Benchmarking Cloud databases: Microsoft Azure vs Google Cloud

Advanced Database (INFO-H415)

Caiola Ludovica (588085)
Coessens Simon (583180)
Rocca Valerio (589084)
Salazar Maria (587449)

December 15, 2023

Contents

1 Introduction **1**
- 1.1 Background and Context . . . 1
- 1.2 Project Objective . . . 1
- 1.3 Scope and Limitations . . . 1

2 Technologies fundamentals **2**
- 2.1 Introduction to PostgreSQL . . . 2
  - 2.1.1 Why PostgreSQL? . . . 2
- 2.2 Cloud Computing foundations . . . 2
  - 2.2.1 Comparison with traditional technologies . . . 3
- 2.3 Introduction to Microsoft Azure . . . 4
  - 2.3.1 Azure Database for PostgreSQL - Flexible Server . . . 4
- 2.4 Introduction to Google Cloud . . . 5
  - 2.4.1 Google Cloud SQL . . . 5

3 Introduction to the Business Case **7**
- 3.1 Taxi Lecco's Data . . . 7
- 3.2 The Database . . . 7
  - 3.2.1 Operations on the Database . . . 8

4 Configuration and Setup **8**
- 4.1 Setup of Cloud Servers . . . 8
- 4.2 Setup of Local Machines . . . 9

5 Benchmark **10**
- 5.1 Project's Benchmark . . . 10
  - 5.1.1 Project's Benchmark Execution . . . 13
- 5.2 TPROC-C . . . 13
  - 5.2.1 TPROC-C Execution . . . 14

6 Results **15**
- 6.1 Project's Benchmark . . . 15
- 6.2 TPROC-C . . . 17

7 Discussion and Conclusion **20**
- 7.1 Transaction Analysis . . . 20
- 7.2 Cost Analysis . . . 20
- 7.3 Conclusion and further research . . . 21

1 Introduction

1.1 Background and Context

Nestled at the southern tip of Lake Como in Italy, the city of Lecco beckons travelers with its picturesque landscapes and rich cultural heritage. Despite its modest size - in 2021 it counted only 46,831 inhabitants - Lecco emerges as a hidden gem for tourism, drawing visitors with its stunning lakeside setting and proximity to the Alps.

"Taxi Lecco" is an organization representing taxi operators in the city. The association is developing an application that enables clients to request taxi rides. This could be a challenge for its current OLTP (Online Transaction Processing) system; thus, the organization has tasked our team with assessing the feasibility of hosting the system on the Cloud.

1.2 Project Objective

This project aims to assess whether it is convenient for the organization to migrate the company's OLTP system to the Cloud. In particular, we focus on Microsoft Azure and Google Cloud solutions.

To pursue this result, we performed two different benchmarks tested locally and on both Cloud providers: one was created specifically for this project, and the other is TPROC-C, derived from TPC-C. Considering Cloud peculiarities, conclusions are drawn not only based on performances of the machines; instead, they focus more on costs and additional advantages of the Cloud compared to local systems.

1.3 Scope and Limitations

The main limitation of this project is linked to the features of Cloud services themselves. As we will discuss later, one of their main advantages is the opportunity to scale the resources based on the system's needs. This constitutes a problem for this project, as we cannot predict a priori the exact amount of resources spent. This issue is further aggravated by the nature of those non-fixed costs, which depend on a series of factors.

We then decided to approximate the "pay-as-you-go" costs as a charge per transaction. This choice is further explained in paragraph 6.2.

2 Technologies fundamentals

2.1 Introduction to PostgreSQL

PostgreSQL is a powerful, open-source object-relational database management system (RDBMS) that uses and extends the SQL language combined with many features to store and scale complex data workloads. PostgreSQL supports both non-relational and relational data types, making it one of the most compliant, stable, and mature relational databases available today. It runs on all major operating systems, has been ACID-compliant since 2001, and has powerful add-ons such as the popular PostGIS geospatial database extender. [9]

2.1.1 Why PostgreSQL?

PostgreSQL stands out as the optimal choice for spatial data due to its seamless integration with the robust geospatial extension, PostGIS. Offering a comprehensive suite of spatial functions, support for Open Geospatial Consortium (OGC) standards, and compatibility with various GIS tools, PostgreSQL ensures efficient handling of complex geospatial analyses. Its open-source nature and active community contribute to ongoing support, updates, and customization.

Taxi Lecco's database will deal with coordinates and, in general, spatial information. While it is currently not planned to include spatial queries, this is a natural evolution for the database, and thus the organization may decide to incorporate them. To make the system future-proof, we thus decided to use PostgreSQL.

2.2 Cloud Computing foundations

During the last decades, a variety of definitions of Cloud Computing emerged. We report a simple yet powerful one made by the Microsoft Azure team: "Cloud computing is the delivery of computing services—including servers, storage, databases, networking, software, analytics, and intelligence—over the internet ("the Cloud") to offer faster innovation, flexible resources, and economies of scale." [12]

The concept of computing in a "Cloud" can be traced back to the origins of utility computing, as proposed by computer scientist John McCarthy in 1961: "If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility. ... The computer utility could become the basis of a new and important industry." [14]

It was only in 2006 that the phrase "Cloud computing" made its debut in the corporate realm when Amazon.com introduced the Amazon Web Services (AWS) platform, offering a collection of business-focused services that deliver remotely provisioned storage, computing resources, and business functionality. [14]

In 2009, Google Play ventured into the provision of Cloud Computing Enterprise Applications, marking the onset of a broader trend within the industry. Subsequently, Microsoft unveiled Microsoft Azure, marking a significant step in the proliferation of Cloud services. Following suit, companies such as Alibaba, IBM, Oracle, and HP introduced their Cloud Services. Today, Cloud Computing has emerged as a highly significant and widely adopted technology, with growing importance as a skill set in the academic and professional domains. [7]

2.2.1 Comparison with traditional technologies

According to a 2021 survey commissioned by the learning platform O'Reilly, roughly 90% of the respondents indicated that their organizations are using the Cloud. [15]. There are plenty of advantages that involve Cloud migration, in particular when it refers to small and medium-sized businesses (SMBs), such as this one.

Cloud migration is a crucial step towards establishing a **secure and protected environment**, if set properly. In general, small companies are more likely to be targeted by attackers than large ones [10]. This is because large companies typically have more resources to invest in robust security measures, making their environments more difficult to breach. Conversely, small companies often lack adequate security in their infrastructure, making them attractive targets for attackers. When considering typical attacks, malware is often one of the first threats that come to mind. If malware infects a local machine where all the company's data is stored, it could lead to a complete loss of that data. Secondly, ransomware poses a significant threat because the theft of data could lead to a company's failure, as the ransom demanded is typically higher than the company's total profits. Therefore, many of these real and pressing issues could be avoided by choosing to transfer our applications to the cloud.

These attacks and many others can be avoided by migrating to the Cloud. The choice of going from a local platform to a Cloud one can be considered as an assurance for small businesses, as big companies that manage Cloud service spend lots of resources on IT security. Companies that have migrated to the Cloud are less likely to be targeted by these types of attacks. Moreover, the availability of multiple backups allows for data restoration in the event of encryption and a ransom demand from an attacker. [12]

For all these reasons, migrating to the Cloud can be an underrated step for a company, but a necessary one for ours. The migration process has to be accompanied by a basic but essential education for the company's employees. This includes for example not granting excessive privileges to everyone, as this could lead to an escalation of privilege issues, having a secure password to keep one's account away from the spotlight of attackers, enabling multi-factor authentication, and other necessary practices [2].

Security and reliability are not the only advantages linked to Cloud solutions; the other pros are presented below.

Considering the **cost**, businesses with on-site servers spend a lot of money on their IT infrastructure and operations, compared to large Cloud services companies, such as Google and Microsoft, which already have ready infrastructures and services in place to maintain company data on their respective platforms. In addition to that, companies provide a Cloud support service, which is generally cheaper than hiring an IT specialist on-site. As IT systems expand, substantial investments are needed, mainly in infrastructure expansion. This includes both the cost of acquiring new infrastructure and the ongoing costs of ownership (such as maintaining technical personnel, managing upgrades and patches, covering utility bills, and ensuring security and access control), which often exceed the initial investment [14].

With regards to the **scalability**, cloud consumers can scale their IT resources up or down to accommodate fluctuations, whether automatically or manually configured. This scalability is a cost-efficient feature, as it allows resources to align with demand, preventing potential business loss due to usage thresholds being exceeded [14]. This advantage is particularly important in our scenario, as it involves a range of workloads that fluctuate under diverse conditions, such as festive days and weather.

Business agility refers to an organization's ability to adapt to both internal and external changes. This is particularly relevant in IT, a fast-changing realm where it often involves scaling resources beyond initial predictions. High initial investments and infrastructure costs might force businesses to settle for sub-optimal IT, impacting their ability to meet real-world demands. [14]

2.3 Introduction to Microsoft Azure

Where Amazon Web Services took the lead in 2006 with their first Cloud services, Microsoft followed and launched their first Cloud service, under the name Windows Azure, to the public in 2010. Microsoft, with its well-established global trust and reputation in business software, could not afford to miss this significant opportunity. The First-Generation of services offered by Azure where a direct response to services offered by AWS. Notable first services launched include web hosting for `ASP.NET` applications and `SQL Azure`. Microsoft Azure has since been expanding and has adapted to trends and changes in the industry.

Currently Microsoft Azure is a Cloud computing platform that offers a wide range of services for building, testing, deploying, and managing applications and services through Microsoft-managed data centers. It provides a range of capabilities, including software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS). Azure supports many programming languages, tools, and frameworks. [8]

In the analysis provided by Synergy Research Group [16], Microsoft Azure has demonstrated a steady increase in market share in the Cloud industry over the last six years, making it the second biggest player in the Cloud market followed by Google Cloud and Alibaba Cloud.

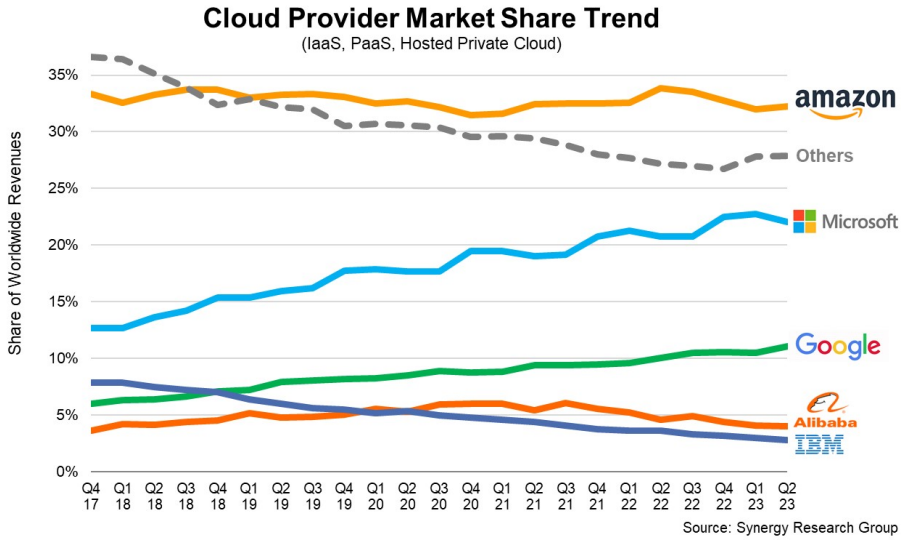

Figure 1: Synergy Research Group market study

2.3.1 Azure Database for PostgreSQL - Flexible Server

Azure Database for PostgreSQL - Flexible Server is a comprehensive database service that is fully managed. It is designed to give users more control and flexibility over how they manage databases, allowing for customization of settings. This service is known for its flexibility, enabling users to finely tune configurations according to their needs. With a flexible server

architecture, users can place the database engine closer to the client tier, reducing latency. Additionally, users have the option to enhance availability by choosing between a single availability zone or multiple availability zones. [1]

2.4 Introduction to Google Cloud

Google Cloud is a broad term that encompasses all of Google's Cloud-based services: this includes Google Cloud Platform as well as other services. Google Cloud Platform (GCP) is a series of cloud computing services offered by Google. Google Cloud Platform provides infrastructure as a service, platform as a service, and serverless computing environments. One of the key features of GCP is its wide range of services, which include computing, storage, networking, big data, and machine learning as well as Cloud management, security, and developer tools.

The first step of the project was choosing the right Cloud service. Our final choice is Google Cloud SQL, but here we want to briefly describe the other possible solutions and why we did not proceed with them.

- **Cloud Spanner**. This is a distributed SQL database management and storage service developed by Google suitable for both relational and non-relational workloads [5]. We did not opt for it as it is horizontally scalable, meaning that the system can handle bigger workloads by adding more servers or nodes to the database cluster. Our project deals with a small number of transactions, so the vertical scalability (i.e., increase the capacity of a single server) offered by Google Cloud SQL was preferred.
- **BigQuery**. The system appealed to us because it provides geospatial analytics that allows for the analysis and visualization of geospatial data. However, BigQuery is a data warehousing tool, while our project focuses on a RDBMS.

2.4.1 Google Cloud SQL

Google Cloud SQL is a fully managed relational database service offered by Google Cloud Platform. It indeed supports PostgreSQL as well as MySQL and SQL Server [4].

The key features of Google Cloud SQL include:

- **Fully managed**: Google Cloud SQL manages databases, automating backups, replication, patches, encryption, and storage capacity increases;
- **Open and standards-based**: it supports popular open-source and commercial engines and it is very user-friendly;
- **Easy migration**: whether from on-premises, on Compute Engine, or other Clouds, Google Cloud SQL's Database Migration Service can migrate databases securely and with minimal downtime; [3]
- **Integrated**: the system can be integrated with services like Compute Engine, allowing companies to build and launch their own application;
- **Backups**: in Google Cloud SQL the user can decide to schedule or create an on-demand backup. Backups help to restore lost data to Cloud SQL instances and protect them from loss or damage.

Each Cloud SQL instance is powered by a Virtual Machine (VM) running on a host Google Cloud server. Each VM is running a specific database program, like MySQL server, PostgreSQL,

or SQL Server. This setup ensures and provides flexibility, scalability, and isolation of resources in a Cloud environment.

Choosing to host our application on Google Cloud SQL offers several advantages that align with our needs:

- **Geospatial Support**: Google Cloud SQL for PostgreSQL supports the popular PostGIS extension for geospatial objects and scalability;
- **Simplicity**: the tool's ease of use perfectly matches the small size of both the organization and the case.
- **Vertical Scalability**: if the number of tourists in the city will increase and the server's resources will not be enough, it will be sufficient to increase the server's computational power.

3 Introduction to the Business Case

This chapter aims to provide a brief introduction to the business case.

3.1 Taxi Lecco's Data

Taxi Lecco provided the team with data about the taxi service in the city. A key piece of information is that the maximum number of taxi rides in a minute ever registered is 44. This is crucial to decide the size of the Cloud servers. In table 1, we highlight some data useful for later cost analyses.

Season	Hour	Average number of rides per hour	Hours per day*
Touristic	Peak	403	3
Touristic	Other daytime	184	13
Touristic	Nighttime	66	8
Non-touristic	Peak	103	1
Non-touristic	Other daytime	70	15
Non-touristic	Nighttime	15	8

Table 1: Taxi rides data. *: hours per day belonging to that category

3.2 The Database

Being a business model based on a mobile application that connects passengers with drivers, captures information, and bases its decisions on the data it collects, our business model is based on consumer demand. The **main features** of our system must be high availability and the ability to deal with multiple transactions per second.

Because of its structure, Taxi Lecco's database is not suitable to host data generated from an Uber-like application. We decided to create our own database that could better match the new organization's objectives. We briefly looked at the following dataset for inspiration: Uber pickups in NYC.

In terms of an operational database, **Uber** receives from passengers a request for a trip, which includes among other information, the origin and destination, so that based on distance, traffic, and other factors. A fare can be returned to the user. Multiple trips can be requested at the same time, due to events in the area, peak hours for work, schools, or universities, so the system must be prepared to receive all transactions successfully.

In Figure 2, the **Entity-Relationship diagram** can be seen. The database creation script can be found in the Appendix. The Database consists of five tables and each one has a primary key. Considering that we are working with spatial data we chose to store the `pickup_location` and `dropoff_location` in `POSTGIS` datatypes. This approach provides more flexibility and allows for sophisticated spatial queries. These will also be tested in the benchmark. Furthermore, the choice of attributes was made to include all the essential fields required for the operational database of our application.

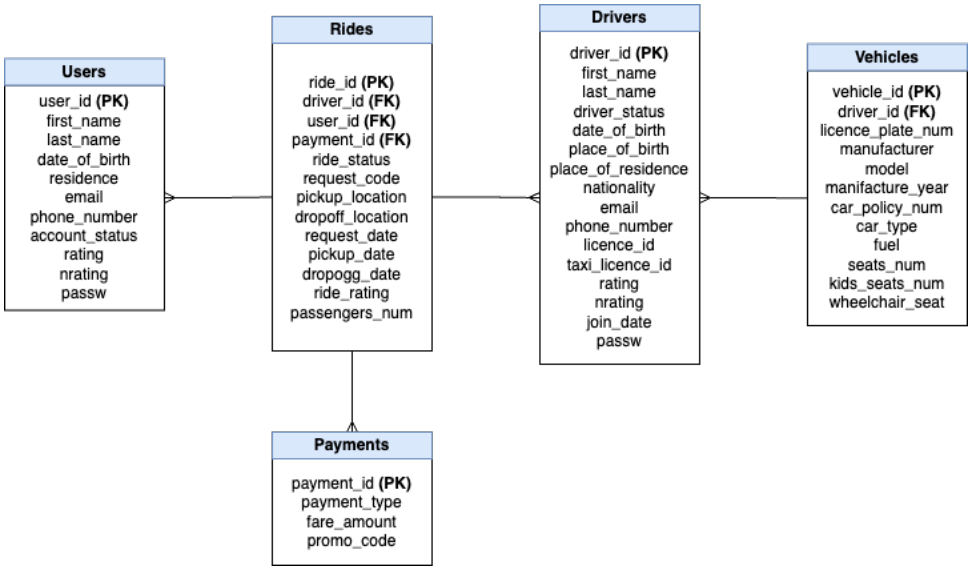

Figure 2: ER schema of the OLTP database

3.2.1 Operations on the Database

This section aims to provide a list of the main operations that are planned to be computed on the server:

1. update driver status after a ride is completed and when the driver stops or resumes working;
2. insert a new ride after a ride is completed;
3. insert a new payment after a ride is completed;
4. if done by the user, update the rating of a driver after a ride is completed;
5. if done by the user, update the rating of a user after a ride is completed;
6. when a client creates an account, insert a new user in the `Users` table;
7. when a new taxi driver registers, insert them in the `Drivers` table;
8. when a new vehicle is registered, insert them in the `Vehicles` table;
9. at 3 A.M. every day, data in tables `Rides` and `Payments` is copied in a `.csv` file and then removed from the database (this operation is later referred also as "daily ETL").

4 Configuration and Setup

4.1 Setup of Cloud Servers

To perform this benchmark it is of course important that both Cloud instances have similar resources. The configuration characteristics can be seen in 3. As described before, our system is supposed to deal with a low amount of transactions. Thus, we configured the two servers with modest computational resources.

In Google Cloud SQL the enterprise edition was chosen. This is a summary of the characteristics:

	Google Cloud	Microsoft Azure
DB version	PostgreSQL 15.4	PostgreSQL 15.4
Location	US Central 1	Central US
vCPUs	1 vCPU	1 vCPU
Memory	3.75 GB	2 GB
Data Cache	Disabled	-
Storage	32 GB	32 GB
Backup	Automated	7 days
Availability	Single zone	Single zone
Edition	Enterprise	-

Table 2: Configuration of Cloud servers

To perform the benchmark, we needed the `POSTGIS` extension. On both Cloud services it was fairly easy to set up: in Google Cloud it is preinstalled for every instance; in Azure the extension needed to be enabled in the server parameters.

We decided to remotely connect to the Cloud instances using our local machines and execute the Project's benchmark via Python. To facilitate this, we added the public IP address of the connecting machine to the server's list of authorized connections in its configuration settings.

4.2 Setup of Local Machines

Local machines have been used to test the benchmarks locally. Follow the specifics of the local machines.

Machine 1	
Type	Laptop
Operating system	Windows 11 Home
Processor	AMD Ryzen 3 3250U 2.60GHz
RAM	8.00 GB

Table 3: Configuration of Machine 1

Machine 2	
Type	Laptop
Operating system	macOS
Processor	Apple M1
RAM	8.00 GB

Table 4: Configuration of Machine 2

5 Benchmark

Conducting a benchmark to compare the performance of PostgreSQL in a cloud environment versus a local implementation is a crucial step in strategic decision-making for database infrastructure. This assessment provides managerial insights into the nuanced trade-offs between leveraging the Cloud's dynamic resources and maintaining a controlled local environment. By measuring key metrics such as query response times, overall efficiency, and scalability, managers gain a comprehensive understanding of how each deployment option aligns with project requirements and business objectives. The benchmarking process facilitates a cost-benefit analysis, shedding light on factors like operational expenses, availability, and adaptability. [13]

Below there are two Benchmarks: the first benchmark was designed by ourselves based on the Cloud Service Benchmarking book [13] and inspired by the Uber business model, the second is TPC-C, recognized benchmark used to evaluate the performance of online transaction processing (OLTP) systems.

5.1 Project's Benchmark

In this benchmarking, it is essential to note that the focus is primarily on the non-functional attributes rather than the functional intricacies of each tool. The goal of this process is to quantify key performance indicators such as scalability, latency, and overall system responsiveness. Unlike a functional assessment that delves into the specific features and capabilities of each deployment option, the benchmark provides a broader understanding of how well PostgreSQL adapts to varying workloads, handles increasing data volumes, and manages latency in different scenarios. However, some features related to the "ease of use" of the tools used will be described.

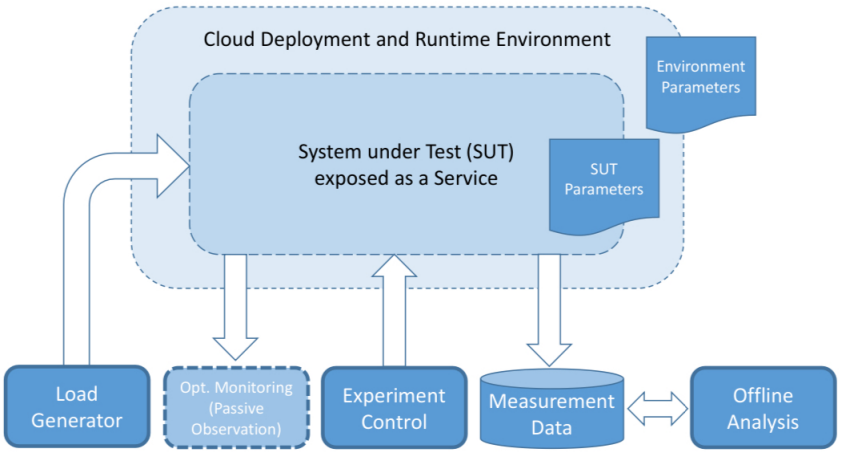

Figure 3: Standard components of Benchmarking [13]

It was decided to follow the steps shown above in designing our own benchmark:

- **Load generator:** as previously stated, the concept of a benchmark is to evaluate various characteristics of a system, which commonly involves exposing the system to stress or limit situations, observing its response to what we call workload. Generally speaking, there are two main categories of workloads: synthetic and trace-based.

  For this case, the workload built was trace-base, which means that it consists of a sequence

of instructions that specifies in detail the requirement. The advantage of this type of workload is that it is totally repeatable. An additional characteristic of the trace-base workload is that it is generally based on a real-life application, in our case, it will be the Lecco taxi business model.

In addition to being a business model based on a mobile application that connects passengers with drivers, and captures information and bases its decisions on the data it collects, it is a business model based on consumer demand, so it not only requires high availability, but it must also be willing to receive multiple transactions per second.

In terms of operational database, Lecco taxi receives from passengers a request for a trip, which includes among other information, the origin and destination, so that based on distance, traffic and other factors, a fare can be returned to the user. Multiple trips can be requested at the same time, due to events in the area, peak hours for work, schools or universities, so the system must be prepared to receive all transactions successfully.

Our workload based on Lecco taxi business case has two parts, database and queries. As already described in the database section, we searched the internet for datasets suitable for our needs but were unable to find any that had all the information required for our application.

The area of the city of Lecco and relevant surrounding towns were divided using Google My Maps. The zones - called “rioni” - are delimited by drawing polygons around them and exported as a KML file. Image 4 shows the zones from a satellite point of view.

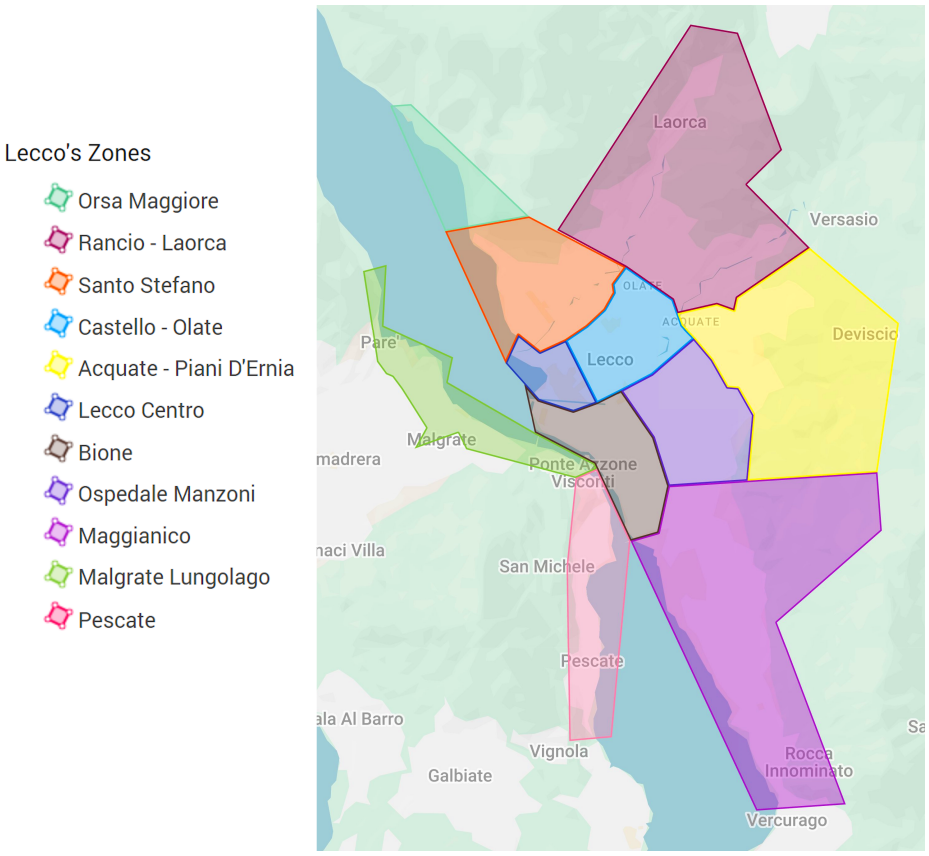

Figure 4: Satellite view of Lecco’s chosen zones

We continued to build our own Data Generator. For this purpose we used Python's Faker library: Faker. The data is written to `csv` files and then pushed to the Database using the Postgres `COPY` command for the non spatial tables.

```
\COPY vehicles FROM 'vehicles.csv' DELIMITER ',' CSV HEADER;
```

Listing 1: Postgres COPY Command for Data Import

For the spatial tables each value is inserted seperately because the `ST_MakePoint` from `POSTGIS` must be used. This function takes the longitude and latitude that is stored in the `csv` files as input.

```
INSERT INTO rides (
    ride_id, driver_id, user_id, ride_status, request_code,
    pickup_location, dropoff_location, request_date,
    pickup_date, dropoff_date, rating, payment_id, passengers_num
) VALUES (
    %s, %s, %s, %s, %s,
    ST_MakePoint(%s, %s), ST_MakePoint(%s, %s),
    %s, %s, %s, %s, %s, %s
)
```

Listing 2: SQL INSERT Command for Rides Table

- **Queries:** as mentioned above, 24 traced based queries were built for this exercise. The number of iterations of these queries depends proportionally on the scale at the time of execution. Within the requested transactions, update, insert, delete, aggregate and read processes were performed, some of these transactions are conventional and others have at least one spatial component. Here are some examples:

```
    UPDATE drivers
    SET driver_status = 2
    WHERE driver_id = %s;
```

Listing 3: Update transaction

```
    INSERT INTO rides (
    ride_id, driver_id, user_id, ride_status, request_code,
pickup_location, dropoff_location, request_date, pickup_date, dropoff_date,
 ride_rating, payment_id, passengers_num)
    VALUES (
    %s, 1, %s, 'TEST', 27164,
    ST_MakePoint(40.89911326259291, -74.19879968),
    ST_MakePoint(40.89119552115688, -73.78466389),
    '2023-05-04', '2023-07-07 20:03', '2023-08-23 07:54',
    4, 1, 2);
```

Listing 4: Insert transaction

```
SELECT v.vehicle_id, SUM(ST_Distance(r.pickup_location, r.dropoff_location
)) AS total_distance
FROM vehicles v, rides r, drivers d
WHERE r.driver_id = d.driver_id AND v.vehicle_id = d.driver_id
GROUP BY v.vehicle_id;
```

Listing 5: Aggregate transaction

```
DELETE FROM rides
WHERE ride_id =%s;

```

Listing 6: Delete transaction

- **Experiment control:** after defining the workload as specified above, measurement scales were established in order to saturate the SUTs (systems under test) with different intensity levels while maintaining the original characteristics of the workload and without violating the rules of the data model. One of the main challenges in the implementation of the benchmark is to maintain the correctness and fairness in the evaluation of the SUTs, and for this it is best to divide the execution phase into two: the first as a controlled experiment that optimally configures the system, loads the data and has some warm-up sessions in order to stabilize the behavior of the SUT, and the second will finally be the experiment to extract the results of the SUT.

  For this particular case, a controlled experiment was done to see the number of iterations needed to start having consistent or constant results, so it was decided to take the first three for the first phase and the remaining ones (7) as stable and observational benchmark results.

- **Measurement data:** the simplest way to collect the data resulting from a benchmark is to write a results file on each system and then bring it to an offline processing or analysis location manually. In our case, the entire experiment is run from a local Python connection that creates `.txt` files for the data load and the ten runs of the queries for each of the scales.

- **Offline analysis:** in the folder "Analysis" the averages are calculated and the graphs are produced. Here we dropped the first three runs for each scale factor run.

5.1.1 Project's Benchmark Execution

For the execution of the workload, it is important to take into account the limited resources that were available in the cloud and the limitations of each one. Based on the limited resources we set scales 2, 10,100,1000. For the Azure account, we used the free resources offered by Azure Postgres Database, and according to these limits, we created the most similar configuration in Google Cloud SQL.

For the local execution, we used Machine 2 and the connection to the Cloud databases through DataGrip, because although for Azure Postgres database it is easy to manipulate directly from bash within the platform, for Google Cloud SQL there are limitations in the file upload (restriction to only `SQL` and `csv`). We conducted six runs for each scale factor, excluding the initial run, which was executed solely to prime the cache structures.

5.2 TPROC-C

The TPC-C is an online transaction processing (OLTP) benchmark approved in 1992. It assesses a blend of five simultaneous transactions of varying types and complexities, executed either online or queued for deferred execution. The database for TPC-C consists of nine types of tables, featuring a diverse range of records and population sizes. Although the benchmark simulates the operations of a wholesale supplier, its applicability extends beyond any specific business segment. Instead, it accurately represents industries that engage in the management, sale, or distribution of products or services. [11]

The TPROC-C is an open-source benchmark derived from the TPC-C. The modifications are made to simplify and make it cost-effective for running on any supported database environments within HammerDB, a software that allows testing the performance of database systems. [6]

We have chosen to benchmark using TPROC-C mainly for three reasons:

- **Reliability**: TPROC-C provides us with the opportunity to compare our results with those generated by an industry-certified and recognized benchmark.
- **Concurrency**: TPROC-C enables us to assess the Cloud server's performance in handling transactions executed by multiple users simultaneously.
- **Cost computation**: HammerDB outputs the number of transactions executed per minute (TPM), which is returned every 10 seconds in a log file. This allows us to approximate the cost per transaction, useful in the cost analysis.

5.2.1 TPROC-C Execution

The systems were benchmarked on four scale factors, with an exponentially growing number of total transactions per user: 500, 5,000, 50,000, and 500,000. The other parameters were fixed for all the runs: number of virtual users (6), number of data warehouses (12), ramp-up time (2 minutes), and test time (10 minutes).

In this case, we performed six runs for each scale factor, the first one being not registered. The TPROC-C benchmark was executed on Machine 1. We also registered the credit reduction for the execution of the benchmark and the total amount of transactions executed, both on Microsoft Azure and Google Cloud.

6 Results

6.1 Project's Benchmark

Now we will look at the results for our Project's Benchmark. We will first analyse the loading times for each scale factor. The result can be seen in 5 and 6. Here we see similar results for Google Cloud and Azure. The time required to create the rides table is significantly longer compared to other tables, primarily because of the utilization of spatial `POSTGIS` functions during its creation.

Operation	**Scale 1 (s)**	**Scale 10 (s)**	**Scale 100 (s)**	**Scale 1000 (s)**
create_database	0.51	0.21	0.16	0.17
users	0.24	0.24	0.51	1.06
vehicles	0.26	0.24	0.24	0.57
drivers	0.24	0.24	0.25	0.91
payments	0.26	0.25	0.47	1.47
rides	24.99	123.02	1243.07	12942.21

Table 5: Execution times for database operations at different scale factors Google Cloud

Operation	**Scale 1 (s)**	**Scale 10 (s)**	**Scale 100 (s)**	**Scale 1000 (s)**
create_database	0.24	0.28	0.35	0.28
users	0.27	0.26	0.54	1.41
vehicles	0.27	0.26	0.25	0.32
drivers	0.27	0.26	0.26	0.35
payments	0.27	0.26	0.43	1.76
rides	28.56	133.62	1381.94	14078.79

Table 6: Execution times for database operations at different scale factors on Azure

Further we will look at the execution time of the queries described in section 5. First we look at figure 5. This figure presents a box plot illustrating the average running times for all queries across all scale factors. Here we can see that Azure and Google Cloud have very similar results. Its notable that the Local performance is better than the two others. There is no latency for the local connection which might explain the difference.

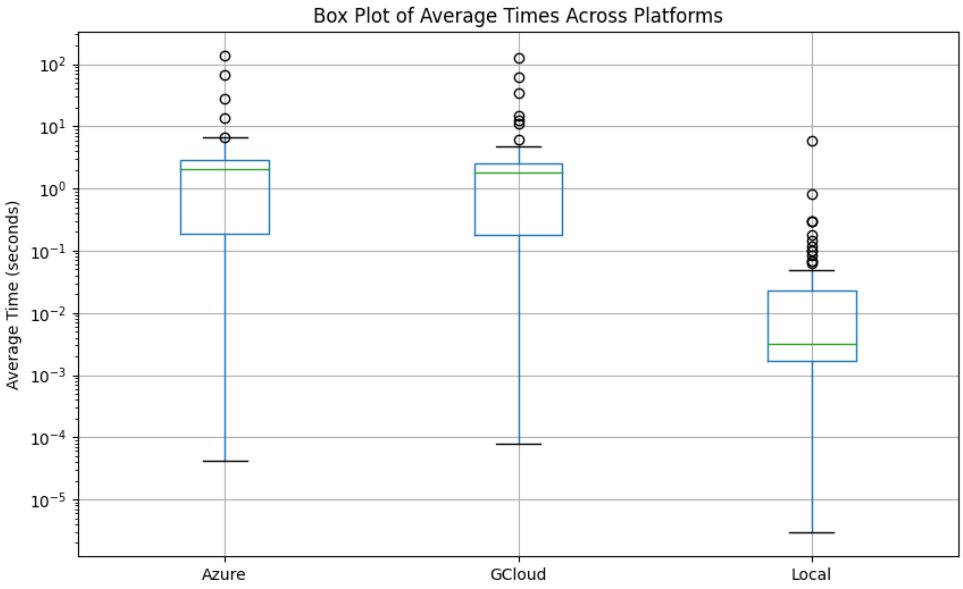

Figure 5: Box plot of average running times for all queries

In Figure 6 we see the benchmark results for query one across the different services and scale factors. Also here we can clearly see that local outperforms both Cloud services. This pattern is consistent across other queries so we will proceed with a more focused comparison between Azure and Google Cloud.

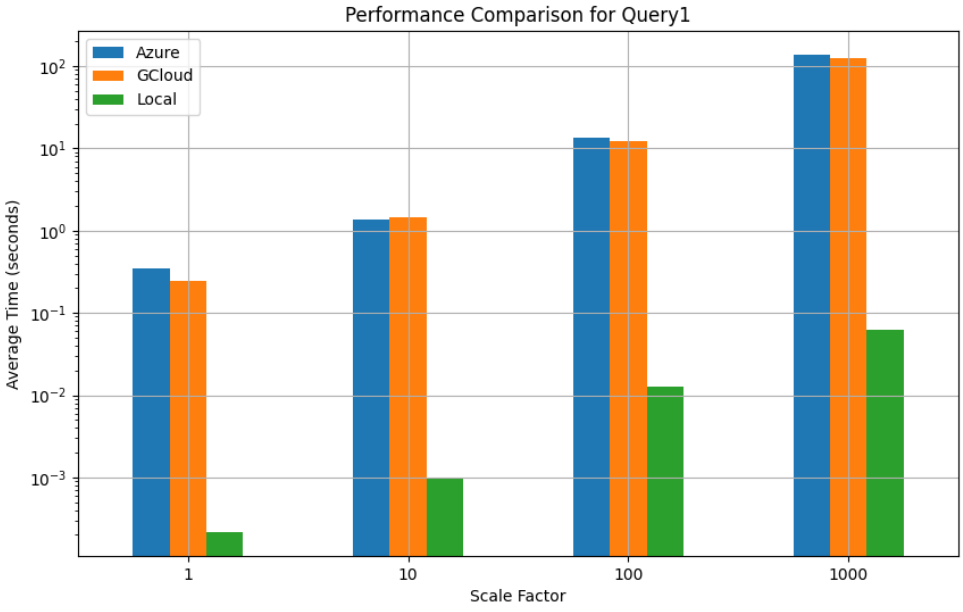

Figure 6: Comparison for Query 1

Figure 7 shows a detailed comparison between Azure and Google Cloud. Here, we again notice that the performance results are very similar, which is expected due to the similar resource settings of both Cloud instances.

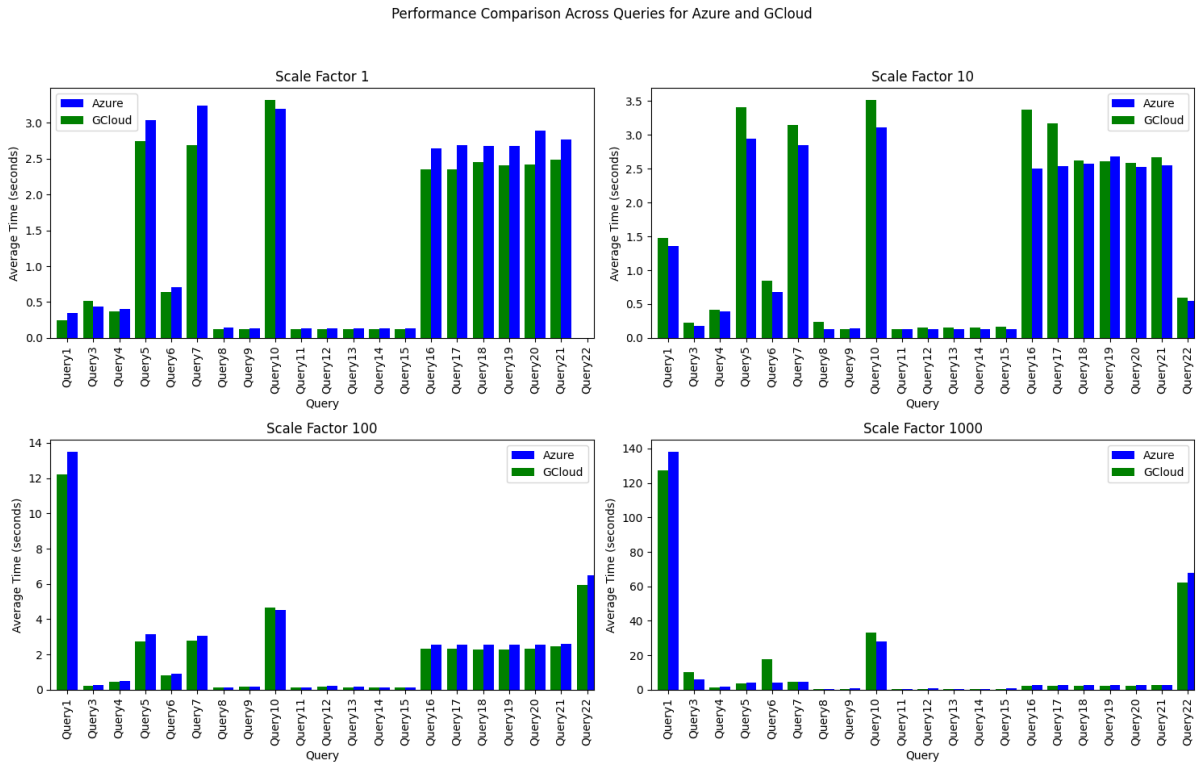

Figure 7: Google Cloud vs Azure comparison

6.2 TPROC-C

The raw log files, showcased in Figures 8, 9, and 10 with time series plots, unveil some intriguing insights. Notably, local runs demonstrate higher TPM values, reaching up to 12,000, compared to Cloud runs, which achieve at most values under 2,000 TPM. This discrepancy also reflects in execution times, with local runs being faster due to their ability to achieve higher TPMs.

An interesting distinction arises between Google Cloud and Azure runs. In Azure logs, sporadically, a single timestamp stands out, registering around 80,000 TPM. This could be attributed to the system handling transactions that the server initially struggled to process. However, this is not registered in Google Cloud runs.

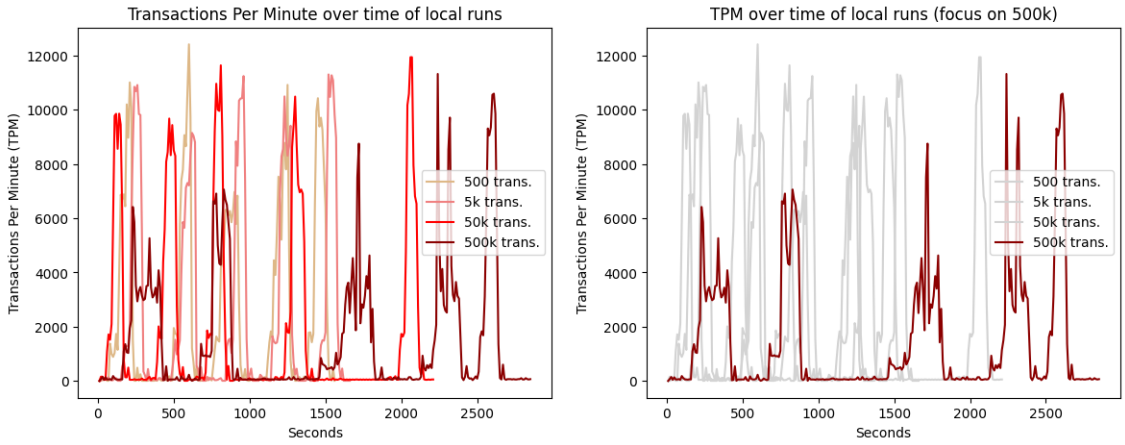

Figure 8: Raw time series of the local runs

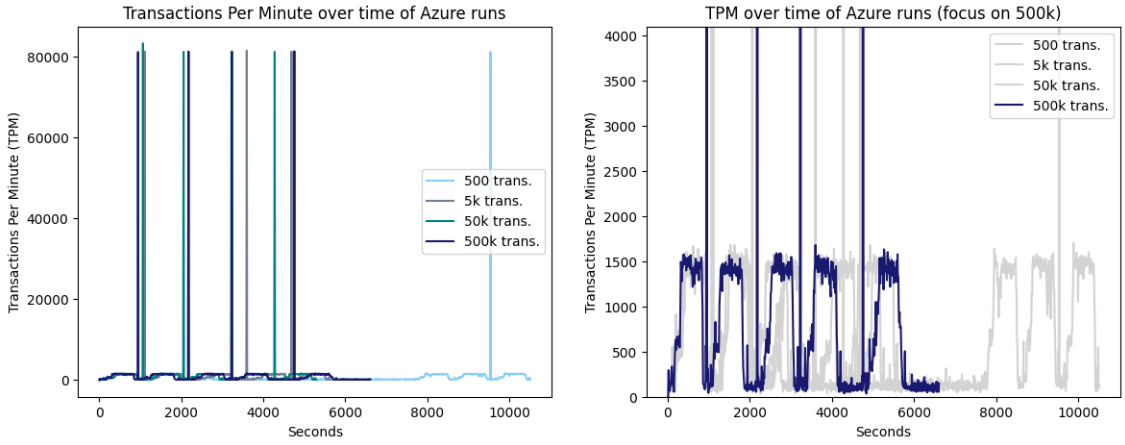

Figure 9: Raw time series of Azure runs

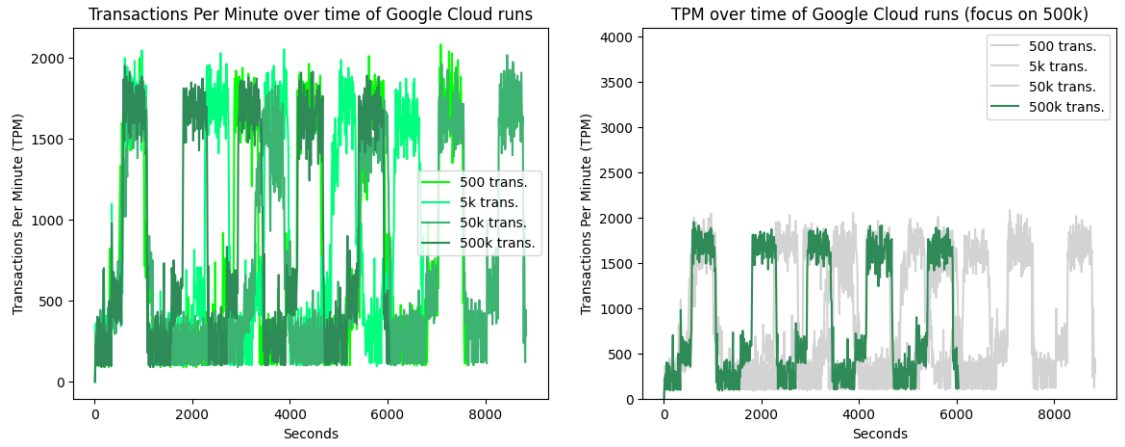

Figure 10: Raw time series of Google Cloud runs

The raw time series described were handled by some tailored functions that computed the average between the runs. Figure 12 emphasizes even more the limit in the TPM the Cloud

servers can handle. Another interesting insight is given by the biggest local run (figure 11): the system decided to spread the test over a longer period, resulting in a lower TPM peak.

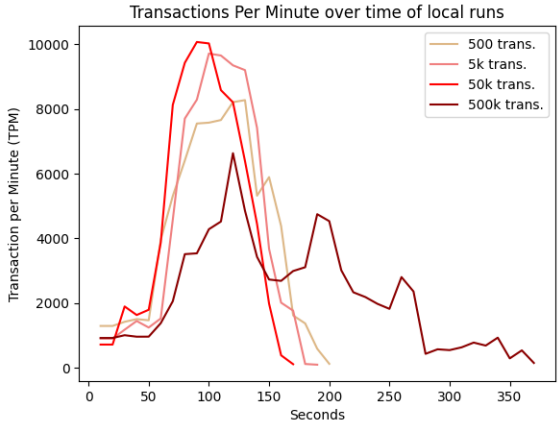

Figure 11: Average time series of the local runs

(a) Azure

(b) Google Cloud

Figure 12: Average time series of Cloud runs

Finally, table 7 summarizes important data for the cost analysis. The conversion rate between US Dollars and Euros is set as 1 USD = 0.92 EUR.

Provider	**Tot. exec. trans.**	**Tot. cost**	**Cost per 10,000 trans.**
Google Cloud	2,354,322	11.27€	0.047869€
Microsoft Azure	3,002,478	20.53€	0.068377€

Table 7: Data on TPROC-C Cloud runs

7 Discussion and Conclusion

7.1 Transaction Analysis

The analyses done in the previous section confirm that the servers do not support enough IOPS. Although this may constitute a limit for the benchmark execution itself, it is not a problem for our analysis. As said in paragraph 5.1, we performed the benchmark to observe how the system reacts to a stressful environment; both Cloud servers were successful in that sense.

7.2 Cost Analysis

As already discussed, Google Cloud and Microsoft Azure impose dual charges on users for PostgreSQL servers, consisting of a fixed quota for each created server and a variable fee dependent on resource consumption. The latter is influenced by multiple factors, encompassing storage usage, the number of vCPUs, and the maximum allowed IOPS. To simplify our calculations, we approximate this variable cost as a **charge per transaction**. We pursued this decision as our project centers around an RDBMS rather than a Data Warehouse. Given that our daily ETL operations remove rides and payments from the system, the impact of storage on costs is low compared to transactions.

As already explained in paragraph 1.3, we want to underline again that the real costs are given by a multitude of factors, and that our analysis is based on an approximation. It is straightforward to conclude that the expenses presented here do not necessarily represent the exact costs the system will experience in a real-life environment.

Now we focus on pay-as-you-go costs, while later we will discuss fixed expenses. The monthly costs by season and by Cloud provider are computed as follows:

$$MVC = [(PARh \cdot PNh) + (OARh \cdot ONh) + (NARh \cdot NNh)] \cdot CPT \cdot 30 \cdot K$$

Where:

- **MVC**: Monthly Variable Cost;
- **PARh, OARh, NARh**: respectively Peak, Other daytime, and Nighttime average number of rides per hour;
- **PNh, ONh, NNh**: respectively Peak, Other daytime, and Nighttime hours per day;
- **CPT**: cost per transaction;
- **30**: number of days in a commercial month;
- **K**: coefficient which determines the number of transactions in the daily operations performed on the database. For each ride, both the `Payments` and `Rides` tables are updated. Daily ETL copy and delete information from both tables. Also, before and after a ride the status of the driver is updated. The other operations are called depending on external factors. We can then safely assume that K is a value between 8 and 15.

This is the metric used to compare Google Cloud SQL and Microsoft Azure's PostgreSQL Servers.

The **final results** are summarized in Table 9, alongside the fixed costs. The column Zone specifies whether geo-redundancy in the case of a regional outage or disaster is enabled ("Multi") or not ("Single"). The table does not contain data about Azure PostgreSQL Servers without geo-redundancy enabled since Microsoft offers this feature without additional costs.

Provider	Season	Zone	Fixed Costs	Variable Costs	Total cost
Google	Touristic	Single	∼54€/mo	∼5.05€/mo	∼59€/mo
Google	Non-Touristic	Single	∼54€/mo	∼1.47€/mo	∼55€/mo
Google	Touristic	Multi	∼108€/mo	∼5.05€/mo	∼113€/mo
Google	Non-Touristic	Multi	∼108€/mo	∼1.47€/mo	∼109€/mo
Azure	Touristic	Multi	∼18€/mo	∼7.21€/mo	∼25€/mo
Azure	Non-Touristic	Multi	∼18€/mo	∼2.09€/mo	∼20€/mo

Table 8: Final monthly costs rate (K=8)

Provider	Season	Zone	Fixed Costs	Variable Costs	Total cost
Google	Touristic	Single	∼54€/mo	∼9.47€/mo	∼63€/mo
Google	Non-Touristic	Single	∼54€/mo	∼2.76€/mo	∼57€/mo
Google	Touristic	Multi	∼108€/mo	∼9.47€/mo	∼117€/mo
Google	Non-Touristic	Multi	∼108€/mo	∼2.76€/mo	∼111€/mo
Azure	Touristic	Multi	∼18€/mo	∼14€/mo	∼32€/mo
Azure	Non-Touristic	Multi	∼18€/mo	∼3.92€/mo	∼22€/mo

Table 9: Final monthly costs rate (K=15)

7.3 Conclusion and further research

Both Google Cloud and Microsoft Azure offer similar solutions for hosting a PostgreSQL server, providing cost efficiency, scalability, and enhanced security without the need for significant upfront investments.

Our cost analysis proved **Azure's Flexible PostgreSQL Servers** to be the most convenient choice for Taxi Lecco, mainly due to the small number of transactions processed by the OLTP system. The result still holds even if the estimated number of transactions is significantly higher than the actual one. The organization could be able to obtain Cloud's advantages - deeply explained in Chapter 2 - with as little as 22€ per month.

Further research may include analyzing the costs associated with the inclusion of cyclical and frequent spatial queries to inform taxi drivers on areas with high ride demand.

References

[1] Azure database for postgresql - flexible server. https://learn.microsoft.com/en-us/azure/postgresql/flexible-server/overview.

[2] Cyber guidance for small businesses. https://www.cisa.gov/cyber-guidance-small-businesses.

[3] Database migration service. https://cloud.google.com/database-migration?hl=en, urldate = 13/12/2023.

[4] Google cloud sql. https://cloud.google.com/sql?hl=en, urldate = 13/12/2023.

[5] Google spanner. https://en.wikipedia.org/wiki/Spanner_(database), urldate = 13/12/2023.

[6] Hammerdb documentation. https://www.hammerdb.com/docs/ch03s02.html.

[7] History of cloud computing. https://www.geeksforgeeks.org/history-of-cloud-computing/.

[8] Microsoft azure. https://en.wikipedia.org/wiki/Microsoft_Azure.

[9] Postgresql documentation. https://www.postgresql.org/about/.

[10] Small businesses are more frequent targets of cyberattacks than larger companies. https://www.forbes.com/sites/edwardsegal/2022/03/30/cyber-criminals/?sh=14e854ed52ae.

[11] Tpc-c benchmark. https://www.tpc.org/tpcc/default5.asp.

[12] What is cloud computing? a beginner's guide. https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-cloud-computing.

[13] David Bermbach, Erik Wittern, and Stefan Tai. *Cloud Service Benchmarking*. Springer, 2017.

[14] Barcelo E. Erl, T. *Cloud Computing: Concepts, Technology, Security and Archtecture*. Pearson, 2023.

[15] Loukides M. The cloud in 2021: Adoption continues. https://www.oreilly.com/radar/the-cloud-in-2021-adoption-continues/.

[16] SRG Research. Quarterly cloud market once again grows by $10 billion from 2022, meanwhile little change at the top, 2022.