Get lower latency for NoSQL workloads in the cloud with Azure Cosmos DB for NoSQL

Azure Cosmos DB delivered lower latency at a lower solution cost in most cases than Amazon DynamoDB

Organizations that rely on NoSQL databases can offer users a better experience by choosing a cloud solution that responds more quickly to requests—or, in other words, has lower latency. In addition to providing faster response times to users, cloud solutions with lower latency can help to reduce costs by consuming fewer resources to process the same workload. We used the Yahoo! Cloud Serving Benchmark (YCSB) to measure the 95th and 99th percentile latency of two fully managed, NoSQL database service cloud solutions: Azure Cosmos DB and Amazon DynamoDB.

We tested both solutions using workload profiles targeting 10,000, 30,000, and 50,000 operations per second (OPS). We assessed the latency of both solutions with a read-only profile, a write-only profile, an updates-only profile, and a mixed profile of 90 percent reads and 10 percent writes. The Azure Cosmos DB solution outperformed the DynamoDB solution in every test at the 99th percentile and in all but one test at the 95th percentile, where the difference was statistically insignificant. To demonstrate the large-scale capabilities of the Azure Cosmos DB solution, we measured its latency at 1 million OPS and found it offered 3.15 ms latencies (100 percent read) and 12.8 ms latencies (100 percent write) at the 99th percentile. Lastly, using publicly available cost information from each service's deployment wizards, we calculated the hourly rates of both solutions for each test and found that the Azure Cosmos DB solution was more affordable than the Amazon DynamoDB solution for all but two workloads. In the two instances where Azure Cosmos DB was less affordable, it offered an average of 74.5 percent better 99th percentile latency at an average cost that was only 24.5 percent higher.

How we tested

We created each database and provisioned them with a limit on the resources they could consume. We used the YCSB workload and measured the full transaction latencies for one hour. For the cost calculations, the database charges depend on the resource limit we provisioned, so to estimate the cost of performing each workload at the target, we chose resource limits large enough to perform the workload with a small amount of additional resources as headroom. We recorded the database charges that each cloud's provisioning tool provided.

To measure the latency of Azure Cosmos DB for NoSQL, we established a resource group, an Azure Cosmos DB account, and a database using the Azure portal. For each workload, we created a container (akin to a SQL table) in the database with a resource setting large enough to handle the workload's I/O. We created the database in provisioned mode, which fixes the maximum rate at which your application can consume request units (RUs). RUs are a measure of the computer resources/costs an application uses to perform one database operation on one KB of data. We set the RU rate to the appropriate value we found in the Azure DB benchmarking repository; it allowed enough headroom to sustain the target OPS for each workload.[1] For the specific configurations we used for each workload, see the science behind the report. To run the YCSB workload, we used the testing framework from a forked copy (an independent copy) of the GitHub repository.[2] Each workload used an Azure template to create one client VM, install YCSB, compile the Azure Cosmos DB for NoSQL driver, execute the load and run phases of the workload, and copy the results to an Azure Blob storage. All database and client resources for this testing were in the East US Azure region.

To measure the latency of DynamoDB on AWS, we created a DynamoDB table using the AWS portal. For each workload, we set the maximum read capacity units (RCUs) and maximum write capacity units (WCUs) to match the workload's target rate. We used the DynamoDB capacity calculator to estimate the RCU and WCU needed for the workload, and added an additional 2,000 capacity units/s for headroom. We found that this headroom sufficed to eliminate failed updates and insertions. You can find the specific RCUs and WCUs we used for each workload in the science behind the report. We created one client VM, running Ubuntu 22.04 on x86-64, with sufficient CPU resources (threads) to drive the database at the target rates. We installed YCSB 0.17.0 on the VM and we compiled the DynamoDB driver with Java 8 (build 351). We then executed the load and run phases of the workload from the Linux command line. All database and client resources for this testing were in the us-east-1 AWS region.

Lower latency at 10K OPS

Using YCSB, we measured the latency of both solutions for 100 percent read, write, and update workloads at a target rate of 10,000 OPS. The YCSB client measures the latency, which is the time between the start of the client's request and the time the client receives the last response from the database server. The Azure Cosmos DB solution offered lower latencies in all the workloads we tested at a target rate of 10,000 OPS except in one instance, where the Azure solution was a statistically insignificant 0.1 percent higher than the Amazon DynamoDB solution. With a 100 percent read workload, the Azure Cosmos DB solution provided 83.9 percent lower latency at the 99th percentile than the Amazon DynamoDB solution.

About Microsoft Azure Cosmos DB

According to Microsoft, Azure Cosmos DB is a "fully managed and serverless distributed database."[3] A serverless solution with no minimum charges, Azure Cosmos DB allows organizations to run NoSQL workloads with unpredictable traffic and pay for only the resources they use.

To learn more about Microsoft Azure Cosmos DB, visit https://azure.microsoft.com/en-us/products/cosmos-db.

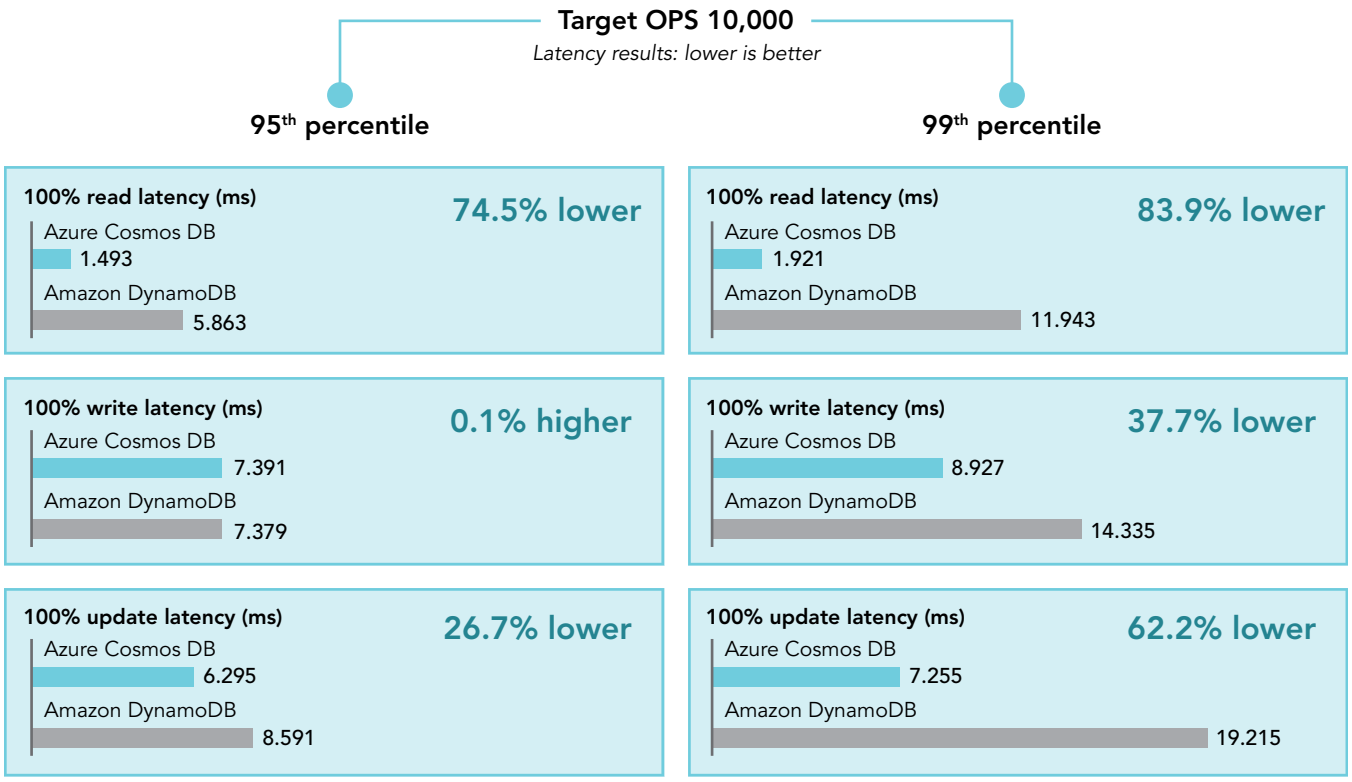

Figure 1: The latencies, in milliseconds, of the solutions at a target rate of 10,000 OPS. Lower is better. Source: Principled Technologies.

Lower latency at 30K OPS

The next series of tests measured the latency of the two solutions for 100 percent read, write, and update workloads at a target rate of 30,000 OPS. The Azure Cosmos DB solution offered lower latencies at the 95th and 99th percentile for every workload we tested. On a 100 percent update workload, Azure Cosmos DB provided 54.6 percent lower latency than the Amazon DynamoDB solution at the 99th percentile.

About the Yahoo! Cloud Serving Benchmark

Yahoo! developed the Yahoo! Cloud Serving Benchmark to evaluate the performance of cloud solutions using a common set of workloads. According to Yahoo!, "the core workloads provide a well-rounded picture of a system's performance" and the YCSB Client "is extensible so that you can define new and different workloads to examine system aspects, or application scenarios, not adequately covered by the core workload."[4]

YCSB supports five operations in a workload:

- Read: query a complete record
- Write: insert a complete record
- Update: change some of the fields in an existing record
- Scan: query a small range of records
- Read-Modify-Write: query a complete record, change part of it, and write it back

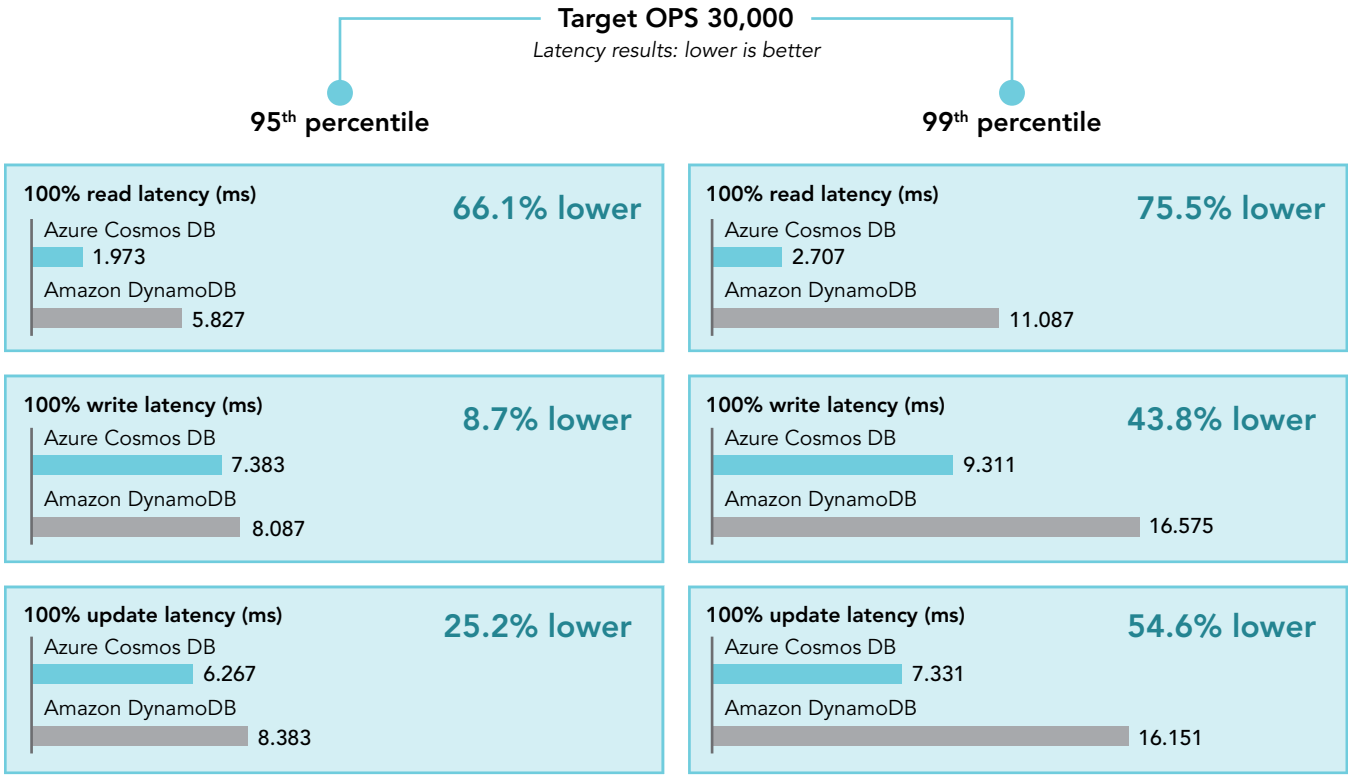

Figure 2: The latencies, in milliseconds, of the solutions at a target rate of 30,000 OPS. Lower is better. Source: Principled Technologies.

Lower latency at 50K OPS

We measured the latency of the two solutions for 100 percent read, write, and update workloads at a target rate of 50,000 OPS and found that the Azure Cosmos DB solution offered lower latencies than the Amazon DynamoDB solution in every instance. Testing the latency of 100 percent read, write, and update workloads provides insight into the performance of the solutions, but to understand how the solutions might perform in a more real-world scenario, we also measured a mixed workload of 90 percent read operations and 10 percent write operations at a target rate of 50,000 OPS. The Azure Cosmos DB solution offered 75.9 less 99th percentile read latency and 54.1 less 99th percentile write latency than the Amazon DynamoDB solution at a target rate of 50,000 OPS.

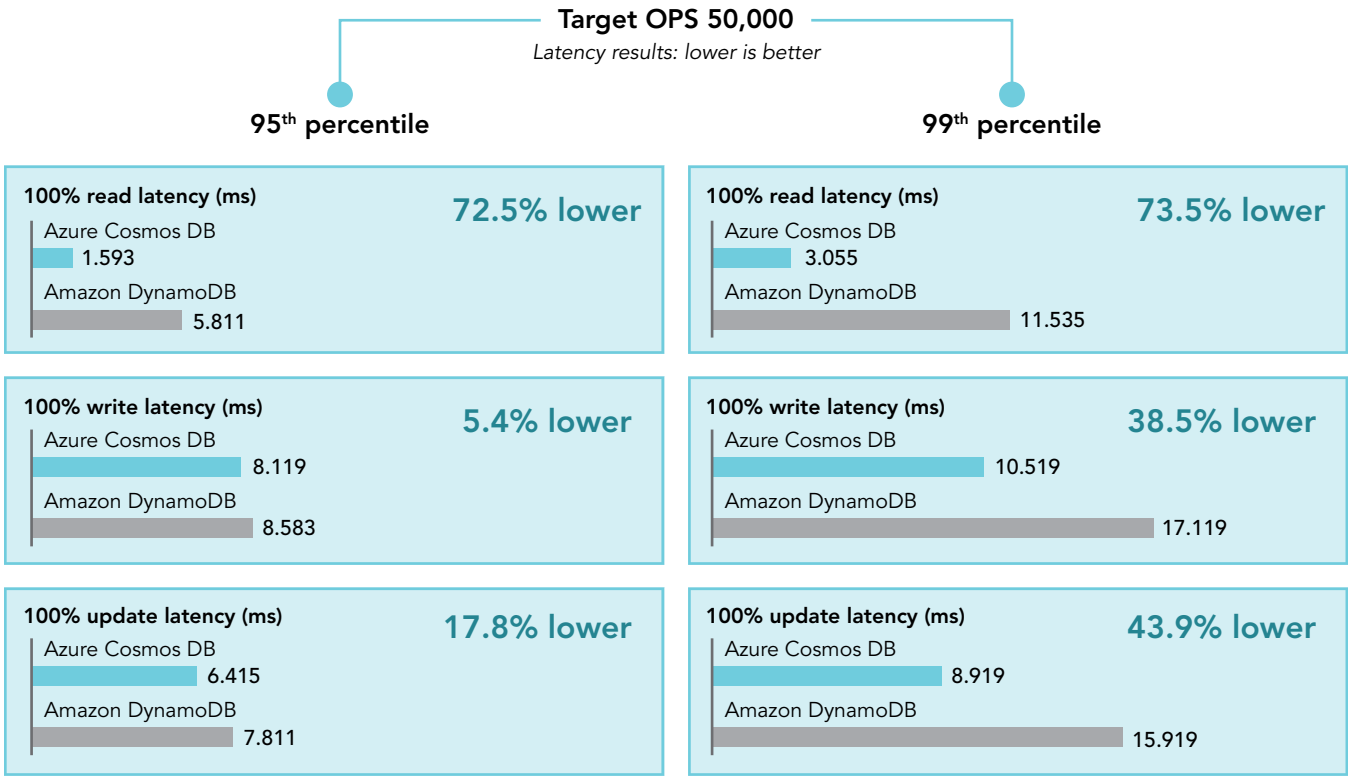

Figure 3: The latencies, in milliseconds, of the solutions at a target rate of 50,000 OPS. Lower is better. Source: Principled Technologies.

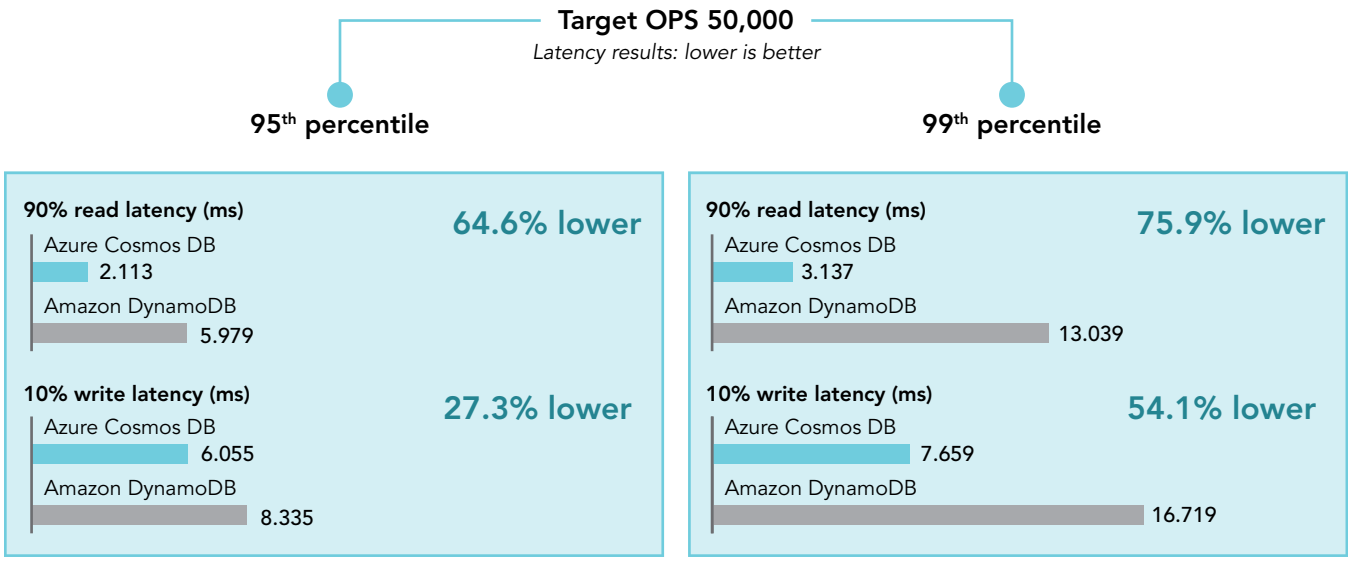

Figure 4: The latencies, in milliseconds, of the solutions at a target rate of 50,000 OPS. Lower is better. Source: Principled Technologies.

Lower costs per hour with Azure Cosmos DB

Higher performance often is available only at a higher price. To evaluate whether this was true in our tests, we calculated the costs per hour of each solution running the workloads we tested for latency. We wanted to focus on the cost of the services themselves, so these estimates are for using the database services we provisioned and do not include the costs for resources the client used running the YCSB benchmark or resources the database drivers used running on the client (e.g., the number of CPU cores). The Azure Cosmos DB container creation tool provided the costs of the Azure Cosmos DB solution. The Amazon DynamoDB deployment tool provided the costs of the Amazon DynamoDB solution. We found that the Azure Cosmos DB solution cost less per hour in all but two instances: the hourly cost of the 100 percent read workloads at target rates of 30,000 OPS and 50,000 OPS was an average of 24.5 percent higher for the Azure Cosmos DB solution, but the average latency was 74.5 percent lower (better) at the 99th percentile.

Table 1: The cost in dollars/hour of each solution at a target rate of 10,000 OPS. Lower is better. Source: Principled Technologies.

Target OPS 10,000			
	Azure Cosmos DB	**Amazon DynamoDB**	**Percentage savings**
100% read	$0.96	$0.99	3.03%
100% write	$10.40	$14.71	29.29%
100% update	$12.95	$15.50	16.45%

Table 2: The cost in dollars/hour of each solution at a target rate of 30,000 OPS. Lower is better. Source: Principled Technologies.

Target OPS 30,000			
	Azure Cosmos DB	**Amazon DynamoDB**	**Percentage savings**
100% read	$2.88	$2.32	-24.13%
100% write	$31.20	$41.21	24.29%
100% update	$38.86	$43.33	10.31%

Table 3: The cost in dollars/hour of each solution at a target rate of 50,000 OPS. Lower is better. Source: Principled Technologies.

Target OPS 50,000			
	Azure Cosmos DB	**Amazon DynamoDB**	**Percentage savings**
100% read	$4.80	$3.84	-25.00%
100% write	$52.00	$67.70	23.19%
100% update	$64.75	$71.15	8.99%
90% read and 10% write	$8.00	$10.33	22.55%

Azure Cosmos DB performance at 1 million OPS

Applications such as financial trading and real-time analytics require high throughput and low response times to provide near-instantaneous processing.

To get a better understanding of the latency of Azure Cosmos DB at a large scale, we measured the latency of the solution for 100 percent read and 100 percent write operations at a target rate of 1,000,000 OPS. This target rate for one hour is a scaling goal other groups have used.[5,6,7]

The Azure Cosmos DB solution achieved a 99th percentile latency of 3.15 ms for the 100 percent read workload and 12.8 ms for the 100 percent write workload. Comparing these response times to the response times in the 50,000 OPS test, we see a similar read latency and only a 2.3 ms increase in the write latency. These results suggest that Azure Cosmos DB can scale to handle even the largest workload needs even at an unusually large scale with 100 percent writes at 1,000,000 OPS.

Table 4: The 95th and 99th percentile latencies in milliseconds for database transactions for each workload. Median of three runs. Lower is better. Source: Principled Technologies.

Azure Cosmos DB 1,000,000 OPS		
Workload	100% reads	100% writes
95th percentile	2.134	9.097
99th percentile	3.152	12.877

Conclusion

When we compared the latency of Azure Cosmos DB to that of Amazon DynamoDB, we found that the Azure Cosmos DB solution outperformed the Amazon DynamoDB solution in all but one instance, where the difference was statistically insignificant. Plus, we found that the Azure Cosmos DB solution was more affordable than the Amazon DynamoDB solution in most instances. In the two instances where the Amazon DynamoDB solution was cheaper, the Azure Cosmos DB solution provided better latency processing those workloads. At a target rate of 1,000,000 OPS the Azure Cosmos DB solution offered 3.15 ms latencies (100 percent read) and 12.8 ms latencies (100 percent write) at the 99th percentile, which suggests that the solution can efficiently scale and handle a high number of queries with minimal delay or interruption.

1. GitHub, "SQL(Core) API," accessed April 6, 2023, https://github.com/Azure/azure-db-benchmarking/blob/main/cosmos/sql/tools/java/ycsb/recipes/.
2. GitHub, "Benchmarking Framework For Azure Databases," accessed April 6, 2023, https://github.com/Azure/azure-db-benchmarking/.
3. Microsoft, "Azure Cosmos DB," accessed March 30, 2023, https://azure.microsoft.com/en-us/products/cosmos-db.
4. Yahoo!, "Yahoo Cloud Serving Benchmark," accessed March 30, 2023, https://research.yahoo.com/news/yahoo-cloud-serving-benchmark/.
5. Jonah Berquist, "One million queries per second with MySQL," accessed April 7, 2023, https://planetscale.com/blog/one-million-queries-per-second-with-mysql.
6. Douglas Hood, "Scaling SQL to millions of transactions per second with a single database," accessed April 7, 2023, https://www.linkedin.com/pulse/scaling-sql-millions-transactions-per-second-single-database-hood.
7. Christos Kalantzis, "Revisiting 1 Million Writes per second," accessed April 7, 2023, https://netflixtechblog.com/revisiting-1-million-writes-per-second-c191a84864cc.

Read the science behind this report at https://facts.pt/YqsF4MH ▶

This project was commissioned by Microsoft.

Principled Technologies®

Facts matter.®