Università di Pisa

Facolta' di Ingegneria

Corso di Laurea Specialistica in Ingegneria Informatica

Tesi di Laurea

Progettazione e sviluppo di un metodo per lo spostamento dei dati in Grid per l'esperimento CDF

Relatore:
Prof. Andrea Domenici

Candidato:
Antonio Dario Cuomo

Relatore:
Prof. Marco Avvenuti

Relatore:
Prof.ssa Donatella Lucchesi (INFN-Padova)

Riassunto analitico

Lo scopo di questa tesi e' quello di descrivere il lavoro di progettazione e sviluppo di un metodo per lo spostamento dei dati in Grid per l'esperimento CDF (Collision Detector at Fermilab). Negli ultimi mesi le necessita di calcolo per l'esperimento CDF sono aumentate notevolmente. Questo ha portato il CDF a muoversi verso un'architettura di calcolo Grid. Uno dei problemi aperti nel passaggio al grid rimane quello della gestione degli output delle simulazioni montecarlo prodotti nei Worker Node. Nella tesi si descrive la fase di progettazione e sviluppo della soluzione adotatta basata sulla tecnologia SAM/SRM .

Indice generale

1 INTRODUZIONE AL GRID COMPUTING

In questo capitolo viene data una definizione di Grid e si spiegano le motivazioni che hanno spinto la comunità scientifica ad orientare molti sforzi (in termini di tempo e denaro) per la ricerca in questo settore.
Successivamente si descrivono i requisiti tecnici che una Grid deve soddisfare.

1.1 Definizione di Grid

Il termine Grid è stato introdotto a metà degli anni '90 nell'ambito di una proposta di infrastruttura di elaborazione distribuita per lo sviluppo e l'esecuzione di applicazioni scientifiche di grande complessità ed elevato costo computazionale. Sviluppi significativi in questa linea di ricerca hanno consentito, negli anni successivi, di mettere a fuoco in modo più preciso il concetto di Grid.
Dal punto di vista tecnico, una Grid è da considerarsi sostanzialmente un'architettura basata su protocolli che consentono la condivisione scalabile, sicura e coordinata delle risorse: CPU, dischi, sensori, strumenti scientifici, applicazioni etc.
Il Grid fornisce risorse di calcolo e dati provenienti da diverse organizzazioni geograficamente sparse, che condividono risorse software e hardware.

Il servizio offerto non si limita al semplice accesso a queste risorse condivise, bensì provvede anche al supporto per lavori collaborativi.

Allo stato attuale, la definizione di Grid su cui concorda la gran parte della comunità scientifica è quella coniata da I. Foster, C. Kesselmann e S. Tuecke, secondo cui una Grid è:
“ l’insieme delle infrastrutture hardware e software che consente la condivisione di risorse e la risoluzione coordinata di problemi nell’ambito di organizzazioni virtuali multi-istituzionali e dinamiche.”

L'idea alla base del grid computing è quella di costituire delle organizzazioni virtuali, le quali collaborino condividendo in maniera coordinata risorse di calcolo e di storage, sfruttando reti di comunicazione ad alta capacità [1].
Tali organizzazioni vengono definite, in terminologia Grid, Virtual Organizations (VO).

1.2 Organizzazioni Virtuali

Nel Grid la condivisione di risorse non si limita allo scambio di file, ma permette l'accesso diretto a computer, software, dati ed altre risorse, in modo da poter sviluppare processi di collaborazione tra differenti soggetti e tipi di attività. Questo richiede un alto livello di controllo, infatti vengono definite precise regole attraverso le quali deve svolgersi la condivisione, in modo da definire in maniera chiara cosa viene condiviso e chi ha i permessi per condividere.
Un insieme di individui, organizzazioni o istituzioni che condividono queste regole è definito Organizzazione Virtuale (Virtual Organization, VO). Da questa definizione si comprende come più Organizzazioni Virtuali possano variare tra loro, le differenze possono comprendere i più svariati aspetti: lo scopo, la dimensione, la durata, il tipo di struttura. Allo stesso tempo una singola Organizzazione Virtuale è un'entità estremamente dinamica, in quanto non solo la natura ma anche il numero dei partecipanti alla VO può cambiare, la dinamicità delle relazioni di scambio che si vengono a creare richiede dei meccanismi per scoprire quali relazioni esistono in un particolare momento ed il loro tipo, infatti quando un nuovo utente entra in una Organizzazione Virtuale dovrebbe poter conoscere a quali risorse ha diritto di accedere e le regole a cui sottostare per eseguire l'accesso. Inoltre una stessa risorsa presente nella VO può essere usata in diversi modi in base allo scopo della condivisione ed alle regole che la controllano.

1.3 Grid Computing e l'Open Grid Services Architecture (OGSA)

Sicuramente l'organizzazione di risorse così eterogenee, richiede un notevole sforzo di standardizzazione.

Tra il 1999 e il 2000, i principali gruppi di ricerca che cominciavano ad occuparsi di Grid Computing, hanno dato vita al Global Grid Forum (GGF); un'organizzazione costituita da centinaia di ricercatori e gruppi di lavoro con lo scopo di sviluppare nuovi standard e coordinare gli sforzi di ricerca.
Il primo passo è stato quello di definire una struttura a strati.
I due livelli più bassi altro non sono che l'hardware utilizzato: le architetture per il calcolo e per lo storage comunicano tra loro tramite la comune infrastruttura di comunicazione (internet).
Ciò che permette agli utenti ed alle loro applicazioni di vedere il grid come un'unica infrastruttura di calcolo uniforme indipendentemente da qualunque aspetto architetturale è l'ampio strato di middleware.

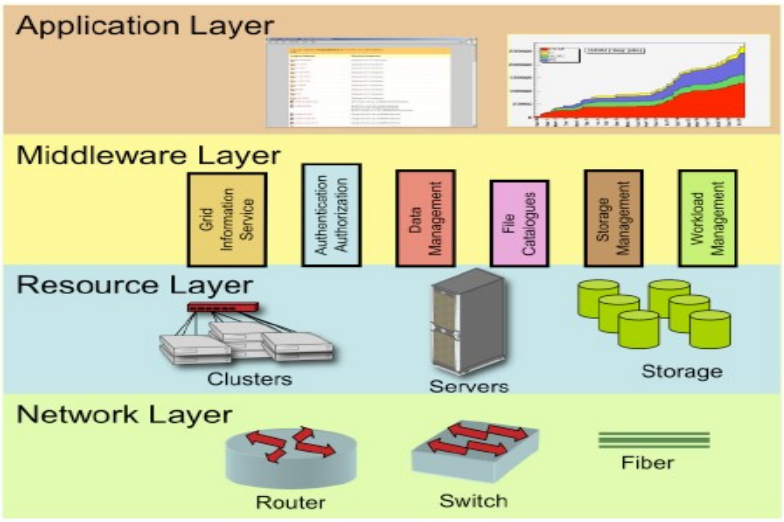


Illustrazione 1: architettura a stati del grid

OGSA identifica otto categorie di servizi a livello middleware, all'interno delle quali sono state individuate e definite interfacce di programmazione di comunicazione che, se implementate da tutti i sistemi, consentono l'integrazione delle Virtual Organization.
Le categorie identificate nell'architettura OGSA, così come vengono pubblicate nel documento ufficiale del GGF [2] sono:
• *Infrastructure Services*
Servizi di intercomunicazione tra le risorse condivise.
• *Resource Management Services*
Controllo, allocazione e distribuzione delle risorse.
• *Data Services*
Trasferimento dati, inclusa la conversione.
• *Context Services*
Descrizione e politiche di utilizzo delle risorse in funzione del contesto e dell'utilizzatore.
• *Information Services*
Servizi di informazione sullo stato ed il funzionamento della griglia.
• *Self-Management Services*
Gestione della efficienza e della complessità del sistema.
• *Security Services*
Servizi di sicurezza ed autenticazione all'interno della Virtual Organization.

• *Execution Management Services*
Gestione dell'esecuzione delle applicazioni.

Alla base delle proposte del GGF, si trova sempre il tentativo di favorire standard consolidati, semplici e già largamente implementati, specialmente per quanto riguarda i servizi di comunicazione intragrid.
L'intera struttura di OGSA è stata interamente progettata per favorire lo sviluppo di applicazioni realizzate tramite l'integrazione di componenti che interagiscono in maniera dinamica e variabile.

Nell'approccio grid indicato dal GGF, ogni problema viene scomposto in servizi atomici, che vengono attivati sotto il coordinamento di un Grid Manager. Il Grid Manager interpreta le richieste dell'applicazione, distribuendone le necessità sui provider di servizio disponibili.
L'accesso ai servizi avviene, secondo l'indicazione di OGSA, tramite interfacce di programmazione che ogni servizio deve esporre per poter entrare a far parte attivamente del sistema di griglia.
La comunicazione fra un servizio e l'altro invece, avviene attraverso i protocolli di trasporto consolidatisi nel mondo Internet, in particolare il TCP.
Ognuno dei servizi può rappresentare sia un'operazione concettualmente atomica, sia una tradizionale procedura software che elabora dati producendo un risultato.
Negli ultimi si è spinto per lo sviluppo di ***middleware*** che implementi le funzionalità di grid aderendo alla specifica OGSA.
Tramite l'introduzione di un ***middleware*** di grid, si riescono ad affrontare e risolvere indipendentemente dall'applicazione problematiche di allocazione delle risorse, autenticazione degli utenti e dei servizi, sicurezza.

1.4 Globus nell'architettura OGSA

Globus è il più grosso progetto per la realizzazione di middleware di griglia sviluppato secondo gli standard OGSA.

Globus costituisce quello che viene denominato un hosting environment, cioè un ambiente software di esecuzione all'interno del quale i servizi vengono istanziati. L'hosting environment definisce il modello ed il linguaggio di programmazione, gli strumenti di sviluppo e mette a disposizione le librerie necessarie all'implementazione delle interfacce adeguate alla semantica del Grid.
Tale semantica, come si è detto, è definita da OGSA: interfacce di creazione, controllo e comunicazione dei servizi. OGSA non definisce requisiti su come un servizio viene eseguito o sugli algoritmi.
Globus è basato sul sistema operativo ospite quale ambiente di esecuzione. Di conseguenza istanzia processi in risposta alle richieste di servizio. É dunque il sistema operativo a gestire l'esecuzione dei servizi, mentre il Globus Toolkit mette a disposizione i comandi necessari all'attivazione e al controllo del loro ciclo di vita.
In tale contesto, la disponibilità di innumerevoli linguaggi di programmazione, offre una molteplicità di possibilità per l'implementazione dei servizi. Ogni linguaggio che disponga di un compilatore o di un interprete compatibile con il sistema operativo scelto come hosting environment può essere utilizzato.

É necessario in ogni caso scegliere un linguaggio per cui sia disponibile una libreria che implementi la semantica dell'ambiente Grid.
La scelta implementativa di Globus è legata ad una filosofia opensource ed altamente orientata ad applicazioni scientifiche, che perciò richiedono grande efficienza nell'esecuzione.

1.5 I componenti di Globus

L'architettura di Globus è organizzata a componenti sovrapposti secondo la struttura a strati rappresentata in figura. I componenti di ogni strato condividono caratteristiche comuni e completano le funzionalità dei livelli inferiori esattamente come accade nel modello a strati del TCP/IP.

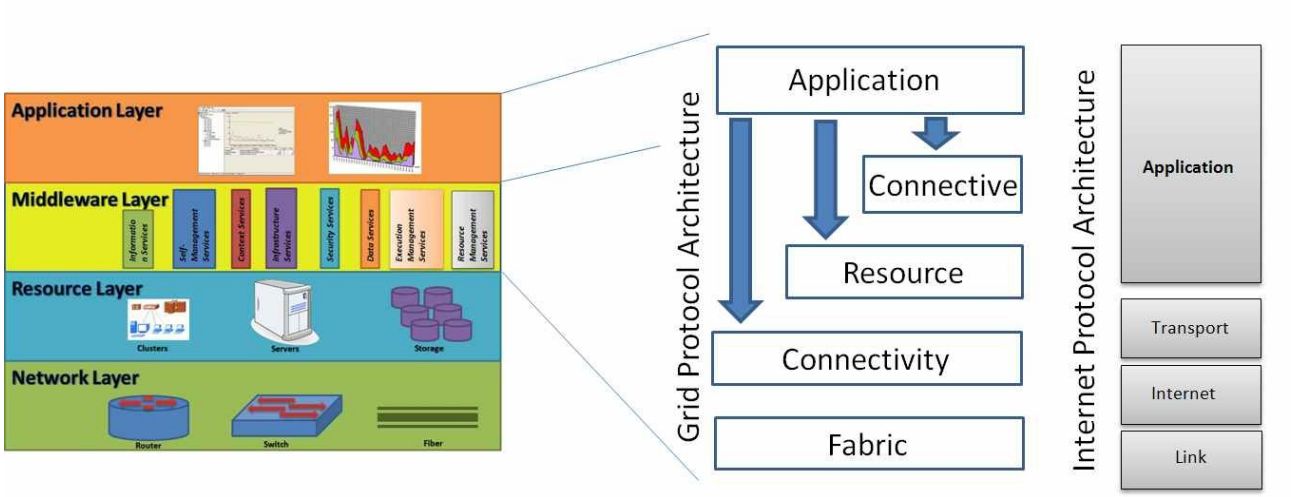


Illustrazione 2: espansione dell'apllication e del middleware Layer

I livelli nei quali è organizzato il middleware Globus sono:
• Application
• Collective
• Resource
• Connectivity
• Fabric

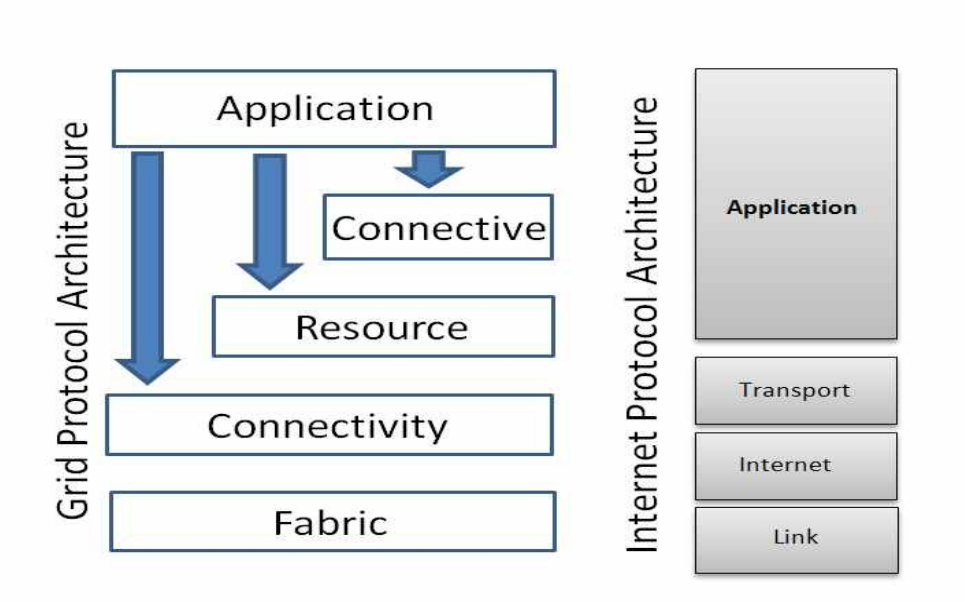


Illustrazione 3: struttura a livelli del middleware Globus

Il livello *Fabric* fornisce l'accesso condiviso alle risorse di calcolo, alle memorie, ai cataloghi, alle risorse di rete ed ai sensori.
Il livello *Connectivity* definisce il nucleo del protocollo di autenticazione e comunicazione richiesto per le transazioni speciche di Grid sulla rete. I protocolli di comunicazione permettono lo scambio di dati tra risorse a livello Fabric.
Il livello *Resource* fornisce un set di protocolli, come ad esempio il GridFTP, una versione estesa di FTP che prevede l'uso di protocolli di sicurezza del livello Connectivity, l'accesso parziale ai file e la gestione del parallelismo dei trasferimenti ad alta velocità.
Il livello *Collective* fornisce protocolli e servizi che gestiscono le interazioni fra collezioni di risorse, come i servizi di Data Replication che attraverso la duplicazione dei dati, consentono di massimizzare le performance di accesso, diminuendo il tempo di risposta ed il costo, e aumentando l'adattabilità.
Il livello *Application* comprende le applicazioni dell'utente che opera all'interno dell'Organizzazione Virtuale.

Complessivamente si può individuare, nella sovrapposizione degli strati, una struttura a clessidra.
Il punto di forza di questo modello architetturale, infatti, consiste proprio nella possibilità che molteplici applicazioni - allo strato più alto - e molteplici software per il controllo diretto delle risorse - allo strato più basso - siano connettibili tramite pochi protocolli di trasporto.
In generale, il posizionamento degli strati dell'architettura di Globus, segue l'andamento degli strati del protocollo TCP/IP.
Chiave di un buon funzionamento di tutto il sistema in ambito molto eterogeneo, è il numero ridotto di protocolli definiti nello strato di connettività e la loro collaudata solidità e diffusione.

CAPITOLO 2

IL MODELLO PER IL COMPUTING DI CDF

Introduzione

In questo capitolo si descrive il modello di computing utilizzato dall'esperimento CDF - Collision Detector at Fermilab.

Gli eventi fisici che avvengono all'interno del Tevatron vengono ricostruiti a partire dai dati di basso livello registrati dai rivelatori.

Nel cosiddetto “raw data recording” tutti gli eventi che soddisfano certe richieste vengono registrati e catalogati come possibili eventi significativi e suddivisi in funzione del tipo di evento che ci si aspetta di aver registrato.

L'architettura di calcolo utilizzata per elaborare ed analizzare questi dati prelevati dal “raw data recording” era composta da grossi cluster Linux, la cosiddetta CAF Cdf Analysis Farm.
Nell'ultimo anno e mezzo a causa delle sempre maggiori necessita' del CDF non piu soddisfatte dalle sole proprie risorse disponibili si è tentato di passare a una CAF distribuita su Grid.

Dopo una breve descrizione dell'architettura per la raccolta e catalogazione dei dati verra' descritta la struttura della CAF per poi passare a descrivere come questa è stata riadattata per poter accedere alle risorse messe a disposizione dal Grid.

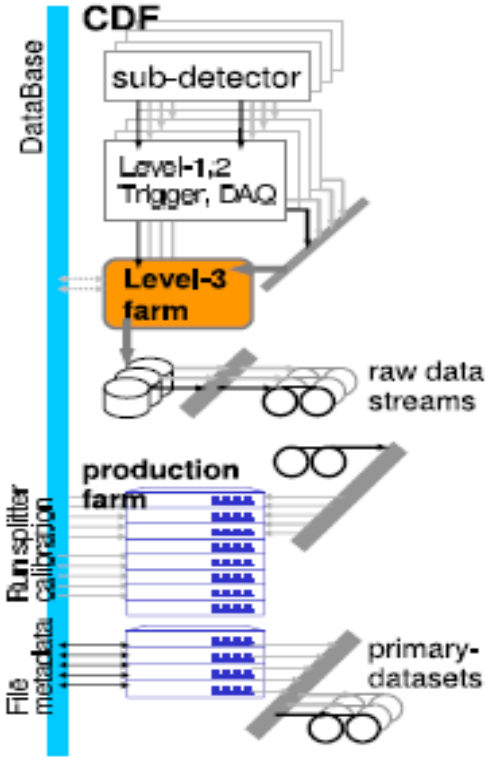
2.1 Raw Data Recording

Il Tevatron, il detector utilizzato dal CDF, è uno dei piu' grandi al mondo e unisce a sensori per la ricostruzione delle traiettorie delle particelle un sistema calorimetrico per la misura dell'energia rilasciata dalla singole particelle prodotte nelle collisioni protone anti-protone.

I dati raccolti a seguito degli scontri di protoni con antiprotoni vengono in primo luogo suddivisi in 3 insiemi detti “Run”. Ogni “run” corrisponde a un periodo di tempo, di solito di circa 12 ore, dopo il quale la qualita' dei fasci di protoni e antiprotoni si e' degradata e risulta poco utile per la produzione di eventi rari.
Per selezionare solo i dati piu interessanti cioe' quelli contenenti eventi utili per la verifica dei modelli fisici vengono utilizzati 3 “filtri” online, chiamati in gergo tecnico “livelli di trigger”.

In figura si mostra il flusso dei dati dal trigger alla farm di analisi.

Illustrazione 4: i dati misurati dei vari sub-detector vengono spediti nella pipeline composta dai tre livelli dove vengono



selezionati in modo progressivo.Solo nell'ultimo livello, il livello3, l'evento viene ricostruito per intero. Gli eventi ulteriormente selezionati vengono registrati in nastri permanenti.Il livello 3 è costiuto da farm di calcolo (CAF).

A livello 1 e 2 i dati vengono selezionati in funzione delle sole quantita' misurate direttamente dai vari rivelatori

Se i requisiti base non sono soddisfatti l'evento e' rigettato. Gli eventi che passano i primi due livelli vengono poi trasferiti al livello 3 dove sono elaborati da algoritmi di ricostruzione online che permettono selezioni piu' raffinate.
Gli eventi che superano il livello 3 sono poi scritti su appositi supporti di storage cosi' come vengono trasmessi dal livello 3. Successivamente questi dati vengono ricostruiti utilizzando risorse dedicate a questo processo e poi scritti e catalogati su nastro.
Gli eventi ricostruiti sono usati dai fisici per le analisi di processi che permettono di capire e verificare I modelli fondamentali della fisica della particelle.

Stream	trigger contents	
	event size (kByte)	ratio to total size (%)
A	monitoring	
	195	2.5
B	high E_T leptons, missing E_T	
	140	11.5
C	high E_T photons, di-photons	
	122	13.0
D	study stream	
	344	11.2
E	tau, lepton pair, Z to bb, higgs	
	146	16.0
G	QCD jet, Di-jet, Miss E_t plus jets, high P_T b-jet	
	139	11.2
H	B to $\pi\pi$, B_s to $D_s\pi$	
	136	24.0
J	J/ψ to leptons, Υ to leptons	
	146	10.6

Tabella 1: Statistiche dei dati raccolti a livello 2 nel 2005.Vengono indicate la grandezza media dei dati per evento e la percentuale del volume dei dati raccolti per ogni data stream sul volume totale

2.1.1 Requisiti di calcolo

Le Farm hanno lo scopo di ricostruire gli eventi a partire dai dati registrati a livello 1 e 2.

Gli eventi da ricostruire sono tra loro indipendenti, nel senso che posso essere ricostruiti senza che ci sia bisogno di informazioni da parte di altri eventi. Questo fa si che l'architettura di calcolo per il CDF possa avvantaggiarsi senza difficolta' del calcolo parallelo: ogni attivita' puo essere facilmente suddivisa in varie sezioni parallele indipendenti.

Le architetture utilizzate dal CDF data processing facility sono per lo piu dei multi core con sistema operativo “Scientific Linux”, una distribuzione appositamente implementata e mantenuta da FermiLab e Cern.

A partire dal 2001 fino al 2004 il CDF ha raccolto dati con un throughput di 20 Mbyte/sec che negli ultimi anni sono diventati 40 Mbyte/sec che corrispondono a circa 25 milioni di eventi registrati ogni giorno.

I dati raccolti venivano precedentemente elaborati da opportuni batch system sviluppati appositamente dal FermiLab e denominati Farm Processing System (FPSNG) [5] poi diventati Condor batch system nel passagio da risorse dedicate a Grid.
Solo per dare un'idea delle risorse di calcolo necessarie basta considerare che fino al 2006 il CDF utilizzava per le proprie attivita' di ricostruzione ed analisi degli eventi circa 5000 CPU, stima non piu fattibile al giorno d'oggi che si è passati al grid.

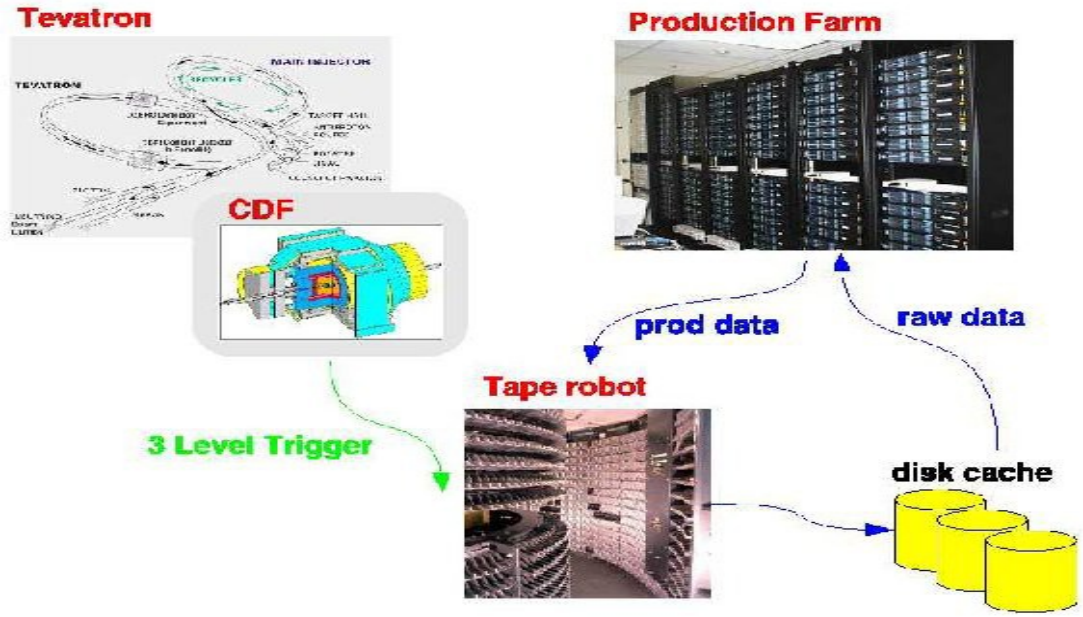
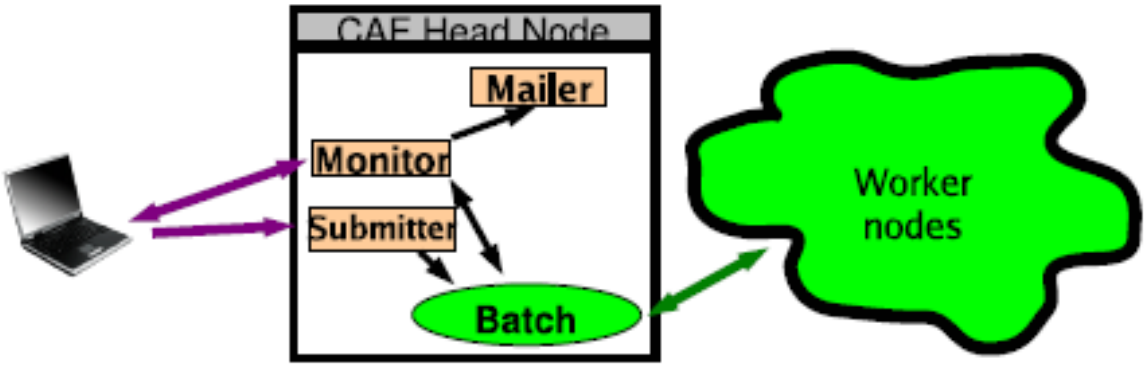


Illustrazione 5: La production farm

2.2 La Cdf Analysis Farm (CAF)



è stata

i debba

Nella CAF ogni elemento ha installato del software che gli permette di interfacciarsi al pool di risorse in modo totalmente trasparente rispetto alle caratteristiche hardware dell'elemento stesso, e che gli permette di accedere al software del CDF, al data management sistem ed ai databases del CDF.

La CAF è basata sull'idea che l'utente scriva ed effetui il debug dei suoi codici sul proprio desktop e, una volta testati, invii i job. I job vengono inviati sotto forma di tarball .tgz comprensivi del codice sviluppato dall'utente e dei dati di ingresso al codice. Il portale della CAF si occupa di suddividere il job in sezioni parallele ed assegnare ogni sezione ad una risorsa del pool.

Il protocollo per il l'invio di job cosi come quello per l'assegnazione delle risorse è stato implementato ad hoc.

Illustrazione 6: CAF head node

Nel caso in cui il job sia stato sottomesso con successo alla CAF l'utente riceve un identificativo del job che potra essere utilizzato dell'utente per monitorarne lo stato di avanzamento del proprio job. Il job puo' essere monitorato tramite un'interfaccia web.
I risultati dei job a fine elaborazione vengono poi inviati all'utente in uno spazio opportunamente indicato dall'utente, indicandolo nel comando CafSubimit nel comando con un tarball comprensivo di output, eventuali messaggi di errore codice originario e tutti gli altri file di ausilio inviati assieme al codice originale.

```
<fcdflnx4.fnal.gov- 11:50:49> CafSubmit
Using python
WARNING: Python C API version mismatch for module krb5:
  This Python has API version 1010, module krb5 has version 1007.

usage: CafSubmit --tarFile=file --outLocation=file --procType=ptype [--start=num --end=num | --sections=startnum-endnum] [--maxParallelSec=num] [--email=addr
] [--dhaccess=method] [--dataset=id] [--farm=name] [--group=group]  command

       --tarFile=file: path for tar file to be submitted (e.g. ./submitme.tar.gz)
   --outLocation=file: full path for output file (e.g. me@ncdfxx.fnal.gov:/home/me/out.tgz)
     --procType=ptype: desired process type (e.g. short)
          --start=num: beginning segment number (e.g. 1)
            --end=num: ending segment number (e.g. 100)
 --sections=start-end: segment range (e.g. 1-100)
   [--maxParallelSec]: max parallel running section number (e.g. 30)
       [--email=addr]: optional email address for summary output
  [--dhaccess=method]: method for dataset acess, options are SAM, DFC, MCGen, rootd, and None
      [--dataset=id]: dataset ID (e.g. hbot0h), only leave blank if not accessing data!
       [--farm=name]: option sets farm name, as opposed to setting CAF_CURRENT
                      environment variable, (e.g. cafcondor
     [--group=group]: special resource group (default is the common group)
             command: command to be executed, normally a shell script (e.g. ./my.sh)

<fcdflnx4.fnal.gov- 11:50:57>
```

```
<fcdflnx4.fnal.gov- 12:14:52> CafSubmit --tarFile=./Transfert_on_grid.tgz --outLocation=canto@fcdfdata113.fnal.gov:icaf/output\$.tgz --procType=test --sectio
n=1-3 --email=canto@fnal.gov --dhaccess=none ./Transfert_on_grid.sh \$
Using python
WARNING: Python C API version mismatch for module krb5:
  This Python has API version 1010, module krb5 has version 1007.

Analysis Farm: caf    Host: fcdfhead3.fnal.gov

JID:  3454213
<fcdflnx4.fnal.gov- 12:15:29>
```

Per garantire l'autenticazione e l'integrita' viene utilizzato un protocollo di autenticazione Kerberos [7].

```
<fcdflnx4.fnal.gov- 12:14:26> CafSubmit --tarFile=./Transfert_on_grid.tgz --outLocation=canto@fcdfdata113.fnal.gov:icaf/output\$.tgz --procType=test --sectio
n=1-3 --email=canto@fnal.gov --dhaccess=none ./Transfert_on_grid.sh \$     comando di sottomissione
Using python
WARNING: Python C API version mismatch for module krb5:
  This Python has API version 1010, module krb5 has version 1007.

Analysis Farm: caf    Host: fcdfhead3.fnal.gov

Error: No credentials cache file found. Please run kinit.     l'utente non possiede un ticket kerberos valido

<fcdflnx4.fnal.gov- 12:14:36> kinit canto     l'utente si autentica e kiede un ticket kerberos
****************************************************************
*** It seems you are about to type your password over the net. ***
*** Are you SURE there's no other way to do this ?             ***
*** If not, is your connection encrypted end-to-end ?          ***
*** Talk with an expert or review the documentation at         ***
*** http://www.fnal.gov/docs/strongauth/html/user.html         ***
****************************************************************

Password for canto@FNAL.GOV:
<fcdflnx4.fnal.gov- 12:14:52> CafSubmit --tarFile=./Transfert_on_grid.tgz --outLocation=canto@fcdfdata113.fnal.gov:icaf/output\$.tgz --procType=test --sectio
n=1-3 --email=canto@fnal.gov --dhaccess=none ./Transfert_on_grid.sh \$
Using python
WARNING: Python C API version mismatch for module krb5:
  This Python has API version 1010, module krb5 has version 1007.

Analysis Farm: caf    Host: fcdfhead3.fnal.gov

JID:  3454213                 Ora che l'utente possiede un ticket kerberos puo mandare job alla CAF
<fcdflnx4.fnal.gov- 12:15:29>
```

2.2.2 come funziona la CAF

La CDF Analysis Farm (CAF) è stata sviluppata come un portale. Un insieme di demoni accetta richieste dagli utenti. Queste richieste vengono poi convertite in comandi per il batch system sottostante che si occupa poi del lavoro reale.

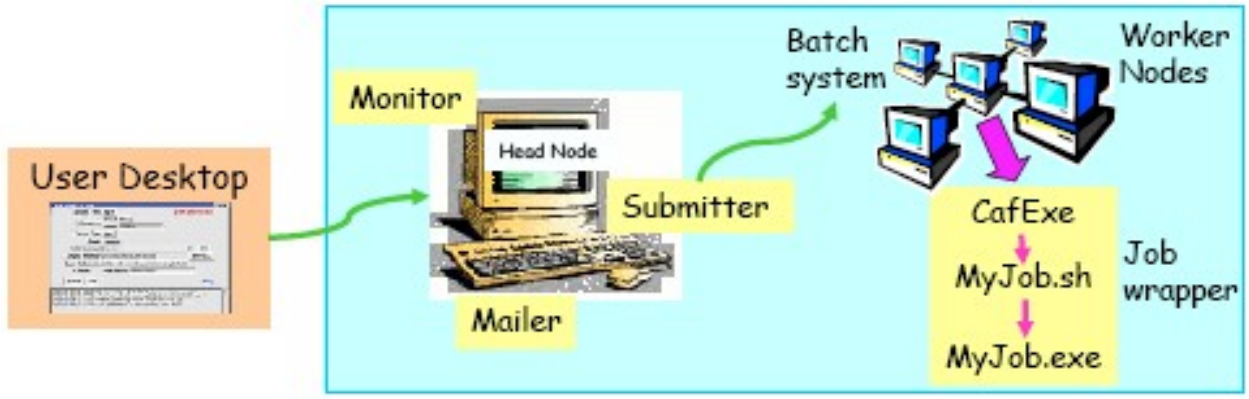


Illustrazione 7: overview del portale CAF: lo head node, dove i demoni sono in esecuzione e i worker node dove sono attivi i wrappere

- *Il demone di sottomissione :* accetta il job dell'utente, si occupa poi di inviare ogni sottosezione al batch system dopo aver creato i files di submission necessari per ogni specifico batch sistem.
  Se il job viene inviato con successo al batch system il demone di sottomissione restituisce all'utente l'id del job.

- *Il demone di Monitoring:* Il monitoring ha due componenti: “interactive monitoring” e “classical batch monitoring” l'interactive monitoring permette agli utenti sia di interagire con il job sia di controllarne lo stato. Ogni utente puo controllare la lista dei propri job siano essi in esecuzione, pendenti o gia terminati.Gli utenti possono inoltre visualizzare il contenuto della directory remota del working node dove il job scrive il suo output e leggere il log-file e lo error-file. Oltretutto gli utenti possono interagire con il job per metterlo in pausa, effettuare il debug o farlo terminare forzatamente mentre è in esecuzione. Il servizio di monitoring tramite web permette di visualizzare alcune informazioni utili riguardo tutti i job di tutti gli utenti.

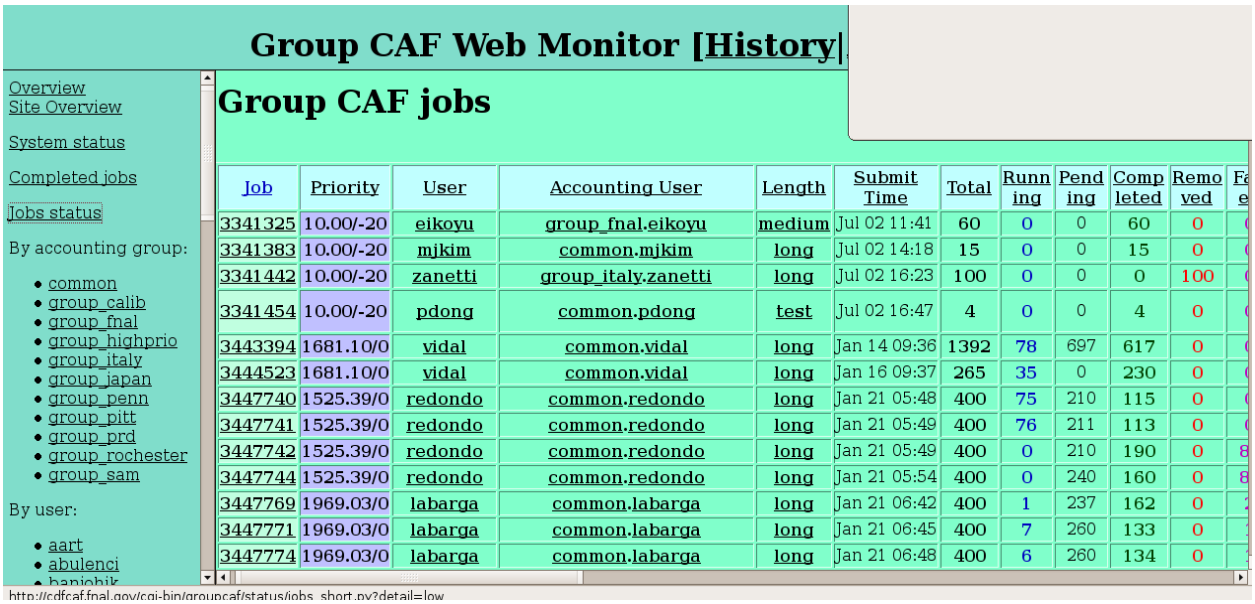


Illustrazione 8: Pagina del Web monitoring

- *Il mailer daemon :* monitora lo stato del job e quando il job finisce la sua esecuzione avvisa l'utente con una mail. Tra le informazioni inviate all'utente ci sono: lo stato di terminazione del job, se è terminato correttamente oppure se è stato abortito; l'id del batch sistem che ha eseguito il job e le sue sottosezioni.

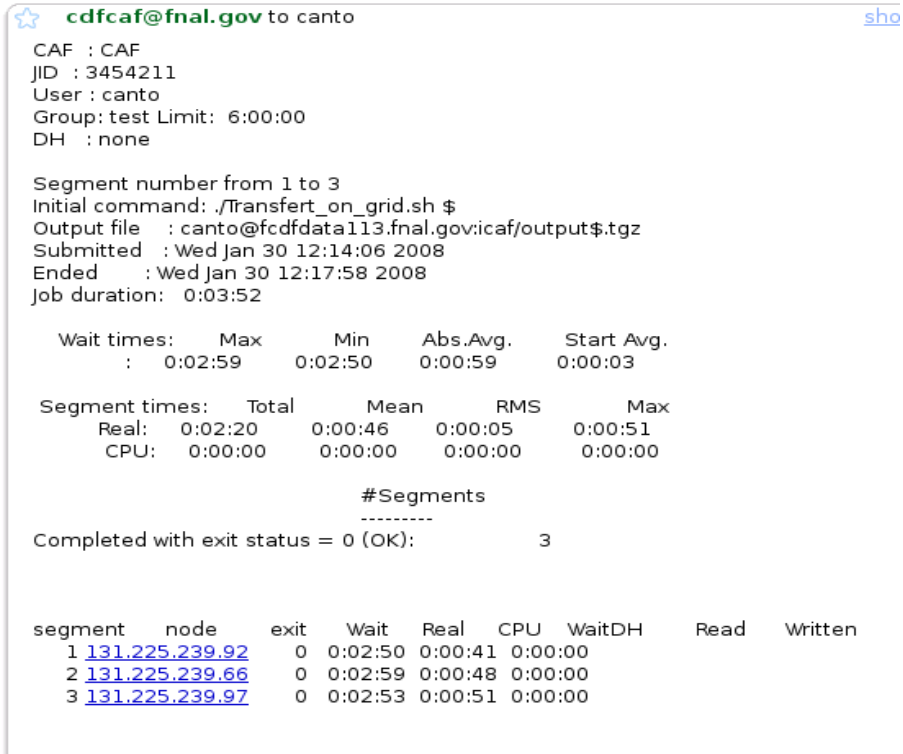


Illustrazione 9: Mail di rapporto sull'esito del job

Ogni macchina con accesso alla caf ha un wrapper. Tale wrapper (CafExe) gira nel worker node, scompatta il tarball inviato dall'utente e suddivide il job in tante sezioni parallele quante sono state indicate dall'utente. Tale wrapper si occupa anche di effettuare alcune operazioni di monitoring come la creazioni di job summary files. Quando il job termina la sua esecuzione il CafExe di occupa di spostare I file di uscita e di log prodotti dal Worker Node remoto dove il job era in esecuzione in una zona specificata dall'utente, eliminando poi questi file dal worker node remoto.

2.2.3 Verso il grid

Come detto il primo approccio al calcolo distribuito del CDF era basato su un pool di risorse gestite e mantenute dal CDF: la CAF.
Una prima evoluzione verso un'architettura distribuita geograficamente è stata la dCAF (distribuited CAF).
I vari nodi della dCAF erano ospitati nelle diverse sedi, sparse per il mondo, dei gruppi che collaboravano al CDF.
Continuava pero a trattarsi di risorse dedicate anche se la manutenzione e la gestione dei nodi veniva ora demandata a livello di sede.
L'architettura mostro presto i suoi limiti in quanto diventava economicamente sempre piu oneroso tenere il passo con l'aumento della necessita' di risorse per il calcolo e per l'allocazione dei dati.

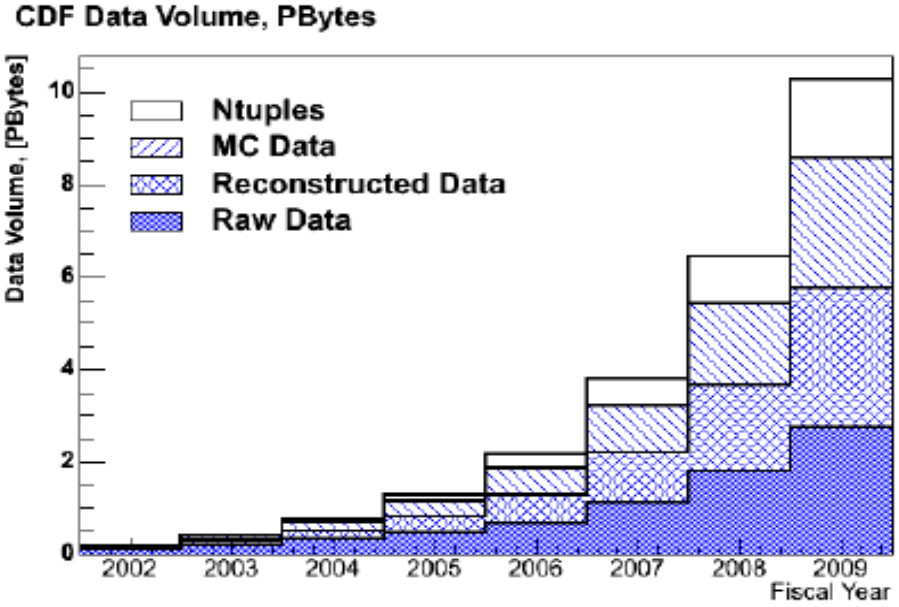


Illustrazione 10: trend dell' incremento del volume dei dati

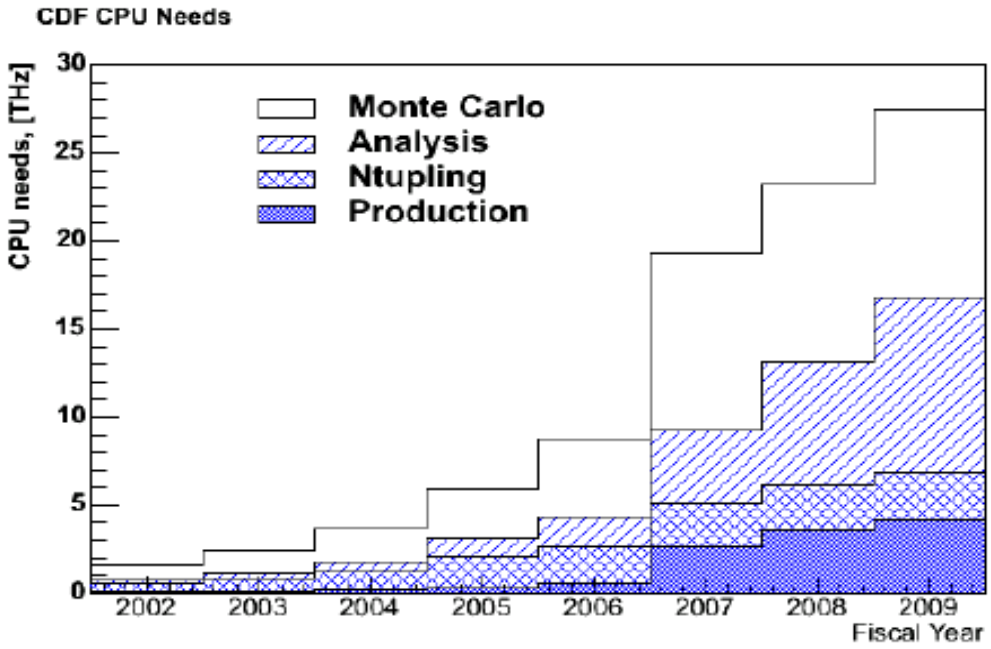


Illustrazione 11: trend dell'aumento delle necessita' di calcolo

Invece che continuare a far crescere le dimensioni della CAF il CDF decise nel 2006 di muoversi verso il grid e riorganizzare la CAF affinche' potesse funzionare su in ambiente grid.
Il vantaggio è stato quello di poter utilizzare fin da subito le grandi risorse di calcolo che la comunita' scientifica ha messo a disposizione per l'esperimento LHC (Large Handron Collider) di Ginevra.
Seppur tale esperimento non sia a tutt'oggi in funzione è gia possibile per i vari esperimenti e centri di ricerca minori utilizzare le sue risorse di calcolo che altrimenti resterebbero inutilizzate.
La possibilita di passare da un pool “privato” ad ambiente grid è stata la chiave del successo e della longevita' della CAF che si è saputa adattare, con cambiamenti non troppo bruschi, alle esigenze della sua utenza.

2.3 Overview delle soluzioni proposte per il passaggio al Grid

Nel passaggio al grid ci si è trovati di fronte alla scelta di quale modello adottare.

I modelli adottati nel grid per gestire le risorse sono infatti molti. Le due principali scuole di pensiero sono quella Nord Americana e quella Europea.

Il modello Nord Americano richiama l'idea di un cluster virtuale con i vari nodi sparsi nel mondo.

Il modello per la gestione delle risorse è di tipo pull, come verra spiegato nel seguito.

Il principale batch system utilizzato a livello di nodo è il Condor Batch sistem [6]. Questo modello risulta essere piu efficiente rispetto a quello europeo per quel che riguarda la gestione delle risorse, è pero meno scalabile.

Nel modello Europeo invece si cerca gestire le risorse nella maniera piu scalabile possibile con un arbitro che gestisce il matching tra risorse e job. Il modello per il matching tra risorse e job è di tipo push, questo porta a qualche decremento delle prestazioni per quel che riguarda la gestione delle risorse a cambio di una maggiore scalabilita'.

Qui nel seguito si spiegano in linea teorica le principali modifiche apportate alla CAF per poter funzionare sia sul grid americano (la Condor based GridCAF) sia sul grid europeo (la gLite WMS based GridCAF). Successivamente verranno poi illustrate le effettive implementazioni della CAF sia per il grid europeo (LcgCAF) si per quello Nord Americano (NamCAF).

2.3.1 Condor based GridCAF (o GlideCAF)

Tutti i nodi che fanno parte di uno stesso pool di risorse hanno condor pre-installato e quando un nodo vuole aggiungersi al pool è necessario abbia installato il middleware di Condor [6].

La Grid CAF basata su middleware Condor utilizza la stessa idea proposta nel modello americano: vengono creati dei pool di risorse virtuali a partire delle risorse del grid.
Vien definito a questo scopo l'idea di “processo demone Condor” o anche Glide-in. Il “processo demone Condor” non è altro che un processo demone che vien inviato ai vari nodi del grid che dispongono del middleware necessario per unirsi al pool virtuale condor based, una volta avviato tale processo non fa altro che far unire il Worker Node al pool di risorse virtuali, dichiarare la risorse del nodo dove si trova e prendere job dalla coda dei processi in attesa.
L'invio dei job contenenti processi demoni condor avviene per mezzo di un' entita' chiamata Glidekeeper, il Glidekeeper è anch'esso un processo demone che viene avviato su una macchina del pool e invia job contenenti processi demoni condor con lo scopo di attivare i vari nodi del grid.
Il Glidekeeper si appoggia al middleware Globus che è uno standard del grid.
Tutti i nodi appartenenti al grid utilizzano i tool del “middleware Globus” sia che possano unirsi ad un pool di risorse Condor sia che no.
Quando su un nodo arriva un job glidein i processi demoni condor vengono risvegliati ed il nodo, se opportunamente configurato (ossia con condor installato), si unisce al pool di risorse virtuali e preleva jobs dalla coda dei processi in attesa.

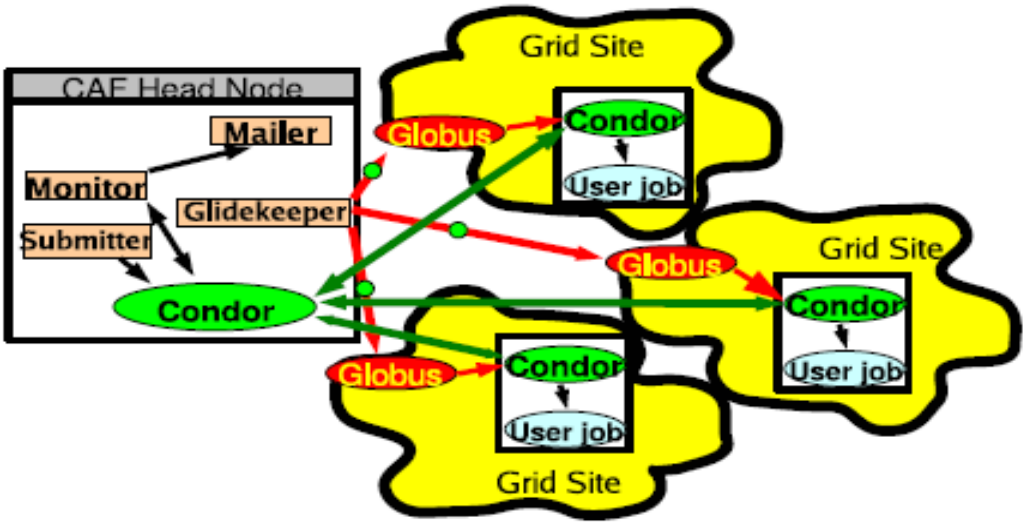


Illustrazione 12: condor besde GridCAF

Il vantaggio di questo approccio è che l'infrastruttura grid viene totalmente mascherata da Condor.
La coda dei job in attesa è comune sia per le risorse locali che per le risorse di griglia ed i job possono essere prelevati indifferentemente sia dalle risorse del pool proprio che da quelle virtuali messe a disposizione dal grid.
Il secondo vantaggio deriva dall'utilizzo di un modello pull per la gestione delle risorse: la CAF non deve preoccuparsi della gestione di code differenziate o di inviare i job ai nodi, saranno i nodi stessi che si occuperanno di scegliere job opportuni rispetto alle loro risorse a disposizione.
Oltretutto poiche' i nodi richiedono volontariamente i job è piu' facile gestire la dinamicita' delle risorse: quando per un qualunque motivo un nodo esce dal pool virtuale smette semplicemente di prelevere job dalla coda, non si rischia in questo modo di inviare job a nodi che non sono momentaneamente disponibili. Questo aumenta notevolmente il “success rate” ossia l'indice di job correttamente terminati rispetto al modello push (infatti col metodo pull se un nodo smette di essere operativo al piu fallisce il job in corso al momento dello shut down).

2.3.2 gLite WMS based GridCAF

La gLite WMS based GridCAF[3] è basata su un modello push. In questa implementazione della CAF il local batch sistem inteso come punto principale di sottomissione dei job viene rimpiazzato dal Grid: il portale CAF si interfaccia direttamente col Workload Management Sistem (WMS) del middleware del grid, detto anche Resource Broker.
E' il Resource Broker che si occupa di eseguire il matching tra risorse a disposizione e richieste in arrivo.
La scelta di utilizzare un arbitro per l'invio dei job da alcuni problemi nel caso di risorse che vengono

aggiunte o levate dinamicamente. Allo stato attuale la gestione dinamica delle risorse è ancora rudimentale, questo implica che a volte job con determinate necessita vengano inviati a nodi che non hanno piu' le risorse opportune per portarli a termine o addirittura (seppur piu raramente) a nodi che non sono piu presenti nel grid.
Questo rende piu basso il "success rate" del WMS based GridCAF.

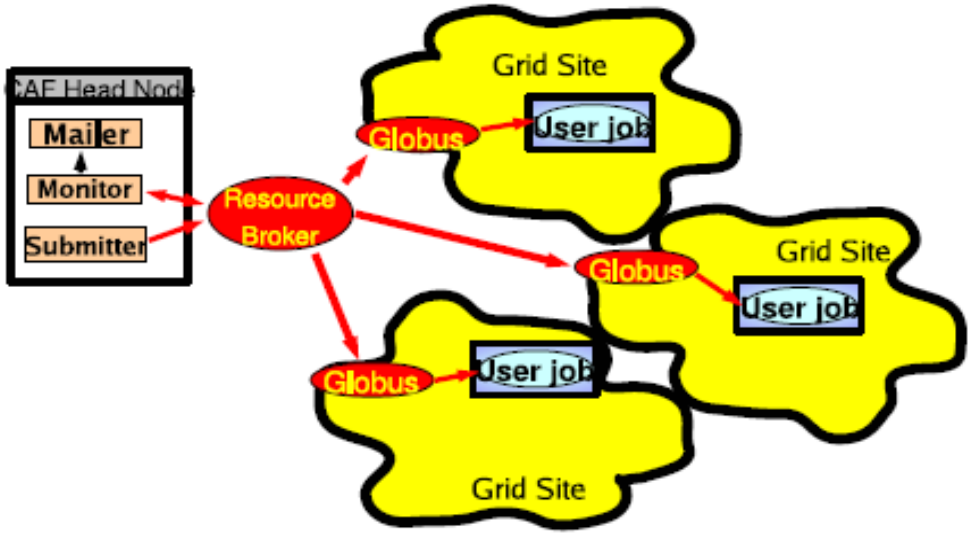


Illustrazione 13: G-lite WMS based GridCaf

Il grande vantaggio della "gLite WMS based GridCAF" è che essa permettere di accedere a praticamente tutti i siti Grid senza richiedere che essi abbiano installato software condor, condizione quest'ultima a volte molto restrittiva.

2.4 Le versioni della CAF per il GRID

Il primo approccio verso una CAF distribuita è stato fatto applicando un modello "Glide-in" ed utilizzando condor come batch sistem.
Il software Condor permette di gestire sia pool di risorse dedicate sia di accedere alle risorse del grid.
Nel modello "Glide-in" i worker node vengono aggiunti dinamicamente ad un pool di risorse aventi tutte condor installato.
Questa reimplementazione della CAF basata su modello "Glide-in" è chiamata GlideCAF ed è stata un notevole successo sia per efficenza che per stabilta'.

In ogni caso la glideCAF da sola non era sufficiente per garantire le necessita' del CDF.
Cosi nell'ultimo anno il CDF si è mosso verso un modello di computing piu distribuito dando il via a due progetti paralleli:

- LcgCAF un portale per accedera alle risorse del grid basato su gLite Workload Management System (WMS), e' su questa versione che si lavorare durante la tesi.
- NAMCAF nata sulla base della GlideCAF, questa verione non verra' presentata.

2.4.1 LCG CAF

LcgCAF è una versione della CAF per la sottomissione dei job degli utenti CDF al Grid Europeo.
Nella LcgCAF è prevista l'esistenza di un head node che funge da punto di sottomissione dei job.

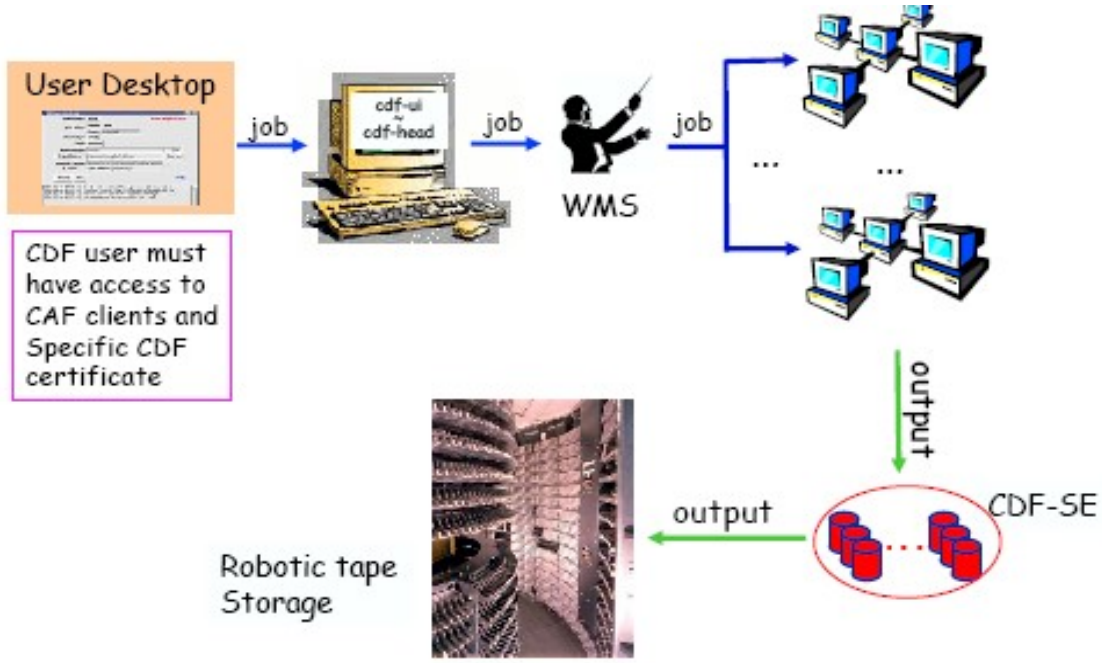


Illustrazione 14: architettura generale della LcgCAF. I job dell'utente passano dal portale, vengono diretti verso il grid submission point e successivamente ai vari siti del Grid. L'output viene poi spedito nei CDF-SE e successivamente scritto su nastro indelebile

Gli utenti possono inviare job da qualunque desktop che abbia accesso al meccanismo standard della CAF per la sottomissione di job.
L'autenticazione grid per gli utenti CDF viene effettuata da dei cron job che girano nello head node e trasformano i ticket kerberos, necessari per l'accesso alle risorse di CDF, in certificazioni X509, il

metodo di autenticazione utilizzato nel Grid.
Il tempo di vita di questa certificazione dipende sia dal setting della LcgCAF che dalla configurazione del VOMS (Virtual Organization Management System)[8].
Attualmente la certificazione Grid dura una settimana, tempo sufficiente per l'esecuzione di una lunga simulazione montecarlo, quando la certificazione scade il job viene terminato.
Una volta che l'utente si è autenticato il client CAF si connette al demone per la sottomissione dei job che inoltra il job al gLite Workload Management System.

A questo punto il Resource Broker, uno degli elementi del WMS distribuisce I job tra i Computing Element (CE) del grid.
I risultati del job vengono poi inviati in opportuni Storage Element (SE) del CDF o copiati in uno spazio definito dall'utente.

A invio ed esecuzione dei job

Come per la CAF anche LcgCAF ospita diversi processi demoni.
I piu importanti tr aquesti son quelli che si occupano dell'invio dei file, del monitoring e della user notification.
La struttura di base rimane quella gia descritta per la CAF ma con funzionalita' diverse.
In figura viene mostrato il portale con le tre principali classi di demoni ed il percorso per l'invio di job nel Grid.

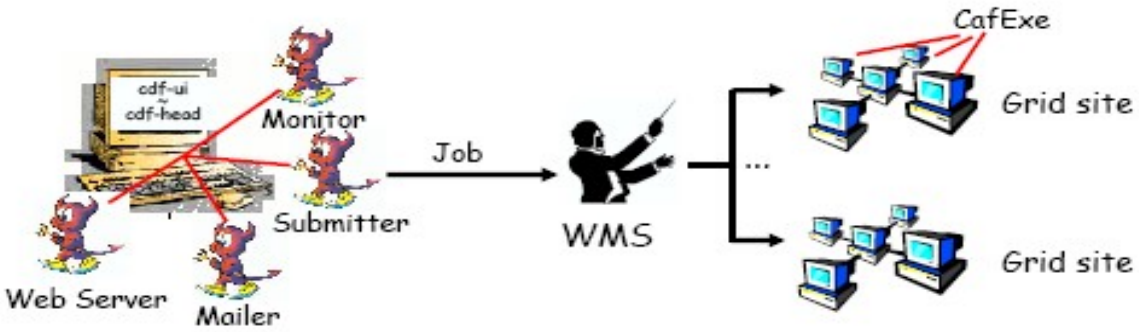


Illustrazione 15: LcgCAF submission and execution .Rappresentazione del portale con I demoni ed il job wrapper in esecuzione sul Worker Node..

Il “submitter” è quel servizio responsabile di accettare le richieste dell'utente di invio dei job e creare un opportuno file JDL (Job Description Language)[9].
La parallelizzazione dei job, che nella CAF originaria viene fatta dividendo il job in vari segmenti, nel grid viene effettuata utilizzando il DAG (Direct Acyclic Graph)
I job non vengono direttamente inviati sul Grid, ma messi in attesa in una coda di sottomissione, questo perche' la versione corrente di WMS impiega tempi molto lunghi per effettuare il matching necessita' del job / risorse disponibili.
Una volta che un job viene inserito nella coda di attesa gli viene assegnato un ID che, seppur il job non

sia ancora stato inviato sul grid, verra utilizzato per scopi di monitoring.
Talvolta successivamente alla spedizione del job nel Grid è necessario cambiare l'ID assegnato al job durante la sua permanenza nella coda di attesa, in questi casi il matching tra il vecchio ed il nuovo ID viene gestito in automatico senza quindi dare difficolta' al servizio di monitoring.
Il servizio che si occupa di prelevare i job dalla coda di attesa e inviarli al WMS si trova, nel modello LcgCAF, sullo head node.
Il WMS successivamente manda il job ad elaborare, la scelte del Computing Element (CE) dipende dal numero di CPU libere: tra tutti i CE che hanno risorse sufficienti per portare a termine il job viene scelto quello col piu alto numero di CPU libere.
Quando tutte le sezioni parallele del job originario sono concluse una mail informativa viene mandata all'utente che ha inviato il job. L'indirizzo a cui mandare la mail viene specificato al momento della sottomissione del job.
In questa mail vengono collezionate tutte le informazioni utili come il nodo che ha eseguito il job, le CPU che hanno eseguito i singoli segmenti, quali segmenti sono andati a buon fine e quali no, se ci sono segmenti che sono stati abortiti e il tempo di CPU.

cdfcaf@fnal.gov to canto sho

CAF : CAF
JID : 3454211
User : canto
Group: test Limit: 6:00:00
DH : none

Segment number from 1 to 3
Initial command: ./Transfert_on_grid.sh $
Output file : canto@fcdfdata113.fnal.gov:icaf/output$.tgz
Submitted : Wed Jan 30 12:14:06 2008
Ended : Wed Jan 30 12:17:58 2008
Job duration: 0:03:52

Wait times:	Max	Min	Abs.Avg.	Start Avg.
:	0:02:59	0:02:50	0:00:59	0:00:03

Segment times:	Total	Mean	RMS	Max
Real:	0:02:20	0:00:46	0:00:05	0:00:51
CPU:	0:00:00	0:00:00	0:00:00	0:00:00

	#Segments
Completed with exit status = 0 (OK):	3

segment	node	exit	Wait	Real	CPU	WaitDH	Read	Written
1	131.225.239.92	0	0:02:50	0:00:41	0:00:00			
2	131.225.239.66	0	0:02:59	0:00:48	0:00:00			
3	131.225.239.97	0	0:02:53	0:00:51	0:00:00			

Illustrazione 16: mail di rapporto sull'esito del job

B MONITOR

*Gli u*tenti CDF possono monitorare lo stato dei job sia con l'interprete dei comandiche da interfaccia Web.

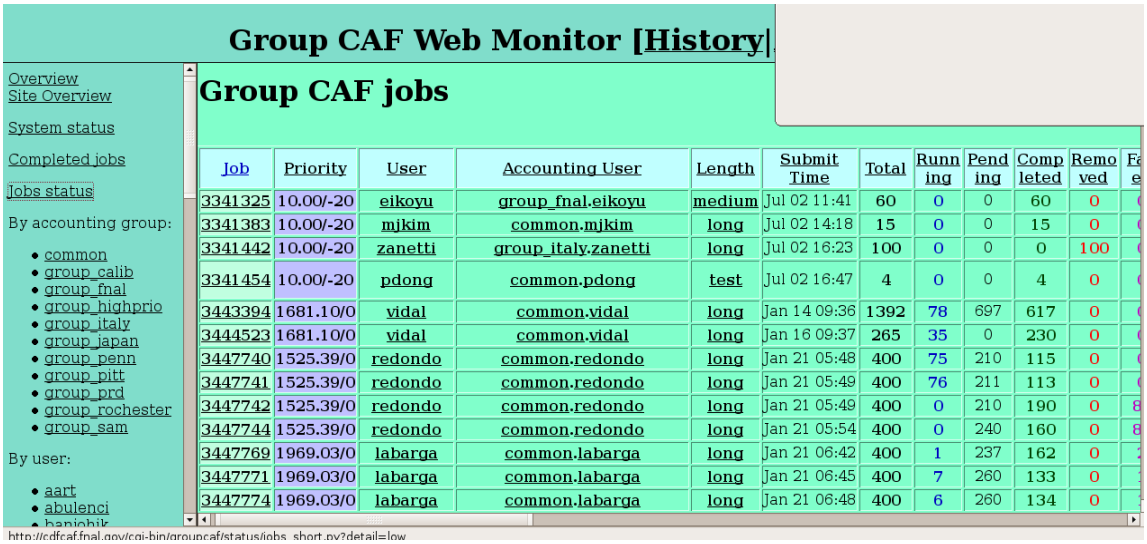


Illustrazione 17: Interfaccia Web per il monitoring

In LcgCAF le informazioni vengono prelevate dal WMS e successivamente dal worker node nella seguente maniera:
Il servizio denominato CafMon provvede a recuperare le informazioni a seguito di queries da parte degli utenti.
Il meccanismo delle queries nella LcgCAF è basato sul WNCollector, un servizio implementato ad- hoc per LcgCAF.
Tale servizio è composto da un server, *WNCollDaemon,* in esecuzione sullo head node ed un client, *WNColl.py,* in escuzione sullo stesso worker node su cui è in esecuzione il job.
Il client periodicamente colleziona diverse informazioni relative al working node, alla CPU ed alla memoria utilizzata, eventuali errori del job ed i file di log relativi al job.
Successivamente queste informazioni vengono trasferite al CDF Information System (CDF-IS) che si trova sullo head node.

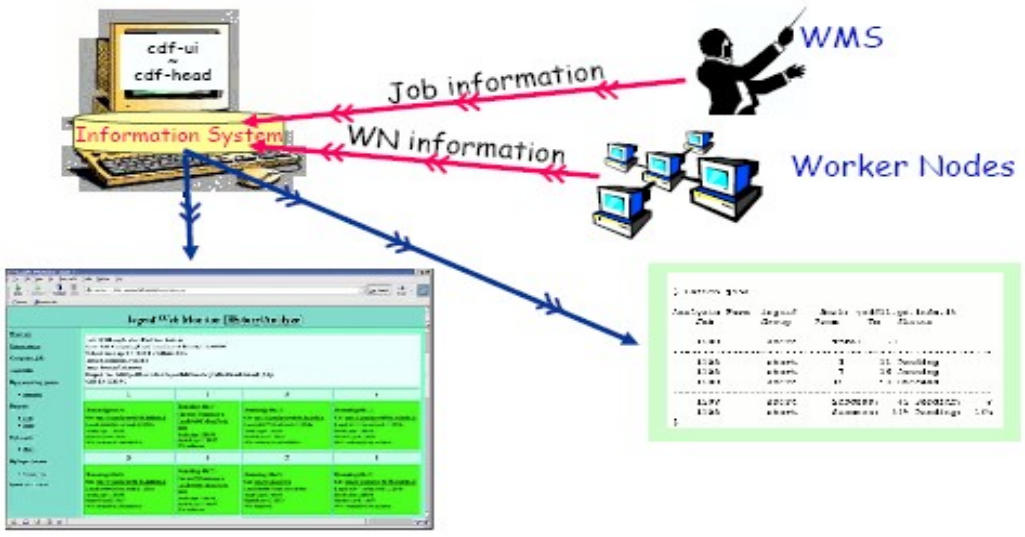


Illustrazione 18: LcgCAF monitors architecture. Le informazioni sui job vengono collezionate dal WMS e dai Worker node e poi inviate al CDF Informantion System. Il CDF Information System puo essere interrogato sia tramite interfaccia WEB che da riga di comando.

Quando un utente invia una richiesta all' information System gli vengono restituite le informazioni in cache in quel momento,il minitoring ha evidentemente un certo ritardo nell'aggiornamento che è tanto maggiore quanto maggiore è il numero dei job di cui tener traccia.
Gli utenti posso interagire con i job solo per forzarne prematuramente la loro terminazione, questa funzionalita' è implementata usando il tool Glite-wms-job-cancel.
Queste informazioni vengono poi trasformate in formato XML dal *xml monitor* che legge le informazioni del CDF-IS e le rende accessibili via Web in un formato rappresentato in figura

C. CDF Code distribution and CDF database access

Per poter essere eseguite correttamente molte simulazioni Montecarlo hanno bisogno di codici studiati ad-hoc dal CDF, le informazioni del Run Condition database ed informazioni sullo stato di configurazione dei detector interni al tevatron.
Sia i codici che le informazioni devono essere consultabili a run-time, ma non è certo auspicabile che tutti i nodi del grid abbiano tali informazioni gia' in loco.
Il Run Condition database contiene informazioni importanti per la ricostruzioni dei dati fisici a partire dai dati del livello 1 e 2 come per esempio la geometria del detector, le configurazioni dei trigger ed altre informazioni molto utili per la ricostruzione degli eventi.
Tutte queste informazioni sono contenute in Databases interni al Fermilab e l'accesso a questi databases rappresenta un grosso collo di bottiglia a causa del numero limitato di connessioni che possono essere accettate contemporaneamente.
Per ovviare a tale problema si è pensato di implementare una soluzione basata sul caching tramite dei

proxy server. In questo modo il carico di richieste al database principale è calato notevolmente.
Anche il software CDF è stato pensato per essere eseguito in ambiente dedicato con accesso ad un ampio insieme di eseguibili, librerie, e files di configurazione.
Anche in questo caso non è possibile pensare che tutto cio' sia gia presente in tutti i nodi del Grid; la maniera piu semplice per replicare il CDF file system nei vari nodi è utilizzare un file sistem virtuale e tecniche di caching.
Il software utilizzato è Parrot [10], un file sistem virtuale di tipo Unix-Like per ambiente remoti.
Ogni volta che un'applicazione esegue una chiamata di sistema l'applicazione si blocca e viene invocato Parrot. Parrot interpreta gli argomenti passatigli dalla chiamata di sistema e si occupa di effettuare una chiamata a sistema remoto utilizzando i protocolli standard quali HTTP, FTP, GridFTP, etc.
Nella versione del CDF parrot usa principalmente il protocollo HTTP, in questo modo anche le chiamate a sistema remoto possono essere memorizzate in chace dal proxy cache.
Utilizzando anche in questo caso un proxy per la precisione Squid è possibile ridurre notevolmente i tempi per l'accesso al codice CDF.

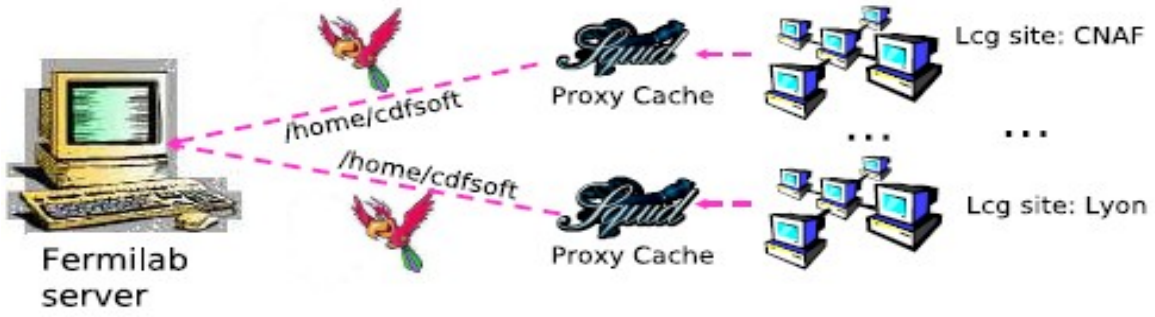


Illustrazione 19: LcgCAF code distribution .Ogni nodo vede il file system del CDF come se fosse sul proprio file system locale.

D Output storage

LcgCAF permette ai suoi utenti di copiare l'output di un job sia su uno Storage Element (SE) del grid opportunamente scelto dall'utente sia all'interno del file server apposito del CDF.
Quando l'output viene copiato in uno Storage Element del Grid l'autenticazione avviene utilizzando il Grid user proxy.
Se l'output viene copiato all'interno del CDF l'autenticazione avviene utilizzando il protocollo kerberos. In quest'ultimo caso è necessario che anche il Worker Node sui quali vengono eseguiti i job possiedano un ticket kerberos valido.
Per far cio si utilizza il servizio *Kdispenser,* questo servizio, ospitato nell'head node, crea un ticket kerberos valido e lo invia al worker node tramite un canale autanticato GSI

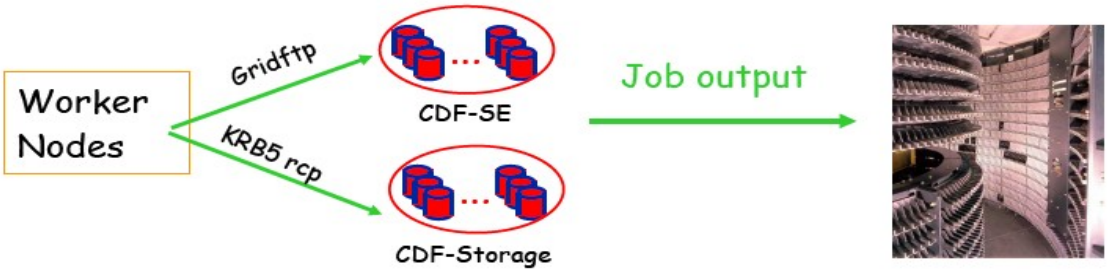


Illustrazione 20: i protocolli utilizzati per spostare l'output possono essere GridFTP se siamo su griglia oppure KRB5 rcp se ci troviamo in locale

Nel caso vengano prodotti un grande numero di file questi file vengono preliminarmente collocati in uno Storage Element intermedio e successivamente inviati a Fermilab dove successivamente, a seguito di una analisi di validazione, verranno catalogati e spostati su nastro indelebile.
Il motivo per il quale viene scelta questa strategia è per impedire che un improvviso picco di traffico in upload possa congestionare la rete.
Questa soluzione è pero insidiosa e presenta delle problematiche ancora aperte come vedremo meglio nel seguito.

3 IL DATA HANDLING

Introduzione

Il problema del quale mi occupo nel mio lavoro di tesi si può catalogare nell'insieme dei problemi di “Data Handlig & Managing Storage”. I problemi di “managining storage” sono tra i principali problemi aperti del Grid computing.

A tal proposito una delle piu' autorevoli riviste del settore iSG (international Science Grid) scrive:

”Allocating storage resources is more complicated than managing compute clusters. Upon completion of a compute job, CPUs are simply returned to the common pool; however, their output must be stored and cannot be deleted indiscriminately.”[1]

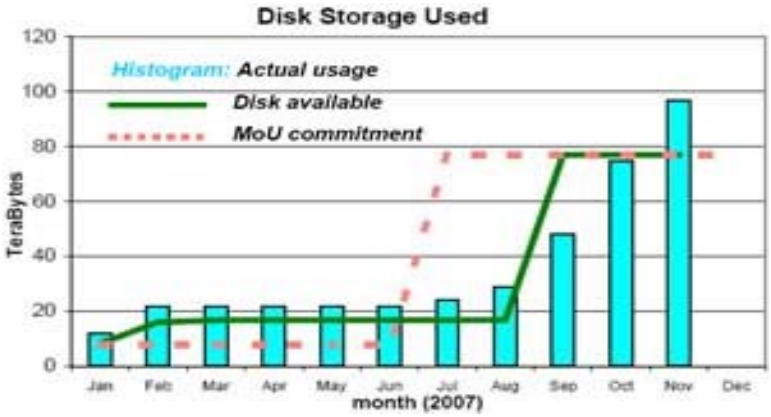


Illustrazione 21: Lo spazio disponibile al Triumph tier1 si è da poco esaurito, ed è stato necessario comprare nuovo hardware. In 3 mesi si è utilizzato l'equivalente di un anno.

Nella prima parte del capitolo si illustra come viene risolto il problema del data handling in 'local' ossia

per il movimento e l'accesso dei file contenuti nei diversi Storage System del Fermilab utilizzando SAM.

Il passaggio al grid ha però riproposto il problema in modo piu' ampio, dato che i file si trovano sparsi nei vari worker node appartenenti a diverse organizzazioni nel mondo. Si accentuano quindi i requisiti di efficienza e sicurezza negli accessi. SAM è stato pensato per lavorare direttamente sopra lo Storage management System e non è stato progettato per funzionare sul Grid, nel prossimo capitolo si descriveranno le modifiche necessarie per funzionare su grid.

Il passaggio al grid porta quindi alla necessità di testare delle nuove versioni di SAM che siano compatibili con il middleware Grid e cercare di provvedere a dei meccanismi che si occupino in modo automatico dello spostamento di questi dati e della loro catalogazione, gestendo eventuali repliche, garantendo la coerenza tra le diverse repliche dello stesso documento, e tenendo traccia della locazione nella quale si trovano questi dati di volta in volta.

3.1 Il Data Handling al CDF

Il problema del data handling non è un problema nuovo al CDF: già per le prime versioni della CAF è stato necessario provvedere a degli strumenti per la catalogazione e gestione dei file.

Per venire incontro a queste necessità alla fine degli anni novanta al Fermilab e` stato avviato il progetto SAM: Sequential data Access via Metadata.

SAM e' un sistema molto complesso, contenente databases, Data base Management system, funzioni per la gestione dei dischi e per lo spostamento dei file in ambiente distribuito.

L sua funzione principale e' quella di catalogo dei file in ambiente distribuito. Ogni file di dati (raccolti al Tevatron o simulati) viene catalogato in base a un insieme di caratteristiche note come metadata. Tra queste molto importanti vi sono il nome del file, numero di eventi nel file, creatore del file, data, locazione, se dati da collisione o simulati. I metadata sono scritti poi in diverse tabelle del database di SAM in modo che quando un utente chiede uno specifico file o un set di files o di eventi il sistema possa ritornare la locazione del file velocemente all'utente.

Grazie a SAM gli utenti del CDF accedono ad oltre 1.5 Pbytes di dati ospitati su disco o nastro a Fermilab o in altri siti.

3.2.1 SAM Data Handling System

SAM è un “file data management system” il cui obbiettivo è di ottimizzare l'uso ed i tempi di accesso delle risorse di storage, e che si colloca, in quella che è la architettura a strati del data management, tra lo Storage management System e le applicazioni di data processing come illustrato dal diagramma.

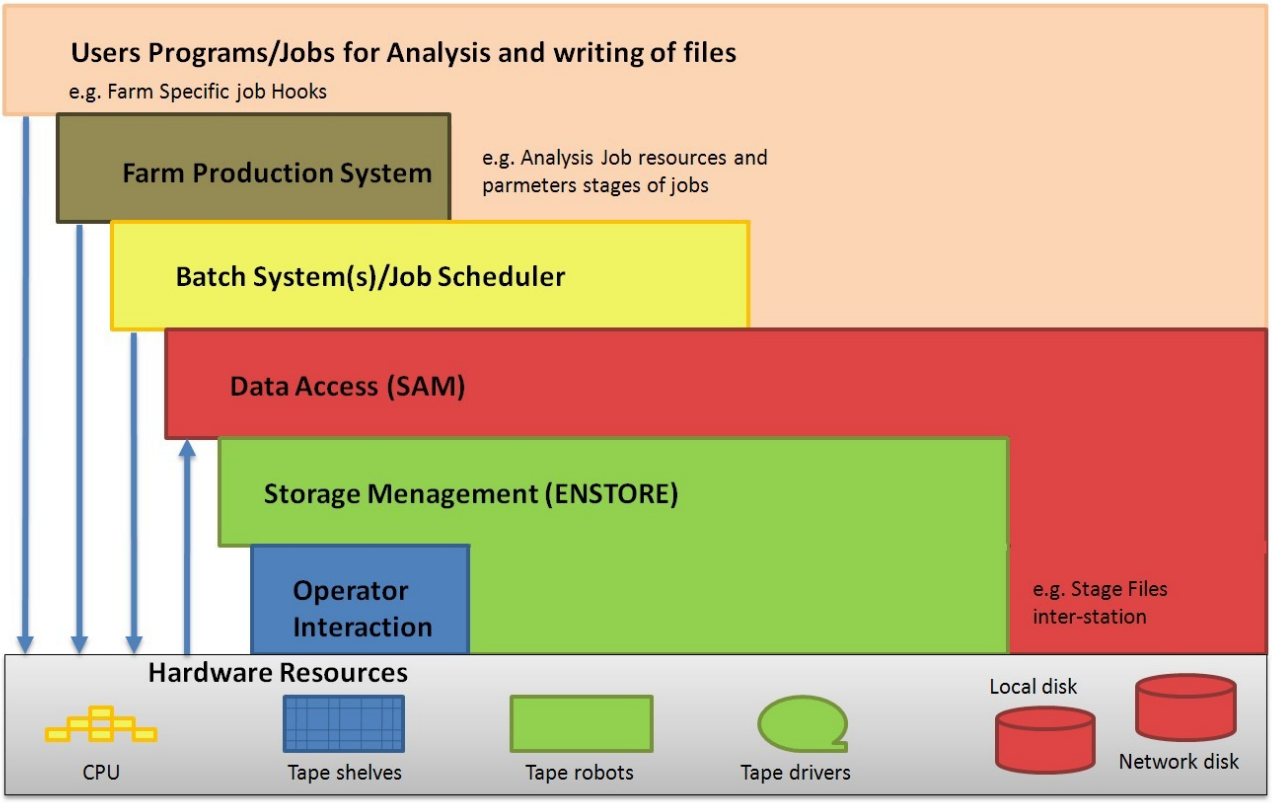


Illustrazione 22: Architettura a strati di una Storage Element, SAM e' il modulo rosso

Il progetto SAM è stato iniziato dall'esperimento D0, il secondo esperimento di fisica delle particelle presente al Fermilab, che si trovava a dover gestire problemi simili a quelli del CDF.

SAM si occupa di permettere l'accesso ai file in modo trasparente e gestire caches di dati.
SAM è stato progettato come sistema distribuito e si appoggia sul CORBA (Common Object Request Broker Architecture) e sfrutta un'architettura Client/Server.
Il sistema è composto da compute systems, database servers e storage systems distribuiti. Tutti gli elementi di storage supportano le funzionalità base per registrare e accedere ai file.
Cataloghi di metadata, cataloghi delle repliche, e databases di dati fisici sono implementati usando Oracle2© relational databases.

Un elemento fondamentale del modello SAM sono le SAM station, che svolgono le funzionalita' di Server. Una SAM station è una entità logica che contiene nel pratico diversi tipi di risorse: CPU, dischi

e altro hardware.
Una SAM station può corrispondere ad uno storage element, oppure integrare piu' storage element di piccole dimensioni oppure, al contrario, può succedere che su uno Storage Element possano essere abilitate piu' SAM station. Una stessa macchina fisica può ospitare diverse Station che condividono quindi le risorse di calcolo ma ognuna di esse possiede una propria partizione indipendente per lo storage.
Il disco fisico dove vengono allocati i file può essere gestito sia dalle Station che esternamente. Piu' dischi gestiti dalla stessa SAM station formano le cosidette “logical disk caches”. Alcune SAM station hanno inoltre la possibilità di scrivere su nastro.

Per accedere ai dati di una station è necessario stanziare un progetto, un progetto permette ad un gruppo di utenti di accedere ai file di un dataset, l'istanza di un progetto infatti prevede la definizione di un dataset associato e di un gruppo di utenti che accedono a quel dataset.
Un progetto è un processo che tiene traccia di quali file sono stati consegnati e quali file sono stati processati con successo. Nella CAF un progetto viene creato dal CAF submitter nel momento in cui viene sottomesso un job. C'è un solo progetto per ogni job, indipendentemente dalle sezioni del job. I job che hanno bisogno di file registrati nel catalogo di SAM come input possono riceverli tramite il loro progetto.

Il ruolo di client e' svolto dai cosidetti consumer, i consumer sono processi avviati dagli utenti, che hanno i privilegi per accedere ai catologhi delle station. Ogni qual volta un utente desidera effettuare l'accesso ai file registrati nelle Station lo fa tramite consumer.

Non è necessario che tutti i dati del data set siano gestiti dalla stessa SAMstation su cui è definito il progetto; infatti, una volta che il progetto è stato creato, i processi possono chiedere i file del dataset, e sarà la SAM station a passare i file iniziando da quelli presenti localmente mentreparallelamente iniziera' a prelevare gli altri dalle altre SAMstation, minimizzando i tempi di attesa.

In una SAM station è presente un servizio server, chiamato station master (SM). Tale servizio coordina le funzionalità per il reperimento dei file per i progetti che stanno utilizzano i dati della SAM station. Al momento dell'istanziazione di un progetto lo Station Master valuta la quantità di cache necessaria per far funzionare il progetto.

I servizi di registrazione e discovery dei file sono stati implementati usando CORBA Naming Service.
Le APIs per I servizi di SAM sono definite usando CORBA IDL (Interface Definition Language).

Un'altro servizio importante è lo *stager.* Lo stager è quel servizio che si occupa di muovere o cancellare file tra differenti Station. I protocolli di trasferimento usati sono **Rcp, kerberized rcp, bbftp, encp.**
Ogni Station ha un cache manager che si occupa di gestire la cache e di attuare le opportune politiche di caching per i diversi gruppi.
Infine lo *Station*s *file storage server* è quel servizio che permette agli utenti di immagazzinare i file nei tape e nei dischi.

Il catalogo dei metadata è un elemento molto importante di SAM e permette agli utenti ed alle applicazioni la ricerca dei dati tramite alcuni parametri che vengono definiti e riempiti dalla comunita' che usa tali file (nel caso di D0 e CDF parametri fisici ma, piu' in generale, parametri caratteristici per l'organizzazione che ha settato l'architettura SAM).

Anziché dover conoscere un file col suo nome è possibile accedere ai file specificando le caratteristiche di interesse di quel file, per esempio è possibile risalire a file specifici tramite la descrizione dei processi che vengono analizzati.
Le query possono essere inoltrate sia tramite riga di comando [3] che con un normale browser [4].

In figura vengono rappresentate le entità piu' importanti di una SAM station.

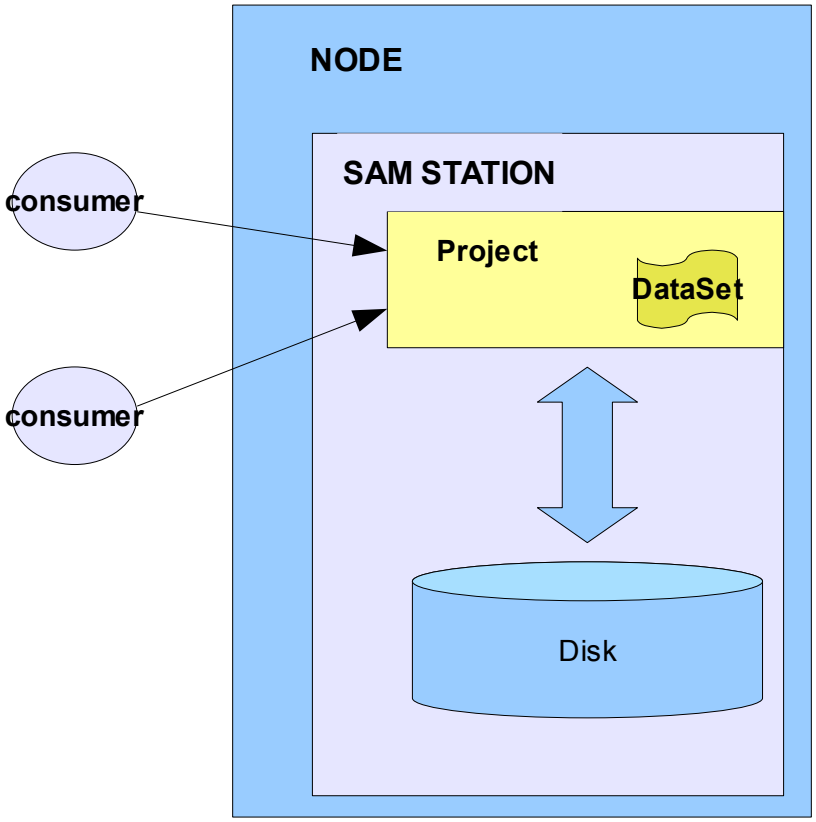


Illustrazione 23: Entita' piu' significative di una SAM Station

3.3 Limiti del datahandlig al CDF

L'architettura del data handling appena descritta, SAM, funziona presso il CDF da diversi anni e con ottimi risultati, ma solo per quel che riguarda la CAF, la D-CAF ed in generale la catalogazione e l'accesso dei file già presenti a Fermilab.
Il passaggio ad una architettura di calcolo su grid, come LcgCAF, ha riproposto il problema del data handling in modo piu' ampio dato che ora i file si trovano sparsi nei vari worker node appartenenti a diverse organizzazioni.

Si accentuano quindi i requisiti di efficenza e sicurezza negli accessi. Al momento attuale SAM è stato pensato per funzionare direttamente sopra lo Storage management System e non è stato progettato per funzionare sul Grid.

Non è ancora attiva infatti un'architettura simile che permetta di gestire e muovere i file del CDF tra i diversi nodi del Grid, problema condiviso sia per LcgCAF che per NamCAF.
Quando i job vengono inviati su Grid, per esempio attraverso LcgCAF, bisogna provvedere, prima della spedizione del job, a reperire i file di input su cui tale job dovrà fare le sue elaborazioni ed inserire questi file nel tarball. Chi si occupa di reperire in tempi brevi questi file è proprio SAM, e' pero importante che questi file siamo gia stati registrati nelle SAM Station interne al FermiLab.
Uno dei problemi relativi al non poter sfruttare le potenzialità di SAM su Grid è l'aumento di volume dei tarball che devono, in questo modo, portare con se anche tutti i file di dati di cui hanno bisogno. Questo riduce l'efficenza in fase di assegnazione delle risorse.
Oltretutto è necessario che i file che si vogliono utilizzare siano già stati registrati negli Storage Element del FermiLab; questa cosa non e` affatto banale dato che tali file potrebbero essere parte dell'output non ancora registrato di altri job precedentemente terminati.
Quest'ultima osservazione introduce il problema che si vuole affrontare nella tesi:

Il CDF al momento attuale non usa tool appositi per il trasferimento dei dati risultanti dalle simulazioni Monte Carlo dai Worker Node agli SE di CDF, sarà quindi necessario provvedere a una soluzione che permetta di gestire lo spostamento di tali dati in modo automatico ed efficiente.

Attualmente la gestione degli output dei job è totalmente demandata agli utenti, sono quindi gli utenti che devono preoccuparsi, al momento della preparazione di un job, di come muovere l'output e dove registrarlo.
Questo comporta un uso estremamente inefficiente delle risorse, sia per quel che riguarda i worker node remoti, sia nella gestione dei file prodotti che diventano poi difficilmente rintracciabili, ed anche per quel che riguarda l'uso delle risorse di rete che non vengono adeguatamente monitorate e controllate.

3.3.1 I Principali problemi legati all'attuale data handling su grid

La maggior parte dei job inviati ai Worker Node sono delle simulazioni Monte Carlo.

Come è lecito aspettarsi, poiche` tutti i segmenti sono istanze diverse di uno stesso codice, con differenti dati in ingresso, la maggior parte dei segmenti terminano piu' o meno allo stesso tempo.

Nella figura sotto viene evidenziato come 3200 sezioni eseguite durante tutta la giornata in diversi nodi finiscano tutte nell'arco di 20 minuti.

Illustrazione 24: estratto dal Web Monitoring dei job su LcgCAF

I vari worker node una volta finito di elaborare i propri job iniziano a copiare i risultati nei CDF storage element interni al fermilab.

Dalla tabella sotto[2] si può vedere a quanto ammonta la quantità di dati che vengono copiati negli Storage Element del CDF:
450 TB è il volume dei risultati delle simulazioni Monte Carlo del solo anno 2007, ossia circa 1,23 TB al giorno, con un incremento di 159.3 TB rispetto all'anno 2006.
Per dare un'idea del volume di dati che dal grid vengono copiati al Fermilab si può notare che in una sola settimana sono stati prodotti circa 1997.5 GB di dati **in piu'** rispetto alla settimana precedente.

Data Type	Volume (TB)	No of Events (M)	No of Files	Yearly Volume Increase (TB)	Yearly Event Increase (M)	Yearly File Increase	Monthly Volume Increase (GB)	Monthly Event Increase (k)	Monthly File Increase	Weekly Volume Increase (GB)
Raw Data	868.5	6309.7	1005200	222.1	1608.8	252341	29664.9	188062.3	32935	7338.8
Calibration	0.0	0.0	0	0.0	0.0	0	0.0	0.0	0	0.0
Production	941.1	7754.6	1031463	309.9	2325.3	275534	16218.5	120499.9	13189	27.0
MC	450.8	3776.7	539259	159.3	1242.0	167281	12815.2	77025.5	11935	1997.5
Stripped-Prd	45.6	525.1	49876	13.8	95.9	12304	2107.6	12741.2	1879	745.0
Stripped-MC	0.5	3.0	533	0.0	0.0	0	0.0	0.0	0	0.0
Ntuple	244.8	9710.6	237967	117.4	3272.5	98560	4208.7	102619.2	3425	0.3
MC Ntuple	112.5	1638.0	104300	93.0	1293.2	85200	10617.3	159774.7	9131	1298.6
Test	0.0	0.0	0	0.0	0.0	0	0.0	0.0	0	0.0
Others	0.0	0.0	0	0.0	0.0	0	0.0	0.0	0	0.0
Total	2663.8	29717.6	2968598	915.6	9837.7	891220	75632.3	660722.7	72494	11407.2

Tabella : Tabella riassuntiva dei dati registrati negli SE del CDF (http://cdfsam-prd.fnal.gov/~sam/data_volume/summary.html)

Il FermiLab si trova quindi periodicamente inondato di dati in ingresso.

Nel momento in cui ho iniziato a lavorare alla tesi non veniva effettuato nessun controllo sul flusso dei dati in ingresso. Quello che succedeva, e tutt'ora succede, è che lunghi periodi in cui il traffico in ingresso era relativamente moderato si alternavano a momenti in cui si avevano forti picchi di traffico.

Questi picchi di traffico sono diventati col tempo sempre piu' problematici.

Infatti tutto questo traffico in ingresso porta periodicamente la banda a congestionarsi, la congestione della banda è una situazione di difficile gestione.
I principali inconvenienti in questi casi sono:

- *Obbligare il Worker Node che ha prodotto l'output ad attendere che il traffico si riporti a livelli standard prima di incominciare il trasferimento dei dati.*

  Obbligare un Worker Node ad attendere per poter inviare l'output, dal punto di vista del Grid, implica tenere occupate risorse di calcolo per un lungo periodo di tempo nonostante nessuna operazione venga di fatto eseguita dai Worker Node. Questo oltre che significare un uso poco efficiente delle risorse va totalmente contro la filosofia del Grid computing.

- *Perdere alcuni degli output in ingresso*

  Alcune volte può capitare che a causa della congestione si perdano alcuni output, poiche' i dati di un Monte Carlo sono generati con un criterio preciso se qualche file viene perso e' necessario ricominciare la simulazione dall'inizio.

Un altro problema è che durante questi picchi gli sforzi dei device di memorizzazione a livello meccanico diventavano troppo intensi con il risultato di accorciare il tempo di vita e soprattutto l'affidabilità delle apparecchiature di memorizzazione. Il che comporta notevoli costi di manutenzione.

Altri problemi precedentemente discussi dovuti ad una cattiva gestione degli output dei Monte Carlo sono legati al fatto che dati già analizzati non sono riutilizzabili finché non vengono ricopiati nei SE del CDF.

3.3 Obbiettivi

La gestione degli output è diventata quindi di primaria importanza.

Quello che si vuole ottenere è un'architettura per il data handling su Grid simile a quella descritta per il data handling interno del CDF ed un framework che si occupi di spostare i dati ottenuti con i Monte Carlo, dai worker node remoti al Fermilab e di permetterne l'accesso in modo trasparente rispetto alla loro posizione nel Grid.
Tale architettura dovrà poi essere affiancata da un sistema che permetta una gestione intelligente del traffico sulla rete per evitare problemi di collasso dell'infrastruttura.
Non è possibile riproporre quanto già implementato per la CAF con SAM senza un lavoro aggiuntivo, perché SAM non è nato per funzionare su Grid. Come si vede anche nella figura n 22 SAM dialoga direttamente con lo Storage Management, non è quindi pensabile di poterlo utilizzare su grid nella sua versione originale data la grande eterogeneità di storage management presenti su grid.

4 SRM e SAM/SRM

Introduzione

In questo paragrafo si vuole dare una breve introduzione su SRM (Storage Resource Manager) spiegando le funzionalita' principali.

A Storage Resource Manager (SRM) *is a middleware component whose function is to provide dynamic space allocation and file management on shared storage components on the Grid [1]*.

SRM è un'interfaccia a livello di middleware che ha lo scopo di standardizzare quelli che sono i metodi di accesso ai vari Storage Element a prescindere dalla loro architettura.
Per diversi tipi di storage system l'implementazione dell'interfaccia SRM può essere anche molto diversa, le specifiche dell'interfaccia rimangono pero sempre le stesse. Questo assicura uniformità di comportamento a prescindere dal tipo di storage system.

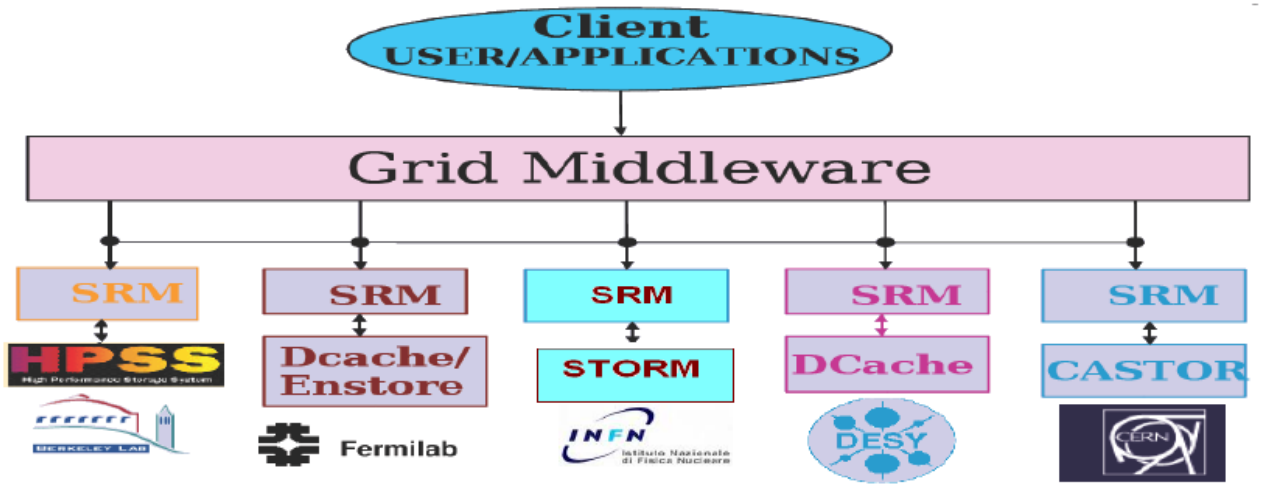


Illustrazione 25: l'interfaccia SRM permette di accedere in modo uniforme a tutti gli Storage Element a pescindere dal tipo. A livello middleware vengono utilizate solo le funzioni d'intefaccia di SRM sia per l'acceso ai file sia per il data motion.

4.1 La gestione dei file di Grid

Si introduce brevemente, senza entrare nel dettaglio, la gestione dei file nei grid, per introdurre quella terminologia che verra' utilizzata in seguito.
Nei Grid la gestione dei file e', ovviamente, molto diversa rispetto ai normali file system.
Un file nella GRID è logicamente identificato, in maniera univoca, dal suo GUID (GRID Unique Identifier), trattare i file utilizzando il GUID non è pero' user friendly:
un esempio di Guid potrebbe essere guid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6

Dello stesso file possono essere poi presenti diverse copie in diversi Storage Element, le varie copie del file vengono riferite tramite il SURL (Site URL) o PFN (Physical File Name)

- Il SURL include l'indirizzo dello Storage Element e il protocollo di accesso:
  srm://fcdflnx2.fnal.gov/cdf/home/cuomo/output10_1
  possono essere presenti piu' SURL corrispondenti ad un unico GUID

- Il LFN (Logical File Name) definisce degli alias del GUID e permete di riferirli con nomi piu' significativi e simili a quelli comunemente usati nei file system:
  fcdf.lnx4:cdf/home/cuomo/output10_1/step1
  Anche in questo caso possono essere presenti piu' LFN corrispondenti ad un unico GUID.

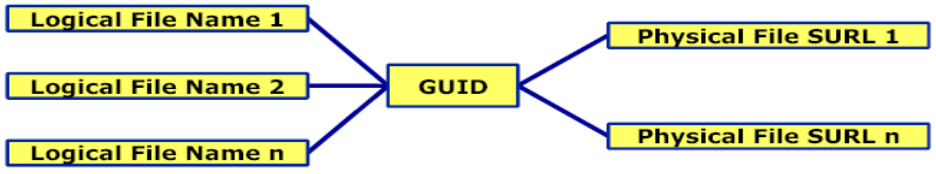
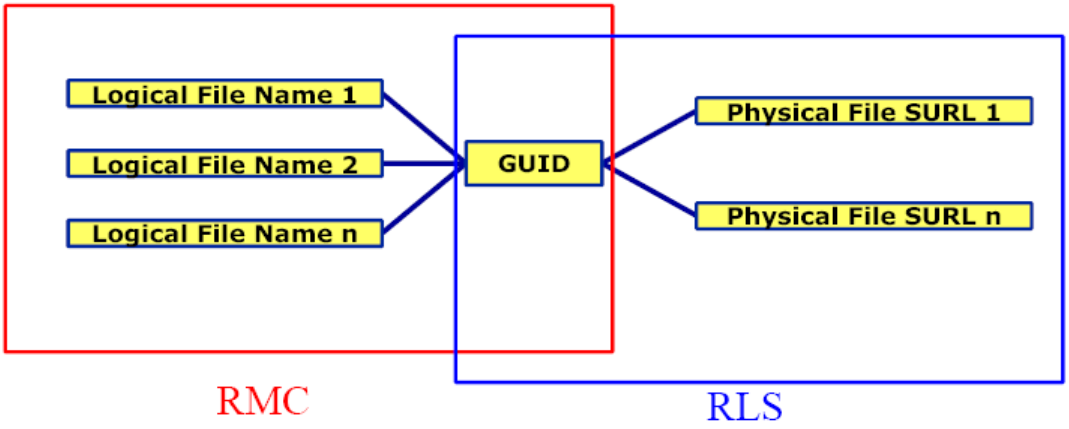


Illustrazione 26: Relazioni tra LFN - GUID - SURL

Il Replica Location Service (RLS) ed il Replica Metadata Catalog (RMC) gestiscono il mapping tra LFN, GUID e PFN:

• Replica Metadata Catalog: Logical File Name <-> Grid Unique Identifier
Il Replica Location Service (RLS) è il servizio che mantiene e rende disponibile le informazioni relative alla posizione fisica delle copie di file di dati, effettua il mapping tra il GUID e il PFN di tutte le repliche di ciascun file.

• Replica Location Service: Grid Unique IDentifier <-> Logical File Name
Il Replica Manager consiste in un set di comandi che l'utente può utilizzare per interagire con il servizio di Storage Management per operare sui file, il Replica Location Service gestisce il mapping tra LFN



4.2 I requisiti di SRM v2.2

I principali requisiti che SRM deve soddisfare sono :

1 *Un'interfaccia* che permetta di eseguire determinate operazioni e che supporti un insieme ben determinato di protocolli di comunicazione e di sicurezza

2 Deve prevedere delle funzioni per l'accesso ai dati.

3 Deve garantire il rispetto delle priorità e delle policies locali.

4 Deve prevedere l'esistenza di ben differenziate classi di storage. Infatti, in ambiente grid, oltre agli utenti normali, che hanno necessità di allocare i loro dati in modo permanente, esistono anche gli utenti "di passaggio". Devono quindi essere differenziate delle classi di storage che definisco in modo chiaro ben determinati attributi.

5 Gli utenti del grid devono avere a disposizione un insieme di operazioni che permettano di prenotare spazio libero sugli Storage Element.

6 Devono essere previsti dei meccanismi per rintracciare i dati.

7 Gli Storage Element devono avere delle funzionalità che gli permettano di dichiarare lo spazio disponibile ed il loro stato ad un qualche information system.

8 Devono essere previste funzioni di *Management* e *Monitoring* per stabilire "lo stato di salute" dei vari servizi del grid.

9 Deve essere offerto supporto per protocolli che permettano l'accesso contemporaneo a più files e protocolli per rintracciare i files in ambiente grid.

Le specifiche SRM si sono evolute nel tempo dando origine a diverse versioni. L'ultima versione è la 2.2.
SRM permette ai client di inviare richieste per allocare e gestire una determinata quota di spazio. Questo spazio può essere usato per creare file nuovi o per accedere a file esistenti.
Un job utente alla fine della sua esecuzione crea un file con i risultati dell'elaborazione. Per poter effettivamente essere sicuri che il file di output non vada perso il job deve assicurarsi che la quantità di spazio necessaria per poter allocare il file nella destinazione finale sia effettivamente disponibile e possa essere prenotata.
Dunque, dopo essersi autenticato ed aver ricevuto l'autorizzazione (requirement 1) il job effettua una richiesta per riservare una determinata quantità di spazio (requirement 5) su uno Storage Element accessibile. L'applicazione in esecuzione su un determinato Computing Element può utilizzare solo certi ben determinati protocolli di trasferimento o per l'accesso ai file (requirement 9 e 1)
Tramite l'interfaccia SRM sarà possibile prenotare uno spazio di modo che si possa copiare l'output. Altri job che debbano utilizzare tali files potranno richiederli tramite SRM senza bisogno che essi conoscano l'esatta locazione dei files. SRM implementa delle operazioni filesystem like (requirement

5) e meccanismi per la gestione trasparente dei file (requirement 7).
Lo spazio che viene riservato ai file può avere un attributo lifetime che ne determina la validità temporale, e può anche essere infinito. Scaduto il tempo di vita lo spazio precedentemente riservato torna ad essere disponibile. Anche i file già allocati in memoria hanno un loro tempo di vita, e una volta scaduto possono essere eliminati.
Di uno stesso file possono essere presenti tante copie, la gestione delle copie viene effettuata da SRM in modo trasparente. Anche per le copie è previsto un tempo di vita.

Per poter organizzare e gestire i file , SRM usa un *namespace* simile a quello UNIX. Le applicazioni possono riferirsi ai file usando UNIX-like paths.

L'interfaccia SRM usa il Web Service SOAP over HTTP. Per l'autenticazione e per verificare le autorizzazioni viene usato GSI.

4.2.1 Alcune funzioni di SMR v2.2

Le funzioni di SRMv2.2 sono descritte in dettaglio in [1]. Sotto sono elencate alcune tra le più significative.

Le funzioni di SRMv2.2 sono organizzate in famiglie: space management functions, permission functions, directory functions, data transfer functions, and discovery functions.

- *Space management functions:*
  - *srmReserveSpace:* permette di riservare un certo spazio con certe proprietà
  - *srmReleaseSpace:* permette di rilasciare uno spazio precedentemente prenotato
  - *srmPurgeFromSpace:* permette di eliminare copie di file da una data locazione

- *Directory functions:* simili alle equivalenti funzioni UNIX
  - *srmMkdir*: Crea un directory nello SRM namespace.
  - s*rmRmdir*: Rimuove una directory dallo SRM namespace.
  - *srmRm*: Rimuove file da SRM.
  - *srmLs*: permette di visualizzare il contenuto di una directory
  - *srmMv*: permette di cambiare il path di un file.

- *Data transfer functions*
  - *srmCopy:* permette ad un utente di copiare un file nello SRM namespace.
  - *srmReleaseFiles: s*etta a 0 il lifetime del file permettendo cosi al sistema di cancellare il file qualora necessitasse spazio

Per permettere ai servizi di middleware ed ai client di contattare gli “SRM endpoint” degli Storage Element è necessario pubblicare le informazioni di configurazione e di stato degli “SRM endpoint” nel Grid Information System. Per fare ciò si utilizza un standard chiamato GLUE[3] . GLUE-Grid Laboratory Uniform Environment- specifica la forma in cui devono essere pubblicate le informazioni

sugli Storage Element e sugli SRM-endpoint.

4.3 SAM/SRM

L'obbiettivo principale del progetto di integrazione di SAM con SRM è quello di estendere le politenzialità del data handling di SAM ad un generico storage system tramite l'intefaccia SRM.
Esiste una certa similitudine tra le operazioni di SAM e quelle rese disponibili da SRM, sopratutto per quel che riguarda alcune operazioni per il trasferire i file, per rimuovere i file e per gestire il namespace che sono molto simile sia per SAM che per SRM.

4.3.1 Il mapping della risorse tra SAM e SRM

Nonostante le similitudini presenti in molte operazioni di base, ci sono comunque alcune differenze tra le politiche di accesso di SRM e di accesso al file system dello standard POSIX (SAM segue infatti lo standard POSIX).
Le differenze più importanti sono:

1. Per accedere ad un file con SRM è necessario avvisare in anticipo con l'uso esplicito delle funzioni “get” o “put”. In SAM l'accesso ai file non ha bisogno prefetch: è lo standard POSIX che si occupa di gestire gli accessi.

2) In SRM un file può cambiare implicitamente il suo “costo di accesso” (sia SAM che SRM reperiscono i file cercando di minimizzare i costi di accesso)

3) Le politiche per assegnare I costi d'accesso differiscono tra SAM e SRM.

Queste differenze vanno gestite per poter permettere un mapping dei comandi copy, put, rm, ls, mkdir di SAM e SRM.

SRM differenzia diverse classi di storage, che tipicamente corrispondono a diversi tipi di Storage Element, per ogni tipo diversa classe di storage bisognerà apportare delle modifiche nelle funzionalità di SAM per poter garantire l'integrazione con SRM.
SAM dovra' inoltre occuparsi di gestire le operazioni di lettura richieste dai propri utenti che dovranno essere trasmesse a SRM tenendo conto del diverso metodo di SRM di accedere ai file, e del diverso modo di riferirli: sara quindi necessaria una logica aggiuntiva che si occupi di effettuare la conversione da URL a TURL.
SAM gestisce una memoria cache per poter ottimizzare la gestione dei file che vengono richiesti dagli utenti. Allo stesso modo tale memoria cache può essere utilizzata per tenere memoria dei file richiesti da SAM via SRM.

4.3.2 Classi di storage

Di seguito si evidenziamo le diverse politiche che SAM dovra' implementare per ogni clesse SRM di Storage Element.

Volatile storage

La classe di storage “volatile” permette allo Storage Element di cancellare implicitamente i file se non utilizzati, questo perché tipicamente una locazione definita “volatile” ha uno spazio disco limitato e deve quindi poter attuare le sue politiche di gestione e pulizia del disco, questo implica che la cache di SAM possa non essere corretta.
Per poter assicurare che il catalogo delle repliche sia consistente con la situazione effettiva SAM deve controllare la disponibilità effettiva dei file ogni volta che questi vengono richiesti. Se il file risulta non più allocato l'indirizzo SRM presente in cache dovrà essere cancellato.

Durable storage

La classe di storage “durable” registra e cancella i file solo a seguito di richieste esplicite. SAM dovrà quindi occuparsi di gestire esplicitamente lo spazio SRM, facendo corrispondere alle richieste dei client i relativi comandi SRM, è molto importante che SAM si occupi di gestire lo spazio in modo da non far superare allo Storage Element la quota di spazio prefissata.

Permanent storage

La classe di storage “permanent” non cancella mai i file. Può succedere che il “costo di accesso” al file cambi secondo la politica dello Storage Element.
Poiché sia SRM che SAM tentano di minimizzare i costi di acceso è opportuno che tali costi vengano gestiti in modo esplicito, per questo nel momento della scelta del file vanno implementate delle funzioni che permettano di leggere il costo di accesso al file stesso, chiamate request prioritization.
Le request prioritization possono essere di due tipi : attiva e passiva.
Attiva quando si chiede a SRM il costo assegnato ai files.
Passiva quando il calcolo del costo viene effettuato direttamente da SAM, il vantaggio di questa strategia è che si riducono i tempi di overhead dovuti a SRM.

4.3.3 Traduzione dello URL

La sincronizzazione tra il catalogo delle repliche ed il contenuto dello storage viene effettuato dalla SAM station.

La SAM station effettua un traduzione da Logical File Name a Site URL ogni volta il file viene inviato al un file richiederte.

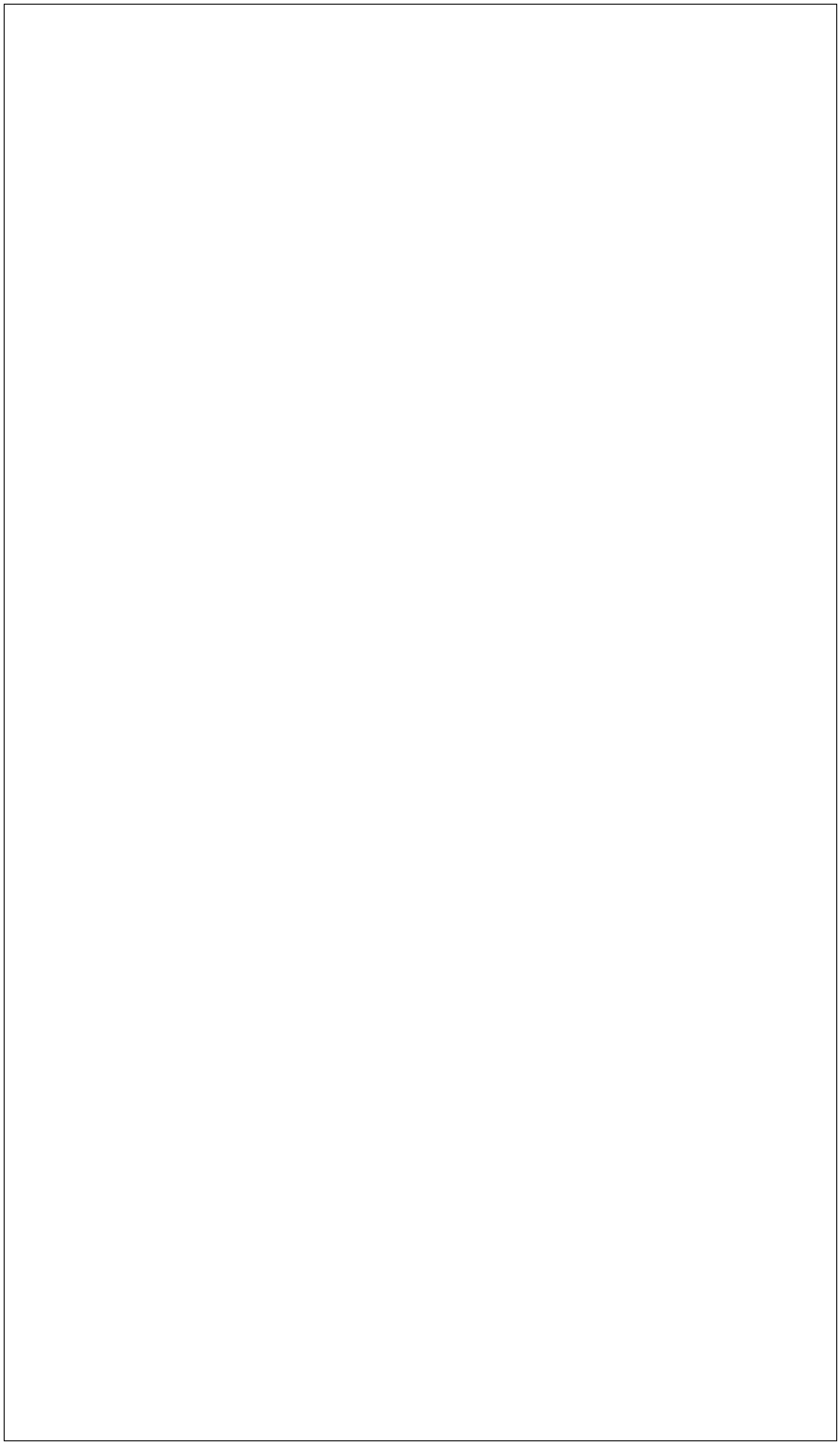
4.3.4 Location mapping

Fare il mapping dei file da SAM a SRM presenta degli inconvenienti legati alla struttura delle directory. Tipicamente SAM copia tutti i file in un'unica directory nella sua cache.
Una prima soluzione per non perdere le struttura a directory di SRM è quella di mettere il path del file in append rispetto alla cartella principale di SAM. Il problema correlato con questa soluzione è il crescere incontrollato dei path.
Una seconda soluzione potrebbe essere di scegliere tra le repliche quella con il path più corto.

4.4 Schema di accesso ai file

Lo schema seguito da una SAM station integrata su SRM nella richiesta di file è più complicato rispetto allo schema tipico di una SAM station.

Il flow può essere riassunto in 13 punti ed è riportato in figura (n) in figura (n+1) vengono messi in evidenza i diversi componenti di SAM che partecipano in ogni azione.
1) start project
2) E' nella cache?
3) Si/no . Se si si passa al punto 8 , altrimenti si va al punto 4
4) get priority
5) SRM get metadata
6) return metadata
7) return priority
8) request transfer / get
9) SRM get/copy
10) SRM get/copy done
11) transfer request complete
12) si registra la nuova replica SRM
13) Il file è disponibile



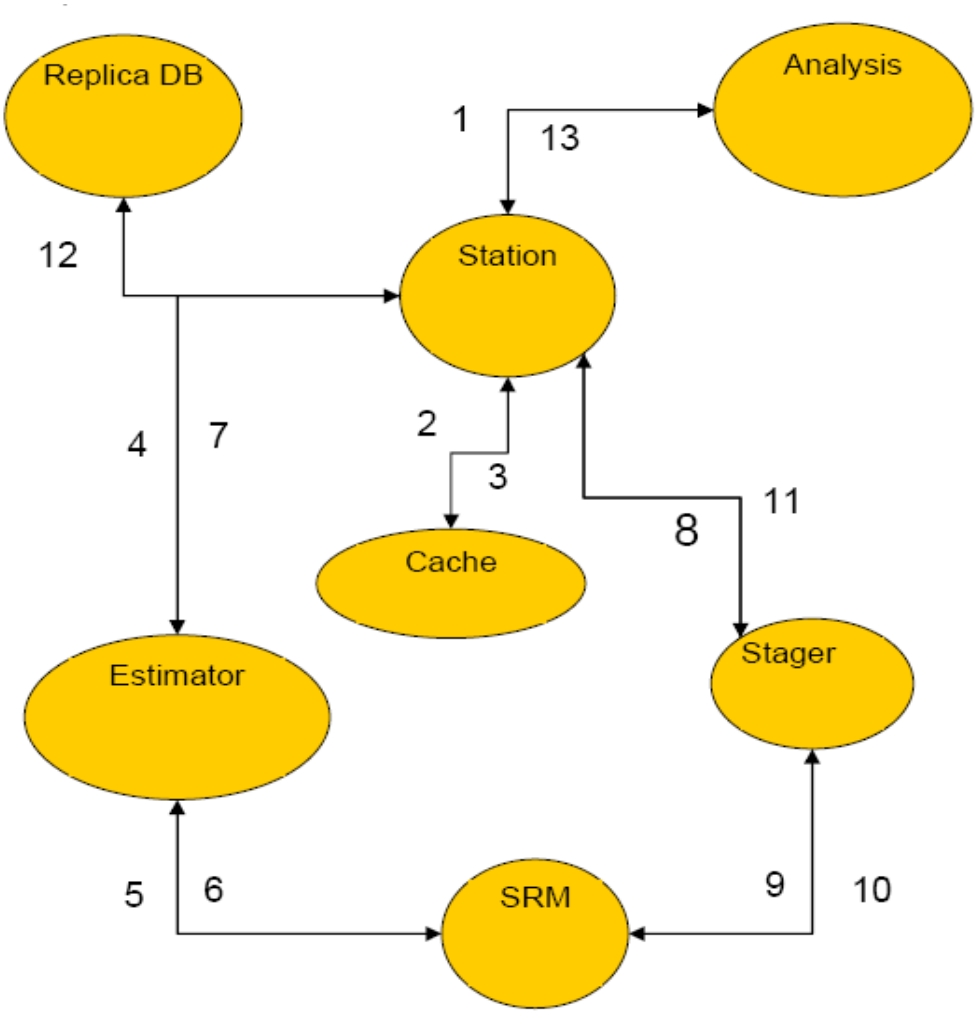


Illustrazione 28: Gli elementi di una SAM station impiegati nei diversi passi dell'algoritmo.

4.5 Uso di SAM/SRM per CDF

Come già discusso precedentemente l'attuale modello di data handling del CDF è basato su un sistema SAM.
Col passaggio ad una architettura di calcolo su Grid il CDF si trova quindi ad aver una grossa quantità di dati che, dopo essere stati prodotti nei vari worker node, devono essere riportati a Fermilab prima di

poter essere utilizzabili. Ciò comporta notevoli inefficienze come discusso nel capitolo precedente.

Come verra' meglio descritto nel quinto capitolo il modello che si pensa di realizzare prevede dei punti di raccolta intermedi nei quali i dati vengano raccolti e catalogati per poi essere riportati al FermiLab in un secondo momento.

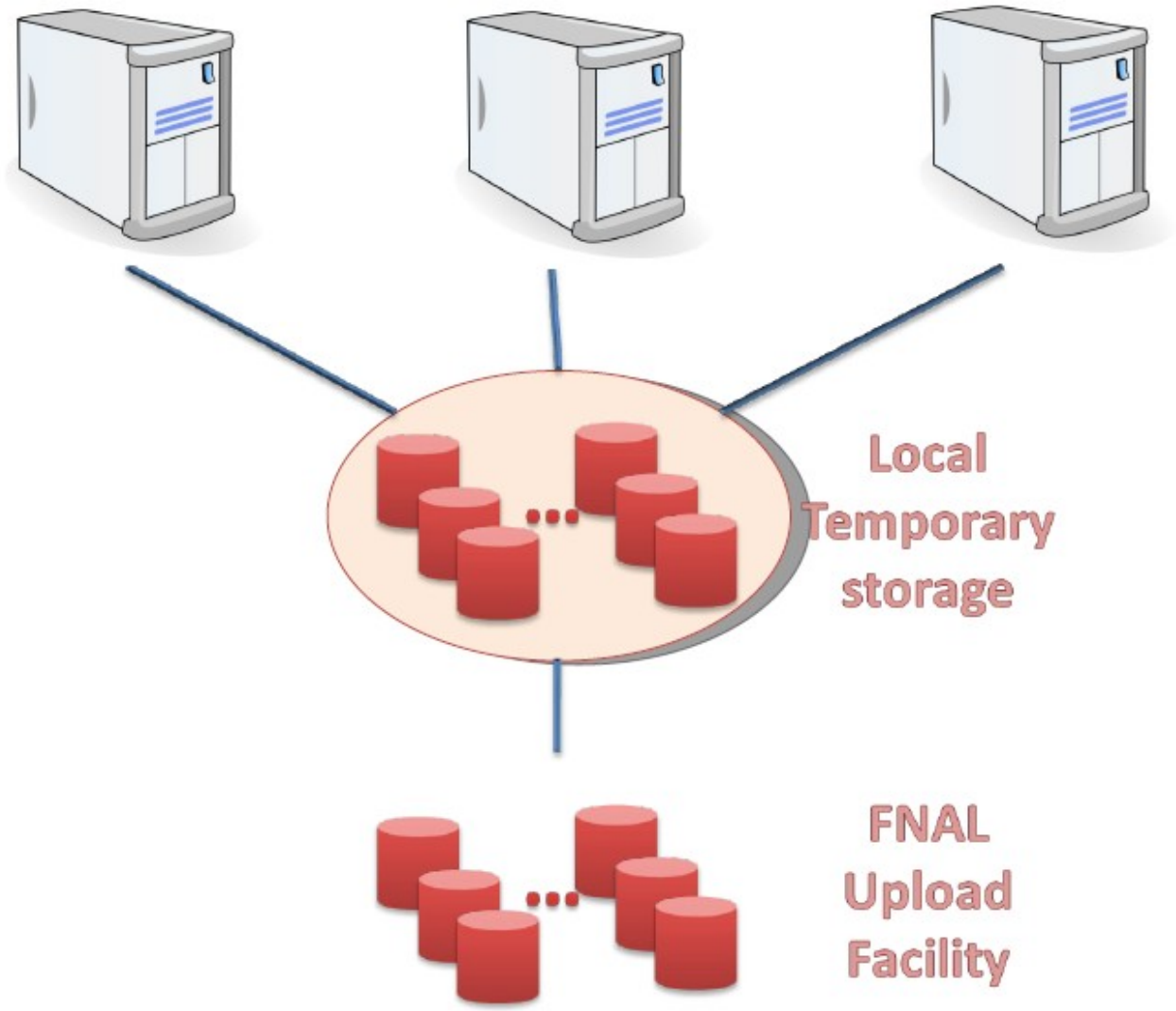


Illustrazione 29: Modello proposto per il Data Moving

Una volta che i dati sono stati copiati nel punto di raccolta temporaneo bisognerà poi provvedere a rendere questi dati subito disponibili alla comunità CDF.
Bisognerà quindi prevedere la presenza di un sistema che si occupi di :

- rendere fin da subito disponibili I dati, anche se essi non si trovano ancora negli Storage Element del Fermilab,
- rendere trasparente agli utenti la struttura dei cluster e dei punti di raccolta,
- mascherare ogni dettaglio riguardo alla collocazione fisica dei file,

- permetta di spostare questi file dai vari nodi del grid in modo automatizzato, preoccupandosi sia di utilizzare le risorse di rete nella maniera migliore possibile sia di gestire le repliche.

Per fare ciò si pensa di utilizzare SAM over SRM come mostrato in figura.

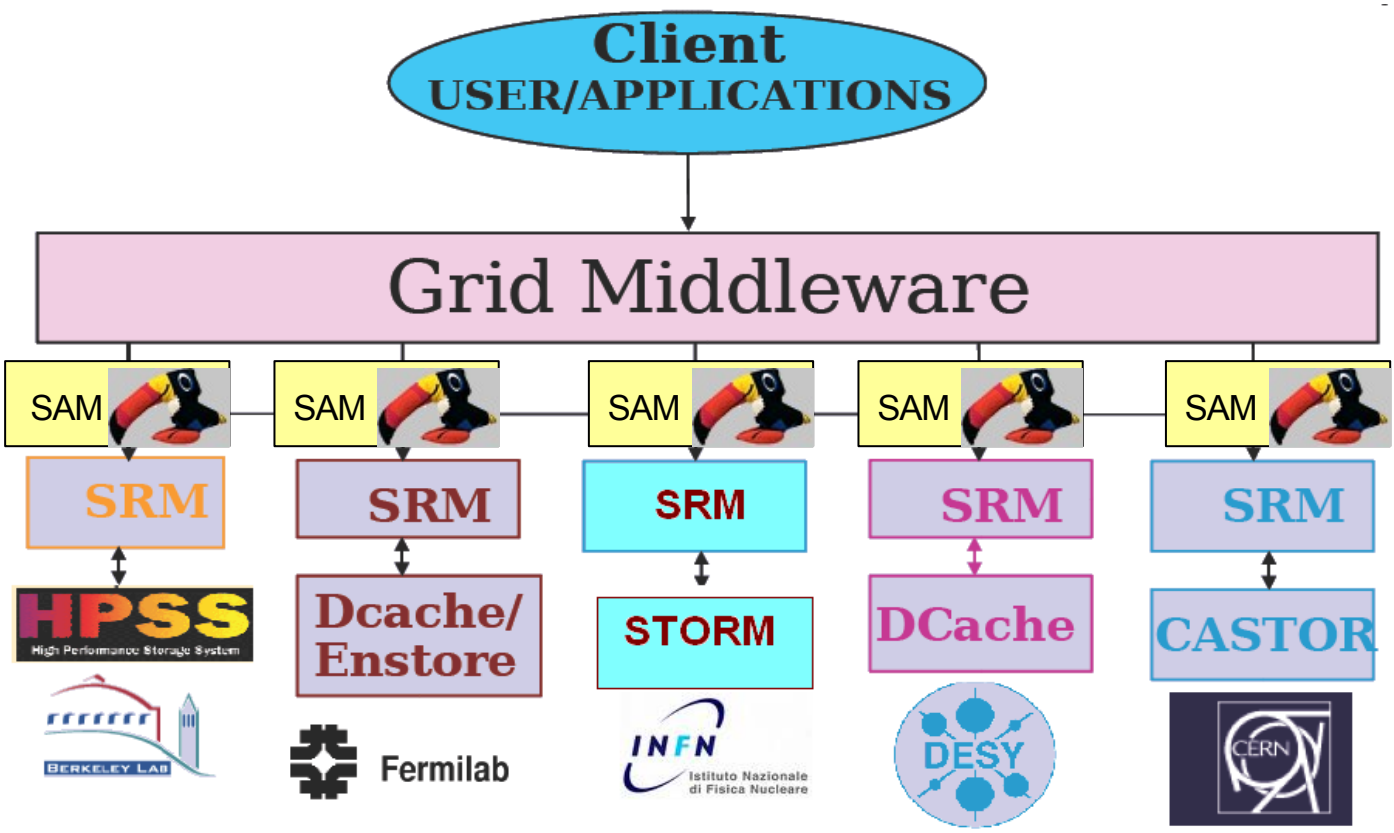


Illustrazione 30: SAM/SRM

L'integrazione di SAM con SRM è ancora sperimentale, però permette di affrontare il problema del data handling su grid in modo più conveniente per il CDF rispetto all'adozione di altre tecnologie usate dagli esperimenti LHC.
E' evidente che poter utilizzare, già dai punti di raccolta intermedi, una architettura SAM che sia integrata con le altre SAM station già presenti al FermiLab offrirebbe enormi vantaggi da un punto di vista dell'efficenza.

- Sarebbe possibile sfruttare il know how già maturato all'interno di CDF, il che renderebbe I costi di manutenzione ed i costi relativi al passaggio tecnologico minimi rispetto ad utilizzare altre tecnologie per il data handling su grid
- Permetterebbe fin da subito di integrare il catalogo dei file presenti al FermiLab con I cataloghi dei punti di raccolta temporanei.
- SAM si occuperebbe di rendere trasparente all'utente l'allocazione fisica dei file, gli utenti accederebbero a tutti i file registrati nei cataloghi, siano essi nei punti di raccolta temporanei o al FermiLab.
- L'accesso ai file, e quindi la ricerca delle repliche, verrebbe effettuato da SRM, in questo modo si sfrutterebbero appieno le potenzialità offerte da SRM.

Quello che ci riproponiamo di con questo lavoro è testare le potenzialità offerte da SAM/SRM.
Rendere operativa una prima struttura del tipo precedentemente descritto e valutare la funzionalità di SAM/SRM. Il nodo che si occuperà di fare da punto di raccoltà temporaneo si trova al CNAF di Bologna ed è uno Storage Element gestito da STORM.

In un primo momento bisognerà quindi valutare sia le funzionalità di SAM/SRM per d-cache e per Storm per poi vedere come le due SAM station riescono a comunicare e sincronizzarsi mascherando la struttura sottostante.

CAPITOLO 5

PROGETTO ED IMPLEMENTAZIONE DEL PROTOTIPO

Introduzione

In questo capitolo si descrive il progetto e la realizzazione di un primo prototipo implementato per risolvere il problema del trasferimento dei files dai Worker Nodes, dove vengono prodotti, agli Storage Element del FermiLab.

Partendo della definizione dei requisiti si presenta la soluzione proposta. Successivamente si specificano le tecnologie utilizzate e si approfondiscono gli aspetti tecnici effettivi della soluzione. Si descrive poi il primo prototipo implementato.

Nello spiegare la soluzione si procede iniziando con la definizione dei requisiti che dovranno essere soddisfatti a prescindere da dettagli implementativi.

Il Grid è per definizione un'entità dinamica e molti cambiamenti avvengono ogni giorno. E' inoltre un settore in continua espansione, le potenzialità vengono sfruttate ogni giorno di più e sempre nuovi utenti accedono alle sue risorse così come sempre nuove risorse vengono aggiunte a quelle già presenti.

Questo comporta una notevole difficoltà nel prevedere, in termini quantitativi, le grandezze su cui si opera. E' possibile riferirsi alla storia dei mesi passati per avere un'idea ma è impossibile indagare in

modo esaustivo o cercare di predire, a questo livello di progettazione, quelle che saranno le necessità in termini di spazio di memoria, di tempi di permanenza, di punte di traffico o anche di traffico medio.

Questo rende estremamente difficile essere precisi in fase di specifica e analisi dei requisiti e, in generale, sarebbe molto complesso e dispendioso seguire uno schema di progettazione a cascata.

Per risolvere i problemi fin qui descritti si è quindi deciso di utilizzare un approccio evolutivo in cui una prima soluzione sperimentale venga migliorata in modo incrementale, attraverso raffinamenti successivi.

Il primo passo sarà quindi realizzare un primo prototipo da affinare successivamente risolvendo i vari problemi che si incontrano via via.

5.1 Definizione dei requisiti

I principali requisiti, per il sistema che vogliamo progettare sono:

- *Dovrà essere portabile*: considerato il grande dinamismo e la continua crescita in termini di risorse ma anche di utenti e di consumi del CDF, e` opportuno che la soluzione possa essere adoperata facilmente su architetture eterogenee.

- *Dovrà essere trasparente all'utente*: finora sono stati gli utenti ad occuparsi dei vari aspetti per lo spostamento dei dati negli Storage Element del CDF, dai protocolli da usare, al percorso da seguire fino alla destinazione dei finale. La soluzione che si vuole implementare dovrà rendere trasparente all'utente qualsiasi aspetto relativo al data movement.

- *Dovrà essere sicura*: utilizzare gli opportuni protocolli di sicurezza , autenticazione e certificazione in ambiente Grid.

- *Dovrà garantire la tracciabilità dei dati:* come specificato nel secondo requisito, i file saranno guidati in modo automatico dai Worker Node verso il FermiLab e durante questo processo deve essere possibile sapere dove sono temporaneamente allocati i file.

Oltre ai Requisiti funzionali sono inoltre stati evidenziati due requisiti di efficienza che la soluzione finale dovrà avere:

- *Dovrà permettere l'accesso ai file:* deve essere possibile accedere ai file sia che essi si trovino già al FermiLab sia che attendano per essere copiati negli Storage Element di CDF.
- *Garanzia di corretto funzionamento in condizioni standard.*
- *Uso efficiente delle risorse condivise*: la soluzione dovrà permettere ai Worker Node di poter trasmettere il prima possibile il risultato dei Monte Carlo in Storage Element appositi in modo da poter essere nuovamente disponibile per elaborare altri job.
- *Uso efficiente delle risorse di rete*: la soluzione dovrà far si che il trasferimento da un punto ad un altro del grid sia gestito senza sprechi di risorse, anche dal punto di vista delle risorse di rete.
- *Gestione del traffico:* la soluzione implementata dovrà permettere di schedulare l'ordine con cui i file devono essere copiati negli Storage Element di FermiLab.

5.2 Overview della soluzione proposta

Si descrivono ora, in corrispondenza dei requisiti appena esposti, le proposte di progetto effettuate.

Requisito 1: portabilità

Data l'estrema dinamicità dell'applicazione è difficile allo stato attuale prevedere con precisione grandezze come il numero medio di file giornalieri e mensili che verranno prodotti da qui a qualche mese, o lo spazio disco che sarà necessario. E' opportuno dunque che l'architettura proposta sia estremamente scalabile.

Quello che si è pensato di fare è organizzare i nodi in maniera gerarchica.

I Worker Node del grid vengono clusterizzati in funzione della loro posizione geografica. Ogni Worker Node del cluster-virtuale, finite le proprie elaborazioni, manda l'output ad un punto di raccolta intermedio.
Dal punto di raccolta intermedio i file vengono poi spediti al FermiLab.

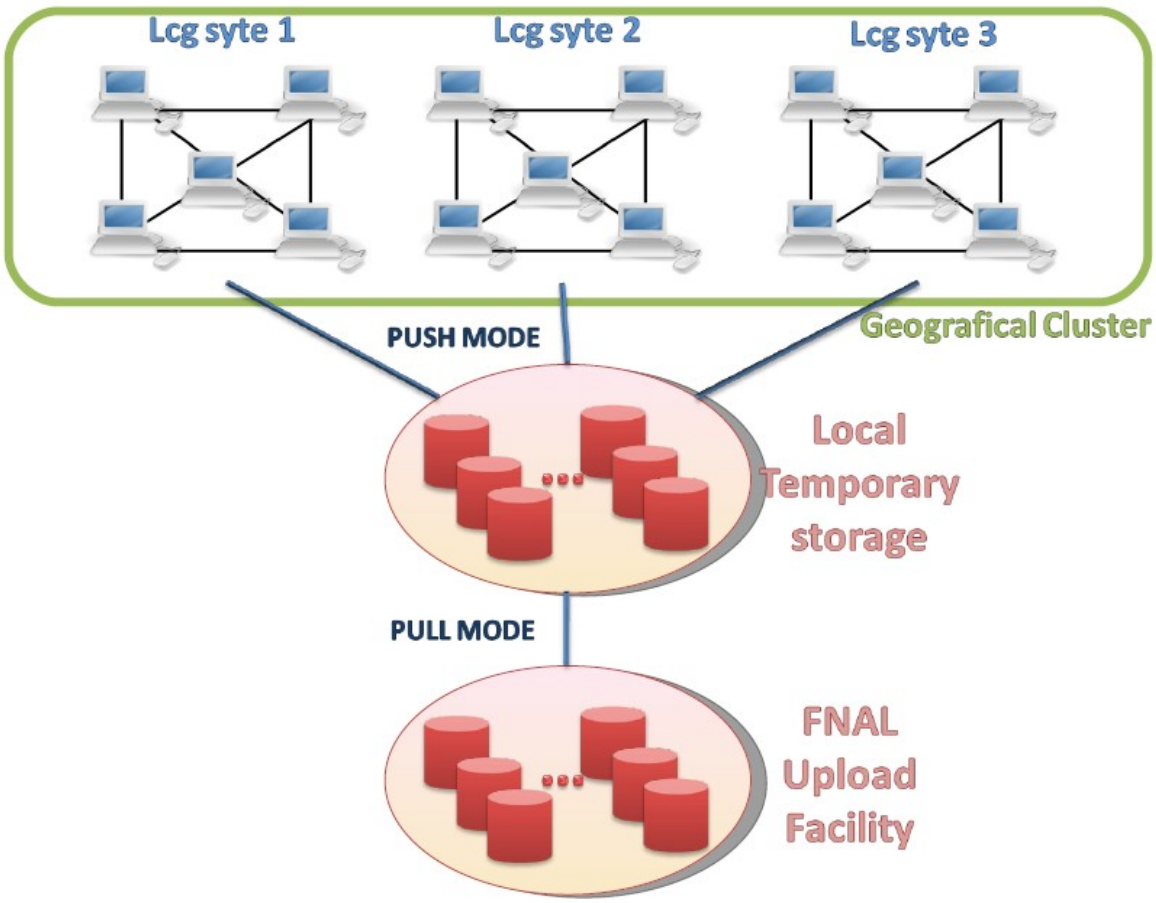


Illustrazione 31: I diversi Worker Node vengono clusterizzati.

La soluzione presenta un alto grado di scalabilità: ogni qualvolta un nuovo nodo si aggiunge al grid questo può essere incorporato ad uno dei cluster virtuali oppure può andare a formare un nuovo cluster assieme ad altri Worker Node qualora la zona sia "scoperta". Può anche succedere che, qualora un Local Temporary Storage non riesca a gestire tutto il traffico dei suoi Worker Node, un cluster possa essere diviso in due o più gruppi senza alterare il funzionamento complessivo del sistema.

La struttura proposta presenta anche altri notevoli vantaggi tra cui quello di abbassare i costi per CDF: la responsabilità del funzionamento di ogni Local Temporary Storage viene demandata ai gruppi locali. Questo semplifica notevolmente la gestione delle risorse e aumenta la tolleranza ai guasti complessiva del sistema, d'altra parte un controllo centralizzato di una tale struttura sarebbe impensabile**.**

Requisito due : Trasparenza

Finora sono stati gli utenti ad occuparsi dei vari aspetti per lo spostamento dei dati negli Storage Element del CDF, dai protocolli da usare, al percorso da seguire fino alla destinazione finale. La

soluzione che si vuole implementare dovrà rendere trasparente all'utente qualsiasi aspetto relativo al data movement. A livello applicazione sarà quindi necessario provvedere a scrivere un'interfaccia che si occupi di nascondere all'utente la presenza di tale architettura.

Requisito 3: sicurezza

Il grid dispone di diversi meccanismi che ne tutelano la sicurezza, e` quindi importante che la soluzione soddisfi i criteri di sicurezza e autenticazione nel grid.

Requisiti di efficenza

E_1 : Accesso ai file

Bisogna permettere l'accesso ai file che si trovano nei *Local Temporary Storage.*

In questo modo si cerca di dare una soluzione ad uno dei problemi che si hanno con l'attuale data handling che sfrutta la potenzialità di SAM come catalogo dei dati solo localmente a FermiLab.

Con il data handling attuale è necessario che i file che si vogliono utilizzare siano già stati registrati negli Storage Element del FermiLab, cosa non banale dato che tali file potrebbero essere parte dell'output non ancora registrato di altri job precedentemente terminati.

Diventa quindi importante scegliere una tecnologia opportuna che**,** utilizzando il middleware grid**,** permetta di accedere in modo uniforme sia ai file presenti al FermiLab sia ai file allocati nei Local Temporary Storage.

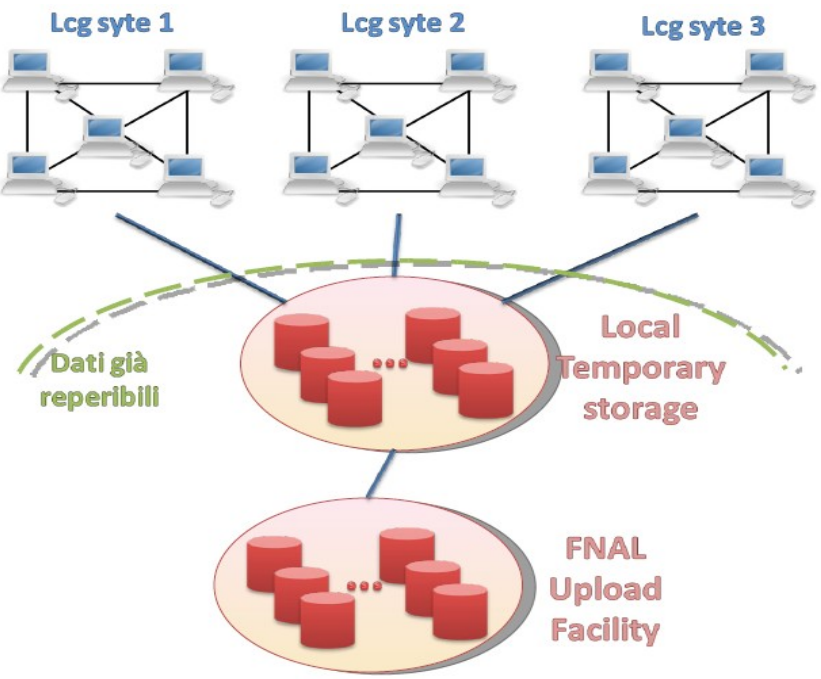


Illustrazione 32: L'accesso ai dati deve essere garantito fin dai LTS

E_2: Garanzia di corretto funzionamento in condizioni standard

La presenza di punti di raccolta temporanei che gestiscono piccoli cluster di Worker Node potrà permettere di fare una stima, a breve e medio termine, di quanto spazio disco possa servire per raccogliere tutti i dati provenienti dal cluster in condizioni stazionarie.

Questo permette ad ogni Local Temporary Storage di gestire le proprie risorse, in modo da garantire per un determinato periodo di tempo di poter tenere in memoria i file in arrivo dai Worker Node.

E_3: Uso efficiente delle risorse condivise

La soluzione dovrà permettere ai Worker Node di poter trasmettere il prima possibile il risultato dei Monte Carlo in Storage Element appositi**,** in modo da poter essere nuovamente disponibile per elaborare altri job.

Si vuole infatti ridurre il tempo di permanenza dei risultati dei Monte Carlo nei Worker Node**,** in attesa di essere traferiti negli Storage Elemet di FermiLab.

Lasciare i risultati delle elaborazioni sui Worker Node implica tenere occupate risorse di calcolo per un lungo periodo di tempo nonostante nessuna operazione venga di fatto eseguita. Questo**,** oltre che

significare un uso poco efficiente delle risorse**,** va totalmente contro la filosofia del Grid computing.

Pertanto bisognerà permettere ai Worker Node di liberarsi il prima possibile dei file prodotti**;** i Worker Node dovranno quindi trasferire i loro dati in modo Push**.**

Per contro**,** dalla parte del FermiLab è possibile, utilizzando un metodo Pull, far si che il flusso in ingresso risulti costante per tutto il tempo che intercorre tra la fine di una simulazione e dell'altra.

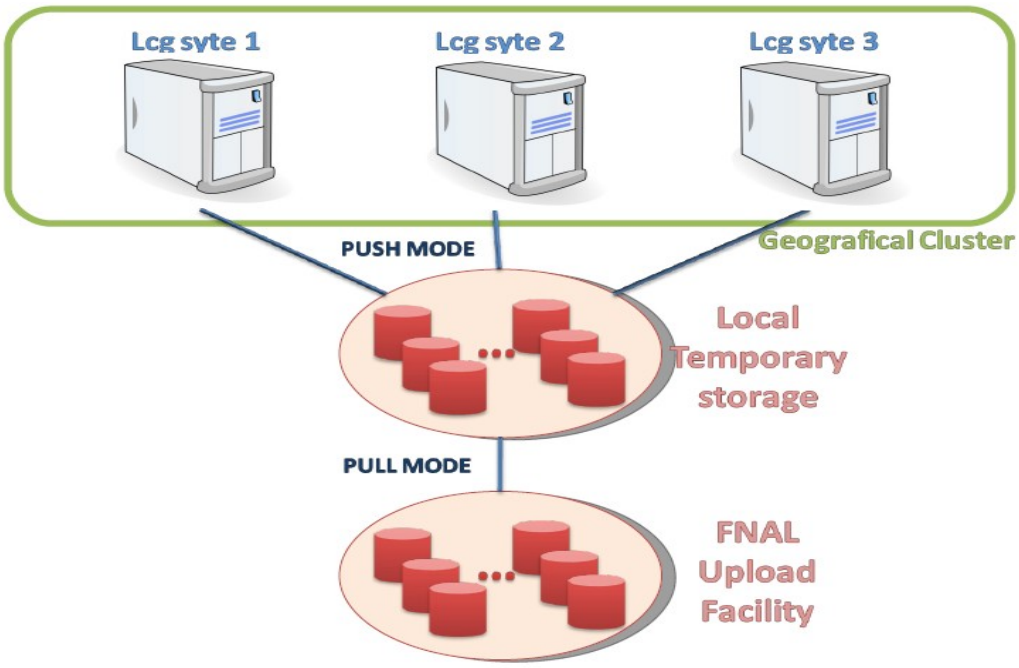


Illustrazione 33: Modello Push/Pull

E_4: Uso efficiente delle risorse di rete

La soluzione dovrà permettere di evitare situazioni di sovraccarico della rete. Deve essere possibile gestire il trasferimento dei dati in maniera dinamica. Utilizzando un approccio Pull mode è il destinatario finale a decidere quando iniziare il trasferimento da un particolare Local Temporary Storage.

Riassumendo, il vantaggio di passare da uno schema puramente Push ad uno schema misto Push/Pull è quindi quello di poter gestire la rete nella maniera più conveniente e contemporaneamente permettere ai Worker node del grid di poter essere utilizzati in maniera più efficiente, in modo che si riesca a tenere il traffico costante, evitando saturazioni.

5.3 Aspetti tecnologici

A partire dalle specifiche della soluzione, si procede ora nello spiegare le varie scelte implementative adottate.

5.3.1 Il Catalogo dei file

L'idea principale su cui poi si basano tutti gli altri aspetti implementativi è quella di implementare un prototipo che permetta di tenere traccia della posizione dei file nel Grid, di accedere a questi file prima che vengano scritti negli Storage Element del FermiLab e che permetta di gestire lo spostamento dei file nella maniera desiderata [Requisiti 2, E1 ed E4].

Una necessità emersa e` anche quella, nel passaggio al grid, di utilizzare tecniche di catalogazione dei file per permettere che tutti i file siano visibili come logicamente appartenenti alla stessa entità, astraendo dalla loro posizione fisica e da altre caratteristiche di tipo implementativo.

Questo rende necessario l'uso di un catalogo dei file temporanei allocati nei Local Temporary Storage. Tale catalogo deve contenere informazioni sui file contenuti nel Local Temporary Storage e deve poter restituire tali informazioni quando interrogato esternamente.

Tale catalogo deve contenere informazioni sia sui file allocati temporaneamente nei Local Temporary Storage che di quelli allocati negli Storage Element del FermiLab e non ancora validati e deve poter permettere l'accesso a tutti questi file in modo omogeneo [Requisito 4].

Un utente che voglia accedere a questi dati deve poterlo fare senza preoccuparsi della posizione fisica dei dati [Requisito 2]**:** con una semplice query al catalogo centrale deve sapere se il file da lui richiesto

esiste ed eventualmente potervi accedere.

Il problema, seppur con dimensioni diverse, non è nuovo all'interno del CDF. Già nel passaggio dalla CAF alla D-CAF (vedere capitolo 2 per maggiori dettagli) era stato formalizzato un problema simile con requisiti praticamente identici.

Quello che si è utilizzato allora, che ha funzionato e tuttora funziona con ottimi risultati**,** sia da un punto di vista delle prestazioni che delle funzionalità, è il middleware SAM**.**

In questo caso si utilizzera' SAM/SRM. Gli aspetti riguardanti le caratteristiche di SAM **e** di SRM così come l'integrazione tra SAM ed SRM**,** sono stati già descritti nei capitoli 3 e 4.

Dopo aver definito il problema (capitolo 3), definito i requisiti e proposto un modello, quello che si vuole fare ora è allestire un prototipo funzionante della soluzione utilizzando SAM/SRM.

Questo servirà per studiare l'effettiva fattibilità della soluzione proposta e valutare lo stato dell'arte dell'integrazione tra SAM/SRM.

5.3.2 Perché utilizzare SAM/SRM:

Non esistono, allo stato dell'arte, delle tecnologie che risolvano appieno il problema del data handling sul grid rispettando i requisiti sopra descritti; nel corso degli ultimi anni sono state proposte diverse soluzioni:

1. Phedex : usato dal esperimento CMS e ancora in fase sperimentale.

- Condor – Stork: Stork non e' un sistema di data handling, e' solo un catalogo locale di disco che si appoggia sul middleware condor (vedere cap 2). Il fatto che si appoggi su questo particolare tipo di middleware la rende poco adatta a lavorare sul grid europeo LCG.

  Stork non permette di utilizzare SRM e prevede funzionalita' solo per il trasferimento dei files e non per l'accesso remoto ai dati**.**

- SAM : originariamente pensata e progettata da D0 e CDF per risolvere problemi simili in ambiente distribuito**,** è attualmente il data handling framework per CDF. Il suo passaggio al Grid e' ancora in uno stato iniziale.

I vantaggi fondamentali nell'usare SAM/SRM si possono riassumere in 4 punti fondamentali:

1) SAM/SRM fornisce già degli strumenti che permettono una gestione trasparente ed omogenea dei file registrati, implementano accurati meccanismi per la gestione delle repliche e lo spostamento dei file.

2) CDF usa già SAM come catalogo.
3) E' possibile utilizzare il know how maturato all'interno di CDF per quel che riguarda SAM.
4) E' possibile prevedere l'integrazione delle SAM Station dislocate nei vari Local Temporary Storage con le SAM Station del CDF. Usare un'altra delle tecnologie precedenti richiederebbe poi un notevole sforzo per poter integrare i cataloghi dei Local Temporary Storage con quelli già presenti al CDF.
5) Ci si appoggia sull'interfaccia middleware SRM: SRM si sta velocemente diffondendo come standard sui Grid.

5.4 Implementazione

Il CNAF, il centro di calcolo delliIstituto Nazionale di Fisica Nucleare, è stato come il primo punto di raccolta temporaneo e proprio al CNAF è stato allestito il primo Local Temporary Storage.

Lo Storage Element è di tipo Storm, con un'interfaccia SRM v2.2**,** con uno spazio disco di 300 GB e una scheda di rete 1 Gbit/s.

5.4.1 StoRM

StoRM è un Grid Storage implementato dall' Istituto Nazionale Fisica Nucleare – INFN e fornito di interfaccia SRM v2.2.

StoRM implementa uno storage resource manager per sistemi di storage “disk based” gestiti da file system paralleli o distribuiti.
StoRM utilizza lo standard POSIX per gli accessi al file system (space reservation, accesso diretto ai file etc.) e si basa sui certificati di grid VOMS X.509 (vedere appendice) per garantire meccanismi di certificazione ed autenticazione. Ogni volta che l'utente chiede di accedere ad un file il sistema controlla che l'utente sia in possesso di un proxy-certificate valido.
StoRM**,** oltre che implementare tutte le funzionalità di SRMv2.2**,** permette l'utilizzo dei più diffusi protocolli grid come gsiftp e rfio.
L'architettura logica è illustrata nella figura n . **S**i può notare come il logic core sia separato dal file system sottostante.

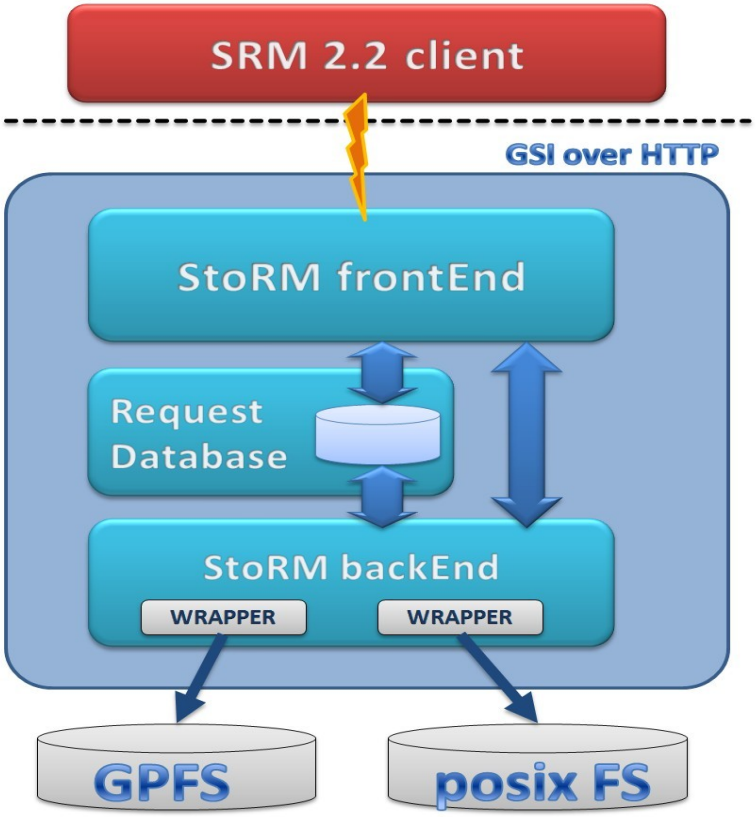


Illustrazione 34: L'architettura di StoRM

5.4.2 Il prototipo

Per poter introdurre i vari aspetti implementativi si inizia con la presentazione del modello di data moving proposto, in modo da poter introdurre i concetti ed i passaggi più importanti di cui si è dovuto tener conto nella messa in campo del prototipo.

Trasferimento dei dati dai Worker Node ai Local Temporary Storage

Un utente sottomette una simulazione Monte Carlo tramite LcgCAF. Il Resource Broker decide di assegnare il job ad uno dei siti che fanno capo al CNAF come Local Temporary Storage.

Una volta terminata la sua esecuzione il Worker Node invia il risultato del Monte Carlo con i metadata relativi al CNAF;

Questo introduce le prime difficoltà implementative:

a) Far sapere ai Worker Node dove inviare l'output:

nella LcgCAF attuale è previsto che il Worker Node spedisca il proprio output in una locazione definita dell'utente con dei protocolli anch'essi specificati dall'utente. L'utente deve quindi preoccuparsi di inserire nel proprio codice anche tutti i comandi per il data movement dell'output. Non è pensabile, nella soluzione,che sia l'utente ad inserire i comandi per ridirigere l'output del proprio Monte Carlo al CNAF per due motivi principali:

1) L'architettura di data handling degli output deve essere trasparente all'utente secondo quanto definito dal requisito numero due: l'utente deve essere estraneo al percorso che verrà effettuato dai risultati delle sue simulazioni, dovrà semplicemente essere avvisato dal mailer daemon (ved. Cap 2) che il suo job è terminato e che i suoi sono reperibili e accessibili tramite il SAM DataBase.
2) Delegare all'utente la gestione degli output implicherebbe pretendere che l'utente conosca l'architettura ed i comandi di basso livello della LcgCAF e dei tool di grid come Globus, SRM e SAM.
3) Nel momento della sottomissione del job non è possibile determinare in quale nodo del Grid tale job verrà eseguito: dopo la sottomissione del job nella LcgCAF il WLS si occupa di effettuare il matching tra risorse disponibili e le richieste pendenti valutando in tempo reale la situazione del Grid (vedere cap 2.3.2 per approfondimenti). Poiché nel modello proposto ogni cluster virtuale sarà associato al suo proprio Local Temporary Storage è impossibile per l'utente sapere a quale Local Temporary Storage farà riferimento il Worker Node che effettua il Monte Carlo.

In fase di progetto sono state considerate due opzioni per risolvere il problema di come informare i Worker Node della destinazione alla quale spedire gli output dei Monte Carlo in modo trasparente all'utente:

- A livello di nodo: ogni nodo appartenente ad un cluster può essere fornito di un framework che si occupi di spedire i dati al Local Temporary Storage.

- A livello di LcgCAF, ossia dotare LcgCAF di un framework opportuno che aggiunga ai tarball degli utenti quelle informazioni e quegli script necessari per inviare correttamente l'output ai Local Temporary Storage.

Lavorare a livello di nodo non sarebbe una buona idea, perché questo porterebbe una scalabilità ridotta ed una manutenzione costosa:

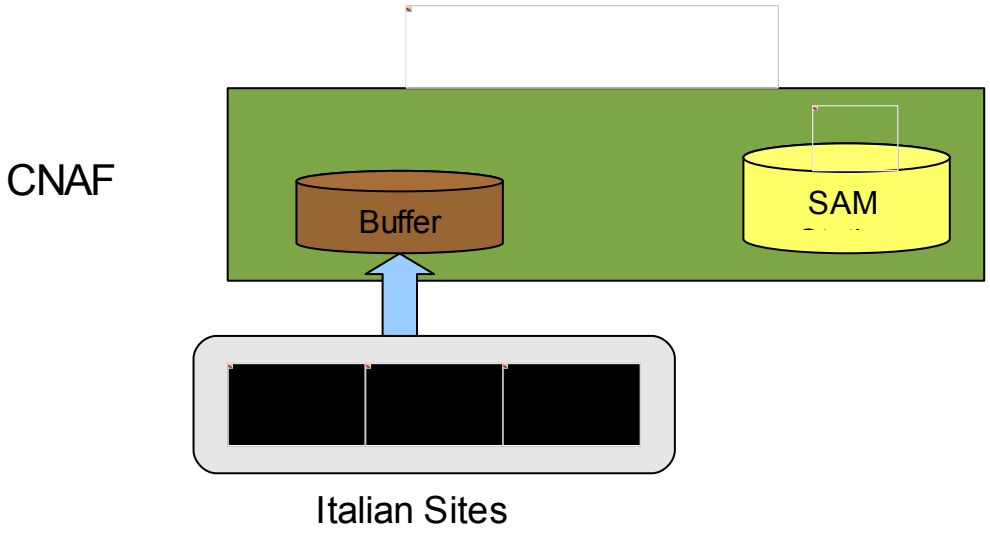
- i nodi potrebbero passare da un virtual cluster ad un'altro qualora diventasse necessario [requisito uno]. In questo caso sarebbe necessario un grosso lavoro per modificare il framework ad ogni cambiamento della topologia.
- Nel caso opposto in cui ci sia invece un cambiamento a livello di Local Temporary Storage che comporti un cambio nelle istruzioni per il trasferimento sarebbe ugualmente necessario modificare le istruzioni per tutti i nodi.

Sarà quindi necessario lavorare a livello di LcgCAF per far sì che vengano aggiunte quelle informazioni e quegli script necessari per l'indirizzamento corretto degli output ai Local Temporary Storage, il compito di implementare quei moduli e' stato affidato al CAFTeam ossia quel gruppo che si occupa della manutenzione della CAF.

In fase di test sono stati utilizzati dei tarball appositamente modificati che vengono inviati ai Worker Node.

Un altro problema relativo allo spostamento dai file dai Worker Node ai Local Temporary Storage è legato ai protocolli da usare, non è possibile infatti fare assunzioni sulle funzionalità offerte dai singoli Worker Node per il data movement l'unica condizione che risulta essere sempre verificata è la presenza dei tool Globus: tale condizione ci viene assicurata dal fatto che tutti i siti appartenenti a LcgCAF sono dotati del middleware Globus toolkit (capitolo 2.3.2)**.**

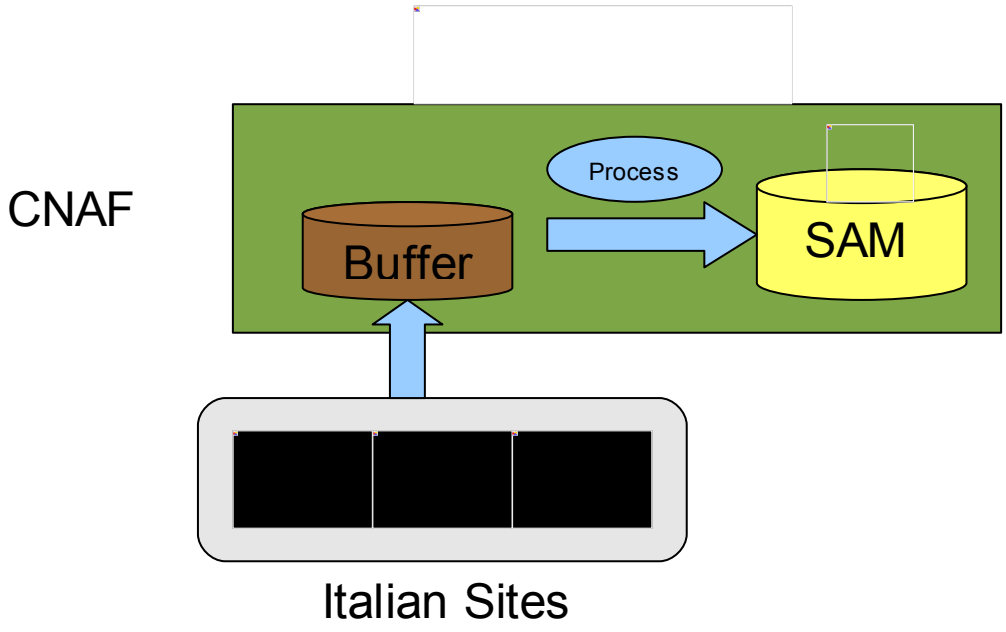
Per il trasferimento dei file da Worker Node al Local Temporary Storage viene quindi utilizzato il tool *globus-url-copy*.Globus-url-copy è quel tool dei Globus toolkit che permette lo spostamento di file tra due nodi di grid.



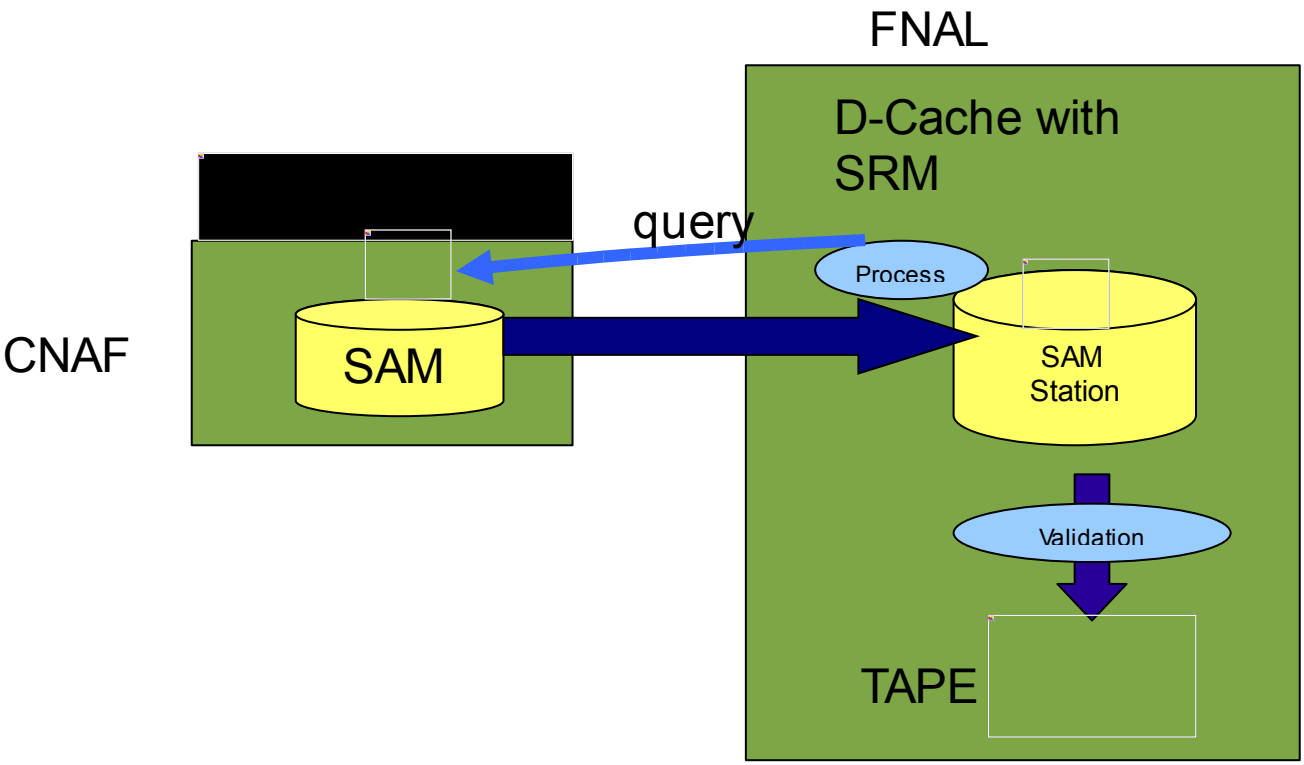
I risultati dei Monte Carlo cosi come i suoi metadata vengono successivamente allocati in un buffer di memoria temporaneo.

Sullo Storage Element presente al CNAF saranno presenti i processi incaricati di controllare la presenza di nuovi files; in caso di esito positivo del controllo si passa alla fase di aggiornamento della SAM station del CNAF con i nuovi dati.

L'autenticazione per tutto il percorso dalla sottomissione del job fino alla scrittura dei dati nel buffer temporaneo viene garantita dal meccanismo di certificazione proxy X509 (vedere appendice sulla autenticazione). Quando gli utenti sottomettono un job tramite LcgCAF il loro ticket di certificazione kerberos, necessario per accedere alle risorse di calcolo del FermiLab e dunque anche a LcgCAF, viene convertito in una certificazione X509, necessaria per accedere alle risorse Grid. (vedere cap. 2.4.1).



Per quel che riguarda il passaggio dai dati dal CNAF al FermiLab la dinamica è leggermente diversa in quanto, in questo caso, la metodologia è Pull: sarà il FermiLab a stabilire in che momento esportare i dati recentemente registrati nelle SAM Station dei LTS per ottimizzare l'uso delle risorse di rete [requisito E_4].



Assieme alla SAM Station dovrà essere presente, al FNAL, un processo che si occupi di interrogare la SAM Station del CNAF per controllare presenza di nuovi file. Qualora si riscontri che nuovi file sono stati dichiarati nella Station del CNAF, tali file verranno importati nella Station del FNAL.

Dopo che i file sono stati copiati con successo presso il FermiLab, la loro copia verra cancellata dagli Storage Element del CNAF.

I file copiati negli Storage Element del FermiLab rimangono in attesa di essere validati. Se i file sono ritenuti significativi dal validation team vengono scritti su nastro in maniera indelebile e rimangono comunque disponibili all'interno della SAM station permettendone cosi la lettura anche da remoto.

Questa parte del prototipo non e' stata realizzata, il trasferimento dei file dal CNAF al FermiLab e in generale da SAM Station SRM x Based a SAM Station SRM x Based è ancora un problema aperto.

Le due metodologie valutate per il trasferimento Station – Station sono:

- *Trasferimento usando i grid file transfer tools:* ossia occuparsi del trasferimento utilizzando le funzionalità di piu' basso livello messe a disposizione dal middleware per lo spostamento di file.

- *Trasferimento "Station to Station"*: utilizzare i servizi messi a disposizione da SAM/SRM per traferire file da una SAM Station SRM based ad un'altra SAM Station SRM based.

Ogni metodo presenta vantaggi e svantaggio anche se nessuno dei due è, allo stato dell'arte, completamente efficace.

Ovviamente la scelta e' quella di implementare la comunicazioni tra SAM Station fruttando i tool di alto livello per la comunicazione Station to Station, vale pero' la pena approfondire questo aspetto della progettazione per poi descrivere lo stato attuale del lavoro.

Trasferimento usando i grid file transfer tools:

E' il metodo di trasferimento a piu' basso livello, non utilizza le utility messe a disposizione da SAM SRM, questo presenta degli svantaggi notevoli i cui piu' significativi sono:

- richiede un notevole sforzo implementativo per effettuare book keeping: al momento del trasferimento dei file bisogna assicurarsi che sia presente una quantita' opportuna di spazio libero e assicurarsi che rimanga tale fino al completamento dello spostamento del file.

– non vengono sfruttati i meccanismi messi a disposizione da SAM SRM per ottimizzare i tempi di trasferimento dei file.

L'unico reale vantaggio offerto da questo metodo e' la possibilita' di ridurre il “time to market”**;** in questo modo infatti non ci si preoccuperebbe dei problemi di compatibilita' tra SAM station con SRM di versioni diverse ed installate su Storege Element di tipo diverso.

Trasferimento Station to Station:

I principali vantaggi nell'uso delle utility di SAM/SRM per lo spostamento di file sono:

- *Gestione dello spazio*: tali utility si occupano di gestire in modo trasparente lo spazio negli SE, a partire dal book keeping dello spazio nello SE di destinazione, alla gestione dei protocolli di comunicazione (tipicamente grid-ftp), fino ad arrivare alla gestione delle repliche, tutti aspetti che altrimenti sarebbe molto complesso gestire in modo corretto utilizzando le funzionalita' di piu' baso livello offerte del Middleware.
- Ottimizzazione: le utlity di SAM/SRM si occupano di ottimizzare i costi, in termini di tempo, delle operazioni di accesso e spostamento dei file (vedere capitolo 4).
- Una maggiore semplicita' nell'implementazione del codice per lo spostamento dei file**.**

Il grande limite di questo approccio è dovuto al fatto che l'integrazione di SAM con SRM è ancora in fase di sviluppo. L'effettiva capacità di due SAM Station, ospitate su due SE diversi, di scambiarsi file e sincronizzarsi va valutata caso per caso.
Sia il CNAF che il FermiLab sono stati dotati di SAM Station SRM based:

- Al CNAF e' stato settato uno Storage Element gestito da StoRM con interfaccia SRM v2.2.

- Al FNAL ,in un primo momento, si e' cercato di utilizzare uno Storage Element d-cache con SRM v1.

Il principale problema incontrato e' che SAM non e' ancora compatibile con SRM v2.2, l'unica versione di SRM perfettamente compatibile con SAM e' SRMv1. Non sara' quindi possibile, almeno in questa prima fase, sfruttare le funzionalita' di SAM per il trasferimento Station to Station.

D'altra parte, installare un interfaccia v2.2 e' stata individuata come unica possibilita' implementativa per due ordini di motivi:

- StoRM funziona solamente con SRM v2.2.
- SRM v2.2 si sta velocemente diffondendo come standard per il Resource Management del grid.

Si e' deciso allora di testare il prototipo attuale usando solamente le funzionalità di SRMv2 per il trasferimento dei file, almeno fino a che il passaggio di SAM da SRM v1 a SRMv2 non sia completato.

A questo scopo si e' deciso di dotare uno Storage Element d-cache con un client SRM v2 ed usare questo spazio testare le fuzionalita' tra StoRM SRM e d-cache SRM.

E' ora importante scegliere quale implementazione di SRM v2 utilizzare.

Nella tabella sotto si illustrano i risultati ottenuti testando la compatibilita' tra interfacce SRM v2 per d-cache e StoRM.

–utilizzando l'implementazione proposta dal FermiLab computing center

	dCache	StoRM
srmping	✗	✗
srmls	✓	✗
srmmkdir	✓	✓
srmrmdir	✓	✓
srmcp (disk to srm)	✓	?
srmcp (srm to srm)	✗	✗

–utilizzando l'implementazione g-Lite proposta da EGEE [http://public.eu-egee.org/]

	dCache	StoRM
Ping	✓	✓
ls	✓	✓
mkdir	✓	✓
rmdir	✓	✓
srm copy (disk to srm)	✓	✓
srm copy (srm to srm)		

 Working

 Not Working

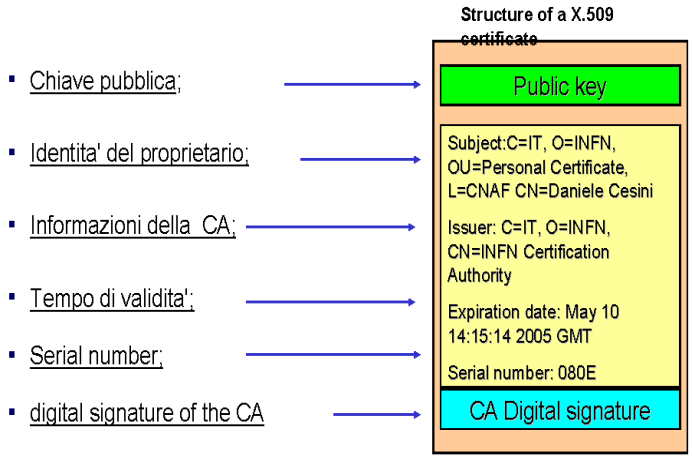
 Work in Progress

In entrambi i casi si e' notato un problema, dovuto a StoRM, nello spostare i file da un'area SRMv2 ad un'altra area SRMv2 col metodo srmcp. Storm non supporta, allo stato attuale, smcp, gli sviluppatori di Storm sono stati avvisati del problema.

Per poter portare avanti il lavoro sara' necessario risolvere i problemi di compatibilita' di SAM con SRM v2.2 e di implementazione delle funzionalita' di SRM per StoRM.

Appendice A:
L'autenticazione su GRID

Per sua natura, una GRID si compone di un insieme di entità organizzate in un'unica struttura collaborativa, ma che possono appartenere ad enti diversi ed essere geograficamente collocate a grande distanza l'una dall'altra. In ogni tipo di applicazione GRID il problema della mutua autenticazione delle componenti coinvolte è cruciale.
Nelle infrastrutture GRID, ogni elemento deve poter attestare la propria identità attraverso un certificato. Esistono diversi standard per la creazione di certificati, uno dei più utilizzati in contesto GRID è il formato X.509.
Un certificato X.509 contiene le informazioni sull'identità del soggetto, la sua chiave pubblica, i propri limiti di validità temporale, l'identità della Certification Authority (CA) che ha firmato il certificato e quindi la firma digitale vera e propria.



Nel momento in cui due elementi A e B del GRID devono interagire, essi devono autenticarsi mutuamente. Il protocollo di autenticazione e' quello standard basato su chiave pubblica e chiave privata.

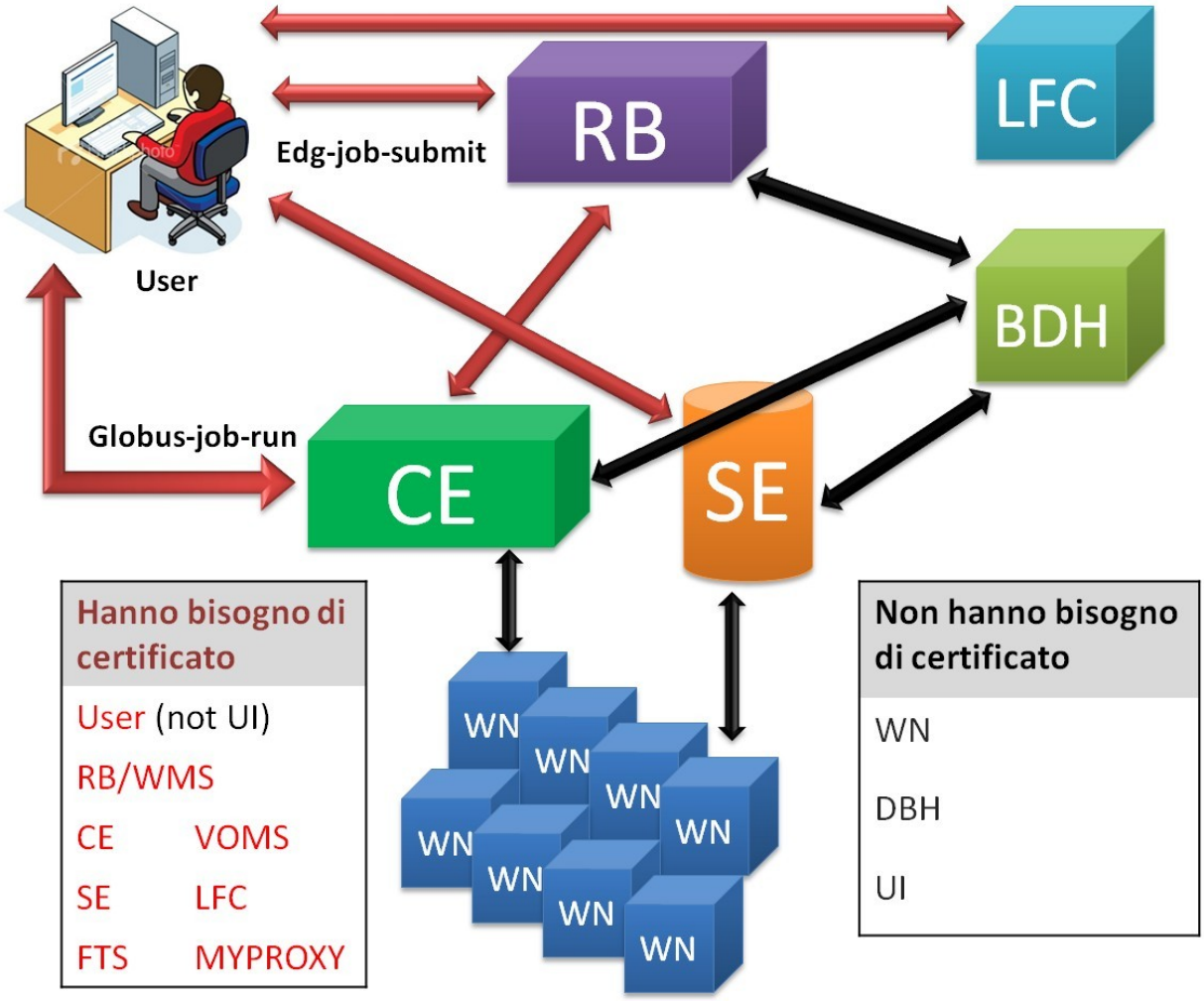
A questo scopo, per prima cosa A stabilisce una connessione con B e gli invia il proprio certificato. L'amministratore di sistema di B avrà in precedenza configurato B in modo che consideri attendibili alcune Certification Authority (CA), di cui avrà installato anche le corrette chiavi pubbliche.
Pertanto B può validare il certificato ricevuto, verificando se la CA a cui fa riferimento è tra quelle conosciute e da considerare attendibili. In caso affermativo, B provvede a calcolare l'hash del testo del certificato, a decriptare la firma digitale e a fare il confronto tra i due risultati.
A questo punto B può avere la certezza che il certificato non sia stato manomesso, ma non può sapere se chi glie lo ha inviato è il legittimo proprietario, oppure qualcuno che ne ha ottenuto illecitamente una copia. Per fare questo genera un testo casuale, lo invia ad A chiedendogli di criptarlo con la sua chiave privata, riceve il risultato e lo decripta con la chiave pubblica presente nel certificato ricevuto. Se il testo finale coincide con quello iniziale significa che l'interlocutore è davvero in possesso della chiave privata associata a quella pubblica che compare nel certificato ricevuto e il certificato ricevuto, su garanzia della CA, associa la chiave pubblica di A all'identità di A. Il procedimento si ripete simmetricamente affinché A possa autenticare B.

Il proxy

In ultima analisi, l'identificazione sicura degli elementi della GRID si basa, oltre che sulla credibilità

della CA del caso, sul fatto che ogni elemento deve avere l'accesso esclusivo alla propria chiave privata. Per questo motivo le chiavi private degli utenti devono essere custodite con la massima cura, oltre che essere salvate in forma criptata tramite una *passphrase*, ossia una password preferibilmente molto lunga, che deve essere fornita dall'utente perché il processo di autenticazione possa completarsi.

Così facendo però, l'utente è costretto a reinserire la passphrase parecchie volte nell'ambito della stessa sessione lavorativa, in quanto ogni volta che esegue un'azione sulla GRID, ossia per ogni comando dato, deve avvenire la mutua autenticazione con gli elementi coinvolti.



Per ovviare a questo problema è stata elaborata un'estensione del protocollo su cui si basa la gestione dei certificati, introducendo il concetto di *proxy*. Un proxy rappresenta l'insieme di una nuova chiave privata e di un nuovo certificato che ogni utente GRID può generare per sé tramite un opportuno comando. Il proxy non è firmato da una CA vera e propria, bensì dallo stesso utente che lo crea, il quale, all'atto della creazione, deve inserire la propria passphrase affinché il sistema possa accedere alla chiave privata necessaria per generare la firma. Inoltre, a differenza del certificato originario dell'utente, un proxy ha una validità molto limitata nel tempo, tipicamente 12 ore, dopodiché diviene inutilizzabile e quindi la chiave privata del proxy può essere salvata in forma non criptata.
Dopo che un utente ha creato un proxy, ogni volta che si rende necessaria una autenticazione con un

elemento della GRID viene inviato sia il proxy sia il certificato originario dell'utente.
I passi da effettuare per la generazione del proxy sono:
· B genera una coppia di chiave pubblica/privata per il proxy certificate.
· B usa la coppia di chiavi per generare una richiesta di certificato, che invia ad A usando un canale sicuro. Questa richiesta di certificato include la chiave pubblica del proxy ma non la chiave privata.
· Supponiamo che A accetti di delegare le proprie credenziali a B. A allora firma la richiesta di certificato utilizzando la propria chiave privata.
· A invia il certificato firmato a B utilizzando un canale sicuro.
· B può ora utilizzare il certificato proxy per agire per conto di A.
Osservare che le chiavi private non vengono mai trasmesse.

Inizialmente verrà analizzato e validato il proxy:
Il processo di convalida di un certificato Proxy è quasi identico al processo di convalida di un certificato X.509. La differenza principale consiste nel fatto che il proxy certificate non è firmato da una Certification Authority, ma da un utente.
Nel nostro esempio il proxy certificate è firmato da A, ciò significa che abbiamo bisogno della chiave pubblica di A per testare l'autenticità del certificato stesso.
Dato che C potrebbe non avere il certificato di A insieme al certificato proxy viene spedito anche il certificato di A. Dato che il certificato di A è firmato da una CA l'ultimo passo necessario è di convalidare la firma della CA.

L'autenticazione non richiede l'immissione di password, perché la sua chiave privata è salvata in forma non criptata e inoltre la sua integrità è garantita dall'ente di certificazione che, in questo caso è l'utente, la cui chiave pubblica è reperibile nel certificato originario.
Il certificato originario è a sua volta validabile in quanto firmato da una vera CA fidata.

Dopo aver creato un proxy, è anche possibile crearne altri a catena, naturalmente senza che venga mai richiesto di inserire una passphrase.
In questo caso ogni proxy funge da ente certificatore del proxy successivo.

MyProxy

La creazione di un proxy è fondamentale quando si intende lavorare in una GRID, tale tecnica presenta

però delle limitazioni significative.
Una prima limitazione è l'impossibilità di eseguire comandi la cui durata superi il tempo di validità del proxy stesso, in quanto alla conclusione del comando non si avrebbe modo di recuperarne l'*output*.
Inoltre l'utilizzo esclusivo dei proxy obbliga l'utente a replicare, o quantomeno a rendere disponibile, su tutti i *client* da cui accede alla GRID la propria chiave privata, necessaria per potere generare il proxy. Replicare la propria chiave privata rappresenta, oltre che una scomodità, un rischio per la sicurezza.
Per questi motivi nelle GRID è previsto l'utilizzo dei cosiddetti myproxy, gestiti da opportuni myproxy server.
Un myproxy server è un servizio presente nelle GRID che si occupa di agevolare l'utente nella gestione dei certificati.
Infatti, tramite opportuni comandi un utente può creare un proxy particolare, detto appunto myproxy, che viene immagazzinato nel myproxy server. La chiave privata relativa al myproxy viene conservata sul myproxy server in forma criptata con una passphrase scelta ad hoc dall'utente e specifica per quel particolare myproxy. La validità temporale di un myproxy è superiore a quella di un normale proxy e tipicamente è di 7 giorni.
Dopo aver creato un myproxy, un utente può ottenere in qualsiasi punto della GRID dei propri proxy validi senza dover avere a disposizione localmente la propria chiave privata e senza dover utilizzare la passphrase del certificato personale: basterà infatti contattare il myproxy server chiedendogli di generare e rilasciare un proxy, e per questa operazione sarà necessario fornire soltanto la passphrase relativa al myproxy.

Inoltre è possibile tramite opportune opzioni generare un myproxy chiedendo al myproxy server di conservare la chiave privata senza proteggerla con passphrase. In questo modo il myproxy server ha la possibilità di generare ed erogare nuovi certificati per l'utente in modo completamente autonomo. Naturalmente, l'utente può decidere di autorizzare il myproxy server a rilasciare un nuovo proxy a chiunque lo richieda, cosa estremamente sconsigliata per ovvi motivi di sicurezza, oppure di attivare l'autorizzazione solo nel caso in cui la richiesta del certificato provenga da parte di servizi specifici.
Uno dei casi in cui questo avviene è quando si rende necessaria una procedura di rinnovo automatico dei certificati, per esempio nel caso in cui si debbano eseguire sulla GRID delle operazioni di cui non si conosce la durata temporale ma che nelle previsioni si protrarranno oltre la data di scadenza del proxy corrente.

Bibliografia

Bibliografia capitolo 1

1] I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid. Enabling Scalable Virtual Organizations

2] Global Grid Forum. Overview of OGSA. http://www.gridforum.org

3] http://www.globus.org/

Bibliografia al capitolo 2

1] http://cdfcaf.fnal.gov/

2] Casarsa, S. C. Hsu, E. Lipeles, M. Neubauer, S. Sarkar, I. Sfiligoi, F.Wuerthwein, *"**The Cdf Analysis Farm**"* AIP Conf. Proc. 794, 275 (2005).

3] LcgCAF F. Delli Paoli,A. Fella,D. Jeans, D. Lucchesi, et al., ***"LcgCAF - The CDF portal to the gLite Middleware"***, Presented at Computing in High Energy and Nuclear Physics, Mumbay, India, Feb 13-17, 2006,148, (2006)

4] J. Antos, M. Babik, D. Benjamin, S. Cabrera, et al., ***"Data processing model for the CDF experiment"*** **physics 9 Jun 2006**

5] .FBSNG Web site. *Next Generation of FBS http://wwwisd.fnal.gov/fbsng/*

6] D. Thain, T. Tannenbaum, M. Livny, ***"Distributed computing in practice: the Condor experience."***, Concurrency - Practice and Experience 17, 2-4, 323 (2005)

7] Kerberos Web site *http://web.mit.edu/Kerberos/*

8] http://vdt.cs.wisc.edu/VOMS-documentation.html

9] http://www.grid.org.tr/servisler/dokumanlar/DataGrid-JDL-HowTo.pdf

10] C. Moretti, I. Sligoi,D. Thain, .Transparently Distributing CDF Software with Parrot., *Presented at Computing in High Energy and Nuclear Physics, Mumbay, India, Feb 13-17, 2006*,**26**, (2006).

11] Generic Connection Brokering, http://www.cs.wisc.edu/ sschang/firewall/gcb/

Bibliografia capitolo 3

1] http://www.isgtw.org/?pid=1000866

2] http://cdfsam-prd.fnal.gov/~sam/data_volume/summary.html

3] http://d0db.fnal.gov/sam_user_api/

4] http://dbb.fnal.gov:8520/cdfr2/databases/

5] Baranovski, A., Bertram, I., Garzoglio, G., Lueking, L., Terekhov, I., Veseli, S., Walker, R., *SAM-Grid: Using SAM and Grid middleware to enable full function Grid Computing.*

6] Carpenter, L., Lueking, L., Moore, C., Pordes, R., Trumbo, J., Veseli, S., Terekhov, I., Vranicar, M., White, S., White, V. *SAM and the Particle Physics Data Grid*, White paper (http://www.ppdg.net/docs/WhitePapers/SAMandPPDG.pdf).

Bibliografia capitolo 4

1] A. Sim, F. Donno and all, *SRM v2.2Specification*, 15 December 2006, http://sdm.lbl.gov/srmwg/doc/SRM.v2.2.html

2] A. Baranovski, R. Mathur https://plone3.fnal.gov/SAMGrid/Wiki/SAM-SRM-Design.pdf

3] http://www.isgtw.org/?pid=1000255