BENCHMARK GCP SPANNER

WITH YCSB TOOLS

Accenture

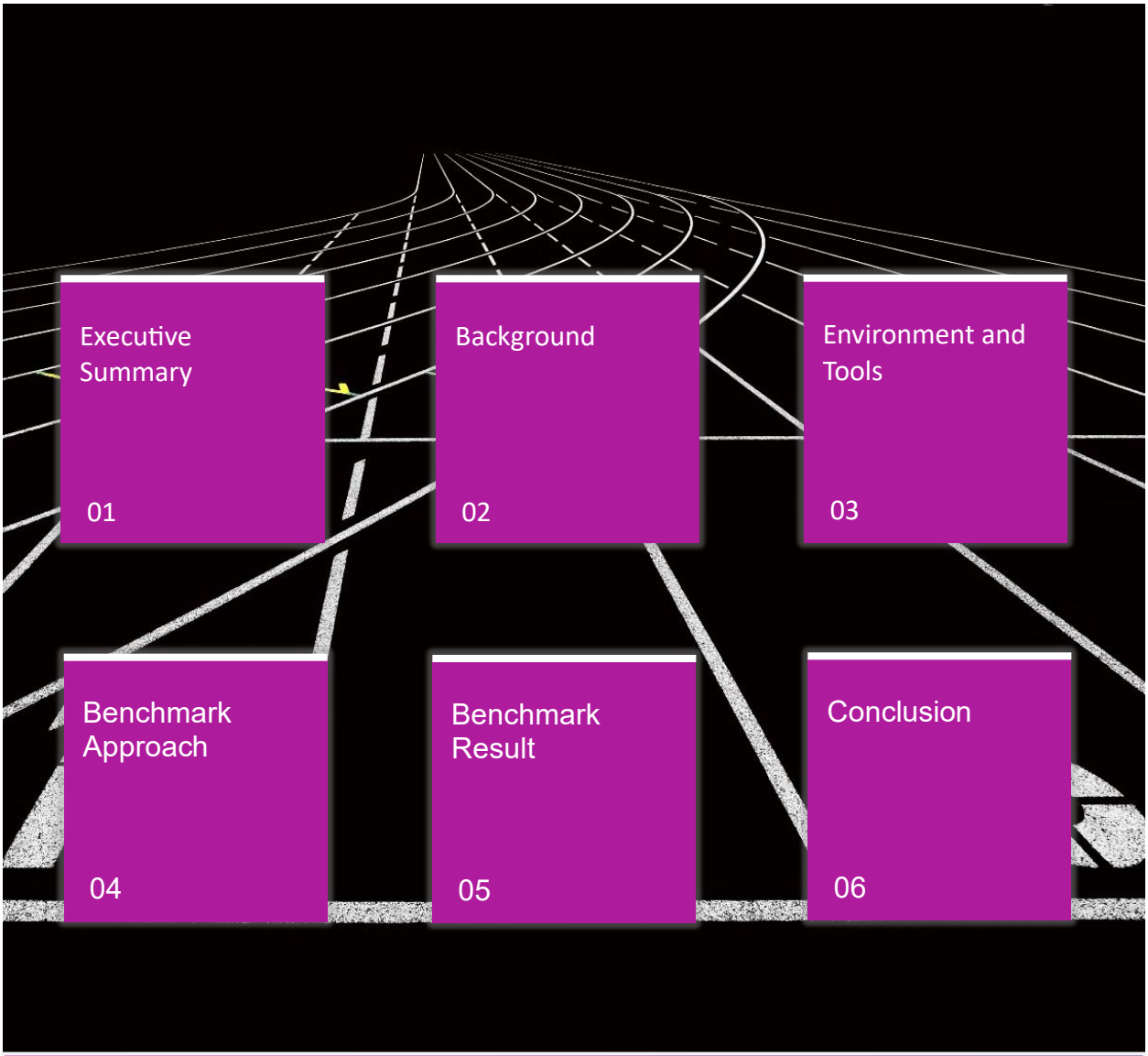

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Google Cloud Platform (GCP) and Amazon Web Services (AWS) are two leading cloud service providers. Both GCP Spanner and AWS DynamoDB are the two leading database offerings that are targeting large workload in response to the rise of data volume in the business community. Both GCP Spanner and AWS DynamoDB are designed for large distributed workloads with high availability, scalability and performance.

While both databases offer solutions for global data distribution, their underlying technologies are designed differently. In addition, GCP Spanner and AWS DynamoDB use different pricing models: the pricing on Spanner is based on compute instances and storage used while DynamoDB charges primarily based on Read/Write throughput and storage with a pay-per-request pricing model.

The objective of this benchmark is to compare the operation performance between Spanner and DynamoDB with the same costing spending. This benchmark remark can guide organizations in understanding the cost and performance comparison to help them to align with their specific technological requirements and business goals.

The benchmark utilizes open-source benchmark tool, Yahoo! Cloud Serving Benchmark (YCSB), to perform the evaluation and comparison of both databases. The evaluation was performed using a regional Spanner instance with three nodes, managing a dataset of 1,000,000,000 rows, or approximately 1 TB of data. The costing information associated with these workloads on Spanner were used to run comparable YCSB jobs on DynamoDB. The execution result, along with the recommendations, are detailed in this report.

INTRODUCTION

This report presents the benchmarking analysis between two fully managed database services: Google Spanner and Amazon DynamoDB. In an era where data is not just an asset but the backbone of business operations, selecting a database solution that offers both scalability and cost effectiveness without compromising on performance is critical. Through a series of methodically designed tests, we evaluate the intricacies of each service's performance, from transactional throughput to total cost under different conditions. By employing the YCSB framework, we aim to provide an objective comparison of Spanner and DynamoDB, offering insights that will guide businesses in choosing the database service that best fits their unique requirements.

GCP Spanner

Spanner is a fully managed database service for both relational and non-relational workloads that offers strong consistency at global scale, high performance at virtually unlimited scale, and high availability with an up to five 9s SLA.

While Spanner is renowned for its relational capabilities, it is also a versatile key-value database that can be used to store and retrieve non-relational data via read and write APIs. According to Google's blog post, a significant portion of internal Spanner usage at Google is non-relational, including applications like Google Photos, Google Ads, and Gmail.

The key features for GCP Spanner include:

GLOBAL SCALE AND DISTRIBUTION

Spanner offers the consistency of a single-machine database for global database footprint. It enables applications to read and write data across the globe with single digit milliseconds latency.

STRONG TRANSACTIONAL CONSISTENCY

Unlike many NoSQL databases that offer eventual consistency, Spanner ensures that all users view the same data at the same time for critical business applications using TrueTime, external consistency, and multi-version concurrency control.

FULLY MANAGED SERVICE

Spanner is always up with no maintenance windows or planned downtime ever with seamless sharding and replication across zones and regions.

WRITE AND READ SCALABILITY WITH NO LIMITS

Spanner is used extensively at Google by numerous projects, such as Google Photos, Google Ads, and Gmail. In total, Spanner serves over 3 billion read/write requests per second at peak.

MIX ANALYTICS WITH YOUR KEY VALUE WORKLOADS

Spanner Data Boost lets you run analytical queries or batch processing jobs without affecting the underlying transactional workload.

Pricing Model:
Spanner's pricing model is designed to offer flexibility and transparency, allowing organizations to scale their costs with their usage. The pricing for Spanner is primarily around three aspects:

1. **Compute Cost:** Spanner charges the compute capacity, measuring in number of nodes provisioned or processing units. 1 node is equal to 1,000 processing unit. This cost is calculated on an hourly basis, enabling organizations to scale up or down as the workload changes.

2. **Storage Cost:** It refers the cost of the amount of storage your database consumes, which calculates for the average amount of data in the Spanner databases, including tables, secondary indexes, backups, and metadata over a one-month period, multiplied by the monthly rate.

3. **Network Usage:** Spanner also charges for the network bandwidth used, particularly for data read from or written to the database from outside the Spanner region. There are no network usage charges for network ingress traffic or Spanner replication.

This model ensures that organizations pay for exactly what they use, providing a cost-effective solution for databases that demand high availability, global distribution, and horizontal scalability.

AWS DynamoDB

AWS DynamoDB is a fast and flexible NoSQL database service targeting high performance workload at any scale. It offers a robust, fully managed NoSQL database service to assist users with database workloads. It is engineered to provide a fast distributed database with high performance, availability, and scalability.

AWS DynamoDB is a key-value database that delivers low latency performance at any scale. It's a perfect fit for mobile, web, gaming, ad tech, IoT, and many other applications that require low-latency data access at scale.

The key features for AWS DynamoDB include:

- **Performance at scale:** DynamoDB can handle more than a large number of requests per day and it is designed for massive scalability.
- **Fully managed:** As a fully managed service, it handles the operational complexity and route maintenance, provides a scaling mechanism as the data and traffic grow.
- **Key-value and document data models:** The support of document and key-value data models provides flexible schema so each item can have many different attributes.
- **Auto partitioning:** As data volume grows, DynamoDB uses its automatic partitioning feature to spread data across partitions and increase throughput without manual intervention.

Pricing Model:
Like Spanner, AWS DynamoDB also separates the costs for compute, storage and network.

1. **Compute Cost:** AWS DynamoDB offers two pricing models to help users to choose the appropriate pricing option for their workloads.
    - **On-demand mode:** it allows users to accommodate the workloads as it ramps up. On-demand can have a significant impact in the consumption cost.
    - **Provisioned mode:** It allows users to set read and write capacity and it is more cost effective compared with on-demand mode.

2. **Storage Cost:** It refers to the cost of the amount of storage the database consumes, which calculates for the average amount of data in the DynamoDB databases, including tables, secondary indexes, backups, and metadata over a one-month period, multiplied by the monthly rate.

3. **Network Usage:** While data transfer into DynamoDB is free, AWS charges for data transferred out of DynamoDB to the internet or other AWS regions.

As we delve into the comparative benchmarking of Spanner and DynamoDB, it is our goal to provide a comprehensive overview on how to compare the cost effectiveness from two key cloud database offerings from two leading cloud providers. The subsequent sections will detail the testing environment, methodology, and a detailed discussion of the benchmark results, leading to the conclusion. By the end of this report, readers are able to gain in-depth knowledge about how these database services can be optimized for their specific use cases, ensuring that their workload can be benefited from understanding the database performance from heavy workload.

ENVIRONMENT AND TOOLS

In this benchmark, we executed our tests in our GCP and AWS lab environment, utilizing a Spanner cluster to test GCP's capabilities and corresponding resources on AWS for DynamoDB. The benchmarking was facilitated by a VM configured with the YCSB tool. By using YCSB benchmarking tool, we were able to evaluate each database's performance and cost expense for YCSB workloads.

Environment on GCP

Spanner Cluster

Spanner is a distributed database and is designed to excel in a multi-node environment with fault tolerance and high availability. In the benchmark, we choose a 3-node configuration of Spanner to allow us to observe how the database handles the load across a Spanner cluster, which mirrors a real-world scenario that requires multiple nodes setup. To isolate the potential impact from replication to multi-regions, we use a single region in the configuration. To better evaluate the performance, we use Manual allocation instead of Autoscaling mode.

The followings are the detail configuration of the 3-node of Spanner cluster.

Name	Value
Spanner region	us-east4 (Northern Virginia)
Regional configuration	Single region
Compute capacity	3 Spanner Nodes
Storage used	1 TB
Scaling mode	Manual allocation

Table 1. Configuration Information of GCP Spanner Cluster

Client VM

We use a VM size that is typical in the real-world production environment. The Table 2 shows the detail information about the GCE VM that is used as YCSB client.

Name	Value
VM instance type	e2-custom-16-32768
vCPU	16
Memory in GB	32
Total storage volume in GB	30
OS	Red Hat Enterprise Linux 9.3
OS image name	5.14.0-362.8.1.el9_3.x86_64
Platform	64-bit

Table 2. Configuration Information of YCSB Client VM on GCP

Environment on AWS

DynamoDB Table

With AWS DynamoDB, there are two capacity pricing modes for processing data reads and writes: On-demand or Provisioned Capacity mode. Provisioned capacity mode allows for a more controlled testing environment. By predetermining the capacity units, we eliminate variables that could arise from the on-demand mode, where DynamoDB automatically adjusts capacity in response to changes in the benchmark workload. This control is crucial for benchmark testing, as it ensures that the results are directly attributable to the workload applied, without external adjustments. As a result, we choose the provisioned mode to capture the cost information to compare the similar cost occurred on GCP Spanner. To isolate the potential impact from replication to multi-regions as a global table, we did not use replica in the DynamoDB table configuration.

The detail information of DynamoDB environment is provided below in Table 3.

Name	Value
DynamoDB region	us-east-1 (North Virginia)
Capacity mode	Provisioned
Storage used	1 TB
Table class	DynamoDB Standard
Auto scaling	Off for both RCU and WCU

Table 3. Configuration Information of AWS DynamoDB Table

Client VM

Similar to Spanner test, we also created an EC2 VM to be used as YCSB client and detail VM information is listed in Table 4.

Name	Value
VM instance size	c5.4xlarge
vCPU	16
Memory in GB	32
Total storage volume in GB	30
OS	Red Hat Enterprise Linux 9.3
OS image name	Linux 5.14.0-362.8.1.el9_3.x86_64
Platform	64-bit

Table 4. Configuration Information of YCSB Client EC2 VM on AWS

YCSB

The YCSB benchmark tool is an open-source framework for evaluating and comparing the performance of multiple types of database systems. It was developed by Yahoo! researchers with the goal of providing a benchmarking tool that can help in understanding the performance of new and existing cloud-based data serving systems, particularly for transaction-processing workloads.

YCSB has several built-in workload distributions and the most popular one including:

- **Uniform**: A database record can be chosen uniformly at random. In other words, all records in the database are equally likely to be chosen.
- **Zipfian**: A database record is chosen based on the Zipfian distribution. For example, some records can be very popular and much more likely to be accessed while most of other records are unpopular.
- **Latest:** It is similar to Zipfian with the only exception that most recently inserted records are in the head of distribution.

Name	Value
YCSB version	YCSB 0.17.0
YCSB connector for Spanner	1,000,000,000 rows or 1 TB of data (the actual storage size is slightly more due to the overhead to row management in the database)
Key distribution	Zipfian
Read/write distribution	80% read and 20% write
Payload size	Use 1000 Byte payload size in YCSB

Table 5. YCSB Workload Information

Other key parameters used in the command-line execution of the benchmark testing.

- `-recoundcount` : the total number of records in the table
- `-operationcount` : the total number of database operations
- `-threads` : the total number of client threads.
- `-target` : the target number of operations per second. It is used in different throughput.
- `-s` : the flag to provide status report every ten seconds.

BENCHMARK APPROACH

In a typical comparison between two relational databases, we can load the same amount of data into two different types of databases with the same kind underlined hardware and OS and execute the workload to compare the performance. As both GCP Spanner and AWS DynamoDB are fully managed databases with dramatical different pricing approaches, we need to identify a common way that can be used in both databases to compare the difference. The idea is to use the same workload throughput to measure the cost between the two databases.

Key Considerations

There are several key considerations in our benchmark approach:

Choosing workload distribution

Among these three workload distributions in YCSB, Zipfian is often the representative of real-world scenarios in which a small percentage of records make up a large portion of access requests. This is especially useful when evaluating performance impact from hot key or rows, which can usually become the bottleneck in database performance. Therefore, we chose Zipfian distribution in our benchmark to represent databases under conditions that are highly representative of actual operational demands.

Setting read and write percentage in the workload

In YCSB configuration, we can specify the percentage for read and write operation. We used a mix of 80% read and 20% write operation mix to closely simulate real-world application scenarios. This ratio is based on typical usage patterns observed in production environments, where read operations often significantly outnumber write operations. By adopting this 80/20 read/write ratio, our benchmarking test targets to mirror the common use cases, especially for those serving as back-ends for web applications, e-commerce platforms, and content management systems. These systems usually experience heavy read traffic as users access data more frequently than they modify it. Therefore, the 80/20 read/write split ratio provides a good way to assess the performance of both databases.

Determine row size of 1,000 bytes

In DynamoDB, for rows (or items) up to 1KB in size, one WCU is used to perform one standard write per second[1]. With the consideration of potential row overhead involved, we set row size at 1,000, slightly less than 1KB, to allow rooms for accommodate row overhead and target 1 WCU per row in the write operation.

[1] "For items up to 1 KB in size, one WCU can perform one standard write request per second" https://aws.amazon.com/dynamodb/pricing/provisioned/

Eliminating auto scaling variables

Disabling auto scaling is another important aspect of this benchmarking approach. While auto scaling is a valuable feature for adapting to workload changes in production environments, it can introduce unpredictable variability in a benchmark test. By turning off auto scaling, we ensure that the capacity remains constant throughout the test, providing a stable basis for evaluating performance and latency.

Pushing around 65% CPU utilization on Spanner

Maintaining CPU utilization at a certain level strikes a balance between achieving high performance and avoiding overutilization, which may lead to increased latency and reduced throughput. In our benchmark testing for Spanner, we chose to target a CPU utilization rate of around 65%. This decision aligns with Google Cloud's best practice recommendations for optimal usage of Spanner. Since each Spanner node is replicated to three different Google Cloud zones in a region, Spanner can remain failover-safe at around 65% utilization. Spanner can operate efficiently, effectively managing its workload while retaining sufficient headroom to handle sudden spikes in demand or temporary increases in workload without degradation in performance.

Considering strongly consistent read

In the OLTP world, one important requirement is that all users should view the same data at the same time, in other words, a consistent view of the data. This is even crucial for applications where up-to-the-moment data accuracy is essential, such as financial transactions, inventory management, and real-time analytics. Spanner by default is using strong consistency. For AWS DynamoDB, there is an option between eventually consistent read and strongly consistent read. Eventually consistent is the default read consistent model for all read operations in DynamoDB.

In our benchmark testing, we chose strongly consistent reads for transactions to simulate real-world database usage scenarios where data accuracy and integrity are critical. By enforcing strong consistency, we aim to evaluate both databases' ability to maintain data integrity and consistency under high workload.

Choosing key cost factors

The pricing structure for both Spanner and DynamoDB is primarily influenced by three cost elements:

- Computing cost
- Storage cost
- The cost of network bandwidth used

Given that our YCSB client resides within the same region as the database, eliminating any network egress charges, our cost analysis can disregard network cost. Instead, our evaluation focuses on the combined costs of computing and storage resources, which are the key factors in our total cost evaluation.

Determining Cost Comparison Approach

The pricing models of Spanner and DynamoDB differ significantly, prompting us to use the throughput metric from YCSB, specifically Queries Per Second (QPS), as a basis for evaluating cost efficiency between the two databases. Our approach begins by establishing the QPS achieved in our Spanner benchmark tests. We then replicate this workload on DynamoDB, maintaining the same QPS level. This method allows us to calculate and compare the hourly costs incurred by both Spanner and DynamoDB under identical workload conditions, providing a clear and direct comparison of their cost-effectiveness.

Our Approach

Here is the overview of our benchmark approach:

Database Table Configuration

For the database table, we use the YCSB table with a row size of 1,000 bytes. To simulate a realistic production environment, our dataset comprises 1 billion rows. With considering typical database overhead, we expect total data storage slightly exceeding 1TB for both databases.

Data Loading Process

The YCSB LOAD function is used to insert these 1 billion rows into the Spanner table, preparing the database for the upcoming tests.

Workload Execution

With the dataset in place on Spanner, we use YCSB's RUN function to execute a workload with an 80% read and 20% write distribution from the YCSB client running from the virtual machine. We execute this workload through multiple parallel sessions, each using multiple threads for concurrent database operations. This setup is designed to generate a workload that maintains approximately 65% CPU utilization on the 3-node Spanner cluster. By running this workload continuously over several hours, we can determine the database performance throughput in terms of Queries Per Second (QPS) and the associated hourly cost for Spanner.

DynamoDB Comparison

Leveraging the QPS benchmark established from Spanner, we then replicate this load on DynamoDB, maintaining the same 80%/20% read/write ratio. It's important to note that achieving similar throughput on DynamoDB might require a higher allocation of Capacity Units to prevent errors during loading. By measuring the actual RCUs and WCUs consumed, we were able to calculate DynamoDB's hourly costs for sustaining a workload identical to that of Spanner in terms of QPS.

The following is the summary of key information used in the benchmark.

Name	Value
YCSB version	YCSB 0.17.0
YCSB connector for Spanner	1,000,000,000 rows or 1 TB of data (the actual storage size is slightly more due to the overhead to row management in the database)
Key distribution	Zipfian
Read/write distribution	80% read and 20% write
Payload size	Use 1000 Byte payload size in YCSB

Table 6. YCSB Workload Information

This approach ensures that our benchmark results can provide relevant and valuable information for organizations looking to understand the cost effectiveness between the two databases under the same workload.

Pricing Information

We use the following information to calculate the hourly cost on both Spanner and DynamoDB.

GCP Spanner

Based on the information from https://cloud.google.com/spanner/pricing.

Compute Cost

The hourly rate is $0.99 per node.

Storage Cost

The hourly rate is $0.00045 per hour.

AWS DynamoDB

Based on the information from https://aws.amazon.com/dynamodb/pricing/provisioned/.

Compute Cost

The hourly rate for DynamoDB Standard table class:
1 WCU : $0.00065
1 RCU : $0.00013

Storage Cost

For DynamoDB Standard table storage, the cost is $0.25 per GB.
Hourly storage rate per GB = $0.25/(744 hours in a month) = $0.00033602

BENCHMARK RESULT

After loading the 1, 000, 000, 000 rows to both Spanner and DynamoDB using YCSB tools, we began our workload benchmark testing.

The QPS to achieve optimal 65% CPU Utilization on 3-node Spanner cluster

Our methodology involved conducting multiple rounds of workload with 80% read and 20% write split ratio, through the YCSB client VM to fine-tune CPU usage in a 3-node Spanner cluster. Aiming for our target of 65% CPU utilization, we initiated several parallel sessions, each tasked with executing 50 million database operations. This setup allowed each session to complete its workload in approximately two hours. Further adjustments were made to the total number of concurrent sessions and the threads per session to achieve the desired CPU usage. To achieve approximately target 65% CPU utilization, we identified the configuration with four sessions running 40 threads each. The results, as illustrated in Figure 1 and Figure 2, are captured from Spanner's System Insight screen.

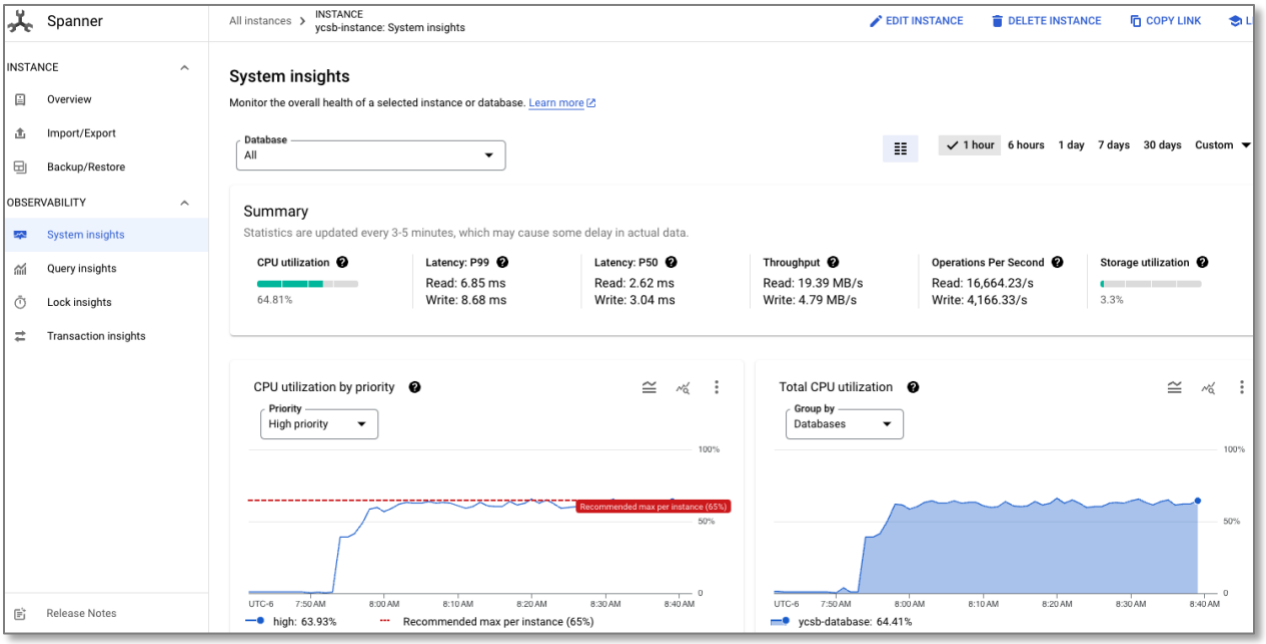

Figure 1. CPU Utilization for the 3-Node Spanner Cluster

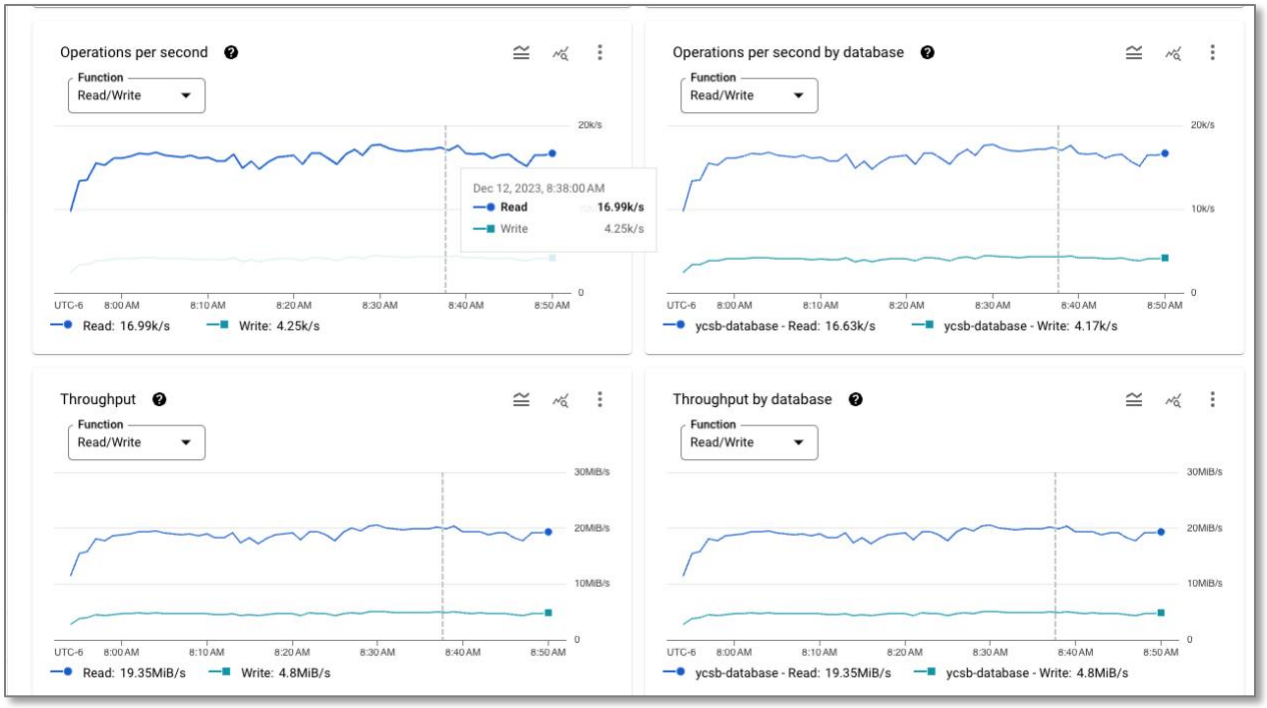

Figure 2. Operation Throughput on Spanner Cluster

At this 65% utilization mark, the cluster achieved a throughput of approximately 21,067 Queries Per Second (QPS) on average, breaking down to about 7,022 QPS per node. To maintain this level of performance in a 3-node Spanner cluster with 1 TB of storage, the hourly cost amounted to $3.43 based on the formula below:

Total Hourly Cost on a 3 node GCP Spanner Cluster
= Hourly Compute Cost + Hourly Storage Cost
= Hourly Compute Rate Per Node x Total Number of Nodes + Hourly Storage Cost
= $0.99 Per Node x 3 Nodes + $0.00045 Per GB x 1024 GB
= $2.97 + $0.46
= $3.43

Performance Comparison for Spanner at Different CPU Utilizations

While the optimal CPU utilization for Spanner is generally recommended to be around 65%, we are interested in understanding the workload throughput as CPU utilization increases in Spanner cluster. Figure 2 illustrates the correlation between varying levels of CPU utilization and the corresponding impact on workload throughput.

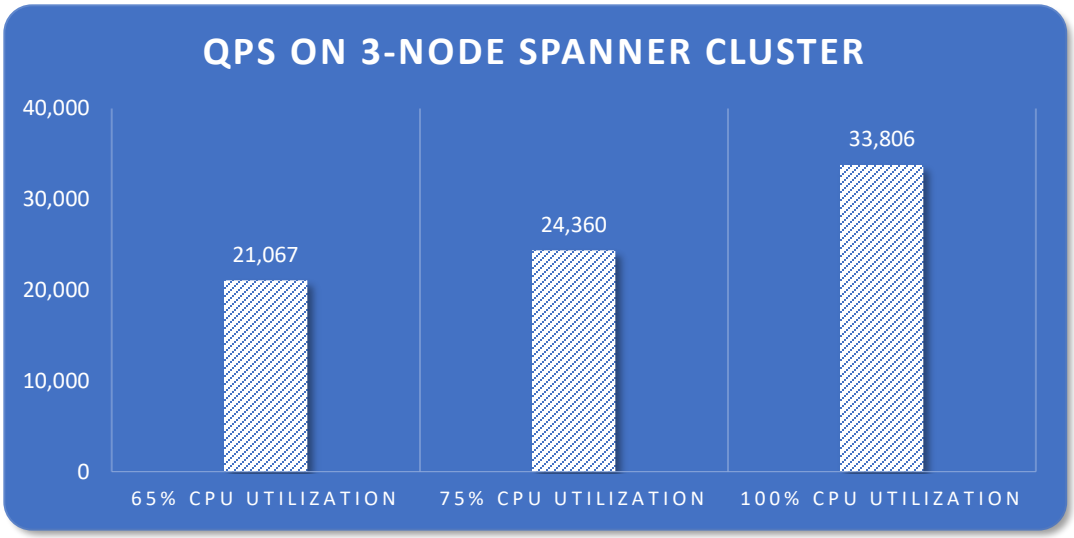

Figure 3. QPS at Different CPU Utilization Level

Figure 3 illustrates that Spanner achieved an overall throughput of 21,067 QPS across a 3-node cluster with approximately 65% CPU utilization. This equates to around 7,000 QPS per node, illustrating how the system effectively manages its resources to deliver optimal performance without overburdening the CPU.

As we increased the, pushing the cluster CPU utilization to 75%, there was a noticeable boost in throughput. The system managed to handle a workload of 24,360 QPS, which breaks down to about 8,120 QPS per node. This higher throughput at elevated CPU utilization demonstrates Spanner's capability to handle increased loads although at a higher resource utilization.

When subjected to a heavy workload that fully utilized the cluster, the 3-node Spanner reached its maximum capacity, demonstrating a throughput of 33,806 QPS, or 11,269 QPS per node. This level of utilization indicates the peak capacity of 11,269 QPS per node when fully maximizing the system where there is little room for additional load without potential performance degradation.

These findings illustrate the direct relationship between CPU utilization and throughput in Spanner, highlighting the importance of balancing load to maintain system efficiency. While higher CPU utilization can lead to increased throughput, it is essential to consider the diminishing returns and potential risks of operating at full capacity, which can impact the system's ability to handle sudden spikes in demand or additional workload. Based on Spanner's documentation, running Spanner near or at 100% CPU utilization for an extended period of time has a risk of degrading normal operation performance and is NOT a safe and recommended approach to achieve consistent performance. Therefore, the optimal CPU utilization level is recommended at 65% CPU utilization, which can achieve the objective of high performance, high availability, low latency, and consistent workload throughput.

The detail configuration for YCSB client to reach these three levels of CPU utilization on Spanner is shown below in Table 7:

CPU Utilization	YCSB Client Session	Threads per Session
65%	4 parallel sessions	32
75%	4 parallel sessions	40
100%	5 parallel sessions	50

Table 7. Configuration Information of GCP Spanner Cluster

Cost Comparison with DynamoDB

To establish a comparable workload throughput in DynamoDB to that of Spanner, we aimed to replicate the same average QPS of 21,067, which was observed over a two-hour execution period on a 3-node Spanner cluster. In DynamoDB's provisioned mode, where only RCUs and WCUs in provisioned mode are applicable, we configured the setup with 16,854 RCUs and 4,213 WCUs based on 80%/20% read/write split ratio to target a similar throughput of 21,067 QPS. This setup assumes that each read operation consumes 1 RCU and each write operation uses 1 WCU, making this a theoretical figure derived from our calculations.

To maintain consistency in our testing approach, we used the same configuration on the YCSB client, running four parallel sessions with 32 threads in each session. A notable feature of YCSB is its `target` parameter, which allows us to set a precise QPS to be directed at DynamoDB. If executed as per our plan, this configuration is expected to sustain the load for approximately two hours. Figure 4 and Figure 5 illustrates the read and write usage, captured from the DynamoDB console, are provided below for a visual representation of the performance under these conditions.

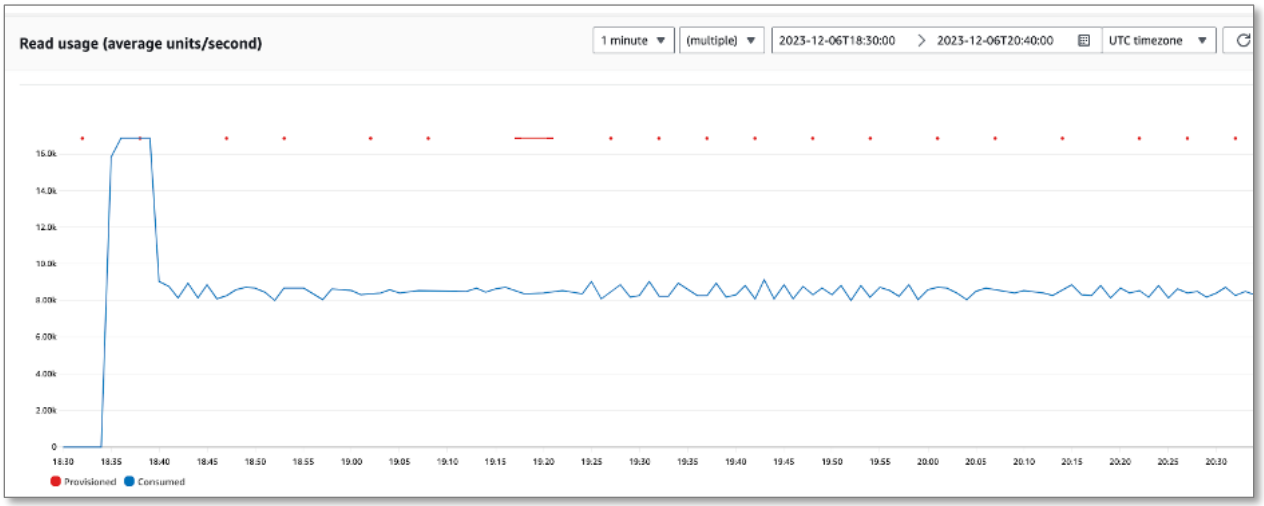

Figure 4. RCU Usage with Write Throttling Enforced

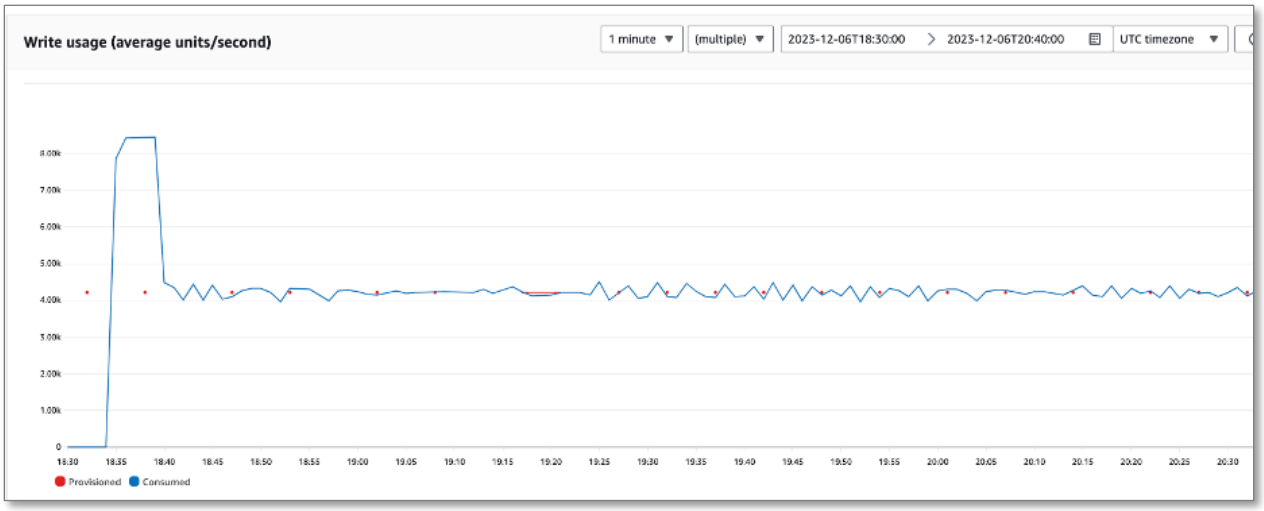

Figure 5. WCU Usage with Write Throttling Enforced

The results of our DynamoDB testing presented some interesting observations. Marked by red dots on the charts, the set limits for RCUs and WCUs were a focal point. Initially, the read operations adhered strictly to the RCU limit we established. However, the WCU usage quickly escalated, nearly doubling the set limit. This situation persisted for a brief period before we encountered update failure errors, as reported by the YCSB client in the logfile, such as "*ProvisionedThroughputExceededException: The level of configured provisioned throughput for the table was exceeded. Consider increasing your provisioning level with the UpdateTable API.*" Following these errors, DynamoDB's throttling mechanism kicked in. This not only pushed the WCU usage back down to our predefined limit but also led to a significant reduction in RCU usage, dropping to around 8,400 QPS, which is about 50% of our target. Consequently, the overall workload throughput was lower than what we anticipated. This throttling issue mirrored our experience during the initial phase of loading 1 billion rows into DynamoDB, where several YCSB client sessions reported data insertion failures. AWS general recommendation is to implement a retry logic within the application to manage such failures during write operations or increase WCU limit.

Based on the actual WCU required shown in Figure 5, we doubled the WCU capacity in the next run. The result was what we expected. There is no throttling in both RCU and WCU observed as shown in Figure 7 and Figure 8.

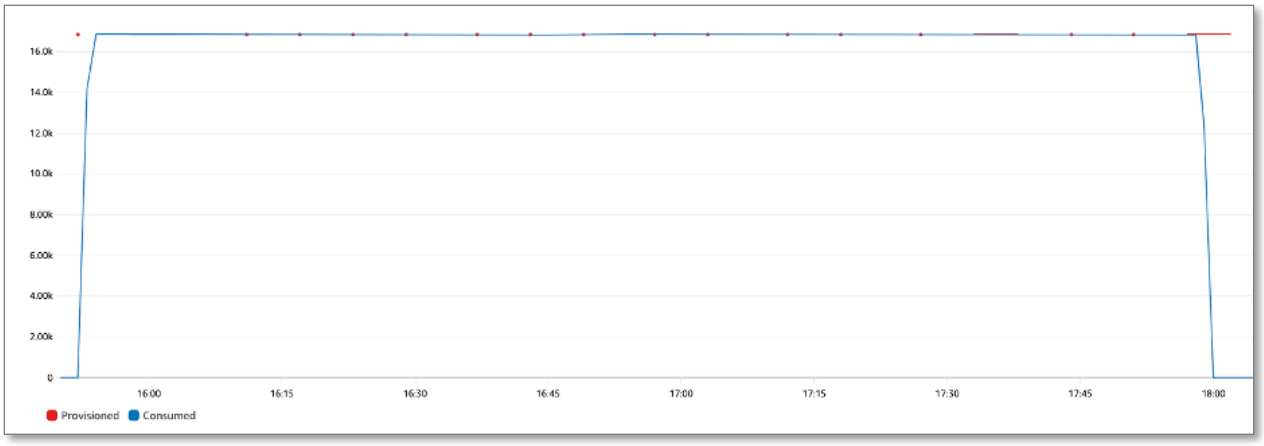

Figure 7. RCU Usage with Double WCU Capacity

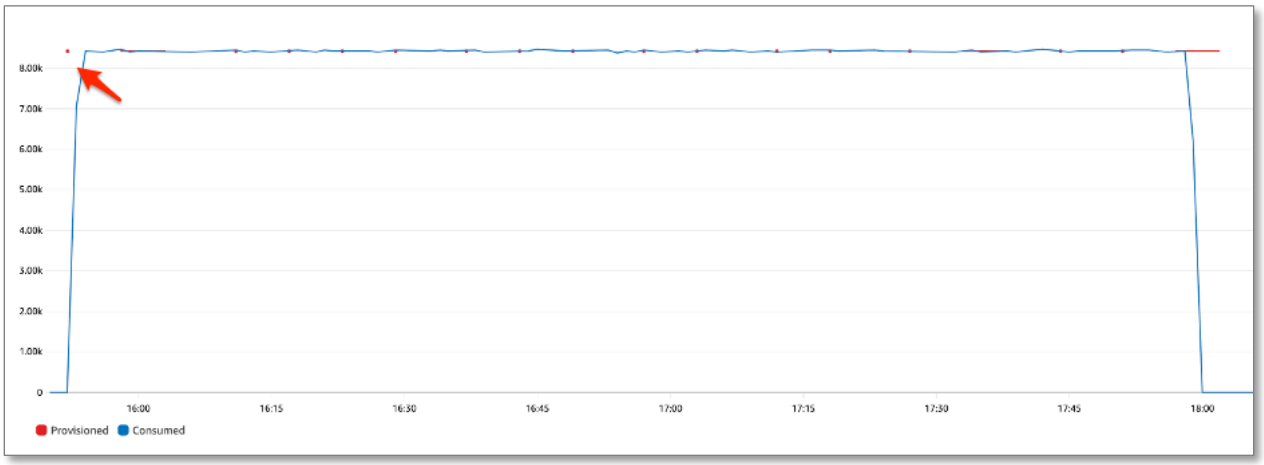

Figure 8. WCU Usage with Double WCU Capacity

Based on AWS documentation, for rows (or items) up to 1KB in size, it requires one WCU. Items larger than 1KB requires additional WCUs. In our tests, with row size defined as 1,000 bytes, technically it is below 1KB in size. But with unknown reasons not in AWS documentation, our 1KB writes required doubling the WCU usage. One potential reason is that the row size may internally exceed 1KB if DynamoDB is adding a small overhead to each row[2].

Given these findings, our cost comparison for DynamoDB will consider two scenarios:

1. **Cost Estimation without Doubling the WCU usage:** This scenario considers the cost without factoring in the impact of requiring doubling the WCU usage for each write.

[2] To verify whether the row size has an impact in the double WCU we observed, we did another test with row size of 900 bytes. The result indicated one WCU is used during one row's write operation. It works as expected and show no throttling in both RCU and WCU.

2. **Cost Estimation with Doubling the WCU usage:** This scenario accounts for the additional cost implications due to the need to double the WCU usage for each write.

For an AWS DynamoDB table without doubling the WCU usage, the hourly cost amounts to $5.27 based on the formula below:

Total Hourly Cost on AWS DynamoDB without Doubling WCU Capacity
= Hourly Compute Cost + Hourly Storage Cost
= (Hourly Provisioned Read Capacity Cost + Hourly Provisioned Write Capacity Cost)
+ Hourly Storage Cost
= ($0.00013 Per RCU x 16854 + $0.00065 Per WCU x 4213)
+ $0.00033602 per GB x 1024 GB
= $2.19 + $2.74 + $0.34
= $5.27

To maintain the real performance with Spanner, we need to double the WCU usage. For an AWS DynamoDB table with doubling the WCU usage, the hourly cost jumps to $8.01 based on the formula below:

Total Hourly Cost on AWS DynamoDB with Doubling WCU Capacity
= Hourly Compute Cost + Hourly Storage Cost
= (Hourly Provisioned Read Capacity Cost + Hourly Provisioned Write Capacity Cost)
+ Hourly Storage Cost
= ($0.00013 Per RCU x 16854 + $0.00065 Per WCU x 4213 x 2)
+ $0.00033602 per GB x 1024 GB
= $2.19 + $5.48 + $0.34
= $8.01

Figure 9 below illustrates a comparative analysis of the hourly costs across three different scenarios:

- The cost of running a 3-node cluster on GCP Spanner.
- The cost of targeting to achieve equivalent QPS throughput on AWS DynamoDB without double WCU.
- The cost of achieving equivalent QPS throughput on AWS DynamoDB with double WCU.

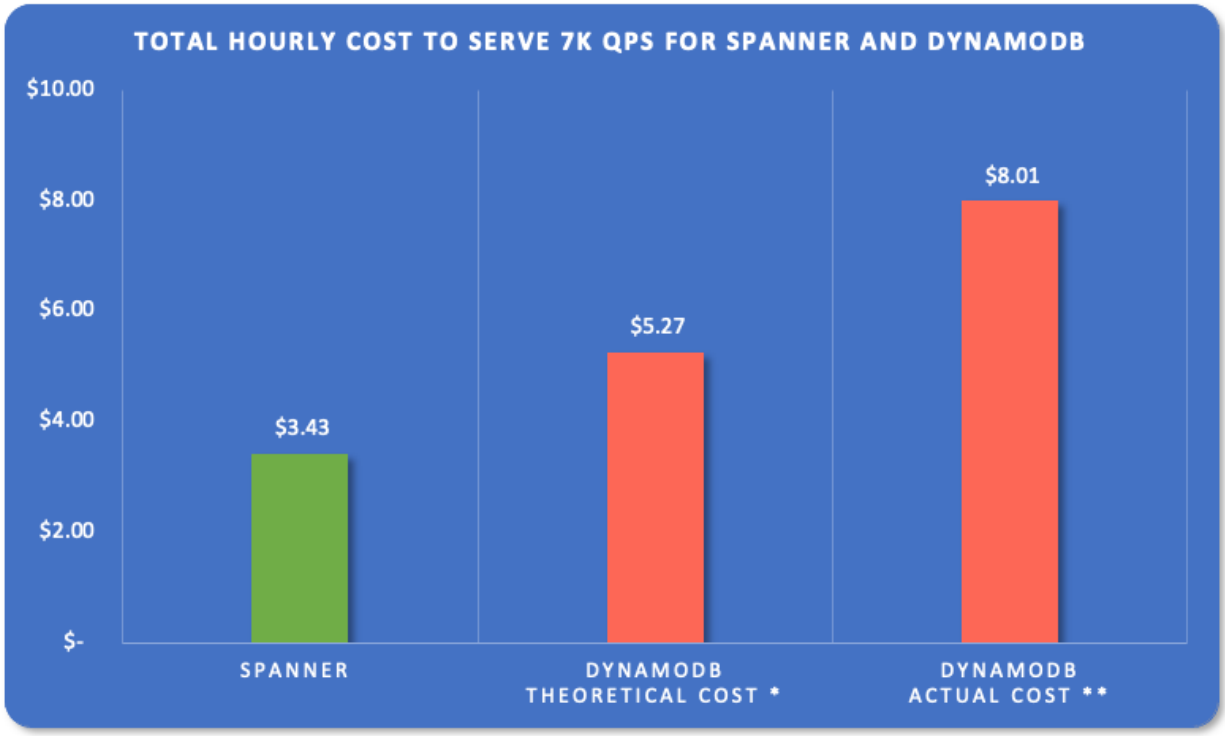

Figure 9. Total Cost to Serve 7K QPS for Spanner and DynamoDB (lower is better)

* Without doubling the WCU usage in DynamoDB, the theoretical cost of RCU + WCU is equal to the QPS achieved from Spanner when row size is 1,000 bytes. This is theoretical cost.

** Doubling the WCU usage in DynamoDB is required to achieve the same 7K QPS from Spanner when row size is 1,000 bytes. This is actual cost.

In the referenced figure above, we observe a distinct cost disparity between Spanner and DynamoDB for an equivalent workload. Operating a 3-node Spanner cluster with the consideration of compute and 1 TB storage, an hourly cost for Spanner amounts to $3.43. Conversely, under the same workload (mirroring the QPS generated by Spanner) and storage capacity, DynamoDB's cost fluctuates between $5.27 and $8.01. In scenarios where DynamoDB faces a necessity for doubled WCU, the cost escalates significantly. This results in DynamoDB being about 54% costlier than Spanner in scenarios without the need for additional WCU, and 134% more expensive when additional WCU is required, demonstrating a marked increase in expenditure under certain operational conditions.

Both platforms offer scalability, but it's crucial to plan how scaling will impact your costs. If your primary concern is cost efficiency, especially under heavy workloads, Spanner appears to be the more economical choice. It provides a lower cost for overall cost at a large scale, making it suitable for businesses looking to optimize their budget while still requiring a robust, scalable database solution.

CONCLUSION

The choice to select a target large global cloud-based database depends on several factors, including specific use cases, scalability requirements, consistency needs, and budget considerations. The analysis reveals that while DynamoDB and Spanner can both be configured to handle high workloads, their performance characteristics and cost implications under load are markedly different. Understanding these nuances is essential for making informed decisions when choosing a database service for high-demand applications.

Based on the benchmark result above, Spanner demonstrated more predictable performance while maintaining a consistent throughput at a lower cost. The comparison of hourly costs across Spanner and DynamoDB, under similar QPS conditions, suggests that Spanner offers stable performance at a lower cost, and the cost dynamics in DynamoDB can vary significantly based on its handling of high load and WCU requirements.

For applications that need to maintain uniform performance even as they scale, Spanner's architecture ensures consistent latency and throughput. Spanner may be more cost-effective for applications with heavy, consistent workloads due to its predictable pricing structure.

In addition, Spanner’s default strong consistency behavior will benefit applications that require strong consistency across globally distributed data and as well as systems that need a horizontally scalable database with global transaction support and high availability.

In summary, database selection should be based on the specific requirements of your application, including the type of data you’re storing, your scalability needs, your consistency requirements, your budget, and how these factors align with the features and pricing models of each database service. For applications that need to maintain uniform performance and cost efficiency even as they scale, Spanner's architecture ensures consistent latency and throughput.

ABOUT

Author

Weidong Zhou, Principal Director, Accenture
Weidong Zhou has more than 20+ years of experience in various fields of IT consulting, mainly focused on financial services, insurance, technology, telecom and retailing customers. His keen understanding of the business problems, with strong technology understanding and technical leadership skills helps him provide simple solutions to complex problems.

Muhammad K. Aslam, Senior Manager, Accenture
With a wealth of expertise, Kashif has cultivated a profound understanding of high-volume transactional databases and data warehousing systems across diverse industries such as Telecom, Banking, Retail, and Healthcare. His professional journey has been marked by invaluable contributions to numerous clients, guiding them through seamless migrations to hyper-scaler cloud environments. Notably, he is also the author of a book on data replication.

Mukesh Sharma, Senior Manager, Accenture
Experienced Database/Big Data Architect with over 21 years of expertise in managing, configuring, and optimizing databases and ERP systems. Skilled in designing and implementing high-performance database environments for business-critical applications. Proficient in various databases technologies. Extensive experience in architecting large-scale databases, cloud migration databases and mission-critical systems.

Accenture

Accenture is a global professional services company with leading capabilities in digital, cloud and security. Combining unmatched experience and specialized skills across more than 40 industries, we offer Strategy and Consulting, Interactive, Technology and Operations services—all powered by the world's largest network of Advanced Technology and Intelligent Operations centers. Our 710,000 people deliver on the promise of technology and human ingenuity every day, serving clients in more than 120 countries. We embrace the power of change to create value and shared success for our clients, people, shareholders, partners, and communities.

Visit us at www.accenture.com