An approach for constructing private storage services as a unified fault-tolerant system

J.L. Gonzalez, a,∗, d, Jesus Carretero Perez, a, Victor Sosa-Sosa, a, Juan F. Rodriguez Cardoso, a, Ricardo Marcelin-Jimenez, b

a Carlos III University, Computer Science Department, Madrid, Spain
b UAM-Iztapalapa, Electrical Engineering Department, Mexico City, Mexico
c Cd. Valles Institute of Technology, San Luis Potosi, Mexico
d Information Technology Laboratory, Center of Research and Advanced Studies of the National Polytechnic Institute (CINVESTAV), Ciudad Victoria, Mexico

A R T I C L E   I N F O

Article history:
Received 7 September 2012
Received in revised form 26 February 2013
Accepted 26 February 2013
Available online 16 March 2013

Keywords:
Storage services
Web-based storage
Fault-tolerance
Virtualization
Federation

A B S T R A C T

Organizations are gradually outsourcing storage services such as online hosting files, backup, and archival to public providers. There are however concerns with this process because organizations cannot access files when the service provider is unavailable as well as they have no control and no assurance on the management procedures related to data. As a result, organizations are exploring alternatives to build their own multi-tenant storage capacities.

This paper presents the design, implementation and performance evaluation of an approach for constructing private online storage services. A hierarchical multi-tier architecture has been proposed to concentrate these services in a unified storage system, which applies fault-tolerant and availability strategies to the files by passing redundant information among the services or tiers. Our approach automates the construction of such a unified system, the data allocation procedure and the recovery process to overcome site failures. The parameters involved in the performance of the storage services are concentrated into intuitive metrics based on utilization percentage, which simplifies the administration of the storage system. We show our performance assessments and the lessons learned from a case study in which a federated storage network has been built from four trusted organizations spanning two different continents.

© 2013 Elsevier Inc. All rights reserved.

1. Introduction

Web-based storage solutions are a set of value-added services such as content synchronization among different devices (Dropbox; sugarsync) (e.g. tablets, phones, computers), online backup (Garg, 2009; Cleversafe) and remote file access via web dashboards (Dropbox).

This kind of services have become a common solution for the large majority of users – individuals or organizations – that need to store or retrieve their files any time from anywhere (ATAW).

The users can assess web-based storage by choosing either a Pay As You Go pricing model in which some quality of service can be contracted (Windows azure data and storage; Amazon) or free file hosting systems (FHS) (Dropbox; sugarsync) in which the quality, security and privacy are based on a best effort. Both storage solutions are becoming a business success case (Annette Jump, 2011; Richard et al., 2011), which has increased both the web and file sharing traffic on the Internet (Waterloo, 2011).

However, the users of this kind of services have expressed their concerns about the lack of control on the data management procedures in charge of their information (Chow et al., 2009). There is also uncertainty of legal aspects as they delegate a private content to a third-party (Smith, 1979; Salveggi, 2004), for instance, the possibility of performing data mining on private data (Google privacy and policies), and the information vulnerability, in case that the storage provider were compromised (Nytimes; Google, 2011).

In addition, regulations in different countries require organizations to guarantee the integrity and confidentiality of their files during large periods of time such as six years in the case of medical contents (Law, 2011) or three for financial information (U. IRS: T.R.R. Guide). Under the current growing rates, this is enough time to achieve massive volumes that will have an impact on middle- and long-term storage costs (Gantz and Reine; 2011; Walker et al., 2010).

Price is a aspect to weigh up when users choose a storage provider (Ion et al., 2011; Fujitsu. and com Masaharu Sato,
long-term costs and penalizations are considered because prices are set per unit of storage and time. The longer some content is stored in the cloud, the higher the price to be charged.

The costs associated to private and public storage solutions have been widely studied (Mazhelis et al., 2012; Chen and Sion, 2011; Walker et al., 2010; Singh and Jangwal, 2012).

In the case of private storage solutions, the most significant cost is the capital investment (site infrastructure and the space required for it), which is observed in the first year and also includes the investment required when the storage must be replaced because of its life expectancy (Mazhelis et al., 2012; Chen and Sion, 2011). The common operating expenses in private solutions are energy, personal and maintenance, which represent a constant cost that depend on the country where the site has been installed.

In the case of storage solutions offered by public providers, the operating expenses are commonly the service costs such as storage, network, databases (server hours and transactions), and data transfer. Additional charges could be applied to the organization’s account when contracting functionalities such as geographical redundancy, auditing and monitoring systems.

Fig. 1 shows the results of a cost comparison over three years (Singh and Jangwal, 2012) for both public (100 EC2 Amazon instances and 53 storage) and the same solution developed in a private site.

The comparison shown in Fig. 1 is based on assumptions. The assumptions affecting the costs of the public solution are 24/7 availability and an increment up 15% in the resources for facing up a given increment of the demand.

The costs assumed in the private solution are the virtual server deployment costs, the yearly support cost they occur as well as the allocated maintenance. It also assumes the organization did not include the professional services to install the private cloud infrastructure because organization technical staff had the skills to do it themselves.

Fig. 1 illustrates the costs of public and private cloud solutions with a budget estimate. This means the costs can be different when the organizations establish different scenarios with different assumptions. The organizations can reproduce this type of exercises either by using a budget estimate web site Planforcloud in which the administrators can either choose cloud services from several providers or use a mathematical model (Walker et al., 2010).

The private storage scenarios are acceptable for organizations having space to allocate infrastructure, handling large volume of data for several years and that are interesting in the development of additional functionalities to avoid both the privacy issues of the outsourcing and the service unavailability produced by the vendor lock-in.

As a result, organizations have started to consider the possibility of building their own in-house storage capacities as information is becoming a very important asset, and this is a promising alternative to keep in control of this valuable resource (Peters, 2011).

This decision is fostering studies that aim to develop new fault-tolerant storage systems based on commodity components as well as guidelines for configuring such environments in a cost-effective manner (Annette Jump, 2011).

This paper presents the design, implementation and performance evaluation of an approach for constructing private online storage systems by using commodity components with minimal proprietary requirements.

This approach introduces the following contributions:

- A multi-tier architecture of storage services. A multi-tier hierarchical architecture has been proposed to address user/organization concerns about data confidentiality and integrity. The first tier or Client tier is the ambit of the user and it includes local storage service. The second one or Trusted Shared tier is the ambit of the organizations and includes online hosting file service. The third one or Trusted Shared tier is a federated ambit including backup and archival services. That architecture preserves the control on the data assets management and offers options to decide the best suited place to store/retrieve each file. The tiers of this architecture perform analogous functions to traditional storage but they are applied to the online storage service. This means that the Client tier can be regarded as the cache, the Trusted tier works on the role of the disk, and the Trusted Shared tier plays the role of tape system. We have defined that hierarchy according to the latency and costs to achieve data availability at each tier.

- A virtualization mechanism. A mechanism automates the virtualization of the multi-tier architecture of storage services as a single unified system. It also automates the placement and distribution of data as well as the recovery procedure to overcome site failures.

- A data allocation and load balancing method. This data placement method has been designed for passing anonymized files through tiers in a balanced manner. The goal of this method is preserving the autonomy of the storage services and keeping metadata in-house.

- Intuitive metrics to simplify the administration of the storage system. Our approach proposes to concentrate the parameters involved in the performance of the storage services into easily understandable metrics called Utilization Factors (UF), which represent the percentage of consumed capacity of a disk, server, or the whole system.

- A set of fault-tolerant strategies. We propose reactive and proactive redundancy strategies to deliver data availability when suffering from failures such as disk/server crashing, virtual machine blackout and even a multi-site disaster.

To show the feasibility of our proposal, this paper also presents a case study describing the implementation of four private online storage systems based on four trusted organizations, which make part of a federated storage network.

The approach provides an isolated storage system for each trusted organization in which users store and retrieve their files in ATAW manner by using the unified organization domain. The users are also able to store/retrieve files by using the federated shared storage when the site of their organization is down.

In the implementation, each organization keeps in-house control on metadata management, file allocation, resource location and access methods, as well as a given data availability and reliability degree. Moreover, organizations decide the amount of resources to be assigned to the federated storage. We show our performance...
assessments and the lessons learned from these experiments, that can be regarded as general operation guidelines.

2. Related work

Amazon Web Storage (Dropbox) and Windows Azure data storage (Windows azure data and storage) are platforms based on a pay-as-you-go pricing model, which apply rates based on the monthly stored contents plus the penalizations stated at the service level agreement (SLA) contracted. As we already mentioned, these conditions could lead to long-term costs that might press organizations to consider an alternative service, such as a private cloud (Chris Peters and Chahal, 2011) or a file hosting service (FHS) (Dropbox; sugarsync).

FHS has been targeted to those users requiring file storage and sharing in ATAW manner, who cannot afford neither the expenses of public services, nor the cost of a private cloud. A common FHS is based on URLs, which are constructed on a rather predictable string of characters (explicit user names or sequential identifiers) and therefore, they are exposed to attacks (Bermbach and Klems, 2011).

In response to the growing concern about outsourcing services, public cloud providers are starting to address some of these problems. For instance, in order to avoid legal issues, Amazon allows users to choose the geographic locations where their files will be stored, which are reduced however to Ireland, Japan, Singapore, USA and Brazil (Amazon, 2013). As for information availability, hot standby host recovery solutions are currently available to withstand a site disaster by online mirroring information to the storage site of a trusted organization (corporation, 2010). Nevertheless, the cost of these solutions could be prohibitive.

Pay-as-you-go storage services and FHS are both delivered by third-party providers which protect themselves with fuzzy conditions on those issues related to service interruptions (Nytimes, 2011). These terms introduce uncertainty on the user data availability. The major threat is that individuals or organizations could have their documents temporarily or permanently unavailable without the possibility of claiming any responsibility (M. file-sharing site shut and down, 2012). As a result, reliability in cloud storage is an open issue that has started to be addressed by schemes such as secure RAID for the cloud (Abu-Libdeh et al., 2010; Kevin et al., 2009), and the swift system commonly used by OpenStack (Gadella and Mattoso, 2011). Also, in order to build a fault-tolerant infrastructure, new approaches are proposing a single virtualized access to a platform gathering several cloud-storage public providers (Bermbach and Klems, 2011). As for FHS, new proposals offer in-house strategies based on P+Q or parity blocks per file by using Virtual Disks (Chai and Uehara, 2009) and distributed web storage (Gonzalez and Marcellin-Jimenez, 2011). Nevertheless, these alternatives could hardly support a major site disaster.

Federations have been proposed in the past for grid environments (Ahuja et al., 2010; Ranjan et al., 2005), and some proposals have focused these approaches on processing and load balancing for cloud-based storage among public providers (Wang et al., 2010). These solutions however have not been oriented to support a private storage services. In this context, in-house online storage based on the virtualization of commodity components is an interesting solution and the object of our proposal. There are data management mechanisms oriented to monitor location, according to legal requirements, but their implementation are stilling in progress (Kousiouris et al., 2011).

Approaches to cloud storage service selection are available for public providers (Ruiz-Alvarez and Humphrey, 2011; Goiri, 2012), which even could be used in our implementation when deciding to send redundancy to public providers.

Distributed file systems have been proposed for building global data persistent systems in GRID (Gridfilesystem; Sánchez et al., 2012; xtremefs) and P2P environments (Bindel et al., 2002). Our multi-tier approach allows the organizations to build this type of global systems by using the native file system of the compute instances in a cloud environment. A qualitative comparison showing similarities and differences between current global data persistent systems and the multi-tier approach is shown in the next section.

3. The construction of private storage services as a unified fault-tolerant system

We consider a scenario where a given set of organizations require to build in-house fault-tolerant storage services. We also consider the organizations have expressed their willingness to be included in a federated scheme composed of trusted parties, which cooperate to create a private shared storage service. This scenario is quite common when each participant is either a business unit of a larger enterprise, or it is part of a collaborative network. This type of scenarios already has been described in sky environments (Keay et al., 2009).

Each organization that takes part in this federation has preserved an in-house infrastructure including a set of web storage nodes or one of them at least. A storage node is either a private cloud computing instance or a physical PC, which are automatically configured by the virtualization mechanism to serve I/O operations such as PUT, GET, DELETE and LIST in ATAW manner. Each storage node is in charge of a set of partitions, which are used for storing files.

3.1. A multi-tier architecture of storage services

Our approach concentrates the storage services of an organization into a multi-tier architecture. We identify the following physical tiers where the storage services of an organization can be placed:

1. Client: This tier is the local storage service, which is placed at the user side. In this condition, the user retains secure and local access by using either her notebook or tablet.
2. Trusted: This tier is the online File Hosting Service (FHS) of an organization. The storage nodes are inside the local organization firewall. This tier offers low latency and full control on the storage nodes.
3. TrustedShared: This tier is the backup and archival service for trusted organizations. In this tier the storage nodes are outside the organization firewall, but inside the firewall of a private federated environment in which the local organization participates. The latency in this tier depends on the distance among the federation members.

The multi-tier architecture is depicted in Fig. 2, which shows the workflow to online store/retrieve files by using the multi-tier architecture of storage services in which Client tier offers local storage service, Trusted tier offers File Hosting Service to store/retrieve files in ATAW manner and TrustedShared tier offers backup and archival services by using a federated storage network.

The idea is quite simple, a unified system receives I/O operations and determines which storage service or tier is the best suited to serve these requests according to the user concerns and the resources availability. To deliver transparent file availability, the approach moving from a higher to a lower tier until to find either the requested file or its redundancy. As a result, the more lower tier, the more latency will be observed and more actions must be
done to achieve data availability and to attend the user concerns about privacy in data management.

### 3.2. Virtualized storage zone

The design approach virtualizes the multi-tier architecture as a single unified domain called Virtualized Storage Zone.

A virtualized zone is a multi-layer system designed to control the resources of the tiers in a transparent manner.

Fig. 3 shows that the Virtualized Storage Zone includes layers such as single access control, multi-tenancy for isolating users and organizations from other users, a data placement to allocate and locating files and recovery strategies for solving failures in each tier. In this example, a given user accesses her organization domain (single point and access control), finds her work groups and files (multi-tenancy) and stores/retrieves her files (data placement) by using the multi-tier architecture, even when the site of her organization is down (file availability and recovery). System monitoring records logs about performance and resources utilization while auditing records logs about data management.

A virtualization mechanism automatically creates and configures the Virtualized Storage Zone as a client–server model in which all storage nodes are clients and some of them are appointed to be servers or tier managers.

The clients only include the management of data placement and recovery layers while the managers are in charge of coordinating the whole multi-layer scheme.

The virtualization mechanism creates and configures a manager per tier, which provides two databases, one for metadata and the other one for resource management. The first database contains information about users accounts, files and logs, which is used to handle the access control and multi-tenancy layers. The second database contains the information required to support data placement and recovery layers.

The confidential access to databases is reinforced by two complementary mechanisms. First, queries are only allowed when issued from the storage nodes of the Virtualized Storage Zone. Second, any exchange is secured using encrypted roles or credentials, which are granted by using private networks.

In a Virtualized Storage Zone, each tier is an autonomous and independent self-managed storage system, which reinforces the control of the tier used to allocate files each time a user accesses the Virtualized Storage Zone.

From a local point of view, the clients send requests to its Trusted tier, which is in charge of negotiations with local and remote nodes. From a global point of view, each organization has a TrustedShared tier that is in charge of data distribution and synchronization with the federated storage network.

The approach includes, in each tier, a metadata controller for synchronization and to achieve data consistency in a centralized manner. A node of the Trusted tier is in charge of the local coordination mechanisms while a node or a set of nodes in the TrustedShared tier handle the remote coordination.

#### 3.2.1. Control access layer

This layer unifies the access to all the tiers (client included) by using a RESTful web client–server model.

A client-side API installed on the user computer synchronizes the user’s files stored in a given folder with the Virtualized Storage Zone. The client is also able to access it by using a web browser.

On the server-side, synchronization and web dashboard RESTful APIs are provided to attend respectively the requests produced by client-side API and those produced when the user stores/retrieves files by using a web browser. These APIs are replicated on storage nodes of the Virtualized Storage Zone, which means that all the nodes (managers included) work as single web domain that can attend to all the users of an organization. In order to clarify the work flow followed by users when accessing their files, we show in Fig. 4 an example of a Virtualized Storage Zone used by an organization called MyOrg.

Fig. 4 shows a single unified storage system in which the storage nodes have been configured as FHS. SNode0 and SNode1 belong to MyOrg and are included in the Trusted tier while SNode2 and SNode3 belong to other organizations and are included in TrustedShared.

As it can be seen, the users of that organization could be attended by both the storage nodes SNode0 and SNode1 from the Trusted tier and the storage nodes SNode2 and SNode3 from TrustedShared.

In this example, the manager of the Trusted tier is denoted by SNode0, while SNode2 is the manager of TrustedShared. Each storage node is in charge of a disk partition in which it stores the files or redundant information.

Fig. 4 also shows an example where a given user that belongs to MyOrg performs a PUT/GET operation of the file F1 by applying the following procedure:

1. A given user sends a request and her credentials to MyOrg domain, which contacts the Trusted manager to authenticate the user’s credentials and determine the best suited node to attend that request. The request is routed to the node SNode1, which forwards the corresponding string to the client. This string consists of a relative URL appointing at a valid temporal path to GET/PUT the file.

2. To complete the I/O operation of the original file (F1 in white), the API placed at the client tier anonymizes that file (F1 in gray) and starts the streaming of it by using relative URL.

Fig. 4 also depicts the procedure to be followed by the virtualized storage zone when the MyOrg site is down (steps 1’ and 2’). In this case, the user is attended by the storage nodes of the TrustedShared tier (federated backup service).

The access control layer applies the multi-tenancy policies for the whole domain. Each tier may have local storage for users with local permissions, but those data are not available in the whole...
domain. To share files, those files must be in the shared area (TrustedShared tier), where the global access rights are applied.

3.2.2. Multi-tenancy management layer

At this point, we have shown the procedure to store files in a Virtualized storage zone in which the disk partitions are used as black boxes. In this section, we describe the technique used to handle files into the disk partitions.

In order to isolate the users/organizations from other users/organizations, a multi-tenancy management layer has been included in the approach.

The multi-tenancy layer is depicted in Fig. 5, which shows the file system of a storage node. As can be seen, two disk partitions have been created, one for the Trusted tier and the other one for TrustedShared tier. The partition for the Trusted tier includes separated directories called virtual lockers, which has been assigned to the user accounts and each locker includes the files of the account owner. The TrustedShared includes a directory for each organization of the federation, which also includes a set of lockers containing redundant information. When a given user is attended by a storage node that belongs to her organization, that node must use the trusted partition. Otherwise, the information to be stored belongs to other organization and the storage node must use the TrustedShared partition for this operation. Fig. 5 shows how the manager stores the file F1 by using the Trusted partition while it stores the file R1, which is encrypted and anonymized redundant information from other organization by using the TrustedShared partition.

All the directories and partitions are created by using IDs, which are random keys at the expense of the managers of the corresponding tier. As a result, the approach can build access paths by concatenating the IDs produced by the multi-tenancy layer.

This logical structure is replicated in each storage node of the Trusted and TrustedShared tiers. As a consequence, a given user is able to store/retrieve her files in ATAW manner by using her virtual locker when accessing the single domain of the Virtualized Storage Zone even when her organization is unavailable.

This mechanism also allows the users to built virtual workgroup spaces, which are pools shared among users working for the same business unit of a given organization.

3.3. A data placement method to virtualized storage zone

We have designed a data placement method to allocate/locate files in the Virtualized Storage Zone by using the paths of IDs concatenated produced by multi-tenancy layer. For instance, Fig. 5 shows the concatenated paths the system could use to allocate/locate both files (F1) and redundancy (R1).

3.3.1. File locating method

The file locating method is based on a client–server metadata system in which the metadata server is placed in the tier managers while a metadata engine or metadata client is placed in all storage nodes of the Trusted and TrustedShared storage nodes.

The main function of the metadata server is to map fields such as the StorageNodeID, fileID, userID and TierPartitionID to a virtual locker generated by the multi-tenancy layer.

The following URL could be used to GET or PUT the fileID in ATAW manner.

StorageNodeDomain/TierID/userID/fileID

However, this is a security issue potentially affecting the users concerns about privacy (Bermbach and Klemz, 2011).

Instead, in our metadata model, the server constructs one unique service URL called StorageServiceKey and delivers it to the user. The StorageServiceKey is registered in the database of the metadata server as one single time service Key, which is a URL as the following one:

StorageNodeDomain.Engine.ServiceKey

where StorageNodeDomain is the domain of the chosen storage node to serve that request and ServiceKey is a random key generated by metadata server for identifying the Virtual Locker, and fileID for a given PUT/GET operation sent by the client-side API. This does not offer information about the real location of the Virtual Locker. Engine is the client-side RESTful API of the metadata system, which is in charge to serve the PUT/GET operation associated to the ServiceKey.

The storage nodes follow Algorithm 3.1 when receiving a StorageServiceKey. This algorithm describes the communication of the engine with the metadata controller and the procedure to receive/deliver files from/to the user.

Algorithm 3.1 (Procedure used by the storage nodes to allocate/locate files).

Input: ServiceKey; Role (encrypted credential for current Storage Node); MS = MetadaServer;
Output: Stream; (to GET/PUT file)
ResponseArray = Query to MC by using Role to find ServiceKey from ValidIOService table;
if ResponseArray != NULL then
TypeID = ResponseArray[type];
Locker = ResponseArray[LockerID];
File = ResponseArray[FileID];
RelativeURL = RandomDirectoryName;
Creates temp path for RelativeURL;
TOC [TypeID] = GET then
Move File to RelativeURL;
Stream_File,GET(RelativeURL); The Client API retrieves the file;
Query to MC by using Role to insert a new register in files (FileID, date, hour and Role);
else
Stream_File,PUT(RelativeURL); The Client API stores the file;
Move RelativeURL to Locker;
end
Query to MC by Role to delete ServiceKey in ValidIOService table;
Delete File in RelativeURL;
Auditing and Monitoring insert a new register in logs (date, hour, StorageServiceKey, and Role);
else
Request is rejected. Query to MC by using Role to insert a new register in the incidents table;
end

3.3.2. DALB: data allocation and load balancing method

The strategy to choose the best suited storage node for serving a given request is a key factor that could affect the overall system performance because this choice could produce unbalanced load or degraded services (Zikos and Karatzas, 2008; Stavrinides and Karatzas, 2010, 2010).

This section describes a resource management called DALB: Data Allocation and Load Balancing Method, which includes strategies such as storage nodes assignment, load balancing, and guidelines to determine when using each tier.

To handle this kind of strategies, a set of parameters must be defined. There are several parameters to be taken into account when managing online storage systems such as latency, size of the buffers and size of queues of each node, number of file downloads and uploads, size of the partitions, storage quotas or maximum file size.

In addition, SLAs (Service Level Agreements) include parameters such as cost per GB, cost per Upload/Download request, availability degree or maximum throughput, which are all required for load balancing observed by different sets of costumers when dealing with multi-tenant environments.

In order to simplify Virtualized Storage Zones administration, DALB concentrates all the parameters involved in performance and
service management into a single metric called Utilization Factor or $UF$. $UF$ represents the relative utilization of a given tier, node or partition, considering its full capacity. This means that a given component is saturated when $UF$ is equal to 1.

$UF$ is a quite intuitive metric both to choose the best qualified tier, node or disk partition for serving each request as well as to understand the utilization degree of each component of a Virtualized Storage Zone.

To choose a storage node in a Trusted tier, the manager calculates an $UF$ for all storage nodes in the tier. The DALB method calculates this metric according to the operation type to be performed because real-world traces show different ratios per type of operation (a GET ratio $= 80\%$ is quite common in storage services) (Juan Manuel Tirado, 2011) and different performance according to the stream rates (Gonzalez and Marcelin-Jimenez, 2011).

The following formula is used by the manager for calculating the $UF$ for a GET operation.

$$UF_{SN} = \frac{CBuff - (Cq + Cr)}{CBuff}$$

where $Cr$ is the file size of the GET operation, $CBuff$ is the memory assigned to the buffer of the task queue and $Cq$ is the capacity currently used from $CBuff$.

In the case of PUT operations, $Cq$ is critical in the selection of a storage node because this minimizes unbalanced load and storage saturation. $UF$ in this case is calculated by the following formula:

$$UF_{SN} = \frac{Cap - (CapUsed + Cr)}{Cap}$$

where $Cap$ is the capacity of the disk partition controlled by the storage node and $CapUsed$ the capacity used of that disk partition.

The storage node to be appointed will be the one with the minimum value of $UF$, over all the available storage nodes in the tier:

$$Choice = \text{Min}(UF_{SN,i})$$

In contrast, the selection of storage nodes in the TrustedShared tier must be negotiated because the nodes of this tier belong to different organizations, which have to settle a trade-off between the attention of their local users and the external users of the federation.

The approach establishes the following governance guidelines to store/retrieve files in the TrustedShared tier.

**Guideline 1**: The scheme limits the utilization of storage nodes only for serving push/pull redundant information required by data availability and fault-tolerance strategies.

In this process, a given Trusted manager requires to push redundancy information in the shared storage and the TrustedShared manager must find a storage node(s) of an organization willing to
receive that information trying to disturb as little as possible the normal operation of the Trusted tier of that organization.

**Guideline 2:** Each organization determines its cooperation degree with the federated scheme by assigning an amount of I/O operations that is able to receive by using the following metrics:

- **MeanRqtsPerDay:** This represents the mean amount of requests expected by a storage node in a day according to the logs of the resources database produced by monitoring system.
- **MeanRqtsPerHour:** This parameter represents the mean amount of requests expected by a storage node in a given hour.

**MeanRqtsPerDay** represents the maximum limit to receive requests and **MeanRqtsPerHour** the minimum one. The manager only calculates **UF** for Trusted tiers having \( (\text{MeanRqtsPerDay} - \text{MeanRqtsPerHour}) > 0 \), which means that these tiers are willing to cooperate by allocating redundant information.

The following formula is used by the TrustedShared manager for calculating UF of Trusted tiers:

\[
\text{UF}_{Ti} = \frac{\text{CurrentRqtsPerDay} - \text{CurrentRqtsPerHour}}{\text{MeanRqtsPerDay}}
\]

The manager precalculates the **UF** of each tier \( i \) by using a monitoring system, which negotiates every day both **MeanRqtsPerDay** and **MeanRqtsPerHour** per type of I/O operation with the managers of the Trusted tiers and calculates **UF** of **Ti**, limit.

\[
\text{UF}_{Ti}\text{, limit} = \frac{\text{MeanRqtsPerDay} - \text{MeanRqtsPerHour}}{\text{MeanRqtsPerDay}}
\]

When **UF** of **Ti**, limit, the Trusted Tier-\( i \) could be chosen to serve the current request.

When the **UF** of **Ti** is less than **UF** of **Ti**, limit, the push redundancy operation is registered in the pending task queue of the resources database. When a **UF** of **Ti** is zero this operation is performed.

Nevertheless, it might be the case that the smallest **UF** is not the best choice when the latency represents prohibitive costs. **FitValuePerRqts** is a fitness function used by the TrustedShared manager to ponderate the mean latency (**MeanLatency**), the mean free space in the buffers (**MeanFreeBuff_i**) of the storage nodes, and the **FileSize** of a given request.

\[
\text{FitPerRqts} = \text{UF}_{Ti} \times \frac{\text{FileSize} - \text{MeanFreeBuff}_i}{\text{MeanLatency}}
\]

The choice of the Trusted tier is determined by the maximum pondered benefit by calculating the **FitPerRqts** for all Trusted tiers with **UF** \( > 0 \) and per I/O type (Upload/Download).

**Choice** = Max(FitPerRqts(Tier \( j \), IOType))

Once the choice has been defined, the manager of chosen Trusted tier determines the storage node best suited for that request by using the **UF_SN**.

**Guideline 3:** The amount of redundancy that each Trusted tier can send to the TrustedShared tier is equal to the capacity of its partition in that tier, which has been defined when constructing the virtualized Storage Zone. This can be administered with a utilization factor for the redundancy reception at TrustedShared or **UF_R2TS**.

\[
\text{UF}_{R2TS} = \frac{\text{Cap}_{R2TS} - \text{CapUsed}}{\text{Cap}_{R2TS}}
\]

where

\[
\text{Cap}_{R2TS} = \sum_{i=0}^{n} SN_{P_i}
\]

**SN_{P_i}** is the size of the partition assigned to the organization into the TrustedShared tier and **CapUsed** is the sum of the file/chunk sizes that each Trusted tier has sent to the TrustedShared tier.

**Guideline 4:** In order to preserve autonomy from an arbitrary exercise of authority by a TrustedShared manager, each Trusted manager registers the metrics of its storage nodes in its resources database.

**Guideline 5:** When the TrustedShared tier manager declares a degraded mode because the site of a given organization is down, all the requests of the users of that organization are accepted in the TrustedShared tier.

In summary, the administrators of each organization cooperating in this federated system are able to know the amount of resources that their clients have used of each storage node of their Trusted tier by using **UF_SN** and the percentage or fraction that their storage services have consumed in the TrustedShared tier by using **UF_R2TS**. This is a quite intuitive approach to keep control on the consume of each storage service.

Resource allocation strategies such as round robin, pseudo-random or based on economics models can be used for choosing storage nodes by using **UFs**.

### 3.4. Virtualized storage zone functionalities

The functionalities used for serving I/O operations based on the multi-tier architecture are described in this section.

#### 3.4.1. Data availability

By data availability we mean the situation in which the storage service delivers a file requested by a given user.

We have defined three configurations to support data availability in the storage system.

- **Proactive Online:** In this configuration, a storage node pushes a replica into other storage node each time it receives a PUT (upload) operation. This configuration increases the data availability by \( n \), where \( n \) represents the number of replicas that have been distributed on the virtualized zone.

- **Reactive On-demand:** In this configuration, each time a storage node receives a GET (download) operation, it pulls a replica from other node. This configuration is commonly used during failure recoveries, but we propose to use it to improve data availability in the tiers of a zone. In this configuration the proactive actions are disabled (neither redundancy is produced, nor distributed when uploading files).

- **Proactive Deferred:** In this configuration, a storage node pushes replicas to other nodes, but during idle times.

These configurations can be applied in either Intra-tier (the replicas exchange is performed only in the Trusted tier), or Inter-tier (the replica exchange is performed in the TrustedShared tier).

Each replica is created as an original file, which means the manager delivers to the user the first file found by DALB (either original or a replica).

In the case of the user uploading a file with the same name of an already uploaded file, the approach handles the consistency by using versions of the files, which means each version is a new file that is represented with the name of previous file (random ID) plus the date and hour of the new version.

Fig. 6 shows the file version management in our storage approach. As it can be seen, the availability and fault-tolerant strategies are also applied to each version by producing replicas and IDAs chunks \( \{VF = R[1 \ldots n] | Uc[1 \ldots n] \} \).

The user decides whether she keeps or deletes previous versions when accessing her locker through the web dashboard.
This means the management of file versions is the responsibility of the owner. The system does not delete anything, as we do not have a policy to do that. A quota mechanism is used to control the storage space occupied by a user according to the organization policies and alarms are sent to the user by email when its virtual locker is getting close to quota.

3.4.2. Fault tolerance and archiving strategies

In this section, we describe the fault-tolerance strategies supporting failure recoveries on disks, nodes, and storage sites. Two basic fault-tolerance strategies are currently supported:

- **Simple replication**: This technique is based on a chained declustering strategy (van Renesse and Schneider, 2004; Gonzalez and Cortes, 2008) where a given file and its corresponding replica are allocated on two consecutive storage nodes, according to a cyclic or revolving order. This implies that the system spends an excess of 100% on redundancy overhead. Using this strategy, the system is only able to withstand one single node failure. We have implemented a chained declustering based on file versions, which means each time a given user uploads a file that has been previously uploaded, a new version of this file is automatically recorded.

- **IDA storage schemes** (Rabin, 1989): this strategy is based on the Information Dispersion Algorithm IDA (n, m) as a mean to provide information integrity (IDA is part of the Reed–Solomon family of error-correcting codes). Let $F$ be a file of size $|F|$, that is transformed into $n$ new files, called dispersed. Each dispersed file is of size $|F|/m$, where $n = m$. Dispersals are stored in $m$ different storage nodes that make part of the storage service zone. From the algorithm properties, it is granted that the original file can be reconstructed, provided that any $m$ dispersals are available. The combination of parameters ($n$, $m$) tolerates up to $(n - m)$ missing dispersals at the price of an excess of redundancy equal to $(n - m)/m$. Table 1 shows the amount of extra redundancy spent for $n$ servers when requiring $m$ dispersals (at least) to support fault-tolerance. We have implemented IDA with parameters (5, 3), this means each file is transformed into $n = 5$ dispersals and the file will remain available provided that any $m = 3$ dispersals are on hand. This implies that the system spends an excess of 66.7% on redundancy overhead. The combination (5, 3) presents a good compromise between reliability (2 chunks can fail), extra space needed for recovery (66.7%) and performance (the system distributes five chunks on the federated storage). The combination (4, 3) is better than the previous one in terms of capacity since it requires less extra space, but it only allows the system to withstand the failure of a single chunk. The combination (6, 4) offers a reliability similar to (5, 3) as well as a better deal in terms of capacity. However, in this combination the performance is affected by the costs of the distribution and computation of more chunks.

<table>
<thead>
<tr>
<th>Table 1</th>
<th>IDA parameters combination and its associated redundancy.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$ (servers)</td>
<td>$m = 1$</td>
</tr>
<tr>
<td>$n = 2$</td>
<td>100%</td>
</tr>
<tr>
<td>$n = 3$</td>
<td>200%</td>
</tr>
<tr>
<td>$n = 4$</td>
<td>300%</td>
</tr>
<tr>
<td>$n = 5$</td>
<td>400%</td>
</tr>
<tr>
<td>$n = 6$</td>
<td>500%</td>
</tr>
<tr>
<td>$n = 7$</td>
<td>600%</td>
</tr>
</tbody>
</table>

Fig. 7 shows a Virtualized Storage Zone of MyOrg in which a very sensitive file has been handled over the time by DALB method. In the first place an anonymous file was allocated in the Trusted tier (F1 in gray). A data availability strategy have distributed a replica of this file (R1 Gray) to other storage node in the Trusted tier. Simple replication based on deferred data availability has also been performed producing an anonymized encrypted replica to the TrustedShared tier (F1 in black) and finally an archival technique based on IDAs (5, 3) has been also applied by dispersing anonymized chunks on the multi-tier architecture ((C1..C4)). Notice that the number of chunks at the TrustedShared manager are less than the minimum necessary to recover a file. Therefore, an unauthorized user who had access to the dispersals at the TrustedShared tier, would be unable to recover the file by himself.

In both replication and codification, the manager is able to release the oldest files from the Trusted tier, and it consider the redundancy of the TrustedShared tier as an archival system. When the users require access to these files, the manager assumes a partial degraded status, reconstructs them from the redundant information in TrustedShared tier and relocates them in the Trusted tier.

Fig. 8 depicts a workflow that summarizes the operations performed in the Virtualized storage zone to store/retrieve files to the user. As it can be seen, the PUT operations could include additional actions according to the availability and fault-tolerant strategies that are required by a given user or file. In the case of GET operations, the basic idea is to obtain the file by the approach searching from a higher to a lower tier until to find either the file or its redundancy.

Table 2 shows similarities and differences between Global data persistent systems and the multi-tier approach. Table 2 shows our multi-tier approach covers the problem of cloud storage federation that is not addressed by other solution in this comparison. As it could be seen, Multi-tier approach includes implementation features for building fault-tolerant storage platforms in the cloud by using a REST API.

4. The implementation of private storage services prototype – a case study

We have implemented an ATAW storage service on four organizations placed at four cities, spanning two different countries: Mexico and Spain.

Each organization contributed to its virtualized zone with a set of hardware and logical resources, including PCs, servers or cloud computing instances, as well as logical volumes or folders. Each node was configured as a web server that was mapped to a public IP. Table 3 shows the features of storage resources shared by each organization.

We built four Virtualized Storage Zones by using the virtualization mechanism, on each organization. The zones are represented by the following acronyms: UC3M-Colme, UC3M-Cloud, UAM, and ITV.
Fig. 7. Data availability and fault-tolerant strategies schemes in the virtualized storage zone "MyOrg".

Fig. 8. The work flow in a virtualized storage zone.

Table 2
A qualitative comparison between global data persistent systems and the multi-tier approach.

<table>
<thead>
<tr>
<th>Target application</th>
<th>GFS: Grid file system</th>
<th>XtreemFS</th>
<th>OceanStore (Bindel et al., 2002)</th>
<th>GAS: Grid autonomic storage (Sánchez et al., 2012)</th>
<th>Multi-tier approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>File system</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fault-tolerant</td>
<td>Required (GFS)</td>
<td>Required (XtreemFS)</td>
<td>Required (OceanStore)</td>
<td>Required (MAPPS-DSI)</td>
<td></td>
</tr>
<tr>
<td>strategies</td>
<td>Replication</td>
<td>Replication</td>
<td>Replication and chunks</td>
<td>Replication</td>
<td></td>
</tr>
<tr>
<td>Programming API</td>
<td>Webdav</td>
<td>File system volumes</td>
<td>Not defined</td>
<td>Not defined</td>
<td></td>
</tr>
<tr>
<td>connexion Environment</td>
<td>GRID</td>
<td>GRID</td>
<td>F2P</td>
<td>GRID</td>
<td></td>
</tr>
<tr>
<td>Storage</td>
<td>Infrastructure</td>
<td>Infrastructure</td>
<td>Infrastructure</td>
<td>Infrastructure</td>
<td></td>
</tr>
<tr>
<td>service-oriented to Date-delivery</td>
<td>2011</td>
<td>2010</td>
<td>2000</td>
<td>2011 2012</td>
<td></td>
</tr>
</tbody>
</table>

Table 3
The features of the storage assets shared by each of the four organizations.

<table>
<thead>
<tr>
<th>Region</th>
<th>SC-ITV</th>
<th>SC-UAM</th>
<th>UC3M-Cloud</th>
<th>UC3M-Colme</th>
</tr>
</thead>
<tbody>
<tr>
<td>City</td>
<td>Northeast – Mex</td>
<td>Center – Mex</td>
<td>Madrid – Spain</td>
<td>Madrid – Spain</td>
</tr>
<tr>
<td></td>
<td>San Luis P.</td>
<td>Mexico City</td>
<td>Leganes</td>
<td>Colmenarejo</td>
</tr>
<tr>
<td>Number of nodes</td>
<td>7</td>
<td>6</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Type of nodes</td>
<td>Homogeneous</td>
<td>PCs</td>
<td>Virtual</td>
<td>PCs</td>
</tr>
<tr>
<td>Virtualized capacity</td>
<td>7TB</td>
<td>1.5TB</td>
<td>1TB</td>
<td>1TB</td>
</tr>
<tr>
<td>Operating system</td>
<td>Ubuntu</td>
<td>Fedora Slackware</td>
<td>OpenStack Ubuntu</td>
<td>Ubuntu</td>
</tr>
<tr>
<td>File system</td>
<td>Ext4</td>
<td>Ext2</td>
<td>Ext4</td>
<td>Ext4</td>
</tr>
<tr>
<td>Intratier network</td>
<td>FastEthernet</td>
<td>FastEthernet</td>
<td>GBEthernet</td>
<td>GBEthernet</td>
</tr>
</tbody>
</table>
The clients were provided with a multi thread java engine called HareSync, which is a synchronization API with the Virtualized Storage Zone.

All the storage nodes were furnished with the RESTful API called PSFile, which is a file hosting system that enables users to upload/download files, create work groups, add users to their work groups and share their files to other users or work groups. One storage node of each organization was configured as Trusted Manager. The TRUSTShared tier has been configured as a federated backup and archiving system, which gathers and coordinates the storage capacities of the four organizations, to be considered as a single domain. Using the virtualization mechanism, a node at the UC3M-Cloud Zone has been appointed to the role of TrustedShared manager.

Based on this unified interface, any user of any organization is supported regardless of the availability of her local Trusted tier.

5. Design of experiments

We have designed a testbed based on a set of scenarios in which the configurations allowed us to measure the virtualized storage zone functionalities. In this process we obtained a set of metrics to be analyzed.

5.1. Evaluation scenarios

We have implemented an I/O load injection API called Benchmark API, which is used to test the storage services through the client APIs (HareSync and PSFile). The Benchmark API produces PUT/GET operations and user credentials to the virtualized storage zones, which, in turn, assume that this artificial load comes from real and valid users.

We have designed two workload scenarios to test the virtualized storage zones.

Cumulative workload scenario

In this scenario, the Benchmark API sends a workload to the virtualized zones by duplicating the file size from 0.5 MB to 256 MB and the number of simultaneous requests varying from 1 to 10.

This scenario makes sense because it allows us to identify the performance parameters impacting on the ATAW storage services.

Real-world workload scenario

In this second scenario, the Virtualized Storage Zones are tested by strictly reproducing the behavior of real users recorded from a workload of an application currently in production.

We have taken the logs from the Phoenix web-storage API, which is a FHS application placed in a monolithic web site (Phoenix and prototype). This API has been in operation during one year, installed at the same facilities where the ITV virtual storage zone resides. The trace includes the work of students, teachers, staff members, and users from other organizations of neighbor cities. The logs include elements such as the identifier key (id-user) that is associated to a real user, the id-file associated to the real file (same than id-user), the operation type (upload/download), the file size and the timestamps (milliseconds of the date and hour when the operation was dispatched by the application). Notice that no personal information is registered in the logs of the Phoenix API and the identifier keys are only valid for testing purposes.

In order to stress the virtualized zones, we have taken the logs from the last four months and compress them in a 24-h trace. The features of the real trace are shown in Table 4.

Table 4: The features of the real trace from the Phoenix API.

<table>
<thead>
<tr>
<th>Number of reqs</th>
<th>5895</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean download file size</td>
<td>2.22 MB</td>
</tr>
<tr>
<td>Mean upload file size</td>
<td>2.54 MB</td>
</tr>
<tr>
<td>Max file size</td>
<td>264.90 MB</td>
</tr>
<tr>
<td>Mean amount of reqs/h</td>
<td>245</td>
</tr>
<tr>
<td>Number of uploads</td>
<td>1327</td>
</tr>
<tr>
<td>Number of downloads</td>
<td>4568</td>
</tr>
<tr>
<td>% of GET operations</td>
<td>77%</td>
</tr>
</tbody>
</table>

This trace allowed us to evaluate the impact of data distribution on the online storage services, the influence of the recovery strategies when solving both individual failures and the overall site failure, as well as the impact of the production and distribution of redundancy (simple replication and IDA codification) when applying disaster recovery strategies.

5.2. Functionality assessment and configurations

The data availability functionality was tested by using Proactive Online and Reactive On Demand configurations. A comparison with AmazonS3 service by using a free account was performed.

The fault-tolerant and Archival functionalities were tested by using Proactive Online, Reactive On Demand, and Deferred configurations.

5.3. Metrics

The following measurements were obtained during the aforementioned experiments:

- **Response Time**: This time was measured from the moment the user clicks the download/upload button until the moment the tier manager returns the control back to the user, meaning that her request has been successfully dispatched. It includes the file streaming (the time employed by the client API for uploading/downloading files and the network round trip latency) and the service time elapsed to perform the PUT/GET operation on the given file.

- **Service Time**: This is the time elapsed during the internal processing at the virtualized storage node. This is the sum of the time spent on metadata management, communications and file transfer. This also includes the time spent for redundancy computation, production, and distribution (if any).

- **Throughput**: The amount of MB/sec handled by the virtualized zones during PUT or GET operations.

6. Experimental evaluation

We analyzed the metrics produced by the approach functionalities when using the configurations previously defined.

6.1. Evaluation of the file availability strategies

In this section we show the critical elements affecting the performance of a Virtualized Storage Zone when offering file availability.

6.1.1. Analysis of cumulative workload scenario

We consider that the Virtualized Storage Zone performance has been correctly assessed by means of the service time. Meanwhile, the response time helps to estimate the user’s degree of satisfaction.

In order to understand the effects of the latency on the file availability, we compare the response time offered by the FHS configured in the storage nodes of the Trusted tiers of the Virtualized Storage Zone with a public account in AmazonS3. Fig. 9 shows in vertical axis the response time observed when uploading/downloading files in the FHS of the UC3M-Colme zone and AmazonS3 (Ireland Data Center). The horizontal axis shows the
file size according to the cumulative scenario. The FHS of UC3M-Colme zone and AmazonS3 both notify the user I/O operation is done when the last byte has been received at the storage node handling that request. This means the experiment does not include additional redundancy management.

In this experiment we can see in-house solutions produce less latency than public providers because of both users and servers of UC3M-Colme are located at the same region, which means a locality improvement. This means that an in-house FHS is feasible to handle the files of an organization.

Fig. 10 shows in vertical axis the response time for file uploads observed by one and ten users while the file size according to the cumulative scenario is shown in the horizontal axis. It also shows the cost of this operation when the UC3M-Colme zone FHS performs online replication in the file upload operation.

In this experiment, the FHS only returns the control to the user when a replica has been stored in a place that guarantees the file availability in one site failure scenario.

Fig. 10 shows the cost of the online replication is significant and this cost is increased by the amount of users performing simultaneously upload operations. This means scheduling policies should be studied for the availability and fault-tolerant strategies, which therefore, we have evaluated in the next sections.

Now we analyze the costs of the immediate redundancy production and distribution even within the Trusted tier by evaluating the service time of the strategies for delivering data availability at any storage node of the Trusted tiers. AmazonS3 is not included because we cannot analyze the service times of its servers.

Fig. 11 shows in vertical axis the service time of the UC3M-Colme Virtualized Storage Zone for the cumulative workload scenario. Fig. 11 shows the service times for both Uploads and Downloads produced by the Proactive Online and Reactive On-demand data availability configurations.

Proactive Online (Upload) and Reactive On-demand (Download) both push and pull replicas in online manner respectively, which also produces a delay when returning the control to the user. In contrast, the latencies associated to the Proactive Online (Download) and Reactive On-demand (Upload) configurations reflect the metadata overhead. The Reactive On-demand configuration is more suitable for improving performance while Proactive Online (Upload) for increasing the reliability for very sensitive workloads.

Fig. 12 depicts the same set of experiments as in Fig. 11 for 10 simultaneous users. We observe that this measurement follows a behavior pretty similar to that of the previous assessment. Nevertheless, there is an overhead produced by the concurrency.

Figs. 13 and 14 depict the same set of experiments as in Figs. 11 and 12 but now showing response times. As it can be seen, when increasing the number of users, the behavior of the tested configurations is similar to the ones shown in Figs. 11 and 12. There is however an overhead on the response time, which depends on both the distance between the user and the final service provider, where the file streaming takes place, as well as the number of simultaneous users in the Storage Node.

The production and distribution of redundancy as well as the streaming of the file from the user both represent the most critical
factors in response time. In contrast, the metadata management is the smallest part of the storage process, which is a constant of few milliseconds. In fact, in this experiment the manager spent in mean 133 ms (±5.96 ms) for all the file sizes of the load.

All the experiments in this section with this load were performed ten times and the points on the graph have 3.45% of error margin in average.

6.1.2. Analysis of real-world workload scenario

Once we have recognized the parameters that shape the virtualized storage zones’ performance, we then study the impact of workload injection on our system. Fig. 15 shows a bursty traffic producing increments/decrements in the number of requests according to the hour of the day. Similar behavior has been observed in publish-subscribe patterns (Juan Manuel Tirado, 2011) and file sharing applications (Waterloo, 2011).

IntraTier data availability costs

Fig. 16 shows in vertical axis the mean service time per hour observed at the UC3M-Colme virtualized zone for Proactive Online and Reactive On-demand configurations. Analyzing altogether uploads and downloads, it can be observed that the Proactive Online configuration increases the service time compared to the Reactive On-demand configuration. This can be explained from the fact that file and redundancy allocation are performed simultaneously. Although this procedure increases data availability, it also produces larger delays on the storage node queues.

InterTier data availability costs

Now, we will pay special attention to the distribution mechanism when the manager sends redundant information from the Trusted to the TrustedShared tier.

Fig. 17 shows in vertical axis the throughput for Proactive Online-Intertier, Proactive Online-Intratier and Reactive

---

**Fig. 13.** Response time for 1 user in UC3M-Colme zone according to the file size.

**Fig. 14.** Response time for 10 concurrent users in UC3M-Colme zone according to the file size.

**Fig. 15.** Requests per hour of the real trace of file hosting service in production.

**Fig. 16.** Mean service times per hour for UC3M-Colme with simple replication, intra-tier manner.

**Fig. 17.** Throughput per hour in UC3M-Cloud zone data availability configurations (simple replication in inter tier and intratier).
On-demand-Intratier, meanwhile horizontal axis shows the elapsed time of the experiment developed on the UC3M-Colme virtualized zone.

As it can be observed, Proactive Online-Intratier and Proactive Online-Intertier exhibit similar throughput. This can be explained from the fact that UC3M-Colme and UC3M-Cloud are separated by only two hops in the traced route and the quality of the network allows to distribute data by taking advantage of the DALB method, which means that sending I/O traffic to the TrustedShared tier also reduces the traffic sent to the Trusted tier.

Fig. 18 shows in vertical axis the throughput for Proactive Online-Intratier and Proactive Online-Intertier for an experiment developed on the ITV virtualized zone.

In this case, Proactive Online-Inter tier presents a major overhead because the closest alternative source to receive redundancy is at the UAM zone, which is seven hops away from the ITV in the traced route. As it can be seen, replication depends on the cost of the streaming latency. Under this circumstance, Proactive Deferred configurations are best suited to support the fault-tolerance strategy. Proactive Deferred actions reduce the effects on performance because they enforce the DALB method to choose storage nodes that are idle. This approach takes advantage of the time-zones.

In this section, we evaluated the Virtualized Storage Zones under extreme conditions.

6.2. The evaluation of site disaster recovery

Administrators from the organizations that took part in this project, admitted that blackouts are the most common failures they have to deal with. Every time this problem happens, it takes one to three hour of downtime in average.

In order to evaluate the effectiveness of the configurations applied to support fault-tolerance, the Colme-UC3M site was automatically turned-off from 7:00 to 10:00 am.

6.2.1. Simple replication strategy

Fig. 19 shows the mean service times for Proactive Online – UC3M-Cloud Zone, Reactive Online (Recovering) and Proactive Online – UC3M-Colme Zone in vertical axis, while the elapsed time of the experiment is shown in the horizontal axis. In this experiment, the requests from the UC3M-Colme users are forwarded through the HareSync or Web browser to the storage nodes of the TrustedShared tier, with enough redundant information to recover and deliver the files to the requesting users. We focused our attention on the storage nodes at the UC3M-Cloud virtualized zone, which have been chosen by the DALB method to receive redundancy from UC3M-Colme Zone because of the low latency between both sites. The Proactive Online – UC3M-Cloud and Proactive Online – UC3M-Colme configurations represent the performance of the Virtualized zones in normal operation. Reactive Online(Recovering) represents performance observed by the users concerning to UC3M-Cloud and UC3M-Colme organizations respectively during the recovering procedure.

Also notice that Proactive Online-Intratier takes advantage of the locality of organization sites but it produces overhead for both UC3M-Cloud and UC3M-Colme virtualized zones.

Fig. 20 shows mean service times for Proactive Online – UC3M-Colme, Reactive Online (Recovering) (same as in Fig. 19) and Reactive Online (Recovering-Deferred), which represents the recovering process when the Proactive-Deferred was applied to distribute files.

In this experiment, DALB has routed the users of the UC3M-Colme to the storage nodes of the ITV and UAM virtualized zones instead of the UC3M Cloud zone because of the time-zones, which means a difference of 7 h between countries. The users of the ITV and UAM virtualized zones are not disturbed by the recovery process because this process was performed overnight and only the users of the crashed Virtualized Storage zone have observed an overhead. This experiment shows that organizations participating in this study were able to withstand the downtime of a whole site,
while preserving the control on data management by using a cost-effective solution based on commodity components.

As we can see, deferred configurations depend on the hour in which the failure occurs. This could be solved by increasing the number of replicas, which also increases the possibility of taking advantage of the difference between time-zones.

We should notice that, for a given file, its availability can be compromised when the primary storage place in which it has been allocated is out of service. If a second failure on a redundant site happens, it may preclude the recovery of this resource. We say that on this condition, there is a window of time when the file recovery becomes vulnerable. Proactive Online configurations minimizes the vulnerability window when handling simple replication. In contrast, using Proactive Deferred the size of the vulnerability window is equal to the time in which the manager decides to send the replica to the TrustedShared tier. During this time a second failure would be a threat when handling simple replication. In this case DALB reconstructs files by using the IDA algorithm.

6.2.2. Archival based on IDAs strategy

The cost of this solution is the most considerable of the work flow followed when recovering failures in different tiers of the online storage system. In contrast, this archival strategy has supported two-site disasters or one disaster and the crashing of the client tier (tablet or notebook). In addition, this is the most dependable strategy for addressing the privacy and reliability concerns of the user.

Fig. 21 shows in vertical axis the mean service time according to the file size, which is shown in horizontal axis.

This experiment shows the costs observed when DALB obtains the file F1 by moving from a higher to a lower tier until finding either the requested file or its redundancy.

In Fig. 21, Trusted Replica represents the service time when DALB obtains F1 file by using a replica from other storage node of the Trusted tier (UC3M-Cloud zone) because the storage node in which F1 originally was allocated is unavailable. TrustedShared Replica represents the service time when the whole Trusted tier is unavailable (site disaster of UC3M-Cloud zone) and DALB recovers a replica of F1 from a storage node of other organization (UC3M-Colme zone) and returns it to the user. Finally, TrustedShared IDAs Archive represents the service times when two site failures have been observed and DALB reconstructs the F1 file from remaining sites that are still alive (ITV and UAM zones) by using IDAs archival scheme.

As we can see, the service times are increased each time DALB changes the tier for recovering the F1 file. This means, Client tier offers immediate data availability. In the Trusted tier the manager spends seconds when recovering a file Trusted Replica, more seconds spent when recovering a replica from TrustedShared (TrustedShared Replica) and a longest time is elapsed when recovering files by using archival solution (TrustedShared IDAs Archive).

Fig. 22 shows the throughput observed by the user in the same experiment shown in Fig. 21. A decrement in throughput per file size according to the recovery strategy is observed when DALB changes the tier for recovering F1 file.

In the case of strategies based on IDA codification, the intersection of activities could represent a critical or stressing period for the overall federated system. This period starts at 15:00 h (time of Madrid), which is also a moment when operations start in Mexico (there are 7 h between both countries), see Fig. 15.

Notice each stage of the recovering process from the client to the TrustedShared tier could be optimized in the current implementation.

7. Conclusions

This paper has presented the design, implementation, and performance evaluation of our solution for constructing a set of private online storage services as a single fault-tolerant system. This unified system is based on a hierarchical multi-tier architecture in which each tier represents a different storage service. The approach applies fault-tolerant and availability strategies to the files by passing redundant information among the tiers, which allows the unified system to face up different types of service failures. Our approach automates the construction of such a unified fault-tolerant system as well as the procedures to distribute data and overcome site failures.

Our case study implementation has revealed that the production and distribution of redundancy as well as the streaming of files are the most critical factors in the performance of the unified storage system. In contrast, the metadata management is the smallest load of the storage process, which represents few milliseconds for all the scenarios that were evaluated. The experimental evaluation revealed that the in-house storage prototype produced less latency than public solutions and this locality improvement was used to compensate the overhead produced when applying availability strategies to the files. The evaluation also revealed that the reactive availability configurations offered acceptable performance and reliability (suitable for regular workloads), the deferred configurations offered the best performance but the lowest reliability (suitable for non sensitive workloads) while the proactive configurations offered the best reliability but the worst performance (suitable for very sensitive workloads).

In the case study implementation, the unified system served file downloads and uploads even in the blackouts of two whole
sites by going down the hierarchy of the multi-tier architecture. As we expected, the costs associated to the data availability were increased each time system changed the tier when recovering a given file. The Client tier offered immediate data availability to the user, the unified system increased the response times when online recovering a replica by using the File Hosting Service placed at the Trusted tier. When this service was unavailable and the system going down to the TrustedShared tier, it spent more time than previous procedure when recovering a replica from the federated backup service. The longest time was elapsed when that file was recovered by using the archival service based on IDA codification, which is the last option of the system.

Organizations participating in this study were able to preserve in-house the control on the data management even when their users were attended by other organization. Intuitive metrics based on utilization percentages allowed the administrators to know the consumption of the storage service, even in federated scenarios and make decisions to meet growth needs.

8. On-going and future work

We have studied the fault-tolerance strategies addressed to recover the files of each user of an organization. We are working on a scheme where each user will be able to determine a set of attributes defining different degrees of file availability which, in turn, will be translated to different replication strategies. We also are adding the untrusted tier to the architecture by developing RESTful APIs of storage of public providers. This tier can be used for non sensitive data and during saturation of resources of a given tier. We are also studying strategies to achieve a trade-off between data availability, energy consumption and confidentiality.

Acknowledgments

This work has been partially funded by the Spanish Ministry of Science and Technology under the grant TIN2010-16497 “Input/Output Scalable Techniques for distributed and high-performance computing environments”. The authors who would like to thank ITV, UAMI and UC3M systems staff.

References


Amazon simple storage service (amazon s3). http://aws.amazon.com/s3/


Google, 2011 March. Software bug used to send malware. http://www.pcmag.com/ article/0/281723813168.00

Grid filesystem. https://docs.pbox.org/author/display/ISPNN/GRID+File+System


Phoenix prototype. http://www.adaptiveview.org/MxTreeStore


J.L. González received his PhD in Computer architecture from UPC Universitat Politècnica de Catalunya, Barcelona (2009). He was a visiting professor at Universidad Carlos III de Madrid, Spain and LTI-Cinvestav, Tamaulipas, Mexico. His research lines are Cloud-Based Storage systems, linguistic archival systems, federated storage networks. His expertise areas are the design of fault-tolerant, adaptability and availability strategies as well as task scheduling and storage virtualization.

Jesús Carretero Pérez is Computer Architecture Professor at the Polytechnical School of University Carlos III of Madrid. His is a senior researcher and his major research lines are high-performance computing and cloud computing, parallel and distributed systems, computational linguistics and real-time systems. He is also the director of the Master in Administration and Management of Information Systems and has published 17 books, has been involved in 52 research projects and he has written 200 journal and congress articles.

Victor Sosa-Sosa is a full-time research-professor in the Information Technology Lab at Center for Research and Advanced Studies of the National Polytechnic Institute of Mexico (CINVESTAV – IPN). Currently, he is a visiting researcher in the Database and Information System group at Max Planck Institute für Informatik in Germany. He has a PhD in Computer Science from Technical University of Catalonia (UPC-Barcelona) and was a member of the Mexican National System of Researchers 2002–2012. He was professor and researcher in the Distributed Systems Group at CENIDET-Mexico until 2006. Participant and responsible of research and development projects funded by the National Council for Science and Technology of Mexico (CONACYT) and private companies. He has been advisor of more than 15 master’s thesis and 2 PhD thesis. Author and co-author of several technical contributions in international conferences and/or journals. His research interest and specialization areas are related to Distributed Information Systems and Databases. Recently, his work is more focused on distributed information at Web scale, especially on topics related to efficient information search, storage and analysis, integration of Web query interfaces (DeepWeb) and knowledge harvesting.

Juan F. Rodríguez Cardozo received a master of Engineering in Computer Science from the Carlos III University of Madrid, which is linked to the profession of Computer Science Engineer and combines advanced education in issues on leadership as well as management aspects with other IT trends. His areas of interest are research, design and implementation of distributed systems based on grid and cloud platforms aimed to deploy HPC solutions. He is currently working on Design and deployment of private cloud architecture based OpenStack Cloud Software, Infrastructure as a Service (IaaS).

Ricardo Marcelín Jiménez received a degree in Electronics Engineering from the Autonomous Metropolitan University (UAM) in 1987, and a PhD in Computer Science from the National Autonomous University of Mexico (UNAM) in 2004. He is a full professor at the Department of Electrical Engineering at the UAM. His research interests are in the theory and practice of distributed computing, especially the issues related to coordination and fault-tolerance.