File Systems

2024 Semester 2 COMPSCI 340: Operating Systems
Talia Xu

Lecture 1
1.0.0

What can happen on a crash?

Assume we are creating and writing to a new file /foo/bar
Write in general (not just the first write)

Operation	data bitmap	inode Bitmap	root inode	foo's inode	bar's inode	root data	foo data	bar data[0]	bar data[1]	Step #
create(bar)			read							1
						read				2
				read						3
							read			4
		read								5
		write								6
							write			7
					read					8
					write					9
				write						10
write()					read					11
	read									12
	write									13
									write	14
					write					15

What can happen on a crash?

Assume we are creating and writing to a new file /foo/bar

A crash after the completion of:

- Step 1-5: No corruption.
- Step 6: i-node bitmap points to garbage data.
- Step 7-9: Inconsistent metadata.
- Step 10-12: No corruption.
- Step 13: data bitmap points to garbage data.
- Step 14: Corrupted data.
- Step 15: The new data cannot be found.

What can happen on a crash?

Lost Files: New files being created might not be fully registered in the directory structure, leading to them being ”lost” even if their data exists on disk.

Inaccessible Files: Existing files might become inaccessible due to corrupted metadata or lost pointers to their data blocks.

Incorrect File Sizes: The file system might report incorrect file sizes due to incomplete writes or metadata updates.

Inconsistent Metadata: Applications trying to access affected files might crash, freeze, or produce errors due to the inconsistencies.

Recovery from failure

Consistency checking compares data in directory structure with data blocks on disk, and tries to fix inconsistencies.

- Tools: fsck for Unix systems, chkdsk for Windows.
- Scanning the entire disk, slow, very slow ...

Journaling: A system that keeps track of changes that will be made in the file system to ensure data integrity.

Versioning and Snapshots: Maintain historical versions or states of files and systems to roll back when needed.

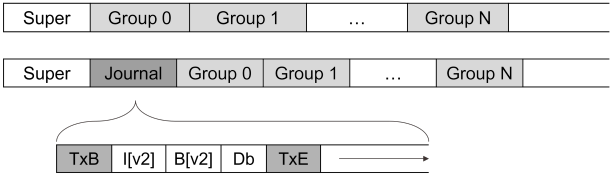
Backup and Restore: Periodic backups to external devices or cloud storage, ensuring data can be restored from a known good state.

Journaling (a.k.a Write-Ahead Logging)

Used by many modern file systems:

- Linux (ext3, ext4), Windows (NTFS), IBM (JFS)

First write down what you are going to do, and then carry out the action



Journaling (a.k.a Write-Ahead Logging)

Physical journal

- all changed data is recorded in the journal first

Logical journal

- only changed metadata (like inode tables, directory structures) is recorded

Journaling (a.k.a Write-Ahead Logging)

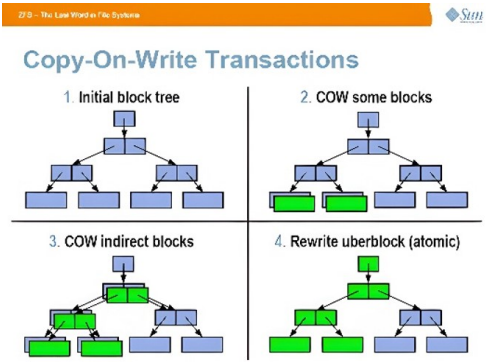
What if absolute data integrity is crucial?

- for example, in banking, aviation systems, stock exchange?

How do we ensure that data integrity is always maintained?

ZFS

What happens if system crashes before each step can be completed?



Why distributed system? An example

2004: Facebook started on a single server

- Web server front end to assemble each user's page.
- Database to store posts, friend lists, etc.

2008: 100M users

2010: 500M users

2012: 1B users

2019: 2.5B users

Why distributed system?

Nature of the application

- Multiplayer games
- Collaborative Projects & Cloud App

Availability despite unreliable components

- A service shouldn't fail when one computer does. •

Conquer geographic separation

- A web request in NZ is faster served by a server in NZ than by a server in US.

Scale up capacity

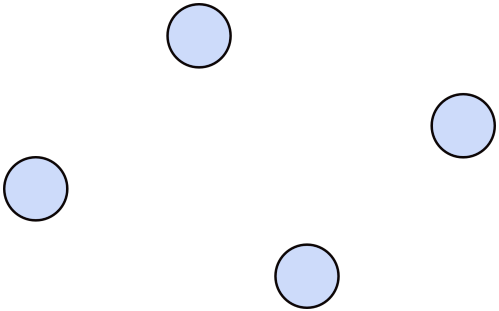
- More CPU cycles, more memory, more storage

What is a distributed system?

“A collection of loosely coupled nodes interconnected by a communication network.”

— textbook

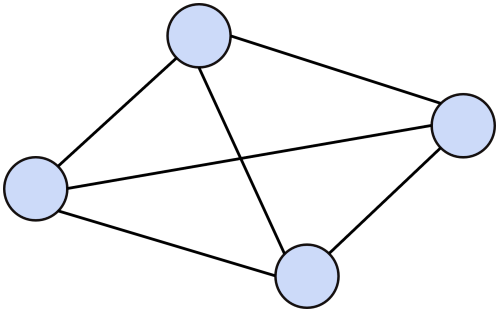
What is a distributed system?



Independent components or elements

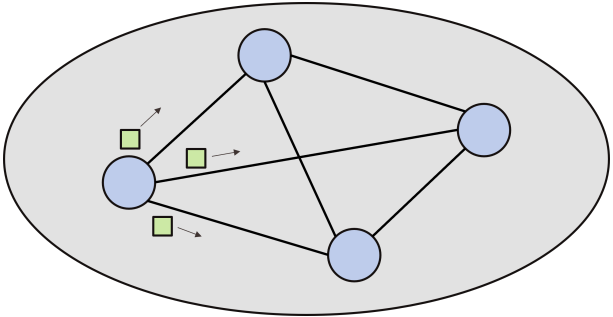
- software processes or hardware used to run a process, store data, etc.

What is a distributed system?



Independent components or elements that are **connected by a network.**

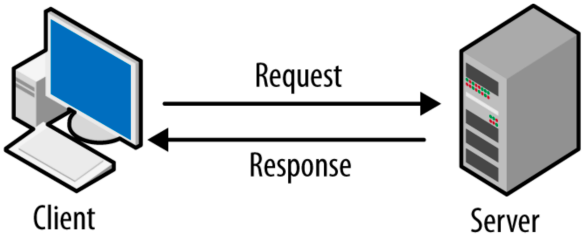
What is a distributed system?



Independent components or elements that are **connected by a network** and **communicate by passing messages** to achieve a common goal, appearing as **a single coherent system**.

When one (or more) component fails, the system does not fail.

The client-server model



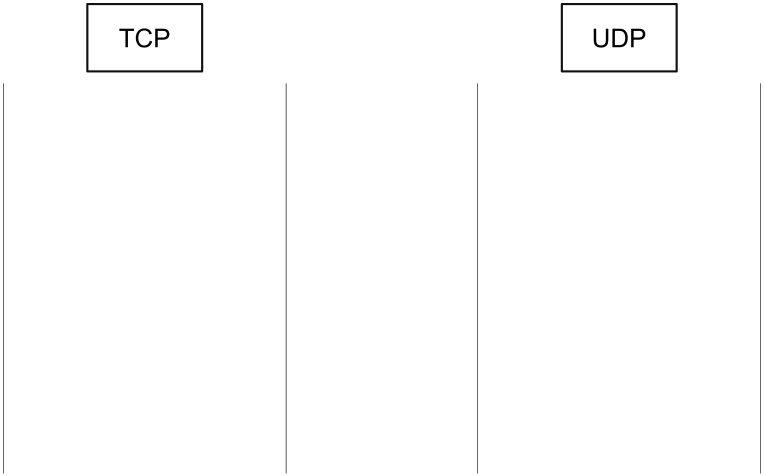
Unstructured communication

- Use shared memory or shared data structures

Structured communication

- Message-oriented model

Communication protocols



Communication protocols

TCP (safe but slow)

Double lines convey security and two-way transmission

UDP (fast but unreliable)

Arrowheads suggest speed, but dotted pattern suggests poor reliability

Remote service - stateful

Server knows:

- who has the file open
- for what type of access
- and where it is in the file etc.
- When the client calls open, it receives an identifier to be used to access the file.
- Looks very similar to traditional local file access to the client process.
- Efficient, the needed data may be read ahead by the server.
- Information about the file is held in memory.
- If the server crashes
  - it is difficult to start again since all the state information is lost.
- Server has problems with processes which die
  - needs to occasionally check.

Remote service - stateless

Server does not maintain information on the state of the system. It merely responds to requests.

Open and close calls don't send messages to the server. Handled locally. (Except for access privileges.)

Requesting processor has topass all the extra information with each read/write

- e.g., the current file location the process is reading from, accesibility

Server doesn't have to worry about processes stopping

No complicated recovery process if the server goes down