# A Study Of RDBMS Performance on Cloud Platforms

**Midterm Project : CSCE 678**

Pranay Dhariwal (UIN: 128001762)
Sowmya Gaja Kumar (UIN: 526009328)

## Contents

# Motivation

The current report is a study of the three of the leading Cloud platforms, namely, Amazon Web Services, Microsoft Azure and Google Cloud Platform. The metric we have used for comparison is the performance of each of these platforms with respect to relational databases (MySQL and PostgreSQL).

Today, our industry is heavily data driven. The world of data is constantly evolving every passing second. Every day we generate a mind-boggling 2.5 quintillion bytes of data. Of all the data in the world, 90% of it was generated in the last two years. This evolution calls for efficient management of data. By collecting seemingly insignificant data, companies are able to address a multitude of challenges that they are facing.

When we are presented with structured data which needs to be consistent and has multiple relations, RDBMS is the most effective solution.  Database management is critical to an application and it requires a specialized team of engineers to maintain and update these complex resources. The use of cloud to maintain such a database solves the above problem and provides advantages like better collaboration and communication, greater flexibility, cost effectiveness, data protection etc.

The largest providers for RDBMS cloud services are Amazon, Microsoft and Google. The existence of multiple vendors has made choosing the right cloud provider for a given task an increasingly nuanced discussion that goes well beyond scale. Our report discusses the performance of RDBMS (specifically MySQL and PostgreSQL) on each of these platforms. Identifying the crucial parameters for this evaluation is a critical step and multitude of parameters present makes this decision a daunting task.

| | | |
|---|---|---|
| **AWS** | **Azure** | **GCP** |

## Methodology

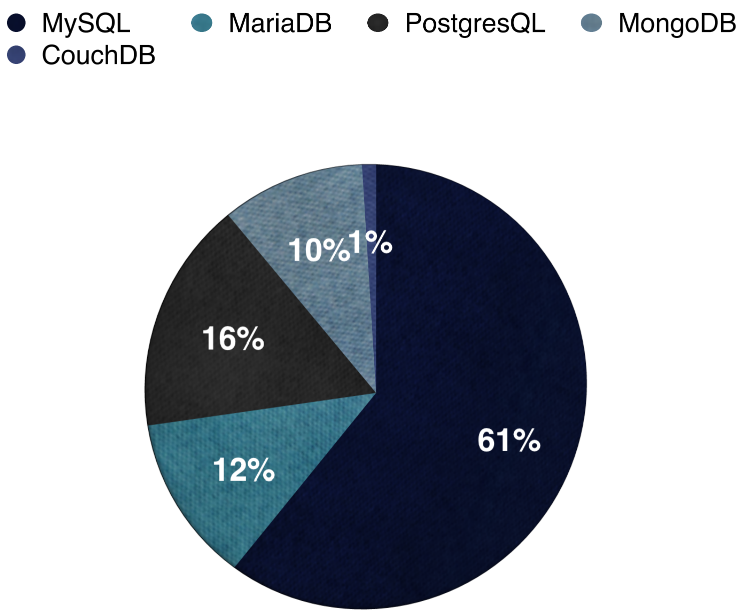
To evaluate a cloud relational database, we first tried to identify the issues that effect the performance and measure the performance in terms of response time. For the first part of the problem we did a systematic literature review.

When evaluating the services provided by each of these cloud platforms, there are numerous factors one should take into account. In our report we have taken into consideration the following metrics:

- Time taken to establish a connection.
- Query execution times.

There is a possibility that running an experiment might not always lead to the same performance. This could be due to various reasons like network latency, failed node, network connection problem, increased load on a node being used etc. Hence we run a given experiment multiple times and calculate statistical parameters like mean, variance, confidence intervals to obtain uniform results as this would help us minimize the effects of outliers.

Our experiment is divided into multiple use cases. We have considered inputs of varying length to see the variation in each of these cases. Each experiment is run twenty (20) times. The value obtained in each case is recorded and we calculate the mean and confidence intervals on each of the platforms.

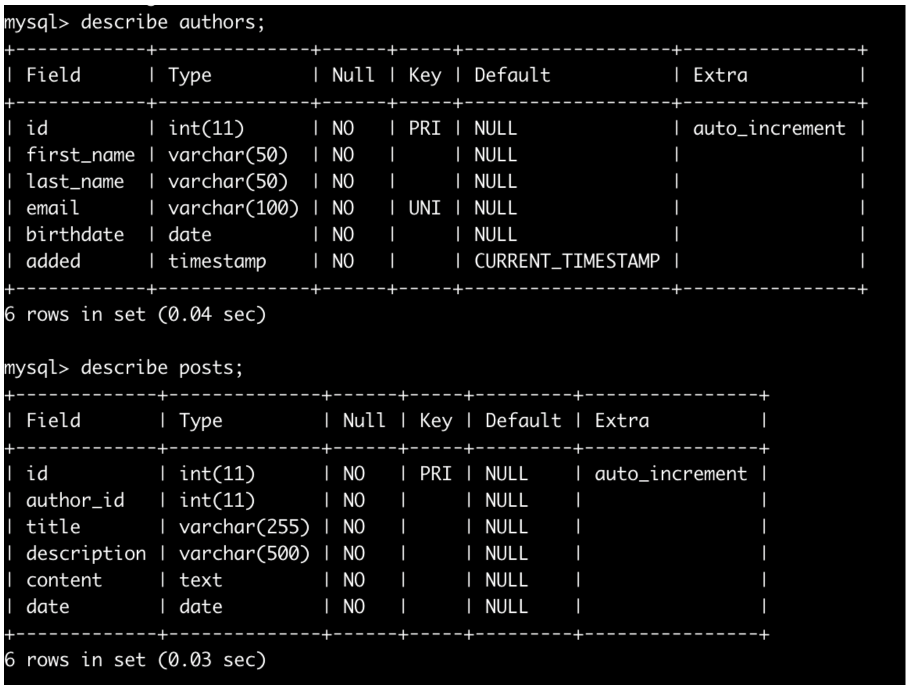


The next step involved deciding which databases to evaluate. For this, we tried to understand the marketshare of each of the databases. As we can see that MySQL is the biggest player with a stake of 61%, followed by PostgreSQL with 16%.

Query selection played an important role in our decision making process. When executing a JOIN in SQL, there is a tremendous amount of overhead because the system must pull data from different tables and align them with keys. When data is retrieved, it pulls all of the key-value pairs at the same time. This lead us to chose a join query as one of our experimental use cases.

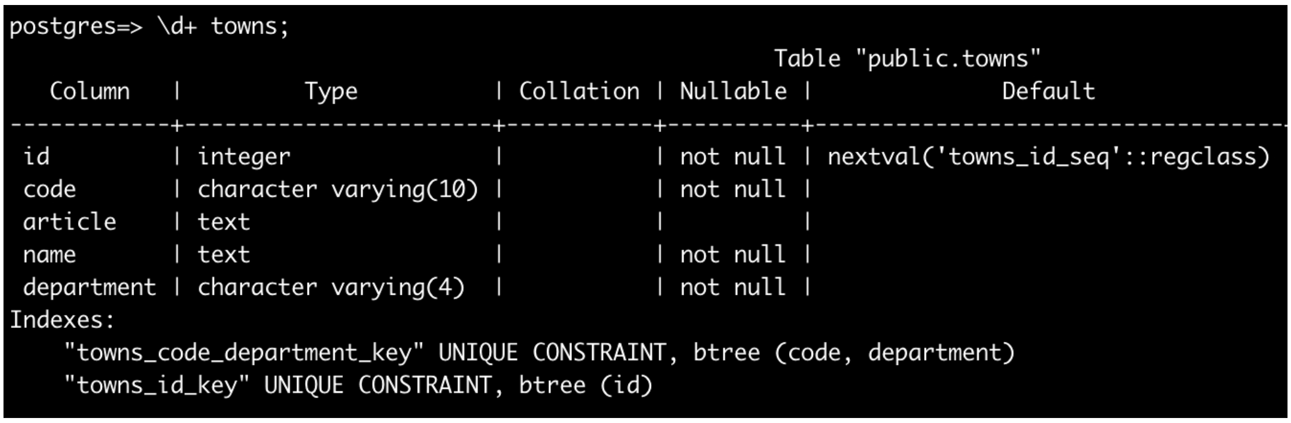
First, we start our experiment by computing the estimated time for a connection establishment for which we use Python3 as our scripting language. The below table and images summarize the environment configuration on the platforms on which we conducted our experiments and has the schema description respectively.

| Configuration | Amazon Web Services | Microsoft Azure | Google Cloud Performce |
|---|---|---|---|
| Version | 5.6.40 | 5.6.7 | 5.7 |
| PostgreSQL Version | 10.6 | 10 | 9.6 |
| vCore | 4 | 4 | 1 |
| Storage | 20 GB | 100 GB | 10 GB |
| RAM | 4 GB | 5GB | 3.75 GB |
| Location | US East | South Central US | Central US |



```
mysql> describe authors;
+------------+--------------+------+-----+-------------------+----------------+
| Field      | Type         | Null | Key | Default           | Extra          |
+------------+--------------+------+-----+-------------------+----------------+
| id         | int(11)      | NO   | PRI | NULL              | auto_increment |
| first_name | varchar(50)  | NO   |     | NULL              |                |
| last_name  | varchar(50)  | NO   |     | NULL              |                |
| email      | varchar(100) | NO   | UNI | NULL              |                |
| birthdate  | date         | NO   |     | NULL              |                |
| added      | timestamp    | NO   |     | CURRENT_TIMESTAMP |                |
+------------+--------------+------+-----+-------------------+----------------+
6 rows in set (0.04 sec)

mysql> describe posts;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| id          | int(11)      | NO   | PRI | NULL    | auto_increment |
| author_id   | int(11)      | NO   |     | NULL    |                |
| title       | varchar(255) | NO   |     | NULL    |                |
| description | varchar(500) | NO   |     | NULL    |                |
| content     | text         | NO   |     | NULL    |                |
| date        | date         | NO   |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
6 rows in set (0.03 sec)
```

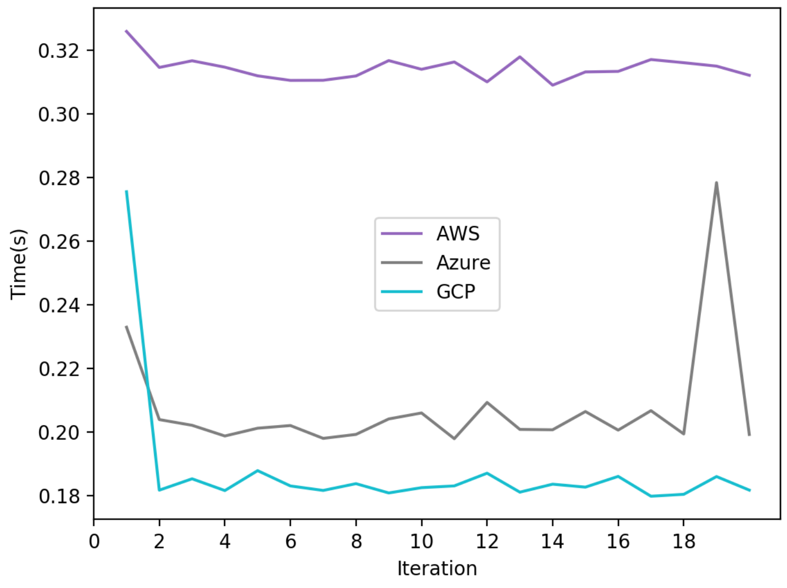


```
postgres=> \d+ towns;
                                                 Table "public.towns"
   Column   |          Type          | Collation | Nullable |                Default
------------+------------------------+-----------+----------+-----------------------------------
 id         | integer                |           | not null | nextval('towns_id_seq'::regclass)
 code       | character varying(10)  |           | not null |
 article    | text                   |           |          |
 name       | text                   |           | not null |
 department | character varying(4)   |           | not null |
Indexes:
    "towns_code_department_key" UNIQUE CONSTRAINT, btree (code, department)
    "towns_id_key" UNIQUE CONSTRAINT, btree (id)
```

# Results

## MySQL

**Metric One**: Time taken to establish the connection over twenty iterations.



| Platform | Confidence Interval (x̄ ± 1.725 * σ / square root(n)) |
|---|---|
| Amazon Web Services | 0.315 ± 0.0014 |
| Microsoft Azure | 0.2074 ± 0.0068 |
| Google Cloud Platform | 0.1878 ± 0.0078 |

**Metric Two** : Time taken to execute queries over twenty iterations.
We used two queries to measure this metric, the first one involves a join and the second one is a simple select query. The queries are listed as below:

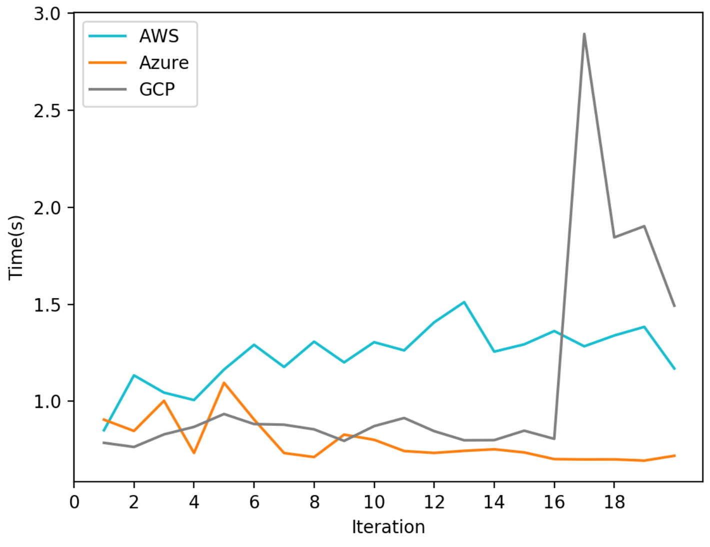
**Query 1:**

```
SELECT * FROM authors WHERE authors.id IN (SELECT posts.author_id FROM posts)
```

**Query 2:**

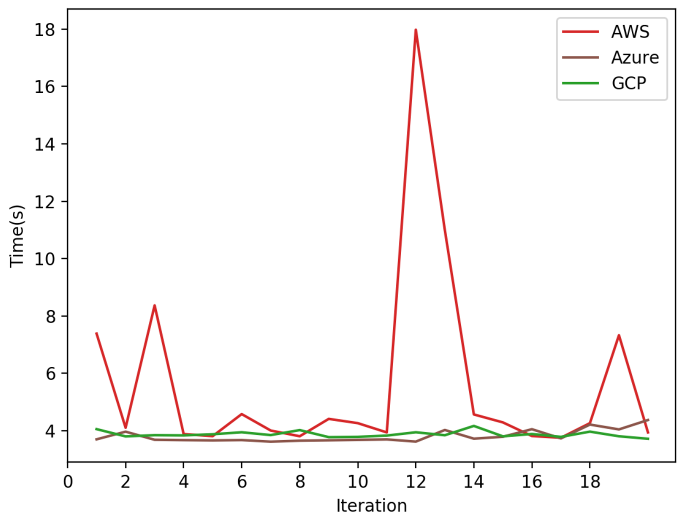
```
SELECT * FROM posts; (50000 rows)
```

Query 1



| Platform | Confidence Interval (x̄ ± 1.725 * σ / square root(n)) |
|---|---|
| Amazon Web Services | 1.2372 ± 0.056 |
| Microsoft Azure | 0.789 ± 0.041 |
| Google Cloud Platform | 1.08 ± 0.205 |

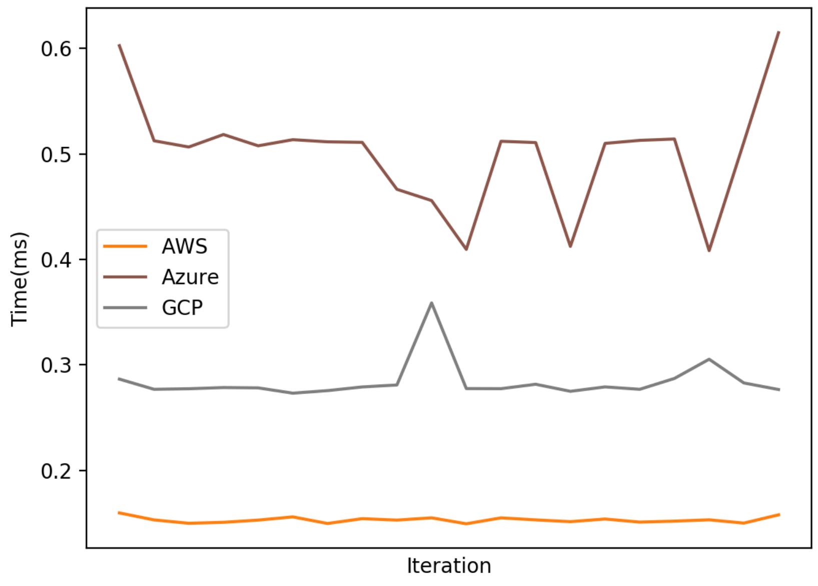
Query 2



| Platform | Confidence Interval (x̄ ± 1.725 * σ / square root(n)) |
|---|---|
| Amazon Web Services | 5.677 ± 1.213 |
| Microsoft Azure | 3.815 ± 0.082 |
| Google Cloud Platform | 3.88 ± 0.041 |

## PostgreSQL

**Metric One:** Time taken to establish the connection over twenty iterations.

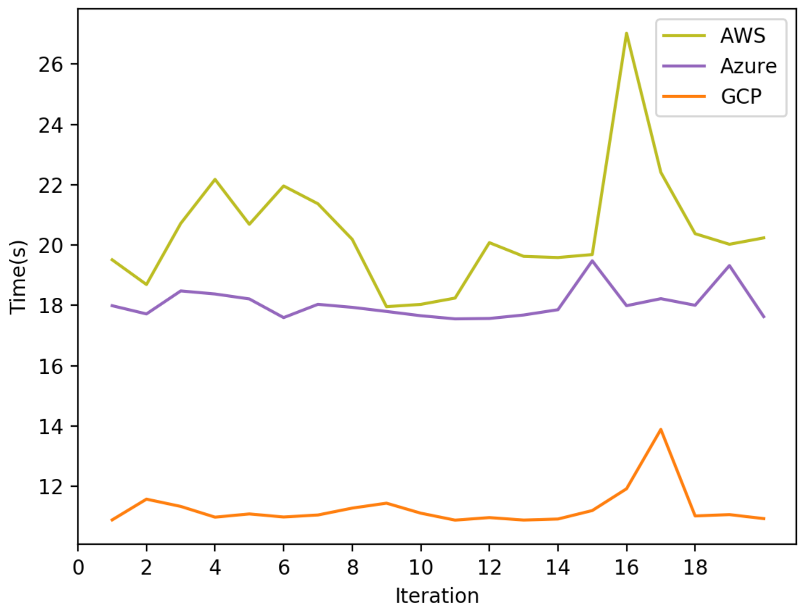


From the above values, we calculated the Average Mean Time and Confidence Interval on each of the platform.

| Platform | Confidence Interval (x̄ ± 1.725 * σ / square root(n)) |
|---|---|
| Amazon Web Services | 0.152 ± 0.001 |
| Microsoft Azure | 0.501 ± 0.019 |
| Google Cloud Platform | 0.284 ± 0.007 |

**Metric Two:** Time taken to execute a select query over twenty iterations.
In the second metric we tried to run the 'select * ' query which displays all the data from the table which has one million rows. The below depicts time taken over twenty iterations and the confidence. The table is a representation of the confidence intervals on each of the platforms.

| Platform | Confidence Interval (x̄ ± 1.725 * σ / square root(n)) |
|---|---|
| Amazon Web Services | 20.4306 ± 0.754 |
| Microsoft Azure | 18.056 ± 0.2 |
| Google Cloud Platform | 11.27 ± 0.252 |



To explain the results obtained we present the possible reasons:

1. With respect to network latency and connectivity, we found out that GCP has the most robust network in the US East region where our experiment was conducted. This could be a factor GCP to outperform other platforms in terms of network latency.

2. To optimize the performance of a join query in a distributed system, three major factors could play a quintessential role. Size of transmitted data, transmission speed and local processing cost. The provider that has the best performance would probably be optimizing on local processing cost to outperform the others.

3. Also, when considering complex join queries, the order of joins also plays an important role. The determination of optimal order could have a large effect on the performance.

## Discussions

In our experiments we have compared the MySQL and PostgreSQL offerings present on all the three platforms. The summary of the results obtained is as follows:

| Feature | Amazon Web Services | Microsoft Azure | Google Cloud Platform |
|---|---|---|---|
| **MySQL connection time** | Third | Second | First |
| **MySQL execution times** | Third | First | Second |
| **PostgreSQL connection time** | First | Third | Second |
| **PostgreSQL execution times** | Third | Second | First |
| **Data import and export** | Supports all technologies like dump and load. | Supports all technologies like dump and load. | Supports all technologies like dump and load. |
| **Failed replicas** | Used to increase availability | Used to increase availability | Can be used as read replicas. |
| **Backup** | Data is backed up every day | Data is backed up every five minutes | Data is backed up every day |
| **Market share** | 62% | 20% | 12% |
| **SLA guaranteed 10% Service Credit** | <= 99.95% | <= 99.99% | <= 99.95% |

This comparison leads us to conclude the following results:
The pros of using **AWS** is that it gives more options to create a DB instance such as memory optimized. The disaster recovery system is one of the best as they have cross region read replicas. The cons of using AWS are that query joins are not optimized. Also, the pricing is high when compared to other cloud providers. In **Azure**, query join is very optimized. High availability and frequent automatic backups makes Azure a reliable provider. However,  it has zero or few administrative tasks. In **GCP**, the servers are optimized so the time taken to establish a DB connection from public Internet is minimum when compared to the others. Auto scaling feature is key advantage of using GCP. If storage has reached the threshold then it automatically scales up. The downside of using GCP is that there is performance impact due to semi synchronous replication.

In conclusion, there is no clear winner in this battle between AWS, Google, and Azure . Being in a competitive market has led all the cloud service providers to attract customers with the extended services at reduced prices.

## References

- https://d36cz9buwru1tt.cloudfront.net/AWS_RDBMS_MS_SQLServer.pdf
- AWS: https://aws.amazon.com/free/
- Azure: https://azure.microsoft.com/en-us/free/
- GCP: https://cloud.google.com/free/