UNIT IV

Storage Systems: Evolution of storage technology, storage models, file systems and database, distributed file systems, general parallel file systems. Google file system. Apache Hadoop, Big Table, Megastore, Amazon Simple Storage Service(S3) , Cloud Security: Cloud security risks, security – a top concern for cloud users, privacy and privacy impact assessment, trust, OS security, Virtual machine security, Security risks.

Storage Systems:

- Storage and processing on the cloud are intimately tied to one another.
- Most cloud applications process very large amounts of data. Effective data replication and storage management strategies are critical to the computations performed on the cloud.
- Strategies to reduce the access time and to support real-time multimedia access are necessary to satisfy the requirements of content delivery.
- Sensors feed a continuous stream of data to cloud applications.
- An ever increasing number of cloud-based services collect detailed data about their services and information about the users of these services. The service providers use the clouds to analyze the data
- Humongous amounts of data - in 2013 The Internet video will generate over 18 EB/month.♦ Global mobile data traffic will reach 2 EB/month.♦ (1 EB = 10^{18} bytes, 1 PB = 10^{15} bytes, 1 TB = 10^{12} bytes, 1 GB = 10^{12} bytes)

A new concept, “***big data***,” reflects the fact that many applications use data sets so large that they cannot be stored and processed using local resources. The consensus is that “big data” growth can be viewed as a three-dimensional phenomenon; it implies an increased volume of data, requires an increased processing speed to process more data and produce more results, and at the same time it involves a diversity of data sources and data types.

Evolution of storage technology:

The technological capacity to store information has grown over time at an accelerated pace.

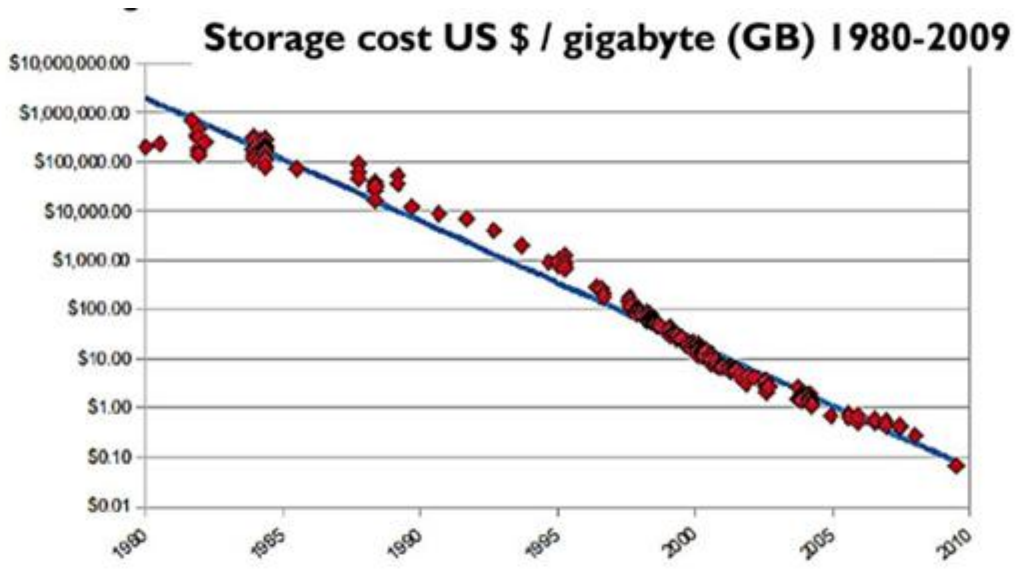
- 1986: 2.6 EB; equivalent to less than one 730 MB CD-ROM of data per computer user.
- 1993: 15.8 EB; equivalent to four CD-ROMs per user.
- 2000: 54.5 EB; equivalent to 12 CD-ROMs per user.
- 2007: 295 EB; equivalent to almost 61 CD-ROMs per user.

Hard disk drives (HDD) - during the 1980-2003 period:

- Storage density of has increased by four orders of magnitude from about 0.01 Gb/in^2 to about 100 Gb/in^2
- Prices have fallen by five orders of magnitude to about 1 cent/MB.
- HDD densities are projected to climb to 1,800 Gb/in^2 by 2016, up from 744 Gb/in^2 in 2011.

Dynamic Random Access Memory (DRAM) - during the period 1990-2003:

- The density increased from about 1 Gb/in^2 in 1990 to 100 Gb/in^2.
- The cost has tumbled from about $80/MB to less than $1/MB.



Since 2010, cloud-based storage has become the new norm for digital storage — one that's more secure and doesn't require new hardware if a business' needs change. While initial growth may have been slow, in the last 10 years, cloud services have expanded significantly.

By 2010, Amazon, Google, Microsoft, and OpenStack had all launched cloud divisions. This helped to make cloud services available to the masses. Since then, cloud services have taken over a large part of the tech industry and cloud transitions or migrations have become common.

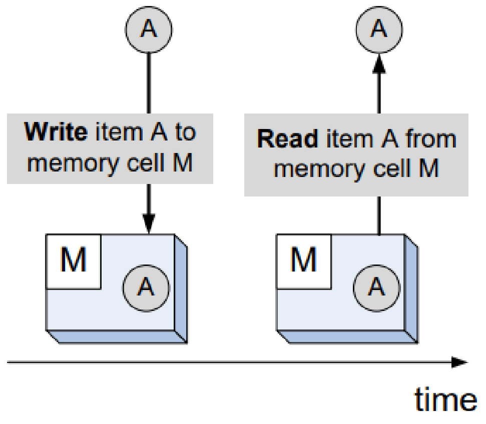
Storage models, file systems, and databases

A **storage model** describes the layout of a data structure in physical storage; a **data model** captures the most important logical aspects of a data structure in a database. The physical storage can be a local disk, a removable media, or storage accessible via a network.

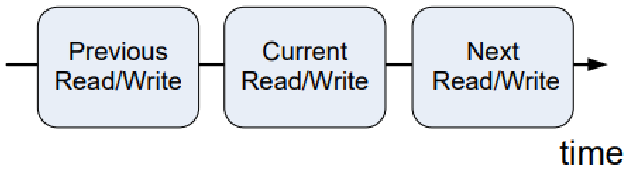
Two abstract models of storage are commonly used:
1. Cell storage
2. Journal storage

Cell storage - assumes that the storage consists of cells of the same size and that each object fits exactly in one cell eg. Cell 1, Cell 2...

This model reflects the physical organization of several storage media; the primary memory of a computer is organized as an array of memory cells and a secondary storage device, e.g., a disk, is organized in sectors or blocks read and written as a unit. read/write coherence and before-or-after atomicity are two highly desirable properties of any storage model and in particular of cell storage.



Read/Write coherence: the result of a **Read** of memory cell M should be the same as the most recent **Write** to that cell



Before-or-after atomicity: the result of every **Read** or **Write** is the same as if that **Read** or **Write** occurred either completely before or completely after any other **Read** or **Write**.

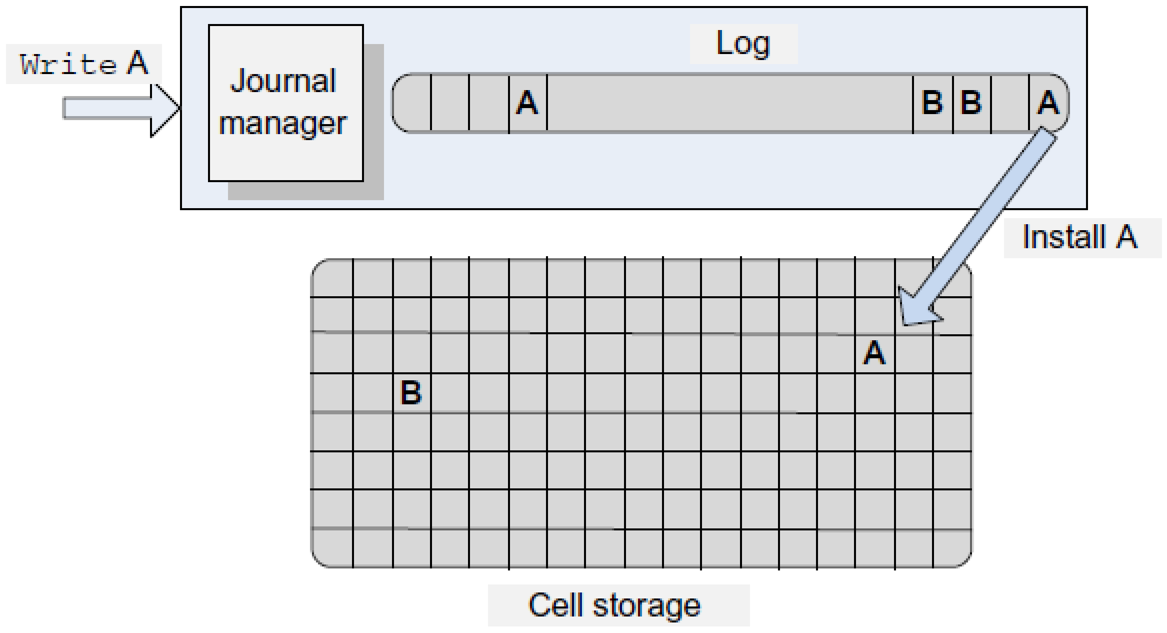
Journal storage is a fairly elaborate organization for storing composite objects such as records consisting of multiple fields. Journal storage consists of a manager and cell storage, where the entire history of a variable is maintained, rather than just the current value. The user does not have direct access to the cell storage ; instead the user can request the journal manager to

(i) start a new action;
(ii) read the value of a cell;

(iii) write the value of a cell;
(iv) commit an action; or
(v) abort an action

The journal manager translates user requests to commands sent to the cell storage:
(i) read a cell;
(ii) write a cell;
(iii) allocate a cell; or
(iv) deallocate a cell.

In the context of storage systems, a log contains a history of all variables in cell storage. The information about the updates of each data item forms a record appended at the end of the log. A log provides authoritative information about the outcome of an action involving cell storage; the cell storage can be reconstructed using the log, which can be easily accessed – we only need a pointer to the last record.



A log contains the entire history of all variables. The log is stored on nonvolatile media of journal storage. If the system fails after the new value of a variable is stored in the log but before the value is stored in cell memory, then the value can be recovered from the log.

If the system fails while writing the log, the cell memory is not updated. This guarantees that all actions are all-or-nothing. Two variables, A and B, in the log and cell storage are shown. A new value of A is written first to the log and then installed on cell memory at the unique address assigned to A.

A file system:

A file system consists of a collection of directories. Each directory provides information about a set of files. Today high-performance systems can choose among three classes of file system: network files systems (NFSs), storage area networks (SANs), and parallel file systems (PFSs).

Traditional – Unix File System.

Distributed file systems:

- **Network files systems (NFSs):**
  The NFS is very popular and has been used for some time, but it does not scale well and has reliability problems; an NFS server could be a single point of failure.

Storage Area Networks (SAN) - allow cloud servers to deal with non-disruptive changes in the storage configuration. The storage in a SAN can be pooled and then allocated based on the needs of the servers. A SAN-based implementation of a file system can be expensive, as each node must have a Fibre Channel adapter to connect to the network.

Parallel File Systems (PFS) - scalable, capable of distributing files across a large number of nodes, with a global naming space. Several I/O nodes serve data to all computational nodes; it includes also a metadata server which contains information about the data stored in the I/O nodes. The interconnection network of a PFS could be a SAN.

Most cloud applications do not interact directly with file systems but rather through an application layer that manages a database. A database is a collection of logically related records. The software that controls the access to the database is called a **database management system (DBMS)**.

A DBMS supports a query language, a dedicated programming language used to develop database applications.

Most cloud applications are data intensive and test the limitations of the existing infrastructure.
Requirements:

- Rapid application development and short-time to the market.
- Low latency.
- Scalability.
- High availability.
- Consistent view of the data.

The NoSQL model is useful when the structure of the data does not require a relational model and the amount of data is very large.

Distributed file systems

Network File System (NFS):

NFS was the first widely used distributed file system; the development of this application based on the client-server model was motivated by the need to share a file system among a number of clients interconnected by a local area network.

A majority of workstations were running under Unix; thus, many design decisions for the NFS were influenced by the design philosophy of the Unix File System (UFS).

It is not surprising that the NFS designers aimed to:

- Provide the same semantics as a local UFS to ensure compatibility with existing applications.
- Facilitate easy integration into existing UFS.
- Ensure that the system would be widely used and thus support clients running on different operating systems.
- Accept a modest performance degradation due to remote access over a network with a bandwidth of several Mbps.

Unix File System (UFS):

The layered design provides flexibility.

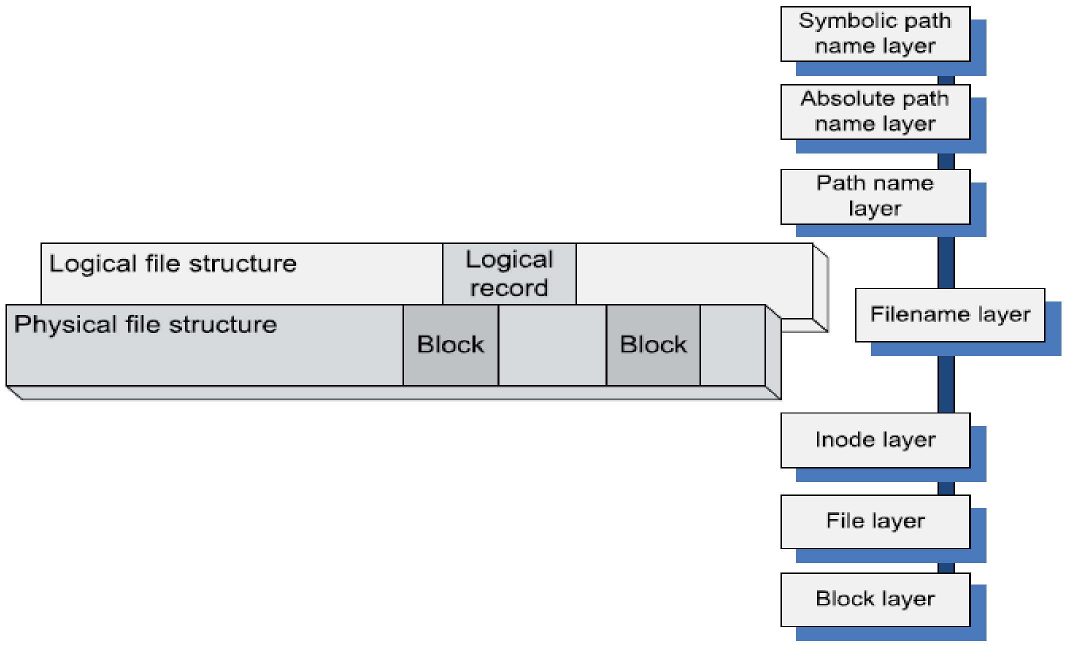
- The layered design allows UFS to separate the concerns for the physical file structure from the logical one.
- The vnode layer allowed UFS to treat uniformly local and remote file access.

The hierarchical design supports scalability reflected by the file naming convention. It allows grouping of files directories, supports multiple levels of directories, and collections of directories and files, the so-called file systems.

The metadata supports a systematic design philosophy of the file system and device-independence.

- Metadata includes: file owner, access rights, creation time, time of the last modification, file size, the structure of the file and the persistent storage device cells where data is stored.
- The inodes contain information about individual files and directories. The inodes are kept on persistent media together with the data.

The logical organization of a file reflects the data model – the view of the data from the perspective of the application. The physical organization reflects the storage model and describes the manner inwhich the file is stored on a given storage medium.



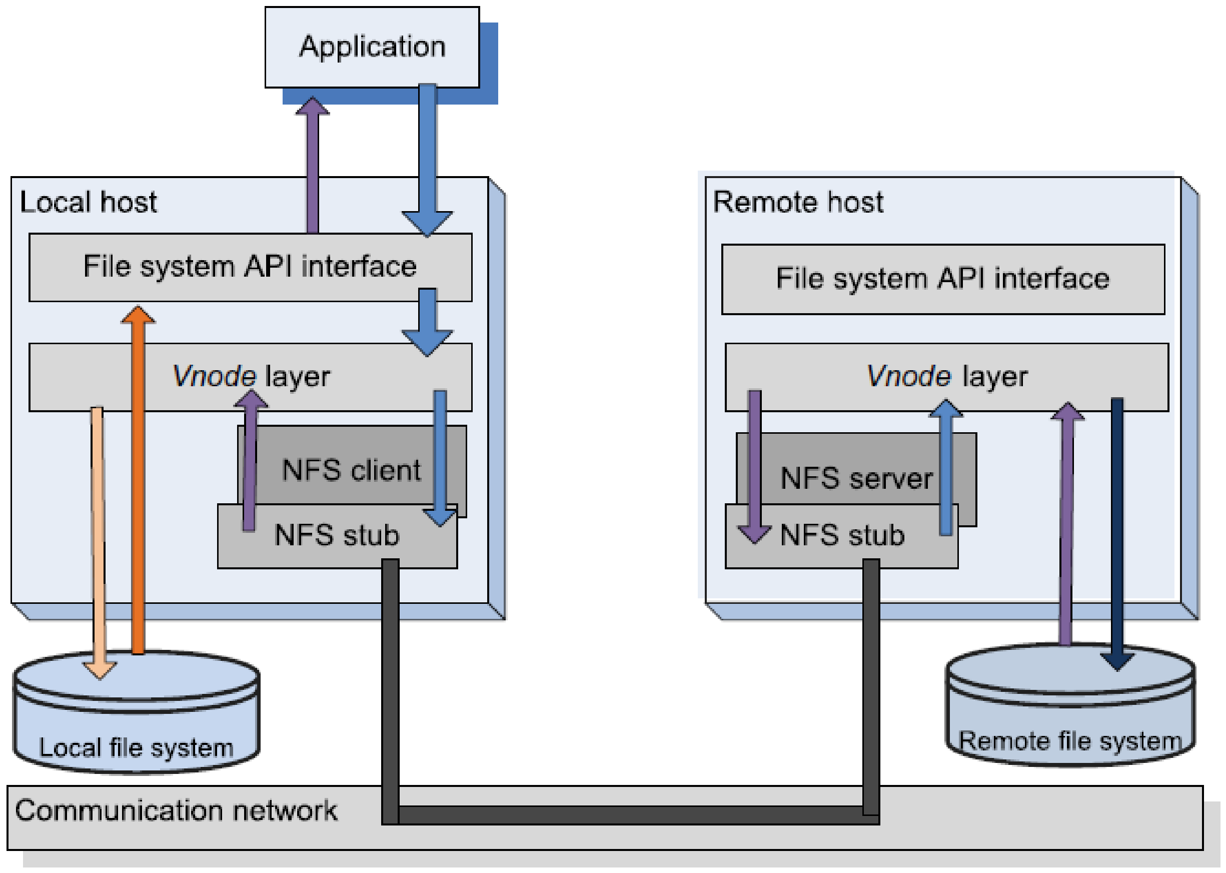
The **lower three layers** of the UFS hierarchy – the block, the file, and the inode layer – reflect the physical organization. The block layer allows the system to locate individual blocks on the physical device; the file layer reflects the organization of blocks into files; and the inode layer provides the metadata for the objects (files and directories).

The **upper three layers** – the path name, the absolute path name, and the symbolic path name layer – reflect the logical organization. The filename layer mediates between the machine-oriented and the user-oriented views of the file system.

Network File System (NFS)

NFS is based on the client-server paradigm. The client runs on the local host while the server is at the site of the remote file system; they interact by means of Remote Procedure Calls (RPC).

- The API interface of the local file system distinguishes file operations on a local file from the ones on a remote file and, in the latter case, invokes the RPC client.
- API for a Unix File System, with the calls made by the RPC client in response to API calls issued by a user program for a remote file system as well as some of the actions carried out by the NFS server in response to an RPC call. NFS uses a vnode layer to distinguish between operations on local and remote files
- A remote file is uniquely identified by a file handle (fh) rather than a file descriptor. The file handle is a 32-byte internal name, a combination of the file system identification, an inode number, and a generation number.

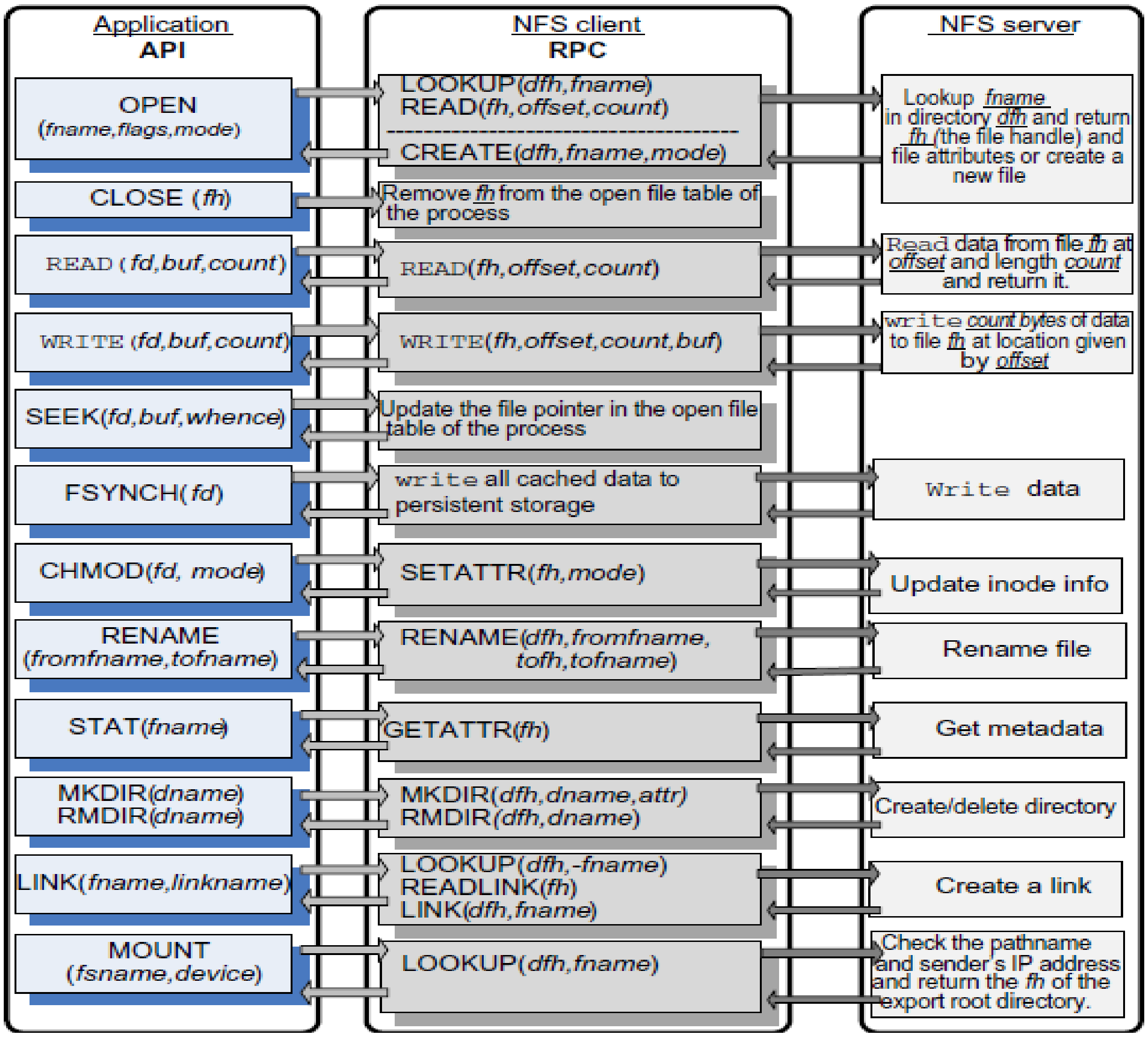


RPC calls, such as read, are idempotent, communication failures could sometimes lead to unexpected behavior. Indeed, if the network fails to deliver the response to a read RPC, then the call can be repeated without any side effects.

By contrast, when the network fails to deliver the response to the rmdir RPC, the second call returns an error code to the user if the call was successful the first time. If the server fails to execute the first call, the second call returns normally.

The API of the UNIX file system and the corresponding RPC issued by an NFS client to the NFS server.

- fd - file descriptor.
- fh - for file handle.
- fname - file name,
- dname - directory name.
- dfh - the directory were the file handle can be found.
- count - the number of bytes to be transferred.
- buf - the buffer to transfer the data to/from.
- device - the device where the file system is located.


Application
API
NFS client
RPC
NFS server
OPEN (fname,flags,mode)
LOOKUP(dfh,fname)
READ(fh,offset,count)
CREATE(dfh,fname,mode)
Lookup fname in directory dfh and return fh (the file handle) and file attributes or create a new file
CLOSE (fh)
Remove fh from the open file table of the process
READ (fd,buf,count)
READ(fh,offset,count)
Read data from file fh at offset and length count and return it.
WRITE (fd,buf,count)
WRITE(fh,offset,count,buf)
write count bytes of data to file fh at location given by offset
SEEK(fd,buf,whence)
Update the file pointer in the open file table of the process
FSYNCH(fd)
write all cached data to persistent storage
Write data
CHMOD(fd, mode)
SETATTR(fh,mode)
Update inode info
RENAME (fromfname,tofname)
RENAME(dfh,fromfname, tofh,tofname)
Rename file
STAT(fname)
GETATTR(fh)
Get metadata
MKDIR(dname)
RMDIR(dname)
MKDIR(dfh,dname,attr)
RMDIR(dfh,dname)
Create/delete directory
LINK(fname,linkname)
LOOKUP(dfh,-fname)
READLINK(fh)
LINK(dfh,fname)
Create a link
MOUNT (fsname,device)
LOOKUP(dfh,fname)
Check the pathname and sender's IP address and return the fh of the export root directory.

Andrew File System (AFS):

In distributed networks, an AFS relies on local cache to enhance performance and reduce workload. For example, a server responds to a workstation request and stores the data in the workstation's local cache. When the workstation requests the same data, the local cache fulfills the request.

Sprite Network File System (SFS):

SFS is a component of the Sprite network operating system. SFS supports non-write-through caching of files on the client as well as the server systems. Processes running on all workstations enjoy the same semantics for file access as they would if they were run on a single system.

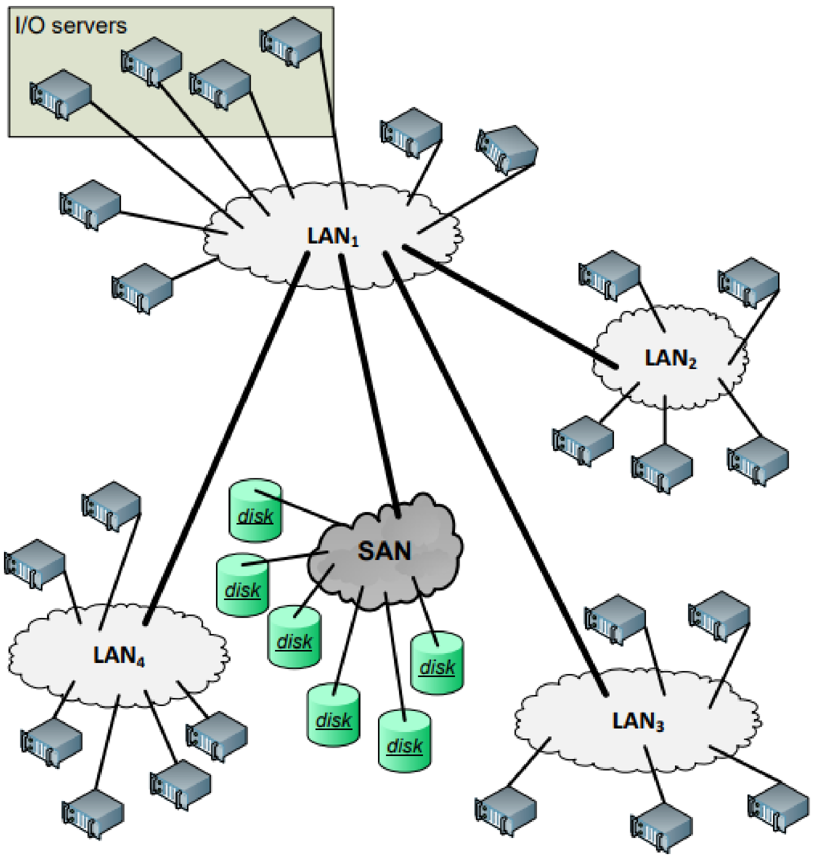
General Parallel File System (GPFS)

Parallel I/O implies execution of multiple input/output operations concurrently. Support for parallel I/O is essential to the performance of many applications. Parallel file systems allow multiple clients to read and write concurrently from the same file.

Concurrency control is a critical issue for parallel file systems. Several semantics for handling the shared access are possible. For example, when the clients share the file pointer successive reads issued by multiple clients advance the file pointer; another semantics is to allow each client to have its own file pointer.

GPFS:

- Developed at IBM in the early 2000s as a successor of the TigerShark multimedia file system.
- Designed for optimal performance of large clusters; it can support a file system of up to 4 PB consisting of up to 4,096 disks of 1 TB each.
- Maximum file size is (2^{63} -1) bytes.
- A file consists of blocks of equal size, ranging from 16 KB to 1 MB, stripped across several disks.



GPFS reliability:

To recover from system failures, GPFS records all metadata updates in a write-ahead log file.

- Write-ahead - updates are written to persistent storage only after the log records have been written.
- The log files are maintained by each I/O node for each file system it mounts; any I/O node can initiate recovery on behalf of a failed node.
- Data striping allows concurrent access and improves performance, but can have unpleasant side-effects. When a single disk fails, a large number of files are affected.
- The system uses RAID devices with the stripes equal to the block size and dual-attached RAID controllers.
- To further improve the fault tolerance of the system, GPFS data files as well as metadata are replicated on two different physical disks.

GPFS distributed locking:

In GPFS, consistency and synchronization are ensured by a distributed locking mechanism. A central lock manager grants lock tokens to local lock managers running in each I/O node. Lock tokens are also used by the cache management system.

Lock granularity has important implications on the performance. GPFS uses a variety of techniques for different types of data.

 Byte-range tokens - used for read and write operations to data files as follows: the first node attempting to write to a file acquires a token covering the entire file; this node is allowed to carry out all reads and writes to the file without any need for permission until a second node attempts to write to the same file; then, the range of the token given to the first node is restricted.

 Data-shipping - an alternative to byte-range locking, allows fine-grain data sharing. In this mode the file blocks are controlled by the I/O nodes in a round-robin manner. A node forwards a read or write operation to the node controlling the target block, the only one allowed to access the file.

GPFS uses disk maps to manage the disk space. The GPFS block size can be as large as 1 MB, and a typical block size is 256 KB. A block is divided into 32 sub blocks to reduce disk fragmentation for small files; thus, the block map has 32 bits to indicate whether a subblock is free or used.

An allocation manager running on one of the I/O nodes is responsible for actions involving multiple disk map regions. For example, it updates free space statistics and helps with deallocation by sending periodic hints of the regions used by individual nodes.

Google File System (GFS)

The Google File System (GFS) was developed in the late 1990s. It uses thousands of storage systems built from inexpensive commodity components to provide petabytes of storage to a large user community with diverse needs.

It is not surprising that a main concern of the GFS designers was to ensure the reliability of a system exposed to hardware failures, system software errors, application errors, and last but not least, human errors.

The system was designed after a careful analysis of the file characteristics and of the access models. Some of the most important aspects of this analysis reflected in the GFS design are:

- Scalability and reliability are critical features of the system; they must be considered from the beginning, rather than at some stage of the design.
- The vast majority of files range in size from a few GB to hundreds of TB.
- The most common operation is to append to an existing file; random write operation to a file are extremely infrequent.
- Sequential read operations are the norm.
- The users process the data in bulk and are less concerned with the response time.
- The consistency model should be relaxed to simplify the system implementation but without placing an additional burden on the application developers.

Several design decisions were made as a result of this analysis:

1. Segment a file in large chunks.

2. Implement an atomic file append operation allowing multiple applications operating concurrently to append to the same file.

3. Build the cluster around a high-bandwidth rather than low-latency interconnection network. Separate the flow of control from the data flow; schedule the high-bandwidth data flow by pipelining the data transfer over TCP connections to reduce the response time.

4. Eliminate caching at the client site. Caching increases the overhead for maintaining consistency among cached copies at multiple client sites and it is not likely to improve performance.

5. Ensure consistency by channeling critical file operations through a master, a component of the cluster that controls the entire system.

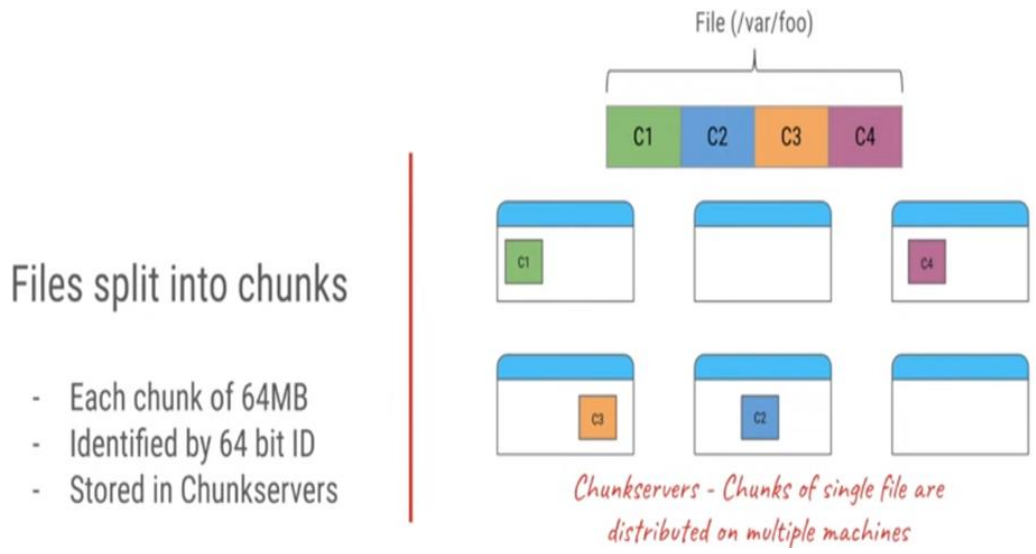
6. Minimize the involvement of the master in file access operations to avoid hot-spot contention and to ensure scalability.

7. Support efficient check pointing and fast recovery mechanisms.

8. Support an efficient garbage-collection mechanism.

GFS chunks:

- GFS files are collections of fixed-size segments called chunks.
- The chunk size is 64 MB; this choice is motivated by the desire to optimize the performance for large files and to reduce the amount of metadata maintained by the system.
- A large chunk size increases the likelihood that multiple operations will be directed to the same chunk thus, it reduces the number of requests to locate the chunk and, at the same time, it allows the application to maintain a persistent network connection with the server where the chunk is located.

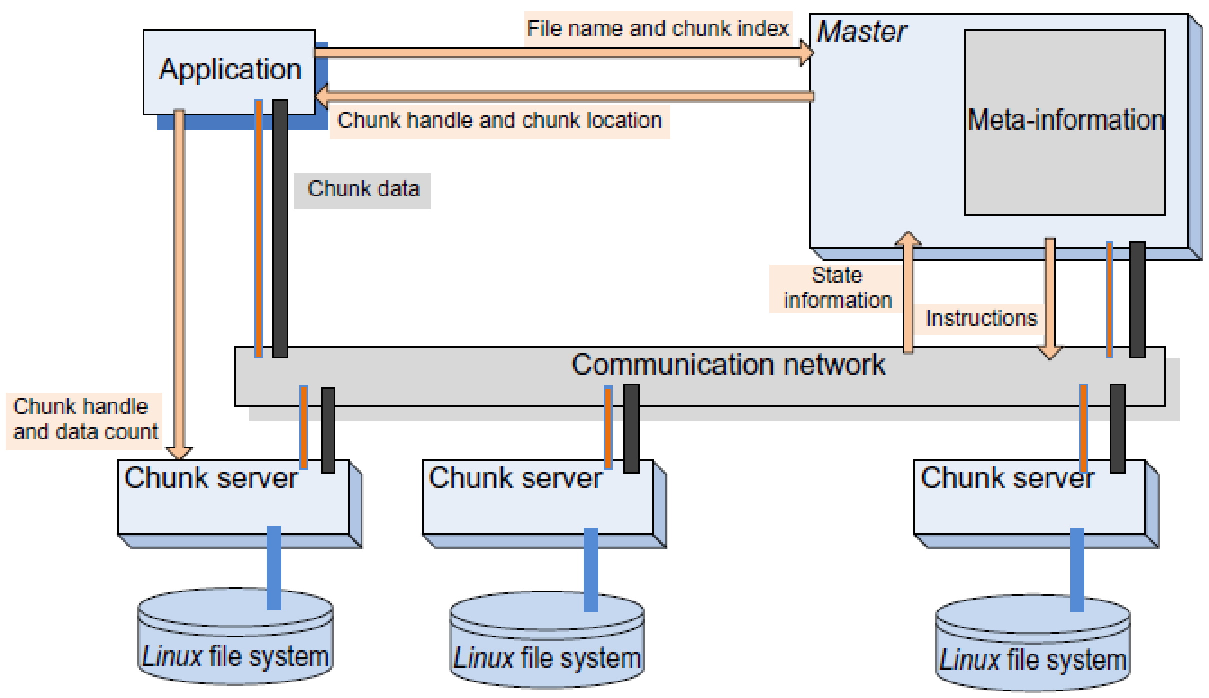

File (/var/foo)
C1
C2
C3
C4
Files split into chunks
- Each chunk of 64MB
- Identified by 64 bit ID
- Stored in Chunkservers
Chunkservers - Chunks of single file are distributed on multiple machines

The architecture of a GFS cluster:

A master controls a large number of chunk servers; it maintains metadata such as filenames, access control information, the location of all the replicas for every chunk of each file, and the state of individual chunk servers. Some of the metadata is stored in persistent storage (e.g., the operation log records the file namespace as well as the file-to-chunk mapping).

The locations of the chunks are stored only in the control structure of the master's memory and are updated at system startup or when a new chunk server joins the cluster. This strategy allows the master to have up-to-date information about the location of the chunks.

System reliability is a major concern, and the operation log maintains a historical record of metadata changes, enabling the master to recover in case of a failure. As a result, such changes are atomic and are not made visible to the clients until they have been recorded on multiple replicas on persistent storage.



When data for a write straddles the chunk boundary, two operations are carried out, one for each chunk. The steps for a write request illustrate a process that buffers data and decouples the control flow from the data flow for efficiency:

1. The client contacts the master, which assigns a lease to one of the chunk servers for a particular chunk if no lease for that chunk exists; then the master replies with the ID of the primary as well as secondary chunk servers holding replicas of the chunk. The client caches this information.

2. The client sends the data to all chunk servers holding replicas of the chunk; each one of the chunk servers stores the data in an internal LRU buffer and then sends an acknowledgment to the client.

3. The client sends a write request to the primary once it has received the acknowledgments from all chunk servers holding replicas of the chunk. The primary identifies mutations by consecutive sequence numbers.

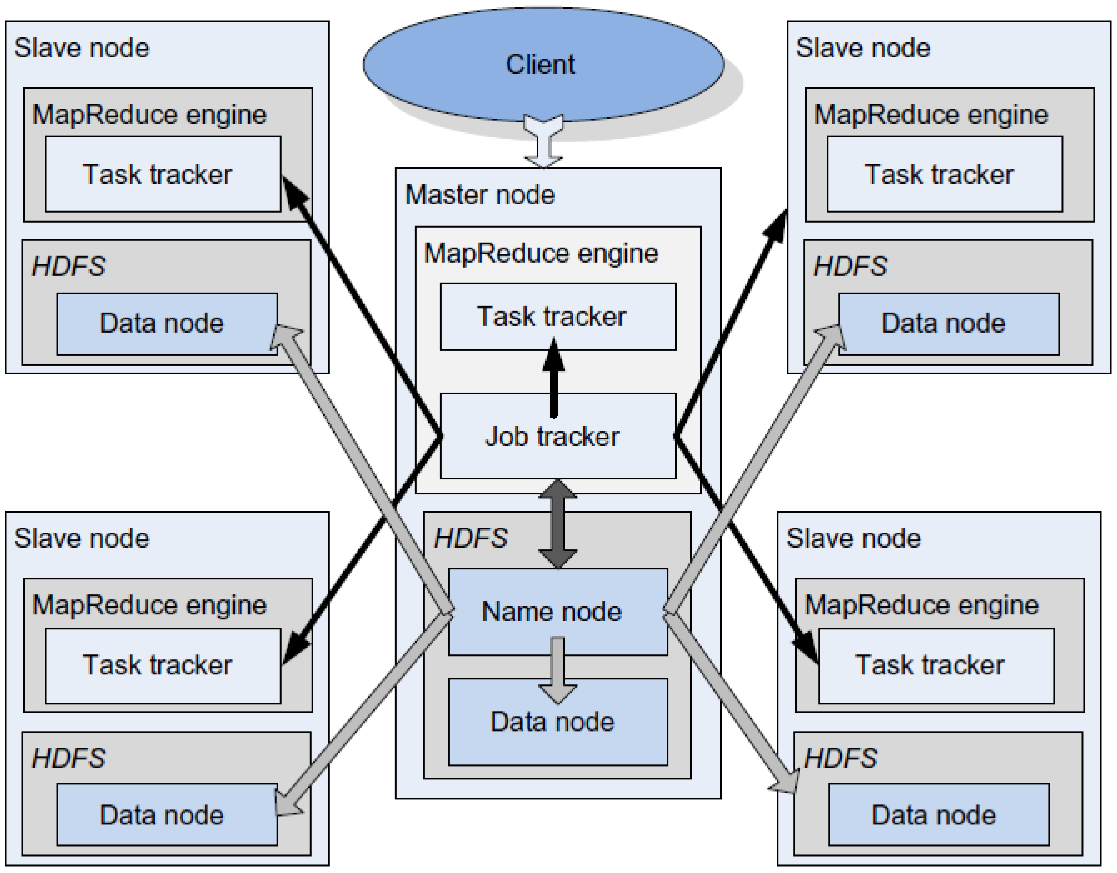
4. The primary sends the write requests to all secondaries.

5. Each secondary applies the mutations in the order of the sequence numbers and then sends an acknowledgment to the primary.

6. Finally, after receiving the acknowledgments from all secondaries, the primary informs the client.

Apache Hadoop

- Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models.
- Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.
- Hadoop runs applications using the MapReduce algorithm, where the data is processed in parallel on different CPU nodes.



❖ Hadoop has a Master Slave Architecture for both Storage & Processing

The Hadoop Distributed File System (HDFS) is a distributed file system for Hadoop. This architecture consist of a single NameNode performs the role of master, and multiple DataNodes performs the role of a slave.

The name node running on the master manages the data distribution and data replication and communicates with data nodes running on all cluster nodes; it shares with the job tracker information about data placement to minimize communication between the nodes on which data is located and the ones where it is needed.

Namenode

- NameNode is the centerpiece of HDFS.
- NameNode is also known as the Master
- NameNode only stores the metadata of HDFS – the directory tree of all files in the file system, and tracks the files across the cluster.
- NameNode does not store the actual data or the dataset. The data itself is actually stored in the DataNodes.
- NameNode knows the list of the blocks and its location for any given file in HDFS..

DataNode

- DataNode is responsible for storing the actual data in HDFS.
- DataNode is also known as the Slave
- When a DataNode is down, it does not affect the availability of data or the cluster. NameNode will arrange for replication for the blocks managed by the DataNode that is not available.
- DataNode is usually configured with a lot of hard disk space. Because the actual data is stored in the DataNode.

Job Tracker

- The role of Job Tracker is to accept the MapReduce jobs from client and process the data by using NameNode.
- In response, NameNode provides metadata to Job Tracker.

Task Tracker

- It works as a slave node for Job Tracker.
- It receives task and code from Job Tracker and applies that code on the file. This process can also be called as a Mapper.

Advantages of Hadoop

- Fast, Scalable, Cost Effective, Resilient to failure

<u>Big Table</u>

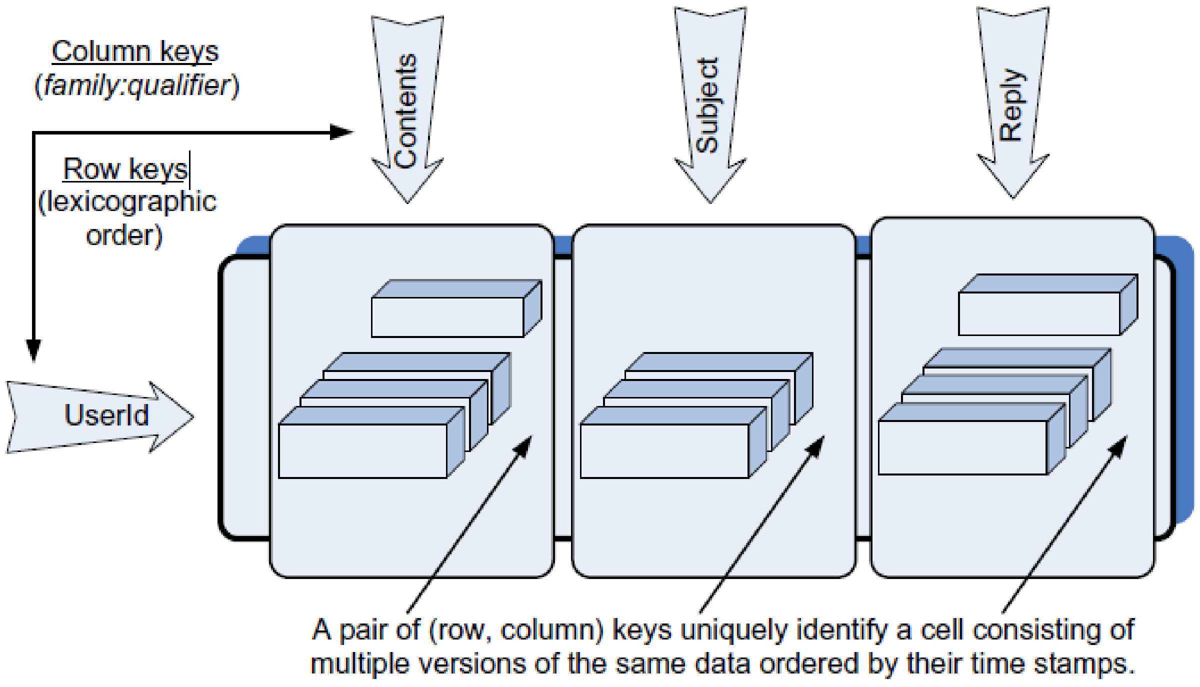
➢ BigTable is a distributed storage system developed by Google to store massive amounts of data and to scale up to thousands of storage servers.

- Cloud Bigtable is a sparsely populated table that can scale to billions of rows and thousands of columns, enabling you to store terabytes or even petabytes of data.
- To guarantee atomic read and write operations, it uses the Chubby distributed lock service; the directories and the files in the namespace of Chubby are used as locks.

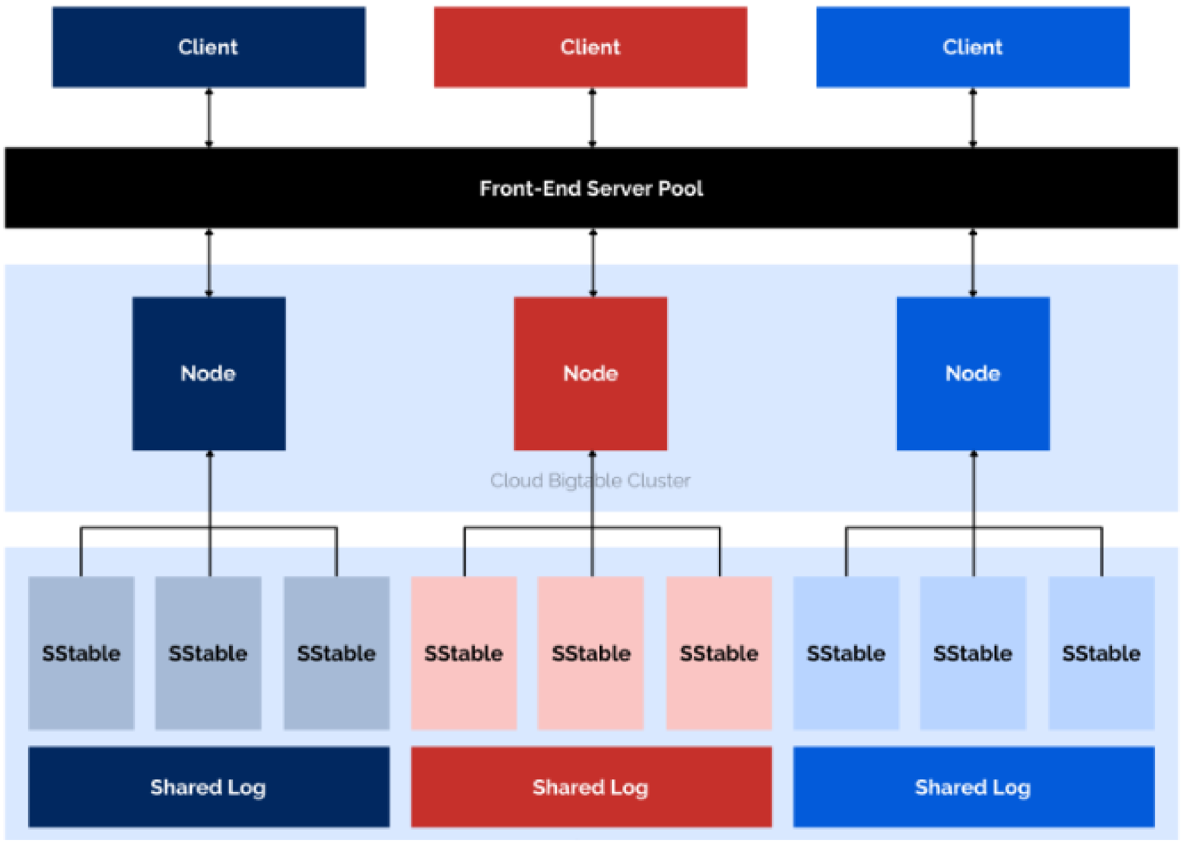
Space management is ensured by a three-level hierarchy,

- root tablet
- metadata tablet
- user tablets

- The organization of a BigTable a sparse, distributed, multidimensional map for an **email application**. The system consists of three major components: a library linked to application clients to access the system, a master server, and a large number of tablet servers.
- The master server controls the entire system, assigns tablets to tablet servers and balances the load among them, manages garbage collection, and handles table and column family creation and deletion.


Column keys
(family:qualifier)
Row keys
(lexicographic order)
Contents
Subject
Reply
UserId
A pair of (row, column) keys uniquely identify a cell consisting of multiple versions of the same data ordered by their time stamps.

- ➢ The organization of an email application as a sparse, distributed, multidimensional map. The slice of the BigTable shown consists of a row with the key “UserId” and three family columns.
- ➢ The “Contents” key identifies the cell holding the contents of emails received, the cell with key “Subject” identifies the subject of emails, and the cell with the key “Reply” identifies the cell holding the replies.
- ➢ The versions of records in each cell are ordered according to their time stamps. The row keys of this BigTable are ordered lexicographically.
- ➢ A column key is obtained by concatenating the family and the qualifier fields. Each value is an uninterpreted array of bytes.



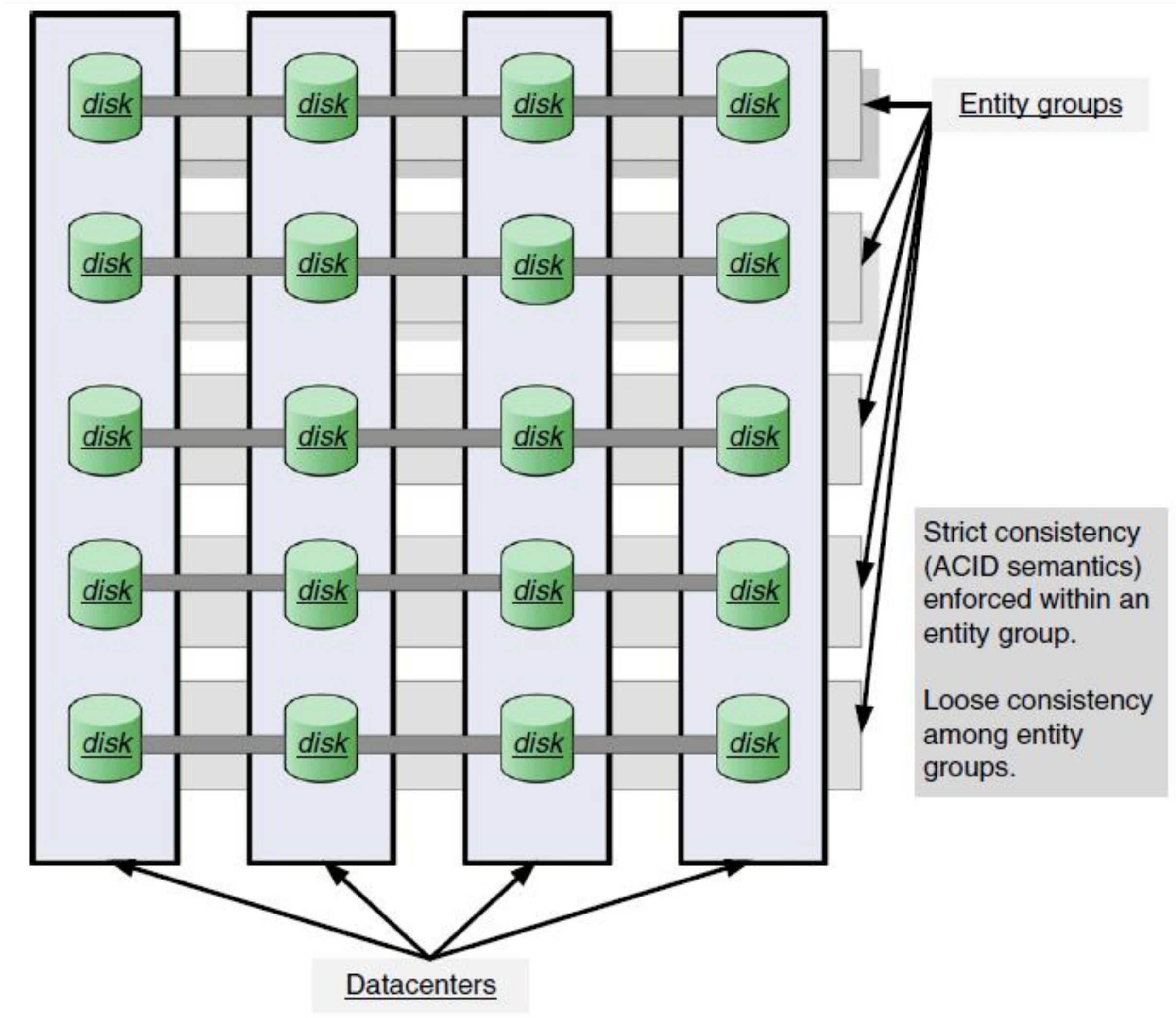
BigTable is used by a variety of applications, including Google Earth, Google Analytics, Google Finance, andWeb crawlers. For example, Google Earth uses two tables, one for preprocessing and one for serving client data.

<u>**Megastore**</u>

Megastore is scalable storage for online services. The system, distributed over several data centers. Providing scalable, highly available storage for interactive services.

Google's Megastore is the structured data store supporting the Google Application Engine. Megastore handles more than 3 billion write and 20 billion read transactions daily and stores a petabyte of primary data across many global datacenters.

The system makes extensive use of BigTable. Entities from different Megastore tables can be mapped to the same BigTable row without collisions. This is possible because the BigTable column name is a concatenation of the Megastore table name and the name of a property. A BigTable row for the root entity stores the transaction and all metadata for the entity group.


disk
Entity groups
Strict consistency (ACID semantics) enforced within an entity group.
Loose consistency among entity groups.
Datacenters

Each partition is replicated in data centers in different geographic areas. The system supports full ACID semantics within each partition and provides limited consistency guarantees across partitions.

Read and write operations can proceed concurrently, and a read always returns the last fully updated version.

A write transaction involves the following steps:

(1) Get the timestamp and the log position of the last committed transaction. (2) Gather the write operations in a log entry. (3) Use the consensus algorithm to append the log entry and then commit. (4) Update the BigTable entries. (5) Clean up.

AWS Simple Storage Service (S3)

AWS Simple Storage Service (S3) is the object storage service provided by AWS. It is probably the most commonly used, go-to storage service for AWS users given the features like extremely high availability, security, and simple connection to other AWS Services.

- S3 is a safe place to store the files.
- It is Object-based storage, i.e., you can store the images, word files, pdf files, etc.
- The files which are stored in S3 can be from 0 Bytes to 5 TB.
- It has unlimited storage means that you can store the data as much you want.
- Files are stored in Bucket. A bucket is like a folder available in S3 that stores the files.
- S3 is a universal namespace, i.e., the names must be unique globally. Bucket contains a DNS address. Therefore, the bucket must contain a unique name to generate a unique DNS address.

Amazon S3 – Components:

1. Buckets
2. Objects
3. Keys
4. Regions
5. Data Consistency Model

1. **Buckets**
    - A bucket is a container used for storing the objects.
    - Every object is incorporated in a bucket.
    - For example, if the object named photos/tree.jpg is stored in the treeimage bucket, then it can be addressed by using the URL http://treeimage.s3.amazonaws.com/photos/tree.jpg.

- A bucket has no limit to the amount of objects that it can store. No bucket can exist inside of other buckets.
- S3 performance remains the same regardless of how many buckets have been created.
- The AWS user that creates a bucket owns it, and no other AWS user cannot own it. Therefore, we can say that the ownership of a bucket is not transferrable.

2. **Objects**
    - Objects are the entities which are stored in an S3 bucket.
    - An object consists of object data and metadata where metadata is a set of name-value pair that describes the data.
    - An object consists of some default metadata such as date last modified, and standard HTTP metadata, such as Content type. Custom metadata can also be specified at the time of storing an object.
    - It is uniquely identified within a bucket by key and version ID.

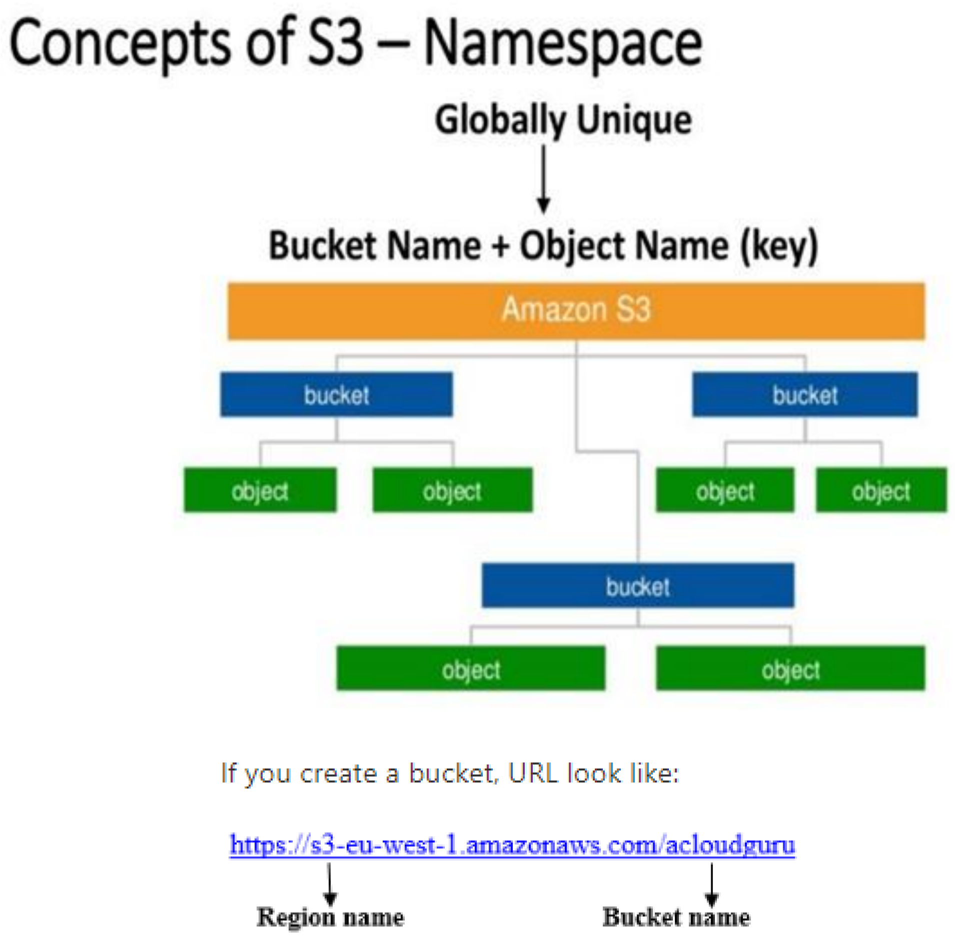
3. **Key**
    - A key is a unique identifier for an object.
    - Every object in a bucket is associated with one key.
    - An object can be uniquely identified by using a combination of bucket name, the key, and optionally version ID.
    - For example, in the URL http://jtp.s3.amazonaws.com/2019-01-31/Amazons3.wsdl where "jtp" is the bucket name, and key is "2019-01-31/Amazons3.wsdl"

4. **Regions**
    - You can choose a geographical region in which you want to store the buckets that you have created.
    - A region is chosen in such a way that it optimizes the latency, minimize costs or address regulatory requirements.
    - Objects will not leave the region unless you explicitly transfer the objects to another region.

5. **Data Consistency Model**
   Amazon S3 replicates the data to multiple servers to achieve high availability.

   Two types of model:

- **Read-after-write consistency for PUTS of new objects.**
  - For a PUT request, S3 stores the data across multiple servers to achieve high availability.
  - A process stores an object to S3 and will be immediately available to read the object.
  - A process stores a new object to S3, it will immediately list the keys within the bucket.
  - It does not take time for propagation, the changes are reflected immediately.



If you create a bucket, URL look like:

https://s3-eu-west-1.amazonaws.com/acloudguru

If you upload a file to S3 bucket, then you will receive an HTTP 200 code means that the uploading of a file is successful.

Cloud security

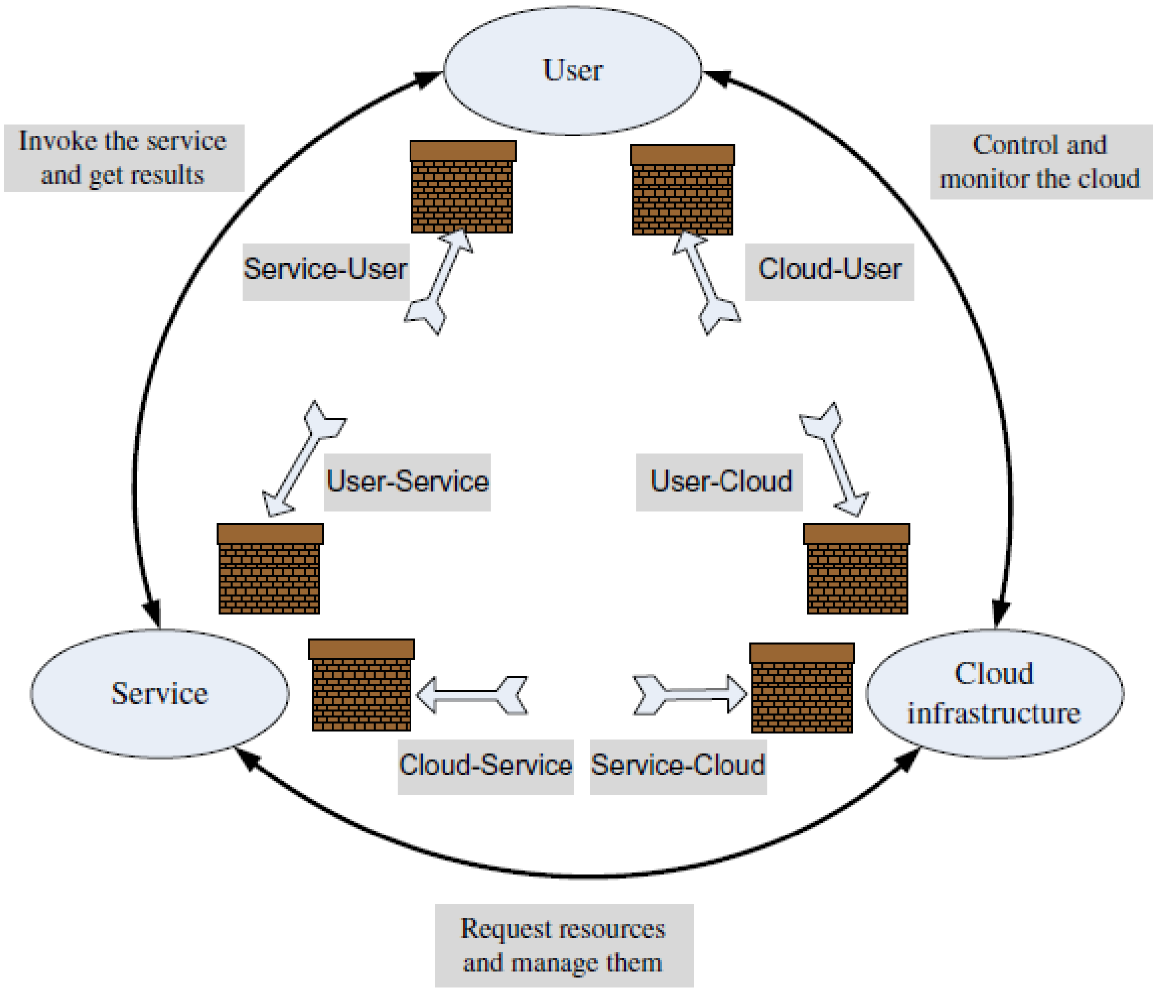
A computer cloud is a target-rich environment for malicious individuals and criminal organizations.

- Major concern for existing users and for potential new users of cloud computing services. Outsourcing computing to a cloud generates new security and privacy concerns.
- Standards, regulations, and laws governing the activities of organizations supporting cloud computing have yet to be adopted.
- Many issues related to privacy, security, and trust in cloud computing are far from being settled.
- There is the need for international regulations adopted by the countries where data centers of cloud computing providers are located.
- Service Level Agreements (SLAs) do not provide adequate legal protection for cloud computer users, often left to deal with events beyond their control.

Cloud security risks:

- **Traditional threats** are those experienced for some time by any system connected to the Internet, but with some cloud-specific twists. The impact of traditional threats is amplified due to the vast amount of cloud resources and the large user population that can be affected. The fuzzy bounds of responsibility between the providers of cloud services and users and the difficulties in accurately identifying the cause of a problem add to cloud users' concerns.
- **New threats** - cloud servers host multiple VMs; multiple applications may run under each VM. Multi-tenancy and VMM vulnerabilities open new attack channels for malicious users. Identifying the path followed by an attacker more difficult in a cloud environment.
- **Authentication and authorization** - the procedures in place for one individual does not extend to an enterprise.
- **Third-party control** - generates a spectrum of concerns caused by the lack of transparency and limited user control.
- **Availability of cloud services** - system failures, power outages, and other catastrophic events could shutdown services for extended periods of time.
- **Data loss or leakages** are two risks with devastating consequences for an individual or an organization sing cloud services. Maintaining copies of the data outside the cloud is often unfeasible due to the sheer volume of data.

- **Account or service hijacking** is a significant threat, and cloud users must be aware of and guard against all methods of stealing credentials. Finally, unknown risk profile refers to exposure to the ignorance or underestimation of the risks of cloud computing.



Attacks in a cloud computing environment:

Three actors involved; six types of attacks possible.

The user can be attacked by:

- Service - SSL certificate spoofing, attacks on browser caches, or phishing attacks.
- The cloud infrastructure - attacks that either originates at the cloud or spoofs to originate from the cloud infrastructure.

The service can be attacked by:

- A user - buffer overflow, SQL injection, and privilege escalation are the common types of attacks.
- The cloud infrastructure - the most serious line of attack. Limiting access to resources, privilege-related attacks, data distortion, injecting additional operations.

The cloud infrastructure can be attacked by:

- A user - targets the cloud control system.
- A service - requesting an excessive amount of resources and causing the exhaustion of the resources.

A Top concern for cloud users

Major user concerns are unauthorized access to confidential information and data theft. Data is more vulnerable in storage than while it is being processed. Data is kept in storage for extended periods of time, whereas it is exposed to threats during processing for relatively short periods of time.

1. Data Breaches:

Data Breaches result from an attack or employee negligence and error. This is a primary cause for concern in cloud platforms. Vulnerabilities in the application or ineffective security practices can also cause data breaches. Employees may log into cloud systems from their phones or personal laptops thus exposing the system to targeted attacks.

2. Account Hijacking

With the increase in adoption of cloud services, organizations have reported an increased occurrence of account hijacking. Such attacks involve using employee’s login information to access sensitive information. Attackers can also modify, insert false information and manipulate the data present in the cloud.

3. Insecure APIs and Interfaces

Customers can tailor their cloud computing experience according to their needs by using Application Programming Interface or APIs.

These are used to extract, manage and interact with information on the cloud. However, the unique characteristics of API leave the door wide open for threats. Hence the security of APIs affects the security and availability of cloud services and platforms.

4. Insider Threat

An Insider threat is the misuse of information through hostile intent, malware, and even accidents. Insider threats originate from employees or system administrators, who can access confidential information They can also access even more critical systems and eventually data.

5. Malware Injections and APT (Advanced Persistent Threats)

Malware injections are scripts or code that is inserted into the cloud services and begin to mimic valid instances. When embedded into the cloud, they begin to change the normal execution of the code.

Once the malware injection and cloud systems begin to operate in sync attackers can affect the integrity and security of the data. SQL injection attack and cross-site scripting attack are seen very often.

6. Denial of Service Attacks

Unlike other kind of cyberattacks, which are typically launched to establish a long-term foothold and hijack sensitive information, denial of service assaults do not attempt to breach your security perimeter. Rather, they attempt to make your website and servers unavailable to legitimate users.

7. Shared Vulnerabilities

Cloud security is a shared responsibility between the provider and the client.

<u>Privacy and privacy impact assessment</u>

The term privacy refers to the right of an individual, a group of individuals, or an organization to keep information of a personal or proprietary nature from being disclosed to others.

- Privacy is protected by law; sometimes laws limit privacy.
- The main aspects of privacy are: the lack of user control, potential unauthorized secondary use, data proliferation, and dynamic provisioning.
- Digital age has confronted legislators with significant challenges elated to privacy as new threats have emerged. For example, personal information voluntarily shared, but stolen from sites granted access to it or misused can lead to identity theft.
- Privacy concerns are different for the three cloud delivery models and also depend on the actual context.

Federal Trading Commission Rules:
Web sites that collect personal identifying information from or about consumers online required complying with four fair information practices:

Notice - provide consumers clear and conspicuous notice of their information practices, including what information they collect, how they collect it, how they use it, how they provide Choice, Access, and Security to consumers, whether they disclose the information collected to other entities, and whether other entities are collecting information through the site.

- Choice - offer consumers choices as to how their personal identifying information is used. Such choice would encompass both internal secondary uses (such as marketing back to consumers) and external secondary uses (such as disclosing data to other entities).
- Access - offer consumers reasonable access to the information a web site has collected about them, including a reasonable opportunity to review information and to correct inaccuracies or delete information.
- Security - take reasonable steps to protect the security of the information they collect from consumers.

Privacy Impact Assessment (PIA):

There is a need for tools capable of identifing privacy issues in information systems, the so-called Privacy Impact Assesment (PIA). There are no international standards for such a process, though different countries and organization require PIA reports.

- The centerpiece of A proposed PIA tool is based on a SaaS service.
- The users of the SaaS service providing access to the PIA tool must fill in a questionnaire.
- The system used a knowledge base (KB) created and maintained by domain experts.
- The system uses templates to generate additional questions necessaryand to fill in the PIA report.
- An expert system infers which rules are satisfied by the facts in the database and provided by the users and executes the rule with the highest priority.

Trust:

Trust in the context of cloud computing is intimately related to the general problem of trust in online activities. Two conditions must exist for trust to develop.

The first condition is risk, the perceived probability of loss; indeed, trust would not be necessary if there were no risk involved, if there is a certainty that an action can succeed.

The second condition is interdependence, the idea that the interests of one entity cannot be achieved without reliance on other entities.

A trust relationship goes though three phases:

1. Building phase, when trust is formed.

2. Stability phase, when trust exists.

3. Dissolution phase, when trust declines.

Internet trust:

- Obscures or lacks entirely the dimensions of character and personality, nature of relationship, and institutional character of thetraditional trust.
- Offers individuals the ability to obscure or conceal their identity. The anonymity reduces the cues normally used in judgments of trust.
- Identity is critical for developing trust relations, it allows us to base our trust on the past history of interactions with an entity. Anonymity causes mistrust because identity is associated with accountabilityand in absence of identity accountability cannot be enforced.
- The opacity extends identity to personal characteristics. It is impossible to infer if the entity or individual we transact with is who it pretends to be, as the transactions occur between entities separated in time and distance.
- There are no guarantees that the entities we transact with fully understand the role they have assumed.

Policies and reputation are two ways of determining trust:

- Policies reveal the conditions to obtain trust, and the actions when some of the conditions are met. Policies require the verification of credentials; credentials are issued by a trusted authority and describe the qualities of the entity using the credential.
- Reputation is a quality attributed to an entity based on a relatively long history of interactions or possibly observations of the entity.
- Recommendations are based on trust decisions made by others and filtered through the perspective of the entity assessing the trust

In a computer science context : trust of a party A to a party B for a service X is the measurable belief of A in that B behaves dependably for a specified period within a specified context (in relation to service X).

Operating system security

An operating system (OS) allows multiple applications to share the hardware resources of a physical system, subject to a set of policies. A critical function of an OS is to protect applications against a wide range of malicious attacks such as unauthorized access to privileged information, tempering with executable code, and spoofing.

The mandatory security of an OS is considered to be “any security policy where the definition of the policy logic and the assignment of security attributes is tightly controlled by a system security policy administrator”

The elements of the mandatory OS security:

1. Access control,
2. Authentication usage,
3. Cryptographic usage policies.

The first policy specifies how the OS controls the access to different system objects. **The second** defines the authentication mechanisms the OS uses to authenticate a principal, **and the last** specifies the cryptographic mechanisms used to protect the data. A necessary but not sufficient condition for security is that the subsystems tasked with performing security-related functions are temper-proof and cannot be bypassed.

Commercial OS do not support a multi-layered security; only distinguish between a completely privileged security domain and a completely unprivileged one.

 Trusted paths mechanisms - support user interactions with trusted software. Critical for system security; if such mechanisms do not exist, then malicious software can impersonate trusted software. Some systems provide trust paths for a few functions, such as login authentication and password changing, and allow servers to authenticate their clients.

Closed-box versus open-box platforms:

Closed-box platforms , e.g., cellular phones, game consoles and ATM could have embedded cryptographic keys to reveal their true identity to remote systems and authenticate the software running on them.

 Such facilities are not available to open-box platforms, the traditional hardware for commodity operating systems.

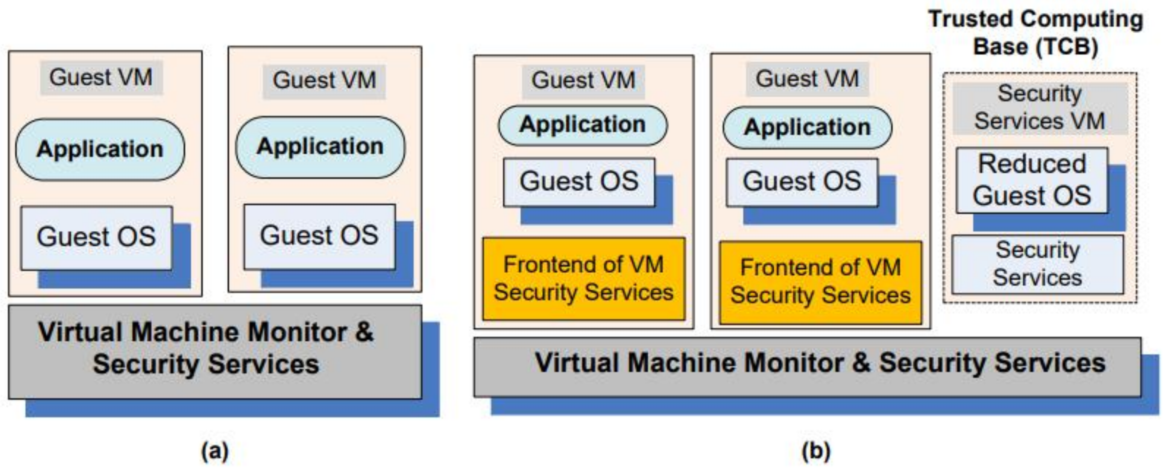
 Commodity operating system offer low assurance. An OS is a complex software system consisting of millions of lines of code and it is vulnerable to a wide range of malicious attacks.

Virtual machine security

Hybrid and hosted VMs, expose the entire system to the vulnerability of the host OS. Virtual security services are typically provided by the VMM.

In a traditional VM the Virtual Machine Monitor (VMM) controls the access to the hardware and provides a stricter isolation of VMs from one another than the isolation of processes in a traditional OS.

- A VMM controls the execution of privileged operations and can enforce memory isolation as well as disk and network access.
- The VMMs are considerably less complex and better structured than traditional operating systems thus, in a better position to respond to security attacks.
- A major challenge - a VMM sees only raw data regarding the state of a guest operating system while security services typically operate at a higher logical level, e.g., at the level of a file rather than a disk block.


Guest VM
Application
Guest OS
Guest VM
Application
Guest OS
Virtual Machine Monitor & Security Services
(a)
Trusted Computing Base (TCB)
Guest VM
Application
Guest OS
Frontend of VM Security Services
Guest VM
Application
Guest OS
Frontend of VM Security Services
Security Services VM
Reduced Guest OS
Security Services
Virtual Machine Monitor & Security Services
(b)

(a) Virtual security services provided by the VMM; (b) A dedicated security VM.

VMM-based threats:

1. Starvation of resources and denial of service for some VMs. Probable causes:

- badly configured resource limits for some VMs.
- a rogue VM with the capability to bypass resource limits set in VMM.

2. VM side-channel attacks: malicious attack on one or more VMs by a rogue VM under the same VMM. Probable causes:
   - (a) lack of proper isolation of inter-VM traffic due to misconfiguration of the virtual network residing in the VMM.
   - (b) limitation of packet inspection devices to handle high speed traffic, e.g., video traffic.
   - (c) presence of VM instances built from insecure VM images, e.g., a VM image having a guest OS without the latest patches.
3. Buffer overflow attacks.

VM-based threats:

1. Deployment of rogue or insecure VM. Unauthorized users may create insecure instances from images or may perform unauthorized administrative actions on existing VMs.

   Probable cause: improper configuration of access controls on VM administrative tasks such as instance creation, launching, suspension, reactivation, and so on.
2. Presence of insecure and tampered VM images in the VM image repository.

   Probable causes: (a) lack of access control to the VM image repository; (b) lack of mechanisms to verify the integrity of the images, e.g., digitally signed image.

Security risks posed by shared images

Even when we assume that a cloud service provider is trustworthy, many users either ignore or underestimate the danger posed by other sources of concern.

Image sharing is critical for the IaaS cloud delivery model. For example, a user of AWS has the option to choose between

- Amazon Machine Images (AMIs) accessible through the Quick Start.
- Community AMI menus of the EC2 service.

Many of the images analyzed by a recent report allowed a user to undelete files, recover credentials, private keys, or other types of sensitive information with little effort and using standard tools.

A software vulnerability audit revealed that 98% of the Windows AMIs and 58% of Linux AMIs audited had critical vulnerabilities.

Security risks:

- Backdoors and leftover credentials.
- Unsolicited connections.
- Malware.

Security risks posed by a management OS:

- A virtual machine monitor, or hypervisor, is considerably smaller than an operating system, e.g., the Xen VMM has ~ 60,000 lines of code.
- The Trusted Computer Base (TCB) of a cloud computing environment includes not only the hypervisor but also the management OS.
- The management OS supports administrative tools, live migration, device drivers, and device emulators.
- In Xen the management operating system runs in Dom0; it manages the building of all user domains, a process consisting of several steps:
  - Allocate memory in the Dom0 address space and load the kernel of the guest operating system from the secondary storage.
  - Allocate memory for the new VM and use foreign mapping to load the kernel to the new VM.
  - Set up the initial page tables for the new VM.
  - Release the foreign mapping on the new VM memory, set up the virtual CPU registers and launch the new VM.

Possible actions of a malicious Dom0:

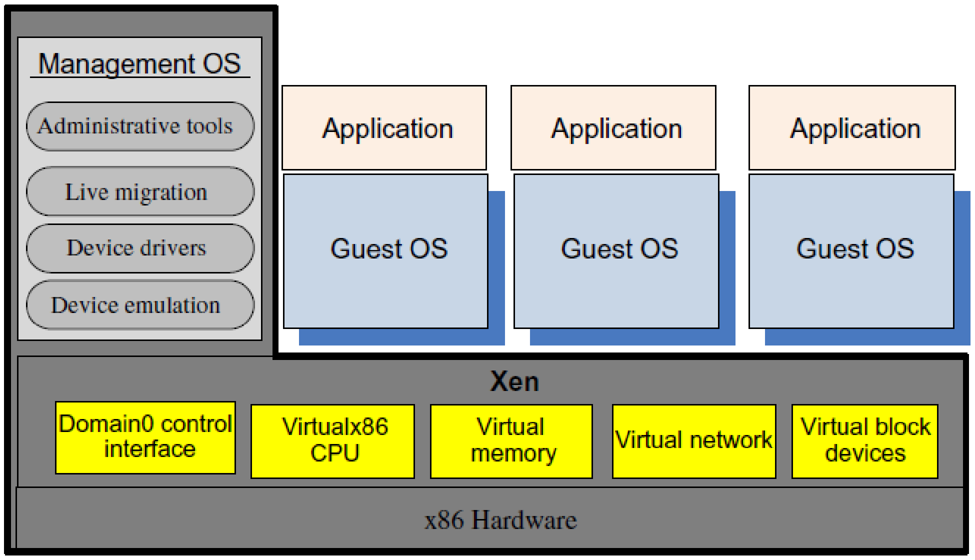
At the time it creates a DomU:

- Refuse to carry out the steps necessary to start the new VM.
- Modify the kernel of the guest OS to allow a third party to monitor and control the execution of applications running under the new VM.
- Undermine the integrity of the new VM by setting the wrong page tables and/or setup wrong virtual CPU registers.

- Refuse to release the foreign mapping and access the memory while the new VM is running.

At run time:

- Dom0 exposes a set of abstract devices to the guest operating systems using split drivers with the frontend of in a DomU and the backend in Dom0.
- We have to ensure that run time communication through Dom0 is encrypted.
- Transport Layer Security (TLS) does not guarantee that Dom0 cannot extract cryptographic keys from the memory of the OS and applications running in DomU



The trusted computing base of a *Xen*-based environment includes the hardware, *Xen*, and the management operating system running in *Dom0*. The management OS supports administrative tools, live migration, device drivers, and device emulators. A guest operating system and applications running under it reside in a *DomU*.