JES
Journal of Engineering Sciences

Vol 13, Issue 03, MARCH /2022

ISSN NO:0377-9254

# CHARON: A SECURE CLOUD - OF- CLOUDS SYSTEM FOR STORING AND SHARING BIG DATA

[1]CHEERLA MOUNIKA, [2]MURALI PONAGANTI
[1]MCA Student, [2]Assistant Professor
Department Of MCA
Sree Chaitanya College Of Engineering, Karimnagar

**Abstract :**

We present CHARON, a cloud-backed storage system capable of storing and sharing big data in a secure, reliable, and efficient way using multiple cloud providers and storage repositories to comply with the legal requirements of sensitive personal data. CHARON implements three distinguishing features: (1) it does not require trust on any single entity, (2) it does not require any client-managed server, and (3) it efficiently deals with large files over a set of geo-dispersed storage services. Besides that, we developed a novel Byzantine-resilient data-centric leasing protocol to avoid write-write conflicts between clients accessing shared repositories. We evaluate CHARON using micro and application-based benchmarks simulating representative workflows from bioinformatics, a prominent big data domain. The results show that our unique design is not only feasible but also presents an end-to-end performance of up to 2:5better than other cloud- backed solutions.

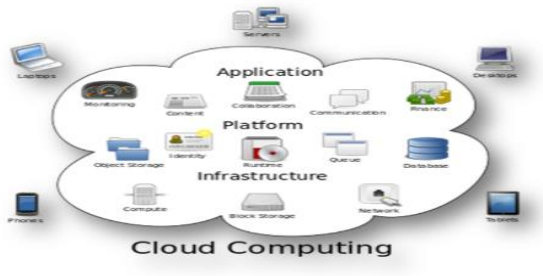
## I. INTRODUCTION

### 1.1 Introduction of the Project

**What is cloud computing?**

Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the common use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation. Cloud computing consists of hardware and software resources made available on the Internet as managed third-party services. These services typically provide access to advanced software applications and high- networks of server computers.

**How Cloud Computing Works?**




Structure of cloud computing

The goal of cloud computing is to apply traditional supercomputing, or high-performance computing power, normally used by military and research facilities, to perform tens of trillions of computations per second, in consumer-oriented applications such as financial portfolios, to deliver personalized information, to provide data storage or to power large, immersive computer games. The cloud computing uses networks of large groups of servers typically running low-cost consumer PC technology with specialized connections to spread data-processing chores across them. This shared IT infrastructure contains large pools of systems that are linked together. Often, virtualization techniques are used to maximize the power of cloud computing.

### 1.2 Characteristics and Services Models:

The salient characteristics of cloud computing based on the definitions provided by the National Institute of Standards and Terminology (NIST) are outlined below:

**On-demand self-service:** A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider.

**Broad network access:** Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).

**Resource pooling:** The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location-independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or data center). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.

**Rapid elasticity:** Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.

**Measured service:** Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be managed, controlled, and reported providing transparency for both the provider and consumer of the utilized service.

## II. LITERATURE SURVEY

### 2.1 Introduction

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength.Once these things are satisfied, ten next steps are to determine which operating System and Language can be used for developing the tool

**1) BYZANTINE DISK PAXOS: OPTIMAL RESILIENCE WITH BYZANTINE SHARED MEMORY**

We present Byzantine Disk Paxos, an asynchronous shared-memory consensus algorithm that uses a collection of n > 3t disks, t of which may fail by becoming non responsive or arbitrarily corrupted. We give two constructions of this algorithm; that is, we construct two different t-tolerant (i.e., tolerating up to t disk failures) building blocks, each of which can be used, along with a leader oracle, to solve consensus. One building block is a t-tolerant wait-free shared safe register. The second building block is a t-tolerant regular register that satisfies a weaker termination (liveness) condition than wait freedom: its write operations are wait-free, whereas its read operations are guaranteed to return only in executions with a finite number of writes. We call this termination condition finite writes (FW), and show that wait-free consensus is solvable with FW terminating registers and a leader oracle. We construct each of these ttolerant registers from n > 3t base registers, t of which can be non-responsive or Byzantine. All the previous t-tolerant wait-free constructions in this model used at least 4t + 1 fault-prone registers, and we are not familiar with any prior FW terminating constructions in this model.

**2) UNIDRIVE: SYNERGIZE MULTIPLE CONSUMER CLOUD STORAGE SERVICES**

Consumer cloud storage (CCS) services have become popular among users for storing and

synchronizing files via apps installed on their devices. A single CCS, however, has intrinsic limitations on networking performance, service reliability, and data security. To overcome these limitations, we present UniDrive, a CCS app that synergizes multiple CCSs (multi-cloud) by using only few simple public RESTful Web APIs. UniDrive follows a server-less, client-centric design, in which synchronization logic is purely implemented at client devices and all communication is conveyed through file upload and download operations. Strong consistency of the metadata is guaranteed via a quorum-based distributed mutual-exclusive lock mechanism. UniDrive improves reliability and security by judiciously distributing erasure coded files across multiple CCSs. To boost networking performance, UniDrive leverages all available clouds to maximize parallel transfer opportunities, but the key insight behind is the concept of data block over-provisioning and dynamic scheduling. This suite of techniques masks the diversified and varying network conditions of the underlying clouds, and exploits more the faster clouds via a simple yet effective in-channel probing scheme. Extensive experimental results on the global Amazon EC2 platform and a real-world trial by 272 users confirmed significantly superior and consistent sync performance of UniDrive over any single CCS

## III. Existing System

Byzantine disk Paxos [26] is a consensus protocol built on top of untrusted shared disks. More recently, an enhanced version of this protocol specifically designed to use file synchronization services (e.g., DropBox, Google Drive) instead of disks was published [21]. These algorithms could be used to implement mutual exclusion satisfying deadlock- freedom (a stronger liveness guarantee than obstruction-freedom). However, these solutions would require a much larger number of cloud accesses. Our lease protocol, on the other hand, requires only two to four cloud accesses for acquiring a lease.

To the best of our knowledge, there are only two fault-tolerant data-centric lease algorithms in the literature [15], [39]. The lease algorithm of Chockler and Malkhi [39] has two important differences when compared with CHARON's BFT composite lease. First, it does not provide an always-safe lease as it admits the existence of more than one process with valid leases. Second, it tolerates only crashes, requiring thus some trust on individual cloud providers. The BFT mutual exclusion algorithm from DepSky [15] is a natural candidate to regulate access contention in CHARON. However, our composite lease algorithm is 410 faster than DepSky's (see x5.2), does not require clients to have synchronized clocks, and neither rely on weakly-consistent operations such as object storage' list.

Systems like Hybris [23], SCFS [24] and RockFS [70] employ a hybrid approach in which unmodified cloud storage services are used together with few computing nodes to store metadata and coordinate data access. The main limitation of these systems is that they require servers deployed in the cloud providers, which implies additional costs and management complexity. The same limitation applies to modern (single-provider) geo- replicated storage systems such as Spanner [71], SPANStore [25] and Pileus [72], if deployed in multiple clouds.

A slightly different kind of work proposes the aggregation of multiple file synchronization services (e.g., DropBox, Box, Google Drive) in a single dependable service [20], [21], [22]. CYRUS [20] does not implement any kind of concurrency control, allowing different clients to create different versions of files accessed concurrently.

## IV. Proposed System

The system proposes a CHARON which is a distributed file system that provides a near-POSIX interface to access an ecosystem of multiple cloud services and allows data transfer between clients. The preference for a POSIX interface rather than using data objects resorts to the fact the envisioned users are likely to be non-experts, and existent life sciences tools use files as their input most of the times. In particular, the system needs to (1) efficiently deal with multiple storagee locations, (2) support reasonably big files, and (3) offer controlled file sharing. These

challenges are amplified by our goals of excluding user-deployed servers and of requiring no modifications to existing cloud services (for immediate deployability).

All techniques used in CHARON were combined considering two important design decisions. First, the system absorbs file writes in the client's local disk and, in the background, uploads them to their storage location. Similarly, prefetching and parallel downloads are widely employed for accelerating reads. This improves the usability of CHARON since transferring large files to/from the clouds take significant time (see x5). Second, the system avoids write-write conflicts, ruling out any optimistic mechanism that relies on users/applications for conflict resolution.

The expected size of the files and the envisioned users justify this decision. More specifically, (1) solving conflicts manually in big files can be hard and time-consuming; (2) users are likely to be non-experts, normally unaware of how to repair such conflicts; and (3) the cost of maintaining duplicate copies of big files can be significant. For instance, collaborative repositories, such as the Google Genomics [31], require such control since they allow users to read data about available samples, process them, and aggregate novel knowledge on them by sharing the resulting derived data into the bucket containing the sample of interest.

## V. MODULE DESCRIPTION

### Model and Guarantees

In this section, we refer to cloud services as base objects. A client may invoke, on the same base object, several parallel operations that are executed in FIFO order. Since a lease implies timing guarantees, an upper bound interval is assumed for the message transmission between clients and base objects. However, this assumption is required only for liveness since safety is always guaranteed. Our algorithm ensures that at most one correct client at a time can access the shared resource and only for a limited duration (the lease term [37]). Each resource provides three lease-related operations: lease(T) and renew(T) acquires and extends the lease for T time units, respectively, while release() ends the lease. These operations satisfy the following properties:

Mutual Exclusion (safety): There are never two correct clients with a valid lease for the same resource.

Obstruction-freedom (liveness): A correct client that tries to lease a resource without contention will succeed.

Time-boundedness (liveness): A correct client that acquires a lease will hold it for at most T time units unless the lease was renewed.

### Metadata Organization

Metadata is the set of attributes assigned to a file/directory (e.g., name, permissions). Independently of the location of the data chunks, CHARON stores all metadata in the cloud-of-clouds using single-writer multi-reader registers to improve their accessibility and availability guarantees. we redesigned and optimized the SWMR register implementation of DepSky [15] to improve the performance and concurrency

### Data Management

CHARON uses the local disk to cache the most recent files used by clients. Moreover, it also keeps a fixed small main-memory cache to improve data accesses over open files. Both of these caches implement least recently used (LRU) policies. The use of caches not only improves performance but also decreases the operational cost of the system. This happens because cloud providers charge data downloads, but usually uploads are free as an incentive to send data to their facilities. Managing large files in cloud-backed file systems brings two main challenges. First, reading (resp. writing) whole (big) files from the cloud is impractical due to the high downloading (resp. uploading) latency [24]. Second, big files might not fit in the (memory) cache employed in cloud-backed file systems for ensuring usable performance. CHARON addresses these challenges by splitting(large) files into fixed-size chunks of 16MB, which results in blocks with a few megabytes after compression and erasure codes.

This small size has been reported as having a good tradeoff between latency and throughput

**Cloud-backed Access Control**

CHARON implements a security model where the owner of the file pays for its storage and is able to define its permissions. This means that each client pays for all private data and all the shared data associated with the shared folders he created (independently on who wrote it). CHARON clients are not required to be trusted since access control is performed by the cloud providers, which enforce the permissions for each object.

## VI. RESULTS


FIGURE 8.3 THRESHOLD DETAILS

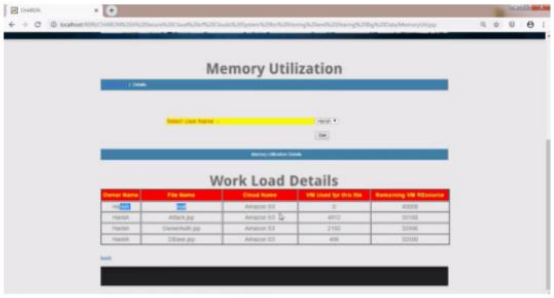



FIGURE 8.4 WORK LOAD DETAILS

## VII .CONCLUSION

CHARON is a cloud-backed file system for storing and sharing big data. Its design relies on two important principles: files metadata and data are stored in multiple clouds, without requiring trust on any of them individually, and the system is completely data centric. This design has led us to develop a novel Byzantineresilient leasing protocol to avoid write-write conflicts without any custom server. Our results show that this design is feasible and can be employed in real-world institutions that need to store and share large critical datasets in a controlled way.

**References :**

1.Adriano Serckumecka, Ibéria Medeiros, Bernardo Ferreira, Alysson Bessani, "A Cost-Effective Cloud Event Archival for SIEMs", Reliable Distributed Systems Workshops (SRDSW) 2019 38th International Symposium on, pp. 31-36, 2019.

2.Vinicius Cogo, Alysson Bessani, "Enabling the Efficient Dependable Cloud-Based Storage of Human Genomes", Reliable Distributed Systems Workshops (SRDSW) 2019 38th International Symposium on, pp. 19-24, 2019.

3.Adriano Serckumecka, Ibéria Medeiros, Bernardo Ferreira, Alysson Bessani, "SLICER: Safe Long-Term Cloud Event Archival", Dependable Computing (PRDC) 2019 IEEE 24th Pacific Rim International Symposium on, pp. 43-4309, 2019.

4.Mengtian Xu, Guorui Feng, Yanli Ren, Xinpeng Zhang, "On Cloud Storage Optimization of Blockchain With a Clustering-Based Genetic Algorithm", Internet of Things Journal IEEE, vol. 7, no. 9, pp. 8547-8558, 2020.