1. **Introduction**

Cloud-based storageservices commonly serves millions of users with storage capacity for each user can reach to several gigabytes to terabytes of data. People use cloud storage for the daily demands, for example backing-up data, sharing file to their friends via social networks such as Face book, Zing Me. Users also probably upload data from many different types of devices such as computer, mobile phone or tablet. After that, they can download or share them to others. System load in cloud storage is usually really heavy.

Thus, to guarantee a good quality of service for users, the system has to face many difficult problems and requirements: Serving intensity data service for a large number of users without bottle-neck; Storing, retrieving and managing big-files in the system efficiently; Parallel and resumable uploading and downloading; Data de-duplication to reduce the waste of storage space caused by storing the same static data from different users. In traditional file systems, there are many challenges for service builder when managing a huge number of big file: How to scale system for the incredible growth of data; How to distribute data in a large number of nodes; How to replicate data for load-balancing and fault-tolerance; How to cache frequently accessed data for fast I/O, etc. A common method for solving these problems which is used in many Distributed File Systems and Cloud Storages is splitting big file to multiple smaller chunks, storing them on disks or distributed nodes and then managing them using a meta-data system . Storing chunks and meta-data efficiently and designing a lightweight meta-data are significant problems that cloud storage providers have to face. After a long time of investigating, realized that current cloud storage services have a complex meta-data system; at least the size of metadata is linear to the file size for every file. Therefore, the space complexity of these meta-data system is *O*(*n*) and it is not well scalable for big-file. Propose new big-file cloud storage architecture and a better solution to reduce the space complexity of meta-data.

BFC Native App BFC Web Interface

Application Layer

Workers

BFC API

Storage Logic Layer

Metadata Storage Service File Chunk Storage Service

Object Store Layer

Distributed Key value Store

KV

KV

Persistent Layer

KV

KV

KV

**Figure 1:** BFC Architecture

**Key-Value** stores have many advantages for storing data in data-intensity services. They often outperform traditional relational databases in the ability of heavy load and large-scale systems. In recent years, key-value stores have an unprecedented growth in both academic and industrial field. They have low-latency response time and good scalability with small and medium key-value pair size. Current key-value stores are not designed for directly storing big-values, or big file in our case. Executed several experiments in which put whole file-data to key-value store, the system did not have good performance as usual for many reasons: firstly, the latency of put/get operation for big-values is high, thus it affects other concurrent operations of key-value store service and multiple parallel accesses to different value reach limited. Secondly, when the value is big, there is no more space to cache another objects in main memory for fast access operations. Finally, it is difficult to scale-out system when number of users and data increase. This research is implemented to solve those problems when storing big-values or big-file using key-value stores. It brings many advantages of key-value store in data management to design a cloud-storage system called Big File Cloud (BFC). These are our contributions in this research: – Propose a light-weight meta-data design for big file. very file has nearly the same size of meta-data. BFC has *O*(1) space complexity of meta-data of a file, while size of meta-data of a file in Dropbox[1], HDFS[4] has space complexity of *O*(*n*) where *n* is size of original file. See Fig 9 – Propose a logical contiguous chunk-id of chunk collection of files. Those make it easier to distribute data and scale-out the storage system. – Bring the advantages of key-value store into big-file data store which is not default supported for big-value.ZDB[16] is used for supporting sequential write, small memory-index overhead. These contributions are implemented and evaluated in Big File Cloud (BFC) that serve storage for Zing Me Users. Disk Image files of VNG’s CSM Boot diskless system are stored in Big File Cloud.

* 1. **Objective of the Project**

These days, cloud-based capacity administrations are quickly developing and turning into a rising pattern in information stockpiling field. There are numerous issues when planning a proficient stockpiling motor for cloud-based frameworks with a few prerequisites, for example, enormous document handling, lightweight meta-information, low inactivity, parallel I/O, de-duplication, circulated, high adaptability. Key-worth stores assumed an essential part and indicated numerous points of interest when taking care of those issues. In Big File Cloud (BFC) with its calculations and construction modeling to handle most of issues in a major document distributed storage framework in view of key value store.

It is finished by proposing low-confused, settled size meta-information outline, which backings quick and exceedingly simultaneous, dispersed record I/O, a few calculations for resumable transfer, download and basic information de-duplication technique for static information.

This examination connected the upsides of ZDB - an in-house key value store which was upgraded with auto-increase whole number keys for taking care of enormous document stockpiling issues proficiently. The outcomes can be utilized for building versatile appropriated information distributed storage that bolster huge document with size up to a few terabytes.