5-18-2013

THE SUSTAINABILITY OF CLOUD STORAGE

Ryan Thompson
Georgia Southern University, ryan_s_thompson@georgiasouthern.edu

T.C. Friel
Georgia Southern University, thomas_c_friel@georgiasouthern.edu

THE SUSTAINABILITY OF CLOUD STORAGE

Ryan Thompson
Georgia Southern University
ryan_s_thompson@georgiasouthern.edu

T.C. Friel
Georgia Southern University
thomas_c_friel@georgiasouthern.edu

ABSTRACT

This paper will attempt to determine the sustainability of cloud storage; that is, systems where the database lies in a separate vendor network other than the application itself. In this paper, we will discuss the reliability, usability, cost and overall ease of use of current vendor solutions while also describing the typical setup of cloud storage. We will also look at the evolution of cloud storage moving into the future, and attempt to determine whether integrations into cloud storage can be relied upon to transport vital information.

Keywords

Cloud computing, databases, Firebase, Amazon Simple DB, Amazon Dynamo DB

INTRODUCTION

Current commercial systems generally aim to have applications and database a stone's throw from each other. Part of the reason why the two systems are usually kept within the same network map is for fear that the reliability of the system will be compromised. The less distance information has to travel, the less likely that information will be lost or interrupted. The idea of cloud computing is nothing new. The terminology has been seized in modern day by major marketing firms, but the basic concept of remote systems has been around almost since the birth of modern communication systems.

In the 1970s the phone company used terminal computers to configure telephone lines. These systems, though not widely utilized or talked about at the time, allowed for engineers to connect into their system and test lines that were being worked on.

With the birth of the web application, most designers and developers identify their application as being remote or existing "on the cloud". Still, the idea of separating an application too far from the database it utilizes does not sit well with most professionals. Cloud storage, for the purpose of this paper, is a database that runs in a separate cluster other than the application that utilizes it. There are two common deployment models; databases on the cloud independently (using a Virtual machine image) or databases used by a service, typically provided by a cloud storage provider. Of the databases available on the cloud, some are SQL based and some use a No-SQL data model.

For the purpose of this paper, we used some of the more mainstream commercial products available; Firebase, Amazon Simple DB, Amazon Dynamo DB, and a local copy of Microsoft SQL 2008 R2 for control purposes.

With the onset of cloud services, there has been increasing demand to accept new technologies as they become available, because it is "widely accepted that a cloud data processing system should provide a high degree of elasticity, scalability and fault tolerance."[11]

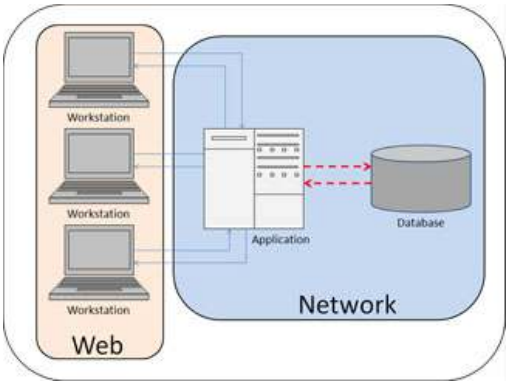


Figure 1. An example of a standard database setup

RELIABILITY

One of the major tenants in computing has always been reliability and sustainability. Network administrators fear server down time while programmers fear untested changes. Programmers create services for clients who rarely understand the intricacies of the product that they use. With a general understanding, most people understand that computer systems sometimes stop working.

However, as we saw with the April 2011 Amazon AWS outage, reliability is often on the minds of the end user. The Amazon outage lasted for four days and affected hundreds of notable sites such as Reddit and The New York times. [10] If a cloud storage vendor does go down, customers and developers alike are usually at the mercy of the support team and the urgency that they assign the outage. Such is why the reliability of your database may come down to the credibility of your vendor. "Outages are the most notable, but bumpy product upgrades, faulty product configurations, and weak integrations can all create similar issues." [10] Disaster recovery is another misnomer that's not always addressed in the fine print of a service agreement. [9]

CLOUD VS. STANDARD

Reliability refers to the consistency of a measure and though difficult to calculate correctly, can be estimated by asking, "What could go wrong?" To understand what could go wrong with cloud storage, we must first understand how it communicates.

A standard database communicates with its application through a centralized server. The user, represented by an integrating application, talks to the server and establishes a connection. Database connections are finite, and though often handled by the framework of the application, can be resource expensive. [7] The creation of a connection is often times disproportionately more resource heavy than the act of sending data. [8] See Figure 1.

There are multiple points of failure when connecting to a database. Inconsistent connection failures are typically due to communication issues with the network that houses the hardware. Localization of this hardware lowers the chance of these communication issues.

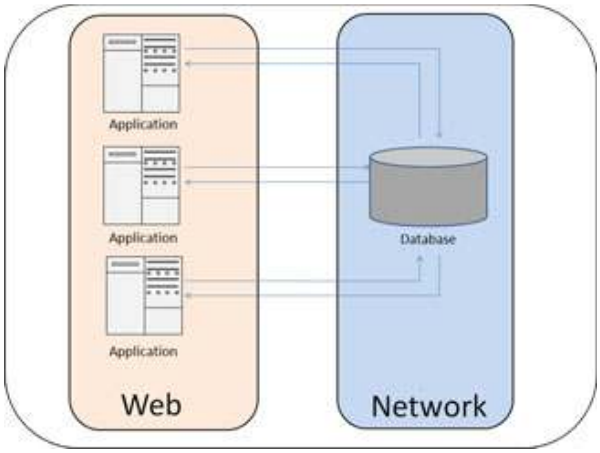


Figure 2. A standard cloud database setup.

Firebase is set up like Figure 2. In this schema, the application is located on the workstation and each of the applications is connected to a database. This introduces multiple points of failure instead of one. This type of setup is undesirable for vital information, such as a financial transaction. A financial transaction relies on encryption methods to distort its sensitivity, but must be transported through a vital system. [1] Failed financial transactions often result is incorrect balances and angry customers. From a hierarchical classification perspective, the needs of users are defined by the specific industry with which he or she is associated.[3] However, cloud storage has its advantages; it lowers the cost of scalable storage for small startups, it provides immediate access to already functioning systems, and it offers easy scalability.[2]

COST

Whenever you want to start a small project and have others provide your storage, cost becomes a big factor. The nature of your project often dictates what option is most cost effective. Will the project have a heavy flow of transactions to the

database? Do you need to keep a constant connection? Analyzing these factors before deciding on a storage provider often cuts down the cost. For our experiment, Amazon Simple DB was by far, the most expensive.

USABILITY

Simple DB had the simplest implementation and a very intuitive SDK. The downside to it was the cost, and more so, the charge system that it used.

Dynamo DB was far more complex and the SDK, while clean, was extremely confusing. The explanation of terms was not relatable to standard database terminology, and there was no effort to draw parallels between the two. The charge system was extremely easy to understand, and the rates were reasonable.

Firebase was easy to work with but the SDK was not standardized. All examples were in JavaScript, and the topic of integration into any other framework was left un-approached. However, Firebase still claims to be in beta, so their incomplete documentation is understandable.

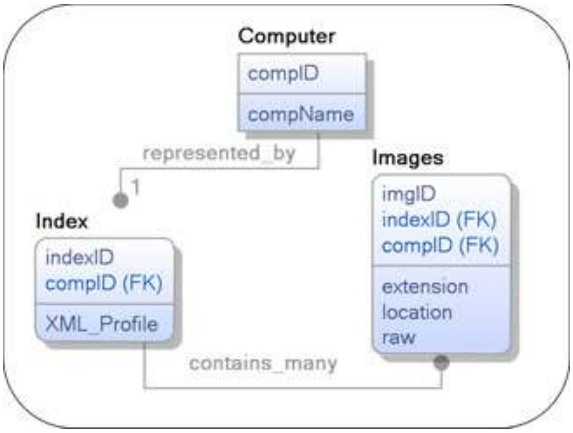


Figure 3. A IDEF1X data model of the experiment database

EXPERIMENT

To provide high quality data and to discern the reliability of cloud storage; an experiment was created to test the fail rate.

First, a traditional database was created using SQL Server 2008 R2 by generating tables and fields under the guise of a relational database that would be used for backups and data restoration (See Figure 3). Information was pulled in high volume from the host computer by way of searching the computer, indexing all of the JPGs on the hard disk, and sending them to the database. The images were converted into base64 and the transactions limited to a payload of 1000 bytes, often generating several hundred parts for each picture sent.

The database was then stress tested by running multiple threads of the application. The data included average read/write times, as well as overall transaction time. The data obtained from the stress test of the standard database acted as our control. The test was then run on a series of cloud storage solutions; Firebase, Amazon Simple DB, and Amazon Dynamo DB. Initially, the experiment was to include several other cloud storage vendors; Mongo, Heroku, and Google. It was realized early on that the fiscal cost of sending all of the JPGs on a standard Windows 7 machine to cloud storage far exceeded the intended budget for this experiment.

THE PROGRAM

The program written for this experiment was meant to stress the various databases so that a picture of their reliability could be drawn. The cloud storage portion of the program can be obtained at (https://github.com/tcwolf/imageTransport). The program operates as a windows console application feeding off a class library that contains the bulk of the code. The program initially identifies the computer by its Windows Machine Name and creates an entry into the database. The intention of this was more design than functionality, as this entry allows all transactions into the database to be identified as a collective. After sending an identifying transaction, it then searches the computer for files with 'jpg' in the filename and creates an XML index of these files. After this is completed, it stores the index in the database. It then loops through all of

the pictures and converts them into base64 strings. Each string is broken up into 1000 bytes and sent over. When completed, the program exits.

ERROR HANDLING AND IMPROVEMENTS

All exceptions that happen while sending over images are logged into a text file and are the main resource for collected data. In places that are appropriate, specific cloud storage exceptions are created also being written to the same text file. All communication errors that occur while sending transactions to any of the Amazon vendor databases fault under the same exception.

The program lacks the functionality to continue where it left off, but small modifications could be made to allow the program to initially search for the index, load it into the program, and then iterate through the collected files names to see which entries exist in the database. This would also cut down on the time spent on building the index.

AMAZON SIMPLE DB

Amazon simple DB is the first of the cloud database services that we analyzed for this paper. Of the three it had the smoothest API and the most support but with it the largest price tag. The benefit of simple DB is that it takes away the need to have an in-house solution and allows you to concentrate your energies on other aspects of your business. Amazon storage solution also adds automatic scalability for you application.[5] Simple DB is a non-relational data store that is, according to the website, " […] optimized to provide high availability and flexibility, with little or no administrative burden."[12] The SDK was cleanly written and the service worked very efficiently. They even offer a free tier for student and small applications.

One of the more troubling aspects was the price and the breakdown of the charges. In the first night of testing the application, the free tier usage quickly evaporated and the test account generated $25 in charges citing 181 hours of an increment referred to as the Amazon Simple DB Machine Hour.

"Amazon Simple DB measures the machine utilization of each request and charges based on the amount of machine capacity used to complete the particular request (SELECT, GET, PUT, etc.), normalized to the hourly capacity of a circa 2007 1.7 GHz Xeon processor." [12]

According to Amazon the 13279 transaction sent over used 181.934 machine hours (regardless, that this happened in less than an actual hour with a single thread of the application running). However, we only used 0.001 GB of storage and bandwidth. Our results mirror a similar experiment who" incurred charges of nearly $500 for our experiments, without actually providing any remote data access. Basically, all we did was create the instance and run our performance test queries on it."[4] This shows that simple DB is very expensive to operate.

AMAZON DYNAMO DB

Dynamo is proclaimed by Amazon CEO's Werner Vogels to be "the fastest growing new service in AWS history." [13] It is a No-SQL database service with a large amount of control for the user. The application is still in Beta but like Simple DB has a clean and easy to use SDK with plenty of sample code available. Dynamo was a little harder to use notably in comparison to Simple DB. The biggest difference came from the cost, running the same 13279 transaction using Dynamo never lifted us out of the free tier. The primary charge for Dynamo was storage and read/write units. The program did not come close to surpassing either of these elements.

FIRE BASE

Firebase is the last of the three cloud storage solutions used for this experiment. It is a web based database that uses a tree system to store data. Firebase is still in the early stages of development and is completely free, as of now, though getting an account may take some effort as not all people who sign up get accounts. Firebase sells itself as a "cloud service that automatically synchronizes data between clients and our cloud servers. It frees developers from worrying about how their data will be communicated and stored, and allows them to focus on their own application logic."[14] It seems to do a good job, though the down side is that it is primarily for web development and the SDK is geared towards JavaScript only. The program was able to communicate with the database in C# with a series of HTTP POSTS. This made it very difficult to use for the experiment. However, once the program was able to communicate, it seemed to have a faster rate than the other two solutions tested.

MICROSOFT SQL 2008 R2

Microsoft SQL 2008 R2 is the database that we used as our control. Microsoft SQL is an industry standard for databases and was implemented using a standard configuration. The application connected to the database over a local network. The database was created using the IDEF1X data model shown in Figure 3[15].

CONCLUSION

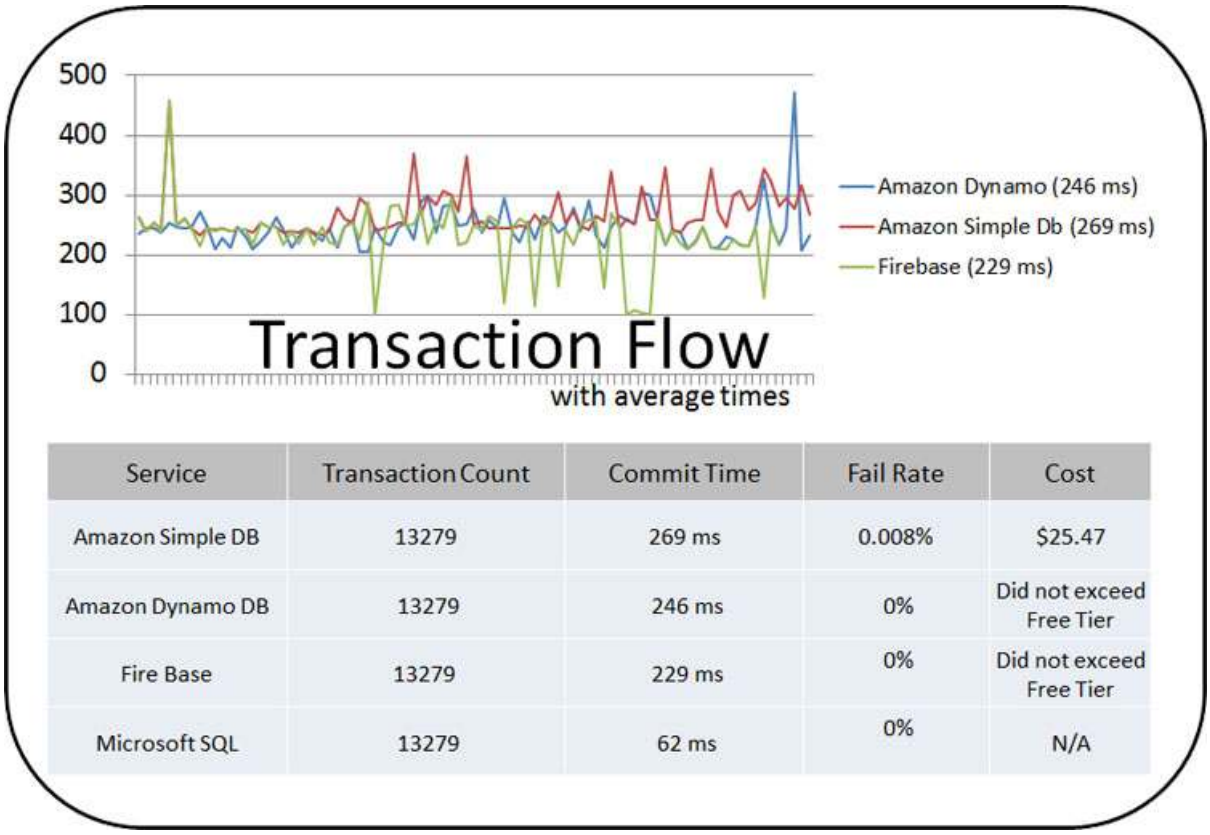
Fail rate is difficult to calculate, and it would not have been surprising to complete the test and received no errors. When looking at the options, a 0.008% fail rate may not seem too bad, but in an environment where hundreds of transactions are sent in an hour, the yield of error may become very visible. From the current measure of reliability, there seems to be little difference in the vendor options available.

This does not label cloud storage a disruptive technology, but rather a yet perfected technology. For the type of application it's designed for, it does exactly what it's supposed to and does offer options for developers who would otherwise have none.

Appendix A shows the transaction flow of each of the vendor solutions. From this data and ignoring the control, Firebase definitely seems the more reliable option for the avid developer. However, by way of the vendors that were tested, vendor cloud storage is only suitable for small scale applications that have low transaction volume.

This research will hopefully guide companies to caution when making the decision to use cloud storage. The number of vendors entering into online data storage seems to be growing daily, but little has been done to address the basic issues in the cloud storage model regarding data restoration. Data restoration services could be improved upon significantly if a system that focused on reliable transport and low fail rate was introduced.

In the future experiments we would like to raise the ceiling on our monetary limit and try to do a true stress test on the amazon web service databases. The application written for our experiment is a perfect way to stress test a database as it sends a large amount of data very quickly. Using Microsoft SQL as the bench mark, as we did below, it allows one to really see how much stress these services can take versus an industry standard like MS SQL or ORACLE.



Service	Transaction Count	Commit Time	Fail Rate	Cost
Amazon Simple DB	13279	269 ms	0.008%	$25.47
Amazon Dynamo DB	13279	246 ms	0%	Did not exceed Free Tier
Fire Base	13279	229 ms	0%	Did not exceed Free Tier
Microsoft SQL	13279	62 ms	0%	N/A

Appendix A. The graph of our transaction flow with average times.

REFERENCES

1. S. Bajaj and R. Sion. Trusted DB: a trusted hardware based database with privacy and data confidentiality. In SIGMOD, pages 205–216, 2011.
2. Edward P. Holden, Jai W. Kang, Geoffrey R. Anderson, and Dianne P. Bills. 2011. Databases in the cloud: a status report. In Proceedings of the 2011 conference on Information technology education (SIGITE '11). ACM, New York, NY, USA, 171-176. DOI=10.1145/2047594.2047642 http://doi.acm.org/10.1145/2047594.2047642.
3. Edward P. Holden, Jai W. Kang, Dianne P. Bills, and Mukhtar Ilyassov. 2009. Databases in the cloud: a work in progress. In Proceedings of the 10th ACM conference on SIG-information technology education (SIGITE '09). ACM, New York, NY, USA, 138-143. DOI=10.1145/1631728.1631765 http://doi.acm.org/10.1145/1631728.1631765.
4. [4] Ani Thakar and Alex Szalay. 2010. Migrating a (large) science database to the cloud. In Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC '10). ACM, New York, NY, USA, 430-434. DOI=10.1145/1851476.1851539 http://doi.acm.org/10.1145/1851476.1851539
5. Ramanathan, S.; Goel, S.; Alagumalai, S.; , "Comparison of Cloud database: Amazon's SimpleDB and Google's Bigtable," Recent Trends in Information Systems (ReTIS), 2011 International Conference on , vol., no., pp.165-168, 21-23 Dec. 2011 doi: 10.1109/ReTIS.2011.6146861 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6146861&isnumber=6146829
6. Christensen, C., 2003. The Innovator's Dilemma, Harper Business Essential Tractinsky, N. (1997) Aesthetics and apparent usability: Empirically assessing cultural and methodological issues, in Steve Pemberton (Ed.) *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI 97)*, March 22 – 27, Atlanta, GA, USA, ACM Press, 115-122.
7. Liang Zhao, Sherif Sakr, Liming Zhu, Xiwei Xu, and Anna Liu. 2012. An architecture framework for application-managed scaling of cloud-hosted relational databases. In Proceedings of the WICSA/ECSA 2012 Companion Volume (WICSA/ECSA '12). ACM, New York, NY, USA, 21-28. DOI=10.1145/2361999.2362004 http://doi.acm.org/10.1145/2361999.2362004
8. A. Ohori and K. Ueno. Making Standard ML a practical database programming language. In ICFP, pages 307–319, 2011. http://www.pllab.riec.tohoku.ac.jp/papers/icfp2011OhoriUenoAuthorVersion.pdf
9. Azza Abouzeid, Kamil Bajda-Pawlikowski, Daniel Abadi, Avi Silberschatz, and Alexander Rasin. 2009. HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads. Proc. VLDB Endow. 2, 1 (August 2009), 922-933.
10. Nautiyal, Siddharth. 2012. bessemer cloud computing law #8: the most important part of saas. http://www.bvp.com/blog/bessemer-cloud-computing-law-8-most-important-part-saas
11. D. Jiang, B. C. Ooi, L. Shi, and S. Wu. The Performance of Map Reduce: An In-depth Study. PVLDB, 3(1), 2010.
12. Amazon SimpleDB Pricing. http://aws.amazon.com/simpledb/pricing/
13. Amazon DynamoDB (beta) http://aws.amazon.com/dynamodb/
14. Firebase A scalable real-time backend for your web app. https://www.firebase.com/
15. IDEF1X Data Modeling Method http://www.idef.com/IDEF1x.htm