UNIVERSITÀ DI NAPOLI FEDERICO II
FACOLTÀ DI SCIENZE MM.FF.NN.

CORSO DI LAUREA IN INFORMATICA

Anno accademico 2001/2002

Tesi di Laurea

Realizzazione di un sistema computazionale distribuito ad alte prestazioni nell'ambito del progetto VIRGO

Relatore:
Prof. Eliana Minicozzi

Candidato:
Rosario Esposito

INDICE

Elenco delle tabelle

Elenco delle figure

1 Motivazioni e obiettivi del lavoro di stage

1.1 Introduzione

Il lavoro presentato nelle seguenti pagine è frutto di un'attività di stage svolta, principalmente, presso il laboratorio VIRGO[1] della Sezione INFN[2] di Napoli.
Gli obiettivi principali dello stage sono i seguenti:

1) Lo studio di un'architettura hardware/software e l'implementazione di un prototipo di Linux farm[3] ad alte prestazioni per l'esecuzione e il test, in locale, di procedure di analisi dei dati acquisiti dall'interferometro di VIRGO.

   Le definizioni e i dettagli di questa prima parte del lavoro sono discussi nell'intero capitolo 2. In particolare, i paragrafi 2.2, 2.3 e 2.4 contengono la descrizione hardware e software della farm inizialmente realizzata e le motivazioni per la sua successiva modifica. Tale modifica è illustrata nel paragrafo 2.5 mentre, nel paragrafo 2.6, sono descritti i test effettuati sull'architettura proposta.

2) L'estensione della farm al modello di "griglia computazionale"[4] geograficamente distribuita, nell'ambito del progetto INFN-Grid[5]. Tale attività, illustrata nell'intero capitolo 3, ha come scopo la realizzazione di un'infrastruttura di base su cui effettuare test di analisi dei dati di VIRGO in un ambiente "grid-oriented"[4]. In particolare, il paragrafo 3.4.4 contiene la descrizione della realizzazione di tale infrastruttura e dei test effettuati su di essa; in tale paragrafo, quindi, è riferita la seconda parte del lavoro originale svolto.

---

[1] Nei paragrafi 1.2 e 1.3 viene presentata una descrizione del progetto VIRGO e delle problematiche di calcolo distribuito da dover affrontare
[2] INFN: Istituto Nazionale di Fisica Nucleare
[3] Per "Linux farm" si intende un insieme di calcolatori con sistema operativo Linux che collaborano per l'esecuzione di processi di calcolo
[4] I termini "griglia computazionale" e "ambiente grid-oriented" saranno introdotti e descritti in dettaglio nell'intero capitolo 3
[5] La descrizione del progetto INFN-Grid è presentata nel paragrafo 3.3.2

Questi due obiettivi, strettamente correlati, sono finalizzati alla realizzazione di un sistema di calcolo ad alte prestazioni "multi-purpose" da utilizzare, in locale, come un potente strumento per l'analisi dei dati e, in un contesto di calcolo distribuito, come risorsa disponibile all'interno di una griglia computazionale.

1.2 Il progetto VIRGO

Obiettivo principiale del progetto VIRGO è quello della prima rivelazione diretta di onde gravitazionali emesse da sorgenti astrofisiche mediante tecniche interferometriche. Tale progetto, formalmente approvato nel 1994 come collaborazione scientifica tra due enti di ricerca, l'italiano INFN (Istituto Nazionale di Fisica Nucleare) e il francese CNRS (Centre National de la Recherche Scientifique), è attualmente in fase di realizzazione presso Cascina (Pisa). Il completamento di tale progetto è previsto per la fine del 2002 ed i primi run sono previsti per l'inizio del 2003, anno in cui comincerà la fase di Commissioning di VIRGO, necessaria per la messa a punto del rivelatore. [1]

L'antenna VIRGO è sostanzialmente un interferometro laser di tipo Michelson costituito da due bracci ortogonali, ciascuno lungo 3 chilometri, costituito da una cavità Fabry-Perot di finesse 50. Tali cavità servono per realizzare un braccio di un Michelson equivalente di lunghezza pari a 150 km, mediante riflessioni multiple. La funzione di tali cavità Fabry-Perot è duplice: hanno la funzione di aumentare la potenza laser in cavità migliorando la sensibilità dell'interferometro, e, contemporaneamente, la funzione di adattare la banda di misura dell'interferometro ad una banda di emissione di sorgenti di onde gravitazionali in cui è elevata la probabilità di rivelare segnale gravitazionale. Questa banda, nel caso di VIRGO, è pari a 4 Hz – 6 kHz.

Infatti, in questa banda dovrebbe essere possibile rivelare segnali gravitazionali emessi da diverse classi di sorgenti astrofisiche: burst, segnali gravitazionali emessi a seguito di esplosioni di supernovae; chirp, segnali gravitazionali emessi da binarie coalescenti; quasi-sinusoidi, segnali gravitazionali emessi da Pulsar. [1]

La rivelazione di onde gravitazionali da parte di interferometri richiede un'estrema sensibilità. In particolare, VIRGO è stato progettato per raggiungere sensibilità di 10^{-21} m/sqrt(Hz) a 10 Hz. Il raggiungimento sperimentale di questa sensibilità richiede notevoli qualità e accuratezza nella progettazione dei diversi componenti

dell'interferometro. Infatti solo abbassando notevolmente le sorgenti di rumore interne (ottico, meccanico, elettronico, ecc.) e quelle esterne (sismico, elettromagnetico, acustico, etc.) o comunque controllandole e misurandole in modo continuativo è possibile pensare di rivelare tali onde gravitazionali. Gli scienziati italiani e francesi si sono prefissi di raggiungere questo difficile obiettivo e stanno sviluppando delle tecniche molto avanzate nel campo dei laser ultrastabili ad alta energia, nel campo della realizzazione di specchi ad alta riflettività, nel campo dell'isolamento sismico ed ambientale in generale e nel campo dei controlli digitali di posizione ed allineamento ottico.

VIRGO, infatti, utilizza una nuova generazione di laser ultrastabili e gli oscillatori più stabili mai costruiti. Per evitare movimenti dei componenti ottici indotti da rumore sismico che mimerebbero segnali gravitazionali, gli specchi dell'interferometro sono isolati mediante un elaborato sistema di pendoli compositi, alto 10 metri (superattenuatore). Poiché la presenza di gas residui potrebbe perturbare le misure, il raggio di luce deve propagarsi sotto vuoto spinto. I due tubi, lunghi 3 km e con un diametro di 1.2 m, sono tra i più grandi sistemi da vuoto spinto mai realizzati al mondo.

VIRGO svolgerà la sua attività giorno e notte e sarà in ascolto di tutti i segnali gravitazionali provenienti da qualsiasi parte dell'universo. I segnali saranno rilevati, registrati e pre-analizzati attraverso un complesso sistema di acquisizione e calcolo on-line. Si prevede un flusso continuo di 5 Mbyte/sec di dati che dovranno essere archiviati, su disco e su nastro, ed analizzati sia on-line che off-line. Tutti questi dati verranno poi successivamente collezionati e resi disponibili alla comunità scientifica per ulteriori analisi. [1]

Figura 1-1 Veduta aerea dell'interferometro di Cascina

1.3 Il problema del calcolo in VIRGO

I rivelatori interferometrici di Onde Gravitazionali pongono significativi problemi nei riguardi dell'analisi dei dati. Il volume di dati prodotti e la potenza di calcolo necessaria per la loro elaborazione raggiungono ordini di grandezza tali da richiedere necessariamente uno sforzo di ricerca e di sperimentazione indirizzati verso nuove soluzioni tecnologiche.

Come quadro generale di riferimento, è opportuno ricordare che VIRGO viene sviluppato in parallelo ai progetti anglo-tedesco (GEO), americano (LIGO) e giapponese (TAMA). La collaborazione tra i vari progetti è strettissima. A tale scopo si è già raggiunto un accordo sulla omogeneizzazione del formato dei dati che coinvolge praticamente tutti i gruppi sopra citati ed è in corso di formalizzazione l'accordo di scambio completo dei dati stessi tra VIRGO e LIGO. Per ciascun esperimento il volume dei dati prodotti continuativamente si misura in Tbytes (con un rate stimabile in diversi Mbytes/s) e le potenze di calcolo necessarie per l'analisi in Tflops.

I dati prodotti dall'interferometro centrale di VIRGO devono essere analizzati sia a fini diagnostici della macchina sia allo scopo di testare e raffinare gli algoritmi di analisi. I dati dell'interferometro completo sono previsti a partire dal 2003. Entro questa data, le procedure d'analisi devono essere mature ed un forte nucleo delle risorse di calcolo necessarie deve essere disponibile.

Attualmente VIRGO è ancora nella fase di Commissioning dell'Interferometro Centrale (CITF), un interferometro a bracci corti realizzato nella zona centrale dell'interferometro, necessario per una prima messa a punto dei sistemi di controllo, acquisizione, ecc.. Il sistema d'acquisizione dati di VIRGO genera un flusso di dati che può essere sintetizzato nella Tabella 1-1 [2]:

	Raw Data	**Processed Data**
Flow/s	4.0 MB	.3 MB
Flow/day	350 GB	26 GB
Flow/year	126 TB	9.5 TB
Tapes/year (50 GB DLT)	2500	190

Tabella 1-1 Flusso dei Raw e Processed Data per differenti scale dei tempi

I dati sono raccolti in strutture chiamate “frame” che costituiscono l’insieme dei “Raw Data”. Vista la natura ed il livello di frontiera del rivelatore, questi flussi sono destinati a variare per eccesso o per difetto in funzione dell’abilità degli sperimentatori nel comprendere e modellare tutte le sorgenti di rumore proprie dell’antenna. Il numero dei nastri magnetici che deve essere utilizzato in un anno è stato valutato assumendo nastri del tipo DLT di 50 GB di capacità. I “Processed Data”, ottenuti con una elaborazione on-line, sono immagazzinati sui dischi di un cluster di computer e sono accessibili tramite un sistema di distribuzione dati. [3]

Per non danneggiare irreversibilmente i dati nel momento iniziale dell’elaborazione e della qualificazione del segnale, non sarà mai possibile utilizzare per calcolo off-line le risorse di calcolo della struttura dedicata all’elaborazione on-line dei dati; è anzi un obbligo evitare interferenze tra le due strutture computazionali di on-line ed off-line.
Riportiamo qui di seguito una lista [2], necessariamente parziale, relativa alle attività di calcolo off-line per VIRGO. Alcune di esse sono già in pieno svolgimento, come quella relativa alla simulazione strumentale, altre sono in fase di sviluppo, come la scrittura di algoritmi relativi alla ricerca delle varie categorie di segnali. Vediamo di elencarle almeno in modo sommario, cercando di metterne in luce il rispettivo peso in termini di richiesta di risorse necessarie ai fini del calcolo:

- **Simulazione dati di VIRGO**. Questa attività è basata sull’elaborazione di dati simulati. Il flusso dati, cosi come il formato, è analogo a quello dei dati reali. La simulazione è fondamentale per definire l’efficienza dei vari passi di analisi previsti lungo la catena d’elaborazione dei dati.

- **Caratterizzazione delle sorgenti di rumore dell’interferometro**. L’analisi è strettamente connessa con lo sviluppo della simulazione ed è attualmente in corso da due anni per il Commissioning dell’interferometro centrale (CITF). Essa implica un pesante lavoro di rielaborazione dei Raw Data.

- **Ricerca off-line di segnali da binarie coalescenti**. Le necessità di calcolo si riferiscono essenzialmente alla ricerca di possibili candidati per sistemi binari in coalescenza. Questa attività richiede un ampio spazio disco per lo storage dei dati e notevoli risorse di elaborazione.

- **Ricerca di segnali continui da stelle di neutroni asimmetriche in rapida rotazione**. La ricerca è basata sullo sviluppo di tecniche di analisi "sub ottimali" che consentono di affrontare la ricerca in tutto cielo di sorgenti di frequenza di rotazione sconosciuta, limitando la potenza di calcolo necessaria nel range di qualche Tflops.

- **Network Analysis**. Lo scambio dati con altri rivelatori è fondamentale per questo tipo di esperimenti. VIRGO scambierà dati con i due interferometri americani di LIGO e con altri rivelatori distribuiti nel mondo per effettuare analisi dati in coincidenza al fine di aumentare il livello di confidenza relativo alla rivelazione. Questa esigenza richiede di definire nuovi protocolli per i quali sono previsti dei test, basati soprattutto sullo scambio di dati del sistema di monitoraggio ambientale. La ricerca dei chirp, ad opera di gruppi di persone geograficamente distribuiti, concorrerà certamente ad aumentare le necessità di calcolo e di velocità di trasmissione dei dati dell'esperimento.

Dallo scenario appena descritto emerge la forte necessità di realizzare dei sistemi di calcolo ad alte prestazioni capaci di elaborare in maniera efficiente l'enorme quantità di dati prodotta dall'interferometro.

In particolare, il gruppo VIRGO della Sezione INFN di Napoli, che ha la responsabilità della ricerca off-line di segnali provenienti da binarie coalescenti, sta studiando tecniche e metodologie per la realizzazione di sistemi di calcolo distribuito ad alte prestazioni basati su farm di computer.

Per riassumere i principali problemi che devono essere presi in considerazione sono:

1) Potenza di Calcolo richiesta
2) Analisi dati in coincidenza con altri esperimenti
3) Distribuzione dati

1.3.1 Potenza di Calcolo richiesta

Dalla Tabella 1-2 [2] si evince che per un'analisi completa relativa alla rivelazione del segnale gravitazionali emessi da binarie coalescenti è necessaria una notevole potenza di calcolo (>300 Gflop) mentre ancora maggiore potenza di calcolo è necessaria per la rivelazione di segnale gravitazionali emessi da pulsar (>1 Tflop).

Sorgente	**Potenza richiesta**
Burst	10 GFlop
Binarie Coalescenti	> 300 GFlop
Pulsar	> 1 TFlop

Tabella 1-2 Potenza di calcolo richiesta per l'analisi dati in relazione al tipo di sorgente gravitazionale

Queste richieste pongono seri problemi alla realizzazione di un sistema di calcolo efficiente in quanto è difficile pensare che attualmente tale potenza possa essere concentrata in un unico posto e comunque dedicata interamente ad un'unica attività di analisi. E' pertanto logico e necessario pensare sia di utilizzare nelle diverse sedi partecipanti al progetto VIRGO macchine capaci di effettuare calcolo distribuito sia di utilizzare risorse computazionali distribuite in sedi diverse. Questo tipo di scelta si rende necessaria al fine di facilitare la gestione e l'adattabilità dei sistemi alle esigenze che man mano dovessero porsi.

1.3.2 Analisi Dati in Coincidenza

L'analisi dati in coincidenza fra rivelatori assume un ruolo importante soprattutto nell'ottica di una prima rivelazione di onde gravitazionali. Infatti, data la debolezza intrinseca del segnale gravitazionale e la presenza di numerose possibili sorgenti rumore, peraltro interpretabili come segnali gravitazionali, un elemento che potrebbe aumentare considerevolmente la certezza di una rivelazione è

senz'altro quello di effettuare analisi in coincidenza con altre antenne della stessa classe di VIRGO in una prima fase, ma di estendere tale analisi anche ad antenne di classe diversa in un prossimo futuro, come ad esempio le antenne a barra, la cui banda di rivelazione è all'interno della banda di rivelazione degli interferometri.

Per realizzare tale obiettivo sono necessari dei solidi supporti informatici che garantiscano, da un lato, meccanismi per la condivisione delle informazioni (con protocolli standard ed efficienti) e, dall'altro, lo sviluppo di algoritmi e procedure di analisi per il confronto e il controllo sull'affidabilità dei dati provenienti da fonti diverse. [2]

1.3.3 Distribuzione Dati

Un ultimo aspetto da tenere in considerazione è che la numerosa comunità di ricercatori, geograficamente distribuita, deve poter accedere efficientemente al grande volume di dati prodotto dall'esperimento.

A fronte di questa esigenza e considerato il rapido e continuo aumento della velocità di interconnessione delle reti di computer, è stato proposto a livello internazionale di sviluppare un sistema trasparente di distribuzione dell'onere computazionale e di memoria, basato su centri di elaborazione di vari gradi di potenza, fortemente interconnessi tra loro.

In tale ottica si inserisce il progetto GRID che nasce con lo scopo di realizzare griglie computazionali di nodi geograficamente distribuiti. GRID è oggetto di grande attenzione e si sta concretizzando in proposte operative di ricerca e sviluppo che coinvolgono enti quali la NASA e la NSF negli USA, l'ESA, il CERN, l'INFN e le agenzie di ricerca dei più grandi paesi europei (CNRS, Ppark, MaxPlanck). Grid, infatti, è un'architettura distribuita che, in modo innovativo, affronta il problema della condivisione di risorse su larga scala e per le sue caratteristiche facilita lo sviluppo di nuove applicazioni.

Il termine "GRID-computing" identifica, in generale, un'infrastruttura di elaborazione distribuita nata per rispondere alle esigenze poste da studi avanzati nel campo della scienza,

dell'ingegneria, della fisica, della biologia e di qualsiasi campo dove vi sia la necessità di elaborare grandissime quantità di dati.

L'infrastruttura Grid si occupa del coordinamento delle risorse condivise, in un ambiente sicuro, dalle cosiddette Virtual Organization (VO), gruppi di persone ed entità geograficamente distribuite, multi-istituzionali, in dinamica evoluzione e con interessi comuni (scientifici, economici, amministrativi, etc.). La condivisione diventa così accesso diretto a computer, dati, software ed altre risorse e non più un semplice scambio di file. La sicurezza integrata garantisce l'autenticazione e l'autorizzazione dei partecipanti, il monitoring delle risorse e policy tra organizzazioni differenti.

Infine, un'architettura Grid lavora in assenza di locazione e controllo centralizzati, esistenza di relazioni di fiducia preesistenti e configurazioni omogenee.

2 Architettura proposta per i processi di computazione in locale: farm di macchine Linux

2.1 Introduzione

In questa sezione verrà descritta l'architettura hardware e software del prototipo di farm realizzata nel laboratorio VIRGO di Napoli come strumento di test per l'analisi di binarie coalescenti.

2.2 Descrizione della farm: Hardware

Il prototipo della farm [18][21] consiste in 12 macchine (SuperMicro 6010H). Ogni macchina è dotata di 2 processori Pentium III – 1 GHz, 512 Mbyte di ram, 2 schede Fast Ethernet e un disco SCSI da 18 Gbyte.

Uno dei nodi della farm, configurato come master, è dotato di una scheda Gigabit Ethernet. I nodi della farm sono interconnessi attraverso 2 switch 100baseT/X a 24 porte. La comunicazione tra i nodi avviene su 2 reti private indipendenti, realizzate utilizzando i 2 collegamenti Fast Ethernet presenti su ogni macchina. La connessione con la rete esterna è invece garantita dal collegamento Gigabit sul nodo master.

Le applicazioni e i dati necessari alla farm sono forniti da un Alpha Server 4100. Tale server è dotato di 2 processori Alpha a 500Mhz, 256 Mbyte di RAM, un set di dischi SCSI per un totale di 144 Gbyte e 2 schede Fast Ethernet. Il server usa uno di questi collegamenti Fast Ethernet per comunicare con la rete privata della farm e l'altro per l'accesso alla LAN pubblica attraverso lo switch del laboratorio VIRGO. La Figura 2-1 riassume l'architettura hardware appena descritta:

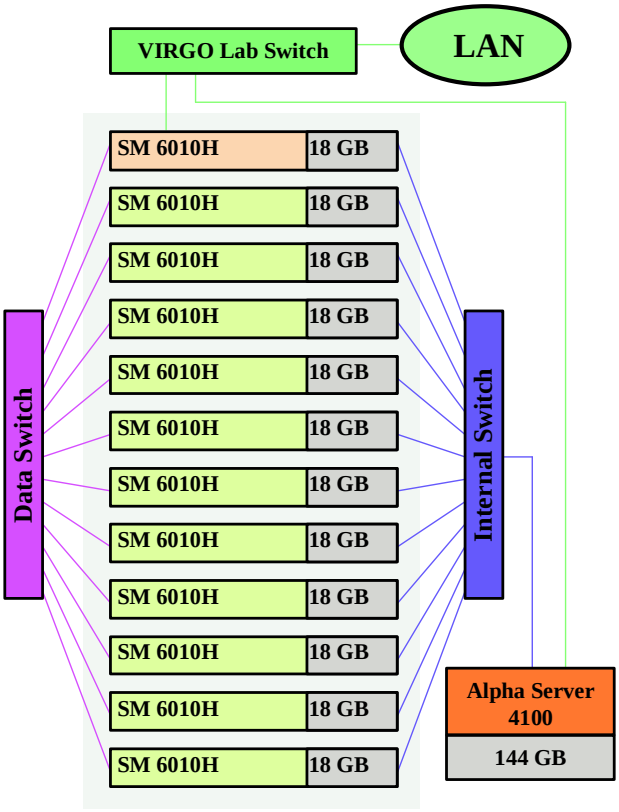


Figura 2-1 Architettura della farm VIRGO di Napoli

2.3 Descrizione della farm: Software

Il sistema operativo installato sulla farm VIRGO di Napoli è Linux (distr. RedHat 7.2 con kernel 2.4.17). Sull'Alpha Server gira invece il sistema operativo Compaq Tru64 (ver. 5.0a). Le librerie installate sulla farm sono le seguenti: Grasp 1.9.8, Numerical Recipes e MPICH 1.2.1.

- Grasp[6] è una libreria per l'analisi dei segnali gravitazionali, scritta in C;

- Numerical Recipes[7] è una libreria di funzioni matematiche utilizzate nella scrittura di algoritmi di analisi. Una parte di questa libreria è usata da Grasp;

- MPICH[8] è un'implementazione di MPI (Message Passing Interface), un set di librerie standard per ambienti di calcolo parallelo;

La scelta di Linux come sistema operativo della farm è dettata da diverse motivazioni. Linux è un sistema operativo opensource efficiente, stabile, affidabile ed estremamente potente, dotato di un completo ambiente di sviluppo e supportato da una comunità di utilizzatori estremamente vasta. La sua versatilità e la sua rapida e continua evoluzione lo hanno reso, nel corso degli ultimi anni, uno dei sistemi operativi più utilizzati, soprattutto negli ambienti di ricerca.
I cluster [4] Linux possono essere usati per:

- garantire affidabilità e disponibilità dei servizi (High Availability).

- ottimizzare l'utilizzo delle risorse attraverso la distribuzione del carico sui nodi costituenti il cluster (Load Balancing)

[6] http://trantor.bioc.columbia.edu/grasp/
[7] http://www.nr.com/
[8] http://www-unix.mcs.anl.gov/mpi/mpich/

- affrontare problemi che, per dimensione e risorse di calcolo necessarie (come il caso di VIRGO), non possono essere trattati da un singolo calcolatore (**High Performance Computing**).

Con l'espressione HPC (High Performance Computing) si identifica un sistema in cui le risorse di calcolo hanno prestazioni particolarmente elevate. Tradizionalmente i sistemi HPC sono stati implementati su macchine monolitiche capaci di alloggiare più CPU e con bus dati estremamente veloci ed efficienti. Più recentemente, grazie all'elevata potenza di calcolo raggiunta dai sistemi a basso costo (i comuni personal computer) e alla disponibilità di sistemi operativi come Linux, è stato possibile realizzare dei cluster HPC di comuni PC interconnessi da una rete locale, con molti vantaggi quali:

- costi decisamente inferiori rispetto ai sistemi monolitici;

- utilizzo di tecnologie comuni e componenti disponibili attraverso la grande distribuzione (COTS: Commodity Off The Shelf);

- architettura scalabile dei sistemi;

Generalmente la configurazione di una farm di PC presenta alcune difficoltà dovute principalmente all'elevato numero di elementi hardware da gestire e tenere sotto controllo.
Il setup di una farm può essere talvolta un processo lungo e laborioso e la probabilità che l'amministratore di sistema commetta degli errori di configurazione cresce all'aumentare dei nodi da installare e gestire. Inoltre, la manutenzione del software installato su una farm può risultare molto complicata quando il numero di workstation da gestire è elevato. L'installazione di un nuovo pacchetto, l'update di una libreria o la rimozione di un'applicazione software possono diventare dei compiti onerosi quando devono essere replicati su tutti i nodi della farm.

Una farm, infine, si può presentare come uno strumento difficile da utilizzare, poiché spesso l'esecuzione di job paralleli richiede

una conoscenza degli host disponibili in un determinato momento e un'allocazione manuale delle risorse da parte degli utenti.
L'architettura software della farm di VIRGO tenta di risolvere i problemi appena descritti facilitando i compiti di installazione, manutenzione e utilizzo delle risorse di calcolo disponibili.

L'architettura è basata sull'utilizzo di nodi "diskless" (privi di disco di sistema). Uno dei nodi, che svolge il ruolo di master, esporta agli altri nodi il sistema operativo e le applicazioni.
Una delle due reti private della farm è usata per il boot remoto dei nodi e lo scambio di dati con il master. Dal punto di vista del management della farm, tutti i nodi (eccetto il master) possono essere visti come unità prive di disco rigido. Di fatto, i nodi utilizzano il proprio disco locale soltanto come area di spool per salvare dati temporanei in fase di elaborazione, se necessario.
Questa soluzione offre diversi vantaggi: l'installazione e l'upgrade software della farm diventa molto facile in quanto tutte le modifiche vengono fatte su una sola macchina, il master, e diventano visibili istantaneamente da tutti i nodi. Ciò riduce drasticamente i tempi di amministrazione dei nodi slave. Inoltre l'assenza di un disco sui nodi comporta una sensibile riduzione dei costi di sistema.

Il boot da rete dei nodi diskless è realizzato con l'ausilio di Etherboot[9] [5], un tool opensource che consente di creare una ROM (Read Only Memory) contenente un codice di startup (dipendente dal tipo di scheda Ethernet installata su una macchina) leggibile da un qualsiasi dispositivo di boot (un floppy, un disco rigido, la EEPROM di una scheda di rete…).
Esaminiamo in dettaglio la sequenza di boot di un generico nodo diskless:

- Il codice di startup generato da Etherboot viene letto ed eseguito all'accensione

- Il nodo invia sulla rete una richiesta utilizzando il protocollo Bootp (o DHCP) per ottenere un indirizzo IP da un server (il nodo master)

[9] http://www.etherboot.org

- Una volta ottenuto l'indirizzo IP il nodo esegue il download del kernel di Linux dal server utilizzando il protocollo TFTP

- Il kernel di Linux viene caricato in memoria e sul nodo inizia la vera sequenza di bootstrap del sistema operativo

- Il kernel, precedentemente compilato con l'opzione "root filesystem over NFS", fa sì che il nodo esegua un mount del suo root filesystem da un server NFS (il nodo master)

- Una volta montato il root filesystem, il nodo completa la sua sequenza di boot montando il proprio disco locale in una directory "/scratch" e inizializzando i vari servizi di sistema.

Poiché ogni nodo diskless deve avere un root filesystem da montare via NFS, il nodo master dovrebbe avere tante immagini di root filesystem (una per ogni client) da esportare agli altri nodi. Non è possibile esportare uno stesso root filesystem a tutti i client per ovvi motivi (ad esempio i file di configurazione e i file di log e sono diversi per ogni client).

Sul nodo master della farm di VIRGO, quindi, è stato installato ClusterNFS[10] [11][12], una versione modificata dell'NFS server standard. ClusterNFS è un prodotto opensource che risolve l'increscioso inconveniente di avere tante immagini di root filesystem da esportare ai nodi della farm. Con ClusterNFS tutti i nodi (compreso il master) possono condividere lo stesso root filesystem utilizzando le seguenti convenzioni:

- Tutti i file sono condivisi per default

- Un file di nome xxx comune a tutti i client NFS (ma non al server) deve essere rinominato come xxx$$CLIENT$$

- Un file xxx specifico per un certo client deve essere rinominato come xxx$$HOST=nomehost.dominio$$ oppure

[10] http://clusternfs.sourceforge.net/

xxx$$IP=aaa.bbb.ccc.ddd$$ dove “nomehost.dominio” è il nome del client oppure “aaa.bbb.ccc.ddd” è l’indirizzo IP del client.

2.4 Limitazioni dovute all'allocazione statica dei processi

La configurazione della farm con nodi diskless facilita di gran lunga l'installazione e la manutenzione dei nodi. Resta comunque il problema dell'allocazione statica dei nodi in fase di sottomissione dei job da parte degli utenti.

Il crescente interesse per i cluster di workstation, sia per il calcolo parallelo che per quello sequenziale, induce, inevitabilmente, alla creazione e alla revisione degli algoritmi di migrazione dei processi tra nodi al fine di garantire alte performance e un utilizzo intenso delle "idle workstation" (macchine la cui CPU è inattiva ma disponibile per l'esecuzione di job).

Attualmente quasi tutti i tools utilizzati per il calcolo parallelo non prevedono la migrazione dei processi verso le workstation più scariche; l'allocazione processo-CPU è di tipo statico. Gli svantaggi sono evidenti: dopo l'allocazione iniziale dei processi su varie CPU, il carico della farm inizia ad essere non uniforme (delle CPU sono idle poiché hanno appena terminato i loro processi, mentre altre sono utilizzate, in maniera concorrente, da più processi).

2.5 Allocazione dinamica dei processi: MOSIX

2.5.1 Cos'è MOSIX

MOSIX[11] [14][15] é un'estensione dei kernel Unix-like, come Linux, ed è costituito da una serie di algoritmi per la condivisione adattiva delle risorse. Questi algoritmi sono progettati in modo da rispondere, in tempo breve, alle variazioni di utilizzo delle risorse, bilanciando il carico computazionale all'interno di un cluster di workstation.

MOSIX (Multicomputer OS for UnIX) nasce nei primi anni 80 su PDP-11/70 all'università di Gerusalemme. La sua prima implementazione viene realizzata su un sistema Unix BSD per PDP-11/70. Negli anni successivi MOSIX viene implementato anche su VAX 11/780 e su Motorola / bus VME con il supporto economico dell'esercito israeliano.
Nel 1994 viene rilasciata una versione di MOSIX per BSD su piattaforma Intel e nel 1997 MOSIX diventa un progetto GNU per Linux. Dal 1997 ad oggi gli sviluppatori di MOSIX lavorano a stretto contatto con quelli del kernel di Linux, contribuendo al continuo rilascio di patch e all'evoluzione di questo sistema operativo. Nel Novembre 2001 il progetto MOSIX ha subito uno split dovuto a motivi commerciali ma la parte opensource del progetto, che prende il nome di OpenMOSIX[12], continua ad essere supportata da validi sviluppatori ed evolve di pari passo con il kernel di Linux.

MOSIX è progettato in modo da garantire il load-balancing tra le workstation di un cluster allo scopo di fornire un supporto efficiente per l'esecuzione di task paralleli o sequenziali.
Questo obiettivo è raggiunto implementando la migrazione di processi, eventualmente in esecuzione, da un nodo all'altro, in modo trasparente, prevenendo la drastica diminuzione (trashing) dell'efficienza delle applicazioni in caso di continui trasferimenti di blocchi di dati tra la RAM e la memoria di massa (memory swapping). Il compito di MOSIX è quello di garantire alte prestazioni globali e di creare un ambiente multi-user e time-sharing.

[11] http://www.mosix.org/
[12] http://www.openmosix.org

2.5.2 Componenti di MOSIX

L'implementazione corrente di MOSIX [19][20] ne consente l'utilizzo su cluster di workstation x86/Pentium-based con architettura UniProcessor (UP) o Symmetrical Multi Processor (SMP), connesse attraverso una rete locale standard. Le configurazioni possibili vanno da piccoli cluster di PC connessi con una rete Ethernet a 10Mbit fino a cluster di sistemi ad altissime prestazioni costituiti da migliaia di server SMP interconnessi da LAN Gigabit, ATM o Myrinet.
L'architettura di MOSIX si basa su due componenti:

- un meccanismo per la migrazione di processi eventualmente in esecuzione (**Preemptive Process Migration**)

- un set di algoritmi per la condivisione adattiva delle risorse

2.5.3 Preemptive Process Migration

Il modulo PPM (Preemptive Process Migration) è in grado di far migrare ogni processo, in qualsiasi momento, verso uno qualsiasi dei nodi disponibili del cluster. Di solito le migrazioni vengono decise in maniera automatica dal sistema operativo, sulla base delle informazioni fornite da algoritmi di condivisione delle risorse. Gli utenti però possono forzare manualmente i propri processi a migrare verso altri nodi. Tali migrazioni manuali possono essere utili per garantire una particolare politica di gestione dei processi o per testare differenti algoritmi di scheduling. L'amministratore di sistema (superutente) può inoltre definire delle politiche generali e stabilire quali nodi sono disponibili per la migrazione dei processi.

A ciascun processo viene associato un UHN (Unique Home Node) che identifica il nodo in cui è stato creato (in genere rappresenta il server al quale sono collegati gli utenti).
Nel modello SSI (Single System Image) di MOSIX ogni processo sembra essere in esecuzione sul proprio UHN e tutti i processi di una sessione utente condividono l'ambiente e le risorse dell'UHN.

In realtà i processi vengono creati sul server al quale sono collegati gli utenti, ma, successivamente, migrano in maniera trasparente verso i nodi meno carichi. Il PPM è lo strumento principale per gli algoritmi di gestione delle risorse. La granularità della distribuzione del carico in un cluster MOSIX è a livello di processo.

Gli utenti possono eseguire le loro applicazioni inizializzando più processi su un solo nodo (gateway), sarà compito di MOSIX far migrare questi processi verso i nodi del cluster. Se durante l'esecuzione dei vari task vengono rese disponibili nuove risorse, allora gli algoritmi di condivisione delle risorse provvederanno alla ridistribuzione dei processi sui nodi della farm. La capacità di smistare dinamicamente i processi all'interno del cluster è particolarmente importante nella creazione di ambienti di esecuzione multi-user e time-sharing.

Un processo in fase di migrazione viene diviso in due contesti; lo "user context", che può essere migrato a tutti gli effetti, e il "system context" che è dipendente dall'UHN e non può essere migrato.

Lo user context, chiamato "remote", contiene il codice del programma, lo stack, i dati, le mappe di memoria e i registri del processo. Il system context, chiamato "deputy" contiene una descrizione delle risorse a cui il processo è legato e uno stack per l'esecuzione di codice di sistema per conto del processo.

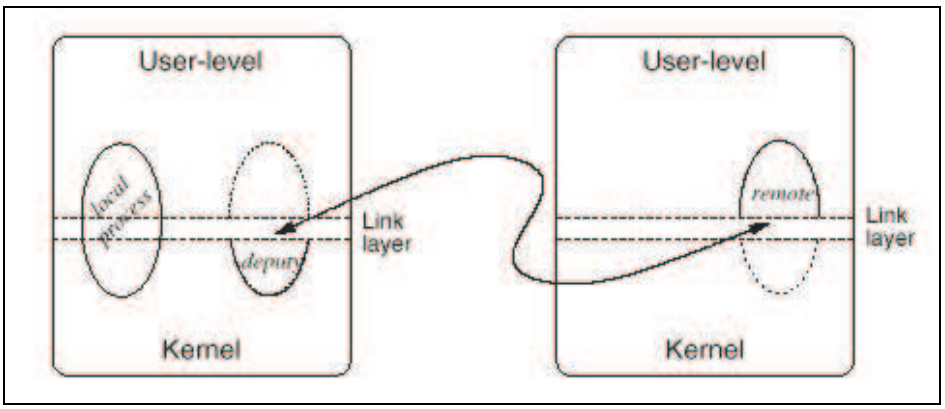


Figura 2-2 Meccanismo remote-deputy

Uno degli svantaggi dell'approccio remote-deputy è l'overhead dovuto all'esecuzione di system call via rete. Per questo motivo, ad esempio, i processi che creano dei socket sul proprio UHN o utilizzano shared memory non vengono fatti migrare, altrimenti si avrebbe un overhead eccessivo di chiamate di sistema dal remote verso il deputy.

Questo problema sarà comunque superato nelle versioni future di MOSIX, in quanto il team di sviluppatori sta implementando un supporto per la migrazione dei socket e per l'utilizzo di shared memory distribuita.

MOSIX non prevede un controllo di tipo centralizzato oppure delle relazioni master-slave tra i nodi; ciascun nodo può operare come un sistema autonomo e indipendente. La configurazione di una MOSIX farm è molto dinamica: ciascun nodo può entrare a far parte o uscire dalla farm senza creare particolari problemi alle altre workstation.

2.5.4 Algoritmi di condivisione delle risorse

I principali algoritmi di MOSIX per la condivisione delle risorse sono:

- l'algoritmo di load-balancing dinamico
- l'algoritmo di ottimizzazione dell'utilizzo della memoria.

2.5.4.1 Algoritmo di load-balancing dinamico

Questo algoritmo cerca continuamente di ridurre le differenze di carico tra coppie di nodi, migrando i processi dal nodo più carico verso quello più scarico. L'esecuzione è totalmente decentralizzata: ciascun nodo esegue lo stesso algoritmo e la riduzione di carico è effettuata in maniera indipendente per coppie di workstation. Il numero di processori di ciascun nodo e la loro velocità sono dei fattori presi in considerazione nelle scelte che vengono effettuate da questo algoritmo.

L'algoritmo di scheduling dei processi è basato su un modello matematico che riflette alcuni principi dell'economia e dell'analisi

competitiva. L'idea è quella di associare dei costi di utilizzo alle varie risorse, come CPU, memoria, dischi, etc. e di assegnare dinamicamente i processi ai nodi minimizzando continuamente questi costi.

2.5.4.2 Algoritmo di ottimizzazione dell'utilizzo della memoria (Memory ushering)

I processi vengono fatti migrare verso i nodi che hanno una maggiore memoria disponibile per prevenire il trashing dovuto a continue operazioni di swap su disco. A seguito di un memory swapping causato da poca memoria disponibile da parte di un nodo, questo algoritmo fa migrare il processo verso un nodo con memoria libera sufficiente.

L'algoritmo di memory ushering ha priorità rispetto a quello di load-balancing quindi i processi vengono migrati verso i nodi con più memoria disponibile anche a scapito del corretto bilanciamento del carico all'interno del cluster.

2.5.5 MFS - MOSIX File System

MOSIX è molto efficiente nell'eseguire in maniera distribuita processi CPU-bound.
Per gestire in maniera efficiente anche i casi di processi I/O-bound si possono utilizzare dei filesystem con supporto DFSA (Direct File System Access)

In un file system che supporta le specifiche DFSA le operazioni di I/O vengono effettuate da un processo in locale, sul nodo in cui è in esecuzione e non via rete, per evitare un overhead.
Inoltre un filesystem con supporto DFSA gode delle seguenti caratteristiche:

- Vi è uno stesso mount point su tutti i nodi

- File consistency. L'accesso ad uno stesso file può avvenire contemporaneamente dai vari nodi e le modifiche che il file subisce devono presentarsi nello stesso ordine in cui sono state realmente effettuate

- Time-stamp consistency. Se un file A è modificato dopo un file B allora A deve avere un time-stamp maggiore del time-stamp di B

Attualmente pochissimi file system supportano le specifiche DFSA; MOSIX è dotato di un proprio filesystem: MFS (MOSIX File System), conforme al tali specifiche. [17]
L'algoritmo di load-balancing di MOSIX consente ai processi di migrare verso i nodi su cui devono fare operazioni di I/O.
Con NFS i dati seguono le applicazioni mentre con MFS sono le applicazioni che migrano verso i dati.

MFS è un file system simile a "/proc" di Linux e fornisce ad ogni nodo una visione completa di tutti i file system montati fisicamente su tutti i nodi. MFS inoltre garantisce una consistenza della cache su uno stesso file nel caso di accessi concorrenti (la cache è mantenuta sul nodo in cui risiede fisicamente il file)

2.5.6 Monitoring dei processi, userland-tools e MOSIXview

MOSIX pubblica le informazioni relative allo stato del cluster (utilizzo di cpu, memoria, velocità dei processori...) all'interno del "/proc file system" standard di Linux.
Esistono delle utility , chiamate "userland-tools", che leggono il contenuto di "/proc/hpc/info" e consentono un monitoraggio in tempo reale dei processi che girano sul cluster. Le userland-tools consentono, tra l'altro, di effettuare il tuning dell'intero cluster e la migrazione manuale dei processi.

E' disponibile, inoltre, una interfaccia grafica (MOSIXview) molto user-friendly che fa da front-end per le userland-tools e fornisce uno strumento veloce ed intuitivo per il monitoraggio e la gestione dei processi, facilitando i compiti di gestione del cluster all'amministratore di sistema.

2.6 Test preliminari sulla farm VIRGO di Napoli

2.6.1 Introduzione

In questa sezione verranno descritti i test preliminari effettuati sulla farm, necessari ad avere delle indicazioni di base sulle prestazioni e sui tempi di elaborazione per l'analisi dei dati.

2.6.2 Test tecnici

L'obbiettivo di questi test è quello di valutare la capacità computazionale di ogni nodo della farm, sia in termini di potenza di calcolo che in termini di velocità di accesso ai dati su RAM, dischi e rete locale. A tale scopo sono stati utilizzati dei tool di benchmark disponibili su internet e comunemente usati da molti laboratori e centri di calcolo.

Nella Tabella 2-1 e nella Tabella 2-2 sono riportate le potenze di calcolo e le velocità di accesso al disco per ogni nodo della farm. I tool utilizzati per questi test sono rispettivamente glibench[13] 0.2.5 [7] e bonnie++[14] 1.02. [6]

I risultati dei test sulla potenza di calcolo dei nodi sono rilevanti per il dimensionamento delle farm usate per l'analisi on-line dei dati di VIRGO e per l'analisi off-line sulle binarie coalescenti. Forniscono, infatti, un limite inferiore sul numero di nodi necessari a sostenere il flusso di dati proveniente dall'esperimento.

La velocità di accesso ai dischi, in un certo senso, è un parametro meno critico in quanto sono possibili delle configurazioni in cui i dati distribuiti da un nodo master non vengono mai salvati sul disco dei nodi di calcolo ma piuttosto vengono mantenuti nella RAM dei nodi per tutto il tempo di elaborazione. Se i nodi sono dotati di una notevole quantità di memoria RAM una simile configurazione consente di eseguire un algoritmo alla massima velocità disponibile su un nodo. La velocità di accesso ai dischi è comunque un parametro importante da tenere in considerazione nello studio di nuove configurazioni di calcolo.

[13] http://glibench.sourceforge.net/
[14] http://www.coker.com.au/bonnie++/

CPU (glibench 0.2.5)	
Dhrystones[15] (MIPS)	**Whetstones**[16] (MFLOPS)
1867	**602**

Tabella 2-1 Potenza di calcolo per ogni nodo

DISK (bonnie++ 1.02)	
Block read (Kbyte/s)	Block write (Kbyte/s)
28968	32629

Tabella 2-2 Velocità di accesso ai dischi locali per ogni nodo

Nella Figura 2-3 e nella Figura 2-4 sono riportate le velocità di accesso alla memoria In lettura e scrittura, per ogni nodo. Nelle figure citate la "block size" è la quantità di memoria occupata da un blocco di dati, gli "strides" rappresentano la distanza (misurata in pagine di memoria) tra due blocchi di dati e la terza dimensione rappresenta la velocità di trasferimento. Il tool di benchmark usato per ottenere questi dati è memperf[17] 0.9e [8].

Dalle figure citate è possibile osservare che la velocità di trasferimento dei dati dipende fortemente dalla dimensione dei blocchi di dati trasferiti. I trasferimenti sono più efficienti quando la dimensione dei blocchi varia tra 16 e 128 Kbyte. In questo caso è possibile ottenere una velocità di trasferimento di circa 2000 Mbyte/s. Questo valore è molto importante se la memoria viene utilizzata per eseguire gli algoritmi alla loro massima velocità, come descritto in precedenza. E' inoltre importante notare che quando la dimensione dei blocchi supera la dimensione della memoria cache del processore, la velocità di trasferimento dei dati decresce rapidamente. Questo è un problema serio per la gestione di grandi flussi di dati, situazione che può

[15] Dhrystone è un test di benchmark che misura la velocità con cui un processore effettua operazioni su numeri interi. I risultati del test sono espressi in unità chiamate "dhrystones" (Milioni di operazioni intere al secondo).
[16] Whetstone è un test di benchmark che misura la velocità con cui un processore effettua operazioni in virgola mobile. I risultati del test sono espressi in unità chiamate "whetstones" (Milioni di operazioni in virgola mobile al secondo).
[17] http://www.cs.inf.ethz.ch/cops/software/

verificarsi ad esempio nel caso della ricerca di onde gravitazionali provenienti da una stella con una massa piccola.

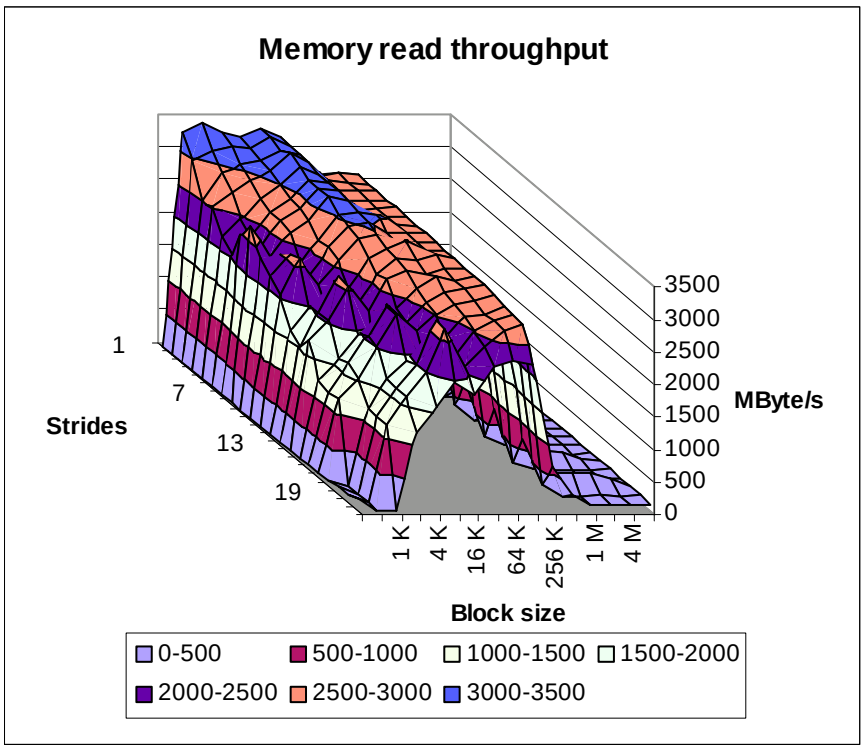


Figura 2-3 Velocità di accesso alla memoria in lettura per un singolo nodo

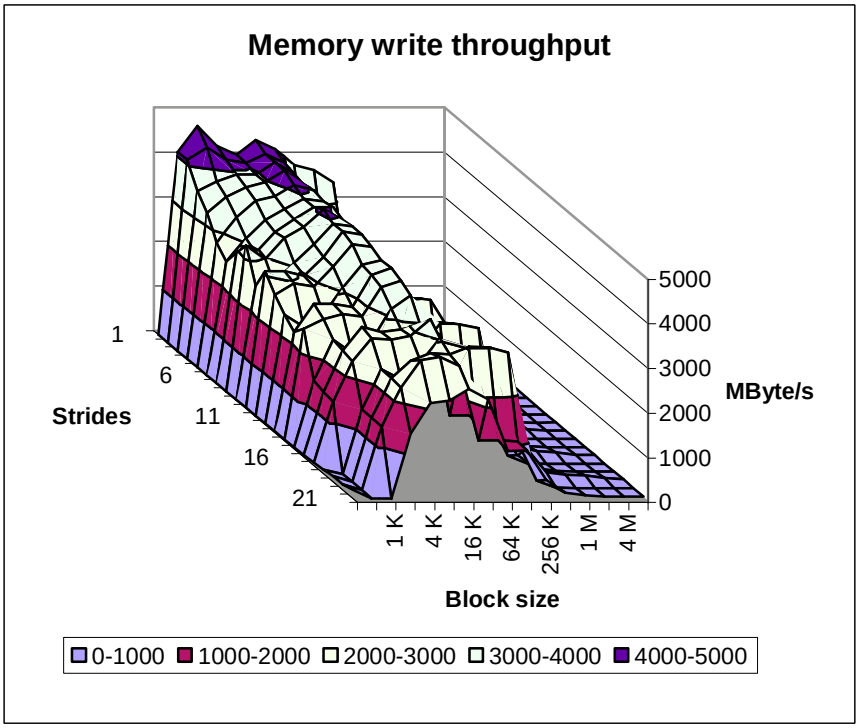


Figura 2-4 Velocità di accesso alla memoria in scrittura per un singolo nodo

Nel test successivo abbiamo considerato la velocità di trasferimento dei dati su LAN Fast Ethernet.
La Figura 2-5 mostra la velocità di trasferimento dei dati tra due nodi usando il protocollo TCP/IP. Come è possibile osservare dalla figura, la velocità massima di circa 90 Mbit/s è raggiunta quando la dimensione dei blocchi è maggiore o uguale a 1 Mbyte. Il tool di benchmark utilizzato per questo test è netpipe[18] 2.4 [9].

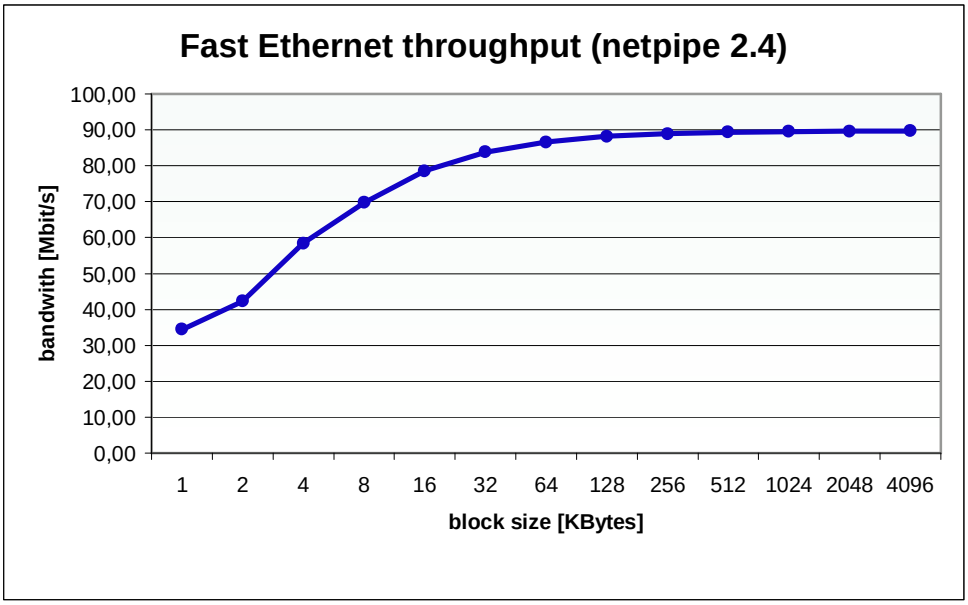


Figura 2-5 Velocità di trasferimento dati tra 2 nodi usando il protocollo TCP/IP

Nell'implementazione di algoritmi che usano MPI come protocollo di comunicazione tra i nodi è importante misurare la velocità di trasferimento su rete di blocchi di dati inviati utilizzando questo protocollo. La Figura 2-6 mostra i risultati di un test di comunicazione MPI ottenuti con il tool di benchmark pallas[19] 2.2 [10].

Dalla Figura 2-6 si può notare che la velocità raggiunta usando MPI è massima per valori intermedi della block size ma, per confronto con la Figura 2-5, è inferiore a quella ottenuta col protocollo

[18] http://www.scl.ameslab.gov/netpipe/
[19] http://www.pallas.com/e/products/pmb/index.htm

TCP/IP. Ciò è in accordo col fatto che i blocchi di dati trasferiti con MPI non devono essere di grandi dimensioni.

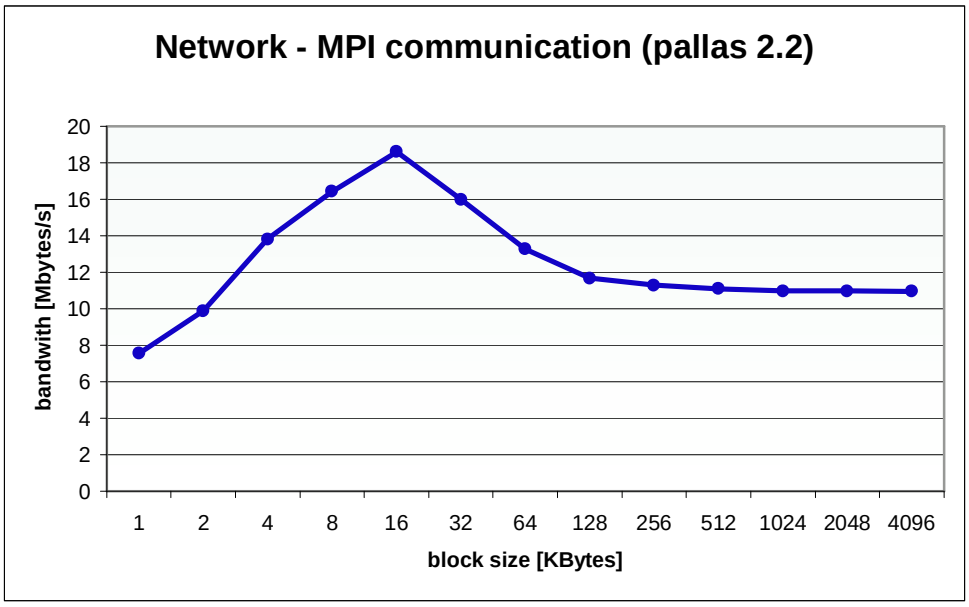


Figura 2-6 Velocità di trasferimento dati tra 2 nodi usando MPI

2.6.3 Test di analisi dei dati di VIRGO

Il prototipo della farm VIRGO di Napoli è stato usato per testare alcune procedure di analisi dei dati. Il primo passo è stato quello di valutare le prestazioni della farm con algoritmi di ricerca di binarie coalescenti. L'algoritmo usato è il "Matched Filter", una procedura per la ricerca di forme d'onda conosciute, all'interno di un segnale affetto da rumore di fondo; esso è ottimale ma richiede un elevato costo computazionale. Questo algoritmo, infatti, è basato su un confronto esaustivo tra il segnale di origine e tutte le possibili forme d'onda prese in considerazione, chiamate "templates".
Al crescere del numero di templates aumenta la qualità con cui viene identificato il segnale ma un numero sempre maggiore di operazioni sono richieste in fase di elaborazione.

L'implementazione dell'algoritmo Matched Filter usata per i test si basa su MPI per parallelizzare su più processori il carico computazionale.
I test di esecuzione dell'algoritmo Matched Filter su dati di prova hanno mostrato un progressivo aumento della velocità di elaborazione all'aumentare del numero di processori utilizzati. La figura 6 mostra la curva di speed-up osservata rilevata sperimentalmente:

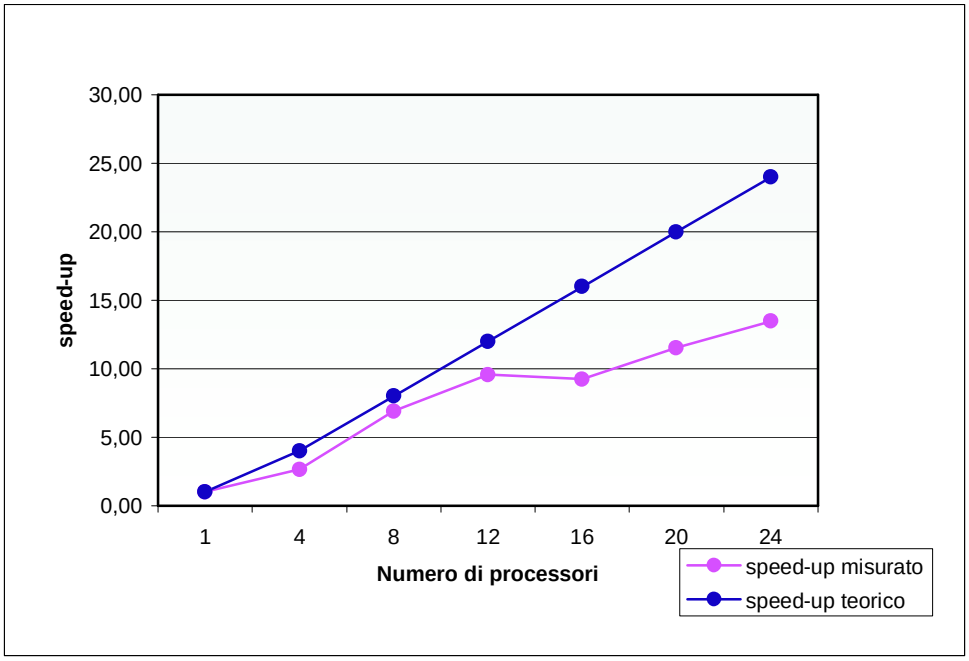


Figura 2-7 Speed-up dell'algoritmo Matched Filter parallelo

L'aumento della velocità non segue un andamento lineare rispetto al numero di processori utilizzati. Quest'effetto di saturazione è comunque presente in tutti i sistemi di calcolo paralleli basati su rete locale. [22]

2.6.4 Test generali di analisi dei dati

La farm è stata anche testata con procedure di calcolo parallelo utilizzate in altri campi della fisica. In particolare, è stato testato un programma (PWSCF[20] Plane-Wave Self-Consistent Field), basato su MPI, per l'analisi della struttura di un cluster di atomi. Una sintesi dei risultati ottenuti in questo caso è mostrata in Figura 2-8.

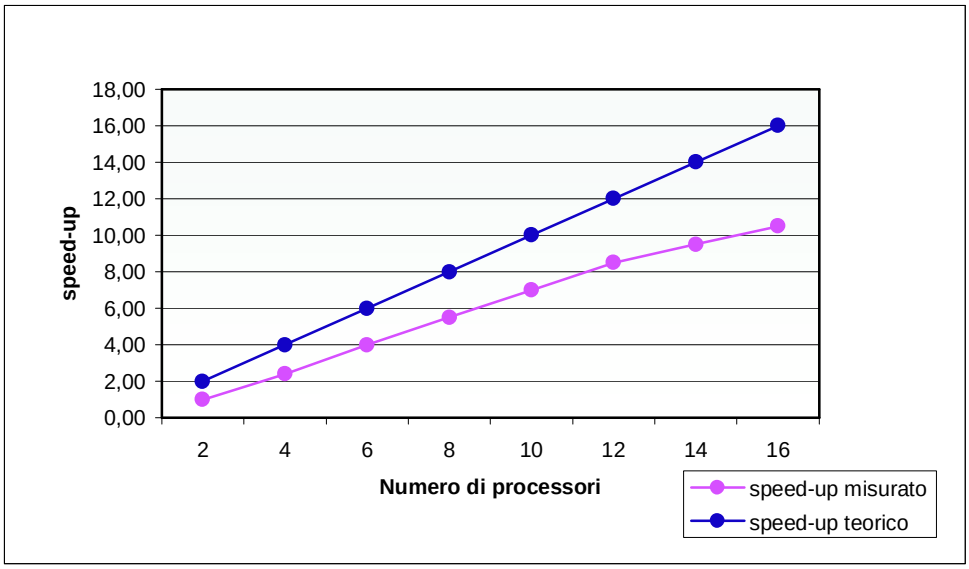


Figura 2-8 Speed-up del codice PWSCF

Dalla figura si può osservare che, in questo caso, la potenza di calcolo della farm cresce quasi linearmente all'aumentare del numero di processori. [22]

[20] http://www.pwscf.org/

3 GRID

3.1 Introduzione

Qualcuno potrebbe pensare che l'enorme potenza di calcolo generata da un moderno sistema di computer configurati in cluster potrebbe essere sufficiente a soddisfare i requisiti di ogni applicazione umanamente concepibile, ma non è sempre così.

Esistono già delle applicazioni (come ad esempio quelle di VIRGO) e altre se ne stanno pensando, che necessitano di risorse di calcolo superiori (a volte per ordini di grandezza) a quelle odiernamente fornite dagli attuali strumenti di calcolo.

Nasce quindi l'esigenza di andare oltre quella che è stata fino ad oggi la comune struttura di un sistema di calcolo.

Una delle strade recentemente intraprese per far fronte a tutto questo è l'implementazione delle cosiddette "griglie computazionali distribuite", più comunemente chiamate "grid".

3.2 Cos'è una grid

Le grid [24] sono infrastrutture che mirano a collegare su larga scala risorse di calcolo che possono essere eterogenee, distanti tra loro, nonché gestite da persone ed enti diversi.

Il termine griglia computazionale, "computational grid", è stato scelto perché in inglese presenta una forte analogia con il termine atto a designare la rete elettrica: "electric power grid"; è infatti al modello delle reti di distribuzione dell'energia elettrica che il progetto delle grid si ispira. Una rete elettrica rende oggigiorno ampiamente disponibili e collega fonti di energia eterogenee (vi sono centrali elettriche di innumerevoli tipi diversi, che sfruttano le più svariate tecnologie), distanti, gestite da persone ed enti diversi: basti pensare al fatto che intere nazioni comprano e vendono costantemente energia elettrica. Allo stesso modo si può pensare a dei prototipi di grid quando si guarda alle reti di comunicazione stradali, ferroviarie, marittime e aeree, alle reti telefoniche, ai servizi di trasporto e distribuzione delle merci e così via. Una cosa importante è però il

fatto che gli esempi citati sono realizzazioni diverse di una stessa struttura astratta.

Allo stesso modo una griglia computazionale dipende da altre infrastrutture, senza le quali sarebbe impensabile; in un certo senso si può addirittura pensare che ne sia la naturale evoluzione e il completamento: ci riferiamo alle strutture di calcolo classiche (personal computer, macchine multiprocessore, cluster di PC...) e alla rete Internet.

Le applicazioni che maggiormente necessitano della potenza di calcolo e della versatilità messa a disposizione da una infrastruttura di questo tipo e sono le seguenti:

- **Supercalcolo distribuito**. Questo tipo di applicazione può usare le grid per unire le risorse di calcolo di alcuni, o molti, supercomputer per risolvere problemi inaffrontabili con l'ausilio di un unico calcolatore, per quanto potente, tra quelli attualmente disponibili.

- **High Throughput computing**. In questo tipo di applicazioni, una grid può essere utilizzata per organizzare il lavoro di un grande numero di programmi, che siano tra loro poco o per nulla collegati. Lo scopo di questo sistema è generalmente quello di sfruttare il tempo macchina inutilizzato.

- **On demand computing**. Le applicazioni on demand (a richiesta) si appoggiano sulle grid per rendere disponibili, una tantum, risorse di calcolo che, per il loro saltuario utilizzo, non sarebbe pratico, né economicamente conveniente, avere a disposizione in locale.

- **Data intensive computing**. Nelle applicazioni di questo tipo, il cui scopo è la gestione di gigantesche quantità di dati distribuiti geograficamente, si riscontrano sovente problemi legati all'alto carico di calcolo e all'ingente trasferimento di dati. In questo caso il modello di griglia computazionale si presenta come una valida soluzione per lo scheduling e la gestione di flussi di dati complessi e di notevoli dimensioni.

- **Calcolo collaborativo**. Le applicazioni di questo tipo mirano soprattutto a favorire le comunicazioni e le collaborazioni tra le persone, pertanto sono spesso pensate in termini di spazi virtuali. Molte di queste applicazioni devono rendere disponibili risorse di calcolo condivise, così come archivi di dati.

3.3 Le grid nel mondo

Attualmente, in tutto il mondo, molti istituti di ricerca stanno implementando dei prototipi di grid, ciascuno rivolto alla soluzione dei particolari problemi che maggiormente interessano gli sviluppatori. Molto probabilmente quella in futuro che sarà la Grid nel suo stadio definitivo verrà proprio dall'unione di alcune di queste particolari implementazioni.

Negli Stati Uniti, ad esempio, la National Partnership for Advanced Computational Infrastructure (NPACI) e la National Computer Science Alliance (NCSA) stanno sviluppando il prototipo di una National Technology Grid[21], che mira a collegare le risorse di molti centri di supercalcolo, laboratori di ricerca, college e campus universitari. Il loro obiettivo principale e costruire la base di una grid su scala nazionale il cui utilizzo sia diffuso quanto quello di Internet oggigiorno.

Un altro progetto di questo tipo è in fase di implementazione alla NASA, si tratta della Information Power Grid (IPG[22]), essa mira a unire tutte le risorse di calcolo della NASA in un unico supercomputer da utilizzare per la risoluzione di problemi di calcolo nella ricerca aerospaziale e per quelli del monitoraggio delle condizioni climatiche.

3.3.1 EDG (European DataGrid)

In Europa 6 partner maggioritari e 15 minoritari hanno deciso di unire le proprie risorse umane e tecnologiche per lo sviluppo di una grid europea con lo scopo di risolvere i problemi di calcolo della fisica delle alte energie, il progetto è denominato EDG[23] (European DataGrid). [27]

Questo progetto nasce dall'esigenza di trattare l'enorme quantità di dati che verranno generati dagli esperimenti di fisica delle alte energie, in programma nei prossimi anni presso il CERN.

Presso i laboratori del CERN è infatti attualmente in costruzione un nuovo acceleratore di particelle, denominato LHC, presso il

[21] http://access.ncsa.uiuc.edu/
[22] http://www.ipg.nasa.gov/
[23] http://www.eu-datagrid.org/

quale saranno attivi quattro esperimenti denominati: ATLAS, CMS, ALICE, LHCb.

La grid che il progetto EDG intende realizzare per gestire ed analizzare i dati prodotti da questi esperimenti sarà costituita inizialmente da un insieme ridotto di risorse di calcolo geograficamente distribuite che fornisca i servizi e le funzionalità di base: il testbed.

Il testbed servirà a dimostrare la reale fattibilità del progetto, contribuendo a sviluppare le tecnologie non ancora disponibili e a risolvere i problemi, sia teorici che pratici, che si presenteranno durante questa fase mista di sviluppo e test. Oltre a fornire un ambiente dimostrativo, il testbed dovrà poter essere usato come strumento di produzione per qualche vera applicazione, usando dati reali, ad esempio nei processi di simulazione di alcuni esperimenti. La struttura di base fornita dal testbed, una volta consolidata, sarà ampliata fino a costituire la vera e propria grid di produzione.

3.3.2 Il progetto INFN-Grid

L'INFN, da sempre interessato alle problematiche del calcolo distribuito, è uno dei maggiori partecipanti nello sviluppo delle attività inerenti a LHC, in particolare di quelle legate alla realizzazione di infrastrutture per il Grid-computing. Il progetto INFN-Grid[24] [28] ha come obiettivo primario l'installazione e lo sviluppo, su scala nazionale, di una griglia computazionale in grado di gestire e utilizzare efficacemente sia i supercomputer che i cluster di personal computer distribuiti sui vari nodi della rete italiana per la ricerca: Garr-b. Queste risorse di calcolo, distribuite geograficamente e finora utilizzate normalmente dalle singole Sezioni INFN, dovranno poter essere integrate utilizzando la grid nazionale per formare un sistema coerente di High Throughput Computing, accessibile in modo trasparente ai ricercatori.

Questa grid nazionale dovrà inoltre essere integrata con quella del progetto europeo EDG e con infrastrutture simili, attualmente in fase di sviluppo in altre nazioni europee, in Giappone e negli USA.

[24] http://www.infn.it/grid

Come per il progetto europeo EDG, anche il progetto INFN-Grid ha come obiettivo primario la realizzazione di un testbed che dovrà convalidare il modello adottato e servire come piattaforma di test per lo sviluppo delle applicazioni.

3.3.3 Virgo & Grid

Oltre agli esperimenti legati a LHC, il progetto INFN-Grid è stato pensato anche per fornire a VIRGO l'infrastruttura necessaria a gestire ed analizzare i dati prodotti dall'interferometro di Cascina in un ambiente "grid-oriented" (che si avvale cioè di un'infrastruttura Grid).

Il testbed che il progetto INFN-Grid si propone di realizzare, verrà utilizzato come piattaforma di test su cui effettuare inizialmente l'analisi dei dati di VIRGO per la ricerca di segnali provenienti da sistemi di binarie coalescenti e da stelle Pulsar.

3.4 Organizzazione delle risorse della grid

La grid proposta da EDG e da INFN-GRID è organizzata sfruttando il cosiddetto modello a centri regionali, denominazione che sta a indicare un sistema di organizzazione gerarchico delle risorse di calcolo geograficamente distribuite.

I vantaggi di questa soluzione sono molteplici; innanzitutto una struttura distribuita su ampia scala geografica minimizza la necessità di spostamenti per il personale tecnico e per gli sperimentatori e massimizza, al contempo, la possibilità di coinvolgimento attivo nelle varie fasi dei lavori.

Un'organizzazione regionale è la soluzione migliore anche dal punto di vista della velocità di trasmissione dati: è un dato di fatto che le reti su corta distanza saranno sempre più economiche e efficienti rispetto a reti su lunga distanza. Pertanto una gerarchia di centri di calcolo, ciascuno con un adeguato spazio di archiviazione, che può servire sia da cache sia da repository dei dati per gli sperimentatori locali, permette di ottimizzare il rapporto costo/prestazioni della rete, un po' come avviene per il sistema di proxy gerarchico per l'http. Inoltre è spesso oggettivamente difficile, per motivi logistici o di reperibilità del know-how, concentrare in un unico luogo tutte le risorse necessarie, mentre distribuirle geograficamente risulta molto più semplice.

Una ricaduta secondaria di questo modello organizzativo è la capacità di poter decidere di elaborare i dati dove sono, piuttosto che dove si disponga di una maggior potenza di CPU, oppure vicino allo sperimentatore, potendo così ottimizzare l'efficienza dell'elaborazione a seconda delle circostanze.

3.4.1 I Tier

Il modello gerarchico appena descritto può essere schematizzato in una serie di **tier**[25], centri regionali, con cinque livelli decrescenti di complessità e capacità operative:

Il **tier 0** è il livello più alto della catena gerarchica ed è costituito da tutta quella parte della struttura di calcolo e acquisizione dati che non può, per evidenti ragioni pratiche, essere replicata altrove. Un esempio di questo genere e l'archivio dei Raw Data prodotto nella fase di acquisizione di un esperimento.

I **tier 1** forniscono tutti i servizi tecnici e i set di dati sperimentali necessari per le procedure di analisi e simulazione. Al livello 1 della gerarchia di tier sono anche definite le politiche di accesso alle risorse. Il sistema di autenticazione assegna a seconda dei casi priorità diverse di accesso, discriminando gli utenti secondo vari parametri, come il gruppo di ricerca di appartenenza, la vicinanza geografica alle risorse o il tempo macchina richiesto.

I **tier 2** sono centri di calcolo più piccoli, rispetto a quelli del livello superiore, generalmente dipendenti da un tier 1. Un tier di questo livello è in grado di fornire la maggior parte dei servizi di analisi, anche se non allo stesso livello di completezza di un tier 1. Il bacino di utenza di questo tier è molto ristretto, principalmente composto da utenti locali. L'uso dei tier 2 da parte di utenza remota è visto come un evento eccezionale, ad esempio causato da un guasto in un altro tier 2 o 1.

I **tier 3** e **4** sono gli ultimi due livelli di questa scala gerarchica e possono essere considerati, in un certo senso, elementi satelliti di tier 1 o 2, costituiti prevalentemente da piccole risorse di calcolo o di storage temporaneo, ad esempio una piccola farm di istituto per quanto riguarda il livello 3, o i PC usati come client dai ricercatori per il livello 4. Questi livelli sono totalmente dipendenti dai livelli

[25] Dall'inglese tier: fila

superiori, sia per quanto riguarda la disponibilità dei dati, sia per il collegamento agli altri tier.

Per quanto riguarda il progetto INFN-Grid il tier 0 degli esperimenti legati a LHC è il CERN, dov'è situato l'acceleratore di particelle e dove verranno svolte le operazioni di acquisizione dei dati. VIRGO, invece, avrà come tier 0 l'interferometro di Cascina.

L'INFN ha deciso di implementare un unico tier 1 al CNAF[26] di Bologna, punto di accumulazione verso cui convergeranno i dati provenienti dal CERN e da Cascina.

In fase di sviluppo del testbed, i tier 2 di VIRGO saranno situati nelle Sezioni INFN di Napoli e Roma1, in cui saranno sviluppate delle procedure di analisi, rispettivamente, su sistemi di binarie coalescenti e su stelle Pulsar utilizzando delle farm locali.

I tier 3 di VIRGO, responsabili della rielaborazione dei dati e di specifiche procedure di analisi in astrofisica e astronomia, saranno situati nelle Sezioni di Firenze/Urbino e Perugia

[26] IL CNAF è un Centro Nazionale dell'INFN il cui scopo è la ricerca e lo sviluppo nel campo delle discipline informatiche applicate agli esperimenti di fisica nucleare e delle alte energie.

3.4.2 Il tool-kit globus

Una grid è una struttura eterogenea, tenuta insieme da un certo numero di applicazioni, ciascuna delle quali si occupa di un particolare aspetto, avendo cura di gestire i necessari scambi di informazioni con le altre applicazioni. Per descrivere l'infrastruttura di una grid è quindi opportuno concentrarsi, volta per volta, sull'analisi di un particolare aspetto del problema (e quindi su una particolare applicazione), piuttosto che tentare di descrivere l'architettura nel suo complesso, cosa che risulterebbe indubbiamente dispersiva. Il modello a centri regionali, precedentemente presentato, fornisce una descrizione logica della composizione di una grid su larga scala; non permette però di analizzare la struttura da un punto di vista più tecnico e dettagliato.

Nel seguito, invece, saranno descritti i protocolli principali e i sottosistemi software su cui si basano le funzionalità di una grid. Per l'implementazione del prototipo iniziale di grid all'interno del progetto europeo EDG e di INFN-Grid è stato scelto di utilizzare il tool-kit **globus**[27]. [23]

Il software globus è definito "tool-kit" (collezione di utensili) in quanto è costituito da numerose piccole parti interagenti tra loro, ma utilizzabili anche singolarmente. Proprio la natura granulare di questo software è risultata la principale motivazione della sua scelta: questo tipo di struttura permette infatti di dover riscrivere solo porzioni molto limitate del codice, qualora sia necessario modificarne il comportamento. Globus, inoltre, si può facilmente estendere con delle componenti personalizzate, in modo da creare una propria "versione" del tool-kit, senza dover apportare sostanziali modifiche alla distribuzione originaria.

Il tool-kit, fornisce le funzioni di security, di accesso ai dispositivi di memorizzazione, di allocazione delle risorse e di gestione delle informazioni relative allo stato della grid. Il trasporto dei dati, oltre che agli strumenti interni di globus, è stato affidato anche a un altro software, GridFTP, che è una versione estesa del protocollo FTP

[27] Il progetto globus (http://www.globus.org) è sviluppato alla Mathematics and Computer Science Division dei Laboratori Nazionali di Argonne, all'Information Science Institute dell'Università della Southern California's e ai Distributed Systems Laboratory dell'Università di Chicago.

capace di interfacciarsi con il sistema di security della grid e di risolvere alcuni problemi, legati alla scarsa efficienza del protocollo FTP standard.

Globus serve essenzialmente come collante dell'intera grid e come strato software intermedio (middleware) su cui si appoggiano le applicazioni e le interfacce utente.
Esamineremo in dettaglio, nei prossimi paragrafi, i sottosistemi che compongono il tool-kit globus.

3.4.2.1 GSI (Grid Security Infrastructure)

La Grid Security Infrastructure [25] è quella parte della grid su cui si basano le procedure di autenticazione degli utenti e delle risorse. Essa è pertanto la struttura fondamentale su cui si appoggiano tutte le altre. Ogni volta che due entità, siano esse persone o risorse di calcolo, vogliono comunicare utilizzando la grid come infrastruttura, dovranno mutuamente autenticarsi, per essere sicure ciascuna dell'identità dell'altra. Questo è necessario per garantire che solo il personale autorizzato possa utilizzare le risorse di calcolo condivise.

Ogni singola comunicazione all'interno della grid deve essere autenticata, cosa che impone un notevole sforzo nello stabilire una policy di sicurezza adeguata. Non è invece strettamente necessario che le comunicazioni siano sempre crittografate, cosa che comporterebbe un notevole aumento delle dimensioni dei dati, basti pensare ai dati grezzi degli esperimenti; è evidente che non vi e nessuna necessità di crittografarli.

Il sistema di autenticazione di globus segue le direttive ratificate nello standard denominato X.509 della International Standards Organization (ISO). In questo standard sono definite le specifiche necessarie per un'autenticazione basata sul sistema dei certificati. I certificati, anche detti Digital ID, sono l'equivalente elettronico di un passaporto, in pratica servono a provare che qualcuno sia effettivamente chi sostiene di essere; inoltre essi permettono di effettuare comunicazioni riservate tramite l'utilizzo della tecnica di crittazione in chiave pubblica. Grazie allo standard X.509, i certificati sono un sistema di autenticazione indipendente dal software utilizzato, cosa che ne ha favorito una larga diffusione.
Il sistema di certificazione X.509 viene utilizzato per due scopi distinti:

- garantire la sicurezza della comunicazione, impedire cioè che qualcuno possa decifrare o alterare i dati trasferiti tra le due parti;

- verificare l'identità dei corrispondenti, ovvero, garantire all'utente di una transazione elettronica che l'entità all'altro

capo della comunicazione sia la persona giusta e non qualcun'altro, magari con intenzioni non proprio onorevoli.

Nel sistema di certificazione X.509 l'implementazione di queste due funzionalità avviene tramite l'utilizzo di una tecnica chiamata crittografia (asimmetrica) in chiave pubblica. Questa prevede che ogni utente possegga due chiavi che servono per criptare o decriptare i messaggi. Una è chiamata chiave pubblica e viene distribuita a chiunque voglia comunicare in modo sicuro con l'utente. La seconda chiave e detta chiave privata ed è custodita dall'utente e mantenuta segreta. Tipicamente la chiave privata è conservata sulla macchina personale dell'utente ed è criptata con una password. Questa chiave è necessaria tutte le volte che l'utente usa il suo certificato per una comunicazione sicura.

La chiave pubblica e quella privata sono parti complementari di una funzione matematica. Qualsiasi dato criptato con una delle due chiavi può essere decifrato soltanto dall'altra chiave. Una chiave può, inoltre, decriptare soltanto informazioni criptate dalla sua chiave complementare.

I certificati vengono rilasciati da enti appositi, chiamati Certification Authorities e comunemente abbreviati in CA, i quali godono della fiducia delle parti coinvolte nello scambio di informazioni. Una Certification Autorithy ha il compito di verificare, al momento del rilascio del certificato, la veridicità delle informazioni ivi contenute, cioè l'identità della persona per cui il certificato viene rilasciato. La CA riceve dall'utente, generalmente via e-mail, una copia della chiave pubblica, generata con un apposito software, e la usa per compilare il certificato, inserendovi le seguenti informazioni:

- una stringa (detta "subject" del certificato) che identifica la persona o la risorsa certificata;

- la chiave pubblica appartenente al proprietario;

- la data di scadenza del certificato;

- l'identità della Certificate Authority (CA) che ha firmato il certificato;

- la firma digitale della CA;

Una CA può emettere un certificato anche per un'altra CA, in questa maniera si possono creare delle catene gerarchiche di CA. Controllare la validità di un certificato significa quindi risalire la sua catena di autorizzazioni fino a raggiungere quella di una CA "fidata". In cima alla catena gerarchica c'è una "RootCA" che ha un certificato auto-firmato (root certificate).

Il GSI di globus si basa sul sistema di certificazione appena descritto e ne estende le funzionalità per soddisfare le esigenze derivanti dall'utilizzo in un sistema di calcolo distribuito. Queste infatti differiscono, sotto alcuni aspetti, da quelle nate con il trasferimento di informazioni riservate.

Nell'ambito di una grid, infatti, non si ha a che fare con la necessità di stabilire una comunicazione sicura tra un client e un server, ma tra un numero non definito a priori di entità distribuite su molti domini, amministrati in modo eterogeneo. Questo rende indispensabile l'uso di un'infrastruttura di autenticazione che si appoggi sui sistemi di sicurezza locali, senza la necessità di doverli sostituire.

Le problematiche da tenere in considerazione sono diverse:

- popolazione di utenti variegata e distribuita tra varie organizzazioni;

- risorse eterogenee inserite in un ambiente in evoluzione continua e pressoché imprevedibile;

- un processo di calcolo, durante la sua esecuzione, può avere la necessità di iniziare altri processi e allocare risorse in modo dinamico;

- le varie componenti di un processo di calcolo possono avere la necessità i comunicare tra loro;

- le risorse possono trovarsi in ambienti in cui le policy di sicurezza sono molto diverse le une dalle altre;

- sulle varie risorse lo stesso utente può trovarsi ad avere userID, variabili di ambiente e credenziali completamente diversi;

- risorse e utenti si trovano in nazioni diverse, con legislazioni diverse e a volte incompatibili tra loro (vedi il problema dell'esportazione di codici crittografici dagli USA);

La struttura stessa della grid e le problematiche legate al suo sviluppo hanno poi posto le seguenti condizioni al contorno:

- "one time login": Un utente dovrebbe potersi autenticare un'unica volta (ad esempio all'inizio della giornata o al momento di lanciare il calcolo) e poter lanciare programmi che acquisiscono risorse in tutta la grid senza doversi autenticare nuovamente;

- protezione delle credenziali: le credenziali personali (password, chiavi private, . . .) dell'utente devono rimanere al sicuro in ogni momento;

- esportabilità del codice: è necessario che il codice utilizzato non contrasti con le legislazioni dei paesi che collaboreranno al progetto;

- sistema di credenziali/certificazione uniforme: è necessario utilizzare uno standard comune, almeno per il sistema di mutua identificazione tra domini diversi;

Il GSI offre un sistema uniforme per la gestione delle credenziali adottando lo standard ISO X.509. Per l'esportabilità del codice il problema è risolto utilizzando il software di crittografia SSLeay[28], sviluppato al di fuori degli Stati Uniti.

[28] http://www.openssl.org/

Il problema del “one time login” e della protezione delle credenziali viene affrontato nel GSI con l’introduzione del meccanismo dei certificati proxy [29]. Per usare le risorse della GRID è necessario avere un certificato personale rilasciato dalla propria CA.

Un certificato proxy non è altro che un nuovo certificato (con una nuova chiave pubblica) e una nuova chiave privata. Il nuovo certificato è firmato dall’utente (e non dalla CA). Il certificato proxy ha una validità temporale limitata (di default 12 ore).

Il certificato proxy autofirmato, insieme al vero certificato dell’utente (contenente la chiave pubblica) viene mandato al soggetto che possiede la risorsa. Quest’ultimo può verificare la veridicità delle informazioni associate all’utente richiedente, ripercorrendo a ritroso la catena gerarchica di firme dei certificati. La chiave pubblica del richiedente (estratta dal certificato inviato) viene usata per validare il certificato proxy. La chiave pubblica della CA viene usata per validare il certificato dell’utente.

Il certificato proxy viene memorizzato nell’area dell’utente e può essere usato (senza ulteriori procedure di autenticazione) per tutta la durata del certificato stesso. L’amministratore di un host della GRID (gatekeeper) per consentire agli utenti di usare le risorse della macchina, deve abilitarne l’accesso. Per fare ciò è necessario conoscere il subject del certificato dell’utente remoto che si vuole abilitare, e mapparlo su un utente locale dell’host in modo che l’utente remoto possa accedere alle risorse con i permessi dell’utente locale.

Questa operazione viene effettuata aggiungendo una entry in un file di configurazione dell’host, detto “grid-mapfile”, in cui viene specificata la corrispondenza tra il subject del certificato dell’utente remoto e lo username locale.

3.4.2.2 GASS (Globus Access to Secondary Storage)

Il Globus Access to Secondary Storage è il sistema principale utilizzato da globus per fornire, ad utenti e programmi, gli strumenti necessari per accedere ai dispositivi di memorizzazione di massa delle risorse. Esso fornisce essenzialmente due funzionalità. La prima consiste nel mettere a disposizione dell'utente un protocollo, denominato "x-gass", tramite il quale accedere ai file sulla macchina remota. Questo può avvenire tramite l'utilizzo di appositi programmi forniti nel tool-kit, oppure, all'interno del software creato dall'utente tramite apposite chiamate di libreria, che sostituiscono le chiamate classiche di lettura e scrittura su file. Questo è analogo a quanto e offerto da altri protocolli basati su tecnologia client/server come, ad esempio, http o ftp. La seconda funzionalità fornisce al sistema la possibilità di creare e gestire in modo trasparente, sia per l'utente che per l'applicazione, un sistema di cache remota dei dati. In questo modo, un'applicazione lanciata dall'utente tramite la grid registrerà temporaneamente sia i dati in input che il proprio output sul disco della macchina che esegue l'elaborazione. In questo modo il processo non sarà appesantito durante l'elaborazione dal problema di trasportare i dati attraverso la rete. Al momento della sottomissione del job, l'utente può comunicare (tramite il linguaggio RSL[29]) al sistema le proprie preferenze in merito alla gestione della cache, specificando se l'output debba essere restituito durante l'esecuzione del programma, nel qual caso si parla di modalità interattiva, oppure se debba essere conservato nella cache fino a una successiva richiesta da parte dell'utente. In quest'ultimo caso si parla di modalità batch.
Questo sistema di cache fornisce i seguenti vantaggi:

- velocizza le operazioni di I/O;
- rende trasparenti le operazioni di gestione dei file remoti;
- permette l'implementazione di un efficace sistema di trasporto degli eseguibili;
- rende uniforme la gestione dei processi batch o interattivi;
- permette di controllare gli eventi disastrosi.

[29] RSL (Resource Specification Language): linguaggio strutturato utilizzato per descrivere i job da sottomettere al manager delle risorse della grid

Naturalmente vi sono degli effetti collaterali indesiderati, più precisamente si ha che questo sistema:

- aggiunge un overhead nelle operazioni di I/O;
- comporta la necessità di fornire sufficiente spazio disco sulle macchine che eseguono le elaborazioni.

Il primo è dovuto al fatto che, sostanzialmente, i file vengono scritti un numero maggiore di volte rispetto a quelle strettamente indispensabili in un sistema senza cache. Questo non è un controsenso rispetto a quanto detto in precedenza perché, se è vero che un sistema di questo tipo rende più lenta l'esecuzione di un programma rispetto a quello che si avrebbe in un'esecuzione locale, è anche vero che i benefici che si ottengono tramite l'elevata distribuzione del calcolo superano di gran lunga l'entità dell'overhead accumulato.

3.4.2.3 GRAM (Globus Resource Allocation Manager)

Il Globus Resource Allocation Manager è il livello più basso dell'architettura preposta alla gestione delle risorse, esso rende disponibile la possibilità di eseguire job in remoto, fornendo un'API[30] per la sottomissione, il monitoraggio e la conclusione dei job.

Quando un job viene sottomesso la richiesta viene inoltrata al "gatekeeper", situato sul computer remoto. Il gatekeeper è una parte di GRAM che risiede sulla risorsa di calcolo (tecnicamente un server). Esso interpreta la richiesta ricevuta dal cliente remoto e la passa ad un "jobmanager". Il jobmanager è un sottosistema software che esegue e tiene sotto controllo il programma remoto comunicandone i cambiamenti di stato all'utente che ha sottomesso il job. Quando l'applicazione remota finisce, sia normalmente, sia generando un errore, il jobmanager segue la stessa sorte.
Il GRAM si occupa inoltre di:

- interpretare le richieste in linguaggio RSL, che contengono la descrizione delle risorse necessarie all'esecuzione di un job. Questo compito è svolto creando uno o più processi per soddisfare la richiesta, oppure rifiutando il permesso di eseguire il job;

- abilitare la possibilità di controllare e monitorare da remoto lo stato dei job appena creati;

- aggiornare il Metacomputing Directory Service (un componente di globus, descritto nel prossimo paragrafo) con le informazioni relative alla disponibilità delle risorse che amministra;

Il GRAM tecnicamente è un sistema di tipo client/server. In generale le risorse della grid hanno un server GRAM, detto gatekeeper, che rimane in ascolto delle richieste su una porta TCP/IP

[30] API: Application Programming Interface, set di librerie che permettono di includere particolari funzionalità in un programma

(normalmente la 2119). Tutte le macchine, che in qualche modo debbono poter effettuare delle richieste di sottomissione dei job, dispongono invece del client, che si occupa di comunicare le richieste al server utilizzando il linguaggio RSL.

3.4.2.4 MDS (Metacomputing Directory Service)

L'MDS, acronimo di Metadirectory Data Structure [26], anche detto GIS (Grid Information Service), è il servizio che si occupa di mantenere un database con informazioni sullo stato delle risorse della grid. Questo servizio si basa su un sistema di database di tipo LDAP[31].

Il servizio MDS ha un carattere prettamente gerarchico; in esso, infatti, i dati collezionati dalle singole risorse risalgono una catena piramidale di server LDAP fino a giungere al database di livello più alto. Lo schema di funzionamento è abbastanza semplice: ogni risorsa dispone di un software chiamato GRIS (Grid Resource Information Service), che si occupa di reperire le informazioni relative allo stato della risorsa e inviarle periodicamente a un server di livello superiore detto GIIS (Grid Index Information Service), ad intervalli di tempo stabiliti (ad esempio ogni 5 minuti). Un GIIS si comporta essenzialmente come una cache per le informazioni relative alle risorse, ma svolge anche il compito di organizzare un gruppo di GRIS in modo coerente, secondo gli standard del database in uso (LDAP). Normalmente non esiste un unico GIIS che riceve le informazioni da tutti i GRIS esistenti; è infatti notevolmente più efficiente una struttura in cui esista una gerarchia di GIIS di livello diverso, dove ognuno di questi provveda periodicamente a inviare le informazioni raccolte a un server di livello superiore. Ad esempio, nell'ambito del progetto INFN-Grid si è deciso di avere un GIIS per ogni sezione dell'INFN, che raccoglie le informazioni locali; ne esiste poi uno di livello superiore, sito a Bologna, che colleziona le informazioni inviategli dai GIIS di sezione. In futuro probabilmente vi saranno anche dei GIIS intermedi che collezioneranno le informazioni relative ai GRIS delle risorse appartenenti a un singolo esperimento.

[31] Lightweight Directory Application Protocol, protocollo standard per la consultazione di grossi database distribuiti.

3.4.2.5 GridFTP

Il tool-kit globus comprende un potente strumento per il trasferimento dei dati all'interno di una griglia, il protocollo GridFTP (detto anche GSIFTP).
GridFTP è una versione estesa del protocollo FTP standard ed è dotato delle seguenti caratteristiche:

- Integrazione nativa con la Globus Security Infrastructure. Ciò consente di sfruttare in maniera "trasparente" i meccanismi di autenticazione basati su certificati X.509 durante il trasferimento di dati da un nodo all'altro della griglia;

- Supporto di canali multipli per trasferimenti paralleli. A differenza dell'FTP standard, GridFTP può effettuare un trasferimento di file tra due host utilizzando più connessioni (socket TCP/IP) in parallelo, garantendo un utilizzo efficiente di tutta la banda disponibile;

- Trasferimenti parziali di file. Questa caratteristica è utile quando si ha a che fare con file di grandi dimensioni e si vuole accedere soltanto ad alcune informazioni contenute al loro interno, senza dover necessariamente trasferire l'intero set di dati.

- Trasferimenti di tipo "Third-party" (server-to-server). Utilizzando questa modalità, un utente può eseguire un trasferimento di dati tra due host remoti, senza che le informazioni passino dall'host su cui è collegato, con una conseguente diminuzione del traffico di rete ed un notevole incremento delle prestazioni;

- Possibilità di recupero di trasferimenti interrotti. Questa è una caratteristica molto importante che consente, a seguito di errori in fase di trasmissione dati, di non dover ritrasmettere un intero file ma soltanto quella parte che non è stata ancora trasmessa.

3.4.3 Architettura e componenti del middleware di DataGrid

Una Grid può essere descritta in vari modi. Da un punto di vista funzionale può essere rappresentata come un insieme di servizi disponibili per le comunità di utenti. Un servizio è un programma che è continuamente in esecuzione in attesa di ricevere richieste dai client. I client fanno le loro richieste attraverso una connessione internet standard. Ci sono vari tipi di servizio; ognuno può essere composto da vari sottosistemi.

Il progetto European DataGrid propone un modello [30], adottato anche da INFN-Grid, costituito da un insieme di "grid-elements" [31], computer connessi a una rete che forniscono alcuni servizi grid.

3.4.3.1 Gli elementi di partenza

In questo modello, i "mattoni" di base che costituiscono una grid sono:

- **Computing Element (CE)**: è una risorsa grid in grado di fornire cicli di CPU per l'esecuzione di job. Un CE può anche essere il gateway di un cluster di PC, un supercomputer per l'esecuzione di job paralleli, o una postazione standard di calcolo interattivo in grado di gestire applicazioni grafiche e I/O verso dispositivi di storage.

- **Storage Element (SE)**: è un nodo di una grid che fornisce le facilities di accesso ai dati. Fornisce i servizi necessari a immagazzinare, localizzare e replicare i dati. Un SE, inoltre, fornisce ad altri nodi della grid le informazioni relative alla disponibilità dei dati.

- **Network**: la rete, in questo modello, è una risorsa fondamentale ed è intesa come un insieme di collegamenti, dotati di certe caratteristiche (tipologia, capacità di banda, traffico medio…), che interconnettono i vari grid-elements.

Questa prima immagine del modello di DataGrid è schematizzata in Figura 3-1:

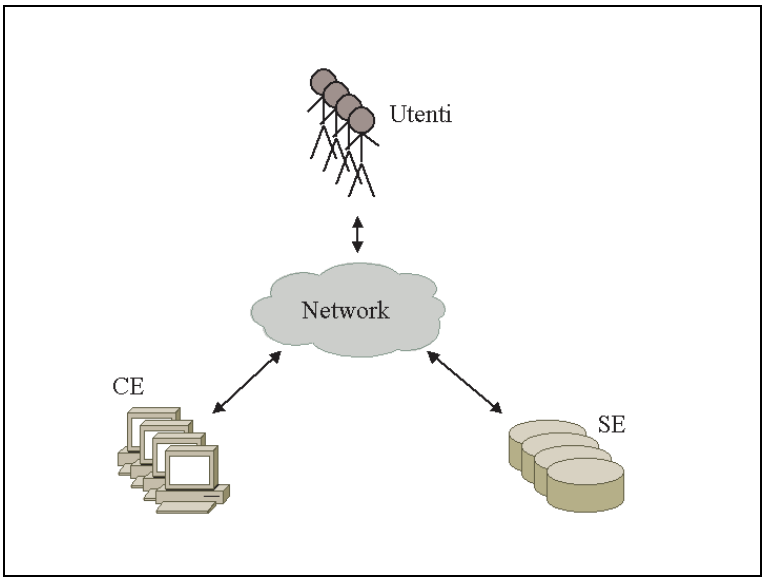


Figura 3-1 Schema di base di DataGrid

3.4.3.2 User Interface e Worker Node

Ai grid-elements di base si aggiungono altre risorse per ottenere uno schema di grid più esteso:

- **User Interface (UI)**: è un nodo di una grid a cui gli utenti si collegano per sottomettere i propri job. Una UI offre agli utenti un set di comandi e un ambiente testuale, grafico o di tipo web per la sottomissione e il management dei job;

- **Worker Node (WN)**: è un nodo generico che offre potenza di calcolo. Un WN può essere considerato ad esempio come un elemento di una farm locale, o un computer che può offrire, come unico servizio, la sua cpu per l'esecuzione di job.

Il nuovo schema è rappresentato in Figura 3-2:

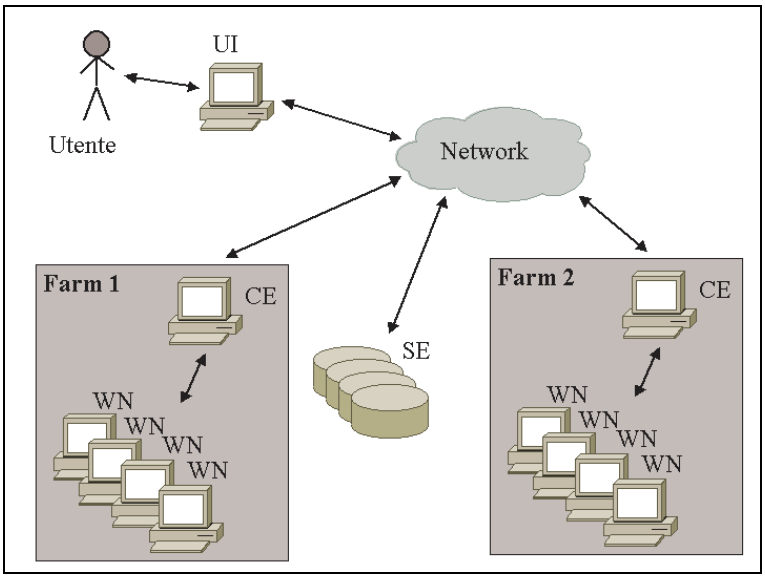


Figura 3-2 Schema esteso di DataGrid con WN e UI

L'assegnazione di un job al nodo di una farm viene effettuata da un sottosistema software detto **jobmanager**. Il più semplice jobmanager a cui si può pensare è quello di tipo "fork". Tale sottosistema è costituito dalle sole chiamate di sistema fork() che

creano un nuovo processo in corrispondenza di ogni job da eseguire sulla farm. Un jobmanager di tipo “fork” non gestisce il caso di una farm composta da più computer (CE + diversi WN) in quanto i processi generati dalle chiamate fork() rimangono confinati al CE. Per sfruttare la potenza di calcolo di tutti i nodi di una farm si possono utilizzare altri jobmanager. Quello che viene utilizzato di default è **PBS**[32] **(Portable Batch System)**. PBS si occupa dello scheduling e dell’allocazione (statica) dei processi in un cluster locale di macchine fornendo degli strumenti per la gestione di code, accounting delle risorse di calcolo ed elaborazioni di tipo batch.

[32] http://www.openpbs.org

3.4.3.3 Estensione della grid: requisiti

Una grid deve essere dotata di altri grid-elements in grado di garantire tre requisiti fondamentali:

1) **Indipendenza dalla locazione dell'ambiente di esecuzione**. Un utente non deve avere alcuna conoscenza, a priori, di quale CE (o WN) eseguirà il proprio job;

2) **Indipendenza dalla locazione dei dati**. Gli utenti devono essere in grado di accedere ai propri dati senza conoscere in quale SE essi sono realmente memorizzati.

3) **Sicurezza**. Gli utenti devono poter accedere ai propri dati in maniera sicura, nel rispetto delle policy stabilite all'interno delle loro organizzazioni virtuali.

Nei prossimi paragrafi saranno descritti in dettaglio questi tre aspetti, fornendo una visione della grid progressivamente estesa con nuove funzionalità e meccanismi di controllo.

3.4.3.4 Indipendenza dalla locazione dell'ambiente di esecuzione

L'indipendenza dalla locazione dell'ambiente di esecuzione viene realizzata introducendo un nuovo grid-element: il **Resource Broker (RB)**. L'RB è il perno su cui si basa il sottosistema di gestione del carico di lavoro (**Workload Management System**) all'interno della grid. Una delle attività principali di un RB è quella di trovare una corrispondenza tra i requisiti espressi dagli utenti per l'esecuzione dei propri job e le risorse disponibili sulla griglia, utilizzando opportuni algoritmi di scheduling. Una volta trovate queste corrispondenze, l'RB è in grado di decidere a quali CE devono essere assegnati i job sottomessi, attraverso il **Job Submission System (JSS)**.

La Figura 3-3 mostra un primo schema del Workload Management System (per semplicità sono stati omessi eventuali Worker Node):

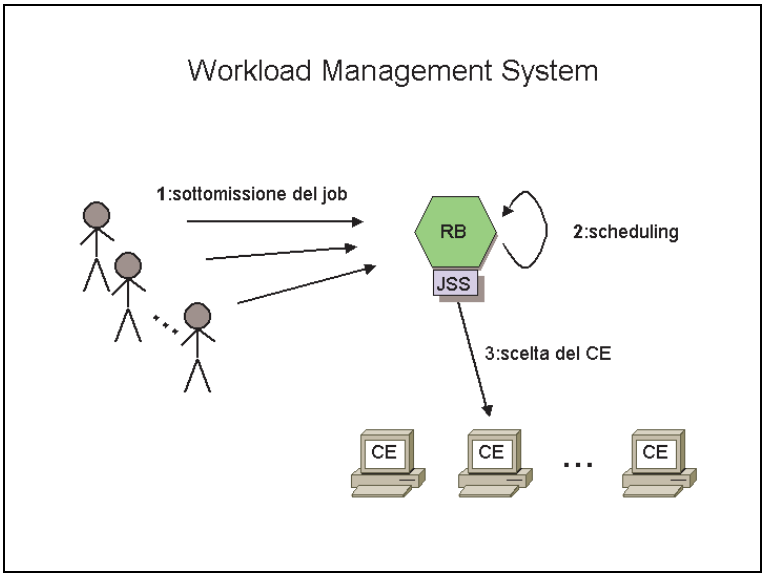


Figura 3-3 Schema di base del Workload Management System

I job sottomessi dagli utenti sono descritti utilizzando un linguaggio, detto **JDL (Job Definition Language)**. Il JDL adottato da DataGrid è il "Classified Advertisement (ClassAd) Language" definito dal progetto **Condor**[33] per la descrizione di job, workstation ed altre risorse. Il JDL è un semplice linguaggio basato su espressioni, con cui è possibile specificare le caratteristiche di un job (parametri di input/output, requisiti di sistema, jobmanager da utilizzare ed altri attributi) in maniera da facilitare la ricerca, da parte del Resource Broker, delle corrispondenze tra requisiti del job e disponibilità di risorse. Per ogni job sottomesso alla griglia, l'utente deve scrivere un file in JDL con tutte le caratteristiche e gli attributi del job; tra questi, di fondamentale importanza sono l'**Input Sandbox** e l'**Output Sandbox**. Questi due attributi indicano, rispettivamente, il set di file

[33] Lo scopo del progetto Condor (http://www.cs.wisc.edu/condor) è quello di implementare meccanismi e politiche di High Throughput Computing (HTC) su vasti insiemi di risorse di calcolo geograficamente distribuite. Condor-G, una versione del software di Condor specifica per gli ambienti di tipo Grid, è uno dei componenti su cui si basa il funzionamento dei Resource Broker.

che il job riceve in input (l'eseguibile del job e di tutti i suoi file di input) e il set di file che il job restituisce in output.

Le descrizioni dei job, scritte dagli utenti in JDL, vengono "tradotte" dall'RB in comandi RSL della sottostante architettura globus e passati al JSS. I criteri di scheduling adottati dal Resource Broker si basano su:

- autorizzazioni di accesso alle risorse da parte degli utenti;
- disponibilità dei dati;
- requisiti dei job;
- preferenze dei job;
- politiche di accounting;
- considerazioni di tipo euristico.

Affinché l'RB possa decidere quali risorse debbano essere allocate per l'esecuzione di un job, tutti i grid-elements devono essere in grado di fornire delle informazioni sul loro stato e pubblicarle, periodicamente, su un database ospitato da un altro grid-element, detto **Information Index (II)**. Il meccanismo utilizzato per questa operazione è quello offerto dal Metacomputing Directory Service (GIIS + GRIS) della struttura globus sottostante. La Figura 3-4 mostra il nuovo schema del Workload Management System, esteso con un Information Index:

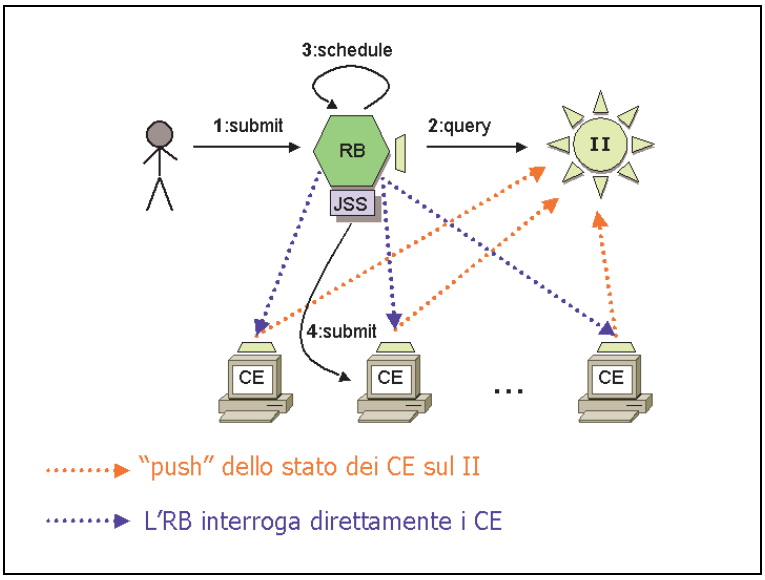


Figura 3-4 Workload Management System con Information Index

Tutti gli eventi legati alla sottomissione, l'esecuzione e il completamento di un job vengono registrati su un altro grid-element, detto **Logging & Bookkeeping (L&B)**. Il database con tutti questi eventi può essere consultato dall'utente attraverso la User Interface per avere informazioni relative allo stato dei job che ha sottomesso. Il nuovo schema è illustrato in Figura 3-5:

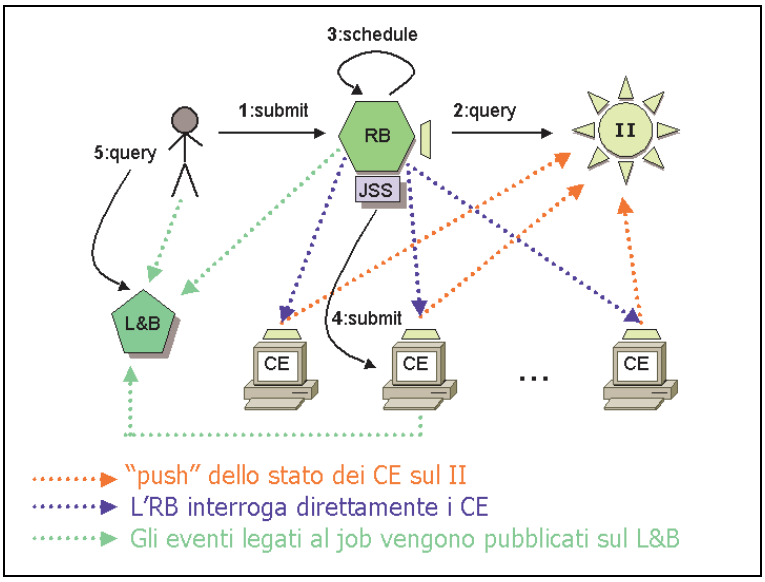


Figura 3-5 Workload Management System con Logging & Bookkeeping

3.4.3.5 Indipendenza dalla locazione dei dati

Attualmente l'intero sistema di accesso ai dati all'interno della griglia è basato sull'utilizzo di file. Tutti gli oggetti che interagiscono con un'applicazione vengono mappati su file. Anche lo standard input e lo standard output dei job vengono rediretti su file per consentire un'elaborazione di tipo batch. L'indipendenza dell'elaborazione di un job dalla locazione fisica dei dati è ottenuta grazie al cosiddetto meccanismo delle repliche. Uno stesso file (master), memorizzato su un certo Storage Element può esistere in copie multiple (repliche) su altri Storage Element distribuiti all'interno della griglia. Ogni file ha 2 nomi:

- un **Logical File Name (LFN)** che è un indirizzo del tipo:
  `lfn://<VO>/<path>`
  dove `<VO>` è la Virtual Organization in cui è memorizzato il file. Ad esempio l'LFN:
  `lfn://virgo.org/virgofile-1.dat`
  identifica il file di nome "`virgofile-1.dat`" all'interno della VO di nome "`virgo.org`".

- un **Physical File Name (PFN)** che ha la forma:
  `pfn://<SE>/<path>`
  dove `<SE>` è lo Storage Element su cui il file è fisicamente memorizzato. Ad esempio il PFN:
  `lfn://virgo-se.na.infn.it/test/file1.dat`
  identifica il file di nome "`file1.dat`" che si trova nella directory "`test`" di uno Storage Element che ha come indirizzo "`virgo-se.na.infn.it`".

Ad ogni LFN possono essere associati diversi PFN, uno per la copia master del file ed uno per ogni copia (replica) dello stesso file. Un nuovo grid-element, detto **Replica Catalogue (RC)** gestisce un database di tipo LDAP con tutte le associazioni tra LFN e PFN. Affinché il database rimanga aggiornato, gli Storage Element registrano, periodicamente, il contenuto del proprio file system sul Replica Catalogue. Per ogni file pubblicato sul database viene memorizzata una serie di attributi, tra cui:

- un flag che ne identifica il tipo (master oppure replica);
- il proprietario;
- i diritti di accesso (lettura, scrittura, esecuzione…);
- data dell'ultimo accesso;
- checksum e codici di controllo d'errore.

Le applicazioni possono accedere ad un file attraverso il suo LFN. Sarà poi compito del Resource Broker interrogare il Replica Catalogue per ottenere il PFN del file all'atto della sottomissione di un job. In questo modo, se uno stesso file è replicato in più punti della griglia, il Resource Broker potrà scegliere come PFN quello in cui compare lo Storage Element più "vicino" al Worker Node che dovrà eseguire il job. Il Worker Node prescelto potrà recuperare il file dallo Storage Element utilizzando il protocollo GridFTP. La Figura 3-6 mostra il nuovo schema della griglia, esteso con le funzionalità offerte dal Replica Catalogue:

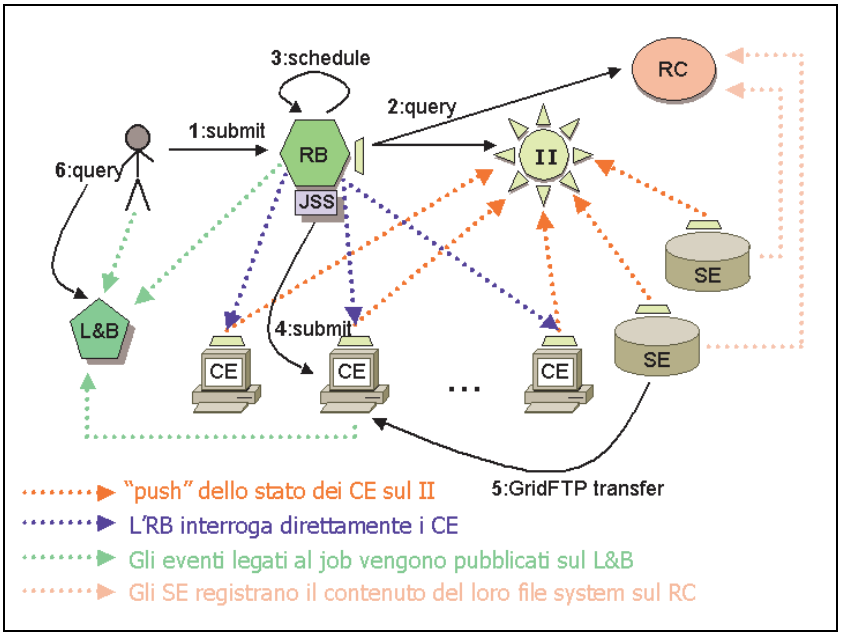


Figura 3-6 Meccanismo di accesso ai dati via Replica Catalogue

Per replicare i set di dati da uno Storage Element all'altro e affinché queste repliche siano registrate sul database del Replica

Catalogue in maniera automatica, si utilizza un tool chiamato **GDMP (Grid Data Mirroring Package)**. GDMP è un software con architettura client/server che viene installato sugli Storage Element. GDMP è in grado di replicare (mirroring) in maniera automatica ed asincrona dei file arbitrari (in qualsiasi formato essi siano stati scritti) all'interno della griglia. Il contenuto di un server GDMP può essere replicato su dei client GDMP in maniera sicura ed efficiente, utilizzando alcuni componenti del tool-kit globus. I trasferimenti dei dati tra due Storage Element replicati avvengono utilizzando il protocollo GridFTP e tutte le procedure di autenticazione remota si appoggiano sulla Globus Security Infrastructure.

Quando un nuovo file viene creato su uno Storage Element, la componente server di GDMP avvisa le componenti client residenti sugli altri Storage Element (fornendo loro il nome del nuovo file) e aggiorna il Replica Catalogue, inserendo una nuova entry con l'LFN e il PFN del nuovo file (copia master). I GDMP client, successivamente, provvedono autonomamente al trasferimento (via GridFTP) del file dallo Storage Element remoto e all'aggiornamento del Replica Catalogue, inserendo delle nuove entry con altri PFN per lo stesso file, uno per ogni replica. Lo schema esteso delle griglia che utilizza il Grid Data Management Package per la replica dei file è rappresentato in Figura 3-7:

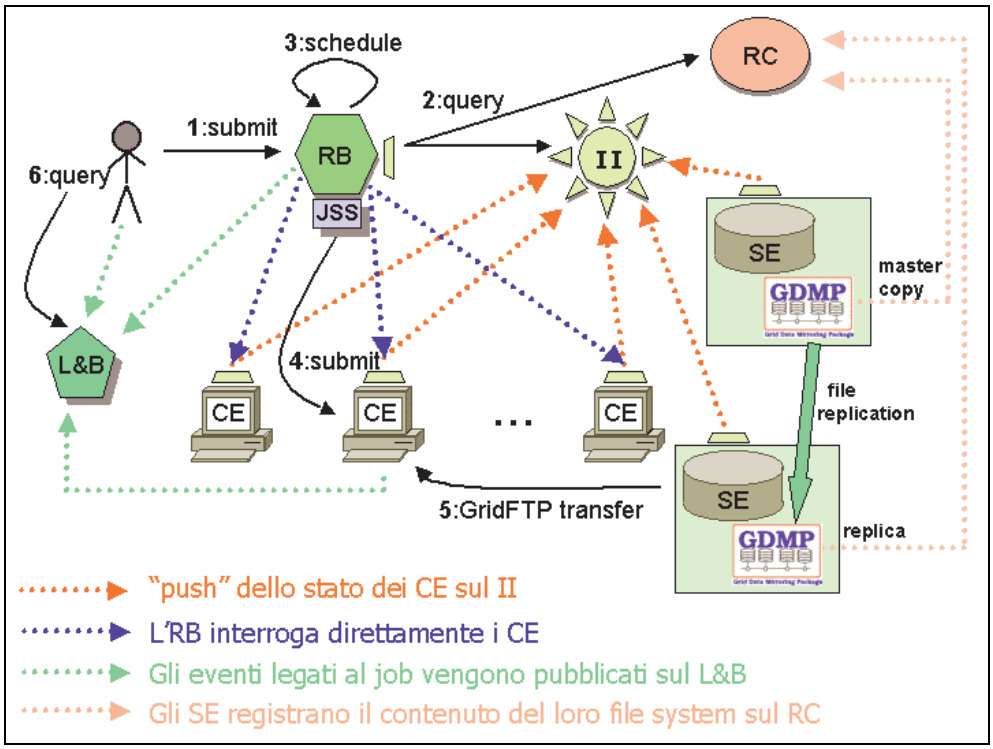


Figura 3-7 Replica dei file con GDMP

3.4.3.6 Sicurezza

I meccanismi di autenticazione e di accesso alle risorse, basati sui certificati X.509, sono quelli offerti dalla Security Infrastructure (GSI) della sottostante architettura globus, descritti nel paragrafo 3.4.2.1.

Gli utenti della grid, identificati univocamente con il loro certificato, fanno parte di Organizzazioni Virtuali, che corrispondono a gruppi di lavoro, enti di ricerca, aziende, società, etc. Uno stesso utente può essere membro di più Organizzazioni Virtuali e spesso, per motivi di lavoro, può essere costretto a spostarsi fisicamente da un sito all'altro della grid, pur rimanendo nella stessa Organizzazione Virtuale.

Per tenere traccia delle Organizzazioni Virtuali e degli utenti che ne fanno parte, nasce l'esigenza di avere un nuovo grid-element, detto **Virtual Organization Server (VO)**. Il VO gestisce un database LDAP in cui è memorizzato l'elenco delle Organizzazioni Virtuali e, per ognuna di esse, l'elenco dei membri che ne fanno parte. Per ogni utente il VO memorizza una serie di campi, tra cui nome, cognome, email, descrizione, chiave pubblica del certificato, etc. Il VO può essere usato come una sorta di "yellow pages" per recuperare informazioni specifiche per un certo utente o per una certa Organizzazione Virtuale. Le registrazioni sul database del VO vengono effettuate, di solito, attraverso un'interfaccia web.

Come descritto nel paragrafo 3.4.2.1 il meccanismo di gestione delle autorizzazioni per l'accesso alle risorse della grid è attualmente implementato nel testbed con un "grid-mapfile". Questo file è presente su ogni Computing Element e Storage Element e contiene dei mapping tra il subject dei certificati utente e le username locali delle macchine. Il grid-mapfile è un oggetto statico e se il database del VO viene aggiornato frequentemente i grid-mapfile dei vari CE ed SE possono diventare obsoleti in breve tempo. Per evitare che ciò accada i CE e gli SE eseguono periodicamente un perl script, detto **mkgridmap**. Questo script legge un file di configurazione in cui ci sono definite delle politiche locali (ACL access control list), interroga il VO e costruisce un grid-mapfile aggiornato con le corrispondenze tra i subject dei certificati degli utenti autorizzati e un elenco di username locali. Questo schema è illustrato nella Figura 3-8:

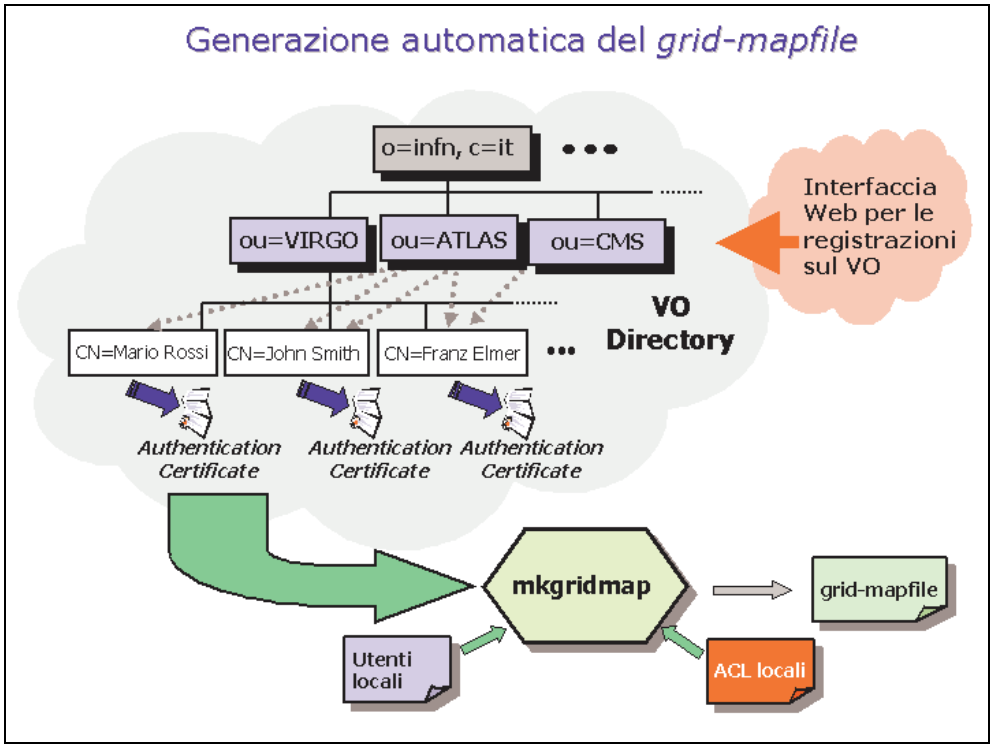


Figura 3-8 Generazione automatica del grid-mapfile

3.4.3.7 Installazione e management dei grid-elements

Il numero di grid-elements in ogni sito connesso alla grid può diventare estremamente elevato. In un ente di ricerca molto grande, ad esempio, il numero di grid-elements può essere dell'ordine delle migliaia. Le configurazioni dei grid-element evolvono rapidamente, in maniera molto dinamica. L'intero sistema software su cui si basa l'infrastruttura grid è in continuo e rapido sviluppo. Versioni aggiornate del tool-kit globus e di tutto il middleware di DataGrid vengono rilasciate molto frequentemente per correggere gli inevitabili errori di programmazione di componenti software così sofisticati. Le versioni del software che gira sui grid-element devono essere allineate per evitare problemi di incompatibilità e malfunzionamenti distribuiti. Può capitare, inoltre, che dei set di risorse devono essere allocati per un breve periodo ad un certo processo di elaborazione (ad esempio un calcolo che dura una settimana) e riallocati, successivamente, per un'altra elaborazione. Ciò comporta un notevole sforzo da parte dei system administrator, costretti a impiegare ore di lavoro in onerose operazioni di installazione e riconfigurazione delle macchine. Per far fronte a questi evidenti problemi di management dei grid-elements devono essere utilizzati degli strumenti software opportuni. Quello utilizzato nei progetti EDG e INFN-Grid è **LCFG**[34] (**L**ocal **C**on**Fi**g**G**uration System).

LCFG è un tool di installazione automatica, configurazione e management centralizzato di macchine con sistema operativo Linux. L'architettura client/server su cui si basa LCFG è illustrata in Figura 3-9:

[34] http://www.lcfg.org

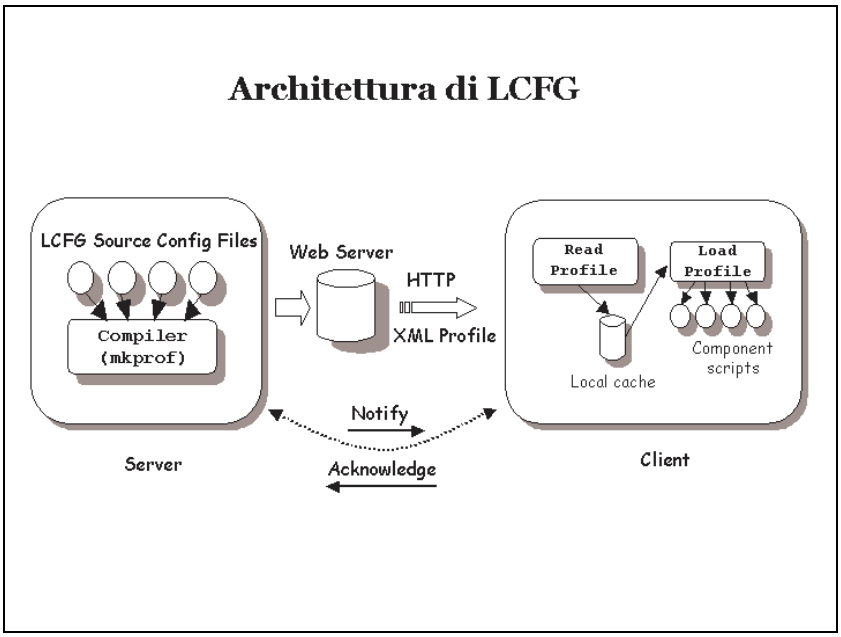


Figura 3-9 Architettura client/server di LCFG

La configurazione di un intero sito (inteso come insieme di computer) è descritta nei "source files", mantenuti su un server centrale. Ognuno di questi file non corrisponde (necessariamente) ad una macchina specifica ma descrive, piuttosto, un "aspetto" della configurazione complessiva. Un source file, ad esempio, può descrivere i parametri necessari alla configurazione di una macchina per gli studenti, quelli necessari all'installazione di una generica macchina Linux Redhat 6.2, quelli specifici per un Computing Element di una grid, etc.

I source files vengono "compilati" (con il tool mkprof) in profili individuali, specifici per le singole macchine. Ogni profilo contiene i parametri necessari alla configurazione di una macchina target. Quando un profilo (che risiede sul server) viene modificato, la macchina client associata a quel profilo riceve una notifica UDP. Il client recupera dal server, via HTTP, la versione aggiornata del suo profilo e lo memorizza in una cache locale. I client, interrogano periodicamente il server per verificare se vi siano ulteriori cambiamenti nel proprio profilo non ancora notificati.

I client, inoltre, inviano periodicamente al server degli "acknowledgement" UDP; il server usa questi messaggi per generare

dinamicamente una pagina web in cui sono visualizzate le informazioni relative allo stato dei client.

Sui client, i "component scripts" sono responsabili della lettura e del parsing del profilo ricevuto dal server, nonché dell'implementazione della configurazione descritta in tale profilo. Quest'ultima procedura viene svolta, in generale, rigenerando opportunamente i file di sistema della macchina client e notificando i demoni dei servizi coinvolti in tali modifiche.

Un profilo LCFG è, di fatto, un documento XML con una struttura semplice, contenente una sezione per ogni componente. La configurazione di ogni componente è descritta da una lista di coppie chiave/valore. Il profilo fornisce una esplicita dichiarazione dell'intera configurazione della macchina client.

Il meccanismo dei profili centralizzati consente di installare e configurare una macchina "da zero" in brevissimo tempo, con il minimo intervento da parte dell'amministratore di sistema. L'installazione di un grid-element (di cui è presente un profilo sull'LCFG server), ad esempio, si riduce a un reboot della macchina e all'avvio di una procedura automatica che:

- legge da un dispositivo di boot (floppy, cdrom o scheda di rete) una versione minimale del sistema operativo;
- partiziona il disco di sistema e installa il sistema operativo (la lista dei pacchetti di Linux da installare è descritta nel profilo ottenuto dal server LCFG);
- esegue degli script di post-installazione che completano la configurazione e l'aggiornamento del software.

Il sistema di installazione "non presidiata" offerto da LCFG si avvale di un server DHCP per l'assegnazione degli indirizzi IP alle macchine da installare e di un server NFS per l'esportazione ai client di una directory in cui sono presenti tutti i pacchetti di Linux.

Questo sistema risolve in maniera efficiente i problemi legati al management di un elevato numero di grid-elements all'interno di un sito.

3.4.4 Test di analisi dei dati di VIRGO con GRID

Nei paragrafi successivi sono descritti i passi seguiti per l'implementazione dell'infrastruttura grid iniziale, il layout geografico dei grid-elements utilizzati e il meccanismo di sottomissione dei job di analisi dei dati di VIRGO.

3.4.4.1 Realizzazione dell'infrastruttura iniziale

L'obiettivo di questo lavoro è quello di creare un pool iniziale di macchine da configurare e utilizzare come grid-elements. In questa prima fase di test è stato deciso di utilizzare soltanto un sottoinsieme dei nodi diskless della Linux Farm descritta nel Cap. 2. In particolare, dal set di 12 computer sono stati scelti 4 nodi diskless da convertire, rispettivamente, in 1 User Interface, 1 Computing Element e 2 Worker Node.

Il sistema operativo (Linux RedHat 6.2) e il middleware di DataGrid[35] (EDG 1.1.4), sono stati installati su una partizione del disco da 18Gbyte in modo da fornire ai 4 nodi una doppia configurazione e utilizzarli, a seconda delle esigenze, come grid-elements o come nodi diskless della Linux farm locale.

Oltre al pool di 4 macchine scelte tra i nodi della farm è stato utilizzato un altro computer come LCFG server per le installazioni e il management centralizzato dei grid-elements.

Il setup delle macchine ha richiesto i seguenti passi:

- Acquisizione del know-how. In questa fase sono state raccolte tutte le informazioni e la documentazione necessaria all'utilizzo del tool-kit globus e del middleware di DataGrid;

- Installazione e configurazione del server LCFG. Sulla macchina scelta per questo compito è stato installato il sistema operativo (da CDROM) e le componenti server di LCFG, tra cui: un server DHCP per l'assegnazione automatica degli indirizzi IP ai grid-elements, un server NFS per l'esportazione dei pacchetti Linux (in formato rpm) e un server web

[35] http://marianne.in2p3.fr/

(Apache 1.3.12) da cui i grid-element effettuano il download del proprio profilo.

- Creazione dei profili per i 4 grid-elements. Il team di sviluppo del middleware di DataGrid fornisce dei "modelli" di file di configurazione (source files) che descrivono dei generici grid-elements. Questi modelli sono stati personalizzati con tutti i parametri "site-dependent" della rete locale di Napoli e con tutti i parametri specifici per i 4 grid-elements (nomi, indirizzi IP, schema di partizionamento dei dischi, filesystem da montare, pacchetti software aggiuntivi...). Una volta personalizzati, i source files sono stati compilati con "mkprof", il tool di LCFG che genera i profili XML e resi disponibili ai client, via web.

- Creazione di un floppy per il boot iniziale. Uno dei componenti di LCFG consente di creare un dischetto di boot per l'installazione "non presidiata" dei grid-elements. Con questo tool sono stati creati i boot-floppy per i 4 grid-elements (1 UI, 1 CE e 2 WN). Una volta inseriti i floppy nel drive delle 4 macchine, le procedure di installazione automatica hanno generato, in breve tempo e senza alcun intervento manuale, dei sistemi pronti per essere utilizzati come grid-elements.

3.4.4.2 Layout geografico dei grid-elements

Il pool iniziale di 4 grid-elements descritto nel paragrafo precedente è soltanto uno dei "mattoni" di base che compongono l'infrastruttura utilizzata come testbed per l'esecuzione di job di analisi dei dati di VIRGO su grid.

Il layout geografico [32] dell'intero pool di grid-elements utilizzati nella VIRGO Virtual Organization è illustrato nella Figura 3-10:

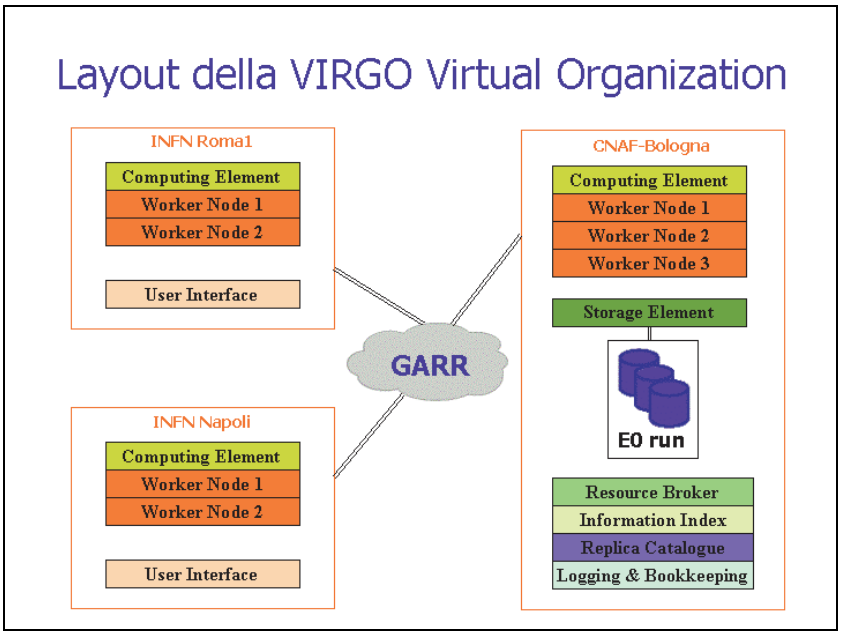


Figura 3-10 Layout geografico dell'infrastruttura grid usata da VIRGO

I grd-elements utilizzati sono disposti su tre siti INFN connessi alla rete GARR: Napoli, Roma e Bologna. I siti di Napoli e Roma hanno due configurazioni "gemelle" con una User Interface e una farm locale costituita da un Computing Element e due Worker Node. Al CNAF di Bologna, tier 1 per gli esperimenti di LHC e VIRGO, sono installati i grid-elements che offrono i servizi di base per il funzionamento di tutto il testbed INFN-Grid: Resource Broker, Information Index, Replica Catalogue, Logging & Bookkeeping. A Bologna, inoltre, è installata una farm locale costituita da un Computing Element e tre Worker Node, ed infine uno Storage Element con uno

spazio disco di 1Tbyte contenente un mirror dei dati acquisiti durante l'Engineering Run 0[36] di VIRGO (E0 run).

[36] L'interferometro di VIRGO è ancora in fase di realizzazione. I test periodici delle apparecchiature necessarie all'acquisizione dei dati sono chiamati "Engineering Run", durano in genere qualche giorno e hanno dei numeri progressivi (0,1,2...)

3.4.4.3 Sottomissione dei job di analisi dei dati

Questo paragrafo descrive il modo in cui sono stati effettuati i test di analisi di binarie coalescenti utilizzando l'infrastruttura grid di VIRGO, illustrata nei due paragrafi precedenti.

Nella situazione iniziale lo Storage Element di Bologna contiene i "Raw Data" (dati "grezzi") acquisiti durante il Run E0 di VIRGO. Questi dati sono organizzati in particolari strutture dette "frame" e memorizzati all'interno di una serie di file. I Raw Data non vengono pubblicati sul Replica Catalogue perché non devono essere mirrorati su altri Storage Element, né devono essere direttamente processati dalle farm. Il processo di analisi vero e proprio, infatti, deve essere eseguito su alcuni subset di questi dati grezzi, scelti in modo opportuno dai ricercatori di VIRGO (ad esempio dai dati grezzi devono essere estratti soltanto quelli relativi ad una certa banda di frequenze del segnale acquisito dall'interferometro). Questi subset di dati devono essere pubblicati sul Replica Catalogue per poter poi essere analizzati utilizzando le farm di Computing Element e Worker Node dei tre siti. L'analisi viene fatta applicando l'algoritmo Matched Filter ai dati estratti, con circa 40.000 template. Se tutti i template venissero usati da un solo processo Matched Filter ci sarebbe un solo job allocato su uno dei Worker Node e non si sfrutterebbero i vantaggi offerti dal calcolo distribuito all'interno della griglia. Per distribuire il carico computazionale tra tutti i Worker Node disponibili, l'insieme dei 40.000 template viene diviso in più sottoinsiemi, ognuno dei quali è utilizzato da un processo Matched Filter su un certo subset di dati estratti.
Quanto appena descritto può essere sintetizzato in 4 passi:

1) L'utente si collega ad una delle due UI ed effettua la procedura di autenticazione sulla grid. Nella home-directory dell'utente vi è una subdirectory di nome ".globus" che contiene il suo certificato e la sua chiave privata. L'utente esegue il comando "grid-proxy-init", digita la passphrase che protegge il suo certificato e ottiene l'autorizzazione di eseguire job sulla grid. Tale autorizzazione dura di default 12 ore, periodo dopo il quale il suo certificato proxy (creato con il comando grid-proxy-init) non è più valido.

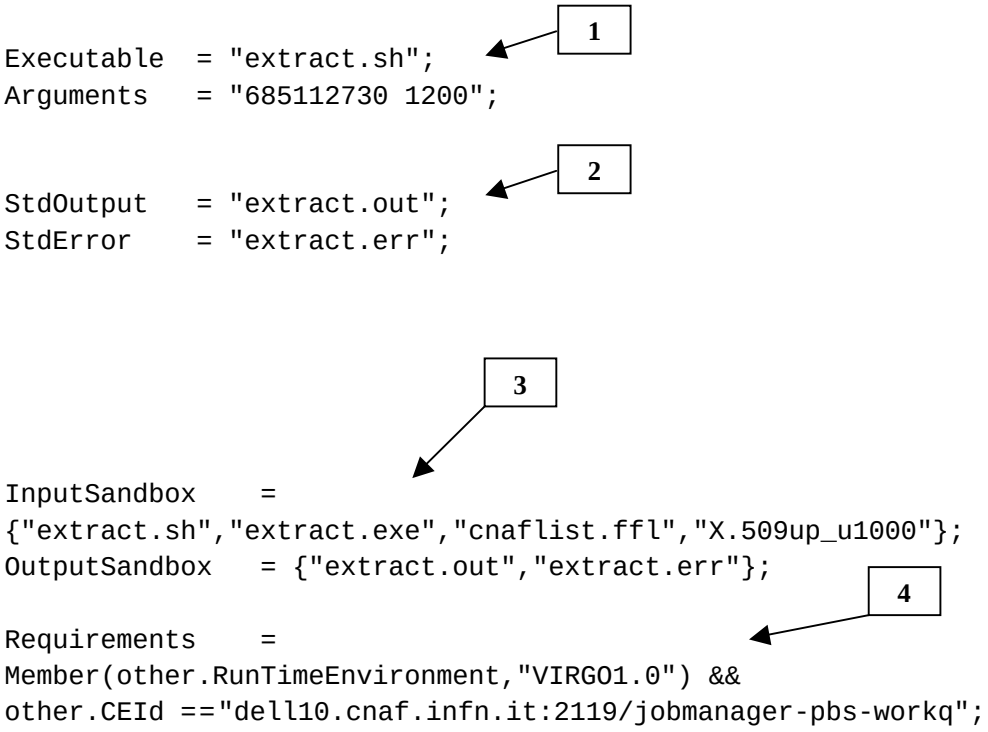
2) L'utente sottomette alla grid un job per l'estrazione di un subset di dati dallo Storage Element di Bologna. I dati estratti vengono pubblicati sul Replica Catalogne.

3) Per ogni sottoinsieme di template viene sottomesso alla grid un job per l'esecuzione di un processo Matched Filter sul subset di dati ottenuto al punto 2). Il Resource Broker provvede, in maniera autonoma, all'allocazione dei job sui Worker Node disponibili.

4) L'utente può in ogni momento interrogare il Logging & Bookkeeping, col comando "dg-job-status", per conoscere lo stato dei job che ha sottomesso. Una volta che tutti i job raggiungono lo stato "completed" l'utente può recuperare il risultato delle elaborazioni nella sua Output Sandbox, con il comando "dg-job-get-output", ed effettuare il "logout" dalla griglia con il comando "grid-proxy-destroy", distruggendo così il suo certificato proxy, creato al punto 1).

I passi 1) e 4) sono banali mentre il 2) e 3) richiedono ulteriori dettagli.

Generalmente, per l'estrazione di un subset di dati dai frame contenuti nei Raw Data, i ricercatori di VIRGO utilizzano un eseguibile di nome "extract.exe". Affinché questo programma possa essere eseguito sulla grid è necessario scrivere un file in JDL che ne descriva le caratteristiche. Questo file, che chiameremo ad esempio "extract.jdl", deve poi essere sottomesso alla grid con il comando "dg-job-submit". Esaminiamo il contenuto del file "extract.jdl":


1
2
3
4
Executable = "extract.sh";
Arguments = "685112730 1200";
StdOutput = "extract.out";
StdError = "extract.err";
InputSandbox =
{"extract.sh","extract.exe","cnaflist.ffl","X.509up_u1000"};
OutputSandbox = {"extract.out","extract.err"};
Requirements =
Member(other.RunTimeEnvironment,"VIRGO1.0") &&
other.CEId =="dell10.cnaf.infn.it:2119/jobmanager-pbs-workq";

Nella sezione n.1 sono specificati il nome dell'eseguibile del job e i parametri di input. L'eseguibile non è il vero "extract.exe" ma uno shell-script "extract.sh". Ciò è dovuto al fatto che, oltre ad eseguire il programma "extract.exe", devono essere compiute una serie di operazioni (settare variabili di ambiente, assegnare dei permessi di accesso ai file, pubblicare i dati ottenuti sul Replica Catalogue). Tutte queste operazioni sono specificate nello script "extract.sh", il cui contenuto è il seguente:

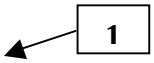
```
####### Shell script “extract.sh”
#
#!/bin/sh
#
#  Memorizza il nome della directory corrente nella
#  variabile curdir
#
curdir=`pwd`
#
#  Memorizza il contenuto del certificato proxy nella
#  variabile “myproxy”
#
myproxy=`echo $X.509_USER_PROXY`
#
#  Assegna i diritti di accesso al certificato proxy
#
chmod u+r ${curdir}/X.509up_u1000
chmod go-rwx ${curdir}/X.509up_u1000
#
#  Esporta il contenuto della variabile X.509_USER_PROXY
#
export X.509_USER_PROXY=${curdir}/X.509up_u1000
#
#  Assegna un nome al file di output
#
outfile=`echo V-$1-$2.out`
#
#  Esegue l’estrazione dei dati con il programma
#  “extract.exe”
#
echo $outfile
$curdir/extract.exe cnaflist.ffl $1 $2 $outfile
#
#  Copia i dati estratti in una directory dello
#  Storage Element
#
cp $outfile /flatfile/virgo
#
#
#
#  Pubblica i dati estratti sul Replica Catalogue
#
globus-job-run dell11.cnaf.infn.it /bin/bash -c 'export
LD_LIBRARY_PATH=/opt/globus/lib:/opt/edg/lib:/usr/local/lib;
gdmp_register_local_file -d /flatfile/virgo'
#
globus-job-run dell11.cnaf.infn.it /bin/bash -c 'export
LD_LIBRARY_PATH=/opt/globus/lib:/opt/edg/lib:/usr/local/lib;
gdmp_publish_catalogue'
```

Nella sezione n.2 del file “extract.jdl” sono specificati i nomi dei file su cui devono essere rediretti lo standard output (“extract.out”) e lo standard error (“extract.err”) del job.

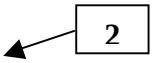
Nella sezione n.3, invece, è specificato l’elenco dei file che costituiscono l’Input Sandbox e l’Output Sandbox del job. Nel file “cnaflist.ffl” c’è l’elenco dei nomi dei file contenenti i frame da cui estrarre i dati. Il file “X.509up_u1000” contiene il certificato proxy.

Nella sezione n.4, infine, sono specificati i requisiti del job. In particolare viene esplicitamente richiesto al Resource Broker che il job venga eseguito su una macchina che abbia settato una variabile di ambiente di nome “VIRGO1.0” e il cui indirizzo è “dell10.cnaf.infn.it” (che corrisponde a quello dello Storage Element di Bologna). In questo modo l’esecuzione del job di estrazione dei dati verrà eseguito direttamente sullo Storage Element, evitando così inutili trasferimenti di dati verso altri Computing Element.

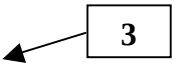
Per effettuare il passo 3) del processo di analisi è necessario creare un file JDL che descriva il job per l’esecuzione dell’algoritmo Matched Filter su un subset di dati. L’eseguibile dell’algoritmo Matched Filter si chiama “mf.exe” ed accetta come input un file contenente 200 template (di nome “template-200”), un file di configurazione (di nome “Mf.cfg”) e un file con i dati da analizzare. Il file JDL, che chiameremo “compute.jdl” avrà il seguente contenuto:



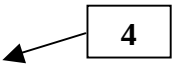

```
Executable  = "compute.sh";
Arguments   = "template-200";
```



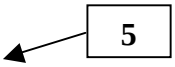

```
StdOutput   = "mf.out";
StdError    = "mf.err";
```



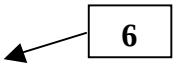

```
InputData       = "LF:V-685112730-600.out";
```



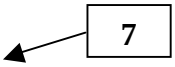

```
InputSandbox    =
{"compute.sh","X.509up_u1000","mf.exe","template-200",
 "Mf.cfg","LFN2PFN.pl"};
OutputSandbox     = {"mf.out","mf.err"};
```




```
ReplicaCatalog  = "ldap://grid011g.cnaf.infn.it:9011/rc=VIRGO
Testbed1 Replica Catalog,dc=grid011g,dc=cnaf,dc=infn,dc=it";
```




```
DataAccessProtocol = "gridftp";
Rank        = other.FreeCPUs;
```




```
Requirements    =
Member(other.RunTimeEnvironment,"VIRGO1.0");
```

Nella sezione n.1 sono specificati il nome dell'eseguibile del job e il parametro di input (il file contenete i template). Anche in questo caso, l'eseguibile è uno shell-script, di nome "compute.sh", che, oltre a lanciare il vero eseguibile dell'algoritmo Matched Filter, esegue altre operazioni (settare variabili di ambiente, effettuare la traduzione da Logical File Name a Physical File Name del file di input, trasferire il file di input dallo Storage Element al Computing Element su cui verrà eseguito il job). Il contenuto del file "compute.sh" è il seguente:

```
####### Shell script "compute.sh"
#
#!/bin/sh
#
#  Memorizza il nome della directory corrente nella
#  variabile curdir
#
curdir=`pwd`
#
#  Memorizza il contenuto del certificato proxy nella
#  variabile "myproxy"
#
myproxy=`echo $X.509_USER_PROXY`
#
#  Assegna i diritti di accesso al certificato proxy
#
chmod u+r ${curdir}/X.509up_u1000
chmod go-rwx ${curdir}/X.509up_u1000
#
#  Esporta il contenuto della variabile X.509_USER_PROXY
#
export X.509_USER_PROXY=${curdir}/X.509up_u1000
#
#  Effettua la traduzione da LFN a PFN del file di input
#
pfn=`$curdir/LFN2PFN.pl`
#
#
#  Trasferisce il file di input dallo Storage Element
#  al Computing Element
#
/opt/globus/bin/globus-url-copy $pfn
file://`/bin/hostname`$curdir/file.input
#
#
#  Esegue l'algoritmo Matched Filter
#
$curdir/mf.exe Mf.cfg $1 file.input
```

Nella sezione n.2 del file “compute.jdl” sono specificati i nomi dei file su cui devono essere rediretti lo standard output (“compute.out”) e lo standard error (“compute.err”) del job.

Nella sezione n.3, invece, è specificato il Logical File Name del file di input, contenente i dati da analizzare. La traduzione da Logical File Name a Physical File Name è fatta all’interno dello script “compute.sh”, chiamando un perl-script di nome “LFN2PFN.pl”. La vera traduzione da LFN a PFN è effettuata dal Resource Broker, all’atto della sottomissione del job. L’RB scrive il PFN del file di input, insieme ad altre informazioni, in una sottodirectory di nome “.brokerinfo” nella home-directory dell’utente collegato alla UI. Il perl-script “LFN2PFN.pl” ha il compito di leggere il PFN dalla sotto-cartella “.brokerinfo” e restituirlo come output.

Nella sezione n.4, come nel caso del file “extract.jdl” è specificato l’elenco dei file che costituiscono l’Input Sandbox e l’Output Sandbox del job.

Nella sezione n.5 è specificata una stringa che identifica l’indirizzo del Replica Catalogue.

Nella sezione n.6 sono specificati 2 attributi: il protocollo da utilizzare per il trasferimento del file di input dallo Storage Element al Computing Element (gridftp) e una preferenza (rank) per la scelta del Computing Element su cui eseguire il job (con FreeCPUs si indica che è preferibile che il job sia eseguito su un CE con una o più CPU inutilizzate).

Nella sezione n.7, infine, è specificato l’unico requisito per l’esecuzione del job. Viene richiesto che l’RB allochi il job su un CE che abbia settato una variabile di ambiente di nome “VIRGO1.0”.

Per automatizzare la sottomissione dei job di analisi si può utilizzare uno shell-script da far girare sulla UI. Questo script esegue un ciclo in cui, viene fatto un “dg-job-submit” del file “compute.jdl” facendo variare, ad ogni passo successivo, il contenuto del file di template. In questo modo il carico computazionale dovuto all’esecuzione di tanti processi Matched Filter viene distribuito, in maniera uniforme ed automatica, su tutti i Worker Node disponibili. Per ogni “dg-job-submit”, viene restituita una stringa, detta PID (Process ID), che identifica univocamente il job all’interno della grid. Attraverso il PID è possibile, interrogando il Logging & Bookkeeping

col comando “dg-job-status”, ottenere informazioni sullo stato di un job. Un possibile shell-script che esegua la sottomissione automatica dei job di analisi è il seguente:

```
######### Shell-script submit.sh
#
#!/bin/sh
#
#
#
#  Viene assegnato un nome ad un file contenente i PID
#  dei job sottomessi alla grid
#
TIMESTAMP=`date +%s`
JOBLIST=`echo dgjoblist-$TIMESTAMP`
#
#
# La directory “templates-archive” contiene tutti i
# sottoinsiemi di template
#
for TEMPLATE_FILE in `ls -1 templates-archive`; do
      cp -f templates-archive/$TEMPLATE_FILE ./template-200
      /opt/edg/bin/dg-job-submit compute.jdl -output $JOBLIST
      rm -f template-200
      sleep 2
done
```

3.4.4.4 Un'interfaccia user-friendly per l'accesso alla grid

I meccanismi di accesso alla grid e di sottomissione dei job descritti nei paragrafi precedenti obbligano gli utenti ad un considerevole lavoro di organizzazione degli script da far girare sulla User Interface attraverso una riga di comando.

Uno dei traguardi che le grid tentano di raggiungere è, invece, quello di fornire agli utenti una visione astratta delle risorse di calcolo e la possibilità di accedere ad esse attraverso delle interfacce semplici ed intuitive.

Un primo approccio, in tal senso, si è avuto nel progetto INFN-Grid con la realizzazione di un "portale web" di accesso alla griglia. Tale sistema prende il nome di **GENIUS**[37] (**G**rid **E**nabled web e**N**vironment for site **I**ndependent **U**ser job **S**ubmission). [33]

GENIUS si è rivelato uno strumento fondamentale durante la seconda parte dello stage; le operazioni di accesso alla grid (con la generazione automatica dei certificati proxy), sottomissione di script JDL, interrogazione di database LDAP e monitoraggio dello stato dei job sono state notevolmente agevolate dall'utilizzo di semplici interfacce web.

Per completezza, nella Figura 3-11 è illustrato uno schema logico che descrive l'accesso alla grid attraverso il portale web GENIUS:

[37] https://genius.ct.infn.it

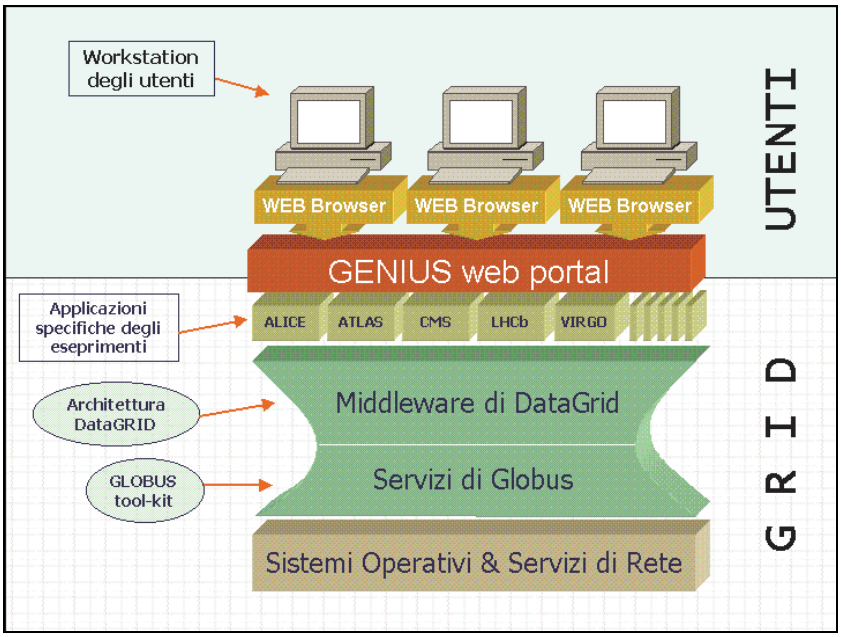


Figura 3-11 Accesso alla grid attraverso GENIUS

4 Conclusioni

Attualmente, la Linux farm realizzata durante lo stage è utilizzata dai ricercatori di VIRGO come un valido strumento per la progettazione, lo sviluppo e il testing di procedure di analisi dei dati, sia in locale che in un contesto di calcolo geograficamente distribuito. Tale strumento è particolarmente versatile in quanto la doppia configurazione dei nodi consente, in maniera rapida, di allocare o deallocare le risorse disponibili a seconda delle esigenze di calcolo (locale o grid-oriented).

La prima parte dello stage (Par. 2.2 e 2.3), particolarmente significativa da un punto di vista della progettazione hardware e software, ha portato alla realizzazione di un potente strumento di calcolo, costituito da un cluster di macchine Linux, da utilizzare in maniera efficiente per l'analisi, in locale, dei dati di VIRGO. Tale sistema, dotato di importanti caratteristiche quali scalabilità, rapidità di installazione, facilità di configurazione, monitoraggio, utilizzo e manutenzione, ha dimostrato di avere un buon rapporto prezzo/prestazioni.

I meccanismi offerti da MOSIX (Par. 2.5) per la migrazione dei processi, il bilanciamento dinamico del carico computazionale e l'ottimizzazione sull'utilizzo della memoria, hanno aumentato in modo significativo le prestazioni del sistema.

Nella seconda parte dello stage (Cap. 3) il sistema di calcolo è stato esteso implementando l'infrastruttura di base che costituisce una grid. A tale scopo sono state utilizzate tecnologie tra le più avanzate disponibili nel campo del calcolo distribuito, del trasferimento dati, dell'Information Technology e delle procedure di sicurezza telematica.

La validità del modello adottato e la fattibilità del progetto sono state confermate dai test effettuati sia durante (Par. 2.6) che dopo (Par. 3.4.4) la fase di prima implementazione.

Per quanto riguarda i servizi principali si può asserire che la gestione delle risorse di calcolo e della security sono già abbastanza adatte allo scopo ma necessitano indubbiamente di miglioramenti nella facilità e comodità di utilizzo.

I problemi aperti sono ancora innumerevoli ed è probabile che in futuro il modello di grid adottato subisca variazioni anche notevoli

dovute, ad esempio, alla interconnessione di griglie computazionali con architetture eterogenee. In tale direzione si muovono, attualmente, progetti quali DataTAG[38], con l'ambizioso traguardo di realizzare delle griglie computazionali transoceaniche, tra l'Europa e l'America.

Il lavoro descritto in queste pagine è stato solo uno dei primi passi di un lungo processo che occuperà, presumibilmente, fisici e informatici per il prossimo decennio ma, da quanto ottenuto, si può concludere che a oggi esistono sia le conoscenze che le tecnologie necessarie allo sviluppo di griglie computazionali capaci di soddisfare le richieste del calcolo in importanti applicazioni scientifiche, quali l'analisi di segnali acquisiti da antenne interferometriche.

[38] http://datatag.web.cern.ch/datatag/

Glossario

API: Acronimo di Application Programming Interface, termine generico che indica un set di librerie che permettono ad un programmatore di inserire alcune funzionalità nei propri programmi.

Benchmark: Programma utilizzato per misurare le prestazioni di un sistema informatico o di un suo specifico componente.

BootP: Boot Protocol, è uno dei protocolli standard che utilizzano un'architettura di tipo client/server per l'assegnazione dinamica di indirizzi IP agli host di una LAN.

Cache: Termine che indica un'area di memoria, spesso notevolmente più veloce delle altre, in cui vengono registrate le informazioni che si intendono utilizzare più di frequente.

Cluster: Termine che indica un gruppo di computer che lavorano congiuntamente per eseguire un unico compito. Si parla, in genere, di cluster di calcolatori per indicare un insieme di calcolatori che usano le proprie risorse in un sistema di calcolo parallelo. Spesso la parola cluster è usata come sinonimo di Farm.

Computing Element (CE): è una risorsa grid in grado di fornire cicli di CPU per l'esecuzione di job. Un CE può anche essere il gateway di un cluster di PC, un supercomputer per l'esecuzione di job paralleli, o una postazione standard di calcolo interattivo in grado di gestire applicazioni grafiche e I/O verso dispositivi di storage

DHCP: Dynamic Host Configuration Protocol, è uno dei protocolli standard che utilizzano un'architettura di tipo client/server per l'assegnazione dinamica di indirizzi IP agli host di una LAN.

Farm: Questo termine indica un gruppo di computer adibito ad una produzione specifica, come ad esempio il lavoro di simulazione o analisi dati per un esperimento. Una farm può essere gestita in modi più o meno complicati dal software che si occupa del job management.

FLOPS: FLoating-point Operations Per Second, è l'unità di misura per la velocità nei calcoli a virgola mobile. Rappresenta il numero di operazioni di calcolo a virgola mobile (con numeri frazionari) eseguiti in un secondo da un processore.

Fork: La fork è la chiamata di sistema (Unix) con cui un processo in esecuzione può generare un sottoprocesso. La chiamata fork può essere utilizzata come jobmanager su un Computing Element.

FPU: Floating Point Unit, è il processore che effettua calcoli in virgola mobile. A volte è integrato nel processore principale, altre volte (nelle architetture più vecchie) è un coprocessore dedicato a questo compito.

FTP: File Transfer Protocol, è un protocollo standard usato comunemente su nternet per il trasferimento di file.

Globus: Insieme di pacchetti software (tool-kit) che costituiscono il nucleo dei servizi forniti da una griglia computazionale grid. Globus viene spesso considerato come un "collante" che tiene insieme e rende omogenee le parti che costituiscono la grid.

GASS: Global Access to Secondary Storage, è un protocollo utilizzato da globus per il trasporto automatico dei file da un punto all'altro della grid.

GDMP: Grid Data Mirroring Package, è un software, con architettura client/server, in grado di replicare (mirroring) in maniera automatica ed asincrona dei file arbitrari (in qualsiasi formato essi siano stati scritti) all'interno della griglia.

GIIS: Grid Index Information Service, parte del servizio MDS che riceve le informazioni dalle varie risorse (tramite i GRIS) e le organizza in un database LDAP.

GIS: Grid Information Service, è la parte del servizio MDS che si occupa di collezionare, organizzare e diffondere le informazioni relative allo stato delle risorse della grid.

GRAM: Globus Resource Allocation Manager, è la parte di globus che gestisce le richieste di un utente o di un processo per quanto riguarda l'allocazione delle risorse.

Grid: Termine generico che indica un'infrastruttura per la condivisione e il coordinamento nell'utilizzo di risorse all'interno di comunità vaste, dinamiche e multi-istituzionali.

Grid-element: elemento di una grid in grado di offrire un particolare servizio.

Grid-oriented: con questo termine si indica un ambiente di calcolo distribuito che si avvale di una infrastruttura Grid.

GridFTP: E' un estensione del protocollo FTP standard che ha, come principale caratteristica, la possibilità di saturare il link su cui avviene la trasmissione dei dati, utilizzando più socket contemporaneamente.

GRIS: Grid Resource Information Service, parte del servizio MDS che si occupa di comunicare ai livelli superiori della struttura gerarchica dell'MDS lo stato di una risorsa.

GSI: Grid Security Infrastructure, è la parte del software che gestisce l'infrastruttura adibita alla sicurezza di una grid.

Host: sinonimo di computer, elaboratore.

HPC: High Performance Computing, con questa espressione si identifica un ambiente di elaborazione in cui le risorse di calcolo hanno prestazioni particolarmente elevate con tempi di risposta molto brevi.

HTC: High Throughput Computing, con questa espressione si identifica un ambiente di elaborazione in cui è richiesta una enorme potenza di calcolo per l'esecuzione di job che durano lunghi periodi di tempo.

HTTP: HyperText Markup Language, è il protocollo standard utilizzato su Internet per la trasmissione e lo scambio di ipertesti.

Information Index (II): grid-element che colleziona in un database le informazioni relative allo stato delle risorse di una Grid.

ISO: International Organization for Standardization, è l'organismo internazionale che si occupa della definizione di standard, quali protocolli, dispositivi, schemi, etc.

JDL: Job Definition Language, è un linguaggio basato su espressioni, con cui è possibile specificare le caratteristiche di un job (parametri di input/output, requisiti di sistema, jobmanager da utilizzare ed altri attributi) in maniera da facilitare la ricerca, da parte del Resource Broker, delle corrispondenze tra requisiti e disponibilità di risorse.

Jobmanager: Un jobmanager è un software che ha il compito di gestire le risorse di un singolo computer o di un cluster, scegliendo come e quando eseguire un job in base alle risorse disponibili e a una serie di regole definite dall'amministratore di sistema.

JSS: Job Submission System, è il sottosistema software che si occupa dell'allocazione di un job su un Computing Element scelto dal Resource Broker.

LAN: Local Area Network, è una rete di calcolatori interconnessi, situati all'interno di un'area limitata del raggio di poche centinaia di metri.

LDAP: Lightweight Directory Application Protocol, protocollo standard per la consultazione di grossi database distribuiti.

LCFG: Local ConFiGuration System, è un tool, con architettura client/server, per l'installazione automatica, configurazione e management centralizzato di macchine con sistema operativo Linux.

Linux: è un sistema operativo Unix che ha, tra le sue caratteristiche, quello di essere opensource, multipiattaforma, multiutente e supportato, a livello mondiale, da una vasta comunità di utilizzatori e sviluppatori.

Logging & Bookkeeping (LO): grid-element che colleziona in un database le informazioni relative allo stato dei job in esecuzione all'interno di una Grid.

MDS: Metadirectory Data Structure, sinonimo di GIS.

MFS: MOSIX File System, è il file system offerto da MOSIX per la condivisione di file all'interno di un cluster di macchine Linux.

MIPS: Million Instructions per Second, unità di misurazione della velocità di un computer, in milioni di operazioni (intere) al secondo. Viene utilizzata nelle prove (benchmark) per confrontare le prestazioni dei diversi modelli di CPU.

MOSIX: Multicomputer OS for unIX é un'estensione del kernel di Linux ed è costituito da una serie di algoritmi per la condivisione adattiva delle risorse, progettati in modo da rispondere, in tempo breve, alle variazioni di utilizzo delle risorse, bilanciando il carico computazionale all'interno di un cluster di workstation.

MPI: Message Passing Interface, è un set di librerie standard per lo sviluppo di applicazioni di calcolo parallelo

NFS: Network File System, è uno dei protocolli standard, basato su un'architettura client/server, per la condivisione di file tra due o più sistemi Unix.

Overhead: E' la quantità di operazioni aggiunte all'esecuzione di un programma da una qualsiasi infrastruttura. Ha come effetto generale una perdita di performance del programma.

PBS: Portable Batch System, è un jobmanager che si occupa dello scheduling e dell'allocazione (statica) dei processi in un cluster locale di macchine fornendo degli strumenti per la gestione di code, accounting delle risorse di calcolo ed elaborazioni di tipo batch

Policy: Una policy, è una serie di regole che stabiliscono il comportamento di un sistema di security, queste regole servono a chiarire in modo univoco chi, quando e come è autorizzato ad accedere ad un sistema informatico.

Replica Catalogue (RC): grid-element che gestisce un database di informazioni relative ai file contenuti sugli Storage Element di una grid.

Resource Broker (RB): è un grid-element che ha il compito di trovare una corrispondenza tra i requisiti espressi dagli utenti per l'esecuzione dei propri job e le risorse disponibili sulla griglia, utilizzando opportuni algoritmi di scheduling.

RSL: Resource Specification Language, è il linguaggio utilizzato in globus per descrivere al GRAM le caratteristiche del job da eseguire.

Socket: E' un canale logico su cui viene trasmesso un flusso di dati tra due computer.

SSL: Security Socket Layer, è un software in grado di creare dei socket per lo scambio di dati crittati tra due computer, utilizzando un meccanismi di crittografia asimmetrica (chiave pubblica/chiave privata).

Storage Element (SE): è un nodo di una grid che fornisce i servizi necessari a immagazzinare, localizzare e replicare i dati. Un SE, inoltre, fornisce ad altri nodi della grid le informazioni relative alla disponibilità dei dati.

Switch: dispositivo hardware multiporta utilizzato per lo scambio di pacchetti tra due o più calcolatori.

TCP/IP: Trasmission Control Protocol/Internet Protocol, è il protocollo standard utilizzato su internet per lo scambio di dati tra due applicazioni.

TFTP: Trivial FTP, è una versione limitata del protocollo FTP standard che non offre, ad esempio, nessun meccanismo di autenticazione.

Tool-kit: è un insieme di pacchetti software, facilmente estendibili, che svolgono compiti specifici e possono essere utilizzati separatamente o in maniera congiunta.

UDP: User Datagram Protocol, protocollo standard per la trasmissione di dati utilizzato in applicazioni che non richiedono garanzia di ricezione dei pacchetti.

User Interface (UI): è un nodo di una grid a cui gli utenti si collegano per sottomettere i propri job. Una UI offre agli utenti un set di comandi e un ambiente testuale, grafico o di tipo web per la sottomissione e il management dei job.

Virtual Organization (VO): gruppo di persone ed entità geograficamente distribuite, multi-istituzionali, in dinamica evoluzione e con interessi comuni, scientifici, economici o amministrativi. Un server VO è un grid-element che colleziona in un database l'elenco e i membri delle Virtual Organization di una Grid.

Worker Node (WN): è un nodo generico che offre potenza di calcolo. Un WN può essere considerato ad esempio come un elemento di una farm locale, o un computer che può offrire, come unico servizio, la sua cpu per l'esecuzione di job.

X.509: Si tratta di uno standard ISO che definisce un sistema di certificazione usato per rendere sicure le comunicazioni elettroniche.

XML: Extensible Markup Language è un linguaggio di codifica estensibile, e rappresenta un formato universale per dati e documenti strutturati sul Web.

Bibliografia

[1] AA.VV. “General overview of the VIRGO Project”
http://www.virgo.infn.it/central.html

[2] F. Barone, L. Milano et al.: “Il problema del calcolo in VIRGO”, addendum al proposal in commissione di gruppo II, INFN, 2001

[3] F. Barone, L. Milano et al.: “VIRGO Computing Plan” VIR-PLA-DIR-7000-122, December 2001

[4] AA.VV. “Cluster Computing White Paper”, ver. 2.0 Final Release, Ed. Mark Baker, University of PortsMouth, UK, December 2000
http://www.dcs.port.ac.uk/~mab/tfcc/WhitePaper/final-paper.pdf

[5] Ken Yap, Markus Gutschke: “Etherboot User Manual”, 2002
http://www.etherboot.org/doc/html/userman.html

[6] R. Coker : “Bonnie++ Documentation“, 1999
http://www.coker.com.au/bonnie++/readme.html

[7] M. M. Weber: “Glibench Documentation“, 2001
http://clibench.daemonware.ch/index.php?seite=articles

[8] C. Kurmann, T. Stricker: “Memory System Performance of High End SMPs, PCs and Clusters of PCs”, 1999
http://www.cs.inf.ethz.ch/cops/software/doc/

[9] Q. O. Snell, A. R. Mikler, J. L. Gustafson: ”NetPIPE: A Network Protocol Independent Performance Evaluator”, 1996
http://www.scl.ameslab.gov/netpipe/paper/full.html

[10] Pallas GmbH: “Pallas MPI Benchmark documentation”, 2000
ftp://ftp.pallas.com/pub/PALLAS/PMB/PMB-MPI1.pdf
ftp://ftp.pallas.com/pub/PALLAS/PMB/PMB-MPI2.pdf

[11] G. R. Warnes: “Simplifying Linux Clusters: MOSIX + ClusterNFS”, 1999

http://clusternfs.sourceforge.net/Presentation.pdf

[12] G. R. Warnes: “Recipe for a diskless MOSIX cluster using ClusterNFS”, 2000
http://clusternfs.sourceforge.net/Recipe.pdf

[13] A. Barak, A. Braverman, I. Gilderman, O. La'adan: “Performance of PVM with the MOSIX Preemptive Process Migration”, Proc. 7th Israeli Conf. on Computer Systems and Software Engineering, Herzliya, pp. 38-45, June 1996.
http://www.mosix.cs.huji.ac.il/ftps/pvm.ps.gz

[14] A. Barak, O. La'adan: “The MOSIX Multicomputer Operating System for High Performance Cluster Computing”, Journal of Future Generation Computer Systems, Vol. 13, March 1998
http://www.mosix.cs.huji.ac.il/ftps/mosixhpcc.ps.gz

[15] A. Barak, O. La’adan, A. Shiloh: “Scalable Cluster Computing with MOSIX for LINUX” Proc. Linux Expo '99, pp. 95-100, Raleigh, N.C., May 1999
http://www.mosix.cs.huji.ac.il/ftps/mosix4linux.ps.gz

[16] S. McClure, R. Wheeler: “MOSIX: How Linux Clusters Solve Real World Problems”, Proc. 2000 USENIX Annual Tech. Conf., pp. 49-56, San Diego, CA., June 2000.
http://www.mosix.cs.huji.ac.il/ftps/usenix.ps.gz

[17] L. Amar, A. Barak, A. Eizenberg, A. Shiloh: “The MOSIX Scalable Cluster File Systems for LINUX”, 2000
http://www.mosix.cs.huji.ac.il/ftps/mfs.ps.gz

[18] F. Barone, L. Milano, R. Esposito et al.: “Evaluation of Mosix-Linux Farm Perforamces in GRID Environment”, Proc. of CHEP (International Conference on Computing in High Energy and Nuclear Physics), pag. 702-703, September 2001

[19] M. Bar: “Distributed OSs: General description of OpenMosix”, 2002

http://sourceforge.net/docman/display_doc.php?docid=9562&group_id=46729

[20] M. Bar: “OpenMosix Internals: How OpenMosix Works”, 2002
http://sourceforge.net/docman/display_doc.php?docid=10390&group_id=46729

[21] F. Barone, L. Milano, R. Esposito et al.: “Mosix Linux Farm Prototype for GW Data Analysis”, 6th Gravitational Wave Data Analysis Workshop, December 2001
http://gwdaw2001.science.unitn.it/abstracts/abstractsgwdaw.pdf

[22] F. Barone, L. Milano, R. Esposito et al.: “Preliminary tests on the Napoli farm prototype for coalescing binary analysys”, VIR-NOT-NAP-1390-196, March 2002

[23] I. Foster, C. Kesselman: “Globus: A Metacomputing Infrastructure Toolkit”, Intl J. Supercomputer Applications, 1997
ftp://ftp.globus.org/pub/globus/papers/globus.pdf

[24] I. Foster, C.Kesselman: “The Grid: Blueprint for a New Computing Infrastrucure”, Morgan Kaufmann Publishers, 1998, Chap. 2
http://www.globus.org/research/papers/chapter2.pdf

[25] I. Foster, C.Kesselman, G. Tsudik, S. Tuecke: “A security architecture for computational grids“, Proc. 5th ACM Conference on Computer and Communications Security, 1998, pag. 83-92
ftp://ftp.globus.org/pub/globus/papers/security.pdf

[26] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman: “Grid Information Services for Distributed Resource Sharing”, Proc. Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.
http://www.globus.org/research/papers/MDS-HPDC.pdf

[27] B. Segal: “Grid Computing: The European Data Project”, 2000
http://web.datagrid.cnr.it/pls/portal30/docs/1442.DOC

[28] AA.VV.: “A Computational and Data Challenge for future INFN experiments; a GRID approach - Outline of the INFN-GRID Project”, 2000
http://www.infn.it/grid/doc

[29] A. Forte: “Certificates How-To”, INFN-Torino, 2001
http://www.to.infn.it/grid/seminari/18042001/

[30] A. Ghiselli: “DataGrid Prototype 1”, INFN-CNAF, 2002
http://grid.infn.it/grid/doc/TERENA-EDG-release1.pdf

[31] F. Giacomini: “Architettura e componenti del middleware di DataGrid”, INFN-CNAF, Workshop “Lezioni sul software e sul calcolo moderno”, Feb. 2002
http://server11.infn.it/grid/doc/gualino-giaco-02.pdf

[32] R. Esposito, G. Tortone et al.: “A Grid Approach to Geographically Distributed Data Analysis for Virgo”, Gravitational Wave Advanced Detector Workshop, May 2002
http://131.215.114.135:8083/related/talks/22/tortone-palomba.pdf

[33] R. Barbera, A. Falzone: “GENIUS (Grid Enabled web eNvironment for site Independent User job Submission) User’s Guide”, May 2002
https://genius.ct.infn.it/manual/genius-manual-v1.3.pdf