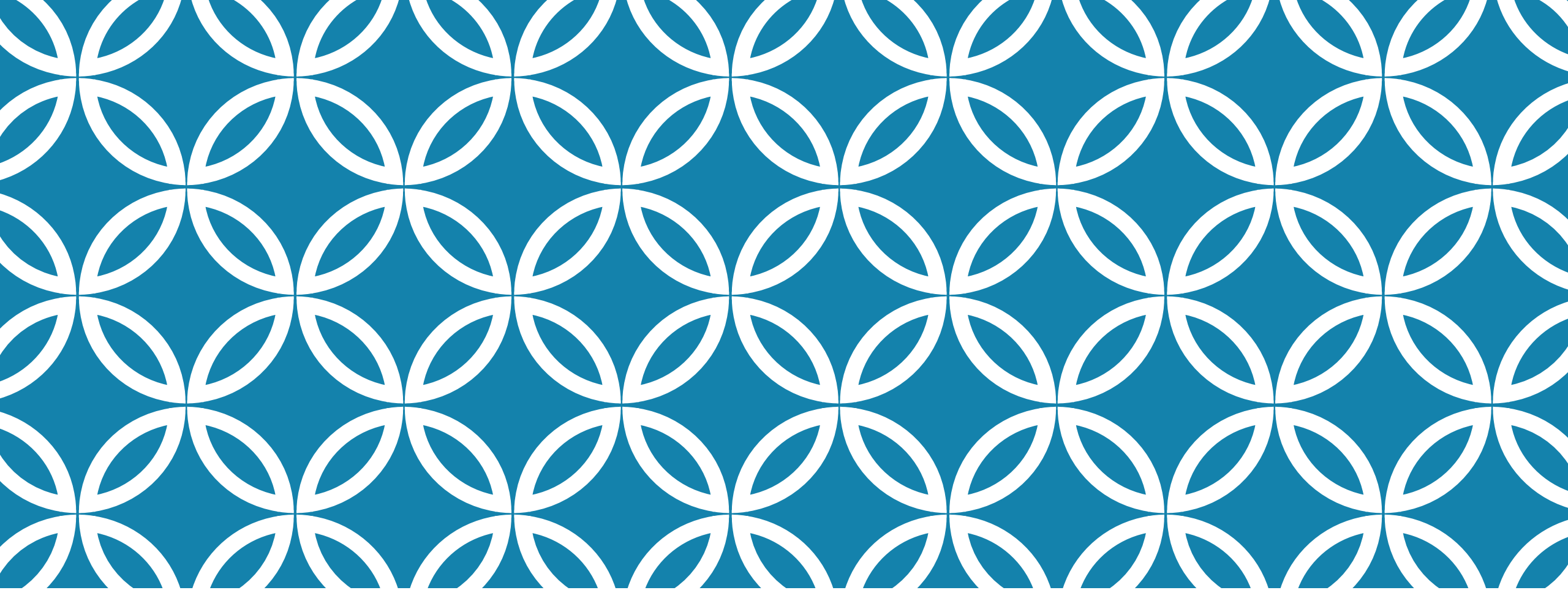


NETWORKED FILE SYSTEMS

Professor Ken Birman

CS4414 Lecture 20

IDEA MAP FOR TODAY

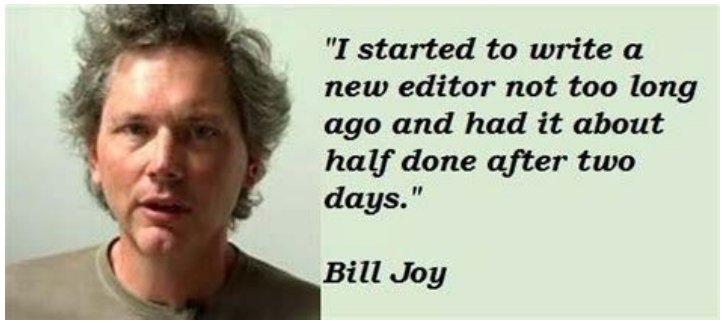
With TCP, it is possible for Linux itself to reroute file system requests to some remote machine.

This enables a "remote" file system

Ceph: A remote file system specifically tuned for storing objects

Zookeeper: A remote file system for distributed coordination

BILL JOY



Bill was a UC Berkeley PhD student in 1980. He created the BSD version of Linux (among other things, like vim editor).

At the time Unix (pre-Linux) used to crash and end up with damaged file systems, which had to be repaired by hand. Bill rewrote the file system. His innovations improved speed 10x and eliminated the crash issue.

He did it using the kinds of ideas we have talked about in CS4414!

SUN MICROSYSTEMS

Bill started a company, early in the .com boom era.

The company took the BSD Linux and created a workstation computer carefully designed for modern users, as of 1980.

Their potential clients were purchasing DEC computers at the time, which also had Linux, but included a “cluster” option

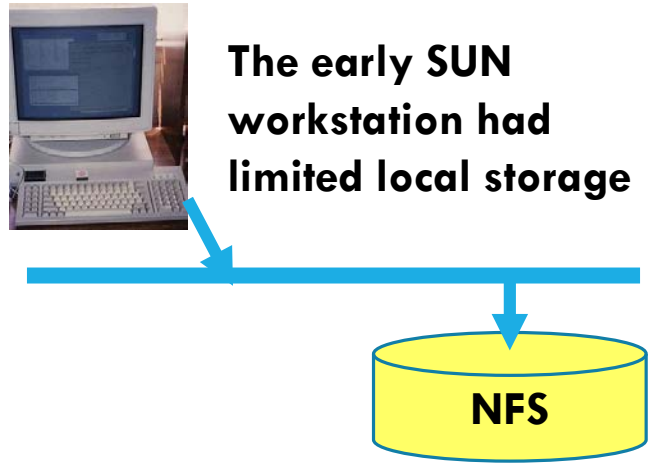
HOW TO COMPETE AGAINST DEC?

SUN took the approach that each computer is "on its own"

It might be a client of some remote services, but it would be able to keep running even if those were down.

This differed from DEC, where the whole cluster needed to be working for the computer itself to be "available". In fact DEC was doing an early NUMA design.

SUN REMOTE FILE SYSTEM IDEA



SUN introduced the "network file system" or NFS

Bill created it. His basic idea was very simple.

He mimicked the hardware interface to a normal disk, but talked to it over a network, via a SUN remote procedure call package (like Google GRPC).

HOW DID THEY DO IT?

First, a little context: The earliest version of Unix only supported a fairly small local file system on a single disk.

This quickly was inadequate. Users wanted to share files. The idea of "mounting" a remote file system emerged.

Then this was expanded to deal with "disk array" hardware, where there are multiple disks in a single big storage unit.

CONCEPT: MOUNT POINT

Any directory in the active file system, like /mnt/c

The mount command maps it to the root of a file system on some storage drive

Now, / mnt/c/x lets you access x in the other file system.

CONCEPT: DEVICE DRIVER

A kernel such as Linux talks to devices through “device driver” modules, specific to the hardware.

So for example, a Western Digital USB hard drive would have an associated device driver that Linux can load and run.

ROLE OF THE DEVICE DRIVER?

Recall the layered abstractions Dijkstra proposed.

A device driver takes a general purpose system, like a WD USB, and creates a uniform way for Linux to talk to it.

- Linux is one of many operating systems, and Ubuntu is one of many versions of Linux
- Internally, each of these has its own "opinion" about the best way to manage a storage device
- Device driver makes a wide range of devices look standard/uniform.

… SO (BACK ON TOPIC)

… SUN created a device driver for a remote storage unit that it would talk to over a network.

Now mount can work as usual.

- Local file system is fast to access, but private and has less capacity.
- Linux booted from the local file system, then mounts the remote one.
- Remote file system is massive. We incur a network delay to access it, but it can be shared with others, which is convenient in many ways.

ADVANTAGE SUN?

DEC's NUMA clustering approach was amazing, but fairly complex. Many customers found it "immature" and unreliable.

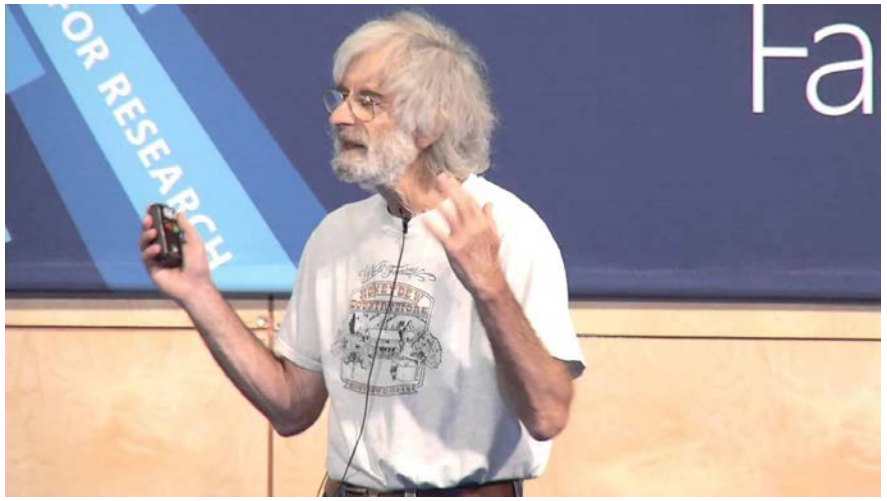
SUN's NFS was very simple and robust… in part because Linux was extended by SUN to survive even if the NFS crashed or rebooted when the workstation was running.

FAMOUS 1980'S QUOTE

Bill Joy (SUN founder, CTO; created NFS): *"I came to work one morning and continued to edit a program I was writing. It was open on the screen – I hadn't closed it when I left work.*

At lunch, someone asked me how I liked the new storage product: They had upgraded during the night and I never even noticed!"

FAMOUS 1980'S QUOTE



Leslie Lamport (Turing Award for distributed computing)

Leslie Lamport: "*Since the start of my career I've worked on the theory of fault tolerant distributed computing. Yet we have lacked real distributed systems where I could apply this theory.*

But now the future finally arrived: Today I was unable to work due to the failure of a computer I have never even heard of!"

ISSUES WITH NETWORKING

Bill and Leslie were highlighting two dimensions of modern computing.

Bill's experience with NFS was quite good: his workstation never even noticed the new server!

Leslie's experience (at SRI and then Microsoft, not SUN) highlighted a risk: dependencies on remote components, fault-tolerance issues.

TCP VERSUS UDP

SUN remote procedure call used UDP: "User Datagram Protocol". Each message is sent by itself. Routing is the same as with TCP, but if any packet is lost, the whole UDP message will be dropped.

Leslie was using a system similar to GRPC that ran on TCP streams, which are reliable and sequenced. But TCP also checks that the two endpoints both remain alive.

SUN RPC RETRANSMITS LOST MESSAGES, THEN EVENTUALLY GIVES UP

SUN gave each message a unique id. It would resend a few times if a request or response was dropped. This is all we need normally, when interacting with a server in an active way.

When Bill stopped editing, his SUN workstation had no reason to send more messages. So it never noticed that the server was replaced.

THE EARLY VERSION OF SUN RPC HAD NO REAL SECURITY

Anyone could access any NFS server, read or write any file

In fact it even trusted user ids: Anyone could access any server *as any legitimate user*.

After two years, SUN finally fixed this. Security just wasn't the priority for those first two years!

MOUNTING AN NFS DISK

The NFS driver needed to know a computer's name, to turn this into an IP address. NFS used a standard port number.

But then it "uses" this information only when actually sending requests (to read or write blocks on the remote device)

NFS has no reason to "chit chat" if a machine is idle.

… AT XEROX

Leslie's computer was using a Xerox RPC over TCP.

With TCP, if a storage server was removed some evening and replaced with a larger, faster one, even if it has the same IP address and port number, the connection would break

- TCP is continuously numbering bytes in each direction (SEQ numbers)
- The new computer wouldn't know about the old, open, TCP session
- So… the session will break next time we use it.

THE PUZZLE WITH NETWORKING FAILURES

Consider these two cases:

1. Some computer catches fire, crashes, and will never recover. The "roles" it played (like running an NFS file server) will be moved to some other computer (maybe with the same name)
2. The cat is playing with the network cable, and manages to pull it loose. Someone will notice and plug it back in. No computer crashed at any time.

THE PUZZLE WITH NETWORKING FAILURES

… they look identical!

So, in a networked file system, you can see errors that do not ever arise with a local disk.

If a local disk "fails" this is a hardware issue. It may be seriously broken. If a networked disk fails, perhaps this just means there is a transient networking issue and packets cannot get through.

LINUX HAS NEVER DEALT WELL WITH THIS!

With NFS, we can get some very strange file system outages.

For example, many "older" Linux programs create a file as a way to do distributed locking.

- The application creates a file "lock" in some working directory
- If other applications try and create this same file, their requests fail, and they loop.
- To release the lock, remove ("unlink") the file

RECALL THAT NFS RETRIES IF AN RPC FAILS

Consider this (real) case:

- Bill opens nsf.c from vim (he wrote this editor, too).
- The editor wants to lock nsf.c against concurrent edits, so first it creates a file .nsf.c-lock ("invisible" unless you know to look for it)
- The RPC request gets through, but the response is somehow dropped.

  … so, the lock file was created, but Bill's machine doesn't know
- Bill's machine re-issues the request. But it fails: the lock file exists.
- Bill's sees "Error: File is locked against edits". **Why is this wrong?**

WHAT DOES BILL DO?

In fact, he reboots

Linux has tools to automatically remove temporary files, and Bill sets them up to also look for and remove any lock files.

So, this clears the issue… but in an ugly way!

OTHER LINUX OPTIONS

Modern Linux has a **file replace** feature:

- If a file is large, it may take a while to write out, and a crash could leave it corrupted. So how do you rewrite a large file?
- Create the new version first as a temporary file, and "fsync" it to disk. Then, replace the old one with the new one.
- Linux has an atomic rename operation that can do this for you.
- Rename is atomic on any given machine, including a remote server (NFS just sends the rename request to the server). But if a crash occurs at this moment, the process that called rename can get an error.

FAULT TOLERANCE IS A FUN TOPIC!

Leslie Lamport was a pioneer in this area. (So, in fact, was Ken)

But SUN, with its ultra-simple NFS solution still "won".

- DEC was successful with their Vax Cluster product, but their approach was very complicated and the tools weren't easy to use.
- SUN systems sometimes acted in weird ways, but rebooting (which removed all lock files!) generally fixed the issues.
- Neither company exists today… But these same puzzles remain!

MODERN VERSION OF NFS

In fact, NFS persisted and is still available and very widely used

But in recent years, people began to use file systems to store objects.

- They gave each object a name, like a file name.
- The serialized data would be the file contents.
- One object ≅ one file (with a few exceptions, like log files tracking the history of changes to some object).

CEPH FILE SYSTEM

Ceph responds to an issue this form of object storage causes:

- We can easily end up with directories that hold billions of objects
- NFS file servers were not designed for this and often crashed due to software and design limitations

CEPH FILE SYSTEM

Size distributions for objects are very different than for older NFS files:

- File systems are optimized for the most common cases.
- In NFS these tended to be small text files and large photos or videos
- With objects, the distribution simply looks different.

CEPH FILE SYSTEM

Lifetimes for objects are very different than for older Linux files:

- Even in Linux, a surprising number of files are created, but then immediately deleted. Some, like lock files, are never even read.
- Linux has some optimizations to avoid pointless I/O for such cases.
- With objects, the lifetime distribution simply looks different.

CEPH HAS NO IDEA WHAT IS IN AN OBJECT

It only sees the serialized form of an object.

Recall that when we serialize, an object is converted into a byte array of some size. Later the application would deserialize and reconstruct the object contents.

Ceph never sees object definition and cannot call the methods

YOU CAN ENCRYPT YOUR OBJECTS

With every form of Linux, you can specify that you want a file volume to be encrypted. *Don't lose the key!*

In such cases, all the bits that are written will first be encrypted, and later when you do a read, the data will be decrypted.

In Linux file systems this is block by block and invisible to you, once you enable it.

CEPH SUPPORTS ENCRYPTION TOO

You can actually encrypt your own serialized objects.

But the Ceph feature will cover entire storage systems, even if they have many servers.

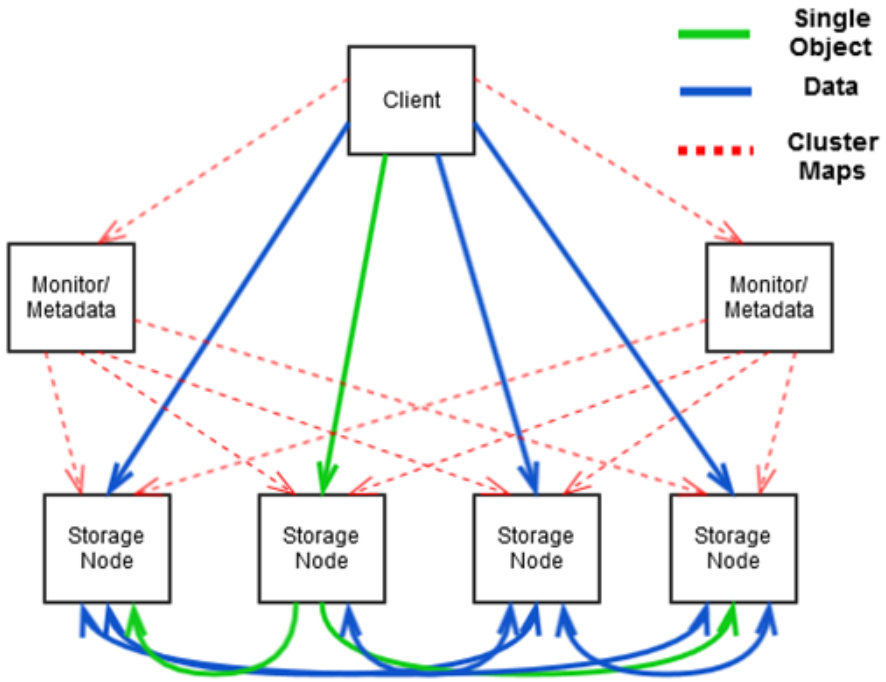
Thus if someone were to steal a storage drive, they just see random bits
(unless they are friends of the keymaker)

Keymaker in the Matrix films

CEPH "ARCHITECTURE"

We discussed complex systems built from multiple processes.

Ceph has this sort of structure! Each type of element is a separate process. New more storage? Just run more storage nodes!



EXAMPLES OF CEPH FEATURES

Efficient ways of managing directories with immense numbers of objects in them, such as fast "lookup" to find the file control information (the inode).

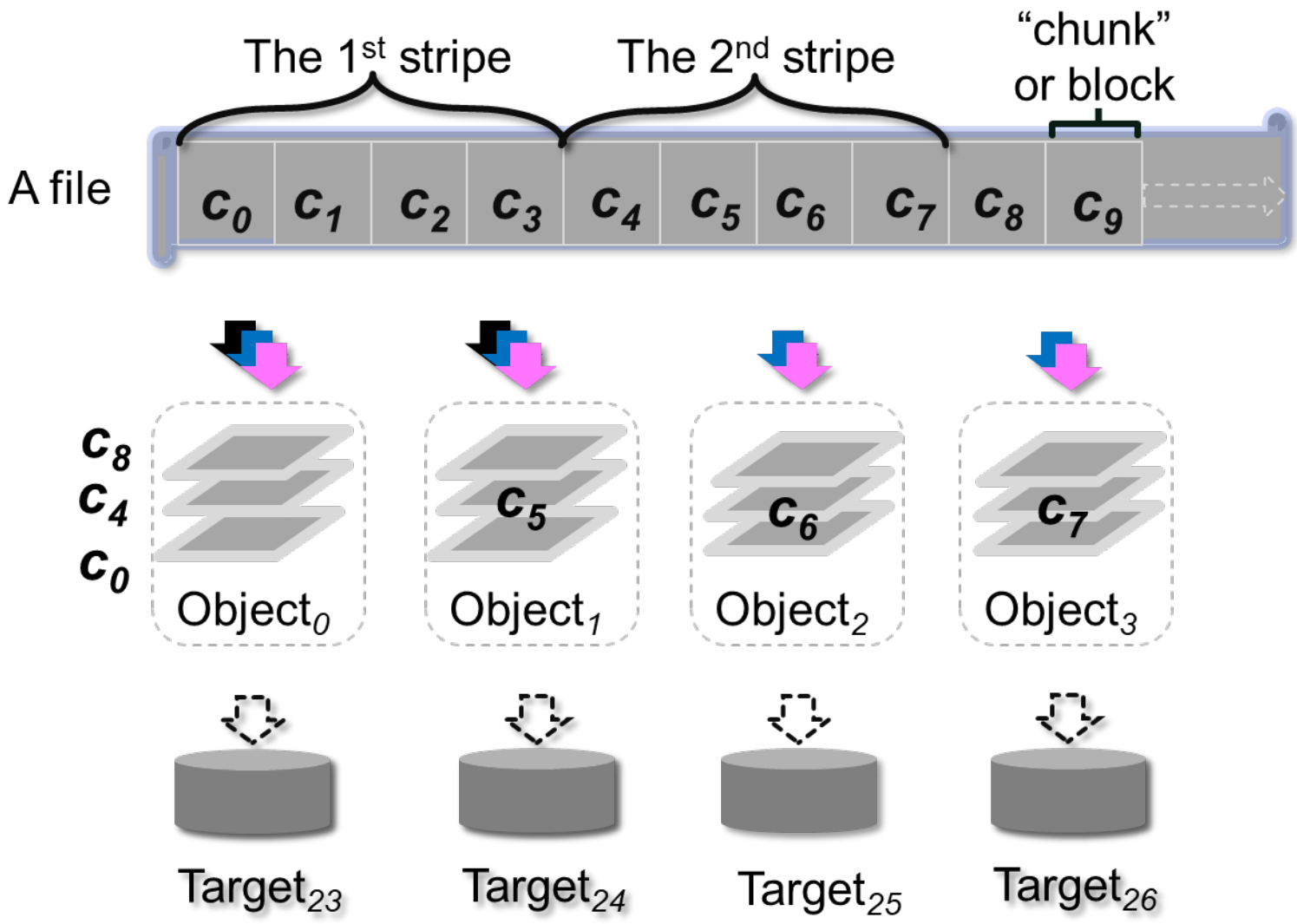
Special ways of packing data so that with large numbers of small objects, a whole block isn't wasted for each.

With really big files, "striping" over multiple storage units.

FILE STRIPING CONCEPT

Slice file into chunks and then write chunk by chunk across several storage units.

Here, each "stipe" has 4 chunks.



Graphic is by Jeff Chase @ Duke.

CEPH RADOS CONCEPT: RELIABLE ARRAY OF DISTRIBUTED OBJECT STORAGE DEVICES

Ceph has one set of servers that handle object "metadata"

This include file names, sizes, date of creation, lifetime, etc…

The "RADOS" layer is a set of servers that hold striped data. It is also used to hold the metdata blocks. Storage can be provided by any networked servers that support the RADOS API.

MORE CEPH FEATURES

Crashes are a worry, so Ceph automatically makes a backup copy (replica) for each object when it is first stored.

Replication is done inside RADOS. Each storage server has a *primary* and a *backup*. When data is stored to the primary, it is simultaneously copied to the backup. The backup can be accessed (read only) when if the primary is down.

MORE CEPH FEATURES

Ceph tries to avoid wasted I/O.

Object "lifetimes" can be short. Many are created, but then deleted within a few milliseconds!

- Ceph is very quick to make the replica but holds it in memory
- No disk I/O occurs unless the object hangs around for a while.

CEPH IS OPEN SOURCE!

… which doesn't always mean "free"

But in fact there *is* a free version of Ceph that you can download and use.

There is also a productized version with many features aimed at big professional users. Like CERN's giant particle accelerator!

A SECOND POPULAR NETWORKED FILE SYSTEM

We saw that failures cause problems for Linux locks.

Ceph replication doesn't address the core problem of a client not being able to tell whether a machine crashed or is healthy. In fact Ceph itself won't allow writes unless RADOS has an operational primary and a backup.

Zookeeper is a file system specifically targeting high availability.

APACHE ZooKeeper™

Flaviu Junqueria

Created by Flaviu Junqueria in the Apache project

Role is to hold data like “which machines are healthy?” or other kinds of configuration files. Free, open source.

HOW ZOOKEEPER IS USED

We often see systems that use Zookeeper plus Ceph plus normal Linux file storage on the local machine or even a datacenter FS.

Each kind of networked file system is really a specialist!

Zookeeper rarely has many files, never allows large files, and the files themselves tend to be kept for as long as the system runs

FAULT-TOLERANCE

APACHE ZooKeeper™

Zookeeper will "self heal" if it is damaged by a crash. It also automatically reports if any of your processes fails.

Includes a somewhat broken checkpointing option, for use if the whole system suddenly shuts down.

- Core issue: it can suffer amnesia about recent events if it crashes.
- We'll discuss this more in Lecture 21.

CLEVER SPECIAL FEATURE IN ZOOKEEPER

Normal Linux file systems (including Ceph) don't track file versions. You just create the file, write data, and perhaps later rewrite the file (or replace it).

With Zookeeper, files also have version numbers.

A process can request a file replace "if the version number is currently 21".

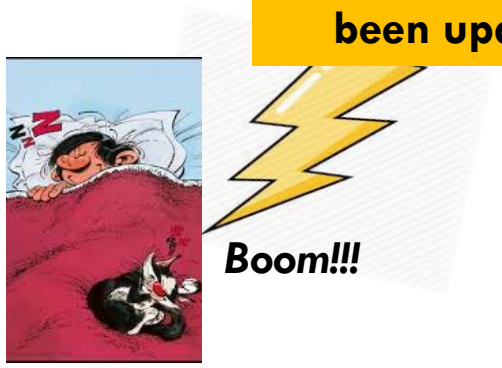
HOW TO USE THIS FEATURE.

When process A reads configuration file *conf*, A notes that the current version is 18.

A computes an update to the configuration and asks Zookeeper to replace *conf* if the version number is still 18.

- Fails if the version number is some larger value now. A lost a race!
- If it succeeds, A definitely creates version 19.

WATCH NOTIFICATIONS



With all of these approaches, a Linux program can also request a signal if a file, or directory, changes. You use **inotify.**

With Zookeeper, applications can learn that *conf* has been updated. They reread the configuration and adapt.

This allows a complex system to coordinate configuration updates.

NETWORKED FILE SYSTEM SUMMARY

With networking, you can easily access remote file storage.

This is valuable in many ways!

- In the cloud, the total capacity could be billions of times more than a local disk could hold.
- There are solutions specialized for storing serialized objects (Ceph)
- And even solutions specialized for tracking configurations (Zookeeper)

LEARN MORE ABOUT FAULT TOLERANCE?

This is a big topic and Cornell is famous for our work on it.

- Our current best solution is Derecho, an open-source C++ library
- It sets world records for data replication speed, and can use TCP or leverage modern networking hardware features that are even faster
- We are building a Ceph and Zookeeper API over Derecho.

With modern "high availability" services, the things you use are probably able to ride out failures. Your application, however, may need to restart.