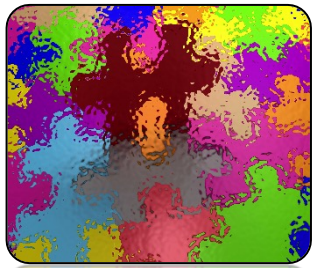
Introduction to Cloud COMPUTING

Giuliano Taffoni - I.N.A.F.

"**Data Cloud and Cloud Security**"

2023 @ Università di Trieste

Outline



Data Cloud

Cloud Security

Overview

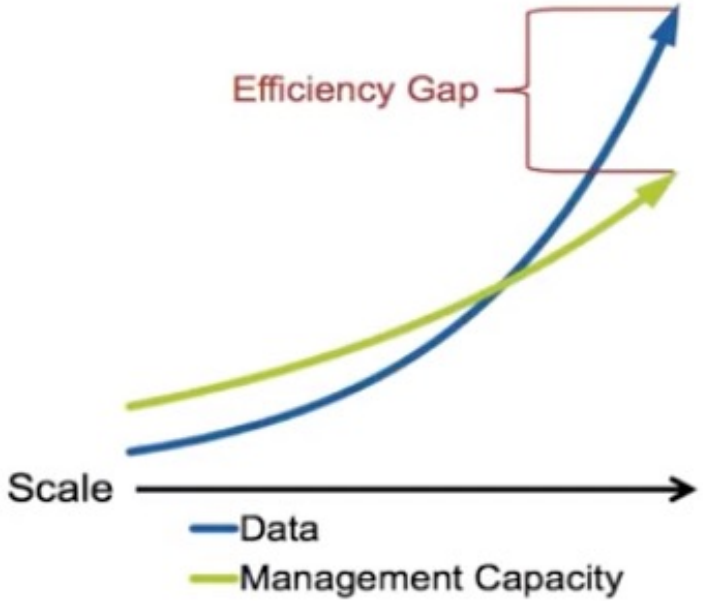
- Traditional Storage architectures
- Cloud storage theory and implementations
- Cloud storage and storage in the Cloud.
- Cloud storage and Cloud Distributed storage.

Data Efficency gap

Cloud storage is a service model in which data is maintained, managed and backed up remotely and made available to users over a network (typically the Internet).

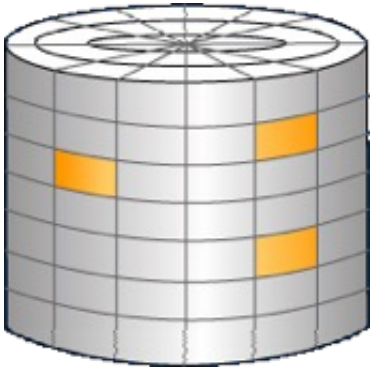
Data explosion and Mobile device growth

- improve scalability (up+out) and security
- improve performance
- simplify storage management
- on demand access
- unstructured data

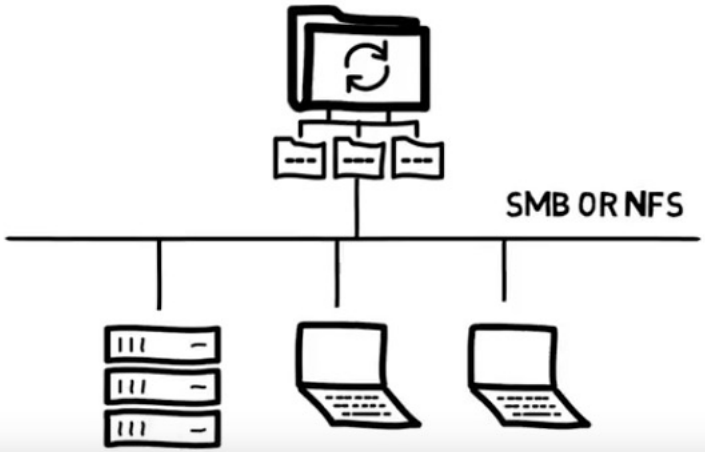


Traditional Storage Architecture

Block Storage



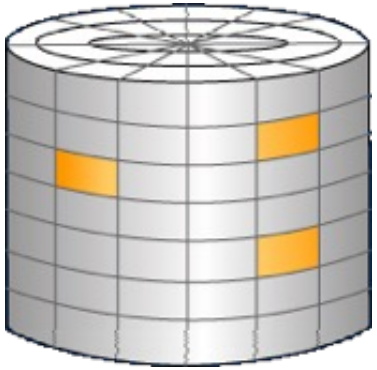
File Storage



Block Storage

Block storage

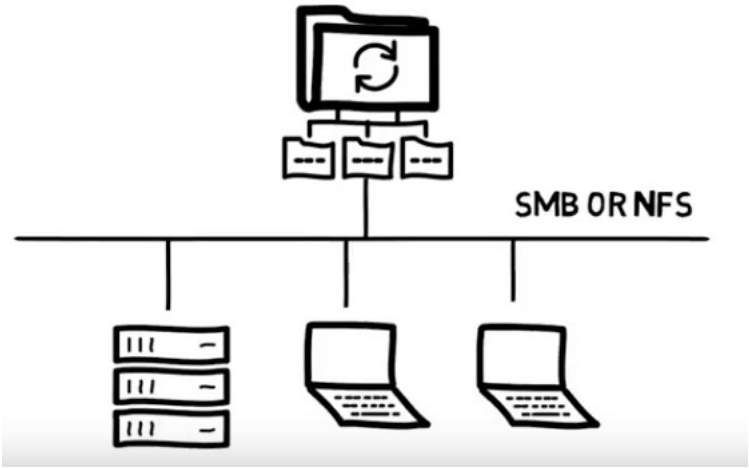
- Volumes
- Blocks (read and Write)
- Fibre Channel or iSCSI protocol
- Local
- Low Latency, high IOPs, low size (<1PB)
- Complex to expand and expensive



File Storage

File Storage

- Files and directories
- Network Shared (LAN)
- SMB, NFS, OCFS etc
- High throughput, large size (PBs)
- Scale out capabilities
- Multi-tiered architecture
- Expensive



Distributed FileSystem

"File system that is shared by many distributed clients
The resources (file+dir) on a particular machine are local to itself. Resources on other machines are remote
Basic layer for many distributed systems (clients) and applications"

A DFS provides a service for clients. The server interface is the normal set of file operations: create, read, etc. on files.

Servers allow clients to perform operations on resources that resides on servers.

Distributed Filesystem Properties

Transparency:

- Location: a client cannot tell where a file is located
- Migration: a file can transparently move to another server
- Replication: multiple copies of a file may exist
- Concurrency: multiple clients access the same file

Flexibility

- Servers may be added or replaced
- Support for multiple file system types

Distributed Filesystem Properties

Dependability

- Consistency: conflicts with replication & concurrency
- Security: users may have different access rights on clients sharing files & network transmission
- Fault tolerance: server crash, availability of files

Performance

- Requests may be distributed across servers
- Multiple servers allow higher storage capacity

Distributed Filesystem Properties

Caching

- Reduce network traffic by retaining recently accessed disk blocks in a cache, so that repeated accesses to the same information can be handled locally.
- If required data is not already cached, a copy of data is brought from the server to the user.
- Perform accesses on the cached copy.
- Files are identified with one master copy residing at the server machine,
- Copies of (parts of) the file are scattered in different caches.

Cache Consistency Problem -- *Keeping the cached copies consistent with the master file*

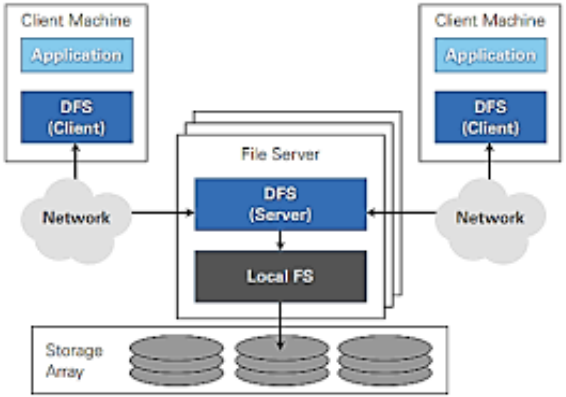
Distributed Filesystem Properties

Ideally, the client would perceive remote files like local ones.

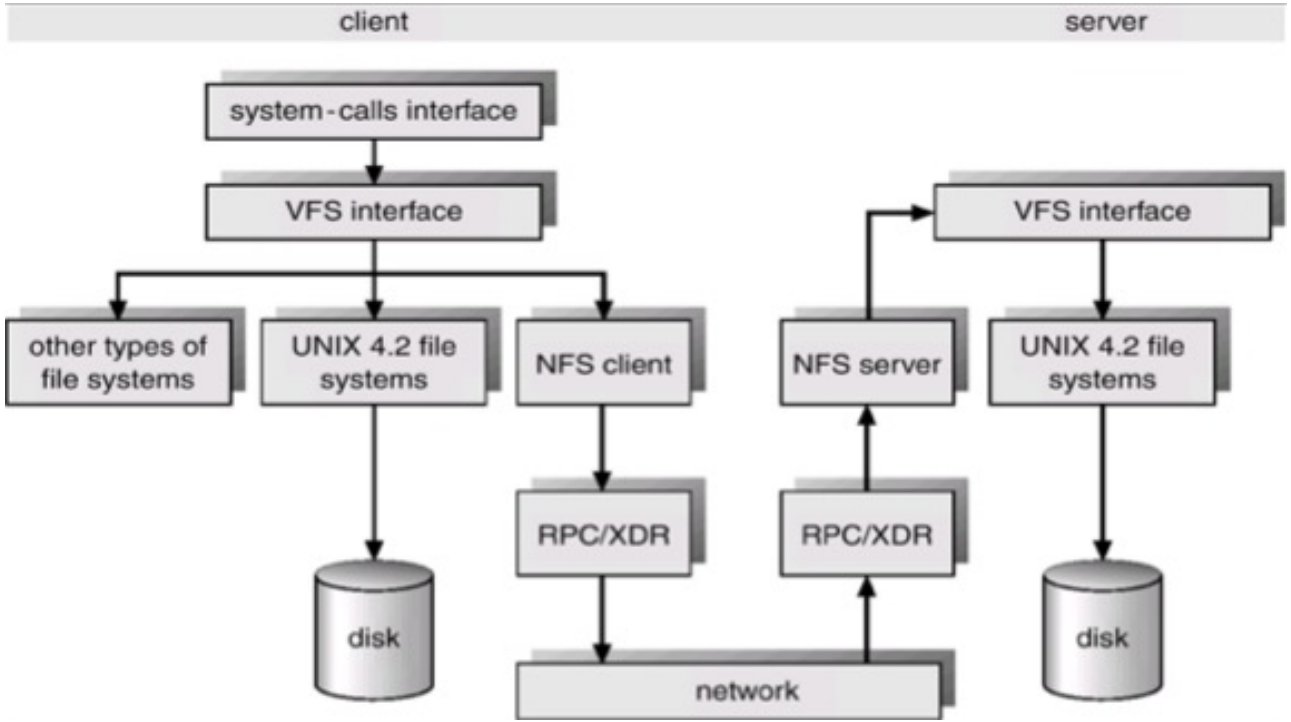
Clients, servers, and storage are dispersed across machines.

Configuration and implementation may vary:

- Servers may run on dedicated machines, OR
- Servers and clients can be on the same machines.



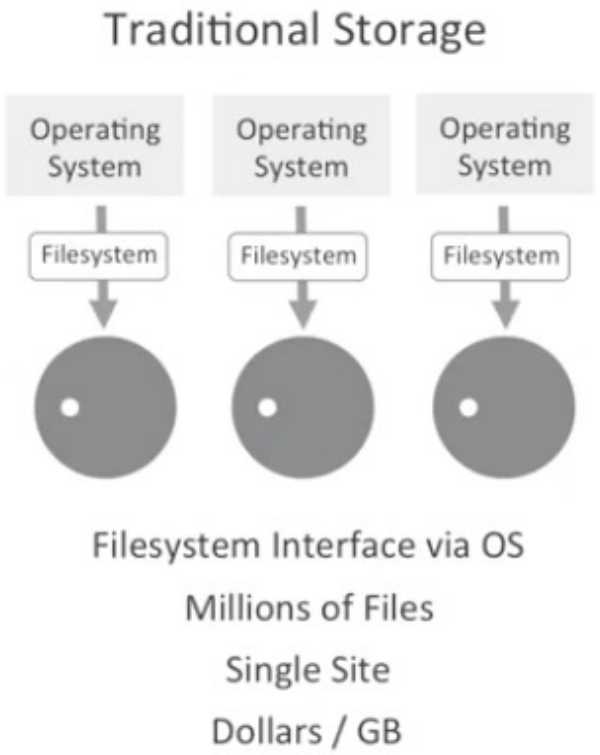
Network filesystem (NFS)



Limitations of traditional Storage

- Handle increasing number of files and users
- Growth over geographic and administrative areas
- Growth of storage space
- No central naming service
- No centralised locking
- No central file store

Applications must be aware of volumes/data location
Data is structured and isolated in Filesystems and volumes



Google File system

Motivation

One single distributed file system
Store big data reliably
Allow parallel processing of big data

Assumptions

Inexpensive components that often fail
Large files (million of files 100+MB)
Large streaming reads and small random reads (500Mb/s read/write load)
Large sequential writes
Multiple users append to the same file
High bandwidth is more important than low latency.

GFS interface

No common standard like POSIX.

Provides familiar file system interface:

`Create, Delete, Open, Close, Read, Write`

Snapshot: low cost copy of a whole file with copy-on-write operation

Record append: Atomic append operation

GFS Design Overview

Files split in fixed size chunks of 64 MByte

Chunks stored on chunk servers

Chunks replicated on multiple chunk servers

GFS master manages name space

Clients interact with master to get chunk handles

Clients interact with chunk servers for reads and writes

No explicit caching

GFS Design Overview

Master server: Single master - Keep metadata - accept requests on metadata - Most management activities

Chunk servers: Multiple - Keep chunks of data- Accept requests on chunk data

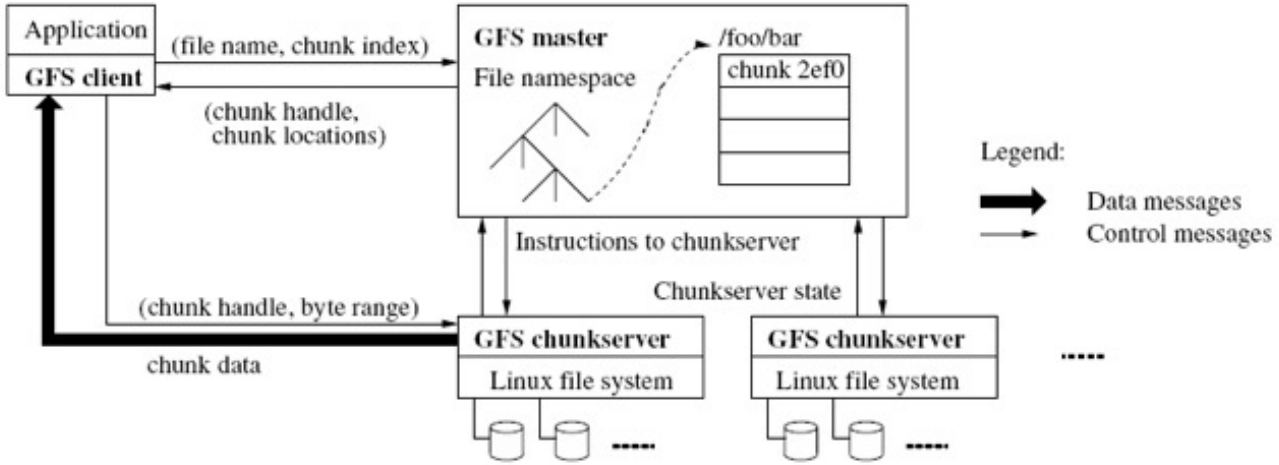
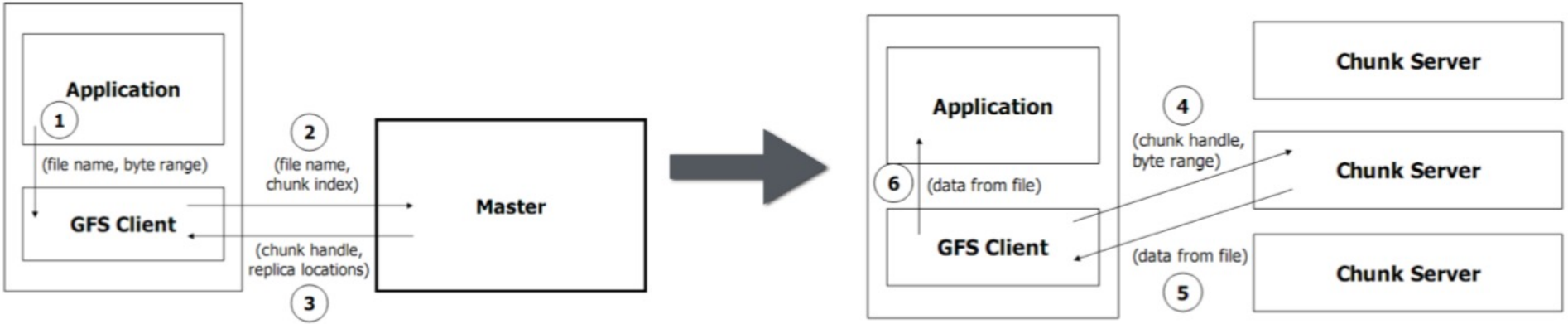
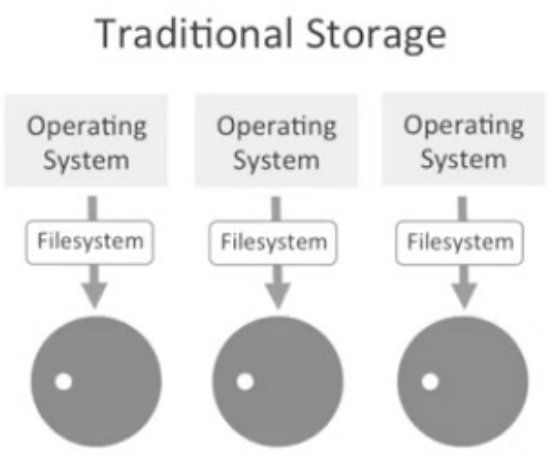


Figure 1: GFS Architecture

GFS example: read files



Object storage

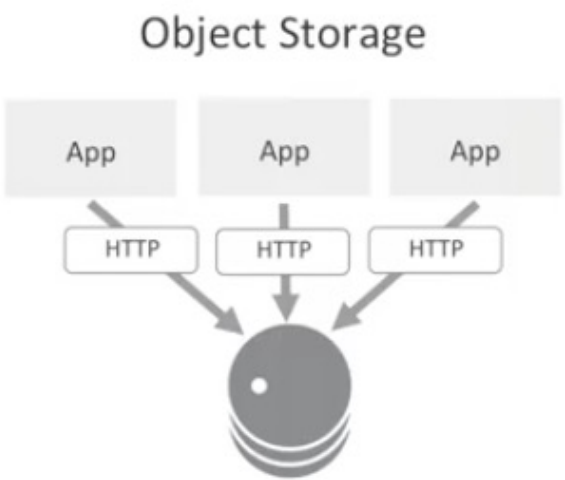


Filesystem Interface via OS

Millions of Files

Single Site

Dollars / GB



HTTP API directly accessible by Applications

Billions of Files

Geo-Distribution

Cents / GB

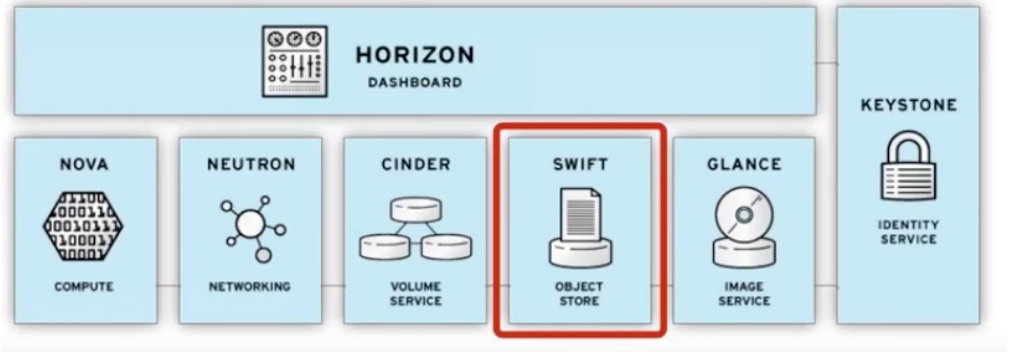
Example of Object storage

Public cloud services

- Amazon S3
- Google Storage (not Google Drive!!!)
- HP Cloud Object Storage
- etc…

Object Storage Systems

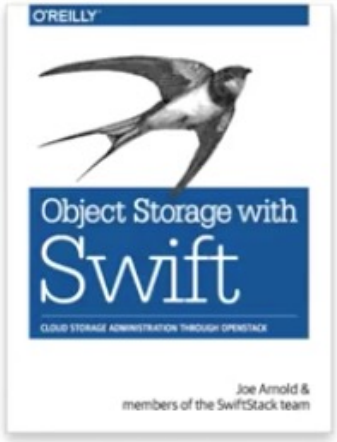
- OpenStack Object Storage System (swift)
- CHEP



SWIFT

OpenStack Swift

SwiftStack



- OpenStack Swift – Worlds most popular Object Storage System
- Powers extremely large storage clouds
- 100% Open-source (Apache 2)
- Rich ecosystem of tools and applications
- SwiftStack is core developer and leads project
- +180 developers today. Contributors include:

SwiftStack redhat IBM intel rackspace the open cloud company hp

What is an Object

An object is a logical unit of storage

- ID (Identification)
- Application data
- Metadata which includes block allocation and length
- Attributes that is accessible by users

Objects have file-like methods

`open, close, read, write`

Object = File + Metadata

What is metadata

Describes the object
- Helps you to find the right one
- Tells you what it is
- The specifications
- Used where and when
- Access permissions

Any and all objects
- Different attributes per object
- Add attributes later

Name
utf8Name
type
size
hashScheme
hash
retention
retentionString
retentionClass
changeTimeString
ingestTime
ingestTimeString
accessTime
accessTimeString
updateTime
updateTimeString
uid
gid
permissions
hold
shred
dpl
index
replicated
customMetadata
acl
namespace
objectPath
owner

Astronomical Data Analisys

Use Case:
Cost effective On-line archiving enriched with metadata.

Challenge:
-Astronomical FITS files are self descriptive
-Content needs to be available for image reduction
-Content must be content of a permanent archival record securely stored

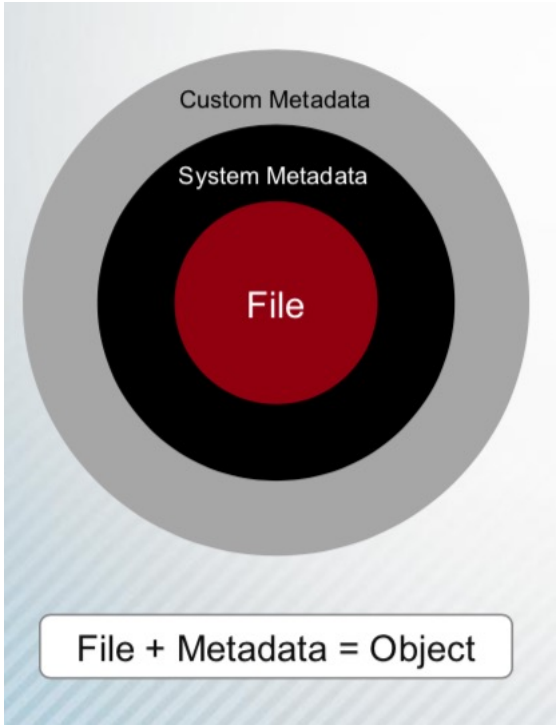
FITS files

***FITS**. Flexible Image Transport System (**FITS**) is an open standard defining a digital **file format** useful for storage, transmission and processing of data: formatted as N-dimensional arrays (for example a 2D image), or tables. **FITS** is the most commonly used digital **file format** in astronomy.*

Data Reduction in python example

https://github.com/gtaffoni/Learn-Python/blob/master/Lectures/PythonLecture05-Astronomy_Data_Reduction.ipynb

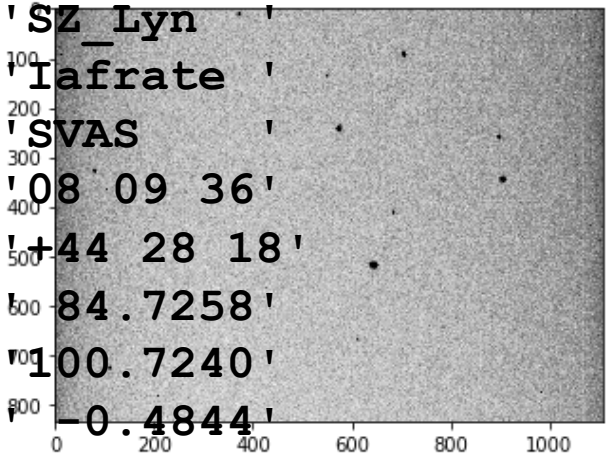
Objects and Metadata




```
File class = image

Filename = image.FITS
Created = Oct  3, 2018 13:29:59
Last modified: Oct  10, 2018
11:00:00

OBJECT  = 'SZ_Lyn   '
OBSERVER= 'Iafrate  '
TELESCOP= 'SVAS     '
OBJCTRA = '08 09 36'
OBJCTDEC= '+44 28 18'
OBJCTALT= '84.7258'
OBJCTAZ = '100.7240'
OBJCTHA = '-0.4844'
```



Mining Metadata

Metadata:

- stored with the object in extend attributes
- can be changed, removed or added (later)
- can be indexed

Identify sets of related objects based on system and custom metadata

Understand the object store – gather object or content metrics on sets of objects based on metadata

- Content discovery
  - Return all files owned by "Bob Smith" created after "3/15/2011" with unexpired retention
  - Return objects with a specific retention class defined
  - Return objects under retention hold, for mitigation purposes
- Metrics gathering
  - What is the size distribution of files in the system?
  - What percentage of my files are Word documents?
  - Which owners have generated the most content?

Access an object

Objects have a 64 bit unique ID

Objects live on a flat namespace

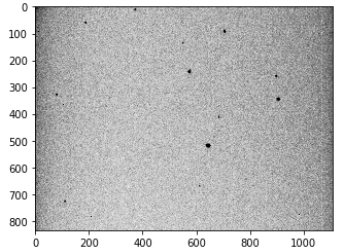
Request for storage are made with HTTP using RESTfull APIs (or SWIFT client)

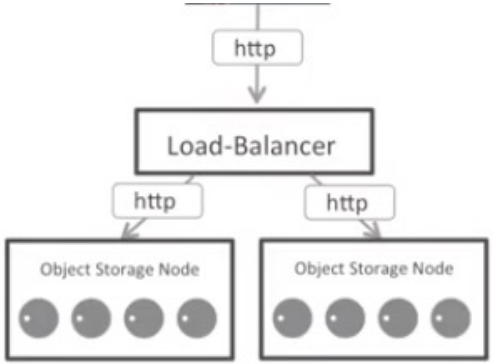
Three primary components of the request:

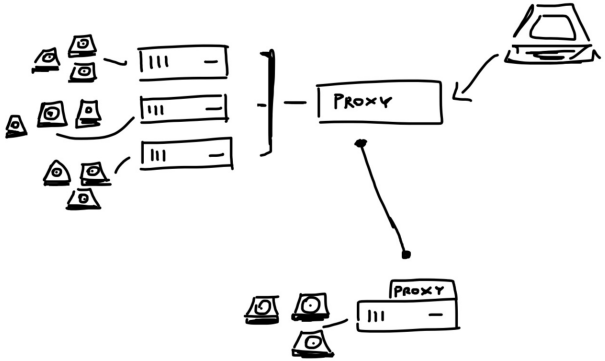
1. HTTP Verb (PUT, GET, DELETE, etc.)
2. Authentication Information
3. SURL
4. Metadata (*Optional*)

HTTP API and SWIFT Architecture

PUT http://XXX.XXX.XXX.XXX/v1/AUTH_application1/pic/image.fits







Manage Object with HTTP

- **Storage area** within the cluster
- NOT a user account
- Each account has its own URL

Swift is multi-tenant

- Namespaces used to group objects within an account
- Containers are unlimited
- Like folders, but can't nest them

- File + Metadata
- Each object is addressed as a URL
- Users name the object
- Objects are not organized based on hierarchy
- Instead, object names may contain "/", so pseudo-nested directories are possible.

Method	Description
GET	Downloads an object with its metadata
PUT	Creates new object with specified data content and metadata
COPY	Copies an object to another object
DELETE	Deletes an object
HEAD	Retrieves object metadata
POST	Creates or updates object metadata

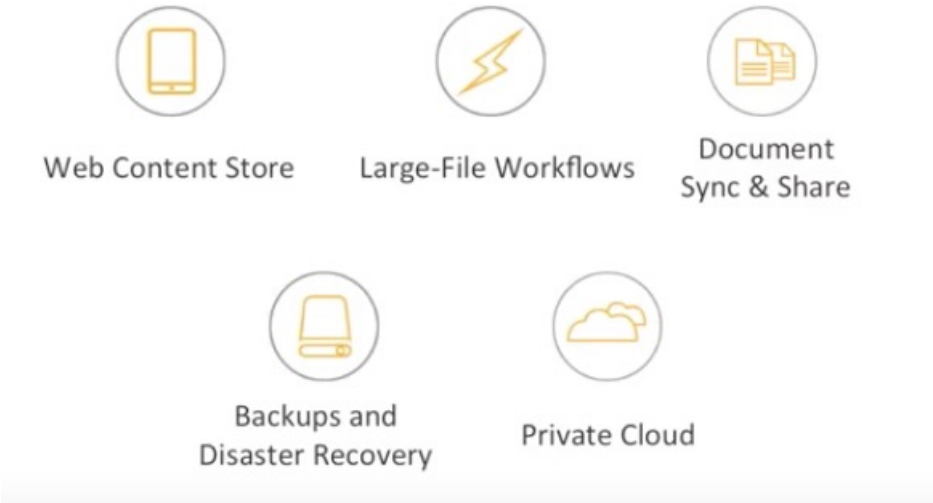
There are also API operations for Accounts and Containers.

Object Storage Usecase

Video streaming and processing

Map-Reduce

Astronomical data analysis



Cloud Security

Outline

Introduction

Infrastructure security

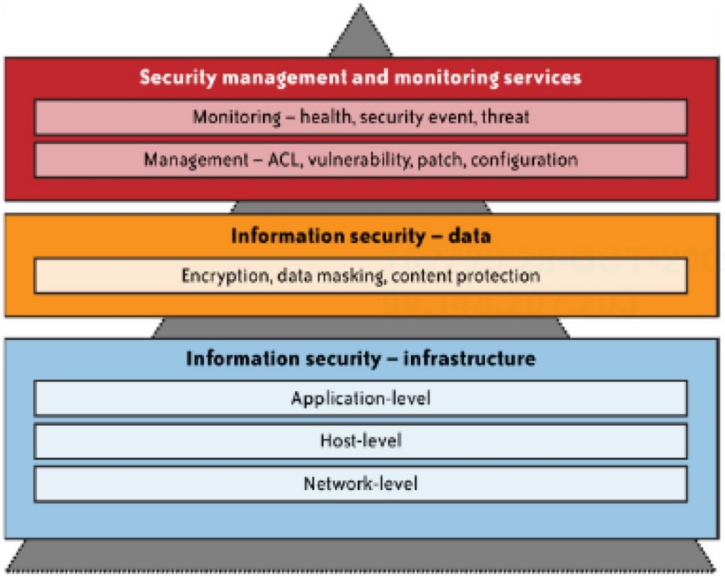
Data security

Identity and access management

Introduction

What should I do when I use the cloud and I want some privacy?

Many security problems in non-cloud environment are still applicable!



General considerations

Security boundary separating the client's and vendor's responsibilities.

Different types of cloud computing service models provide different levels of security

Focus on public clouds

Different levels:

- Network level
- Host level
- Application level

What is necessary on a cloud

Auditing

Data integrity

e-Discovery for legal compliance Privacy

Recovery

Regulatory compliance

Before approaching the Cloud

- Determine which resources (data, services, or applications) you are planning to move to the cloud.
- Determine the sensitivity of the resource to risk.
- Risks that need to be evaluated are loss of privacy, unauthorized access by others, loss of data, and interruptions in availability.
- Determine the risk associated with the particular cloud type for a resource
- Take into account the particular cloud service model that you will be using.
- If you have selected a particular cloud service provider, you need to evaluate its system to understand how data is transferred, where it is stored, and how to move data both in and out of the cloud.

Security Boundary

Who has responsibility for a particular security mechanism?

Where is the boundary between the responsibility of the service provider and the responsibility of the customer?

Cloud Security Alliance: CSA is an industry working group that studies security issues in cloud computing and offers recommendations to its members.

Multi- tenancy security

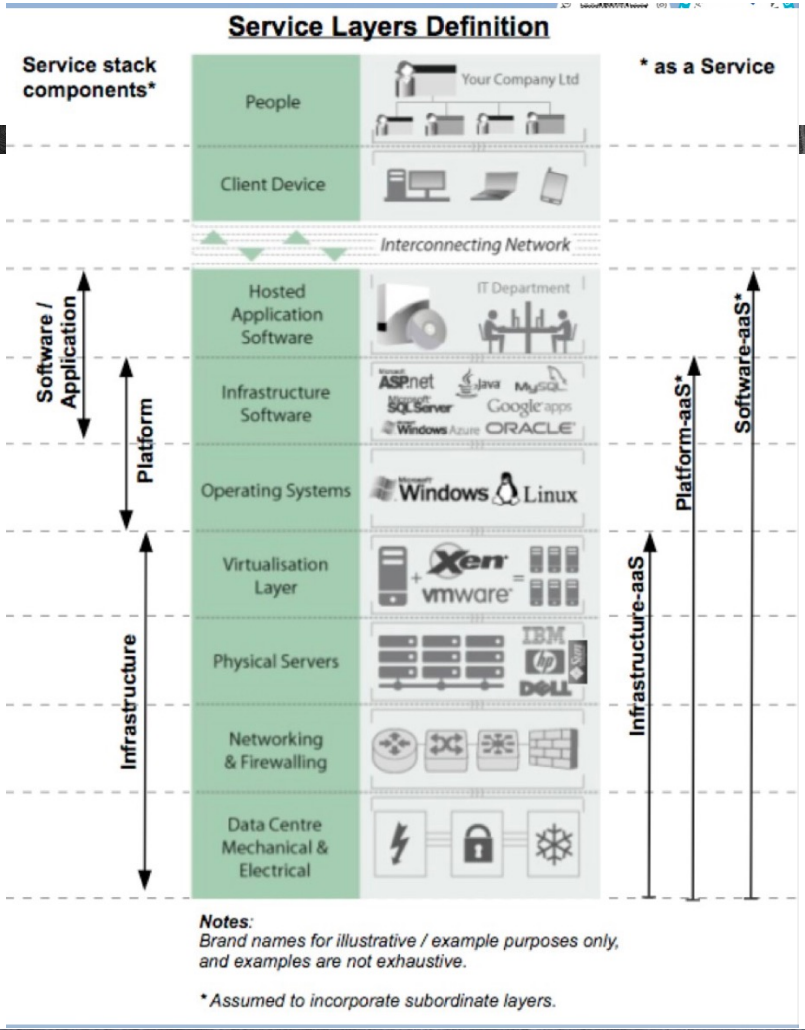
Different customers must be isolated,

Customer data segmented,

Customer services identified and isolated.

Multi project approach (e.g. OpenStack tenants)

XaaS Boundaries



The platform security level is reduced to the security level of the most vulnerable application running on the platform. (isolate with VM and containers)

Network Level Security

Confidentiality and integrity of data-in-transit
Amazon had security bugs with digital signature on SimpleDB, EC2, and SQS accesses (in 2008)

Less or no system logging /monitoring
- Only cloud provider has this capability
- Thus, difficult to trace attacks

Reassigned IP address
- Expose services unexpectedly
- Spammers using EC2 are difficult to identify

Availability of cloud resources
Some factors, such as DNS, controlled by the cloud provider.

Physically separated tiers become logically separated E.g., 3 tier web applications

Host Level

- Hypervisor security
  - “zero-day vulnerability” in VM, if the attacker controls hypervisor

- Virtual machine security
  -ssh private keys (if mode is not appropriately set) -
  - VM images (especially private VMs)
  - Vulnerable Services

VM/Containers security advantages

Simple way to implement resource management policies

Improved intrusion prevention and detection.

Secure logging and intrusion protection.

More efficient and flexible software testing

VM/containers security risks

- Explosion of number of VMs
- Snapshots (roll back to state that can be exploited)
- Shared Images
  - backdoors and leftover credentials,
  - unsolicited connections,
  - Malware

22% of the scanned Linux AMIs contained credentials allowing an intruder to remotely log into the system.

OS Security

Implement minimal security on: Access control, authentication usage, and cryptographic usage policies are all elements of mandatory OS security.

Applications with special privileges that perform security- related functions are called trusted applications. Such applications should only be allowed the lowest level of privileges required to perform their functions.

An OS poorly isolates one application from another, and once an application is compromised, the entire physical platform and all applications running on it can be affected.

Data Security

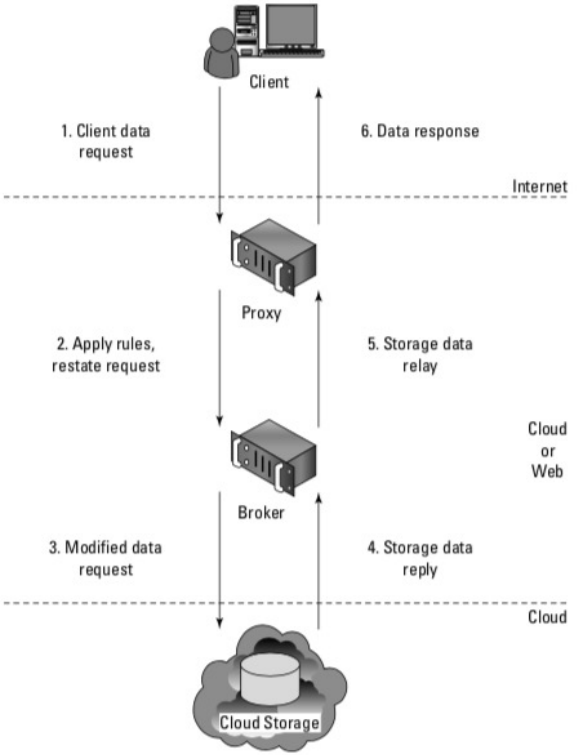
Identify the **security boundary** separating the client's and vendor's responsibilities
Determine the **sensitivity** of the data to risk
Data should be transferred and stored in an **encrypted format**.
Separate clients from direct access to shared cloud storage.
These are the key mechanisms for protecting data mechanisms:

- Access control
- Auditing
- Authentication
- Authorizations

Data Segregation and Isolation

Isolate data from direct client access creating a layered access to the data.

Data segregation based on tenants



Encryption

Most cloud service providers store data in an encrypted form (e.g. Amazon S3 256-bit Advanced Encryption Standard) on server side or client side.

Some example of java code here:

https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingClientSideEncryption.html

Problems:

- a problem with encrypted data may result with data that may not be recoverable.
- it does nothing to prevent data loss: keep your keys!!!!!

that’s all, have fun