|  |
| --- |
| BCIT IOT |
| BCIT IOT Cloud Services Postmortem |
| BCIT ARLO & Telus |
|  |
| **Connor Vinchoff**  **Nathan Chung**  **Natalie Doan**  **Kevin Shi**  **Matthew Marty**  **Muhammad Bilal Zubair**  **Cody Chand** |
|  |

# 

# Introduction

One of the objectives of the BCIT NB-IOT project was to evaluate IOT platforms (AWS/Azure) for future use in research and education for BCIT programs. This involved assessing the respective IOT APIs, database performance and functionality, ease of SDK integration, and availability of documentation.

Overall, we believe AWS should be preferred for IOT programs going forth, primarily due to its mature SDK and superior documentation.

# Issues

## Azure

* Confusing and unintuitive documentation
  + Class documentation contains no complete instances of the class
  + Method functionality and returned value definitions are split across multiple pages
  + Reference ordered alphabetically, instead of by type (i.e. storage classes, errors, requests, etc)
  + There were times where the official Microsoft documentation was presented in such a way that it looked like a third party “tips and tricks” website
  + Deprecated features sometimes do not have links to the features that replaced them
* Many service configurations locked-in on creation
  + The functional app (Lambda function Azure-equivalent) we inherited was configured for a specific NodeJS environment
  + Azure functions within that app thus had to use that specific NodeJS environment; AWS allowed for runtime selection on a per-function basis
  + CosmosDB accounts were locked into a specific API and couldn’t be reconfigured
* Documentation and feature mismatch
  + While the CosmosDB mongo API documentation stated that index creation was supported, createIndex operations through mongo shell failed to properly create indexes

## AWS

* DynamoDB table keys locked-in on creation
  + This was significantly less problematic than CosmosDB’s key configuration; since DynamoDB allows for sort/partition key configuration per-table, it was simple to configure a new table properly and migrate the data via Lambda function

## Database Throughput

Azure database functionality was severely constrained by low (i.e. free-tier) provisioned throughput. This resulted in rate-limiting data requests from the webapp, which increased the latency between user actions being reflected in the charts to unacceptably long times. While a caching strategy alleviates a degree of the latency, we believe that students using Azure should work with significantly smaller amounts of data (i.e. fewer devices, or less frequent device readings) in the interests of cost and performance.

Azure has a system where if it detects a request being made that is too large, it will calculate the throughput and if the current throughput is not enough to run the request , it will not run it at all and give an error with a header informing you when you may try again. This can be useful to prevent yourself from making an excessively large and expensive query, but if you do want the query to be run it has been exceedingly difficult to get it to do so in a timely manner. There is a very recent preview feature that is said to simplify mongo rate errors, so it is possible future teams working on this project in Azure will not have this problem. Further testing is advised.

# Proposal

We believe a course similar to ACIT 2420 Linux Systems Administration with student groups using Raspberry Pis would be ideal for a potential IOT course. As BCIT is still assessing IOT devices, whatever sensor/shield configuration is eventually settled upon need only be GPIO-compatible to interface with the Raspberry Pi. Students would configure the complete devices to send data to the chosen IOT hub service, then pipe the data into the cloud provider’s analytics services.

The proposed IOT course may also work as a joint project between multiple already existing classes in the CIT and CST programs, done in small teams, where each class has a chance to work on the project throughout the term in their own discipline. A class focused on either hardware or programming would create a program that runs on the devices to collect data and transmit it to the cloud. A class that already deals with cloud services such as AWS and Azure could integrate a part of its time to collecting the sensor’s data and putting it into a database. Different groups could each use different engine/api’s and present their experience with them during a class. A programming or web development class could create a RESTful API that lets you query the data from the cloud, and a second application could display the data. This covers and combines many topics that are in other classes (hardware, software, cloud, application development) into a single project introducing students to IOT and giving them hands-on experience.