Elsevier Editorial System(tm) for Journal of Network and Computer Applications
Manuscript Draft

Manuscript Number: JNCA-D-13-00551

Title: Leveraging Client-side Storage Techniques for Enhanced Use of Multiple Consumer Cloud Storage Services on Resource-Constrained Mobile Devices

Article Type: Regular Article

Keywords: Multiple cloud storage, mobile devices, erasure coding, fault-tolerance, storage techniques

Corresponding Author: Prof. Hyotaek Lim, Ph.D

Corresponding Author's Institution: Dongseo University

First Author: Hui-Shyong Yeo, M.D

Order of Authors: Hui-Shyong Yeo, M.D; Hyotaek Lim, Prof

Abstract: Consumers increasingly desire to share content and expect to have continuous access to their data using multiple computing devices such as smartphones and tablets. As a result, consumer cloud storage service has become a trend in ubiquitous data access, and it has revolutionized the way users access their personal data. There are many cloud storage services available in the marketplace including paid and free subscriptions. However, the former is costly while the latter possesses many limitations such as low storage capacity, limited features and unsatisfactory performance. In addition, there are several security and privacy concerns related to cloud storage services that are often overlooked by consumers such as vendor lock-in issue, frequent service outages, data corruption and government subpoena. In this paper, we propose a solution that unifies storage from multiple cloud providers into a centralized storage pool that is better in terms of availability, capacity, performance, reliability and security. First, we study the feasibility of applying several storage technologies in resource-constrained mobile devices to address the limitations and issues related to cloud storage usage model. Then, we validate our proposed solution over typical single cloud storage with a working prototype implementation. Our results show that how it can improve the usage of consumer cloud storage at zero monetary cost while the minor overheads imposed on the client-side are actually compensated by the performance gained.

Cover Letter

Leveraging Client-side Storage Techniques for Enhanced Use of Multiple Consumer Cloud Storage Services on Resource-Constrained Mobile Devices

Hui-Shyong Yeo

Department of Ubiquitous IT, Dongseo University, 617-716 Busan, South Korea

Email: hsyeo@dongseo.ac.kr

Hyotaek Lim (Corresponding author)

Division of Computer and Information Engineering, Dongseo University, 617-716 Busan, South Korea

Email: htlim@dongseo.ac.kr

Phone: +8251-320-1718

Abstract: Consumers increasingly desire to share content and expect to have continuous access to their data using multiple computing devices such as smartphones and tablets. As a result, consumer cloud storage service has become a trend in ubiquitous data access, and it has revolutionized the way users access their personal data. There are many cloud storage services available in the marketplace including paid and free subscriptions. However, the former is costly while the latter possesses many limitations such as low storage capacity, limited features and unsatisfactory performance. In addition, there are several security and privacy concerns related to cloud storage services that are often overlooked by consumers such as vendor lock-in issue, frequent service outages, data corruption and government subpoena. In this paper, we propose a solution that unifies storage from multiple cloud providers into a centralized storage pool that is better in terms of availability, capacity, performance, reliability and security. First, we study the feasibility of applying several storage technologies in resource-constrained mobile devices to address the limitations and issues related to cloud storage usage model. Then, we validate our proposed solution over typical single cloud storage with a working prototype implementation. Our results show that how it can improve the usage of consumer cloud storage at zero monetary cost while the minor overheads imposed on the client-side are actually compensated by the performance gained.

Keywords: Multiple cloud storage, mobile devices, erasure coding, fault-tolerance, storage techniques

***Suggested List of Potential Referees**

Mohammed Atiquzzaman, School of Computer Science, University of Oklahoma

Jiang Bian, Division of Biomedical Informatics, Unversity of Arkansas for Medical Sciences

Josef Spillner, Computer Networks, Dresden University of Technology.

Leveraging Client-side Storage Techniques for Enhanced Use of Multiple Consumer Cloud Storage Services on Resource-Constrained Mobile Devices

Hui-Shyong Yeo

Department of Ubiquitous IT, Dongseo University, 617-716 Busan, South Korea

Email: hsyeo@dongseo.ac.kr

Hyotaek Lim (Corresponding author)

Division of Computer and Information Engineering, Dongseo University, 617-716 Busan, South Korea

Email: htlim@dongseo.ac.kr

Phone: +8251-320-1718

Abstract: Consumers increasingly desire to share content and expect to have continuous access to their data using multiple computing devices such as smartphones and tablets. As a result, consumer cloud storage service has become a trend in ubiquitous data access, and it has revolutionized the way users access their personal data. There are many cloud storage services available in the marketplace including paid and free subscriptions. However, the former is costly while the latter possesses many limitations such as low storage capacity, limited features and unsatisfactory performance. In addition, there are several security and privacy concerns related to cloud storage services that are often overlooked by consumers such as vendor lock-in issue, frequent service outages, data corruption and government subpoena. In this paper, we propose a solution that unifies storage from multiple cloud providers into a centralized storage pool that is better in terms of availability, capacity, performance, reliability and security. First, we study the feasibility of applying several storage technologies in resource-constrained mobile devices to address the limitations and issues related to cloud storage usage model. Then, we validate our proposed solution over typical single cloud storage with a working prototype implementation. Our results show that how it can improve the usage of consumer cloud storage at zero monetary cost while the minor overheads imposed on the client-side are actually compensated by the performance gained.

Keywords: Multiple cloud storage, mobile devices, erasure coding, fault-tolerance, storage techniques

1. Introduction

Cloud storage is a variety of cloud computing flavor [1], specifically known as Data as a service (DaaS). It provides online storage where data is stored in virtualized pools of storage hosted by third parties, usually span over large data centers in different geographical locations. In short, it is simply an improved online storage service that allows users to access it anywhere, anytime and using any device, through the internet. It simplifies how users access their personal data and eliminates the need to carry along external storage device all the time, such as USB flash drive or SD card. It also became a viable backup solution for users to store their precious personal data in the cloud. Cloud

storage provides some key features that are extremely useful to users, such as file versioning, synchronization between devices, file sharing and collaboration.

Average storage per household is expected to grow from 464 Gigabytes (GB) in 2011 to 3.3 Terabytes (TB) in 2016 [2], mainly due to the high adoption rate of mobile devices equipped with high quality camera and hard disk drives with large capacity among general consumers. Media quality has improved tremendously over the past decade, and today's consumers are craving for High Definition (HD) content such as FULL HD (1080p) and even UHD (4K) resolution. These contents require large storage capacity, and therefore are usually stored on cheap priced hard disk drives. However, shortage of hard disk supplies resulting from Thailand's floods [3] had caused sudden price hikes on the storage markets and hence it provides an impetus for cloud storage adoption among consumers. In addition, the growth of smart mobile devices also indirectly pushed the adoption of cloud storage services. It is because internal storage capacity that comes along with the device is usually very limited (16-32GB). In addition, external secure digital (SD) card expansion is not supported by all variant of mobile devices such as iPhone and certain Android phones. Therefore, cloud storage could be a great alternative to extend storage capacity on these mobile devices. It allows ubiquitous data access anytime, anywhere and it satisfied the user's desire of accessing data and share contents on multiple devices. Gartner [4] predicted consumers would store 36 percent of their digital content in the cloud by 2016, compared to a mere 7 percent during 2011.

Table 1. Comparison between several most popular consumer cloud storage services available in current market.

	Dropbox	**Google Drive**	**Box**	**SkyDrive**[1]
Free quota	2GB	5GB	5GB	7GB
Monthly subscription fee[2] in U.S. dollar ($)	100GB $9.99 200GB $19.99 500GB $49.99	25GB $2.49 100GB $4.99 200GB $9.99 400GB $19.99 1000GB $49.99	25GB $9.99 50GB $19.99	20GB $10[1] 50GB $25[1] 100GB $100[1]
Monthly cost per GB	$0.1	$0.1 (25GB) $0.05 (Others)	$0.4	$0.0416
Media streaming on mobile app	Yes	No	No	No
File size limitation	No limit	10GB	250MB	2GB
File versioning	30 days	30 days	No	25 days

Many cloud storage service providers offer free and paid subscription to the mass. The most popular choices of provider among consumers are Dropbox, Google Drive, Box and SkyDrive [5]. The offered services vary from one to another in terms of pricing, extra features and performance. A simple comparison in Table 1. would help us to

[1] SkyDrive pricing is based on annual fee, which is converted to $0.83, $2.08 and $4.16 per month, respectively.

[2] Pricing information is for consumer plans only. Business plans are omitted.

visualize the main differences. We noticed that most of the available cloud storage services offered a very limited free storage capacity. The pricing for paid subscription is also relatively expensive [6] because the subscription fee is charged monthly or annually. It means the total cost of ownership (TCO) [7] keeps increasing overtime as compared to one-time investment of purchasing local storage devices. There are also limitations such as limited file upload size, lack of media streaming support and slow performance [1]. In addition, there are several non-performance related issues that cannot be directly inferred from the table such as vendor lock-in issue [1,8], frequent service outages [1,22], data corruption [19,22] and government subpoena [20,21].

To address these limitations, we study the feasibility of leveraging client-side efforts to apply several storage technologies in cloud storage system. In particular, we propose to use simple storage techniques and exploit services from multiple providers to improve the usage model of cloud storage in current ecosystem. In this paper, we demonstrate through a detailed experimental study that our solution can achieve extra availability, capacity, reliability, security and performance at zero monetary cost. The key insight of this paper is that those limitations of individual cloud storage can be easily overcome by applying storage techniques on the client-side. Nevertheless, we restrict the scope of our study on consumer cloud storage services and mobile devices due to the fact that sales of mobile device have topped PC [9] and may be replacing the conventional desktop in the near future. In addition, several interesting challenges are raised when applying such computational intensive storage techniques on mobile devices due of its constrained nature. In summary, this paper makes the following contributions:

- We study the limitations and issues exist in current consumer cloud storage ecosystem and the feasibility of applying several client-side storage techniques to address these limitations.

- We propose and design the implementation of a middleware application for efficiently using of multiple cloud storage services on resource-constrained mobile devices such as smartphones or tablets.

- By prototyping on real hardware device and evaluating the performance in real world scenario, we show that our proposed solution can significantly improve the usage of consumer cloud storage services at zero cost plus minor overheads.

The remaining of this paper is structured as follows. Section 2 outlines current limitations and issues related to consumer cloud storage services. Section 3 provides the design and architecture of our proposed solution. In Section 4, we present the prototype system and some implementation considerations. In Section 5, we evaluate the performance alongside with discussions. Section 6 gives an overview of related work. Finally, in section 7 we conclude our paper and discuss on how this research can be further explored.

2. Background

2.1 The Main Problems

2.1.1 Limited Storage Capacity

In today's cloud storage marketplace, most service providers offer limited free storage capacity ranging from 2 to 7 Gigabytes. This amount of storage is barely enough to satisfy users as it is merely enough to store a couple of small files such as documents, pictures and low-resolution video files. Recent advancements in mobile device technology have been tremendous especially on the camera and screen quality [10]. 8 or 13 Megapixels (MP) camera lens and Full HD (1920*1080) or WQHD (2560*1440) screens have become the norm on high-end devices. It allows high definition media content to be created and presented to users ubiquitously. These contents usually consume large storage spaces, whereas the limited free storage capacity offered by cloud storage is unquestionably not enough.

2.1.2 Expensive Pricing

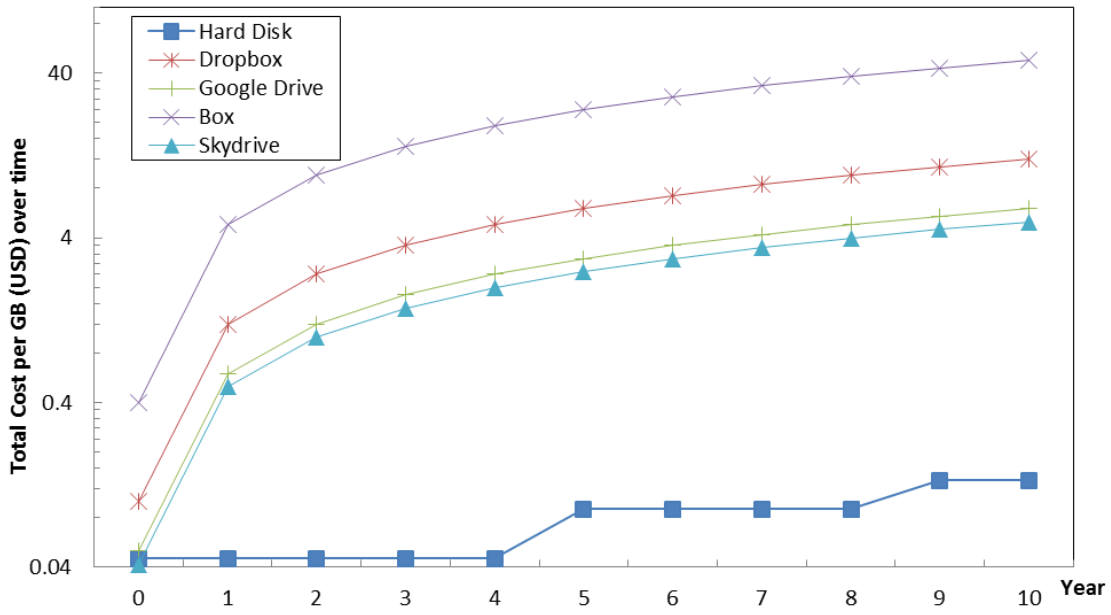


Figure 1. Total cost of ownership of local storage vs. different cloud storage services.

Paid subscriptions offer much higher storage capacity, but the cost is expensive and can be a burden to typical consumers. It is because of the pay per use model where as long as users are subscribed to cloud storage service, they need to pay a monthly or annually fee. Figure 1 visualizes the total cost of ownership (TCO) per GB over time compared to local storage devices. For comparison, a 4 Terabytes hard disk drive is priced at $180 today, which converts to $0.045 per GB. Assuming a standard 4 years lifespan; it costs $0.0009375 per GB each month[3], which is much lower than the cost of subscribing to cloud storage services ($0.05~$0.1 per GB each month). It reveals that while the entry cost into cloud storage services is relatively cheap, it is actually terribly expensive in the long term [6-8]. Besides, typical consumers are also reluctant to pay and prefer the available free services [11], even though the paid services offer extra storage capacity and advanced features. According to Forbes [12], only 4% of Dropbox's users are on paid subscriptions while the rest are on free subscriptions. As a result, some users have opted for an alternative solution, which is sign up for multiple free cloud storage services offered by different providers

[3] Neglecting electricity cost

(figure 2). By doing this, they can instantly boost the available storage capacity, and it appears to be a free and ingenious solution. However, as users' data collection grows larger over time, their files start to scatter in the different cloud storage, and eventually users will have a hard time in finding their files. They would not remember exactly in which cloud they had stored a particular file. In the worst-case scenario, user needs to access each of the cloud storage one by one to search for it, like searching for a needle in the haystack. The situation worsens when the user needs to install and run different software to access each of the cloud storage from different providers. This can be extremely frustrated and the hassle may not worth a few saved pennies.

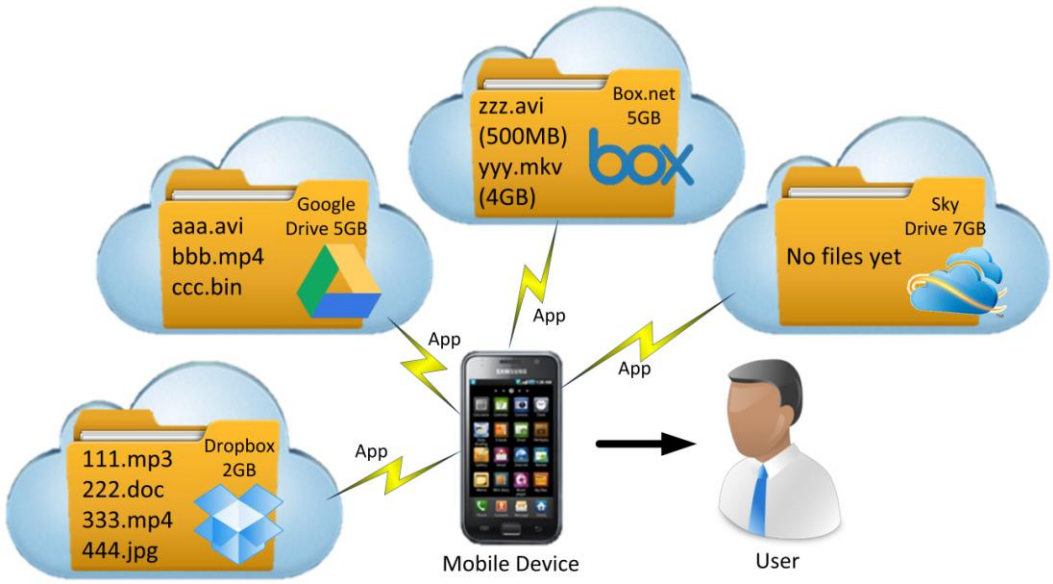


Figure 2. Example of typical consumer usage pattern on multiple cloud storage services.

2.1.3 Alternative Solution

There are several third party services that came into rescue, such as StorageMadeEasy, Otixo, Primadesk, Joukuu and Gladinet [13]. However, all these services merely provide a single interface to access different cloud storage, but do not actually merge them into one centralized storage pool. It can simplify the management of multiple cloud services, but the unorganized files are still scattered across different cloud locations. Besides, these services also require user sign up, which means user needs to tie/bound to another service provider and its terms and conditions. These services store user information and authorization token of each cloud storage that the user has granted access for, which means they have access to user's data in the cloud, and it may raise privacy concerns. Moreover, network connection is routed to their server (similar to proxy) instead of direct connection to cloud storage services, which will raise other issues like performance bottleneck, service outage, and man-in-the-middle security concern. Finally yet importantly, those services cost money or only provide limited bandwidth (250MB) for trial. In short, these services did not solve any actual problem but incur extra cost upon the users.

2.2 Features and Performance Related Problems

2.2.1 Low Performance and Bandwidth Utilization

It is an inevitable fact that I/O performance of using cloud storage is much slower compared to local storage, mainly due to the latency and throughput in accessing data stored in the cloud servers via network. In most cases, the data throughput is not limited by user's bandwidth but is rather bottlenecked on the service provider's side or the slowest link between them. As cloud storage service is generally offered to a vast amount of public users, the servers need to handle many users at the same time. The available bandwidth is shared with multiple users and therefore sluggish performance can be expected, especially on peak hours of the day. To ensure steady operation and quality of service (QoS) of their services, some providers perform bandwidth control and throttling[4] to allocate limited bandwidth for each user effectively. The throttling can be based on per-account or per-connection depends on the defined policy by providers. As a result, user's network bandwidth is being wasted and not fully utilized. Slow data throughput and short traffic bursts result in long waiting time and idle periods [14-16], during which a device keeps the radio channel occupied. This data access pattern is undesired because it will further results in low efficiency of radio resource and energy usage, causing battery power wastage. The power issue is not a concern in the desktop environment, but it is particularly crucial in the context of mobile devices due to the limited battery capacity. In contrast, high data throughput allows download/upload operations to complete earlier, which led to reduction of the "tail time" and allow the device switch its wireless radio into idle state as soon as possible. In idle state, power consumption is approaching zero and is negligible compared to keeping the wireless radio constantly in low power or full power state, as shown in figure 3. In short, it is better to transmit the data over a short period and put the radio into sleep instead of transmitting the equal data over a longer period.

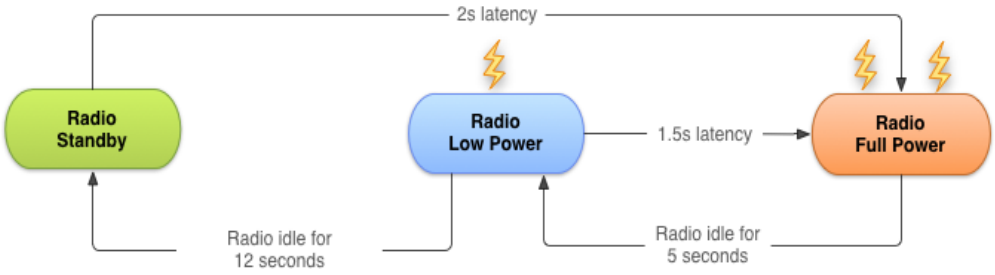


Figure 3. Typical 3G wireless radio state machine. (extracted from Android developers [14])

2.2.2 Limited Features

Service providers usually introduce extra features into their service in order to distinguish and compete with each other, but those premium features are only available to paid subscription users. Studies [11,12] show that most consumers prefer free services over the paid services. Besides, there are many functional limitations in current cloud storage services such as limited file upload size, lack of download acceleration and lack of media streaming support. Those features are key factors for achieving seamless data access especially on mobile devices. For example, users

[4] We observed bandwidth throttling on Dropbox and SkyDrive during our evaluation.

want to play a media content instantly instead of fully download the whole content first because the latter is both time and bandwidth consuming. While some of the providers such as Dropbox and SugarSync do provide media streaming on their mobile application, only a few formats are supported. In addition, the official mobile application provided by service providers only allows one concurrent download/upload operation at one time. It is seriously bottlenecking the whole system, causing low bandwidth utilization and high-energy wastage.

2.3 Non-Performance Related Issues

2.3.1 Vendor Lock-in Issue

Depends solely on a single cloud provider has risk of experiencing vendor lock-in [1,8]: users are trapped with the particular provider and it can be prohibitively expensive for them to switch provider, in terms of monetary, time and bandwidth cost. The longer a user is "trapped" with the provider, the higher the switching cost because more data needs to be migrated. This phenomenon is commonly known as "data inertia". Furthermore, data migration requires two ways operations (download and re-upload), which results in doubling the initial cost. Hence, users are vulnerable to price hikes or new pricing terms (i.e. service provider no longer offering free subscription, introduction of extra charges for bandwidth usage). Users are also subjects to the possibility of data loss if the provider goes out of business suddenly[5]. When new providers enter the cloud storage marketplace and offer better pricing terms or features, users may tempted to switch but they are probably held back by the high switching cost that might be imposed on them.

2.3.2 Frequent Service Outages

Despite guaranteed high availability and uptime (99.9%) in Service-Level Agreement (SLA), there are many cases where public cloud services encountered occasional service outages[6]. It may due to human error, hardware failure or natural disaster. In such cases, depending solely on a single provider results in a single point of failure and therefore users are not able access their data. Outages can result in severe financial losses or other kind of losses on both the provider and consumer. The damage caused by such events can vary among different type of users.

2.3.3 Security Breaches

Many providers encountered security breaches in the past[7]. Since most of the providers maintain encryption key by themselves, all users' data will be in risk if the key is compromised. The security scheme and its level of security occupied by the service provider are also not well understood and controllable by users. Recent security breaches revealed that even tech giants are using inappropriate security scheme. For example, Microsoft Store India stored user password in plaintext [17] while LinkedIn stored user password as hash without cryptographic salt [18].

---

[5] Federal prosecutors have shut down Megaupload file sharing services.

[6] Amazon AWS outages (April 2011, June, October 2012), Dropbox service disruptions (August, October 2012).

[7] Dropbox accounts were publicly accessible for several hours (June 2011), Dropbox employee's account password was stolen and spam messages had been send to users (August 2012).

2.3.4 Data Loss and Data Corruption

Despite service providers claimed to be using redundancy protection scheme such as RAID, total data losses have occurred in the past[8]. Most of the providers also do not responsible and take liability for any data loss, as stated in their terms and service [19]. Users are always responsible for backing up their data into several locations by replication or erasure coding.

2.3.5 Privacy Concern

Privacy concern is a serious issue and is used as the main marketing point of some client-side encryption based cloud storage providers such as SpiderOak and Wuala. It is also commonly known as "Zero-Knowledge Policy" [20]. For example, Twitter had been issued a subpoena by United States government on December 2010 [21]. Twitter appealed successfully and was able to disclose its existence to its users. However, there may be many similar incidents happened but did not disclose to the public such as the existence of PRISM[9] program that allows the government to spy on users' data directly. Since providers maintain the encryption key by themselves, both the provider and government can access users' data without any difficulties.

3. Proposed Architecture Overview

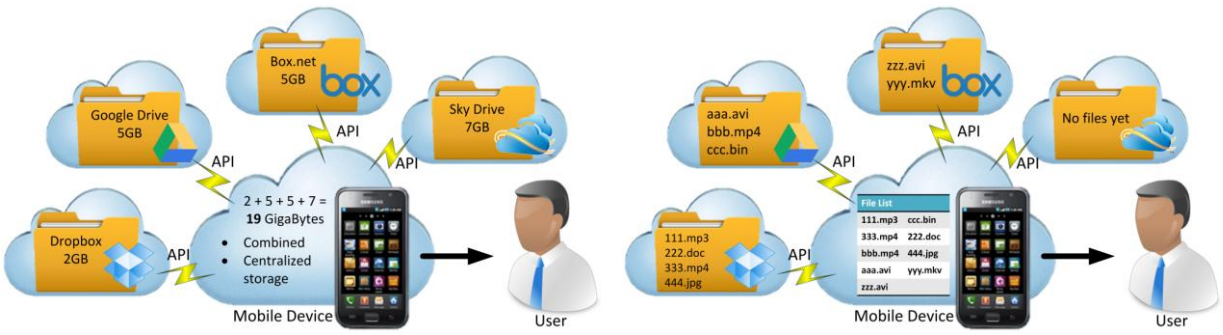
In this section, we will discuss about our proposed architecture and each of the storage techniques involved. We describe each technique and the incentives of using such techniques in our system, and then we describe the actual system design. In this paper, we aims to tackle all the limitations and issues outlined in Section 2. Our perspective is similar to a typical consumer, and our main objective is to improve the usage of cloud storage services at zero cost, whether it is paid or free subscription. We can achieve this by aggregating cloud storage from multiple providers (figure 4(a)) and apply several essential storage technologies into the system. Techniques such as data striping, erasure coding, encryption, compression, caching and data de-duplication will be applied on the client-side while the cloud storage is treated as just bunch of disks (JBOD). In short, our proposed solution can be described as "poor man's solution", "don't put all eggs in one basket" and "if you want a thing done well, do it yourself".

As mentioned in Section 2.1, most of the available multi cloud services require sign up at their website. In contrast, our proposed solution is designed as a portable system that can run on any compatible device and does not require any sign up on third party website. It means users do not need to tie/bound to another service provider. The authentication process is performed directly with each of the cloud storage provider using OAuth2 protocol, which means our system do not keep track of any user's id and password. Our system only stores the OAuth2 tokens that can be revoked anytime, which means it is less vulnerable to attack. The authorization process is a one-time only process that only needs to be performed once by users during installation.

---

[8] Microsoft Sidekick and Amazon EC2 service lost unrecoverable customer data in widely publicized incidents.

[9] PRISM is a surveillance program operated by United States National Security Agency (NSA).



(a) (b)

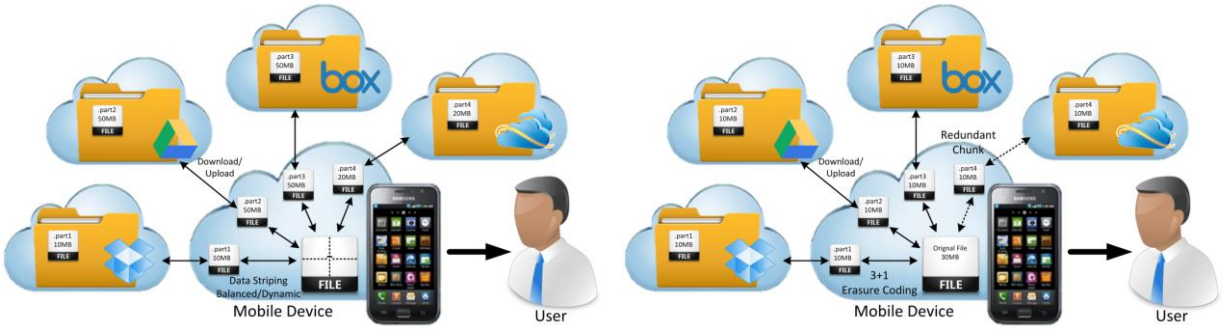
Figure 4. (a) Aggregating multiple cloud storage services. (b) A single list view of all files in all services.

3.1 Unified Cloud Storage

To achieve unified cloud storage and seamless data access, we need to provide users with a high level of abstraction where they are dealing with a single storage pool (figure 4(a)) and can easily access or organize their entire files collection across multiple providers (figure 4(b)). Hence, single point of contact, centralized management and intuitive user interface are critical factors in designing the architecture of our system. All files in each of the cloud storage should be accessible via a single view instead of different views as in current available multi cloud services. To achieve this, all files' attributes and metadata are fetched from each of the cloud storage in parallel using the specific provider's API, and then combined under a single namespace, and finally presented to user in a single list view (figure 4(b)). Additionally, users can perform simple file manipulation operations on any of the files such as Create, Rename, Update and Delete (CRUD). Searching, filtering and sorting can also be performed on all these files even though they are actually stored in different cloud storage.

3.2 Data Striping and Merging

As discussed in Section 2.2.2, most of the available consumer cloud storage services limit the maximum upload size. To bypass this limitation, we apply data striping (figure 5(a)) and upload each data chunk to the same provider or different providers using Round-Robin selection. Another advantageous side effect of data striping is vendor lock-in issue can be mitigated. As the amount of data stored in each of the provider is only $1/n$ (where n is the total numbers of cloud storage services used), the cost of data migration is effectively reduced. Other benefits that arise from data striping such as improved overall bandwidth utilization and load balancing will be further discussed in Section 3.4 and Section 3.5 respectively.



(a) (b)

Figure 5. (a) Data striping + parallel upload/download. (b) Erasure coding + parallel upload/download.

3.3 Replication and Erasure Coding

Replication can be used to provide fault-tolerance and extra availability in cloud storage ecosystem, but it incurs high overhead. Recent studies [8,23-32,45] propose to stripe data across multiple cloud vendors and retain the reliability by applying techniques such as redundant array of inexpensive disks (RAID) or erasure coding (figure 5(b)). Erasure coding breaks an object into *k* equal size fragments and generates extra *n* fragments resulting in a total of *m* fragments such that the original object can be recovered from any of the *k* fragments. It ensures data reliability at lower overhead compared to replication. Besides, extra security can be achieved by efficiently dispersing data to several targets using information dispersal algorithm (IDA) [33,] that is based on secret sharing [34]. It can provide data integrity and protect against data corruption. In short, by applying erasure codes in cloud storage, users are more resilient to disaster and outage risks. However, we must take into consideration the overheads incurred because the coding and decoding process can be quite CPU intensive and power hungry. Therefore, it must be modified and optimized for mobile devices that are designed to be low power consumption due to the limited battery capacity.

3.4 Parallelization and Acceleration

In fact, mobile devices nowadays have dual-cores or quad-cores processor and a large amount of RAM (1-2GB). Data can be divided into multiple chunks so that each chunk is processed concurrently. Besides, different storage techniques discussed in this section can be executed in a pipeline fashion without much difficulty. Parallelization and pipeline processing can improve the system's overall performance significantly [35,36] and avoid idle times. In addition, storage operations can be pipelined with network operations such as upload and download [37,38] to improve the overall bandwidth utilization (figure 5). Figure 6 illustrates the concept where the completion time is much faster than sequential processing.

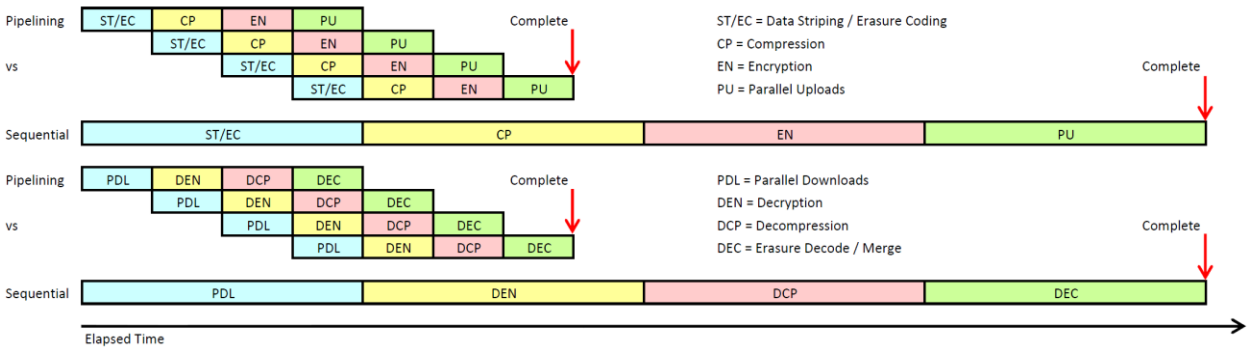


Figure 6. Illustration of pipelining vs. sequential architecture.

Since some of the cloud storage service providers perform per-connection bandwidth throttling, we leverage parallel and multi-chunks upload/download to bypass this limitation. It can improve the transmission speed and keep the user's bandwidth saturated all the time [37,38]. Multiple connections are established to either the same or different cloud servers to transmit different parts of the file concurrently [39]. The improvement is extremely obvious especially in the case where the server throttles each connection bandwidth. It is because users can aggregate the bandwidth from multiple cloud providers, resulting in higher effective bandwidth and bandwidth utilization.

For example, assume there are *n* providers of roughly similar speed. Downloading a file from either one of these sites will take about roughly the same time. However, if the file is mirrored or stripped across different cloud providers, then all *n* parts can be retrieved in parallel. There will be speed improvement of approaching *n* times better, minus some minor processing delay in the striping or merging process.

3.5 Load Balancing and Automated Tiering

Since not all the cloud storage provides equal network speed, each of the operation in parallel download or upload usually completes at different time, which leads to inefficient bandwidth utilization and high energy consumption [14]. The effective throughput is proportional to the slowest link bandwidth. In worst-case scenario, the "tail time" [16] can be significantly long. In order to reduce this effect, we perform load balancing by assigning different priority (automated tiering) to different provider [37-39] instead of using simple Round-Robin selection. When combining with data striping mechanism, we can load balance based on different parameters. For example, we can transmit more data back and forth between client and the i) favorite provider selected by user ii) provider with largest storage capacity iii) provider with the highest link throughput iv) provider with the largest storage capacity left unused v) provider with lowest network delay and response time. Instead of letting the user to assign the priority manually, our system can performs a network benchmark to acquire the effective bandwidth, round-trip time (RTT) and bandwidth delay product (BDP) of each provider, and then automatically determine an optimal ratio for each provider. Due to the fact that network condition is highly unstable and hardly predictable, a running average of the ratio is also being dynamically updated every time a network operation is completed, so that it can adapts better to network condition or geographical location changes. This ratio is applied when uploading processed file (either data

striping or erasure coding) or when downloading from file replicated across multiple cloud storage providers (by using partial download with HTTP GET range). Furthermore, as typical users tend to download more often than upload in normal use cases, the ratio should be more biased towards download bandwidth. Since our method is based on historical performance data, it does not guarantee the theoretical best performance, but it should provide reasonably good performance as perceived by users.

Load balancing is pretty straight forward for data replicated across different providers. More portion of a file is retrieved from the fastest provider (with highest calculated ratio) while less portion from the slower provider. However, file size is an important factor that can impact the result as perceived by the user. Normally, high throughput is favorable because the transfer time will dominate the whole operation. But in the case of small file, low response time is more critical because the operation would complete almost instantly and it does not matter if the throughput is low. For load balancing data processed with data striping, the upload ratio will become the limiting factor because the uploaded portion to each provider must be retrieved fully the next time. Luckily, most of the providers are having symmetric upstream/downstream bandwidth so we can safely neglect this factor. The current capacity should also be taken into consideration as there is no point to keep upload data to the fastest provider when it is low in capacity even though it may be faster than other providers by a big margin.

By automatically tiering different providers into different tiers, load balancing technique can be applied on data processed with erasure coding, albeit it makes the fault-tolerance slightly complex. The higher tier provider (with better performance such as throughput or storage capacity) will stores more erasure coded parts. For example, in a 4+2 (k=4, n=2, m=6) scheme, two parts can be stored in tier-1 provider while the rest are stored in three tier-2 provider and local cache. In this case, it can tolerate two failures of any of the tier-2 location, but cannot tolerate two failures that include the tier-1 location. For the retrieving the data back to the client, only the k fastest providers are selected for retrieving the data because k coded data are sufficient to reconstruct the original file.

3.6 Encryption

As discussed in Section 2.3.3, the level of security associated with current cloud storage services is not well known and may not be sufficient. Privacy concerns grow among consumers due to the recent disclosure of PRISM program. A simple yet effective solution is to perform client-side encryption instead of relying on the security scheme provided by the service provider. By doing this, even if the cloud storage services are under security breach or government subpoena, users' data are not exposed. Client-side encryption also enables users to flexibly choose between different encryption algorithms (DES, AES, Twofish) and key length (128 bits, 256 bits) to adjust the level of security for different untrustworthy providers.

Another major security threat exists where many users tend to use similar password across multiple online services [40]. This is extremely risky because if the password is compromised due to security breach in particular provider, accounts in other online services might be compromised in a chain effect. Similarly, users may use the same key for file encryption, which is also exposed to the similar risk mentioned above. Encryption with a single key is not secure, but using different keys is impractical because inherent difficulties for human to remember all the keys.

Hence, we introduce a two-pass encryption mode (figure 7) where users only need to provide two passwords, a database password (P_d) and a master password (P_m). Then, a database encryption key (K_d) and a master encryption key (K_m) are derived using Password-Based Key Derivation Function (PBKDF2) with high iteration counts. Multiple partial encryption keys ($K_{i...n}$) is then derived from the master key (K_m) using PBKDF2 with lower iteration counts. The file to be encrypted is then split into many small slices (e.g. 4MBytes) and each slice is encrypted with each of the partial keys ($K_{i...n}$) in an interleaved fashion [35,36]. Each of the random salt ($S_{i...n}$) used in partial keys derivation and Initialization Vector (IV) used in Advanced Encryption Standard (AES) encryption is stored along with file metadata in a database encrypted by the database encryption key (K_d). Optionally, user can specifically encrypt different file with a different password. In order to fully decrypt a file, the adversary must fulfill the following requirements: i) Acquire *n* cloud storage passwords (for simple data striping) or at least *k* cloud storage passwords (for erasure coded file). ii) Database password (P_d) to derive database key (K_d) in order to retrieve all the salts ($S_{i...n}$) and IV. iii) The master password (P_m) to derive the master key (K_m) and further deriving each partial encryption key ($K_{i...n}$). Nevertheless, keys management is extremely crucial in this scheme. In worst-case scenario, a file cannot be decrypted successfully if the database is corrupted. Therefore, the encrypted database is also replicated and distributed to each of the *n* subscribed cloud storage services. Note that replication is used instead of erasure coding because the database size is small and negligible.

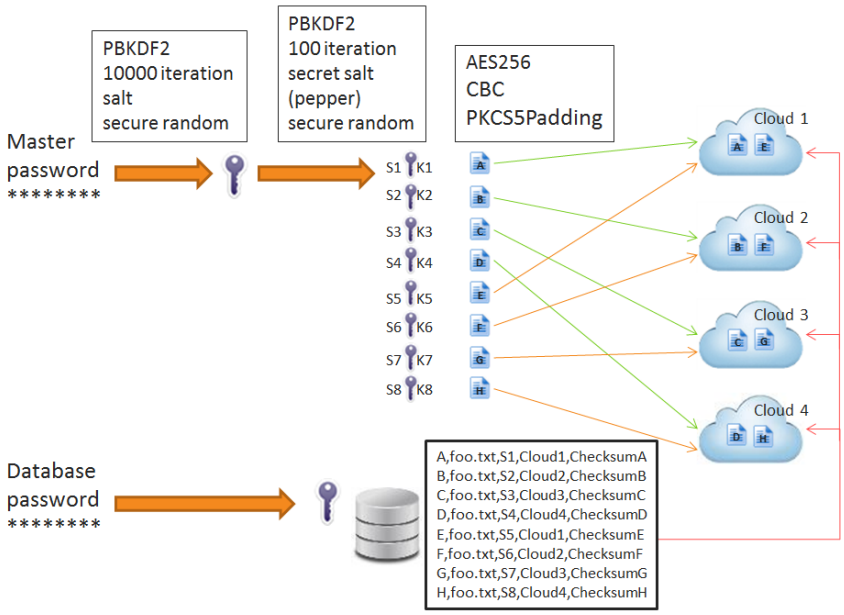


Figure 7. Two-pass encryption mode for file encryption.

3.7 Data Type Aware Compression

Cloud storage quota is expensive (as shown in table 1), as well as the data bandwidth, especially on mobile network such as 3G or LTE. To allow bandwidth and storage quota saving, data can be compressed before transmit through the network. In fact, client application such as Dropbox desktop application performs compression before sending

the data to their server but they calculate the quota based on the original data size instead of the compressed size. By moving this responsibility to the client-side, users can regain the saved quota that they deserved. However, different file possesses different compressibility and not all files can be compressed equally. Some files can be highly compressed (e.g. text, documents, raw media content) while some incompressible files (e.g. pdf, docx, jpeg, mp3, mp4) do not compress much [41]. It is because incompressible are already in highly compressed state. Besides, different file types also yield different compression efficiency when using different compression algorithm (e.g. LZ77, LZW, JPEG). Hence, we can utilize data type aware compression technique in our system. Compression is applied only on compressible data [42], and the best algorithm that will yield high compression ratio is chosen automatically. Without this mechanism, simply applying compression on any file may introduce extra overhead without significant benefits.

3.8 Caching, Prefetching and Network Awareness

One of the main limitation of cloud storage is internet connection is required all the time for it to function properly, as the data are stored remotely and retrieved on demand via network connection. There may be occasions when internet connectivity are disrupted. Therefore, whole file is cached locally to allow offline access even though there is no internet connectivity. Partial cache contains a subset of the total data and when combined with erasure coding mechanism, it can increase the fault-tolerance at no extra overhead on the cloud storage side. For example, in a 3 + 2 erasure coding scheme (where $k = 3$, $m = 5$), one coded part is kept in local cache while four coded parts are stored in different cloud storage locations. In this case, it can actually tolerate two cloud servers outage at only 0.33x storage overhead in the cloud, and 0.33x storage overhead in the local cache.

Data access pattern usually possesses temporal locality or file locality. Hence, we can intelligently prefetch data earlier before it is needed (during cheaper network connection), and hope that it will be requested by user later. Besides reducing response time, it can also avoid sending short burst of data over the network frequently. Network awareness [46] is extremely beneficial because using mobile internet such as 3G or LTE can be terribly expensive. For erasure coded file, uploading all m encoded parts immediately is not necessary. It is because only k parts must be uploaded first while the extra n parts can be delayed to a later time. In addition, if the network condition is unstable or lost, the upload operation may fail. Thus, the data that have not yet been written to the cloud are temporary stored in a cache and it will continue to upload when internet connection is returned to normal or when a cheaper network (Wi-Fi) connection is available. This network awareness mechanism can substantially reduce the expensive mobile data charges by simply delaying the non-priority tasks.

Yet, heavy caching can occupy a lot of internal storage capacity and hence it must be done intelligently to avoid wasting precious storage spaces blindly on the mobile device. It can be achieved by only keeping the new and frequently accessed data in the cache, while periodically performs clean-up for old and inactive data, similar to the Least Frequently Recently Used (LFRU) policy. A SQLite database is responsible to keep track of all files, files metadata and usage frequency. It is replicated in each of the cloud storage to allow seamless switching over to a new

device while still maintaining all files' information. Combining these three techniques can provide users with a smooth and seamless user experience in almost every use cases.

3.9 Data De-duplication

As cloud storage capacity is limited and mobile data bandwidth is relatively expensive especially on 3G and 4G LTE network, it is rational to reduce the storage and bandwidth usage as low as possible. By performing source-based inline data de-duplication, data needed to send through the network can be substantially reduced. Hence, it can save bandwidth usage and improve network transmission speed. In addition, data de-duplication technique can be combined with data compression technique to further reduce the data size.

Data de-duplication aims to reduce storage consumption by identifying distinct chunks of data with identical content and removing them. Only a single copy of the unique chunk is stored along with metadata about how to reconstruct the original files from the chunks. The cost of data de-duplication is the processing power required to calculate the fingerprints, retrieve each block, and reconstruct it back to the original file. It may introduce overhead, but if the reduction in transmission time outweighs the extra time spend on processing de-duplication, it is still a profitable investment.

However, data de-duplication is in conflicting interest with erasure coding because it aims to reduce the storage spaces by removing redundancy, but erasure coding aims to improve the fault-tolerance by introducing redundancy. Nevertheless, we aim to merge both techniques into a mutual state that can benefits from both side. In multiple cloud storages case [47], data de-duplication is performed first follows by erasure coding similar to the approach in R-ADMAD [43].

3.10 Progressive Media Streaming

Not all cloud storage service providers offer media streaming capability. In addition, it requires extra efforts on the server-side to create temporary streaming link for the user to streams from it. Therefore, we apply progressive streaming to overcome this limitation. We run a lightweight proxy server and perform normal download operation in the background, then push the content to media player as soon as data arrives (figure 8). In this case, users can stream from any provider even though the provider does not support media streaming feature.

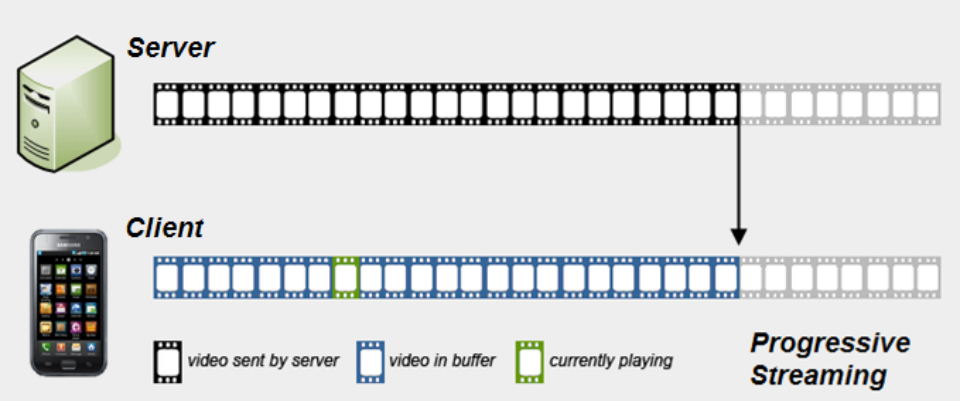


Figure 8. Illustration of progressive media streaming.

3.11 Exploiting File Versioning

Most of the cloud storage services include file versioning, an useful feature where users can restore unintentionally deleted files, or revert the files to the previous version in case of data corruption. For free subscription, it is limited to 30 days or 100 revisions only. The interesting point of this feature is the deleted data can be easily recovered, but it does not count against the user storage quota. It is quite similar to the "recycle bin" concept in desktop environment except the data stored inside can be theoretically unlimited. Hence, user can exploits this feature to temporary boost the storage spaces. This can be easily achieved by temporary deleting some large files to regain some storage capacity, and then recover those deleted files at a later time. This feature can be further exploited to achieve an unlimited storage capacity illusion, by repeating the process indefinitely. To prevent the files stored in "recycle bin" get deleted forever, it must be recovered at least once before the expiration window (30 days). This process can be easily done by restore the file and immediately delete it again, and then the file will last for another expiration time window. This process can also be done programmatically by the application and does not require any user intervention. However, heavy exploiting of this feature maybe be unethical and does not comply with the terms and regulations of service providers.

3.12 Overall Architecture Design

Utilizing the aforementioned concepts, we propose our overall architecture design in figure 9. It follows the layered design outlined in RAOC [23]. There are three main layers in the architecture: i) Mini cloud gateway controller. ii) Data processing layer. iii) Data presentation layer. The data processing layer is further divided into three internal layers. The first internal layer consists of four modes (normal, data striping/merge, encode/decode, replication) and only single mode can be chosen at a time. Each mode has different benefits and tradeoffs as discussed in section 3.2 and section 3.3. The second and third internal layers consist of compression and encryption module, which can be pipelined. The system is designed to be modular so that new module can be added easily to extend the features such as supporting more cloud providers and adding new storage techniques. Figure 10(a) and figure 10(b) shows our simplified design of metadata and naming convention.

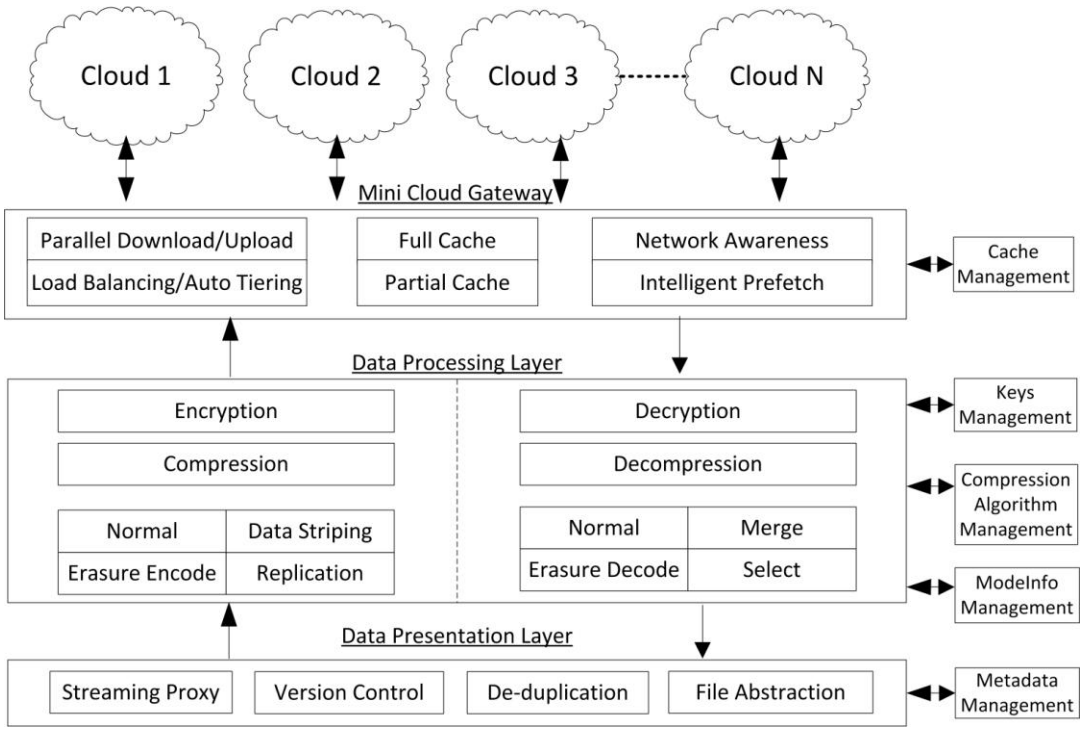
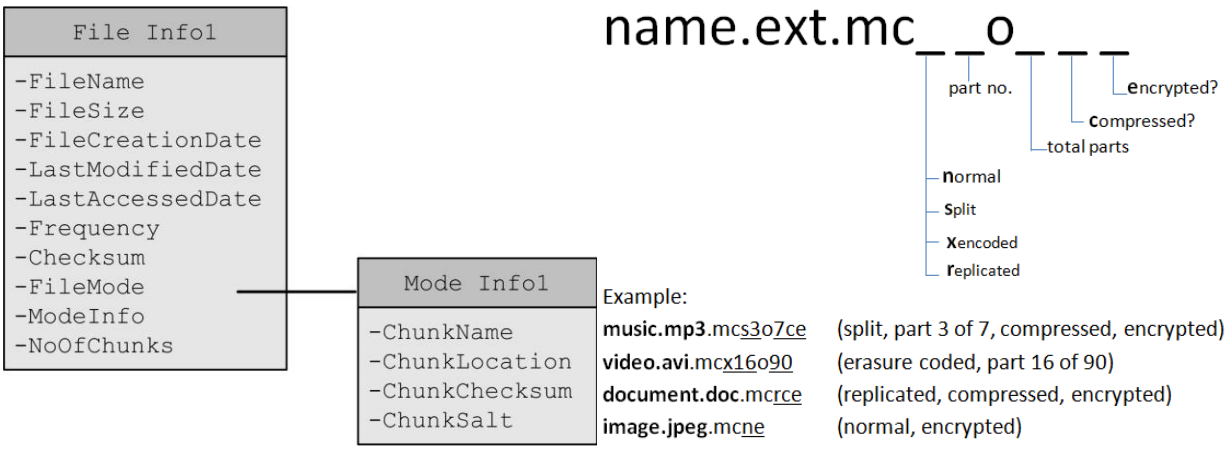


Figure 9. Proposed overall system architecture design.



(a) (b)

Figure 10. (a) Proposed file's metadata structure in database. (b) Proposed file naming convention.

4. Implementation

In this section, we present the realization of our proposed solution, through a prototype implementation on Android mobile device, as Android is the most popular mobile platform [44]. Our solution is implemented as a software application instead of file system due to several reasons: i) It is easier to develop a proof-of-concept prototype. ii) It is easier to distribute the solution to a vast amount of users via .apk file in Google market, and it is easier for users to

perform the installation. iii) Mounting Filesystem in Userspace (FUSE) file system requires root level access, which means users need to "root" their device and void the manufacturer warranty. iv) Most of the official Android ROM does not include FUSE support except some unofficial third party custom ROM. It means users need to flash custom ROM or kernel, and it is a relatively hard process especially for average non tech-savvy users. Flashing custom ROM or kernel also possesses high chance of bricking the device, rendering the device unusable. On the other side, there are several drawbacks of our approach, which are: i) It is not mountable like file system and therefore not full storage virtualization, which means data can be access using the application only. ii) Performance may not as good as file system implementation. Nevertheless, we believe the advantages outweigh the drawbacks in most circumstances.

We utilized jigDFS [24] library for erasure coding. The other libraries used in our prototype such as Bounty Castle cryptography library, Gzip library, NIO and SQLite library are already included in Java SDK and Android development tools. To connect to cloud storage service by different providers, we use Dropbox SDK, Box SDK, Google Drive SDK and Microsoft Live SDK for Android.

Our prototype (figure 11) included most of the techniques mentioned in Section 3 except data de-duplication (Section 3.9), compression algorithm auto selection (Section 3.7), intelligent prefetching (Section 3.8) and file-versioning auto restoration (Section 3.11), which will be further explored in our future works. The current prototype is able to connect to multiple cloud storages, index all files located in different cloud storages, and present it to users in a single list view under the same namespace. Simple CRUD operations and advanced storage techniques such as data striping, erasure coding, compression, encryption, parallel network operations, pipeline data processing and caching are also possible (The advance operations are optional). As a prototype, the "exploit file versioning" feature mentioned on Section 3.11 is implemented on only one of the cloud storage provider (Dropbox).

Our load balancer does not take RTT and BDP into consideration because parameters such as RTT and BDP are not easily obtained by using vendor specific API and requires much works to be done. We denote $S_i \ldots S_n$ as the effective network speed for i to n-th provider. Each provider has its own download speed ($S_{d,i}$) and upload speed ($S_{u,i}$).

$$S_i = \frac{(S_{d,i} * f_d) + (S_{u,i} * f_u)}{f_d + f_u} \qquad R_i = \frac{S_i}{\sum_i^n S} \text{ where } R_i \leq 1$$

Where f_d and f_u are the frequency of download and upload respectively. Then we find the effective ratio for each provider $R_{i \ldots n}$ and use it to load balance across multiple providers.

Figure 11. Screenshots of the prototype application on Android device.

4.1 Implementation Considerations and Rationales

As resources on a mobile device are rather limited compared to its desktop counterparts, the performance might be degraded when performing CPU intensive task. Thus, we must carefully consider the impact and overhead incurred by our system to justify the feasibility of deploying it. To achieve high performance, we mainly utilize the strength of parallelization based on the fact that file can be divided into smaller chunks and each chunk can be processed independently without affecting the overall file.

For erasure coding, we utilize jigDFS library [24], which is the Java implementation of Jerasure library [25] based on Cauchy Reed Solomon algorithm. Study in [26] shows that Jerasure is superior to other erasure codes libraries, in terms of speed and efficiency. There are essentially three main operations in the encoding process, which consist of: i) Hashing the whole file for a fingerprint. ii) Encodes the file to produce $k + n$ chunks. iii) Hashing each encoded chunk for a fingerprint. The operations are similar but in a reversed way for decoding process: i) Hashing each retrieved chunk for a fingerprint and compare it with the fingerprint obtained earlier for integrity checking. ii) Decode k encoded chunks to reconstruct the original file. iii) Hashing the reconstructed file for a fingerprint and comparing it with the fingerprint obtained earlier for integrity checking. The hashing mechanism provides data integrity checking where any data tampering or data corruption can be detected if the newly calculated hash does not match with the previously stored hash. However, these hashing processes are very costly in terms of time and

processing power, especially on mobile devices with sparse resources. Therefore, we utilize Cyclic Redundancy Check (CRC) instead of SHA-1 hashing for better performance and lower overheads, while sacrificing level of data integrity. CRC can protect against data corruption and only unintentionally data tampering, which is more than sufficient in typical cloud storage usage. Of course, users can always manually specify the desired level of data integrity by selecting appropriate hashing algorithm such as SHA-1, SHA-128 or SHA-256, which are also provided in our system.

Besides improving the processing performance, slice-by-slice data processing can reduce the cost of retransmission in case of network failure or disconnection because there is no need to retransmit the whole file again. Since slice-by-slice encoding will yields many small encoded slices (total slices of 6MBytes each * *m* encoded slices), it may cause extra overheads in network transmission when uploading to cloud storage server (figure 12). Hence, it is also rational to merge these slices into a bigger slice before uploading, which we denoted it as a container (figure 13).

Battery capacity on a mobile device is extremely limited. Performing CPU intensive operations such as encoding and encryption on a mobile device may consume a lot of battery power. However, we argue that the actual power consumed by CPU is lesser than the battery power consumed by a slow network operation (upload or download) that keeps the wireless radio in high power state [14-16], due to the fact that wireless radio consumes the second most power on mobile devices, just slightly after the screen. Consequently, our solution can actually resulted in battery power savings in most of the circumstances.

To retrieve an erasure coded file, all related blocks are first downloaded and sorted according to the part number. For pipeline processing, as soon as the first *k* chunks in the first 6MBytes slice have arrived, they are passed to the decoding module for processing, while the *k* chunks in the second slice continue to be downloaded in the background. Each of the decoded blocks is then appended to the previous block to reconstruct the original file. Similarly, to retrieve a stripped file, all related blocks are downloaded in parallel and merged together.

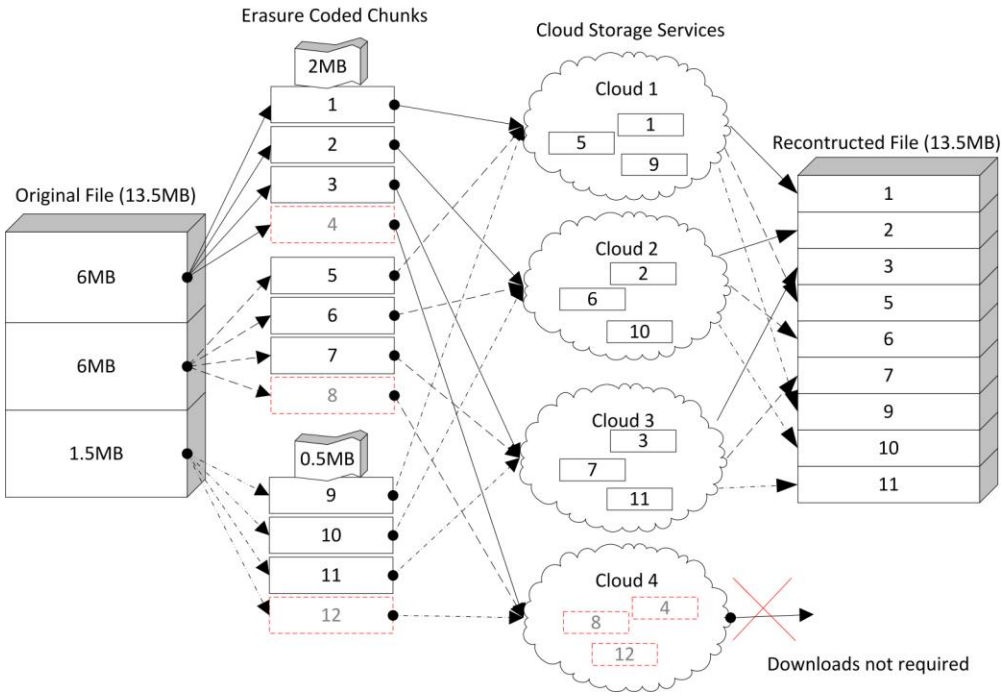


Figure 12. Erasure coded blocks are uploaded directly to cloud storage to allow parallel upload acceleration.

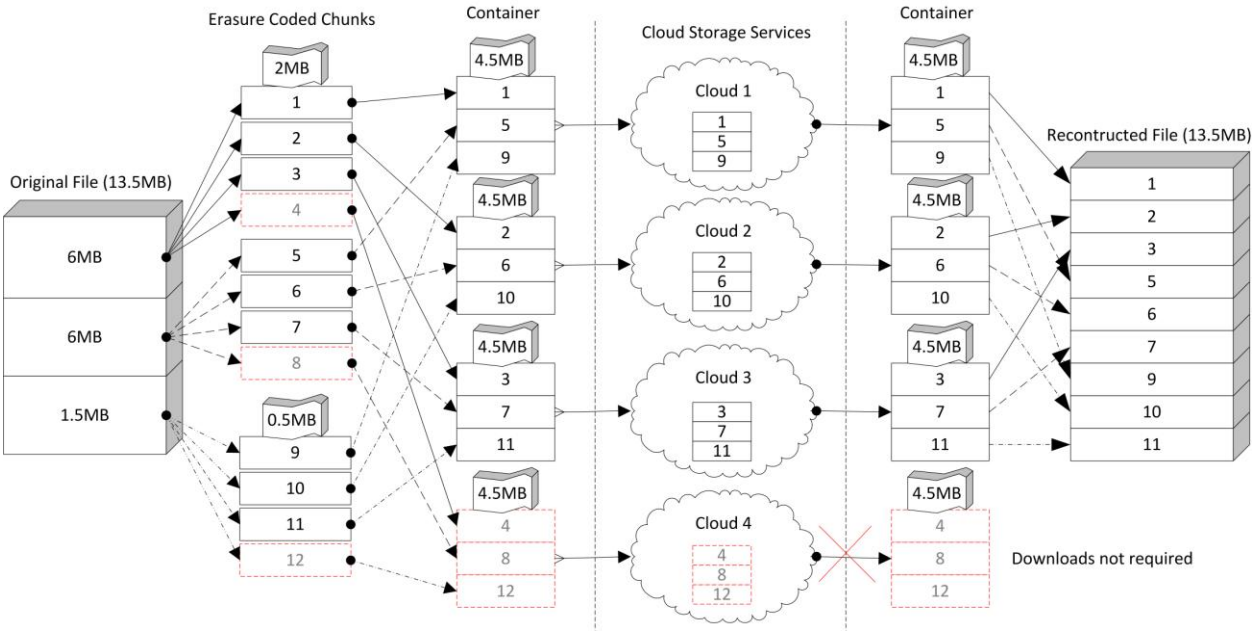


Figure 13. Erasure coded blocks are packed into a container before uploaded to cloud storage.

5. Performance Evaluation

In this section, we present the performance evaluation of our system implementation which tries to answer three main questions: *What is the advantage in terms of performance, cost and availability of using multiple providers? Is the performance gain outweighing the overheads? Did it solve the limitations outlined in Section 2?*

We evaluated the system on three Android smartphone devices from different generations, launched in year 2010, 2011 and 2012 respectively. They consist of a low-end Galaxy S (Single-core processor with 512MB RAM), one middle level Galaxy Nexus (Dual-cores processor with 1GB RAM) and one high-end Galaxy S3 (Quad-cores processor with 2GB RAM). To simulate real world environment, we evaluated our system on two networks consist of SK-Telecom (SKT) LTE mobile network and Korea Telecom (KT) residential internet network via Wi-Fi connection. Bandwidth benchmark with nearest server using Ookla speed test service[10] during non-peak hour results in an average download/upload bandwidth[11] of 37/34 Mbits/s and 29/23 Mbits/s respectively. We run several different performance tests in this section. Each test is performed for five times and the average time consumption is recorded, unless otherwise specified.

5.1 Comparison between Different Cloud Storage Providers and Combined Approach

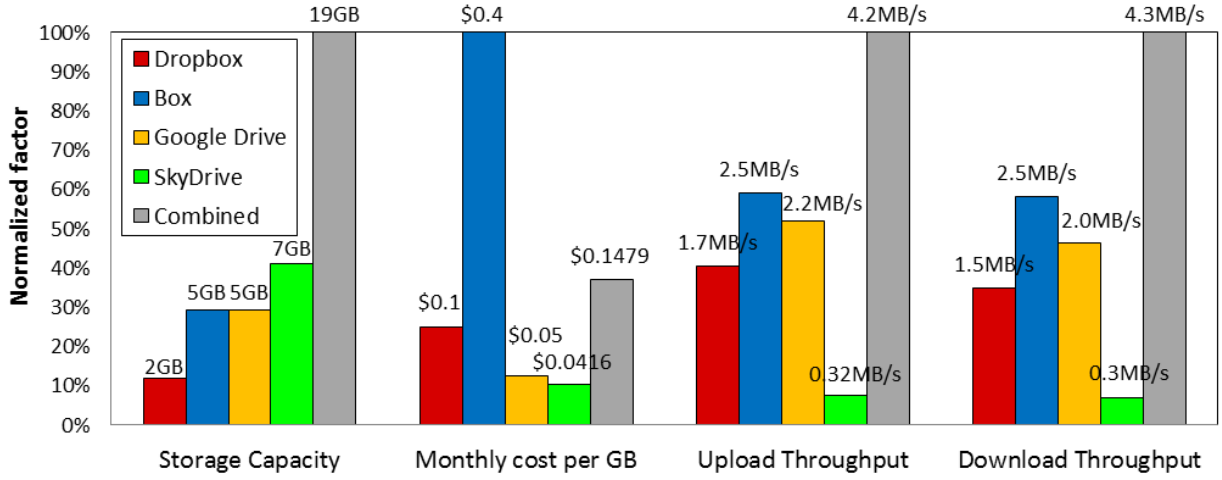


Figure 14. Comparison between different cloud storage service providers.

Figure 14 illustrates the storage capacity, pricing and performance difference on different providers, normalized with respect to the highest factor. We measure the time consumption and calculate the network throughput for uploading and downloading a 25Mbytes test file with each of the different provider and then with multiple providers in parallel. The results show that the effective storage capacity and network performance is improved significantly by aggregating resources from multiple cloud storage services. We also noticed that while SkyDrive offers the largest storage capacity for free subscription and cheapest pricing for paid subscription ($0.0416 per GB each month), the network performance is comparatively lower than other providers. On the other hand, Google Drive paid

[10] http://www.speedtest.net/

[11] The effective bandwidth may vary according to location, network condition and device's capability.

subscription offers much better network performance than SkyDrive but is only slightly more expensive (priced at $0.05 per GB each month for 100GB plan and above). Box charges the highest monthly subscription fee, but it offers the highest network bandwidth, albeit lower than the combined bandwidth of our approach. By using our approach, we reach a combined bandwidth of 4.2 Mbytes/s upload and 4.3 Mbytes/s download. It only peaks at this value due to different limitations on the device and the network itself (802.11n or LTE). The maximum throughput is expected to be higher (more than 6 Mbytes/s) on better hardware such as on 802.11ac devices or on better network such as LTE-Advance. By using multiple cloud storage, the effective monthly cost is also being averaged down from the most expensive provider. In this case, the Box pricing of $0.4 per GB is reduced to $0.1479 per GB, which is roughly 60% cheaper. Although the combined pricing ($0.1479) is still higher than the other three providers, users can actually gain several advantages discussed in Section 3 (avoiding vendor lock-in, lower migration cost, fault-tolerance and improved performance) which are not possible when using single provider.

5.2 Performance Evaluation of Different Data Processing Techniques

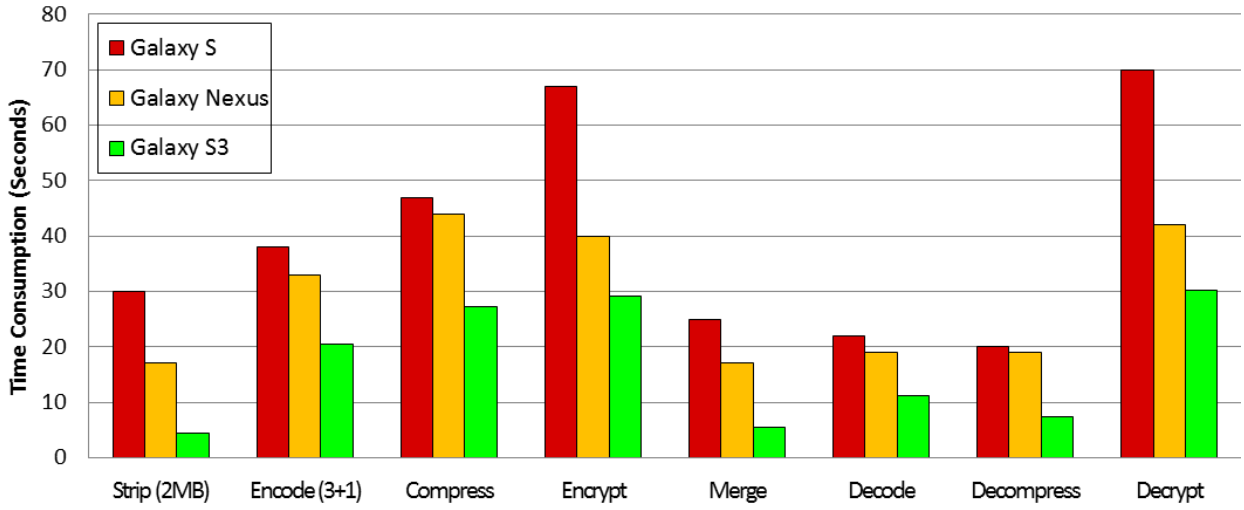


Figure 15. Time consumption of different data processing techniques on three Android devices (100Mbytes file).

We measure the time consumption of processing different storage operations by on a 100Mbytes incompressible file retrieved from Linode[12] website. It is repeated on three mobile devices with different specifications and the results are shown in figure 15. As observed, device from newer generation performs nearly two times faster than device from previous generation in most of the operations.

5.3 Comparison between Sequential Processing and Parallel Processing

[12] www.linode.com/speedtest

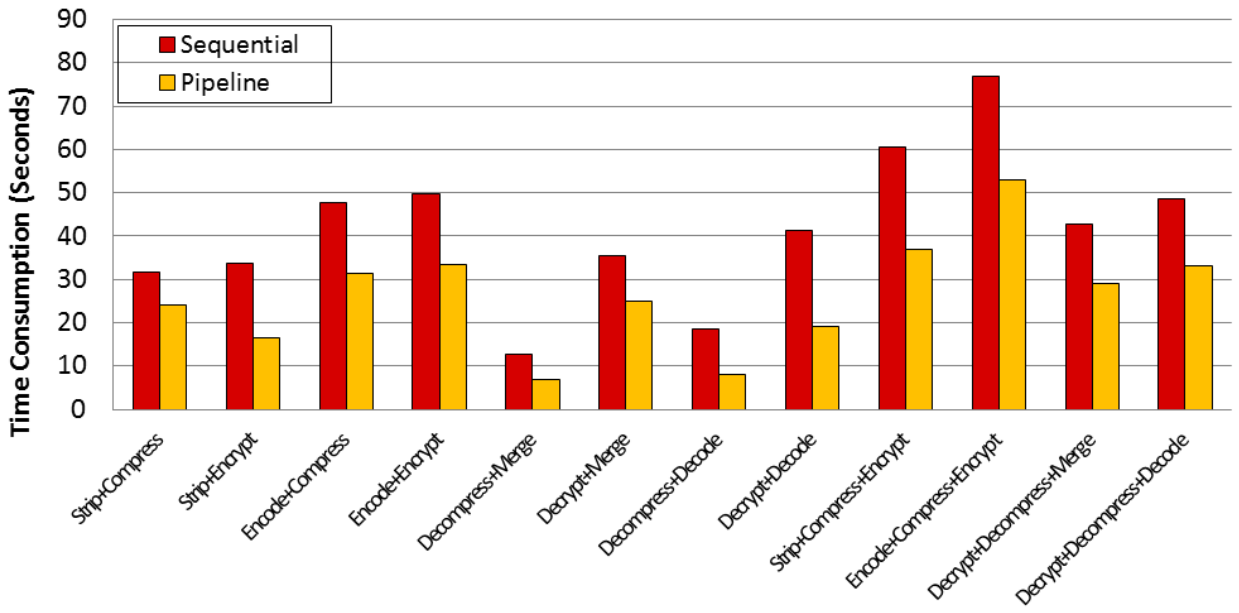


Figure 16. Sequential vs. Parallel Performance.

Next, time consumption of parallel and pipeline processing using a combination of different data processing methods discussed in Section 3 is measured and compared with sequential processing. The results are shown in figure 16 (results for Galaxy S3 only) and we can observe that data processing time is reduced significantly. Parallelization and pipelining allows each chunk of data to be processed concurrently, which can optimize processor utilization and improve the execution time. Nevertheless, the amount of speed-up is limited according to Amdahl's law.

5.4 Processing Performance of Mobile Devices from Different Generations

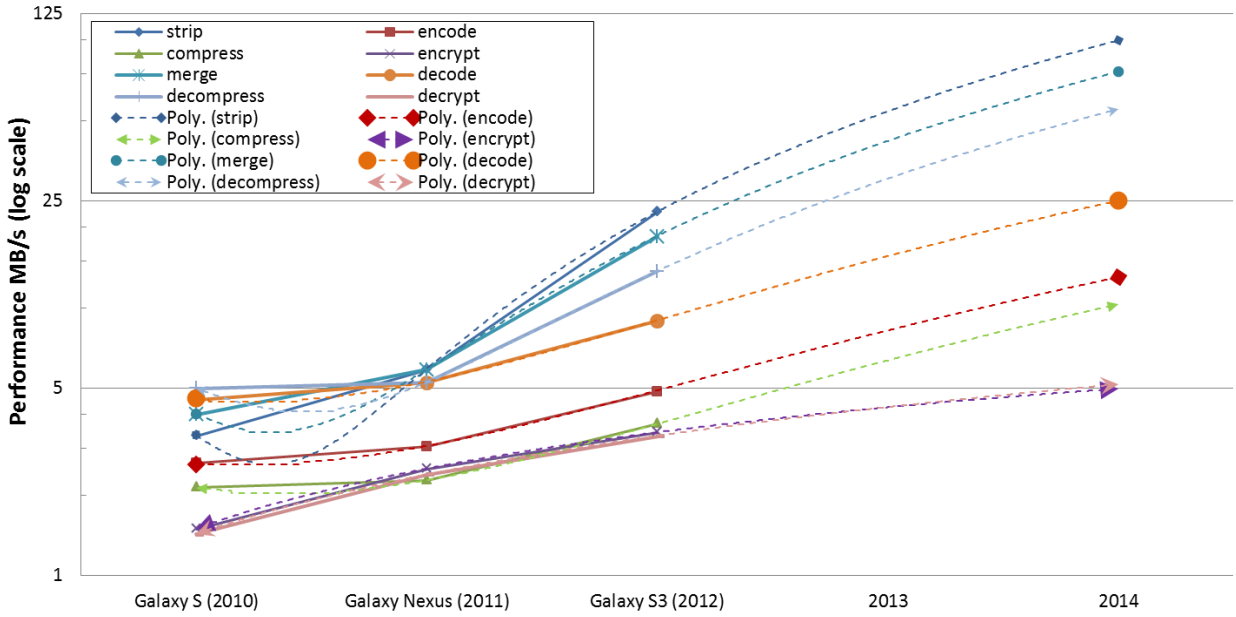


Figure 17. Forecasting performance improvement over the coming years.

We calculate the processing performance by dividing the file size over the time consumption. The performance improvement is approaching two times over each generation, as the device moves from single-core to dual-cores to quad-cores and so on. Therefore, it is safe to assume the performance to improve continuously over the coming years. For a more realistic estimation, we plot a polynomial graph of order two (figure 17) over the performance of devices from previous generations. As the performance continues to grow, the overheads incurred by data processing will become lower and insignificant, which means our proposed solution of applying storage techniques on the client-side will become more feasible. Our estimation is rough but it can be more accurate when more data are available, i.e. performance of newer generation devices (2013 and 2014).

5.5 Evaluation of Erasure Coding Performance by Varying Different Parameters

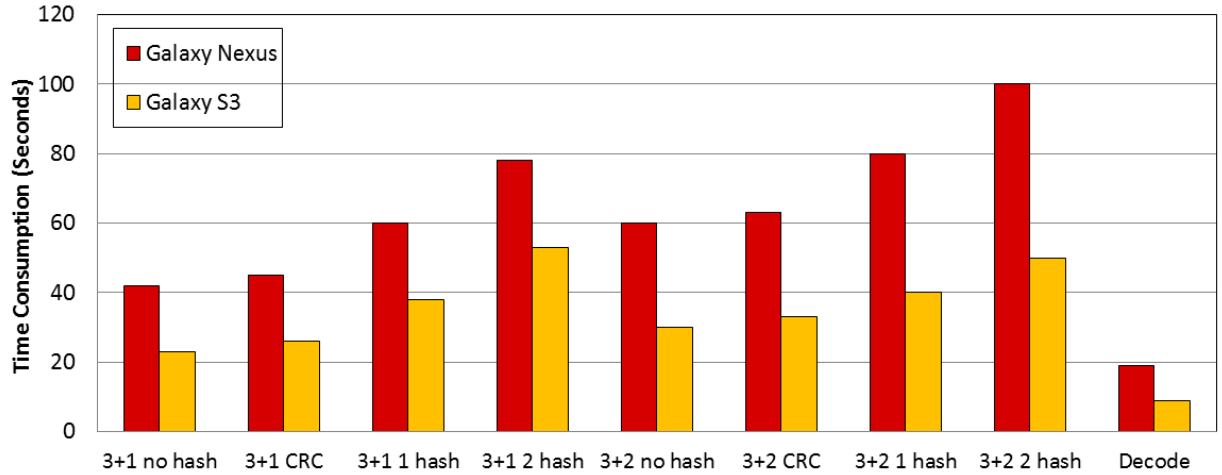


Figure 18. Erasure coding 100MBtyes file with different parameters (k=3, n=1, n=2) and checksum algorithm.

The erasure coding performance is influenced by many factors. The main contributors are the degree of redundancy, the checksum algorithm and the device's processing power. We evaluate the performance by measuring the time consumption for erasure coding a 100Mbytes incompressible with different parameters. It is repeated on two different devices equipped with multiple cores processor (Galaxy Nexus and Galaxy S3). The results are shown in figure 18. We can observe that the time consumption of performing erasure coding is surprisingly low even on mobile devices with limited processing power. By varying the checksum algorithm (CRC32 or SHA1/SHA256) for data integrity checking, the time consumption increase accordingly.

5.6 Evaluation of Network Performance

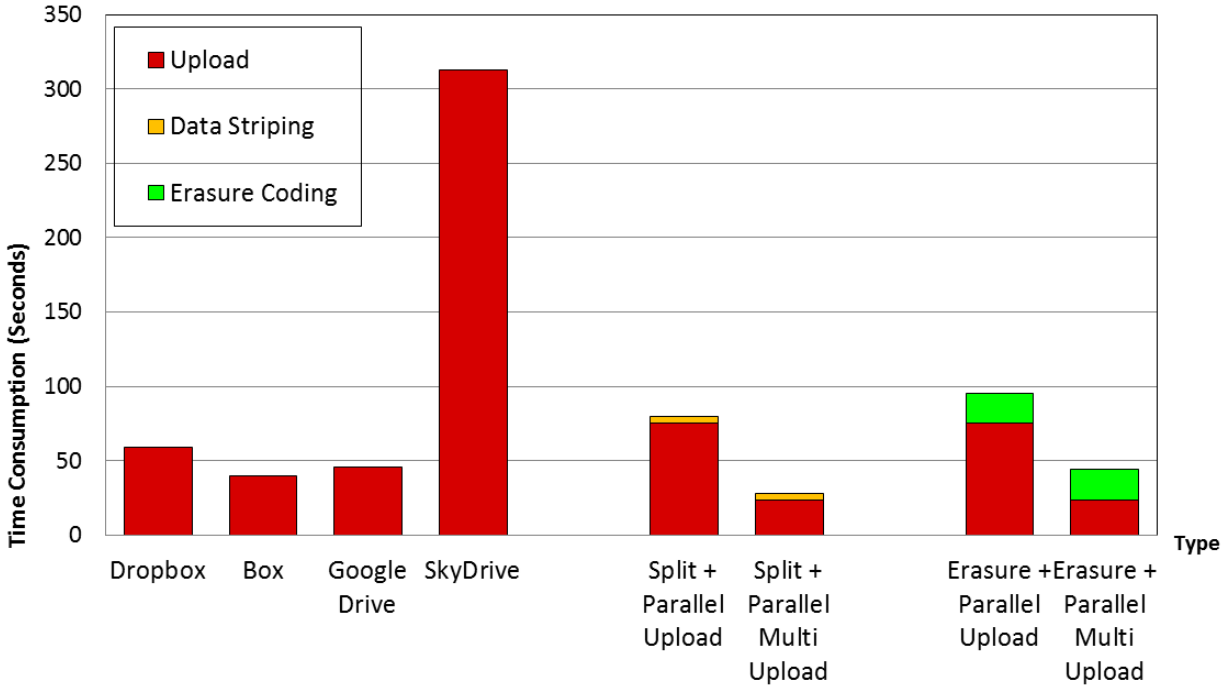


Figure 19. Time consumption of upload operations on Wi-Fi network (100Mbytes file).

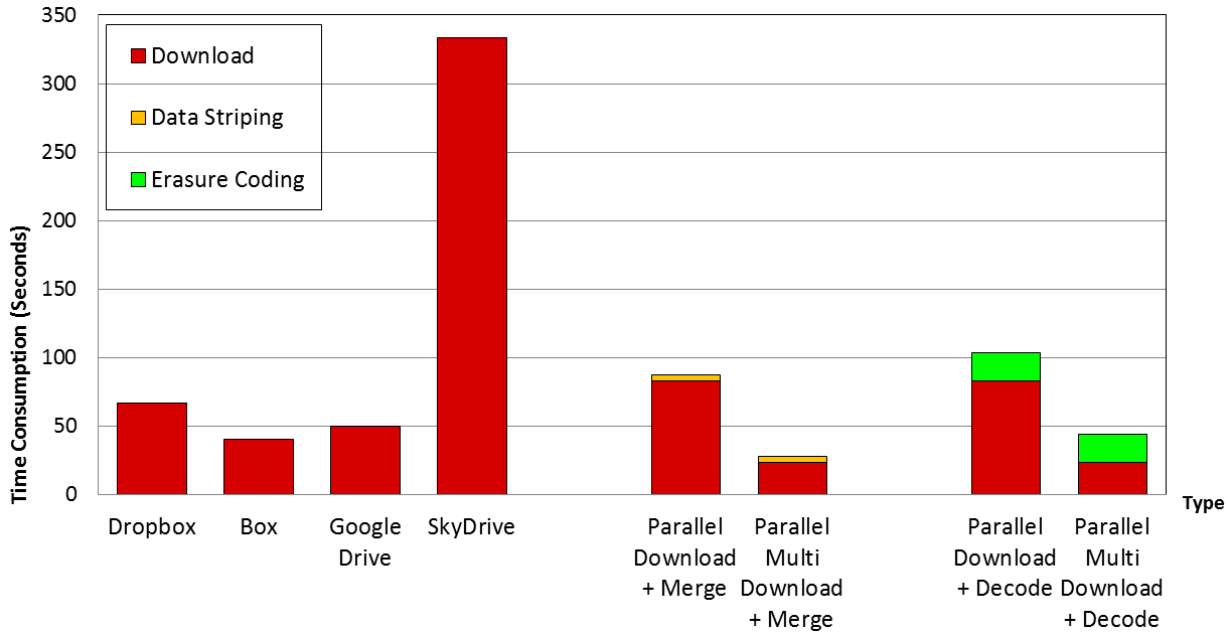


Figure 20. Time consumption of download operations on Wi-Fi network (100Mbytes file).

An evaluation of network performance is performed. We compare the time consumption between using single provider and multiple providers. The results are shown in figure 19 and figure 20. Based on the results presented in the figure, we can observe that the overheads incurred by data striping and erasure coding process are actually remarkably little compared to the overall time for network operations. We attribute this to the strength of parallelism and pipelining. The network I/O will almost always be the limiting factor rather than the storage I/O, hence the time consumption is dominated by the network operations. Using multiple providers is slower than using a single fastest provider because it needs to tolerate the slowest provider (tail time) where the time consumption is actually being averaged down. By establishing multi connections (denoted as "Parallel Multi" in figure 19 and figure 20) to each provider, this bottleneck can be avoided. As a result, the time consumption can be further reduced, resulting in a performance faster than any single fastest provider does. It also results in higher bandwidth utilization and battery energy saving as discussed in section 2.2.1. Therefore, the incurred overheads are actually compensated by the performance gained.

5.7 Dynamic Load Balancing

Figure 21. Parallel downloads without/with load balancing.

We trace the network pattern using Dalvik Debug Monitor Server (DDMS) tools provided in Android SDK. Figure 21 represent the general pattern of bandwidth utilization when performing network operations back and forth with multiple cloud storage simultaneously. Each color region indicates the bandwidth utilization of network connections to different cloud storage, as stated in the figure. We can observe that by using a dynamic ratio load balancing favoring the faster provider, the overall network throughput is higher. It ensures that all of the download and upload operations complete at almost similar time frame, and hence the "tail time" is effectively mitigated. As a result, the wireless radio in mobile device can go into idle state earlier, which leads to lower battery energy consumption.

5.6 Discussions

Single interface to access all the cloud storages as a centralized storage is essential to users because of its intuitiveness and simplicity. It allows agile management of data across multiple services. It also allows user to aggregate resources from multiple providers, increases the storage capacity at zero cost (for free subscriptions) or reduces the migration cost (for paid subscription). Using multiple providers enables us to apply essential storage techniques on the client-side, but it can cause overheads especially on resource-constrained mobile devices. Nevertheless, today's high-end smartphones are equipped with quad-cores processor and 2GB of RAM. By utilizing parallelization and pipeline processing, overheads are reduced while performance is improved. With high-end device

such as LG Google Nexus 4 (quad-cores, 2GB RAM) selling as low as 299USD unlocked[13], our proposed solution is economically viable.

There are still a few limitations existed in our solution. Users are required to sign up for multiple cloud storage services manually. Although this is a simple and one-time only process, it may appear to be troublesome for certain people. Otherwise, if users are already using multiple cloud storage services, only OAuth2 authorization process is required to start using the system. In our experiment, the process of authorizing four cloud storage services took less than one minute. Only new files uploaded via the system can benefit from the techniques applied in data processing layer. Hence, a conversion tool [47] that will convert the previously stored and unprocessed files into processed files is planned for the future works. Users must also aware that the processed files (either stripped or erasure coded) cannot be retrieved correctly when using the web interface on the official website. This side effect is actually the intention of information dispersal algorithm (IDA) which efficiently hides the data from any single service provider. There are clearly trade-offs between using different storage techniques (data striping vs. erasure coding vs. replication) and users must choose depends on their level of demand. Since our solution is not developed as file system, traditional file system semantics and consistency is not provided. File can be only access via the system, or can be access locally after it is stored locally, similar to typical cloud storage services.

6. Related Work

Several studies [8,23-32,45] proposed the similar approach of improving cloud storage services by combining the storage from different vendors. However, each of study focused on different objective. RACS [8] focuses on avoiding vendor lock-in issue by reducing the migration cost. NCCloud [27] focuses on reducing the repair cost by introducing F-MSR coding scheme. ROAC [23] focuses on creating optimal cloud storage systems by considering non-functional properties. jigDFS [24] focuses on a file system with strong encryption and a certain level of plausible deniability. AONT-RS [28] described a new dispersal algorithm that can achieve high security with low computational and storage costs. DEPSKY [29] focuses on confidentiality, integrity and availability (CIA) of information stored in the cloud. Similarly, HAIL [30] acts as a distributed cryptographic system, which allows cloud servers to compute the proof of availability and integrity of the stored data. μLibCloud [31] is most similar to our approach, but their choice of cloud providers are mainly targeted at the enterprise level instead of average household consumers. Most of all, they have used erasure coding or information dispersion algorithm (IDA) outlined by Rabin [33]. Unlike existing studies, our works focus on improving the overall performance from every aspect while minimizing the incurred overheads. Our main research target also differs from previous studies that focus on enterprise or desktop environment where CPU and network resources are not a concern. Our study focuses on mobile devices that are ubiquitous and allow access to cloud storage even on the go but are resources constrained in terms of computing power, network resources and battery power. In addition, our solution is implemented as

[13] Unlocked phone implies that users are not bound to any contract. The total cost of ownership is the price of the device.

software appliances that can be easily distributed and installed on the device itself instead of requiring dedicated hardware or proxy [23].

7. Conclusion and Future Works

We present the prototype of a multi cloud storage application middleware that allows users to enjoy better cloud storage services from multiple providers at zero cost, plus minor efforts on the client-side. It addresses the limitations of today's cloud storage services and delivers improved performance on several aspects including speed and energy consumption. It allows users to take advantage of each provider's strongest features while minimizing their weakness. Our results show that it is feasible to be applied even on resource-constrained mobile devices at the cost of minor overheads. Our system can be distributed to public users via Google Android market and can be installed easily.

In future works, we aim to improve the encoding and encryption performance by using native code or Renderscript in Android SDK. Other techniques that have been proposed in section 3 but yet to be implemented, such as data de-duplication and data-aware compression are planned for the near future. More choices of provider will be added into the cloud gateway layer, and it will be upgraded to the more powerful REST or SOAP protocol. Then, a more throughout performance analysis on the network and battery utilization will be performed with ARO tools [15,16]. Finally, we plan to prototype a client application for the desktop environment as to complete the whole ecosystem of our proposed solution.

8. Acknowledgements

This work was supported in part by National Research Foundation of Korea under Grant 2011-0009349.

References

[1] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., ... & Zaharia, M. (2010). A view of cloud computing. Communications of the ACM, 53(4), 50-58.

[2] Hady, F. and Payne, M. (2012) Why we need whole home storage architecture [online], available http://www.intelconsumerelectronics.com/Consumer-Electronics-3.0/Home-Storage-Architecture.aspx [accessed 9 December 2012]

[3] BBC (2011) Hard disk and camera makers hit by Thai floods [online], available: http://www.bbc.co.uk/news/technology-15534614 [accessed 9 December 2012]

[4] Gartner (2012) Gartner Says That Consumers Will Store More Than a Third of Their Digital Content in the Cloud by 2016 [online], available http://www.gartner.com/newsroom/id/2060215 [accessed 9 December 2012]

[5] Ellis, H. (2012) Google Drive vs. Dropbox, SkyDrive, SugarSync, and others: a cloud sync storage face-off [online], available http://www.theverge.com/2012/4/24/2954960/google-drive-dropbox-skydrive-sugarsync-cloud-storage-competition [accessed 9 December 2012]

[6] Naldi, M., & Mastroeni, L. (2013, April). Cloud storage pricing: a comparison of current practices. In Proceedings of the 2013 international workshop on Hot topics in cloud services (pp. 27-34). ACM.

[7] Han, Y. (2011). Cloud computing: case studies and total cost of ownership. Information technology and libraries, 30(4), 198-206.

[8] Abu-Libdeh, H., Princehouse, L., & Weatherspoon, H. (2010, June). RACS: a case for cloud storage diversity. In Proceedings of the 1st ACM symposium on Cloud computing (pp. 229-240). ACM.

[9] Canalys (2012) Smart phones overtake client PCs in 2011 [online], available http://www.canalys.com/newsroom/smart-phones-overtake-client-pcs-2011 [accessed 9 December 2012]

[10] Ross, C. (2013) Best full HD smartphones [online], available: http://www.pcworld.idg.com.au/roundup/453755/best_full_hd_smartphones/ [accessed 10 July 2013]

[11] Gartner (2012) Gartner Says Free Apps Will Account for Nearly 90 Percent of Total Mobile App Store Downloads in 2012 [online], available http://www.gartner.com/newsroom/id/2153215 [accessed 23 March 2013]

[12] Victoria, B. (2011) Dropbox: The Inside Story Of Tech's Hottest Startup [online], available http://www.forbes.com/sites/victoriabarret/2011/10/18/dropbox-the-inside-story-of-techs-hottest-startup/ [accessed 10 July 2013]

[13] Genius Geeks (2013) Manage Multiple Cloud Storage Services Efficiently With These Tools [online], available http://geniusgeeks.com/manage-multiple-cloud-storage-services-efficiently [accessed 23 March 2013]

[14] Google (2012) Optimizing Downloads for Efficient Network Access [online], available http://developer.android.com/training/efficient-downloads/efficient-network-access.html [accessed 9 December 2012]

[15] Alexandre, G., Subhabrata, S., Oliver, S. (2011) A Call for More Energy-Efficient Apps [online], available http://www.research.att.com/articles/featured_stories/2011_03/201102_Energy_efficient?fbid=plCYzUNB0e3 [accessed 9 December 2012]

[16] Qian, F., Wang, Z., Gerber, A., Mao, Z., Sen, S., & Spatscheck, O. (2011, June). Profiling resource usage for mobile applications: a cross-layer approach. In Proceedings of the 9th international conference on Mobile systems, applications, and services (pp. 321-334). ACM.

[17] Sean, G. (2012) Microsoft's store site in India defaced; hackers find plain text passwords [online], available http://arstechnica.com/business/2012/02/microsofts-store-site-in-india-defaced-hackers-find-plain-text-passwords/ [accessed 9 December 2012]

[18] Kamp, Poul-Henning, et al. "LinkedIn Password Leak: Salt Their Hide." ACM Queue 10.6 (2012): 20.

[19] Andrew, C. (2012) Upload at your own risk: Most cloud storage services offer no data guarantee [online], available http://www.digitaltrends.com/computing/upload-at-your-own-risk-most-cloud-storage-services-offer-no-data-guarantee/ [accessed 9 December 2012]

[20] SpiderOak (2013) Is SpiderOak really "zero knowledge"? Could you read a user's data if forced at gunpoint? [online], available https://spideroak.com/faq/questions/23/is_spideroak_really_zero_knowledge_could_you_read_a_users_data_if_forced_at_gunpoint/ [accessed, 10 July 2013]

[21] Erik, L. (2011) U.S. Twitter Subpoena Is Harassment, Lawyer Says [online], available http://www.bloomberg.com/news/2011-01-10/u-s-twitter-subpoena-on-wikileaks-is-harassment-lawyer-says.html [accessed 10 July 2013]

[22] Kumar, K., & Lu, Y. H. (2010). Cloud computing for mobile users: Can offloading computation save energy?. Computer, 43(4), 51-56.

[23] Spillner, J., Müller, J., & Schill, A. (2013). Creating optimal cloud storage systems. Future Generation Computer Systems, 29(4), 1062-1072.

[24] Bian, J., & Seker, R. (2009, March). Jigdfs: A secure distributed file system. In Computational Intelligence in Cyber Security, 2009. CICS'09. IEEE Symposium on (pp. 76-82). IEEE.

[25] Plank, J. S., Simmerman, S., & Schuman, C. D. (2008). Jerasure: A library in C/C++ facilitating erasure coding for storage applications-Version 1.2. University of Tennessee, Tech. Rep. CS-08-627, 23.

[26] Plank, J. S., Luo, J., Schuman, C. D., Xu, L., & Wilcox-O'Hearn, Z. (2009, February). A Performance Evaluation and Examination of Open-Source Erasure Coding Libraries for Storage. In FAST (Vol. 9, pp. 253-265)

[27] Hu, Y., Chen, H. C., Lee, P. P., & Tang, Y. (2012, February). NCCloud: applying network coding for the storage repair in a cloud-of-clouds. In Proceedings of the 10th USENIX conference on File and Storage Technologies (pp. 21-21). USENIX Association.

[28] Resch, J. K., & Plank, J. S. (2011, February). AONT-RS: blending security and performance in dispersed storage systems. In Proceedings of the 9th USENIX conference on File and storage technologies (pp. 14-14). USENIX Association.

[29] Bessani, A., Correia, M., Quaresma, B., André, F., & Sousa, P. (2011, April). DepSky: dependable and secure storage in a cloud-of-clouds. In Proceedings of the sixth conference on Computer systems (pp. 31-46). ACM.

[30] Bowers, K. D., Juels, A., & Oprea, A. (2009, November). HAIL: a high-availability and integrity layer for cloud storage. In Proceedings of the 16th ACM conference on Computer and communications security (pp. 187-198). ACM.

[31] Mu, S., Chen, K., Gao, P., Ye, F., Wu, Y., & Zheng, W. (2012, September). µLibCloud: Providing High Available and Uniform Accessing to Multiple Cloud Storages. In Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on (pp. 201-208). IEEE.

[32] Cachin, C., Haas, R., & Vukolic, M. (2010). Dependable storage in the Intercloud. IBM Research, 3783, 1-6.

[33] Rabin, M. O. (1989). Efficient dispersal of information for security, load balancing, and fault tolerance. Journal of the ACM (JACM), 36(2), 335-348.

[34] Shamir, A. (1979). How to share a secret. Communications of the ACM, 22(11), 612-613.

[35] Dongara, P., & Vijaykumar, T. N. (2003, March). Accelerating private-key cryptography via multithreading on symmetric multiprocessors. In Performance Analysis of Systems and Software, 2003. ISPASS. 2003 IEEE International Symposium on (pp. 58-69). IEEE.

[36] Ashokkumar, S., Karuppasamy, K., Srinivasan, B., & Balasubramanian, V. (2010). Parallel Key Encryption for CBC and Interleaved CBC. International Journal of Computer Applications, 2(1), 21-25.

[37] Mohamed, N., Al-Jaroodi, J., & Eid, A. (2013). A Dual-Direction Technique for Fast File Downloads with Dynamic Load Balancing in the Cloud. Journal of Network and Computer Applications.

[38] Rodriguez, P., & Biersack, E. W. (2002). Dynamic parallel access to replicated content in the Internet. IEEE/ACM Transactions on Networking (TON), 10(4), 455-465.

[39] Philopoulos, S., & Maheswaran, M. (2001, August). Experimental Study of Parallel Downloading Schemes for Internet Mirror Sites. In Thirteenth IASTED International Conference on Parallel and Distributed Computing Systems (PDCS'01) (pp. 44-48).

[40] Alastair, M., Niger, P. (2013) A Survey of Australian Attitudes Toward Password Use and Management [online], [accessed 10 July 2013]

[41] Winzip (2012) Why don't some files compress very much? [online], available http://kb.winzip.com/kb/?View=entry&EntryID=104 [accessed 9 December 2012]

[42] Harnik, D., Kat, R., Margalit, O., Sotnikov, D., & Traeger, A. (2013, February). To Zip or not to Zip: Effective Resource Usage for Real-Time Compression. In Proceedings of the 11th USENIX conference on File and Storage Technologies. USENIX Association.

[43] Liu, C., Gu, Y., Sun, L., Yan, B., & Wang, D. (2009, June). R-ADMAD: High reliability provision for large-scale de-duplication archival storage systems. In Proceedings of the 23rd international conference on Supercomputing (pp. 370-379). ACM.

[44] IDC (2012) Worldwide Mobile Phone Growth Expected to Drop to 1.4% in 2012 Despite Continued Growth Of Smartphones, According to IDC [online], available http://www.idc.com/getdoc.jsp?containerId=prUS23818212#.UL56zI5JA21 [accessed 10 July 203]

[45] Decasper, D., Samuels, A., & Stone, J. (2012). U.S. Patent No. 20,120,047,339. Washington, DC: U.S. Patent and Trademark Office.

[46] Lingafelt, C. S., Murray, J. W., Swantek, J. T., & Worley, J. S. (2011). U.S. Patent No. 20,110,307,573. Washington, DC: U.S. Patent and Trademark Office.

[47] Prahlad, A., Kottomtharayil, R., Kavuri, S., Gokhale, P., & Vijayan, M. (2010). U.S. Patent No. 20,100,332,818. Washington, DC: U.S. Patent and Trademark Office.