

Università degli Studi di Perugia
Facoltà di Scienze Matematiche, Fisiche e Naturali
Anno Accademico 2003/2004



Tesi di Laurea triennale in
INFORMATICA

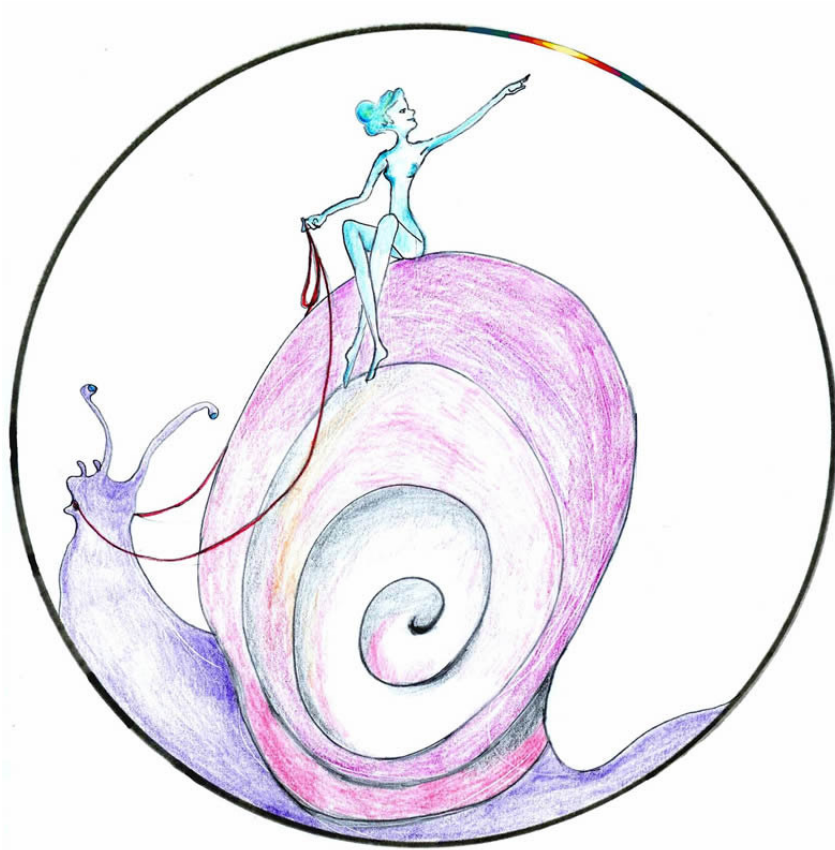
Installazione e configurazione di un sito Grid di sviluppo

Candidato *Leorsini Daniele*

Relatore Prof. *Attilio Santocchia*

Co-relatore *Mirko Mariotti*





“Questo sistema viene visto come un prodigio della natura, sia perché ha (racchiuso) in una macchina un’arte che è prerogativa della mente umana, sia perché significa l’aver scoperto come riprodurre perfettamente tutte le operazioni di quest’arte senza impiegare la mente umana.”

Descrizione del calcolatore di Pascal, 1642

INDICE GENERALE

CAPITOLO 1 INTRODUZIONE.....	7
1.1 INTRODUZIONE GENERALE	7
1.2 STRUTTURA DELLA TESI.....	8
CAPITOLO 2 INTRODUZIONE A GRID	9
2.1 COSA È GRID	9
2.2 LE VARIE REALTÀ ESISTENTI.....	12
2.3 IL CERN, IL PROGETTO LCG E PERSONALIZZAZIONE ITALIANA INFN	13
2.4 IL CONCETTO DI VO.....	16
2.5 PROSPETTIVE FUTURE.....	18
CAPITOLO 3 STRUTTURA DI UN SITO INFN GRID.....	19
3.1 CARATTERISTICHE GENERALI DI INFN GRID.....	19
3.2 IL NODO LCFGNG	20
3.2.1 Il ruolo del Resource Broker	21
3.2.2 Il ruolo dello Storage Element.....	22
3.2.3 Il ruolo del Computing Element	23
3.2.4 Il ruolo dei Worker Node	23
3.2.5 Il ruolo della User Interface.....	23
3.3 SET-UP DI UN SITO INFN GRID.....	24
3.3.1 I file di configurazione	26
3.3.2 I profili XML.....	28
3.4 LA NUOVA RELEASE LCG 2.3.0.....	29
3.5 CENNI SU QUATTOR.....	30
3.6 IL RUOLO DELLE VO	31
3.6.1 Ottenere un certificato	31
CAPITOLO 4 IL SITO DI SVILUPPO GRID-DEV	33
4.1 SITUAZIONE ATTUALE DI GRID A PERUGIA	33
4.2 LO SCOPO DEL SITO DI SVILUPPO.....	34
4.3 DIFFERENZE CON IL SITO DI PRODUZIONE	34
4.4 CONFIGURAZIONE DEL FIREWALL/GATEWAY.....	36
4.5 LA CONFIGURAZIONE DEL SERVER LCFGNG.....	43
4.6 LA CONFIGURAZIONE DEL DNS	44
4.7 LE MODIFICHE AI FILE DI CONFIGURAZIONE: SITE.CFG	48
4.8 L'INSTALLAZIONE DEGLI ALTRI NODI	49
4.9 PROBLEMI RISCONTRATI CON IL KERNEL UFFICIALE	51

INDICE

4.10 MONTAGGIO DEL FILE-SERVER VIA NFS	53
4.10.1 Personalizzazione del file StorageElement-local-cfg.h	54
CAPITOLO 5 TESTING SULLA GRID DI SVILUPPO	55
5.1 IL FILE SERVER	55
5.2 TEST NFS	56
5.3 TEST IOZONE	56
5.4 TEST BONNIE++	58
5.5 LO SCRIPT PERL	59
5.6 CREAZIONE DEGLI UTENTI	61
5.7 CREAZIONE DELLE CODE SUL BACH-SYSTEM DEL CE	63
CAPITOLO 6 CONCLUSIONI	65
6.1 PROSPETTIVE FUTURE DEL SITO DI SVILUPPO	65
6.2 CONSIDERAZIONI FINALI	65
APPENDICE A – LISTATO DELLO SCRIPT PERL	67
APPENDICE B – LISTATO DEL FILE MY_KERNEL.SPEC	69
RINGRAZIAMENTI	73
BLOGRAFIA	75
INDICE DELLE FIGURE	79

CAPITOLO 1 INTRODUZIONE

1.1 Introduzione generale

Il gruppo di ricerca *CMS* (Compact Moun Solenoid) [1] di Perugia partecipa al progetto *LHC* del *CERN*. Il principale obiettivo è quello di analizzare i dati che il rivelatore *CMS* produce in uscita tramite varie tipologie di sensori. La mole notevole di informazioni viene analizzata con applicazioni software sviluppate *ad-hoc* ed ottimizzate per consentire a tutti i membri della collaborazione (situati in Europa, America, Asia) di accedere ai dati prodotti.

Ogni gruppo locale di *CMS* gestisce una propria *farm* di *computing* indipendente ed autonoma per effettuare l'analisi dei dati:

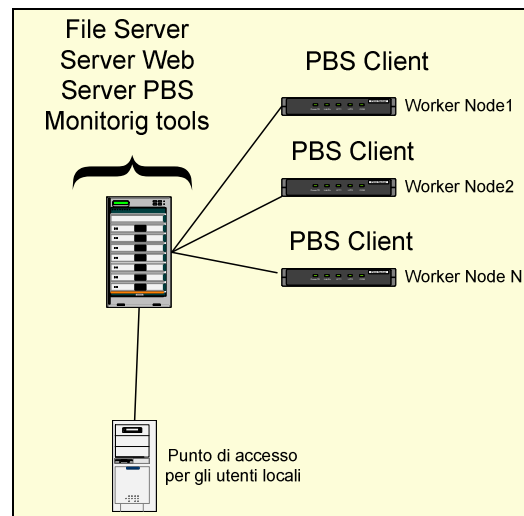


Figura 1 - Farm di produzione CMS Perugia

La configurazione originale della *farm* prevedeva che gli utenti si autenticavano ed entravano nel sistema tramite *SSH* [2] da un unico punto di accesso. I diversi *job* venivano gestiti sui nodi di calcolo dal server *batch PBS* [3] in esecuzione sul server principale (vedi figura sopra). Sullo stesso punto di

accesso era in esecuzione l'interfaccia a *PBS* che consentiva l'interazione con il server da parte degli utenti. La stessa macchina svolgeva il ruolo di servizio *Web* e spazio disco per memorizzare i dati degli utenti.

L'evoluzione dei software di analisi, la necessità di avere più quota disco e più potenza di calcolo, ha portato alla decisione di aderire al progetto *INFN Grid* che nel 2004 ha consentito alla comunità *CMS* Perugia di avere un proprio sito *Grid* locale. Questa decisione ha avuto la naturale conseguenza della dismissione della vecchia *farm* di produzione.

Il mio lavoro di tesi è consistito nel mettere a punto un sito *INFN Grid* di sviluppo (non visibile dal mondo esterno) che ci consentirà di effettuare test di compatibilità di tipo *hardware* e *software* che non possono essere fatti sul sito di produzione reale.

Inoltre, dal momento che il *software* di produzione è molto complesso e nel prossimo futuro cambierà radicalmente, la stessa *farm* di sviluppo ci consentirà di gestire al meglio il passaggio da una versione alla successiva. L'esperienza sviluppata durante le prime installazioni ha infatti suggerito la realizzazione di tale *farm* di test per minimizzare il "down-time" dovuto ai vari *upgrade* che si succedono costantemente. In questo modo i problemi di compatibilità *hardware*, dipendenza del *software* e compatibilità tra i vari pacchetti possono essere studiati e risolti senza interrompere l'utilizzo della *farm* di produzione.

1.2 Struttura della tesi

La tesi si articola su 6 capitoli compresa questa introduzione. Nel secondo capitolo viene introdotto il concetto di *Grid*. Nel terzo viene spiegata nel dettaglio la struttura dell'*INFN Grid* adottata da tutti i siti *INFN* italiani. Nel quarto capitolo viene descritto come è stata configurata la griglia di sviluppo di Perugia. Nel quinto capitolo vengono indicati tutti i principali test che sono stati fatti nella griglia messa a punto. Nel sesto capitolo vi sono le conclusioni e le considerazioni finali.

CAPITOLO 2 INTRODUZIONE A GRID

2.1 Cosa è Grid

Il termine *Grid*, anche se è al singolare, può essere pensato come un insieme di tecnologie che rappresentano l'ultima frontiera nell'ambito del calcolo computazionale distribuito per scopi scientifici e commerciali. *Grid* (significa griglia) consente di utilizzare in maniera efficiente dei nodi interconnessi tramite la rete *Internet*. Immaginate di collegare il vostro pc in rete e di avere subito a disposizione tutte le risorse di calcolo di cui avete bisogno, senza preoccuparvi di dove esse siano e come vi accediate: ebbene *Grid* vuole essere la soluzione a questo problema, che consentirà a diverse istituzioni di condividere risorse distribuite su scala mondiale per raggiungere un obiettivo comune in assenza di un controllo centrale.

Agli inizi degli anni '70, le tecnologie di rete erano troppo primitive per essere sfruttate ed il calcolo computazionale su vasta scala veniva realizzato con l'ausilio di *mainframe* [4] e calcolatori vettoriali [5]. Queste macchine, che come potenza di calcolo oggi possono essere paragonate ai più semplici microcontrollori moderni, avevano costi esorbitanti, richiedevano una manutenzione continua ed i programmatori dovevano realizzare programmi efficienti per sfruttare al meglio tutte le loro potenzialità. La loro storia coincide con i primi elaboratori comparsi sul mercato: la prima generazione era a valvole ed ingombravano enormemente, successivamente con l'invenzione del *transistor* [6], furono introdotti i *mainframe a transistor discreti*, che a differenza dei primi erano un po' più compatti. Con l'evoluzione tecnologica, con l'introduzione sul mercato da parte di IBM del primo computer a basso costo e con l'evoluzione delle reti telematiche si incominciarono a costruire i primi *cluster* [7]: insieme di macchine configurate in maniera tale da poter colloquiare tra di loro attraverso una rete. Il calcolo computazionale era diventato distribuito. I programmatori e sviluppatori non si

dovevano più scontrare con i limiti tecnologici dei primi *calcolatori vettoriali* e dei *mainframe*, la loro attenzione si spostò quindi alla ricerca di algoritmi efficienti per la gestione di comunicazioni all'interno di un *cluster*. E' in questo periodo che vennero concepiti software dal calibro di *MPI (Message Passing Interface)* [8], *PVM (Parallel Virtual Machine)* [9], *HPF (High Performance Fortran)* [10] insieme ai paradigmi di programmazione parallela quali *Task Farm* [11] e *PipeLine* [12].

La necessità di avere una potenza di calcolo sempre maggiore, si scontrò alla fine con i limiti dei collegamenti di rete: infatti, le reti incominciavano a costituire dei colli di bottiglia. Questa esigenza portò allo sviluppo di tecnologie in grado di ottimizzare la ricerca di informazioni: è in questo contesto che si inserisce e viene concepita la tecnologia di *Grid*. Il concetto di calcolo computazionale su *Grid* nasce con l' esigenza di elaborare un ingente quantitativo di dati (*Petabyte*) da parte di molte discipline scientifiche e soprattutto da parte della Fisica delle alte energie. La griglia funziona grazie all'enorme sviluppo che ha avuto la rete *Internet* negli ultimi anni, che consente di mettere in contatto i diversi centri di ricerca sparsi nel mondo permettendo così un'elaborazione più efficiente dei dati sperimentali. *Grid* ha infatti l'obbiettivo principale di mettere a disposizione di un utente o gruppo di persone collocati in qualsiasi località del mondo, le risorse ottimali per quel tipo di dato condiviso dall'utente o gruppo stesso.

Nel 1990 gli sviluppatori e ricercatori, si riunirono prima nel *Grid Forum* e poi nell'attuale *Global Grid Forum* [13] dove vengono definiti gli standard per questo tipo di tecnologia. Le principali aree di interesse sono:

- Gestione sistemi ed automazione
- Gestione del lavoro e delle prestazioni
- Sicurezza
- Gestione Servizi
- Gestione dei *Logical Resource*
- Servizi di *clustering*
- Servizi di connettività
- Gestione della parte fisica (*hardware*)

Oggi come oggi, *Grid* è un'infrastruttura globale che coinvolge nello sviluppo varie collaborazioni appartenenti a tutti i continenti.

Dal punto di vista fisico, *Grid* può essere vista come una combinazione di computer con architetture eterogenee, collegati tramite una rete. Questo *network* appare alla vista di un ipotetico utilizzatore come un unico super computer che consente di realizzare un modello computazionale per risolvere problemi che utilizzano un enorme numero di dati.

Le risorse che mette a disposizione la tecnologia *Grid* sono numerose e possono comprendere ad esempio:

- Portali scientifici: *software user-friendly* in grado di far accedere in maniera semplice l'utente alle risorse disponibili.
- Calcolo distribuito: *workstation* di fascia alta e reti telematiche insieme costituiscono una potenziale risorsa di calcolo.
- Analisi su larga scala: servizi in grado di mettere a disposizione all'utente applicazioni in grado di ottimizzare l'analisi di dati su larga scala.
- Lavoro di gruppo: *Grid* non offre solo l'agglomerazione di dati ma anche di risorse umane.

Negli ultimi dieci anni, lo sviluppo di *Grid* in genere è andato sempre più nella direzione di applicativo sviluppato a strati:

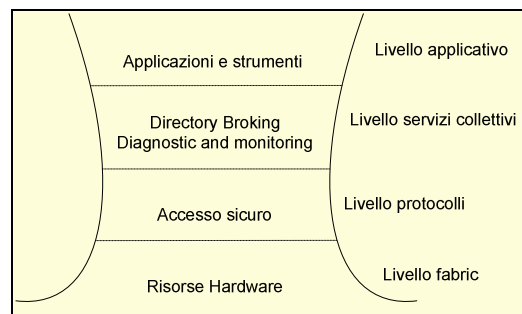


Figura 2 - gli strati del middleware Grid

In cui:

- *Livello Fabric* comprende le componenti *hardware*.
- Livello Protocolli contenente le autenticazioni e comunicazione all'interno della griglia.
- Livello Servizi Collettivi: contiene i servizi e le *API* [14] che permettono la comunicazione fra applicazioni.
- Livello Applicativo: contiene le applicazioni rivolte agli utenti.

2.2 Le varie realtà esistenti

Per poter realizzare una griglia computazionale (*Grid*) abbiamo bisogno di un particolare software predisposto allo scopo. Di implementazioni di *Grid* ce ne sono più di una, ma il concetto base è sempre quello: condividere in maniera ottimale le risorse messe a disposizione.

Il primo software in grado di implementare *Grid* è datato anni '90. Questa griglia venne presentata in una conferenza sul calcolo distribuito in America e venne denominata *I-WAY* [15] ed era in grado di mettere in comunicazione 17 *farm* scientifiche. Oggi come oggi, i tentativi di implementare una *Grid* comune sono molti; di seguito viene proposto un elenco di progetti tutti focalizzati a questo scopo:

- *Condor* [16]: è nato come progetto nel 1985 da Miron Livny [17] presso l'Università del Wisconsin. Dimostrò che all'interno di un ateneo molte stazioni di lavoro erano inattive. Da qui, utilizzando la natura “*Multitasking*” [18] di *Unix* [19] diede origine a questo progetto, che è attualmente in grado di utilizzare migliaia di computer nei periodi di inattività degli stessi.
- *Seti@home* [20]: nato dall'idea di David Anderson [21] che pensò di fare utilizzare su base volontaria (anche privati), l'enorme quantità di pc sparsi in tutto il mondo per analizzare i dati provenienti dal radio telescopio di Arecibo (Messico). Il tutto sfruttando i “tempi morti” dei pc stessi.
- *Globus* [22]: è un particolare *toolkit* su cui si basano tutti i moderni software di *Grid* (*Datagrid*, *LCG* ecc...) *opensource* rilasciato con la licenza *Globus Toolkit Public License (GTPL)* [23]. I servizi di *Grid* sono implementati tramite *API* ed *SDK* [24] nel *Globus toolkit*. Il generico programmatore può sfruttare queste *API* o le chiamate a *Globus* per la programmazione in ambiente C dei servizi che vuole realizzare all'interno della griglia computazionale. Il *Globus toolkit* si concentra su quattro caratteristiche: sicurezza, gestione delle risorse, gestione dei dati, gestione delle informazioni.
- *European Datagrid (EDG)* [25]: è un progetto finanziato della comunità europea, che coinvolge anche le realtà *Grid* nazionali quali *INFN*, *UK* ecc...

- *EGEE* [26]: è anche questo un progetto sponsorizzato dalla comunità europea. Ha l'obiettivo di realizzare una infrastruttura *Grid* robusta e coerente, mantenere ed aggiornare continuamente il *software* con cui si realizza *Grid* e supportare l'industria e la comunità scientifica nel suo utilizzo.

2.3 Il CERN, il progetto LCG e personalizzazione italiana INFN

Il *CERN* (Centro Europeo per la Fisica delle particelle) [27] è oggi il più importante centro di ricerca mondiale per la fisica delle particelle. In un'Europa appena uscita dalla seconda guerra mondiale, le condizioni per fondare il *CERN* non erano certo semplici, ma alcuni stati erano fortemente convinti dell'importanza del centro e convinsero gli altri ad aderire all'iniziativa: così il 29 settembre 1954, l'organizzazione europea per la ricerca nucleare, il *CERN*, vide la luce.

Oggi il centro è impegnato soprattutto alla costruzione dell'*LHC* (*Large Hadron Collider*) [28], il nuovo acceleratore di particelle che dovrebbe iniziare a funzionare nel 2007 grazie ad una collaborazione internazionale che vede coinvolti anche gli Stati Uniti ed alcuni paesi asiatici.

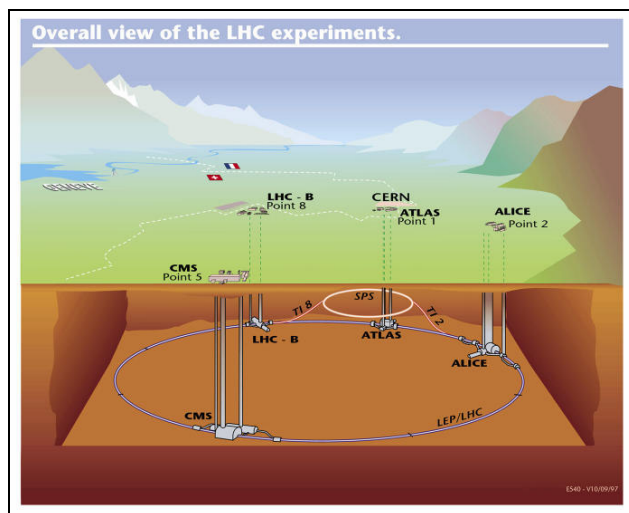


Figura 3 - Il progetto LHC

Questo progetto complessivamente comprende oltre sei mila persone appartenenti a centri di ricerca di tutto il mondo suddivisi in quattro esperimenti:

- *ATLAS* [29] l'obiettivo di questo esperimento è quello di raccogliere informazioni sulla particella di *Higgs* [30] e di scoprire particelle super-simmetriche previste da estensioni del *Modello Standard* [31].

- *CMS (Compact Moun Solenoid)* [1] come nell'esperimento *ATLAS* questo ha l'obiettivo di raccogliere informazioni sulla particella di *Higgs*. *CMS* differisce da *ATLAS* per i metodi e per le tecnologie impiegate nelle osservazioni delle traiettorie e nella misura dell'energia delle particelle prodotte negli urti. Il magnete di *CMS* produrrà un campo magnetico di 4 *Tesla* [32] pari a 80000 volte il campo magnetico terrestre.
- *ALICE* [33] rivelatore di prodotti di collisione tra ioni pesanti. Questo esperimento in particolare, indagherà sul plasma di *quark* [34] e *gluoni* [35]: lo stato in cui si viene a trovare la materia in presenza di energia molto elevata.
- *LHC-B* [36] esperimento che studia le particelle che contengono il *quark b*. Queste particelle saranno prodotte in grande quantità dalle interazioni di protoni in *LHC*.

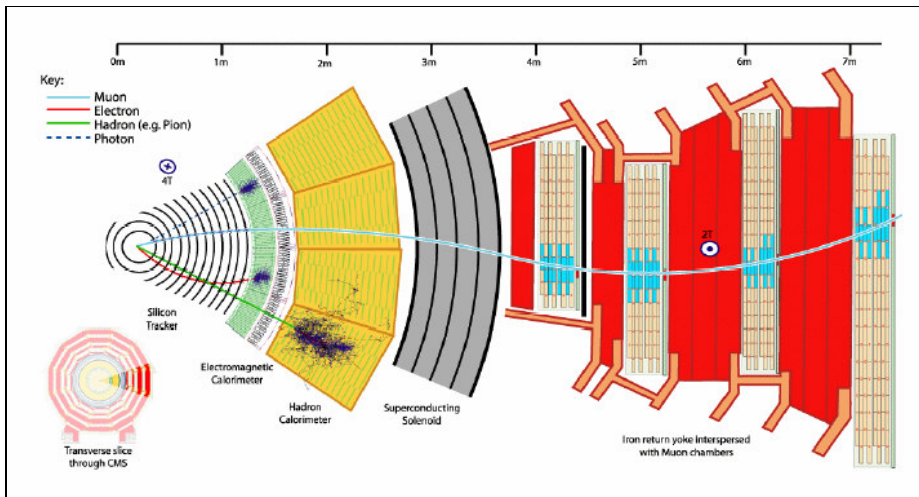


Figura 4 - sezione di CMS

Quando l'acceleratore sarà messo in funzione produrrà tra i dodici e i quattordici *PetaByte* [37] di informazioni all'anno, paragonabili a oltre venti milioni di CD. Il progetto *LCG (Lhc Computing Grid project)* [38] è stato concepito per gestire al meglio tutta questa mole di informazioni con l'intento di realizzare un'infrastruttura *Grid* globale, integrando i centri di ricerca scientifici sparsi per l'Europa, l'America e l'Asia. *LCG* ha cominciato i lavori nel 2002 con

la versione 1.0 e terminerà nel 2008 accompagnando la prima fase operativa di *LHC*.



Figura 5- siti LCG nel Mondo

Attualmente, la *Grid LCG* è utilizzata per effettuare test ed analisi su dati simulati, dal momento che l'acceleratore non è ancora attivo. Attualmente viene adottata la versione 2.3 di *LCG* da parte dei diversi siti che compongono la griglia di produzione *LHC*.

L'Italia come nazione, partecipa a questo progetto con i gruppi locali di ricerca *CMS*, *ALICE* e *ATLAS*. Tali gruppi sono coordinati in termini nazionali dall'*INFN* [39]. La stessa *INFN* costituisce un partner sulla realizzazione del software *LCG* ed effettua lei stessa delle personalizzazioni. Tale software è stato denominato *INFN Grid*. Il progetto *INFN Grid* nasce nel 1999 per realizzare la prima *Grid* italiana basata sulla rete del *GARR* [40]. Questo progetto, oggi comprende più di venti siti appartenenti ciascuno alle diverse università (soprattutto dipartimenti di Fisica e *INFN*) dislocate nel territorio nazionale.



Figura 6 - siti INFN Grid in Italia

2.4 Il concetto di VO

Un aspetto che riveste un ruolo fondamentale all'interno dell'infrastruttura di tipo *Grid LCG* sono gli utenti, i loro gruppi e le politiche di autorizzazione e di autenticazione nel sistema. La griglia, a livello organizzativo, è composta da livelli gerarchici di decisione:

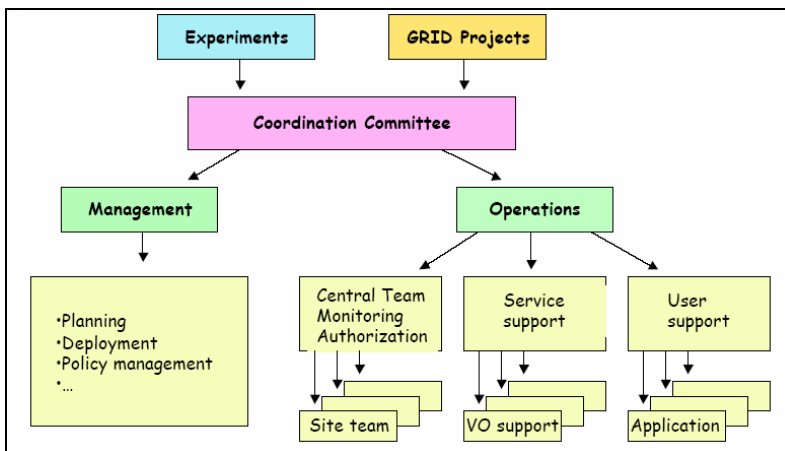


Figura 7 - gerarchia di LCG

Le *Virtual Organization (VO)* [41], sono le comunità di utenti che intendono accedere all'infrastruttura di produzione. Queste concordano con il management le politiche di condivisione ed uso delle risorse. Le *VO* possono far parte di una o più strutture gerarchiche, sono dinamiche (in pratica il numero di utenti che

comporre una *VO* può variare nel tempo come le relazioni tra intere *VO*), un utente può appartenere a più *VO*, ed infine sono coinvolte nelle politiche di autorizzazioni e della gestione dell'informazione in un sito *Grid*. All'interno del *Globus Toolkit*, le *VO* sono implementate tramite il meccanismo dei *grid-mapfile*: un file contenente l'elenco degli utenti che sono autorizzati ad accedere alle risorse. Questo file è organizzato per righe, su ogni riga è collocato il nome utente ed il suo certificato per essere riconosciuto dal sistema. Es:

```
"/C=IT/O=INFN/CN=Daniele Leorsini/Email=leorsini@pg.infn.it" leorsini
```

Le *VO LDAP* costituiscono il primo tentativo di implementazione del concetto di *VO* realizzata dall'*INFN* nell'ambito del progetto *EDG*. Nel particolare, la *VO* è organizzata da un albero *LDAP* (*Lightweight Directory Access Protocol*) [42] contenente informazioni sugli utenti. Gli utenti sono a loro volta organizzati in gruppi, che a loro volta possono essere organizzati in sotto-gruppi.

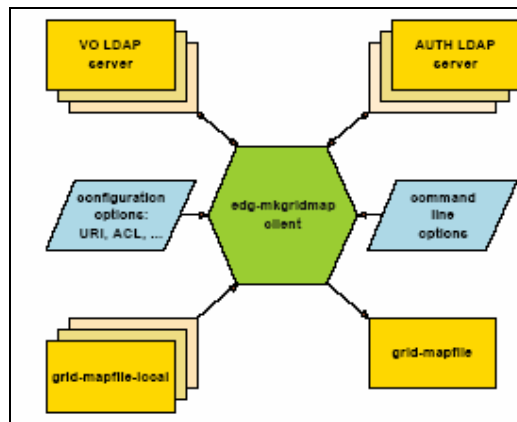


Figura 8 - Rappresentazione delle VO tramite LDAP

Tecnicamente non è un sistema molto flessibile ed è per questo motivo che recentemente è stato introdotto il sistema *VOMS* (*Virtual Organization Membership Service*) [43]. Tale sistema è realizzato da un team facente capo all'*INFN*. Il *VOMS* è un'estensione del sistema *GSI* [44] a cui si aggiungono informazioni di autorizzazioni verificabili in modo indipendente da parte dei membri della *VO*. Il *VOMS* inoltre offre supporto agli utenti stessi ed ai gruppi. Il database che gestisce il *VOMS* è di tipo relazionale (*RDBMS*) ed i dati di ciascuna *VO* sono conservati in database distinti.

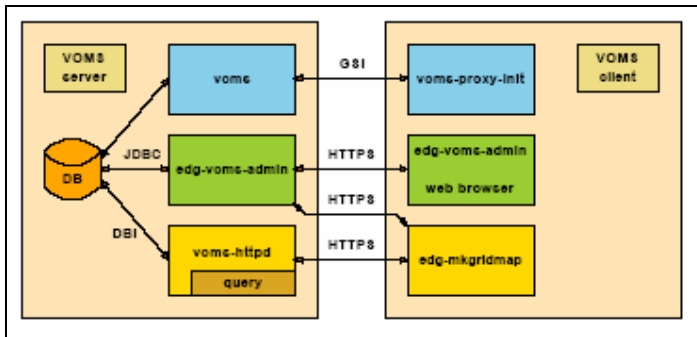


Figura 9 - Organizzazione delle VO con VOMS

2.5 Prospettive future

Il *Grid computing* è già una realtà. In futuro si avrà a disposizione una enorme potenza di calcolo e un enorme quantità di spazio disco che fino a pochi anni fa erano inimmaginabili. Il computer connesso potrà ulteriormente abbassare il suo costo (per l'azienda o per la famiglia) purché faccia parte di una *Grid* che renumeri, in qualche misura, il suo utilizzo. La comunità scientifica sarà quella che ne trarrà sicuramente più vantaggio: potrà estendere ed eguagliare le opportunità di ricerca e di accesso alle frontiere anche ai ricercatori più lontani, con un effetto dirompente sulla produttività e conoscenza del singolo ricercatore. L'ambiente *Grid* potrà sicuramente generare nuove comunità al di fuori dell'ambiente scientifico (*P2P*). Si potranno creare database globali costruiti intorno a comunità professionali: planetario universale (*NASA* [45]), archivio televisivo globale, archivio del Genoma umano ecc...

E' interessante notare inoltre che l'implementazione del *Grid computing* derivi dalla comunità scientifica, la stessa che ha creato il web (*CERN*): la storia sembra ripetersi, le innovazioni fondamentali della rete non provengono dall'industria commerciale ma dalla ricerca, quell'industria sarà poi uno dei tanti soggetti che potrà beneficiare da tale innovazione!

CAPITOLO 3 STRUTTURA DI UN SITO INFN GRID

3.1 Caratteristiche generali di INFN Grid

Per realizzare un'infrastruttura *Grid*, l'Istituto di Fisica Nucleare (*INFN*), si avvale di un particolare software denominato *LCG2*, sviluppato al *CERN*. Tale *software* viene modificato per adattarlo alle esigenze dei gruppi di ricerca italiani. Le differenze più sostanziali sono molteplici e vanno dal tipo di supporto per i *job MPI* (Message Passing Interface), fino al tipo di installazione (con rete privata e non). La versione italiana di *LCG2* prende il nome di *INFN Grid*, ed entrambe sono *open source*.

Volendo dare una classificazione a questo tipo di *software*, si può dire che rientra all'interno di una particolare classe di programmi, denominata *middleware*. Quest'ultima consiste in un insieme di servizi che, una volta installati, vanno ad interporre tra il sistema operativo e l'utente, gestendo l'autenticazione per entrare nel sistema, l'autorizzazione per l'utilizzo delle risorse ad essi destinate e in generale, della sicurezza di tutto l'ambiente. Questi servizi, sono di fondamentale importanza perché vengono utilizzati all'interno di un sistema distribuito.

Gli utenti gestiti da *INFN Grid* vengono raggruppati in *Virtual Organizations (VO)*. La loro autenticazione ed il trasferimento dati all'interno di *INFN Grid*, viene gestita da un infrastruttura chiamata *Grid Security Infrastructure (GSI)*, propria del *middleware*. Per realizzare ciò, vengono utilizzati i certificati *X.509* [46], la crittazione della chiave pubblica dell'utente ed il protocollo *Secure Socket Layer (SSL)* [47].

INFN Grid permette la condivisione di risorse eterogenee, fornendo agli utenti la possibilità di utilizzarle in maniera ordinata.

INFN Grid è composta da un insieme di siti interconnessi tramite la rete *Internet* dove sono installate delle *farm* di computer composte da diverse macchine che svolgono funzioni diverse:

- *Resource Broker (RB)*,
- *Computing Element (CE)*,
- *Storage Element (SE)*,
- *User Interface (UI)*,
- *Worker Node*,
- *Server LCFGng*.

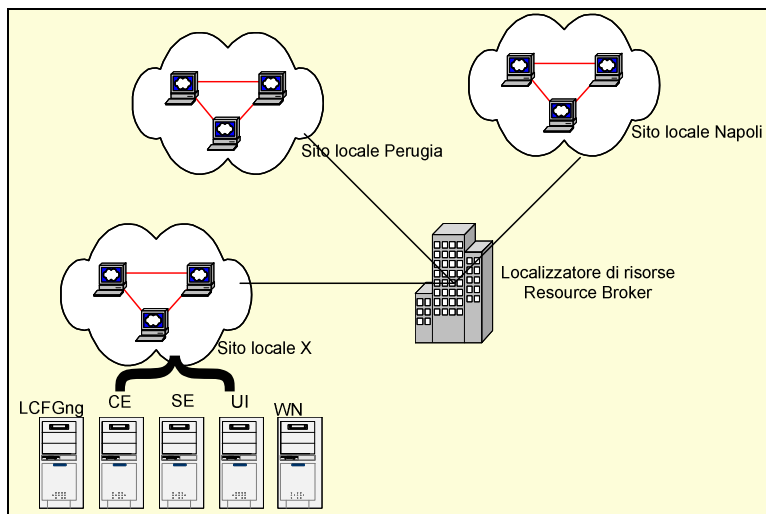


Figura 10 - Struttura globale di INFN Grid

3.2 Il nodo LCFGng

Il nodo *LCFGng* è lo strumento predisposto all'installazione o all'aggiornamento automatico dei nodi presenti in un sito *INFN Grid*. La sua presenza è dovuta al fatto che l'installazione manuale del *middleware* è molto complicata e dispendiosa in termini di tempo, anche per siti medio/piccoli. E' quindi un componente esterno a *Grid* sviluppato da terze parti. E' composto da un server centrale e da dei client che sono in esecuzione sulle macchine del sito locale *Grid*.

L'installazione di un nuovo nodo avviene agendo su dei file di configurazione che andranno poi interpretati con un apposito script. Il prodotto di questa

operazione genererà un file *XML* [48] contenente tutte le informazioni (*kernel*, *account* ecc..) per la macchina interessata all'aggiornamento.

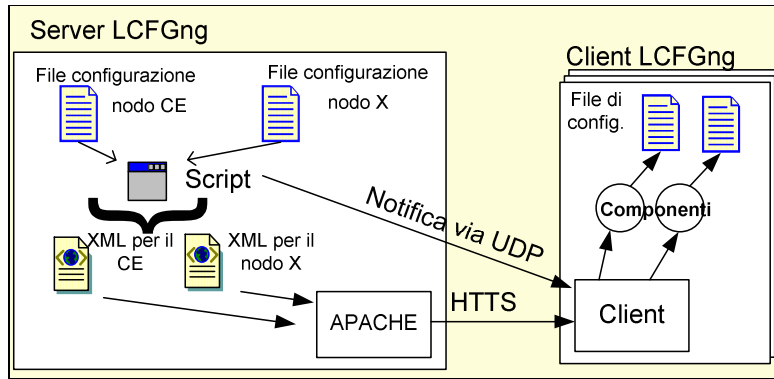


Figura 11 - Funzionamento di LCFGng

L'*upgrade* del nodo configurato avviene in maniera automatica: il server notifica il cambiamento del profilo via *UDP* [49] e successivamente, il client aggiorna il proprio profilo che risiede sul server, tramite una connessione protetta *HTTPS* [50]. In alternativa si può forzare questa operazione tramite un apposito *script* (da eseguire sul server o sul client) oppure facendo il *reboot* della macchina: in quest'ultimo caso un demone in esecuzione sul client andrà automaticamente a controllare se il profilo è stato cambiato o meno, eseguendo se necessario l'aggiornamento.

3.2.1 Il ruolo del Resource Broker

Prima di dare una definizione rigorosa su come funziona il *Resource Broker*, occorre definire un elemento fondamentale per la sottomissione di *job* in *INFN Grid*: il *WMS (Workload Management System)*. Questo componente è l'elemento centrale per la gestione/sottomissione di *job* e la loro distribuzione verso un *Computing Element* appropriato, in accordo alla richieste dell'utente ed alle risorse disponibili in quell'istante. Il *Resource Broker* è la macchina su cui girano i servizi di *WMS* e comprendono:

1. *Il Network Server*, adibito all'accettazione dei *job* provenienti dalle UI.
2. *Workload Manager*, che costituisce il nucleo centrale del *WMS*.
3. *Match Maker*, grazie al quale vengono individuate le risorse migliori per il tipo di *job* sottomesso.

4. *Job Adapter*, componente in grado di preparare l'ambiente per l'esecuzione del *job* sottomesso.
5. *Job control Service*, che fornisce l'interfaccia grazie al quale è possibile gestire i *job*.

Le richieste di *job* vengono generate dagli utenti tramite un file chiamato *JDL* [51] che tra l'altro, permette di specificare direttamente il *CE* su cui girare i *job*, specificare un *SE* dove memorizzare l'output dei *job* e specificare speciali attributi per una individuazione più consona del *CE* stesso.

Se prevediamo la presenza di questo componente all'interno del nostro sito, dobbiamo configurare anche altre due macchine: il server *BDII* e il *proxy server*. Il server *BDII* è adibito alla gestione di un servizio *LDAP* [42] per le *VO*, mentre il server *proxy* ha la funzione di generare dei certificati temporanei per la sottomissione dei *job*.

3.2.2 Il ruolo dello Storage Element

All'interno di un sito *INFN Grid* lo *Storage Element* è il componente a cui viene affidata la fase di memorizzazione delle informazioni: esso infatti offre un accesso uniforme allo spazio disco presente sulla macchina. Questo elemento della griglia può comodamente controllare un enorme numero di *array* di dischi di varia natura, e quindi diversi *Terabyte* di dati.

All'interno dello *Storage Element* è in esecuzione il server *GSIFTP*, in grado di offrire le funzionalità di un normale servizio *FTP* [52], utilizzando però l'infrastruttura *GSI* per un più veloce e sicuro trasferimento file da o verso lo *Storage Element*.

Oltre al *GSIFTP* è presente anche lo *Storage Resource Manager (SRM)*, in grado di gestire dinamicamente il contenuto dello spazio disco in qualsiasi momento. Questo componente interagisce con il sistema operativo del nodo per le operazioni di archiviazione e con diversi protocolli, quale quello utilizzato proprio dal servizio *GSIFTP*, per il trasferimento file.

Inoltre sono presenti i servizi di *Replica Catalogue* e *Replica Locations Service*, in grado di individuare e tracciare le copie di dati.

Lo *Storage Element* deve avere un certificato installato valido per poter essere mappato dal *Resource Broker*. Tale certificato è da richiedere presso la *INFN Certification Authority (CA)*.

All'interno di un sito *INFN Grid* è possibile prevedere la presenza di uno o più macchine configurate per svolgere il compito di *Storage Element*. Nella versione corrente di *INFN Grid*, tutti i siti ne hanno almeno una.

3.2.3 Il ruolo del Computing Element

Il *Computing Element* è l'elemento in grado di controllare e monitorare la potenza computazionale del sito *Grid*, fornendo un'interfaccia uniforme a queste risorse. Nel nostro caso, la potenza di calcolo è contraddistinta da dei nodi chiamati *Worker Node*.

Il *Computing Element* può essere visto come una porta, in grado di gestire le comunicazioni tra *Worker Node* e gli altri elementi della griglia.

Le richieste di *job* vengono fornite dal *Resource Broker*, la loro gestione e sottomissione avviene tramite il *batch system OPENpbs*.

Come avviene per lo *Storage Element* anche questo nodo deve possedere un certificato di riconoscimento.

3.2.4 Il ruolo dei Worker Node

I *Worker Node* sono quelle macchine adibite al calcolo computazionale all'interno di un sito *Grid INFN*. Per la loro gestione/monitoraggio vengono installati su di essi diversi client. I due più importanti sono *openPBS* per il carico di lavoro e *GRIDICE* per il monitoraggio. Questi client a loro volta comunicano con i *Computing Element* presenti nel sito, adibiti al loro controllo.

3.2.5 Il ruolo della User Interface

La macchina che ha il ruolo di *User Interface (UI)* all'interno di un sito *INFN Grid*, è forse quella che riveste il ruolo più importante per l'accesso alla griglia da parte dell'utente finale. In questa, il generico utente deve avere un proprio *account* personale ed un certificato installato valido, per poter sottomettere *job* in *Grid*. Essa inoltre, offre un ampio set di comandi per la gestione dei file. Generalmente questi permettono:

- Effettuare repliche di dati
- Eseguire la ricerca di repliche di dati
- Copia di file verso uno o più *SE*

Inoltre è possibile effettuare anche il monitoraggio dei propri *job*, infatti:

- Dalla *UI* si possono sottomettere job
- Dalla *UI* si possono controllare i job
- Dalla *UI* si può visualizzare gli output dei job

Generalmente le *UI* sono configurate per sottomettere *job* ad un *Resource Broker* predefinito.

Di *User Interface* all'interno di un sito *Grid* ce ne possono essere più di una, tutto dipende da quanti utenti sono attivi all'interno del sito locale e da quanti utilizzano la stessa macchina per svolgere carichi di lavoro di vario genere (si pensi ad esempio ad utenti che sviluppano codice di analisi ed hanno quindi bisogno di avere l'output del codice in breve tempo). Quindi è molto importante saper dimensionare bene l'*hardware* della stessa *User Interface* in relazione al tipo di utenza che si avrà ed al tipo di lavoro che si vorrà svolgere.

Nelle future *release* di *INFN Grid*, sarà previsto un pacchetto *software* da installare sul proprio pc personale che consentirà di “trasformare” il proprio *desktop* in *User Interface*. Questo porterà indubbiamente innumerevoli benefici, quali:

1. La possibilità di avere più *UI* all'interno di un sito locale contenendo i costi
2. La possibilità di poter utilizzare la *User Interface* per svolgere calcoli anche complessi senza penalizzare altri utenti attivi sulla griglia locale
3. Diminuzione delle fasi di gestione/manutenzione di un sito, perché le *UI* presenti fisicamente nelle *farm* di produzione potranno essere una o addirittura nessuna.

3.3 Set-up di un sito INFN Grid

L'installazione di un sito *INFN Grid* viene gestita interamente dal server *LCFGng*. La configurazione di questo nodo avviene in passi successivi:

1. Installazione della distribuzione *Linux Red Hat 7.3* con servizi di rete
2. *Download* ed installazione nel sistema installato del pacchetto *git-lcfgutils*

La versione 7.3 di *Red Hat* è dovuta alle esigenze delle varie *VO*: il *software* di ogni esperimento è compatibile solo con questa distribuzione. Il *kernel* che troveremo nel sistema è il 2.4.27.

L'installazione del pacchetto *git-lcfgutils* ci consente di utilizzare appositi strumenti per il *set-up* del nodo *LCFGng*, in dettaglio:

1. *git-download* (esegue il *download* di tutti i pacchetti necessari per la versione di *INFN Grid* specificata nell'argomento del comando)
2. *git-updatesrvpackages* (installa il server *LCFGng*)
3. *git-buildinstallroot* (crea l'immagine del sistema operativo da utilizzare per l'installazione sugli altri nodi)
4. *git-createhdfiles* (crea gli *hdfiles* che consentono al server *LCFGng* di risolvere tutte le dipendenze *software* del sistema)

Una volta completate tutte queste operazioni, si avranno a disposizione tutti gli strumenti necessari all'amministrazione del sito *Grid*. Le directory che sono state create durante la precedente fase e che ci interesseranno sono:

1. `/var/obj/conf/server/source/` contenente i file che andranno compilati per generare l'*XML*;
2. `/var/obj/conf/server/source/git-2.2.0/` contenente i file di configurazione del sito *INFN Grid* ;
3. `/var/obj/conf/server/web/` directory principale del server *web apache*, contenente i profili *XML*. Viene utilizzata per l'installazione/aggiornamento dei nodi;
4. `/tftpboot/` contenente l'immagine base del *kernel* utilizzata dai nodi per l'installazione via *Preboot eXecution Enviroment (PXE)* [54].
5. `/usr/src/redhat/` che come vedremo servirà nell'installazione di pacchetti personalizzati all'interno di un nodo, non previsti dalla *release* ufficiale di *INFN Grid*.

Come già accennato in precedenza, ogni nodo ha i propri file di configurazione all'interno dei quali vengono definite le proprie variabili. I file hanno una struttura gerarchica ottenuta utilizzando la funzione *include*. Il linguaggio, e quindi anche la sintassi, è quella utilizzata dal linguaggio C/C++.

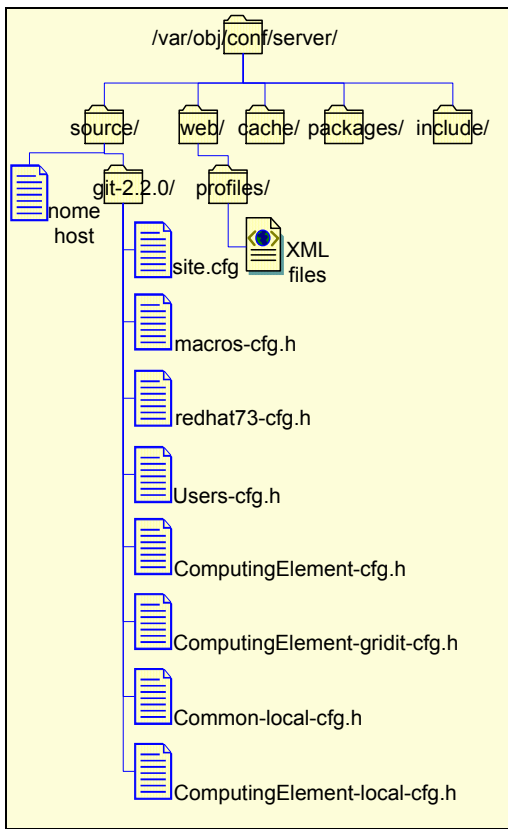


Figura 12 - Struttura directory di LCFGng

In particolare si avrà per l'ipotetico file nomehost:

```

#define HOSTNAME nomehost

#include "git-2.2.0/macros-cfg.h"
#include "git-2.2.0/site-cfg.h"
#include "git-2.2.0/redhat73-cfg.h"
#include "git-2.2.0/Users-cfg.h"
#include "git-2.2.0/ComputingElement-cfg.h"
#include "git-2.2.0/ComputingElement-gridit-cfg.h"
#include "git-2.2.0/Common-local-cfg.h"
#include "git-2.2.0/ComputingElement-local-cfg.h"
    
```

3.3.1 I file di configurazione

I file che vengono utilizzati per la configurazione del sito sono:

1. *site.cfg*;
2. *StorageElement-local-cfg.h*;
3. *ComputingElement-local-cfg.h*;
4. *UserInterface-local-cfg.h*;

5. *Common-local-cfg.h*;
6. *WorkerNode-local-cfg.h*;
7. ed in generale tutti i file **-local.cfg.h*, in base al tipo di nodo che si vuole installare.

Il file *site.cfg*:

Questo file è contenuto all'interno della cartella principale `git-2.2.0/` e contiene la definizione delle variabili principali per la configurazione di tutti i nodi del sito *Grid*. In esso vengono indicati:

- le *VO* operanti sul sito;
- la versione di *INFN Grid* installata;
- il tipo di macchina del sito (driver *hardware*);
- i nomi e gli indirizzi *IP* [55] di tutti i nodi;
- se i *WN* sono in una rete nascosta o no;
- il tipo di *scheduler* e *batch system* installato sul *CE*, con relative code;
- gli indirizzi di eventuali risorse esterne al sito (ad esempio l'indirizzo dell' *RB* se non previsto nel sito);

Il file *StorageElement-local-cfg.h*:

E' anche questo contenuto all'interno della directory `git-2.2.0/` e ci consente di poter dichiarare in maniera flessibile, configurazioni specifiche per lo *Storage Element* sovrascrivendo eventuali variabili già definite nel file *site.cfg*. Un tipico esempio è il modulo del *kernel* da caricare per la scheda di rete, se questa risultasse essere diversa da quella dichiarata nel *site.cfg*.

```
+hardware.modlist eth1 eth0
+hardware.mod_eth0 alias eth0 via-rhine
+hardware.mod_eth1 alias eth1 8139too
```

Il file *ComputingElement-local-cfg.h*:

Questo file ha lo stesso scopo del precedente, l'unica differenza sta nel fatto che riguarda il *Computing Element* e non lo *Storage Element*.

Il file `UserInterface-local.cfg.h`:

In questo file, oltre a svolgere le funzioni dei precedenti, ci consente di definire gli utenti per la *User Interface*. Esempio:

```
EXTRA(auth.users) utentel
EXTRA(auth.groups) utentel
auth.userpwd_utentel NVOAdvCJ.xbRI (PWD criptata)
auth.useruid_utentel 501
auth.groupgid_utentel 501
auth.usergroup_utentel utentel
auth.usercomment_utentel Nome Cognome
```

Il file `WorkerNode-local-cfg.h`:

Funzione identica ai precedenti, riguardante i *Worker Node*.

Il file `Common-local-cfg.h`:

All'interno di questo file, vengono definite le impostazioni comuni a tutti i nodi, ad esempio la password di *root* per i nodi del sito Grid.

3.3.2 I profili XML

I profili XML sono gestiti dal server LCFGng. Questi vengono generati tramite il comando `mkxprox` che accetta come parametro in ingresso il nome del file di configurazione del nodo contenuto nella *directory* `/var/obj/conf/server/source/`. Il nome del file di configurazione deve necessariamente essere uguale al nome dell'*host* interessato all'installazione. Una volta generato il profilo XML, viene pubblicato nella directory pubblica di *apache* `/var/obj/conf/server/web/`.

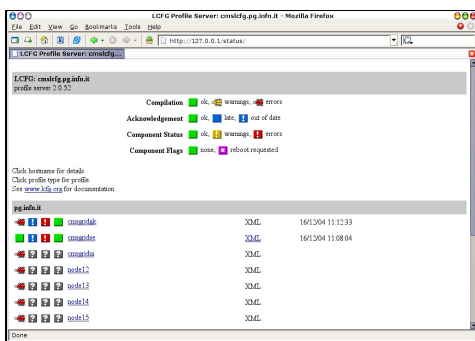


Figura 13 - Home page del web server su LCFGng

La struttura *XML* è la seguente:

```
<?xml version="1.0"?>
<profile
...
  <components>
    <perlex>
      ...
        <server>
          LCFGngserver.com
        </server>
      ...
    </perlex>
  </components>
...
</profile>
```

La struttura del file indica tutte le caratteristiche del nodo (*kernel*, versione di *INFN Grid*, directory montate via *NFS*, moduli caricati ecc..) e l'indirizzo del server *LCFGng*.

Quando un profilo viene cambiato, il server notifica il cambiamento al nodo interessato e via *HTTP* tramite il server web, l'aggiornamento è automatico. In alternativa si può forzare l'aggiornamento sia da lato client che server tramite un comando (`rdxprof -u http://lcfg.domain/profiles`).

3.4 La nuova release LCG 2.3.0

L'evoluzione di *INFN Grid* è legata allo sviluppo di *LCG2*. Recentemente è uscita la nuova versione 2.3.0 che ha piccole differenze con la precedente:

- *openPBS* è stato sostituito da *Torque* e *Maui*
- E' stata inserita un'altra coda nel *batch system*: la coda *cert* per le certificazioni
- E' stato introdotto il supporto per il tipo di *job DAG* [55]
- Sono state aggiunte altre *VO* (biomed, magic, esr, planck)
- *VO* aggiuntive possono essere aggiunte dinamicamente tramite uno script *python* [56].
- Via *LCFGng* è possibile configurare per diverse *VO*, diverse code del *batch system*

Ovviamente *INFN Grid* ha seguito la stessa evoluzione.

3.5 Cenni su Quattor

Nella versione *LCG2.3.0*, il *CERN* ha deciso di non supportare più il server *LCFGng*. Al suo posto è stato adottato *Quattor*.

Quattor è un insieme di strumenti di amministrazione, in grado di fornire tool potenti, modulari e portatili per la gestione o configurazione di *cluster* e *farm* in ambiente *Unix* e derivati (*Linux* e *Solaris*).

Le configurazioni dei nodi sono distinte in due categorie:

- La configurazione attuale del nodo, memorizzata sul Configuration Cache Memory (CCM) presente sul nodo stesso.
- La configurazione che si desidera avere su quel nodo, memorizzata all'interno del Configuration Database (CDB) che risiede sul server *Quattor*.

Uno speciale linguaggio di programmazione (*PAN*) è adibito come validatore dei profili di ciascun nodo.

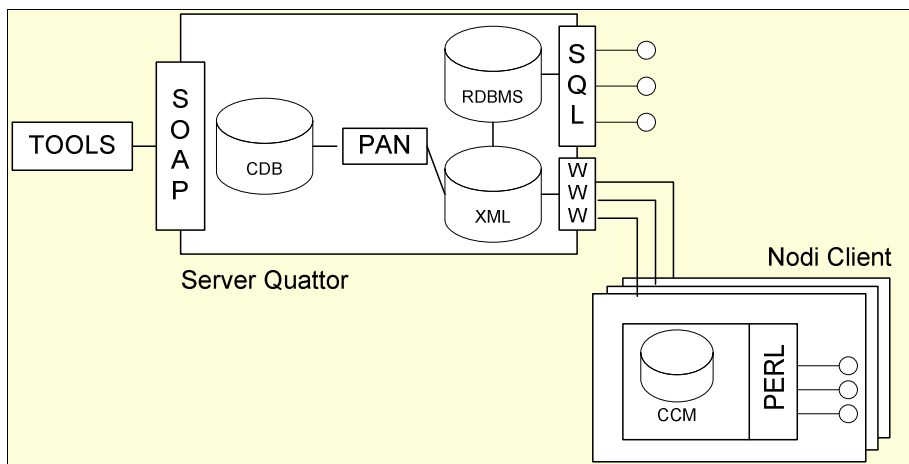


Figura 14 - Struttura di Quattor

Attualmente *Quattor* supporta queste distribuzioni *Linux*:

- *Red Hat 7.3*
- *Fedora (IA64,i386)*
- *Scientific Linux (IA64,i386)*
- *Solaris 9*

Quattor non è supportato ufficialmente da *INFN Grid*.

L'installazione del nuovo *middleware INFN Grid* avverrà con strumenti di installazione manuali (*apt-get* e *yaim*) per siti con supporto a *Scientific Linux*,

mentre rimarrà *LCFGng* per siti con *Red Hat 7.3*. Viene data inoltre, la possibilità a quei siti con un gran numero di nodi (*TIER 1*) di poter utilizzare *Quattor*, dato che il suo uso è consigliato per quel tipo di situazioni.

3.6 Il ruolo delle VO

Le *Virtual Organizations* rivestono un ruolo fondamentale all'interno di *INFN Grid*, e più in generale in *LCG2*: sono loro che con i loro programmi di analisi utilizzano Grid.

Le *VO* possono essere definite come un insieme dinamico di utenti e regole in grado di condividere risorse facenti parte ad una o più organizzazioni fisiche.

Generalmente, un utente che vuole utilizzare *INFN Grid* deve necessariamente appartenere ad una *VO* ed avere un certificato valido.

3.6.1 Ottenere un certificato

Per poter accedere a *Grid*, si deve avere un certificato personale rilasciato dalla *Certification Authority (CA)* dell'*INFN*. Il certificato dovrà essere poi installato all'interno dell'*User Interface*, dove l'utente possiede un account. Per ottenere un certificato occorre seguire diversi passi:

- Installare da <https://security.fi.infn.it/CA/mgt/getCA.php>, il certificato dell'*INFN* sul proprio *browser*
- Identificarsi all'interno della *Certification Authority* del proprio dipartimento. Alla fine di tale operazione verrà fornito un *ID*.
- Richiedere il certificato personale usando il proprio *ID* presso l'*INFN CA*. Si riceverà poi una mail contenente l'indirizzo dove scaricare il proprio certificato sul *browser*.
- Esportare il proprio certificato dal *browser*.
- Copiare il certificato ottenuto, sulla *UI*.

Una volta realizzato ciò, occorre convertire il certificato nel formato *PEM* [57] e copiarlo all'interno della *directory home* dell'utente sulla *UI .globus/*. La conversione avviene digitando questi semplici comandi:

```
openssl pkcs12 -nocerts -in mycert.p12 -out  
~/globus/userkey.pem  
openssl pkcs12 -clcerts -nokeys -in mycert.p12 -out  
~/globus/usercert.pem  
chmod 0400 ~/globus/userkey.pem  
chmod 0600 ~/globus/usercert.pem
```

Dove mycert.p12 è il certificato esportato.

CAPITOLO 4 Il sito di sviluppo Grid-Dev

4.1 Situazione attuale di Grid a Perugia

La sezione *INFN* di Perugia nel 2004 ha deciso di contribuire allo sviluppo di *INFN Grid*, installando un proprio sito, per permettere alla *VO CMS* locale di poter usufruire delle potenzialità computazionali che la griglia mette a disposizione.

La configurazione originale di un sito *INFN Grid* prevede la presenza di più macchine destinate alle funzioni computazionali: i *Worker Node*. Tali macchine, nelle vecchie versioni, dovevano avere tutte un indirizzo *IP* pubblico, necessario per la loro gestione. Il gruppo *CMS* di Perugia non poteva accettare una situazione simile, dato che non era in grado di assegnare molti indirizzi *IP* a tale progetto. Si è pensato quindi di modificare il *middleware*, rendendo possibile l'installazione di questi nodi all'interno di una rete nascosta o privata. Tale modifica è stata accettata dal *Central Management Team* (organo preposto alla gestione globale *INFN Grid*) ed attualmente a Perugia la si adotta.

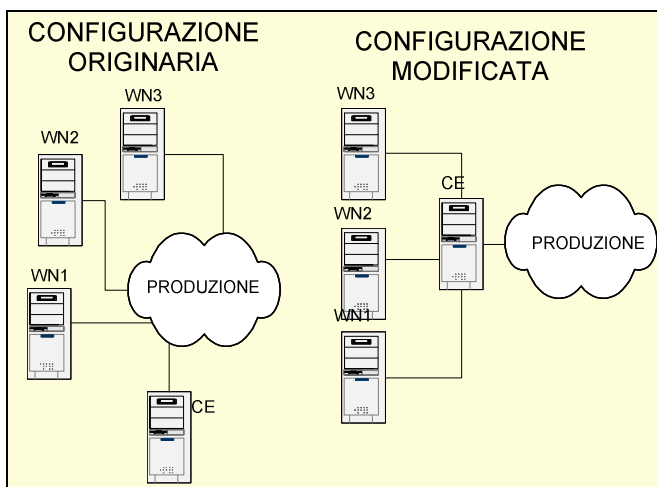


Figura 15 - Sito normale e uno modificato con rete privata

4.2 Lo scopo del sito di sviluppo

I motivi che hanno portato alla creazione di un sito *INFN Grid* di sviluppo sono molteplici e di varia natura. La presenza di un sito del genere, può rendere più semplice le fasi di aggiornamento di un sito di produzione, minimizzandone i tempi di “*downtime*”. Infatti, gli aggiornamenti *software* (ad esempio la versione nuova del *middleware*) prevedono un periodo di sospensione del servizio sul sito locale, che può essere di breve o medio/lungo termine (a seconda degli eventuali problemi di compatibilità del software). Tale periodo è dato all’amministratore del sito per poter effettuare concretamente l’aggiornamento e correggere eventuali anomalie. La presenza del sito di sviluppo eliminerebbe quasi totalmente il “*downtime*” dovuto alle potenziali incompatibilità di *software*, essendo questo stesso sito quasi identico a quello di produzione (i test verrebbero realizzati in questa sede) e di fatto l’intera “nuova” configurazione sarebbe semplicemente copiata su di esso. Inoltre potrebbero essere testati servizi di diversa natura, quale *NFS*: servizi che hanno dato problemi in *INFN Grid* e che sono il motivo principale che ci ha spinto alla realizzazione del sito di test.

4.3 Differenze con il sito di produzione

Il sito di sviluppo è una copia esatta di quello di produzione dal punto di vista della configurazione della rete, è infatti composto da due sottoreti (*pg.infn.it* e *cmsgrid.pg.infn.it*) interfacciate con il mondo esterno tramite un *firewall/gateway*, in particolare:

- Si ha la rete *pg.infn.it*, dove sono collegati i nodi *LCFG*, *cmsgridk*, *cmsgridse*, con *IP* dei nodi uguali a quelli reali
- La rete privata *cmsgrid.pg.infn.it*, dove eventualmente si potrebbero agganciare i *Worker Node* del sito.

Tutti i nodi quindi, sono equipaggiati con due schede di rete.

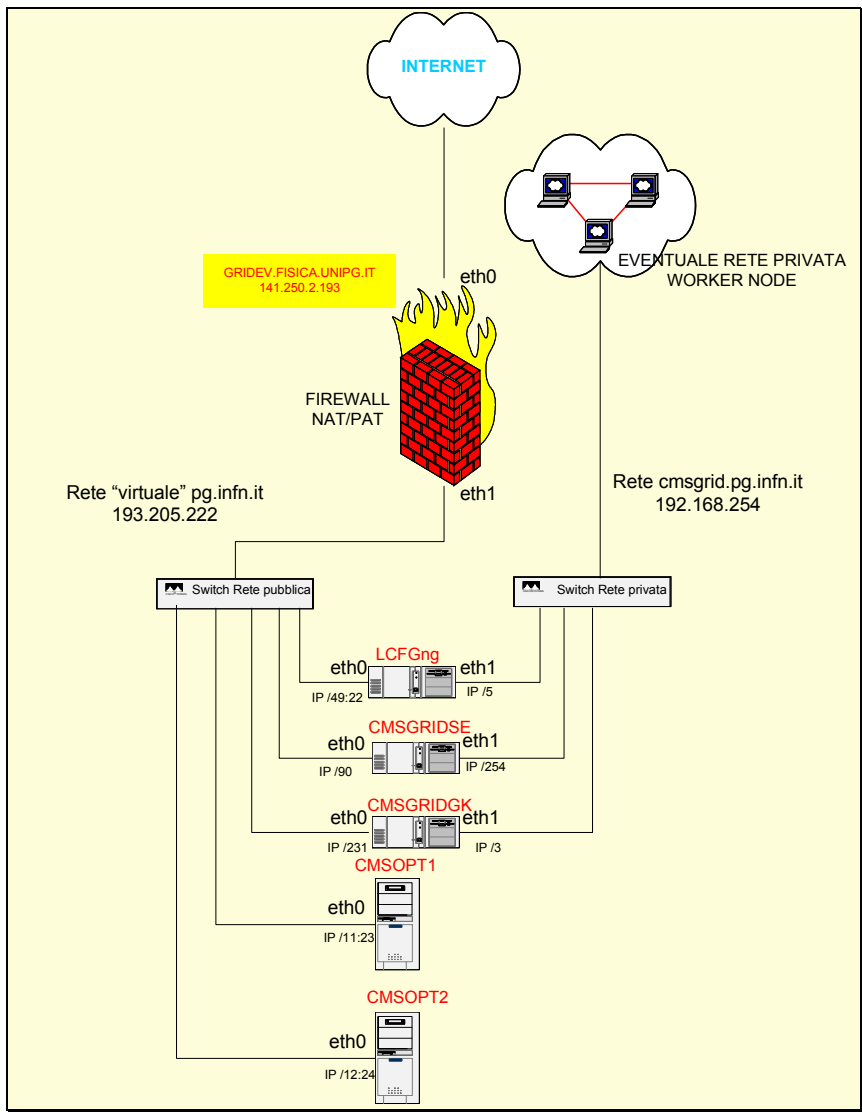


Figura 16 - Struttura di Grid-Dev

L'hardware utilizzato viene riassunto in questa tabella:

Hw Nome	Sistema / Processore	RAM	Hard Disk	Eth0	Eth1
Lcfg	Intel PIII 800 MHz	500 MByte	IDE 100 GByte	VIA RIINE 2 10/100 Mbit	VIA RIINE 2 10/100 Mbit
cmsgridgk	Intel PIV 2,5 GHz	500 Mbyte	IDE 80 GByte	VIA RIINE 2 10/100 Mbit	VIA RIINE 2 10/100 Mbit
cmsgridse	Intel PIV 3 GHz	1 GByte	IDE 400 GByte	VIA RIINE 2 10/100 Mbit	VIA RIINE 2 10/100

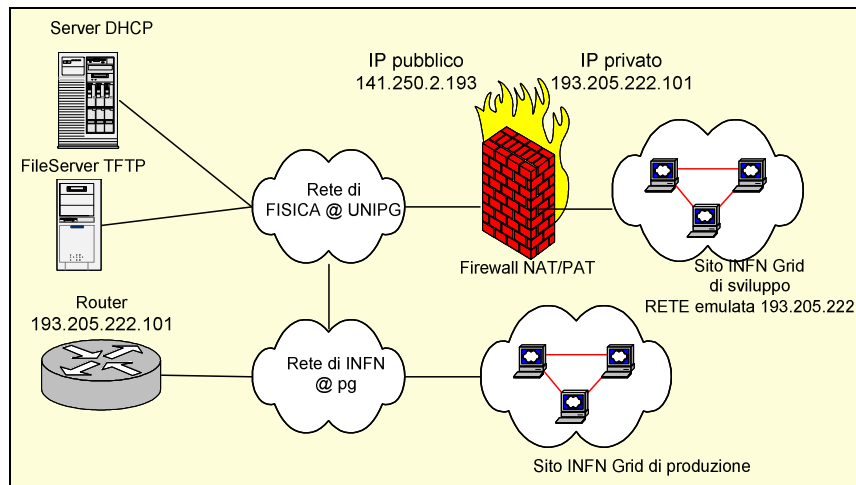
Griddev	VIA EPIA ITX 1 GHz	500 MByte	Diskless	VIA RIINE 2 10/100 Mbit	Realteck 10/10 Mbit
Cmsopt1	2 AMD Opteron 64bit 2 GHz	4 GByte	SCSI 30 GByte	VIA RIINE 2 10/100 Mbit	Realteck 10/100 Mbit
Cmsopt2	2 AMD Opteron 64bit 2 GHz	4 GByte	SCSI 30 GByte	VIA RIINE 2 10/100 Mbit	Realteck 10/100 Mbit
SWITCH	2 Digicom 8 porte 10/100				

Le differenze sostanziali con il sito reale sono poche e comprendono la non possibilità di poter sottomettere *job*, non avendo un certificato valido, e non essendo quindi mappato dal *Resource Broker* e le differenze con le componenti *hardware* (diverso tipo di processore, schede di rete ecc...).

4.4 Configurazione del firewall/gateway

La realizzazione di una copia del sito di produzione, è stata resa possibile grazie al nodo che svolge il ruolo di *Gateway/firewall* e che ci consente di interfacciarsi con il mondo esterno. Questa macchina quindi, dovrà essere fornita di due interfacce di rete: una collegata alla nostra sottorete di sviluppo, l'altra collegata alla rete di Fisica dell'Università di Perugia.

Una possibile soluzione per implementare questo sistema è quella di installare un sistema *Linux* completo (con servizi di rete attivi), eseguire il *NAT (Network Address Translation)* [58] della sottorete privata e proteggere la stessa rete con un *firewall* [59]. Essendo pochi i servizi attivi che sono richiesti, la soluzione che si è adottato nella realtà è l'utilizzo di una macchina *diskless* (senza disco), che consente una manutenzione del sistema più agevole e meno dispendiosa in termini di tempo. Il sistema operativo base (*Linux*) andrà interamente caricato in memoria principale all'avvio della macchina, eseguito in *run-time* con le regole del *firewall*, senza accessi interattivi. La sua gestione non avverrà quindi da remoto tramite *SSH (Secure SHell)* [2] ma direttamente dalla macchina.


Figura 17 - vista globale di Grid-Dev

La realizzazione di questo tipo di infrastruttura, ha previsto la configurazione del server *DHCP* (Dynamic Host Configuration Protocol) [60], del server *TFTP* (Trivial File Transfer Protocol) [61]; della configurazione e la compilazione del *kernel linux* (montato poi via rete sul nostro nodo) e la configurazione dell'*initrd* con lo script per il firewall.

Configurazione del server DHCP:

Il server *DHCP* che si è utilizzato è quello in uso nella rete del dipartimento di Fisica dell'Università di Perugia. E' necessario il suo utilizzo per gestire la macchina ad ogni sua accensione o riavvio, dal momento che abbiamo a che fare con un dispositivo senza memoria di massa. L'indirizzo *IP* assegnato è il 141.250.2.193 e questo andrà registrato sul file di configurazione del server *DHCP* (`/etc/dhcpd.conf`).

```
host gridev {
    hardware ethernet 00:02:b3:46:f3:fe;
    fixed-address 141.250.2.193;
    option host-name "gridev";
    next-server fileserv.fisica.unipg.it;
    if substring (option vendor-class-identifier, 0, 9) =
    "PXEClient" {
        filename "pxelinux.0";
    }
}
```

Spiegando le direttive imposte da questo file si può dire che:

- l'indirizzo *IP* 141.250.2.193 viene assegnato alla macchina che ha scheda di rete con *MAC address* 00:02:B3:46:F3:FE (il nostro nodo);
- a questo indirizzo viene associato il nome *gridev* (*gridev.fisica.unipg.it*);
- viene indicato l'indirizzo del server *TFTP* con i file di configurazione da cui scaricare il *kernel* del nodo (*fileserv.fisica.unipg.it*).

La configurazione del server TFTP:

Il *TFTP* è una particolare protocollo, definito nella *RFC 1350*, che consente di trasferire file utilizzando il protocollo *UDP* sulla porta 69. E' molto simile al protocollo *FTP* (*File Transfer Protocol*) ma ha molte limitazioni: non può leggere il contenuto delle directory ed è sprovvisto di autenticazione.

La configurazione del server *TFTP* avviene inserendo all'interno del file *xinetd* [62] del server (*/etc/xinetd.d*) le seguenti direttive:

```
service tftp
{
    disable = no
    socket_type          = dgram
    protocol            = udp
    wait                = yes
    user                 = root
    server               = /usr/sbin/in.tftpd
    server_args          = -s /tftpboot
}
```

All'interno della *directory* principale (*/tftpboot/*), devono essere create queste cartelle con i relativi file che, come vedremo, serviranno al client per poter caricare il sistema operativo:

```
/tftpboot/initrds/gridev-2.6.4
/tftpboot/kernels/gridev-2.6.4
/tftpboot/pxelinux.0
/tftpboot/pxelinux.cfg/8DFA0261
```

Nella *directory* */tftpboot/initrds/gridev-2.6.4* è contenuto l'*initrd* del nostro sistema mentre nella *directory* */tftpboot/kernels/gridev-2.6.4* è contenuto il *kernel*.

All'interno delle directory `/pxelinux.0` e `/tftpbboot/pxelinux.cfg/8DFA0261` sono presenti i file di configurazione di *PXELINUX* (Pre-eXecutions Environment LINUX).

La configurazione del Kernel:

Il *kernel* che si è deciso di utilizzare ha versione 2.6.4. La configurazione è avvenuta con i consueti strumenti (`make menuconfig` ecc...). Il tipo di *kernel* configurato è di tipo monolitico per una più facile gestione. I servizi resi attivi sono quelli di tipo rete e quelli per la gestione del *RAM* (Random Access Memory) *disk*. Una volta ottenuta la sua immagine (*bzimage*), è stata copiata nella directory `/tftpbboot/kernels/gridev-2.6.4` del server *TFTP*.

La configurazione del file *initrd*:

Per i nostri scopi oltre al *kernel*, abbiamo bisogno del file *initrd*. L'*initrd* può essere definito come un disco *RAM* che viene inizialmente montato come un *filesystem* dal *kernel Linux* allo scopo di avviare un sistema minimo per poter eseguire delle funzioni preliminari. Normalmente, al termine di queste operazioni l'*initrd* monta il *filesystem* standard ed il controllo passa al demone *init*.

```
/bin
/bin/bash
/bin/cat
/bin/echo
/bin/grep
/bin/mount
/bin/sh
/bin/umount
/bin/infinite
/bin/rm
/bin/tftp
/bin/chmod
/dev
/dev/console
/etc
/etc/fstab
/etc/mtab
/lib
/lib/ld-2.3.2.so
/lib/ld-linux.so.2
/lib/libc-2.3.2.so
/lib/libc.so.6
/lib/libdl-2.3.2.so
/lib/libdl.so.2
/lib/libpthread-0.10.so
/lib/libpthread.so.0
/lib/libtermcap.so.2
/lib/libtermcap.so.2.0.8
/lib/iptables
/lib/iptables/libipt_MASQUERADE.so
/lib/iptables/libipt_DNAT.so
/lib/iptables/libipt_DSCP.so
/lib/iptables/libipt_ECN.so
/lib/iptables/libipt_LOG.so
/lib/iptables/libipt_MARK.so
```

```

/lib/iptables/libipt_MIRROR.so
/lib/iptables/libipt_REDIRECT.so
/lib/iptables/libipt_REJECT.so
/lib/iptables/libipt_SAME.so
/lib/iptables/libipt_SNAT.so
/lib/iptables/libipt_TARPIT.so
/lib/iptables/libipt_TCPMSS.so
/lib/iptables/libipt_TOS.so
/lib/iptables/libipt_TTL.so
/lib/iptables/libipt_ULOG.so
/lib/iptables/libipt_ah.so
/lib/iptables/libipt_contrack.so
/lib/iptables/libipt_dscp.so
/lib/iptables/libipt_ecn.so
/lib/iptables/libipt_esp.so
/lib/iptables/libipt_helper.so
/lib/iptables/libipt_icmp.so
/lib/iptables/libipt_iplimit.so
/lib/iptables/libipt_length.so
/lib/iptables/libipt_limit.so
/lib/iptables/libipt_mac.so
/lib/iptables/libipt_mark.so
/lib/iptables/libipt_multiport.so
/lib/iptables/libipt_owner.so
/lib/iptables/libipt_physdev.so
/lib/iptables/libipt_pkttype.so
/lib/iptables/libipt_recent.so
/lib/iptables/libipt_rpc.so
/lib/iptables/libipt_standard.so
/lib/iptables/libipt_state.so
/lib/iptables/libipt_tcp.so
/lib/iptables/libipt_tcpmss.so
/lib/iptables/libipt_tos.so
/lib/iptables/libipt_ttl.so
/lib/iptables/libipt_udp.so
/lib/iptables/libipt_unclean.so
/lib/libreadline.so
/lib/libreadline.so.4
/lib/libreadline.so.4.3
/linuxrc
/proc
/sbin
/sbin/ifconfig
/sbin/route
/sbin/iptables
/sbin/servercontrol
/scripts
/scripts/reload-chains
/tmp
/var
/var/lock
/var/log

```

Come già detto precedentemente, la gestione del *RAM disk*, deve essere abilitata durante la configurazione del *kernel*. Nel nostro specifico caso al termine del caricamento *dell'initrd*, invece di caricare il *filesystem* vero e proprio (cosa che noi non abbiamo e che non serve) viene caricato il file *linuxrc*, in cui abbiamo inserito la configurazione del *firewall*.

La configurazione di PXELINUX:

PXELINUX è un programma derivato da *SYSLINUX* [63] per l'avvio di macchine equipaggiate con schede di rete conformi allo standard Intel *PXE*. In pratica è un

bootloader che consente di caricare il sistema operativo dalla rete. La scheda madre del nodo interessato dovrà supportare questo tipo di avvio della macchina (*boot on LAN*).

La configurazione standard prevede la presenza all'interno della directory `/tftpboot/` del server *TFTP*, del file `pxelinux.0` e della directory `pxelinux.cfg/`. Il file `pxelinux.0` consente di fare il *boot* della macchina mentre la directory `pxelinux.cfg/` deve contenere il file di configurazione del nostro nodo. Dal momento che possiamo avere più *host* che leggono il file di configurazione dallo stesso server *TFTP*, ai file di configurazione deve essere assegnato un nome che dipende dall'indirizzo *IP* oppure dal *MAC* address del richiedente (in modo tale da distinguere le configurazioni): tale nome è lo stesso indirizzo convertito in esadecimale. Ad esempio per l'*host* con indirizzo *MAC* `88:99:AA:BB:CC:DD` si dovrà creare un file con nome `01-88-99-AA-BB-CC-DD`, oppure per l'*host* con indirizzo *IP* `192.0.2.91` si dovrà avere un file di configurazione con nome `C00002B`. Nel nostro caso il contenuto del file è il seguente:

```
default linux
label linux
    kernel kernels/gridev-2.6.4
    append initrd=initrds/gridev-2.6.4 root=/dev/ram0 init=/linuxrc
    rw
```

all'interno del quale viene indicata l'immagine del *kernel* con i relativi parametri da caricare (*initrd*, *linuxrc* ecc..)

Nella figura seguente è possibile vedere l'iter che il *gateway* configurato esegue una volta acceso.

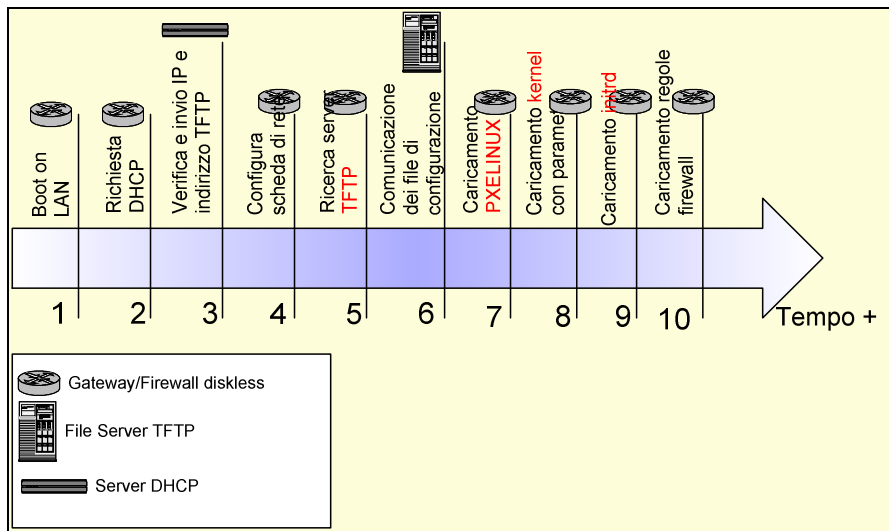


Figura 18 - Linea temporale di caricamento del Kernel

La configurazione del firewall: Le regole

Il *firewall* che si è deciso di utilizzare è *iptables* [64], incluso nel *kernel* precedentemente configurato. Il *firewall* è stato utilizzato anche per implementare il servizio di *NAT/PAT* (*Port Address Translation*). Il servizio *NAT* (*RFC 1631*) è un meccanismo grazie al quale è possibile modificare gli indirizzi *IP*, generalmente con lo scopo di consentire ad una rete privata di accedere all'esterno: gli indirizzi *IP* privati vengono cambiati dinamicamente con indirizzi *IP* pubblici a disposizione. Il servizio *PAT* [65] è un servizio analogo al precedente: in questo caso la sostituzione avviene per le porte *TCP* e *UDP*. Nella maggior parte dei casi quest'ultimo viene utilizzato in reti private con un solo *IP* pubblico a disposizione per poter accedere all'esterno (come nel nostro caso).

Lo script che include le regole del *firewall* è stato inserito all'interno di un file caricabile da web con *wget*:

```
#!/bin/bash
ifconfig eth1 193.205.222.101 netmask 255.255.255.0
iptables -t nat -A POSTROUTING -s 193.205.222.0/24 -d !
193.205.222.0/24 -j SNAT --to-source 141.250.2.193
iptables -t nat -A PREROUTING -p tcp -d 141.250.2.193 --dport 22 -
j DNAT --to-destination 193.205.222.49
iptables -t nat -A PREROUTING -p tcp -d 141.250.2.193 --dport 23 -
j DNAT --to-destination 193.205.222.11:22
iptables -t nat -A PREROUTING -p tcp -d 141.250.2.193 --dport 24 -
j DNAT --to-destination 193.205.222.12:22
echo 1 > /proc/sys/net/ipv4/ip_forward
```

in cui:

- viene configurata la seconda interfaccia del nodo (*IP, netmask ecc.*),
- Vengono redirette tutte le connessioni ricevute da *gridev* sulla porta 22 verso l'host 193.205.222.49 (server *LCFG*)
- Vengono redirette le connessioni sulla porta 23 verso 193.205.222.11 (*cmsopt1*)
- Vengono redirette tutte le connessioni verso la porta 24 verso 193.205.222.12 (*cmsopt2*)
- Viene abilitato l'*IP forward* del nodo, in modo da consentire ai client di poter accedere all'esterno.

Con questo *script* è stato possibile “emulare” la rete *INFN* di Perugia senza causare nessun tipo di problema (conflitto di indirizzi *IP* tra nodi) perché il tutto è mascherato dietro il nostro *gateway/firewall*. Ovviamente all'interno della nostra rete di test, è previsto un server *DNS (Domain Name Server)* [66] in esecuzione sul server *LCFG* per poter gestire i nomi e gli indirizzi delle due sotto-reti.

4.5 La configurazione del Server *LCFGng*

Per l'installazione e la configurazione del server *LCFG* si sono seguite le procedure standard, come descritto nel paragrafo 3.3.

Il sito *INFN Grid* di produzione a Perugia, come già accennato, prevede la gestione dei *Worker Node* all'interno di una rete nascosta. Per realizzare ciò, il server è stato dotato di due schede di rete: la prima (*eth0*) gestisce la configurazione dei nodi collegati alla rete pubblica (*CE, SE, UI*), mentre la seconda (*eth1*) gestisce la configurazione di tutti i nodi (*Worker Node, CE, SE, UI*) collegati alla rete privata. Partendo dal presupposto che la prima scheda di rete è stata configurata durante l'installazione del server *LCFG*, i parametri che sono stati assegnati alla seconda scheda di rete sono: (basta modificare il file `/etc/sysconfig/network-scripts/ifcfg-eth1`)

```
DEVICE=eth1
BOOTPROTO=static
BROADCAST=192.168.254.255
IPADDR=192.168.254.5
NETMASK=255.255.255.0
NETWORK=192.168.254.0
ONBOOT=yes
```

4.6 La configurazione del DNS

Per la gestione dei nomi dei singoli nodi, si è configurato un server *DNS* in grado di gestire la rete “emulata” *INFN* di Perugia (193.205.222.0) e la rete *cmsgrid.pg.infn.it* (192.168.254.0). Tale server è in esecuzione su *LCFG*. Il server che si è utilizzato è *BIND* 9 [67]. La sua configurazione ha riguardato il file principale *named.conf* (*/etc/named.conf*), e i file delle zone singole collocati in */var/named*. All’interno del file *named.conf* viene definita la locazione della directory principale del server (*/var/named/*) ed il nome dei singoli *file zone* gestiti dal *DNS*, in particolare:

```
options {
    directory "/var/named";
};
controls {
    inet 127.0.0.1 allow { localhost; }
    keys { rndckey; };
};
zone "." IN {
    type hint;
    file "named.ca";
};
zone "localhost" IN {
    type master;
    file "localhost.zone";
    allow-update { none; };
};
zone "0.0.127.in-addr.arpa" IN {
    type master;
    file "named.local";
    allow-update { none; };
};
```

Il file è stato suddiviso in due parti per comodità. In questa prima parte viene definita la fonte per la risoluzione dei nomi del dominio principale contraddistinto dal punto (ad esempio *leorsini.it* ecc...), la risoluzione diretta (da nome di dominio a *IP*) della rete dell’elaboratore locale (*localhost*) e la risoluzione inversa (da *IP* a nome di dominio) della rete dell’elaboratore locale (*0.0.127.in-addr.arpa*).

```

/***/ configurazione di cmsgrid *****/
zone "cmsgrid.pg.infn.it" IN {
    type master;
    file "cmsgrid.zone";
    allow-update { none; };
};
zone "254.168.192.in-addr.arpa" IN {
    type master;
    file "cmsgrid.rev";
    allow-update { none; };
};
//Emulazione rete pg.infn.it
zone "222.205.193.in-addr.arpa" IN {
    type master;
    file "pg-infn.rev";
    allow-update {none;};
};

zone "pg.infn.it" IN {
    type master;
    file "pg-infn.zone";
    allow-update {none;};
};
include "/etc/rndc.key";

```

In questa seconda parte del file, vengono definite i percorsi delle zone (in pratica dove risiedono effettivamente i file sul server) per la rete `pg.infn.it` e `cmsgrid.pg.infn.it` sia per la risoluzione diretta che inversa dei nomi di dominio.

Come detto in precedenza, i *file zone* sono collocate all'interno della directory `/var/named` e sono rispettivamente:

- `named.ca`: contenente le informazioni dei *root file server*. E' utilizzato per inizializzare la *cache* del server *DNS*. E' un file predefinito scaricabile dalla rete.
- `named.local`: contengono le informazioni per la risoluzione diretta dei nomi per la rete dell'elaboratore locale.

```

$TTL      86400
@         IN      SOA     localhost. root.localhost. (
                                1997022700 ; Serial
                                28800      ; Refresh
                                14400      ; Retry
                                3600000    ; Expire
                                86400     ) ; Minimum
         IN      NS      localhost.
1         IN      PTR     localhost.

```

- *localhost.zone*: contenente le informazioni per la risoluzione inversa dei nomi per la rete dell'elaboratore locale

```

$ORIGIN localhost.
@ 1D IN SOA @ root (
    42      ;serial (d. adams)
    3H     ;refresh
    15M    ;retry
    1W     ;expiry
    1D )   ;minimum
1D IN NS  @
1D IN A   127.0.0.1
    
```

- *pg-infn.rev*: contenente le informazioni per la risoluzione inversa dei nomi degli elaboratori facenti parte della rete 193.205.222.0 (pg.infn.it)

```

$TTL 3600
@ IN SOA cmslcfg.pg.infn.it. root.cmslcfg.pg.infn.it. (
    1998031800;Serial
    28800;Refresh
    7200;Retry
    604800;Expire
    86400);Minimum
49.222.205.193.in-addr.arpa. NS    cmslcfg.pg.infn.it.
231.222.205.193.in-addr.arpa. PTR   cmslcfg.pg.infn.it.
90.222.205.193.in-addr.arpa. PTR   cmsgridgk.pg.infn.it.
165.222.205.193.in-addr.arpa. PTR   cmsgridse.pg.infn.it.
                                PTR   cmsgridui.pg.infn.it.
    
```

- *pg-infn.zone*: contenente le informazioni per la risoluzione diretta dei nomi degli elaboratori facenti parte della rete pg.infn.it

```

$TTL 3600
@ IN SOA cmslcfg.pg.infn.it. root.cmslcfg.pg.infn.it. (
    1998031800;Serial
    28800;Refresh
    7200;Retry
    604800;Expire
    86400);Minimum
cmslcfg.pg.infn.it. NS    cmslcfg.pg.infn.it.
cmslcfg.pg.infn.it. A     193.205.222.49
cmsgridse.pg.infn.it. A   193.205.222.90
cmsgridgk.pg.infn.it. A   193.205.222.231
cmsgridui.pg.infn.it. A   193.205.222.165
    
```

- *cmsgrid.rev*: contenente le informazioni per la risoluzione inversa dei nomi degli elaboratori facenti parte della rete 192.168.254.0

(cmsgrid.pg.infn.it)

```
$TTL 3600
@ IN SOA cmslcfg.cmsgrid.pg.infn.it. grid.pg.infn.it. (
    2101200403 ; Serial
    3600      ; Refresh every hour
    900      ; Retry every 15
minutes
    3600000 ; Expire 1000 hours
    3600 ) ; Minimum 1 hour

    IN      NS      cmslcfg.cmsgrid.pg.infn.it.
1 IN      PTR     gwnode.cmsgrid.pg.infn.it.
2 IN      PTR     cmslx2.cmsgrid.pg.infn.it.
..
..
```

- *cmsgrid.zone*: contenente le informazioni per la risoluzione diretta dei nomi degli elaboratori facenti parte della rete cmsgrid.pg.infn.it:

```
$TTL 3600
@ IN SOA cmslcfg.cmsgrid.pg.infn.it. grid.pg.infn.it. (
    2101200403 ; Serial
    3600      ; Refresh every hour
    900      ; Retry every 15 minutes
    3600000 ; Expire 1000 hours
    3600 ) ; Minimum 1 hour

cmsgrid.pg.infn.it. IN NS cmslcfg.cmsgrid.pg.infn.it.

gwnode                IN A 192.168.254.1
cmslx2                IN A 192.168.254.2
..
..
```

Inoltre all'interno del file *networks* (/etc/networks) sono state inserite le sotto-reti gestite dal server:

```
localdomain    127.0.0.0
pg.infn.it     193.205.222
cmsgrid.pg.infn.it 192.168.254.0
```

Infine, all'interno del file *resolv.conf* (/etc/resolv.conf) sono stati inseriti gli indirizzi dei server *DNS* interni ed esterni al sito locale:

```
bind,hosts
main pg.infn.it
search pg.infn.it
search cmsgrid.pg.infn.it

nameserver 193.205.222.49
nameserver 193.205.222.2
nameserver 141.250.2.7
```

4.7 Le modifiche ai file di configurazione: site.cfg

La configurazione del server *LCFG* ha riguardato anche le modifiche necessarie al file *site.cfg*, riguardanti:

- I parametri riguardanti tutti gli *host* del sito:
 - CE_HOSTNAME il nome del *Computing Element*;
 - SE_HOSTNAME il nome dello *Storage Element*;
 - SITE_LOCALDOMAIN il nome della rete locale;
 - SITE_MAILROOT la mail dell'amministratore del sito;
 - SITE_GATEWAYS il *gateway* del sito;
 - SITE_ALLOWED_NETWORKS le reti gestite la sito;
 - SITE_NAMESERVERS l'indirizzo del server *DNS* locale del sito;
 - SITE_NETMASK la *netmask*;
 - SITE_NETWORK la rete principale gestita;
 - SITE_BROADCAST il *broadcast* della rete;
 - SITE_WN_HOSTS il nome dei *Worker Node*;
- I parametri riguardanti la gestione dei *Worker Node* su rete privata:
 - SITE_PRIVATE_NETWORK utilizzo o meno della rete privata per i *WN*;
 - SITE_INT_LOCALDOMAIN nome della rete privata;
 - SITE_INT_GATEWAYS il *gateway*;
 - SITE_INT_NAMESERVERS l'indirizzo del *DNS*;
 - SITE_INT_NETMASK la *netmask* della rete;
 - SITE_INT_NETWORK la rete privata gestita;
 - SITE_INT_BROADCAST il *broadcast* della rete;
 - SITE_INT_LCFG_SERVER l'indirizzo del server *LCFG*;
 - SITE_INT_CE_HOSTNAME l'indirizzo privato del *CE*;

Il file StorageElement-local-cfg.h

All'interno di questo file, sono state inserite le personalizzazioni *hardware* per il nodo *cmsgridse* (lo *Storage Element*) che ovviamente sono diverse da quelle del sito di produzione. In particolare si sono indicati i driver delle due schede di rete:

```
+hardware.modlist      eth1 eth0
+hardware.mod_eth0     alias eth0 via-rhine
+hardware.mod_eth1     alias eth1 8139too
```

Il file `ComputingElement-local-cfg.h`

Come nel caso del precedente, all'interno di questo file si sono inserite le personalizzazioni hardware del nodo *cmsgridgk*, in particolare quelle riguardanti le due schede di rete:

```
+hardware.modlist    eth1 eth0
+hardware.mod_eth0   alias eth0 via-rhine
+hardware.mod_eth1   alias eth1 8139too
```

Il file `Common-local-cfg.h`

In questo file si è solo indicato la password di *root* per tutti i nodi:

```
+auth.rootpwd    $1$jQW4LrWI$bvIli7ZmeY6J6E4FFxEp/1
```

La password è inserita in maniera cifrata. Per cifrare la password si è utilizzato lo script `git-cryptpasswd`.

4.8 L'installazione degli altri nodi

L'installazione dei nodi è avvenuta subito dopo la configurazione del server *LCFG*. I nodi di un sito *INFN Grid* si possono installare in due diverse maniere: tramite un disco floppy di *boot* o tramite *PXE*. L'installazione che si è deciso di eseguire è quella tramite *PXE*, anche perché già in precedenza avevamo configurato un altro *host* con questo meccanismo (*Gateway/firewall*). Questa installazione prevede la configurazione del servizio *DHCP* sul server *LCFG*, per la prima fase di installazione:

```
deny unknown-clients;
not authoritative;
option dhcp-class-identifier "PXEClient";
option vendor-encapsulated-options 01:04:00:00:00:00:ff;
filename "pxelinux.0";

subnet 193.205.222.0 netmask 255.255.255.0 {

    option routers 193.205.222.101;
    option domain-name-servers 193.205.222.49, 192.168.254.5;

    host cmsgridgk {
        option user-class "http://cmslcfg.pg.infn.it/profiles";
        hardware ethernet 00:50:FC:EC:D3:7F;
        fixed-address 193.205.222.231;
    }
    host cmsgridse {
        option user-class "http://cmslcfg.pg.infn.it/profiles";
        hardware ethernet 00:50:FC:ED:85:A3;
        fixed-address 193.205.222.90;
    }
    host cmsgridui {
        option user-class "http://cmslcfg.pg.infn.it/profiles/";
        hardware ethernet 00:20:ED:11:0C:B8;
        fixed-address 193.205.222.165;
    }
}
}
```

All'interno di questo file, oltre i consueti parametri (assegnazione *IP* ai client, segnalazione *TFTP* server ecc...), troviamo una particolare classe che definisce l'indirizzo web del profilo di ciascun singolo nodo (`option user-class "http://..."`). Tale indirizzo è usato dai client *LCFG* per scaricare il proprio profilo *XML* di configurazione. Ovviamente il *DHCP* deve gestire oltre gli *host* appartenenti alla rete `pg.infn.it`, anche quelli apparenti alla rete `cmsgrid.pg.infn.it`, quindi:

```
subnet 192.168.254.0 netmask 255.255.255.0 {

    option routers 192.168.254.5;
    option domain-name-servers 192.168.254.5, 193.205.222.49;

    host node17 {
        option user-class "http://cmslcfg.cmsgrid.pg.infn.it/profiles/";
        hardware ethernet 00:20:ED:11:0D:71;
        fixed-address 192.168.254.17;
    }
    host nodeX {
        option user-class "http://cmslcfg.cmsgrid.pg.infn.it/profiles/";
        hardware ethernet XX:XX:XX:XX:XX:XX;
        fixed-address XXX:XXX:XXX:XXX;
    }
}
}
```

Per poter indicare ad un particolare nodo il tipo di installazione, esiste un apposito script *CGI* (Common Gateway Interface) [68] invocabile dallo spazio web di *LCFG* (<http://<YourLcfgngServer>/install/install.cgi>) oppure esiste uno *script bash* (*git-bootselect*) invocabile dalla *shell* dello stesso server. Tramite questi due script si possono scegliere:

- Installazione del nodo senza conferme interattive;
- Installazione del nodo con conferme interattive passo-passo;
- Avvio del nodo dal proprio *HD* (Hard Disk).

In particolare, quest'ultima opzione è da selezionare una volta terminata l'installazione, e ci consente di cambiare la modalità di avvio della macchina (in questo caso dall'hard disk): se ciò non avvenisse, al successivo riavvio si avrebbe una nuova installazione.

4.9 Problemi riscontrati con il kernel ufficiale

L'installazione standard di un generico sito *INFN Grid*, prevede la presenza su tutti i nodi del *kernel 2.4.27*. Purtroppo durante l'installazione del nodo *cmsgridse* (*Storage Element*), si sono riscontrati problemi di vario genere, evidentemente dovuti al fatto che la macchina aveva un tipo di *hardware* con *driver* non presenti in quella versione del *kernel*. Per ovviare a questi problemi e rendere attivo il nodo, si è provveduto alla configurazione ed alla compilazione di un *kernel* personalizzato e più recente, predisposto solo per quella macchina.

L'installazione del nuovo *kernel* è stata suddivisa in *step*, per rendere più agevole la lettura:

STEP1: Il *kernel* che abbiamo deciso di utilizzare ha versione 2.4.6. I sorgenti sono stati reperiti dal mirror <http://kernel.org> in formato *tarball* (*linux-2.4.33.tar.bz2*). Per configurarlo abbiamo utilizzato i consueti strumenti (*make menuconfig* ecc...) messi a disposizione dalla comunità *opensource*. Si è deciso inoltre, di compilare un *kernel* di tipo modulare.

La configurazione e la compilazione è stata fatta sul server *LCFG*, perché è da questo server che poi abbiamo fatto l'*upgrade* del nodo tramite il profilo *XML*. I sorgenti sono stati copiati all'interno della directory */usr/src/*. Una volta configurato e compilato il *kernel*, si è dovuto fare una copia del file *.config* (file di configurazione del *kernel*) e rinominarla in *.config_smp*. Tale copia serve per la

configurazione di sistemi con due o più processori (non è comunque il nostro caso).

STEP2: Una volta compilato il *kernel* ed ottenuti i file *.config* e *.config_smp*, abbiamo dovuto copiare il *tarball* del *kernel* che abbiamo precedentemente scaricato all'interno della directory `/usr/src/redhat/SOURCES` del server *LCFG*.

STEP3: All'interno della directory `/usr/src/redhat/SPECS` è stato collocato il file delle specifiche per il nostro *kernel*. Questo file servirà poi per la creazione del pacchetto *rpm*. Il nome assegnato è `2-4-6.spec` e dal sito web ufficiale di *INFN Grid* si è scaricato un *template* dove all'interno di esso sono state assegnati i valori di alcune variabili:

- `Version` impostata a 2.4.6, indica la versione del *kernel*
- `Release` il nome che vogliamo dare al *kernel* es: *mykernel*
- `Version2.4.27` variabile impostata a 1 per indicare che è un *kernel* della famiglia 2.4
- Le altre variabili presenti non occorre modificarle.

Il file delle specifiche è stato inserito nell'appendice B di questa tesi.

STEP4: Modificato il file `2-4-6.spec`, si è provveduto ad una nuova compilazione del *kernel* ed alla generazione del pacchetto *rpm* tramite lo *script* `rpmbuild -ba /usr/src/redhat/SPECS/2-4-6.spec`. Una volta terminata questa fase avremo:

- All'interno della directory `/usr/src/redhat/RPMS/i386/` i pacchetti binari (*rpm*) per macchine mono e bi-processore;
- All'interno della directory `/usr/src/redhat/SRPMS/i386/` i file sorgente del *kernel* personalizzato

STEP5: Per l'aggiornamento via *LCFG* del nodo *cmsgridse* si è dovuto copiare il pacchetto *rpm* ottenuto al passo precedente in `/usr/src/redhat/SRPMS/i386/` e lanciare lo *script* `git-genhdfiles` per creare gli *hdfiles* e risolvere eventuali dipendenze del pacchetto stesso.

STEP6: Ora abbiamo dovuto modificare il file di configurazione del nodo, in particolare il file `StorageElement-local-cfg.h`. All'interno di questo abbiamo indicato il *kernel* che il nodo deve caricare:

```
#define KERNEL_VERSION 2.4.27-mykernel
```

STEP7: Ora abbiamo tutto il necessario per generare i nuovi profili *XML* tramite `mkxprof nomenodo`. Una volta fatto ciò, per verificare se tutto è corretto, basta spostarsi all'interno della directory `/var/conf/server/web/profile/pg.infn.it/` e ricercare all'interno del profilo *XML* `cmsgridse.xml` i parametri di impostazione del nostro *kernel*, in particolare:

```
<initrd>(hd0,0)/initrd-2.4.27-mykernel .img</initrd>
<kernel>(hd0,0)/vmlinuz-2.4.27-mykernel</kernel>
```

ed inoltre:

```
<imethod cfg:name="mkinitrd">install run chroot /root mkinitrd
-f /boot/initrd-2.4.27-daniele .img 2.4.27-mykernel</imethod>
```

Il nodo ora è pronto per essere aggiornato con il nostro nuovo *kernel*.

4.10 Montaggio del file-server via NFS

Il *file-server* che era installato sulla vecchia *farm* di *CMS*, aveva dato problemi di vario genere quando fu montato via *NFS* (*Network File System*) [69] sulla griglia di produzione locale. Lo scopo del sito di sviluppo, principalmente era quello di riuscire a montare quel *file-server* sul sito di test, testarlo e poi eventualmente montarlo sul sito reale. Per fare ciò, l'installazione che si è fatta è avvenuta tramite il server *LCFG*, grazie al quale è stato possibile montare una cartella creata ad hoc del file-server, sul nodo *cmsgridse* (*SE*) del sito di test ed eseguire gli eventuali test.

Il montaggio di uno spazio disco su di un sito *INFN Grid*, può essere fatto in tre diversi modi:

- Il primo consiste nel montare a mano il *fileserv* via *NFS*
- Il secondo consiste nel modificare e nell'aggiornare il profilo *dell'SE* tramite *LCFG*
- Il terzo consiste nell'installare all'interno del sito un ulteriore *Storage Element*.

La soluzione che si è adottata è quella di ridefinire il profilo tramite *LCFG*. Infatti, una soluzione come quella di installare un ulteriore *Storage Element* avrebbe comportato un quantitativo di lavoro sproporzionato rispetto allo scopo, mentre una soluzione come quella di montare a mano il file-server via *NFS*, avrebbe comportato la perdita della configurazione ad ogni eventuale riavvio della macchina *cmsgridse* (*Storage Element*).

Per realizzare ciò, si è dovuta modificare leggermente la configurazione del file `StorageElement-local-cfg.h`, ricompilare i profili ed effettuare questi test.

4.10.1 Personalizzazione del file `StorageElement-local-cfg.h`

Oltre al file `site.cfg`, si è dovuto modificare il file `Storage-Element-local-cfg.h`. Le modifiche hanno riguardato l'aggiunta di queste variabili:

```
EXTRA(nfsmount.nfsmount) test
nfsmount.nfsdetails_test /export/data05 192.168.254.8:/data05 rw
```

In cui viene indicato il server *NFS* (192.168.254.8), la cartella da montare con i permessi (`/data05 rw`), e la cartella locale a *cmsgridse* su cui montare quella remota (`/export/data05`).

Avvenute queste modifiche, si è ri-generato il profilo (`mkxprof nomenodo`) e lo si è aggiornato sul nodo (`rdxprof`). Per effettuare una verifica sulla configurazione, basta controllare sul file *XML* del profilo se viene indicata correttamente la cartella da montare via *NFS*:

```
<nfsmount cfg:template="nfsdetails_$">
  <nfsdetails cfg:name="lcfg">/export/local/linux/7.3
  cmslcfg.pg.infn.it:/opt/local/linux/7.3
  ro,nfsvers=2,nolock
  </nfsdetails>
  <nfsdetails cfg:name="test">/export/data05
  192.168.254.8:/data05 rw
  </nfsdetails>
</nfsmount>
```

CAPITOLO 5 Testing sulla Grid di sviluppo

5.1 Il File Server

Il *File Server* acquistato e da montare via *NFS* sulla *farm* di produzione ha questi caratteristiche hardware:

Tipo di Hardware	Quantità	Modello
CPU	2	INTEL XEON 3 GHz
RAID	2	Controller 3Ware Raid 5 + disco spare
HD	16	Quantum fireball 250 GB SCSI
RAM	2 x 512 MB	

Lo scopo ultimo che si vorrà ottenere, sarà quello di avere su tutte le macchine della *farm* reale (quindi *CE*, *SE*, *UI*, *WN*) lo spazio disco utilizzato per memorizzare i dati di simulazione dell'esperimento *CMS* memorizzati all'interno del *File Server*.



Figura 19 - Il FileServer

5.2 Test NFS

I test che sono stati fatti sono molteplici, tutti effettuati per controllare l'affidabilità del collegamento *NFS* tra il *fileserver* ed il nodo *cmsgridse (SE)* del sito di sviluppo. I programmi che sono stati utilizzati sono: *IOzone*, *Bonnie* ed uno script realizzato in *perl*.

5.3 Test IOzone

IOzone è un pacchetto applicativo che mette a disposizione un insieme di strumenti grazie ai quali è possibile misurare le prestazioni di un generico *filesystem*. Gli strumenti in questo caso sono un insieme di operazioni che comprendono:

- *Read / Write* (lettura / scrittura di un nuovo file);
- *Re-read / Re-write* (lettura / scrittura di un file esistente);
- *Read backwards* (lettura di un file all'indietro, dalla fine all'inizio);
- *Read strided* (lettura del file a "salti", esempio: vengono letti 4 Kbyte sul file, ne vengono saltati 200 Kbyte, ne vengono riletti 4 ecc...);
- *Fread* (lettura di un file utilizzando la funzione *fread()* del C);
- *Fwrite* (scrittura di un file utilizzando la funzione *fwrite()* del C);
- *Random read /write* (lettura / scrittura del contenuto del file, in questo caso la locazione della lettura o scrittura all'interno del file è casuale);
- *Aio_read / aio_write* (test per l'Async I/O in scrittura ed in lettura);
- *Mmap* (test per la mappatura del file all'interno dell'user address space);

Quando viene lanciato lo *script* dalla *shell* di comando, *IOzone* effettua sequenzialmente tutte le operazioni che sono state precedentemente elencate su dei blocchi di memoria che variano di volta in volta. Ad esempio:

File size in KB	File stripe in KB	Reclean in KB/Sec	Write in KB/Sec	Rewrite in KB/Sec	Read in KB/Sec	Reread in KB/Sec
64	64	4	7620	7662	137903	141578
64	64	8	7644	7676	167544	166673
64	64	16	7617	7645	183912	182884
64	64	32	7571	7600	194538	192760
64	64	64	7569	7475	200019	198729
128	64	4	9159	9177	140507	140675

Per mettere sotto stress il nostro *fileserver* via *NFS*, si è sempre eseguito lo script con gli argomenti *Rac* passati come parametro, grazie ai quali è stato possibile utilizzare la funzione *close()*: questa funzione è necessaria se il *client NFS* ha versione 3 e consente di ridurre l'effetto cache sulla parte *client* di *NFS* (come nel nostro caso):

```
# ./iozone Rac
```

Comunque, insieme all'opzione *Rac* si sono indicati di volta in volta il file di output e la dimensione massima del file da scrivere, per poter effettuare i test su porzioni diverse del *filesystem*.

Per verificare l'efficienza e la stabilità del sistema, i test sono stati fatti sullo spazio condiviso via *NFS*. Questi hanno riguardato file di dimensioni variabili: 1 Gigabyte e 10 Gigabyte, dal momento che la dimensione media de file utilizzati dall'esperimento *CMS* sono compresi tra 500 *MegaByte* e 1 *GigaByte*. I dati rilevati comprendono solo le operazioni in scrittura ed in lettura.

NFS (1G scrittura):

File size in KB	File stripe in KB	Transfer in KB/sec
131072	64	214646
262144	64	212699
524288	64	45522
1048576	64	29645

NFS (1G lettura):

File size in KB	File stripe in KB	Transfer in KB/sec
131072	64	689026
262144	64	528826
524288	64	44629
1048576	64	33669

NFS (10G lettura):

File size in KB	File stripe in KB	Transfer in KB/sec
1048576	64	29723
2097152	64	25647
4194304	64	24373
8388608	64	22607

NFS (10G scrittura):

File size in KB	File stripe in KB	Transfer in KB/sec
1048576	64	1811528
2097152	64	86130
4194304	64	88680
8388608	64	87780

E' da notare, che nei test da 10 *Gigabyte* si abbia un crollo delle prestazioni dovuto all'esaurimento dell'effetto della cache di sistema.

5.4 Test Bonnie++

Bonnie++ è un programma basato sul *benchmark Bonnie* scritto e sviluppato da Tim Bray [70]. Questo consente di testare *filesystem* di dimensioni superiori a 2 *Gigabyte*, con macchine a 32bit ed effettua i test utilizzando le macro funzioni *creat()*, *stat()*, *unlink()*. L'output che si ottiene è convertito in *CSV* [71], un particolare formato per fogli elettronici. Successivamente, il file di output *CSV* può essere convertito sia in *HTML* [72] che in testo normale, tramite appositi *script* (*bon_csv2html*, *bon_csv2text*). I test che vengono eseguiti da *bonnie++* sono:

- I primi 6 test vengono fatti su file di dimensioni conosciute (200MByte di default, modificabile). Per ogni file viene riportato il numero di *Kilobyte* processati al secondo e la percentuale di *CPU* utilizzata. Questi test sono identici a quelli che effettua *bonnie*.
- I successivi 6 test riguarda l'utilizzo delle funzioni *creat()*, *stat()*, *unlink()*.

In dettaglio, i test di I/O riguardano:

- Input sequenziale
- Output sequenziale
- Ricerche casuali

I test sono stati effettuati su un *filesystem* montato via *NFS*, con una dimensione di 2 *GigaByte* per file.

NFS:

```
# bonnie++ -s 2000 -d /export/data05/test/Bonnie/risultati/ -u root
Using uid:0, gid:0.
Writing with putc()...done
Writing intelligently...done
Rewriting...done
Reading with getc()...done
Reading intelligently...done
start 'em...done...done...done...
Create files in sequential order...done.
Stat files in sequential order...done.
Delete files in sequential order...done.
Create files in random order...done.
Stat files in random order...done.
Delete files in random order...done.
Version 1.03      -----Sequential Output----- --Sequential Input- --Random-
                -Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
Machine          Size K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP /sec %CP
cmsgridse.pg. 2000M 10579 68 10459 2 6009 2 10773 67 11364 1 759.5 1
                -----Sequential Create----- -----Random Create-----
                -Create-- --Read--- -Delete-- -Create-- --Read--- -Delete--
                files /sec %CP /sec %CP /sec %CP /sec %CP /sec %CP /sec %CP
                  16 199 1 795 1 222 1 192 1 1006 3 229 1
cmsgridse.pg.infn.it,2000M,10579,68,10459,2,6009,2,10773,67,11364,1,759.5,1,16,199,
1,795,1,222,1,192,1,1006,3,229,1
```

5.5 Lo script perl

Oltre ai programmi *bonnie* e *IOzone*, si è utilizzato anche un ulteriore *script* scritto in *perl* per poter effettuare un ulteriore controllo sul *fileserv* NFS. Lo *script* in questione crea un certo numero di file: la quantità e la dimensione è impostabile come parametro al programma stesso. Il funzionamento del programma è molto semplice:

- Viene impostata la quantità di file
- Viene impostata la dimensione dei file
- Viene lanciato il programma con *./mytest.pl*
- Viene creata una cartella *template* che successivamente conterrà tutti i file
- Viene memorizzato il tempo di inizio del test
- Vengono scritti i file
- Vengono riletti i file
- Viene calcolato il tempo di fine dei test
- Viene prodotto come output il tempo impiegato per il test.

I test sono stati eseguiti su uno spazio disco normale e su uno spazio disco NFS.

Questi riguardavano :

- 20000 file da 1 Kbyte ciascuno
- 20000 file da 10 Kbyte ciascuno

L'output dei test è il seguente:

NFS (1K):

```

cmsfs Myscript # ./mytest.pl > test1
cmsfs Myscript # cat test1
Test starting with 20000 files, 1240 bytes each
/data05/test/Myscript/nfs_test_1CvNyC
files writing: 100%
 11 wallclock secs ( 4.41 usr +  3.08 sys =  7.49 CPU)
directory listing: 100%
  1 wallclock secs ( 0.22 usr +  0.34 sys =  0.56 CPU)
files status: 100%
  0 wallclock secs ( 0.29 usr +  0.41 sys =  0.70 CPU)
files reading: 100%
  2 wallclock secs ( 1.44 usr +  0.54 sys =  1.98 CPU)
    
```

NFS(10K):

```

cmsfs Myscript # ./mytest.pl > test1
cmsfs Myscript # cat test10
Test starting with 20000 files, 10240 bytes each
/data05/test/Myscript/nfs_test_n8onVV
files writing: 100%
  9 wallclock secs ( 4.49 usr +  3.38 sys =  7.87 CPU)
directory listing: 100%
  1 wallclock secs ( 0.25 usr +  0.30 sys =  0.55 CPU)
files status: 100%
  2 wallclock secs ( 0.32 usr +  0.37 sys =  0.69 CPU)
files reading: 100%
  3 wallclock secs ( 1.85 usr +  1.05 sys =  2.90 CPU)
    
```

Normale (1K):

```

cmsfs MYscript # ./mytest.pl > test1
cmsfs MYscript # cat test1
Test starting with 20000 files, 1240 bytes each
/data04/test/MYscript/nfs_test_1ll1AFC
files writing: 100%
 10 wallclock secs ( 4.42 usr +  3.25 sys =  7.67 CPU)
directory listing: 100%
  0 wallclock secs ( 0.24 usr +  0.31 sys =  0.55 CPU)
files status: 100%
  1 wallclock secs ( 0.36 usr +  0.31 sys =  0.67 CPU)
files reading: 100%
  3 wallclock secs ( 1.37 usr +  0.58 sys =  1.95 CPU)
    
```

Normale (10K):

```

cmsfs MYscript # ./mytest.pl > test10
cmsfs MYscript # cat test10
Test starting with 20000 files, 10240 bytes each
/data04/test/MYscript/nfs_test_UMfbRZ
  files writing: 100%
    9 wallclock secs ( 4.20 usr +  3.79 sys =  7.99 CPU)
  directory listing: 100%
    1 wallclock secs ( 0.29 usr +  0.27 sys =  0.56 CPU)
  files status: 100%
    2 wallclock secs ( 0.32 usr +  0.38 sys =  0.70 CPU)
  files reading: 100%
    2 wallclock secs ( 1.96 usr +  0.69 sys =  2.65 CPU)

```

5.6 Creazione degli utenti

Oltre ad eseguire il test di affidabilità e di prestazioni sul *file server* montato via *NFS*, si è provveduto anche alla creazione di tutti gli utenti (*Grid* e non) per la *farm* di produzione locale. Questo è stato fatto perché il sito di produzione dovrà essere configurato per accettare *job* provenienti da *account* di utenti *Grid* ed *account* di utenti non *Grid*: per realizzare ciò gli utenti sono stati creati via server *LCFGng* e all'interno del batch system è stata creata una coda per i job locali. In questo modo si avrà una *farm* ibrida.

Gli *account* via *LCFGng* vengono creati seguendo un *template* fornito con la versione ufficiale di *INFN Grid* collocato nella directory `/var/obj/conf/server/source/git-current/UserInterface-local-cfg.h` corrispondente al file di configurazione della User Interface:

```

EXTRA(auth.users)          utente1
EXTRA(auth.groups)        utente1
auth.userpwd_utente1      NV0AdvCJ.xbRI
auth.useruid_utente1      501
auth.groupgid_utente1     501
auth.usergroup_utente1    utente1
auth.usercomment_utente1  nome cognome user

```

All'interno di questo file vengono indicate:

- `auth.users` *UserName* utente (in questo caso `utente1`);
- `auth.groups` crea un gruppo con nome `utente1`;
- `userpwd_gino` Frase segreta per entrare nel sistema per `utente1` (questa è criptata);
- `useruid_gino` UID corrispondente all'utente `utente1`;

- `groupgid_gino` UID per il gruppo utente1;
- `usergroup_gino` gruppo di appartenenza per il gruppo utente1;
- `usercomment_gino` commento per il gruppo utente1;

Nel nostro specifico caso, non avendo a disposizione una *User Interface*, si sono creati degli utenti fittizi nella macchina *cmsgridse* di sviluppo (tramite il file `/var/obj/conf/server/source/git-current/StorageElement-local-cfg.h.`) :

```

/* Utenti extra creati per test */
EXTRA(auth.users)          leorsini testing1 testing2
EXTRA(auth.groups)         leorsini testing1
auth.userhome_leorsini     /home/leorsini
auth.userhome_testing1    /home/testing1
auth.userhome_testing2    /home/testing2
auth.userpwd_leorsini      NV0AdvCJ.xbRI
auth.userpwd_testing1     NV0AdvCJ.xbRI
auth.userpwd_testing2     NV0AdvCJ.xbRI
auth.groupgid_leorsini     520
auth.groupgid_testing1    521
auth.useruid_leorsini      520
auth.useruid_testing1     521
auth.useruid_testing2     522
auth.usergroup_leorsini    leorsini
auth.usergroup_testing1   testing1
auth.usergroup_testing2   leorsini
auth.usercomment_leorsini  User leorsini
auth.usercomment_testing1  User gruppo leorsini
auth.usercomment_testing2  User gruppo leorsini
/*

```

Successivamente, dopo aver aggiornato il profilo della macchina con i consueti strumenti, si è verificato che le cartelle *home* degli utenti *leorsini*, *testing1* e *testing2* erano state create, e all'interno del file `/etc/groups` vi erano i gruppi utenti creati (*leorsini*, *testing1*, *testing2*). Infine, si sono eseguiti test sull'affidabilità del sistema di protezione e gestione delle risorse. Questi test riguardarono:

- Creazione e cancellazione di file/cartelle da parte di utenti appartenenti allo stesso gruppo (test eseguito con successo);
- Creazione e cancellazione di file/cartelle da parte di utenti con gruppi differenti (test eseguito con successo);
- Modifica di file/cartelle da parte di utenti facenti parte dello stesso gruppo (test eseguito con successo);

- Modifica di file/cartelle da parte di utenti non facenti parte dello stesso gruppo (test eseguito con successo).

5.7 Creazione delle code sul *bach-system* del CE

Per poter permettere alla comunità locale *CMS* di usufruire della *farm* di produzione per sottomettere job locali, senza passare per il *Resource Broker*, si è provveduto alla creazione di una coda locale sul *bach system*. La sua creazione è avvenuta sul *Computing Element (CE)* eseguendo i seguenti script:

```
qmgr -c "create queue plocal queue_type=execution"
qmgr -c "set queue plocal started=true"
qmgr -c "set queue plocal enabled=true"
qmgr -c "set queue plocal resources_default.nodes=1"
qmgr -c "set queue plocal resources_default.walltime=3600"
qmgr -c "set queue plocal acl_hosts=h1+h2+h3"
qmgr -c "set queue plocal acl_host_enable=true"
qmgr -c "set queue plocal priority=20"
```

In cui vengono settati i seguenti parametri:

- Linea 1: creazione della coda *PLOCAL* del tipo *execution* (specifica che deve essere settata per tutte le code che si vorranno creare in futuro);
- Linea 2: specifica che il *job* inserito nella coda *PLOCAL* può essere eseguito;
- Linea 3: specifica se la coda può accettare *job*;
- Linea 4: specifica il numero predefinito di nodi da concedere ai *job* inseriti nella coda *PLOCAL*;
- Linea 5: specifica il tempo massimo di concessione delle risorse per il *job* inserito nella coda *PLOCAL*;
- Linea 6: specifica gli *host* autorizzati a sottomettere *job* in *PLOCAL*: in questo caso *h1*, *h2* e *h3* (da usare in congiunzione con *acl_host_enabled*);
- Linea 7: specifica che i *job* possono essere sottomessi nella coda *PLOCAL* solo dagli *host* elencati in *acl_host_enabled*;
- Linea 8: specifica la priorità della coda con le altre presenti.

CAPITOLO 6 CONCLUSIONI

6.1 Prospettive future del sito di sviluppo

Come accennato in precedenza, il sito di sviluppo è stato costituito per poter eseguire svariati test in prospettiva futura, non solo destinati al *fileserver*. Infatti, la prossima versione di *INFN Grid* abbandonerà gli strumenti di installazione tramite server *LCFGng* ed adotterà il *repository apt* gestito tramite *YAIM* [73]: il sito di test sarà un ottimo strumento per imparare ad usare al meglio questi nuovi strumenti. Inoltre sarà necessario provare altre configurazioni, che riassunte sono:

- Installazione sito tramite *YAIM e repository APT* [74];
- Installazione di una farm ibrida: *Red Hat 7.3 e Scientific Linux*;

Tutta la configurazione sviluppata nel sito di test sarà poi applicata a quello di produzione.

6.2 Considerazioni finali

L'installazione e la configurazione del sito di sviluppo *Grid-Dev*, ha consentito di testare il funzionamento del *File Server* che era attivo sulla vecchia *farm* di produzione per *CMS* di Perugia (ormai dismessa). Lo spazio disco ora, grazie al lavoro di test svolto, viene montato automaticamente via *NFS* su tutti i nodi del sito *Grid* di produzione tramite server *LCFGng*. Tale risultato è stato reso possibile modificando solo alcune variabili nel file di configurazione degli *host* del sito. I test che si sono eseguiti non avevano lo scopo di analizzare le prestazioni del collegamento *NFS* (questo lo si sarebbe fatto anche senza una *farm Grid* di test) ma solo di certificare il corretto funzionamento all'interno di *Grid*. I risultati che si sono ottenuti sono eccellenti ed in linea con le nostre aspettative.

APPENDICE A – Listato dello script perl

```
#!/usr/bin/perl -Tw
use strict;
use Benchmark;
use File::Temp qw(tempdir tempfile);
use IO::Dir ();
use IO::File ();

### what directories to use for testing
my @DIRS      = qw(
                    /data05/test/Myscript/
                );
### how many temporary files should it be created?
my $FILES     = 20_000;
### what should be the size of each temporary file?
my $FILESIZE  = 1_240; #1k
#my $FILESIZE = 1;
### what template should it use for file creation? (use at least 4
'X')
my $TEMPLATE  = 'nfs_test_XXXXXX';
### should I delete temporary files at program end?
my $CLEANUP  = 1;

$I = 1;
{ ### helper routines
  my ($t0, $t1);
  sub Tinit {
    print " @_: 000%";
    $t0 = new Benchmark;
  }
  sub Tend {
    $t1 = new Benchmark;
    print $/, ' ', timestr(timediff($t1, $t0)) , $/;
  }
  sub Show {
    print "\b" x 4;
    printf "%03d%", $_[0] * 100/$FILES;
  }
  sub dir_based_test {
    my ($tmpdir, $type, $title) = @_;
    Tinit($title);
    my $d = new IO::Dir $tmpdir or warn "Can't open directory
$tmpdir: $!\n", retur$
    my $idx = 0;
    (my $tmatch = $TEMPLATE) =~ s/X/./g;
    while (defined($_ = $d->read)) {
      next unless /$tmatch/o;
      if ($type eq 'read') {
        my $fh = IO::File->new("$tmpdir/$_") or warn
"$!\n", next;
        while (<$fh) {}
        $fh->close;
      }
      elsif ($type eq 'stat') {
        stat("$tmpdir/$_") or warn $!, $/, next;
      }
      Show(++$idx);
    }
    $d->close;
  }
}
```

APPENDICE A

```
        Tend;
    }
}

{ ### running tests for each directory
print "Test starting with $FILES files, $FILESIZE bytes each\n";
    foreach my $dir (@DIRS) {
        ### creating a temporary subdirectory in the test
        directory.
        my $tmpdir = tempdir( $TEMPLATE, DIR => $dir, CLEANUP =>
$CLEANUP);
        print "$tmpdir\n";
        ### creating test files
        Tinit 'files writing';
        foreach (1 .. $FILES) {
            my ($fh, $fname) = tempfile($TEMPLATE, DIR => $tmpdir,
UNLINK => 0);
            print $fh 'A' x $FILESIZE;
            close $fh;
            Show($_);
        }

        Tend;

        dir_based_test($tmpdir, 'list', 'directory listing');
        dir_based_test($tmpdir, 'stat', 'files status');
        dir_based_test($tmpdir, 'read', 'files reading');
    }
}
```

APPENDICE B – Listato del file my_kernel.spec

my_kernel.spec

```
# Customize it with the kernel version you are running
Version: 2.4.27

# Customize here your config files location
%define KernelConfDir /usr/src/linux-%{version}

# Customize the sub-release (it will be used in the "EXTRAVERSION"
# variable of the main Makefile)
Release: test_1

# Write 1 if you are compiling a 2.4 kernel, 0 if a 2.6 kernel
%define kernel24 1

Name: kernel
Summary: The Linux Kernel
License: GPL
Group: System Environment/Kernel
Vendor: The Linux Community
URL: www.kernel.org
Source: linux-%{version}.tar.bz2
BuildRoot: /var/tmp/%{name}-%{version}-root
Provides: module-info, kernel = %{version}
Provides: kernel-drm = 4.1.0, kernel-drm = 4.2.0

%define __spec_install_post /usr/lib/rpm/brp-compress || :
%define debug_package %{nil}

%description
Linux Kernel; version recompiled by INFN-PADOVA for EGEE/LCG Grid.
Xeon + No Hyperthreading + Intel Pro 1000 + 3Ware.
Single processr version.

%prep

%setup -n linux-%{version}

%build
[ -d $RPM_BUILD_ROOT ] && rm -fr $RPM_BUILD_ROOT

cp -f %{KernelConfDir}/.config .config

make clean
# For 2.4 make EXTRAVERSION=<blabla> does not works correctly
# (it does not update version.h) so we modify the Makefile :-('
perl -p -i -e 's/^EXTRAVERSION.*/EXTRAVERSION=-%{release}/'
Makefile
%if %{kernel24}
rm -f include/linux/modversions.h include/linux/version.h
make oldconfig
make dep
make include/linux/version.h
%else
make oldconfig
%endif
make -j2 bzImage
```

APPENDICE B

```
make -j2 modules
# Create directories
mkdir -p $RPM_BUILD_ROOT/boot $RPM_BUILD_ROOT/lib
$RPM_BUILD_ROOT/lib/modules
# Install kernel + modules
cp arch/i386/boot/bzImage $RPM_BUILD_ROOT/boot/vmlinuz-%{version}-
%{release}
cp System.map $RPM_BUILD_ROOT/boot/System.map-%{version}-
%{release}
cp .config $RPM_BUILD_ROOT/boot/config-%{version}-%{release}
make INSTALL_MOD_PATH=$RPM_BUILD_ROOT modules_install

# Building kernel with SMP support
cp -f %{KernelConfDir}/.config_smp .config

# For 2.4 make EXTRAVERSION=<blabla> does not works correctly
# (it does not update version.h) so we modify the Makefile :-(
perl -p -i -e 's/^\^EXTRAVERSION.*\/EXTRAVERSION=-%{release}smp/'
Makefile
%if %{kernel24}
rm -f include/linux/modversions.h include/linux/version.h
make oldconfig
make dep
make include/linux/version.h
%else
make oldconfig
%endif
make -j2 bzImage
make -j2 modules
# Install SMP kernel + modules
cp arch/i386/boot/bzImage $RPM_BUILD_ROOT/boot/vmlinuz-%{version}-
%{release}smp
cp System.map $RPM_BUILD_ROOT/boot/System.map-%{version}-
%{release}smp
cp .config $RPM_BUILD_ROOT/boot/config-%{version}-%{release}smp
make INSTALL_MOD_PATH=$RPM_BUILD_ROOT modules_install

%install

%files
%defattr (-, root, root)
%dir /lib/modules
/lib/modules/%{version}-%{release}
/boot/vmlinuz-%{version}-%{release}
/boot/System.map-%{version}-%{release}
/boot/config-%{version}-%{release}

#####
##
### SMP !
#####
##
%package smp
Summary: Kernel SMP
Group: System Environment/Kernel
Provides: module-info, kernel = %{version}
Provides: kernel-drm = 4.1.0, kernel-drm = 4.2.0

%description smp
Linux Kernel; version recompiled by INFN-PADOVA for EGEE/LCG Grid.
Dual Xeon + No Hyperthreading + Intel Pro 1000 + 3Ware.
Dual SMP version.
```

```
%files smp
%defattr (-, root, root)
%dir /lib/modules
/lib/modules/{version}-{release}smp
/boot/vmlinuz-{version}-{release}smp
/boot/System.map-{version}-{release}smp
/boot/config-{version}-{release}smp
```


RINGRAZIAMENTI

Vorrei ringraziare calorosamente tutto il dipartimento di Fisica dell'Università di Perugia e l'INFN che in questi mesi mi ha fornito tutto il materiale necessario per svolgere il mio lavoro di tesi. Un particolare ringraziamento lo rivolgo a Mirko Mariotti, il prof. Leonello Servoli ed il prof. Attilio Santocchia per la pazienza, i suggerimenti e l'esperienza che mi ha consentito di crescere professionalmente.

Ai miei insegnanti universitari.

A Daniela e ai miei genitori che hanno creduto in me.



Perugia, Febbraio 2005

BLOGRAFIA

- [1] CMS URL : <http://cmsinfo.cern.ch/Welcome.html/>
- [2] SSH: SSH, The Secure Shell: The Definitive Guide di Daniel J. Barrett, Richard Silverman; Paperback ulteriori informazioni URL: <http://it.wikipedia.org/wiki/SSH>
- [3] PBS URL : <http://www.openpbs.org/>
- [4] MAINFRAME URL: <http://it.wikipedia.org/wiki/Mainframe>
- [5] CALCOLATORI VETTORIALI URL: <http://it.wikipedia.org/wiki/Mainframe>
- [6] TRANSISTOR URL: <http://it.wikipedia.org/wiki/Transistor>
- [7] CLUSTER URL : Distributed Systems: Principles and Paradigms di Andrew S. Tanenbaum, Maarten van Steen
- [8] MPI: Distributed Systems: Principles and Paradigms di Andrew S. Tanenbaum, Maarten van Steen
- [9] PVM : Distributed Systems: Principles and Paradigms di Andrew S. Tanenbaum, Maarten van Steen
- [10] HPF : Distributed Systems: Principles and Paradigms di Andrew S. Tanenbaum, Maarten van Steen
- [11] TASK FARM : Distributed Systems: Principles and Paradigms di Andrew S. Tanenbaum, Maarten van Steen
- [12] PIPE LINE : Distributed Systems: Principles and Paradigms di Andrew S. Tanenbaum, Maarten van Steen
- [13] GLOBAL GRID FORUM URL: <http://www.gridforum.org>
- [14] API URL : <http://it.wikipedia.org/wiki/API>
- [15] I-WAY URL : <http://gridcafe.web.cern.ch/gridcafe/Gridhistory/ancestors.html>
- [16] CONDOR URL : <http://cs.wisc.edu/condor>
- [17] MULTITASKING URL : <http://it.wikipedia.org/wiki/Multitasking>
- [18] MIRON LIVNY URL : <http://www.cs.wisc.edu/~miron/>

- [19] UNIX: UNIX for Dummies di John R. Levine, Margaret Levine Young; Paperback ulteriori info URL :
<http://it.wikipedia.org/wiki/Unix>
- [20] SETI@HOME URL : <http://setiathome.ssl.berkeley.edu/>
- [21] DAVID ANDERSON URL :
<http://setiathome.ssl.berkeley.edu/project.html>
- [22] GLOBUS URL : <http://globus.org>
- [23] GTPL URL: <http://www-unix.globus.org/toolkit/license.html>
- [24] SDK URL: <http://www.google.com>
- [25] EDG URL : <http://eu-datagrid.web.cern.ch/eu-datagrid/>
- [26] EGEE URL : <http://cern.ch/egee>
- [27] CERN URL : <http://cern.ch>
- [28] LHC URL: <http://lhc-new-homepage.web.cern.ch/lhc-new-homepage/>
- [29] ATLAS URL : <http://atlas.web.cern.ch/Atlas/Welcome.html>
- [30] HIGGS URL: http://it.wikipedia.org/wiki/Bosone_di_Higgs
- [31] MODELLO STANDARD URL :
[http://it.wikipedia.org/wiki/Modello_standard_\(fisica\)](http://it.wikipedia.org/wiki/Modello_standard_(fisica))
- [32] TESLA URL : <http://it.wikipedia.org/wiki/Tesla>
- [33] ALICE URL : <http://alice.web.cern.ch/Alice/AliceNew/>
- [34] QUARK URL : <http://it.wikipedia.org/wiki/Quark>
- [35] GLUONI URL : <http://it.wikipedia.org/wiki/Gluone>
- [36] LHC-B URL: <http://lhcb.web.cern.ch/lhcb/>
- [37] PETABYTE URL : <http://it.wikipedia.org/wiki/Byte>
- [38] LCG URL : <http://lcg.web.cern.ch/LCG/>
- [39] INFN URL : <http://infn.it>
- [40] GARR URL : <http://garr.it>
- [41] VO URL : <http://grid-it.cnaf.infn.it>
- [42] LDAP: LDAP Directories Explained: An Introduction and Analysis di Brian Arkills
- [43] VOMS URL: <http://www.egrid.it/doc/sysadm/voms>
- [44] GSI URL: <http://www-unix.globus.org/toolkit/docs/>
- [45] NASA URL : <http://nasa.gov>

- [46] X.509 URL: <http://www.ietf.org/html.charters/pkix-charter.html>
- [47] SSL Network Security with OpenSSL di John Viega ulteriori info:
URL <http://openssl.org>
- [48] XML Learning XML, Second Edition di Erik T. Ray ulteriori info
URL: <http://it.wikipedia.org/wiki/XML>
- [49] UDP A TCP/UDP Network Protocols Handbook di Technologies
Javvin Technologies
- [50] HTTP HTTP: The Definitive Guide di David Gourley, Brian Totty
- [51] JDL URL : <http://www.google.com>
- [52] FTP: TCP/IP Bible (Bible) di Rod Scrimger, Paul LaSalle, Mridula
Parihar, Meeta Gupta ulteriori informazioni URL :
<http://it.wikipedia.org/wiki/FTP>
- [53] IP: TCP/IP Bible (Bible) di Rod Scrimger, Paul LaSalle, Mridula
Parihar, Meeta Gupta ulteriori informazioni URL:
http://it.wikipedia.org/wiki/Internet_Protocol
- [54] PXE URL: <http://www.kegel.com/linux/pxe.html>
- [55] DAG : URL <http://grid-it.cnaf.infn.it/>
- [56] PYTHON URL: <http://www.python.org>
- [57] PEM : URL : <http://www.openssl.org/docs/>
- [58] NAT: URL : <http://it.wikipedia.org/wiki/NAT>
- [59] FIREWALL : Firewall di HENNING MANKELL ulteriori info URL:
<http://it.wikipedia.org/wiki/Firewall>
- [60] DHCP : The DHCP Handbook (2nd Edition) di by Ralph Droms, Ted
Lemon; Paperback ulteriori informazioni URL:
<http://it.wikipedia.org/wiki/DHCP>
- [61] TFTP : URL : <http://www.faqs.org/rfcs/rfc1350.html>
- [62] XINETD URL: <http://www.xinetd.org/>
- [63] SYSLINUX : URL : <http://syslinux.zytor.com/>
- [64] IPTBLES: Linux iptables Pocket Reference di Gregor N. Purdy
ulteriori info URL: <http://www.netfilter.org/>
- [65] PAT : URL : <http://it.wikipedia.org/wiki/NAT>
- [66] DNS : DNS and BIND, Fourth Edition di by Paul Albitz, Cricket Liu;
Paperback ulteriori info URL: <http://it.wikipedia.org/wiki/DNS>

- [67] BIND: DNS and BIND, Fourth Edition di by Paul Albitz, Cricket Liu;
Paperback ulteriori info URL: <http://it.wikipedia.org/wiki/DNS>
- [68] CGI : CGI Programming with Perl di Gunther Birznieks, et al;
Paperback ulteriori info URL: <http://it.wikipedia.org/wiki/Cgi>
- [69] NFS : Managing NFS and NIS, 2nd Edition di Hal Stern, et al;
Paperback ulteriori informazioni URL:
http://it.wikipedia.org/wiki/File_system
- [70] TIM BRAY URL : <http://www.textuality.com/>
- [71] CSV URL: <http://www.google.com>
- [72] HTML HTML & XHTML: The Complete Reference di Thomas
Powell ulteriori info URL: <http://www.w3c.org>
- [73] YAIM URL: <http://lcg.web.cern.ch/LCG/>
- [74] APT Debian GNU/Linux Bible di Steve Hunger ulteriori info URL:
www.apt-get.org/

INDICE DELLE FIGURE

Figura 1 - Farm di produzione CMS Perugia.....	7
Figura 2 - gli strati del middleware Grid	11
Figura 3 - Il progetto LHC	13
Figura 4 - sezione di CMS	14
Figura 5- siti LCG nel Mondo.....	15
Figura 6 - siti INFN Grid in Italia.....	16
Figura 7 - gerarchia di LCG	16
Figura 8 - Rappresentazione delle VO tramite LDAP.....	17
Figura 9 - Organizzazione delle VO con VOMS	18
Figura 10 - Struttura globale di INFN Grid	20
Figura 11 - Funzionamento di LCFGng.....	21
Figura 12 - Struttura directory di LCFGng	26
Figura 13 - Home page del web server su LCFGng	28
Figura 14 - Struttura di Quattor.....	30
Figura 15 - Sito normale e uno modificato con rete privata	33
Figura 16 - Struttura di Grid-Dev	35
Figura 17 - vista globale di Grid-Dev.....	37
Figura 18 - Linea temporale di caricamento del Kernel	42
Figura 19 - Il FileServer.....	55

