CLOUD COMPUTING
Cloud Data Storage

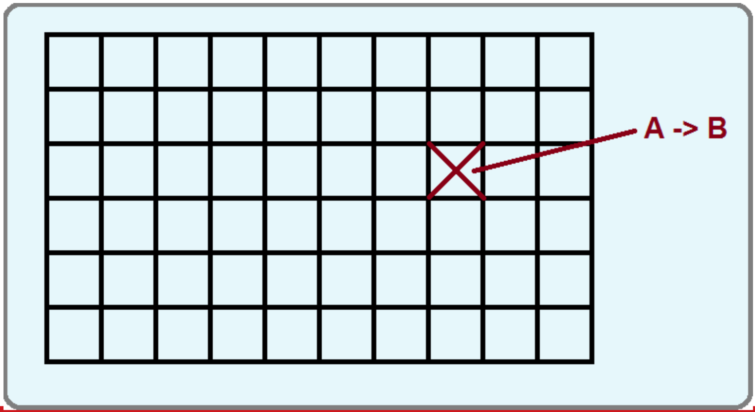
Zeinab Zali
Isfahan University Of Technology

Storage models

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

Cell Storage

- Cell storage assumes that the storage consists of cells of the same size and that each object fits exactly in one cell
  - The physical organization of several storage media
  - All changes to these cells will be atomic. If the system crashes as we are changing the contents of a cell from A to B, then the cell will either contain A or B, but never garbage that is a mixture of A and B

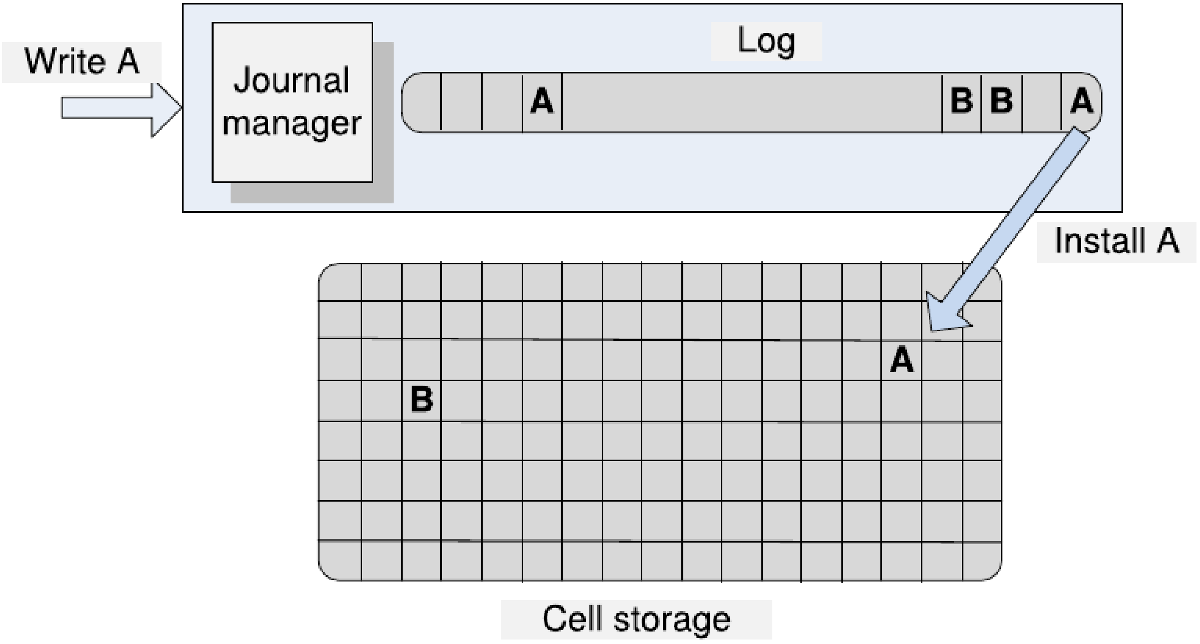


Journal Storage

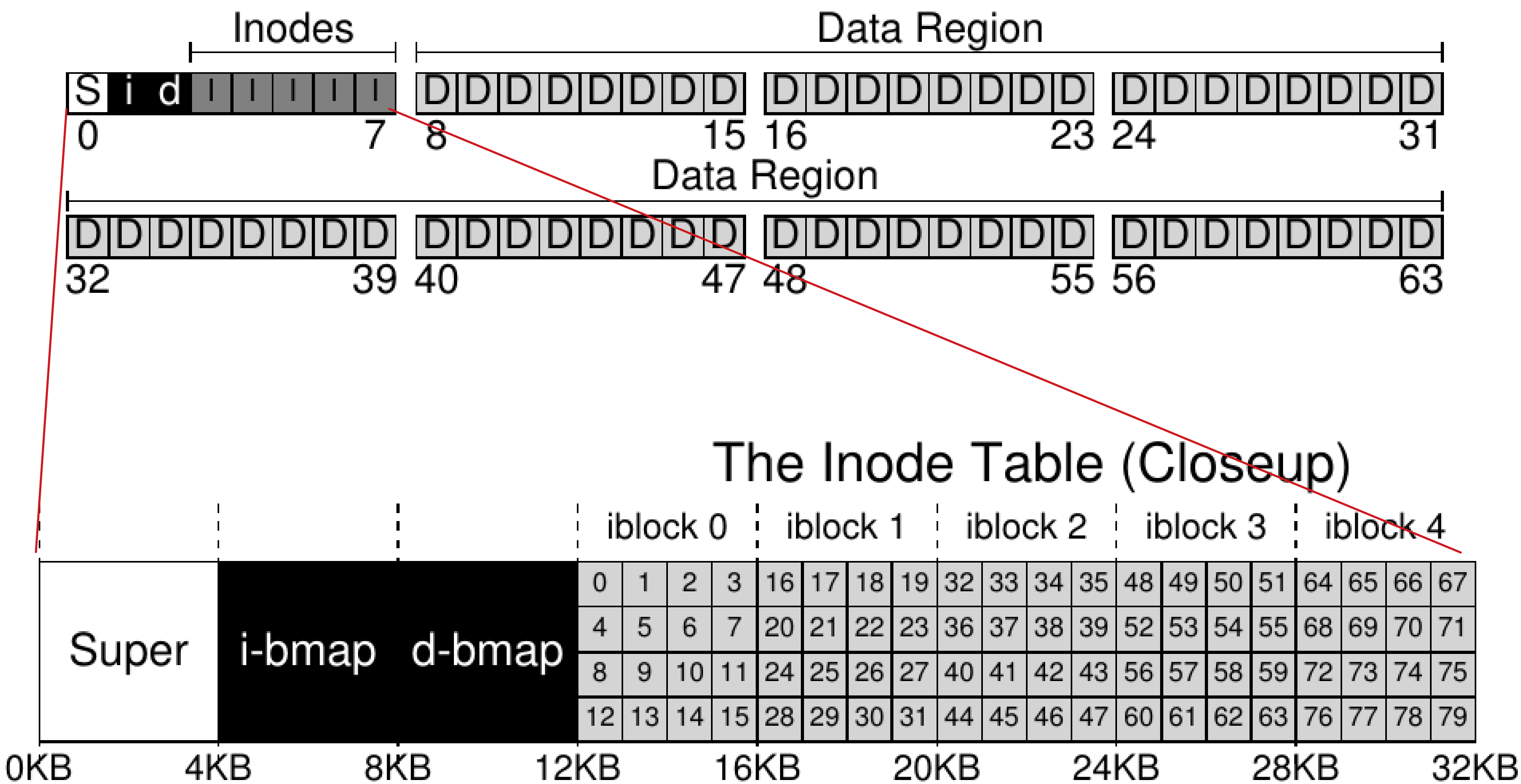
- Journal storage consists of a manager and a cell storage where the entire history of a variable is maintained, rather than just the current value.
  - A fairly elaborate organization for storing composite objects such as records consisting of multiple fields.
- The journal manager translates user requests to commands sent to the cell storage:
  - read a cell; write a cell; allocate a cell; deallocate a cell

Journal Storage

- A journal is a buffer that stores a record of all the changes that we make to the file system



Simple file system with blocks



inode

- Metadata: all of the extra information stored by a file system that is needed for the management of files.
  - To store this information, file systems usually have a structure called an inode (index node)
- Most modern systems have some kind of structure like this for every file they track, but perhaps call them different things (such as dnodes, fnodes, etc.)

Other structures

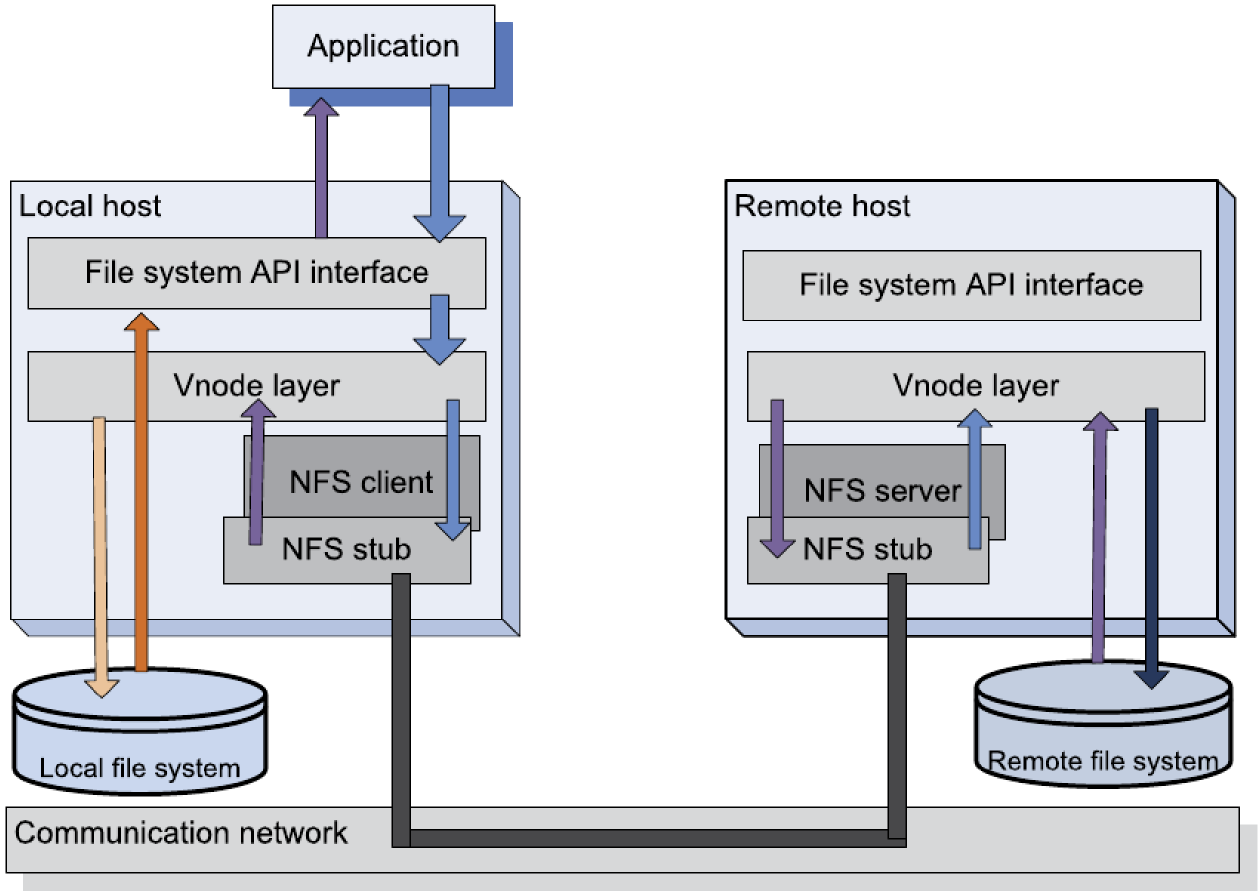
- inode table: simply holds an array of on-disk inodes and locates on a portion of disk
- allocation structures: some way to track whether inodes or data blocks are free or allocated
- Ex: two bitmaps for data and inode, each bit is used to indicate whether the corresponding object/block is free (0) or in-use (1)
- superblock contains information about this particular file system, including, for example, how many inodes and data blocks are in the file system

Distributed File Systems

NFS (Network File System)

- NFS was the first widely used distributed file system and is based on the client-server model
- Motivation: the need to share a file system among a number of clients interconnected by a local area network
- NFS Provides the same semantics as a local FS to ensure compatibility with existing applications (and locally transparent)
- NFS is very popular and have been used for some time, but does not scale well and has reliability problems

NFS architecture (I)

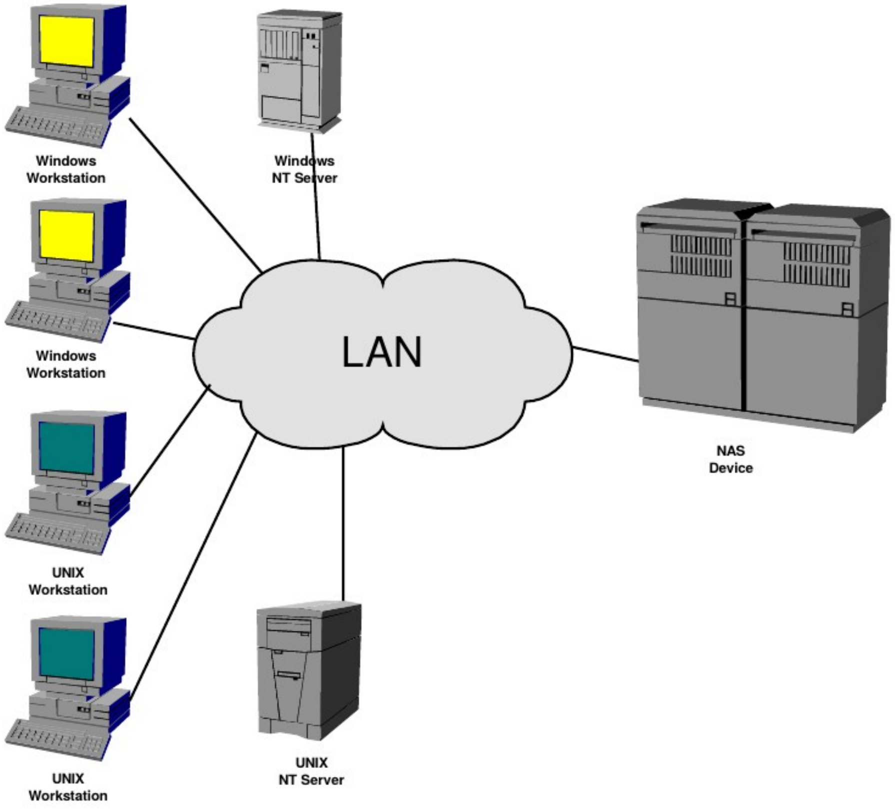


NFS architecture (II)

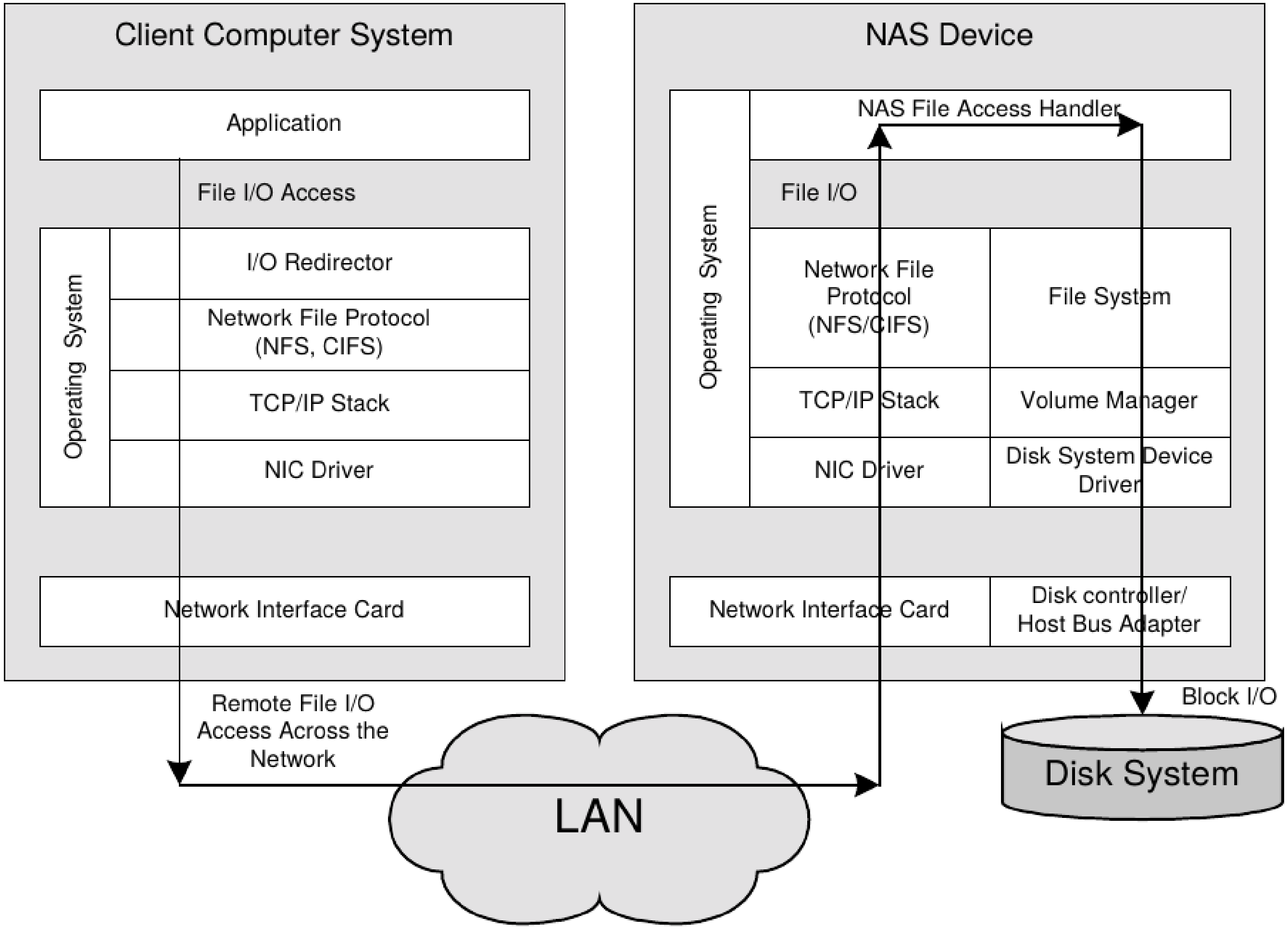
- NFS protocol: A set of remote procedure calls that provide the means for clients to perform operations on a remote file store
- NFS server module: resides in the kernel on each computer that acts as an NFS server
- Requests referring to files in a remote file system are translated by the client module to NFS protocol operations and then passed to the NFS server module at the computer holding the relevant file system

NAS (Networked Attached Storage)

- NAS is generally referred to as storage that is directly attached to a computer network (LAN) through network file system protocols such as NFS and CIFS
- Connectivity is through the Network Interface Card and the Ethernet frames carrying the remote file access commands

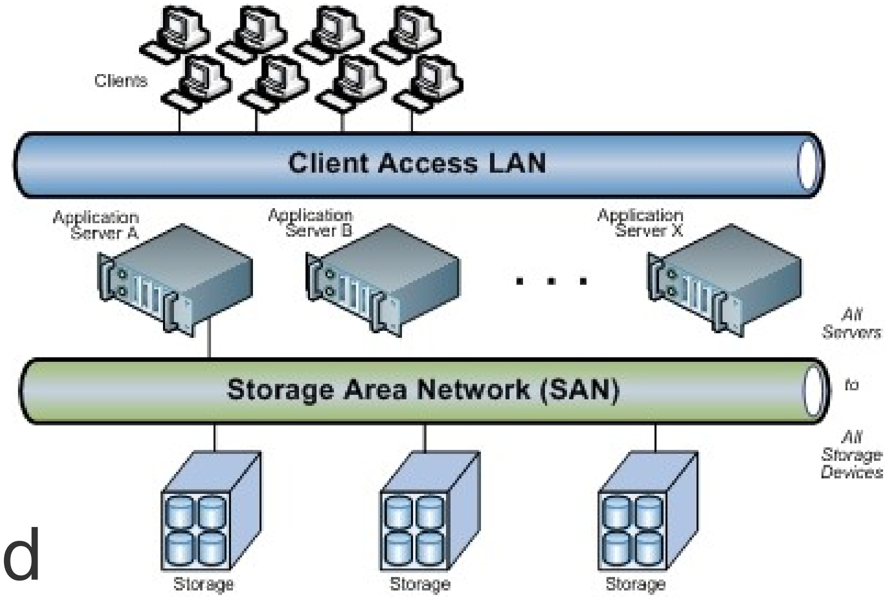


NAS Software Architecture

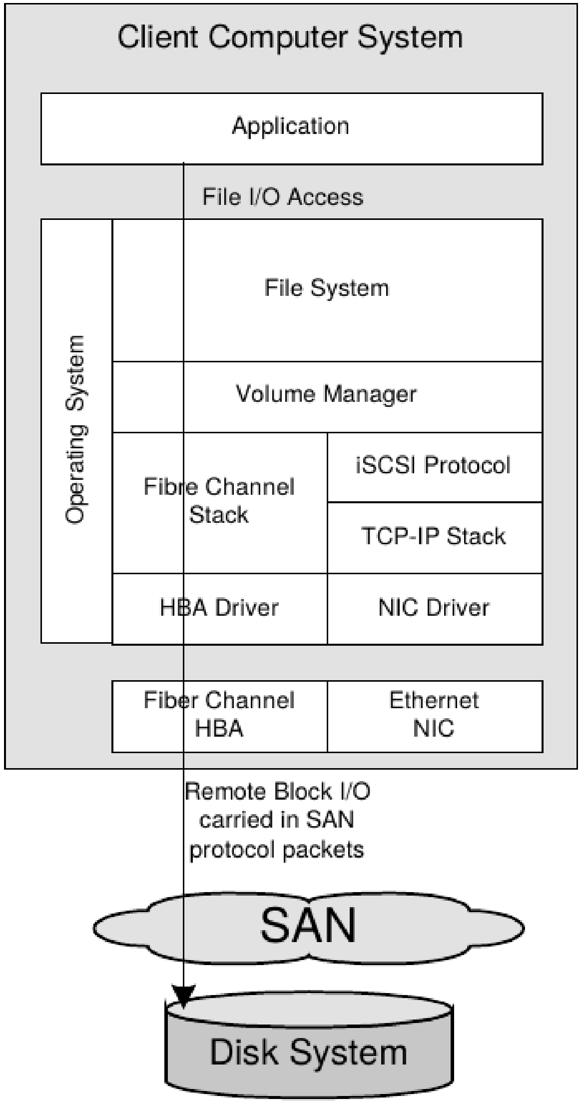


SAN (Storage Area Network)

- Advances in the networking technology allow the separation of the storage systems from the computational servers
- SAN is the network used to interconnect the storage and computational servers
- A SAN-based implementation of a file system can be expensive, as each node must have a Fiber Channel adapter to connect to the network



SAN software architecture



SAN vs NAS

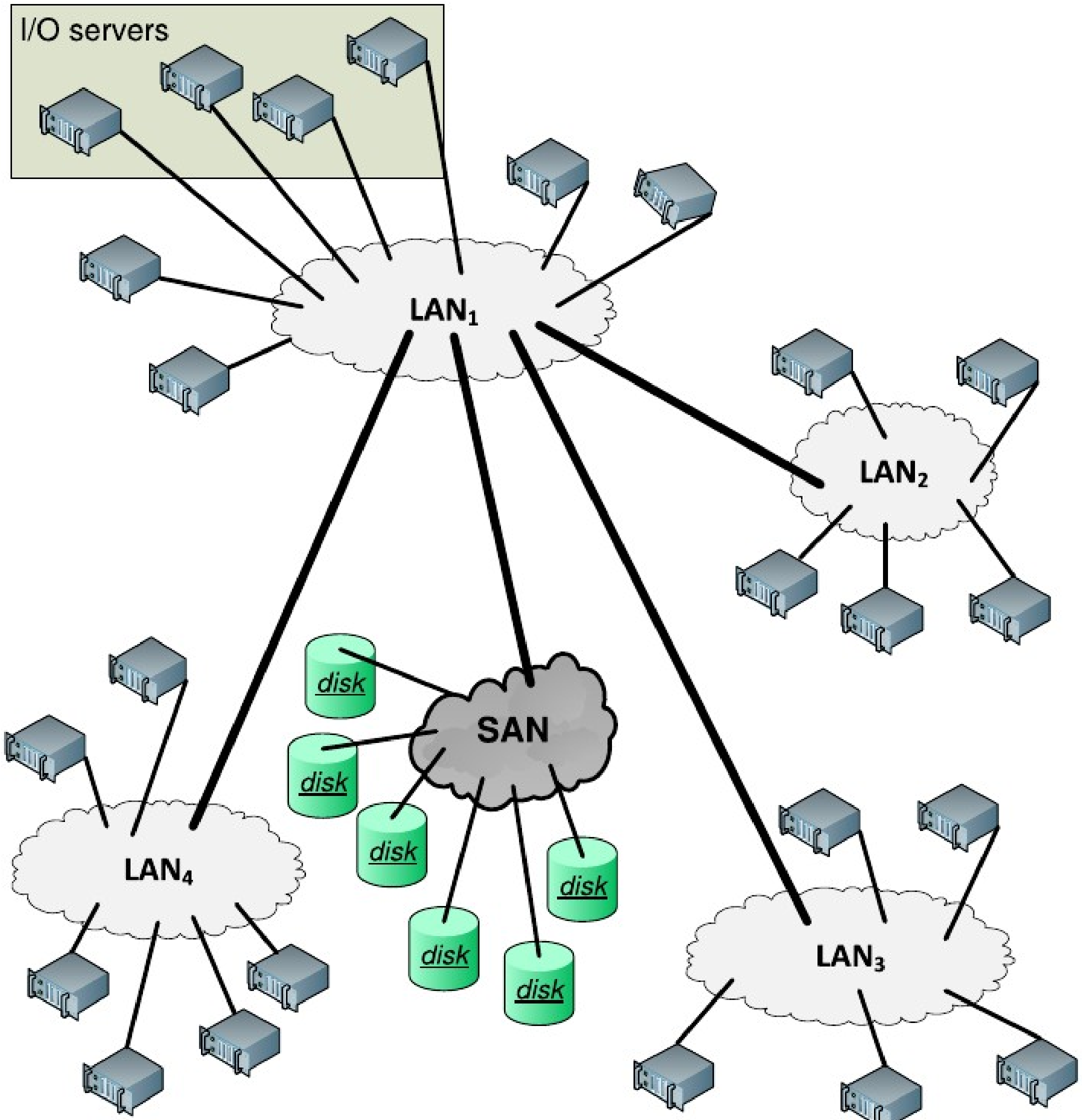
Architecture	I/O	Connectivity
SAN	block-orient	Dedicated (fiber channel)
NAS	File-oriented	Ethernet

features	speed	scalability	cost
SAN	Hight through FC	Capacity can be added as required	expensive
NAS	Not fast enough for HPC	Two many NAS devices is not desired and efficient	Lower than SAN

PFS (Parallel File System)

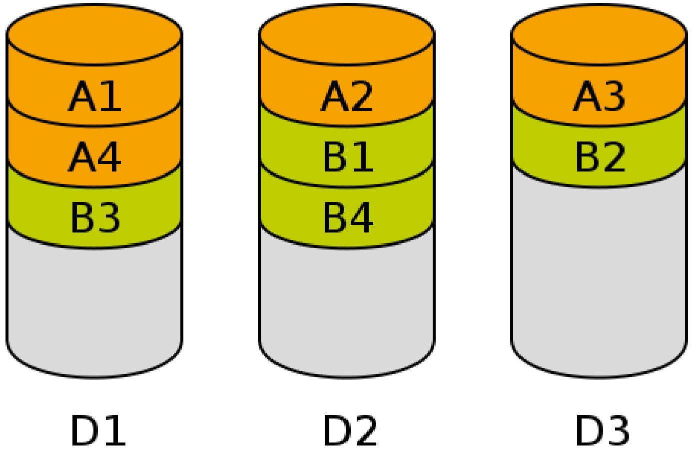
- Once the distributed file systems became ubiquitous, the natural next step in the file systems evolution was supporting parallel access.
- Parallel file systems allow multiple clients to read and write concurrently from the same file
- The General Parallel File System (GPFS) was developed by IBM in early 2000s
  - It was designed for optimal performance of large clusters
  - GPFS can support a FS of up to 4 petabytes consisting of up to 4096 disks of 1 TB each

GPFS configuration



GPFS properties (I)

- GPFS uses data stripping
  - So processing devices can request data more quickly than a single storage device can provide it
  - It is useful for parallel computing in clouds



An example of data striping. Files A and B, of four blocks each are spread over disks D1 to D3.

GPFS properties (II)

- To recover from system failures GPFS records all metadata updates in a write-ahead log file.
  - Write-ahead means that 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 and any I/O node is able to initiate recovery on behalf of a failed node
- GPFS consistency and synchronization are ensured by a distributed locking mechanism

Google File System (GFS)

- A scalable distributed file system for large distributed data-intensive applications
  - It provides fault tolerance
  - Can run on inexpensive commodity hardware
  - It delivers high aggregate performance to a large number of clients

GFS architecture (I)

- GFS cluster:
  - Single Master
  - Multiple Chunkserver
  - multiple clients access the cluster

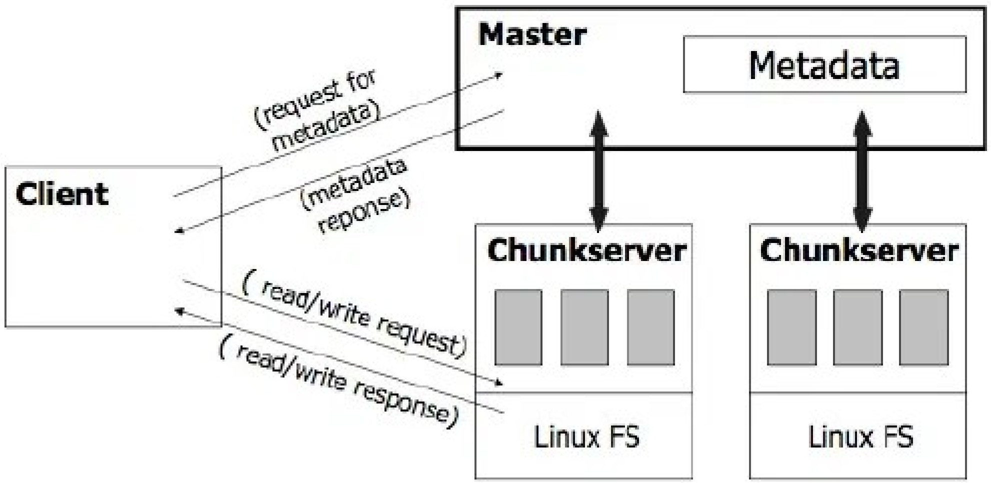
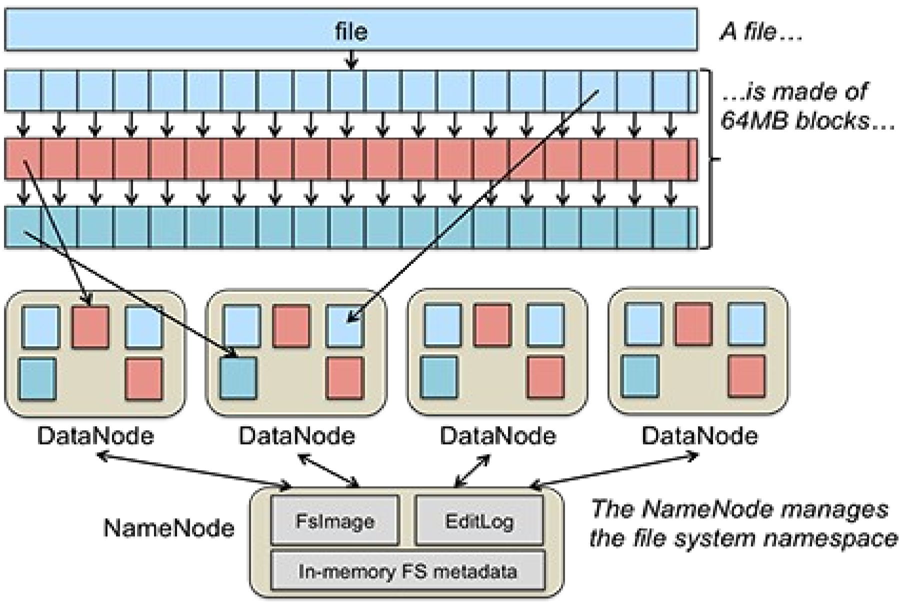


Figure 1

GFS architecture (II)

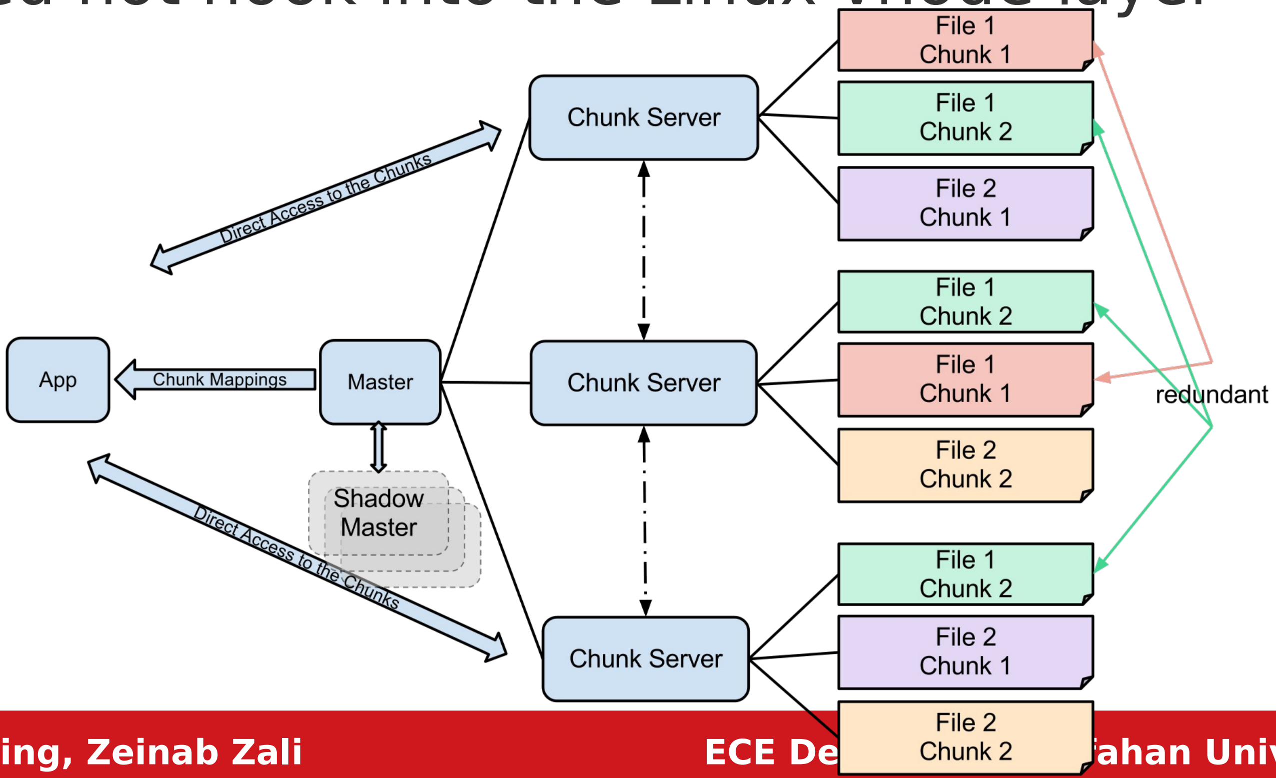
- Chunk: Files are divided into fixed-size chunks
- Each chunk is identified by an immutable and globally unique 64 bit chunk handle assigned by the master at the time of chunk creation



- Chunk servers store chunks on local disks as Linux files and read or write chunk data specified by a chunk handle and byte range
- For reliability, each chunk is replicated on multiple chunk servers

GFS architecture (III)

- The master periodically communicates with each chunkserver in HeartBeat messages to give it instructions and collect its state
- GFS does not provide the POSIX API and therefore need not hook into the Linux vnode layer



Data Base Management System (DBMS)

DBMS

- Database: a collection of logically-related records
  - Most cloud applications do not interact directly with the file systems, but through an application layer which manages a database
- Data Base Management System (DBMS): The software that controls the access to the database
  - The main functions of a DBMS: enforce data integrity, manage data access and concurrency control, and support recovery after a failure
  - Query language: each DB has a dedicated programming language used to develop database applications

DBMS history

- navigational model: 1960
- relational model: 1970
  - Oracle, MySQL, SQLServer, and Postgres
- object-oriented model: 1980
- NoSQL model: first decade of 2000
  - mongoDB

ACID properties

- **A**tomicity: a transaction must be all or nothing
- **C**onsistency: a transaction takes the system from one consistent state to another consistent state.
- **I**solation: each transaction must be performed without interference from other transactions.
- **D**urability: after a transaction has completed successfully, all its effects are saved in permanent storage.

DBMSs and Clouds

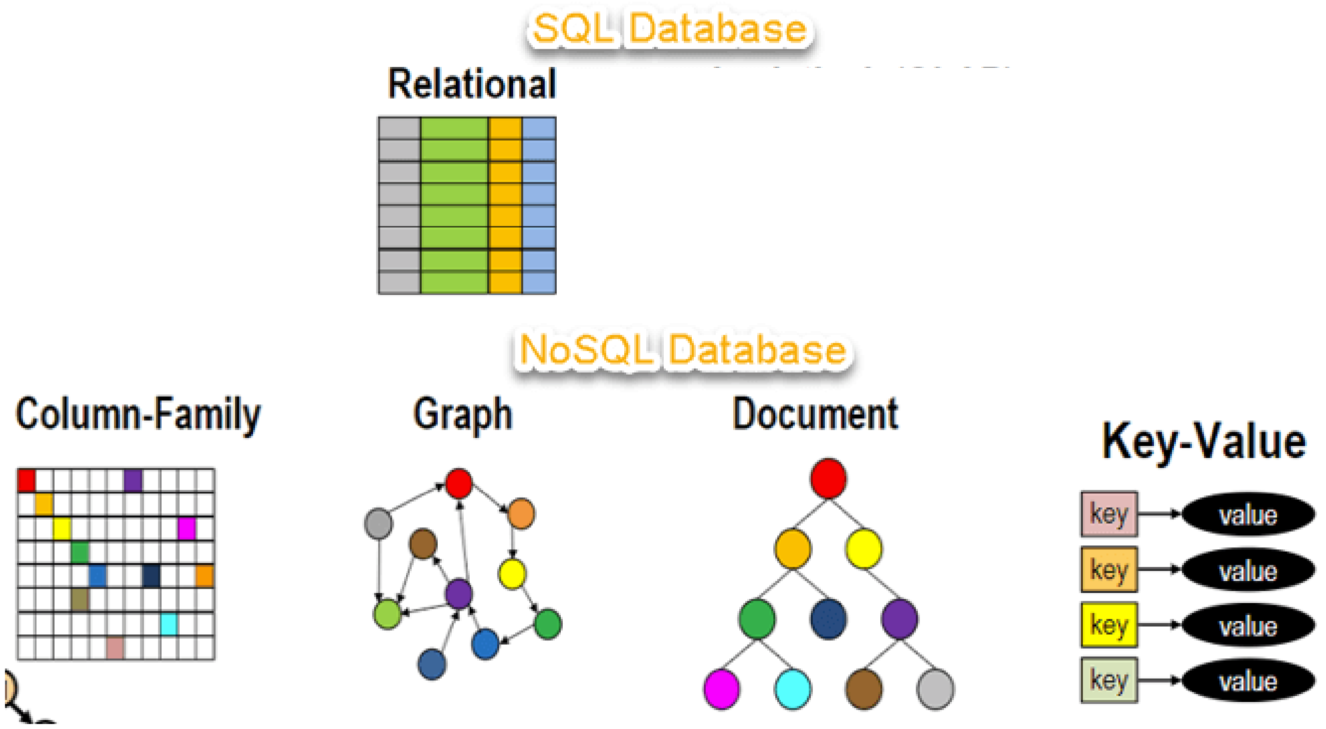
- cloud applications require:
  - low latency, scalability, high availability, and demand a consistent view of the data
  - storing unstructured or semi-structured data
- The requirements cannot be satisfied simultaneously by existing database models
  - relational databases are easy to use for application development, but do not scale well.
- The NoSQL model is useful when
  - the structure of the data does not require a relational model
  - the amount of data is very large

Problems with traditional RDBMS

- Not scalable:
  - Replication is required for ensuring fault-tolerance of large-scale systems built with commodity components
- High overhead origins from:
  - implementations of ACID transactions, multi-threading, and disk management

NoSQL concepts vs RDBMS

- storing unstructured or semi-structured data in noSQL vs structured data in RDBMS



NoSQL concepts vs RDBMS

- Different names in NoSQL:
  - a partition became a shard
  - a table is a document root element
  - a row is an aggregate/record
  - a column is an attribute/field/property
- NoSQL model does not support SQL as a query language

NoSQL DB Types

- Key-value model data as an index key and a value.
  - Ex: Riak and Amazon's Dynamo

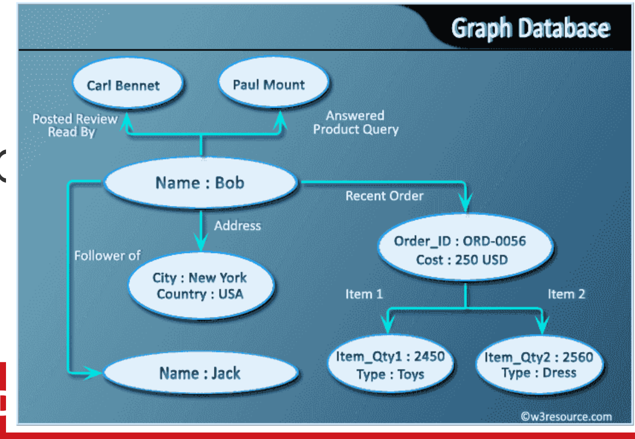
Key	Value
"India"	{"B-25, Sector-58, Noida, India – 201301"}
"Romania"	{"IMPS Moara Business Center, Buftea No. 1, Cluj-Napoca, 400606",City Business Center, Coriolan Brediceanu No. 10, Building B, Timisoara, 300011"}
"US"	{"3975 Fair Ridge Drive. Suite 200 South, Fairfax, VA 22033"}

- Document store are similar to key-value, but the value associated with a key contains structured or semi-structured data.
  - Ex: MongoDB

```
{officeName:"3Pillar Noida",
{Street: "B-25, City:"Noida", State:"UP", Pincode:"201301"}
}
{officeName:"3Pillar Timisoara",
{Boulevard:"Coriolan Brediceanu No. 10", Block:"B, Ist Floor", City: "Timisoara", Pincode: 300011"}
}
{officeName:"3Pillar Cluj",
{Latitude:"40.748328", Longitude:"-73.985560"}
}
```

NoSQL DB Types

- Column-family are large sparse tables with a very large number of rows and only a few columns.
  - Columnar databases store all the cells corresponding to a column as a continuous disk entry thus makes the search/access faster.
  - Google’s BigTable and HBase & Cassandra
- Graph databases where the nodes represent entities and the edges the relationships among the entities.
  - Ex: InfoGrid and InfiniteGrap



NoSQL Properties (I)

- is designed to scale well
- does not exhibit a single point of failure
- Supports partitioning and replication as basic primitives
- has built-in support for consensus-based decisions

NoSQL Properties (II)

- soft-state approach in the design of NoSQL
  - allows data to be inconsistent
  - transfers the task of implementing only the subset of the ACID required by a specific application to the application developer.
- The NoSQL systems ensure that data will be eventually consistent at some future point in time, instead of enforcing consistency at the time when a transaction is “committed.”
  - BASE instead of ACID

BASE properties

- **Basically Available:** availability of data even in the presence of multiple failures through a highly distributed approach to database management
- **Soft state:** abandoning the consistency requirements of the ACID model (data consistency is the developer's problem and should not be handled by the database)
- **Eventually consistent:** guaranteeing at some point in the future, data will converge to a consistent state

Online Transaction Processing (OLTP)

- Many cloud services are based on Online Transaction Processing (OLTP)
  - business models such as google, amazon, facebook, ...
- OLTP applications have to deal with
  - extremely high data volumes
  - providing reliable services for very large communities of users
  - real-time demands of online applications
  - operate under tight latency constraints

General solutions for OLTP

- The relational schema is of little use for OLTP applications

1) Decrease the latency: by caching frequently used data in memory on dedicated servers, rather than fetching it repeatedly

2) decreases the response time: by Distributing data to a large number of servers allows multiple transactions to occur at the same time

General solutions for OLTP

- The overhead of OLTP systems is due to four sources with equal contribution: logging, locking, latching, and buffer management
  - A latch is a counter that triggers an event when it reaches zero. For example a master thread initiates a counter with the number of worker threads and waits to be notified when all of them have finished

3) Logless, single threaded, and transaction-less databases could replace the traditional ones for some cloud applications

4) Data replication is critical not only for system reliability and availability, but also for its performance

OLTP caching

- Memcaching: refers to a general purpose distributed memory system that caches objects in main memory.
- memcaching is based on a very large hash table distributed across many servers.
- A memcached system is based on a client-server architecture and uses the LRU cache replacement strategy
- The servers maintain a key-value associative array. (key <= 250 bytes and value<=1MB)
- The API allows clients to add entries to the array and to query it;

BigTable

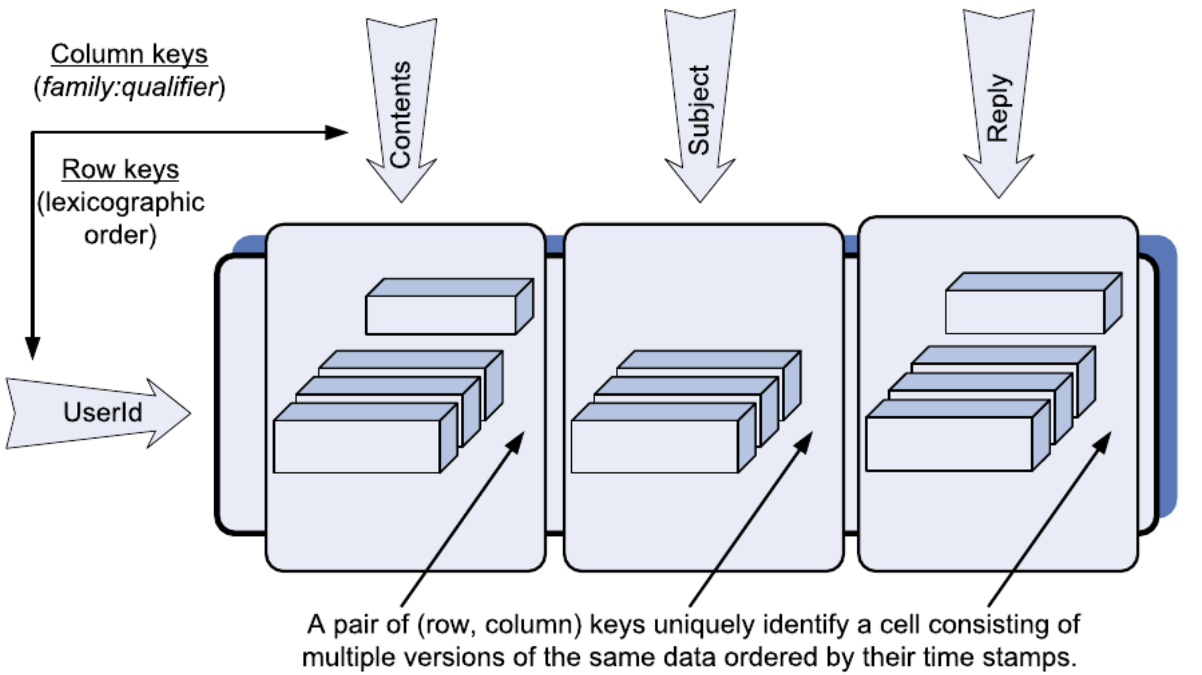
- BigTable is a distributed storage system developed by Google to store massive amounts of data and to scale up to thousands of storage servers
  - Used for Web indexing, Personalized Search, Google Earth, Google Analytics, Google Finance
- The system uses the GFS to store user data, as well as system information
- To guarantee atomic read and write operations, BigTable uses the Chubby distributed lock service

BigTable concepts

- Column keys identify units of access control called column families including data of the same type.
  - Ex: a string defining the family name(a set of printable characters) and an arbitrary string as qualifier.
- A row key is an arbitrary string of up to 64 KB and a row range is partitioned into tablets serving as units for load balancing.
- Any read or write row operation is atomic even when it affects more than one column.
- Time stamps used to index different versions of the data in a cell are 64-bit integers.

BigTable Example

- the organization of an Email application:
  - a row with the UserId key: ordered lexicographically
  - three family columns: Email contents , Email Subjects, Email Replies (a column key is obtained by concatenating the family and the qualifier fields)
  - the version of records in each cell are ordered according to their time stamps.



BigTable Example

- Storing and querying events of a social network
  - Such as following information

"follows" column family

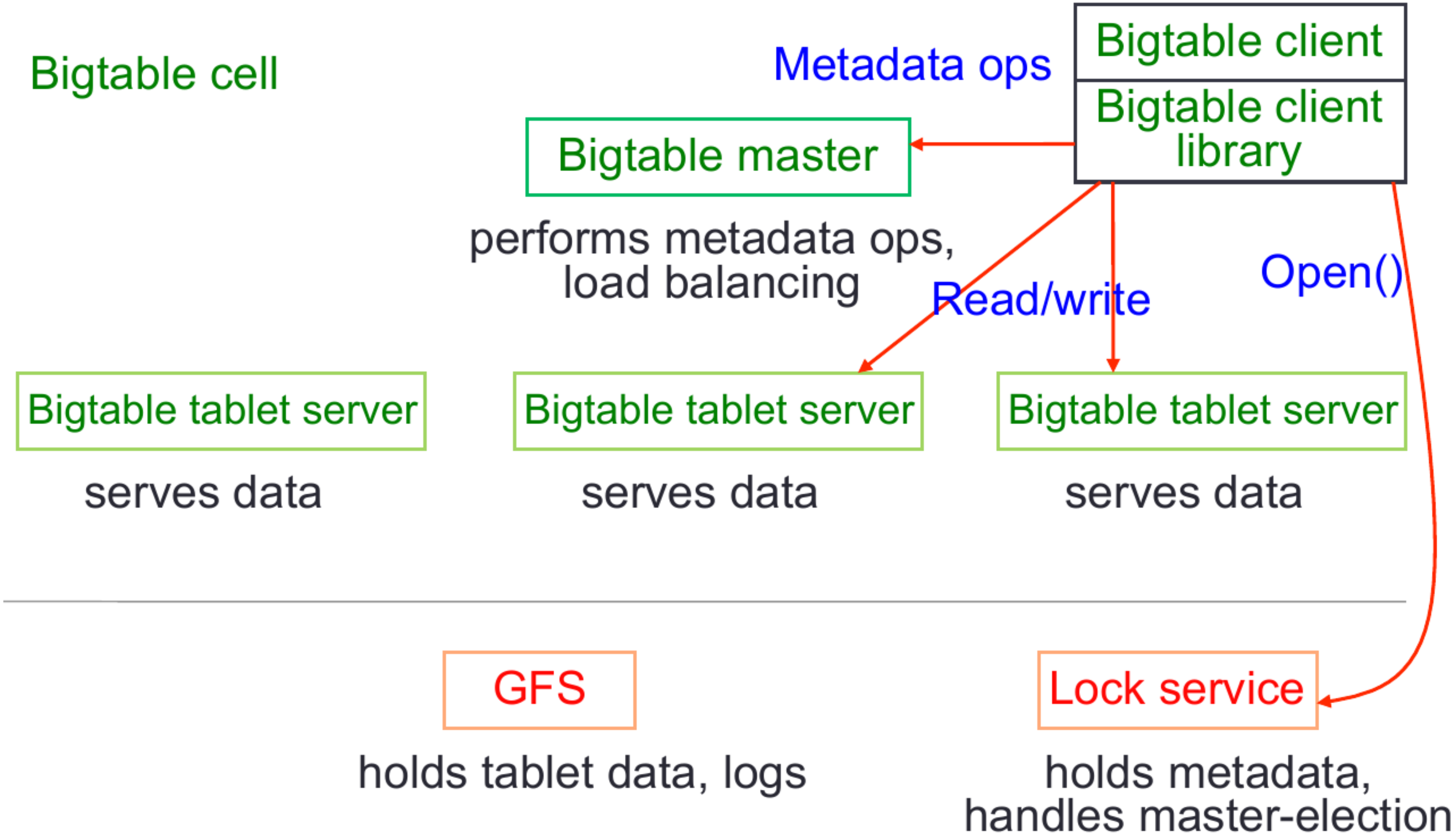
	Follows			
Row Key	gwashington	jadams	tjefferson	wmckinley
gwashington		1		
jadams	1		1	
tjefferson	1	1		1
wmckinley			1	

Multiple versions

BigTable applications

- Time-series data: such as CPU and memory usage over time for multiple servers.
- Marketing data: such as purchase histories and customer preferences.
- Financial data: such as transaction histories, stock prices, and currency exchange rates.
- Internet of Things data: such as usage reports from energy meters and home appliances.
- Graph data: such as information about how users are connected to one another.

BigTable architecture (I)



BigTable architecture (II)

- The system consists of three major components:
  - a library linked to application clients to access the system
  - a 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
  - a large number of tablet servers:
    - A Bigtable table is partitioned into many tablets based on row keys
    - Each tablet contains blocks of contiguous rows

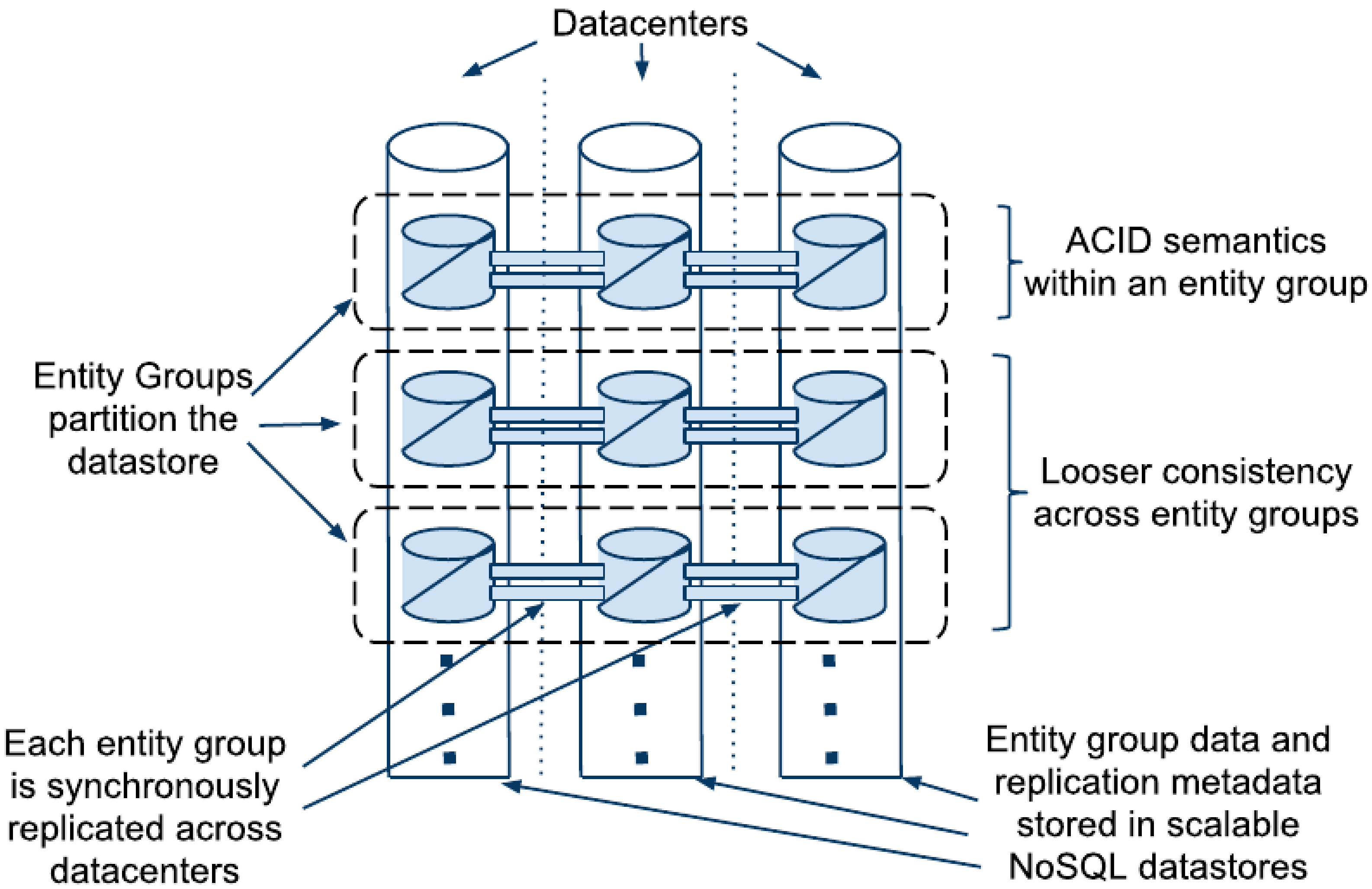
BigTable Implementation

- GFS
  - For storing log and data files
- Chubby
  - Ensure there is only one active master
  - Store bootstrap location of Bigtable data
  - Discover tablet servers
  - Store Bigtable schema information
  - Store access control lists

MegaStore

- Megastore is widely used internally at Google.
- The basic design philosophy of the system is to partition the data into entity groups and replicate each partition independently in data centers located in different geographic areas
- The system supports full ACID semantics within each partition and provides limited consistency guarantees across partitions

MegaStore structure



MegaStore application Example(I)

- Email: Each email account forms a natural entity group.
  - Operations within an account are transactional and consistent: a user who sends or labels a message is guaranteed to observe the change despite possible fail-over to another replica.
  - External mail routers handle communication between accounts.

MegaStore application Example(II)

- Blogs A blogging application would be modeled with multiple classes of entity groups.
  - Each user has a profile which is naturally its own entity group.
  - a second class of entity groups to hold the posts and metadata for each blog