Table of Contents

Intro

Jun Zhen | CSS 436 | Professor Robert Dimpsey | Program 4

How to run program

To view and run the program please visit this URL:
https://junzhencss436prog4.azurewebsites.net

Query Button – Query the table base for given the first name and last name
Load Button – Get the text file from Professor Dimpsey's URL and store them into my Azure Blob storage. Then from Blob storage store into Azure Table storage.
Clear Button – Wipe the entire Blob storage and Table Storage.

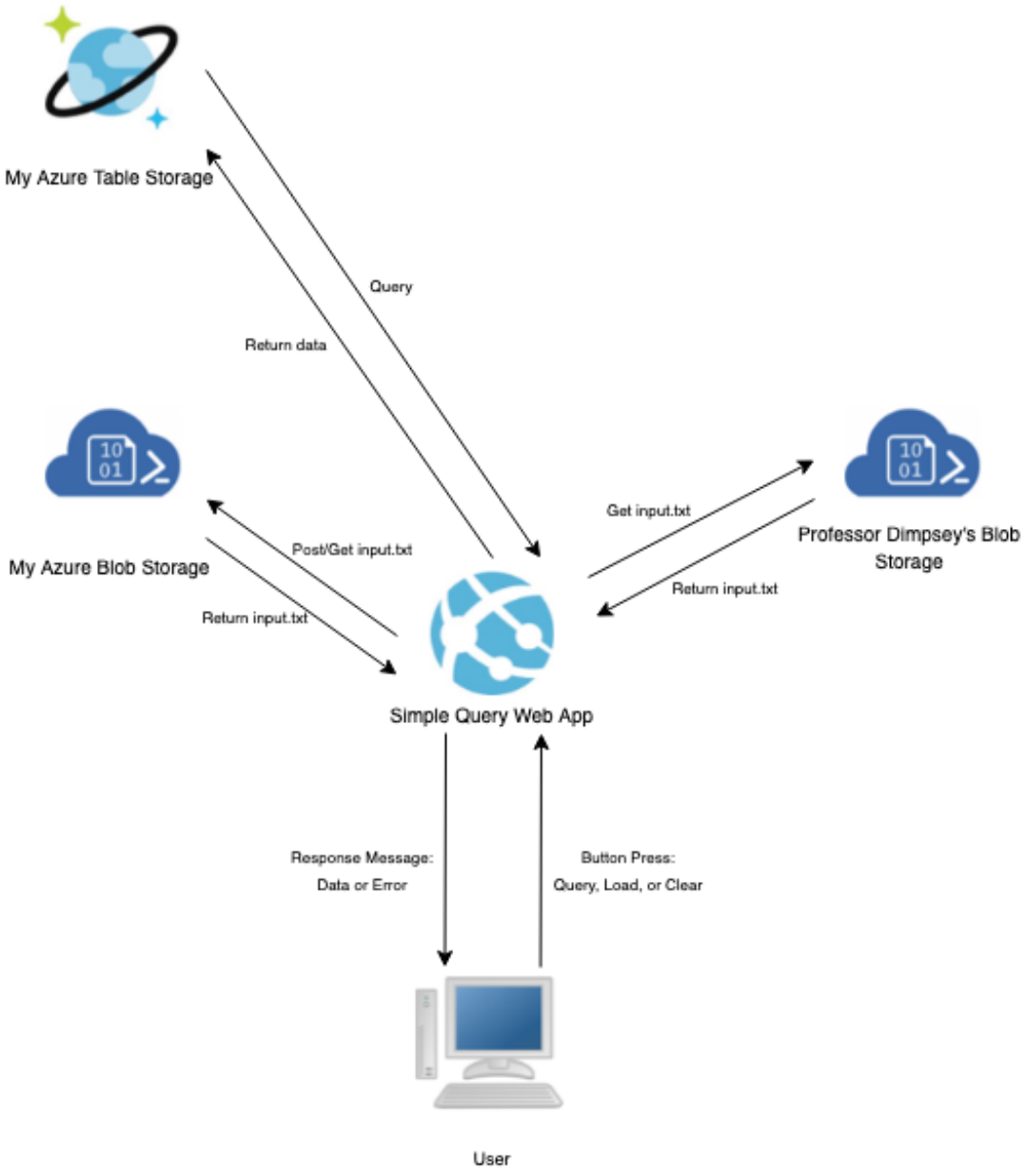
The location of my storage:
Blob Storage: https://css436prog4storage.blob.core.windows.net/program4inputs/textblob.txt
Table Storage: https://css436prog4storage.table.core.windows.net/prog4table

Design Logic

The web application is built with JavaScript and html. Specifically, with ExpressJS and NodeJS. When a user visits the URL, they will take to the landing page, index.html. Here they will have the option to query, load data, and clear data. The ExpressJS framework will provide all the routing logic when the client and server communicate. Each press of a button will hit a different end point of my server. This web application is then hosted on Microsoft Azure's web app service and connected to their Storage Service.

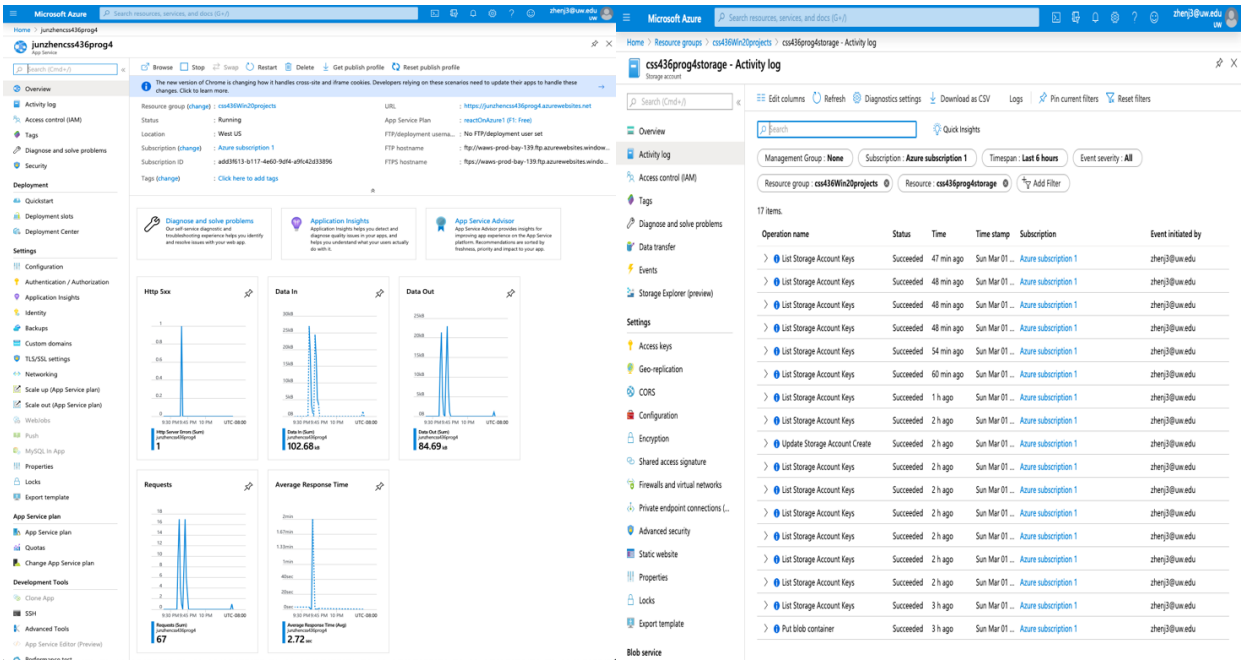
The ideal scalability of my web server is to scale out. The reason is because the server is running on NodeJS. NodeJS is asynchronized and single threaded. By scaling out, we can handle more clients while scaling up will yield little to no benefits.

Design diagram



Monitoring

Majority of the monitoring will be done primary through my azure portal web page. This web page gives us a view of all the activities and traffic of our program. Graphs that show data in, data out, http requests, and average response time. Azure portal also allows me to create alerts, which will notify me whenever there is an outage on one of my dependency. I could set alerts that tell me if my server is reaching capacity. For the storage, I could see all the data that are loaded into my database and out of my database.



Furthermore, azure provides each application with a feature called Application Insights. It is an extensible Application Performance Management server for developers. It is used to monitor live applications, detects performance anomalies, and have powerful analytics. With Application insights, I can find out which page are most popular, at what time of day, and where users are. Given this information, I can configure out my service plans to the demand of my web app. This feature also allows me to see dependency rates which I can debug and pinpoint where to improve my services.

Scalability

Azure Web Apps offer 5 tiers of services for hosting web applications: Free, Basic, Standard, Premium, and Isolated.

	FREE **Try for free**	BASIC **Dedicated environment for dev/test**	STANDARD **Run production workloads**	PREMIUM **Enhanced performance and scale**	ISOLATED **High-Performance, Security and Isolation**
Web, mobile, or API apps	10	Unlimited	Unlimited	Unlimited	Unlimited
Disk space	1 GB	10 GB	50 GB	250 GB	1 TB
Maximum instances	–	Up to 3	Up to 10	Up to 30**	Up to 100*
Custom domain	–	Supported	Supported	Supported	Supported
Auto Scale	–	–	Supported	Supported	Supported
VPN hybrid connectivity	–	–	Supported	Supported	Supported
Network Isolation					Supported
Price	Free	Promotional Price $0.018/hour	$0.095/hour	$0.111/hour	$0.38/hour

Specifically, for our web application, I choice the free service plan. This service plan grants no multiple virtual machine instances and only able to handle 10 clients at once. To able to get benefit of scalability, I would have to upgrade my service plan to a more premium plan such as the standard service plan that provides up to 10 multiple virtual machine instances.

The easiest way for any websites to handle high traffic load is to either scale out or scale up. By scaling up, we would increase the capability of our servers. Scaling out mean we would increase the amount of our services and set up a load balancer to direct traffic evenly. The beauty of the cloud is that all of these will be handle by Microsoft. Clients will able to help if they allow our web app to be cached in their local machines.

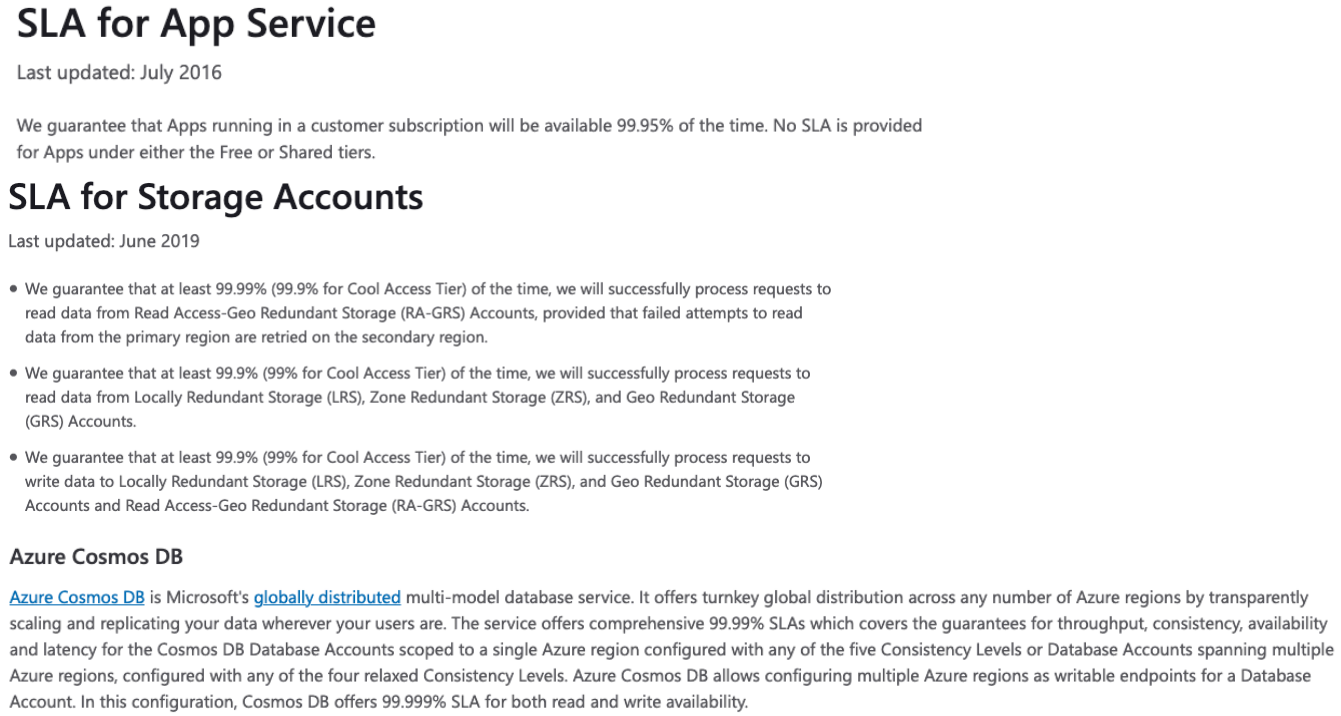
As for my database, azure table storage configuration allows automatically adjust the read and write capacity. There is no need for any configuration with table itself because it is a NoSQL database. New entries will not ‘break’ the database because every entry are key-value pairs. I expect the durability of my database will be intact as the traffic load increase. With azure’s storage, all my data are copied and storage in 3 different location within the same location to further increase durability.

SLA

Our application depends four different services. They are my own azure App service, blob storage, table storage, and the last one is provided by Professor Dimpsey's blob storage. All of them has different availability rate. According to Microsoft Azure's documentation the availability for their services are:

My Azure App Service is 99.95%
My Azure Blob Storage is 99.9%
My Azure Table Storage is 99.99%
Professor Dimpsey's Azure Blob Storage is 99.9%

To get the total availability rate we would multiple them together and get 0.997402249 availability.



SLA for App Service

Last updated: July 2016

We guarantee that Apps running in a customer subscription will be available 99.95% of the time. No SLA is provided for Apps under either the Free or Shared tiers.

SLA for Storage Accounts

Last updated: June 2019

- We guarantee that at least 99.99% (99.9% for Cool Access Tier) of the time, we will successfully process requests to read data from Read Access-Geo Redundant Storage (RA-GRS) Accounts, provided that failed attempts to read data from the primary region are retried on the secondary region.
- We guarantee that at least 99.9% (99% for Cool Access Tier) of the time, we will successfully process requests to read data from Locally Redundant Storage (LRS), Zone Redundant Storage (ZRS), and Geo Redundant Storage (GRS) Accounts.
- We guarantee that at least 99.9% (99% for Cool Access Tier) of the time, we will successfully process requests to write data to Locally Redundant Storage (LRS), Zone Redundant Storage (ZRS), and Geo Redundant Storage (GRS) Accounts and Read Access-Geo Redundant Storage (RA-GRS) Accounts.

Azure Cosmos DB

Azure Cosmos DB is Microsoft's globally distributed multi-model database service. It offers turnkey global distribution across any number of Azure regions by transparently scaling and replicating your data wherever your users are. The service offers comprehensive 99.99% SLAs which covers the guarantees for throughput, consistency, availability and latency for the Cosmos DB Database Accounts scoped to a single Azure region configured with any of the five Consistency Levels or Database Accounts spanning multiple Azure regions, configured with any of the four relaxed Consistency Levels. Azure Cosmos DB allows configuring multiple Azure regions as writable endpoints for a Database Account. In this configuration, Cosmos DB offers 99.999% SLA for both read and write availability.

For the durability rate for our application is all depended on the storage. The plan is I opt for is locally redundant storage (LRS) which provide three synchronous copies within a single physical location. Because of this 3 copies backup, Azure Storage promises 11 9's for their storage durability. For here I'll be assuming Professor Dimpsey's blob storage will have the same service plan as I do.

To get the total durability rate, we would multiple them together and get 0.99999999997 durability.

Final Result:
Availability: 99.74%
Durability: 99.99999999%

Sources

https://azure.microsoft.com/en-us/support/legal/sla/storage/v1_5/
https://azure.microsoft.com/en-us/support/legal/sla/app-service/v1_4/
https://docs.microsoft.com/en-us/azure/azure-monitor/overview'
https://azure.microsoft.com/en-us/blog/scaling-up-and-scaling-out-in-windows-azure-web-sites/
https://aws.amazon.com/s3/sla/
https://azure.microsoft.com/en-us/support/legal/sla/summary/