

Distributed File Systems: AFS Jan 25, 2021





- Why Distributed File Systems?
- Basic mechanisms for building DFSs
 - Using NFS and AFS as examples
- Design choices and their implications
 - Caching
 - Consistency
 - Naming
 - Authentication and Access Control

Client Caching in NFS v2



- Cache both clean and dirty file data and file attributes
 - Memory cache
 - Sub-file caching granularity
- File attributes (e.g., last modified time) in the cache expire after 60 seconds (file data doesn't expire)

Will retrieve updated attributes from server every 60s

- If server has a more recent modified time, grab the up-to-date data in cache from server
- Dirty data are buffered (in cache) on the client until file close or up to 30 seconds
 - If the machine crashes before then, the changes are lost

Looking back at the campus-wide use-case



- NFS gets us partway there, but
 - Probably doesn't handle scale (* you can buy huge NFS appliances today that will, but they're \$\$\$).
 - Is very sensitive to network latency
 - Consistency is.. what do we even call that? Highly implementation specific.
- How can we improve this?
 - More aggressive caching (AFS caches on disk in addition to just in memory)
 - Prefetching (on open, AFS gets entire file from server, making later ops local & fast).

Client Caching in AFS



- Callbacks! Clients register with server that they have a copy of file;
 - Server tells them (calls them back): "Invalidate" if the file changed (but only does so on file close!)
 - This trades state (at server) for improved consistency
- Key AFS bit: read from local disk copy unless server indicates new copy exists (via callback)
- What if server crashes? Lose all callback state!
 - Reconstruct callback information from clients
 - ask everyone "who has which files cached?"

AFS v2 RPC Procedures



- Procedures that are not in NFS
 - Fetch: from client to server, return status and optionally data of (entire) file/dir to client + add callback on it
 - RemoveCallBack: from C to S, specify a file that the client has flushed from the local machine
 - **BreakCallBack**: from S to C, revoke the callback on a file or directory (this is the callback **call** to client)

• **Store**: from C to S, store the status and optionally data of a file on the server

AFS v2 RPC Procedures



- Procedures that are not in NFS
 - Fetch: from client to server, return status and optionally data of (entire) file/dir to client + add callback on it
 - RemoveCallBack: from C to S, specify a file that the client has flushed from the local machine
 - BreakCallBack: from S to C, revoke the callback on a file or directory (this is the callback call to client)
 - What should the client do if a callback is revoked?
 - **Store**: from C to S, store the status and optionally data of a file on the server

AFS v2 RPC Procedures



- Procedures that are not in NFS
 - Fetch: from client to server, return status and optionally data of (entire) file/dir to client + add callback on it
 - RemoveCallBack: from C to S, specify a file that the client has flushed from the local machine
 - BreakCallBack: from S to C, revoke the callback on a file or directory (this is the callback call to client)
 - What should the client do if a callback is revoked?
 - Delete existing cached copy / refetch from server on open
 - **Store**: from C to S, store the status and optionally data of a file on the server





- Why Distributed File Systems?
- Basic mechanisms for building DFSs
 - Using NFS and AFS as examples
- Design choices and their implications
 - Caching
 - Consistency
 - Naming
 - Authentication and Access Control

Topic 2: File Access Consistency



- In UNIX local file system, concurrent file reads and writes have "sequential" consistency semantics
 - Each file read/write from user-level app is an atomic operation
 - The kernel locks the file vnode
 - Each file write is immediately visible to all file readers
- Neither NFS nor AFS provides such concurrency control between distributed processes
 - NFS: "sometime within 30 seconds"
 - AFS: session semantics consistency (next slide)
 - Same machine processes in AFS do have seq. consistency



What it means:

- A file write is visible to processes on the same box immediately, but not visible to processes on other machines until the file is closed
- When a file is closed, changes are visible to new opens, but are not visible to "old" opens
 - Last closer wins!
 - AFS writebacks the *entire* file (not a mix of updates like NFS)
- All other file operations are visible everywhere immediately
- Implementation
 - Dirty data are buffered at the client machine until file close, then flushed back to server, which leads the server to send "break callback" to other clients



		Client ₁		Client ₂
-	\mathbf{P}_1	\mathbf{P}_2	Cache	\mathbf{P}_3 (
	open(F)		-	
	write(A))	A	
	close()		A	
local to		open(F)	A	
		$read() \rightarrow A$	A	
		close()	A	
Client	open(F)		A P	
	write(D)	onon(E)	D B	
		read() \rightarrow B	B	
		close()	B	
			B	open(F)
			В	read() \rightarrow A
Clients 1.2			В	close()
	close()		В	
concurrent			В	open(F)
CONCUTENT			В	$read() \rightarrow B$
			В	close()
-			В	open(F)
-	open(F)		В	

write(D)

close()

-	А	
-	А	
-	А	
-	А	
-	А	
-	А	
-	А	Local processes
-	А	see writes immediately
-	А	2
А	А	Remote processes
А	А	do not see writes
А	А	
Å	В	until close()
̈́Β	В	has taken place
В	В	1
В	В	
В	В	
В	В	
В	В	
С	В	
С	C	
¢	D	
Ď	D	Unfortunately for P_3
D	D	the last writer wins
D	D	
_		

Comments

File created

Server

Disk

_

Cache

write(C)

close()

open(F)

close()

read() \rightarrow D

D

D

D

D

D

D D



	P ₁	Client ₁ P ₂	Cache	Client ₂	Cache	Server Disk	Comments
	$\frac{1}{\text{open}(F)}$	• 2	-	13	-	- DISK	File created
P1 and P2	write(A)	А		-	-	
	close()	/	A		-	А	
local to		open(F)	A		-	А	
		$read() \rightarrow A$	A		-	А	
		close()	A		-	А	
Client1	open(F)		A		-	A	
Olicitti	write(B))	В		-	A	
		open(F)	В		-	A	Local processes
		$read() \rightarrow B$	В		-	A	see writes immediately
		close()	В		-	A	
			В	open(F)	A	A	Remote processes
Olianta 1.0			В	$read() \rightarrow A$	A	A	do not see writes
Clients 1.2			В	close()	A	A	
,	close()		В		A	В	until close()
concurrent			В	open(F)	̈́Β	В	has taken place
CONCUMENT			В	$read() \rightarrow B$	В	В	_
			В	close()	В	В	
			В	open(F)	В	В	
	open(F)		В		В	В	
	write(D)	D		В	В	
			D	write(C)	C	В	
			D	close()	C	C	
	close()		D		¢	D	
			D	open(F)	Ď	D	Unfortunately for P_3
			D	$read() \rightarrow D$	D	D	the last writer wins
			D	close()	D	D	



	Р.	Client ₁ \mathbf{P}_{2}	Cache	Client ₂	ache	Server Disk	Comments
	$\frac{1}{\text{open}(F)}$	12	-	13	-	-	File created
P P1 and P2	write(A)	А		-	-	
	close()	, ,	А		-	А	
local to		open(F)	A		-	A	
		$read() \rightarrow A$	A		-	A	
		close()	A		-	A	
Client1	open(F)		А		-	A	
Onoriti	write(B)		В		-	A	
		open(F)	B		-	A	Local processes
		read() \rightarrow B	В		-	A	see writes immediately
		close()	В	$(\mathbf{\Gamma})$	-	A	Denselation
			B	open(F)	A	A	Remote processes
Cliente 1 2			D B	read() \rightarrow A		A	do not see writes
	1 ()			close()			
	close()		В	$(\mathbf{\Gamma})$	A	В	until close()
concurrent			D D	open(F)	D D	D D	has taken place
oonourion			D	reau() \rightarrow D	D D	D D	
			B	close()	D B	B	
	open(F)		B	open(r)	B	B	
	write(D)	D		B	B	
	Wille(D	/	D	write(C)	Č	B	
			D	close()	Č	Ċ	
	close()		D	~	¢	D	
	e100e()		D	open(F)	Ď	D	Unfortunately for P ₂
			D	read() \rightarrow D	D	D	the last writer wins
			D	close()	D	D	



		Client ₁		$Client_2$		Server	Comments
	\mathbf{P}_1	\mathbf{P}_2	Cache	\mathbf{P}_3	Cache	Disk	
	open(F)		-		-	-	File created
P I and PZ	write(A)	А		-	-	
	close()		A		-	А	
local to		open(F)	A		-	A	
		read() $\rightarrow A$	A		-	A	
	$(\mathbf{\Gamma})$	close()	A		-	A	
Client	open(F)	,	A P		-	A A	
•	write(D)	(\mathbf{E})	D B		-	A A	I agal processos
		$read() \rightarrow B$	B		-	A A	see writes immediately
		close()	B		-	A	see writes infinediately
		clobe()	B	open(F)	А	A	Remote processes
			B	read() \rightarrow A	A	A	do not see writes
Clients 1.2			В	close()	А	А	
	close()		В	~	Á	В	until close()
aanaurrant			В	open(F)	B	B	has taken place
concurrent			В	read() \rightarrow B	В	В	1
			В	close()	В	В	
			В	open(F)	В	В	
	open(F)		В		В	В	
	write(D)	D		В	В	
			D	write(C)	C	В	
			D	close()	C	С	
	close()		D		¢	D	
			D	open(F)	D	D	Unfortunately for P_3
			D	read() \rightarrow D	D	D	the last writer wins
			D	close()	D	D	



	Client ₁			Client ₂		Server	Comments
	\mathbf{P}_1	\mathbf{P}_2	Cache	\mathbf{P}_3	Cache	Disk	
D1 and D2	open(F)		-		-	-	File created
	write(A)	A		-	-	
	close()		A		-	A	
local to		open(F)	A		-	A	
		read() $\rightarrow A$	A		-	A	
	opon(F)	close()	A A		-	A A	
Client	write(B))	B		_		
	write(D)	open(F)	B		-	A	Local processes
		read() \rightarrow B	B		-	A	see writes immediately
		close()	В		-	Α	5
			В	open(F)	А	A	Remote processes
Cliente 1 0			В	$read() \rightarrow A$	A	A	do not see writes
• Clients I.2			В	close()	A	A	
,	close()		В		A	В	until close()
concurrent			В	open(F)	В	В	has taken place
Concurrent			B	read() \rightarrow B	B	B	
			В	close()	В	В	
	onon(E)		D D	open(F)	D D	D D	
	write(D)			B	B	
	write(D)	D	write(C)	C	B	
			D	close()	Č	Č	
	close()		D	~	¢	D	
			D	open(F)	Ď	D	Unfortunately for P ₃
			D	read() \rightarrow D	D	D	the last writer wins
			D	close()	D	D	



P1 and P2 local to Client1

Clients 1,2
concurrent

	Client ₁		$Client_2$		Server	Comments
\mathbf{P}_1	\mathbf{P}_2	Cache	\mathbf{P}_3	Cache	Disk	
open(F)		-		-	-	File created
write(A)	А		-	-	
close()		А		-	А	
	open(F)	А		-	А	
	$read() \rightarrow A$	A		-	А	
	close()	А		-	А	
open(F)		А		-	А	
write(B)		В		-	А	
	open(F)	В		-	A	Local processes
	$read() \rightarrow B$	В		-	А	see writes immediately
	close()	В	()	-	A	
		В	open(F)	A	A	Remote processes
		В	$read() \rightarrow A$	A	A	do not see writes
		В	close()	A	А	
close()		В		Ă	В	until close()
		В	open(F)	B	В	has taken place
		В	$read() \rightarrow B$	В	В	
		В	close()	В	В	
		В	open(F)	В	В	
open(F)		В		В	В	
write(D)	D		В	В	
		D	write(C)	С	В	
		D	close()	C	С	
close()		D		¢	D	
		D	open(F)	Ď	D	Unfortunately for P_3
		D	$read() \rightarrow D$	D	D	the last writer wins
		D	close()	D	D	



 \sim

			$Client_1$		Client ₂		Server	Comments
		\mathbf{P}_1	\mathbf{P}_2	Cache	\mathbf{P}_3	Cache	Disk	
_		open(F)		-		-	-	File created
•	P I and PZ	write(A)	A		-	-	
		close()	<i>(</i>)	A		-	А	
	local to		open(F)	A		-	A	
			read() \rightarrow A	A		-	A	
		onon(E)	close()	A		-	A A	
	Client	write(B)	1	B		-	A Δ	
		wine(D)	open(F)	B		-	A	Local processes
			read() \rightarrow B	B		-	A	see writes immediately
			close()	В		-	А	5
				В	open(F)	А	А	Remote processes
	O			В	$read() \rightarrow A$	А	А	do not see writes
•	Clients 1.2			В	close()	A	А	
	,	close()		В		A	В	until close()
	concurrent			В	open(F)	В	В	has taken place
	Concurrent			B	read() \rightarrow B	B	B	
				В	close()	В	В	
		opon(E)		D B	open(F)	D	D B	
		write(D))	D		B	B	
		WIIIC(D)	D	write(C)	C	B	
				D	close()	Č	Č	
		close()		D	~	¢	D	
		01000()		D	open(F)	Ď	D	Unfortunately for P ₃
				D	read() \rightarrow D	D	D	the last writer wins
				D	close()	D	D	



	P ₁	Client ₁	Cache	Client ₂	ache	Server Disk	Comments
_ /	$\frac{1}{\text{open}(F)}$	12	Cacilie	13	-	DISK	File created
P1 and P2	write(A)		Δ		_	_	The created
	close()		A		-	А	
	clobe()	open(F)	A		-	A	
local to	1	$read() \rightarrow A$	A		-	A	
	(close()	A		-	A	
Cliont1	open(F)		А		-	А	
Chefiti	write(B)		В		-	А	
		open(F)	В		-	А	Local processes
	1	$read() \rightarrow B$	В		-	А	see writes immediately
		close()	В		-	А	
			В	open(F)	А	А	Remote processes
			В	$read() \rightarrow A$	А	А	do not see writes
Clients 1.2			В	close()	А	A	
•······,_	close()		В		A	В	until close()
aanaurrant			В	open(F)	΄B	В	has taken place
concurrent			В	read() \rightarrow B	В	В	-
			В	close()	В	В	
			В	open(F)	В	В	
	open(F)		В		В	В	
	write(D)		D		В	В	
			D	write(C)	С	В	
			D	close()	C	C	
	close()		D		¢	D	
			D	open(F)	Ď	D	Unfortunately for P_3
			D	$read() \rightarrow D$	D	D	the last writer wins
			D	close()	D		



	Client ₁			Client ₂	.	Server	Comments
	\mathbf{P}_1	\mathbf{P}_2	Cache	\mathbf{P}_3	Cache	Disk	
	open(F)		-		-	-	File created
PT and PZ	write(A)	A		-	-	
	close()		A		-	А	
local to		open(F)	A		-	А	
iucai lu		$read() \rightarrow A$	A		-	А	
	()	close()	A		-	A	
Client1	open(F)		A		-	A	
	write(B)	B		-	A	~ .
		open(F)	B		-	A	Local processes
		read() \rightarrow B	В		-	A	see writes immediately
		close()	B	(F)	-	A	Demoster
			B	open(F)	A	A	Remote processes
Cliante 1 2			D D	read() \rightarrow A	A	A	do not see writes
	1 0		D	Close()	A	A	
	close()		В		A	В	until close()
concurrent			В	open(F)	В	В	has taken place
Concurrent			B	read() \rightarrow B	D D	D D	
			D D	close()	D D	D D	
	$\operatorname{opon}(\mathbf{F})$		D B	open(r)	D B	D B	
	write(D)			B	B	
	wille(D)		$\operatorname{write}(C)$	C	B	
				close()	C	D C	
	alaca				đ		
	close()			onon(E)			Unfortunately for D
				read() D			the last writer wine
			D	$close() \rightarrow D$	D	D	the last willer wills



		Client ₁		Client ₂		Server	Comments
	\mathbf{P}_1	\mathbf{P}_2	Cache	\mathbf{P}_3 (Cache	Disk	
D1 and D2	open(F)		-		-	-	File created
P I anu PZ	write(A)	A		-	-	
• • ·	close()	$(\mathbf{\Gamma})$	A		-	A	
local to		open(F)	A		-	A	
		$reau() \rightarrow P$			-	A A	
Client1	open(F)	close()			-		
Clienti	write(B))	B		_	A	
	WIIIC(D)	open(F)	B		-	A	Local processes
		read() \rightarrow B	В		-	A	see writes immediately
		close()	В		-	А	, ,
			В	open(F)	А	А	Remote processes
O			В	$read() \rightarrow A$	А	А	do not see writes
Clients 1.2			В	close()	A	А	
	close()		В		Á	В	until close()
concurrent			В	open(F)	В	В	has taken place
CONCUMENT			В	$read() \rightarrow B$	В	В	
			B	close()	В	B	
	$(\mathbf{\Gamma})$		В	open(F)	В	В	
	open(F))	D		D B	D	
	write(D)		$\operatorname{write}(C)$	D C	B	
			D	close()	C	C D	
	close()		D		đ		
	close()			open(F)	у D		Unfortunately for Pa
			D	$read() \rightarrow D$	D		the last writer wins
			D	close()	D	D	



		Client ₁		Client ₂		Server	Comments
	\mathbf{P}_1	\mathbf{P}_2	Cache	\mathbf{P}_3	Cache	Disk	
	open(F)		-		-	-	File created
PT and PZ	write(A	.)	A		-	-	
	close()		A		-	А	
local to		open(F)	A		-	А	
iucai lu		$read() \rightarrow A$	A		-	A	
		close()	A		-	A	
Client1	open(F)		A		-	A	
	write(B)	В		-	A	T 1
		open(F)	В		-	A	Local processes
		read() \rightarrow B	D D		-	A	see writes immediately
		close()	D D	onon(E)	-	A	Pomoto processos
			D B	$read() \land \Lambda$	A	A A	do not soo writes
Cliente 1 2			B	$reau() \rightarrow R$		A A	do not see writes
	-1()			close()	$\overline{\Lambda}$	Л	
	close()		D	onon(E)	A D	D	until close()
concurrent			D B	$road() \setminus B$	D R	D B	has taken place
oonoanone			B	$reau() \rightarrow D$	B	B	
			B	open(F)	B	B	
	open(F)		B	open(i)	B	B	
	write(D))	D		B	B	
	(_	/	D	write(C)	Ċ	В	
			D	close()	C	С	
	close()		D	^v	¢	D	
	00000		D	open(F)	Ď	D	Unfortunately for P ₂
			D	$read() \rightarrow D$	D	D	the last writer wins
			D	close()	D	D	



		Client ₁		Client ₂	.	Server	Comments
	\mathbf{P}_1	\mathbf{P}_2	Cache	\mathbf{P}_3	Cache	Disk	
	open(F)		-		-	-	File created
P I and PZ	write(A	.)	A		-	-	
	close()		A		-	А	
local to		open(F)	A		-	A	
100a110		$read() \rightarrow A$	A		-	A	
		close()	A		-	A	
Client1	open(F)		A		-	A	
	write(B) 	В		-	A	T 1
		open(F)	D D		-	A	Local processes
		reau() \rightarrow D	D		-	A	see writes inifiediately
		close()	B	opon(F)	_	A A	Romoto processos
			B	road() \ \			do pot soo writes
Cliphte 1 2			B	$close() \rightarrow R$		A	do not see writes
	alaca()		P	e105e()	Å	D	until close()
1	close()		B	opon(F)	^{/1} B	D B	has taken place
concurrent			B	$read() \rightarrow B$	B	B	has taken place
			B	close()	B	B	
			B	open(F)	B	B	
	open(F)	1	В	- F - ()	В	В	
	write(D)	D		В	В	
	,	,	D	write(C)	C	В	
			D	close()	C	С	
	close()		D		¢	D	
	()		D	open(F)	Ď	D	Unfortunately for P_3
			D	$read() \rightarrow D$	D	D	the last writer wins
			D	close()	D	D	



		Client ₁		Client	2	Server	Comments
	\mathbf{P}_1	\mathbf{P}_2	Cache	\mathbf{P}_3	Cache	Disk	
D1 and D2	open(F)		-		-	-	File created
P I and PZ	write(A	.)	A		-	-	
	close()		A		-	A	
local to		open(F)	A		-	A	
		read() $\rightarrow A$	A		-	A	
	opon(E)	close()	A		-	A A	
Client	write(B		B		-	A	
	WIIIC(D	open(F)	B		-	A	Local processes
		read() \rightarrow B	B		-	A	see writes immediately
		close()	В		-	А	5
		0	В	open(F)	А	А	Remote processes
O			В	$read() \rightarrow A$	A	А	do not see writes
Clients 1.2			В	close()	A	А	
	close()		В		Á	В	until close()
concurrent			В	open(F)	̈́Β	В	has taken place
CONCUMENT			В	read() \rightarrow B	В	В	
			В	close()	В	В	
			В	open(F)	В	В	
	open(F)		В		D D	D D	
	write(D)		$\operatorname{Autrito}(C)$	D C	D B	
			D	close()	C	D C	
	close()		D	0.000()	đ	n	
	C105C()		D	open(F)			Unfortunately for Pa
			D	$read() \rightarrow D$	D	D	the last writer wins
			D	close()	D	D	



		Client ₁		Client ₂		Server	Comments
	\mathbf{P}_1	\mathbf{P}_2	Cache	\mathbf{P}_3 (Cache	Disk	
	open(F)		-		-	-	File created
P I and PZ	write(A)	A		-	-	
	close()		A		-	A	
local to		open(F)	A		-	A	
		$read() \rightarrow A$	A		-	A	
	(\mathbf{T})	close()	A		-	A	
Client	open(F)	N N	A P		-	A	
	write(D	(\mathbf{E})	D		-	A A	Local processos
		$read() \rightarrow B$	B		-	A A	see writes immediately
		$close() \rightarrow D$	B		_	A	see writes ininectately
		close()	B	open(F)	А	A	Remote processes
			B	read() \rightarrow A	A	A	do not see writes
Clients 1.2			В	close()	А	А	
	close()		В	~	Á	В	until close()
a a la a u vira a t	00000()		B	open(F)	B	B	has taken place
concurrent			В	read() \rightarrow B	В	В	I
			В	close()	В	В	
			В	open(F)	В	В	
	open(F)		В		В	В	
	write(D)	D		B	В	
			D	write(C)	C	B	
			D	close()	C	C	
	close()		D	()	¢	D	
			D	open(F)	D	D	Unfortunately for P_3
			D	read() \rightarrow D	D	D	the last writer wins
			D	close()	D	D	



		Client ₁		Client	2	Server	Comments
	\mathbf{P}_1	\mathbf{P}_2	Cache	\mathbf{P}_3	Cache	Disk	
	open(F)		-		-	-	File created
	write(A)	A		-	-	
	close()		A		-	A	
ocal to		open(F)	A		-	A	
		read() $\rightarrow A$	A		-	A	
	$amam(\Gamma)$	close()	A		-	A	
	open(F))	A B		-		
	wille(D)	open(F)	B		_		I ocal processes
		$read() \rightarrow B$	B		-	A	see writes immediately
		close()	B		_	A	see writes miniculately
			В	open(F)	А	A	Remote processes
			В	$read() \rightarrow A$	A	A	do not see writes
Clients 1.2			В	close()	A	A	
	close()		В		A	В	until close()
oonourront	0		В	open(F)	̈́Β	В	has taken place
Soncurrent			В	read() \rightarrow B	B	В	-
			В	close()	В	В	
			В	open(F)	В	В	
	open(F)	、 、	B		B	B	
	write(D)		-	B	В	
				write(C)	C	D	
	1()			close()	đ		
	close()			$amam(\Gamma)$			L'afortunatoly for D
				$read() = \sum_{i=1}^{n}$	ם נ		the last writer wins
			D	close()			une last wither wills



		Client ₁		$Client_2$		Server	Comments
	\mathbf{P}_1	\mathbf{P}_2	Cache	\mathbf{P}_3	Cache	Disk	
	open(F))	-		-	-	File created
P I and PZ	write(A	.)	A		-	-	
	close()		A		-	A	
local to		open(F)	A		-	A	
		read() \rightarrow A			-	A	
	оло оло (Г)	close()	A		-	A	
Client	open(F)		A B		-	A A	
	write(D	ρ	B		-		I ocal processes
		$read() \rightarrow B$	B		_	A	see writes immediately
		close()	B		_	A	see writes infinediately
		()	B	open(F)	А	A	Remote processes
			В	$read() \rightarrow A$	А	А	do not see writes
Clients 1.2			В	close()	А	А	
	close()		В		A	В	until close()
oonourront	~		В	open(F)	΄B	В	has taken place
concurrent			В	read() \rightarrow B	В	В	1
			В	close()	В	В	
			В	open(F)	В	В	
	open(F)		B		B	B	
	write(D	')	D		B	B	
				write(C)	C	B	
	1 ()			close()	ć		
	close()		D		<u>y</u>	D	
				open(F)			Unfortunately for P_3
				$reau() \rightarrow D$			the last writer wills
			υ	close()	D		



		Client ₁		Client ₂		Server	Comments
	\mathbf{P}_1	\mathbf{P}_2	Cache	\mathbf{P}_3 (Cache	Disk	
Dd bas 1C	open(F)		-		-	-	File created
	write(A)	A		-	-	
- ,	close()		A		-	A	
ocal to		open(F)	A		-	A	
		$read() \rightarrow A$			-		
21600+1	open(F)	close()			_		
JIENLI	write(B))	B		-	A	
		open(F)	В		-	A	Local processes
		read() \rightarrow B	В		-	A	see writes immediately
		close()	В		-	A	-
			В	open(F)	А	A	Remote processes
Oliopto 1 0			В	$read() \rightarrow A$	A	A	do not see writes
			В	close()	A	A	
,	close()		В	()	A	В	until close()
roncurrent			B	open(F)	B	B	has taken place
Songuitent			В	read() \rightarrow B	В	В	
			D D	close()	D D	D D	
	open(F)		B	open(r)	B	B	
	write(D)	D		B	B	
		,	D	write(C)	C	В	
			D	close()	С	С	
	close()		D		¢	D	
	0		D	open(F)	Ď	D	Unfortunately for P_3
			D	$read() \rightarrow D$	D	D	the last writer wins
			D	close()	D	D	



		Client ₁		$Client_2$		Server	Comments
	\mathbf{P}_1	\mathbf{P}_2	Cache	\mathbf{P}_3	Cache	Disk	
	open(F)		-		-	-	File created
P I and PZ	write(A)	A		-	-	
	close()		A		-	A	
local to		open(F)	A		-	A	
		$read() \rightarrow A$	A		-	A	
		close()	A		-	A	
Client1	open(F)		A		-	A	
	write(B))	В		-	A	T 1
		open(F)	В		-	A	Local processes
		read() \rightarrow B	В		-	A	see writes immediately
		close()	B	(\mathbf{E})	-	A	Dama a ta mua angana
			B	open(F)	A	A	Remote processes
Cliente 1 2			D D	read() \rightarrow A	A	A	do not see writes
	1 ()		D	close()	A	A	
	close()		B		A	B	until close()
concurrent			B	open(F)	B	B	has taken place
Concurrent			В	read() \rightarrow B	В	В	
			В	close()	В	В	
	$\sim \sim \sim (\Gamma)$		B	open(F)	B	D D	
	open(F)	\ \	В		D D	D D	
	write(D)		$\operatorname{Autmito}(C)$	D	D D	
				close()	C	D C	
	1 ()			close()	đ		
	close()			$(\mathbf{\Gamma})$	× I		He feature tale fea D
				open(r)			Uniortunately for P_3
				reau() $\rightarrow D$			the last writer wins
			D	close()	D	D	



		Client ₁		Client ₂		Server	Comments
	\mathbf{P}_1	\mathbf{P}_2	Cache	\mathbf{P}_3	Cache	Disk	
	open(F)		-		-	-	File created
PT and PZ	write(A)	A		-	-	
	close()		A		-	A	
local to		open(F)	A		-	A	
iucai lu		$read() \rightarrow A$	A		-	A	
		close()	A		-	A	
Client1	open(F)		A		-	A	
	write(B)	$(\mathbf{\Gamma})$	B		-	A	T 1
		open(F)	D D		-	A	Local processes
		reau() \rightarrow D	D		-	A	see writes ininediately
		close()	B	opon(F)	_		Romoto processos
			B	$read() \rightarrow A$			do not see writes
Cliente 1 2			B	$close() \rightarrow R$	A		do not see writes
	alaca()		P	close()	Å	D	until alaga()
	close()		D	omon(E)		D D	until close()
concurrent			B	$road() \setminus B$	B	B	has taken place
concarron			B	$reau() \rightarrow D$	B	B	
			B	open(F)	B	B	
	open(F)		B	open(i)	B	B	
	write(D)	D		B	B	
		/	D	write(C)	Ċ	В	
			D	close()	C	C	
1	close()		D	U.	Ć	D	
l l	51000()		D	open(F)	 D	D	Unfortunately for P ₃
			D	read() \rightarrow D	D	D	the last writer wins
			D	close()	D	D	



		Client ₁		Client	2	Server	Comments
	\mathbf{P}_1	\mathbf{P}_2	Cache	\mathbf{P}_3	Cache	Disk	
	open(F)		-		-	-	File created
PT and PZ	write(A)	A		-	-	
	close()	<i>(</i>)	A		-	A	
local to		open(F)	A		-	A	
		read() \rightarrow A	A		-	A	
		close()	A		-	A	
Client	open(F)		A B		-	A A	
	wille(D)	open(F)	B		-	A	Local processes
		read() \rightarrow B	B		-	A	see writes immediately
		close()	В		-	A	5
			В	open(F)	Α	A	Remote processes
O			В	$read() \rightarrow A$	A	A	do not see writes
Clients 1.2			В	close()	A	A	
	close()		В		A	В	until close()
concurrent			В	open(F)	B	В	has taken place
CONCUMENT			В	$read() \rightarrow B$	В	В	
			B	close()	B	B	
			B	open(F)	В	В	
	open(F)	\ \	B		В	В	
	write(D)		u_{i}	D	D D	
				close()	C	D C	
				close()	đ		
	close()			open(F)	Ď		Unfortunately for Pa
			D	$read() \rightarrow D$		D	the last writer wins
			D	close()	D	D	

AFS Write Policy



- Writeback cache (in contrast with write through)
 - Opposite of NFS "every write is sacred"
 - Store contents back to server
 - When cache overflows
 - AFS: On last user close() : last closer "wins"
 - ...or don't (if client machine crashes)
- Is writeback crazy?
 - Write conflicts "assumed rare"
 - Who wants to see a half-written file?

Dealing with crashes in AFS



Client crashes

- Cache is suspect (could miss a break callback)
- Have to check with server if caching latest state
- Server crashes
 - Lose all callback state (kept in memory)
 - All clients must detect server failure + treat their local caches as suspect (as above, but across all clients)
- Contrast this with NFS in which clients don't even notice server crashes

Results for AFS



- Lower server load than NFS
 - More files cached on clients
 - Callbacks: server not busy if files are read-only (common case)
- But maybe slower: Access from local disk is much slower than from another machine's memory over LAN (better with SSD: ~1ms to read 1MB)
- For both:
 - Central server is bottleneck: all reads and writes hit it at least once;
 - is a single point of failure.
 - is costly to make them fast, beefy, and reliable.





- Why Distributed File Systems?
- Basic mechanisms for building DFSs
 - Using NFS and AFS as examples
- Design choices and their implications
 - Caching
 - Consistency
 - Naming
 - Authentication and Access Control

Topic 3: Name-Space Construction and Organization



- NFS: per-client linkage
 - Server: export /root/fs1/
 - Client: mount server:/root/fs1 /fs1
- AFS: global name space
 - Name space is organized into Volumes
 - Global directory /afs;
 - /afs/cs.wisc.edu/vol1/...; /afs/cs.stanford.edu/vol1/...
 - Each file is identified as fid = <vol_id, vnode #, unique identifier>
 - All AFS servers keep a copy of "volume location database", which is a table of vol_id→ server_ip mappings
 - Can move volumes between servers to balance load

Implications on Location Transparency



- NFS: no transparency
 - If a directory is moved from one server to another, client must remount
- AFS: transparency
 - If a volume is moved from one server to another, only the volume location database on the servers needs to be updated (clients do not need to observe the change)









- Why Distributed File Systems?
- Basic mechanisms for building DFSs
 - Using NFS and AFS as examples
- Design choices and their implications
 - Caching
 - Consistency
 - Naming
 - Authentication and Access Control

Topic 4: User Authentication and Access Control



- User U logs onto workstation A, wants to access files on server B
 - How does A tell B who U is?
 - Should B believe A?
- Choices made in NFS V2
 - All servers and all client workstations share the same <uid, gid> name space → B send U's <uid,gid> to A
 - Problem: root access on any client workstation can lead to creation of users of arbitrary <uid, gid>
 - Server believes client workstation unconditionally
 - Problem: if any client workstation is broken into, the protection of data on the server is lost;
 - <uid, gid> sent in clear-text over wire → request packets can be faked easily

User Authentication (cont'd)



- How do we fix the problems in NFS v2
 - Hack 1: root remapping → strange behavior
 - Local NFS remaps local root to nobody for protection
 - Hack 2: UID remapping \rightarrow no user mobility
 - nfsv4 uses usernames instead of UIDs.. still a hack
 - Real Solution: use a centralized Authentication/Authorization/Access-control (AAA) system

A Better AAA System: Kerberos



- Basic idea: shared secrets
 - User proves to KDC [key distribution center] who he is; KDC generates shared secret (S) between client and file server



S: specific to {client,fs} pair; "short-term **session-key**"; expiration time (e.g. 8 hours)

A Better AAA System: Kerberos



- Basic idea: shared secrets
 - User proves to KDC [key distribution center] who he is; KDC generates shared secret (S) between client and file server



S: specific to {client,fs} pair; "short-term **session-key**"; expiration time (e.g. 8 hours)

Distributed file systems require making many trade offs



- Some tradeoffs:
 - consistency, performance, scalability.
- We've learned a lot since NFS and AFS (and can implement faster, etc.), but the general lesson holds. Especially in the wide-area.
- We'll see a related tradeoff, also involving consistency, in a while: the CAP tradeoff.
 Consistency, Availability, Partition-resilience.





- Client-side caching is a fundamental technique to improve scalability and performance
 - But raises important questions of cache consistency
- Timeouts and callbacks are common methods for providing (some forms of) consistency.
- AFS picked close-to-open (session) consistency as a good balance of usability (the model seems intuitive to users), performance, etc.
 - AFS authors argued that apps with highly concurrent, shared access, like databases, needed a different model

Failure Recovery in AFS & NFS



- What if the file server fails?
- What if the client fails?
- What if both the server and the client fail?
- Network partition
 - How to detect it? How to recover from it?
 - Is there anyway to ensure absolute consistency in the presence of network partition?
 - Reads
 - Writes
- What if all three fail: network partition, server, client?

Key to Simple Failure Recovery



- Try not to keep any state on the server
- If you must keep some state on the server
 - Understand why and what state the server is keeping
 - Understand the worst case scenario of no state on the server and see if there are still ways to meet the correctness goals
 - Revert to this worst case in each combination of failure cases