DATA SERVICES FOR INTERNET OF THINGS

An Undergraduate Research Scholars Thesis

by

DAVID LACROIX

Submitted to the Undergraduate Research Scholars program
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Research Advisor: Dr. Dilma Da Silva

May 2016

Major: Computer Science

TABLE OF CONTENTS

ABSTRACT

Data Services for Internet of Things

David LaCroix
Department of Computer Science and Engineering
Texas A&M University

Research Advisor: Dr. Dilma Da Silva
Department of Computer Science and Engineering

The factors involved in choosing between storing data repositories at locally-hosted infrastructures or at (remote) public clouds are well understood for many enterprise application domains. The proliferation of Internet-of-things (IoT) devices (including wearables) is now introducing a new class of applications, for which neither the research community nor the industry players offer guidelines on how to best handle the data. The goal of this research project is to characterize the most effective data architecture in terms of locally or remote hosted for a given IoT workload. Through this research, developers will become aware of various issues dealing with the designation of a host for a given data repository including security, efficiency, and accessibility concerns.

ACKNOWLEDGMENT

I want to thank Dr. Da Silva for all of the time and effort she has committed to me throughout this endeavor. I have truly enjoyed working with her from the beginning.

NOMENCLATURE

API	Application Programming Interface
AWS	Amazon Web Services
B	Byte
CAAPI	Common Access API
DB	Database
DNW	Digital Neighborhood Watch
EDA	Energy Data Analytics
GDP	Global Data Plane
HUD	Heads up Display
IoT	Internet of Things
MTU	Maximum Transmission Unit
OS	Operating System
SaaS	Software as a Service
VM	Virtual Machine

SECTION I

INTRODUCTION

Overview

Before our problem can be discussed in great detail, there are two primary concepts that must be well understood. These concepts are cloud computing and the internet of things (IoT). Both will be explored regarding the technology itself, the current state of the technology, and relevant concerns about their continued development. Following those subsections are two practical examples which combine the cloud and IoT; our research will be to further understand and optimize relations involved in situations such as those. Finally, we will discuss the specifics of our research topic.

Cloud Computing

The cloud (or cloud computing) refers to both the hardware and software components that comprise a given cloud service offering. This service can be a range of anything from storing data to program execution or some other computational utility provided by the organization. The "cloud" aspect reveals that the service is provided as part of a computing platform designed to offer the service with certain characteristics, most often the computing infrastructure's ability to grow and shrink the resources being allocated to the service to meet current demand. This service can be offered in a public or private model. A public cloud is a server that is available to anyone in a pay-as-you-go manner, and the service it provides is termed utility computing. A private cloud is one that is used only internally or at the discretion of a company or organization, and is

carried out on a computing platform owned by the organization. These two types of cloud services will be further investigated as it applies to storing sensor data generated by the IoT.

History of the Cloud

In the last eight years, the computing industry has seen cloud services rise from an interesting idea to a widely adopted approach to build systems where some (if not all) of the computing and data services are hosted on a public datacenter infrastructure, on a pay-as-you-go basis. This new model frees application owners from having to manage their own hardware infrastructure and achieves cost benefits as the cloud provider leverages economy of scale to achieve unprecedented efficiency in usage of computing resources. The computer science research community has dedicated a lot of attention to attacking many of the new problems introduced by the cloud paradigm [1], with a large body of results being published at top conferences (such as SOSP[1], OSDI[2], USENIX ATC[3]) and at new cloud conferences such as ACM SOCC[4] and IEEE IC2E[5].

Cloud Evolution

There are three new aspects of cloud computing that have recently revolutionized the field. First, there are essentially infinite computing resources available on demand which can handle unexpected load surges [1]. Second, users of the cloud no longer have to make an upfront commitment to the resources they will need which allows the user to expand or retract their

[1] ACM Symposium on Operating Systems Principles, http://sosp.org
[2] USENIX Symposium on Operating Systems Design and Implementation, www.usenix.org/conferences
[3] USENIX Annual Technical Conferences, www.usenix.org/conference/atc15
[4] ACM Symposium on Cloud Computing, sites.google.com/site/acm2015socc/
[5] IEEE International Conference on Cloud Engineering, conferences.computer.org/IC2E

services as needed [1]. Finally, resources can be utilized through an hourly or daily rate; therefore, users can release resources based on their current needs and conserve costs [1]. These benefits have been made possible through building datacenters in cost effective areas with respect to power, bandwidth, operations, software, and hardware availability.

Current Range of Cloud Services

Different levels of utility computing abstraction are offered to users which have their own advantages and disadvantages. For example at the lower end of abstraction, Amazon's EC2 instance allows users flexibility of choice in nearly every aspect of the environment, but this makes it nearly impossible to implement automatic scaling and failover due to the difficulty with replicating the given environment in the time scale needed[1]. At the other end of the spectrum is Google's AppEngine which enforces a specific structure used on traditional web applications; it requires the user to follow a specific format but offers efficient scaling and high-availability [1]. For our project we have chosen to use Amazon's cloud services due to the high flexibility of the service.

Cloud vs Local Storage

For most enterprise and scientific computing applications, the difficulty in whether to store the data locally or at the cloud is in determining several factors relating to the application-specific information, for example if the data is personally identifying, health related, confidential, or otherwise privileged. In conjunction with these facets, system developers will also look at how pervasively the data will be accessed.

Costs and Benefits of the Cloud

Cloud computing is becoming a much more prevalent technology, but it comes with its own set of concerns especially regarding security. While the cloud offers many benefits such as access from anywhere, seemingly infinite storage, no upfront commitment to specific capacities in terms of CPU, storage, or networking, and the ability to pay-as-you-go [1], it can also put data at a higher risk being that it is stored within the same hardware as all other applications. While this makes it a larger target for attackers, there are security measures that can be put into place. One such aspect is that the cloud provider is often better positioned to achieve higher levels of security in their datacenters than most customers are able to do on their own. They are able to achieve this through the ability to hire experts and invest in overall procedures and tools that may be too expensive for individual companies.

Obstacles to Continuing Cloud Growth

As discussed by Armbrust et al in [1], the top 10 most relevant obstacles and possible solutions to the future evolution of the cloud are as follows. First, the need for continuous availability can be solved by a user utilizing multiple cloud providers. Second, it can be difficult to transfer data between separate cloud providers (aka data lock-in) and can be solved by standardizing cloud APIs between providers. Third, clouds need to keep data secure (such as through encryption, VLANs, and firewalls) and have the ability to be audited. Fourth, there is a significant risk of data bottlenecks during transfers, this can actually be circumvented by shipping the data on a disk; using this method is more efficient in both time and cost usage. One such example of this service is offered by Amazon through a service called Snowball. Fifth, performance needs to be steady and predictable; to fix this the provider should increase support for Virtual Machines

(VMs) which are operating systems that run on a host, utilize flash memory which can be erased electronically and is both read only and reprogrammable, and gang schedule VMs to allow them to run concurrently. Sixth, storage needs to be scalable and is presently available. Seventh, providers will need to be able to easily debug issues in a large distributed system and need a debugger that interacts with the distributed VMs. Eighth, the service should be able to scale quickly which will require an auto-scaler. Ninth, a reputation fate sharing service should be offered to ensure that if a given component fails the entire system will stop. Lastly, providers should offer a pay-for-use software licenses. There has been a lot of progress in pursuing these ten challenges, and there is also disagreement on the extent of their importance; however, they continue to be a good representation of the challenges still facing cloud computing.

Internet of Things (IoT)

The Internet of Things is a concept of having people, animals, or objects monitored through various sensors (devices that record or measure a given property of their environment) and storing this data stream in repositories that can be analyzed to make decisions. These decisions can be realized through an actuator which is a device that can perform a given operation relative to a given input. IoT devices are now part of a new application domain, and the data in such applications exhibit different characteristics from traditional enterprise and scientific applications toward manufacturing plants and critical physical infrastructures like power grids, in terms of how the data is generated, consumed, and analyzed.

Connected Devices Forecast

Currently the number of connected devices which could be utilized in an IoT solution is growing very rapidly. According to Gartner [2], in 2016 the number of connected devices will grow to 6.4 billion, an increase of around 30 percent from 2015. On average this increase equates to about 5.5 million devices getting connected every day of 2016 [2]. With the addition of all of these devices comes a major opportunity for IoT products and solutions. According to McKinsey Global Institute [3], the IoT has the potential to make an economic impact of 3.9 trillion to 11.1 trillion dollars per year by 2025.

IoT Background

There are many challenges that need to be addressed to realize the full potential of the IoT vision. In this work, we focus on the data services required by an IoT service. The data repository can be realized through databases in locally-hosted storage mediums or accessed remotely in the cloud. IoT can be anything from the multitude of Fitbit watches to a network of cameras to a factory with thousands of sensors monitoring operations. This topic is becoming prevalent with all of the smart devices being developed such as wearables, smart phones, and the like. All of the devices are capable of collecting specific information and yielding useful feedback from the data. A couple of examples could be tracking the average number of steps taken in a given time period or using recognition software to monitor a babysitter and automatically notify the parent of any suspicious activity.

IoT Evolution

IoT is currently experiencing great growth with the explosion of smart devices that all connect to the internet and to each other while constantly sharing information. This change has brought up some significant issues in the field including privacy, security, scalability, latency, bandwidth, availability and durability control [4]. One suggested solution deals with the manipulation of this data in a much more efficient and secure manner; it focuses on the transport, replication, preservation, and integrity of streams of data while enabling transparent optimization for locality and quality of service [4].

Current State of IoT

The development of IoT applications is an emerging field and there are no guidelines on best practices on how to build such systems to achieve efficient usage of resources. Existing applications tend to adopt the cloud for data storage due to the simplicity of the approach, without knowing if such an approach will be able to meet application requirements as the field evolves and more sophisticated data manipulation is needed. The need to access remote databases, depending on how the data is generated and accessed, can introduce delays in processing that are unacceptable to users [4].

Handling IoT Concerns

Regarding privacy and security, the use of many sensors to constantly collect and store information creates an environment of high risk, and development should exercise extreme caution before proceeding [4]. Scalability is another issue that will need to be dealt with since predicted estimates range from 26 – 50 billion devices being connected to the cloud by 2020 and

this amount of data could overwhelm bandwidth requirements if it were not compressed in some way [4]. Since the cloud is actually on the edge of the network and not at the center as some believe, there may be extreme issues even with simple operations using the current network to handle the future load. With the advent of so many smart devices using IoT solutions, the upstream load will become saturated; this is due to the fact that the current system is designed to handle higher downstream loads than upstream. Being that these sensors will control physical actions within the real world, quality of service will become a major issue that cannot be ignored. Lastly, there is the issue of both reliably destroying temporary information and reliably storing long-term data in light of the fact that the cloud's hardware is out of the control of the user [4].

IoT Solutions

Currently the market has seen massive adoption of both cloud usage and the appearance of numerous smart devices, and many current IoT solutions involve connecting these devices to the cloud. These IoT solutions also tend to develop their own APIs and separate gateways to handle the data. Many IoT solutions fall into two categories, ambient data collection or real time applications with low latency requirements [4]. Ambient data collection may come from sensors placed in buildings, in cities, on humans, and so forth collecting continuous data that may also contain highly sensitive or confidential information. Regarding real time applications, they tend to be reactive to environment stimulus for human consumption, robotic processing, or some other monitor and requiring low latency updates.

The Quest for a Universal IoT Solution

A data-centric proposal has been made referred to as the Global Data Plane (GDP) which is focused on "the distribution, preservation, and protection of information" [4]. There are several significant ideas behind using this approach. Regarding security, it uses a combination of single writer logs and public/private keys to ensure that only the given sensor can record information to the log and that only authentic observers can access the data. To successfully utilize publish and subscription updates to an observer, the solution uses a multicast tree which has been shown to work with this kind of structure [4]. Lastly, it is advised to use a Common Access API (CAAPI) to normalize access to the structure. An alternative to a data-centric approach is to organize a system around devices. Such device-centric model would focus on individual device capabilities and how they communicate to each other, aiming at a common architecture to incorporate devices into an IoT solution. A discussion of existing efforts in data-centric versus device-centric solutions can be found at [5].

Bolt: A Data-Centric IoT Approach

Bolt [6] is a data management system designed to handle the emerging class of data sensing applications that target a home environment, but in addition it can be applied to various other domains such as factories, offices, and streets. Bolt has four primary goals; first, that time series data will be supported and that data can be assigned arbitrary tags. Second, the system should support sharing across multiple homes due to the demand of multiple applications needing such functionality. Third, Bolt should allow the flexibility for application specific storage locations being that each has its own specifications. Finally, the system should supply adequate security against potential breaches of security [6].

Bolt's Methodology

Bolt first classifies data by application then by time-tag-value records. It was designed to better utilize even untrusted cloud providers and unsecure networks in a more secure fashion; this is facilitated through encryption at the client before it is sent to the cloud. The transmission process works by first grouping records into chunks that are then compressed and encrypted. The retrieval process works in much the same way by retrieving multiple records grouped as a chunk at once; this is generally more efficient being that queries often involve several, consecutive records.

Application of Bolt

The system was tested using three applications. One application was PreHeat which uses occupational patterns to more efficiently heat the home. Another application is Digital Neighborhood Watch (DNW), a video sharing service; it can detect objects within the video and supports queries across the network of homes to track suspicious activity. The third application is Energy Data Analytics (EDA) which gathers specific energy consumption information within the home; this is accomplished through sensors directly measuring each appliance's usage directly from the outlet. Using all of the information that this system produces, a user can identify which appliances have the highest consumption and allow them to tailor their usage to compensate for unnecessary expenditure.

Bolt Security

Security is maintained through several factors. To ensure confidentiality through the network and cloud, the data is encrypted before being sent out and is decrypted only once it is retrieved. Write

permission is append-only, and the information is encrypted using a private key; a hash of the data is also used to ensure that the chunk has not been modified enroute. Read permissions are granted and when revoked will disallow access to any new data generated.

GigaSight

GigaSight [7] is a system designed to collect continuously streaming video from the use of products such as Google Glass which is a product composed of a glasses band, camera, and HUD that is connected to the internet. It will represent a massive crowd sourced collection of digital video from the world. GigaSight aims to be scalable due to the mass volume that is anticipated in the upcoming future by decentralizing the collection servers. This decentralization is achieved through cloudlets which are areas of the cloud dedicated solely to processing a given user's data that will run virtual machines (VMs) on one or more cores based on demand and availability [7]. The project also anticipates the need for automated editing (or denaturing) to ensure privacy to the user. Once stored this video can be queried regarding its content and data tags.

Profitability of GigaSight

As an incentive model, users will be offered compensation for sharing their video streams. This information is highly valuable especially to marketers. Each individual stream can help marketers determine preferences and help them to tailor their ads to better influence their customer base.

Privacy of GigaSight

Denaturing is an important topic due to high confidentiality prevalent with this technology. Hiding all information generated by the stream would insure complete privacy but would have no informational value. Whereas showing all information insures massive value but offers no privacy. There is an important middle ground that the system will target in order to balance these two demands. It is also important that the denaturing process be automated due to the unrealistic task of having the user edit a constant video stream. This automation process will need to be effective enough to get the user base to trust the system to eliminate their sensitive data. As a default, the user can specify if they want their stream to be obscure by default or not altered, but the system can also edit out faces, objects, and scenes. It would also be possible to use a marker such as a QR code to alert the system that recording is unwelcome. While this project recognizes that it cannot fully cover all of the implications and demands of a viable denaturing algorithm it chooses to primarily explore the architectural and performance demands of the process.

Logistics of Uploading Data

Being that Heads up Displays (HUDs) are the proposed target for recording the video, the project has utilized a connection between smart phones and the HUD. This is done to limit the size of the device and its power consumption. The smart phone is used as a proxy to buffer the video until the phone can establish a Wi-Fi link and begin sending the data to a cloudlet. Although it would be preferable to denature the video on the user's device before sending it onto the network, the battery consumption would prove to be extremely cumbersome and is not a realistic solution in relation to current battery technology; therefore, the data will be denatured on the VM assigned to the user once the data reaches the cloudlet.

Comparing IoT Examples

To better understand the three IoT examples (GDP, Bolt, and GigaSight), we will summarize their differences. Regarding the generic IoT solution, Global Data Plane, it is designed to be a framework for establishing a generic, data-centric model for any given IoT solution. GDP describes a method to ensure greater security through single writer logs and public key cryptography, and it suggests using a CAAPI to universalize access methods for the system. Also, another option for a generic solution is the use of a device-centric model as opposed to a data-centric model.

Bolt

The Bolt project is designed to be a manager of IoT solutions within a home through an integrated interface; however, it can also be adapted for various environments other than a home. Bolt is designed to be flexible relative to the individual solutions it manages, so it allows each solution to specify the specific location of its database and, where applicable, allows sharing across multiple homes. Lastly, to better secure the data, Bolt encrypts the records locally before sending them to the given database.

GigaSight

As opposed to the generic solution of GDP or the IoT solution manager of Bolt, GigaSight is a specific example of an IoT solution. It proposes a method for managing and analyzing data generated for wearable camera devices such as Google Glass. Being that data generated from such a device is highly personal, GigaSight proposes that these streams of data be automatically denatured; this process can take in several factors to determine what should be obscured (faces,

license plates, QR codes requesting anonymity) along with the user's preferences toward more or less privacy. GigaSight also utilizes a user's smartphone as a buffer for storing and uploading data; the denaturing process would take place on a cloudlet specific to that user due to undue battery and computational demands that would be placed on the user's smartphone.

Storage and Access Concerns

The three examples mentioned are affected by a lack of comparison data between possible storage locations; specifically, the solutions do not analyze the various factors involved with selecting the location of their server (local or public cloud). Even though Bolt allows each application to select its database, it must still facilitate things such as sharing between homes which will require integration through some type of sever whether local or public cloud, and neither GDP nor GigaSight address this central issue. This lack of knowledge could result in various unknown inefficiencies such as increased latency, cost, security risks, etc. Through this research, solutions such as these can make a more informed decision on the storage and access of their data.

Our Research

Our research focuses on the location that should be used to collect, store, and analyze IoT data. We intend to test various artificial data generators that will approximate real world scenarios in order to better explore the problem space. To simulate the three scenarios (wearable device, video camera, and factory sensor network), we will vary the amount of data generated by a given "sensor" and vary the number of concurrent operations to reflect both sensor type and number of sensors respectively. We will separately test using a public cloud in the form of Amazon's EC2

service and a local machine to better define their characteristics. We will consider variables such as cost, latency, security, and pervasive access in our analysis.

Local vs Cloud

For cost, we consider the expenses of buying a device versus the pay-as-you-go service while also considering the ability to conserve expenditures with the public cloud, since it can be dialed up or down to meet demand. Considering only cost, the benefits of the cloud to more precisely meet demand nearly always out-weight the costs of maintaining a local machine. Regarding latency, we time both insertion and query events while also analyzing concurrent operations taking into consideration that larger machines may be able to compute operations at a faster rate and handle more requests at once. Both security and pervasive access depend more strictly on non-testable factors. Security concerns will always be lower on a local machine being that the user has total control of the hardware with respect to other users and physical access. When utilizing a public cloud the user's data will be placed in proximity to other applications, and there is also the issue of persistent storage where the provider may not adequately delete or dispose of faulty drives.

Data Scenarios

To best cover various types of data generation we look at three primary scenarios. First, we consider the wearable category such as a Fitbit monitor; this monitor has a low number of sensors and small, intermittent data output. Second, we consider a video stream from a camera; this represents a single sensor but has large, continuous data output. Finally, we consider a factory with hundreds or thousands of sensors and actuators; here we have a large number of

sensors with continuous or intermittent data output and actuators responding relatively to the sensor data. Each of these three scenarios helps us to test the impacts of a variable number of sensors along with a variable magnitude of data generation. While not placed into a distinct testing category, the range of tests simulate the various combinations of the specified characteristics from each sensor group.

Summary

Using artificial data generators we will simulate data similar to that generated in three IoT scenarios (wearables, video cameras, and a factory sensor network). Each case will be tested using two separate storage mechanisms (local and cloud) running InfluxDB for the database software. The test cases will primarily observe latency while varying the magnitude of data per operation and number of concurrent operations to help propose the best storage solution for a given IoT workload.

SECTION II

METHODS

Benchmarking Objectives

Our primary goals regarding the benchmark of our systems are to ensure that our results cover a broad range of data scenarios, provide statistically valid results, and provide fully transparent results to our audience. To help us achieve this last goal and so that we can avoid such pitfalls from the early phases in our work, we have investigated some common benchmarking fallacies and mistakes that Gernot Heiser [8] has stated.

Selective Benchmarking

This fallacy refers to selecting only a small subset of data scenarios to represent the full set, especially when it is done to further support the author's position. An example of this could have occurred in our research if we had chosen to test our various storage solutions using only a subset of our three data scenarios (wearables, cameras, factory sensors); a data subset, such as wearables only, would not have covered the full range of data throughput scenarios and could have mislead the reader into using an inappropriate storage solution. We feel that our three scenarios will help us clearly represent the full range of scenarios needed to test each of our solutions. Time constraints in our work may limit the extent to which data scenario is explored, and we will address such limitations by carefully linking the applicability of our conclusions to the workloads that we explored in sufficient detail.

Comparing Percentage Values

This fallacy can be exemplified through a comparison of changes in the percentage of overhead found in a system. For example, if the original overhead is 10% and with the new system the overhead is 15%, the change in overhead was not 5%. This is really an increase of 50% to the original. This type of fallacy can occur anytime one is comparing percentage values. It is a simple mistake that happens more often than one would expect. Our method for verifying our conclusions will inspect for such mistakes.

Dataset Choice (Calibration vs Validation)

Many times a given system model must be calibrated to specific operating conditions through the use of a calibration workload. The model is then tested using an evaluation workload to determine accuracy. In order to show the true accuracy of the model, the workloads must be different and completely disjoint. This must be done to discover the predictive power of the model rather than simply showing that the model fits the workload it was designed for. Our system model does not require workload calibration, as we will set a priori the workload scenarios that we experiment with. Our work investigates the behavior of IoT data services through synthetic workloads. It is out of our scope to calibrate or validate the workload generator.

Infrastructure

We utilize InfluxDB [9] as our database to store the entries generated by our “sensors” which are simulated by our data generator.

InfluxDB

InfluxDB [9] is a time series database that can perform various analytics over the stored data. It has no external dependencies, is written in the Go language, and supports HTTP(S) API insertion and queries. It was specifically designed to handle sensor data and real-time analytics. Each data point contains a measurement designation, a set of tags, and a set of fields. Each database can contain multiple measurements which in our case could be things such as “wearable”, “camera”, and “factory”. The tags are key-value pairs that are indexed for efficient queries. The fields are also key-value pairs and are designed to store the specific, relevant information for a given point. The database can also maintain redundant copies of the data on multiple servers known as clustering which as a future work could be tested as a hybrid system.

General Setup

The database will be stored relative to our two main environments on the cloud and locally. The test software is executed within the local system and communicates to either the local database or the cloud database, as specified. All of our testing software is coded in Java and is built and executed using Maven [10].

AWS Instance (The Cloud)

For the cloud, we are using an Ubuntu AWS instance. We have installed InfluxDB which will be the database holding all of our time series data points and where we can insert and query information. We communicate with the instance via TCP connection using a java plugin supplied by InfluxDB.

Local Ubuntu Server

We utilize an Ubuntu machine for our local tests. It uses Linux 3.19.0, has 8 cores (Intel i7-4790 3.6GHz), and has 16GB of RAM. The file system is ext4. We will use the same database software (InfluxDB) as on AWS. We run all of the "client" code from this machine which mean that this is where all of the insertions and queries originate from. In addition, all of the timing information was collected on this system.

Experiment Specifications

We will examine four experiments of behavior with the InfluxDB database (DB) on both the local and cloud locations (Insertion Size Latency, Query Size Latency, Insertion Rate Latency, and Query Rate Latency). For each experiment, we run a series of repetitive tests and average the results. We also record the maximum and minimum values of these tests to better display the variance. Latency is measured in milliseconds (ms) and data sizes are measured in bytes (B).

Insertion Size Latency

For this experiment, we will simulate a range of sensor data that needs to be inserted into the DB. We will compare the times of insertion with respect to the cloud and local environments. We run a series of four tests which each calculate the average latency of 1000 insertions for a given data size, and the results of the four tests is averaged to give a final value. We will also report the maximum and minimum of these four tests. The data size ranges from 1B to 4,194,304B and increases by powers of two.

Query Size Latency

Here, we closely mirror the Insertion Size experiment except for querying instead of inserting. This simulates various possibilities for the magnitude of data being requested at a given time. Using the same range of data sizes, we compare the times for querying the local and cloud DB's. We also run a series of four tests which calculate the average latency of 1000 queries, and the final value is the resulting average of the four tests. The maximum and minimum tests are reported as well.

Insertion Rate Latency

Now that the operations of varied data sizes have been tested to simulate a range of sensor types, we will test the concurrent operations needed by a network of sensors. We will simulate various numbers of sensors reporting to the database through concurrent operations occurring on a number of threads. In this case, we can view a thread as a simulated sensor. Here we chose a median value for the data size (1,024B), and each thread will perform 20 insertions during a test. The number of threads we test range from 1 to 128 and increase by powers of two. As with before, the cloud and local times will be compared and both the maximum and minimum values of each test are recorded.

Query Rate Latency

This experiment is very similar to the Insertion Rate experiment. It simply uses the query operation instead of insertion. This simulates multiple users or applications querying the database simultaneously. It uses the same range of threads, the same number of queries per

thread, and the same data size. The cloud and local performance times will be compared and the maximum and minimum test values will also be reported.

SECTION III

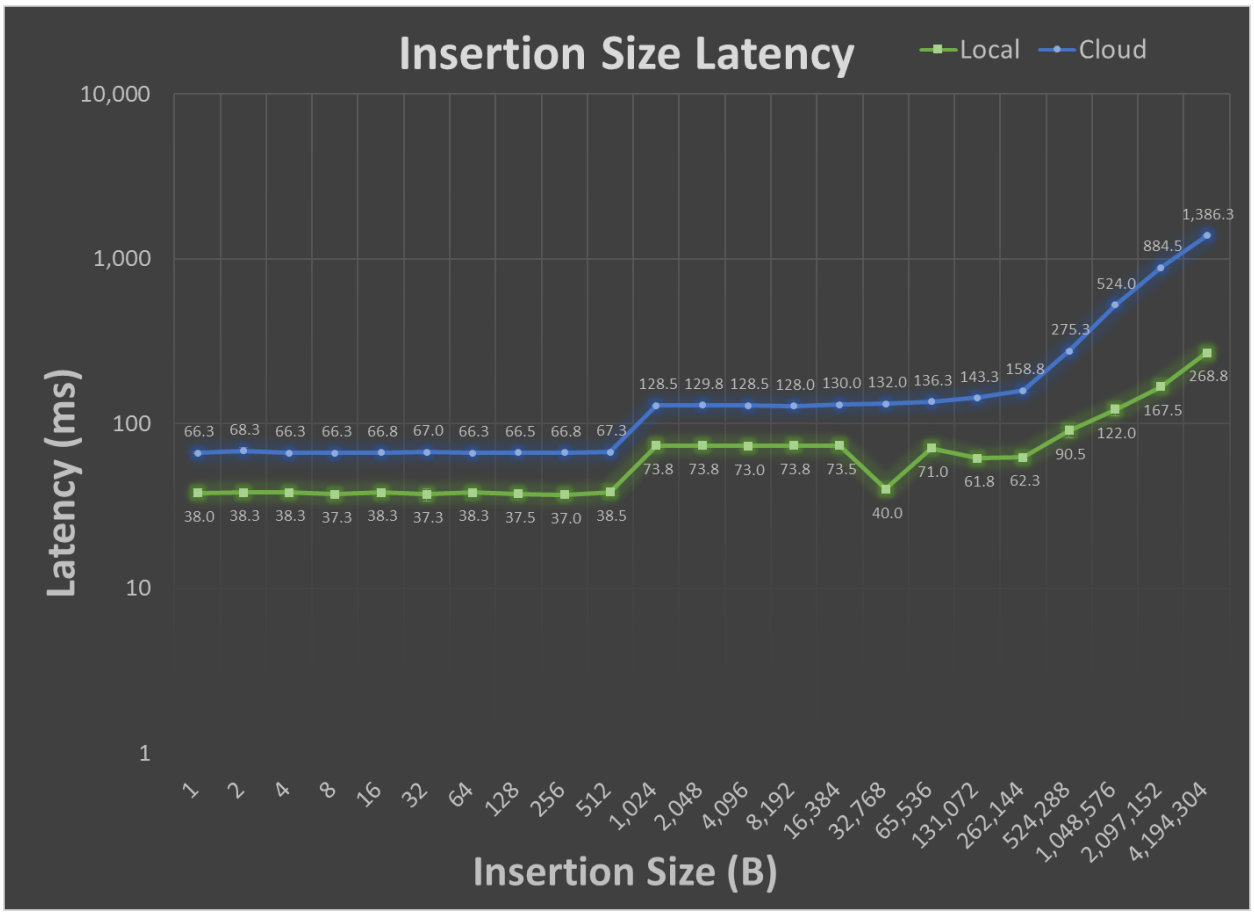
RESULTS

Overview

We will now report on the calculated values of the four experiments previously defined. Each set of data is represented using a graph of latency with regard to either the data size or the number of threads. On each graph the cloud is represented through a blue line with circles for markers and the local machine is represented through a green line with squares for markers. While not always evident on every graph, there are bars indicating the maximum and minimum values of the given tests for a specific data point; the closer these bars are to the data point, the smaller the variance of the tests. We will first present the collected data alongside a very high level depiction of the information. Then, we will express our analysis of the results.

Latency for Insertion Size

For this set of experiments, we tested the response time of inserting various amounts of data to the cloud and local DB. Below is the graph of our results.



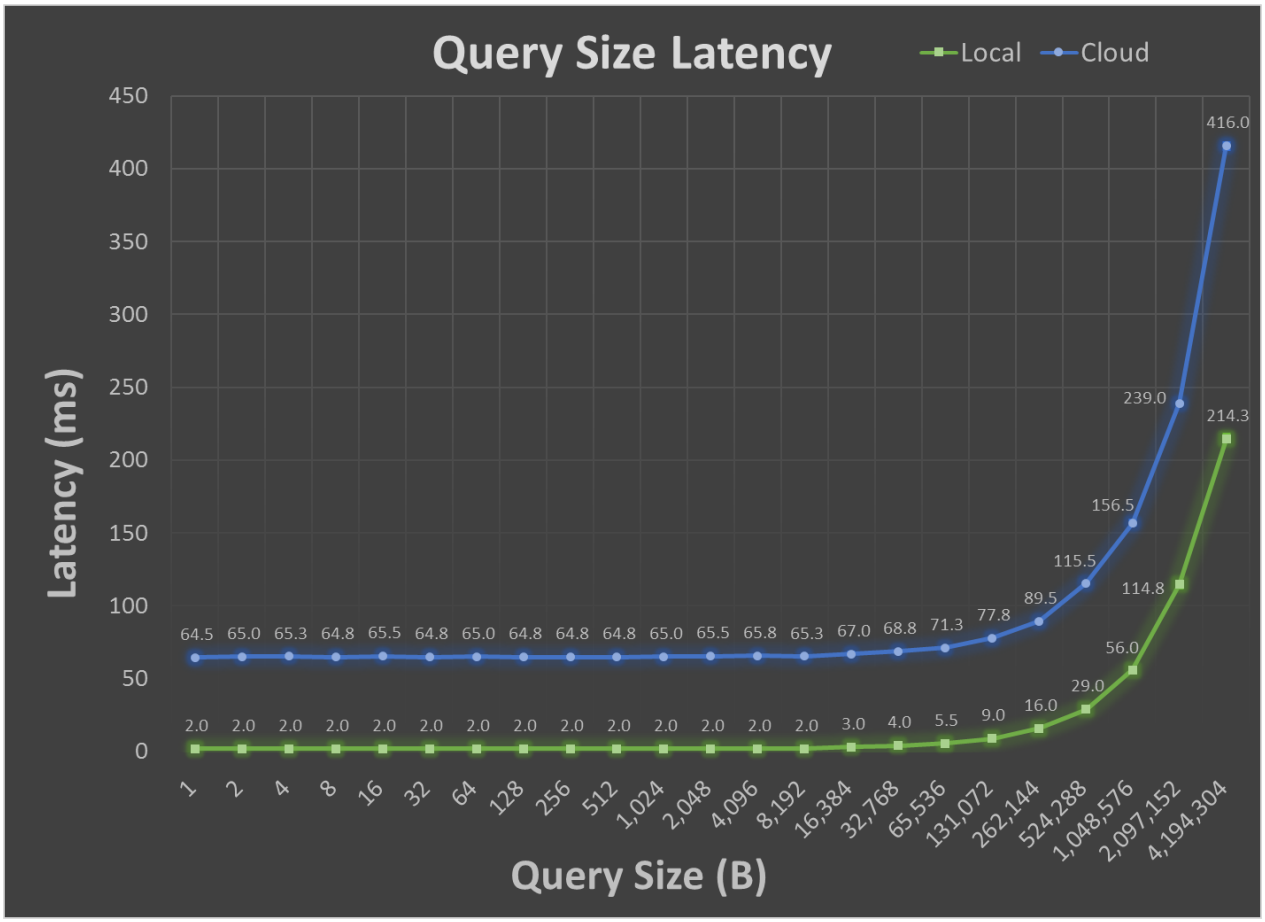
Observations

We noticed that for values under 1,024B, the latency remains relatively constant, ~38ms for the local DB and ~66ms for the cloud DB. After this, it raises to another fairly constant level for both local (~73ms) and the cloud (~130ms). There was a drop of the local latency at 32,768B when the value decreased to 40ms; this was tested and retested multiple times along with the values on both sides of the irregularity. While we ran short on time, further investigation could

be pursued by collecting additional information about the state of the database host and the network. After 262,144B, the latency begins to increase regularly.

Latency for Query Size

For this set of experiments, we tested the response time of querying various sizes of data from the cloud and local DB. Below is the graph of our results.



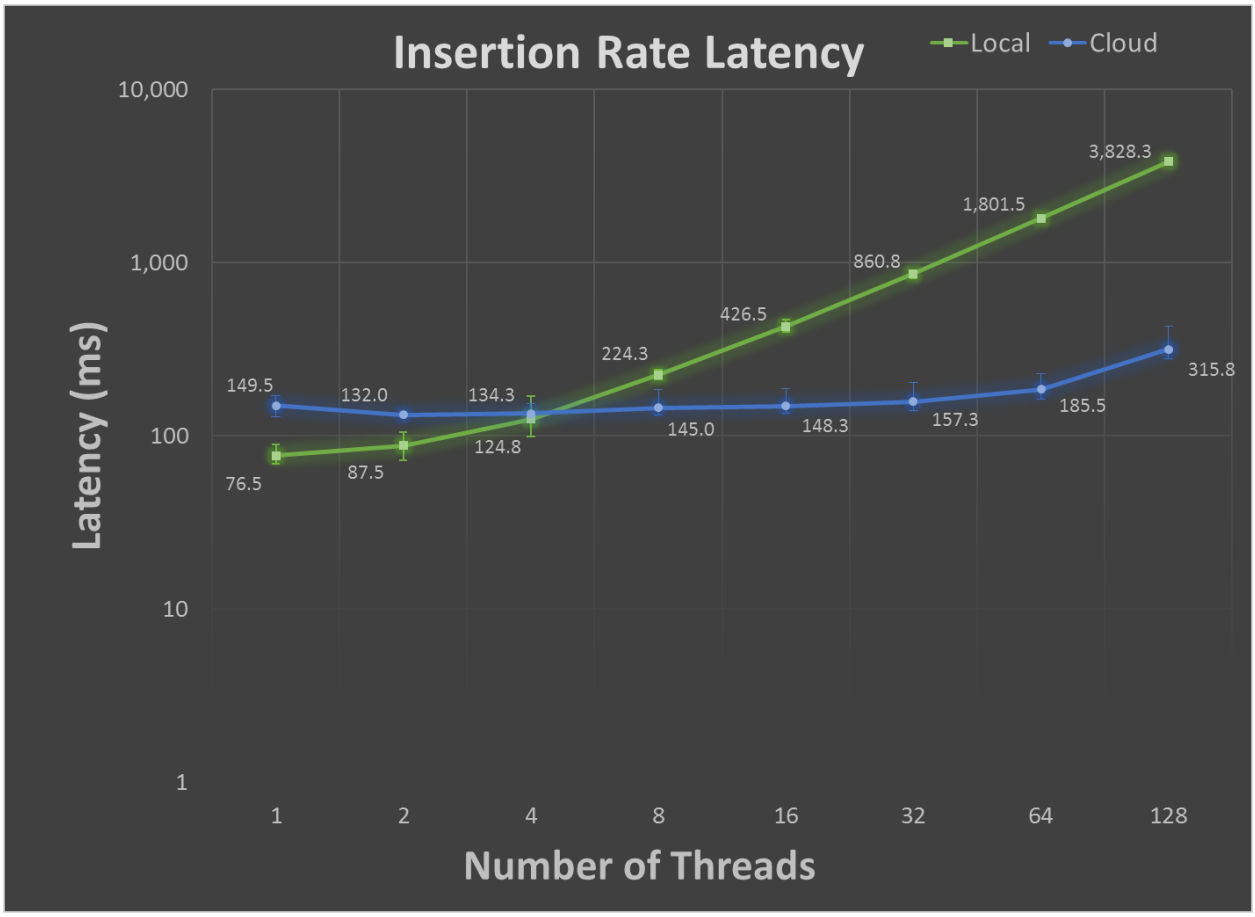
Observations

Both the local (2ms) and cloud (65ms) latencies appeared to remain constant up to 8,192B, and they begin to rise after this point. We recognize the oddity for the lower byte ranges to remain constant even while increasing the amount of data being sent. This is due to the data sizes being

small enough to fit in a single TCP MTU (maximum transmission unit). For the IPv4 network protocol, it is common to have MTUs of 64 or 128 KBs, therefore indicating that all the payload in the experiment can be sent in a single packet. Given these results, future work should use different data size values for evaluating query latency.

Latency for Rate of Insertion

For this set of experiments, we tested the response time of inserting a constant amount of data into the cloud and local DB while varying the number of concurrent insertions through the use of threading (each performing 20 insertions). Below is the graph of our results.

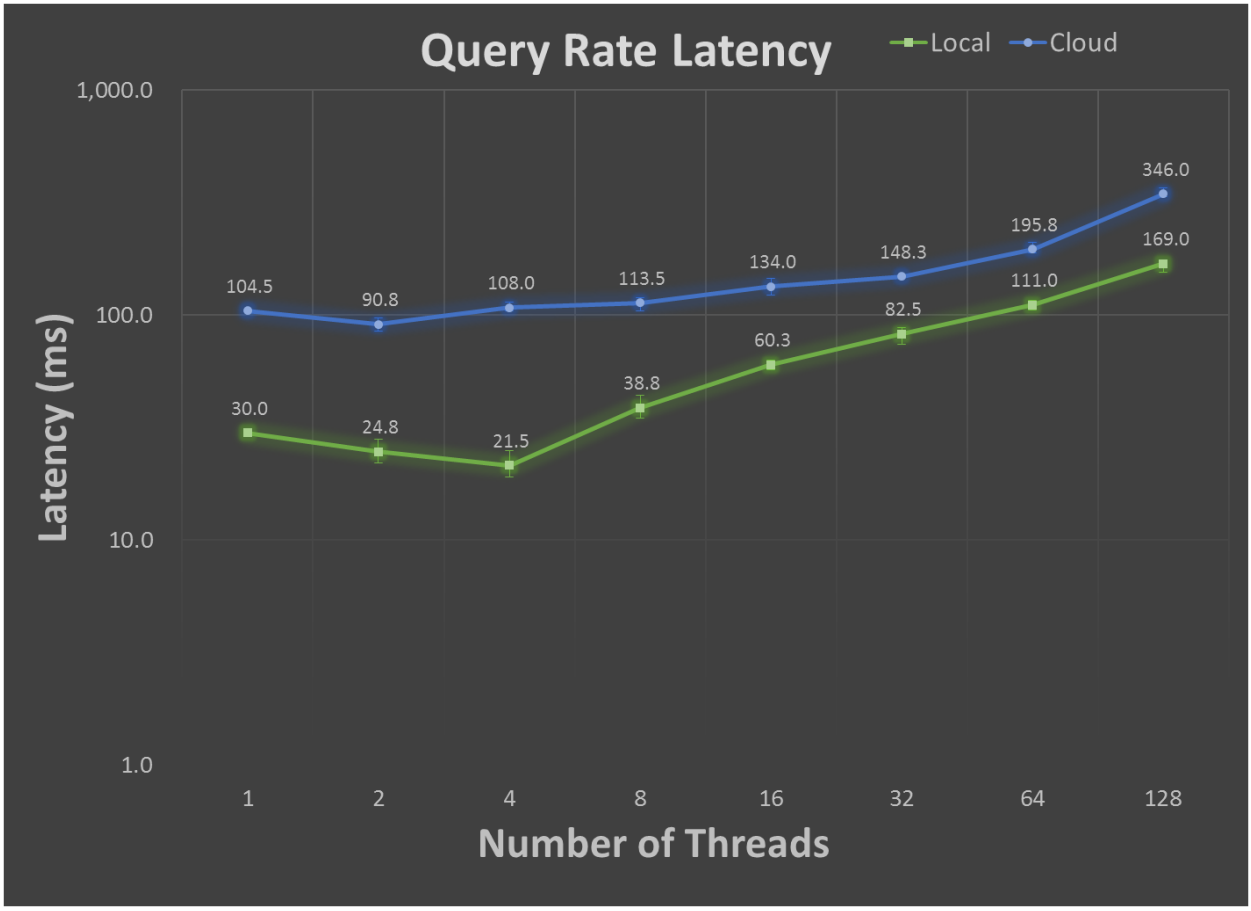


Observations

The latency of inserting into the cloud remains relatively constant while increasing the number of concurrent operations, and the latency for the local DB increases regularly with the addition of threads.

Latency for Rate of Queries

For this set of experiments, we tested the response time of querying a constant amount of data from the cloud and local DB while varying the number of concurrent queries through the use of threading. Below is the graph of our results.



Observations

Here both the cloud and local DB increase at a regular rate, and the cloud remains at a higher latency for all values (approximately double the value than local). The local also exhibits a "knee" where the values first descend as the number of threads increase from 1 to 4, and then it begins to rise. We believe that the improvement between 1 and 4 threads for the local scenario is a result of employing a 4-socket machine; additional investigation would be necessary to demonstrate this is the case.

Analysis

For the majority of the experiments, latency heavily favors a local machine as one would predict; however, regarding the case of concurrent insertion operations the cloud may favor larger networks of sensors. We must also remember that this is only one characteristic which must be taken into consideration when selecting a storage medium; cost, security, and pervasive access must also be analyzed.

Latency for Insertion Size

We see that the performance of both the cloud and local DB closely mirror the same behavior. For sizes less than 65,536B the cloud took roughly double the amount of time to insert locally. For the cloud, sizes larger than this seem to increase in latency much faster than the local machine. For applications that require low latency operations, such as real time system, a local machine may be preferable. All of the tests for this experiment were very similar to the reported average and is the reason that the max/min bars are extremely close to their data points. One possible reason we observe a constant latency for data sizes 1B to 512B is that the database

stores records in generic sizes which may be equal to or larger than 512B. The same may be true for the next level of relatively constant data size between 1024B and 262,144B.

Latency for Query Size

Here we also see a constant latency for a wide range of data sizes (1B – 8,192B). While it seems odd that it is such a wide range of values which remain constant, as mentioned before, the TCP/IP MTU size imposes similar transmission time for data sizes smaller than the MTU. Also for this graph, the max and min bars are nearly non-existent due to the low variability of the results.

Latency for Rate of Insertion

Unlike the other experiments, these results favor the cloud for larger numbers of concurrent operations. This is likely due to the fact that our local machine was unable to handle the magnitude of requests being made at once. Something that should also be noted is the fact that our test software was running locally and may have inhibited the performance of the local results. The experimental method should be improved to add one more machine to the setup, dedicated exclusively to run the client code (i.e., the threads issuing insertion and query requests). New experiments varying the timing of the concurrent requests could also reveal additional factors impacting the results. Even while considering this, the cloud in general may be a more apt environment to handle increased requests and faster internal processing where a less powerful local machine is offered.

Latency for Rate of Queries

Here the local machine appears to operate more efficiently than the cloud. Both results increase at a fairly steady rate which mirror each other. Other than the “knee” mentioned earlier, the graph seems fairly ordinary.

SECTION IV

CONCLUSION

Background

Our research centers on a new class of applications for the Internet of Things which do not yet have the information they need; to fully analyze the differences for data storage on the cloud versus a local machine, we attempt to explore the relevant facts. Through analysis of both general factors around this topic and the direct experimentation of latencies, we hope to better inform our audience of these factors so that they can make a more informed decision.

Experiments

We chose to evaluate two characteristics relevant to these applications which affect their latencies. Specifically, this involves the analysis of various data sizes which correlate to the range of a given sensor's configuration and the analysis of various concurrent operations which correlate to the number of sensors sending data or the number of requests for data being made. These two characteristic were recorded with respect to both the insertion and query commands.

Results

In general our tests favor the use of a local machine with respect to reducing latency as one would expect; however, the cloud may actually offer lower latencies for a large number of concurrent operations. We must also remember that this is only one facet that should be analyzed when selecting the appropriate environment. Other factors include cost, security, and pervasive access. Cost nearly always favors the pay-as-you-go service of the cloud due to much greater

efficiencies. Security generally favors a local machine as the physical security of a machine can be monitored, but as far as digital security is concerned, the cloud is likely more secure due to the ability for the organization to provision an entire staff to manage network security and the expertise that comes with this. For pervasive access, the cloud may be preferred due to its innate proficiency in this area, and local machines requiring more effort to create this ability also contributes to this preference. Depending on all of these factors, one can make a more informed decision depending on the requirements of a given application.

Closing Remarks

IoT is a very interesting field that will continue to develop and evolve. The experimental part of this work represents an initial step in analyzing how the choice of location for storing sensor data can impact the behavior of the system. Through this document we hope to have better informed those that will aid in this evolution and aid the effort to make beneficial design decisions.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," *Commun. ACM*, 2010.

[2] Rob van der Meulen, "Gartner Says 6.4 Billion Connected 'Things' Will Be in Use in 2016, Up 30 Percent From 2015." Stamford, Conn., 2015.

[3] D. Manyika, James; Chui, Michael; Bisson, Peter; Woetzel, Jonathan; Dobbs, Richard; Bughin, Jacques; Aharon, "The Internet of Things: Mapping the Value Beyond the Hype," 2015.

[4] B. Zhang, N. Mor, J. Kolb, D. S. Chan, N. Goyal, K. Lutz, E. Allman, J. Wawrzynek, E. Lee, and J. Kubiatowicz, "The Cloud is Not Enough: Saving IoT from the Cloud," *7th USENIX Work. Hot Top. Cloud Comput. (HotCloud 15)*, 2015.

[5] M. Fazio and A. Puliafito, "Cloud4sens: a cloud-based architecture for sensor controlling and monitoring," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 41–47.

[6] T. Gupta, R. P. Signh, A. Phanishayee, J. Jung, and R. Mahajan, "Bolt: Data Management for Connected Homes," *11th USENIX Symp. Networked Syst. Des. Implement. (NSDI 14)*, pp. 243–256, 2014.

[7] Z. Chen, K. Ha, P. Pillai, M. Satyanarayanan, P. Simoens, and Y. Xiao, "Scalable Crowd-Sourcing of Video from Mobile Devices," 2013.

[8] G. Heiser, "Systems Benchmarking Crimes," 2015. [Online]. Available: https://www.cse.unsw.edu.au/~gernot/benchmarking-crimes.html. [Accessed: 01-Jan-2016].

[9] "InfluxDB." [Online]. Available: influxdata.com. [Accessed: 01-Jan-2016].

[10] "Maven." [Online]. Available: maven.apache.org. [Accessed: 10-Apr-2016].