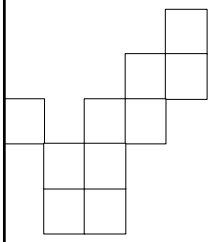


File system Distribuiti

Corso di "Sistemi per Elaborazione dell'Informazione"
Prof. Carpentieri Bruno
A.A. 2004/2005

Aucella Vincenzo
Sangiovanni Raffaele

Cos'è un file system?

- Un file system è il mezzo logico con cui un sistema operativo memorizza e recupera dati su/da hard disk di un computer, si tratti di unità locali, volumi disponibili in rete, o risorse condivise esportate in un'area di memorizzazione di rete (SAN, Storage Area Network).

Fonte: **Linux File System -** Moshe Bar - Mcgraw-hill

Operazioni fondamentali

1. creazione e cancellazione di file
2. apertura di file per la lettura e la scrittura;
3. ricerca all'interno di un file;
4. chiusura di file;
5. creazione di directory;
6. elencazione del contenuto di una directory;
7. eliminazione di file da una directory.

Fonte: **Linux File System -** Moshe Bar - Mcgraw-hill

Cos'è un sistema distribuito?

- Un sistema distribuito è un insieme di macchine debolmente connesse tramite una rete di comunicazione.
- Il termine ***macchina*** si usa per indicare sia un mainframe sia una stazione di lavoro
- Una macchina di un sistema distribuito considera ***remote*** le macchine restanti e le rispettive risorse; ***locali*** le proprie.

Fonte: **Sistemi Operativi -** Silberschatz

Introduzione

- Un ***file system distribuito*** (DFS) è un'implementazione distribuita di un file system su un sistema time-sharing, i cui *client*, *server* e *dispositivi di memoria* sono sparsi tra le varie macchine facenti parte del sistema stesso.

Fonte: **Sistemi Operativi -** Silberschatz

Struttura di un DFS

- **Servizio**
  Entità software in esecuzione su una o più macchine che fornisce una funzione a *client* non conosciuti a priori
- **Server**
  Software di servizio in esecuzione su una singola macchina
- **Client**
  Processo che può richiedere un servizio mediante una serie di operazioni a lui consentite (*interfaccia del client*)

Fonte: **Sistemi Operativi -** Silberschatz

Trasparenza di un DFS

- Un DFS deve apparire ai suoi client come un file system centralizzato convenzionale
- La molteplicità e la dispersione dei suoi server e dispositivi di memorizzazione devono essere rese trasparenti
- L'interfaccia del client di un DFS non deve distinguere tra file locali e file remoti
- La trasparenza facilita la mobilità dell'utente

Fonte: **Sistemi Operativi -** Silberschatz

Prestazioni di un DFS

- Quantità di tempo necessaria per soddisfare le varie richieste di servizio.
- È spesso necessario calcolare un overhead aggiuntivo dovuto alla struttura distribuita (rallentamento nell'esecuzione da parte della CPU dei vari protocolli di comunicazione).
- Obiettivo: raggiungere prestazioni paragonabili a quelle di un file system convenzionale.

Fonte: **Sistemi Operativi -** Silberschatz

Unità Componente

- "*È il più piccolo insieme di file che può essere memorizzato su una singola macchina, indipendentemente da altre unità*".
- Lo spazio di memoria totale è formato da spazi più piccoli localizzati in posizioni remote, spesso corrispondenti agli insiemi dei file stessi.
- Tutti i file che appartengono alla stessa *unità componente* devono risiedere nella stessa locazione.

Fonte: **Sistemi Operativi -** Silberschatz

Nominazione

- Naming: funzione di mapping tra oggetti logici e oggetti fisici.
- Gli utenti trattano oggetti logici rappresentati dai nomi dei file.
- Il sistema manipola blocchi fisici di dati, memorizzati sulle traccie di un disco rappresentati da un identificatore numerico
- Mapping multilivello: gli oggetti hanno nomi diversi ad un livello diverso nel sistema.

Fonte: **Sistemi Operativi -** Silberschatz

Strutture di nominazione

- ***Trasparenza di locazione***: il nome di un file non rivela in nessun modo la locazione di memoria fisica del file.
- ***Indipendenza dalla locazione***: il nome di un file non deve essere modificato se cambia la locazione di memoria fisica del file stesso.

Fonte: **Sistemi Operativi -** Silberschatz

Mapping statico

- Schema di nominazione che supporta la trasparenza della locazione a livello utente, ma non la ***migrazione dei file***.
- ***Migrazione dei file***: i dischi possono essere spostati manualmente da una macchina ad un'altra, ma automaticamente il sistema operativo riesce a mapparli nuovamente sullo stesso nome (supportato dal mapping dinamico).

Fonte: **Sistemi Operativi -** Silberschatz

Mapping dinamico

- *Il mapping dinamico* è uno schema di nominazione indipendente dalla locazione, in quanto può mappare lo stesso nome di file su locazione diverse in due momenti diversi.
- Uno schema di nominazione, indipendente dalla locazione, è anche trasparente, ma non è sempre vero il viceversa.

Fonte: **Sistemi Operativi -** Silberschatz

Trasparenza di locazione

- Il nome di un file deve indicare gli attributi più significativi del file stesso, come il suo contenuto, piuttosto che la sua locazione.
- Fornisce agli utenti un modo per condividere convenientemente i dati.
- Il nome di un file indica un gruppo specifico, anche se nascosto, di blocchi fisici di dischi (trasparenza di locazione statica).
- Può mostrare corrispondenze tra unità componenti e macchine (trasparenza di locazione statica).

Fonte: **Sistemi Operativi -** Silberschatz

Indipendenza della locazione

- I file indipendenti dalla locazione possono essere considerati come contenitori di dati logici che non vengono assegnati ad una specifica locazione di memoria.
- Separa la gerarchia di nominazione dalla gerarchia dei dispositivi di memoria e dalla struttura intercomputer.
- Consente una migliore condivisione dello spazio di memoria e dei dati.

Fonte: **Sistemi Operativi -** Silberschatz

Client privi di dischi

- Grazie alla separazione dei nomi dei file dalla locazione, è possibile avere client privi di dischi di memorizzazione.
- Nasce il problema dell'avviamento del sistema operativo; per ovviarlo, si memorizza un protocollo nella ROM per abilitare la connessione alla rete e recuperare il nucleo del s.o.
- I vantaggi dati da questo tipo di struttura sono tanti, tra i quali minor costo ed una maggiore convenienza.
- Lo svantaggio è dato dalla complessità del protocollo di avviamento e dal calo delle prestazioni.
- La tendenza è quella di evitare questo tipo di struttura

Fonte: **Sistemi Operativi -** Silberschatz

Schemi di nominazione:il caso *Ibis*

- Primo approccio: nomina i file mediante una combinazione dei loro nomi *host* e *local*.
- In ***IBIS***: *host*:*nome-locale*, dove *nome-locale* indica un percorso di tipo Unix del file;

  -No trasparenza, no indipendenza locazione;

  La struttura del DFS viene rappresentata da un insieme di unità componenti isolate costituite da interi file system convenzionali.

Fonte: **Sistemi Operativi -** Silberschatz

Schemi di nominazione:il caso ***NFS***

- ***Network file system*** (**NFS**): offre funzionalità per unire le directory remote con quelle locali, dando all'utente l'impressione di un albero di directory coerente.
- I montaggi delle directory avvengono su richiesta in base ad una tabella di punti di montaggio e nomi di strutture di file.
- L'integrazione, anche se supportata, è limitata e non uniforme perché ogni macchina può aggiungere diverse directory remote al proprio albero.
- Struttura complessa e difficile da mantenere.

Fonte: **Sistemi Operativi -** Silberschatz

Schemi di nominazione: terzo caso

- E' possibile avere una integrazione totale dei componenti del file system.
- Una sola struttura globale di nomi si estende a tutti i file del sistema.
- Prova a rendere la struttura del DFS molto vicina a quella di un file system convenzionale.

Fonte: **Sistemi Operativi -** Silberschatz

Realizzazione (1)

- Il mapping necessita di aggregare i file in unità componenti e fornire la mappatura non sul singolo file, ma sulla base delle un'unità componente.
- E' necessaria una tecnica che preveda di introdurre identificatori di file di basso livello indipendenti dalla locazione, ma che indichino a quale unità componente appartiene il file.
- E' necessario un secondo livello di mapping che associa le unità componenti a locazioni di memoria fisica.

Fonte: **Sistemi Operativi -** Silberschatz

Realizzazione (2)

- Gli identificatori di basso livello hanno nomi strutturati. Tali nomi sono stinghe di bit formati normalmente da una parte che identifica l'unità componente, ed una che identifica il file nell'unità componente.
- Bisogna solo far attenzione a non utilizzare un nome già usato, per avere l'unicità.

Fonte: **Sistemi Operativi -** Silberschatz

Accesso ai file remoti

- Un utente richiede un file su un server remoto.
- Il server remoto viene individuato grazie allo schema di nominazione
- Il server remoto può fornire il file attraverso il metodo del ***servizio remoto***

Fonte: **Sistemi Operativi -** Silberschatz

Servizio Remoto

- Le richieste di accesso dell'utente vengono gestite dal server che risponde con opportuni risultati
- Ogni richiesta di accesso ai file remoti comporta l'interazione col server

Svantaggi Principali

- Basse prestazioni
- Eccessivo traffico sulla rete

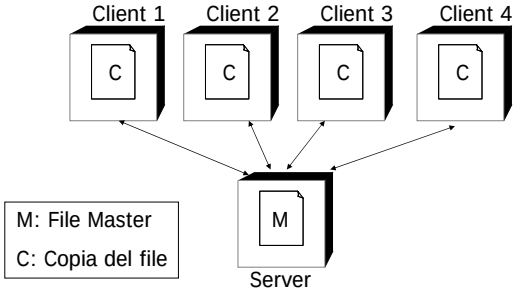
Fonte: **Sistemi Operativi -** Silberschatz

Uso della cache

- Quando al server arriva una richiesta di servizio remoto, per prima cosa si controlla nella cache se è possibile soddisfarla.
- Se la risposta è positiva, allora si trasferiscono i dati al client.
- Altrimenti il server cerca i dati in questione, li copia nella propria cache e li invia al client richiedente.
- Gli accessi ai dati remoti avvengono solo quando sono stati copiati in locale.

Fonte: **Sistemi Operativi -** Silberschatz

Dislocazione dei file nella rete



Fonte: **Sistemi Operativi -** Silberschatz

Caratteristiche del caching

Vantaggi

- Diminuzione del traffico sulla rete
- Migliori tempi di risposta del sistema client

Problemi

- Coerenza fra file master e copie nella cache dei client

Fonte: **Sistemi Operativi -** Silberschatz

Dimensione dei dati

I dati sottoposti a caching sono in blocchi generalmente più grandi di quelli richiesti per il singolo accesso.

Unità di caching più grandi comportano:

- Maggiore hit ratio (tasso di successi)
- Maggior costo per fallimento della ricerca sulla cache

Fonte: **Sistemi Operativi -** Silberschatz

Supporti per il caching

- Come memoria cache si possono usare sia la memoria centrale che il disco.
- La cache su disco offre maggiore affidabilità in caso di guasto e di ripristino del sistema.
- Il caching in memoria centrale presenta diversi vantaggi.

Fonte: **Sistemi Operativi -** Silberschatz

Caching in memoria centrale

- I client possono fare a meno dei dischi.
- Maggiore velocità nell'accesso ai dati.
- La tecnologia attuale produce memorie sempre più grandi a costi più bassi.
- La cache del server è in memoria centrale, dunque si usa una stessa tecnologia, maggiore omogeneità.

Fonte: **Sistemi Operativi -** Silberschatz

Aggiornamento della cache

- Quando un client apporta delle modifiche su un file remoto sottoposto a caching occorre regolare la sincronizzazione fra copia master e copia cache
- Esistono diverse politiche:

  Scrittura diretta, ritardata e su chiusura

Fonte: **Sistemi Operativi -** Silberschatz

Scrittura diretta

- Quando un client apporta delle modifiche ai dati in cache, avviene un aggiornamento di dati anche sulla copia master che risiede sul server.

Vantaggio:

- L'affidabilità, nel caso si guasti un client.

Svantaggio:

- Scarse prestazioni in scrittura, per scrivere i dati sul server.

Fonte: **Sistemi Operativi -** Silberschatz

Scrittura Differita

- Si differiscono gli aggiornamenti della copia master. Si cominciano a scrivere nella cache, e in un secondo momento si applicano le modifiche alla copia master del server.

Vantaggi:

- Veloce in scrittura, perché bisogna accedere solo alla cache
- E' possibile sovrascrivere gli stessi dati nella cache, ed inviare al server solo la versione più aggiornata

Svantaggio:

- Ovviamente, se per qualsiasi motivo un client subisce un guasto, si perdono i dati non ancora aggiornati

Fonte: **Sistemi Operativi -** Silberschatz

Scrittura su chiusura

- Le modifiche sulla copia cache vengono apportate alla copia master solo su chiusura del file remoto

Vantaggio:

- Migliori prestazioni rispetto al precedente, soprattutto quando i file rimangono aperti per più tempo

Svantaggio:

- Se modifichiamo di frequente un file, ci sarà un traffico eccessivo sulla rete

Fonte: **Sistemi Operativi -** Silberschatz

Coerenza dei dati

- Con l'uso della cache, nasce il problema della consistenza dei dati
- Se i dati in cache del client non sono coerenti con la copia master dei dati questi devono essere scartati e sottoposta a caching una copia aggiornata
- Due approcci possibili al problema:
    1. *Approccio iniziato dal client*
    2. *Approccio iniziato dal server*

Fonte: **Sistemi Operativi -** Silberschatz

Metodo iniziato dal client

- Il client si occupa di controllare la validità dei suoi dati
- Tale controllo può essere fatto:
    1. ad ogni accesso al file
    2. all'apertura del file
    3. ad intervalli di tempo fissi
- Questo approccio rallenta l'accesso alla cache e carica rete e server

Fonte: **Sistemi Operativi -** Silberschatz

Metodo iniziato dal server

- Il server viene informato per ogni accesso ad un file aperto da parte dei client, e ne registra il tipo di apertura (scrittura o lettura).
- In questo modo può intervenire in caso di conflitti, ed impedire l'inconsistenza.

Fonte: **Sistemi Operativi -** Silberschatz

Vantaggi del Caching

- Minore carico sul server e sulla rete
- Maggiore scalabilità
- Accessi più veloci ai dati remoti
- Minore overhead sulla rete, dovuto alle dimensioni dei blocchi sottoposti a caching
- Maggiore ottimizzazione del server

Fonte: **Sistemi Operativi -** Silberschatz

Svantaggi del Caching

- Maggiore complessità del sistema
- Necessita presenza di dischi o di grossa quantità di memoria
- Problema della coerenza della cache in caso di frequenti scritture

Fonte: **Sistemi Operativi -** Silberschatz

Tipi di Servizio

- Informazioni che mantiene il server
- **Servizio con informazioni di stato**
  - Il server tiene traccia di tutti i file cui ha accesso ogni *client*
- **Servizio senza informazioni di stato**
  - Il server si limita a rilasciare i blocchi richiesti dal *client*, senza interessarsi di come verranno utilizzati

Fonte: **Sistemi Operativi -** Silberschatz

Servizio con informazioni di stato

1. Il *client* richiede l'`open` su un file
2. Il *client* richiede l'accesso a quel file
3. Il *server* registra in memoria il file
4. Il *server* fornisce un identificativo di connessione unico per quel client ed il file aperto
   - Indice alla struttura di *i-nodes* presente in memoria centrale.
5. Questo indice viene utilizzato per gli accessi successivi fino al termine della sessione.

Fonte: **Sistemi Operativi -** Silberschatz

Servizio senza informazioni di stato

1. La richiesta fatta dal *client* identifica completamente il file e la sua posizione all'interno del server
2. Il *server* non mantiene una tabella dei file aperti
3. Non è necessario né aprire o chiudere connessioni in relazione alle operazioni `open` e `close`
4. Le operazioni di lettura e scrittura avvengono tramite messaggi remoti
5. Le operazione sui file (comprese quelle di cancellazione) sono indipendenti ed *idempotenti*, cioè daranno sempre lo stesso effetto in qualsiasi momento vengano eseguite

Fonte: **Sistemi Operativi -** Silberschatz

Confronto dei servizi Blocco del sistema

- **Servizio con informazioni di stato**
  - Server: le informazioni in memoria vengo perse; lo stato può essere recuperato solo mediante un protocollo di ripristino basato su un dialogo con i client.
  - Client: il server deve esserne informato per poter terminare i processi client rimasti in sospeso (*rilevamento e terminazione degli orfani*)
- **Servizio senza informazioni di stato**
  - Una volta riparato, il server può continuare a rispondere alle richieste che vengono ritrasmesse automaticamente dai client.

Fonte: **Sistemi Operativi -** Silberschatz

Confronto dei servizi: Prestazioni

- **Servizio con informazioni di stato**
  - Informazione è contenuta in memoria centrale
  - Se un file è stato aperto per accesso sequenziale, il server ne conosce lo stato e può quindi leggerne in anticipo i blocchi successivi.
  - Indispensabile se i server utilizzano lo stesso metodo di validazione della cache (ad esempio i descrittori di file e gli offset impliciti di UNIX)
- **Servizio senza informazioni di stato**
  - Messaggi di richiesta e tempi di elaborazione più lunghi per mancanza di informazione precaricata in memoria

Fonte: **Sistemi Operativi -** Silberschatz

Replicazione dei file

- Ridondanza (*indipendente)* di file per migliorarne la disponibilità
- Può aumentare le prestazioni generali quando si scelgono file su server più vicini al client richiedente
- Deve essere una struttura trasparente ed invisibile sia all'utente, sia ai livelli superiori di ciascun server.
- Viene controllata stabilendo il grado e la disposizione delle repliche.
- Può essere vista anche come una serie di accessi virtuali allo stesso file logico.

Fonte: **Sistemi Operativi -** Silberschatz

Aggiornamento delle repliche

- Per l'utente, le repliche indicano la stessa entità logica, quindi l'aggiornamento deve riflettersi su tutte le copie fisiche del file per mantenere la consistenza dell'informazione
- La *coerenza* consiste nell'assicurare che le repliche del file si riferiscano allo stesso file logico
- La *disponibilità* rappresenta la capacità di reperire in breve tempo i file sui server, utilizzando eventualmente copie più reperibili
- Si cerca di ottenere un compromesso accettabile tra *coerenza* e *disponibilità*
  - Il sistema Locus, ad esempio, nel caso in cui la rete venga partizionata, utilizza estensivamente la replicazione a scapito della coerenza

Fonte: **Sistemi Operativi -** Silberschatz

Esempio replicazione dei file: Sistema Ibis

- Utilizza una variante dell'approccio di copia primaria
- Il dominio del mapping dei nomi è rappresentato da una coppia <*id-replica-primaria,id-replica-locale*> relativa ad un singolo sistema componente
  - Se localmente non sono presenti repliche, la coppia viene sostituita con un valore speciale che lega quindi il mapping ad ogni singola macchina
  - Se la replica locale coincide con quella primaria, i due *id* saranno identici
- Supporta la *replicazione su richiesta*

Fonte: **Sistemi Operativi -** Silberschatz

Sistema Ibis: Replicazione su richiesta

- È una politica per il controllo automatico della replicazione, analoga al caching dell'intero file
- La lettura di una replica non locale genera la copia nella cache locale, generando quindi una nuova replica non primaria
- Gli aggiornamenti si applicano solo alla copia primaria e ciò causerà l'invalidazione di tutte le altre repliche.
- Per fare questo si utilizza un opportuno sistema di scambio messaggi
- Non è garantita l'invalidazione atomica e serializzata di tutte le repliche non primarie
  - Una replica vecchia può essere considerata valida
- In caso di accessi di scrittura remoti, la copia primaria viene trasferita sulla macchina richiedente.

Fonte: **Sistemi Operativi -** Silberschatz

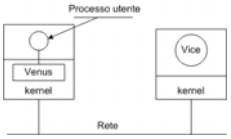
Caso di Studio: AFS

- Ambiente di calcolo distribuito progettato e sviluppato a partire dal 1983 presso la Carnegie-Mellon University (CMU)
- Il suo nome è in onore di Andrew Carnegie e Andrew Mellon, fondatori della CMU; viene comunemente chiamato AFS
- Scopo del progetto è di fornire a studenti e docenti della CMU un ambiente di tipo UNIX con un file system condiviso
- La Transarc Corporation, che ne ha continuato lo sviluppo, è stata acquistata dall'IBM, la quale ha prodotto diverse versioni commerciali, in particolare il *Transarc DFS (assai simile all'AFS)*
- AFS è scalabile: il sistema è stato progettato per comprendere più di 5000 workstation

Fonte: **Sistemi Operativi -** Silberschatz

AFS - Architettura

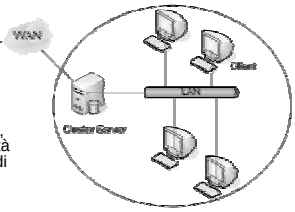
- Distinzione tra *macchine client* (workstation) e *macchine server*, connessi tra loro attraverso LAN o WAN
- **Vice**: server dedicati che presentano ai client
  - uno spazio di nomi condivisi in forma di gerarchia omogenea
  - trasparenza di locazione
- **Virtue**: protocollo, integrato nel kernel,che i client utilizzano per comunicare con i *Vice*.
- **Venus**: processo client che comunica con i Vice



Fonte: **Sistemi Operativi -** Silberschatz

AFS - Cluster

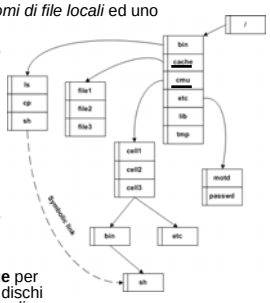
- Client e server sono strutturati in cluster interconnessi da una LAN
- Un cluster consiste di una collezione di workstation e un *cluster server* è connesso alla WAN tramite un router
- I client utilizzano soprattutto i server del proprio cluster, rendendo meno frequente il riferimento a file intercluster
- Si scarica lavoro dei server sui client, tenendo conto del fatto che la velocità della CPU del server è il reale collo di bottiglia del sistema



Fonte: **Sistemi Operativi -** Silberschatz

AFS – Architettura Client

- Le macchine client hanno uno *spazio di nomi di file locali* ed uno *spazio di nomi condivisi* offerto dai *Vice*
- Su ogni client, lo spazio dei nomi visibile ai programmi utente appare come un albero UNIX tradizionale, con l'aggiunta delle directory */cmu*, e */cache*
- La directory */cache* contiene i file remoti posti nella cache
- la directory */cmu* contiene i nomi delle workstation condivise, raggruppati in *cellule*, sotto cui si trovano i rispettivi file system
- I file system remoti sono montati in */cmu*
- le altre directory e file sono strettamente locali e non condivisi
- Le workstation utilizzano il protocollo **Virtue** per comunicare con i Vice e devono possedere dischi locali per memorizzare lo spazio dei nomi locali



Fonte: **Sistemi Operativi -** Silberschatz

AFS – Architettura Server

- I server *Vice* presentano lo spazio dei nomi condivisi ai client come una gerarchia di file omogenea e indipendente dalla locazione
- L'insieme dei server è responsabile della memorizzazione e gestione dello spazio dei nomi condivisi

Fonte: **Sistemi Operativi -** Silberschatz

AFS - Caratteristiche

- **Mobilità dei client:** I client possono accedere a qualsiasi file si trovi nello spazio dei nomi condivisi
- **Sicurezza:** Sui Vice non vengono eseguiti programmi client; le funzioni di convalida e trasmissione sono fornite da un pacchetto basato sul modello RPC. La comunicazione è cifrata
- **Protezione:** Sono fornite **liste d'accesso** per la protezione delle directory
- **Eterogeneità:** La specifica dell'interfaccia permette l'integrazione di architetture e sistemi operativi.

Fonte: **Sistemi Operativi -** Silberschatz

AFS – Spazio dei nomi condivisi

- È composto da unità chiamate volumi
  - generalmente associati con i file su un singolo client
  - Uniti tra loro tramite un meccanismo simile al montaggio dello Unix
- Un file o una directory del Vice sono identificati tramite un *fid*. Un fid è lungo 96 bit ed è costituito da tre elementi della stessa lunghezza:
  - Numero di volume
  - Numero di *vnode*: indice in un array contenente gli *inode* dei file in un singolo volume.
  - Un *unicizzatore*: permette il riuso dei valori di *vnode*, mantenendo le strutture compatte.
- I *fid* sono trasparenti alla locazione, quindi gli spostamenti di un file da server a server non invalidano il contenuto della directory che si trova nella cache
- Le informazioni sulla locazione sono mantenute in una **base di dati di locazione-volume** e l'informazione replicata su ciascun server
- Bilanciamento dei volumi tramite l'operazione di spostamento dei volumi (operazione atomica)

Fonte: **Sistemi Operativi -** Silberschatz

AFS - Caching

- AFS effettua il caching degli interi file
  - blocchi di 64K
  - *Venus* intercetta la *open* e scarica il file in */cache*
  - Lettura e scrittura dei byte del file sono effettuate dal kernel senza intervento di *Venus*
  - alla chiusura il file viene memorizzato sul server
  - politica LRU
- Venus effettua il caching del contenuto delle directory e dei link simbolici;
  - Ricerche sulle directory prelevate, sono eseguite localalmente
- Eccezioni alla politica di caching sono le modifiche alle directory che sono fatte direttamente sul server responsabile per quella directory

Fonte: **Sistemi Operativi -** Silberschatz

AFS – Coerenza dei file

- Le modifiche al file saranno visibili ad altri siti solo all'atto della chiusura
- Venus presuppone che i dati della cache siano validi
  - Non interverrà il Vice per validare la copia
  - Riduce il numero di richieste di validazione

Fonte: **Sistemi Operativi -** Silberschatz

AFS – Coerenza dei file: l'approccio

- Il client copia nella propria cache un file o una directory
- Il server aggiorna le informazioni di stato e invia un contrassegno di validità per quel file
- Se un client modifica un file, il server revoca la validità per tutti i client che possiedono nella cache quel file.
- Il client può caricare un file nella cache solo se contrassegnato come valido
- Se il file non è valido, il client ne richiede la nuova versione dal server

Fonte: **Sistemi Operativi -** Silberschatz

AFS – Coerenza dei file: considerazioni

- Traffico sulla rete presente per la validazine della cache
- Quando si riavvia una workstation il processo Venus considera sospetto il contenuto della cache
  - Genera una richiesta di validazione per ogni risorsa nella cache
- Ogni server deve mantenere le informazioni per la validità
  - se il carico è eccessivo Il server interrompe l'uso di contrassegni, richiede memoria e revoca la validità per i file contenuti nelle cache dei client

Fonte: **Sistemi Operativi -** Silberschatz

Caso di studio: Network File System

- NFS è la soluzione proposta da Sun al problema del DFS
- NFS è sia una specifica che un sistema software implementato dalla Sun
- Si basa sul protocollo UDP Ethernet

Fonte: http://www.disi.unige.it/person/DoderoG/so2/fsdistri.htm

NFS - Panoramica

- Nella rete non si fa distinzione fra macchine client e server
- Workstation indipendenti ciascuna con un proprio File System
- Non esiste quindi un "FS globale", ma solo file che un client accede in remoto

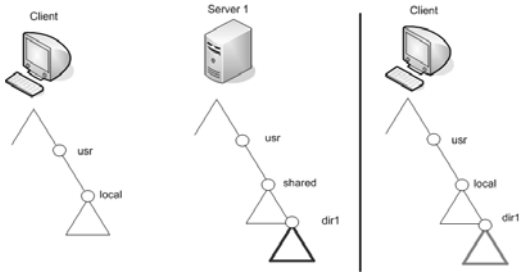
Fonte: http://www.disi.unige.it/person/DoderoG/so2/fsdistri.htm

NFS - Montaggio di File System

- Per poter usare su un client una directory remota, occorre poterla **"montare"**
- L'operazione e' permessa solo al superuser (della macchina client) e non e' trasparente
  - occorre esplicitare la macchina host
  - gli accessi successivi sono trasparenti
- Una macchina può montare su una (o più) proprie directory parti del file system di altre macchine
- Montaggio a cascata

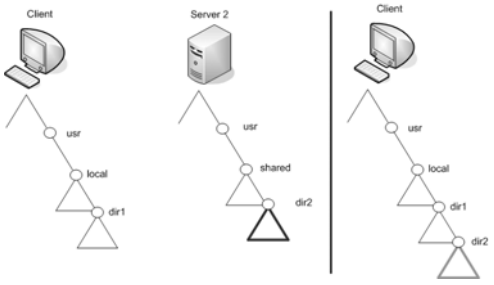
Fonte: http://www.disi.unige.it/person/DoderoG/so2/fsdistri.htm

NFS - Esempio di montaggio



Fonte: http://www.disi.unige.it/person/DoderoG/so2/fsdistri.htm

NFS - Esempio di montaggio a cascata



Fonte: http://www.disi.unige.it/person/DoderoG/so2/fsdistri.htm

NFS – considerazioni sul montaggio

- Se c'erano dei file locali in */usr/local*, su *client* non sono piu' visibili.
- Il montaggio a cascata **non ha effetto** su *server1*
  - i file al di sotto di *dir1* continuano ad essere visibili (salvo mount effettuati dal superuser di *server1*, ovviamente).
- Analogamente, mount effettuati dal superuser di *server1* sulla directory *dir1* non sono visibili da *client*,
  - sempre che non siano "duplicati" dal superuser di *client*
- La visione del FS è tutt'altro che univoca da parte delle varie workstation
  - salvo che eseguano tutte uno stesso script iniziale
  - la visione locale del FS è indipendente dalle altre wotkstation

Fonte: http://www.disi.unige.it/person/DoderoG/so2/fsdistri.htm

NFS - Come funziona il montaggio ?

- Su ogni macchina abbiamo un processo server esterno al kernel
- Si specifica il nome della directory remota e della macchina su cui si trova
- La richiesta di montaggio viene instradata al server
- Il server controlla nella lista di esportazione se la macchina che la richiede è abilitata a montare quella directory
- Il server restituisce al client una *file handle* per accedere al file system remoto montato

Fonte: http://www.disi.unige.it/person/DoderoG/so2/fsdistri.htm

Eterogeneità

- NFS è costituito da tre parti indipendenti:
  - il protocollo NFS stesso
  - *Remote Procedure Call* (RPC)
  - *eXternal Data Representation* (XDR)
- Inter-operabilità di Workstation con sistemi operativi differenti

Fonte: http://www.disi.unige.it/person/DoderoG/so2/fsdistri.htm

NFS – Servizi offerti

- Attraverso il *file handle,* le operazioni previste dal protocollo NFS sono:
  - Cercare un file su una directory
  - Leggere elementi di una directory
  - Manipolare directory e link
  - Modificare attributi file
  - Leggere e scrivere file

Fonte: http://www.disi.unige.it/person/DoderoG/so2/fsdistri.htm

NFS – Considerazioni sul protocollo

- NFS funziona senza informazione di stato
  - mancano infatti open e close
- I server non mantengono dati relativi ai client
  - lista directory esportate
- Ogni client deve specificare a ogni accesso tutti i parametri della RPC usata
- NFS non gestisce la concorrenza, garantisce solo che le scritture siano atomiche

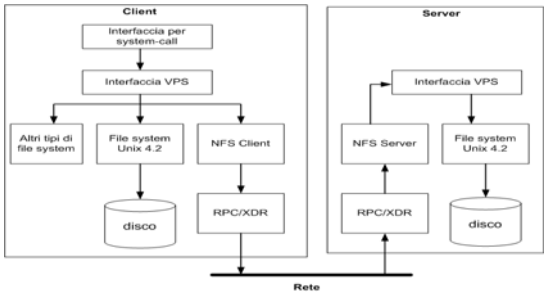
Fonte: http://www.disi.unige.it/person/DoderoG/so2/fsdistri.htm

NFS – Cenni Implementazione Sun

- Per motivi di efficienza NFS e' integrato nel kernel
- Viene inserito uno strato intermedio detto *VFS* (virtual file system)
  - interfaccia verso altri file system (Ext2 o FAT)
- Gestisce una struttura detta *v-node*
  - *I v-node sono unici nel sistema*
  - *Gli i-node sono unici per ogni singola macchina*

Fonte: http://www.disi.unige.it/person/DoderoG/so2/fsdistri.htm

Architettura



Fonte: http://www.disi.unige.it/person/DoderoG/so2/fsdistri.htm