

## An architecture for synchronising cloud file storage and organisation repositories

Gil Andriani, Eduardo Godoy, Guilherme Koslovski, Rafael Obelheiro & Mauricio Pillon

To cite this article: Gil Andriani, Eduardo Godoy, Guilherme Koslovski, Rafael Obelheiro & Mauricio Pillon (2018): An architecture for synchronising cloud file storage and organisation repositories, International Journal of Parallel, Emergent and Distributed Systems, DOI: [10.1080/17445760.2017.1422500](https://doi.org/10.1080/17445760.2017.1422500)

To link to this article: <https://doi.org/10.1080/17445760.2017.1422500>



Published online: 09 Jan 2018.



Submit your article to this journal [↗](#)





View related articles [↗](#)



View Crossmark data [↗](#)



# An architecture for synchronising cloud file storage and organisation repositories

Gil Andriani, Eduardo Godoy, Guilherme Koslovski , Rafael Obelheiro and Mauricio Pillon 

Graduate Program in Applied Computing, Department of Computer Science, Santa Catarina State University, Joinville, Brazil

## ABSTRACT

Cloud computing providers have disseminated dynamic storage provisioning delivered to end users as on-demand services. Although cloud file storage and sharing has become popular among home users, the access requirements, performance expectations and usage characteristics are different for organisations, and were not originally considered by popular applications and tools for synchronising files between cloud providers and local repositories. Moreover, multisite organisations traditionally have legacy file storage and wide-area networking solutions to support their business systems. Typically, the file repositories are replicated between sites using private communication links. The combination of legacy storage solutions interconnected through private links with cloud-based file storage is a challenging task. In this context, this paper introduces Cloud4NetOrg, a client architecture for cloud file storage and multisite repository synchronisation. We implemented prototypes of this architecture that interact with two popular cloud file services (DropBox and OneDrive), and the experimental results indicate a promising application in collaborative environments with several LANs. Indeed, Cloud4NetOrg decreases the synchronisation time and the total data transferred from/to cloud repositories by using the organisation repositories as a hierarchical cache system.

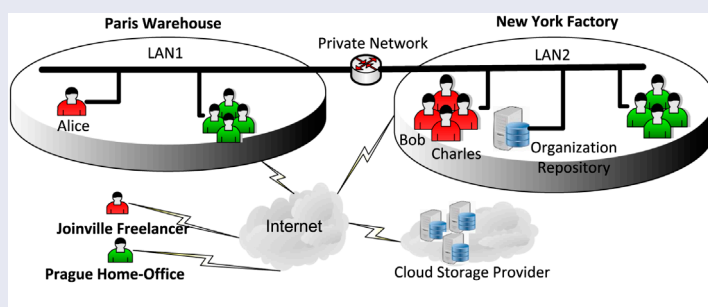
## ARTICLE HISTORY

Received 11 April 2017

Accepted 27 December 2017

## KEYWORDS

Cloud storage; file synchronisation; legacy storage devices; synchronisation client; Cloud4NetOrg



Cloud4NetOrg is proposed for geographically distributed organisations composed of dynamic and temporary collaborative groups. The interaction between employees is based on file sharing. Commonly, sites are interconnected by a private network and have an internal data storage repository. A single site can have multiple subnetworks to interconnect the collaborative groups. In addition, home-office users collaborate through the Internet, usually using size-limited storage devices.

## 1. Introduction

Cloud computing revolutionised the use of processing, communication and storage resources, delivering such functionalities to users as on-demand services [1]. The adoption of cloud storage is a reality for home users: storage, editing, and file sharing were popularised by disseminating clients for file synchronisation (e.g. DropBox [2], Google Drive [3] and OneDrive [4]) across local devices and hosted cloud repositories. However, these synchronisation tools are designed with a focus on home users with simple access and sharing requirements.

For instance, users on academic campuses have repositories with an average size of 4.23 GB, and most users tend to store many files (over 1000), including users with over 20,000 stored files [5]. Concerning dynamic updates, it was observed that about 82% of update operations carry up to 1 MB, featuring a concentration of use on small files. Still, most of the connection sessions between clients and providers (85% of the sessions) have no significant file transfers, while file sharing between groups with large numbers of users is not a reality for academic settings [5–7]. Although popular, the file synchronisation tools developed for home users do not fully meet the specific requirements of a major part of the enterprise spectrum. With regarding the file accessing, usually in organisations are built dynamic and temporary groups composed of collaborative users spontaneously formed according to professional interests and needs. The collaborative users tend to be concentrated in private networks accessing legacy storage systems. Eventually, the local repositories are synchronised atop private networks. Moreover, collaborative users have more simultaneously shared files compared to home users. Consequently, synchronisation generates higher network traffic on each client, and latency becomes a critical factor. Even though these users are mostly in private networks, existing cloud synchronisation clients ignore the interaction with local repositories. Recently, synchronisation tools and services focusing on enterprises were launched [8,9] partially addressing the enterprise challenges. However, the full integration of cloud-based file storage with multisite organisations is an open research field [10,11].

Cloud synchronisation tools keep files synchronised with the cloud storage service by periodically copying all changed remote data to user's devices and vice-versa. This is not a limiting factor for home users as usually the capacity of virtual space (the cloud space quota) is similar or higher than the storage space on their personal devices. In other words, the data volume is not a critical factor. In short, the cloud synchronisation tools are upper-bounded by the local storage capacity as cloud providers deliver the illusion of infinite remote storage space.

Organisation's devices have limited repository space, and consequently the volume of data stored on the cloud exceeds the storage capacity of local workstations [12]. In this sense, a recent proposal introduced the selective file synchronisation given to users the freedom for selecting which files must be periodically synchronised [8]. Unselected files are still indexed and on-demand available.

The upfront investment for acquiring data storage equipment declined in recent years motivating the implementation of local solutions at different scales of capacity. Indeed, the private data repositories available in organisations have higher storage capacity compared to the local disk capacity of workstations. Instead of replacing the consolidated storage mechanisms by cloud storage, *we claim that such local storage solutions can be used for composing a caching layer between cloud storage services and synchronisation tools*. Moreover, collaborative users can select only files they need, when they need them, for being automatically synchronised. Although simple, the cache architecture can soften the access latency for recurrence used files.

The ubiquitous access to files over the Internet and the concentration of professionals in one place are conflicting factors. On the one hand, the possibility of accessing data from anywhere allows greater interaction and collaboration between teams; on the other hand, the concentration of professionals in one place accessing the same volume of data in the cloud can overload the Internet link, consequently increasing latency and networking costs [13]. In this sense, a second characteristic of multisite organisations that can be explored for optimization is the existence of private network links between sites. This solution offers features such as high availability and symmetric download

and upload speed. Consequently, the dedicated interconnection can attenuate the performance variability observed in the traditional Internet, and the resulting networking stability is beneficial for TCP connections supporting the synchronisation tools. *Our second claim is that the existing private network links can be used for synchronising files, thereby reducing Internet link usage.*

In short, we believe that consolidated network and storage services and tools should work in parallel with cloud storage. In this context, the main contribution of this paper is the definition of a client architecture for synchronising legacy repositories of geographically-distributed organisations and cloud file storage services, termed Cloud Storage for Network-based Organisation File System (Cloud4NetOrg). This architecture<sup>1</sup> has been validated through the implementation and experimental evaluation of two prototypes that interact with well-known cloud storage services (DropBox and OneDrive). The results highlight that an organisation can reap significant benefits from using existing local storage repositories to intermediate access to remote cloud storage. Moreover, the cloud provider also enjoys reduced network and server load when files are directly retrieved from local caches.

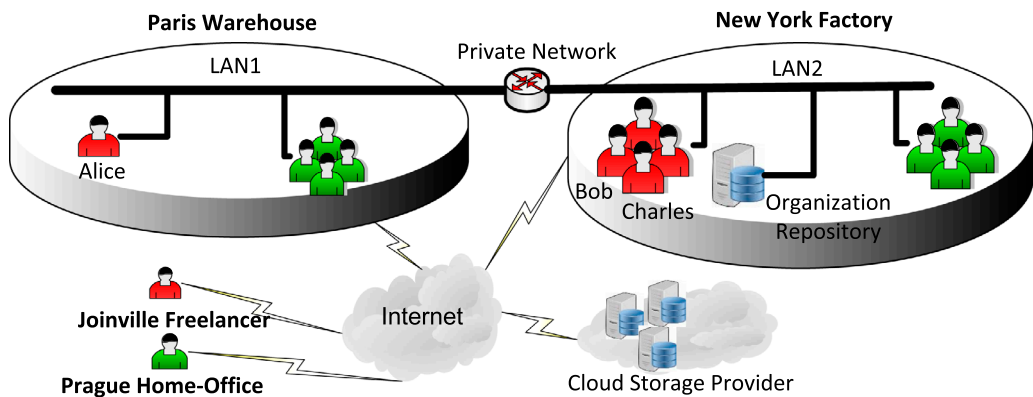
The remainder of this paper is organised as follows: Section 2 reviews the concepts and issues that motivate our research, while Section 3 discusses related work. The proposed architecture is presented in Section 4. Experimental analysis is discussed in Section 5 while conclusion and perspectives are presented in Section 6.

## 2. Concepts, issues and motivations

Cloud-based file storage comprises two main actors: data storage repositories and synchronisation clients. The internal architecture of repositories is usually composed of modules for monitoring and indexing data, implemented as a distributed file system, focusing on delivering availability, authenticity, and confidentiality to users. Complementarily, cloud providers offer services to control versioning, quota sharing and collaborative editing. For their part, file synchronisation clients are responsible for maintaining a directory structure and files synchronised with the cloud repository. File transfer events are realised by synchronisers and observers, that act on the metadata associated with the files. There are two observers, locally- and remotely-placed, responsible for identifying changes in files generated by sharing users. When needed, a synchronising module performs the transfer of files (partial or total) between local and remote repositories. Some common features are found in popular clients such as incremental synchronisation, encryption and data compression.

Cloud-based file synchronisation clients are widespread among home users, satisfactorily serving their needs related to easy and on-demand access. The content stored in the cloud is synchronised with local devices bounded by the minimum storage space between source and destination directories. Usually, this limitation is perceived at user's work devices (e.g. workstation, smartphones, laptops). Local storage devices can have available capacity close to hundred gigabytes while cloud storage can offer terabytes of storage capacity [15]. Although, conceptually speaking, a selective and partial repository synchronisation [8] is not in accordance with cloud principles [16] (some applications allows the selection of which files and directories must be synchronised with user's devices). However, configuring and managing a hierarchy of synchronised directories is a tricky task, which places the burden of complexity on the user.

Recently, cloud storage providers have started offering specific services to organisations, increasing storage capacity and service availability [8,17]. However, such services fall short of meeting the expectations of enterprise environments. Usually, such organisations are composed of multiple users collaborating on common projects, often placed in geographically-distributed environments interconnected by private networks. Figure 1 depicts a scenario where an organisation is composed of two groups, identified by green and red, distributed across New York, Paris, Joinville, and Prague. This organisation has two physical offices at New York and Paris which concentrate the largest number of employees. Both sites are interconnected by a private network and have an internal data storage repository. Complementary, two users are remotely collaborating through the Internet, placed at Joinville and Prague.



**Figure 1.** A common scenario of collaborative users from a multisite organisation. Home-office users (Joinville and Prague) interact with users concentrated at Paris and New York. A private network is available for interconnecting data repositories.

Storage systems based on the Network File System (NFS) protocol [18], Common Internet File System (CIFS) standard effort [19] and Server Message Block (SMB) protocol [20] are largely used by enterprise environments. Such solutions are useful for servers replicated through virtual private networks. While the storage systems can be deployed in Paris and New York, however, collaborators in Joinville and Prague must use a different approach for sharing files, even if a virtual private network (VPN) over the Internet is available. Although a VPN can offer a private communication channel, the traffic is routed atop the Internet and quality-of-service guarantees are not provided (differently from a private network between Paris and New York). Summing up, the packet losses currently observed on Internet affect the performance of TCP-congestion control algorithm [21] propagating a performance degradation to final applications. Consequently, the synchronisation time is increased, mainly for small files. Moreover, for collaborators within a single site (for instance, users at the Paris and New York facilities in Figure 1), the Internet access link can become a bottleneck that degrades the perceived quality-of-experience of file synchronisation with the cloud.

A workaround for the users in Joinville and Prague is to retrieve files directly from cloud repositories. Even communicating over the Internet, commercial cloud-based mechanisms use content-delivery networks to approximate services and users, alleviating the quality-of-service obstacles [22,23]. In short, with cloud-based approaches, the networking bottleneck (latency and/or bandwidth) is observed on the users access-points, while with VPN it is present at both connection end-points (enterprise VPN server and collaborators). Thus, remotely-located users must use cloud storage services.

Regarding the race conditions when multiple users are concurrently accessing the same files, the popular synchronisation tools follow a well-known design principle, called the end-to-end argument [24], which claims that only the data consuming actor (in this case, the user) can decide which file version is the correct one. Thus, when the same file is written simultaneously by several users, synchronisation systems just create multiple versions of the file, and later a user can decide which one should prevail.

To summarise, the current architecture of synchronisation tools, allied with geographically clustered collaborators, imposes a barrier on cloud storage adoption by enterprises [25]. The use of synchronisation applications developed for home and small-scale scenarios is not in accordance with business and organisational requirements [8]. In short, two main architectural barriers are identified on available popular solutions: (i) lack of integration with private and legacy storage systems; (ii) synchronisation latency in interactions with cloud servers.

### 3. Related work

Data and file synchronisation between local and remote repositories have received attention from academic and industrial communities. Specifically considering the interaction with cloud repositories, BlueSky [10] and SCFS [11] share similar objectives with Cloud4NetOrg. The former provides a networked file system based on cloud storage (natively integrated with Amazon S3 and Windows Azure) which supports multiple protocols such as NFS and CIFS. The latter provides access to cloud storage services through a near-POSIX interface with internal design based on FUSE-J [26].

Regarding commercial tools for synchronising cloud and local repositories, Nasuni [27], Twin-Strata [28], Panzura [29], and StorSimple [30] are highlighted. Nasuni and Panzura offer virtual network-attached storage (NAS) appliance services and CIFS/NFS gateways, respectively. BlueSky, Nasuni and Panzura fulfill the requirements of geographically concentrated organisations, but ignore the existence of home-office collaborators, an expanding practice today, as well as multisite organisations. The internal design of neither mechanism is disclosed, and consequently a detailed comparison is unfeasible.

Cloud storage providers deliver high availability and scalable on-demand file-systems services, such as GFS [31], S3FS [32], CSTORE [33], and Hadoop [34]. However, the application programming interface (API) calls and implementation do not adhere to POSIX standards, making it hard to integrate them with the legacy storage systems available on organisations. Indeed, administrators have to choose between maintaining legacy solutions or increasing the functionality that comes with file systems linked to cloud computing.

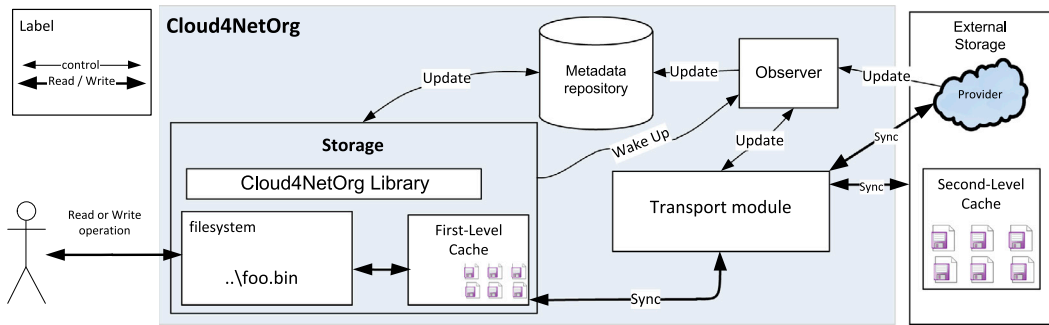
The popular applications for file synchronisation across cloud storage and local devices spread the use of cloud computing in the non-technical community. DropBox, OneDrive and Google Drive apps keep local files synchronised with cloud storage requiring minimum administrative overhead. In short, these clients perform a local copy of all shared files. This approach is convenient for home users usually connected through a non-bottleneck Internet access point. For these users, the storage capacity restriction is placed on cloud side (the available remote capacity, for instance). Dropbox Smart Sync innovates by splitting metadata from data synchronisation [8]. With a Smart Sync-aware client, all files information and team folders summaries are displayed (the metadata) even when files are not locally-available. In short, the key goal of Smart Sync is to save local-storage space. However, the user is in charge of selecting which files must be synchronised (a default setting can be configured for new files and folders). Cloud4NetOrg shares the same view regarding the parameterized synchronisation and innovates by using consolidated storage systems as local caches. Moreover, local caches are promising approach for collaborative teams with different Internet access, as discussed in Section 4.

Cache techniques are commonly applied in distributed systems to speed up data access. For instance, the Coda file system [35] supports disconnected operations and relies on a callback cache consistency protocol, keeping only the latest version of the file and propagating the modifications in parallel. In this sense, Cloud4NetOrg supports disconnected read and write operations. In case of conflict, Coda invalidates the caches, while Cloud4NetOrg, like other cloud-based storage tools, maintains all versions of the given file.

Considering cloud-based storage, some mechanisms use different levels of cache techniques [10, 11, 36]. For instance, BlueSky [10] implements a single level of cache using the write-back coherence approach, while SCFS [11] has two levels of cache, the first on disk (hundreds of GB) and the second on memory (hundreds of MB). The least recently used (LRU) algorithm is applied for data replacement.

Coral [36] introduces a new data layout and cache management scheme for cloud-based storage. The experimental analysis showed promising results when compared to BlueSky (lower access time and cost per GB of storage). However, Coral is mostly concerned with data blocks instead of files, which precludes its integration with legacy file repositories. Cloud4NetOrg implements a configurable two-level cache based on a networked file system, as presented in the next section.





**Figure 2.** Cloud4NetOrg architecture.

Some synchronisation clients can simultaneously use multiple providers [37–39], implementing additional service support, fragmentation to multiple repositories and decentralised index. However, synchronisation with local repositories (caches) and collaborative team work are ignored.

Recent studies on data traffic characterization contemplating home [5,7,40,41] and academic users [7] indicate that the traffic for file synchronisation reached 1/3 of the traffic for on-demand video streaming in the scenarios analysed. These works characterise the traffic profile by measuring data volume, average file size, and synchronisation activities (management traffic). Moreover, they offer a perspective on home-placed cloud users which guides the present proposal. However, no attention was directed to multi-homed organisations with collaborative users. In these environments, flexibility and heterogeneity are essential characteristics for a file storage system [18–20,42], while scalability, high availability and security are desirable properties [42]. The present proposal, Cloud4NetOrg, meets the essential characteristics, and also helps with scalability and high availability by combining three key aspects: local disk storage, network-based storage systems, and cloud storage, as discussed in Section 4.

With regarding to synchronisation performance, Wang et al. [43] and Li et al. [44] propose techniques to decrease the total traffic across the Internet, sync time, and overuse (when data traffic is greater than the amount of real data). Furthermore, optimization strategies were applied to decrease the volume of data traffic between cloud storage repositories and synchronisation clients. DropBox deserves a special discussion on this point: the client implements direct synchronisation between users placed on a single broadcast domain by using the LanSync protocol [45]. For instance, sync clients can detect that a set of users are located on the same local area network (LAN) and apply techniques to enable local synchronisation, avoiding traffic from/to cloud servers. However, such approaches are only applicable on a single broadcast domain. In short, two sharing users placed on the same office but on different domains (i.e. a routed datagram network) will synchronise their files through cloud services. Cloud4NetOrg innovates by allowing synchronisation between clients that are concentrated on private networks, whether on the same broadcast domain or not.

#### 4. Cloud4NetOrg architecture

Popular clients for synchronising files between local devices and cloud storage, such as DropBox and OneDrive, have a set of common modules (storage, transport, observer, and metadata repository) in their architectures [46,47]. The Cloud Storage for Network-based Organisation File System (Cloud4NetOrg) architecture, depicted in Figure 2, leverages this modular architecture, and provides extra functionality for catering to multisite collaborators.

A key component of the architecture is a two-level cache strategy that speeds up access to frequently used files. The first-level cache is local to each client, while the second-level cache uses the organisation repositories, minimising the use of the Internet link and consequently decreasing the access latency. Cloud4NetOrg is transparent to applications and software developers since the

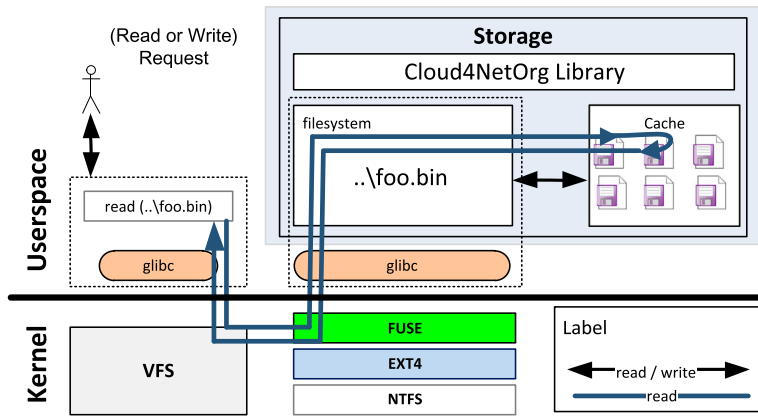


Figure 3. Detailed view of the Cloud4NetOrg storage module.

standard operating system interface is still used for accessing files. With regard to directory hierarchy, Cloud4NetOrg mounts the remote repositories as traditionally performed for any network file storage system. All modules are independent of cloud providers. In fact, just the transport module interacts with cloud application programming interfaces (APIs). The modules and data flow are individually discussed in the following sections.

#### 4.1. Storage module

The Cloud4NetOrg storage module is responsible for managing the file system as well as for performing input and output operations. The module combines virtual file system (VFS) with FUSE [48] to create a custom file system in user space with minimum intervention of kernel calls. In short, the Cloud4NetOrg storage module consists of: (i) a local cache space; (ii) an access library which provides functions to retrieve, read, write and remove files; and (iii) a file system, which exports to the operating system a structure of directories and files. The export operation is analogous to the directory structure control performed by traditional sync clients from the perspective of functionality and compatibility with running applications.

The local cache is populated on demand. The Cloud4NetOrg architecture provides the user with access to the entire set of files linked to their account on the cloud provider. As soon as a read or write operation is triggered, if the file involved is not in the local cache or is outdated, it is transferred from external storage to the local cache. The content replacement algorithm applied to the local cache is LRU (the least recently used files are discarded).

Figure 3 details the Cloud4NetOrg internal data flow for performing a read request submitted by an application. At the bottom, the VFS, FUSE, and traditional storage kernel modules receive calls from applications. For exemplifying the data flow in Figure 3, an application starts a read request by calling the C library (*glibc*). Through this library, the request is forwarded to the kernel modules (VFS and FUSE). The Cloud4NetOrg storage module, when triggered, interacts with the metadata repository (Figure 2) to locate the requested file and manipulate the local indices. For accomplishing the request, a returning flow is started when the file is placed at the local cache (first level). Since the cache is implemented with a write-back approach, the storage module waits until the content is copied.

The VFS overcomes the local space limitation by providing information (metadata) about all existing files, even when they are not locally available or while remote synchronisation is in progress. Although the available storage volume is smaller than the total required, the user still has knowledge about all the files. This approach is aligned with recently launched commercial tools, such as Dropbox Smart Sync [8].



## 4.2. Observer and transport modules

The observer module is in charge of running asynchronous tasks to update the local base metadata when a change is detected on the remote cloud storage repositories, invalidating cache entries when needed. In addition, the observer module calls the transport module for uploading not-yet-synchronised files to the organisation and/or cloud repositories, and updates the metadata repository after uploads are finished. It is worthwhile to highlight that Cloud4NetOrg is not designed to be used without access to a cloud storage repository for long periods of time. While the local metadata repository is complete, only a subset of files may be actually present (with their full content) in the L1 and L2 caches; in the case of disconnection, users can access only those cached files.

The Cloud4NetOrg transport module differs from traditional synchronisation architectures because, once parameterized, it can interact with multiple external repositories (cloud providers or legacy repositories). The transport module abstracts from others modules the existence of distinct external repositories dealing with providers APIs and legacy repositories access. Moreover, this module identifies (by periodically probing) which repository has lower latency for synchronising the remote files with the local cache. The default repository is the second-level cache. However, on running time, Cloud4NetOrg verifies the latency to all available repositories with ICMP protocol. When the user has access to the organisation's private network and latency to private repository is lower than to cloud provider, Cloud4NetOrg retrieves files from local storage. Otherwise, Cloud4NetOrg sets cloud-based storage as default repository. In this sense, a user can be connected to an organisation's private network (second-level cache) and retrieve the file from a private repository, or be connected to the Internet and retrieve the file directly from cloud storage.

## 4.3. Metadata repository

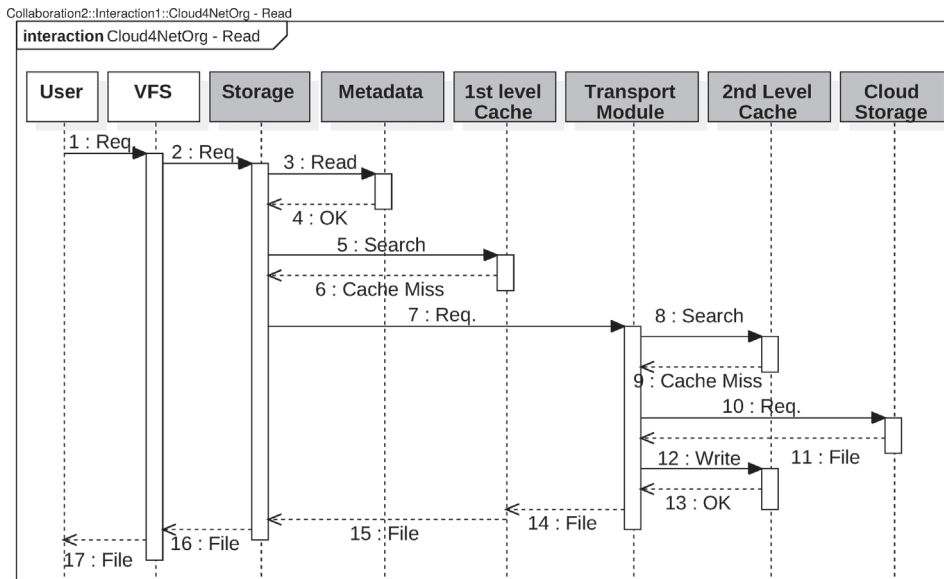
The metadata repository, managed by the observer module, indexes all shared files. Each file has only one index in the metadata repository but can be replicated in the first- and in second-level cache and in the cloud. This repository comprises information such as identifiers, object path (hierarchy), object type (file or directory), object size, creation and modification dates, as well as a control of completed and pending operations. In short, the database structure comprises a single table within few attributes, specifically, *id*, *name*, *created\_date*, *last\_modified\_date*, *size*, *url*, *path*, *type*, *parent\_reference*, *cloud\_pending*, *repository\_pending*, and *operation*. The attributes *id*, *name*, *type*, *parent\_reference\_id*, and *path* are used for composing the filesystem structure (folders and files), while the remaining is applied for controlling synchronisation between repositories.

According to the current state of objects (e.g. synchronised with the cloud, created, or synchronisation pending), the observer module triggers sync calls. Cloud4NetOrg relies on the metadata repository to mount the VFS, exporting to users a view of all existing shared files, regardless of their physical location at the time of the request.

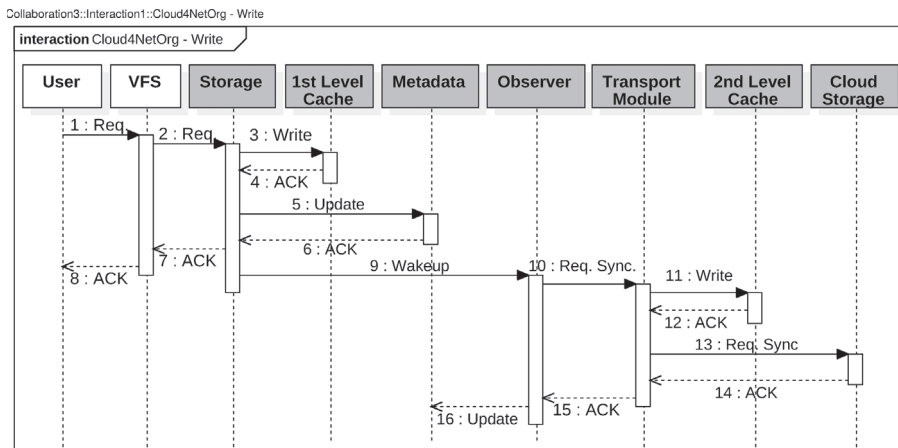
## 4.4. Read and write operations

Read and write operations are the most frequent demands managed by Cloud4NetOrg, and each involves several steps. For a read operation, the execution flow is completely synchronous, while for a write operation the file is synchronously written to the first-level cache and then asynchronously propagated to the second-level cache and to cloud storage.

Figure 4(a) shows the flow for retrieving a file stored in the cloud. We first discuss the case where the file is not cached and must be transferred from cloud storage, and then we discuss how caching affects the process. Actions performed by the operating system are represented by white boxes, while those executed by Cloud4NetOrg are in grey. The read flow is started by an application request (step 1), submitted to and translated by the VFS. In step 2, the request is sent to the Cloud4NetOrg storage module, which searches the file in the metadata repository (step 3) and uses the metadata (returned in step 4) to search the L1 cache (step 5). Since the file is not cached locally, in step 6 a *Cache Miss* is returned, and the file is requested from the transport module (step 7), which in turn searches the



(a) Flow for reading a file using Cloud4NetOrg.



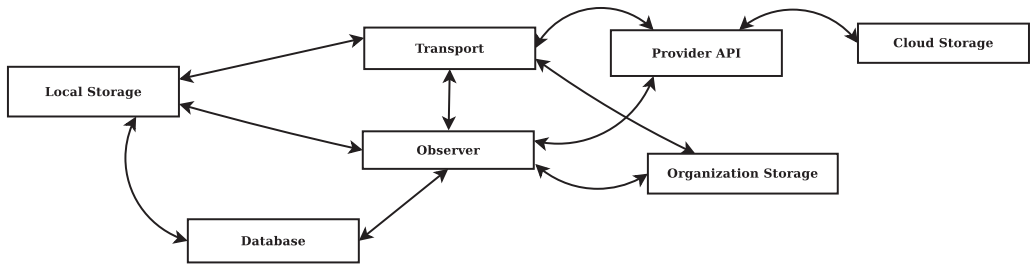
(b) Flow for writing a file using Cloud4NetOrg.

**Figure 4.** Data flow for reading and writing a file using Cloud4NetOrg.

file in the L2 cache (step 8). Since the file also cannot be found in the L2 cache (step 9), it is retrieved from the cloud provider (steps 10–11), and propagated back to the client (steps 12–17). As soon as the transport module performs step 12 (L2 cache fill) and receives OK (step 13), it sends the same file to fill the L1 cache (step 14).

If the file is already cached, it will be returned either in step 6 or step 9, depending on whether it is in the first- or in the second-level cache. If no metadata can be found in step 3, the metadata repository returns a *Not Found* message and the read is aborted, returning an error to the user. As mentioned, all steps are executed synchronously with a write-back cache.

For performing a write operation (Figure 4(b)), the request is submitted to VFS (step 1) and forwarded to the Cloud4NetOrg storage module (step 2), which executes three tasks in parallel: (i) writing to the L1 cache (step 3); (ii) updating the metadata repository (step 5), indicating that the file has been written to the cache and recording a pending synchronisation with the external repositories; and



**Figure 5.** Common components for Cloud4NetOrg-OneDrive and Cloud4NetOrg-DropBox.

(iii) waking up the observer module (step 9). Once the storage module receives acknowledgments from both the L1 cache and the metadata repository (steps 4 and 6), a response indicating success is sent back to the user (steps 7–8). The synchronisation process is managed by the observer module, which is in charge of scheduling an operation in the transport module (step 10). At this point, two tasks are executed in parallel: (i) writing to the L2 cache (step 11); and (ii) synchronising with cloud repositories (step 13). When the synchronisation is finished (step 14), the observer receives an acknowledgment from the transport module (step 15), and updates the metadata repository (step 16).

Following the approach implemented on popular synchronisation clients, Cloud4NetOrg does not address concurrency problems. When a concurrent operation is detected (e.g. two or more clients writing the same file), the synchronisation tool just duplicates the file, recording the date and time of the event. The final decision on version correctness is realised by the end user (the end-to-end design principle for distributed systems [24]).

## 5. Experimental analysis

As a proof of concept, we implemented two prototypes, one integrating Cloud4NetOrg with OneDrive (Cloud4NetOrg-OneDrive) and another with DropBox (Cloud4NetOrg-DropBox). We performed an experimental evaluation of these prototypes, comparing them with other traditional storage devices and solutions. We first describe components composing both prototypes followed by our testbed. Then, we assess the performance of the storage module and the efficiency gains provided by the second-level cache compared to DropBox with LanSync. Finally, we model the scalability provided by Cloud4NetOrg.

### 5.1. Prototypes implementation

The common components for composing both prototypes are presented in Figure 5. As a design decision, Cloud4NetOrg prototypes only support Windows operating system. The development of Linux-based prototypes is let as future work. The first component is the local storage, developed with Dokan FUSE<sup>2</sup> library, a user mode file system for Windows. This component allows Cloud4NetOrg to be fully integrated with the Windows file system, abstracting file and directory operations such as create, remove, update, among others.

The local storage communicates with transport, observer and database components. The database component acts as the metadata repository where each file is accounted, and interacts only with the local storage and observer modules. In turn, the observer component orchestrates all operations requested by other components. For instance, when triggered by the local storage, the observer coordinates with the provider's API and the transport module the synchronisation between cloud and local repositories. Finally, it asks the update of the corresponding metadata.

Complementary, the transport module is in charge of abstracting and dealing with different cloud providers. With regarding the libraries and tools, the provider's API is based on OneDrive SDK-C#<sup>3</sup> and DropBox SDK-DotNet,<sup>4</sup> for OneDrive and DropBox, respectively.

## 5.2. Testbed

Cloud4NetOrg prototypes and market-available synchronisation clients were executed on a controlled testbed, composed of virtual machines hosted by two HP EliteBook devices 840 with Core i5-4300U 2.5 GHz processor, 16 GB RAM, 500 GB SATA local disk, running Windows 10 and VirtualBox 5.1.8 virtualization software. Each virtual machine was configured with two vCPUs, 1 GB RAM and 80 GB for storage. As a file system, NTFS was used with default settings. The devices were interconnected by a switched Gigabit LAN, and a Wifi connection was configured with a D-LINK DWR-922 router (144.4 Mbps). The writing tests on USB storage were performed on a 64 GB Kingston Data Travel 101 flash drive, accessible on the virtual machine. Some tests required an external network-based storage device, which was implemented on a machine with the same hardware configuration.

## 5.3. Cloud4NetOrg storage module performance

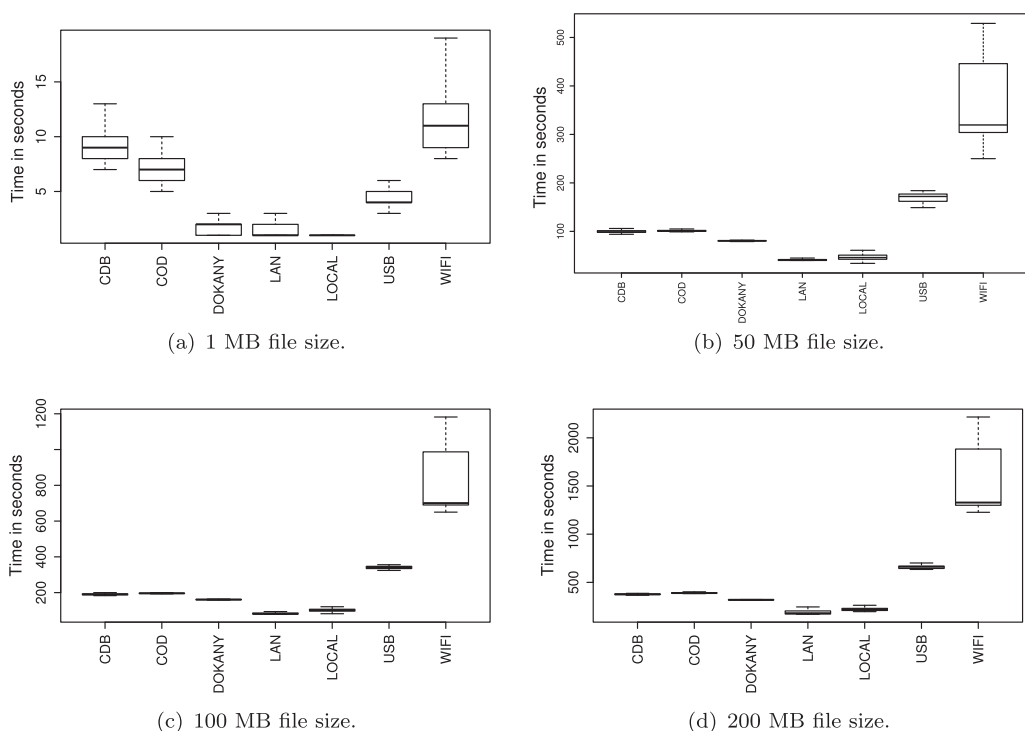
This scenario quantifies the additional time required to write, read, and delete files introduced by the Cloud4NetOrg storage module. The synchronisation time is a crucial metric for collaborative users as it represents the latency for accessing a given file. This experiment represents a scenario where users are in the same collaboration group and on the same LAN (e.g. Bob and Charles in Figure 1).

Following previous work [49], the input and output operations were performed with different file sizes (1 MB, 50 MB, 100 MB and 200 MB), generated and submitted by the *postmark* tool [50]. For each experiment, 100 rounds were performed (10 files are manipulated per round), and the graphs show the median, maximum, minimum, first and third quartiles of total execution time. The tests were run with five storage scenarios (composing the main scenario depicted in Figure 1):

- Local storage: Read and write operations were submitted to the same storage device in Bob's desktop (with NTFS) that the operating system is located on. This scenario is used as the baseline for performance analysis.
- Wired LAN: Operations were carried out on a remote repository with CIFS file system (over NTFS), on a Gigabit Ethernet LAN, termed organisation repository in Figure 1.
- Wifi LAN: Operations were performed on a organisation repository (CIFS over NTFS) accessed through an access point, configured in infrastructure mode with maximum bandwidth of 144.4 Mbps. For this scenario, LAN2 is controlled by the Wifi access point.
- USB storage: Read and write calls were performed on a flash drive connected to Bob's desktop by a USB 2.0 interface (FAT32).
- Dokany filesystem [51]: Dokany is a C++ library for implementing FUSE file systems on Windows. Since our prototypes are built over Dokany, this scenario provides a baseline for their performance.

The results are shown in Figure 6; data for our prototypes are represented as COD for Cloud4NetOrg-OneDrive and CDB for Cloud4NetOrg-DropBox. For 1 MB files, LAN storage performs almost as well as local storage, but has higher variability. COD is nearly 20% better than CDB, and both prototypes are outperformed by USB storage. For files larger than 1 MB, the user-mode file systems (COD, CDB, and Dokany) have higher execution time than both local and LAN storage, and lower than USB and WiFi; for these file sizes, both Cloud4NetOrg prototypes had roughly equivalent performance regardless of the cloud service provider used. Overall, WiFi storage had the higher execution times for all file sizes, and exhibited higher variability. The times for Dokany (the baseline for a user space file system) were between 33 and 50% higher than the numbers for LAN storage across all file sizes.

In short, this analysis indicates that Cloud4NetOrg induces low overhead compared to Dokany, and does not introduce an architectural bottleneck that could increase the synchronisation latency perceived by clients, even when all files must be retrieved from the second-level cache.



**Figure 6.** Cloud4NetOrg storage module performance compared with other storage solutions.

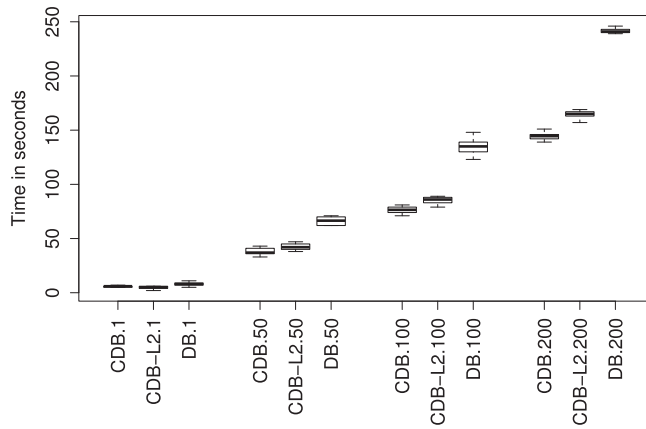
#### 5.4. Cloud4NetOrg-DropBox vs. DropBox

After individually quantifying the storage module performance of Cloud4NetOrg, a discussion is carried out comparing the full architecture performance with the DropBox client. For this experiment, all components of Cloud4NetOrg are involved: storage, metadata repository, transport module, observer, and external storage.

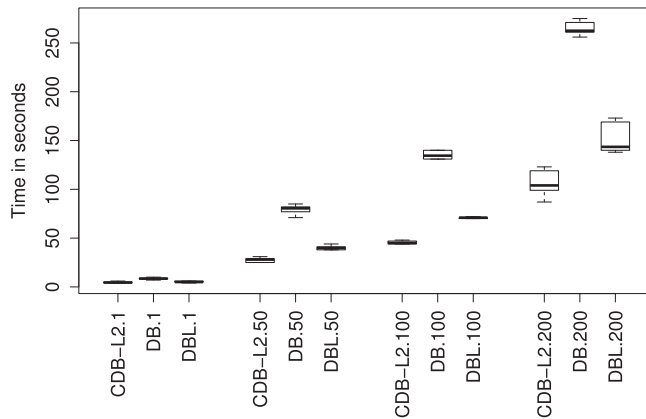
This scenario corresponds to an organisation that has several subnets as described in Figure 1. Besides the large use nowadays for synchronising files with cloud storage repositories, the DropBox solution (version 14.4.19) was selected to serve as baseline for comparison due to the implementation of optimization techniques. DropBox natively uses data compression and deduplication before transfer from/to cloud and offers an optional module for LAN synchronisation (LanSync).

DropBox LanSync decreases the data flow from/to the cloud provider by transferring files directly between two DropBox clients located on the same subnet. In this sense, the comparison highlights the two levels of caching implemented by Cloud4NetOrg versus DropBox LanSync. The first level resides on a client's local disk, while the second one resides on the local network (which can be disabled for debugging and evaluation purposes). The results were collected for Cloud4NetOrg, with and without the second-level cache; for the DropBox clients in the same subnet, when LanSync can act; and for DropBox clients on different subnets, where LanSync is not applicable.

In Figure 7, Cloud4NetOrg without second-level cache is identified by CDB while a version with cache (accessible by the private organisation network link) is labeled as CDB-L2. The file size under analysis is concatenated with the label. Figure 7(a) shows the results with clients placed in two different domains. This scenario is illustrated by Figure 1 where Alice, placed in Paris, is collaborating with Bob, placed in New York, on a shared set of files. Initially, with 100 and 200 MB files, the waiting time for DropBox users is 67% higher compared to Cloud4NetOrg. Considering the use of the organisation repository as a cache, Cloud4NetOrg with second-level cache decreased by 14% the synchronisation



(a) Cloud4NetOrg (with and without second-level cache) vs. DropBox.



(b) Cloud4NetOrg vs. DropBox (results for one and two subnets).

**Figure 7.** Time required for a client to synchronise a file updated by another user.

time for 200 MB. This low impact is occasioned by the benchmarking tool: the number of read and write operations performed by *postmark* is similar, representing a worst-case sharing scenario, as the second-level cache is updated but the files are immediately removed.

Figure 7(b) shows the time required for a client to synchronise a file that has been updated by another user. The results for Cloud4NetOrg (CDB-L2) have the second-level cache enabled. For DropBox, we consider two scenarios. In the first scenario (represented by DBL), both users are in the same broadcast domain, and so LanSync is used; this corresponds to Bob and Charles in Figure 1. In the second scenario (DB), the users are in different subnets, which precludes the usage of LanSync; this corresponds to Alice and Bob in Figure 1. The results for DBL point out a 93% performance improvement compared to DB, showing the efficiency of LanSync in this scenario. Cloud4NetOrg is more efficient than DropBox (with or without LanSync), especially for large files, due to lower synchronisation latency. The reason is that LanSync only allows file synchronisation between clients when a file is available (after the upload is finished), while Cloud4NetOrg decreases the delay by starting the synchronisation to other clients immediately after sending a control message (updating the metadata repository). Finally, when the second-level cache is available, the placement of collaboration users affects the execution time of Cloud4NetOrg. In the first scenario (Figure 7(a)), Cloud4NetOrg needs to transfer data between Paris and New York to fill the L2 cache, and its execution time is worse than in the second scenario (Figure 7(b)).

When using the DropBox synchronisation client, a user only has access to a file (including its metadata) after it has been transferred to the local disk. This may lead to unneeded delays and to wasting network bandwidth and disk space if files are brought in from the cloud and never accessed by the user on that device, which is likely in large repositories. In Cloud4NetOrg, the metadata repository always has information about all files. This provides Cloud4NetOrg users with knowledge about all the files available in the cloud provider without having to pre-populate either of the two cache levels. On the flip side, Cloud4NetOrg users cannot work off-line, except with previously cached files.

To quantify the impact of these distinct approaches, another experiment was conducted based on different usage profiles. We measured the time it takes for a user to be able to access between 10 and 100% of the files in a directory hierarchy, assuming none of the files are on the local disk. The results depicted in Figure 8 indicate that, for a DropBox client, the usage profile has no impact on total access time, as all files and metadata are transferred from the cloud to the local device, regardless of usage. Therefore, the time measured for DropBox represents the total time required to completely visualise the directory contents. The transfer time for Cloud4NetOrg clients varies according to the directory size (10 MB, 500 MB, 1 GB and 2 GB) and the fraction of files actually accessed. The differences between DropBox and Cloud4NetOrg are larger when files are available on the LAN (Figure 8(a)) than when they need to be transferred from the cloud (Figure 8(b)). This indicates that the L2 cache in Cloud4NetOrg provides better performance than DropBox, even with LanSync enabled.

To summarise, our experiments show that, for all collaborative users in Figure 1, Cloud4NetOrg decreases Internet link usage, requires less space in local storage, and minimises the waiting time for file synchronisation.

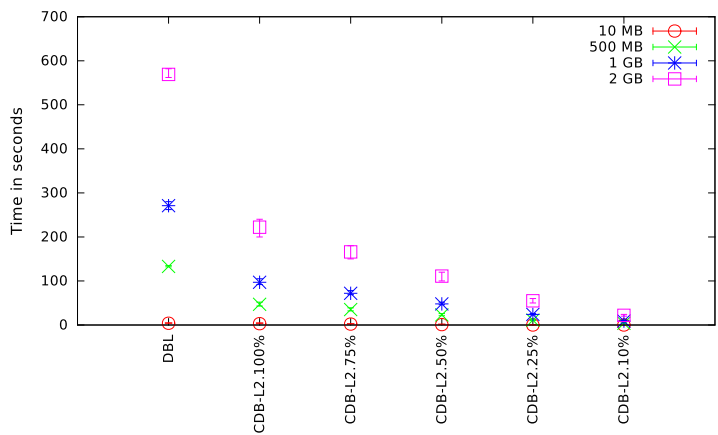
### 5.5. Cloud4NetOrg scalability

A cloud storage characterization study performed by Gonçalves et al. on academic campuses [41] highlighted that 333 users had 3 million of files totaling approximately 1.38 TB of data. If Cloud4NetOrg was used with a repository with the same characteristics, all files would be indexed and made available to users, consuming only the storage space required for the metadata repository, without need for a previous download and synchronisation phase. In the metadata repository, each file is represented by a register in the database and requires approximately 787 bytes. Thus, the total metadata repository volume can be estimated by multiplying the register's size by the number of files. For indexing 3 million files, the Cloud4NetOrg-OneDrive prototype consumes close by 2.2 GB, 0.16% of total storage volume.

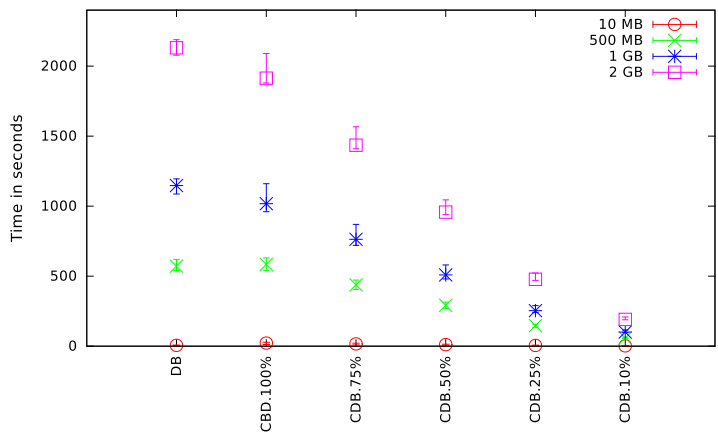
In the Cloud4NetOrg architecture, the first-level cache is configurable and independent of the storage capacity of the second-level cache and external repositories. In this sense, the number of download and upload events (and volume) from/to external storage is decreased as only actually requested files are synchronised. In addition, the second-level cache reduces the Internet traffic for multisite organisations as collaborative users interconnected by a private network can share files via a local cache repository.

The volume of data synchronised between an L2 cache and cloud repositories (i.e. data traversing an Internet link) can be modeled using the equation  $v = b_e \cdot \alpha(1 + n)$ , where  $b_e$  is the volume of bytes transferred from/to the cloud,  $n$  the number of clients to synchronise, and  $\alpha \leq 1$  is a coefficient of data compression. For popular clients (DropBox and OneDrive), the number of collaborative users has direct impact on the Internet link traffic, while for Cloud4NetOrg,  $v$  is given by  $v = b_e \cdot \alpha$  (thus, it is independent of the number of collaborative clients). To illustrate, Table 1 presents results given by the model for an organisation with varying numbers of employees sharing an Internet link. The estimated results show the volume of traffic for different numbers of collaborative users. We assumed no compression ( $\alpha = 1$ ), and that each employee writes on average 7 MB to the cloud (this amount is taken from [41]); thus,  $b_e = 7 \times \text{Employees}$ . Cloud4NetOrg has constant Internet traffic data volume independent of the number of collaborative users. In the worst case, with 100 collaborative users,





(a) Files available on the L2 cache (Cloud4NetOrg) or retrieved via LanSync (DropBox).



(b) Files downloaded from cloud provider.

**Figure 8.** Access time comparison between DropBox and Cloud4NetOrg clients for different usage profiles.

**Table 1.** Data volume sent to a cloud repository (GB).

Employees	Collaborative users (n)			Cloud4NetOrg
	2	20	100	
1000	20	143	690	7
2000	41	287	1380	14
5000	102	717	3452	35
8000	164	1147	5523	56
10000	295	1435	6904	70

popular synchronisation clients would consume on the order of 6,904 GB, while Cloud4NetOrg would require only 70 GB.

## 6. Conclusion

The Internet has revolutionised the workflow of modern organisations. Freelancers, home-office, mobile users, and other remote collaborations are commonly found in companies from different business sectors. In this context, cloud-based storage has been acting as a driving force, increasing the availability of files, which are delivered as services. Moreover, tools for file synchronisation with cloud repositories as commonly available and well-accepted by home users.

The present work reviewed the issues and challenges related to the adoption of such cloud-based services by multisite organisations. The current synchronisation clients lack technologies for enterprise collaborative users. The collaborative users tend to be concentrated in private networks accessing legacy storage systems. Eventually, the local repositories are synchronised atop private networks. In this sense, the present work proposed Cloud4NetOrg, an architecture for synchronising cloud file storage and organisation repositories. Cloud4NetOrg uses the existing repositories for composing a two-level cache architecture, while the private network is used for decreasing the Internet traffic. Both optimizations decrease the latency perceived by clients while accessing files.

Two prototypes were developed indicating that Cloud4NetOrg is compatible with public synchronisation tools (OneDrive and DropBox). The experimental results indicate a low overhead of Cloud4NetOrg storage module, and a promising usage on topologies with multiple subnetworks. Compared to DropBox, Cloud4NetOrg decreases the synchronisation latency by 67% for 100 and 200 MB files. Although efficient for the experimental scenarios analysed, in future work we aim to investigate the application of robust algorithms for enhanced cache coherence and control. In addition, we aim to develop a multi-provider prototype simultaneously accessing files from different cloud repositories. We also intend to improve the transport module in order to further reduce synchronisation traffic.

## Notes

1. An earlier version of this paper circulated in Portuguese only [14] comprising an initial architecture proposal and preliminary results.
2. <https://github.com/dokan-dev/dokany/wiki/FUSE>
3. <https://github.com/OneDrive/onedrive-sdk-csharp>
4. <https://github.com/dropbox/dropbox-sdk-dotnet>

## Disclosure statement

No potential conflict of interest was reported by the authors.

## Funding

This research was developed at the Laboratory of Parallel and Distributed Programming (LabP2D), and was partially supported by UDESC and FAPESC.

## ORCID

Guilherme Koslovski  <http://orcid.org/0000-0003-4936-1619>

Mauricio Pillon  <http://orcid.org/0000-0001-7634-6823>

## References

- [1] Zhang Q, Cheng L, Boutaba R. Cloud computing: state-of-the-art and research challenges. *J Internet Serv Appl*. 2010;1:7–18.
- [2] DropBox. DropBox, Tech. rep., Dropbox Corporation; 2017. Available from: <https://www.dropbox.com/>
- [3] Drive G. Google Drive, Tech. rep., Google Driver Corporation; 2017. Available from: <https://www.google.com/drive/>
- [4] OneDrive. OneDrive, Tech. rep., One Drive Corporation; 2017. Available from: <https://onedrive.live.com/>

- [5] Gonçalves G, Drago I, Silva APC, et al. Analyzing the impact of dropbox content sharing on an academic network. In: Proceedings of the Brazilian Symposium on Computer Networks and Distributed Systems (SBRC); SBRC; 2015. p. 100–109.
- [6] Gonçalves G, Drago I, Silva APC, et al. Characterizing and modeling the dropbox workload. In: Proceedings of the Brazilian Symposium on Computer Networks and Distributed Systems (SBRC). Florianópolis, Brazil: SBRC; 2014.
- [7] Drago I, Mellia M, Munafo MM, et al. Inside dropbox: understanding personal cloud storage services. In: Proceedings of the Internet Measurement Conference (IMC); Boston, MA, USA: IMC; 2012.
- [8] Dropbox. Smart Sync, Tech. rep.; 2017. Available from: <https://www.dropbox.com/business/smartsync>
- [9] Google. G Suite for Teams, Tech. rep.; 2017. Available from: <https://gsuite.google.com/solutions/teams/>
- [10] Vrable M, Savage S, Voelker GM. BlueSky: a cloud-backed file system for the enterprise. In: Proceedings of the USENIX Conference on File and Storage Technologies (FAST); San Jose, CA: FAST; 2012.
- [11] Bessani A, Mendes R, Oliveira T, et al. SCFS: a shared cloud-backed file system. In: Proceedings of the USENIX Annual Technical Conference (USENIX ATC); Philadelphia, PA: USENIX; 2014. p. 169–180.
- [12] Douceur JR, Bolosky WJ. A large-scale study of file-system contents. In: Proceedings of the ACM SIGMETRICS; Atlanta, GA, USA: ACM; 1999. p. 59–70.
- [13] Hofmann P, Woods D. Cloud computing: the limits of public clouds for business applications. IEEE Internet Comput. 2010;14:90–93.
- [14] Andriani G, Koslovski G, Pillon MA. Sincronização de Arquivos entre Nuvens de Armazenamento e Repositórios Geograficamente Distribuídos, in Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC) 2016. SBRC, Salvador, Bahia: Maio; 2016.
- [15] Katzer M, Crawford D. Office 365: moving to the cloud. In: Office 365. Springer; 2013. p. 1–23.
- [16] Mell P, Grance T. The NIST definition of cloud computing. In: NIST SP 800–145, Technical Report. National Institute of Standards and Technology; 2011.
- [17] Naldi M, Mastroeni L. Cloud storage pricing: a comparison of current practices. In: Proceedings of the International Workshop on Hot Topics in Cloud Services; Prague, Czech Republic: ACM; 2013.
- [18] Shepler S, Eisler M, Robinson D, et al. Network file system (NFS) version 4 protocol; RFC 3530, Technical Report. 2003.
- [19] Microsoft Corporation. Common Internet File System (CIFS) protocol; 2014. Available from: <http://msdn.microsoft.com/en-us/library/ee442092.aspx>
- [20] French SM and Samba Team. A new network file system is born: comparison of SMB2, CIFS and NFS. In: Proceedings of the Ottawa Linux Symposium (OLS). Ottawa, Canada: OLS; 2007. p. 131.
- [21] Dukkkipati N, Mathis M, Cheng Y, et al. Proportional Rate Reduction for TCP. In: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, IMC '11, Berlin, Germany; New York, NY, USA: ACM; 2011. p. 155–170.
- [22] Stallings W. Foundations of modern networking: SDN, NFV, QoE, IoT, and Cloud. 1st ed. Addison-Wesley Professional; 2015. ISBN: 0134175395 and 978-0134175393.
- [23] Schlinder B, Kim H, Cui T, et al. Engineering egress with edge fabric: steering oceans of content to the world. In: Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2017; August 21–25, 2017; Los Angeles, CA, USA; 2017. p. 418–431.
- [24] Saltzer JH, Reed DP, Clark DD. End-to-end arguments in system design. ACM Trans Comput Syst. 1984;2:277–288.
- [25] Smith DM, Petri G, Natis YV, Scott D, Warrilow M, Heiser J, Bona A, Toombs D, Yeates M, Bova T, Lheureux BJ. Cloud computing affects all aspects of IT. Gartner Group: Tech. rep; 2013.
- [26] FUSE-J. FUSE-J, Tech. rep.; 2017. Available from: <https://sourceforge.net/projects/fuse-j/>
- [27] Nasuni. UniFS – a true global file system, Tech. rep., Nasuni Corporation; 2017. Available from: <https://www.nasuni.com/what-is-a-global-file-system/>
- [28] TwinStrata. TwinStrata, Tech. rep., TwinStrata Corporation; 2017. Available from: <https://www.emc.com/domains/cloudarray/index.htm>
- [29] Panzura. Hybrid Cloud NAS, Tech. rep., Panzura Company; 2017. Available from: <http://panzura.com/solutions/hybrid-cloud-nas/>
- [30] StorSimple. StorSimple, Tech. rep., StorSimple Corporation; 2017. Available from: <http://www.storsimple.com/>
- [31] Ghemawat S, Gobiuff H, Leung ST. The Google file system. In: Proceedings of the ACM Symposium on Operating Systems Principles (SOSP). New York, NY; 2003. p. 29–43.
- [32] Muniswamy-Reddy KK, Macko P, Seltzer M. Provenance for the cloud. In: Proceedings of the USENIX Conference on File and Storage Technologies (FAST); San Jose, CA; 2010.
- [33] Duan H, Yu S, Mei M, et al. CSTORE: a desktop-oriented distributed public cloud storage system. Comput Electr Eng. 2015;42:60–73.
- [34] Shvachko K, Kuang H, Radia S, et al. The hadoop distributed file system. In: Proceedings of the IEEE Symposium on Mass Storage Systems and Technologies (MSST). Incline Village, Nevada; 2010. p. 1–10.
- [35] Kistler JJ, Satyanarayanan M. Disconnected operation in the Coda file system. ACM Trans Comput Syst. 1992;10:3–25.
- [36] Chang C, Sun J, Chen H. Coral: a cloud-backed frugal file system. IEEE Trans Parallel Distrib Syst. 2016;27:978–991.
- [37] Machado GS, Bocek T, Ammann M, et al. A cloud storage overlay to aggregate heterogeneous cloud services. In: 38th IEEE Conference on Local Computer Networks (LCN); Sydney, Australia: IEEE; 2013. p. 1–12.

- [38] Otixo. Otixo, Tech. rep.; 2017. Available from: <http://www.otixo.com>
- [39] oDrive. ODrive, Tech. rep.; 2017. Available from; <http://www.odrive.com/>
- [40] Drago I, Bocchi E, Mellia M, et al. Benchmarking personal cloud storage. In: Proceedings of the Internet Measurement Conference (IMC); Barcelona, Spain: ACM; 2013.
- [41] Gonçalves G, Drago I, Silva APC, et al. Modeling the Dropbox client behavior. In: Proceedings of the IEEE International Conference on Communications (ICC), June. Sydney, Australia: IEEE; 2014. p. 1332–1337.
- [42] Tate J, Lucchese F, Moore R. Introduction to storage area networks. IBM Redbooks; 2006. ISBN: 0738428345 and 978-0738428345.
- [43] Wang H, Shea R, Wang F, et al. On the impact of virtualization on dropbox-like cloud file storage/synchronization services. In: Proceedings of the 20th International Workshop on Quality of Service; Coimbra, Portugal: IEEE; 2012.
- [44] Li Z, Wilson C, Jiang Z, et al. Efficient batched synchronization in Dropbox-like cloud storage services. In: Eysers D, Schwan K, editors. Middleware 2013. Vol. 8275, Lecture notes in computer science. Berlin Heidelberg: Springer; 2013. p. 307–327.
- [45] LANSync. LANSync, Tech. rep.; 2017. Available from: <https://www.dropbox.com/en/help/137>
- [46] Houston D, Ferdowsi A. Network folder synchronization. US Patent 8,825,597. 2014.
- [47] Besen A, Cheong H, Mueller H, et al. Sharing and synchronizing electronically stored files. US Patent 8,949,179. 2015.
- [48] FUSE. Fuse, Tech. rep.; 2017. Available from: <http://fuse.sourceforge.net/>
- [49] Kanaujia V, Giridhar C. FUSEing Python for development of storage efficient filesystem. Python Paper, Technical Report. 2012;7:4.
- [50] Katcher J. Postmark: a new file system benchmark, Tech. Rep. TR3022. Network Appliance; 1997.
- [51] Dokany. Dokany, Tech. rep.; 2017. Available from: <https://github.com/dokan-dev/>