CECS 526

Fall 2015

Term Paper Assignment

HELI SHAH

DHWANI SUNIL PATEL

Raindrop File System – A Network File System for Mobile Devices and the Cloud

Abstract

Nowadays, access to dependable storage for mobile devices is becoming a vital problem to which cloud storage is the optimal solution. Although cloud storage addresses this issue, it introduces a number of multifarious challenges like unpredictable wireless network connectivity and information security across the network.

A feasible solution overcoming these challenges is Raindrop File System, addressed as RFS hereafter, which is a wireless-friendly network file system for mobile devices and the cloud using client-centric design. RFS functions by connecting mobile devices with the cloud by extending undependable mobile storage to dependable and scalable cloud storage without requiring the client nodes to be always connected over internet. This gives the mobile users an edge by leveraging cloud's dependable and unlimited storage thereby providing data privacy.

We analyse how RFS is better than other network file systems and its various operations done on client side and server side and demonstrate the issues faced while implementing RFS over multiple operating systems and how memory is synchronized between client and server. The empirical results of RFS delivers a better user experience, good file system performance and less communication overhead explaining the feasibility of running a complete mobile system from the cloud.

Table of Contents

Introduction

In the present day, mobile devices are replacing laptops and traditional computers in order to provide better mobility and ease of use. These devices require enormous storage space due to use of multimedia applications. Thus the storage space available on these devices limits how much multimedia files can be used on the device and the user is constantly removing files to make space to add new ones. There is a need to permanently solve this problem and integration to cloud based storage elegantly solves this problem. Cloud based file system provides anytime and anywhere access to the unlimited storage of a cloud to the mobile device users. [2]

However, in practicality there is a continuous fluctuation in the bandwidth and network connectivity which makes the effective use of cloud storage challenging. One way to connect mobile devices with the cloud is network file systems. This gives rise to two fundamental issues which must be addressed. Firstly, there is unpredictable wireless connectivity and secondly there are data privacy issues. To date, most of the popular network file systems are still not suitable for file sharing between mobile devices and the cloud. These file systems have different costs and performance characteristics which mobile devices are not designed for.

Drawbacks of Existing Solutions

Existing solutions include using external cards or online storage applications[2] but there are some limitations associated with it which are as follows:

- External cards cannot help to secure user's data in case the mobile device gets lost. So a user needs to continuously backup the data in order to keep it secured.
- Online storage applications have different user interface which makes it difficult for a user to view the system. Many cloud systems are not yet made responsive to all the devices making it difficult for the user to operate such applications.

Hence, cloud computing is a better solution.

Advantages of Cloud Storage

Cloud storage provides improved capacity, processing power and improved reliability. Cloud computing also helps in reducing the running cost of compute- intensive applications that take long time and large amount of energy when performed on limited resource devices. Cloud

computing can efficiently support various tasks for data warehousing, managing and synchronizing multiple documents online.

Storing data or running applications on cloud is an effective way to improve the reliability since data is stored and backed up on the cloud. This reduces the chance of data loss. Furthermore, the cloud can be used to protect copyrighted digital content from being abused and unauthorized distribution. Also, the cloud can provide security from viruses and malicious codes as scanning and detection is done online.[2]

RFS Overview

To demonstrate the concept of cloud based file system we present Raindrop File System (RFS) [1] which provides device aware cache management that supports different networks, different costs and energy- awareness. It provides multiple levels of security and privacy policies. RFS tracks the usage of previous users so that the storage server can predict the new user's access pattern and apply server pre push optimization. Hence the speed of file delivery increases. As present day's mobile devices have large code and datasets, RFS implements partial caching of files. In RFS, client portion is implemented in Linux kernel and server is integrated with Amazon S3 cloud storage to reduce the complexity of storing the data at server. [1]

Objectives of RFS

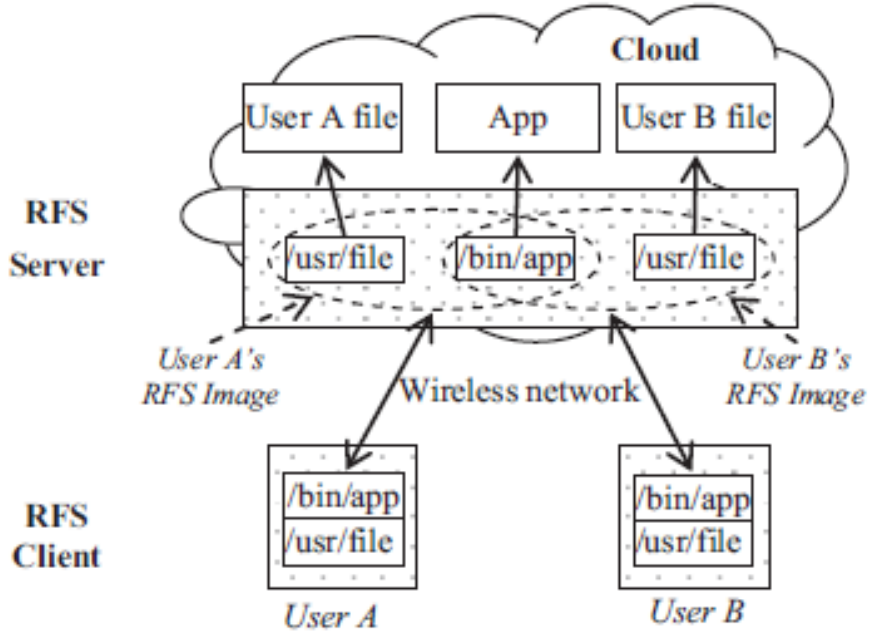
While designing RFS as a wireless network file system, following objectives must be addressed:

- Network may be accessed in multiple ways such as free connectivity like WiFi /Bluetooth or using operator cost structures
- Assuming that a single user uses the mobile device, privacy of the user must be retained
- Based on user's data access pattern, mostly accessed data can be pre pushed to server and cached on-demand

Design goals of RFS

The device usage characteristics, limitations in battery and network connectivity must be considered. RFS must provide mobile friendly system that connects mobile clients with the cloud storage without being always connected. In general, there are three underlying principles for RFS as below:

- Cloud Storage abstraction



In this client server model which is followed by RFS, the user is unaware of the files present in the cloud. Whenever a user requests for a file, it just interacts with the server. So a user need not directly depend on the cloud. RFS server is responsible for mapping requested files to the cloud.

Current file systems are not designed to address small-file traffic, so to provide server-scalability is the key role of RFS.

As users are mobile, the file system should be made which must be compatible to all the networks when a user moves from one network to another. Currently, HTTP is supported by all the networks. So RFS uses HTTP between client and server.

- Connection Optimization: Connection optimization requires to serve: demand fetching (retrieving file and storing it on local storage), synchronization and maintaining consistency amongst multiple clients.
- Client data protection: Mobile users are always concerned about privacy of their data. Therefore, RFS provides mobile clients an option to keep their data encrypted. So, the

public cloud will work as a transparent storage server which does not know anything about user's private data.

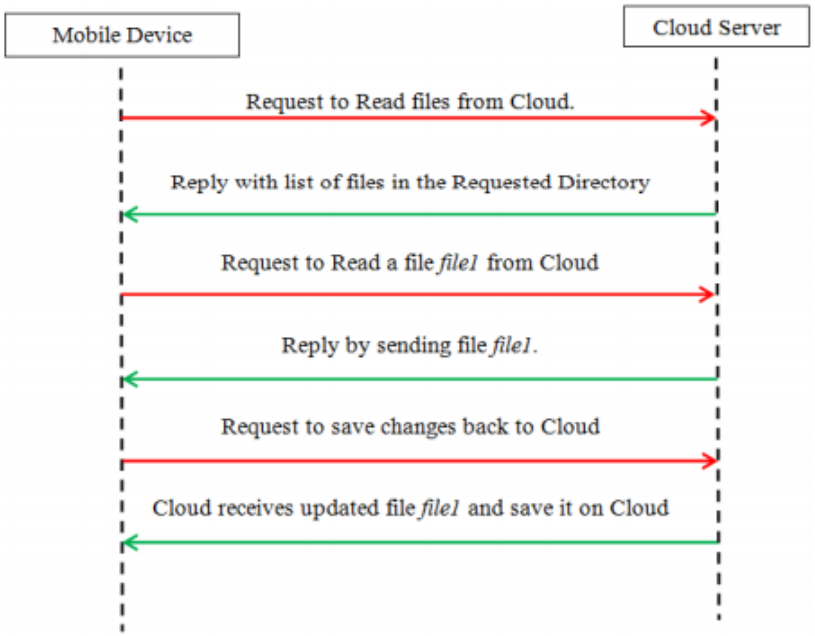
RFS is designed in a way that data security and privacy is controlled by client.

However, when data security is client-centric, it becomes difficult to share the resource over the network. Thus, encryption should be placed on the files. User might now will to send the key over a network. Now, when a user sends data to other user, we need a system which provides access to only those users who are given access.

Thus, a solution is determined in the article of RFS which uses multiple keys (public and private key) to transfer files with encryption. Here, the public key is shared between the users while a private key is kept secret to the sender. Using an algorithm, the receiver will decode the private key and can get access to the file.

System flow

The general flow [3] of the Cloud based file system is:



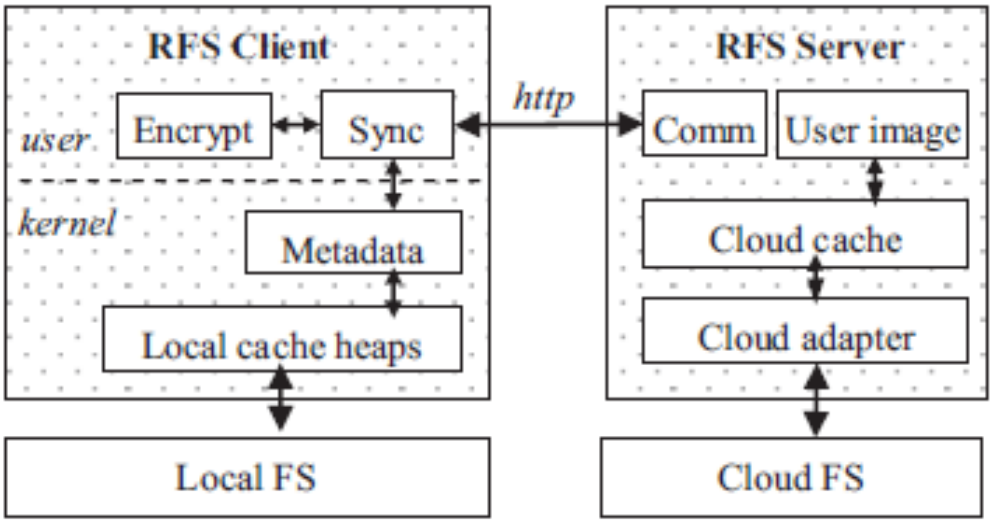
Each instance of mobile application requests a file to the cloud server. On receiving the request, it processes the request and returns the desired information to the mobile device.

Sequence of events

1. Request to read the file: This is the first step in mobile-cloud interaction. Here, user is prompted to provide directory path that he/she wants to access.

2. Reply with List of Files in the Requested Directory: In response to the submitted path, the cloud application will process the request and send the list of files available in the directory.
3. Request to Read a File from Cloud: On receiving the list of files, user selects the files which he requires.
4. Reply by Sending Requested File: On receiving the request, cloud processes the request and sends the requested file which a user accesses it.
5. Request to Save Changes Back to Cloud: On making necessary changes, the user saves the file which is sent along with the updated version of the file.

RFS Architecture



The above figure depicts the RFS client server architecture. Client and server interacts using HTTP.

RFS client

RFS client is placed on the top of the local file system. It includes four components:

- synchronization daemon which synchronises transactions between client and server and maintains consistency.
- an encryption engine: It provides client-side encryption and decryption.

- metadata manager: Keeps track of all the files stored locally and allows access to the synchronisation daemon for keeping data consistent.
- local cache storage

RFS server

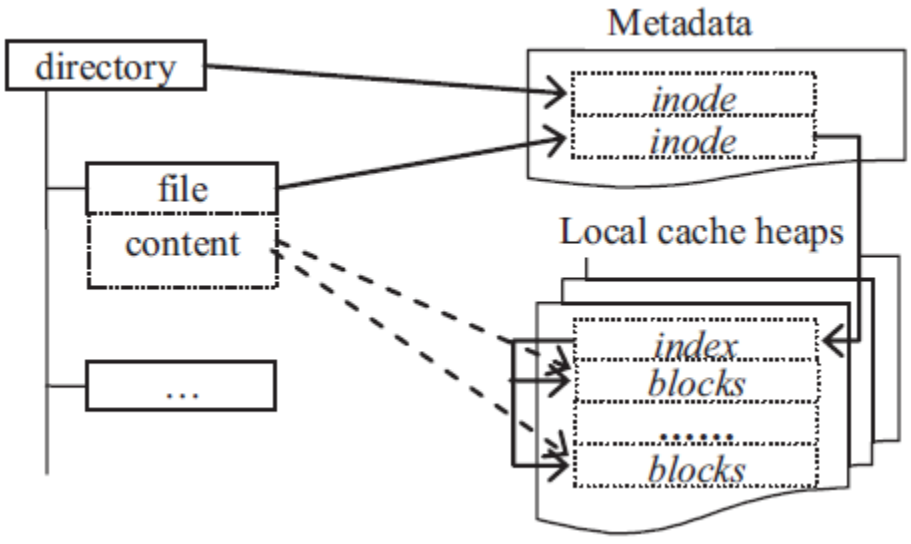
The foundation for RFS server is the underlying cloud storage system which includes four services:

- user images: It manages RFS image for each RFS user
- communications: It enables web service communication using REST services
- cloud caching: It provides a local cache for all RFS users
- cloud adapter: It provides the API corresponding to different cloud storage systems

RFS implementation

RFS client is implemented on Linux kernel while RFS server is built on Amazon S3 cloud storage system. In addition to provide cloud storage system, RFS uses HTTP 1.1 and SOAP web application protocols and zlib compression algorithm to compress data transmitted over network. RFS provides a facility to map local files stored on the client machine with the RFS directory. By using partial file caching, it uses a metadata file for storing inodes which has a block area and an index area.

All inodes are referred to their index where index area represents location of file block.



RFS integration with the cloud:

RFS server stores the metadata on distributed network database like Amazon's RDS whereas file data is stored on an API defined by an FTP server. This API has three functions: get, put and delete, to retrieve, store and remove the given file from the server respectively. RFS enforces Advanced Encryption Standard (AES) algorithm to encrypt/decrypt data files stored in cloud storage system thus ensuring data privacy. Instead of client pre-fetching, RFS applies server pre-pushing which is based on sending numerous blocks to the client based on the user's data access pattern.

Performance of RFS over other file systems:

After analysing different file systems and executing them on Wired, WiFi and WCDMA, it was found that RFS achieved better performance than FScache and Coda. [1]

On reviewing different papers we found that RFS was implemented on mobile devices having different operating systems and over different kinds of network and concluded that RFS achieved better performance as compared to other file systems.

Different benchmarks were set to measure performance over each network connection. These benchmarks include file input output operations. [1]

- Grep: Grep the text "test" in a reduced Android system that is mounted over a network file systems.
- Copying: Copying android system from local to remote file system.
- Copyout: Copy android system from network to local file system.
- Install : Installing android system in the network file system

Different modes were applied such as cold mode and warm mode.

Cold mode is used to measure performance of the file system that have to fetch data from the remote server. It is observed that Coda has low performance due to high network connectivity.

While in warm mode, performance of the file system that have to fetch data from local machine is calculated. Due to better synchronisation offered by RFS, it outperforms the other file systems.

Below figure shows detailed analysis of performance for each FS, Grep, copying, copy out and install. [1]

Mode	Network	FS	Grep	Copyin	Copyout	Install
Cold	Wired	*FScache*	*11.63*	*80.61*	*11.58*	*49.83*
		Coda	*34.65*	*13.16*	*39.26*	*17.03*
		RFS	***10.58***	***6.01***	***15.59***	***12.21***
	WiFi	*FScache*	*41.65*	*130.91*	*47.78*	*168.22*
		Coda	*63.95*	*13.40*	*63.98*	*32.00*
		RFS	***50.50***	***6.92***	***49.49***	***35.44***
	WCDMA	***RFS***	***2710***	***7.01***	***4217***	***1693***
Warm	Wired	*FScache*	*1.23*	*74.99*	*1.78*	*48.89*
		Coda	*0.65*	*5.42*	*0.76*	*7.92*
		RFS	***0.27***	***2.05***	***1.01***	***4.35***
	WiFi	*FScache*	*3.27*	*105.71*	*4.26*	*145.28*
		Coda	*0.58*	*5.69*	*0.86*	*8.04*
		RFS	***0.28***	***1.82***	***1.23***	***4.40***
	WCDMA	***RFS***	***0.41***	***2.01***	***1.20***	***4.01***

From the above results, it was determined that RFS is suitable to run over 3G networks which no other file system offers.

Alternate methods to RFS

Until now external memory cards were used for backing up files, uploading media and transferring data. Due to increase in the quality of images and files, there is need to increase the storage size. By backing up files in external memory card, a user can protect his or her files as the data will be stored in the device and not online.

In order to transfer files from one device to another, a user just needs to store in the external memory card and insert it into other device. This method is a secured way to transfer files as nothing is available online and it does not require any extra protection to data. While in online transfer, we need to make sure if the data is not getting forged while transferring files.

Conclusion

RFS provides better file system performance and provides seamless user experience for the mobile users. It achieves comparable performance with CODA and at the same time facilitates minimum overhead for data privacy over wireless 3G networks.

Report Contributions

Heli Shah:

- Studied various file systems and discussed with the project partner
- Analysed the system flow of client-server system.
- Analysed RFS architecture and role of each component used in it.
- Analysed the performance of RFS over FScache and Coda.
- Documentation, Editing and review of final report with the project partner

Dhwani Patel:

- Studied various file systems and discussed with the project partner
- Analyzed the existing file systems & the business scenario for RFS and it's problem statement
- Analysed the RFS design goals and its implementation
- Documentation, Editing and review of final report with the project partner

References:

[1] Yuan Dong, Haiyang Zhu, Jinzhan Peng, Fang Wang, Michael P. Mesnier, Dawei Wang, Sun C. Chan. Raindrop File System – A Network File System for Mobile Devices and the Cloud, http://dl.acm.org/citation.cfm?id=1945036

[2] M. Satyanarayanan, J. J. Kistler, P. Kumar, M. E. Okasaki, E. H. Siegel, David, and C. Steere. Coda: A highly available file system for a distributed workstation environment. IEEE Transactions on Computers, 39:447–459, 1990.

[3] https://sdsudspace.calstate.edu/bitstream/handle/10211.10/3564/Kaur_Sukdeep.pdf?sequence=1

[4] http://www.cs.cmu.edu/~coda/docdir/scalable90.pdf

[5] https://en.wikipedia.org/wiki/Mobile_cloud_storage