IMPLEMENTAZIONE CLUSTER DI CALCOLO PRESSO INAF/IASF Bologna

A.Tacchini,A.DeRosa,M.Genghini

IASF-Bologna Report Interno, 559/2009
Dicembre,2009

Sommario

Ridefinizione cluster hpc

Per aumentare le prestazioni del cluster HPC (High Performance Computing) presente all'istituto IASF di Bologna si è deciso di rivederne l'impostazione e la struttura.
Sono state acquistate nuove unità, più moderne e potenti delle vecchie, organizzate in un sistema DELL. Tale sistema è diventato il fulcro per il calcolo parallelo.
Un numero maggiore di nodi di calcolo, e per ciascun nodo, una potenza di elaborazione superiore sono stati i cambiamenti principali.
Sul fronte del collegamento tra i nodi si è deciso di mantenere il protocollo Ethernet, per ragioni di costi e compatibilità con il resto dei sistemi di istituto, passando all'uso di switch dotati di 24 porte a 1 Gb/s, con possibilità di passare agevolmente anche ai 10 Gb/s in futuro.

Sistemazione vecchio cluster

Per armonizzare in una unica entità il vecchio cluster con le nuove unità si è operato sul fronte hardware e software.
Per quel che concerne l'hardware si è effettuato un ampliamento della memoria RAM su ognuno dei vecchi nodi (node1….node4), portandola a 8 Gigabyte.
Per il software invece si è provveduto ad installare, sempre su ognuno dei vecchi nodi, una versione aggiornata di Scientific Linux a 64bit.
Si è scelto di utilizzare la versione a 64 bit del sistema operativo per essere in grado di sfruttare appieno la potenza dei processori Intel a 64 bit.
Per conservare la compatibilità con le applicazioni a 32 bit è stato deciso di mantenere disponibili le versioni a 32 bit di librerie e moduli.

GPFS

Il GPFS (General Parallel File System) è un File System sviluppato dall'IBM per essere usato nei sistemi di tipo Cluster.
E' un File System di Tipo Clustered ovvero viene montato contemporaneamente su diversi server.
Usato dalla maggior parte dei supercomputer nel mondo il sistema GPFS fornisce elevata velocità nell'accesso ai dati (i dati vengono suddivisi fisicamente su diversi supporti fisici e letti/scritti in parallelo) garantendo al contempo una elevata affidabilità. A questo si unisce una buona flessibilità derivante dai vari tool di amministrazione messi a disposizione ed all'accesso condiviso al File System permesso anche da postazioni remote.
In pratica uno o più nodi del cluster mettono a disposizione dello spazio disco fisico (il che significa una partizione che può essere anche l'intero hard disk) questo è aggregato e messo a disposizione del File System GPFS attraverso l'NSD (Network Shared Disk).
Un File System GPFS è associato ad uno spazio disco fisico ed è identificato da un nome proprio (pe. Scratch1). Possono coesistere diversi File System GPFS contemporaneamente, ciascuno con il

proprio nome e ciascuno montato ed accessibile da una cartella cha ha il suo stesso nome (ie. /Scratch1).
Tutto questo per ogni nodo del cluster.
Quindi è necessario assicurarsi che su ogni nodo venga montato ogni File System GPFS desiderato (non basta aggiungere un nodo al cluster per averlo attivo).

Passi necessari all'installazione del sistema

Il punto di partenza sono state macchine prive del sistema operativo, quindi il primo passo è stato quello di installare un Sistema Operativo Linux, nel nostro caso Scientific Linux 5.2 64bit.
La scelta è caduta su Scientific Linux per la sua compatibilità con Red Hat (ne è un derivato) e per la grande disponibilità di librerie usate nel calcolo scientifico.

Problema

L'installazione di linux è stata forzata a vedere solo 2 Gbyte di RAM altrimenti una incompatibilità del kernel l'avrebbe bloccata. Una volta completata l'installazione il sistema è in grado di vedere la quantità corretta di RAM.

Opzione di installazione : linux mem=2048M

Per farlo funzionare sono stati usati alcuni accorgimenti.
Per prima cosa ci si è assicurati che fossero presenti i sorgenti del Kernel nella directory */usr/src/kernels*. In seguito si è fatto credere al sistema di avere montato una Red Hat release invece della Scientific Linux, questo lo si è ottenuto editando il file */etc/redhat-release* e sostituendo la dicitura "SL Linux" con "Red-Hat".

Altri accorgimenti sono stati:

- assicurarsi che il file */etc/hosts* contenesse indirizzi e nomi di tutti i nodi;
- assicurarsi che l'utente "root" potesse accedere via SSH ad ogni nodo senza bisogno di PASSWORD;
- assicurarsi che i file RPM di GPFS fossero presenti sulla macchina principale usata come server (i.e. tonno);
- assicurarsi che fosse installato il pacchetto kernel-devel;

Per garantire totale accessibilità ai demoni di GPFS è stato necessario disabilitare il Firewall, per questa ragione ogni nodo è stato reso accessibile solo attraverso un Gateway (nel nostro caso è sempre tonno che è dotato di Firewall) in modo da mantenere un certo grado di protezione negli accessi.
Inoltre si è deciso di disabilitare tutti i servizi non utili al funzionamento del Cluster.

A questo punto è bene sottolineare che il funzionamento del sistema GPFS prevede la presenza di uno o più nodi che fungono da SERVER mentre i nodi restanti fungono da CLIENT.
La linea di condotta scelta è stata quella di iniziare le procedure sul nodo SERVER (nel nostro caso tonno) che richiede un maggiore sforzo dato che offre più servizi, ed esportare file e configurazioni agli altri nodi mediante una semplice operazione di copia dello stretto necessario.

Il nodo di nome tonno possiede ora tutti gli strumenti per installare il sistema GPFS.
Per prima cosa è stato necessario modificare il file */usr/lpp/mmfs(src/config/site.mcr* avendo avuto cura di indicare correttamente la distribuzione linux (riga LINUX_DISTRIBUTION), che nel nostro caso era REDHAT_AS_LINUX, e le directory dove era situato il kernel ed i suoi moduli (*/lib/modules/2.6.18-92.1.22.el5/build/include*).
Si è poi proceduto all'installazione del sistema (make InstallImages).
Una volta andata a buon fine l'installazione si è copiata la directory */usr/lpp/mmfs* in tutti gli altri nodi.

Il passo successivo è stato creare il Cluster GPFS e per farlo si è operato sul nodo che funge da server avente a disposizione il pacchetto gpfs.gpl-3.2.1-6.noarch.rpm.
E' stato fatto partire il demone linux che gestisce il GPFS su tutti i nodi, dopodiché si è impartito il comando mmcrcluster specificando il nodo che sarà il server (eventualmente indicando anche il server secondario se lo si volesse presente) e la lista dei nodi che compongono il cluster, tale lista si trova nel file scratch.nodes che è strutturato secondo la semantica richiesta dal comando mmcrcluster.
A questo punto il cluster GPFS è stato creato ma esso non possiede alcuno spazio disco da gestire col proprio File System, è stato necessario fornirgliene uno.
Ricordando che GPFS è un File System distribuito esisterà uno o, verosimilmente, più nodi che metteranno a disposizione dello spazio disco (partizioni o interi Hard Disk detti NSD o Network Shared Disk) per creare un unico spazio accessibile attraverso il File System GPFS.
Tale spazio è stato creato attraverso il comando mmcrnsd a cui è stato passato come parametro il file scratch1.disks che contiene, nel formato richiesto dal comando, la lista delle partizioni da unire.
Infine attraverso il comando mmcrfs è stato creato il File System vero e proprio ed indicato il punto di montaggio.
Il nome indicato come punto di montaggio (uguale a quello del File System GPFS) è univoco ma possono esistere diversi File System GPFS montati su un nodo (/scratch2, scratch3 ecc.).
Il punto di montaggio indicato sarà presente su tutti i nodi come una normale cartella.

Per aggiungere un nodo nuovo(che per semplicità non condivide nessuno spazio disco fisico) va usato il comando mmaddnode seguito poi da mmstartup per far partire i demoni gpfs sul nodo.
Infine va eseguito il montaggio del File System con mmmount.
Per verificare i File System presenti su un nodo è possibile sia usare il comando mmlsfs che visualizzare il file fstab.

Configurazione attuale

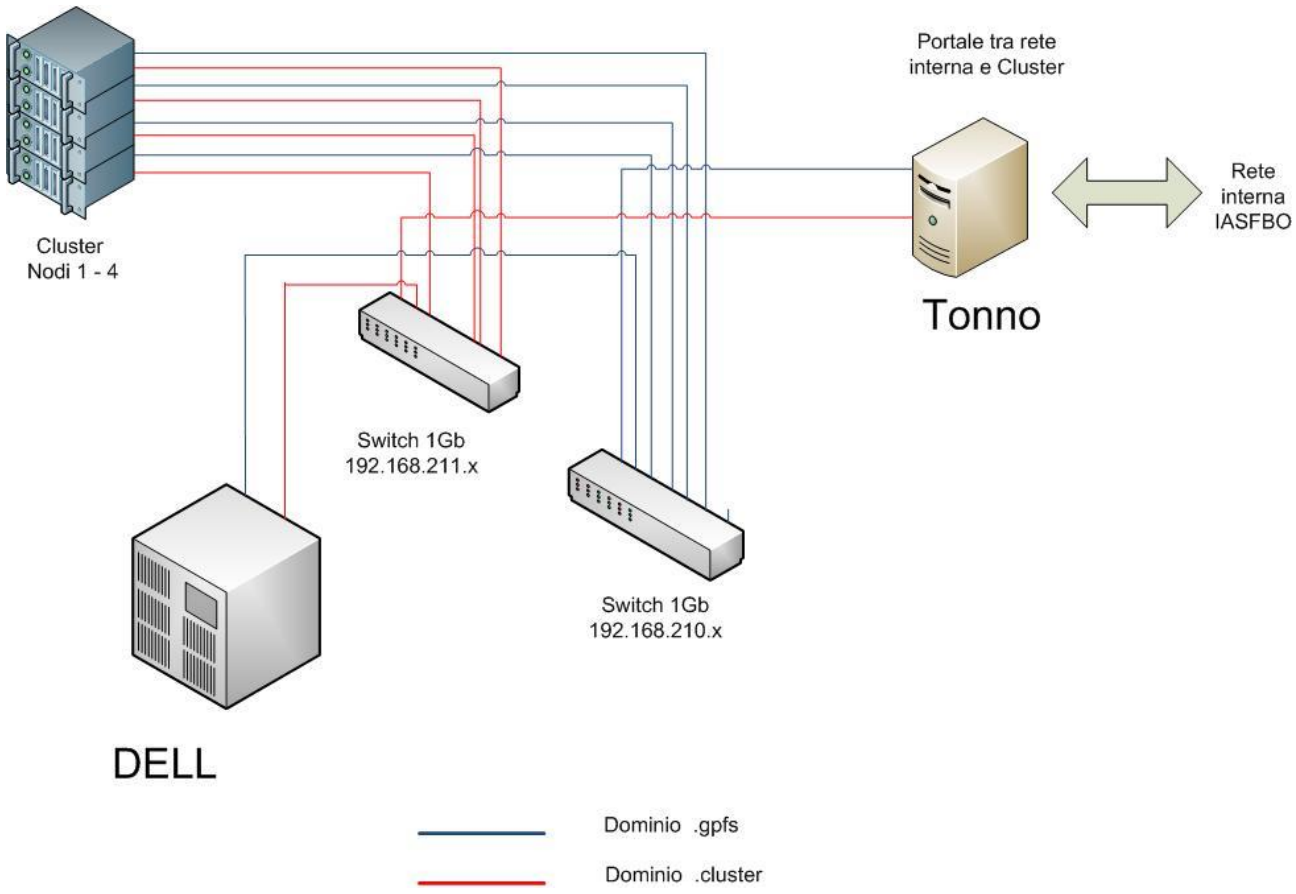
La cartella /scratch1, che è il punto di montaggio del File System GPFS di nome scratch1, ha dimensione di 1 Tbyte ed è composta fisicamente da 8 Hard Disk della stessa dimensione nella configurazione raid 1+0 posto nella macchina master.gpfs.

Schema collegamenti cluster

Poiché ogni elemento del cluster è dotato di doppia interfaccia Ethernet si è deciso di diversificare fisicamente la rete usata dal sistema GPFS da quella usata dal cluster HPC.
Lo scopo di tale operazione è di alleggerire il carico sulla rete dedicata al calcolo di tutti i dati che i nodi appartenenti al File System GPFS si scambiano tra di loro.
Sono stati creati quindi due sottodomini:

- .gpfs per riferirsi agli elementi appartenenti alla rete che crea il File System GPFS.
- .cluster per riferirsi agli elementi appartenenti alla rete HPC.

Tali sottodomini sono fisicamente e logicamente indipendenti, nel file /etc/hosts sono indicati, per ogni nodo, il nome host e l'indirizzo ip.



Aggregazione interfacce

E' stata provata anche la soluzione di aggregare le interfacce di rete così da raddoppiare il canale ma ciò ha dato numerosi problemi ed incompatibilità con il sistema operativo.
Nello schema seguente viene riassunto il concetto dello sdoppiamento fisico della rete. Sono indicati col colore blu i collegamenti relativi al sottodominio .gpfs e col colore rosso i collegamenti relativi al sottodominio .cluster.

Uso del GPFS

Il sistema GPFS (General Parallel File System) è stato scelto come base del Cluster di calcolo, attenzione esso non va a sostituire il File System del sistema operativo, ma risiede ed opera su uno strato diverso e per certi versi vi si affianca.
Come spiegato in precedenza su un sistema operativo Scientific Linux sono state lasciate delle partizioni vuote (ma possono essere dei dischi interi) ma formattate; sono stati poi installati e resi operativi dei demoni linux che hanno il compito di creare per prima cosa una struttura logica, a partire dal file scratch.nodes, che altro non è che un cluster di tutte le partizioni che compongono lo spazio disco condiviso del GPFS.
Successivamente è stato formattato il File System usando le informazioni contenute nel file scratch1.disks creando così uno spazio unico condiviso da tutti i nodi e che ha le suddette caratteristiche di sicurezza e di performance.
Nel nostro sistema quindi è presente una cartella */scratch1* che contiene le cartelle personali degli utenti e che è condivisa su tutti i nodi del cluster. Da queste cartelle verranno lanciati gli eseguibili.
La cartella */scratch1*, di dimensione di 1 TeraByte è stata pensata come luogo di lavoro e non come contenitore dei dati utente a tempo indeterminato.
Anche se, considerando il numero non elevato di utenti che usufruiranno delle risorse di calcolo, non è stato previsto al momento attuale un sistema di quote specifico, viene mantenuto un controllo sulla dimensione dello spazio disco utilizzato effettuando una scansione e successiva eliminazione di file e cartelle che non sono attivi da un certo periodo di tempo (ovvero i file non modificati negli ultimi 20 giorni).
Lo script che esegue tale operazione si chiama wipeold ed è situato nella cartella */usr/local/bin*.

Aggiunta nuove macchine

Il cluster di calcolo dell'istituto è attualmente costituito da 12 nodi di calcolo: 4 nodi aventi 2 processori dual core a 2,33 Ghz e 8 MB di ram ciascuna, e da 6 nodi aventi 2 processori quad core a 2,5 Ghz e 16 MB di ram più altri 2 nodi con 2 processori quad core e 32 MB di ram per un totale di 80 core.

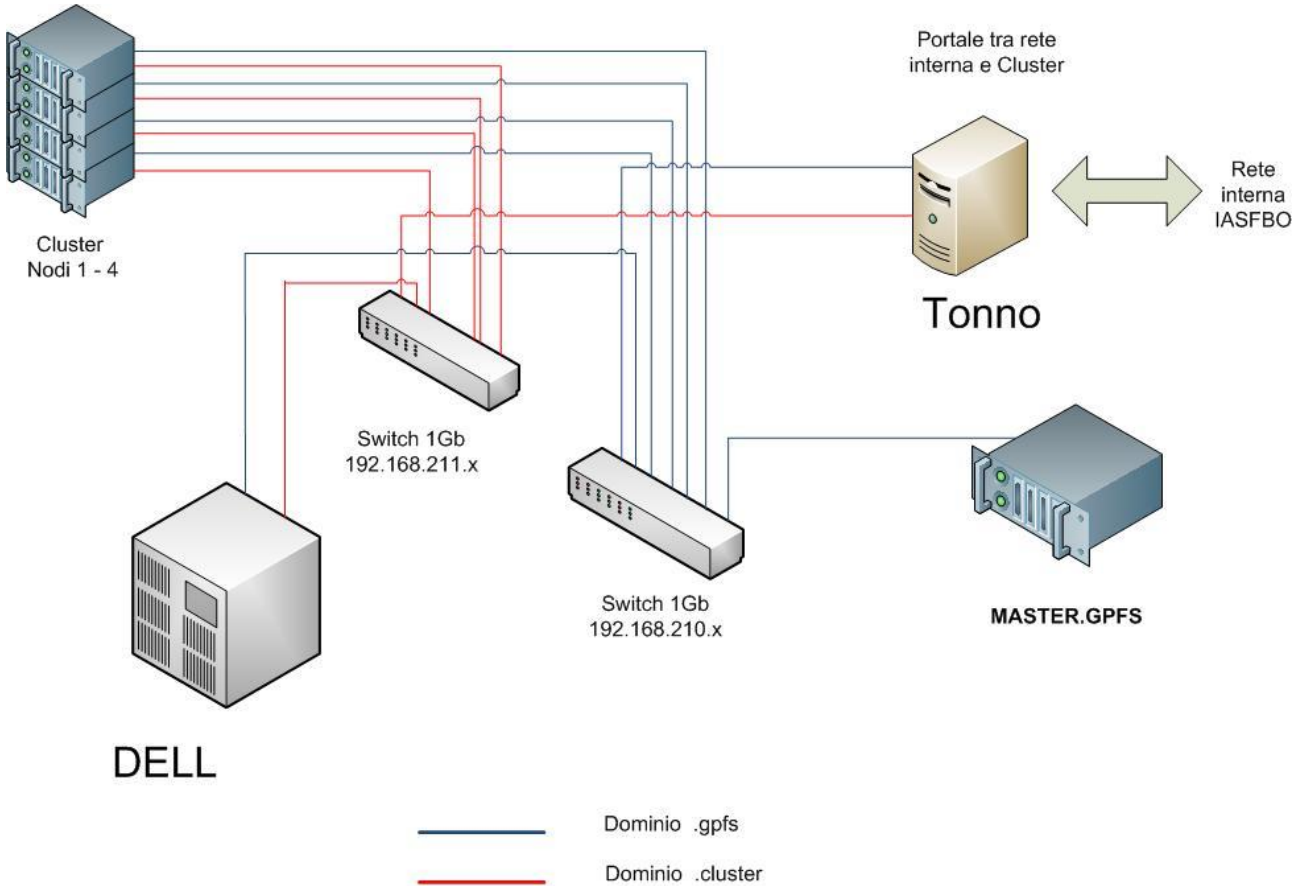
Nome nodo	N° Core	RAM
node1.cluster	4	8 GB
node2.cluster	4	8 GB
node3.cluster	4	8 GB
node4.cluster	4	8 GB
node5.cluster	4	4 GB
node6.cluster	4	4 GB
node7.cluster	8	32 GB
node8.cluster	8	32 GB
node9.cluster	8	16 GB
node10.cluster	8	16 GB
node11.cluster	8	16 GB
node12.cluster	8	16 GB
node13.cluster	8	16 GB
node14.cluster	8	16 GB

L'ultimo arrivato in famiglia è il nuovo fulcro del cluster ovvero un sistema di tipo rack Dell PowerEdge M1000e con Server Blade M600.
Il PowerEdge M1000e è un enclosure modulare capace di ospitare server di tipo Blade.
E' un sistema molto flessibile e scalabile con alimentatori e ventole di raffreddamento ridondati, dotato di 2 moduli I/O PowerConnect M6220 che garantiscono ciascuno 4 uplink Ethernet a 1 Gbit ed internamente fungono da switch (permettendo anche supporto a protocolli tipo OSPF e RIP, IPv6, ACL).
Il tutto in una struttura compatta che mette a disposizione un sistema remoto di gestione e controllo sia dello chassis che degli apparati in esso contenuti ed è raggiungibile all'indirizzo IP 192.168.166.210.
I veri nuovi protagonisti però sono i server Blade M600, dotati di 2 processori quad-core Intel Xeon per un totale di 8 Core ciascuno. Ogni Blade ha poi 16 GByte di RAM, Hard disk di 80 GByte ed è collegato allo switch interno dello chassis.
Questo sistema è la punta di diamante per quel che riguarda il calcolo parallelo nell'istituto IASF di Bologna fornendo la maggior parte della potenza di calcolo del cluster nel suo complesso.

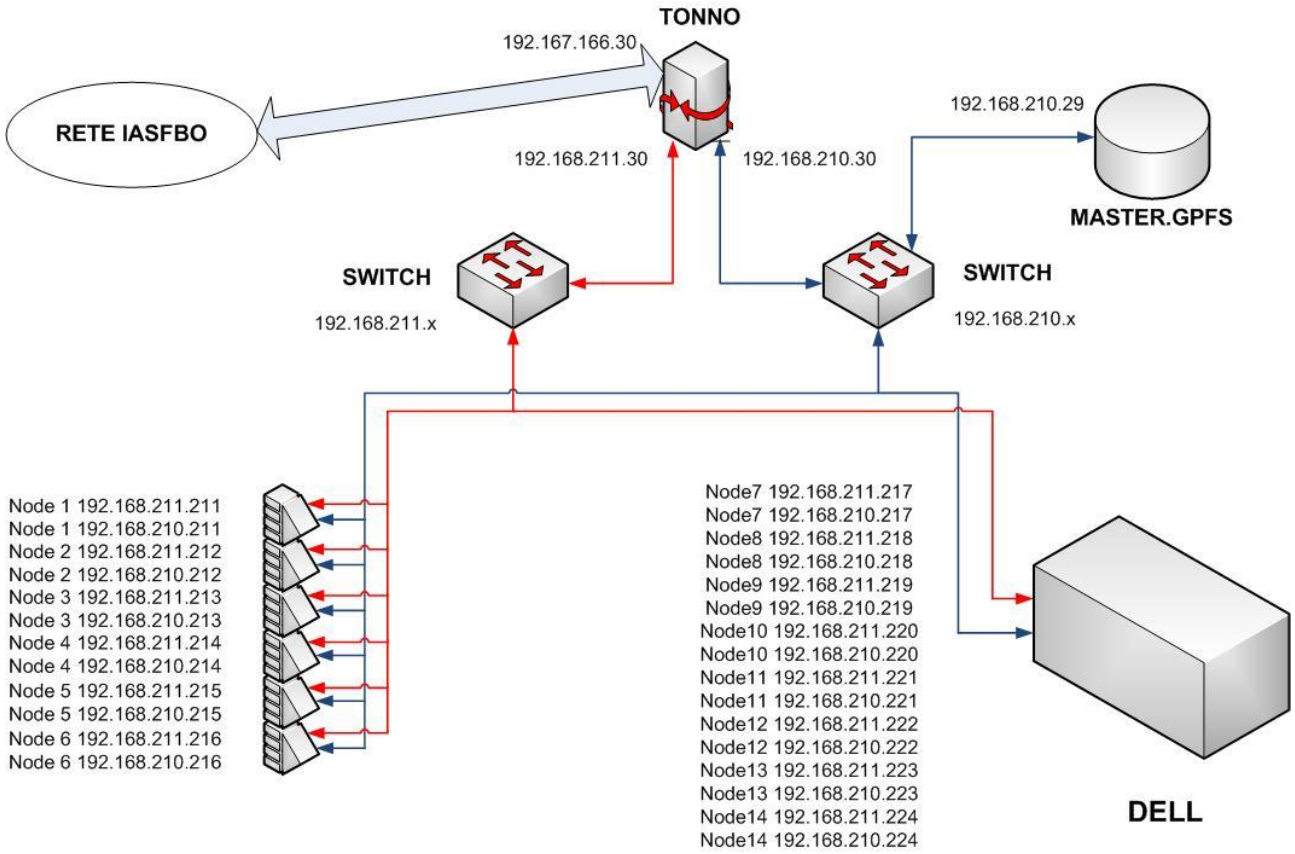
Schema con master.gpfs

Il nodo master.gpfs contiene i volumi fisici che forniscono lo spazio dico a tutto il sistema GPFS.
Tale sistema può all'occorrenza servirsi dei volumi fisici presenti sui nodi per creare uno spazio disco aggregato.
Nel nostro caso invece abbiamo un nodo particolare che fornisce lo spazio.
Su master.gpfs ci sono 8 dischi da 300 GByte in raid 1+0 che formano un volume unico, il cui nome +.AA

è sdb, di circa 1TByte.
Questo è messo a disposizione di tutti i nodi del cluster, attraverso GPFS, nella cartella */scratch1*.
La si utilizza andando nella sottocartella */scratch1/users* dove c'è (o si può creare) una cartella relativa ad ogni utente e dalla quale si devono lanciare i job.



Mappa indirizzi IP



Software installato

Vediamo innanzitutto i due software che consentono l'utilizzo comune del cluster, ovvero il gestore delle code **Torque** e lo scheduler **Maui.**

Torque

Il sistema ottenuto dalle operazioni fin'ora effettuate è un sistema ad alte prestazioni ma che non consente di effettuare alcun calcolo in maniera organizzata, in pratica è possibile utilizzare un'interfaccia per il calcolo parallelo come **MPI** per lanciare un proprio programma ma senza alcun controllo sulle risorse utilizzate, lasciando il sistema aperto a chiunque ne voglia usare tutte le risorse.
Ci viene in soccorso Torque.
Torque gestisce le risorse del sistema, intese come CPU, tempo di calcolo e RAM, mettendole a disposizione degli utenti in maniera razionale attraverso l'uso di code.
Quando un utente vuole eseguire un calcolo (in gergo *job*) fa una richiesta eseguendo il comando qsub ed inserendo il proprio job in una coda tra quelle disponibili.
La scelta della coda verrà fatta in base alle esigenze peculiari del *job*.

La gestione delle priorità con cui i *job* vengono eseguiti è affidata ad un altro elemento che si chiama Maui.
Le code presenti sono : serial, parallel, debug.

-serial: coda per il calcolo seriale che fa uso di un solo processore, priorità bassa, walltime = 24 ore

-parallel: coda per il calcolo parallelo, priorità medio/alta, walltime = 6 ore, i job di questa coda verranno eseguiti sui nodi da node7.cluster a node14.cluster.

-debug: coda usata per il debug di programmi di calcolo parallelo, priorità alta, walltime = 15 minuti

Con il parametro walltime si intende il tempo (previsto) di esecuzione di un *job.*

Si può usare il comando qstat -Qf per avere una descrizione dettagliata di tutte le code esistenti nel cluster.
Vediamo quali sono i parametri più importanti di una delle nostre code:

- Priority: è un indice di priorità. A valori più alti corrisponde una priorità maggiore.
- Max_running: è il numero massimo di *job* abilitati ad essere eseguiti.
- Resources_max.walltime: indica il tempo massimo di esecuzione assegnato a ciascun job.
- Resources_default.neednodes: specifica una caratteristica peculiare richiesta perché un nodo appartenga alla coda.

Per potere eseguire un proprio *job* è necessario usare il comando qsub con alcune opzioni obbligatorie.
Il formato del comando è il seguente: qsub –q . [[opzione]..]script.sh
Per una lista generica delle opzioni di qsub si rimanda all'appendice A, mentre per una descrizione più approfondita è possibile usare il comando man qsub su qualsiasi nodo del cluster.

Passi effettuati nell'installazione di Torque

Anche per Torque è stata effettuata la procedura di installazione completa sul nodo tonno.cluster, dopodiché sono state copiate le cartelle sugli altri nodi.
L'architettura di Torque prevede la presenza di un nodo principale che esegue il demone pbs_server e di nodi secondari (tipo client) che eseguono il demone pbs_mom.
Dopo aver scaricato ed installato il pacchetto torque-2.3.5.tar.gz si è configurato il demone pbs_server su tonno.cluster (file */var/spool/torque/server_name e /var/spool/torque/server_priv/nodes*) e si sono create le code. In seguito si sono copiati file e cartelle sui vari nodi e si è configurato ogni nodo (file */var/spool/torque/mom_priv/config*).
Si sono poi fatti partire tutti demoni, pbs_server sul nodo primario e pbs_mom su quelli secondari.
Librerie, moduli e file binari sono stati installati sulla cartella */opt* che è montata in remoto su ogni nodo.

Maui

Maui è uno scheduler, il suoi compito è stabilire l'ordine con cui i job presenti nelle code di Torque verranno eseguiti.

Le regole per stabilire la precedenza di esecuzione di un job vengono chiamate policies del cluster e si applicano configurando una serie di parametri che si trovano all'interno del file */opt/maui/maui.cfg*.
La procedura di installazione di Maui è stata più semplice di quelle fin'ora utilizzate, è stato sufficiente scaricare ed installare il pacchetto di tipo rpm (maui-oscar-3.2.6p19.8.src.rpm).
La configurazione è stata minima, quanto sufficiente per interfacciarlo con Torque.
Infine è stato fatto partire il demone corrispondente (maui).
Anche Maui è stato posto nella cartella */opt* che è montata via nfs su tutti i nodi.

Librerie e moduli

Le librerie di calcolo sono state poste nella cartella */opt* montata via nfs su tutti i nodi, inoltre è stato scelto un approccio modulare
L'ambiente modulare permette di gestire in modo più versatile e soprattutto più semplice l'ambiente di sviluppo oltre che le librerie e i programmi installati sul cluster.
In sostanza l'ambiente modulare non fa altro che impostare, in base al modulo caricato, le variabili d'ambiente del sistema per permettere l'utilizzo del determinato pacchetto.
Le variabili d'ambiente impostate dal gestore dei moduli sono legate alla particolare shell aperta, in altre parole, è possibile avere più shell aperte con moduli differenti caricati, come conseguenza però bisogna sempre ricordarsi per ogni shell che si apre di caricare i moduli necessari per compilare od eseguire un codice.

Breve lista delle opzioni del comando module

Con il comando module av è possibile visualizzare i moduli presenti sul sistema

Con il comando module load è possibile caricare un particolare modulo o una serie di moduli. Ad esempio:

```
[derosa@tonno ~]$ module load GNU-openmpi-1.3.0 GNU-SCALAPACK-64bit GNU-MKL-11-64bit GNU-HEALPIX-2.10-64bit GNU-CFITSIO-64bit GNU-BLACS-64bit
```

Con module list vedere quali sono i moduli caricati in memoria

```
[derosa@tonno ~]$ module list

Currently Loaded Modulefiles:
  1) switcher/1.0.13          3) GNU-openmpi-1.3.0        5) GNU-MKL-11-64bit         7) GNU-CFITSIO-64bit
  2) oscar-modules/1.0.5      4) GNU-SCALAPACK-64bit      6) GNU-HEALPIX-2.10-64bit   8) GNU-BLACS-64bit
[derosa@tonno ~]$
```

Il comando module unload permette di scaricare il modulo dall'ambiente

[derosa@tonno ~]$ module unload GNU-openmpi-1.3.0 GNU-SCALAPACK-64bit GNU-MKL-11-64bit GNU-HEALPIX-2.10-64bit GNU-CFITSIO-64bit GNU-BLACS-64bit

Il vantaggio di utilizzare un ambiente modulare per le librerie permette di poter rapidamente cambiare il compilatore o la versione della libreria senza effettuare modifiche al Makefile, questo grazie all'uso delle variabili d'ambiente associate al pacchetto.

Con il comando module load GNU-openmpi-1.3.0 GNU-SCALAPACK-64bit GNU-MKL-11-64bit GNU-HEALPIX-2.10-64bit GNU-CFITSIO-64bit GNU-BLACS-64bit, ad esempio, vengono caricati in memoria i moduli delle librerie scalapack, mkl, healpix, cfitsio e blacs.
Per ognuna di queste librerie vengono definite in memoria delle variabili d'ambiente costruite con la seguente regola:

NOME_LIBRERIA_INCDIR
NOME_LIBRERIA_LIBDIR

la _LIBDIR è sempre presente la _INCDIR solo per le librerie che hanno file .h da utilizzare nei codici

con il comando set è possibile vedere rapidamente quali siano le variabili impostate

BLACS_LIBDIR=/opt/gnu-lib/BLACS_patched/LIB
CFITSIO_INCDIR=/opt/gnu-lib/cfitsio/include
CFITSIO_LIBDIR=/opt/gnu-lib/cfitsio/lib
HEALPIX_INCDIR=/opt/gnu-lib/Healpix_2.10/include
HEALPIX_LIBDIR=/opt/gnu-lib/Healpix_2.10/lib
MKL_INCDIR=/opt/intel/Compiler/11.0/069/mkl/include
MKL_LIBDIR=/opt/intel/Compiler/11.0/069/mkl/lib/em64t
SCALAPACK_LIBDIR=/opt/gnu-lib/scalapack-1.8.0_patched

Un po' diverso è il discorso per openmpi, il pacchetto che si occupa del message passing all'interno dei codici paralleli: per come è stato compilato ed installato sul cluster, ricordatevi sempre che DEVE essere caricato SIA IN FASE DI COMPILAZIONE CHE IN RUN TIME, non richiede tuttavia di essere richiamato con variabili d'ambiente all'interno del Makefile, ma sono i wrapper ai compilatori (intel o gnu) che si occupano direttamente di includere e di linkare ciò che è necessario al codice. Quindi i compilatori mpif90, mpicc, mpif77, mpicc hanno lo stesso nome sia che si riferiscano al compilatore gcc-gfortran che icc-ifort e non richiedono in sostanza modifiche nei Makefile.

Uso del cluster

Per una breve guida ai comandi necessari ad utilizzare il cluster si rimanda alla guida presente sul sito IASF Bologna nella sezione help and faqs, sottosezione iasf-bo related topics.
Se ne riporta qui di seguito una versione estesa.

Come accedere

Per accedere al cluster si usa la stessa procedura utilizzata per accedere a snoopy o spike, cioè si digita ssh [nomeutente]@tonno seguito dalla propria password di istituto (quella per la posta, snoopy ecc.). Si accederà così alla propria home su tonno che è il portale per accedere al cluster.
Da notare come la home è proprio la stessa che si ha su spike o snoopy.
Per accedere al cluster da casa o comunque fuori dai locali dell'istituto è necessario innanzitutto accedere a snoopy o spike e da qui, tramite un ssh, accedere a tonno.

Come utilizzare il cluster

Per lanciare un programma è necessario spostarsi nella cartella */scratch1/users/[proprio nome]* e copiarvi ciò che serve per i propri jobs, ovvero dati; eseguibili; ecc.. Questo perché la propria home non è vista all'interno del cluster di calcolo.

ATTENZIONE!! Questa area è accessibile tutti nodi del cluster ma anche da tutti gli utenti!!! Quindi va usata solo come cartella di lavoro temporaneo.
Il comando per lanciare un job sul cluster si chiama "qsub", questi ha diverse opzioni che è consigliato conoscere.

Uso di qsub ed opzioni comuni

Il comando qsub viene invocato nella forma: qsub [[opzione]..]script.sh. Questo è uno script bash o tcsh la cui struttura deve essere di questo tipo:

```
#!/bin/bash      /* incipit obbligatorio per bash

#PBS [opzione]   /* opzione di qsub
..
#PBS –l walltime=xx:xx:xx  /* impostare il walltime del job
#PBS –q [nome coda]  /* scelta della coda, da impostare SEMPRE
#PBS –l nodes=x:ppn=y  /* per richiedere un certo numero di nodi e di processori per nodo
module load IDL-7.0  /* caricare i moduli necessari al job
.
comando di shell
comando di shell
eseguibile       /* ad esempio un comando per il calcolo parallelo
```

Per le code disponibili vedere il paragrafo relativo a Torque.
Un esempio dettagliato di come costruire il file script lo si può trovare nella cartella */scratch1*, il file si chiama template.sh.
E' "**necessario**" usare qsub con l'opzione "-d ." per indicare a qsub la working directory, questa fa si che lo standard output ed error vengano scritti nella cartella da cui è stato lanciato qsub.
Standard output ed error sono nella forma: [nome script].o[job_id], [nome script].e[job_id].

In alternativa è possibile usare il comando runjob che ha questa opzione già abilitata e funziona come qsub.

Dati su cartella scratch

Lo spazio disco sulla cartella scratch1 è di 1TB ed è condiviso su tutti i nodi e tra tutti gli utenti. E' concepito come spazio dati condiviso e temporaneo (e non come spazio su cui archiviarli), per questa ragione, giornalmente, verrà effettuata una pulizia della cartella che cancellerà i file più vecchi di 20 giorni. E' pertanto fortemente consigliato di utilizzare lo spazio sulla propria 'home' per conservare i dati.

Monitorare i propri job

E' possibile tenere sotto controllo i job che sono stati accodati attraverso alcuni comandi: Showq, showstart, qstat.

-Il comando showq è relativo allo scheduler Maui, si invoca senza particolari parametri e visualizza i job trattati dallo scheduler.
Vengono stampati su schermo alcune informazioni divise in colonne: nome del job, nome del proprietario del job, stato, processori allocati, tempo stimato rimasto prima del completamento, data di inizio elaborazione.
Vi è una ulteriore divisione in tre gruppi:

ACTIVE JOBS job che sono in esecuzione e stanno consumando le risorse del cluster.

IDLE JOBS job validi o che non violano nessuna policy ma non sono ancora in esecuzione. Sono visualizzati in ordine di priorità.

NON-QUEUED JOBS job che non possono essere eseguiti per diverse ragioni, ad esempio per violazione di una policy.

Se viene invocato con: showq –h visualizza un help.

-Il comando showstart è relativo allo scheduler Maui, può essere invocato solo dall'amministratore e dal possessore di un job e visualizza una stima del tempo che manca al job per essere eseguito.
Si invoca con:

showstart jobname [-h]

Dove jobname è il nome del job che si può ottenere tramite il comando *showq.*

Ancora l'opzione –h fornisce un help.

-Il comando qstat è relativo al gestore delle code e visualizza lo status di un job di cui si è proprietari.

Si invoca con:

qstat [-f [-1]] [-W site_specific] [-x] [job_id ..| destination ..]

dove job_id è l'identificatore del job che può essere scritto o solo come numero o come numero seguito dalla stringa ".tonno.cluster".
Il job_id può ancora essere ottenuto dal comando showq.
L'opzione –f fornisce lo status completo del job.

Un altro modo per monitorare in tempo reale i risultati del proprio job consiste nel ridirigere verso un file apposito l'output del comando di calcolo parallelo, ad esempio MPI, che è presente all'interno dello *script.sh* che si usa nell'invocazione con qsub.

Cancellare i propri job

Si può cancellare un job di cui si è proprietari attraverso il comando:

qdel [{-m <message>|-p|-W <delay>}] <job_id> [job_id]…| 'all'| 'ALL'

Il job_id è ancora il numero ottenuto tramite showq a cui si può aggiungere il suffisso .tonno.cluster.

Considerazioni finali e Sviluppi futuri

In conclusione le principali scelte operate sono state:

- Dato l'hardware mantenere Ethernet come mezzo fisico di comunicazione.
  Utilizzare la fibra ottica o Ethernet a 10Gb/s in luogo di Ethernet a 1 Gb/s avrebbe migliorato le prestazioni ma la scelta è stata dettata da ragioni economiche.

- Usare Scientfic Linux come sistema operativo.
  Questa scelta è stata fatta per la facile reperibilità di librerie per il calcolo scientifico e la compatibilità con Red Hat Linux (ne è una derivata), di contro c'è stato qualche problema di reperibilità di pacchetti nei normali repository.

- Usare GPFS come File System condiviso e base per la comunicazione all'interno del cluster.
  Rispetto a File System open source (ad esempio Lustre) il File System dell'IBM ha il vantaggio di una maggiore compatibilità con le applicazioni (ad esempio MPI) e performance migliori (ma minore robustezza) rispetto a nfs.

- Usare Torque e Maui per gestire le risorse del cluster.
  La scelta di queste due applicazioni è stata dettata dalla loro grande diffusione e stabilità, dal fatto che sono open source e dalla loro grande duttilità, anche a

dispetto di una interfaccia più ostica che ha richiesto un tempo maggiore di apprendimento.

Per quel che riguarda la configurazione attuale del cluster di calcolo questa consente delle buone prestazioni.
Anche la versatilità è buona, potendo dedicare i nodi più performanti al calcolo parallelo più spinto e lasciando a quelli più "vecchi" il calcolo più leggero.
Fronti di miglioramento ce ne sono e due sono i più evidenti; aumentare la velocità di interconnessione tra i nodi; aumentare ulteriormente la potenza del singolo nodo.
Per quel che riguarda la velocità di interconnessione le soluzioni sono due, o passare ad una rete ethernet a 10 Gbit/s o passare alla tecnologia in fibra come Myrinet o Infiniband.
Entrambe le soluzioni prevedono di cambiare completamente i sistemi di rete, ovvero schede di trasmissione (NIC), cavi e switch.
L'altro fronte è il potenziamento dei nodi.
Ora come ora il Cluster è sbilanciato, ci sono nodi veloci e potenti e nodi che lo sono molto meno.
E' per questa ragione che si usano delle code per differenziare il tipo di calcolo.
Scorporare i vecchi nodi più lenti, riutilizzandoli per altre attività, ed aumentare i nodi appartenenti all'Enclosure Dell avrebbe l'effetto di armonizzare e bilanciare il Cluster con benefici tanto più estesi per i codici che sfruttano in modo massiccio i principi e le architetture del calcolo parallelo.
Infine sarebbe auspicabile una serie di misure per stabilire con precisione le prestazioni rispetto a cluster più importanti.
Alcuni raffronti superficiali rispetto ad un cluster del CINECA di Bologna hanno evidenziato buone prestazioni di calcolo che non richieda una grande quantità di operazioni di I/O.

Appendice A

Comandi GPFS

Comandi usati per installare il File System GPFS su un calcolatore.
Si raccomanda di leggere le pagine MAN disponibili in linea.

Per creare il Cluster:
mmcrcluster –A -C gpfs –p tonno.cluster –N /ibm/scratch.nodes –r /usr/bin/ssh –R /usr/bin/scp

per creare lo spazio disco condiviso:
mmcrnsd –F /ibm/scratch1.disk –V no

per creare il file system:
mmcrfs /scratch1 scratch1(.gpfs?) –F /ibm/gpfs-disk –A yes –B 256k –n 60 –m 1 –M 2 –r 1 –R 2 –V no

per aggiungere un nodo al Cluster:
mmaddnode –N node5.gpfs

per montare il file system su tutti I odi del Cluster:
mmmount /scratch1 -a

Per fare partire I demoni di GPFS su tutti I nodi:
mmstartup –a

per visualizzare lo stato del Cluster
mmlscluster

per visualizzare lo stato del disco virtuale condiviso:
mmlsnsd –m

per fare partire I demoni di Torque:
service pbs_server start
service pbs_mom start

Comando per far partire un job:
qsub [-a date_time] [-A account_string] [-c interval] [-C directive_prefix] [-e path] [-h] [-l] [-j join] [-k keep] [-l resource_list] [-m mail_options] [-M user_list] [-N name] [-o path] [-p priority] [-q destination] [-r c] [-S path_list] [-u user_list] [-v variable_list] [-V] [-W additional_attributes] [-z] **[**script**]**

CONTENUTO DEL FILE SCRATCH1.DISK

/dev/sdb1:master.gpfs: :dataAndMetadata:1:disk01db1
disk01db1: : :dataAndMetadata:1: :

CONTENUTO DEL FILE SCRATCH.NODES

tonno.gpfs
master.gpfs: quorum-manager
node1.gpfs
node2.gpfs
node3.gpfs
node4.gpfs
node5.gpfs
node6.gpfs
node7.gpfs
node8.gpfs
node9.gpfs
node10.gpfs
node11.gpfs
node12.gpfs
node13.gpfs
node14.gpfs