File System Distribuiti

File System Distribuito

a) Un **file system distribuito** è un file system residente su computer differenti che offre una vista integrata dei dati memorizzati sui diversi dischi remoti.

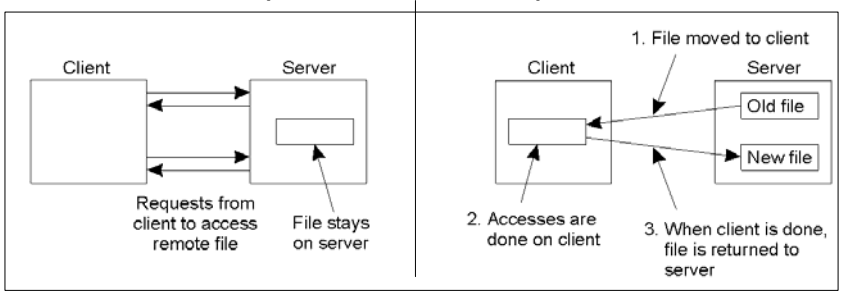
b) Esempi di file system distribuiti
   a) NFS
   b) AFS
   c) Coda
   d) Plan9
   e) xFS

Network File System (NFS)

- Originariamente sviluppato alla Sun Microsystems per le workstation SUN con sistema operativo UNIX.
- E' un modello per integrare file system differenti.
- Basato sull'idea che ogni file server fornisce una vista unificata del suo file system locale.
- NFS può essere usato su gruppi eterogenei di computer.

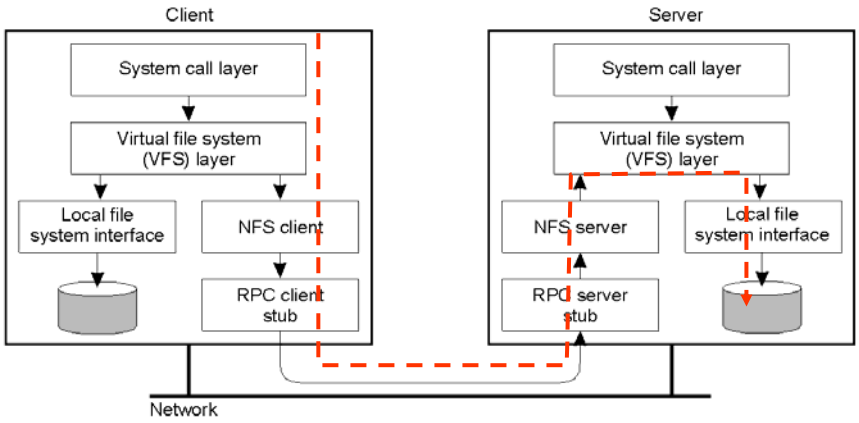
Architettura di NFS (1)

- NFS usa un **remote access model**:
  - I nodi clienti non conoscono le reali locazioni dei file.
  - I server esportano un set di operazioni sui file.



Il modello remote access. **Il modello upload/download.**

Architettura di NFS (2)



L'architettura di base di NFS per sistemi UNIX.

Architettura di NFS (3)

- NFS è indipendente dall'organizzazione del file system locale.
- Integra file systems usati in UNIX, Linux, Windows, e altri sistemi operativi.
- Il modello offerto all'utente è simile a quello dei file system UNIX-like, basato su files organizzati come sequenze di byte.

Modello del File System

Operazione	v3	v4	Descrizione
`Create`	Si	No	Crea un file
`Create`	No	Si	Crea un file non regolare (link simbolici, directory, file speciali)
`Link`	Si	Si	Crea un hard link ad un file
`Symlink`	Si	No	Crea un symbolic link ad un file
`Mkdir`	Si	No	Crea una subdirectory in una data directory
`Mknod`	Si	No	Crea un file speciale
`Rename`	Si	Si	Cambia il nome ad un file
`Rmdir`	Si	No	Rimuove una subdirectory vuota da una directory
`Open`	No	Si	Apre un file
`Close`	No	Si	Chiude un file
`Lookup`	Si	Si	Accede ad un file tramite il filename
`Readdir`	Si	Si	Legge Il contenuto di una directory
`Readlink`	Si	Si	Legge il pathname memoriazzato in un symbolic link
`Getattr`	Si	Si	Legge gli attributi di un file
`Setattr`	Si	Si	Modifica gli attributi di un file
`Read`	Si	Si	Legge i dati contenuti in un file
`Write`	Si	Si	Modifica i dati contenuti in un file

Una lista incompleta di operazioni sul file system supportati da NFS.

Comunicazioni (1)

a) In NFS tutte le comunicazioni tra i server e clienti sono implementate tramite Remote Procedure Call (RPC).

b) Il protocollo usato é: Open Network Computing RPC.

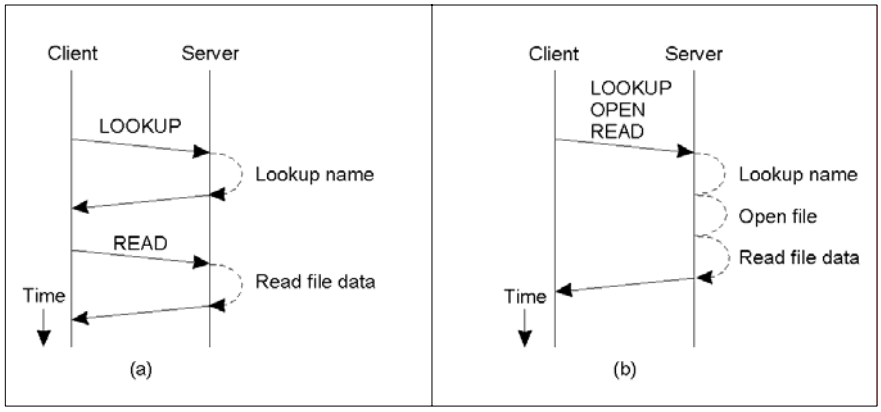
c) Prima della versione 4, NFS usava server **stateless**.

d) I clienti avevano il compito di mantenere lo stato delle operazioni correnti su un file system remoto.

Comunicazioni (2)

- Nella versione 4, NFS ha introdotto le **compound operations** che comprendono più richieste di operazioni in una singola chiamata.
- Usate per ridurre il numero di chiamate RPC e migliorare le prestazioni delle comunicazioni.
- Questo approccio è particolarmente adatto a wide-area file systems.
- Le compound operations non vengono gestite come transazioni:
  - Se una operazione in una compound procedure fallisce, le successive operazioni non vengono eseguite.
  - Viene ritornato un messaggio con le informazioni sulle operazioni eseguite e l'errore che si è verificato
  - Non conviene inviare operazioni non correlate tramite una compound procedure

Comunicazioni (3)



(a) **Lettura di dati da un file in NFS versione 3.**

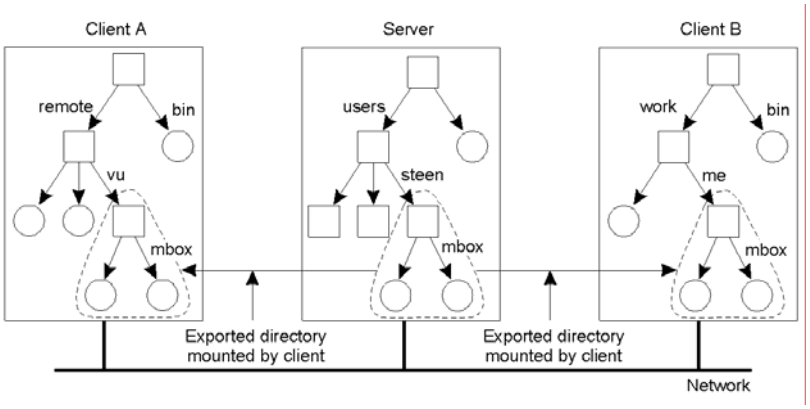
(b) **Lettura di dati da un file usando una compound procedure in NFS versionw 4.**

Comunicazioni (4)

- Nella versione 4, i server NFS mantengono lo stato di alcune operazioni.
- Questo modello è stato introdotto per gestire operazioni su file systems in reti geografiche (wide-area network), come:
  - File locking
  - Protocolli di cache consistency
  - Callback procedures.

Naming (1)

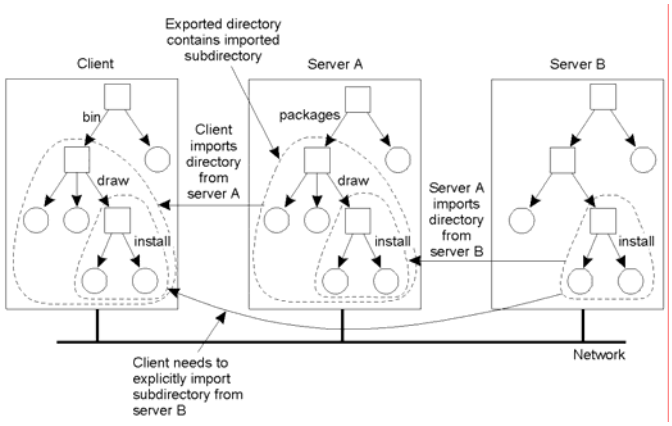
File sharing è basato su operazioni di **mounting**.



Mounting (parte di) un file system remoto in NFS.

Naming (2)

Un NFS server può montare directory esportate da altri server, ma non può esportarle ad altri clienti.

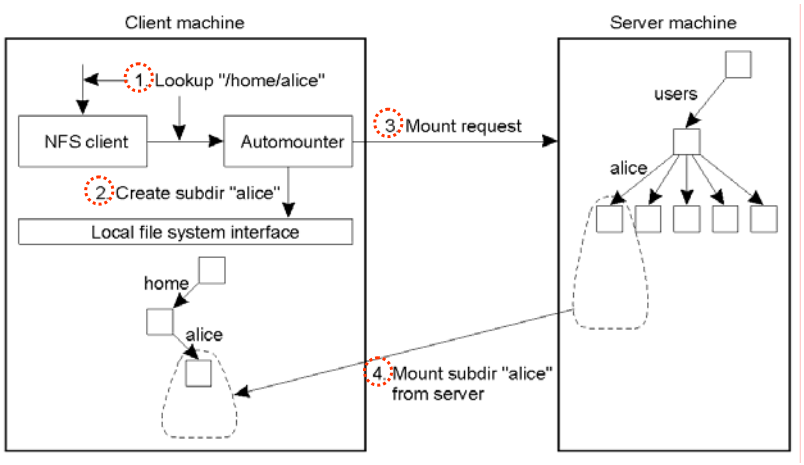


Mounting di directory innestate da più server in NFS.

Automounting (1)

- Quando un file system dovrebbe essere montato su un nodo cliente ?
- Una procedura automatica è implementata da un **automounter** per NFS che
  - effettua il mount della home directory di un utente accede al client e
  - effettua il mount di un file system on demand (quando i file sono acceduti).

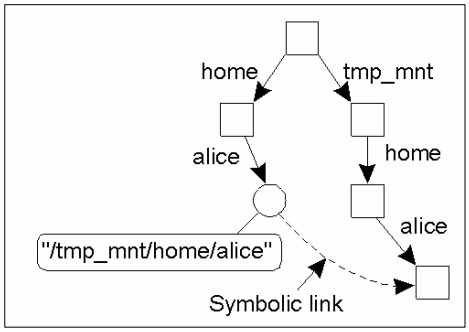
Automounting (2)



Un semplice funzionamento dell'automounter per NFS.

Automounting (3)

- Per evitare di invocare l'automounter ogniqualvolta il file viene letto, le directory possono essere montate su una sotto-directory speciale e quindi usando un link simbolico ad ogni directory montata.



Uso di symbolic links con automounting.

- Si può usare un tempo fissato per la sotto-directory speciale.

Attributi dei File (1)

- Gli attributi che usa NFS per i file sono raggruppati in due classi:
- 12 **obbligatori** (supportati da ogni implementazione) e
- 43 **raccommandati** ma non obbligatori.

Attributo	Descrizione
TYPE	Il tipo del file (regolare, directory, symbolic link)
SIZE	Dimensione in bytes del file
CHANGE	Indicatore per un client di verificare se e quando il file è stato modificato
FSID	Identificatore unico sul server del file nel file system

Alcuni degli attributi obbligatori dei file in NFS.

Attributi dei File (2)

Attributo	Descrizione
ACL	una access control list associata al file
FILEHANDLE	Il file handle fornito dal server per un file
FILEID	Identificatore unico del file nel file system
FS_LOCATIONS	Locazioni nella rete dove il file system può essere trovato
OWNER	L'identificatore del proprietario del file
TIME_ACCESS	Tempo ultimo accesso
TIME_MODIFY	Tempo ultima modifica
TIME_CREATE	Tempo di creazione

Alcuni degli attributi non obbligatori dei file in NFS.

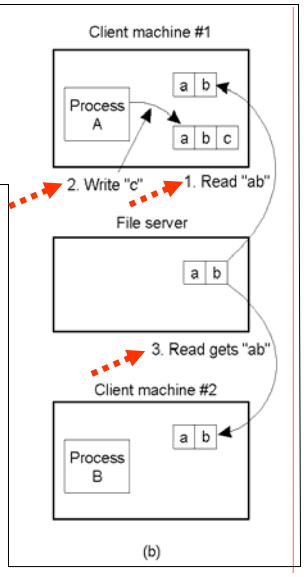
Semantica del File Sharing (1)

- Secondo la **semantica UNIX** in un file system sequenziale che permette di condividere files
  - una read dopo una write, ritorna il valore scritto
  - Dopo due successive writes una read ritorna il valore dell'ultima scrittura.

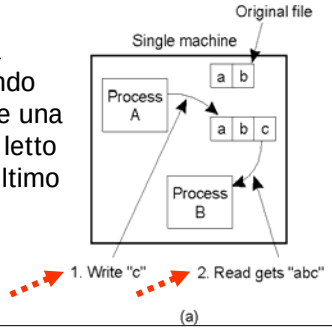
- In un sistema distribuito, la semantica UNIX può essere garantita solo se vi è un solo file server e i clienti non mantengono I file nella cache.

Semantica del File Sharing (2)

In un sistema distribuito con uso di cache, possono essere letti valori obsoleti.



Su una singola macchina quando una *read* segue una *write*, il valore letto dalla *read* è l'ultimo valore scritto.



Semantica del File Sharing (3)

- Sebbene NFS in principio usa il modello *remote access*, molte implementazioni usano cache locali, che in pratica corrisponde ad usare il modello *upload/download*.

- NSF implementa la **session semantics**:

  Le modifiche ad un file aperto sono inizialmente visibili solo al processo che ha modificato il file. Quando il file viene chiuso tutte le modifiche sono visibili agli altri processi (computer).

Semantica del File Sharing (4)

Cosa accade quando due processi memorizzano localmente un file e lo modificano?

Metodo	Commento
UNIX semantics	Ogni modifica su un file è instantaneamente visibile a tutti i processi
Session semantics	Nessuna modifica è visibile ad altri processi prima che il file venga chiuso
Immutable files	Non sono permesse modifiche; una modifica crea un nuovo file
Transaction	Tutte le modifiche sono atomiche

Quattro differenti modelli per gestire file condivisi in un sistema distribuito.

File Locking in NFS (1)

- NFS versione 4 usa uno schema di file locking.
- I Read lock non sono mutualmente esclusivi.
- I Write lock sono esclusivi.

Operation	Description
Lock	Crea un lock (r o w) per un blocco di bytes
Lockt	Controlla se un lock in conflitto è stato acquisito
Locku	Rimuove un lock per un blocco di bytes
Renew	Rinnova un lease sull'uso del lock specificato

Le operazioni di NFS vers. 4 per il file locking.

File Locking in NFS (2)

- NFS implementa anche una **modalità implicita** per il lock di un file detta **share reservation** che indica il tipo di accesso e il tipo di diniego

Richiesta di accesso

Denial state corrente

	NESSUNO	READ	WRITE	BOTH
READ	Succeed	Fail	Succeed	Fail
WRITE	Succeed	Succeed	Fail	Fail
R/W	Succeed	Fail	Fail	Fail

(a)

Richiesta di Denial state

Stato di accesso corrente

	NESSUNA	READ	WRITE	BOTH
READ	Succeed	Fail	Succeed	Fail
WRITE	Succeed	Succeed	Fail	Fail
R/W	Succeed	Fail	Fail	Fail

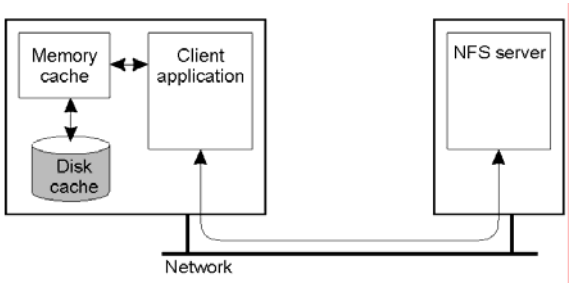
(b)

Il risultato di una *open* su un file già aperto da un'altro cliente con ***share reservations*** in NFS.

(a) **Quando un client richiede una accesso condiviso in presenza di un denial state.**
(b) **Quando un client richiede un denial state in presenza di uno stato di accesso.**

NFS Client Caching (1)

- Mentre la versione 3 non usa una cache, dalla versione 4 NFS implementa un sistema di caching sul lato client che include una Memory cache e una Disk cache.
- Per un file, i dati, gli attributi, gli handle, e le directory possono essere memorizzati nella cache.

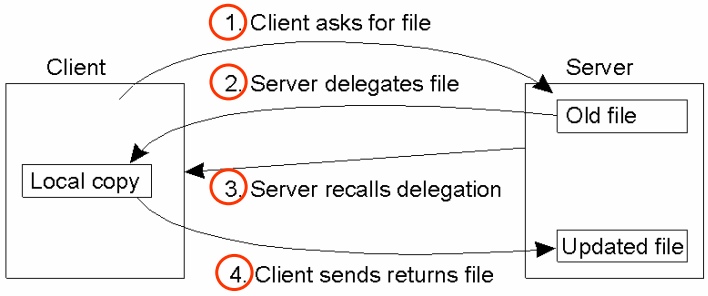


NFS Client Caching (2)

- Il sistema di caching dei dati di un file usa la *session semantics* : le modifiche dei dati nella cache vengono spostate sul server quando un client chiude il file.
- I dati possono essere mantenuti nella cache, ma se il file sarà riaperto, dovranno essere *rivalidati*.
- NFS usa anche la **open delegation** per delegare alcuni diritti al client che ha aperto il file.
- Il client può prendere decisioni relative al proprio nodo senza chiedere al server. Altre decisioni rimangono al server.

NFS Client Caching (3)

- Un server può aver bisogno di ritirare una delega quando un altro client su una macchina differente machine chiede i diritti di accesso per un file.
- Il meccanismo di **callback** è usato che ritirare una file delegation.



NFS Client Caching (4)

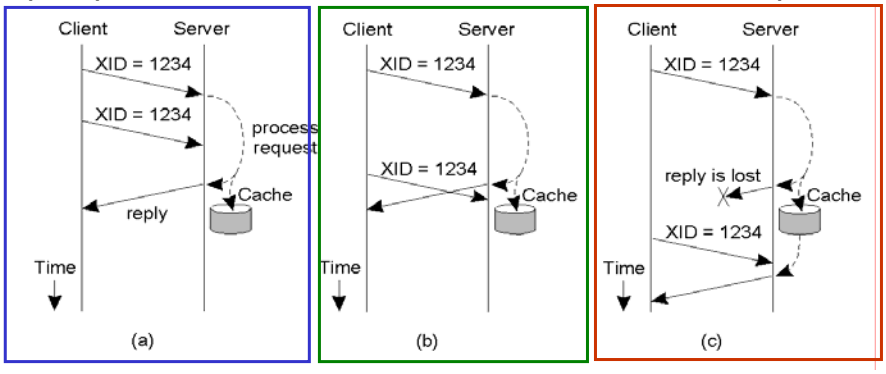
- Attributi, file handle, e directory possono essere memorizzati nella cache, ma le modifiche di questi valori devono essere inviate al server.
- Le informazioni nella cache sono automaticamente invalidate dopo un dato intervallo di tempo. Questo obbliga i clienti di ri-validarli prima di poterle riusare.
- NFS v4 offre un supporto per la **replicazione** di un file system tramite una **lista di locazioni** dove il file system può essere memorizzato (su diverse macchine del sistema distribuito).

NFS Fault Tolerance

- Poichè NFS v4 implementa **server stateful** (e gestisce file locking, open delegation, ecc.), sono necessari meccanismi di fault tolerance e di recovery per gestire eventuali fallimenti delle RPC.
- Le RPC di NFS usano protocolli TCP e UDP che hanno diversi livelli di affidabilità.
- Ad esempio: RPC può generare richieste duplicate quando si ha la perdita di una chiamata di procedura remota. Questo può portare il server ad effettuare più volte una richiesta.
- E' necessario gestire la duplicazione di chiamate.

Duplicate-Request Cache

Ogni richiesta di RPC di un client contiene un *transaction id* (***XID***) e viene memorizzata dal server insieme alla risposta.



Tre situazioni in cui è necessario gestire ritrasmissioni.

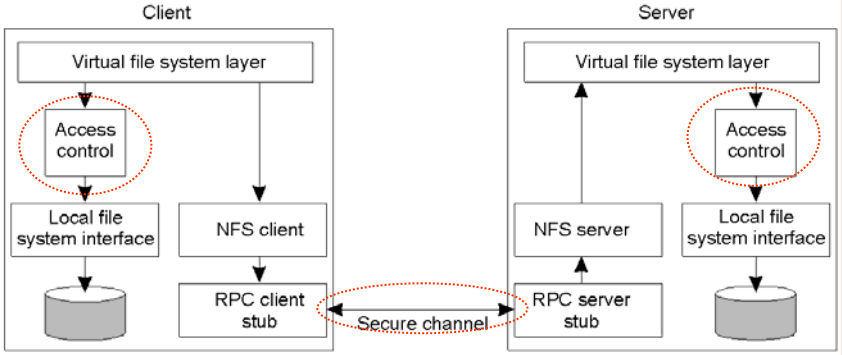
(a) **La richiesta è ancora in fase di gestione.**

(b) **La risposta è stata da poco inviata.**

(c) **La risposta era stata inviata da tempo e era stata persa.**

Sicurezza in NFS

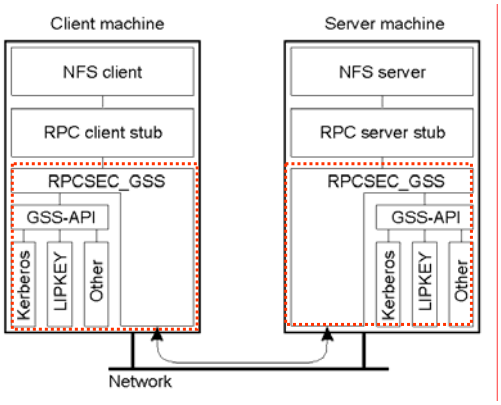
La sicurezza in NFS è basata su canali di comunicazione sicuri (**secure channels**) e meccanismi di controllo degli accessi (**file access control**).



La NFS security architecture.

RPC Sicuro

- RPC Sicuro in NFS v. 4 è basato su RPCSEC_GSS.
- RPCSEC_GSS - Generic Security Service authentication protocol per ONC RPC basato su Generic Security Services API (GSS API)



Controllo degli Accessi

Valori degli attributi ACL

Operazione	Descrizione
Read_data	Permission to read the data contained in a file
Write_data	Permission to to modify a file's data
Append_data	Permission to to append data to a file
Execute	Permission to to execute a file
List_directory	Permission to to list the contents of a directory
Add_file	Permission to to add a new file t5o a directory
Add_subdirectory	Permission to to create a subdirectory to a directory
Delete	Permission to to delete a file
Delete_child	Permission to to delete a file or directory within a directory
Read_acl	Permission to to read the ACL
Write_acl	Permission to to write the ACL
Read_attributes	The ability to read the other basic attributes of a file
Write_attributes	Permission to to change the other basic attributes of a file
Read_named_attrs	Permission to to read the named attributes of a file
Write_named_attrs	Permission to to write the named attributes of a file
Write_owner	Permission to to change the owner
Synchronize	Permission to to access a file locally at the server with synchronous reads and writes

La classificazione di operazioni riconosciute da NFS rispetto al controllo degli accessi.

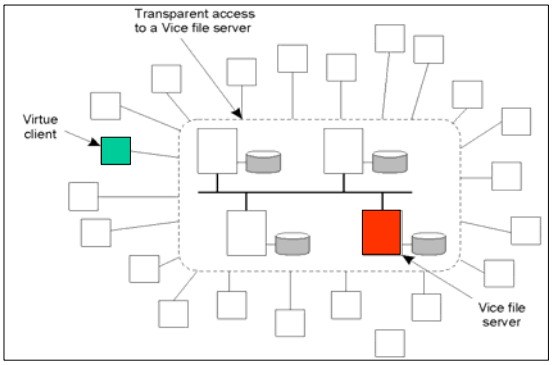
Tipi di Utenti NFS

Tipo di utente	Descrizione
Owner	Proprietario del file
Group	Gruppo degli utenti associato al file
Everyone	Qualsiasi utnte di una processo
Interactive	Qualsiasi processo che accede il file in modalità interattiva
Network	Qualsiasi processo che accede il file dalla rete
Dialup	Qualsiasi processo che accede il file tramite una connessione lenta
Batch	Qualsiasi processo che accede il file come parte di un job batch
Anonymous	Qualsiasi utente che accede il file senza autenticazione
Authenticated	Qualsiasi utente autenticato
Service	Qualsiasi processo di servizio del sistema che accede il file

Differenti tipi di utenti e processi distinti da NFS rispetto al controllo degli accessi.

Il Sistema Coda (1)

- Coda è basato sull' Andrew File System (AFS).
- Obiettivi: **naming, location transparency** e **elevata disponibilità**.

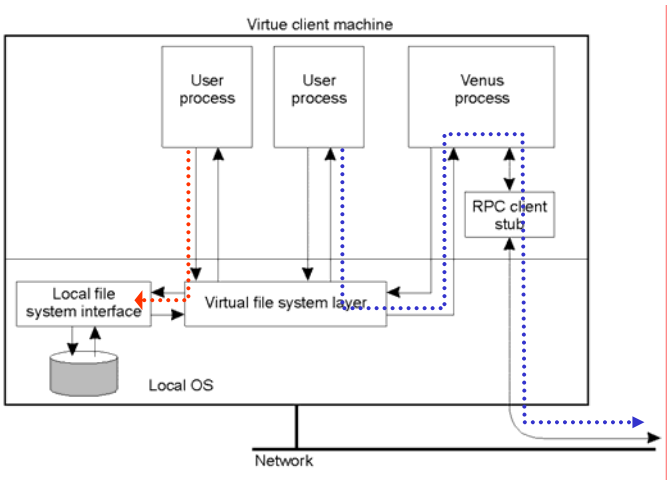


Organizzazione di AFS.

Il Sistema Coda (2)

- In ogni cliente Virtue è in esecuzione un *processo Venus* che svolge un ruolo simile ad un NFS client.
- Il processo Venus permette anche al client di continuare a lavorare anche se il file server non è accessibile.
- Le comunicazioni sono basate su **RPC affidabile**.

Il Sistema Coda (3)



L'organizzazione interna di una workstation Virtue.

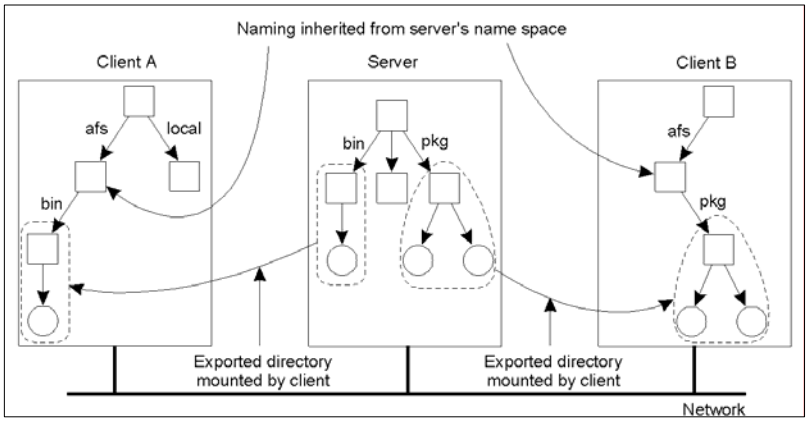
Il Sistema Coda (4)

- Coda implementa un file system UNIX-like con operazioni simili ad NFS.
- Coda implementa un name space condiviso globale mantenuto dai Vice server.
- I clienti accedono il name space globale tramite la sotto-directory speciale (*/afs*). Questo modello è diverso da NFS.
- Quando viene acceduto, una parte viene montata localmente da un processo Venus.

Il Naming in Coda (1)

- Il naming in Coda è simile a quello di UNIX.
- I File sono raggruppati in **volumes** - partizioni di disco che corrispondono a file system - associati ad un utente e memorizzati in un Vice server.
- Diversamente da NFS, in Coda un file condiviso ha lo stesso nome.
- Coda usa *Volumi Logici* e *Replicated Volume Identifiers* (***RVI***) che identifica logicamente un volume.

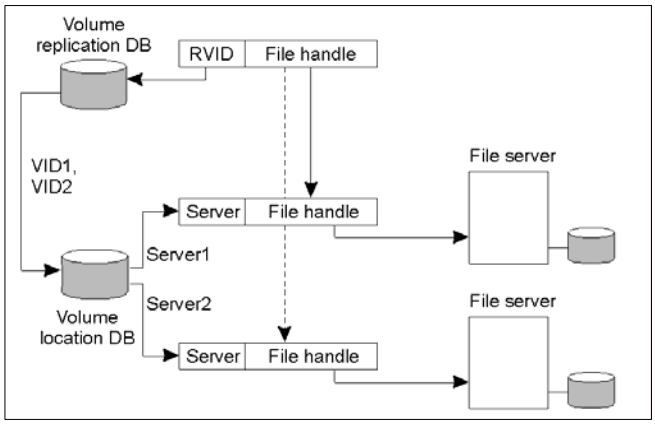
Il Naming in Coda (2)



Clienti in Coda hanno accesso ad un name space singolo condiviso.

Identificatori di File

Gli ident. dei file sono composti da due parti: **RVID** + **vnode**



L'implementazione e la risoluzione di un identificatore di un file in Coda.

Semantica Transazionale

Coda implementa una forma di semantica transazionale "debole" interpretando una sessione come una transazione che gestisce disconnessioni di server.

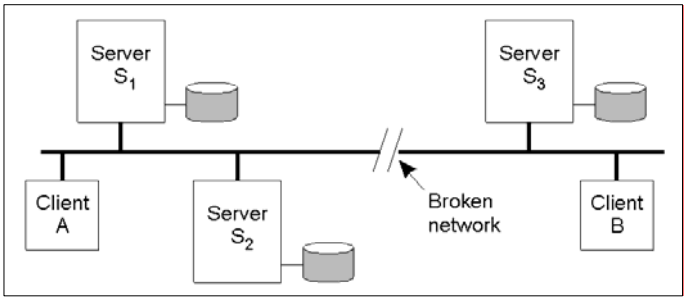
Tipi differenti di sessioni sono definite e system calls differenti sono associate ad un tipo di sessione.

Dati associati ad un file	Letti?	Modificati?
File identifier	Si	No
Diritti di accesso	Si	No
Tempo ultima modifica	Si	Si
Lunghezza del file	Si	Si
Contenuto del file	Si	Si

I metadati letti e modificati per una *store session* in Coda.

Replicazione dei Server

- Coda permette file server replicati chiamati Volume Storage Group (**VSG**).
- Un VSG è reso accessibile per ogni cliente e un protocollo di ***replicated-write*** viene usato per la consistenza.



Due clienti con differenti AVSG per lo stesso file replicato. Coda usa un modello di versioning per gestire le inconsistenze tra copie di uno stesso file

Controllo degli Accessi

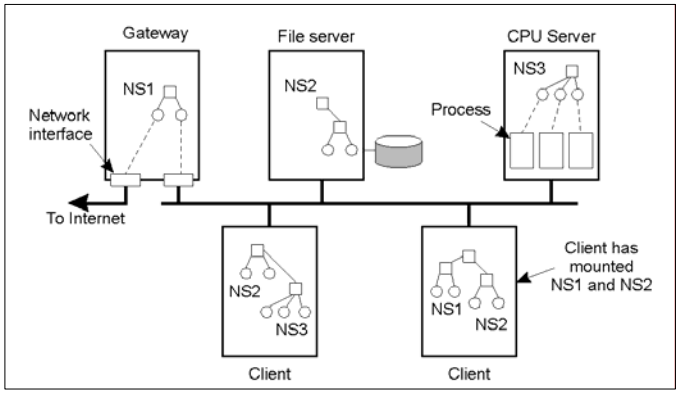
- Le Access control list sono associati con le directory e non con i file. Tutti I file nella stessa directory hanno gli stessi diritti di protezione.

Operazione	Descrizione
Read	Legge un file nella directory
Write	Modifica un file nella directory
Lookup	Verifica lo stato di un file
Insert	Aggiunge un nuovo file alla directory
Delete	Cancella un file esistente
Administer	Modifica la ACL della directory

Classificazione delle operazioni su file e directory di Coda rispetto al controllo degli accessi.

Plan 9: Risorse accedute come File

- Tutte le risorse sono accedute usando una sintassi simile a quella usata per i file su un pool di server.



Organizzazione generale di Plan 9

Comunicazioni in Plan 9

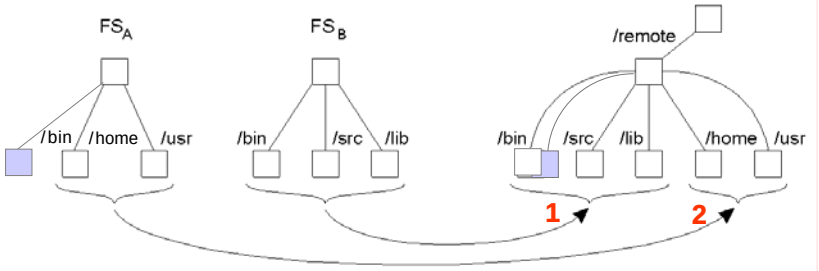
- Per le comunicazioni, Plan 9 usa il protocollo 9P e le interfacce di rete rappresentati come directory.

File	*Descrizione*
ctl	Usato per comandi di controllo specifici del protocollo
data	Usato per leggere e scrivere dati
listen	Usato per accettare richieste di connessione in arrivo
local	Fornisce informazioni sul lato chiamante della connessione
remote	Fornisce informazioni sul lato remoto (chiamato) della connessione
status	Fornisce informazioni sullo stato della connessione

File associati ad una singola connessione TCP in Plan 9.

Il Naming in Plan 9

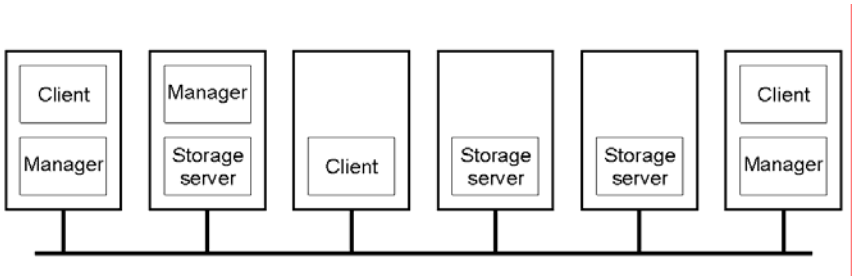
- Un client può montare più di un name space nello stesso mount point componendo una **union directory**.
- L'ordine di mounting è mantenuto durante la ricerca dei file.



Una union directory in Plan 9 dove è stato prima montato FS_B e dopo FS_A.

Principi di xFS

- Il file system xFS è basato sun un modello **serverless**.
- L'intero file system è distribuito sulle diverse macchine inclusi i clienti.
- Ogni macchina può eseguire uno storage server, un metadata server e un processo client.



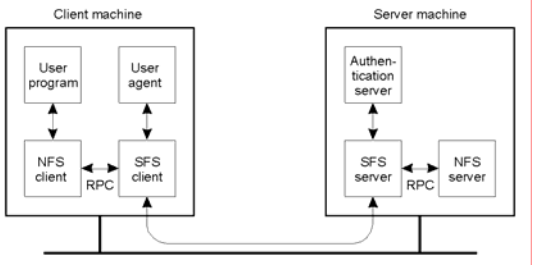
Una tipica distribuzione di processi xFS su più macchine.

Comunicazioni in xFS

- Le prestazioni delle RPC in un ambiente altamente decentralizzato non sono ottimali a causa dei problemi di elevata comunicazione.
- In xFS le RPC sono state sostituite con gli **active messages.** I messaggi attivi sono realizzati tramite processi che gestiscono il loro arrivo.
- Nel modello degli **active message**, quando un messaggio arriva, un handler viene automaticamente eseguito per gestire il messaggio ed effettuare le opportune operazioni.

Principi di SFS

- Il Secure File System (SFS) usa delle chiavi segrete per implementare meccanismi di security nel file system.
- Un client non può accedere un file senza disporre della chiave segreta associata che deve essere prelevata in precedenza.
- La gestione delle chiavi è separata dalla gestione dei file.
- SFS usa NFS v.3 per il controllo degli accessi degli utenti e per le comunicazioni.



Organizzazione di SFS.

Confronto

Issue	NFS	Coda	Plan 9	xFS	SFS
Obiettivi	Trasparenza negli accessi	Alta disponibilità	Uniformità	Sistema serverless	Sicurezza scalabile
Modello di accesso	Remoto	Up/Download	Remoto	Basato su Log	Remoto
Comunicazioni	RPC	RPC	Speciale (9P)	Active msgs	RPC
Processo Client	Leggero/pesante	Pesante	Leggero	Pesante	Medio
Grupoi di Server	No	Si	No	Si	No
Granularità di mount	Directory	File system	File system	File system	Directory
Name space	Per client	Globale	Per process	Globale	Globale
Validità File ID	File server	Globale	Server	Globale	File system
Semantica Condiv.	Session	Transazionale	UNIX	UNIX	N/S
Consistenza cache	write-back	write-back	write-through	write-back	write-back
Replicazione	Minima	ROWA	Nessuna	Striping	Nessuna
Tolleranza ai guasti	Comunicazioni affidabili	Replicazione e caching	Comunicazioni affidabili	Striping	Comunicazioni affidabili
Recovery	Client-based	Reintegrazione	N/S	Checkpoint & write logs	N/S
Canali sicuri	Meccanismi esistenti	Needham-Schroeder	Needham-Schroeder	No pathnames	Self-certificate
Controllo degli accessi	Molte operazioni	Directory operations	UNIX based	UNIX based	NFS BASED

Un confronto tra NFS, Coda, Plan 9, xFS. (N/S indica che nulla è stato specificato)