Project Management Plan
CosmosDB RU Cost Calculation Improvement - Efficiency & Time

Version 1.1

October 3, 2018

Team Members: Joe Do, Ferdinand Tembo, Kevin Tran

https://docs.google.com/document/d/1fwJZauK1jdk3cqosSPZe-YeOyBjWVNZ3gPE2FipCRwc/edit?usp=sharing

Bellevue College - Computer Science Department

Alfred Nehme

Revisions Page

Version	Primary Author	Description of Version	Date Completed
1.0	Kevin Tran	First Draft - Interpretation	10/03/2018
1.1			10//2018

Table of Contents

1. Introduction

1.1 Project Overview

This is the Software Project Management Plan Document that is designed to inform pertinent background, planning, and design details to all parties involved in developing and improving an important function for Cosmos DB - estimate the cost for using Cosmos DB based on user’s workload.

Cosmos DB is a highly reliable cloud application that still requires some adjustments in order to better fulfill its reliability and clarity with customers. For customers, there is an ever-growing concern with transparency, especially with monetary costs. Thus this project is focused on aiding the financial aspect of Cosmos DB.

In the case of this project, we are working with Cosmos DB and we are trying to sharpening its efficiency and accuracy of one of its side functionalities: the estimation calculator of Request Unit throughput. When customers are establishing their cloud database with Cosmos DB, they are required to input an amount of Reserve Unit Throughput (RU/s’s) when attaching a new collection into an Azure Cosmos DB account. Inputting an accurate number will allow the customer to create informed decisions and act accordingly with the accurate calculation. If they were to provide a far estimate of their actual need of Request Unit throughput, then they would either overpay for unnecessary additional throughput or experience sudden additional delays by rate-limiting.

1.2 Literature Review

Cosmos DB is a multidimensional NoSQL database that offers competitive benefits for the end-user. It is a database that can utilize multiple API’s, be easily implemented with multiple SDK’s, offers quick and reliable data transactions whether during an earthquake or during peak use, simplify/reduce data-storage management costs and concerns, and is able to scale dynamically with business needs. Cosmos DB also offers individuals the option to store data using any of the four basic NoSQL storage types: Key-Value, Document, Wide-Column, and Graph.

In terms of the problem, we need to realize that Request Unit throughput will vary between operational requests that are being performed. Request Unit totals are determined by variables within a given Cosmos DB container’s and locality: item size, item property count, data consistency, amount of indexed properties, document indexing, query patterns, and script usage. They are also variable by the type of operational action and type of API that is being used within the Cosmos DB account.

2. Project Organization

2.1 Roles and Responsibilities (Subject to Change)

Name	Role	Responsibility	E-Mail
Joe Do	Project Manager	Administrator of Tool Accounts	joedo0209@gmail.com
Ferdinand Tembo	Tester / Documenter	Ensures code meets expectations and performance goals. Adds and edits documents and notes.	ferdinand.tembo @bellevuecollege.edu
Kevin Tran	Back-end Integration Developer / Documenter	Codes solution to meet expectations and performance goals. Adds and edits documents and notes.	kevin.tran @bellevuecollege.edu

2.2 Tools and Techniques

Tool Name	Use
Azure - Cosmos DB	To implement for testing purposes and possible integration
Google Chrome	For UI testing purposes
Google Drive	Store various files and preliminary code (code may move to GitHub, see GitHub)
Google Docs	To share notes and documents that are pertinent to assignment

Microsoft Edge	For UI testing purposes
GitHub	(In case allowed by other party) To store code
Slack	Communication portal between all members
Trello	Agile Board for Development Process
Visual Studio 2017	To code, test and run solution using C# with the .NET framework and Azure Cosmos DB SDK

3. Project Management Plan

3.1 Tasks

1. Establish concise knowledge of problem via meeting
2. Re-edit various documents; update notes
3. Reassign roles and responsibilities; update and assign tasks
    a. Complete a general comparison research that is two-parted. First, test various test cases using the Microsoft provided RU estimator and then do the exact same tests using a self-made client. After that is done, we should be able to narrow the problem if either only the website has an inaccurate engine or if the CosmosDB estimation engine (that the website might have implemented or copied from) is wrong in general.
        i. Assign and generate various JSON documents and create complex test cases and various amounts of possible CRUD operations/hour.
        ii. Start testing these over on the DocumentDB website
        iii. Create own custom client in testing and retrieving RU values
        iv. Compare and contrast results in step ii and step iii.
4. Assign tasks on Trello and start Agile Development of solution
5. Submit all materials and meet with all involved parties to repeat above tasks or to conclude project

3.2 Assignments

1. Work on Software Design Document
2. Meet with customers and stakeholders to collect project requirements or any related issues
3. Group discussion on approaches to solution
4. Implement various APIs for JSON parsing on Cosmos DB
    a. Do initial testing to get throughput statistics and respective actual cost statistics

    b. Find best-fit model representation on costs for each data
        i. Implement machine learning?
5. Redesign code that deals with estimation
    a. Test code functionality
        i. Meets expected output
        ii. Meets or exceeds expected calculation time
6. Create API/Document for code, calculator, and any other related-implementation

3.3 Timetable

*** Please refer to the Projector Forecast spreadsheet

4. Additional Material

4.1 Definitions/Acronyms/Abbreviations

Word/Acronym/Abbreviation	Definition
Collection	Container of JSON documents and associated JS files.
Cosmos DB	Cosmos **D**ata**b**ase; a Microsoft Service
CRUD	Acronym for **C**reate, **R**ead, **U**pdate, **D**elete operations in a database.
Request Unit	A unit of currency for Cosmos DB. Each time operations are performed in a database container, the user will incur request units.

4.2 References

"Estimate Request Units and Data Storage." Estimate Request Units and Data Storage, Microsoft, 2018, www.documentdb.com/capacityplanner.

Gentz, Mimi, et al. "Request Units and Estimating Throughput - Azure Cosmos DB." Microsoft Docs, Microsoft, 25 June 2018, docs.microsoft.com/en-us/azure/cosmos-db/request-units.

Nevil, Tyson, et al. "MicrosoftDocs/Azure-Docs." GitHub, Microsoft, 3 Oct. 2018, github.com/MicrosoftDocs/azure-docs.

4.3 Appendices

Associated Project Links

Do, Joe, et al. "RU Calculator." RU Calculator, 27 Sept. 2018, bccscapstonecalc.wordpress.com/.