


# Lecture on Parallel Filesystems

## Distributed Filesystems

Jacques-Charles Lafoucriere

ENSIEE| 2018



## What is a distributed system?

**From various textbooks:**

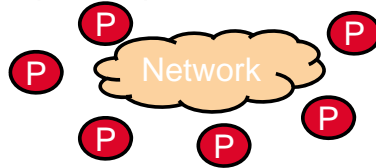
- “A distributed system is a collection of independent computers that appear to the users of the system as a single computer.”
- “A distributed system consists of a collection of autonomous computers linked to a computer network and equipped with distributed system software.”
- “A distributed system is a collection of processors that do not share memory or a clock.”
- “Distributed systems is a term used to define a wide range of computer systems from a weakly-coupled system such as wide area networks, to very strongly coupled systems such as multiprocessor systems.”

SFP 2018

Parallel Filesystems / Distributed and Parallel FS | PAGE 2

## What is a distributed system (cont.)?

- A set of physically separated processors, connected by one or more communication links



- Is any system with more than 2 computers a distributed system?
  - Internet
  - Network printer access
  - Backup
- We don't usually consider these to be distributed systems
  - Except if they respond to a single goal

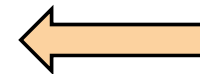
## Systems Classification

SISD: single instruction, single data

SIMD: single instruction, multiple data

MISD: multiple instruction, single data

MIMD: multiple instruction, multiple data



DE LA RECHERCHE À L'INDUSTRIE  
cea **Classification of MIMD Architectures**

```

graph TD
    PC[Parallel Computers] --> MP[Multiprocessors (shared memory)]
    PC --> MC[Multicomputers (distributed / private memory)]
  
```

- Tightly coupled: parallel processing
  - One OS shares proc, clocks and memory
- Loosely coupled: distributed computing
  - Multiple OS, each OS has his own proc, memory

SFP 2018 Parallel Filesystems / Distributed and Parallel FS | PAGE 5

DE LA RECHERCHE À L'INDUSTRIE  
cea **What's wrong with big single OS machines?**

### Symmetric multiprocessors (SMP)

- Multiple processors sharing memory
- Advantages
  - Single - processor applications can run without modifications
  - Inter-processor communication is fast (programs with fine-grain parallelism run fast)
- Problems
  - No easy fault-tolerance - one processor (or cache) fails, the whole system fails
  - Does not scale
  - Hard to provide uniform memory access

SFP 2018 Parallel Filesystems / Distributed and Parallel FS | PAGE 6



## Why (not) use distributed systems?

- Advantages
  - Price / performance
  - Higher performance
    - n processors potentially gives n times the computational power
  - Scalability
    - Modular structure makes it easier to add or replace processors and resources
  - Reliability
    - Replication of processors and resources yields fault tolerance
- Problems
  - Requires paradigm shift for applications programmer : message passing
  - Have to deal with (administer, upgrade, maintain) multiple machines

SFP 2018

Parallel Filesystems / Distributed and Parallel FS | PAGE 7



## Clusters

### A subclass of distributed systems

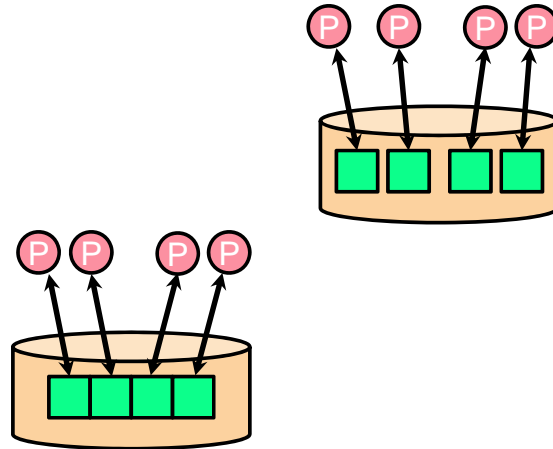
- Mostly homogeneous (the same hardware and OS)
- Array of computers used as cluster acts as a single machine
- Used for classic distributed services
  - Email, file services, ...
  - Distribution is used for load balancing and fault tolerance
- Used for HPC
  - Simulation, data analysis, ...
  - Parallelism: simultaneous coordinated works for a single result

SFP 2018

Parallel Filesystems / Distributed and Parallel FS | PAGE 8

## And Storage?

- Distributed Systems and Parallel Systems need to access data
- Distributed Systems used Distributed FS
  - Loosely coupled
  - Independent access
- Parallel Systems used Parallel FS
  - Highly coupled
  - Simultaneous coordinated access



SFP 2018

Parallel Filesystems / Distributed and Parallel FS | PAGE 9

## Distributed Systems Characteristics

### Failure

- Any component may/will fail
- Communications are unreliable
- But a distributed system is tolerant by design

### Performance

- High potential but easy to break

### Security

- How to be confident in remote sites?

SFP 2018

Parallel Filesystems / Distributed and Parallel FS | PAGE 10



## Communications

### Fundamentally unreliable

- Packet loss
- Packet corruption
- Packet drop

### Solutions

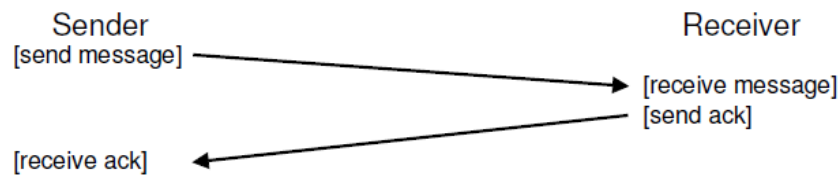
- Checksums for integrity
  - Cyclic Redundancy Codes
- Reliable protocol
  - Acknowledge to handle packet loss
  - Sequence counter to guarantee packet is received exactly once

SFP 2018

Parallel Filesystems / Distributed and Parallel FS | PAGE 11

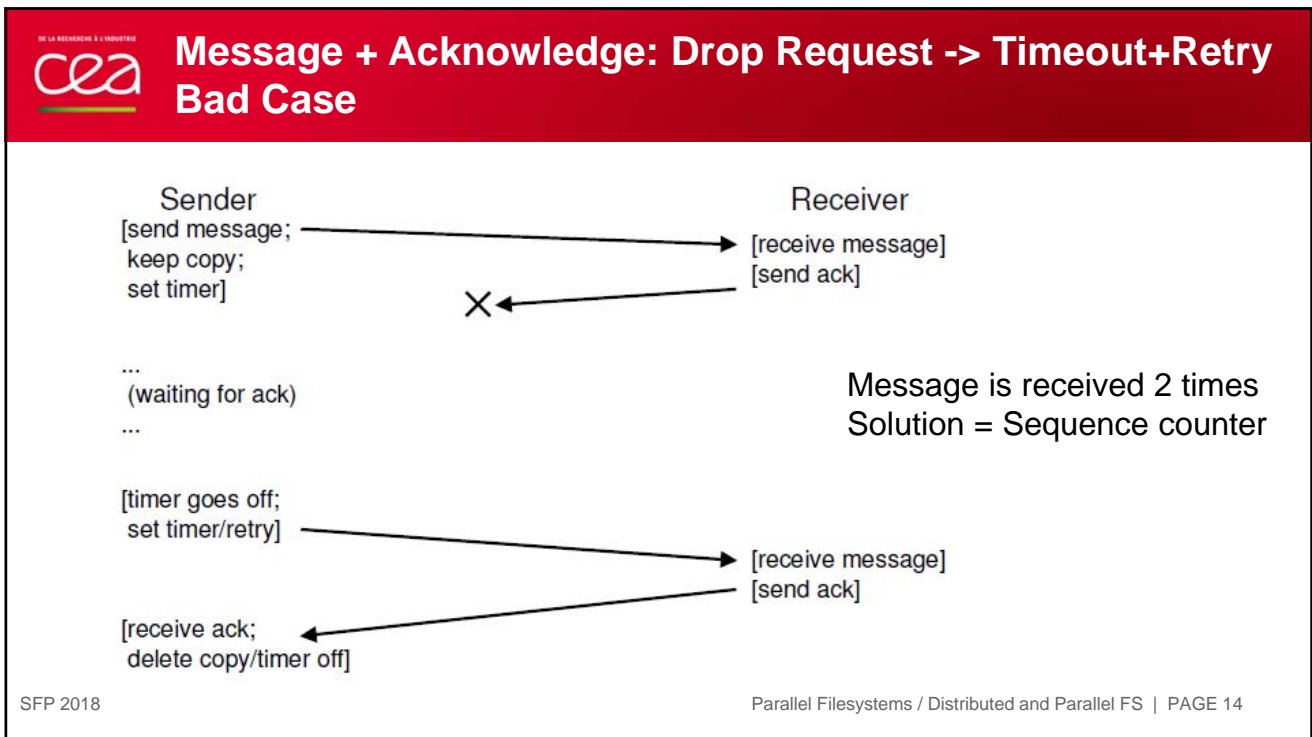
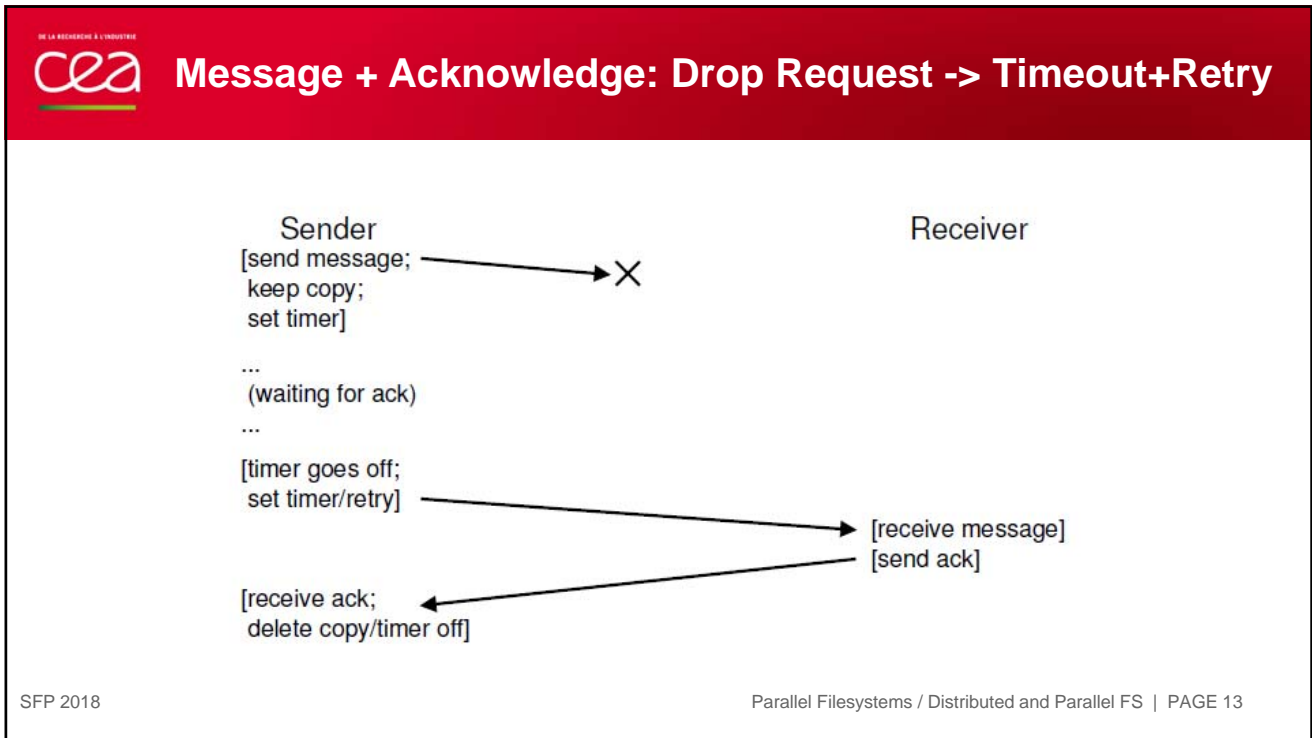


## Message + Acknowledge: Good case



SFP 2018

Parallel Filesystems / Distributed and Parallel FS | PAGE 12





## Remote Procedure Call (RPC)

### Objective

- Make the process of executing code on a remote machine as simple and straightforward as calling a local function

### Principle

- The server defines some routines that it wishes to export
- A stub generator
  - Pack function arguments and results into messages
  - Hide network layers to programmer
  - Handle byte ordering
  - Generates client code and server code
- A run-time library
  - Implement server naming

SFP 2018

Parallel Filesystems / Distributed and Parallel FS | PAGE 15



## Remote Procedure Call (RPC)

### Client

- Create a message buffer
- Pack the needed information into the message buffer
- Send the message to the destination RPC server
- Wait for the reply
- Unpack return code and other arguments
- Return to the caller

### Server

- Unpack the message
- Call into the actual function
- Package the results
- Send the reply
- Return to the caller

SFP 2018

Parallel Filesystems / Distributed and Parallel FS | PAGE 16





DE LA RECHERCHE À L'INDUSTRIE  
**cea** **SUN Network File System**

**Origin: Eighties**

- Many clients
- Few servers
  - Store data on local storage
- Clients request data through well-formed protocol messages

```
graph LR; C0[Client 0] --- N[Network]; C1[Client 1] --- N; C2[Client 2] --- N; C3[Client 3] --- N; N --- S[Server]; S --- DS[Data Storage];
```

SFP 2018 Parallel Filesystems / Distributed and Parallel FS | PAGE 18



## NFS (cont.)

### Advantages

- Easy data sharing across clients
- Centralized administration
- Security
  - Server can be in a locked rooms
  - Simple to backup/restore

SFP 2018

Parallel Filesystems / Distributed and Parallel FS | PAGE 19



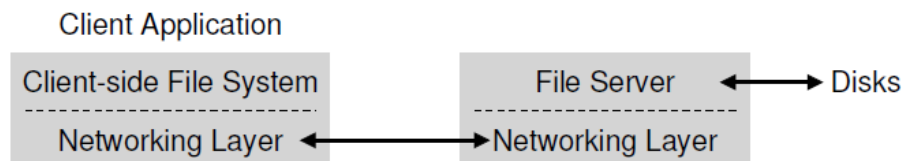
## NFS: Software Architecture

### Client side

- Client application access files and directories
- Client call kernel system calls (open(), read(), write(), mkdir(), readdir(), ...)
- Kernel FS client send RPC

### Server side

- Server program received RPC
- Server do requested function on local disks and reply



SFP 2018

Parallel Filesystems / Distributed and Parallel FS | PAGE 20

DE LA RECHERCHE À L'INDUSTRIE

## NFS V2: Simple Stateless protocol

<span style="color: green;">■</span>	NFSPROC_GETATTR	file handle	attributes
<span style="color: green;">■</span>	NFSPROC_SETATTR	file handle, attributes	nothing
<span style="color: green;">■</span>	NFSPROC_LOOKUP	directory file handle, name of file/directory to look up	file handle
<span style="color: green;">■</span>	NFSPROC_READ	file handle, offset, count	data, attributes
<span style="color: green;">■</span>	NFSPROC_WRITE	file handle, offset, count, data	attributes
<span style="color: green;">■</span>	NFSPROC_CREATE	directory file handle, name of file, attributes	nothing
<span style="color: green;">■</span>	NFSPROC_REMOVE	directory file handle, name of file to be removed	nothing
<span style="color: green;">■</span>	NFSPROC_MKDIR	directory file handle, name of directory, attributes	file handle
<span style="color: green;">■</span>	NFSPROC_RMDIR	directory file handle, name of directory to be removed	nothing
<span style="color: green;">■</span>	NFSPROC_READDIR	directory handle, count of bytes to read, cookie	directory entries, cookie (to get more entries)

SFP 2018
Parallel Filesystems / Distributed and Parallel FS | PAGE 21

DE LA RECHERCHE À L'INDUSTRIE

## NFS: Reading a File

Client	Server
<pre>fd = open("foo", ...); Send LOOKUP (rootdir FH, "foo")</pre>	<pre>Receive LOOKUP request look for "foo" in root dir return foo's FH + attributes</pre>
<pre>Receive LOOKUP reply allocate file desc in open file table store foo's FH in table store current file position (0) return file descriptor to application</pre>	
<pre>read(fd, buffer, MAX); Index into open file table with fd get NFS file handle (FH) use current file position as offset Send READ (FH, offset=0, count=MAX)</pre>	<pre>Receive READ request use FH to get volume/inode num read inode from disk (or cache) compute block location (using offset) read data from disk (or cache) return data to client</pre>
<pre>Receive READ reply update file position (+bytes read) set current file position = MAX return data/error code to app</pre>	
<pre>read(fd, buffer, MAX); Same except offset=MAX and set current file position = 2*MAX</pre>	
<pre>read(fd, buffer, MAX); Same except offset=2*MAX and set current file position = 3*MAX</pre>	
<pre>close(fd); Just need to clean up local structures Free descriptor "fd" in open file table (No need to talk to server)</pre>	

SFP 2018
Parallel Filesystems / Distributed and Parallel FS | PAGE 22



## NFS Failure support

### Protocol is stateless

- Server does not need to memorize client status

### Most functions are idempotent

- The effect of performing the operation multiple times is equivalent to the effect of performing the operation a single time
  - Ok: Read(), Write(), Stat()
  - Ko: mkdir()

SFP 2018

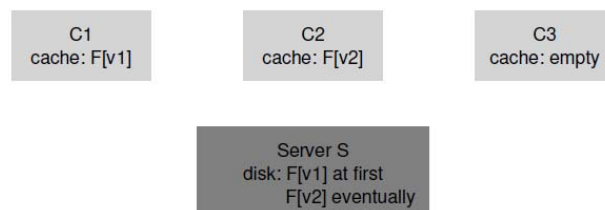
Parallel Filesystems / Distributed and Parallel FS | PAGE 23



## NFS Performances

### How to improve performances?

- Client side caching
  - Write buffering
  - Read caching
- Cache consistency problem: update visibility and stale cache
  - Flush on close
  - File change test



SFP 2018

Parallel Filesystems / Distributed and Parallel FS | PAGE 24



## NFS: Concurrency issue

### Writes from multiple clients in single file

- C1 writes B1/B2/B3
- C2 writes BA/BB/BC
- Protocol does no guarantee write ordering between clients
- Result can be
  - B1 or BA
  - B2 or BB
  - B3 or BC
- Worth in case of read/modify/writes

### NFS is not a parallel FS

SFP 2018

Parallel Filesystems / Distributed and Parallel FS | PAGE 25



## NFS Today

### Previous description is for V2

### New versions

- V3
  - Improve performances with readdirplus()
  - Add kerberos support
- V4
  - Improve windows compatibility
  - Add quotas support
  - Add mechanism to improve client caching
  - Add support of pNFS

SFP 2018

Parallel Filesystems / Distributed and Parallel FS | PAGE 26

## Parallel Filesystems



## Parallel File Systems

### Store application data persistently

- Usually extremely large datasets that can't fit in memory

### Provide global shared namespace (files, directories)

### Designed for parallelism

- Concurrent (often coordinated) access from many clients

### Designed for high-performance

- Operate over high-speed networks
- Optimized I/O path for maximum bandwidth



## Parallel vs Distributed

### Distributed FS is designed for throughput

- Multiple independent access to separate data sets

### Parallel FS is designed for performance

- Multiple simultaneous access to a single data set
- Required a strong coherency
  - Solved by using a Distributed Lock Manager
  - Client request lock on data sets before access

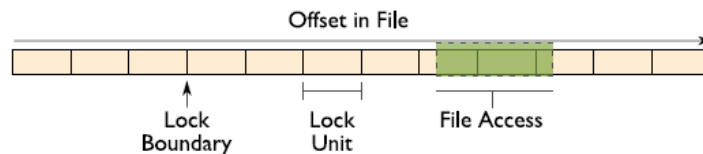
SFP 2018

Parallel Filesystems / Distributed and Parallel FS | PAGE 29



## Locking in Parallel File Systems

- Files are broken up into lock units
- Clients obtain locks on units before access
  - R vs W (or Shared vs Exclusive)
- Enable client caching
- Locks are reclaimed from client when another client desire access
  - Client flush cache before giving back lock
- Locks are delegated and revoked through DLM



SFP 2018

Parallel Filesystems / Distributed and Parallel FS | PAGE 30

DE LA RECHERCHE À L'INDUSTRIE  
**cea** Next?

# Lustre File System

SFP 2018 Parallel Filesystems / Distributed and Parallel FS | PAGE 31

# Thank you for your attention

ENSIIE | 2018

Commissariat à l'énergie atomique et aux énergies alternatives  
Centre DAM-Ile de France | 91297 Bruyères-le-Châtel Cedex  
T. +33 (0)1 69 26 40 00 | F. +33 (0)1 69 26 70 86

Direction des applications militaires  
Département sciences de la simulation et de l'information  
Service informatique scientifique et réseaux

Etablissement public à caractère industriel et commercial | RCS Paris B 775 685 019