Accesso remoto ai file

M. Donatelli, A. Ghiselli , G. Mirabelli

(draft)

28 febbraio 2002

1 Introduzione

Questa proposta si inquadra all'interno del progetto INFN-Grid[1] e l'obiettivo che si propone e' di fornire uno strumento per l'accesso ai file che si trovano su computer remoti, facendo in modo da operare su di essi come se fossero presenti nel filesystem locale; inoltre si vuole mantenere la compatibilità con il file naming di DataGrid.

La piattaforma principale del progetto DATAGRID e INFN-GRID e' di Linux[2], in quanto, dentro questo ambiente e' possibile interagire con gli strumenti del sistema operativo senza problemi di licenze e permessi, ed anche per via della grande documentazione presente su Internet, e alla possibilità di sviluppare software open-source. Questa proposta e' allineata con questa scelta per mantenere la compatibilità con l'ambiente Grid e per le ragioni di cui sopra. Lo studio attuale si basa sul il kernel 2.4 , l'ultima versione stabile disponibile.

Per accedere ai file remoti sono possibili diverse metodologie, per esempio con un'applicazione che utilizza delle API (Application Program Interface), oppure attraverso le system call del sistema operativo.

Durante la ricerca del metodo di accesso ai file remoti sono state vagliate diverse possibilità attualmente disponibili, come FTP[3], FTPFS[4], GridFTP[5], NFSv3[6], NFSv4[7], GFS[8], AFS, CodaFS[9], Intermezzo[10].

Fra le possibili soluzioni che sono state già sviluppate non ne esiste una che soddisfi completamente le richieste di partenza, ma in particolare NFS nelle versioni 3 e 4 e' quella che maggiormente le soddisfa. NFS3 al momento e' integrato nel kernel di Linux, invece NFS4, che e' ancora più interessante per le novità che presenta rispetto a NFS3, e' correntemente in via di sviluppo, ed e' disponibile in una versione(beta) non del tutto stabile, integrato nel kernel 2.4.4 .

Nei capitoli successivi vengono approfondite le caratteristiche specifiche di NFS4 e viene fatta una discussione più ampia dei problemi connessi con l'utilizzo del mount all'interno dell'ambiente Grid.

2 Definizioni e struttura del documento

2.1 Acronimi e concetti sintetici

NFS	Network File System

VFS[11]	Virtual File System
Firewall [12]	Dispositivo capace di filtrare il traffico
Protocollo di mount [13]	Trasporta le informazioni necessarie a montare il filesystem remoto mediante le RPC e restituisce il file handle del mount point al client nfsv3. In nfsv4 il protocollo di mount e' stato integrato nel protocollo NFS.
Replica Catalog[14]	Archivio che mantiene la corrispondenza fra i file logici e quelli presenti fisicamente nei nodi in ambiente Grid
LFN	Logical File Name in ambiente Grid[15]
PFN	Phisical File Name in ambiente Grid
TFN	Transport File Name in ambiente Grid
Nodo	Termine usato per identificare un worker node in ambiente Grid
Porta	Termine utilizzato nei protocolli di trasporto di rete.
NLM [16]	Network Lock Manager, protocollo di gestione dei lock dei file mediante le RPC utilizzato da nfsv3, ed integrato nel protocollo NFS in nfsv4
RPC[17]	Remote Procedure Call
Portmap	Servizio necessario per identificare un protocollo RPC
ACL[18]	Access Contro List, utilizzate per la gestione avanzata dei permessi ai file
ACE	Singola entry di una ACL
Fstab[19]	File utilizzato per automatizzare il mount all'avvio del sistema e per permettere esecuzione successiva di mount a livello utente

2.2 INFN-Grid

Il progetto affronta le problematiche del calcolo distribuito nell'ambito dei modelli GRID che hanno come riferimento l'architettura a layer proposta da Globus. In particolare il progetto europeo DataGrid, di cui l'INFN fa parte, costituisce il riferimento principale per lo sviluppo dei servizi di grid fra cui l'accesso ai dati e' uno degli aspetti rilevanti del progetto. Le soluzioni finora proposte per l'accesso ai dati remoti, si basano su meccanismi di trasferimento dei files piuttosto che su meccanismi di accesso remoto all'interno dei files. L'idea e' quindi di andare a calcolare vicino alla locazione dei dati piuttosto che porsi l problema di un file access remoto (su wide area network) efficiente e sicuro. Andare a calcolare vicino ai dati in un ambiente di calcolo distribuito, significa dover fare parecchie repliche degli stessi files e sviluppare un sistema adatto per gestirle: i Replica Catalog e i Replica manager. Inoltre per soddisfare le esigenze degli utenti di una sintassi di metadata e metafile, per garantire trasparenza sulla locazione fisica dei dati, i Replica Manager hanno anche il compito di fare il mapping fra il metafile e i file fisici.

Un'altra esigenza importante per INFN-Grid e' la security, attualmente la scelta e' basata sui certificati X.509 a chiave pubblica e sul meccanismo delle deleghe per avere una unica autorizzazione quando si accede alla grid.

La proposta di questo documento, oltre a porsi l'obiettivo di realizzare un accesso efficiente e sicuro a grosse quantita' di dati, vuole essere congruente con le scelte di INFN-Grid sia per il file naming sia per la security.

2.3 Struttura del documento

Nel capitolo 3 verrà descritto cosa si intende per file access. Nel capitolo 4 c'e' un introduzione al Network File System e un elenco di alcune differenze fra la versione 3 e la versione 4 di NFS, in particolare quelle connesse con gli obiettivi della proposta. Nel capitolo 5 verranno esaminate in particolare le problematiche relative all'operazione di mount. Nel capitolo 6 viene introdotto l'utilizzo di NFS in Grid. Nel capitolo 7 viene fatta una descrizione delle inefficienze dei protocolli di trasporto. Infine nel capitolo 8 vengono tratte le conclusioni e indicato il programma di lavoro futuro.

3 File Access vs File Transfer

Il file access e' un insieme di routine che permettono l'accesso ad informazioni sui file e sulle directory e ai singoli record o blocchi dei file in lettura e in scrittura su un disco di un determinato host in un ambiente locale.

Il VFS permette di organizzare diversi filesystem fisici, locali o remoti, in una struttura gerarchica e consente di accedere ai file mediante le stesse primitive (chiamate system call). Attraverso NFS, tali routine possono eseguire da un qualsiasi host un accesso nel senso sopra descritto ad un qualsiasi disco della rete locale.

Per i collegamenti a lunga distanza, in un'evoluzione durata parecchi decenni, e' stato costruito un programma detto FTP che permette il trasferimento di un intero file (quindi non un accesso a parti di file) da un nodo ad un altro.

Il nostro scopo all'interno di grid e' di permettere ad un qualsiasi programma di utente che fa uso di file locali di girare su un nodo di grid facendo ancora uso di questi file attraverso un metodo di file access remoto.

Questo ci obbliga a scartare i vari programmi di file transfer, evoluzioni più o meno moderne di ftp, anche se li esamineremo per confrontarne eventuali vantaggi di prestazioni e ci costringe ad esaminare i due maggiori protocolli di file access (AFS[20] e NFS) per individuare la possibilità di un loro uso in ambiente Grid.

AFS viene scartato poiché ha una struttura di filesystem di tipo "centralizzato", che non si sposa con le esigenze di un sistema distribuito, che al contrario richiede la libertà da una qualsiasi forma di vincolo predefinito.

4 Network File System

4.1 Introduzione a NFS

NFS nasce con lo scopo di permettere l'accesso ai file che risiedono in un host remoto e vederli come se fossero presenti localmente in un altro host. L'inserimento all'interno del VFS avviene mediante l'operazione di "mount" .

Il VFS e' una struttura ad albero in cui tutti gli oggetti sono dei file, e ad essi sono associate delle operazioni, le system call, comuni a tutti i file. In realtà i file dentro al VFS sono fisicamente molto diversi tra loro, possono essere file comuni, directory, dispositivi a blocchi, link, dispositivi a carattere.

Il VFS inoltre nasconde la struttura fisica dei vari 'real file system', le informazioni di basso livello, rendendo le funzioni di alto livello omogenee e trasparenti.

Infatti, dentro al VFS un filesystem presente fisicamente su un floppy o su un disco rigido viene visto allo stesso modo di un network file system o di un dispositivo a blocchi. In realtà , a basso livello questi filesystem sono molto diversi e le system call, che nel VFS sono analoghe, sono realizzate con porzioni di codice molto diverse.

Per rendere omogenee queste operazioni, vi e' una struttura associata ad ogni tipo del real file system, in modo che ad ogni system call standard del VFS corrisponda un puntatore ad una routine specifica per quel tipo di real filesystem, in questo modo il sistema operativo e' capace di gestire a basso livello l'implementazione particolare della system call specifica per quel tipo di filesystem.

NFSv3 offre grandi vantaggi, come ad esempio :

- ?? implementazione interna al kernel di Linux
- ?? velocità e leggerezza del protocollo
- ?? presenza su tutte le macchine
- ?? stabilità del protocollo
- ?? utilizzo della cache del VFS (lato client e lato server)
- ?? (letture e scritture) I/O asincrone
- ?? remote file access

Tuttavia sono presenti anche dei problemi:

porting (utilizzo) su rete geografica.

Poiché le macchine con NFS comunicano attraverso il protocolli di rete, vengono coinvolte diverse 'porte', alcune standard ed altre no. Questo, all'interno di una rete locale non causa grossi problemi, anzi funziona in modo molto efficiente, ma volendo estendere queste reti NFS verso la rete geografica internet ci si scontra con problemi di sicurezza. La protezione dei Firewall diventa inutilizzabile non conoscendo a priori le porte associate ad alcuni sottoservizi di NFS, quale ad esempio e' il protocollo di mount che non ha associata una porta standard. La porta associata ad un particolare servizio RPC (remote procedure call) viene conosciuta attraverso il portmap (porta 111) in modo dinamico, oltre al mount ci sono diversi altri sottoservizi come NLM (Network lock manager) che manifestano gli stessi inconvenienti del mount. Per questa ragione NFS risulta molto vulnerabile per quanto riguarda la sicurezza e difficilmente protetto con dei Firewall, addirittura i Firewall devono lasciare aperte un numero elevato di porte per far funzionare questo protocollo su reti geografiche causando seri problemi di sicurezza su tutte le macchine da proteggere.

Security

Altro problema che identifichiamo nel NFSv3 riguarda la sicurezza, in quanto il suo protocollo prevede delle semplici funzionalità di autorizzazione e di autenticazione, valide nel caso in cui si operi all'interno di una rete locale, ma del tutto inefficienti ed inefficaci nel caso in cui si passi su rete geografica.

Anche in ambiente Grid, per la comunicazione tra macchine remote, e' prevista l'adozione di specifiche API riguardanti la sicurezza, le GSS_API.

Queste funzioni applicative si preoccupano di garantire i requisiti di sicurezza in modo indipendente dal protocollo di sicurezza che si vuole adottare, l'importante e' che ci sia lo stesso protocollo di sicurezza su entrambi gli host che comunicano.

Sono possibili tre diversi livelli di sicurezza che possono essere garantiti:

- Autenticazione: Garanzia che i dati siano realmente inviati dal vero mittente. Può anche avere valore di autorizzazione.
- Protezione: Nascondere i dati con opportune tecniche di crittografia
- Integrità: Verifica che i dati trasmessi non siano stati modificati lungo la strada di comunicazione

4.2 NFS versione 4

Per ovviare ai problemi di porting su rete geografica e' stata studiata una sua evoluzione, la versione 4 di NFS, che risolve gran parte dei problemi delle precedenti versioni pur mantenendo praticamente invariate le caratteristiche che hanno reso interessante questo protocollo.

Una novità importante di NFS4 e' l'utilizzo della porta 2049 che e' stata standardizzata. In questo modo sono automaticamente risolti tutti i problemi di sicurezza che si avevano nella precedente versione riguardo alle reti internet e ai Firewall, infatti basta solo lasciare aperta la porta 2049 dei server per far funzionare correttamente il protocollo NFSv4.

E' stato eliminato il vecchio protocollo di mount e quello di NLM nelle RPC e sono stati aggiunti nel protocollo principale, sono state riscritte le Remote Procedure Call, per fare in modo da includere le nuove funzionalità.

Le modifiche[21] che sono state fatte al protocollo NFS per permettere il funzionamento efficiente su internet e su rete geografica:

- Utilizzare la sicurezza forte basata su uno schema a chiave pubblica
- Abilitare il funzionamento attraverso i Firewall
- Definire le operazioni composite per ridurre la latenza di tipo round trip
- Richiesto il TCP come protocollo di trasporto (detto così e' poco data l'inefficienza del TCP sulle alte velocità)
- Definire uno spazio di nomi globale per l'identificazione degli utenti

E' stata introdotto il supporto per la sicurezza mediante le RPCSEC_GSS [22](funzioni indipendenti dal particolare protocollo per garantire la sicurezza) integrate nelle RPC versione 2 con il supporto integrato per kerberos v.5 [23]e lipkey. E' stato introdotto un miglioramento nelle RPC che non sono più trasmesse singolarmente, ma vengono inoltrati al server più comandi insieme (procedura compound), questo permette di risparmiare tempo in connessioni con un round trip time elevato riducendo il numero di richieste/risposte consecutive per svolgere una singola system call. Sono state implementate anche le ACL (access control list) per aumentare l'articolazione delle autorizzazioni associate ai file, le ACL di NFSv4 non sono conformi con lo standard POSIX[24] ma neanche con quello NTFS della Microsoft, però sono tali da poter essere impiegate sia sotto Linux che sotto Windows senza alcun problema di sicurezza.

Le funzioni delle ACL che ci sono in NFSv4 sono :
ALLOW : permette l'accesso ad un'entità
DENY : Non permette l'accesso ad un'entità
AUDIT : fa in modo che l'entità nell'ACE (Access Control Entry)verifichi l'accesso specificato, tracciando poi l'accesso
ALARM : genera un allarme dipendente dal sistema se l'entità nell'ACE verifichi l'accesso specificato

4.3 Altri protocolli di accesso remoto ai file

Per considerare come soluzione valida NFS, sono stati analizzati anche altri protocolli disponibili, che sono commentati di seguito.

File Transfer Protocol o FTP e' un protocollo di trasferimento dei file remoti che lavora con interi file ed e' accessibile a livello applicativo tramite delle API. E' inadatto ai nostri scopi in quanto non garantisce alcun meccanismo di sicurezza sui dati, non gestisce il locking dei file e non permette l'accesso remoto ai file.

GridFTP invece e' un'evoluzione di FTP, a cui sono state aggiunte delle API riguardo alla sicurezza, e' stato potenziato il protocollo tradizionale di FTP permettendo trasferimenti di flussi paralleli e permettendo un semplice meccanismo di accesso remoto ai file. Purtroppo e' possibile utilizzarlo solo a livello applicativo e funziona molto bene soprattutto per trasferimenti di grosse quantità di dati, o di file di grosse dimensioni e infine non gestisce il locking dei file.

E' stato progettato un filesystem chiamato FTPFS, che permette di utilizzare il protocollo FTP per poter montare una directory FTP all'interno di un nodo locale, ma, oltre alle limitazioni di FTP dette prima, al momento il prodotto e' abbastanza instabile e presenta alcuni problemi, come notevoli rallentamenti in caso di accessi paralleli, anche se l'idea di montare un filesystem utilizzando il protocollo di trasporto diffuso come FTP e' stata molto

apprezzata. Infatti in questo modo tutta la parte del protocollo di trasporto viene ignorata, preoccupandosi soltanto della parte dell'integrazione nel Virtual filesystem locale.

Sono stati analizzati altri approcci per poter accedere a file remoti, come per esempio il CodaFS, e' una valida evoluzione di AFS, ma alcune sue caratteristiche di accodamento determinano a volte delle situazioni di conflitto fra diverse versioni dello stesso file che vanno risolte con un intervento manuale, cosa inaccettabile per ciò che vogliamo fare. Un'evoluzione del Coda filesystem e' Intermezzo che e' sviluppato in un linguaggio di alto livello.

Gli ultimi due progetti illustrati sono molto interessanti, ma non adatti per un ambiente di produzione perché presentano delle inconsistenze nelle sincronizzazioni simultanee di diversi file che spesso servono essere risolti a mano da un operatore. In un sistema distribuito questo approccio non e' proponibile.

5 Panoramica sull'utilizzo del mount in NFS

5.1 Comando/operazione di mount di Linux e di NFSv4

Il comando di mount permette ad un host (client) di inserire in un punto qualsiasi del VFS un filesystem tra quelli riconosciuti dal kernel (ext2,fat32,ntfs,nfs,coda,dos, ecc.). Nel sistema operativo di Linux viene aggiunta una nuova entry nella lista dei mount point, e' importante evidenziare che il numero dei mount point disponibili e' limitato, anche se configurabile mediante una variabile di sistema.

Per quanto riguarda NFS con il comando di mount eseguito in un client posso montare un filesystem esportato da un server remoto NFS. Invece il mount nel client di NFSv4 aggiunge un filesystem o **pseudofilesystem** oppure solo una sua porzione, esportato da un server NFSv4. Il punto di inserimento del nuovo filesystem nel client NFS si chiama mount point. Il permesso di montare un nuovo filesystem e' consentito solo al superutente, in quanto e' un'operazione critica che interagisce con delle strutture del kernel del SO. Questa operazione e' possibile farla all'avvio della macchina, oppure dando ad un utente il permesso di eseguirla successivamente.

Come verrà approfondito nel paragrafo successivo, idi questa proposta, non si monta il filesystem di questo o quel server, ma un filesystem "virtuale", gridfs, e successivamente attraverso le 'lookup /grid/nodo/path/../filename' si "montano" i diversi file esportati da corrispondenti server secondo la sintassi /grid/node/path/../filename . Poiché DataGrid ha scelto la sintassi URL per i file, sarà necessario operare un mapping fra la sintassi URL e quella di Gridfs

L'unica directory da 'montare' sarà '/grid' con il filesystem di tipo gridfs, un NFSv4-FS modificato per rispondere alle suddette esigenze.

Per automatizzare il processo di mount del filesystem gridfs in /grid allo startup della macchina, verrà utilizzata una entry aggiuntiva nel file /etc/fstab , permettendo inoltre l'accesso a questo mount point a qualsiasi utente.

5.2 Proposta di modifica del MOUNT in NFSv4

In NFSv4 il protocollo di mount rappresenta la fase iniziale della comunicazione tra client e server. Con questo protocollo il client chiede al server di restituire il file handle (root-fh) relativo alla directory di più alto livello.

Con NFSv4 nella fase di mount e' possibile anche specificare direttamente un PATH diverso da / (ROOT), ed accedere direttamente ad una porzione del filesystem offerta dal server (mediante l'impiego del public-fh). Pertanto il nuovo protocollo di mount e' molto più versatile e potente. Inoltre tale protocollo non e' più separato rispetto a quello di NFS, rendendo più semplice il suo utilizzo nelle reti geografiche. Ad ogni protocollo in NFSv3 e' associata una porta non nota a priori, invece in NFSv4 tutte le RPC di NFS passano attraverso la porta 2049 (standardizzata). Il mount e' integrato nella procedura LOOKUP che permette di risolvere un path ed ottenere il fh corrispondente al file cercato (con una sola RPC di richiesta ed una sola di risposta). PUTROOTFH restituisce il fh di ROOT, invece il PUBLIC-fh puo' rappresentare un qualsiasi file del filesystem esportato dal del server.

La scelta di integrare la porzione interessata del filesystem remoto nel Virtual File System locale, permette di sfruttare i benefici della cache locale, delle system call del sistema operativo e delle funzioni di sistema (open, copy, create, lock, ecc.) di accesso ai file, ereditate automaticamente.

Per fare questo il sistema operativo prevede l'operazione di mount, ma nel caso in cui si abbia a che fare con diversi file system remoti, quest'operazione può diventare frequente, come nel nostro caso, cosa che normalmente non dovrebbe avvenire, poiché il mount è un'operazione di sistema piuttosto pesante che interagisce con il VFS e molte altre strutture dati del kernel.

Ciò che si vuole evitare è il mount permanente di tutti i possibili nodi cui un dato utente può accedere, ma solo quelli correntemente utilizzati. Una possibilità è di dividere l'operazione di mount in due parti:

una prima parte in cui si monta NFS (a livello di sistema) una volta per tutte allo startup, una seconda parte in cui si esegue la connessione con un determinato nodo. Questa seconda parte si fa a livello utente e senza privilegi particolari, poi all'uscita del programma dell'utente si effettua la disconnessione di quel determinato nodo.

Per far questo occorre intervenire a livello di fstab all'interno del mount, che già attualmente permette in molti casi di eseguire a livello utente gran parte delle operazioni di mount. In questo modo viene lasciata all'utente la gestione delle rispettive connessioni o disconnessioni del protocollo verso i nodi nfs-server.

Lo schema seguente di Immagine 1 illustra lo splitting del mount nelle due fasi appena descritte.

Allo startup viene fatto il mount di /grid come appartenente al filesystem 'gridfs' secondo il comando:

mount –t gridfs /grid.

Supponiamo che da un nodo qualunque della Grid ci sia la richiesta di connettere server3 per accedere ad un suo file, chiamato file4. Per fare ciò secondo la sintassi tradizionale bisogna dare il seguente comando:

mount –t nfs server3:/grid/server3 … poi effettuare l'operazione umount.

Con la modifica che proponiamo invece:

il file4 del server3, sarà disponibile nel VFS locale del nodo richiedente con il percorso **/grid/server3/file4 .**

In realtà viene ottimizzata una lookup che stabilisce la connessione con il server3 e restituisce il file handle del file4, in modo tele che sia visto con il percorso: **/grid/server3/file4 .**

Si potrà pertanto utilizzare per questo file tutte le system call che fornisce il sistema operativo e trattare l file proprio come se fosse presente sul nodo locale.

Se un'altra applicazione sullo stesso nodo chiede di accedere al file 3 sul server2, con la Lookup si stabilirà la connessione con il server2 e si otterrà il file handle di file 3, che nel VFS locale corrisponderà al percorso **/grid/server2/file3**.

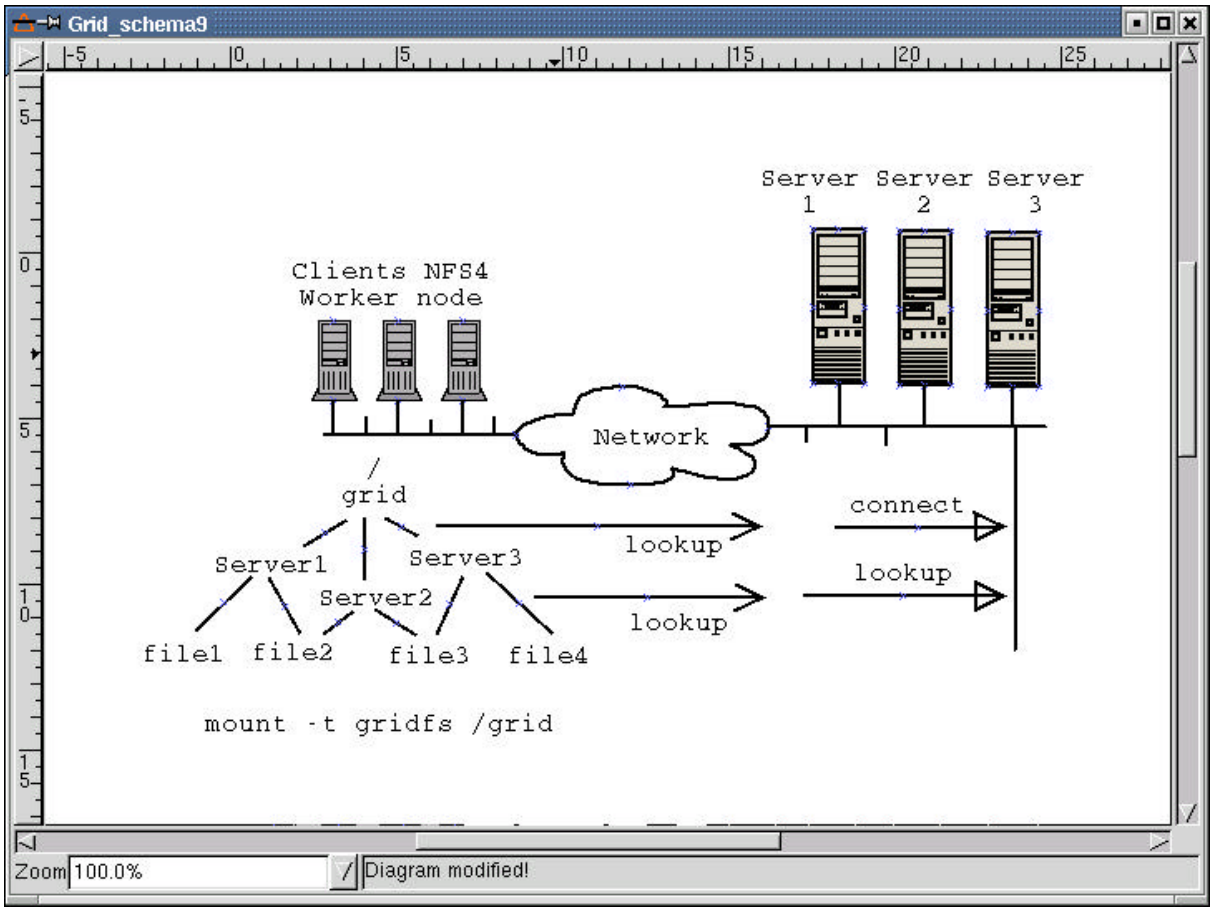

Immagine 1

L'implementazione dell'accesso ai file conNFSv4 si divide in due parti:

?? codice di esecuzione delle system call specifiche di NFSv4 nel SO

?? corrispondenza fra system call del SO e sequenza di procedure di NFS corrispondenti

L'obiettivo di questa proposta è di lasciare inalterato il codice di esecuzione delle system call specifiche di NFSv4 che vanno integrate dal kernel. Questo ci permette di separare il nostro lavoro da quello dell'integrazione di NFSv4 nel kernel di Linux, quindi non sarà necessario curare l'integrazione fra NFS ed il kernel. Questo lavoro sarà demandato agli sviluppatori di NFSv4 sotto la piattaforma Linux.

La parte che andremo a modificare sarà la seconda, cioè cambieremo le corrispondenze tra le system call del SO e la sequenza delle procedure di NFS corrispondenti. Il server non viene toccato , rimane un server NFSv4 a tutti gli effetti. Pensiamo di montare una volta per tutte la directory /grid in ogni client NFSv4 . Il comando mount di NFSv4 nei vari client NFSv4 sarà modificato in modo che all'atto del montaggio del filesystem non venga effettuata alcuna connessione verso alcun server NFSv4. Successivamente, quando sarà necessario accedere ad un file presente in un qualsiasi server NFSv4, allora in quel momento, sarà effettuata la connessione.
In particolare nella system call LOOKUP, verrà risolto il percorso del VFS nel rispettivo file handle e gestita la connessione con il server remoto.

6 Utilizzo di NFS in Grid

Esempio che mostra un possibile utilizzo di questo sistema all'interno della Grid.
Come detto in precedenza uno degli scopi di questa proposta è di rispettare il file naming presente in ambiente Grid.

Si considera che un nodo voglia accedere ad un certo file di nome file1 di cui conosce il logical file name del tipo:

lfn://virtualhostname/file1

Il nodo manderà una richiesta di ricerca del phisical file name (PFN) corrispondente ad un Replica Catalog, quest'ultimo gli restituirà uno o più PFN, con una sintassi del tipo:

pfn://server1/path1/file1

pfn://server2/path2/file1

pfn://server3/path3/file1

Il nodo locale in questione sceglierà tra questi un solo PFN ed utilizzerà il protocollo di trasporto nfs per accedere al file, quindi il transport file name sarà del tipo:

nfs://server1/path1/file1

A questo punto il nodo locale ha tutte le informazioni per connettersi al server 'server1' ed accedere al file file1 in questione, utilizzando il protocollo di nfs, in particolare compirà le seguenti operazioni:

connect server1

open /path1/file1

read /path1/file1

close /path1/file1

disconnect server1

Graficamente la sequenza delle operazioni suddette è la seguente:

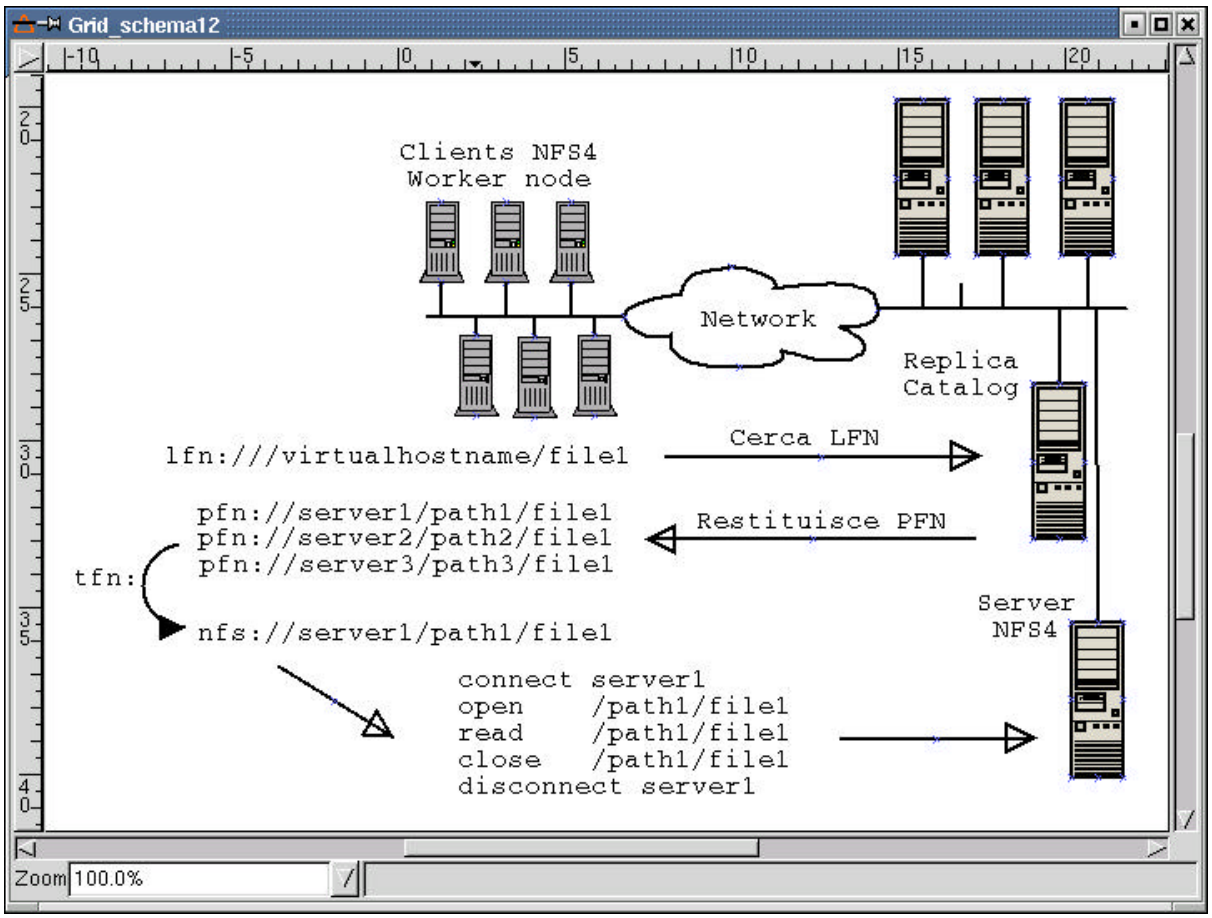

Figura 2

Il file in questione, all'interno del VFS del nodo locale verrà visto come:

/grid/server1/path1/file1

7 Inefficienze dei protocolli di trasporto

Un problema molto sentito ultimamente riguarda l'efficienza dei protocolli di trasporto per le reti ad alta velocità, soprattutto per quanto riguarda il trasferimento di grosse quantità di dati (stream) con protocolli di tipo affidabile e connesso.

Una soluzione molto adottata è di spezzare il trasferimento dello stream in diverse porzioni più piccole che vengono eseguite in parallelo. (ad oggi esistono applicazioni di trasferimento dei file, quali GridFTP che implementano queste funzionalità a livello di API)

Questo approccio presenta i seguenti vantaggi:

?? Vengono superate le inefficienze del protocollo TCP per quanto riguarda le situazioni di congestione nelle comunicazioni di tipo peer-to-peer

?? E' possibile, nel caso in cui siano presenti diverse repliche dei dati che ci interessano, instaurare diverse connessioni verso più host remoti, ottimizzando il carico dei server remoti e migliorando le prestazioni in caso di congestione.

Nel documento non ci si occuperà del dettaglio di questi aspetti, anche se è stato ritenuto utile farne menzione.

E' allo studio attuale un protocollo di trasport, SCTP (Strema Control Transport Protocol) che cerca di risolvere i problemi suddetti. Sono state proposte delle modifiche nel kernel di Linux (e altri sistemi operativi) per supportare questo nuovo protocollo in modo ottimizzato. Per far funzionare questo protocollo nel kernel di Linux, in particolare, si vogliono modificare alcune primitive del sistema operativo e si vuole aggiornare la parte relativa alle interfacce di rete, permettendo l'uso di diverse interfacce di rete affacciate verso reti diverse, tali da poter lavorare parallelamente, senza dover più specificare staticamente alcuna interfaccia di rete predefinita, ma non escludendo la possibilità di farlo per ragioni di sicurezza. In questo modo le macchine risulterebbero anche maggiormente fault tolerance per i problemi di rete. Sono stati condotti anche degli esperimenti riguardo al funzionamento in situazione di congestione di flussi di traffico di tipo SCTP aggregati con quelli di tipo TCP per studiarne l'interazione in caso di congestione. A quanto pare, da test effettuati dagli sviluppatori di SCTP, nessuno dei due tipi di traffico in caso di congestione prevale sull'altro.

In questo documento pertanto viene accennato il problema dell'inefficienza del protocollo di trasporto sulle reti ad alta velocità, ma non viene considerata una soluzione a livello di protocollo applicativo. La soluzione di questo problema può essere spostata a livello di protocollo di trasporto, ad esempio con l'impiego del protocollo SCTP, quindi ci occuperemo soltanto di seguirne i suoi sviluppi.

8 Conclusioni

Lo studio delle nuove funzionalità e caratteristiche architetturali di NFS4 e i primi test fatti ci hanno portato a considerare questo software come sistema di base per l'accesso remoto ai file in un ambiente di grid . La modifica proposta per implementare la connessione al server remoto e utilizzare il caching locale delle strutture remote del file in oggetto, basato sul meccanismo del VFS, permetterà un 'mount' user-oriented, dinamico e multiutente

Verrà fatto uno studio implementativo relativo alle modifiche che riguardano il file fstab per avere un funzionamento dell'operazione di mount a livello utente e le relative problematiche.

Un altro obiettivo che si vuole approfondire e' lo studio dei permessi, intesi come autenticazione dei server remoti. Inoltre si studieranno le prestazioni del protocollo di nfs simulando delle situazioni di uso reale, evidenziando eventualmente punti deboli del protocollo e cercando possibili soluzioni o modifiche.

Infine si pensa di effettuare la simulazione di una situazione operativa di uso del sistema sopra descritto, con la relativa integrazione completa in Grid.

9 Bibliografia

The Grid: Blueprint for a New Computing Infrastructure, Edited by Ian Foster and Carl Kesselman, July 1998

10 Note

[1] INFN-Grid : http://server11.infn.it/grid/

[2] LINUX : www.linux.org

[3] FTP: www.ietf.org - rfc414 , rfc737 , rfc2577 , rfc2228, rfc1639

[4] FTPFS: http://ftpfs.sourceforge.net

[5] GridFTP : http://www.globus.org/datagrid/gridftp.html

[6] NFS v.3 : www.ietf.org - rfc1813 , rpc 2623 (security in NFS3)

[7] NFS v.4 : www.ietf.org – rfc3010,
www.nfsv4.org ,
http://citi.umich.edu/projects/nfsv4

[8] Global File System: http://www.sistina.com/products_gfs.htm

[9] CODA Filesystem: http://www.coda.cs.cmu.edu

[10] Intermezzo: http://inter-mezzo.org/

[11] Virtual File System: http://www.cse.unsw.edu.au/~neilb/oss/linux-commentary/vfs.html
http://www.atnf.csiro.au/~rgooch/linux/vfs.txt

[12] Firewall: http://www.linux.org/docs/ldp/howto/Firewall-HOWTO.html

[13] Protocollo di mount : articolo che parla del mount protocol in nfs

[14] Replica Catalog : www.eu-datagrid.org

[15] Ambiente Grid (LFN, PFN, TFN) : www.eu-datagrid.org
http://150.146.1.82/grid/internal/deliverables.htm

[16] Network Lock Manager : http://docsrv.caldera.com:1997/NET_nfs/nfsC.lock_overview.html
http://www.unet.univie.ac.at/aix/aixbman/commadmn/nfs_netlock.htm

[17] Remote Procedure Call : www.ietf.org - (RPCv2) rfc1831 - rfc2203 (rpcsec_gss)

[18] Access Control List : http://acl.bestbits.at/about.html

[19] /etc/fstab : file configurazione dei dispositivi che devono essere montati automaticamente al boot del sistema, e' possibile in Linux avere altre informazioni con il comando: ***man fstab***

[20] Andrew File System : http://www.transarc.com/Product/EFS/AFS/index.html

[21] novità di NFSv4 : http://www.nfsv4.org/papers/index.html

[22] rpcsec_gss : www.ietf.org - rfc2203

[23] kerberos v.5 : www.ietf.org - rfc2623 –
The Kerberos Version 5 GSS-API Mechanism - rfc1964
The Kerberos Network Authentication Service (V5) – rfc1510

[24] POSIX : http://standards.ieee.org/regauth/posix