UNIVERSITI PUTRA MALAYSIA

MOBILE SYNCHRONIZATION FRAMEWORK TO ENHANCE LARGE OBJECT MANAGEMENT IN MOBILE CLOUD STORAGE SERVICE

YUNUS PARVEJ FANIBAND

FSKTM 2022 1

MOBILE SYNCHRONIZATION FRAMEWORK TO ENHANCE LARGE OBJECT MANAGEMENT IN MOBILE CLOUD STORAGE SERVICE

By

YUNUS PARVEJ FANIBAND

Thesis Submitted to the School of Graduate Studies, Universiti Putra Malaysia, in Fulfilment of the Requirements for the Degree of Doctor of Philosophy

June 2021

Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfilment of the requirement for the degree of Doctor of Philosophy

MOBILE SYNCHRONIZATION FRAMEWORK TO ENHANCE LARGE OBJECT MANAGEMENT IN MOBILE CLOUD STORAGE SERVICE

By

YUNUS PARVEJ FANIBAND

June 2021

Chairman : Ts. Iskandar Ishak, PhD
Faculty : Computer Science and Information Technology

Supporting large file upload and retrieval is crucial for the mobile cloud storage services, as file sizes are trending larger and mobile users' access or share files in large size. These files range from large PDFs to media files in various formats such as MPEG videos. These files need to be used offline, sometimes updated and shared among users. The mobile environment with frequent disconnections and limited bandwidth, impact the data and transaction management as well as the data consistency guarantees. Most of the mobile frameworks use a data-sync paradigm in order to handle disconnected operations, in which data is stored locally on the device and replicated to the cloud asynchronously.

Many applications are storing large amounts of structured and unstructured data. However, it is challenging for mobile synchronization frameworks to manage data consistency in such environment. This is due to the fact that handling large data and maintaining consistency are very challenging in cases of local storage and updates.

This research proposes an enhanced cloud-based mobile synchronization framework to address the two main problems that is data management for large objects and end-to-end support for data consistency for large objects. This research work proposes techniques to improve large file object access and synchronization in mobile cloud environment based on segmentation and object chunking. This framework mainly focuses on large file handling and providing support for both table and objects data models that can be tuned for three consistency semantics: resembling strong, causal and eventual consistency. Experimental results conducted using representative workloads showed that the proposed enhanced Mobile Synchronization framework can handle large files with the size ranging from 100MiB up to 1GiB and is able to reduce synchronization time with object chunking (2, 4, 8 and 16 MiB) in the experiment settings by 65.4% for upload and 93.7% for download on average when compared to other frameworks.

Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia sebagai memenuhi keperluan untuk ijazah Doktor Falsafah

RANGKA KERJA PENYEGERAKAN MUDAH ALIH UNTUK MENINGKATKAN PENGURUSAN OBJEK BESAR DALAM PERKHIDMATAN PENYIMPANAN AWAN MUDAH ALIH

Oleh

YUNUS PARVEJ FANIBAND

Jun 2021

Pengerusi : Ts. Iskandar Ishak, PhD
Fakulti : Sains Komputer dan Teknologi Maklumat

Sokongan muat naik dan mendapatkan fail yang besar adalah penting untuk perkhidmatan storan awan mudah alih kerana saiz fail semakin berkembang dan akses pengguna mudah alih atau berkongsi fail dalam saiz yang besar. Fail-fail ini terdiri daripada PDF besar kepada fail media dalam pelbagai format seperti video MPEG. Fail-fail ini perlu digunakan di luar talian, kadangkala dikemas kini dan dikongsi di kalangan pengguna. Persekitaran mudah alih dengan sambungan yang kerap terputus dan lebar jalur terhad, memberi kesan kepada pengurusan data dan transaksi serta jaminan ketekalan data. Kebanyakan rangka kerja mudah alih menggunakan paradigma penyegerakan data untuk mengendalikan operasi yang terputus sambungan, di mana data disimpan secara setempat pada peranti dan direplikasi ke awan secara tak segerak.

Banyak aplikasi menyimpan sejumlah besar data berstruktur dan tidak berstruktur. Walau bagaimanapun, mengurus konsistensi data dalam persekitaran sedemikian adalah mencabar bagi rangka kerja penyegerakan mudah alih. Ini disebabkan oleh pengendalian data yang besar dan mengekalkan konsistensi adalah sangat mencabar dalam kes storan dan kemas kini tempatan.

Penyelidikan ini mencadangkan rangka kerja penyegerakan mudah aih berasaskan awan yang dipertingkatkan untuk menangani dua masalah utama iaitu pengurusan data untuk objek besar dan sokongan hujung ke hujung untuk konsistensi data bagi objek besar. Kerja penyelidikan ini mencadangkan teknik untuk menambah baik capaian dan penyegerakan objek fail besar dalam persekitaran awan mudah alih berdasarkan segmentasi dan pemotongan objek. Rangka kerja ini tertumpu terutamanya pada pengendalian fail besar dan menyediakan sokongan untuk kedua-dua model data jadual dan objek yang boleh ditala untuk tiga semantik konsisten: menyerupai konsistensi yang kuat, sebab dan akhirnya. Keputusan eksperimen yang dijalankan menggunakan beban

kerja yang mewakili menunjukkan bahawa rangka kerja penyegerakan mudah alih yang dipertingkatkan boleh mengendalikan fail besar dengan saiz antara 100MiB hingga 1GiB dan mampu mengurangkan masa penyegerakan dengan pemotongan objek (2, 4, 8 dan 16 MiB) dalam eksperimen tetapan sebanyak 65.4% untuk muat naik dan 93.7% untuk muat turun secara purata jika dibandingkan dengan rangka kerja lain.

ACKNOWLEDGEMENTS

Alhamdulillah, thank you, Allah, for the strength that He has given me, for the wisdom that He has granted me, and for the unconditional love that He has shown me until I can pursue and completed my Doctor of Philosophy degree. Without You, I would never have the perseverance to make it until the end.

I would like to thank my advisor Dr Iskandar, without whom this thesis would not have been possible. He taught me design, implementation, and evaluation skills that are critical in completing this dissertation. His patience, encouragement and unparalleled expertise and insight in experimental systems research have been a constant source of guidance throughout the thesis research process. I also would like to thank other members of my thesis committee, Dr Fatimah Sidi and Dr Marzanah A. Jabar for their valuable help that significantly improves both the technical content and the presentation of this dissertation.

My stay at the Faculty of Computer Science and Information Technology of UPM would not have been as enjoyable and rewarding without the wonderful people who made it a great place to do research in computer science. I am deeply indebted to my faculty and classmates, for their invaluable help in my development as a researcher both technically and personally. I also would like to thank my officemates, past and present, for creating an intellectually stimulating environment.

My parents taught me the value of hard work and instilled in me the desire to be successful. Without their love and sacrifice, I never could have come this far. Thanks also to my daughter, Hiba and son Ayaan, for reminding me that there is so much more to life than computer science. Finally, my wife, Reshma, deserves greater thanks than I can possibly give. Over the past many years, she has been an immeasurable source of strength, love, comfort, and support.

YUNUS PARVEJ FANIBAND

This thesis was submitted to the Senate of Universiti Putra Malaysia and has been accepted as fulfilment of the requirement for the degree of Doctor of Philosophy. The members of the Supervisory Committee were as follows:

Iskandar bin Ishak, PhD
Senior Lecturer, Ts
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Chairman)

Fatimah binti Sidi, PhD
Associate Professor, Ts
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Member)

Marzanah binti A. Jabar, PhD
Associate Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Member)

ZALILAH MOHD SHARIFF, PhD
Professor and Dean
School of Graduate Studies
Universiti Putra Malaysia

Date: 10 February 2022

Declaration by graduate student

I hereby confirm that:

- this thesis is my original work;
- quotations, illustrations and citations have been duly referenced;
- this thesis has not been submitted previously or concurrently for any other degree at any institutions;
- intellectual property from the thesis and copyright of thesis are fully-owned by Universiti Putra Malaysia, as according to the Universiti Putra Malaysia (Research) Rules 2012;
- written permission must be obtained from supervisor and the office of Deputy Vice-Chancellor (Research and innovation) before thesis is published (in the form of written, printed or in electronic form) including books, journals, modules, proceedings, popular writings, seminar papers, manuscripts, posters, reports, lecture notes, learning modules or any other materials as stated in the Universiti Putra Malaysia (Research) Rules 2012;
- there is no plagiarism or data falsification/fabrication in the thesis, and scholarly integrity is upheld as according to the Universiti Putra Malaysia (Graduate Studies) Rules 2003 (Revision 2012-2013) and the Universiti Putra Malaysia (Research) Rules 2012. The thesis has undergone plagiarism detection software

Signature: _____ Date: _____

Name and Matric No: Yunus Parvej Faniband

Declaration by Members of Supervisory Committee

This is to confirm that:

- the research conducted and the writing of this thesis was under our supervision;
- supervision responsibilities as stated in the Universiti Putra Malaysia (Graduate Studies) Rules 2003 (Revision 2012-2013) are adhered to.

Signature: _____
Name of Chairman of Supervisory Committee: Dr. Ts. Iskandar bin Ishak

Signature: _____
Name of Member of Supervisory Committee: Associate Professor Dr. Ts. Fatimah binti Sidi

Signature: _____
Name of Member of Supervisory Committee: Associate Professor Dr. Marzanah binti A. Jabar

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Programming Interface
C	Causal Consistency
CMP	Consistency scheme with PRACTI property
CRDT	Conflict-Free Replicated Data Types
HU	Hoarding Unit.
E	Eventual Consistency
FSC	Fork-sequential consistency
MBaaS	Mobile Backend as a Service
MCC	Mobile Cloud Computing
MRC	Monotonic read consistency
MWC	Monotonic write consistency
PRACTI	*P*artial *R*eplication, *A*rbitrary *C*onsistency, *T*opology *I*ndependence
RAWC	Read after writes consistency
RPC	Remote Procedure Call
RYWC	Read your writes consistency
S	Strong Consistency
SC	Sequential Consistency
T, O	Table, Object
WFRC	Write Follows Read Consistency

CHAPTER 1

INTRODUCTION

1.1 Background Concepts

The model of mobile cloud computing utilizes the services of cloud computing. The mobile cloud environment comprises portable computing devices, mobile Web and location-based services, supported by wireless communication infrastructure, to provide mobile devices online access to large storage space and unlimited computing power. A wireless network with mobile clients (Figure 1.1) is fundamentally a distributed system but suffers from the primary challenges such as limited computational power and storage of the mobile devices, intermittent loss of connectivity and battery power restrictions. The transmission bandwidth of the mobile device is likely to be lesser than the transmission bandwidth of the mobile support stations (MSSs) and this leads to the phenomenon of communication asymmetry. The effective management of data in systems with mobile client is affected with these limitations. The environment of frequent disconnections and limited bandwidth impact the data and transaction management as well as the data consistency guarantees.

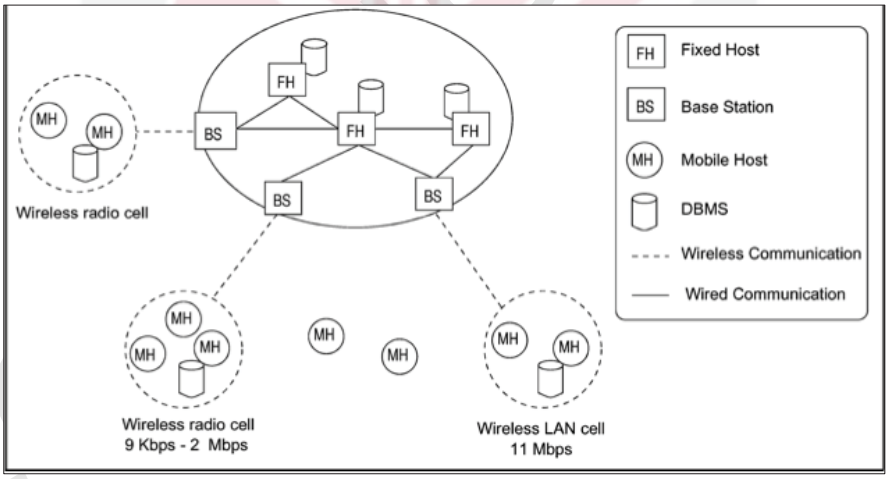

Figure 1.1 : A global architecture of mobile environment

To provide the illusion of uninterrupted data access, the data management must hide the constraints of mobile wireless computing. The technique of replicating data locally on the mobile device enables the user to carry offline data without the need to always be connected to the data server. The ability to disconnect with the network, do local changes, and then reintegrate (synchronize) these changes back into the system makes

the mobile gadget an essential extension to modern distributed databases and collaborative tools.

Data synchronization (Perkins et al., 2015) (Satyanarayanan, 1996) (Pitoura & Samaras, 2012) is an empowering process that eliminates the critical requirement of having steady connectivity and permits users to run data-centric mobile applications while being offline.

Generally, the mobile applications (which are referred to as “Apps') are developed according to the different application programming interfaces (API) abstractions supported by the underlying mobile middleware. The middleware may provide a simple file-based API (possibly extended with replication- specific methods). It may also support complex abstraction such as objects, tuples, relational entities or an object which may contain pointers to other interdependent objects. Middleware with database replication primarily provides query-oriented CRUD APIs (Create, Read, Update and Delete) to application developers for typical operations on data with declaratively defined by SQL queries for update, creation/insertion, and deletion of records.

Supporting large file upload and retrieval is crucial for the mobile cloud storage services, as file sizes are trending larger and mobile users’ access or share files of large size (Z. Li et al., 2016) (Shanon Montelongo, 2019). Practical large object services are, however, only available for PC clients and not for mobile apps. The key observation from the study of both the commercial and open-source cloud storage services for mobile reveal that out of 19 only 9 frameworks (47.36% and most of them are commercial and closed source) support large objects and few also have limitations (in terms of maximum file upload size, option of chunking support and the techniques of handling large objects for better performance). Even though the commercial cloud frameworks provide support for large objects, many frameworks do not handle large files.

Amazon DynamoDB currently limits the size of each item that app store in a table. The maximum item size in DynamoDB is 400 KiB, which includes both attribute name binary length and attribute value lengths. Simba's (Perkins et al., 2015) sync protocol does not support streaming APIs to handle big size objects (e.g. Media file like Videos). SwiftCloud (Preguiça et al., 2014) is a middleware system that implements a Key-CRDT on top of Riak (Klophaus, 2010). The Riak designers do not recommend storing objects over 50 MiB for performance reasons. Izzy (Hao et al., 2013b) is an initial version of Simba and do not support large object. The Parse Server (Parse, 2018) only supports files up to 10 MiB.

1.2 Motivation

A general architecture for mobile cloud computing environment (Figure 1.1) has two unique sets of entities: Fixed hosts (FHs) and Mobile hosts (MHs). FHs are machines (Works stations and Servers) with efficient computation power and reliable storage of data and run large databases (Barbará, 1999) (Jing et al., 1999) (Barbará, 1999) (Singh & Hasan, 2019). FHs that are connected through a fixed network. MHs with limited processing and storage power (cellular phone, palmtops, laptops, notebooks) are not continually communicating with the fixed network. They may be disconnected for various reasons. Additional dedicated fixed hosts called mobile support stations (MSSs) acts as the channel between the FH and MH through wireless LAN (local area network) connections, cells or connections to the network with standard modems.

When the network connectivity becomes unavailable or unacceptable, the MH enters the disconnected state. Disconnected operation (See Figure 1.2) is a three-stage changeover between the following states (Kistler & Satyanarayanan, 1992) (Pitoura & Samaras, 2012) as follows:

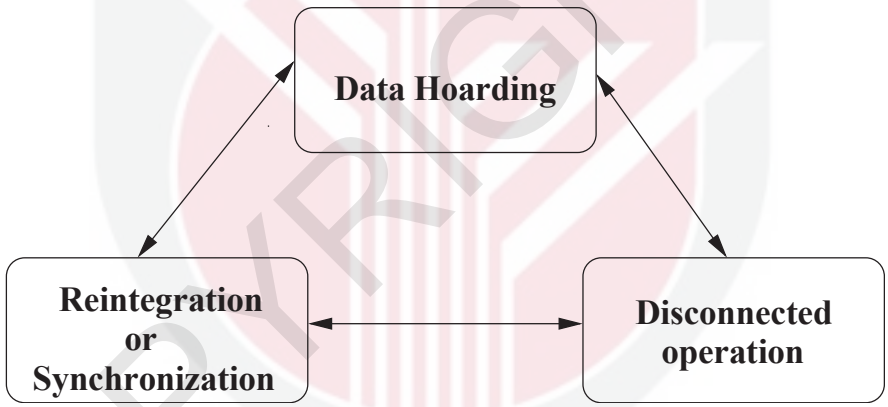

Figure 1.2 : Disconnected Operations

1. **Data hoarding**: This is the process of preloading or pre-fetching the data in anticipation of a foreseeable disconnection. Before going to offline mode (disconnection), the data structures necessary for operation during disconnection are either replicated (cached) or moved (partitioned) at the MH.

Table 1.1 : Issues in Disconnected Operation

State	Problem	Resolution
Hoarding	Unit of caching/hoarding	System dependent (e.g., a file or a database fragment)
	Which items to cache (hoard)?	- Application dependent, based on purpose of the system - Defined distinctly by the user - Generate from the knowledge of past operations
	When to execute hoarding?	- Based on regular intervals - Before disconnection
	Call for locally unavailable data	- Add requests to queue for future service - Raise an exception/error
Disconnection	What to log?	- Timestamps - Data Values - Operations
	When to optimize the log?	- Before synchronization - Incrementally
	How to optimize the log?	- System dependent
	How to synchronize?	- Re-execute an operational log
Reintegrationor Synchronization	How to resolve conflicts?	- Automatic resolution - Use application-semantics Provide utility to aid the user

2. **Disconnected operation**: When the MH is offline (disconnected from the network), data might be changed, added or even removed at either the MH or the FH.
3. **Synchronization or Reintegration**: When the connection is reestablished, each operation executed at the MH should be synchronized (reintegrated) with appropriate updates executed at other sites in order to attain seamless consistency.

For a given distributed system, the complexity of operations in each of the above three states is determined by the interdependence of data operated on. The issues pertaining to three states (Pitoura & Samaras, 2012) are summarized in Table 1.1.

The execution of distributed applications in local-area networks is significantly different from wireless, mobile systems. Wireless applications must use different communication pattern in order to address the high latency, low bandwidth, intermittent connections and communication charges based on time and content. An application operating on a LAN can manage good user interactions in case queries to a non-local database, but the same application operating on a wireless network may become unresponsive due to the delay in response. Hence wireless applications chose data replication, explicit or implicit (caching or data hoarding), as the primary technique to address the Disconnected operation.

The introduction of multi-user and collaborative features for wireless application increase the complexity, as multiple users have to share data objects and thus communicate and collaborate with each other (Munson & Dewan, 1997) (Pitoura & Samaras, 2012). In such cases there must be a sophisticated coordination mechanism other than the conventional mechanism of locks. Thus, addressing the wireless mobile systems constraints in the application development becomes challenging for mobile developers, since they have to retain favourable user interaction and performance along with tackling the data coordination issues.

Mobile services can be developed and deployed in various cloud computing scenarios. The main service models of cloud computing are (Mell & Grance, 2011) Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). With the advent of new service model, Backend as a Service (BaaS), sometimes also referred as Mobile Backend as a Service (MBaaS), the native mobile applications can be easily integrated with the cloud. Throughout this dissertation the term "Mobile Backend as a Service (MBaaS)" always refers to a "sync framework" and sometimes used as interchangeable terms.

Synchronization frameworks should (Perkins et al., 2015) (Gheith et al., 2016):

- Facilitate non-blocking, responsive (ensure high availability) and reliable mobile applications during disconnection.
- Support Cloud-connected multi-user, shared-data mobile apps that need to manage the inter-dependent data not only locally but also across multiple devices with cloud storage.
- Provide a synchronization model with tunable consistency guarantees so that developers have the flexibility to configure how data is synchronized and data conflict are handled.
- Provide a synchronization-aware high-level APIs that support applications for on demand and background synchronization tasks.
- Enable support for large files (i.e., a couple of megabytes or gigabytes) synchronization.
- Needs to be frugal in power consumption and bandwidth usage for mobile clients and hold efficient repeated sync operations.

The model of Sync framework offers a cloud server infrastructure, to store application data and facility of easy configuration. Developer needs to do significant work for the application to remain responsive during interruptions in communication due to poor or no network. The Sync framework offers a solution for the unreliable connection problem with customized synchronization and replication processes and hence helps in synchronizing with multiple clients. An intelligent Sync framework allows enterprise data to take offline and facilitate sync operation by syncing data across multiple mobile devices with the backend systems, detect and resolve the conflicts with configurable, standards-based rules, setting precedence based on policies (Satyanarayanan, 1996)

(Gheith et al., 2016). Ideally Sync framework should provide consistent state at all times (strong consistency). But the CAP theorem for the distributed systems enforces the Sync framework to guarantee immediate availability and tolerate network partitions in order to provide weak form of consistency, commonly known as eventual consistency (Agrawal, Aranya, & Ungureanu, 2013).

Each Sync framework offers a distinctive set of functionalities through APIs (REST or wrapper libraries of the APIs) and allows programs to be written specially to execute in the cloud. Amazon Mobile SDKs provide the means to interact with cloud services through REST APIs. Multi-platform SDKs (iOS, Android, Fire OS, and Unity) are offered to interact with the AWS services, including S3 (storage), DynamoDB (database), Simple Notification Service (SNS) and Mobile Analytics (Mobile, 2016). Apple provide iCloud service (CloudKit SDK) to store and access data in iCloud (Shraer et al., 2018). Mobile applications are broadly classified into two types such as offline applications and online applications (H. Wu et al., 2010a). Unlike online apps, in offline (native) application, the mobile device and back-end system are not connected always. In order to support continuous mobile services, offline applications will process the presentation and business logic with the available local data on the device itself. Periodically data is updated by synchronizing with back-end systems.

Majority of the sync frameworks support either table, or file-only data model. But the Data centric apps need to address the use cases that interact with both inter-dependent structured and unstructured data. While some of the frameworks provide sync protocol that does not support streaming APIs to handle big size objects (e.g., Media file like Videos), the other need to be improved in the area of cross-app synchronization, optimization strategies and caching (Perkins et al., 2015) (Preguiça et al., 2014) (Chun et al., 2012). Sync performance measurement of some cloud services showed that sync protocol implemented in these fails to utilize bandwidth when synchronizing multiple small files or in high real-time traffic (RTT) environment (Drago, Bocchi, Mellia, Slatman, & Pras, 2013) (Bai & Zhang, 2017).

Handling the task of uploading and retrieving large files from and to a mobile app is a cumbersome process for developers due to issues of latency, speed, timeouts and interruptions (Drago et al., 2013) (Perkins et al., 2015) (Bai & Zhang, 2017) . With the growing prevalence of sharing file of larger sizes among mobile users, providing reliable and efficient synchronization service for large files has become an important feature.

The term "consistency" refers to the notion that the state of data such as the latest status or the collection of changes that give rise to that state is decided by various clients accessing a storage system somehow. For two reasons it is difficult to make sure consistency: first, storage systems need to maintain several replicas of fault tolerance and performance data. Second, multiple data items or objects can involve storage operations.

Recently numerous measurement research efforts have been conducted on enterprise cloud storage services and personal cloud storage services. The investigations from (Z. Li et al., 2016) attempted to find out mobile user access behavior in a large-scale mobile cloud storage with a dataset of 350 million HTTP request logs. The study observed the trend of using the cloud storage for large file sharing, with the average volume as large as about 70 MiB, in multiple sessions for retrieving one file.

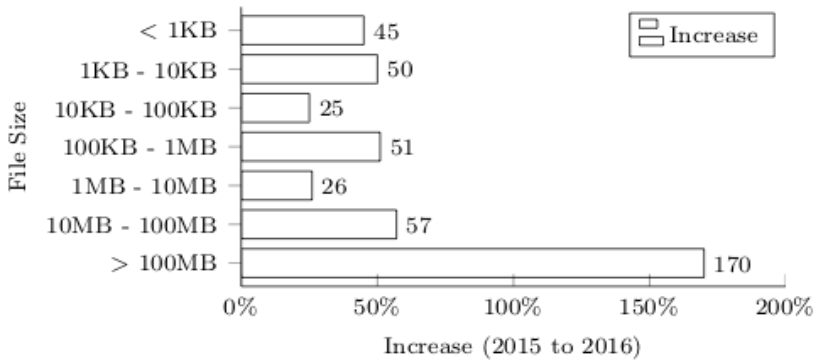

Figure 1.3 : Increase in files uploaded by File size

Another study from a cloud storage service provider (Shanon Montelongo, 2019) analyzed a dataset of 100,000 applications. They provided the services of handling file uploads, transformations, storage, and delivery. Their observation targeted the statistics of uploaded trend of files ranging from different sizes and formats from the year 2015 to 2016 as shown in Figure 1.3. Their analysis concluded that all file increased 50% year over year, but files sized 100 MiB and above increased over 170% year over year. Based on these studies it is evident that the file sizes are trending larger and mobile users' access or share large size of files (above 100 MiB). Some of the mobile operating systems limit the size of the file over which over- the-air (OTA) or app-store downloads are not allowed (Ketola, 2014). For example, Apple's iOS platform ("iOS app OTA limit in cellular network - Apple Community," 2016) limit, the Cellular Data downloads to a file size of 100 MiB. Android OS limit to the size of downloads via cellular data to 150MiB ("Reduce your app size," 2016). Based on above studies and mobile operating system guidelines, it can be concluded that a file with size greater 100 MiB is considered as a large file.

1.3 Problem Statement

Even though many mobile sync frameworks support the data replication and management systems for mobile clients, they lack the support for large objects (more than 100 MiB to Giga bytes) (Perkins et al., 2015) (Hao et al., 2013b) (Preguiça et al., 2014) (Balegas et al., 2015) (Parse, 2018) ("The BaasBox Server," 2019). The key observation from the literature study revealed that out of 19 only 9 frameworks (47.36%

and most of them are commercial and closed source) support large objects and this include the commercial frameworks, additionally few also has limitations (in terms of maximum file upload size, option of chunking support, configuration and the techniques of handling large objects for better performance). Handling large data and maintaining consistency become challenging in cases of local storage and updates, on the cloud, and on other client mobile devices. Reliable (transparent failure handling), Consistent (concurrent updates, sync atomicity) and Efficient (minimize traffic/battery usage) Sync as a service framework are needed for building data centric apps that can handle large objects (Go, Agrawal, Aranya, & Ungureanu, 2015) (Hao, Agrawal, Aranya, & Ungureanu, 2013a).

Also, most mobile frameworks use a data-sync paradigm in order to handle disconnected operations, in which data is stored locally on the device and replicated to the cloud asynchronously. However, it is challenging for frameworks to manage data consistency in such environment due to 1) the limited network bandwidth and intermittent connectivity, which are common to mobile devices, and 2) many apps storing inter-dependent structured and unstructured data (Agrawal et al., 2013). Most sync frameworks support either a table, or a data template that is file-only (Perkins et al., 2015). But the data-centric apps have scenarios to communicate with structured and unstructured data that are interdependent, and the sync framework must ensure that no dangling pointers from structured to unstructured exist. Because mobile apps frequently crash or stall for a variety of reasons sources (Agarwal et al., 2010) (Ravindranath et al., 2014), if an app is in the middle of a data operation (a local write or sync) when a failure occurs, the sync framework must detect and recover to a consistent state.

Cloud storage providers use various techniques such as Chunking, Bundling, Segmentation, Compression, Deduplication, and Delta encoding to maximize storage space and speed up transmission of data (Drago et al., 2013). Notwithstanding the efforts, the sync performance of common mobile cloud storage services is still far from being satisfactory, and the sync time is much longer than anticipated under some circumstances. There are several challenges to improve the performance of syncs in the mobile / wireless environment.

De-duplication techniques to reduce redundant data transfers do not always result in sync efficiency (Drago et al., 2013). The distributed nature of storage facilities makes realistic implementation of the delta encoding algorithm difficult, and failure in incremental sync results in high overhead traffic. When synchronizing a collection of files across a slow network, the iterative sync scheme suffers from low throughput (Cui et al., 2017).

While some existing frameworks aim to boost sync efficiency by integrating multiple capabilities, it is still unclear if such capabilities are useful or necessary in mobile / wireless environments for good storage efficiency (Perkins et al., 2015).

As commercial storage services with sync frameworks are largely closed source with encrypted data, the researchers remain unclear regarding their designs and operating processes. It is difficult to investigate the sync protocol specifically and determine the root cause of sync difficulty (Cui et al., 2017a).

Eventually, as a mobile cloud storage and sync framework requires storage and network technologies, storage techniques need to be flexible and operate effectively in a mobile environment where mobility and changing channel conditions make communications vulnerable to high delay or interruption (Hao et al., 2013a).

Although several mobile sync frameworks support mobile customer data replication and management systems, they lack support for large objects (more than 100 MiB to Giga bytes) (Perkins et al., 2015) (Hao et al., 2013b) (Preguiça et al., 2014) (Balegas et al., 2015) (Parse, 2018) (The BaasBox Server, 2019).

Since many of the Mobile sync frameworks does not support large objects and have some limitations (in terms of maximum file upload size, option of chunking support, configuration and the techniques of handling large objects for better performance and latency), this research work proposes an enhanced cloud- based Mobile Sync framework to address the following two main problems:

1) Data management for large objects
2) End-to-end data consistency for large data objects

1.4 Research Objectives

This research aims to improve data management for large objects and improve the end-to-end data consistency for Mobile Sync framework. In order to achieve the aim following are the research objectives:

1. To propose a mobile data management method based on object segmentation and object chunking in order to improve large file object access and synchronization in a mobile cloud environment. (Measured using Latency (in seconds) during Upload (for varying chunk sizes (2, 4, 8 and 16 MiB)), Download and Delete operations and also using cumulative sync protocol % overhead with varied payload sizes.)
2. To propose an enhanced Mobile Sync framework to improve end-to-end data consistency for large file object in mobile cloud environment. (Measured using end-to-end latency (in seconds) for a specific size object for three consistency schemes of Strong, Causal and Eventual, in a defined environment).

1.5 Research Scope

The scope of this research work is defined in the following points:

- This research focus on addressing only a part of data service with the Synchronization and off-line services for mobile devices. Specifically, this research is focused on large file object support providing end-to-end data consistency in MBaaS framework. So other services that are generally integrated into MBaaS framework like Identity services, social network integration and analytics are out of the scope of this research.
- The implementation of data Storage of proposed MBaaS sync framework, use Cassandra to store tabular data and Open Stack Swift object storage, for object data. Internally, the architecture of the Backend as a Service (BaaS) provider determines how data is stored, replicated, and partitioned. This metrics influence the systems scalability, availability, consistency, and flexibility.
- The case studies chosen in this research are based on the mainly three criteria. First the frameworks should be open source for detailed investigations. Second the selected frameworks utilize different technologies to support data management and consistency features. Thirdly support for different types of data models like (1) File-only, (2) Table-only and combination of (3) Table and Object.
- This research considers the investigation of open-source Mobile synchronization frameworks in detail. Hence this work compares the consistency performance of the enhanced framework with the parent open-source framework Simba (Perkins et al., 2015). Also, the performance comparison of enhanced framework for upload and download performance is covered for open-source frameworks like ParseServer (Parse, 2018) and BaasBox ("The BaasBox Server," 2019) only. This work did not compare the consistency performance with other frameworks like ParseServer andBaasBox since it was difficult to make the setup for consistency measurement. It was due to the fact that two frameworks ParseServer and BaasBox does not made available required sample source code for the setup code for Reader (Tr), Writer (Tw) and CausalTester (Tc) for the underlying data storage provider. Also, it was difficult to provide the same evaluation environment for all these frameworks.

1.6 Research Contributions

This research work proposes an enhanced cloud-based Mobile Sync framework to address two main issues concerning data management and support End-to-end data consistency for large data objects. This research work contributes to the body of knowledge in following aspects:

1. **Efficient method for large data management for mobile devices using object segmentation and chunking:** This study proposed the techniques of Object segmentation and Chunking that manage large objects by producing a low number of objects with large chunk size and hence, a low object-to- node ratio resulting in faster read-writes, in a mobile cloud environment. The approach also utilizes efficient data reduction (compression) and bandwidth reduction techniques during the large data transfer.

2. **An enhanced cloud-based sync framework to support End-to-end data consistency and large data object access:** This work design and implemented an enhanced cloud based Mobile Sync framework to support End-to-end data consistency support for large data object management using Open stack swift object storage APIs and Cassandra to support both tabular and large object data. The proposed enhanced cloud based Mobile Sync framework (NetMob) support API Interface that allows the large objects to be written to or read from the cloud storage and also support local reading or writing only a part of the large object. Efficient network transfer is supported through the chunking methods and objects are stored and synced as a collection of fixed-size chunks. The framework also supported three types of consistency guarantees Strong, Causal and Eventual consistency.

1.7 Organization of the Thesis

This research entitled, "Mobile Synchronization Framework to Enhance Large Object Management in Mobile Cloud Storage Service" comprises of an extensive study. Hence, this research work is divided into chapters for reader understand ability.

CHAPTER 1 gives brief introduction on general architecture for mobile cloud computing environment. The challenges in Disconnected operation are discussed in detail along with the need for synchronization frameworks or Mobile Backend as a Service (MBaaS) frameworks for mobile apps to easily manage data. Research objectives, scope and contribution of this research are explained in this chapter.

CHAPTER 2 introduce the background concepts of Mobile Backend as a Service or synchronization frameworks, different Replication strategies to support fault resilience and Consistency models. This chapter also provide an overview of different data stores in distributed environments. Moreover, this chapter presents a review of data consistency and synchronization frameworks in Mobile Cloud Computing for Mobile Apps. Latest studies done from 2010 to 2018 are considered and classified into different types. Three reference implementations in the literature are considered in detail to investigate the approaches to handle consistency support, sync services, conflict handling and offline operations. The pros and cons of three reference implementations in the literature have been presented. This chapter also presents the 3D Design Framework considered for Consistency benchmarking of application frameworks. The report of evaluation parameters of the three reference implementations in the literature have been presented.

Furthermore, this chapter investigate and analyse the support for large files upload and retrieval in mobile data synchronization frameworks with cloud storage services.

CHAPTER 3 presents the first part of methodology applied in this research. It specifies the research design and categorize the process into four main phases as per the objectives of the research. The first phase of this research reviews data consistency and synchronization frameworks in Mobile Cloud Computing (MCC) for Mobile Apps. Next phase investigates and analyze the support for large files upload and retrieval in mobile data synchronization frameworks with cloud storage services. The third phase proposes a cloud- based framework to support End-to-end data consistency support for large data object access and is finally evaluated in the last phase of this research work.

CHAPTER 4 describes proposed enhanced cloud-based sync framework (called as NetMob), to support End-to-end data consistency support for large data object access. The architecture and design are described in detail along with techniques and methods that are followed in the design of proposed enhanced cloud-based sync framework. Different Client and Server-side modules are presented in detail with synchronization protocol. This chapter further provide details of proposed enhanced cloud-based sync framework data model and supported APIs. Large Object Support with Segmentation and Object Chunking both at the client and server side is described with implementation details. Finally, in this chapter the NetMob architecture is compared with two other open-source mobile sync frameworks.

CHAPTER 5 reports on the evaluation methodology for the proposed enhanced cloud-based sync framework. The experimental setup is described with included set of virtual machines and mobile device client. The data collection methodology for conducting tests the ability of proposed enhanced cloud-based sync framework to handle read and write requests is described. Methods to evaluate the performance of object chunking in proposed enhanced cloud-based sync framework is presented along with the mechanism to test the consistency parameters. Finally, this chapter also presents the results of experimental evaluation of the proposed sync framework to prove its significance and efficiency. The experimental evaluation is based on latency i.e., time taken to upload or download the files of different size under different setups. Moreover, the proposed sync framework is analysed on varying values of system parameters, such as different sizes of chunks (2 MiB, 4 MiB, 8 MiB and 16 MiB) and under different consistency configurations (single replica for strong consistency and three replicas for eventual consistency). Furthermore, the system model of proposed sync framework is validated with experimental data.

Finally, **CHAPTER 6** concludes this research work by re-visiting the research objectives. The chapter also provide future research directions of this research.

REFERENCES

A Fast and Lightweight Key/Value Database Library. (2017). https://github.com/google/leveldb

Agarwal, S., Mahajan, R., Zheng, A., & Bahl, V. (2010). Diagnosing mobile applications in the wild. *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 1–6.

Alapati, S. R. (2018). Cassandra data modeling, and the reading and writing of data. In *Expert apache cassandra administration* (pp. 99–148). Springer.

Alvaro, P., Conway, N., Hellerstein, J. M., & Marczak, W. R. (2011). Consistency Analysis in Bloom: A CALM and Collected Approach. *CIDR*, 249–260.

Amazon DynamoDB - Best Practices for Storing Large Items and Attributes. (2019). https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-use-s3-too.html

Apache CouchDB. (2018).

Apache Hadoop 3.2.1 – HDFS Architecture. (2017). https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

Archana Sharma, & Vineet Kansal. (2011). Replication Management and Optimistic Replication Challenges in Mobile Environment. *International Journal of Database Management Systems*, *3*(4), 81–99. https://doi.org/10.5121/ijdms.2011.3407

Bai, Y., & Zhang, Y. (2017). StoArranger: Enabling Efficient Usage of Cloud Storage Services on Mobile Devices. *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference On*, 1476–1487.

Balegas, V., Duarte, S., Ferreira, C., Rodrigues, R., Preguiça, N., Najafzadeh, M., & Shapiro, M. (2015). Putting consistency back into eventual consistency. *Proceedings of the Tenth European Conference on Computer Systems - EuroSys '15*, 1–16. https://doi.org/10.1145/2741948.2741972

Barbará, D. (1999). Mobile computing and databases-a survey. *IEEE Transactions on Knowledge and Data Engineering*, *11*(1), 108–117.

Belaramani, N. M., Dahlin, M., Gao, L., Nayate, A., Venkataramani, A., Yalagandula, P., & Zheng, J. (2006). PRACTI Replication. *NSDI*, *6*, 5.

Bermbach, D., Kuhlenkamp, J., Derre, B., Klems, M., & Tai, S. (2013). A Middleware Guaranteeing Client-Centric Consistency on Top of Eventually Consistent Datastores. *IC2E*, 114–123.

Bermbach, D., & Tai, S. (2014). Benchmarking eventual consistency: Lessons learned from long-term experimental studies. *2014 IEEE International Conference on Cloud Engineering*, 47–56.

Bermbach, D., & Tai, S. (2011). Eventual consistency: How soon is eventual? An evaluation of Amazon S3's consistency behavior. *Proceedings of the 6th Workshop on Middleware for Service Oriented Computing*, 1.

Bermbach, D., Wittern, E., & Tai, S. (2017). *Cloud service benchmarking*. Springer.

Bhajantri, L. B., & Ayyannavar, V. V. (2018). Cognitive Agent Based Data Synchronization in Ubiquitous Networks: A Survey. *International Journal of Advanced Pervasive and Ubiquitous Computing (IJAPUC)*, *10*(2), 1–17.

Brunette, W., Sudar, S., Sundt, M., Larson, C., Beorse, J., & Anderson, R. (2017). Open Data Kit 2.0: A Services-Based Application Framework for Disconnected Data Management. *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, 440–452.

Brzezinski, J., Sobaniec, C., & Wawrzyniak, D. (2004). From Session Causality to Causal Consistency. *PDP*, 152–158.

Brzezinski, J., Sobaniec, C., & Wawrzyniak, D. (2003). Session guarantees to achieve PRAM consistency of replicated shared objects. *International Conference on Parallel Processing and Applied Mathematics*, 1–8.

Burckhardt, S. (2013a). *Bringing TouchDevelop to the cloud*.

Burckhardt, S. (2013b). Bringing TouchDevelop to the cloud. *Inside Microsoft Research Blog, October*.

Burckhardt, S., Fähndrich, M., Leijen, D., & Wood, B. P. (2012). Cloud types for eventual consistency. *European Conference on Object-Oriented Programming*, 283–307.

Burckhardt, S., Gotsman, A., Yang, H., & Zawirski, M. (2014). Replicated data types: Specification, verification, optimality. *ACM SIGPLAN Notices*, *49*(1), 271–284.

Burckhardt, S., Leijen, D., Fähndrich, M., & Sagiv, M. (2012). Eventually consistent transactions. *European Symposium on Programming*, 67–86.

Cao, J., Zhang, Y., Cao, G., & Xie, L. (2007). Data consistency for cooperative caching in mobile environments. *Computer*.

Cattell, R. (2011). Scalable SQL and NoSQL data stores. *Acm Sigmod Record*, *39*(4), 12–27.

Chandra, S., Dahlin, M., Richards, B., Wang, R. Y., Anderson, T. E., & Larus, J. R. (1997). Experience with a language for writing coherence protocols. *Proceedings of the Conference on Domain-Specific Languages on Conference on Domain-Specific Languages (DSL), 1997*, 5.

Chandrashekhara, S., Ki, T., Jeon, K., Dantu, K., & Ko, S. Y. (2017). BlueMountain: An Architecture for Customized Data Management on Mobile Systems. *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, 396–408.

Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., & Gruber, R. E. (2008). Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, *26*(2), 4.

Chekam, T. T., Zhai, E., Li, Z., Cui, Y., & Ren, K. (2016). On the synchronization bottleneck of OpenStack Swift-like cloud storage systems. *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, 1–9.

Chun, B.-G., Curino, C., Sears, R., Shraer, A., Madden, S., & Ramakrishnan, R. (2012). Mobius: Unified messaging and data serving for mobile apps. *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, 141–154.

Conway, N., Marczak, W. R., Alvaro, P., Hellerstein, J. M., & Maier, D. (2012). Logic and lattices for distributed programming. *Proceedings of the Third ACM Symposium on Cloud Computing*, 1.

Cooper, B. F., Ramakrishnan, R., Srivastava, U., Silberstein, A., Bohannon, P., Jacobsen, H.-A., Puz, N., Weaver, D., & Yerneni, R. (2008). PNUTS: Yahoo!'s hosted data serving platform. *Proceedings of the VLDB Endowment*, *1*(2), 1277–1288.

Council, F. C. I. O. (2017). *MSCT*.

Cui, Y., Lai, Z., Wang, X., & Dai, N. (2017a). QuickSync: Improving synchronization efficiency for mobile cloud storage services. *IEEE Transactions on Mobile Computing*, *16*(12), 3513–3526.

Cui, Y., Lai, Z., Wang, X., & Dai, N. (2017b). QuickSync: Improving synchronization efficiency for mobile cloud storage services. *IEEE Transactions on Mobile Computing*, *16*(12), 3513–3526.

Curino, C., Jones, E., Zhang, Y., & Madden, S. (2010). Schism: A workload-driven approach to database replication and partitioning. *Proceedings of the VLDB Endowment*, *3*(1–2), 48–57.

Dabek, F., Kaashoek, M. F., Karger, D., Morris, R., & Stoica, I. (2001). Wide-area cooperative storage with CFS. *ACM SIGOPS Operating Systems Review*, *35*(5), 202–215.

Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, *51*(1), 107–113.

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., & Vogels, W. (2007a). Dynamo: Amazon's highly available key-value store. *ACM SIGOPS Operating Systems Review*, *41*(6), 205–220.

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., & Vogels, W. (2007b). Dynamo: Amazon's highly available key-value store. *ACM SIGOPS Operating Systems Review*, *41*(6), 205–220.

Definitions of the SI units: The binary prefixes. (2019, May 20). https://physics.nist.gov/cuu/Units/binary.html

Diogo, M., Cabral, B., & Bernardino, J. (2019). Consistency Models of NoSQL Databases. *Future Internet*, *11*(2), 43. https://doi.org/10.3390/fi11020043

Drago, I., Bocchi, E., Mellia, M., Slatman, H., & Pras, A. (2013). Benchmarking personal cloud storage. *Proceedings of the 2013 Conference on Internet Measurement Conference*, 205–212.

Drago, I., Mellia, M., M Munafo, M., Sperotto, A., Sadre, R., & Pras, A. (2012). Inside dropbox: Understanding personal cloud storage services. *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, 481–494.

Drive, G. (2016). *Google Drive*.

Dropbox. (2016). *Build your app on the Dropbox platform*.

Engberg, D. (2012). *WhySQL?*

Evernote system limits. (2017). https://help.evernote.com/hc/en-us/articles/209005247-Evernote-system-limits

Fathalla, D. (2019). *ORLease: Optimistically Replicated Lease Using Lease Version Vector For Higher Replica Consistency in Optimistic Replication Systems ORLease: Optimistically Replicated Lease Using Lease Version*. *1080*.

Firebase. (2017a). *Firebase*.

Firebase. (2017b). *Firebase*.

Garcia-Lopez, P., Sanchez-Artigas, M., Cotes, C., Guerrero, G., Moreno, A., & Toda, S. (2013). *StackSync: Architecturing the personal cloud to Be in sync*.

Gheith, A., Rajamony, R., Bohrer, P., Agarwal, K., Kistler, M., White Eagle, B. L., Hambridge, C. A., Carter, J. B., & Kaplinger, T. (2016). IBM Bluemix Mobile Cloud Services. *IBM Journal of Research and Development*, *60*(2–3), 7:1-7:12. https://doi.org/10.1147/JRD.2016.2515422

Gilbert, S., & Lynch, N. (2002). Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *ACM SIGACT News*, *33*(2), 51. https://doi.org/10.1145/564585.564601

Gray, J. N., Lorie, R. A., Putzolu, G. R., & Traiger, I. L. (1976). Granularity of locks and degrees of consistency in a shared data base. *IFIP Working Conference on Modelling in Data Base Management Systems*, 365–394.

Group, I. M. (2014). *Pick one: 5 clouds for building mobile apps*.

Guy, R. G., Heidemann, J. S., Mak, W.-K., Page Jr, T. W., Popek, G. J., Rothmeier, D., & others. (1990). Implementation of the Ficus Replicated File System. *USENIX Summer*, 63–72.

Guy, R. G., Heidemann, J. S., & Page Jr, T. W. (1992). The ficus replicated file system. *ACM SIGOPS Operating Systems Review*, *26*(2), 26.

Hao, S., Agrawal, N., Aranya, A., & Ungureanu, C. (2013a). Building a delay-tolerant cloud for mobile data. *Proceedings - IEEE International Conference on Mobile Data Management*, *1*, 293–300. https://doi.org/10.1109/MDM.2013.43

Hao, S., Agrawal, N., Aranya, A., & Ungureanu, C. (2013b). Building a Delay-Tolerant Cloud for Mobile Data. *2013 IEEE 14th International Conference on Mobile Data Management*, *1*, 293–300.

Huang, Y., Cao, J., Jin, B., Tao, X., Lu, J., & Feng, Y. (2010). Flexible cache consistency maintenance over wireless ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, *21*(8), 1150–1161.

Imam, A. A., Basri, S., & Ahmad, R. (2015). DATA SYNCHRONIZATION BETWEEN MOBILE DEVICES AND SERVER-SIDE DATABASES: A SYSTEMATIC LITERATURE REVIEW. *Journal of Theoretical and Applied Information Technology*, *81*(2), 364.

iOS app OTA limit in cellular network—Apple Community. (2016). https://discussions.apple.com/thread/7797088

Jing, J., Helal, A. S., & Elmagarmid, A. (1999). Client-server computing in mobile environments. *ACM Computing Surveys (CSUR)*, *31*(2), 117–157.

Joseph, A. D., de Lespinasse, A. F., Tauber, J. A., Gifford, D. K., & Kaashoek, M. F. (1995). Rover: A toolkit for mobile information access. *ACM SIGOPS Operating Systems Review*, *29*(5), 156–171.

Karger, D., Lehman, E., Leighton, T., Panigrahy, R., Levine, M., & Lewin, D. (1997). Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, 654–663.

Kemme, B., & Alonso, G. (2000). A new approach to developing and implementing eager database replication protocols. *ACM Transactions on Database Systems (TODS)*, *25*(3), 333–379.

Kermarrec, A.-M., Rowstron, A., Shapiro, M., & Druschel, P. (2001). The IceCube approach to the reconciliation of divergent replicas. *Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing*, 210–218.

Ketola, T. (2014). Quantifying software development: Applying mobile monetization techniques to your software development process. *2014 Computer Games: AI, Animation, Mobile, Multimedia, Educational and Serious Games (CGAMES)*, 1–4.

Kinvey. (2016). *Kinvey BaaS*.

Kistler, J. J., & Satyanarayanan, M. (1992). Disconnected operation in the Coda file system. *ACM Transactions on Computer Systems (TOCS)*, *10*(1), 3–25.

Klems, M., Bermbach, D., & Weinert, R. (2012). A runtime quality measurement framework for cloud database service systems. *Quality of Information and Communications Technology (QUATIC), 2012 Eighth International Conference on The*, 38–46.

Klophaus, R. (2010). Riak core: Building distributed applications without shared state. *ACM SIGPLAN Commercial Users of Functional Programming*, 14.

Kony MobileFabric. (2017). https://www.kony.com/resources/videos/kony-mobilefabrictm-overview/

Ladin, R., Liskov, B., Shrira, L., & Ghemawat, S. (1992). Providing high availability using lazy replication. *ACM Transactions on Computer Systems (TOCS)*, *10*(4), 360–391.

Lakshman, A., & Malik, P. (2009). Cassandra: Structured storage system on a p2p network. *Proceedings of the 28th ACM Symposium on Principles of Distributed Computing*, 5.

Lamport, L. (1979). How to make a multiprocessor computer that correctly executes multiprocess programs. *IEEE Transactions on Computers*, *100*(9), 690–691.

Li, C., Porto, D., Clement, A., Gehrke, J., Preguiça, N. M., & Rodrigues, R. (2012). Making Geo-Replicated Systems Fast as Possible, Consistent when Necessary. *OSDI*, *12*, 265–278.

Li, Z., Wang, X., Huang, N., Kaafar, M. A., Li, Z., Zhou, J., Xie, G., & Steenkiste, P. (2016). An empirical analysis of a large-scale mobile cloud storage service. *Proceedings of the 2016 Internet Measurement Conference*, 287–301.

Lloyd, W., Freedman, M. J., Kaminsky, M., & Andersen, D. G. (2011). Don't settle for eventual: Scalable causal consistency for wide-area storage with COPS. *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, 401–416.

Lloyd, W., Freedman, M. J., Kaminsky, M., & Andersen, D. G. (2013). Stronger Semantics for Low-Latency Geo-Replicated Storage. *NSDI*, *13*, 313–328.

Mahmood, T., Narayanan, S. P., Rao, S., Vijaykumar, T., & Thottethodi, M. (2016). Achieving Causal Consistency under Partial Replication for Geo-distributed Cloud Storage. *Department of Electrical and Computer Engineering Technical Reports*.

Malkhi, D., & Terry, D. (2005). Concise version vectors in WinFS. *International Symposium on Distributed Computing*, 339–353.

Mark van Seventer. (2012). *A generic approach to data synchronization for HTML5 apps*. VU University.

Mell, P., Grance, T., & others. (2011). *The NIST definition of cloud computing*.

Milani, B. A., & Navimipour, N. J. (2016). A comprehensive review of the data replication techniques in the cloud environments: Major trends and future directions. *Journal of Network and Computer Applications*, *64*, 229–238.

Minelli, R., & Lanza, M. (2013). Software Analytics for Mobile Applications—Insights & Lessons Learned. *2013 17th European Conference on Software Maintenance and Reengineering*, 144–153. https://doi.org/10.1109/CSMR.2013.24

Mordacchini, M., Ricci, L., Ferrucci, L., Albano, M., & Baraglia, R. (2010). Hivory: Range queries on hierarchical voronoi overlays. *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*, 1–10.

Muntz, D., & Honeyman, P. (1991). *Multi-level caching in distributed file systems*.

Muthitacharoen, A., Chen, B., & Mazieres, D. (2001a). A low-bandwidth network file system. *ACM SIGOPS Operating Systems Review*, *35*(5), 174–187.

Muthitacharoen, A., Chen, B., & Mazieres, D. (2001b). A low-bandwidth network file system. *ACM SIGOPS Operating Systems Review*, *35*, 174–187.

Muthitacharoen, A., Morris, R., Gil, T. M., & Chen, B. (2002). Ivy: A read/write peer-to-peer file system. *ACM SIGOPS Operating Systems Review*, *36*(SI), 31–44.

Nelson, M., Welch, B., & Ousterhout, J. (1987). *Caching in the Sprite network file system* (Vol. 21, Issue 5). ACM.

Nguyen, P. (2016). *Mobile Backend as a Service: The pros and cons of parse*.

Nightingale, E. B., & Flinn, J. (2004). Energy-Efficiency and Storage Flexibility in the Blue File System. *OSDI*, *4*, 363–378.

Noble, B. D., Satyanarayanan, M., Narayanan, D., Tilton, J. E., Flinn, J., & Walker, K. R. (1997). Agile application-aware adaptation for mobility. *ACM SIGOPS Operating Systems Review*, *31*(5), 276–287.

NSURLSessionConfiguration | Apple Developer Documentation. (2015). https://developer.apple.com/documentation/foundation/nsurlsessionconfiguration

OfflineSync-BaasBox: Implementation of offline synchronization feature for BaasBox. (2015). https://github.com/simlaudato/OfflineSync-BaasBox

ONeil, P., Cheng, E., Gawlick, D., & ONeil, E. (1996). The log-structured merge-tree (LSM-tree). *Acta Informatica*, *33*(4), 351–385.

OpenStack Swift Object Storage Service. (2018).

Oprea, A., & Reiter, M. (2006). On Consistency of Encrypted Files. *Proceedings of Distributed Computing: 20th International Symposium (DISC '06)*, 254–268.

Parker, D. S., Popek, G. J., Rudisin, G., Stoughton, A., Walker, B. J., Walton, E., Chow, J. M., Edwards, D., Kiser, S., & Kline, C. (1983). Detection of mutual inconsistency in distributed systems. *IEEE Transactions on Software Engineering*, *3*, 240–247.

Parse. (2018). *Parse Platform—The Complete Application Stack*. https://parseplatform.org/

Parse Case Study. (2014). https://aws.amazon.com/solutions/case-studies/parse/

Perkins, D., Agrawal, N., Aranya, A., Yu, C., Go, Y., Madhyastha, H. V, & Ungureanu, C. (2015). Simba: Tunable end-to-end data consistency for mobile apps. *Proceedings of the Tenth European Conference on Computer Systems*, 7.

Pitoura, E., & Samaras, G. (2012). *Data management for mobile computing* (Vol. 10). Springer Science & Business Media.

Popov, A., Proletarsky, A., Belov, S., & Sorokin, A. (2017). Fast Prototyping of the Internet of Things solutions with IBM Bluemix. *Proceedings of the 50th Hawaii International Conference on System Sciences*.

Preguiça, N. (2018). *Conflict-free Replicated Data Types: An Overview*. 1–41.

Preguica, N., Zawirski, M., Bieniusa, A., Duarte, S., Balegas, V., Baquero, C., & Shapiro, M. (2014). SwiftCloud: Fault-tolerant geo-replication integrated all the way to the client machine. *Proceedings of the IEEE Symposium on Reliable Distributed Systems*, 30–33. https://doi.org/10.1109/SRDSW.2014.33

Preguiça, N., Zawirski, M., Bieniusa, A., Duarte, S., Balegas, V., Baquero, C., & Shapiro, M. (2014). SwiftCloud: Fault-tolerant geo-replication integrated all the way to the client machine. *2014 IEEE 33rd International Symposium on Reliable Distributed Systems Workshops (SRDSW)*, 30–33.

Protocol Buffers. (2017). https://developers.google.com/protocol-buffers

Rabin, M. O. (1981). Fingerprinting by random polynomials. *Technical Report*.

Ramasubramanian, V., Rodeheffer, T. L., Terry, D. B., Walraed-Sullivan, M., Wobber, T., Marshall, C. C., & Vahdat, A. (2009). Cimbiosys: A platform for content-based partial replication. *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, 261–276.

Ravindranath, L., Nath, S., Padhye, J., & Balakrishnan, H. (2014). Automatic and scalable fault detection for mobile applications. *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, 190–203.

Raz, Y. (1993). Extended commitment ordering, or guaranteeing global serializability by applying commitment order selectively to global transactions. *Proceedings of the Twelfth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 83–96.

Reduce your app size. (2016). https://developer.android.com/topic/performance/reduce-apk-size

Ren, K., & Gibson, G. (2012). TABLEFS: Embedding a NoSQL database inside the local file system. *APMRC, 2012 Digest*, 1–6.

Research & Drafts | SPDY | Google Developers. (2018). https://developers.google.com/speed/protocols

Rowstron, A., & Druschel, P. (2001). Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. *ACM SIGOPS Operating Systems Review*, *35*(5), 188–201.

Rupprecht, L., Zhang, R., Owen, B., Pietzuch, P., & Hildebrand, D. (2017). SwiftAnalytics: Optimizing object storage for big data analytics. *Proceedings - 2017 IEEE International Conference on Cloud Engineering, IC2E 2017*, 245–251. https://doi.org/10.1109/IC2E.2017.19

Saito, Y., Karamanolis, C., Karlsson, M., & Mahalingam, M. (2002). Taming aggressive replication in the Pangaea wide-area file system. *ACM SIGOPS Operating Systems Review*, *36*(SI), 15–30.

Saito, Y., & Shapiro, M. (2005). Optimistic replication. *ACM Computing Surveys (CSUR)*, *37*(1), 42–81.

Salmon, B., Schlosser, S. W., Cranor, L. F., & Ganger, G. R. (2009). Perspective: Semantic Data Management for the Home. *FAST*, *9*, 167–182.

Satyanarayanan, M. (1996). Fundamental Challenges in Mobile Computing. *Annual ACM Symposium on Principles of Distributed Computing*, 1–7. https://doi.org/10.1145/248052.248053

Serrano-Alvarado, P., Roncancio, C., & Adiba, M. (2004). A survey of mobile transactions. *Distributed and Parallel Databases*, *16*(2), 193–230.

Shanon Montelongo, F. (2019). *How to upload large files*. https://blog.filestack.com/thoughts-and-knowledge/how-to-upload-large-files/

Shapiro, M., Preguiça, N., Baquero, C., & Zawirski, M. (2011a). *A comprehensive study of convergent and commutative replicated data types*. Inria--Centre Paris-Rocquencourt; INRIA.

Shapiro, M., Preguiça, N., Baquero, C., & Zawirski, M. (2011b). Conflict-free replicated data types. *Symposium on Self-Stabilizing Systems*, 386–400.

Sharma, A., & Kansal, V. (2011). Replication management and optimistic replication challenges in mobile environment. *International Journal of Database Management Systems*, *3*(4), 81.

Shetty, P., Spillane, R. P., Malpani, R., Andrews, B., Seyster, J., & Zadok, E. (2013). Building workload-independent storage with VT-trees. *FAST*, 17–30.

Shraer, A., Aybes, A., Davis, B., Chrysafis, C., Browning, D., Krugler, E., Stone, E., Chandler, H., Farkas, J., Quinn, J., & Others. (2018). Cloudkit: Structured storage for mobile applications. *Proceedings of the VLDB Endowment*, *11*(5), 540–552.

Singh, N., & Hasan, M. (2019, December 1). Efficient method for data synchronization in mobile database. *2019 IEEE Conference on Information and Communication Technology, CICT 2019*. https://doi.org/10.1109/CICT48419.2019.9066122

Sovran, Y., Power, R., Aguilera, M. K., & Li, J. (2011). Transactional storage for geo-replicated systems. *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, 385–400.

Soyata, T., Ba, H., Heinzelman, W., Kwon, M., & Shi, J. (2015). Accelerating mobile-cloud computing: A survey. In *Cloud Technology: Concepts, Methodologies, Tools, and Applications* (pp. 1933–1955). IGI Global.

Spahn, R., Bell, J., Lee, M., Bhamidipati, S., Geambasu, R., & Kaiser, G. E. (2014). Pebbles: Fine-Grained Data Management Abstractions for Modern Operating Systems. *OSDI*, 113–129.

Tanenbaum, A. S., & van Steen, M. (2002). Principles and Paradigms. *Distributed Systems.*

Tanenbaum, A. S., & Van Steen, M. (2007). *Distributed systems: Principles and paradigms*. Prentice-Hall.

Tang, H., Liu, F., Shen, G., Jin, Y., & Guo, C. (2015). UniDrive: Synergize multiple consumer cloud storage services. *Proceedings of the 16th Annual Middleware Conference*, 137–148.

Tatarowicz, A. L., Curino, C., Jones, E. P. C., & Madden, S. (2012). Lookup tables: Fine-grained partitioning for distributed databases. *2012 IEEE 28th International Conference on Data Engineering*, 102–113.

Terry, D. B., Demers, A. J., Petersen, K., Spreitzer, M. J., Theimer, M. M., & Welch, B. B. (1994). Session guarantees for weakly consistent replicated data. *Parallel and Distributed Information Systems, 1994., Proceedings of the Third International Conference On*, 140–149.

Terry, D. B., Prabhakaran, V., Kotla, R., Balakrishnan, M., Aguilera, M. K., & Abu-Libdeh, H. (2013a). Consistency-based service level agreements for cloud storage. *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles - SOSP '13*, 309–324. https://doi.org/10.1145/2517349.2522731

Terry, D. B., Prabhakaran, V., Kotla, R., Balakrishnan, M., Aguilera, M. K., & Abu-Libdeh, H. (2013b). Consistency-based service level agreements for cloud storage. *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, 309–324.

Terry, D. B., Prabhakaran, V., Kotla, R., Balakrishnan, M., Aguilera, M. K., & Abu-Libdeh, H. (2013c). Consistency-based service level agreements for cloud storage. *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, 309–324.

Terry, D. B., Theimer, M. M., Petersen, K., Demers, A. J., Spreitzer, M. J., & Hauser, C. H. (1995a). *Managing update conflicts in Bayou, a weakly connected replicated storage system* (Vol. 29, Issue 5). ACM.

Terry, D. B., Theimer, M. M., Petersen, K., Demers, A. J., Spreitzer, M. J., & Hauser, C. H. (1995b). Managing update conflicts in Bayou, a weakly connected replicated storage system. *ACM SIGOPS Operating Systems Review*, *29*(5), 172–182.

Tewari, R., Dahlin, M., Vin, H. M., & Kay, J. S. (1999). Design considerations for distributed caching on the Internet. *Distributed Computing Systems, 1999. Proceedings. 19th IEEE International Conference On*, 273–284.

The BaasBox Server. (2019). https://github.com/baasbox/baasbox

Tolia, N., Harkes, J., Kozuch, M., & Satyanarayanan, M. (2004). Integrating Portable and Distributed Storage. *FAST*, *4*, 227–238.

Tolia, N., Satyanarayanan, M., & Wolbach, A. (2007). Improving mobile database access over wide-area networks without degrading consistency. *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services*, 71–84.

TouchDB. (2018).

Unhelkar, B., & Murugesan, S. (2010). The enterprise mobile applications development framework. *IT Professional*, *12*(3), 33–39.

van Renesse, R., Dumitriu, D., Gough, V., & Thomas, C. (2008). Efficient reconciliation and flow control for anti-entropy protocols. *Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware - LADIS '08*, 1. https://doi.org/10.1145/1529974.1529983

Vogels, W. (2009). Eventually consistent. *Communications of the ACM*, *52*(1), 40–44.

Wada, H., Fekete, A., Zhao, L., Lee, K., & Liu, A. (2011). Data Consistency Properties and the Trade-offs in Commercial Cloud Storage: The Consumers' Perspective. *CIDR*, *11*, 134–143.

Wang, D., Joshi, G., & Wornell, G. W. (2019). Efficient straggler replication in large-scale parallel computing. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, *4*(2). https://doi.org/10.1145/3310336

Wu, H., Hamdi, L., & Mahe, N. (2010a). TANGO: A flexible mobility-enabled architecture for online and offline mobile enterprise applications. *Proceedings - IEEE International Conference on Mobile Data Management*, 230–238. https://doi.org/10.1109/MDM.2010.58

Wu, H., Hamdi, L., & Mahe, N. (2010b). TANGO: A flexible mobility-enabled architecture for online and offline mobile enterprise applications. *Proceedings - IEEE International Conference on Mobile Data Management*, 230–238. https://doi.org/10.1109/MDM.2010.58

Wu, Z., Butkiewicz, M., Perkins, D., Katz-Bassett, E., & Madhyastha, H. V. (2013). Spanstore: Cost-effective geo-replicated storage spanning multiple cloud services. *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, 292–308.

Wuu, G. T. J., & Bernstein, A. J. (1984). Efficient solutions to the replicated log and dictionary problems. *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing*, 233–242.

Xiao, H. (2017). Practical web-based delta synchronization for cloud storage services. *HotStorage*.

Xue, Y. (2008). The research on data synchronization of distributed real-time mobile network. *Computer Science and Software Engineering, 2008 International Conference On*, *3*, 1104–1107.

Yu, H., & Vahdat, A. (2002). Design and evaluation of a conit-based continuous consistency model for replicated services. *ACM Transactions on Computer Systems (TOCS)*, *20*(3), 239–282.

Zhang, I., Szekeres, A., Van Aken, D., Ackerman, I., Gribble, S. D., Krishnamurthy, A., & Levy, H. M. (2014). Customizable and extensible deployment for mobile/cloud applications. *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, 97–112.

Zhang, Y., Power, R., Zhou, S., Sovran, Y., Aguilera, M. K., & Li, J. (2013). Transaction chains: Achieving serializability with low latency in geo-distributed storage systems. *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, 276–291.

Zhang, Y., Tan, C., & Qun, L. (2013). CacheKeeper: A system-wide web caching service for smartphones. *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 265–274.