**Technology Review**

When I started this project, I didn’t have a clear picture in my heads of where to go to find more information. My project supervisor pointed me in the direction of several introductionary articles and books to help me become familiar with the basic concepts and theory. The first of these, [X], was informative, but I found it difficult to get through at first, The second of these was Joe Celko’s book[X]. This was the second text I read in my project research. It helped me solidify my understanding on what BASE and ACID transactions were, and why BASE transactions were preferred over ACID transactions in certain situations. It also introduced me to several important pieces for background information, such as the CAP theorem.

Finding information on the relevant project itself was difficult. This is because no one has tried to build an emulator like this before. This means that there did not appear to be a lot of material available to me initially. Once I clarified what the word ‘Models’ meant in the original project brief, I began to find more useful information.

Another hurdle that I encountered during my individual research was the fact that along with the consistency models being relatively new to me, most of the databases worked in different ways to the MySQL databases that I normally work with. An example of this would be Dynamo, which is a key value store, I theretofore found the paper at [x[ to be useful in putting the design decisions found in a lot of these cloud databases into context.

I found The Cloud Handbook{2] to be particularly useful. The main reason for this is that it provided descriptions of real cloud systems and the extent to which they implement the BASE propertied as well as brief descriptions of the algorithms themselves. This allowed me to narrow my search for information considerably.

In a similar fashion, I found the location at [X] useful in looking for alternative approaches to enforcing the three base properties. This is particularly true of basic availibllity, as there are more strategies around for this then there are for eventual consistency.

One of the most useful resources was Amazons cloud database. The reason for this is that the paper describes the approach to ensuring consistency is achieved well and in great detail. While reading the appropriate chapter of 7 Databases in 7 weeks, I also was able to download Risk, a database that is built around the same consistency model that Amazon uses. I found this to be one of the most promising souces, as the algoritms discussed here seem to be able to be emulated effectively.

Another highly useful approach was that provided by CouchDB. I found here that the CouchDB handbook, freely available online, provided thorough and in depth descriptions of the technical algorithms used, and I was able to start coming up with some ideas on how I might implement this. It is also worth noting that the algorithms used by couchDB were so effective, that variants of them have been produced, such as at [x]

A book I found very suseful was [x] I found this book particularly useful once I had looked at all of the major cloud databases. The reason for this is because the majority of the databases this book chooses to study are NoSQL databases that import some form or part of BASE transactions. Therefore, this book was useful in getting more information on certain algorithms or techniques that might not be gained in other publications. An example is one of the databases studied is the risk database which is based on Amazon’s Dynamo database. As a result, it gives a very detailed description of the vector clocks algorithm for consistency used by Dynamo, whereas the official paper by Amazon only gives more of an overview. It also shows how these databases are used in practice, which makes the theoretical concepts easier to digest.

Another approach that I considered was Google’s model of achieving eventual consistency. While I find the algorithms employed here to be particularly clever, I haven’t decided to implement them for this project. This is due to several reasons. Firstly, the Google ecosystem that supports GFS is highly complex, not only encapsulating the database, but also a lock server and various other components as well. This means that it would be too complex to emulate within the time available to me and also require the system to have a much closer access to the database. This might not be possible due to the fact that ultimately, the data will be stored on a standard ACID database.

A key component of the literature review for me was finding out about the programming languages and tools that I would be using. Scala for the Impatient, and Akka Concurrency were found to be essential in getting a solid grasp of the Scala programming language down. I also found it useful to refer to Ian Somerville’s Software Engineering for ideas and advice on how to plan such a large piece of work.

Once I had a working prototype of eventual consistency, I then started to look at the other cloud property that I wanted to emulate if there was time. This was basic availability. First of all I considered the approach used by CloudDB, which involves having one Master replica to coordinate the other servers, and this master can be re elected whenever it goes down.

I also looked again t the Google file system This used a similar sor t of master slave system, except the rules and protocols were more elaborate, with only certain kinds of data going through the master and support form other system comonents outside the database. I found this approach incribly interesting, but thought it might be too complicated for the task at hand.

Another approach I considered was to use MongoDB. MongoDB is another popular cloud database, it is fully consistent but it uses replication to ensure it is always available. I like CouchDB, I found it used a similar sort of master salve scheme, with the main differneces being in h masters and slaves were selected. I found it interesting that several of the ajor cloud databases used this approach, and started to question why this is. I wonder if I might be able to find out over the course of the project.

As I am aiming ot eventually run my application n a real cloud platform, I also did some research of real cloud systems to guide me in writing the example programs. I first of all started investigating the cloud Offerings that were provided by Google. The reason for this was because I already had limited experience with the platform as part of the Cloud computing module. I found that Googles datastore was not very well suited to my needs, because it was not eventually consistent.

I then started looking at Amazons Cloud systems. The reason for this was that I am loosely basing the ventuaul consistency implementation on Amazon’s Dynamo Database. So I thought that it would be a good point of comparson. Originally, I started looking at the S3 service. B changed to using Dynamo itself, because then I would be comparing my application against the real thing.

**Anderson, C. J., Lehnardt, J. and Slater, N. (2010) *CouchDB: The Definitive Guide*. Sebastopol, CA: O’Reilly Media, pp. 11–20.**

**Burckhardt, S., Leijen, D., Fähndrich, M. and Sagiv, M. (2012) ‘Eventually Consistent Transactions’, *Programming Languages and Systems*. Springer, pp. 67–86. doi: 10.1007/978-3-642-28869-2\_4.**

**Celko, J. (2013) *Joe Celko’s Complete Guide to Nosql: What Every SQL Professional Needs to Know about Nonrelational Databases*. United States: Morgan Kaufmann. pp 3-14.**

**Chang, F., Dean, J., Ghemawat, S., Hsieh, W., Wallach, D., Burrows, M., Chandra, T., Likes, A. and Gruber, R. E., Google (2006) ‘Bigtable: A Distributed Storage System for Structured Data’, in *OSDI*.**

**DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P. and Vogels, W. (2007) ‘Dynamo: Amazon’s Highly Available Key-value Store’, *ACM SIGOPS Operating Systems Review*, 41(6).**

**Edlich, P. D. S. (*NOSQL Databases*. Available at: http://nosql-database.org (Accessed: 1 April 2015).**

**Furht, B., Escalante, A. and editors. (2010) *Handbook of Cloud Computing*. Edited by B. Furht and A. Escalante. Boston, MA: Springer Science+Business Media.pp 138-158**

**Horstmann, C. S. (2012) *Scala for the impatient*. 1st edn. United States: Addison-Wesley Educational Publishers Inc.**

**Kalin, M. (2009) *Java Web Services: Up and Running*. United States: O’Reilly Media, Incorporated.**

**Kuznetsov, S. D. and Poskonin, A. V. (2014) ‘NoSQL data management systems’, *Programming and Computer Software*. Springer, 40(6), pp. 323–332.**

**Pritchett, D. with eBay (2008) *BASE: An Acid Alternative - ACM Queue*, *ACM Queue*. Available at: http://queue.acm.org/detail.cfm?id=1394128 (Accessed: 30 March 2015).**

This

**Redmond, E. and Wilson, J. R. (2012) *Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement*. Lewisville, TX: The Pragmatic Programmers.**

***Sharding and replication* Available at: http://rethinkdb.com/docs/sharding-and-replication/ (Accessed: 1 April 2015).**

This webpage provides a detailed look at Rethinks replication strategy. This is a possible candidate for a model of basic availability that I could implement if I get the time.

**Sommerville, I. (2011) *Software Engineering*. 9th edn. Harlow: Pearson Education (US).**

**Stirling, S., JavaWorld, S. S. and PT, 2004 1:00 AM (2004) ‘Testing J2EE applications’. JavaWorld. Available at: http://www.javaworld.com/article/2072923/testing-debugging/testing-j2ee-applications.html (Accessed: 1 April 2015).**

**Wyatt, D. K. (2013) *AKKA Concurrency*. Canada: Artima Inc**