COMPARATIVE ANALYSIS FOR SELECTING SOLID DATABASE FOR YOUR RELATIONAL DATA REQUIREMENTS*

BY

APURVA KANDELKAR♣

Assistant Professor, School of Computer Science and Engineering, Ajeenkya D Y Patil University, Pune, India

apurvakandelkar@gmail.com

KIRAN RAJ KG

Student, School of Computer Science and Engineering, Ajeenkya D Y Patil University, Pune

kkiranraj186@gmail.com

ABSTRACT

There are various types of databases available, and they are written in a number of languages. Relational databases excel at storing data in tables of strict schemas that are linked to one another by a common column, such as an id that relates to records about just one document. In this paper we will discuss about databases and why we ended up choosing postgres for our requirements.

KEYWORDS

Postgres, Database, SQL, Non-SQL.

1. Introduction

A rigid schema demands that each record contain the same information and that each column only contain the same specified data type. Non-Relational Databases, such as MongoDB, are ideal for storing data that lacks a rigid schema or fixed format, such as a never-ending chat log. After deciding the type of database needed, you will proceed to selecting a database management system. PostgreSQL is a database management framework that uses the Structured Query Language (SQL). It is common because it is open-source and can be customized according to our purposes. PostgreSQL can be accessed through a variety of paid and free Graphical User Interfaces (GUIs), as well as the command line and terminal.

* Received 22 September 2021, Accepted 09 October 2021, Published 24 October 2021
♣ Corresponding Author

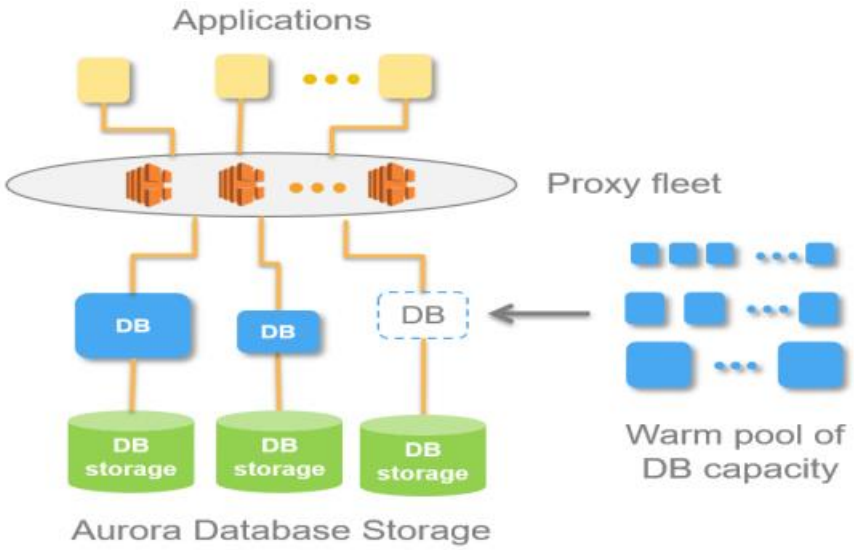

Fig. 1. Image shows the Aurora Database internal Architecture

2. Literature Review

2.1 The accurate considerations for selecting an relational database.

There are several databases available, but most web applications built using an open-source web architecture use PostgreSQL or MySQL. Many clients have asked me why they should use PostgreSQL instead of MySQL for their projects over the years. Data forms are enforced by PostgreSQL. For eg, if you build a stringcolumn in MySQL with a cap of 200 characters and attempt to insert 300 characters, PostgreSQL will raise an exception and your app will crash, while MySQL will insert the first 200 characters and quietly remove the last 100.

Migrations, If a bug occurs when updating the database in PostgreSQL, the entire update is rolled back to where you began. In MySQL, it simply fails at some point it got to (say, halfway through editing your table), and you must find out how to patch it when your development app is down and everybody is crying (I've been there... never again!).

Full text (word) search is available in PostgreSQL. Full text search works really well straight out of the box. But you can load PostgreSQL with a large number of, say, product descriptions and easily browse for any of them without downloading anything else.

UUIDs as primary keys. Instead of /product/1, /product/2, and so on, you now have /product/3c892309-ee0d-49c9-addb-9564c3c60eea. This is advantageous because it makes it more difficult to hack anything by attempting to guess such an ID.

Extra data forms in the database, such as JSON. PostgreSQL will store a JSON object and then index on that JSON object, rendering queries super quick and effectively turning PostgreSQL

into a NoSQL database. It also supports BigInt, Decimal, and a number of other PostgreSQL native data types.

PostgreSQL is still ACID compatible this means that if you write to the client, it will either work or it will not, with no in-between "Yes, it worked, but it didn't write to disc." MySQL is only compatible under specific conditions and for specific table formats.

The PostgreSQL group owns PostgreSQL. The coding is not owned by a company. Oracle owns MySQL, and OracleDB, a rival to MySQL, is their main source of revenue. I don't see them making it (a free product) better than OracleDB. PostgreSQL, frequently competes with OracleDB deployments from EnterpriseDB, a privately owned business that sells corporate-level support.

PostgreSQL follows more SQL requirements than MySQL, making it easier to debug and use.

Materialised viewpoints are supported by PostgreSQL, and they are AMAZING! Assume you have a goods table that you are always querying all products that are linked to orders... a dynamic query which takes few seconds to run each time What you should do is create a view of this query and then "save" it as a materialised view, which functions as a live updated dynamic table from which you can query at Very fast speeds. Very useful for huge size databases.

You can programme within the PostgreSQL database using Ruby and a variety of other languages, which MySQL does not allow.

2.2 Postgres Performance

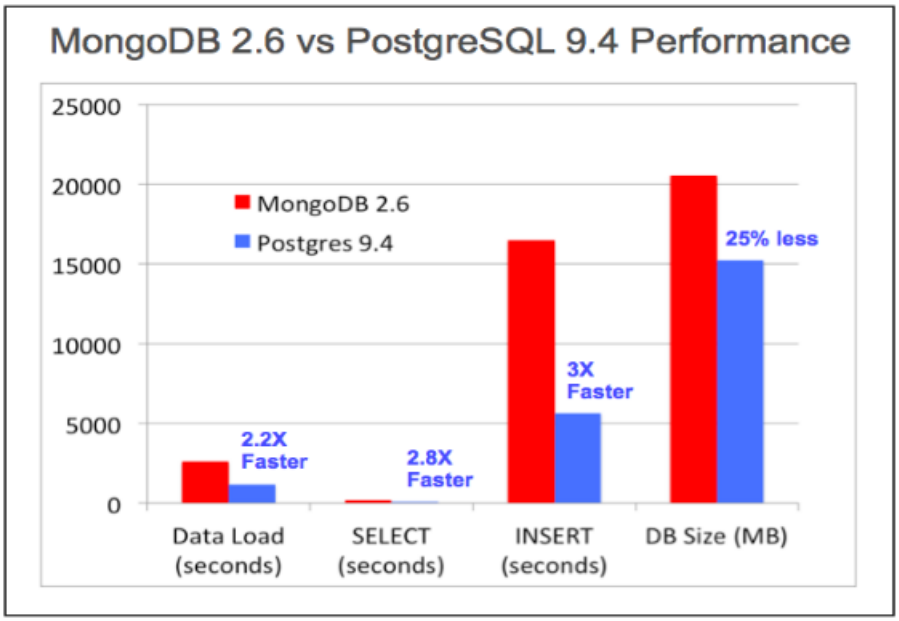

Fig. 2. Performance of postgres and mongodb. It's clear that postgres is faster than mongodb.

Table 1. Postgres vs MySql.

Feature	PostgreSQL	MySQL
Open Source	Yes	Yes
ACID	Yes	Partially
SQL Standard	Yes	Partially
High Performance	Yes	Read only
Community	Yes	Yes

3 Technology Where We Host the Database

The main incentive to deploy (run) your database on EC2 is to save money. As of the time of writing, the cheapest/smallest RDS implementation costs about $30.88 USD per month. In comparison, a t2.micro EC2 instance with 8GB of storage costs about $6 USD per month. This expense will be reduced even more if you want reserved billing.

As a general rule, if your budget makes, always go for RDS. It manages automated fail-over and replication through a standby instance in a separate availability region, Following are some of the best reasons behind it:

Amazon promises domains, not availability districts, 99.99999 percent uptime. Getting a backup replica in a separate region means that even if the main zone where the master database is stored goes down, the application will be able to fall back on the standby replica in a different (and presumably unaffected) region.

Our programme does not need to be able to detect and manage database failures. Amazon does this automatically. If/when the main database crashes; they can redirect your database host URL to the standby database. They will immediately meet up with and promote the restored master database until the crisis has been resolved.

Amazon Aurora is a RDS that incorporates high speed and high availability of high-end enterprise databases with the ease of access and low cost of open-source dbs. Aurora's PostgreSQL-compatible version has up to 3X throughput of regular PostgreSQL operating on the same mono-hardware, allowing current PostgreSQL programmes and tools to operate without change the combination of PostgreSQL database compatibility and Aurora business database capabilities makes Aurora an excellent option for commercial databases.

3 Architecture

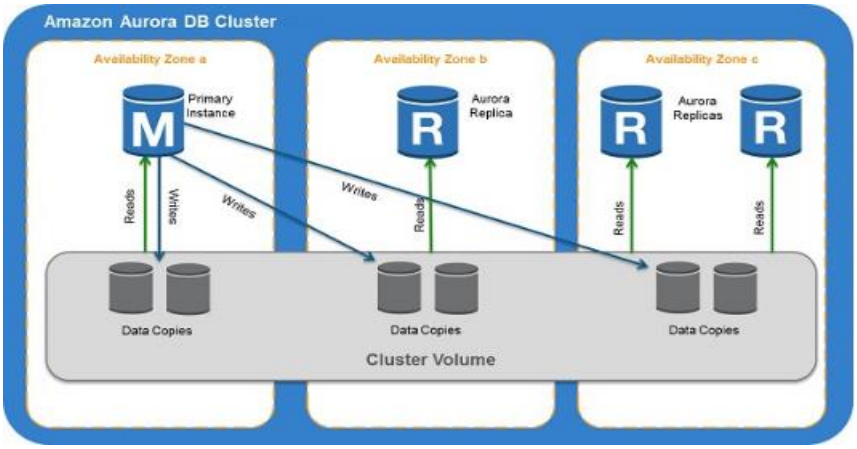

Fig. 3. The postgres High Availability Architecture using Amazon Aurora in 3 availability zones

An Amazon Aurora DB cluster is made up of one or multiple database instances and cluster volumes that handle data for that database. The volume of the aws aurora cluster is a virtual database storage volume that covers several Availability Zones. These have a copy of the DB cluster data in each Availability Zones. AWS Aurora Database clusters are made up of two kinds of DB instances:

Primary Database case Supports read & write operations and handles any data changes to the cluster volume. There is only one primary Database instance in each Aurora Database cluster.

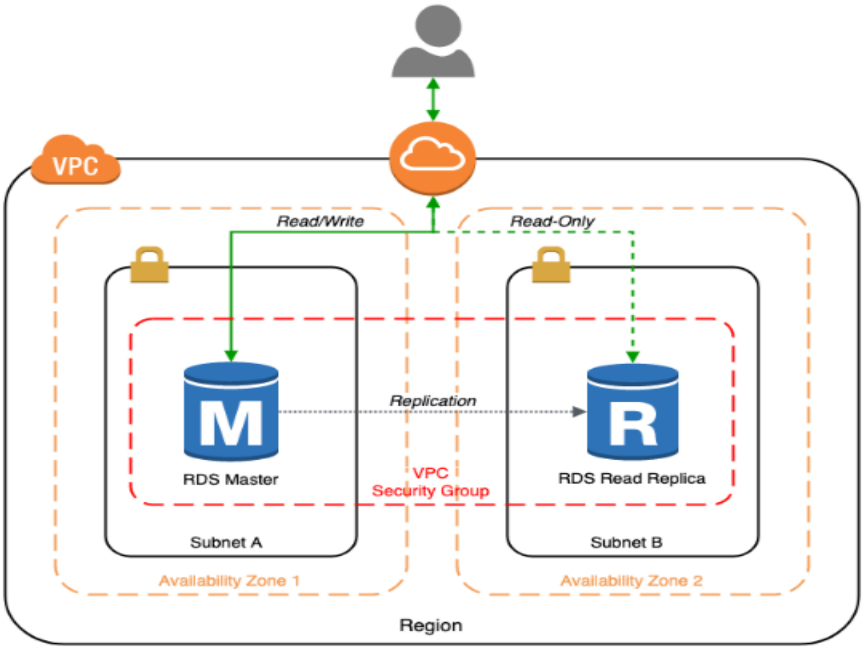

Fig. 4. The postgres High Availability Architecture using Amazon Aurora

Aurora Replica This instance connects to the same data volume as the main DB instance and only supports read operations. In addition to the primary DB case, each Aurora DB cluster may

have up to 15 Aurora Replicas. Keep Aurora Replicas in different Availability Zones to ensure high availability. In the event that the primary DB instance becomes inaccessible, Aurora immediately switches to an Aurora Replica. The failover priority for Aurora Replicas can be set. Read workloads from the primary DB instance may also be offloaded by Aurora Replicas.

Both DB instances of Aurora multi-master clusters can read and write. The distinction between primary instance and Aurora Replica does not matter in this situation. We apply to these writer and reader DB instances when addressing replication topology where clusters may use either single-master or multi-master replication.

3Results

Up to three times the throughput of PostgreSQL Normal benchmarks such as SysBench have revealed up to a threefold improvement in throughput efficiency over stock PostgreSQL 9.6 on comparable hardware. Amazon Aurora employs a number of software and hardware strategies to ensure that the database engine can best use usable compute, memory, and networking resources. To boost output efficiency, I/O operations use distributed systems strategies such as quorums. You can scale the compute and memory resources running your deployment up and down using the Amazon RDS APIs or with a few clicks in the AWS Management Console. Compute scaling operations normally take a few minutes to complete. As your database storage requirements expand, Amazon Aurora can automatically increase the size of your database volume. Your amount will rise in 10 GB increments up to a limit of 128 TB. You don't need to add extra capacity to your database to accommodate potential expansion.

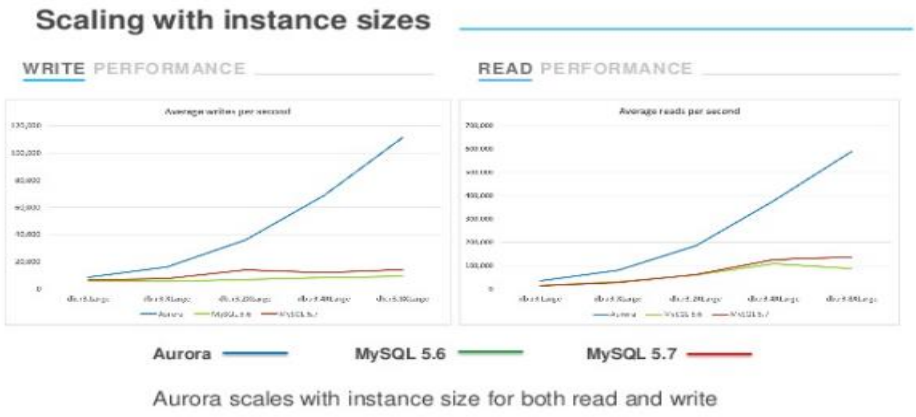

Fig. 5. Aurora Scales with instance sizes

Build up to 15 database read replicas to improve read throughput and accommodate high-volume application requests. Amazon Aurora Replicas use the same underlying storage as the source case, reducing costs and removing the need for replica nodes to execute writes. This frees up more computing resources to serve read requests and decreases replica latency time –

which is mostly in the single digit milliseconds. Amazon.com Aurora Serverless is an on-demand, auto-scaling Aurora configuration in which the database can dynamically start-up, shut down, and scale up or down power depending on the needs of the programme.

With Aurora Serverless, you can run your database in the cloud without having to handle any database instances Amazon RDS tracks the fitness of your Amazon Aurora database and underlying EC2 instance in real time. In the case of a database outage, Amazon RDS will restore the database and related processes automatically. Amazon Aurora does not need the replay of database redo logs after crash recovery, which significantly reduces restart times.

Amazon Aurora also keeps the storage buffer cache separate from the database. Amazon Aurora uses RDS Multi-AZ technology to simplify failover to one of up to 15 Amazon Aurora Replicas you've built in either of three Availability Zones in the event of an instance failure. If no Amazon Aurora Replicas have been provisioned,

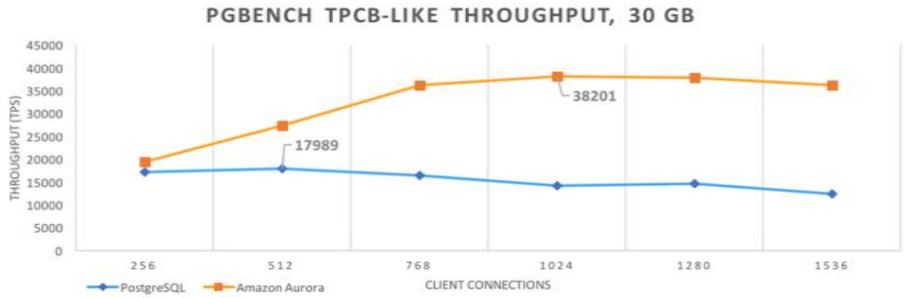

Fig. 6. Amazon Aurora gives us the highest through put than normal postgres

You can use Global Database for globally dispersed applications, where a single Aurora database can span several AWS regions to allow fast local reads and easy disaster recovery. Global Database replicates a database through several AWS Regions using storage-based replication, with typical latency of less than 1 second, If you need to rebound rapidly from a regional degradation or outage, you can use a secondary area as a backup alternative. In less than a minute, a database in a secondary area may be promoted to maximum read/write capability. Each ten-gigabyte chunk of the disc capacity is repeated six times through three Availability Zones. Amazon Aurora storage is fault-tolerant, addressing the failure of up to two copies of data transparently without impacting database write availability and up to three copies without affecting read availability. Amazon Aurora storage is self-healing as well. Data blocks and discs are constantly checked for faults and automatically substituted.

4. Future Scope

Add Redis Caching to the Architecture. 'Redis' is an abbreviation for Remote Dictionary Server. According to the official Redis website, Redis is an open-source (BSD licenced), in-memory

data structure store that can be used as a storage, cache, and message broker. AWS, on the other hand, describes Redis as a 'fast, open-source, in-memory key-value data store suitable for use as a database, cache, message broker, and queue. The term "in-memory data structure store" refers to an in-memory archive that can be used for device primary database and caching, as well as message broker and queue.

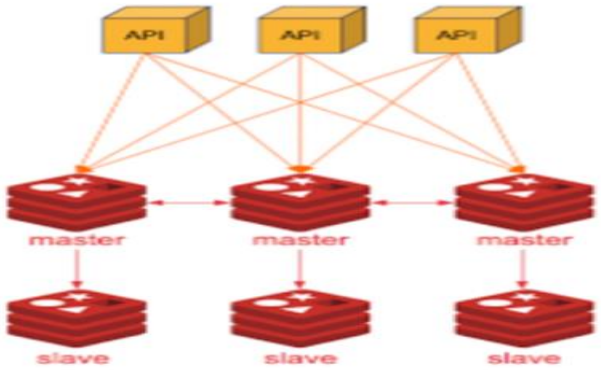

Fig. 7. Redis Multi-Master cluster setup to add on our architecture

An in-memory storage is one that holds the whole dataset in RAM. What does this imply? This ensures that if you query a database or edit data in a database, you are only accessing main memory. As a result, there is no Disk involved with these operations. And this is a positive thing since main memory is much faster than any disc. So we may conclude that Redis is a key-value storage system that allows one to conveniently store data that can be used for device databases, cashing, message brokers, and queues. It is also extremely easy to react due to the use of in-memory data storage mechanism.

Users can build high-performance chat and messaging services across all of their software and services since Redis supports the use of publish and subscribe (Pub/Sub) commands. This requires the ability to use list data structures to execute atomic operations and the ability to block. Redis can process data with sub-millisecond latency, making it suitable for real-time analytics, online advertisement campaigns, and AI-driven machine learning processes, Fast response database: since it stores data in memory rather than on a disc or solid-state drive (SSD), it responds faster to read and write operations than most.

5. Conclusion

Even on longer runs (2 hours), there is no difference in results. The throughput is consistent. After executing for 48 hours. There were still no changes. As expected, better storage results in higher throughput for Percona Server. 3000 IOPS outperforms Amazon Aurora, particularly in IO-intensive cases. Amazon Aurora does poorly for fewer data sets. When it comes to large datasizes, Aurora outperforms Percona Server (with general purpose SSD and provisioned SSD

2000IOPS volumes). It seems that Amazon Aurora does not benefit from increased memory throughput does not improve dramatically with limited datasizes. I believe it validates my hypothesis that Aurora has some kind of write-through cache that produces better results in IO-heavy workloads. Provisioned IO volumes have slightly better efficiency than general purpose volumes, despite being more costly. In terms of cost (when compared to provisioned IO volumes), 3000 IOPS is more cost effective (in this case, although the workload can differ) than 2000 IOPS, in the sense that it provides more throughput per dollar.

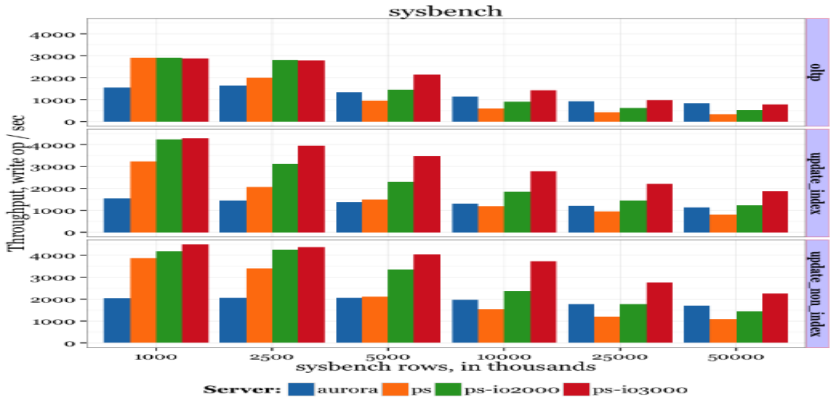

Fig. 9. Sysbench aurora performance results for updating index and non indexes

References

[1] Verbitski, A., Gupta, A., Saha, D., Brahmadesam, M., Gupta, K., Mittal, R., Krishnamurthy, S., Maurice, S., Kharatishvili, T. and Bao, X., 2017, May. Amazon aurora: Design considerations for high throughput cloud-native relational databases. In Proceedings of the 2017 ACM International Conference on Management of Data (pp. 1041-1052).

[2]https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/CHAP_AuroraOverview.html

[3] Verbitski, A., Gupta, A., Saha, D., Corey, J., Gupta, K., Brahmadesam, M., Mittal, R., Krishnamurthy, S., Maurice, S., Kharatishvilli, T. and Bao, X., 2018, May. Amazon aurora: On avoiding distributed consensus for i/os, commits, and membership changes. In Proceedings of the 2018 International Conference on Management of Data (pp. 789-796).

[4] https://aws.amazon.com/rds/aurora/faqs/

[5] https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Aurora.Overview.html

[6] https://aws.amazon.com/rds/aurora/postgresql-features/

[7] https://aws.amazon.com/blogs/database/tag/aurora/

[8]https://www.percona.com/blog/2018/07/17/when-should-i-use-amazon-aurora-and-when-should-i-use-rds-mysql/