The science behind the report:

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

This document describes what we tested, how we tested, and what we found. To learn how these facts translate into real-world benefits, read the report Get lower latency for NoSQL workloads in the cloud with Azure Cosmos DB for NoSQL.

We concluded our hands-on testing on March 31, 2023. During testing, we determined the appropriate hardware and software configurations and applied updates as they became available. The results in this report reflect configurations that we finalized on March 6, 2023 or earlier. Unavoidably, these configurations may not represent the latest versions available when this report appears.

Our results

To learn more about how we have calculated the wins in this report, go to http://facts.pt/calculating-and-highlighting-wins. Unless we state otherwise, we have followed the rules and principles we outline in that document.

Table 1: The 95th percentile latencies in milliseconds for database transactions for each workload. Median of three runs. Lower is better. Source: Principled Technologies.

Target OPS	10,000			30,000			50,000				
Workload	100% reads	100% writes	100% updates	100% reads	100% writes	100% updates	100% reads	100% writes	100% updates	90% reads	10% writes
Azure Cosmos DB	1.493	7.391	6.295	1.973	7.383	6.267	1.593	8.119	6.415	2.113	6.055
Amazon DynamoDB	5.863	7.379	8.591	5.827	8.087	8.383	5.811	8.583	7.811	5.979	8.335
Relative percentage difference	74.5	-0.1	26.7	66.1	8.7	25.2	72.5	5.4	17.8	64.6	27.3

Table 2: The 99th percentile latencies in milliseconds for database transactions for each workload. Median of three runs. Lower is better. Source: Principled Technologies.

Target OPS	10,000			30,000			50,000				
Workload	100% reads	100% writes	100% updates	100% reads	100% writes	100% updates	100% reads	100% writes	100% updates	90% reads	10% writes
Azure Cosmos DB	1.921	8.927	7.255	2.707	9.311	7.331	3.055	10.519	8.919	3.137	7.659
Amazon DynamoDB	11.943	14.335	19.215	11.087	16.575	16.151	11.535	17.119	15.919	13.039	16.719
Relative percentage difference	83.9	37.7	62.2	75.5	43.8	54.6	73.5	38.5	43.9	75.9	54.1

Table 3: The 95th percentile latencies in milliseconds for database transactions for the workloads. Median of three runs. Lower is better. Source: Principled Technologies.

Target OPS	1,000,000	
Workload	100% reads	100% writes
Azure Cosmos DB	2.134	9.097

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

Target OPS	1,000,000	
Workload	100% reads	100% writes
Azure Cosmos DB	3.152	12.877

System configuration information

Table 5: VM specifications for the 10,000 OPS testing.

Cloud configuration information	Azure Cosmos DB	Amazon Dynamo DB
General information		
Cloud service provider	Microsoft Azure®	AWS®
Date testing ended	March 31, 2023	March 29, 2023
Region	East US	us-east-1
Workload information		
Workload name and version	YCSB 0.17.0 with Azure Cosmos update at 10,000 OPS	YCSB 0.17.0 at 10,000 OPS
Three workloads	100 percent reads, 100 percent writes, and 100 percent update	
Iterations and result choice	Median of three runs	
Cloud VM or instance details		
Number of VMs	1	1
VM or instance size	Standard_D8s_v4	m6i.2xlarge
vCPU	8	8
Number of cores/threads	8	8
Memory (GB)	32	32
Underlying processor	Intel® Xeon® Platinum 8272CL	Intel Xeon Platinum 8375C
Operating system information		
Image or template name and UUID	Image Ubuntu® Server 20.04 LTS – Gen 2	ami-007855ac798b5175e
Operating system name	Ubuntu Server 22.04 LTS	Ubuntu Server 22.04 LTS
Kernel version	5.15.0-1035-azure #42-Ubuntu SMP Tue Feb 28 19:41:23 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux	5.15.0-1031-aws #35-Ubuntu SMP Fri Feb 10 02:07:18 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
Date patches last applied	March 6, 2023	February 11, 2023
Changes made from CSP image	We added several packages. See the How we tested section.	
Instance storage		
Number of volumes	1	1
Volume use in this test	OS	OS
CSP volume type	Standard SSD LRS	gp2
Volume size (GB)	20	20
Encryption type	None	None

Table 6: VM specifications for the 30,000 OPS testing.

Cloud configuration information	Azure Cosmos DB	Amazon Dynamo DB
General information		
Cloud service provider	Microsoft Azure	AWS
Date testing ended	March 31, 2023	March 29, 2023
Region	East US	us-east-1
Workload information		
Workload name and version	YCSB 0.17.0 with Azure Cosmos update at 30,000 OPS	YCSB 0.17.0 at 30,000 OPS
Four workloads	100 percent reads, 100 percent writes, and 100 percent update	
Iterations and result choice	Median of three runs	
Cloud VM or instance details		
Number of VMs	1	1
VM or instance size	Standard_D32s_v4	m6i.8xlarge
vCPU	32	32
Number of cores/threads	32	32
Memory (GB)	128	128
Underlying processor	Intel Xeon Platinum 8272CL	Intel Xeon Platinum 8375C
Operating system information		
Image or template name and UUID	Image Ubuntu Server 20.04 LTS – Gen 2	ami-007855ac798b5175e
Operating system name	Ubuntu Server 22.04 LTS	Ubuntu Server 22.04 LTS
Kernel version	5.15.0-1035-azure #42-Ubuntu SMP Tue Feb 28 19:41:23 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux	5.15.0-1031-aws #35-Ubuntu SMP Fri Feb 10 02:07:18 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
Date patches last applied	March 6, 2023	February 11, 2023
Changes made from CSP image	We added several packages. See the How we tested section.	
Instance storage		
Number of volumes	1	1
Volume use in this test	OS	OS
CSP volume type	Standard SSD LRS	gp2
Volume size (GB)	20	20
Encryption type	None	None

Table 7: VM specifications for the 50,000 OPS testing

Cloud configuration information	Azure Cosmos DB	Amazon Dynamo DB
General information		
Cloud service provider	Microsoft Azure	AWS
Date testing ended	March 31, 2023	March 29, 2023
Region	East US	us-east-1
Workload information		
Workload name and version	YCSB 0.17.0 with Azure Cosmos update at 50,000 OPS	YCSB 0.17.0 at 50,000 OPS
Four workloads	100 percent reads, 100 percent writes, 100 percent update, and 90 percent reads-10 percent writes	
Iterations and result choice	Median of three runs	
Cloud VM or instance details		
Number of VMs	1	1
VM or instance size	Standard_D48s_v4	m6i.12xlarge
vCPU	48	48
Number of cores/threads	48	48
Memory (GB)	192	192
Underlying processor	Intel Xeon Platinum 8272CL	Intel Xeon Platinum 8375C
Operating system information		
Image or template name and UUID	Image Ubuntu Server 20.04 LTS – Gen 2	ami-007855ac798b5175e
Operating system name	Ubuntu Server 22.04 LTS	Ubuntu Server 22.04 LTS
Kernel version	5.15.0-1035-azure #42-Ubuntu SMP Tue Feb 28 19:41:23 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux	5.15.0-1031-aws #35-Ubuntu SMP Fri Feb 10 02:07:18 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
Date patches last applied	March 6, 2023	February 11, 2023
Changes made from CSP image	We added several packages. See the How we tested section.	
Instance storage		
Number of volumes	1	1
Volume use in this test	OS	OS
CSP volume type	Standard SSD LRS	gp2
Volume size (GB)	20	20
Encryption type	None	None

Table 8: VM specifications for the 1,000,000 OPS testing.

Cloud configuration information	Azure Cosmos DB
General information	
Cloud service provider	Microsoft Azure
Date testing ended	March 31, 2023
Region	East US
Workload information	
Workload name and version	YCSB 0.17.0 with Azure Cosmos update at 1,000,000 OPS
Two workloads	100 percent reads and 100 percent writes.
Iterations and result choice	Median of three runs
Cloud VM or instance details	
Number of VMs	15
VM or instance size	Standard_D64ds_v5
vCPU	64
Number of cores/threads	64
Memory (GB)	256
Underlying processor	Intel Xeon Platinum 8370C
Operating system information	
Image or template name and UUID	Image Ubuntu Server 20.04 LTS – Gen 2
Operating system name	Ubuntu Server 22.04 LTS
Kernel version	5.15.0-1035-azure #42-Ubuntu SMP Tue Feb 28 19:41:23 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
Date patches last applied	March 6, 2023
Changes made from CSP image	We added several packages. See the How we tested section.
Instance storage	
Number of volumes	1
Volume use in this test	OS
CSP volume type	Standard SSD LRS
Volume size (GB)	20
Encryption type	None

How we tested

Azure Cosmos DB for NoSQL testing

We created all resources in the same Azure region, East US.

The choice of indexing and the data-consistency model used by each NoSQL database affects performance for many operations. For these two areas, we followed the default YCSB configuration. We did not add extra indexing, and used the default data-consistency mode.

For Azure Cosmos DB benchmarking, the default policy indexes "every property of every item and enforces range indexes for any string or number."[1] This will incur extra cost for writes but "allows you to get good query performance without having to think about indexing and index management upfront."[2] The consistency mode we used was session consistency. For AWS DynamoDB, the consistency mode we used was eventual read consistency.

Azure has submitted several updates to YCSB that improve Azure Cosmos DB performance. The YCSB maintainers have not incorporated them into the YCSB code. Accordingly, we forked Azure's GitHub repository for YCSB to our account in order to use these updated features. Copying the code a second time may seem unnecessary, but it ensures that we work with a fixed version of the code (March 6, 2023) because the Azure templating code we used to perform the Azure Cosmos DB testing does not appear to accept a specific branch or release, and so uses the main branch.

Forking two Azure GitHub repositories

This code will be copied to the VM by the testing framework via an Azure template.

1. Fork the Azure YCSB repository https://github.com/Azure/YCSB to your GitHub account.
2. Fork the Azure testing framework repository https://github.com/Azure/azure-db-benchmarking to your GitHub account.

Generating the Azure template for one workload not in the GitHub repository

The testing framework has Azure template files for all workloads except the mixed one: 90 percent reads and 10 percent writes at a rate of 50,000 OPS. We created one based on the 100 percent read workload at 50,000 OPS.

1. From your GitHub account, go to directory `azure-db-benchmarking/blob/main/cosmos/sql/tools/java/ycsb/recipes/read/50-thousand-rps-read`, and copy the file azuredeploy.json to azuredeploy9010.json.
2. Modify the file azuredeploy9010.json with the following patch file:

```
azuredeploy.json    2023-03-21 12:55:19.458971221 -0400
+++ azuredeploy9010.json        2023-03-28 15:38:31.476627274 -0400
@@ -269,6 +269,18 @@
                     "diagnosticsLatencyThresholdInMS": {
                         "value": "[parameters('diagnosticsLatencyThresholdInMS')]"
                     },
+                    "readproportion": {
+                        "value": "0.9"
+                    },
+                    "updateproportion": {
+                        "value": "0"
+                    },
+                    "scanproportion": {
+                        "value": "0"
+                    },
+                    "insertproportion": {
+                        "value": "0.1"
+                    },
                     "writeOnlyOperation": {
                         "value": false
                     },
```

3. Commit these changes.

1. Microsoft, "Indexing policies in Azure Cosmos DB," accessed April 11, 2023, https://learn.microsoft.com/en-us/azure/cosmos-db/index-policy.
2. Microsoft, "Indexing policies in Azure Cosmos DB."

Creating an Azure Cosmos DB for NoSQL database

1. Create an Azure resource group in region East US:
    a. Login into the Azure portal.
    b. Go to the Azure Resource Group browser.
    c. Click Create.
    d. Add the subscription name, name the resource group, and set the region.
    e. Click Review + create.
    f. Click Create.
2. Create a storage account to hold the results of each run:
    a. Go to the Azure Storage Account dashboard.
    b. Click Create.
    c. Add the subscription name, select the resource group from step 1, name the storage account, and set the region.
    d. Click Review.
    e. Click Create.
    f. From the Storage Account overview page, click Access Keys.
    g. For the key1 connection string, click Show.
    h. Copy and save the Storage Account Connection string.
3. Create an Azure Cosmos DB for NoSQL Account:
    a. From the Azure portal, click Azure Cosmos DB.
    b. Click Create.
    c. In the Azure Cosmos DB for NoSQL box, click Create.
    d. Add the subscription name, select the resource group from step 1, name the account, and set the region.
    e. Click Create+review.
    f. Click Create.
    g. On the Azure Cosmos DB Account overview, page, click Keys.
    h. Select the Read-write Keys tab.
    i. Copy and save the URI (e.g., https://DDDD.documents.azure.com:443/).
    j. Copy and save the PRIMARY KEY.
4. From the Azure Cosmos DB Account Overview page, select Cost Management:
    a. Enter the maximum value of Request Units (RU/s) needed for the workload. Consult Table 9.
    b. Click SAVE.
5. Create an Azure Cosmos DB for NoSQL container:
    a. From the Azure Cosmos DB Account Overview page, click New container.
    b. In the pop-up, enter ycsb for the database id.
    c. For the container id, enter usertable.
    d. Set the partition id to /id.
    e. Under Container throughput, select Manual.
    f. For the maximum RU/s for the workload, consult Table 7, and enter it.
    g. If necessary, to accept the estimated cost for the container, click the box.
    h. Click OK.
6. Wait for the container to be created. Note: For the largest workloads, we waited an additional 30 minutes before running a test.

Table 9: Number of YCSB threads needed for a workload's test.

Workload	Request Units/s
100 percent reads at 10,000 OPS	12,000
100 percent writes at 10,000 OPS	130,000
100 percent updates at 10,000 OPS	161,900
100 percent reads at 30,000 OPS	36,000
100 percent writes at 30,000 OPS	390,000
100 percent updates at 30,000 OPS	485,700
100 percent reads at 50,000 OPS	60,000
100 percent writes at 50,000 OPS	650,000
100 percent updates at 50,000 OPS	809,400
90 percent reads, 10 percent writes at 50,000 OPS	120,000
100 percent reads at 1,000,000 OPS	2,000,000
100 percent writes at 1,000,000 OPS	20,000,000

Running the workload on Azure Cosmos DB for NoSQL

1. Load the workload's deployment template into the Azure editor.
    a. For all workloads except for the 90 percent reads and 10 percent writes at 50,000 OPS workload, paste the URL found in Table 10 into a browser.
    b. For the 90 percent reads and 10 percent writes at 50,000 OPS workload, load the deployment template via these steps.
        i. From the Azure portal, click Deploy a Custom Template.
        ii. Click Build your own template in the editor.
        iii. Click Load file.
        iv. Select the file azuredeploy9010.json.
        v. Click Save.
2. Select the correct subscription.
3. Select the resource group created in step 1 in the previous section.
4. Enter the Storage connection string from 2h in the previous section into Results Storage Connection String.
5. Enter the Cosmos URI from step 3i in the previous section into Cosmos URI.
6. Enter the Cosmos PRIMARY KEY from step 3j in the previous section into Cosmos Key.
7. Enter a password for the account "benchmarking" for the VM.
8. In the line Ycsb Git Hub Repo Name, replace Azure with your GitHub account name.
9. Do the same for the line Benchmarking Tools Repo Name.
10. Note: For the 1,000,000 OPS workloads, we changed the VM type to Standard_D64ds_v5.s.
11. Click Review+Create.
12. Click Create.
13. The deployment will automatically copy test results to a unique blob container under your storage account. Use the Storage browser to view them.

Table 10: Azure template URLs for each workload.

Workload	URL
100 percent reads at 10,000 OPS	https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2FAzure%2Fazure-db-benchmarking%2Fmain%2Fcosmos%2Fsql%2Ftools%2Fjava%2Fycsb%2Frecipes%2Fread%2F10-thousand-rps-read%2Fazuredeploy.json
100 percent writes at 10,000 OPS	https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2FAzure%2Fazure-db-benchmarking%2Fmain%2Fcosmos%2Fsql%2Ftools%2Fjava%2Fycsb%2Frecipes%2Fwrite%2F10-thousand-rps-write%2Fazuredeploy.json
100 percent updates at 10,000 OPS	https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2FAzure%2Fazure-db-benchmarking%2Fmain%2Fcosmos%2Fsql%2Ftools%2Fjava%2Fycsb%2Frecipes%2Fupdate%2F10-thousand-rps-update%2Fazuredeploy.json
100 percent reads at 30,000 OPS	https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2FAzure%2Fazure-db-benchmarking%2Fmain%2Fcosmos%2Fsql%2Ftools%2Fjava%2Fycsb%2Frecipes%2Fread%2F30-thousand-rps-read%2Fazuredeploy.json
100 percent writes at 30,000 OPS	https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2FAzure%2Fazure-db-benchmarking%2Fmain%2Fcosmos%2Fsql%2Ftools%2Fjava%2Fycsb%2Frecipes%2Fwrite%2F30-thousand-rps-write%2Fazuredeploy.json
100 percent updates at 30,000 OPS	https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2FAzure%2Fazure-db-benchmarking%2Fmain%2Fcosmos%2Fsql%2Ftools%2Fjava%2Fycsb%2Frecipes%2Fupdate%2F30-thousand-rps-update%2Fazuredeploy.json
100 percent reads at 50,000 OPS	https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2FAzure%2Fazure-db-benchmarking%2Fmain%2Fcosmos%2Fsql%2Ftools%2Fjava%2Fycsb%2Frecipes%2Fread%2F50-thousand-rps-read%2Fazuredeploy.json
100 percent writes at 50,000 OPS	https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2FAzure%2Fazure-db-benchmarking%2Fmain%2Fcosmos%2Fsql%2Ftools%2Fjava%2Fycsb%2Frecipes%2Fwrite%2F50-thousand-rps-write%2Fazuredeploy.json
100 percent updates at 50,000 OPS	https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2FAzure%2Fazure-db-benchmarking%2Fmain%2Fcosmos%2Fsql%2Ftools%2Fjava%2Fycsb%2Frecipes%2Fupdate%2F50-thousand-rps-update%2Fazuredeploy.json
90 percent reads, 10 percent writes at 50,000 OPS	N/A. See instructions
100 percent reads at 1,000,000 OPS	https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2FAzure%2Fazure-db-benchmarking%2Fmain%2Fcosmos%2Fsql%2Ftools%2Fjava%2Fycsb%2Frecipes%2Fread%2F1-million-rps-read%2Fazuredeploy.json
100 percent writes at 1,000,000 OPS	https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2FAzure%2Fazure-db-benchmarking%2Fmain%2Fcosmos%2Fsql%2Ftools%2Fjava%2Fycsb%2Frecipes%2Fwrite%2F1-million-rps-write%2Fazuredeploy.json

We deleted the Azure Cosmos DB container after each run, and created a new one before performing the next workload. Delete the container by going to the Azire Cosmos DB Account Data Explorer, select delete from the pop-down menu, and confirm the deletion request.

DynamoDB on AWS testing

We created all resources in the same AWS region, Us-East-1.

Creating a DynamoDB table

1. Log into AWS console and set the default region to Us-East-1.
2. Proceed to the EC2 instances dashboard, and click Create Table.
3. Name the table usertable.
4. Set the primary key to be firstname.
5. Click Customize Settings.
6. Click Off to disable auto-scaling for Read Capacity.
7. In the "Provisioned capacity units" field, enter the RCU value for this workload from Table 11.
8. Click Off to disable auto-scaling for Write Capacity.
9. In the "Provisioned capacity units" field, enter the WCU value for this workload from Table 11.
10. Click Create Table.

Table 11: Resource capacity values for DynamoDB testing by workload.

Workload	RCU	WCU
100 percent reads at 10,000 OPS	7,000	100
100 percent writes at 10,000 OPS	1000	22,000
100 percent updates at 10,000 OPS	7,000	22,000
100 percent reads at 30,000 OPS	17,000	100
100 percent writes at 30,000 OPS	1000	62,000
100 percent updates at 30,000 OPS	17,000	62,000
100 percent reads at 50,000 OPS	27,000	400
100 percent writes at 50,000 OPS	1000	102,000
100 percent updates at 50,000 OPS	27,000	102,000
90 percent reads, 10 percent writes at 50,000 OPS	27,000	1,020

Creating one AWS VM and installing the YCSB framework on it

1. Log into AWS console, and set the default region to Us-East-1.
2. Proceed to the DynamoDB home, and click Launch Instances.
    a. Add a name.
    b. Click on the Ubuntu OS icon, and select the Ubuntu Server 22.04 LTS on 64-bit (x86) from Canonical.
    c. Select the instant type for the workload.
        i. Use m6i.2xlarge for the 10,000 OPS workloads.
        ii. Use m6i.8xlarge for the 30,000 OPS workloads.
        iii. Use m6i.12xlarge for the 50,000 OPS workloads.
    d. Create and save a new SSH keypair.
    e. Set "Allow SSH traffic from" to My IP.
    f. Use one 20 GB gp2 volume for the root volume.
    g. Click Launch instance.
3. After the instance starts, view its details to get its public IP address.
4. Log onto the VM using SSH as user ubuntu with the private key and IP address.
5. Update the operating system:

```
sudo apt update
sudo apt upgrade
```

6. Download the Oracle Java Development Kit for version 8 build 351 (jdk-8u351-linux-x64.tar.gz), from https://www.oracle.com/java/technologies/javase/javase8u211-later-archive-downloads.html.
7. Install the JDK into /usr/lib/jvm:

```
sudo mkdir -p /usr/lib/jvm
sudo tar -xf jdk-8u351-linux-x64.tar.gz -C /usr/lib/jvm
```

8. Make this version the system's default Java:

```
sudo update-alternatives --install "/usr/bin/java" "java" \
  "/usr/lib/jvm/jdk1.8.0_351/bin/java" 1
sudo update-alternatives --set java /usr/lib/jvm/jdk1.8.0_351/bin/java
```

9. Install maven, and Python 2:

```
sudo apt install maven python2 unzip
```

10. Clone YCSB from its git repository. Add explicit reference to Python 2:

```
cd ~
git clone https://github.com/brianfrankcooper/YCSB.git
sed -i 'sZ#!/usr/bin/env pythonZ#!/usr/bin/env python2#' YCSB/bin/ycsb
```

11. Update the YCSB configuration to use the latest version of the DynamoDB Java SDK:

```
sed -i 'sZ<version>1.11.812</version>Z<version>1.11.1000</version>Z' ~/YCSB/dynamodb/pom.xml
```

12. Compile the YCSB code for DynamoDB:

```
cd ~/YCSB
mvn -pl site.ycsb:dynamodb-binding -am clean package
```

13. Modify YCSB's configuration file for DynamoDB for all workloads per the following patch file:

File: ~/patch-dynamodb-conf.txt

```
diff -u YCSB/dynamodb/conf/dynamodb.properties*
--- YCSB/dynamodb/conf/dynamodb.properties      2023-03-15 12:03:20.558018824 -0400
+++ YCSB/dynamodb/conf/dynamodb.properties-orig     2023-01-26 15:40:59.358602862 -0500
@@ -19,10 +19,10 @@
 ## Mandatory parameters

 # AWS credentials associated with your aws account.
-dynamodb.awsCredentialsFile = /home/ubuntu/YCSB/dynamodb/conf/AWSCredentials.properties
+#dynamodb.awsCredentialsFile = <path to AWSCredentials.properties>

 # Primarykey of table 'usertable'
-dynamodb.primaryKey = firstname
+#dynamodb.primaryKey = <firstname>

 # If you set dynamodb.primaryKeyType to HASH_AND_RANGE, you must specify the
 # hash key name of your primary key here. (see documentation below for details)
@@ -62,7 +62,7 @@
 # know the performance capabilities of a single logical hash partition so
 # they can plan their application accordingly.

-dynamodb.primaryKeyType = HASH
+#dynamodb.primaryKeyType = HASH

 #Optionally you can specify a value for the hash part of the primary key
 #when testing in HASH_AND_RANG mode.
```

```
@@ -71,7 +71,7 @@
 # AWS Region code to connect to:
 # https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.
html#concepts-available-regions
 # Set this parameter, unless you are using the default value ('us-east-1).
-dynamodb.region = us-east-1
+#dynamodb.region = us-east-1

 # Endpoint to connect to. If not set, the endpoint will be set automatically
 # based on the region and for HTTP connections. When using a non-standard
@@ -80,14 +80,14 @@
 #dynamodb.endpoint = http://dynamodb.us-east-1.amazonaws.com

 # Strongly recommended to set to uniform.Refer FAQs in README
-requestdistribution = uniform
+#requestdistribution = uniform

 # Enable/disable debug messages.Defaults to false
 # "true" or "false"
 #dynamodb.debug = false

 # Maximum number of concurrent connections
-dynamodb.connectMax = 310
+#dynamodb.connectMax = 50

 # Read consistency.Consistent reads are expensive and consume twice
 # as many resources as eventually consistent reads. Defaults to false.Create the DynamoDB table
```

14. Generate AWS access credentials:
    a. From the AWS console, go to the IAM dashboard.
    b. Click Manage access keys.
    c. Click Create access key.
    d. Click Download .csv file.
    e. Click Done.
15. Add AWS credentials in the CSV file to the YCSB configuration per the following patch file:

    File: aws-creds-patch.txt

```
diff --git a/dynamodb/conf/AWSCredentials.properties b/dynamodb/conf/AWSCredentials.properties
index e337b391..d38e2ea9 100644
--- a/dynamodb/conf/AWSCredentials.properties
+++ b/dynamodb/conf/AWSCredentials.properties
@@ -15,5 +15,5 @@

 # Fill in your AWS Access Key ID and Secret Access Key
 # http://aws.amazon.com/security-credentials
-#accessKey =
-#secretKey =
+accessKey = AKIA3MJHXZVTWLPY6PGE
+secretKey = nVhzo8I2m4Zvtco0VHC7VjcVWslU6AxCaakllUSk
```

Running the workload on DynamoDB

When the usertable table was in an active state on the DynamoDB table dashboard, we performed a workload test.

In the following, replace each XX with 10, 30, or 50 for the workload's target rates for 10,000; 30,000; or 50,000 OPS, respectively.

Consult Table 12 to get the number of YCSB threads needed for the run. Replace YY by that value.

We placed common YCSB workload properties in files, such as workload-10k. See the end of the section for details.

1. For 100 percent read workloads.
    a. Prepare the table by loading data:

```
cd ~/YCSB
./bin/ycsb load dynamodb -s -P ../workload-XXk -P ../workload-reads \
  -P dynamodb/conf/dynamodb.properties  -target 100 -t YY|& tee -a loading.txt
```

    b. Run the test. The results are displayed on the screen and written to the file results.txt.

```
cd ~/YCSB
./bin/ycsb load dynamodb -s -P ../workload-XXk -P ../workload-reads \
  -P dynamodb/conf/dynamodb.properties  -target XX000 -t YY|& tee -a results.txt
```

2. For 100 percent update workloads.
    a. Prepare the table by loading data:

```
cd ~/YCSB
./bin/ycsb load dynamodb -s -P ../workload-XXk -P ../workload-updates \
  -P dynamodb/conf/dynamodb.properties  -target 100 -t YY|& tee -a loading.txt
```

    b. Run the test. The results are displayed on the screen and written to the file results.txt.

```
cd ~/YCSB
./bin/ycsb load dynamodb -s -P ../workload-XXk -P ../workload-updates \
  -P dynamodb/conf/dynamodb.properties  -target XX000 -t YY |& tee -a results.txt
```

3. For 100 percent write workloads, we do not need to load data. Run the test:

```
cd ~/YCSB
./bin/ycsb load dynamodb -s -P ../workload-XXk -P ../workload-writes \
  -P dynamodb/conf/dynamodb.properties  -target XX000 -t YY |& tee -a results.txt
```

4. For 90 percent read 10 percent write workloads.
    a. Prepare the table by loading data:

```
cd ~/YCSB
./bin/ycsb load dynamodb -s -P ../workload-XXk -P ../workload-read-writes \
  -P dynamodb/conf/dynamodb.properties  -target 100 -t YY|& tee -a loading.txt
```

    b. Run the test. The results are displayed on the screen and written to the file results.txt.

```
cd ~/YCSB
./bin/ycsb load dynamodb -s -P ../workload-XXk -P ../workload-read-writes \
  -P dynamodb/conf/dynamodb.properties  -target XX000 -t YY|& tee -a results.txt
```

We deleted the DynamoDB table at the end of each run, and created a new one before performing the next workload. Delete the table by selecting the table of the DynamoDB home, click Delete, and confirm the deletion request.

File: workload-10k

```
workload=site.ycsb.workloads.CoreWorkload
table=usertable
maxexecutiontime=3800
insertorder=hashed
requestdistribution=uniform
operationcount=36000000
```

File: workload-30k

```
workload=site.ycsb.workloads.CoreWorkload
table=usertable
maxexecutiontime=3800
insertorder=hashed
requestdistribution=uniform
operationcount=108000000
```

File: workload-50k

```
workload=site.ycsb.workloads.CoreWorkload
table=usertable
maxexecutiontime=3800
insertorder=hashed
requestdistribution=uniform
operationcount=180000000
```

File: workload-reads

```
readproportion=1.0
updateproportion=0.0
insertproportion=0.0
recordcount=1000
```

File: workload-updates

```
readproportion=0.0
updateproportion=1.0
insertproportion=0.0
recordcount=100000
```

File: workload-writes

```
readproportion=0.0
updateproportion=0.0
insertproportion=1.0
```

File: workload-read-writes

```
readproportion=0.9
updateproportion=0.0
insertproportion=0.1
recordcount=5000
threadcount=200
```

Table 12: Number of YCSB threads needed for a workload's test.

Workload	Number of YCSB client threads
100 percent reads at 10,000 OPS	23
100 percent writes at 10,000 OPS	60
100 percent updates at 10,000 OPS	60
100 percent reads at 30,000 OPS	135
100 percent writes at 30,000 OPS	180
100 percent updates at 30,000 OPS	180
100 percent reads at 50,000 OPS	200
100 percent writes at 50,000 OPS	300
100 percent updates at 50,000 OPS	300
90 percent reads, 10 percent writes at 50,000 OPS	200

Read the report at https://facts.pt/48qgf0S

This project was commissioned by Microsoft.

Principled Technologies®

Facts matter.®

Principled Technologies is a registered trademark of Principled Technologies, Inc.
All other product names are the trademarks of their respective owners.

DISCLAIMER OF WARRANTIES; LIMITATION OF LIABILITY:
Principled Technologies, Inc. has made reasonable efforts to ensure the accuracy and validity of its testing, however, Principled Technologies, Inc. specifically disclaims any warranty, expressed or implied, relating to the test results and analysis, their accuracy, completeness or quality, including any implied warranty of fitness for any particular purpose. All persons or entities relying on the results of any testing do so at their own risk, and agree that Principled Technologies, Inc., its employees and its subcontractors shall have no liability whatsoever from any claim of loss or damage on account of any alleged error or defect in any testing procedure or result.

In no event shall Principled Technologies, Inc. be liable for indirect, special, incidental, or consequential damages in connection with its testing, even if advised of the possibility of such damages. In no event shall Principled Technologies, Inc.'s liability, including for direct damages, exceed the amounts paid in connection with Principled Technologies, Inc.'s testing. Customer's sole and exclusive remedies are as set forth herein.