CIS 4010 Assignment 1, Task 3
Christian Cornelis - ccorneli@uoguelph.ca

Implementation

Task 1
Implementing this task for Amazon Web Services (AWS) was much easier in my opinion. The syntax for S3 resource and client methods in the Python SDK was very easy to understand, with a *lot* of documentation and examples to help online. Handling file upload and download operations especially was a much simpler task in AWS than was the case in Azure. In Azure, a file had to be opened in order to be uploaded or downloaded, which I found to be confusing to use. The Azure SDK also did not provide a lot of clarification in the method headers regarding which exceptions could be thrown by failed operations, which added to the frustration using it.

Task 2
Implementing this task was far easier with Azure's CosmosDB Tables. Querying was extremely easy to perform and all parameters to queries were comma-separated strings for the most part. One caveat that I felt was quite the roadblock at first were the restrictions on what characters the RowKey can contain. This led to my having to replace forward-slashes and question marks with other text patterns to avoid exceptions being thrown, which led to increased difficulty in querying for titles via my script. In AWS, I was quite fond of the Attr() class once I got used to it, but chaining queries was a little more difficult since I was working with DynamoDB classes as queries, as opposed to simple strings as was the case with Azure Table.

Performance

Task 1

Task	**AWS Time (s)**	**Azure Time (s)**
Create all containers and objects	22.944	11.794
Connect to populated containers.	0.651	0.484
List objects in all containers	1.083	0.459
List objects in a specific container	0.456	0.122
List objects with a specific name	3.013	0.304
Download a specific object	0.970	0.491

Overall, Azure appeared to be substantially faster in almost all measured tests for this task, aside from connecting to containers that were already created, which is essentially only reliant on network speeds.

Task 2

Task	AWS Time (s)	Azure Time (s)
Create and populate table	238.139	239.096
Connect to populated table.	0.27	0.44
Query with no filters*	5.965	4.619
Query with primary key > 1980 as only filter*	5.709	4.927
Query with primary key > 1980, rank gt 7*	6.940	6.428
Query with 1980 < primary key < 2015 and rating gt 8, 7 attributes**	1.746	0.581

* - attributes retrieved were year, title, rank, plot, actors, directors, sorting by primary key.
** - attributes retrieved were year, title, rank, rating, plot, actors, directors, sorting by rating

Overall, Azure appears to have a slight edge when it comes to querying large sets of data. For the most part, the two services appear to be quite comparable. This was quite surprising to me considering DynamoDB sends more requests due to the paginating-nature of the scan() method. Downloading was not measured, as saving to a CSV utilized the response from the query, and would not be indicative of the speed of either service.

Permissions

Task 1 and Task 2
As permissions had to be set in the same way for both tasks, I will address both in one section. Permissions across both the AWS and Azure SDKs were not overly-difficult to set up. For Azure, all that was required was getting a connection string associated with the storage account used by my account, and saving it in an environment variable. This environment variable could then be referenced via my Python scripts and used to validate my connection to the Blob and CosmosDB clients via the SDK.

In AWS, a similar task had to be done to validate my connection. My credentials had to be retrieved from the AWS console and pasted into ~/.aws/credentials. Creating a client using the SDK would automatically validate my connection to my AWS account using the previously-mentioned credentials file. I found that this was slightly easier as the SDK automatically invoked a valid connection so long as the credentials were valid; however, I did find it very annoying that my session token expire every time I log out, which requires me to re-paste my credentials locally in order for my script to work. I was not able to determine a way to check that credentials were valid, so invalid credentials cause my AWS scripts to raise an exception and terminate.

Available Documentation and Tutorials

Task 1
I found that there was a lot of documentation available for this task online. Microsoft's online Azure Quickstart docs were a huge help, as they provided examples on how to perform most of the required

operations for this task. I didn't find I needed to look up any tutorials for the Azure implementation due to the detail that the aforementioned quickstart guide contained. For AWS, the boto3 documentation proved to be extremely useful in that it also contained all documentation with tutorials for using the Python SDK.

Task 2
There was an abundance of documentation online for task 2 for both Azure's CosmosDB service and AWS's DynamoDB service. I found that Azure's implementation of querying in the CosmosDB Python SDK was much simpler, which required far less referencing of the documentation. AWS, on the other hand, had me searching through the documentation for long periods of time attempting to find solutions to issues that I had with the scan() method. All information needed to fix my issue was in the documentation, but I found that it got buried quite easily in the heavily-detailed guides to each method that are available. Both services' documentation had code snippets showing how to implement features of their respective SDKs, which was very useful, and Azure also had a lot of documentation on the GitHub repo of their Azure CosmosDB Python SDK.

References

[1] Microsoft, "Quickstart: Azure Blob storage library v12 - Python" https://docs.microsoft.com/en-us/azure/storage/blobs/storage-quickstart-blobs-python, 2019, online; accessed 20 Jan 2020.
[2] Amazon Web Services Inc. "Uploading Files" https://boto3.amazonaws.com/v1/documentation/api/latest/guide/s3-uploading-files.html, 2019, online; accessed 19 Jan 2020.
[3] StackOverflow. "How can I easily determine if a Boto 3 S3 bucket resource exists?" https://stackoverflow.com/a/26871885, 2014, online; accessed 20 Jan 2020.
[4] StackOverflow. "check if a key exists in a bucket in s3 using boto3" https://stackoverflow.com/a/34562141, 2015, online; accessed 20 Jan 2020.
[5] StackOverflow. "What is the Python equivalent for a case/switch statement? [duplicate]" https://stackoverflow.com/a/11479840, 2014, online; accessed 19 Jan 2020.
[6] Amazon Webservices Inc. "S3 - Boto 3 Docs 1.11.9 documentation" https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/s3.html, 2019, online; accessed 20 Jan 2020.
[7] D. Stacey, "CIS 4010 AWS and Azure Course Notes/Sample Code", https://courselink.uoguelph.ca/d2l/home/606015, 2020, online; accessed 20 Jan 2020.
[8] StackOverflow. "How would you make a comma-separated string from a list of strings? "https://stackoverflow.com/a/44781, online, 2008; accessed 24 Jan 2020.
[9] Microsoft. "TableService class" "https://docs.microsoft.com/en-ca/python/api/azure-cosmosdb-table/azure.cosmosdb.table.tableservice.tableservice?view=azure-python#create-table-table-name--fail-on-exist-false--timeout-none-, online, unknown; accessed 25 Jan 2020.
[10] Microsoft. "Get started with Azure Table storage and the Azure Cosmos DB Table API using Python" https://docs.microsoft.com/en-us/azure/cosmos-db/table-storage-how-to-use-python, online, unknown; accessed 24 Jan 2020.
[11] GeeksForGeeks. "Ways to sort list of dictionaries by values in Python – Using lambda function" https://www.geeksforgeeks.org/ways-sort-list-dictionaries-values-python-using-lambda-function/, online, unknown; accessed 24 Jan 2020.
[12] Microsoft. "Querying tables and entities" https://docs.microsoft.com/en-us/rest/api/storageservices/querying-tables-and-entities, online, unknown; accessed 24 Jan 2020.

[13]PyPi. "Pretty Table 0.7.2" https://pypi.org/project/PrettyTable/, online, 2013; accessed 25 Jan 2020.
[14] GitHub. "Azure/azure-cosmos-python" https://github.com/Azure/azure-cosmos-python, online, 2019; accessed 24 Jan 2020.