cluster. After completing all the tasks, a stage gets completed, and after completing all the stages, a job gets completed. Navigating through the jobs and stages tabs of the Spark UI, it is very beneficial to see how jobs and/or stages are going. In this way, it is possible to see the stuck jobs and work in order to solve that. So, jobs and stages tabs of the Spark UI provides a detailed information about jobs, stages and tasks, this allows us to identify the problems about jobs and solve them. Another tab of the Spark UI is storage, in this tab, created datasets and dataframes can be seen. The only part of the storage tab that we benefited is the information about caching. It is very critical to undo caching when the dataset is no longer needed in memory because it is not desired to have unnecessary load in memory. This is done with the unpersist method on dataframes and datasets. Sometimes we forgot to unpersist a dataset, so it is important to check the storage tab regularly to use your RAM efficiently. Next in the environment tab, configurational and environmental settings about the current Spark application can be found, this tab was not very useful while optimizing. In the executors tab, it is possible to view all the executors that the cluster has. In addition to that, you can see the features of each executor such as storage memory, number of cores, number of tasks that is completed by that executor etc. Lastly in the SQL tab, the structure and the details of the queries that you created during the job can be viewed. Analysing these queries, it is possible to make improvements to do the jobs faster and more efficient.

### 4.5.2 Data Filtering (Lazy Loading)

While reading the data or doing operations on the data, commonly, the whole data is read or loaded. This is a common mistake that causes slower jobs on the data processing. Daniel Tomes from Databricks suggests the data filtering for enhancing the performance and the speed of the Spark jobs. Basically, data filtering means reducing the data that will be read or loaded. This can be done by applying a filter on the data by a value of some feature. Most commonly this feature is date. In our case, we applied this technique when we were processing specific columns of a data to do analytics. Rather than reading or loading the whole data, we took to columns that we need to process and dealt with that. This relatively simple approach to reading the data saved us a surprising amount of time.

### 4.5.3 Managing the Number and the Size of Partitions

Partitions is a very critique aspect of distributed processing systems like Spark. Spark divides the data into partitions and does the desired processes on these partitions parallelly by using different cores. So, when we are applying an operation on the data, Spark manages to do this operation on all the partitions of the data and collects the desired result. In Spark, number of partitions of a dataframe can be determined with setting N in spark.conf.set(“spark.default.parallelism”, N). As Karau and Warren states, having at least the same number of partitions as the number of cores in your cluster is a must. For an optimal value, Kozlowski claims that number of partitions should be 2-3 times the number of cores. In our practices, it is seen that this is correct in almost all cases. In addition, Tomes states that size of the partitions should not be more than 200 MB if possible. Surely, the optimal size of the partitions strongly depends on the power of the cores in your system but setting the partition size to 150-200 MB was a good starting point for this project and it did not change most of the cases. What is more important than these normal partitions is shuffle partitions. To understand shuffle partitions, first the term “shuffle” needs to be explained. Shuffle is a process that happens when a transformation that requires to re-distribute (shuffle) happens. Transformation is a function that produces new dataset from the existing datasets. Some of these functions are, map(), filter(), sample(), union(), groupbyKey(), reducebyKey(), join(), repartition() etc. As it is explained before, Spark divides the input data into partitions and executes transformations on them. To execute some of these transformations, Spark must re-distribute the partitions. It is not possible to execute all transformations on a distributed data without communication between partitions, some transformations require partitions to exchange data between them. As Damji explains, these types of transformations are called “Wide Transformations”. Some of these wide transformations are, groupbyKey(), reducebyKey(), join(), repartition() etc. Transformations that are not wide are named “Narrow Transformations”. As it can be guessed, this process of exchanging data between partition is an expensive operation in terms of time because during this shuffle process, Spark creates new partitions called “shuffle partitions” and re-arrange them in order to perform desired transformation on the dataset. So, shuffling needs to be avoided if possible. To optimize the shuffling process and do it faster, shuffle partitions can be re-arranged. Again, Tomes suggests that size of each shuffle partition should be less than 200 MB. Then the number of shuffle partitions can be calculated with dividing stage input data in MBs to shuffle partition size (equal or less than 200 MB) after that it can be set to a desired number by N in spark.conf.set(“spark.sql.shuffle.partitions”, N). For example, if the stage input data is 200 GB, dividing this with 200 MB, the number of shuffle partitions should be 1000 but if number of cores in the cluster is more than the number that this calculation gives, number of shuffle partitions should be set to the number of cores in the system. This is because it is always optimal to use all the cores. If the number of shuffle partitions is set to a number less than the number of cores, remaining cores would not be used as they do not have any partition assigned to them. Size of the shuffled data can be seen in Spark UI. It’s demonstrated in the Figure 4.

Figure 4

To sum up, shuffle is an expensive operation and if possible, it should be avoided but it is not possible to completely remove shuffles from the job. There will always be shuffles as transformations like groupby, join or repartitions always used in data processing. To optimize it, shuffle partition should be focused on. Number of partitions in shuffle stage should be changed dynamically to achieve better speed.

### 4.5.4 Caching

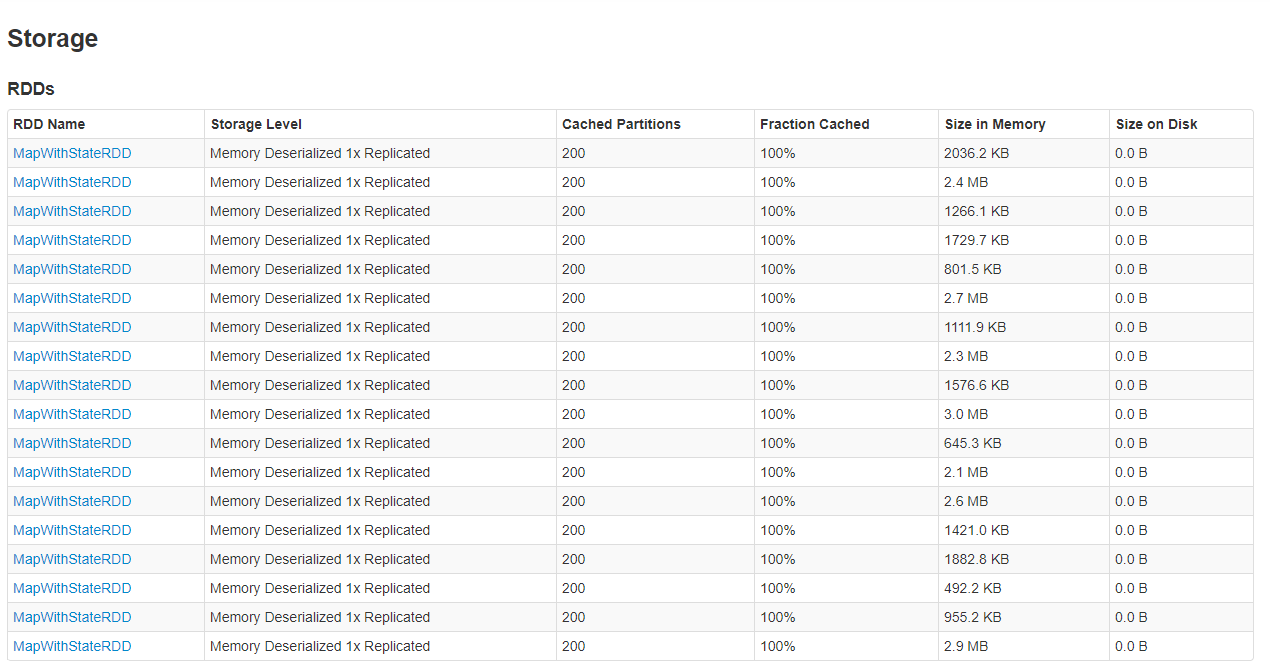
Spark’s ability of in-memory processing does not necessarily mean that all the data is stored and processed in memory. In most of the time, some of the data is stored in the disk, and as it is known, disk is really slow compared to memory. For this problem, Spark has an option to make the data retrieval process faster, this is called caching. Caching is done with cache() method. Other than this, persist() can be used to specifically determine where the data will be stored. For example, when persist(‘MEMORY\_AND\_DISK’) called on a dataset, this dataset is stored in memory and disk, if persist(‘MEMORY\_ONLY’) is called, dataset is stored only in memory. Simply, cache() is the same with persist(‘MEMORY\_ONLY’). Although we have an option to retrieve the data faster, using this on every dataset is not logical as this will overuse the memory. More logical way is caching is to cache the data that used repeatedly in the job. By doing this simple method in this way datasets, surprising improvement in speed and efficiency of the job can be seen. In this project, usage of caching saved the WifiSpeed a lot of time. While using cache, duration of the cache data in memory should be managed properly. When the cached data is no longer needed, unpersist() should be called on this dataset, this will remove the data from memory. Removing the data from cache is necessary because storing unused data in memory is not good resource management. To make this resource management easier, the storage tab of Spark UI helps. In here cached datasets are shown. Below, Figure 5 shows this.

Figure 5

### 4.5.5 Broadcast Join

Joins are one of the most expensive operations. As it is explained before, join is a wide transformation. This means that join operations require shuffle. This makes join even more expensive. So, optimizing joins is one of the critical points of optimizing a Spark job. Unfortunately, most of the improvement techniques about joins are advanced. Applying these advanced methods was not possible in the scope of this project. Tomes and Grah suggests the method of Broadcast Join which is easy to apply and understand. Join operation is occurs between two tables. If one of the tables is small, Tomes suggests using broadcast join. Deciding how many MBs is small is up to the cluster, but if the table can fit in memory without problem, it can be said that it is small. Broadcast join broadcasts the smaller table to all of the partitions, this saves a lot of time because when this broadcasting is done, there is no shuffle needed to complete the join because one of the tables (the small one) is already present in all of the tables. This method is immensely beneficial in terms of time, but it only operates well when one of the tables is small as explained before. In this project, broadcast join is not used very frequently because most of the time two tables of join operation were large, but it is used when one of them is small.

## 4.6 Results

In the tests that are conducted on sample data, it is observed that caching made some of the jobs 1.5-2 times faster. Other techniques such as data filtering, right sizing of partitions showed positive results. In addition, understanding Spark UI helped a lot about figuring out how Spark works. In the light of the tests that are conducted, some of the techniques like caching, right sizing of partitions, data filtering implemented to the WifiSpeed’s data processing. Furthermore, data related part of WifiSpeed is improved with observing and analysing the information in Spark UI. Although this project improved the data processing of WifiSpeed to some extent, it is clear that WifiSpeed has a lot of potential improvement in terms of utilizing Spark to enhance the performance of its data processing. Because of that, it is reasonable to state that this project is not completed. But Mr. Ali, who is the CEO of the QuantWifi, showed great interest and enthusiasm about Spark and its advantages. After the results of this project, company plans to develop their data team and continue to work on Spark to advance their data processing. To sum up, some techniques that are found and tested in this project have been put into action in WifiSpeed’s data processing, and these techniques improved the performance of it. On the other hand, there are still so many possible improvements about the WifiSpeed’s data processing utilizing Spark. So, it is safe to say that this project was successful but not completed.

# 5.INTERNSHIP EXPERIENCE

## 5.1 Learning

In this internship I have learned a lot. I learned Spark which is the most important framework for data engineers whose field is about big data, machine learning, AI etc. In addition, I learned about the logic behind the distributed systems and how they work while learning Spark. Also, I learned about other data related frameworks that are used in WifiSpeed’s data part. For example, Kafka and Clickhouse. Clickhouse was the database that the company used to store their data. Kafka was used to build real-time streaming data pipelines. Other than these technical learnings, I learned how to address and solve optimization problems. For example, doing research and reviewing the literature about the optimization problem taught me a lot. Furthermore, as this was my first experience of being in a workplace physically, I learned how to communicate with other team members. Collaborating with the team helped me develop work related social skills. I learned what it is like to be a team member, and how to be a part of a bigger project.

## 5.2 Relation to Undergraduate Education

As I took CS 210(Introduction to Data Science) and CS 306(Database Systems) I was familiar with data. Also, both courses were including a group project. Information and experience from these courses helped me understand Spark more easily. For example, in CS 306, I have learned about SQL and its structure. In this project, we tried to optimize and improve lots of queries written in SQL. Also, basic Python knowledge on data from CS 210 made it easier for me to understand PySpark (Python API of Spark) which is very similar to standard Python and Pandas. Apart from these knowledges, I wish I could be more familiar with distributed systems. Trying to understand how a distributed data processing system works without any background about distributed systems was challenging. In addition to that, I wish I would be more experienced and familiar with working on big data because it is a lot different than working with normal data.

## 5.3 Difficulties

### 5.3.1 Understanding a New Framework

At the start of this project, I was not familiar enough with Spark. I did brief research about it before my starting date but of course it was not enough. To overcome this difficulty, I worked extra hours to understand and learn Apache Spark. E-books like Learning Spark by Jules S. Damji helped me a lot. Other than E-books I read a lot of articles. Besides that, it can be said that the biggest help came from YouTube. Watching the videos of Matei Zaharia’s, who is the creator of Spark, seminars was very beneficial. So, using internet, I was able to understand and learn Spark.

### 5.3.2 Adapting to an Ongoing Project

When I began to work in QuantWifi, the project WifiSpeed was an ongoing and well-developed project. Adapting to a project like this was not easy at first. I was having difficulties about understanding the purpose of the project. Understanding the service of this project offers was not an easy task. To overcome this, I did not hold back my questions about the WifiSpeed. I asked a lot of questions to all my co-workers even when it was not about my department or project. By doing this, I was able to understand the service that WifiSpeed offers.

## 5.4 A Typical Day

I wake up at 8.00 AM every day, and I be at the office at 9.00 AM. After arriving, I get a cup of tea and I begin to work. Around 12.00, we go to have a dinner together in the cafeteria. In dinner we chat about both work and other things. It was a good way of socializing with my co-workers. After that I continue to work. At 4 PM, I finish my work. If it was a work about doing research, I would report my daily findings to Mr. Ali, and we discuss them together. If my work is something practical like testing an optimization technique, I would report the results and my recommendations about the technique to Mr. Ali. After our meeting with Mr. Ali ends, I pack my things up and leave around 5-6 PM.

# 6. CONCLUSIONS

Main goal of this project was to optimize data processing jobs of the WifiSpeed using Apache Spark. To do this, first task was to understand Spark thoroughly. After that, by scanning the literature and resources about the subject, we were able to collect some methods and techniques to optimize Spark. Next step was to pick the techniques that were suitable and effective in the scope of this project. Lastly, these chosen techniques were tested on sample data, results of the used techniques reported to the supervisor. If the test results are good. Mr. Ali applies these techniques to the real jobs of WifiSpeed. Some of these methods are caching, right sizing of partitions, data filtering etc. So, it can be concluded that Spark was the right way to go as it was able to optimize the data processing jobs in such a short time. Because of the same reason, it can be said that this project was successful. Another conclusion point is the depth of optimization with Spark. Although it is optimized to some degree using these techniques, there are a lot of more techniques to optimize the Spark jobs of WifiSpeed’s data processing.

# 7. RECOMMENDATIONS

The first recommendation is something that I also did not do. It is much better if you know the hiring processes of companies before. To do this, applying companies before the internship period helps to understand what to do in interviews etc. This one did not affect my internship experience a lot, but it would be better if I would know how and where to apply to internship jobs. Besides that, discussing the project with your company before your start is the most important thing to do because in Sabancı University it is mandatory to provide a project and a project report. That is why I asked my company to discuss and determine a project together to work on before starting my internship. This helped me a lot because I knew what to do and I did not have any issues on writing my report. So, deciding on a project with company before starting of the internship is a must. In addition, studying the subject of the project 2 weeks before the start of the internship is a recommendation too. After starting the internship, a good recommendation of mine would be importance of the interaction between your co-workers. Even though the project is about engineering and other technical stuff, it is very important to be a team member in the company. To do this, you need to build a good relationship with your co-workers in the workplace. This also encourages you to ask more questions to them and maximize your learning during the internship.

# REFERENCES

Apache Spark document. Retrieved September 15, 2022, from

<https://spark.apache.org/docs/latest/>

Damji, J. S., Wenig, B., Das, T., & Lee, D. (2020). *Learning Spark*. California: O'Reilly Media.

Grah, S. (2021, November 24). 6 recommendations for optimizing a Spark job. Retrieved August

25, 2022, from

[https://towardsdatascience.com/6-recommendations-for-optimizing-a-spark-job-](https://towardsdatascience.com/6-recommendations-for-optimizing-a-spark-job-5899ec269b4b#:~:text=Ideally%2C%20Spark%20organises%20one%20thread,execution%20of%20the%20Spark%20job.)

[5899ec269b4b#:~:text=Ideally%2C%20Spark%20organises%20one%20thread,execution](https://towardsdatascience.com/6-recommendations-for-optimizing-a-spark-job-5899ec269b4b#:~:text=Ideally%2C%20Spark%20organises%20one%20thread,execution%20of%20the%20Spark%20job.)

[%20of%20the%20Spark%20job.](https://towardsdatascience.com/6-recommendations-for-optimizing-a-spark-job-5899ec269b4b#:~:text=Ideally%2C%20Spark%20organises%20one%20thread,execution%20of%20the%20Spark%20job.)

Grishchenko, A. Spark Architecture. Retrieved September 10, 2022, from [https://0x0fff.com/wp-](https://0x0fff.com/wp-content/uploads/2015/11/Spark-Architecture-JD-Kiev-v04.pdf)

[content/uploads/2015/11/Spark-Architecture-JD-Kiev-v04.pdf.](https://0x0fff.com/wp-content/uploads/2015/11/Spark-Architecture-JD-Kiev-v04.pdf) 42-59.

Karau, H., Warren, R. (2017). *High performance Spark: best practices for scaling and optimizing Apache Spark*. California: O'Reilly Media. 285.

Kozlowski, N. (2017, November 2). Partitioning in Apache Spark. Retrieved August 25, 2022,

from

<https://medium.com/@don_khozzy/partitioning-in-apache-spark-8134ad840b0>

L. Gu and H. Li (2013). Memory or time: Performance evaluation for iterative operation on

Hadoop and Spark, *2013 IEEE 10th International Conference on High Performance*

*Computing and Communications & 2013 IEEE International Conference on Embedded*

*and Ubiquitous Computing*, 721-722.

Salloum, S., Dautov, R., Chen, X. et al (2016). Big data analytics on Apache Spark.

*International Journal of Data Science and Analytics*, 145.

Tomes, D. (2019). Apache Spark proper optimization. Retrieved August 20, 2022, from

<https://www.youtube.com/watch?v=daXEp4HmS-E&t=3133s.>

QuantWifi. (2019). Retrieved September 24, 2022, from <https://quantwifi.com>