
Indice

Capitolo I	3
Evoluzione Del Calcolo Computazionale	3
I-1 Prestazioni dei Calcolatori	4
I-2 Il Calcolo Computazionale Parallelo	7
I-3 I Cluster	12
Capitolo II	17
Componenti Di Un Cluster Beowulf	17
II-1 Caratteristiche Hardware	19
II-2 Componenti Software	22
II-2.1 Strumenti e servizi per la condivisione	23
Il file /etc/exports	24
I file /etc/hosts.allow e /etc/hosts.deny	26
Avvio Dei Servizi	28
II-2.2 Configurazioni Per La Comunicazione	30
II-2.3 Strumenti E Servizi Per L'Account	33
Server NIS	34
II-2.4 Strumenti E Servizi Per L'Accesso Remoto	37
II-2.5 Strumenti E Servizi Per La Parallelizzazione	40
II-2.6 Strumenti E Servizi Per Il Monitoraggio	44
Capitolo III	46
Una Metodologia Per L'Installazione Di Un Cluster Beowulf	46
III-1 Il file kickstart	49
III-2 Installazione Di Un Cluster Mediante Kickstart	51
III-2.1 Network Boot	52
III-2.2 Pubblicazione del kickstart	55
III-2.3 Sorgenti Della Distribuzione	58
III-3 Software Per La Creazione Automatica Di Cluster	59
III-3.1 OSCAR	60
III-3.2 ROCKS	63

Capitolo IV	66
Kickstart Cluster Configurator	66
IV-1 Dettagli implementativi.....	72
IV-2 Supporto Ed Estensione A Sistemi Linux Differenti	74
IV-3 Definizione Di Nuovi Tools.....	81
IV-4 Aggiunta di nuovi nodi.....	84
IV-5 Confronto Con OSCAR E ROCKS.....	86
Capitolo V	88
Esempio Applicativo	88
V-1 PostFirstReboot	92
Capitolo VI	95
Direzioni Future	95
VI-1 Gli Scenari.....	95
VI-2 Cloud Computing e Virtualizzazione.....	98
Conclusioni	101

Capitolo I

Evoluzione Del Calcolo Computazionale

Il calcolo computazionale è una scienza relativamente giovane se consideriamo che i moderni calcolatori, intesi come macchine basate sull'architettura di Von Neumann¹, non sono stati sviluppati prima di 50 anni fa. Esso pone il suo obiettivo nella risoluzione di problemi di natura scientifica attraverso l'esecuzione sequenziale di semplici operazioni.

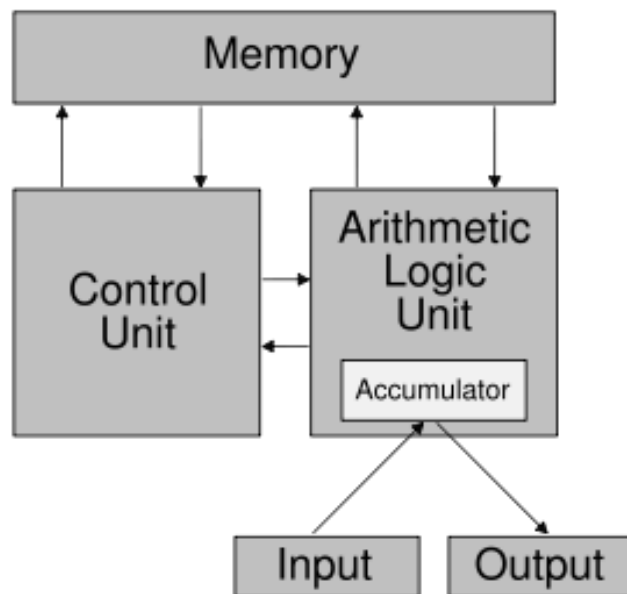


Figura I-1. Schematizzazione dell'architettura di Von Neumann

¹ John von Neumann, matematico e informatico ungherese naturalizzato statunitense, definì, nel 1946, l'architettura di base dei calcolatori. L'architettura da esso definita prevede l'iterazione fra dispositivi di calcolo (CPU + ALU), memoria e dispositivi di I/O attraverso un canale di comunicazione (BUS).

Per quanto possano sembrare macchine molto potenti (ed in passato anche molto grandi), capaci di eseguire velocemente e in modo relativamente preciso le elaborazioni richieste, i calcolatori sono in realtà strumenti molto semplici, se consideriamo le operazioni elementari che sono capaci di eseguire. Tutta la loro potenza risiede in due caratteristiche: la velocità e la qualità delle operazioni sequenziali che devono eseguire; la prima intrinseca alla macchina e la seconda propria delle capacità umane di descrivere un fenomeno attraverso passi operativi elementari e temporizzati.

Tali passi operativi sono definiti in quello che è più comunemente conosciuto come algoritmo, un procedimento che consente di ottenere un risultato eseguendo, in un determinato ordine, un insieme di operazioni semplici e finite.

I-1 Prestazioni dei Calcolatori

I primi calcolatori elettronici sono stati macchine estremamente grandi ed altrettanto costose, ma le loro capacità non superavano di molto quelle di un attuale telefono cellulare. Risolvere problemi più grandi, e soprattutto in tempi sempre più brevi, è stato, ed è tutt'oggi, l'obiettivo principale di scienziati ed ingegneri.

	Dimensioni	Costo	Potenza di calcolo
ENIAC ²	9 x 2 x 30 m	500.000 \$	50.000 OPS
SMARTPHONE	3 x 15 x 1 cm	400 \$	200.000.000 IPS ³

Tabella I-1. Confronto fra il supercalcolatore ENIAC e gli smartphone moderni.

Il tempo t , necessario alla elaborazione di un processo di calcolo, è proporzionale ai due fattori descritti prima: T (la complessità di tempo dell'algoritmo) , μ (il periodo di clock identificabile come il numero di operazioni eseguite al secondo) e k un fattore di proporzionalità.

$$t = kT\mu$$

Per aumentare le prestazioni di un sistema di calcolo bisogna necessariamente agire su una delle due variabili T e μ .

L'aumento della velocità di un processore è strettamente legato alla sue dimensioni, ed in particolare alla distanza fra i suoi componenti.

Consideriamo quindi la relazione

$$v = \frac{s}{t} \rightarrow s = vt \rightarrow s = \frac{v}{f}$$

² Electrical Numerical Integrator and Calculator, realizzato nel 1946 è considerato essere il primo calcolatore.

³ Istruzioni al secondo. Per eseguire un'operazione possono essere necessarie più istruzioni

La Legge sulla Relatività Ristretta, pubblicata da Einstein nel 1905, impone in $2 \cdot 10^{10}$ *cm/s* la massima velocità di propagazione di un segnale elettrico nel rame (essendo $v = Ck$, con C velocità della luce e k un valore di proporzionalità che nel rame è di 0.7).

Ne risulta che con un clock di 10 *Ghz* e di 1 *Thz* i circuiti all'interno del calcolatore non possono superare rispettivamente la dimensione di 2 *cm* e 0.2 *mm*. Circuiti così piccoli dissipano enormi quantità di energia termica causando notevoli problemi di raffreddamento. Per migliorare la potenza di calcolo, agendo sulle componenti hardware, sarebbero necessarie, quindi, ingenti risorse economiche ed il superamento di barriere tecnologiche di difficile risoluzione.

Trovando nel costo e nella tecnologia un limite invalicabile per quel periodo, gli scienziati si sono concentrati nell'ideazione di algoritmi più performanti, capaci cioè di ottimizzare l'utilizzo delle risorse e di ridurre al minimo possibile il numero di operazioni necessarie.

Sebbene nella risoluzione di un problema esistano infiniti algoritmi, gli scienziati sono giunti alla definizione di algoritmi considerati "ottimali"⁴, cioè non ulteriormente semplificabili e migliorabili.

⁴ Un problema ha complessità superiore $O(g(n))$ se esiste un algoritmo che lo risolve con complessità $O(g(n))$, mentre ha complessità inferiore $\Omega(f(n))$ se ogni algoritmo che lo risolve ha complessità di almeno $\Omega(f(n))$. In tal senso un algoritmo è considerato ottimale se la sua complessità $O(f(n))$ è uguale alla complessità $\Omega(f(n))$ del problema.

Ben presto anche questa soluzione non ha più garantito miglioramenti prestazionali, spingendo gli scienziati all'ideazione di nuovi strumenti e metodologie.

I-2 Il Calcolo Computazionale Parallelo

Basandosi sul paradigma “*divide et impera*”⁵, è stato definito un nuovo modello di algoritmo detto parallelo.

Un algoritmo parallelo si può considerare come un insieme di procedimenti (omogenei o non) eseguiti simultaneamente allo scopo di risolvere un problema comune. I singoli procedimenti, che si possono intendere come singole unità elaborative, con un proprio input ed un proprio output, assumono significato solo nel momento in cui vengono relazionati agli altri.

Una delle prime definizioni di parallelismo (Murli, 2006) è stata supportata da un esempio che è ormai divenuto uno standard nella letteratura. L'esempio prende in considerazione una squadra di operai a lavoro per la costruzione di una casa.

⁵ Il paradigma *divide et impera* prevede la risoluzione di un problema attraverso la sua scomposizione in sottoproblemi di minore complessità.

Nell'architettura della macchina di Von Neumann è come se un solo operaio svolgesse l'intera costruzione della casa mentre nella nuova architettura più operai lavorano insieme.

Nella costruzione di un muro possiamo identificare tre azioni principali: applicazione del cemento, posizionamento del mattone e livellatura del mattone. A seconda da chi, e quando, sono svolte queste tre azioni, possono essere specificate due tipologie di parallelismo:

- Parallelismo temporale

- Parallelismo spaziale

Il parallelismo temporale, conosciuto anche come pipeline, vede la squadra di operai svolgere le tre azioni in modo sincronizzato, tale che ognuno degli operai svolge un'unica operazione. In questo contesto, il muro può essere visto come l'intero processo da svolgere, le tre azioni possono essere considerate come le espressioni elementari da svolgere ed i mattoni, i dati necessari a risolvere il problema. E' da notare che il posizionamento di ogni mattone prevede l'iterazione con ognuno degli operai.

Il parallelismo spaziale invece vede la squadra di operai svolgere tutte le azioni necessarie alla costruzione del muro agendo su mattoni

diversi, e ciò significa che ogni operaio si occupa dell'intero procedimento per il posizionamento di un singolo mattone. In questo caso, ogni mattone è utilizzato da un singolo operaio. Rimanendo ancora sullo stesso esempio, è possibile definire una terza tipologia di parallelismo, detto asincrono.

Il parallelismo asincrono vede la squadra di operai svolgere azioni diverse su parti diverse della casa non strettamente legate l'una con l'altra. Ad esempio un operaio costruisce i muri, un altro si occupa del posizionamento di porte e finestre, e un altro ancora si occupa della pavimentazione.

I tre tipi di parallelismo, descritti nell'esempio, rientrano nella classificazione dei sistemi di calcolo definita dall'ingegnere e informatico statunitense Michael J. Flynn, nel 1966, comunemente conosciuta come Tassonomia Di Flynn (M.J. Flynn).

Flynn ha suddiviso i sistemi di calcolo in quattro aree principali:

- SISD (Single Instruction Single Data), categoria in cui rientrano i normali calcolatori basati sull'architettura di Von Neumann

- SIMD (Single Instruction Multiple Data), categoria in cui rientrano i calcolatori paralleli che realizzano il parallelismo spaziale
- MISD (Multiple Instruction Single Data), categoria in cui rientrano i calcolatori basati su pipeline, che realizzano quindi il parallelismo temporale
- MIMD (Multiple Instruction Multiple Data), categoria di calcolatori che realizzano il parallelismo asincrono

Rientrano nella categoria SIMD gli array processor ed i vector processor, entrambi una naturale risposta all'utilizzo di strutture dati come matrici e vettori. Gli array processor sono costituiti da più processori capaci di elaborare ognuno più dati simultaneamente, mentre i vector processor hanno un singolo processore che elabora più dati. Un esempio di sistemi di tipo MISD sono i systolic array⁶, che di fatto implementano un sistema pipeline.

L'architettura consiste in una serie di unità elaborative coordinate da un unico clock globale, ognuna delle quali riceve un dato, lo elabora, e lo restituisce all'unità successiva. Questo tipo di architettura ha sempre

⁶ Gli array sistolici prendono il nome dall'omonima caratteristica del sistema circolatorio sanguigno. Il sangue, pompato dal cuore, segue un percorso ben definito all'interno del sistema venoso.

suscitato poco interesse, per la difficoltà di definizione di algoritmi paralleli.

I sistemi di tipo MIDM sono costituiti da un insieme di più unità elaborative interconnesse fra loro, ma completamente indipendenti l'una dall'altra. Il sistema è così capace di svolgere simultaneamente operazioni diverse su dati diversi. La connessione fra i processori è effettuata attraverso un sistema di memoria condivisa, che può essere globale o distribuito. I sistemi realizzati con memoria condivisa globale, sono detti anche UMA (Uniform Memory Access) in quanto consentono un accesso uniforme alla memoria, mentre sono detti NUMA (Non Uniform Memory Access), i sistemi a memoria condivisa distribuita che non garantiscono un accesso uniforme alla memoria.

Queste architetture, unite a software paralleli, garantiscono altissime prestazioni, ma prevedono un hardware specializzato e molto costoso.

I-3 I Cluster

Alle tipologie di base fornite da Flynn, si sono aggiunte nuove tecnologie, identificabili come la fusione della tipologia SIMD e MIMD. Rientrano in tali architetture i cluster che, ideati inizialmente dall'IBM negli anni 60, al fine di collegare fra loro diversi mainframe, hanno avuto larga diffusione solo negli ultimi decenni. I cluster consistono in un insieme di calcolatori singoli, collegati fra loro attraverso reti ad alta velocità che consentono, attraverso software o strumenti specializzati, di realizzare supercomputer per il calcolo parallelo con la caratteristica di essere molto economici ed altamente dinamici. Più generalmente un cluster può anche essere definito come l'utilizzo di due o più computer per risolvere lo stesso problema (<http://www.beowulf.org/overview/index.html>). La diffusione dei sistemi cluster si è sviluppata parallelamente a quattro caratteristiche:

- sistemi di interconnessione ad alta velocità
- aumento delle prestazioni dei microprocessori
- bassissimo costo dei microprocessori
- diffusione di software e sistemi operativi Open Source

Esistono diversi tipi di cluster che assumono nomenclature diverse, a seconda della tipologia di problemi da risolvere a cui sono. Principalmente si possono considerare tre tipologie diverse: i cluster utilizzati per HA (High Availability), i cluster per HPC (High Performance Computing) ed i cluster per HTC (High Throughput Computing).

I cluster per HA sono generalmente composti da due macchine singole ed il loro scopo è di garantire la disponibilità di un servizio nel migliore dei modi. Ne esistono due modelli principali: A-S (Active-Standby) e A-A (Active-Active). Nel primo modello, i servizi sono disponibili inizialmente su un solo nodo e traslati sul secondo nodo solo nel caso in cui il primo non fosse più disponibile.

Nel modello A-A entrambi i nodi eseguono gli stessi servizi ed il carico di lavoro viene bilanciato su entrambi. Esempi di software per la realizzazione di cluster high availability sono Linux Virtual Server⁷ e Linux-HA⁸.

I cluster per HPC sono specializzati nel calcolo ad alte prestazioni e sono particolarmente adatti all'elaborazione di simulazioni scientifiche.

⁷ Linux Virtual Server, è costituito da un server virtuale che riceve le richieste di servizio dagli utenti (o più in generale dai client) e ridistribuisce il carico su un insieme di server reali in modo totalmente trasparente.

⁸ Linux-HA consente di definire un insieme di nodi su cui redistribuire il carico di lavoro ricevuto da diversi servizi (web server, mail server, dns, dhcp, firewall).

Sono formati da un gran numero di nodi la cui presenza è totalmente trasparente all'utente. Il clustering HPC è supportato da numerosi software, proprietari e non; tra questi ultimi particolarmente diffusi sono:

- Mpich⁹
- Torque¹⁰
- Pvm¹¹

High Throughput Computing è il termine con cui si descrive l'utilizzo, per un lungo periodo, di più risorse computazionali al fine di risolvere un problema. Dalla definizione fornita non sembrerebbe esserci molta differenza fra HPC ed HTC; in realtà la differenza è sostanziale.

La capacità elaborativa di un sistema HPC è misurata in termini di FLOP¹² al secondo, mentre quella di un sistema HTC è misurata in FLOPS al mese o all'anno (Livny, 1997).

Nella pratica, un sistema HPC è specializzato nell'utilizzare grandi quantità di calcolo per eseguire un job nel minor tempo possibile;

9 Mpich è un'implementazione libera e portabile dello standard MPI (Message Passing Interface)

10 Torque è l'evoluzione del resource manager OpenPBS

11 Pvm è un software che consente l'utilizzo di macchine eterogenee per eseguire elaborazioni parallele.

12 Floating-Point Operations, Numero di operazioni in virgola mobile. I moderni microprocessori sono dotati di una componente FPU (Floating Point Unit) specializzata nel elaborazione di operazioni in virgola mobile. Le prestazioni di un calcolatore sono relazionate alla capacità di tale unità sebbene la valutazione della FPU non è un indicazione precisa della potenza della CPU. Per tale motivo si utilizzano benchmark standard rilasciati dalla SPEC (Standard Performance Evaluation Corporation).

un sistema HTC, invece, utilizza grandi risorse di calcolo per risolvere il maggior numero possibile di problemi nel lungo periodo di tempo.

Tra i software più diffusi specializzati nella realizzazione di cluster HTC, troviamo Condor e Globus Toolkit (specializzato in ambienti Grid).

La tecnologia più diffusa nella realizzazione di sistemi cluster prende il nome di Beowulf¹³. E' stata sviluppata da Donald Becker presso la NASA, nel 1994 con lo scopo di fornire ad aziende, istituti di ricerca e università sistemi a basso costo per il calcolo intensivo.

Nella realizzazione di un cluster di tipo Beowulf, bisogna rispettare diverse caratteristiche:

- i nodi sono dedicati al cluster e non utilizzati per nessun altro scopo
- la rete di interconnessione fra i nodi dev'essere completamente dedicata al cluster
- i nodi sono normali computer o workstation a basso costo
- i nodi eseguono software open source
- il cluster è utilizzato per HPC

¹³ Poema epico in lingua anglosassone che prende il nome dall'eroe della storia.

- i nodi eseguono sistemi operativi linux
- esiste un nodo speciale, spesso chiamato head node o master node, interconnesso agli altri nodi attraverso una rete privata, ed all'esterno, su una rete pubblica. Il master node si occupa di offrire servizi ai node slave, come ad esempio spazio di memorizzazione condiviso
- i nodi, generalmente, sono omogenei (stessa cpu, stessa scheda madre, stessa ram, stesso disco)
- i nodi, generalmente, eseguono una sola elaborazione per volta.

Un cluster di tipo beowulf dev'essere supportato da librerie e strumenti per la parallelizzazione del software come ad esempio Mpi (Message Parsing Interface) e PVM (Parallel Virtual Machine).

Capitolo II

Componenti Di Un Cluster Beowulf

Consideriamo Beowulf un'architettura a multicomputer che può essere utilizzata per eseguire calcoli paralleli. E' un sistema normalmente composto da un nodo server e da uno o più nodi client, connessi tra loro utilizzando connessioni Ethernet; componenti hardware comuni e facilmente reperibili (come qualunque PC che possa far girare il sistema operativo GNU/Linux), normali adattatori di rete Ethernet ed apparati di commutazione di rete di tipo switch. Di solito non contiene alcun componente hardware speciale ed è facilmente realizzabile. Beowulf utilizza, inoltre, software comune, come il sistema operativo GNU/Linux, e Parallel Virtual Machine (PVM) o Message Parsing Interface (MPI) per l'implementazione di software parallelo. Al nodo server, d'ora in poi chiamato master node, sono demandati compiti di controllo su tutto il cluster, mentre i nodi client sono dedicati unicamente all'elaborazione massiccia di dati. Nel caso di sistemi Beowulf di notevoli dimensioni, possiamo avere anche più di un nodo server e possibilmente altri nodi

dedicati ad assolvere compiti particolari, come ad esempio stazioni di login, di monitoraggio e di storage.

I nodi client possono essere macchine complete e individualmente funzionali ma anche sistemi privi di unità di memorizzazione. In questo caso, i nodi sono detti diskless e per poter funzionare dipendono totalmente dal nodo master. Nella maggior parte dei casi, i client sono sprovvisti di tastiere e monitor e per accedervi si devono utilizzare strumenti come il login remoto. Un nodo Beowulf, pertanto, può essere pensato come un pacchetto composto da CPU, memoria ed interfaccia di rete, che può essere inserito nel cluster, proprio come una CPU oppure un modulo di memoria può esserlo all'interno della piastra madre di un personal computer. Da questa riflessione si evince che Beowulf non è un pacchetto software, nè un tipo di rete, nè una particolare implementazione del kernel di GNU/Linux, bensì una tecnica di clustering per realizzare supercomputer paralleli virtuali.

Possiamo definire due tipologie di cluster Beowulf:

- Beowulf di Classe 1 : costituiti da macchine i cui componenti sono comuni e di facile reperibilità. Questa caratteristica garantisce prezzi di acquisto e manutenzione molto bassi.

- Beowulf di Classe 2: costituiti da macchine i cui componenti sono specializzati per alte prestazioni ed il cui costo di acquisto e manutenzione è generalmente molto più alto.

La scelta di una delle due tipologie dipende esclusivamente dalle prestazioni che si vogliono raggiungere e dal budget a disposizione.

II-1 Caratteristiche Hardware

Le componenti critiche in un cluster Beowulf sono: la velocità del processore, la velocità della rete, la quantità di memoria ram e velocità e capacità dei dispositivi di memorizzazione. Per il nodo master queste caratteristiche sono tutte fondamentali e di norma sono anche richieste performance maggiori rispetto alle stesse sui nodi slave. Il nodo master svolge molti più compiti rispetto ai nodi client e, nel caso partecipi attivamente alle computazioni, la velocità del processore diviene particolarmente critica. Da esso dipende il funzionamento dell'intero cluster, per cui basse prestazioni si ripercuotono sull'intero sistema, rendendolo altamente inefficiente.

La maggior parte del lavoro prodotto da un cluster è destinato ad essere memorizzato su disco, tenendo conto anche del fatto che, normalmente, i

nodi fanno uso del NFS, e che quindi saranno sempre in concorrenza per l'accesso all'unità disco del nodo head. Per il nodo master è quindi preferibile un'unità di tipo SCSI, o un sistema di tipo RAID, ma i classici IDE assicurano comunque un funzionamento accettabile.

Il tempo trascorso nella trasmissione dei dati è tutto tempo sprecato, o che comunque concorre drasticamente alla riduzione delle prestazioni di un supercomputer distribuito. Sono necessari quindi interfacce di rete molto veloci, come ad esempio Gigabit Ethernet, supportate da strumenti di interconnessione (switch molto meglio che hub) ugualmente performanti.

La quantità di RAM è sempre fondamentale in un sistema di calcolo, che si tratti di un supercomputer o di un personal computer. Fra tutti i dispositivi di memorizzazione di un calcolatore (dai velocissimi registri ai lentissimi hard disk), la memoria RAM garantisce un incremento prestazionale notevole, mantenendo un costo relativamente basso. Maggiori quantità di RAM garantiscono ad un sistema multitask l'esecuzione di più processi contemporaneamente, limitando l'accesso al disco¹⁴.

¹⁴ Ogni processo, per essere eseguito, deve essere caricato in memoria RAM. Per garantire l'esecuzione di un numero di processi la cui dimensione supera le capacità della RAM, i processi sono sottoposti a swapping. In un sistema multitask, i processi sono in concorrenza per l'utilizzo delle risorse e ad ognuno di essi vengono assegnati brevi lassi di tempo per utilizzarle. Il meccanismo di swapping consiste nel caricare in memoria, per poi successivamente eliminarlo, solo il processo a cui momentaneamente sono state assegnate le risorse. Nei sistemi attuali, lo swapping è utilizzato in concomitanza con un'altra tecnica, definita memoria virtuale.

Il cluster nasce e vive parallelamente alla disponibilità di una interconnessione fra i singoli nodi. La rete è il collante che tiene insieme tutti i componenti e le funzionalità del cluster e la sua efficienza è fondamentale. L'hardware, necessario per costruire una rete ad alte prestazioni fra i nodi di un cluster, è molto semplice e consiste in:

- interfaccia di rete Ethernet per ogni nodo (Fast Ethernet, Gigabit Ethernet)
- sistema di cablaggio con cavi RJ45 categoria 5/6
- switch layer 2 con porte 10/100/1000 Megabit

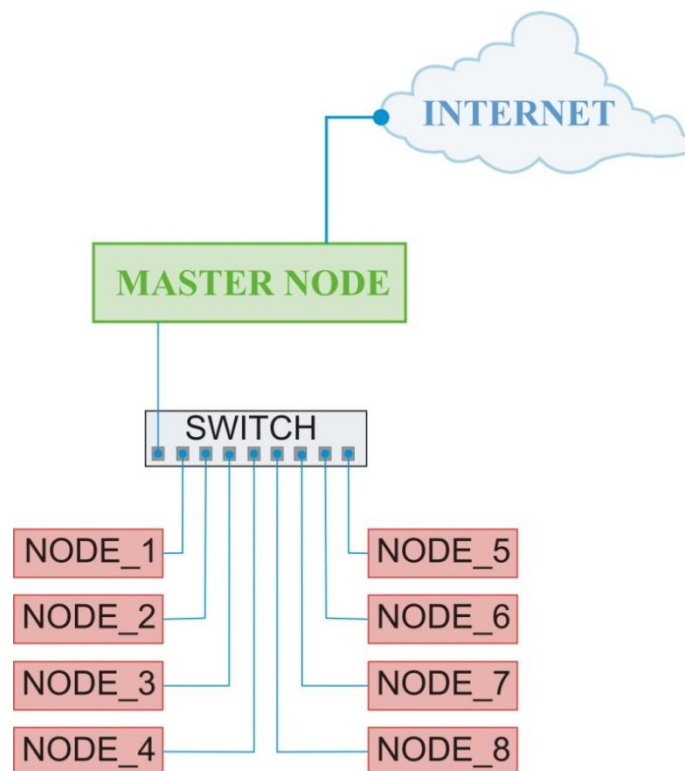


Figura II-1. Composizione Standard Di Un Cluster Beowulf

Il nodo master deve necessariamente essere dotato di due interfacce di rete, la prima dedicata al collegamento con i nodi e la seconda per garantire un collegamento dell'intero cluster con il mondo esterno. In questo senso, il master svolge anche la funzione di gateway per i nodi client.

II-2 Componenti Software

Un cluster beowulf necessita di diverse componenti software e di speciali configurazioni che possiamo suddividere in sei categorie:

- strumenti e servizi per la condivisione
- configurazioni per la comunicazione
- strumenti e servizi per l'account
- strumenti e servizi per l'accesso remoto
- strumenti e servizi per la parallelizzazione
- strumenti e servizi per il monitoraggio

II-2.1 Strumenti e servizi per la condivisione

La creazione di un spazio di memorizzazione condiviso è una delle caratteristiche principali di un cluster e consente di definire posizioni, all'interno del file system globalmente disponibili. Ciò significa che una stessa locazione, e tutti i suoi file, sono sempre disponibili a tutti i nodi del cluster. Ad esempio, una singola installazione di software può essere resa disponibile per l'esecuzione a tutti i nodi. Molto spesso è condivisa anche la directory /home degli utenti che, in questo modo, possono accedere agli stessi dati indipendentemente dal nodo sul quale stiano effettuando un elaborazione.

Uno strumento divenuto ormai standard nei sistemi linux è il protocollo NFS (Network File System). Ideato inizialmente da Sun Microsystems nel 1984, e poi definito dagli RFC¹⁵ 1094, 1813, 3010 e 3530, è un protocollo che consente, a macchine collegate in rete, di “montare” una partizione remota come se fosse locale.

Il Network File System (Tom McNeal, 2002) prevede due componenti fondamentali: il server ed il client, in esecuzione rispettivamente, sul nodo maser e sui nodi slave. La installazione del server nfs sul nodo master si

¹⁵ Un Request for Comments (RFC) è un documento che riporta informazioni o specifiche riguardanti nuove ricerche, innovazioni e metodologie dell'ambito informatico.

effettua attraverso due passaggi: la configurazione di alcuni file e l'avvio del servizio. La configurazione prevede l'utilizzo dei file: `/etc/exports` , `/etc/hosts.allow` e `/etc/hosts.deny`; di questi solo `/etc/exports` è fondamentale per il funzionamento di `nfs`, mentre gli altri sono indispensabili per garantire un minimo di sicurezza al sistema.

Il file `/etc/exports`

Questo file consente di specificare cosa e come dev'essere condiviso e chi può accedere alla condivisione. La sua forma standar è la seguente:

directory [machine([option])]

dove:

directory è la partizione o directory che si vuole condividere;

machine è l'indirizzo ip (o hostname) della macchina abilitata;

option sono una lista di opzioni fra:

- `ro`, per questa macchina la *directory* è condivisa in modalità “sola lettura”

- `rw`, per questa macchina la directory è condivisa in modalità “lettura-scrittura”
- `no_root_squash`, consente all'utente `root` di un client di agire sulla directory condivisa come se fosse l'utente `root` del nodo master. Di default è attiva l'opzione `root_squash`, grazie alla quale, un utente `root` che effettua operazioni sulla directory condivisa viene trattato come se fosse l'utente `nobody`.
- `no_subtree_check`, Se solo una parte di un volume viene esportato, una routine di controllo verifica che il file richiesto dal client appartenga a quel volume. Se l'intero volume viene esportato, disabilitando questa verifica, si potranno accelerare i trasferimenti
- `sync`, con questa opzione un file scritto sul server non è considerato trasferito finchè i dati non siano materialmente scritti su un disco rigido. Di default questa opzione è disabilitata.

```
/opt      node01(ro) node02(ro) node03(ro) node04(ro) \  
         node05(ro) node06(ro) node07(ro) node08(ro)  
/home    node01(rw) node02(rw) node03(rw) node04(rw) \  
         node05(rw) node06(rw) node07(rw) node08(rw)
```

Codice II-1. Contenuto del file `/etc/exports` (specificando tutti i nodi).

Nel caso in cui il cluster abbia molti nodi, si può utilizzare un'altra sintattica con cui, invece di specificare singolarmente ogni nodo, si identifica l'intera subnet del cluster.

```
/opt 10.0.0.0/255.255.255.0(ro)
/home 10.0.0.0/255.255.255.0(rw)
```

Codice II-2. Contenuto del file /etc/exports (specificando la subnet del cluster).

I file /etc/hosts.allow e /etc/hosts.deny

I file `host.allow` e `hosts.deny` consentono di abilitare un minimo di sicurezza nell'accesso ai servizi in esecuzione su un server. Quando un client richiede l'accesso ad un servizio, il sistema controlla le specifiche per quel client e per quel servizio, all'interno del file `host.allow`; se in questo file non viene riscontrata nessuna regola, si passa al controllo del file `host.deny`. Se anche nel file `host.deny` non vi è nessuna regola per il servizio richiesto rispetto a quel client, l'accesso viene consentito. Il file `host.allow` definisce una relazione fra i servizi ed i client che possono accedere a quel servizio, mentre `host.deny` specifica quali client non possono accedere al servizio.

I due file hanno una struttura uguale che può essere riassunta come un elenco di:

```
service: {[ hostname | ip | network/netmask ] ,ALL}
```

Nel file `/etc/hosts.allow`, specificando la riga:

```
sshd: 10.0.0.0/255.255.255.0
```

Si consente a tutti i client, provenienti dalla subnet 10.0.0.0 di accedere al servizio ssh. Volendo disabilitare l'accesso a tale servizio a tutti gli altri client, nel file `hosts.deny` deve essere presente la riga: `sshd: ALL`. Con questa configurazione un client con indirizzo ip 10.0.0.2 riesce ad accedere al servizio ssh, mentre un client con ip 10.0.1.1 non può accedere al servizio.

Il primo servizio il cui accesso dev'essere configurato è `portmap`. Il servizio `portmap` si occupa di mappare le richieste dei client, per i servizi che fanno uso di RPC, sulla porta per cui questi sono in ascolto. Limitare l'accesso al servizio `portmap` è un primo controllo sull'accesso ai servizi. Un client che con conosce la porta di un server non può comunicare con esso. Questa è già un buona configurazione; tuttavia è possibile controllare anche l'accesso ai singoli demoni del Network File System, proteggendosi ulteriormente nel caso in cui un utente malintenzionato riesca ad eludere il

controllo del portmap. Allo stesso modo del servizio portmap, configuriamo in `hosts.allow` e `hosts.deny` i servizi: `lockd`, `rquotad`, `mountd` e `statd`.

```
portmap: 10.0.0.0/255.255.255.0
lockd:   10.0.0.0/255.255.255.0
rquotad: 10.0.0.0/255.255.255.0
mountd:  10.0.0.0/255.255.255.0
statd:   10.0.0.0/255.255.255.0
```

Codice II-3. Contenuto del file `/etc/hosts.allow`.

```
portmap: ALL
lockd:   ALL
rquotad: ALL
mountd:  ALL
statd:   ALL
```

Codice II-4. Contenuto del file `/etc/hosts.deny`

Avvio Dei Servizi

Nfs dipende dal demone portmap che può essere avviato con il comando `/usr/sbin/portmap` o `rpc.portmap`, ed è strettamente necessario che questo servizio sia avviato prima dei servizi NFS. Lavora attraverso 5 demoni che devono essere avviati rispettando questo ordine: `rpc.mountd`, `rpc.nfsd`, `rpc.statd`, `rpc.lockd` e `rpc.rquotad`.

Tutti i servizi citati devono essere avviati in modo automatico durante il boot della macchina. In tal senso possiamo utilizzare, in caso di sistemi

RedHat-Like e Debian, uno script per init.d (/etc/init.d/nfs) creato con l'installazione del pacchetto nfs-utils. L'avvio automatico dei servizi in init.d può essere gestito con tools, come chkconfig, o manualmente creando/eliminando i link nelle runlevel-directory (/etc/rc.d/rcX.d). L'abilitazione dei client consiste nell'avvio dei servizi portmap, rpc.statd e rpc.lockd, che devono essere avviati, come nel caso del master node, al boot.

Per accedere al file system condiviso, bisogna necessariamente “montarlo” all'interno del file system locale. Tale operazione può essere effettuata manualmente, attraverso il comando mount, oppure automaticamente all'avvio della macchina, dopo aver effettuato le opportune configurazioni. La definizione di un punto di mount si effettua attraverso il file /etc/fstab che presenta un insieme di righe del tipo:

device mountpoint fs-type options dump fsckorder

dove:

- *device* rappresenta il dispositivo a caratteri o a blocchi a cui accedere (nel nostro caso il dispositivo dev'essere indicato come IpServerNFS:/shared/directory)

- mountpoint rappresenta la posizione nel file system locale in cui “montare” la directory
- fs-type è il tipo di file system del dispositivo (Ad esempio: ext, ext2, fat, vfat, proc, nfs)
- options, (Ad esempio: ro,rw,sync,user,suid)
- dump specifica se è necessaria un’operazione di dump
- fsckorder specifica l'ordine con cui fsck controlla i file system (1 per il file system root 2 per gli altri)

```
10.0.0.254:/opt /opt nfs ro,bg,intr,hard 0 0
10.0.0.254:/home/home nfs rw,bg,intr,hard 0 0
```

Codice II-5. Contenuto del file /etc/fstab.

II-2.2 Configurazioni Per La Comunicazione

La comunicazione dei nodi all'interno di un cluster non può che aver luogo attraverso le interfacce di rete. Ogni nodo del cluster deve necessariamente avere un indirizzo ip univoco appartenente ad una delle subnet di ip privati 10.0.0.0 oppure 192.168.0.0.

Di norma l'indirizzo ip è assegnato in accordo con il nome del nodo; nel senso che ad un nodo con nome "node01" verrà assegnato l'indirizzo ip 10.0.0.1 (oppure 192.168.0.1). Per convenzione, l'indirizzo 10.x.x.254 oppure 192.168.x.254 è riservato al nodo master.

L'interfaccia di rete è gestita attraverso il comando `ifconfig`, che ci consente di configurare un'interfaccia specificandone il dispositivo, il nome e tutte le caratteristiche di rete (indirizzo ip, gateway, subnet, netmask).

Per configurare le interfacce di rete al boot, o comunque per avere a disposizione un meccanismo per avviare velocemente un'interfaccia di rete, ci viene incontro un script (tipico dei sistemi RedHat-like) memorizzato con il nome di `network` nella directory `/etc/init.d/`. Lo script gestisce l'avvio delle interfacce di rete che possono essere configurate attraverso un file di configurazione nella directory `/etc/sysconfig/network-script/`.

Le interfacce di rete di tipo ethernet assumono un nome del tipo `ethx`, dove `x` identifica la generica interfaccia, un singolo dispositivo ethernet assumerà il nome `eth0`. I file di configurazione si dovranno chiamare `ifcfg-ethX` ed al loro interno dovranno specificare diverse opzioni.

```
DEVICE=eth0
BOOTPROTO=none
HWADDR=00:11:22:33:44:55
ONBOOT=yes
TYPE=Ethernet
USERCTL=yes
IPV6INIT=no
IPADDR=10.0.0.16
NETMASK=255.255.255.0
GATEWAY=10.0.0.254
```

Codice II-6. /etc/sysconfig/network-script/ifcfg-eth0

Il nodo master, avrà due interfacce di rete:

- eth0, utilizzata per comunicare con i nodi
- eth1, utilizzata per comunicare con l'esterno

Per fissare l'associazione degli indirizzi ip dei nodi, con i loro hostname, è necessario configurare il dns locale attraverso il file /etc/hosts: un elenco di corrispondenze fra indirizzo ip e hostname.

```
127.0.0.1    localhost.localdomain localhost
10.0.0.1    node01.clusterdomain.domain.it node01
10.0.0.2    node02.clusterdomain.domain.it node02
.....
10.0.0.254  node0254.clusterdomain.domain.it node254
```

Codice II-7. /etc/hosts

Il nodo master avrà una riga in più per il suo indirizzo ip pubblico.

Per consentire la comunicazione all'esterno ai nodi slave, il master svolge anche la funzione di gateway. Tale servizio può essere abilitato attraverso un'apposita configurazione del firewall del kernel linux (iptables).

```
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Codice II-8. Comandi per la configurazione del nodo master come gateway del cluster.

Questi due comandi, se inseriti nel file `/etc/rc.local`, verranno eseguiti al boot.

II-2.3 Strumenti E Servizi Per L'Account

Gli account utente sui sistemi linux hanno un ruolo fondamentale, in quanto permettono, insieme al meccanismo dei permessi, di definire in modo preciso a chi appartengono i file e chi è autorizzato ad utilizzarli. Ogni utente ha il completo controllo solo su una locazione particolare del file system: la sua home directory. Risulta quindi immediato all'interno di un cluster un utente deve avere una home directory su tutti i nodi. Tale meccanismo può essere realizzato attraverso due configurazioni: la condivisione delle home-directory attraverso nfs e la creazione, per ogni utente, dello stesso account su tutti i nodi.

La creazione di un utente può essere effettuata attraverso i comandi *useradd <user>* e *passwd <user>*. Dove il primo crea un nuovo account per l'utente <user>, creandone anche la home directory, mentre il secondo gli assegna una password.

La gestione degli account è effettuata attraverso tre file: */etc/passwd*, */etc/group*, */etc/shadow*. Replicando questi file su ogni nodo e condividendo la directory */home*, ogni utente potrà loggarsi con la medesima coppia username-password su qualsiasi nodo del cluster. Ad ogni nuovo utente o alla modifica di uno già esistente questa operazione dev'essere effettuata nuovamente.

Questo meccanismo, sebbene funzionale, non è consigliabile per sistemi che hanno molti account. Nella realtà, si utilizzano servizi specializzati come NIS (Network Information Service) e LDAP (Lightweight Directory Access Protocol), il primo specializzato per sistemi Unix-like ed il secondo valido per la maggior parte dei sistemi operativi.

Server NIS

Le Yellow Page (yp), un altro nome per riferirsi al servizio NIS (Kukuk, 2003), possono essere abilitate attraverso tre demoni: *ypserv*, *yppasswd*

(lato server) e ypbind (lato client). Il primo passo nella configurazione di NIS consiste nella definizione di un dominio di riferimento a cui tutti i client devono afferire. Il dominio del nis è fondamentale, in quanto su una stessa rete possono esserci diversi server NIS, ognuno dei quali gestisce diversi gruppi di utenti. La definizione del dominio nis va effettuata su tutti i nodi, compreso il master, attraverso i comandi:

- `/bin/ypdomainname <nomedominio>`
- `/bin/domainname -y <nomedominio>`
- oppure manualmente, aggiungendo nel file `/etc/sysconfig/network` la macro: `NISDOMAIN=nomedominio`

Attraverso il file `/var/yp/Makefile` (del nodo server), ed in particolare nella direttiva `all`, sono indicate le “mappe” che i client NIS possono importare dal server per la gestione degli account.

```
all: passwd group hosts rpc services netid mail      \  
    # netgrp shadow publickey networks ethers      \  
    # bootparams printcap amd.home auto.master     \  
    # auto.home auto.local passwd.adjunct timezone \  
    # locale netmasks
```

Codice II-9. Direttiva “all” nel file `/var/yp/Makefile`

Ad esempio, rimuovendo la mappa “mail” non verrà esportato il file che contiene gli alias degli indirizzi di posta elettronica (*/etc/aliases*).

Attraverso il file */etc/ypserv.conf* si possono definire delle regole di accesso alle mappe singolarmente per ogni client. Successivamente all'avvio dei server *ypserv* e *yppasswdd* è necessario inizializzare il database delle mappe attraverso il comando */var/yp/lib/ypinit -m*. Ogni qual volta si crea un nuovo utente, tale operazione è sufficiente e necessaria per esportare l'account su tutte le macchine appartenenti allo stesso dominio NIS.

Dal punto di vista del client, l'unico servizio indispensabile per accedere ad un server nis è il servizio *ypbind*. L'unica configurazione necessaria alla funzionalità di *ypbind*, oltre alla definizione del dominio nis nel file */etc/sysconfig/network*, consiste nella specifica del server a cui richiedere il database. Nelle prime versioni, *ypbind* cercava, attraverso un messaggio broadcast, autonomamente un server NIS sulla rete. Per garantire maggiore sicurezza tale funzionalità è stata disabilitata aggiungendo il file di configurazione */etc/yp.conf*, in cui sono memorizzate le informazioni per accedere al server.

```
domain nisdomain server 10.0.0.254
```

Codice II-10. Configurazione del file */etc/yp.conf*.

II-2.4 Strumenti E Servizi Per L'Accesso Remoto

Ogni utente con un account deve poter effettuare l'accesso su ogni nodo e da qualsiasi postazione. Tale funzionalità è offerta da servizi come SSH (Secure Shell) e RSH (Remote Shell).

Quando un utente vuole accedere ad un nodo attraverso ssh o rsh, deve inserire username e password, ma per facilitare gli spostamenti interni di un utente, e soprattutto consentirne il libero movimento da un nodo all'altro di processi parallelizzati, i due servizi possono essere configurati in modo da non richiedere una nuova autenticazione, se si proviene da un nodo del cluster e si vuole accedere ad un altro nodo dello stesso cluster. Rsh è un servizio eseguito attraverso xinet.d e per abilitarlo è sufficiente quindi modificare l'opzione `disable=yes` in `disable=no` all'interno dei file `/etc/xinet.d/rsh` e `/etc/xinet.d/rlogin`. La configurazione di rsh per permettere l'accesso tra i nodi senza richiedere autenticazione si effettua editando il file `/etc/hosts.equiv`. Questo file contiene semplicemente il nome di tutti gli host ai quali è concesso effettuare login.

```
node01
node02
node03
node04
.....
node254
```

Codice II-11. Configurazione del file `/etc/hosts.equiv`.

Ssh è un demone eseguito come servizio di init.d e per l'autenticazione può essere abilitato all'utilizzo di certificati RSA, in modo da non trasferire informazioni di account attraverso la rete. Questo tipo di autenticazione si basa su un sistema di crittografia a chiave pubblica e privata.

Consideriamo ad esempio due host che chiamiamo A e B, e prendiamo in considerazione una operazione di trasferimento di dati effettuata tramite il comando scp. A è l'host che vuole trasferire e B è l'host che dovrà ricevere.

Con il sistema di autenticazione basato su password, verrà richiesta la password, sull'host B, dell'utente al quale si vuole inviare il file. Con il sistema basato sulle chiavi invece, i due host si autenticano attraverso le loro chiavi. L'host B possiede la chiave pubblica dell'host A, genera un numero casuale, codifica il numero attraverso la chiave pubblica di A, ed invia il messaggio all'host A. L'host A, attraverso la sua chiave privata, è l'unico capace di decodificare il messaggio e di estrarre il numero. Inviando tale numero all'host B (che lo riconosce come quello da lui generato) viene confermata l'identità di A e l'autenticazione può avvenire. Le due chiavi possono essere codificate attraverso una password (nello specifico chiamata passphrase), la cui immissione verrà richiesta ad ogni tentativo di connessione ad un host. La richiesta della passphrase è effettuata dall'host che richiede l'accesso e quindi è locale. Tuttavia, definendo una passphrase

non si elimina il problema di effettuare operazioni di scambio dati e di login remoti senza l'immissione di password da parte dell'utente.

L'Abilitazione all'rsa del servizio ssh si effettua attraverso il suo file di configurazione in `/etc/ssh/sshd_config`, ed in particolare abilitando le macro:

RSAAuthentication yes

PubkeyAuthentication yes

AuthorizedKeysFile .ssh/authorized_keys

Ogni utente, al quale si vuole concedere login attraverso questo metodo, deve generare le sue chiavi attraverso il comando : `ssh-keygen -t rsa`. Il comando è interattivo e, durante la sua esecuzione, richiede dove memorizzare le chiavi (di default nella directory nascosta `.ssh` della home dell'utente) e la passphrase per codificarle. Le chiavi generate prendono il nome di `id_rsa` e `id_rsa.pub`.

La chiave pubblica, memorizzata nel file `id_rsa.pub`, dev'essere concatenata nel file `$HOME/.ssh/authorized_keys` (come specificato nella macro di `/etc/ssh/sshd_config`) di tutti gli host destinatari ai quali si vuole accedere.

In un cluster in cui le home degli utenti sono condivise via nfs, ogni utente crea la propria coppia di chiavi e copia quella pubblica nel file `authorized_keys`, senza ovviamente effettuare nessun spostamento fisico di tale file.

II-2.5 Strumenti E Servizi Per La Parallelizzazione

Uno strumento per la creazione di sistemi di calcolo paralleli in ambiente linux sono le MPI (Message Passing Interface) ed in particolare la loro implementazione open source, che prende il nome di MPICH.

L'obiettivo dell'MPI è quello di definire uno standard per la portabilità, l'efficienza e la flessibilità di programmi paralleli che utilizzano, per la comunicazione dei loro processi, lo scambio di messaggi. MPI non è un standard ISO o IEEE, ma si può considerare come uno standard de facto per la scrittura di programmi paralleli su piattaforme HPC. MPICH sono delle librerie che consentono di scrivere programmi paralleli nei linguaggi C, C++ e Fortran. Per consentire la comunicazione fra i processi in esecuzione su nodi diversi, è necessario configurare il login attraverso ssh o rsh, a seconda del metodo che si vuole utilizzare. Il meccanismo di comunicazione da utilizzare può essere specificato in fase di

installazione e modificato a runtime definendo la variabile d'ambiente *RSHCOMMAND=/path/to/command*.

L'utilizzo di mpich avviene attraverso due comandi: mpicc (ma anche mpicxx, mpiCC, mpif77, mpif90) e mpirun, il primo per compilare il programma, ed il secondo per eseguirlo. Il comando mpirun accetta diverse opzioni ma le fondamentali sono:

- *-np <n>* , con cui si specifica il numero di processori/nodi da utilizzare
- *-machinefile <filename>*, con cui si indicano esplicitamente le macchine da utilizzare
- *-nolocal*, con cui si richiede di non utilizzare la macchina dalla quale si esegue il programma

Con la definizione del nuovo standard MPI2, è stata rilasciata una nuova versione di mpich nota con il nome di mpich2. La nuova versione racchiude tutte le funzionalità di mpi1 ed mpi2 ed introduce il concetto di ring. Il ring è un insieme di nodi e processori realizzato eseguendo, su ognuno di essi, il servizio mpd. Ogni utente, prima di eseguire un programma parallelo, avvia il proprio ring, definendo così quanti o quali nodi vuole a disposizione e

successivamente sottometta l'esecuzione del programma. E' possibile anche creare un unico ring che tutti gli utenti possano utilizzare.

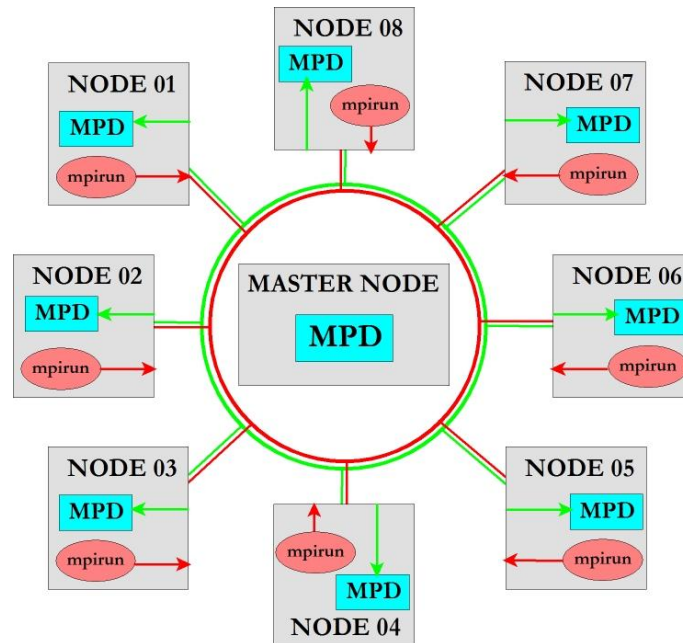


Figura II-1. Rappresentazione grafica di un ring condiviso fra i nodi di un cluster

Un altro strumento molto utile nel calcolo ad alte prestazioni su un cluster è il *resource manager*. Questo strumento permette di gestire le risorse disponibili e schedulare l'esecuzione di job sui vari nodi del cluster.

La versione open source più diffusa prende il nome di Torque, già conosciuto come openPBS (Portable Batch System). Torque fornisce strumenti per il monitoraggio, per lo scheduling e per l'esecuzione di job sui nodi di calcolo di un cluster. Un job sottomesso attraverso Torque può essere specificato attraverso uno script in cui, oltre a definire i comandi da

eseguire, vengono fornite allo scheduler indicazioni su quanti nodi utilizzare e con quali caratteristiche. Attraverso il job descriptor possono essere specificate le caratteristiche hardware e software dei nodi da utilizzare, come ad esempio sistema operativo, memoria disponibile, storage disponibile, software disponibile, e così via. Il funzionamento del resource manager è garantito da tre demoni, chiamati *pbs_server*, *pbs_schedd* e *pbs_mom*. Il primo consente di monitorare e gestire i nodi di calcolo, il secondo si occupa di schedulare i nodi sui quali eseguire, ed il terzo si occupa di eseguire il job.

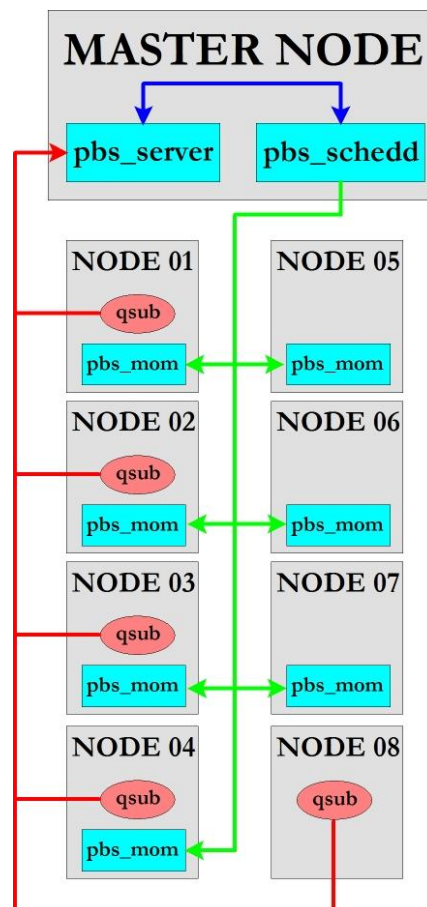


Figura II-2. Rappresentazione grafica dei componenti di torque.

In base a quali demoni siano in esecuzione, possiamo suddividere le macchine in 3 categorie:

- master node (esegue server e scheduler)
- compute node (esegue mom)
- submit node (non eseguono nessun demone, ma possono utilizzare i client per sottomettere e monitorare i job)

Come mostrato dalla figura II-2 i nodi 1-2-3-4 possono sottomettere ed eseguire job, il nodo 8 può solo sottomettere job, ed i nodi 5-6-7 sono solo compute node.

II-2.6 Strumenti E Servizi Per Il Monitoraggio

Il monitoraggio delle risorse in un cluster è una funzionalità fondamentale che permette di controllare lo stato del sistema, l'efficienza del suo utilizzo, ma anche anomalie di funzionamento.

Il software *Ganglia* è specializzato nel monitoraggio delle risorse in ambienti cluster e grid adempiendo molto bene alla realizzazione di tali funzionalità. Il cuore di Ganglia è il demone *gmond* il quale, eseguito come servizio sulla macchina che si vuole monitorare, interagisce con essa,

determinando le risorse totali e disponibili del sistema. Il server Gmond resta in attesa di richieste, rispondendo ad esse con le informazioni raccolte sul sistema attraverso un file xml. Il demone può inviare ad altri gmond in esecuzione, e ricevere dagli stessi, le loro informazioni, fornendo così anche una funzione di aggregazione. Si può definire, così, all'interno di un cluster, un nodo particolare che si occupa di aggregare le informazioni di tutto il cluster in un'unica struttura xml. Utilizzando il demone gmetad, è possibile aggregare in un'unica struttura xml i dati provenienti da più cluster offrendo così un meccanismo di controllo su un intero centro di calcolo.

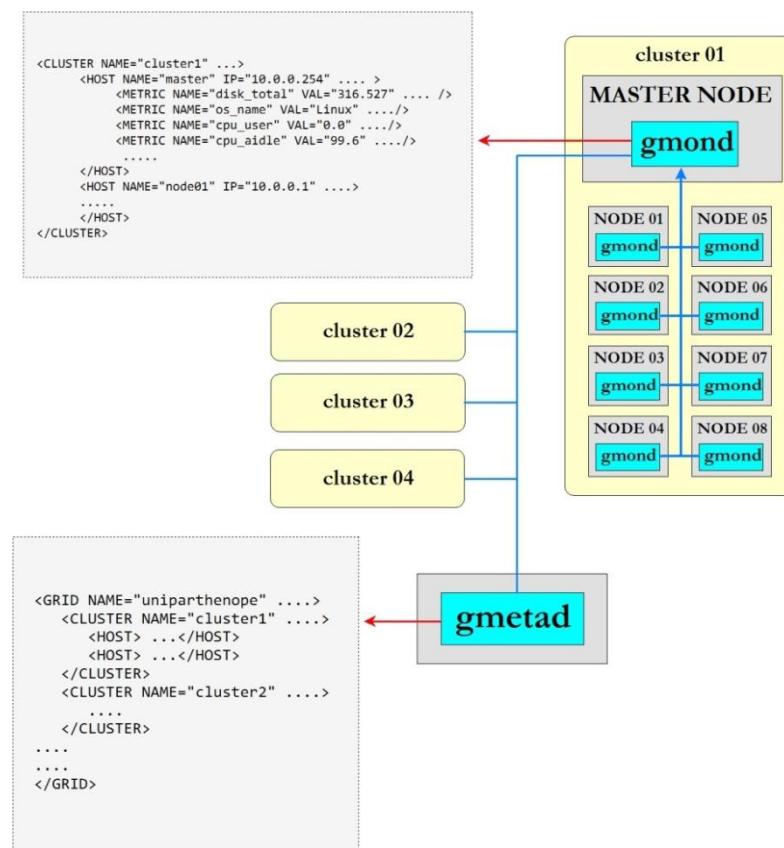


Figura II-3. Rappresentazione grafica del sistema di monitoraggio e di aggregazione.

Capitolo III

Una Metodologia Per L'Installazione Di Un Cluster Beowulf

La creazione di un cluster beowulf, sebbene non particolarmente complicata per un utente esperto, richiede molto tempo per l'installazione e la configurazione di tutti i nodi. La situazione si complica notevolmente se il cluster è di grandi dimensioni o se si devono installare diversi cluster tra loro non omogenei.

Una delle metodologie spesso utilizzate consiste nella clonazione degli hard disk dei nodi. Dopo aver installato e configurato manualmente il primo nodo vengono, in sequenza, installati tutti gli altri, estraendo il loro hard disk ed installandolo come dispositivo slave sul nodo configurato. Dopo aver creato manualmente le stesse partizioni si può avviare la clonazione attraverso il comando:

```
dd if=/dev/hdaX of=/dev/hdbX
```

dove hdaX sono le generiche partizioni dell'hard disk originale e hdbX la stessa generica partizione dell'hard disk da clonare.

Al termine di tale procedura è necessario montare nel file system il dispositivo clonato e modificare manualmente le configurazioni per il nodo a cui esso dovrà appartenere. Questa metodologia consente di installare velocemente i nodi (ad ogni passo i nodi installati raddoppiano), tuttavia c'è da tener conto di tre problematiche:

- durante la clonazione si possono verificare errori nei file che ne compromettono l'installazione
- sono necessarie operazioni di configurazione per ogni nodo (e nell'effettuarle si possono commettere facilmente degli errori)
- ogni nuovo nodo dev'essere smontato per estrarre l'hard disk e poi rimontato

Le distribuzioni RedHat e relative fork¹⁶ (Fedora, Scientific Linux) prevedono l'installazione e la configurazione di una macchina, attraverso un file denominato Kickstart che permette di rispondere a tutte le configurazioni previste in fase di installazione. Il kickstart prevede l'esecuzione di comandi post-installazione fornendo così la possibilità di installare nuovo software o di effettuare modifiche ai file di configurazione.

¹⁶ Il termine fork nel gergo informatico indica la biforcazione di un flusso di elaborazione in due processi separati, che fino a quel momento hanno condiviso istruzioni e dati.

Esso inoltre permette di installare Linux, sia tramite i media tradizionali (cd-rom, hard disk), sia via network (http,ftp,nfs).

Il vantaggio nell'utilizzare kickstart deriva dalla possibilità di:

- Rendere più facili e veloci i processi di installazione.
- Customizzare e rendere omogenee le installazioni.
- Automatizzare le attività di post-installazione.

Dunque risulta essere la scelta ideale se si vuole avere un parco macchine uniforme o se si prevede un considerevole numero di nuove installazioni. Il kickstart può essere inserito in un disco di boot creato ad-hoc, oppure reso disponibile via network.

III-1 Il file kickstart

Il file kickstart è un semplice file di testo contenente un elenco di voci, ognuna identificata da una parola chiave.

Si può creare in diversi modi:

- modificando il file `sample.ks` presente nel cd di documentazione di Red Hat Linux
- utilizzando l'applicazione Kickstart Configurator della RedHat
- utilizzando quello generato da una installazione manuale e memorizzato nel file `/root/anaconda-ks.cfg`
- scriverlo da zero

Il file kickstart è suddiviso in 4 sezioni definite necessariamente in questo ordine:

- *Option section* , in cui sono specificate le opzioni per l'installazione
- *Package section*, in cui vengono identificati i pacchetti da installare

- *Pre-install section*, in cui vengono definiti i comandi da eseguire prima dell'installazione
- *Post-install section*, in cui vengono definiti i comandi da eseguire dopo l'installazione

```
#Option Section
#platform=x86, AMD64, o Intel EM64T
# System authorization information
auth --enablenis --nisdomain=myNisDomain --
nisserver=192.168.0.254
# System bootloader configuration
bootloader --location=mbr
# Partition clearing information
clearpart --none
# Firewall configuration
firewall --disabled
# System keyboard
keyboard it
# System language
lang en_US
# Use CDROM installation media
cdrom
# Reboot after installation
reboot
# SELinux configuration
selinux --disabled
# System timezone
timezone Europe/Rome
# Install OS instead of upgrade
install
# Disk partitioning information
part / --bytes-per-inode=4096 --fstype="ext3" --grow
# Install OS instead of upgrade
install
.....
# Package Section
%packages
*
# Pre-Install Section
# Post-Install Section
%post --interp /bin/bash
/bin/echo "Kickstart Post Install Command" >> /root/post.log
```

Codice III-1. Esempio di file kickstart.

III-2 Installazione Di Un Cluster Mediante Kickstart

Il file kickstart si presta benissimo alla configurazione dei nodi di un cluster beowulf, che sono omogenei e che variano solo per alcune configurazioni (come ad esempio indirizzo ip e hostname).

Nel file possono quindi essere specificate le configurazioni base e nella sezione post-install le personalizzazioni per i particolari nodi. Abbiamo un insieme di file kickstart pari al numero di nodi del cluster, che differiscono solo della fase di post-installazione e solo per alcuni valori.

Gli strumenti necessari e fondamentali per effettuare tale installazione sono:

- una macchina di servizio che offre al nodo gli strumenti di cui ha bisogno:
 - un bootloader
 - il file kickstart
 - i file della distribuzione da installare

- una connessione di rete fra il nodo e la macchina di servizio

III-2.1 Network Boot

Una macchina, per avviarsi, ha bisogno di un immagine di boot e di un bootloader per caricarla; entrambe possono essere fornite attraverso dispositivi locali,(cd-rom, floppy,hard disk) o attraverso la rete. Quest'ultima possibilità è attuabile solo se la macchina in questione permette il boot da rete, caratteristica legata alla presenza di un particolare firmware nel bios.

Il boot da rete è solitamente utilizzato per avviare macchine diskless (che hanno solo processore, ram, scheda di rete e scheda madre) ed è possibile attraverso il Preboot Execution Environment (PXE). Per poter utilizzare il pxe sono necessari i servizi tftp e dhcp ed il bootloader pxelinux.0.

Il Trivial File Transfer Protocol (TFTP) è un protocollo di trasferimento file molto semplice che offre le funzionalità base dell'FTP ed è particolarmente adatto al trasferimento di piccoli file. Relativamente al pxe, è utilizzato per inviare alla macchina richiedente il bootloader.

Il Dynamic Host Configuration Protocol (DHCP) è un protocollo usato per assegnare, in modo dinamico, gli indirizzi IP ai calcolatori di una rete.

Il servizio, che esso offre al pxe, consiste nell'assegnare un indirizzo ip al nodo e nel fornirgli informazioni riguardanti il boot, come l'IP del server che esegue tftp e la posizione del bootloader da richiedere a tale server.

```
allow booting;
allow bootp;

# Standard configuration directives...

option domain-name "domain_name";
option subnet-mask subnet mask;
option broadcast-address broadcast_address;
option domain-name-servers dns_servers;
option routers default_router;

# Group the PXE bootable hosts together
group {
    # PXE-specific configuration directives...
    option dhcp-class-identifier "PXEClient";

    next-server 10.0.0.254;
    filename "/tftpboot/pxelinux.0";

    host node01 {
        hardware ethernet <mac-address>;
        fixed-address 10.0.0.1;
    }
    host node02 {
        hardware ethernet <mac-address>;
        fixed-address 10.0.0.2;
    }
}
```

Codice III-2. Esempio di configurazione del file /etc/dhcpd.conf.

Il bootloader pxelinux.0 fa parte del package syslinux, e di default la sua locazione è in /usr/lib/syslinux/ (oppure /usr/share/lib/syslinux/); in ogni caso si può ottenere tale file dal pacchetto di installazione di syslinux

scaricabile da <http://www.kernel.org/pub/linux/utils/boot/syslinux/>. Tale file dev'essere posizionato nella directory root del tftp.

Il bootloader si occupa di avviare l'immagine di boot; è possibile specificarne le opzioni attraverso un opportuno file. I file di configurazione per l'avvio dell'immagine si trovano nella stessa directory del bootloader ed in particolare nella sottodirectory pxelinux.cfg. Possono esserci diversi file di configurazione, ognuno per una macchina differente, scelti dal pxeboot in base all'indirizzo ip o il mac-address. Supponiamo che la macchina richiedente il boot abbia mac-address 88:99:AA:BB:CC:DD ed indirizzo ip 192.0.2.91 (in esadecimale C000025B) il file selezionato per il boot è determinato secondo il seguente ordine:

- i. /tftpboot/pxelinux.cfg/01-88-99-aa-bb-cc-dd
- ii. /tftpbootpxelinux.cfg/C000025B
- iii. /tftpbootpxelinux.cfg/C000025
- iv. /tftpbootpxelinux.cfg/C00002
- v. /tftpbootpxelinux.cfg/C0000
- vi. /tftpbootpxelinux.cfg/C000
- vii. /tftpbootpxelinux.cfg/C00

- viii. /tftpbootpxelinux.cfg/C0
- ix. /tftpbootpxelinux.cfg/C
- x. /tftpbootpxelinux.cfg/default

Al nostro scopo può bastare il file *default* al cui interno devono essere specificati solo la posizione dell'immagine da caricare ed eventuali opzioni di avvio.

```
kernel vmlinuz
append initrd=initrd.img
```

Codice III-3. File di configurazione del bootloader /tftpboot/pxelinux.cfg/default.

I file *vmlinuz* e *initrd.img* sono contenuti nel cd della distribuzione sotto la directory *images/pxeboot/* ed è necessario copiarli nella stessa directory in cui è memorizzato *pxelinux.0* (ovvero /tftpboot).

III-2.2 Pubblicazione del kickstart

L'utilizzo e la posizione del file *kickstart* sono specificati come opzione del bootloader nel file *pxelinux.cfg/default*. Il file *kickstart* può essere

pubblicato attraverso ftp, http o nfs. Come già detto, ogni nodo differisce dagli altri solo per alcune configurazioni, tutte pressochè legate al suo indirizzo ip ed al suo hostname. Una soluzione consiste nel creare, per ogni nodo, sia il file kickstart che il file di configurazione per il boot, generando così tanti file che differiscono fra loro solo per piccole parti. Ciò comporta, oltre al tempo necessario per la generazione dei file, l'inevitabile introduzione di errori dovuti alle piccole ma numerose modifiche da apportare ai file stessi. Il problema può essere risolto attraverso l'utilizzo di un file kickstart dinamico, che si adatti al particolare nodo a cui è destinato. Questo meccanismo può essere effettuato attraverso l'utilizzo di un web server (apache) e di un linguaggio di scripting lato server, come ad esempio PHP. Quando un nodo richiede al server http il file kickstart questi lo elabora modificando le configurazioni in base al suo indirizzo ip e fornendo così, in modo totalmente trasparente, un file personalizzato per tale nodo. In questo modo non è più necessario creare tanti file kickstart ed altrettanti file di boot, ma un solo kickstart opportunamente modificato con istruzioni php.

```
kernel vmlinuz
append initrd=initrd.img ks=http://10.0.0.254/ks_node.php
```

Codice III-4. File di configurazione del bootloader /tftpboot/pxelinux.cfg/default.


```

.....
$MASTER_NAME="master";
$DOMAIN="uniparthenope.it";
$NODE_NAME="node";
....
# HOSTNAME
/bin/sed 's/^HOSTNAME=localhost.localdomain/HOSTNAME=      \
<?php $LASTDOT=stripos($_SERVER['REMOTE_ADDR'],".");      \
    $nNODE=substr($_SERVER['REMOTE_ADDR'], $LASTDOT + 1);  \
    if ($nNODE < 10){                                     \
        echo $NODE_NAME."0".$nNODE. ". ".$MASTER_NAME.". ".$DOMAIN;\
    }                                                       \
    else {                                                 \
        echo $NODE_NAME.$nNODE. ". ".$MASTER_NAME.". ".$DOMAIN;\
    }?>                                                    \
/etc/sysconfig/network | /usr/bin/tee                      \
/usr/sysconfig/network >> /dev/null

```

Codice III-5. Particolare del kickstart dinamico /var/www/html/ks_node.php.

Con i comandi riportati nell'esempio, inseriti nella sezione %post del kickstart, si modifica il file /usr/sysconfig/network sostituendo alla macro HOSTNAME=localhost.localdomain il valore HOSTNAME=node01.master.uniparthenope.it (nel caso il nodo richiedente il kickstart abbia indirizzo ip 10.0.0.1 oppure 192.168.0.1).

III-2.3 Sorgenti Della Distribuzione

Attraverso il kickstart verrà specificata il tipo di installazione con le opzioni:

- *cdrom*, che indentifica un'istallazione attraverso il cdrom locale
- *harddrive --dir=/path/to/distribution/tree --partition=hda1*,
l'immagine della distribuzione è memorizzata sul disco
- *url --url=ftp://user:passwd@10.0.0.254/path/to/distribution/tree*,
l'immagine della distribuzione è pubblicata da un server ftp
(nell'esempio richiede anche login e password)
- *url --url=http://10.0.0.254/path/to/distribution/tree*, l'immagine
della distribuzione è pubblicata da un server http
- *nfs --server=10.0.0.254 --dir=/distro/path/to/distribution/tree*,
l'immagine è distribuita via nfs

Sfruttando uno dei meccanismi di pubblicazione attraverso la rete le macchine del cluster non avranno bisogno di un dispositivo Cd-rom o (Dvd), costituendo ancora un risparmio sull'acquisto dell'hardware.

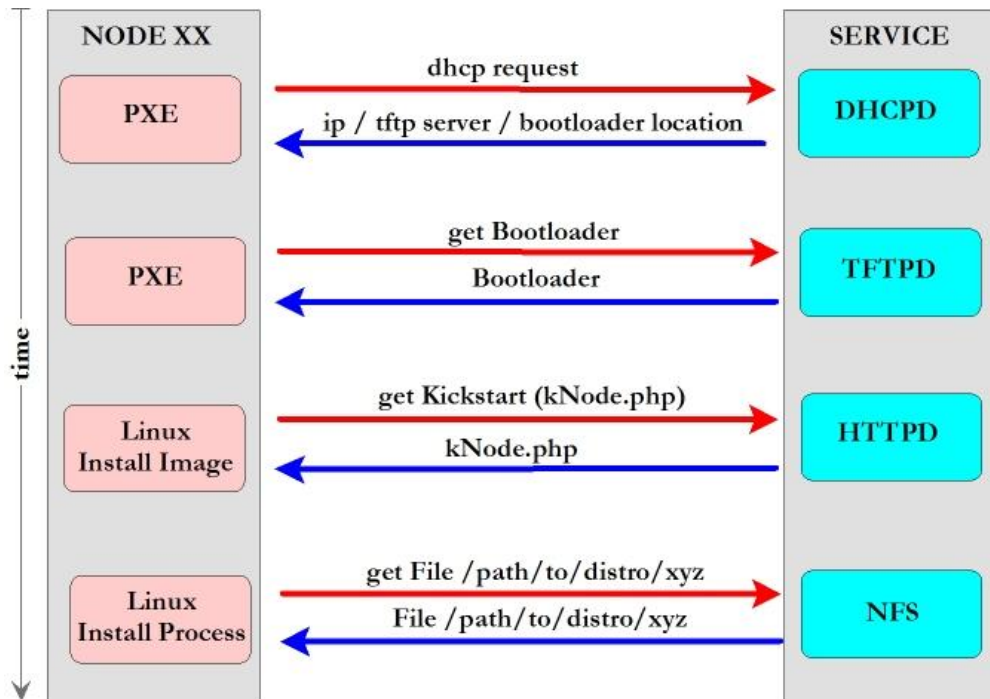


Figura III-1. Sequenza temporale della procedura di installazione attraverso kickstart.

III-3 Software Per La Creazione Automatica Di Cluster

Nell'ambito del clustering sono nati progetti come OSCAR e ROCKS che consentono di ottenere la creazione automatizzata di sistemi cluster HPC in breve tempo. Le tecniche utilizzate da tali progetti sono fondamentalmente diverse ma il risultato ottenuto è essenzialmente lo stesso. Oscar può essere definito come un insieme di applicazioni per la configurazione di un cluster, mentre Rocks è una distribuzione cluster, cioè una vera e propria distribuzione linux.

III-3.1 OSCAR

Il progetto OSCAR (Open Source Cluster Application Resources) consente ad utenti non particolarmente esperti con sistemi linux di installare un cluster HPC di tipo Beowulf. E' composto da due insiemi principali di strumenti:

- System Installation Suite con cui fornisce un insieme di programmi open source per automatizzare l'installazione e la configurazione dei nodi
- C3 (The Cluster Command and Control tool) con cui fornisce un insieme di strumenti a riga di comando per operare dal frontEnd sui nodi di calcolo.

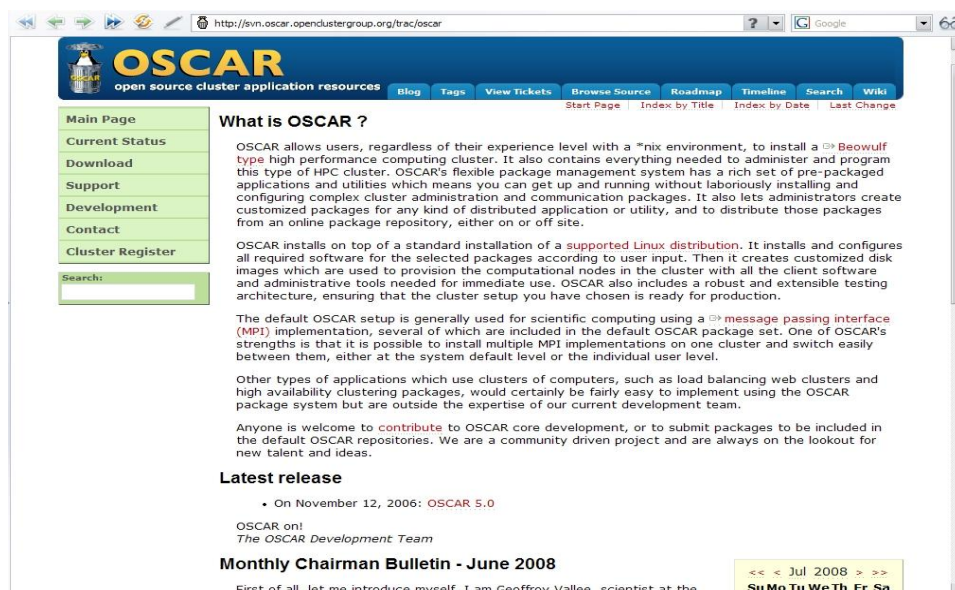


Figura III-2. Home page del progetto OSCAR (<http://svn.oscar.openclustergroup.org/trac/oscar>)

Oscar permette l'installazione di software come MPI, Torque, Maui e Ganglia, tutti precompilati e preconfigurati in modo da risultare immediatamente utilizzabili all'interno del cluster.

La procedura di installazione è effettuata attraverso uno script eseguito sulla macchina frontEnd necessariamente installata con un ambiente X (Gnome o Kde).



Figura III-3. Finestra principale del wizard di configurazione di OSCAR.

Lo script esegue le seguenti operazioni:

- crea le directory di lavoro di OSCAR
- Copia gli rpms nella directory /tftpboot/rpm
- Installa gli rpms dei server e dei tools (PVM,SIS,NFS,DHCP,LAM,C3)
- Aggiorna i file di sistema (hosts, exports, profile, init.d/dhcpd, init.d/nfs)
- Avvia i servizi ed un wizard grafico di configurazione

La procedura di installazione è effettuata attraverso una interfaccia grafica che si occupa di configurare il frontEnd e di generare una immagine con cui verranno installati i nodi slave.

All'avvio dei nodi slave vengono eseguite le seguenti operazioni:

- dal server dhcp vengono presi IP ed HOSTNAME
- da /var/lib/systemimager/scripts viene scaricato lo script corrispondente al nome del client
- Via rsync si preleva ogni utility eventualmente necessaria

- Il disco viene partizionato usando sfdisk in base alle informazioni contenute nell'immagine
- Le nuove partizioni vengono montate su /a
- Il client esegue un chroot /a e via rsync copia tutti i files dell'immagine
- Viene eseguito systemconfigurator per adattare l'immagine al particolare hardware (networking e bootloader)
- Vengono smontati i filesystems e la macchina viene riavviata

III-3.2 ROCKS

Rocks Cluster Distribution (in origine chiamato NPACE Rocks) è una distribuzione linux specifica per cluster HPC. Inizialmente era basato sulla distribuzione linux Red Hat ma le ultime versioni sono basate su CentOS e dotate di una versione modificata di anaconda che semplifica l'installazione di molte macchine. Rocks include diversi strumenti (come MPI) che non fanno parte dell'installazione base di CentOS ma che sono necessari per raggruppare un insieme di computers in un cluster. Durante l'installazione è possibile personalizzare i pacchetti software da includere nel sistema

utilizzando uno o più CD speciali chiamati “Roll CDs” che estendono le funzionalità integrando in modo automatico nuovi strumenti e funzionalità. Attualmente sono disponibili i roll: kernel, os, base, hpc e web-server.

Per installare il nodo frontEnd bisogna avviare la macchina con il cd denominato Kernel/Boot Roll che richiede tutte le normali informazioni di installazione, la selezione dei rolls e le informazioni del cluster.

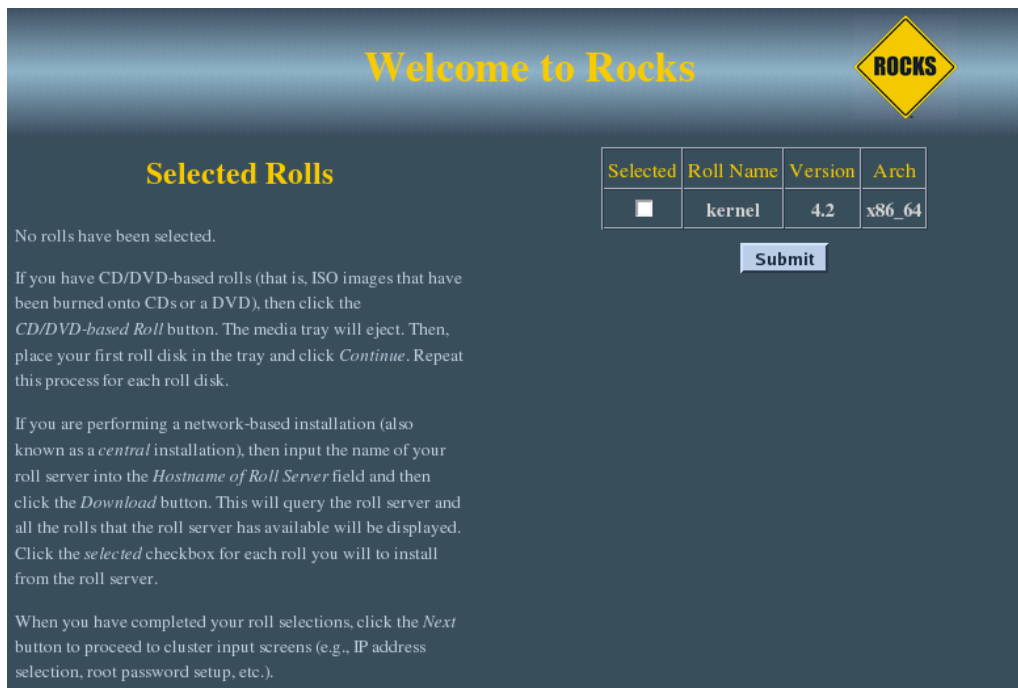



Figura III-4. Selezione dei roll disponibili.

Welcome to Rocks 

Help

Fully-Qualified Host Name:
This must be the fully-qualified domain name (required).

Cluster Name:
The name of the cluster (optional).

Certificate Organization:
The name of your organization. Used when building a certificate for this host (optional).

Certificate Locality:
Your city (optional).

Certificate State:

Cluster Information

Fully-Qualified Host Name	cluster.hpc.org
Cluster Name	Our Cluster
Certificate Organization	SDSC
Certificate Locality	San Diego
Certificate State	California
Certificate Country	US
Contact	admin@place.org
URL	http://www.place.org/
Latitude/Longitude	N32.87 W117.22

[Back](#) [Next](#)

Figura III-5. Informazioni sul cluster.

Per l'installazione dei nodi di calcolo è necessario avviare sul nodo frontend lo script *insert-ethers* il quale si occupa di catturare le richieste dhcp dei client per memorizzarne il loro mac-address ed avviarne l'installazione.

Capitolo IV

Kickstart Cluster Configurator

La caratteristica più importante di un cluster Beowulf, e il propulsore principale della sua diffusione, è il costo contenuto dell'hardware e di quello praticamente nullo del software. Tuttavia, la sua creazione e manutenzione richiede comunque personale altamente specializzato, che a questo punto diviene il costo principale.

Gli studi svolti per questo lavoro di tesi di laurea hanno come obiettivo la realizzazione di strumenti software finalizzati alla creazione e alla gestione di un cluster Beowulf in modo semplice ed intuitivo, tale da permettere a chiunque di poter realizzare e amministrare un cluster in breve tempo. Lo scopo principale del KclusterConfigurator è quello di generare un file di configurazione per automatizzare il processo di installazione di un cluster.

Esso è stato disegnato per seguire le seguenti linee guida:

- rapidità nell'installazione del cluster
- semplicità di utilizzo

- elevata personalizzazione nelle configurazioni
- estendibilità a nuovi tools
- portabilità per distribuzioni unix-like differenti.

L'obiettivo è quello di fornire strumenti in grado di adattarsi in modo flessibile ad esigenze specifiche, consentendo lo start-up del sistema operativo e di tutte le componenti software necessarie alla realizzazione di un cluster di tipo Beowulf, nel modo più semplice, intuitivo e rapido possibile, abbattendo i tempi di lavoro e manutenzione dell'intero sistema.

Il tempo necessario alla creazione del cluster, escludendo quello necessario alla sistemazione dell'hardware, è legato principalmente alle interazioni fisiche con i singoli nodi che, con installazioni standard, divengono inevitabili e numerose.

Con gli strumenti utilizzati è possibile ridurre le interazioni con i nodi a due operazioni :

- primo avvio delle macchine per leggere il mac-address dell'interfaccia di rete
- riavvio delle macchine per iniziare l'installazione

Nel caso in cui le macchine abbiano già un sistema operativo installato, è necessaria una ulteriore operazione che consiste nel modificare, all'interno di bios, la gerarchia dei dispositivi di boot, in modo da impostare la rete come primo device della lista. L'annotazione del mac-address non è indispensabile, ma è l'unica modalità con cui, grazie al dhcp, si può avere una mappatura fisica dei nodi. Questa operazione può essere automatizzata attraverso uno script che cattura le richieste dhcp ricevute dai client. L'unica accortezza necessaria consiste nell'accendere le macchine in successione, attendendo tra l'avvio di un nodo e l'altro, del tempo necessario allo script per il riconoscimento del mac-address.

Lo scenario tipico prevede una macchina linux, definita con il nome di ClusterService, utilizzata per fornire i file ed i servizi necessari ad automatizzare l'installazione su uno o più master.

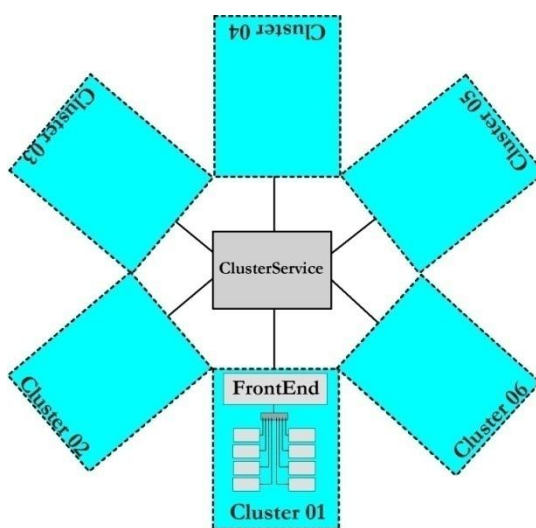


Figura IV-1. Rappresentazione grafica delle connessioni di più cluster al ClusterService.

Sebbene il cluster service possa essere utilizzato per servire anche l'installazione dei nodi, la scelta ideale prevede di configurare ogni master come service per i suoi nodi. Questa possibilità consente di configurare automaticamente il master e di non porre un limite al numero di cluster installabili contemporaneamente, poiché ognuno di essi lavora su una propria rete privata.

Attraverso il `kClusterConfigurator` verranno generati i file di configurazione per ogni cluster che in seguito saranno pubblicati via http o nfs. Nel caso più semplice, ogni cluster avrà due file di configurazione:

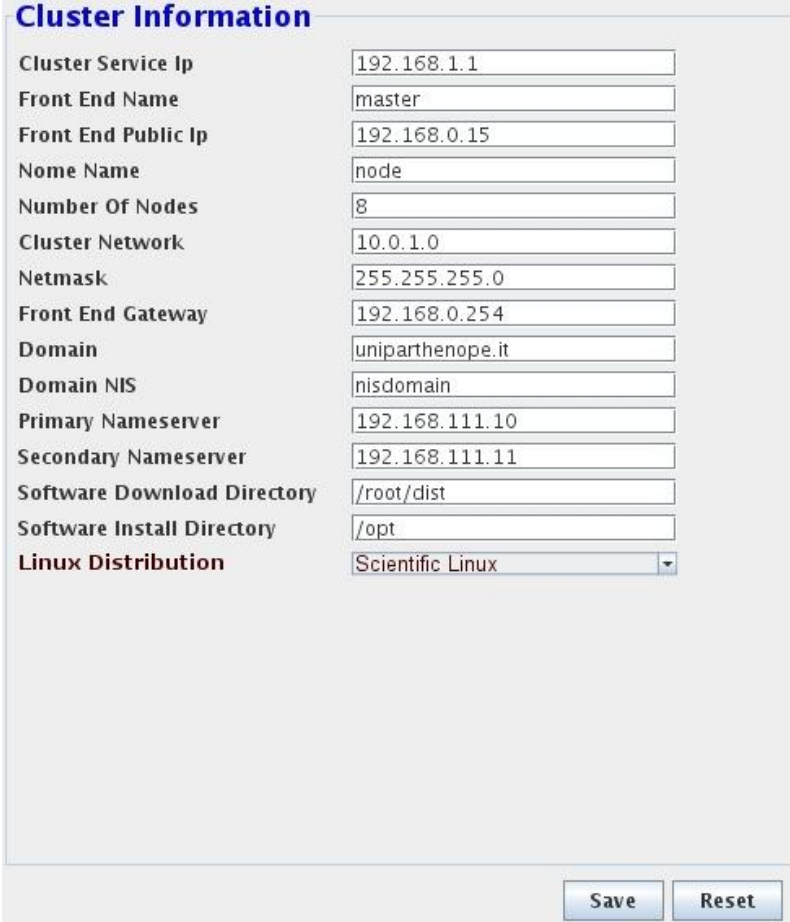
- `kFrontEnd.ks`, per l'installazione del master
- `kNodes.php`, per i suoi nodi

Il nodo master (chiamato anche `FrontEnd`) utilizzerà il `kFrontEnd.ks`, per automatizzare la sua installazione, ed effettuerà localmente una copia del `kNodes.php` per renderla successivamente disponibile ai propri nodi.

Il software progettato vuole essere: uno strumento ideale per l'amministratore di sistema, il quale può gestire facilmente le sue innumerevoli installazioni; ma anche e soprattutto, uno strumento che consenta di raggiungere lo stesso obiettivo ad un utente non particolarmente esperto.

Difatti le informazioni da fornire sono minime e principalmente riguardano:

- la scelta della distribuzione
- informazioni sul cluster
- opzioni di installazione (tipiche di ogni distribuzione linux)
- informazioni legate ai software da utilizzare
- informazioni sulle interfacce di rete delle macchine (mac-address)



Cluster Information	
Cluster Service Ip	192.168.1.1
Front End Name	master
Front End Public Ip	192.168.0.15
Nome Name	node
Number Of Nodes	8
Cluster Network	10.0.1.0
Netmask	255.255.255.0
Front End Gateway	192.168.0.254
Domain	uniparthenope.it
Domain NIS	nisdomain
Primary Nameserver	192.168.111.10
Secondary Nameserver	192.168.111.11
Software Download Directory	/root/dist
Software Install Directory	/opt
Linux Distribution	Scientific Linux

Save Reset

Figura IV-2. Informazioni di base del cluster.

Tutte le operazioni necessarie alla configurazione del nodo frontEnd e dei nodi di calcolo sono costruite in base a queste informazioni. Se necessario, i file generati possono essere ulteriormente modificati per aggiungere manualmente caratteristiche ed opzioni particolari. Il software non è legato a nessuna distribuzione particolare, ma può essere utilizzato con una qualunque distribuzione che preveda un meccanismo di installazione automatizzato. Alcuni esempi sono:

- kickstart di RedHat
- FAI di Debian
- autoYast di Suse

La possibilità di aggiungere nuove distribuzioni e tools per quelle già esistenti è una delle caratteristiche più importanti di cui si è voluto fornire lo strumento. Questo prerequisito è stato soddisfatto attraverso l'utilizzo di strumenti di larga diffusione come:

- linguaggio di programmazione JAVA
- file di configurazione in XML
- linguaggi di shell scripting (sh,bash,perl)
- linguaggi di web-scripting lato server (php)

L'aggiunta di un nuovo strumento si riduce alla creazione di una classe java e all'aggiunta delle informazioni per caricarla all'interno dei file di configurazione della distribuzione (o delle distribuzioni) per cui esso è definito. Gli utenti meno esperti nell'utilizzo di linguaggi di programmazione ad alto livello come JAVA, possono usufruire di strumenti appositamente creati per l'autogenerazione del codice java, attraverso la definizione di un file xml.

IV-1 Dettagli implementativi

La scelta del linguaggio JAVA per l'implementazione del software è dovuta principalmente alla sua larga diffusione. Ma fattori dominanti sono stati anche l'alta portabilità, gli strumenti di alto livello per lo sviluppo di applicazioni grafiche e l'ottimo interfacciamento con l'XML. Tutto il software è gestito attraverso tre classi principali:

- *kClusterConfigurator.java*.

Costituisce la classe di gestione dello strumento e si occupa di fornire l'interfaccia grafica principale, l'accesso ai tools di configurazione ed il caricamento dinamico delle classi che li definiscono. Ogni distribuzione ha i suoi tools di

configurazione, ognuno definito attraverso una classe java che `kClusterConfigurator` è capace di caricare dinamicamente in base alle scelte dell'utente.

```
.....
try{
  for ( int I = 0; I < toolsPackageList.size(); i++ ){
    Class c = Class.forName((String)toolsPackageList.get(i));
    toolsObjectList[i]=c.newInstance();
    ((Configuration)toolsObjectList[i]).setKclusterConfigurator(this);
    ((Configuration)toolsObjectList[i]).setConfigurationFile((String)toolsConfigFileList.get(i));
    toolsListModel.addString( ((Configuration)configurationObjectList[i]).getName() );
  }
} catch (ClassNotFoundException cnfe){cnfe.printStackTrace();}
} catch (InstantiationException ix) { ix.printStackTrace();}
} catch (IllegalAccessException iae) {iae.printStackTrace();}
.....
```

Codice IV-1. Caricamento dinamico delle classi dei tools nel file `KClusterConfigurator.java`.

➤ *InitConfiguration.java*

E' la classe di configurazione principale e si occupa di gestire l'immissione delle informazioni iniziali del cluster, come la distribuzione da utilizzare, le nomenclature e le caratteristiche di rete. Tali informazioni vengono poi passate alla classe principale e messe a disposizione di ogni tool che li richieda, attraverso metodi pubblici.

➤ *Configuration.java*

E' una classe astratta utilizzata per specificare i metodi essenziali che ogni tool deve implementare per interfacciarsi al software ed essere utilizzato.

IV-2 Supporto Ed Estensione A Sistemi Linux Differenti

La gestione delle distribuzioni e delle classi ad esse appartenenti avviene attraverso diversi file di configurazione in formato xml, memorizzati in una apposita struttura gerarchica nella directory etc del pacchetto software. Il file principale per la definizione delle distribuzioni è etc/distribution.xml, in cui sono definite tutte quelle disponibili e le informazioni necessarie alla classe KClusterConfigurator. Ogni distribuzione è caratterizzata attraverso due elementi:

- nome
- un file di configurazione per le classi java che definisce ed utilizza

```
<distributionList>
  <distribution>
    <name>Scientific Linux</name>
    <classDefinitionFile>
      redhat/redhat java classes.xml
    </classDefinitionFile>
  </distribution>

  <distribution>
    <name>Fedora Core</name>
    <classDefinitionFile>
      redhat/redhat_java_classes.xml
    </classDefinitionFile>
  </distribution>

  <distribution>
    <name>Debian</name>
    <packageConfigurationFile>
      debian/debian_java_classes.xml
    </packageConfigurationFile>
  </distribution>
</distributionList>
```

Codice IV-2. Esempio di file /KCluster/install/path/etc/distribution.xml.

Il secondo file di configurazione fondamentale, specificato nell'element `<classDefinitionFile>`, è necessario per permettere al software di caricare tutte le classi (e quindi i tools) di configurazione della distribuzione. Con tale file si specificano quattro caratteristiche della classe:

- Il nome
- una breve descrizione
- il package a cui appartiene
- il file di configurazione da utilizzare

A disposizione di tutte le distribuzioni ci sono un insieme di classi, definite nel package `kCluster.base`, con cui si gestisce l'immissione delle informazioni di installazione.

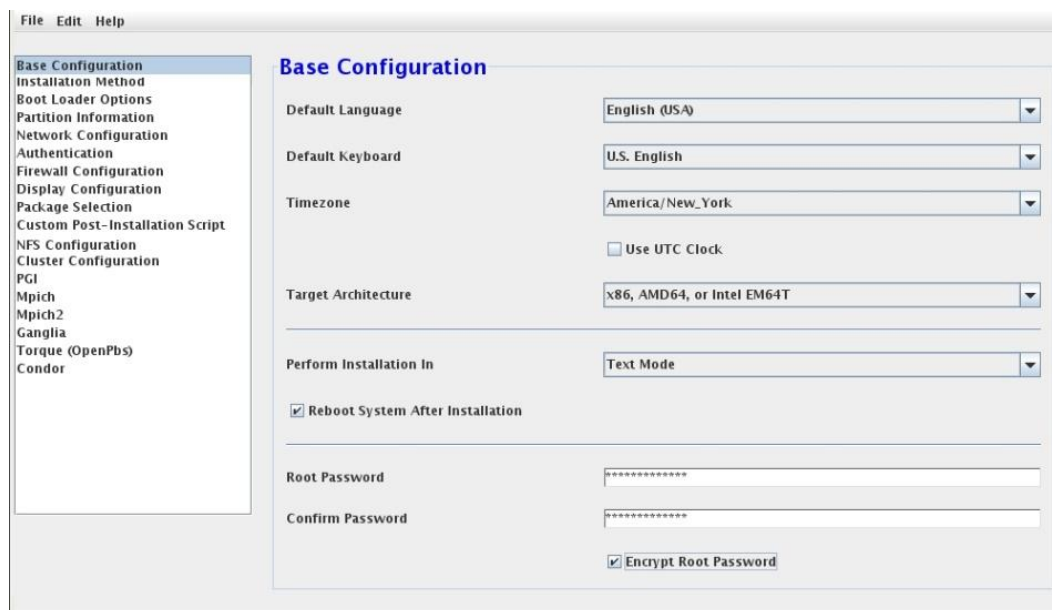


Figura IV-3. Finestra delle configurazioni base dell'installazione

Per includere tutte le classi del package base si può utilizzare, l'elemento `<include>`, oppure selezionarne solo alcune specificandole singolarmente. Con tale meccanismo diverse distribuzioni (di norma quelle appartenenti alla stessa famiglia, come ad esempio ScientificLinux e Fedora), possono facilmente condividere classi e relativi file di configurazione.

```
<classList>
  <include>base/base java classes.xml</include>
  <class>
    <name>PostInstall</name>
    <description>
      PostInstall configuration
    </description>
    <package>kCluster.redhat</package>
    <config>
      scientificLinux/sl_conf_dir/PostInstall.xml
    </config>
  </class>
  <class>
    <name>Mpich-gcc</name>
    <description>
      Mpich installation with gcc compiler
    </description>
    <package>kCluster.redhat.software</package>
    <config>
      redhat/redhat_conf_dir/Mpich-gcc.xml
    </config>
  </class>
  <class>
    <name>Softenv</name>
    <description>Softenv installation</description>
    <package>kCluster.redhat.software</package>
    <config>
      redhat/redhat_conf_dir/Softenv.xml
    </config>
  </class>
</classList>
```

Codice IV-3.

Esempio di file di configurazione delle classi: kCluster/install/path/etc/redhat/redhat_java_classes.xml

Con l'element `<include>` vengono aggiunte tutte le classi del package specificato con gli stessi file di configurazione. Se si vogliono utilizzare file di configurazione diversi, bisogna necessariamente specificare singolarmente le classi.

Con l'element `<classConfigurationFile>` si definisce la directory in cui sono memorizzati i files di configurazione, necessari alle classi che, solo per convenzione, hanno il nome della classe stessa. Con questo meccanismo, distribuzioni diverse possono utilizzare le stesse classi, configurate però in maniera diversa. Le distribuzioni ScientificLinux e Fedora, appartengono entrambe alla famiglia RedHat, condividono tutte le classi, ma in alcuni aspetti di configurazione hanno delle disuguaglianze. Prendiamo ad esempio in considerazione la classe `PostInstall.java` che si occupa di creare il cluster configurando tutti i servizi necessari. Le due distribuzioni possono utilizzare la stessa classe definendo un opportuno file di configurazione.

<pre> <postInstallConfiguration> <hostAllowDeny> <all> <service> Portmap </service> </all> <frontEnd> <service>lockd</service> <service>mountd</service> <service> rquotad </service> <service>statd</service> </frontEnd> </hostAllowDeny> </postInstallConfiguration> </pre>	<pre> <postInstallConfiguration> <hostAllowDeny> <all> <service> rpcbind </service> </all> <frontEnd> <service> rpc.mountd </service> <service> rpc.rquotad </service> <service> rpc.statd </service> </frontEnd> </hostAllowDeny> </postInstallConfiguration> </pre>
---	--

Codice IV-4. Differenze di configurazione fra due distribuzioni (ScientificLinux e Fedora) che utilizzano lo stesso tool (PostInstall.java).

Ogni classe di configurazione implementa i metodi definiti dalla classe astratta Configuration.java ed ognuna di esse fornisce la porzione del file di autoinstallazione relativa al proprio scopo. Il meccanismo di costruzione del file kickstart (o equivalenti) avviene esclusivamente attraverso il file xml di configurazione .

Prendiamo in considerazione la classe KCluster.base.Base, che gestisce informazioni relative alla lingua, alla tastiera, al fuso orario ed altre informazioni di base.

```

<BaseConfiguration>
  <languageList>
    <language>
      <value>en_US</value>
      <name>English (USA)</name>
    </language>
    .....
    <default>en_US</default>
    <kick>lang ${language}</kick>
  </languageList>
  <timezoneList>
    .....
    <default>America/New_York</default>
    <kick>
      timezone ${timezone} [[ (${isutc}) --isutc ]]
    </kick>
  </timezoneList>
  .....
</BaseConfiguration>

```

Codice IV-5. Struttura del file di configurazione della classe Base.java.

L'element <kick>, per ogni opzione di configurazione, definisce le regole per la costruzione del file kickstart. L'interfacciamento con la classe e con i valori immessi dall'utente, viene gestita attraverso i costrutti \${variabile}

che, in fase di elaborazione dell'element `<kick>` vengono sostituiti col valore immesso o scelto dall'utente. Le variabili disponibili appartengono alla classe stessa (e sono specificate attraverso la documentazione) oppure appartengono alla classe principale `KClusterConfigurator`. Nella stringa `<kick>` possono essere specificati anche costrutti di controllo ad esempio con la sintattica `[[(${variabile}) print-to-file]]` la stringa `print-to-file` viene aggiunta al `kickstart` solo se la variabile specificata all'interno dei delimitatori “(“ e “)” assume valore vero o, più in generale, se ha un valore significativo.

```
etc/
distribution.xml
  base/
    base_java_packages.xml
    base_conf_dir/
      Base.xml
      Bootloader.xml
      Partition.xml
      Network.xml
      Authentication.xml
  redhat/
    redhat_java_packages.xml
    redhat_conf_dir/
      PostInstall.xml
      Nfs.xml
      Mpich.xml
      Torque.xml
      Softenv.xml
  scientificLinux/
    scientificLinux_conf_dir/
      PostInstall.xml
      Mpich.xml --> ../redhat/redhat_conf_dir/Mpich.xml
      ....
  debian/
    debian_java_packages.xml
    debian_conf_dir/
      Base.xml
      InstallMethod.xml
      ....
```

Esempio IV-1. Struttura della directory `/KCluster/install/path/etc`.

IV-3 Definizione Di Nuovi Tools

Attraverso l'implementazione di un nuovo tool si può fornire il software di una nuova tipologia di configurazione per il sistema, ma si può anche definire l'installazione e la configurazione di un nuovo applicativo.

La sua realizzazione avviene mediante due operazioni:

- creare una classe java che estende la classe Configuration.java implementandone tutti i metodi.
- informare il sistema della presenza di tale classe attraverso il file di configurazione etc/distro/distro_java_classes.xml.

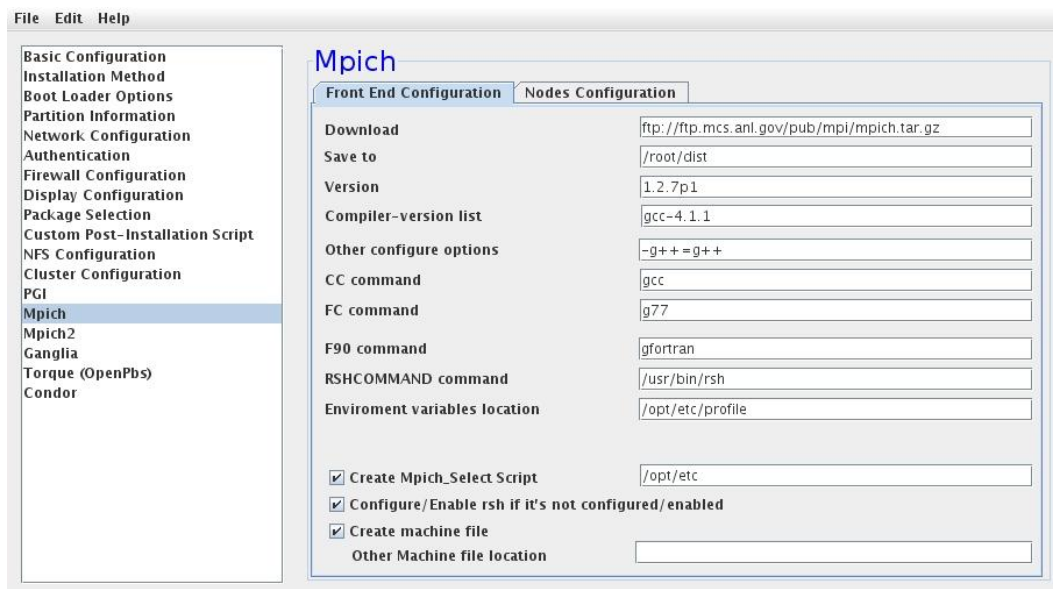


Figura IV-4. Configurazione di Mpich.

La classe Configuration.java definisce diversi metodi astratti:

- *String getName();*

Restituisce un nome descrittivo per la classe, che verrà visualizzato nella finestra di selezione dei tools.

- *void setKclusterConfigurator(KClusterConfigurator kClusterConf);*

Alcuni strumenti hanno bisogno di interfacciarsi alla classe KClusterConfigurator per ottenere informazioni sul cluster (indirizzi ip, nomi dei nodi, dominio di appartenenza, distribuzione linux selezionata). Attraverso questo metodo il tool ottiene un handle della classe di gestione in modo da poter accedere ai suoi metodi pubblici ed ottenere così le informazioni sul cluster.

- *void setConfigurationFile(String xmlFile);*

Con quest metodo, la classe KClusterConfigurator informa il tool sulla locazione del file di configurazione da utilizzare.

- *String getFrontEndKs(); String getNodeKs();*

Questi metodi sono invocati al momento della creazione dei file kickstart e restituiscono la parte di configurazione del tool di competenza.

➤ *void addToPanel(javax.swing.JPanel panel);*

Tale metodo è invocato dalla classe `kClusterConfigurator` quando il tool viene selezionato e consente di visualizzare la sua interfaccia di configurazione

➤ *String checkFrontEndInput(); String checkNodeInput();*

Tali metodi sono invocati prima della creazione del file kickstart (o equivalente), in modo da consentire al tool di effettuare un controllo sui dati immessi dall'utente. Il valore restituito può essere una stringa vuota (ad identificare nessun errore) o una stringa contenente la descrizione dell'errore riscontrato. Al verificarsi dell'errore, la classe `main` seleziona automaticamente il tool in cui esso si è riscontrato, visualizzando la descrizione fornita.

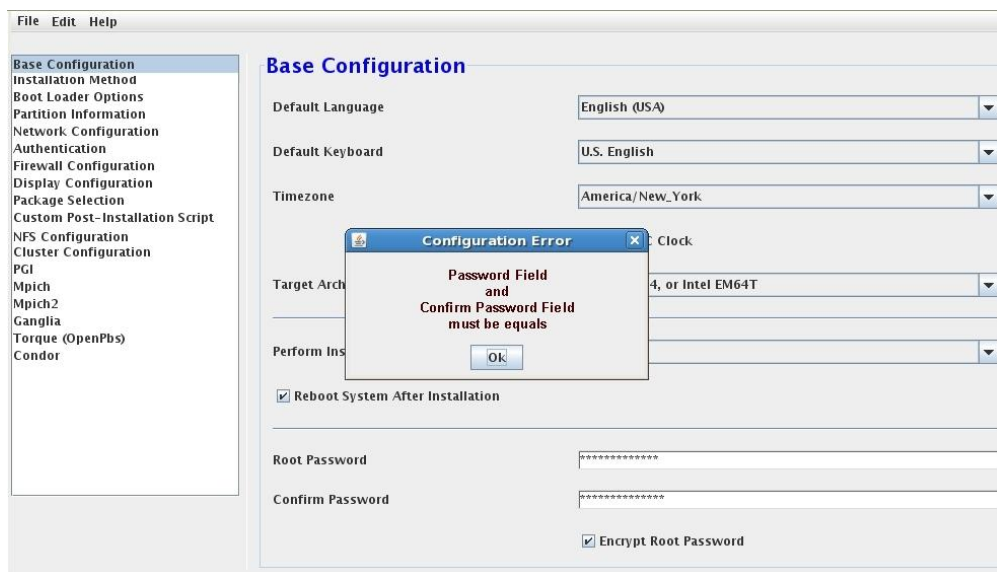


Figura IV-5. Notifica dell'errore nell'inserimento della password di root.

IV-4 Aggiunta di nuovi nodi

La scalabilità dei cluster è un'altra delle caratteristiche principali dei sistemi Beowulf. In ogni momento, si può ottenere un aumento della potenza di calcolo aggiungendo nuovi nodi, in modo totalmente trasparente: per i nodi già presenti, per i software e per gli utenti.

Aggiungere un nuovo nodo ad un cluster, installato utilizzando il Kickstart Cluster Configurator, consiste in due passaggi principali:

- installazione del nuovo nodo, utilizzando il `knode.php` generato al momento della creazione del cluster oppure semplicemente ricreandolo con gli stessi parametri
- riconfigurazione dei nodi già presenti

Quest'ultimo passaggio riguarda principalmente due file di configurazione: `/etc/hosts` e `/etc/hosts.equiv`; il primo per aggiornare il server dei nomi locale di ogni nodo ed il secondo per consentire l'accesso da e verso il nuovo nodo, con `rsh`.

Il Kickstart Cluster Configurator crea script personalizzati per il cluster che consentono queste ed altre operazioni, in modo quasi automatico.

Alcuni esempi di tali script sono:

- `forall.sh` – Consente di eseguire uno o più comandi, in modo sequenziale o parallelo, su tutti, su un gruppo, o su un elenco di nodi.
- `reclustering.sh` – Consente di rigenerare il cluster modificando rapidamente le informazioni statiche dei nodi, come ad esempio la subnet, le directory condivise e gli account utenti.
- `makeNewUser.sh` – Consente di aggiungere un nuovo utente al cluster, semplicemente specificando username e password.

```
#!/bin/bash

if [ -z "$1" -o -z "$2" ]; then
    echo "Usage $0 <user> <password>"
    exit 1
fi

USER=$1
PASSWD=$2
YPINIT="/usr/lib/yp/ypinit"

echo "^M" > auto_enter
useradd $USER
echo $PASSWD | passwd --stdin $USER
$YPINIT -m < auto_enter >> /dev/null

rm -rf auto_enter
echo "CREATE RSA KEYS FOR USER"
echo "/home/$USER/.ssh/id_rsa" > /home/$USER/user_auto_enter
su - $USER -c "ssh-keygen -t rsa < /home/$USER/user_auto_enter" >>
/dev/null
rm -rf /home/$USER/user_auto_enter
su - $USER -c "cp /home/$USER/.ssh/id_rsa.pub
/home/$USER/.ssh/authorized_keys"
exit 0
```

Codice IV-6. Script `makeNewUser.sh`

IV-5 Confronto Con OSCAR E ROCKS

Il Kickstart Cluster Configurator ha lo stesso obiettivo dei sistemi Oscar e Rocks, ma introduce una grande malleabilità nella personalizzazione del sistema e del software installato. Le differenze principali si possono evidenziare in tre caratteristiche:

- portabilità
- personalizzazione
- estendibilità

PORTABILITA'		
ROCKS	OSCAR	KCLUSTER
E' una distribuzione linux basata su CentOS e non prevede quindi l'utilizzo di distribuzioni differenti	E' un insieme di software e strumenti che gestiscono l'installazione e la configurazione del sistema attraverso pacchetti rpm. E' compatibile con: Red Hat, ScientificLinux, Fedora, CentOS e Mandriva.	Basa il suo funzionamento sul file di autoconfigurazione dell'installazione. Tale caratteristica ne permette l'utilizzo con tutte le distribuzioni linux che forniscono simili meccanismi (Red Hat, Scientific Linux, Fedora, Mandriva, Gentoo, CentOS, Debian,...).

Tabella IV-1

PERSONALIZZAZIONE DEL CLUSTER		
ROCKS	OSCAR	KCLUSTER
L'interfaccia grafica consente di specificare il nome del frontEnd e la subnet del cluster.	E' possibile specificare, attraverso la configurazione del servizio dhcp, l'hostname e l'indirizzo ip dei nodi.	L'interfaccia grafica consente di personalizzare tutte le possibili caratteristiche del cluster.

Tabella IV-2

PERSONALIZZAZIONE DEL SOFTWARE		
ROCKS	OSCAR	KCLUSTER
Tutti i software sono precompilati. Consente di selezionare i rolls da installare, ma non i singoli software.	Tutti i software sono precompilati. Consente di selezionare i software da installare, ma non ne consente la configurazione.	I software sono compilati in fase di installazione e questa caratteristica ne consente la totale personalizzazione.

Tabella IV-3

ESTENDIBILITA'		
ROCKS	OSCAR	KCLUSTER
Essendo una distribuzione linux completa, l'estensione a nuovi strumenti necessita di una nuova release.	I software installabili sono messi a disposizione dal team del progetto. Gli utenti possono comunque creare pacchetti srpm personali.	L'aggiunta di nuovi strumenti di configurazione e l'installazione di nuovi software può essere effettuata attraverso la creazione di una classe java. Per strumenti più semplici, è possibile utilizzare uno script per l'autogenerazione della classe attraverso un file xml. Ogni nuovo strumento può essere facilmente condiviso e riutilizzato dalla community di sviluppatori.

Tabella IV-4

Capitolo V

Esempio Applicativo

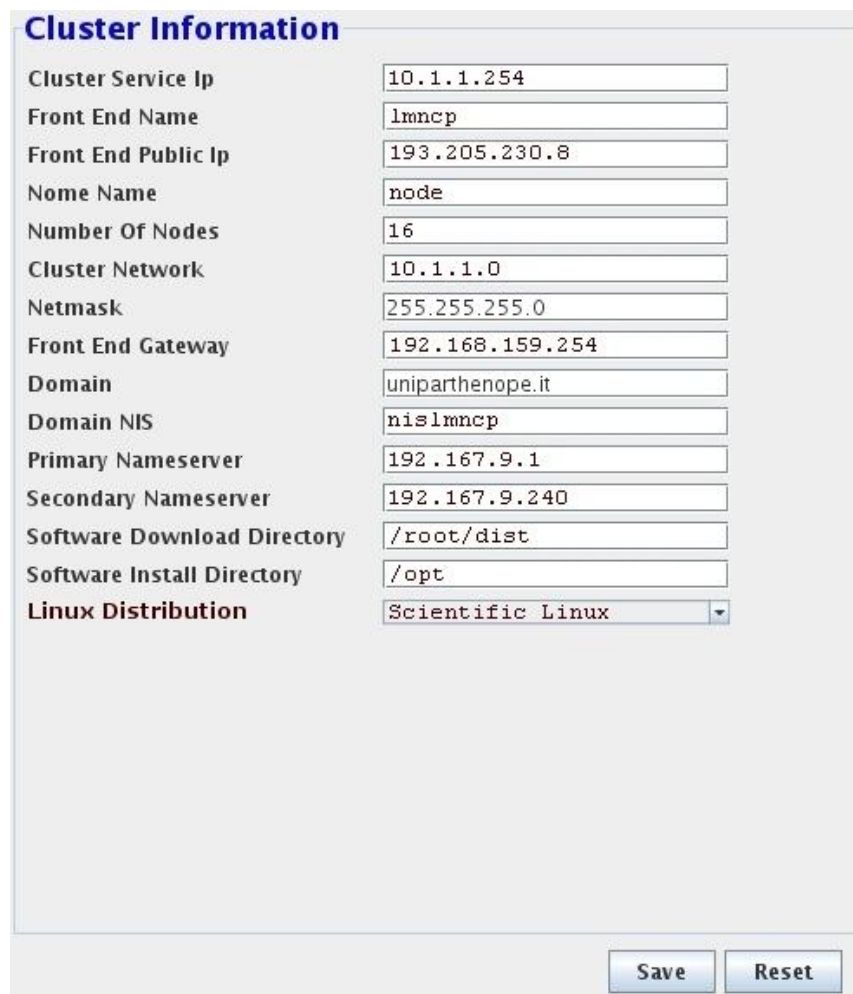
Il Kickstart Cluster Configurator è stato utilizzato con successo presso il dipartimento di Scienze Applicate dell'Università Parthenope per la creazione di due sistemi cluster:

- lmncp.uniparthenope.it
- magi2.uniparthenope.it

Il cluster lmncp (Laboratorio di Modellistica Numerica e Calcolo Parallelo) viene utilizzato a scopo sperimentale dagli studenti della facoltà di Scienze e Tecnologie. E' fornito di 16 nodi di calcolo, ognuno dei quali è una macchina completa, fornita quindi di schermo, tastiera, mouse, dvd-rw, che gli utenti possono utilizzare per avere accesso diretto alle risorse del cluster.

Il cluster lmncp era già presente nel laboratorio, per cui è stato necessario solo l'aggiornamento dei nodi slave che sono stati installati, configurati e

resi completamente attivi in poche ore. In questa circostanza, il nodo master, essendo già attivo, è stato configurato ed utilizzato come ClusterService per i nuovi nodi.



Cluster Information	
Cluster Service Ip	10.1.1.254
Front End Name	lmncp
Front End Public Ip	193.205.230.8
Nome Name	node
Number Of Nodes	16
Cluster Network	10.1.1.0
Netmask	255.255.255.0
Front End Gateway	192.168.159.254
Domain	uniparthenope.it
Domain NIS	nislmcnp
Primary Nameserver	192.167.9.1
Secondary Nameserver	192.167.9.240
Software Download Directory	/root/dist
Software Install Directory	/opt
Linux Distribution	Scientific Linux

Save Reset

Figura V-1. Informazioni di base sul cluster lmncp.uniparthenope.it

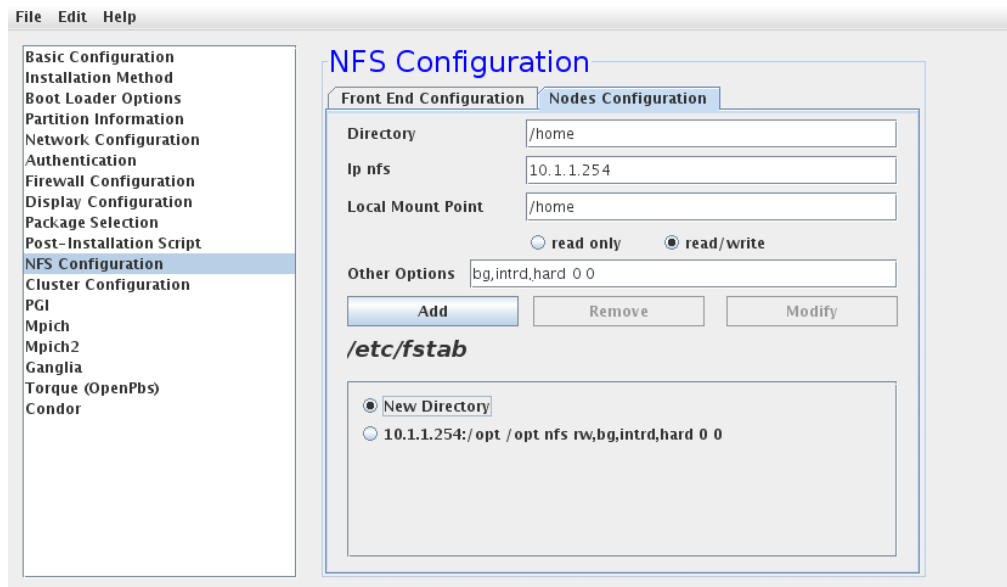


Figura V-2. Configurazione del Network File System.

La realizzazione del cluster avviene principalmente attraverso il tool PostInstall Configuration, che inserisce nel file kickstart tutte le configurazioni necessarie.

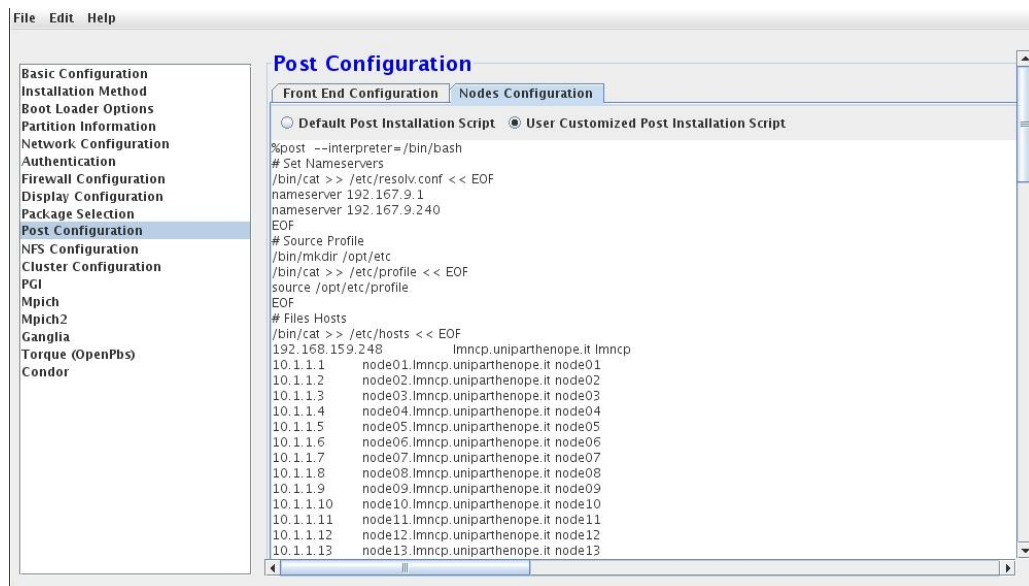


Figura V-3. Comandi eseguiti nella sezione post-install

Il `knodes.php`, oltre a fornire le informazioni di installazione ha configurato le macchine affinché si integrassero nel cluster, rendendole immediatamente funzionali attraverso la configurazione di servizi e strumenti come: `Mpich2` (Codice V-1) , `Ganglia` (Codice V-2), `Torque` e `Condor`.

Le principali operazioni per la configurazione di tali strumenti sono generate automaticamente dal `KCluster` e solitamente non necessitano di nessun input dell'utente (per i nodi slave), in quanto consistono nella creazione di script di avvio e file di configurazione.

```
/bin/cat > /etc/init.d/mpich2 << EOF
#!/bin/sh
.....
MPDSTART=/opt/mpich2-1.0.7r1-pgi-7.0.2/bin/mpd
PORT=55555
RINGHOST=lmncp.uniparthenope.it
PIDFILE=/var/run/mpd.pid # default with --pid option
.....
case "$1" in
  start)
    echo -n "Starting mpd: "
    daemon $MPDSTART --host=$RINGHOST --port=$PORT --pid --daemon
    echo
    ;;
  stop)
    echo "$Shutting down mpd ring."
    killproc -p $PIDFILE
    ;;
  restart)
    $0 stop
    $0 start
    ;;
.....
EOF
```

Codice V-1. Creazione dello script in `init.d` per l'aggiunta del nodo al ring `mpich2` condiviso del cluster

```
%post
.....
# GANGLIA gmond CONFIGURATION
/bin/cat > /etc/gmond.conf << EOF
globals {
  daemonize = yes
  setuid = yes
  user = nobody
  debug_level = 0
  max_udp_msg_len = 1472
  mute = no
  deaf = no
  host_dmax = 0 /*secs */
  cleanup_threshold = 300 /*secs */
  gexec = no
}
cluster {
  name = "lmncp"
  owner = "DSA"
}
host {
  location = "Laboratorio MNCP - DSA"
}
udp_send_channel {
  host = lmncp.uniparthenope.it
  port = 8649
  ttl = 1
}
EOF
```

Codice V-2. Creazione del file di configurazione per il demone gmod di ganglia (/etc/gmond.conf)

V-1 PostFirstReboot

Nella creazione del cluster `lmncp.uniparthenope.it` è stato utilizzato e sperimentato il tool `PostFirstReboot` del `Kickstart Cluster Configurator`. Questo strumento consente di creare uno script, che esegue comandi personalizzati, e permette di eseguire tale script solo al primo riavvio della macchina.

In alcune situazioni, questa funzionalità risulta fondamentale e nel caso specifico è stato utilizzato per l'installazione dei driver della scheda video. I nodi di `lmnbp` sono forniti di una scheda video Nvidia, i cui driver non sono presenti nel kernel linux. L'installazione di tali driver consiste nella creazione dei moduli da caricare nel kernel. Questa operazione è necessaria per ogni kernel che si intende utilizzare.

Essendo disponibili kernel più aggiornati rispetto a quello presente sul DVD della distribuzione, si è optati per un aggiornamento del sistema attraverso repository rpm ufficiali. Prima di compilare i moduli è stato necessario riavviare il nodo, in modo da caricare il nuovo kernel; di conseguenza, al riavvio, i moduli possono essere generati ed utilizzati.

Il funzionamento del `PostFirstReboot` è molto semplice e consiste in tre operazioni:

- Copia di backup del file `/etc/rc.local`
- Creazione dello script `postFirstReboot.sh`
- Aggiunta dell'esecuzione dello script in `/etc/rc.local`

Lo script `postFirstReboot.sh` (Codice V-5), dopo aver eseguito i comandi specificati dall'utente, ripristina la copia backup di `rc.local`, elimina se stesso e, se richiesto, riavvia nuovamente la macchina.

```
%post
.....
/bin/cp /etc/rc.local /etc/rc.local.bkp
/bin/cat > /root/postFirstReboot.sh << EOF
#!/bin/sh
/sbin/init 3
/root/dist/NVIDIA-Linux-x86-169.12-pkg1.run -sNX
/bin/mv -f /etc/rc.local.bkp /etc/rc.local
/bin/rm -f /root/postFirstReboot
reboot
EOF
```

Codice V-2. Script `postFirstReboot` per l'installazione dei driver nvidia.

Capitolo VI

Direzioni Future

Il software sviluppato consente la creazione di cluster Beowulf standard (front end + work node), ma la sua evoluzione dovrà fornire la possibilità di generare cluster, con conformazioni differenti, e specializzati in ambiti anche diversi dall'HPC.

Per adeguarsi all'emergere delle nuove tecnologie legate alla virtualizzazione, il software consentirà di installare sia sistemi reali che virtuali, rendendosi un valido strumento anche per quello che è conosciuto come *Cloud Computing*.

VI-1 Gli Scenari

La possibilità di utilizzare e definire scenari differenti consente di creare facilmente sistemi di qualunque tipologia. È possibile ad esempio installare cluster con particolari nodi di servizio, ma anche cluster di laboratorio e aule didattiche, in cui gli ambienti di lavoro sono già configurati per le

particolari necessità degli utenti. Per consentire tale funzionalità il software dovrà generare file di configurazione diversi per ogni tipologia di nodo, tali file saranno statici nel caso in cui il nodo in questione sia unico, o dinamici (gestiti con script php) nel caso in cui siano necessari più nodi della stessa tipologia.

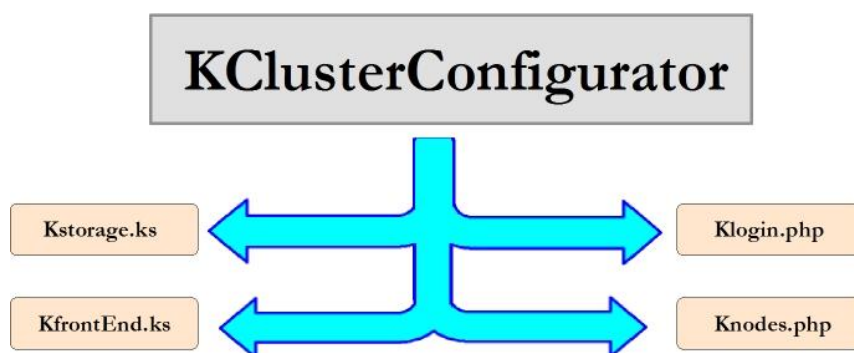


Figura VI-1. File generati dal KClusterConfigurator per uno scenario (frontEnd+workNode+Storage+Login)

L'utente potrà creare il proprio cluster selezionando uno degli scenari disponibili o idearne uno proprio scegliendo le singole componenti ad esso appartenenti e la loro iterazione con gli altri.

Un esempio (figura VI-2) può essere un cluster configurato con nodi di login (per bilanciare il carico di accesso al cluster da parte degli utenti) e nodi di storage (con cui i nodi di un cluster, ma anche di più cluster, possono condividere spazio di memorizzazione e software). A quest'ultimo

scenario si sta già lavorando presso il dipartimento di Scienze Applicate con la creazione del cluster bluejeans.

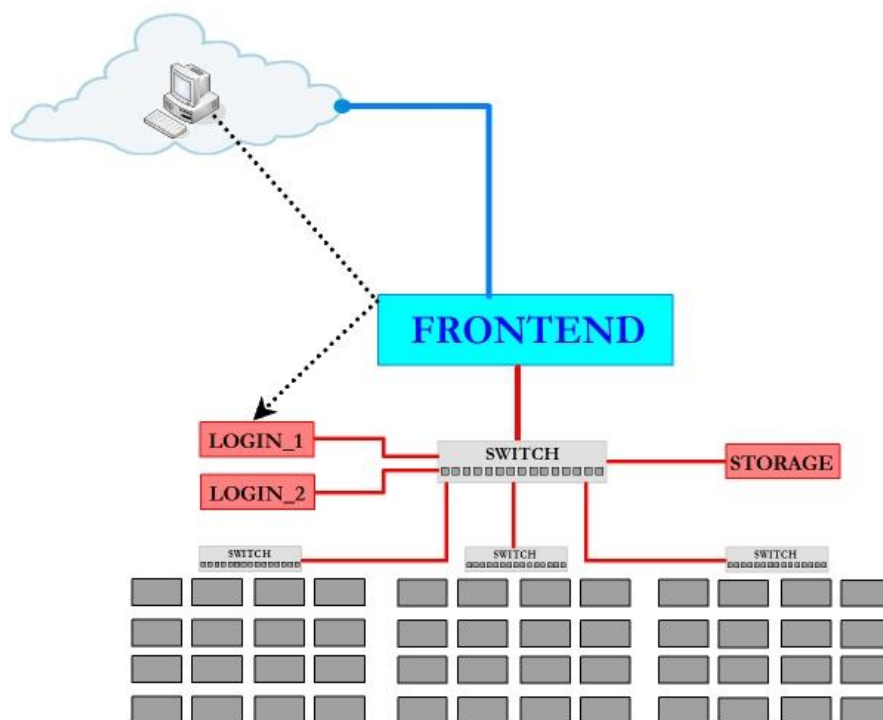


Figura VI-2. Scenario con nodi di servizio (login e storage).

Sarà possibile creare facilmente, con le stesse macchine, scenari differenti che consentiranno di caricare la giusta configurazione del sistema a seconda delle necessità. Prendiamo ad esempio in considerazione un' aula didattica, in cui le macchine hanno la possibilità di caricare un ambiente già pronto per l'esercitazione, a seconda della tipologia di lezione.

VI-2 Cloud Computing e Virtualizzazione

Con il termine Cloud Computing, si identifica un insieme di tecniche e tecnologie informatiche che consentono l'utilizzo di risorse eterogenee e distribuite, come se fossero un sistema singolo e standard. Uno dei primi esempi di cloud computing, sono sicuramente i servizi Google Apps con cui l'azienda di Mountain View mette a disposizione “gratuitamente” ed universalmente le comuni applicazioni “Office”. In questo caso, la risorsa offerta è un'applicazione software che può essere totalmente sostituita alle versioni standard installate sui propri personal computer. Tali tecnologie renderanno i nostri computer (ma anche palmari, telefonini ed ogni strumento capace di interfacciarsi alla rete) dei semplici punti di accesso alle risorse presenti sul web.

Il cloud computing si può considerare come l'evoluzione del web 2.0 che, unitamente alla diffusione capillare sul territorio delle linee di trasmissione dati a banda larga, ha generato un eccezionale boom della rete globale, la quale è passata da un “semplice” strumento per la condivisione delle informazioni, ad una più complessa tecnologia capace anche di generarle e gestirle. L'obiettivo del cloud computing è quello di fornire, sotto forma di servizio web, qualsiasi tipologia di risorsa che sia essa un

software, un servizio di backup dati, un insieme di processori o un intero server, tutto completamente virtualizzato.

La virtualizzazione consente l'astrazione completa di una risorsa, normalmente fornita fisicamente, su di una virtuale, adatta allo scopo. Ad esempio, con una singola risorsa hardware la virtualizzazione consente l'esecuzione contemporanea di più sistemi operativi completamente indipendenti l'uno dall'altro. Tali sistemi, essendo completamente indipendenti dalla risorsa hardware che li ospita, possono facilmente ed in modo trasparente essere migrati da un sistema reale ad un altro, rendendoli particolarmente adatti al cloud computing.

Attraverso queste tecnologie, è possibile generare cluster virtuali che possono essere eseguiti su altri cluster (reali) o sulle risorse totalmente eterogenee, messe a disposizione di un fornitore esterno.

Un possibile scenario potrebbe coinvolgere l'insieme dei computer presenti all'interno di un dipartimento universitario, o negli uffici di una azienda. Tali macchine lavorano, nella migliore delle ipotesi, solo per otto ore al giorno e per non più di cinque giorni a settimana, restando inattive per più del 76% del tempo. Una delle soluzioni potrebbe prevedere la presenza, su ognuna di queste macchine, oltre al sistema reale utilizzato dall'impiegato, un sistema virtuale che configuri la macchina come

appartenente ad un cluster, in modo da poter essere sfruttata durante il periodo di inattività per l'esecuzione di software paralleli. Software specializzati effettuano un monitoraggio della macchina reale, attivando o disattivando quella virtuale, a seconda che un operatore la stia utilizzando o meno.

Dotare il Kickstart Cluster Configurator di strumenti per la creazione di cluster virtuali diviene una delle prerogative più importanti.

Conclusioni

I cluster possono fornire una grande quantità di potenza di calcolo a cui tutti gli utenti vorrebbero e dovrebbero aver accesso.

La diffusione di una tecnologia è direttamente proporzionale alla sua semplicità. Il raggiungimento di tale obiettivo rende necessaria la definizione di strumenti che ne consentano l'utilizzo ad un insieme di persone sempre più esteso. La semplicità con cui lo strumento ideato consente di creare e gestire in modo intuitivo e veloce sistemi di calcolo ad alte prestazioni, utilizzando esclusivamente hardware a basso costo e software open source, permette di divulgare una tecnologia, ormai consolidata ed efficiente come il clustering, ad un gruppo di utenza che non ne avrebbe altrimenti la possibilità (se non dopo aver effettuato un percorso di formazione specifico). In questo modo, le aziende, gli istituti di ricerca e le università possono dotarsi rapidamente di tali sistemi, senza la necessità di investire in personale altamente specializzato o in consulenze esterne, potendo così beneficiare dei conseguenti risparmi economici.

Utilizzando linguaggi di programmazione e tecniche di larga diffusione, sono stati creati strumenti per automatizzare tutte le operazioni necessarie alla realizzazione di un cluster ed alla configurazione del software per il calcolo ad alte prestazioni. Le scelte effettuate in tale ambito hanno consentito di creare uno strumento altamente e facilmente estendibile, rendendolo particolarmente adatto ai miglioramenti offerti dalla crescente comunità dell'Open Source.

BIBLIOGRAFIA

A. Murli - *Lezioni di Calcolo Parallelo* – 2006
Liguori Editore Srl ISBN:8820738198

M.J. Flynn, K. W. Rudd - *Parallel Architectures* -1996
Computer Systems Laboratory Stanford University
<ftp://arith.stanford.edu/tr/rudd.acm396.ps.Z>

M. Livny – *High Throughput Computing: An Interview with Miron Livny*
Intervista di A. Beck – 27 giugno 1997
<http://www.cs.wisc.edu/condor/HPCwire.1>

M. Tartamella, M. Sajeva, B. Vassallo
Amministrazione avanzata Di Server Linux - 2004
Springer ISBN:8847002346

G. C. Fox, R. D. Williams, P. C. Messina
Parallel Computing Works – 1994
Tratto da <http://www.npac.syr.edu/copywrite/pcw/BOOK.html>

What makes a cluster Beowulf?
Tratto da <http://www.beowulf.org/overview/index.html>.

T. McNeal – *Linux NFS-HOWTO* – 2002
Tratto da <http://tldp.org/>: <http://tldp.org/HOWTO/NFS-HOWTO/>

T. Kukuk – *The Linux NIS(YP)/NYS/NIS+ HOWTO* – 2003
Tratto da <http://www.linux-nis.org/nis-howto/HOWTO/NIS-HOWTO.html>