Guida introduttiva e regole per l'uso dei cluster CASPER e HACTAR

Lo Staff di HPC@Polito - 18 novembre 2016

Questa guida è rivolta a tutti gli utenti dei nostri cluster
e si applica ai casi d'uso più comuni.
Qualora fossero necessarie funzionalità più avanzate, si invita a contattare lo staff di hpc@polito all'indirizzo di posta elettronica
hpc.dauin@polito.it

Gentlemen Agreements

- Utilizzando i nostri sistemi per le attività di ricerca, si autorizza automaticamente lo Staff di hpc@polito alla pubblicazione dei propri dati personali (nome, cognome, gruppo di ricerca) e quelli del progetto sul nostro sito (hpc.polito.it) ed in tutte le pubblicazioni di hpc@polito (report annuali, presentazioni, paper).
- Utilizzando i nostri sistemi per le attività di ricerca, ci si impegna a citare il centro di calcolo in tutti i propri articoli scientifici su rivista, conferenza o libri.
  Suggeriamo questo tipo di acknowledge:

  "Computational resources provided by hpc@polito,which is a project of Academic Computing within the Department of Control and Computer Engineering at the Politecnico di Torino (http://www.hpc.polito.it)".

  Alternativamente una versione più breve come
  "Computational resources provided by hpc@polito (http://www.hpc.polito.it)".

Regole per l'uso del cluster

- NON devono mai essere avviati calcoli, simulazioni, ecc. direttamente da linea di comando rischiando così di rendere inutilizzabile il nodo di login o altre risorse condivise (pena il blocco dell'account). Passare SEMPRE attraverso le code di esecuzione dello scheduler.
- NON è consentito qualunque uso delle risorse che esuli l'attività didattica e/o di ricerca.
- Ogni utente è responsabile per qualunque attività svolta o riconducibile allo stesso, è pertanto scoraggiato l'account sharing, consentito solo se segnalato per tempo allo Staff di HPC. In generale, NON è permessa la condivisione a terzi di dati sensibili (nome utente, password) che l'utente si impegna a custodire adeguatamente.
- Come norma generale, NON è consentito fare nulla che NON sia espressamente consentito.

Il presente documento integra il regolamento del centro di calcolo hpc@polito con informazioni di carattere tecnico e pratico sul funzionamento dello stesso.
Versioni aggiornate del regolamento e del relativo manuale d'uso possono essere scaricate dal sito web del progetto: *http://hpc.polito.it*.
In particolare si ricorda che tutti gli utenti sono tenuti al rispetto del Regolamento.

L'iniziativa HPC del Politecnico di Torino è un centro di Academic Computing, ovvero fornisce risorse di calcolo e supporto tecnico per attività di ricerca accademica e didattica.
Lo sviluppo dell'iniziativa *HPC* è stato ufficialmente affidato dal Consiglio di Amministrazione al laboratorio LABINF (Laboratorio Didattico di Informatica Avanzata), presso il *DAUIN* (Dipartimento di Automatica e Informatica).

Casper

Le specifiche di Casper sono:

Architecture	Linux Infiniband-DDR MIMD Distributed Shared-Memory Cluster
Node Interconnect	Infiniband DDR 20 Gb/s
Service Network Gigabit	Ethernet 1 Gb/s
CPU Model	2x Opteron 6276/6376 (Bulldozer) 2.3 GHz (turbo 3.0 GHz) 16 cores
Performance	4.360 TFLOPS
Power Consumption	7 kW
Computing Cores	544
Number of Nodes	17
Total RAM Memory	2.2 TB DDR3 REGISTERED ECC
OS	ROCKS Clusters 6.1
Scheduler	GridEngine 2011.11p1

Hactar

Le specifiche di Hactar sono:

Architecture	Linux Infiniband-QDR MIMD Distributed Shared-Memory Cluster
Node Interconnect	Infiniband QDR 40 Gb/s
Service Network Gigabit	Ethernet 1 Gb/s
CPU Model	2x Xeon E5-2680 v3 2.50 GHz (turbo 3.3 GHz) 12 cores
GPU Node	2x Tesla K40 - 12 GB - 2880 cuda cores
Performance	9.7 TFLOPS
Computing Cores	360
Number of Nodes	15
Total RAM Memory	1.9 TB DDR4 REGISTERED ECC
OS	CentOS 6.6
Scheduler	GridEngine 2011.11

Storage

Le specifiche per gli storage sono:

Home Storage	140 TB on RAID 6, throughput near 200 MB/s
Lustre Storage	87 TB, throughput greater then 2.2 GB/s
Storage Interconnect	Ethernet 10 Gb/s

Primo Accesso

L'account corrispondente al nome utente assegnato in fase di richiesta (es. *username*), è da considerarsi attivo sui Cluster dal momento in cui si riceve la email di conferma riportante i dati di accesso.

La modalità di accesso al cluster dipende dal sistema operativo sul client in uso.
Nel caso si disponga di un sistema *Linux*, *Unix* o *OSX* si può utilizzare il client *ssh* da un qualunque terminale tramite il comando:

[Casper] *$ ssh username@casperlogin.polito.it*

[Hactar] *$ ssh username@hactarlogin.polito.it*

sostituendo a *username* il nome utente riportato nella email, cui seguirà la richiesta della password (anch'essa riportata nella email) e infine l'accesso vero e proprio al cluster.

Nel caso si disponga di un sistema *Windows* si consiglia l'uso dell'appicativo *PuTTY (disponibile all'indirizzo - http://www.putty.org)* da configurarsi come in figura 1.

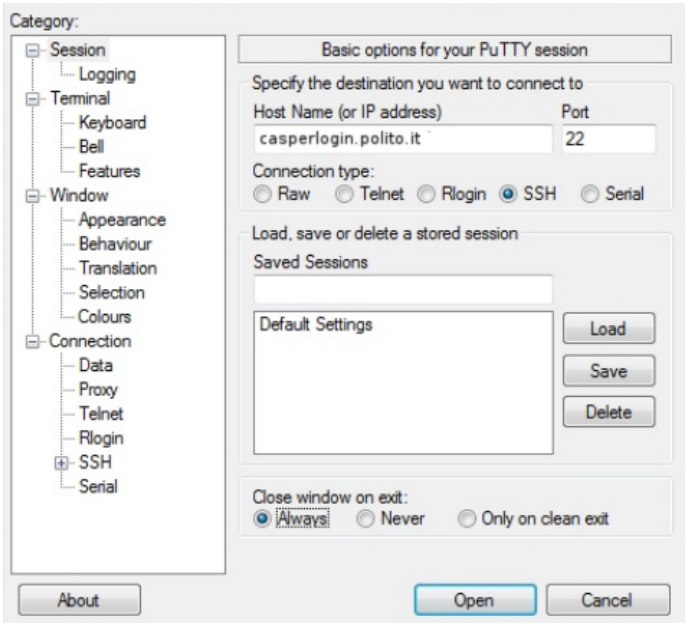

Figura 1: Putty - Configurazione di *PuTTY* per l'accesso a CASPER da Windows

Al primo accesso è di FONDAMENTALE IMPORTANZA cambiare la password!
Per farlo sarà sufficiente digitare il comando:

$ passwd

Non esistono limitazioni sulla lunghezza e sulla composizione delle password d'accesso, tuttavia si consiglia di scegliere una password composta da 8 o più caratteri, contenente numeri e lettere maiuscole e minuscole.

Si ricorda, che non è consentita la pratica dell'Account Sharing (ovvero la condivisione dello stesso utente tra più persone), inoltre, chi ha fatto richiesta dell'account deve prendersi cura delle proprie credenziali: scegliendo password opportune, non rivelando a terzi dati sensibili e segnalando per tempo allo Staff di HPC l'eventuale compromissione della propria utenza.

Trasferimento File

Una volta effettuato l'accesso ci si troverà direttamente nella propria home directory attraverso l'*Home Storage*, accessibile su entrambe i cluster da:

/home/username/

Su questa sistema di storage ogni utente può utilizzare fino a 1TB di spazio disco, eventualmente espandibile.
Gli utenti possono richiedere un aumento della quota loro riservata, inviando una mail ad *hpc.dauin@polito.it* e includendo le oppurtune motivazioni; la richiesta verrà quindi valutata dalla Staff di HPC@POLITO.
Per trasferire i file sul cluster dal proprio host è possibile utilizzare il comando *scp*:

[Casper] *$ scp -r /path/to/local/dir/ username@casperlogin.polito.it:/home/username*

[Hactar] *$ scp -r /path/to/local/dir/ username@hactarlogin.polito.it:/home/username*

questi comandi copiano la directory */path/to/local/dir/* e il suo contenuto all'interno della directory */home/username/* presente sui cluster.

Viceversa è possibile anche copiare file dal cluster sulla propria macchina:

[Casper] *$ scp -r username@casperlogin.polito.it:/home/username /path/to/local/dir/*

[Hactar] *$ scp -r username@hactarlogin.polito.it:/home/username /path/to/local/dir/*

Il comando *scp* funziona quindi in modo simile al comando *cp* nei sistemi Unix.

$ cp SOURCE DESTINATION

Per trasferire file sul cluster è inoltre possibile l'utilizzo del protocollo SFTP.
Ad esempio, è possibile sfruttare un programma come FileZilla, configurato come in figura.

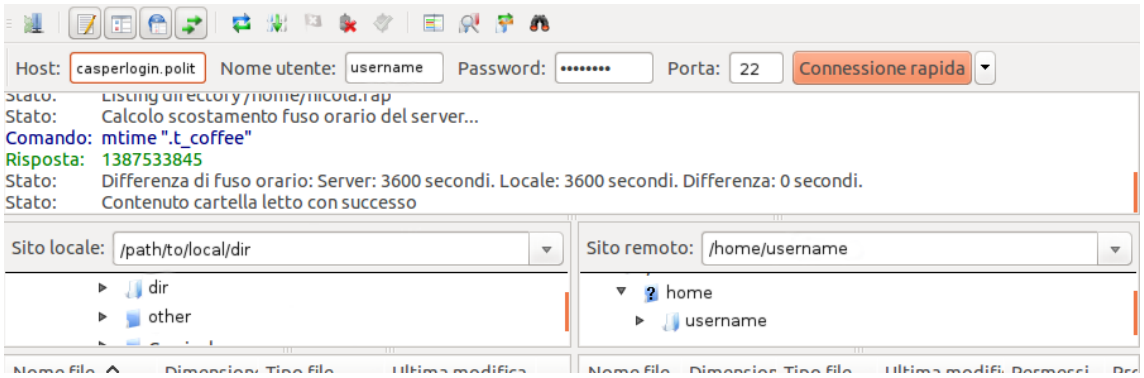

Figura 2: Configurazione FileZilla per il trasferimento file su CASPER

Storage ad alte prestazioni

Gli utenti possono inoltre accedere ad un addizionale storage ad alte prestazioni provvisto di filesystem Lustre, pensato per incrementare le prestazioni di tutti i processi che fanno ampio uso di file.

Per accedere a questo storage è sufficiente riferirsi al path:

/work

qui ogni utente possiede già una directory con il proprio nome e gli adeguati permessi per poter gestire i propri file.

Per poter spostare i file dal normale storage in uso allo storage ad alte prestazioni, sarà quindi sufficiente l'uso di *cp*:

cp /home/nome_utente/nome_file /work/nome_utente/nome_file

cp -r /home/nome_utente/nome_directory /work/nome_utente/nome_directory

Su questa unità di storage è riservato per ogni utente un ulteriore 1TB di spazio disco (*soft quota*), ma in caso di necessità sarà possibile oltrepassare tale soglia fino a 1.5TB (*hard quota*) per non più di 7 giorni.
Per conoscere lo stato della propria quota usare il comando:

$ get-my-lustre-quota -h

Lo storage ha una capacità di 87TB e può raggiungere performance oltre 1.1GB/s usato da CASPER e oltre 2.2 GB/s da HACTAR. Gli utenti possono richiedere un aumento della quota loro riservata, inviando una mail ad *hpc.dauin@polito.it* e includendo le oppurtune motivazioni; la richiesta verrà quindi valutata dalla Staff di HPC@POLITO.

Storage locale ai nodi

È possible sfruttare dello spazio addizionale presente sui dischi locali di ogni nodo di calcolo. Tale spazio può essere temporaneamente utilizzato seguendo i path:

[Casper] */state/partition1* *210GB*

[Hactar] */scratch* *1TB*

Uso del Cluster

I processi eseguiti su un cluster sono detti processi batch, vale a dire che non sono interattivi e che la loro esecuzione può essere rimandata nel tempo. Un job è composto da uno o più processi che lavorano insieme per raggiungere un certo risultato.
Per far si che i job vengano eseguiti nella rete di calcolatori, questi devono essere "schedulati" ossia inseriti in una lista, in attesa che le risorse di calcolo necessarie siano disponibili.
Sia HACTAR che CASPER utilizzano lo scheduler Grid Engine per gestire le code di attesa e quindi la disponibilità delle risorse di calcolo condivise.

Su CASPER esistono due code ad accesso pubblico:

all.q	Coda di default ad accesso pubblico, include tutti i nodi (544 Core su 17 nodi), massima durata di un job 10 giorni.
fast.q	Coda ad alta priorità, 8 Core su 2 nodi, massima durata di un job 60 min.

La coda *fast.q* è ideale per tutti quei job che richiedono poco tempo e poche risorse di calcolo e che quindi trarrebbero poco o alcun beneficio dall'uso del cluster, tuttavia è molto utile per le attività di test (*run* di prova) per i job più grandi.

Anche HACTAR disponde di due code ad accesso pubblico:

all.q	Coda di default ad accesso pubblico, include tutti i nodi (360 Core su 15 nodi), massima durata di un job 10 giorni.
cuda.q	Coda ad alta priorità per il nodo GPU (compute-1-14), 2 Core su 1 nodo + 2 Slot gpu su un nodo, massima durata di un job 10 giorni.

La coda *cuda.q* è ideale per tutti i job che richiedono l'utilizzo di GPU per il calcolo (es. *rendering*, *librerie CUDA*, *etc.*).
Per l'utilizzo del nodo GPU riferirsi all'esempio in appendice A.

Sottomissione e Controllo di un Job
- qsub, qstat, qdel, qhost qlogin -

QSUB

Tutti i task devono essere eseguiti tramite l'ausilio del comando *qsub*, che deve ricevere come argomento il riferimento a uno script contenente le informazioni necessarie alla schedulazione e all'esecuzione del task.
Alcune delle direttive presenti nello script sono obbligatorie(*), pena l'attesa permanente in "*qw*" (coda d'attesa).

[script.qsub]

```
#!/bin/bash
#$ -N Nome_Del_Task
#$ -M email_utente@mail.com
#$ -m abes
#$ -cwd
#$ -j y
#$ -S /bin/bash
#$ -pe shared 32
#$ -l h_rt=HH:MM:SS
#$ -l virtual_free=16G
#$ -q all.q
task_di_esempio.bin
```

Direttive:

-M	Email alla quale verrano inviate le informazioni sul task.
-m	Eventi per i quali è richiesta una notifica via mail (*es. abes* - abort, begin, end, suspend).
-cwd	Se presente il task verrà eseguito partendo dalla directory corrente.
-j y|n	Indica se redirigere o meno lo stderr sul file di output.
-S	Interprete che verrà usato per lo svolgimento del task.
-pe	(*)Specifica il contesto di parallelismo in cui il task verrà eseguito. Ogni Job necessita di un certo numero di slot (ovvero di core) per l'esecuzione, che possono essere "scelti" sullo stesso nodo (max. 32 per CASPER, max. 24 per HACTAR) o su più nodi in base a quanto specificato in questa direttiva. I più comuni e utilizzati sono *shared*, *orte*, *mpirr* e *mpisp*: **shared** - Quando è neccessario eseguire più thread o più processi in una singola macchina multi-core (nodo) specificando il numero di slot (*es. -pe shared 16*). **orte** - Quando è necessario parallelizzare più processi che fanno uso di MPI su un certo numero di slot, massimizzando l'uso di ogni singolo nodo; Politica di schedulazione "fill-up" (*es. -pe orte 16*). **mpirr** - Quando è necessario parallelizzare più processi che fanno uso di MPI su un certo numero di slot, massimizzando la distribuzione dei processi sui nodi; Politica di schedulazione "round robin" (*es. -pe mpirr 16*). **mpisp** - Quando è necessario eseguire un ben determinato numero di processi MPI su ogni nodo assegnato. Vantaggioso soprattuto per i processi MPI che istanziano thread OpenMP (*è possibile trovare un esempio nell'appendice A*).
-l h_rt	(*)Indica il tempo di calcolo (presunto) necessario al processo per il completamento, fornendo così allo scheduler un utile strumento per la gestione dei processi. Questo valore deve essere minore a 10 giorni (<240) per la coda *all.q*.
-l virtual_free	Memoria da pre-allocare per lo svolgimento del task
-q	(*)Coda di riferimento per la schedulazione del task (se non diversamente specificato usare *all.q*
-l gpu	Usato insieme alla coda cuda.q, permette l'utilizzo delle GPU presenti sul nodo 14 di HACTAR (*è possibile trovare un esempio nell'appendice A*)

QSTAT

Per ottenere informazioni sui job schedulati tramite *qsub*, come il *job-ID*, lo *status*, le *code* e gli slot occupati si utilizza il comando *qstat*:

$ qstat

```
job-ID  prior   name   user    state submit/start at    queue                  slots
-----------------------------------------------------------------------------------
494180 0.51008 test   username  r  01/01/2015 00:00:00 all.q@compute-0-0.local  4
494182 0.51008 test   username  r  01/01/2015 00:00:00 all.q@compute-0-0.local  4
494183 0.51008 test   username  r  01/01/2015 00:00:00 all.q@compute-0-0.local  4
494184 0.51008 test   username  r  01/01/2015 00:00:00 all.q@compute-0-0.local  4
```

Per ottenere maggiori informazioni sullo stato di un singolo job si può usare invece il comando:

$ qstat -j {job-ID}

```
$ qstat -j 123456
==============================================================
job_number:                 123456
exec_file:                  job_scripts/123456
submission_time:            Thu Jan 01 00:00:00 2015
owner:                      username
uid:                        000
group:                      usergroup
gid:                        000
sge_o_home:                 /home/username
sge_o_log_name:             username
sge_o_path:                 ...
sge_o_shell:                /bin/bash
sge_o_workdir:              /home/username/...
sge_o_host:                 casper
account:                    sge
cwd:                        /home/username/...
merge:                      y
hard resource_list:         h_rt=720000
mail_list:                  username@casper.local
notify:                     FALSE
job_name:                   jobname
jobshare:                   0
hard_queue_list:            all.q
shell_list:                 NONE:/bin/bash
script_file:                run
parallel environment:       orte range: 32
usage    1:                 cpu=185:03:51:38, mem=2380221.15046 GBs, ...
scheduling info:            ...
```

Per ottenere informazioni su tutti i job in esecuzione sul cluster si può usare il comando:

*$ qstat -u **

Se si vuole consultare lo stato delle code:

$ qstat -g c

```
CLUSTER QUEUE                   CQLOAD   USED    RES  AVAIL  TOTAL aoACDS  cdsuE
--------------------------------------------------------------------------------
all.q                             0.31    214      0    330    544       0      0
bioeda.q                          0.40     18      0     78     96       0      0
fast.q                            0.43      0      0     36     36       0      0
...
```

se invece si vuole lo stato delle code per nodo:

$ qstat -f

```
queuename                      qtype resv/used/tot. load_avg arch          states
---------------------------------------------------------------------------------
all.q@compute-1-1              BIP   0/0/24         0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-1-10             BIP   0/0/24         0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-1-11             BIP   0/0/24         0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-1-12             BIP   0/0/24         0.00     linux-x64
---------------------------------------------------------------------------------
all.q@compute-1-13             BIP   0/0/24         0.02     linux-x64
---------------------------------------------------------------------------------
...
```

QDEL

Infine, per arrestare l'esecuzione di un job usare il comando

$ qdel {job-ID}

che causerà l'interruzione del job e la rimozione dalla coda.

QHOST

Il comando *qhost* permette di conoscere lo stato dei nodi del cluster in un dato momento.

$ qhost

```
HOSTNAME                ARCH         NCPU  LOAD  MEMTOT  MEMUSE  SWAPTO  SWAPUS
-------------------------------------------------------------------------------
global                  -               -     -       -       -       -       -
compute-0-0             linux-x64      32  2.00  126.2G    2.1G 1000.0M  394.7M
compute-0-1             linux-x64      32  2.41  126.2G   32.0G 1000.0M  601.8M
compute-0-10            linux-x64      32 20.02  126.2G    5.1G 1000.0M  166.9M
compute-0-11            linux-x64      32  1.03  126.2G    3.8G 1000.0M  210.4M
compute-0-12            linux-x64      32  2.02  126.2G    5.7G 1000.0M  336.6M
compute-0-13            linux-x64      32  2.04  126.2G    5.2G 1000.0M  174.6M
...
```

È inoltre possibile conoscere la distribuzione del carico di lavoro sui nodi di un utente utilizzando l'opzione *-u username*.

$ qhost -u username

Sessioni Interattive sui nodi di calcolo

In tutti i casi in cui sia richiesto l'utilizzo di una sessione interattiva su un nodo di calcolo è possibile utilizzare il comando *qlogin*.

Questo è l'unico modo consentito per aprire una sessione remota su un nodo di calcolo.

QLOGIN

Dal nodo di login:

$ qlogin -q {all.q} -pe shared {n.slot} -l h_rt {HH:MM:SS}

Le opzioni corrispondo con le direttive di un normale script di qsub (vedere *qsub*).

L'utilizzo del comando *qlogin* è sottoposto a costante monitoraggio secondo quanto definito dalla politica di QLOGIN ABUSE SUPPRESSION.

Si ha un abuso quando una sessione *qlogin* viene lasciata inattiva alterando così le politiche di scheduling del cluster.

Il meccanismo di soppressione funziona in questo modo:
alle ore 3:00 di ogni giorno per ogni *qlogin* in stato 'r' (running) viene calcolato il rapporto fra la media dell'uso della CPU e le CPU effettivamente richieste in fase di sottomissione, se tale rapporto è inferiore a una certa threshold il job viene terminato con *qdel*.
Tale soglia è fissata al 30% (ratio = 0.3).

```
Esempio 1:
   JOB-ID = 300112
   STARTED AT = 1395741955 (UNIX time)
   ELAPSED TIME = 106900 s
   CPU SECONDS = 607772 s
   LOAD\_AVG = 5.68
   REQUESTED CPU = 32
   RATIO = .17
---------------> KILLED

Esempio 2:
   JOB-ID = 300113
   STARTED AT = 1395741955 (UNIX time)
   ELAPSED TIME = 106900 s
   CPU SECONDS = 1215544 s
   LOAD\_AVG = 11.36
   REQUESTED CPU = 32
   RATIO = .35
---------------> KEEP ALIVE
```

Sessioni grafiche interattive e X11 Forwarding

Il comando *qlogin* permette, oltre al normale uso, di aprire sessioni grafiche interattive direttamente sui nodi di calcolo. Per far ciò è necessario stabilire fin dall'accesso che la sessione corrente farà uso di *X11 Forwarding*, cioè:

[Casper] *$ ssh -XC username@casperlogin.polito.it*

[Hactar] *$ ssh -XC username@hactarlogin.polito.it*

dopo di che si dovrà utilizzare il comando *qlogin* per ottenere una sessione interattiva e, infine, richiamare il programma che si vuole eseguire.

References

La documentazione completa è disponibile al link:

http://gridscheduler.sourceforge.net/htmlman/manuals.html

MODULES

Il software *modules* (o nella sua forma estesa *Environment Modules Project*) è un software che permette la modifica dinamica delle variabili d'ambiente di una sessione utente. Il suo utilizzo è fortemente consigliato all'interno degli script *qsub*, perché fornisce un facile sistema per l'utilizzo di diverse versioni di uno stesso software e inoltre facilita l'esportazione di certe variabili ambiente per i software che ne fanno uso riducendo gli errori.
Una lista dei moduli utilizzabili sul *Cluster in uso* può essere ottenuta con il comando:

username@login-0-0$ module avail

I moduli possono essere caricati (*load*) o rimossi (*unload*) dalla sessione utente.
Ad esempio, per caricare il modulo per l'utilizzo di *openmpi*, si usa il comando:

[Casper] *username@login-0-0$ module load OpenMPI/1.6.2/gcc/module*

[Hactar] *username@login-0-0$ module load OpenMPI/1.8.4/gcc/module*

per rimuovere dalla sessione i moduli non necessari (al limite tutti "*") si può usare il comando:

*username@login-0-0$ module unload **

infine, la lista dei moduli caricati nella sessione utente corrente si ottiene con il comando:

username@login-0-0$ module list

I moduli possono essere inoltre caricati direttamente nell'ambiente di lavoro di un nuovo job indipendentemente da quelli caricati nella sessione.
Ad esempio, per l'utilizzo di *openmpi* su rete infiniband, bisognerà aggiungere allo script qsub la linea:

[Casper] *module load OpenMPI/1.6.2/gcc/module*

[Hactar] *module load OpenMPI/1.8.4/gcc/module*

GANGLIA

Ganglia è un'interfaccia web che permette di monitorare l'attività del cluster. È possibile raggiungerla agli indirizzi:

[Casper] *http://casper.polito.it/ganglia/*

[Hactar] *http://hactar.polito.it/ganglia/*

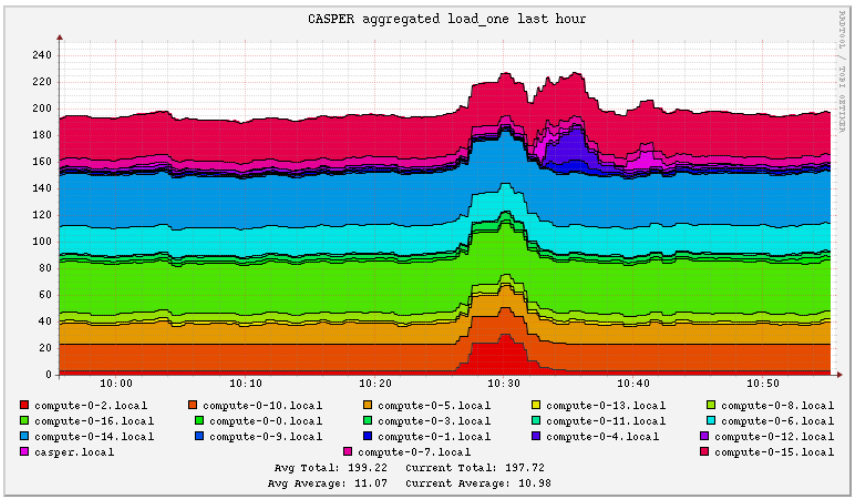

Figura 3: Ganglia - Esempio di monitoraggio delle load sui nodi

APPENDICE A

Sono qui riportati una serie di template per alcuni applicativi presenti sui Cluster.
(Alcuni degli esempi si riferiscono a software ancora in fase di testing su HACTAR e quindi non diponibili agli utenti)

.....Matlab [matlab.qsub] .....

```
#!/bin/bash
#$ -N test_matlab
#$ -M mail_utente@mail.com
#$ -m abes
#$ -cwd
#$ -j y
#$ -S /bin/bash
#$ -pe shared 16
#$ -q all.q
#$ -l h_rt=00:10:00

module load matlab/R2014b/module

matlab -r test_matlab
```

.....Blender [blender.qsub] .....

```
#!/bin/bash
#$ -N test_blender
#$ -M mail_utente@mail.com
#$ -m beas
#$ -cwd
#$ -j y
#$ -S /bin/bash
#$ -pe shared 24
#$ -q all.q
#$ -l h_rt=01:00:00

module load blender/2.76b/module
```

```
    blender -noaudio -b BMW1M-MikePan.blend.1 -o Cycles_Benchmark -F PNG -f 1 -t 32
```

Comsol [comsol.qsub]

```
    #!/bin/bash
    #$ -N comsol_test
    #$ -M mail_utente@mail.com
    #$ -m abe
    #$ -cwd
    #$ -S /bin/bash
    #$ -pe shared 16
    #$ -q all.q
    #$ -l virtual_free=64G
    #$ -l h_rt=120:00:00

    module load comsol/5.0/module

    comsol -np $NSLOTS batch -inputfile input_file.mph -outputfile output_file.mph
```

Cplex [cplex.qsub]

```
    #/bin/bash
    #$ -N test_cplex
    #$ -M mail_utente@mail.com
    #$ -m abes
    #$ -cwd
    #$ -j y
    #$ -S /bin/bash
    #$ -pe shared 8
    #$ -q all.q
    #$ -l h_rt=01:00:00
    #$ -l virtual_free=32G

    module load cplex/12.6/module

    cplex < problem.lp.cplex > problem.cplex.out
```

problem.lp.cplex

```
      read problem.lp
      set timelimit 41472
      set mip display 0
      set mip tol mipgap 0
      set parallel 1
      set threads 8
      set workmem 8192
      set mip limits treememory 32696
      optimize
      quit
```

PROCESSI MPI

I job composti da più processi paralleli MPI possono essere distribuiti sul cluster con due differenti logiche: *fill-up* o *round-robin*.

fill-up cerca di riempire i nodi di calcolo, sfruttando le risorse presenti e velocizzando l'esecuzione dei processi. È la politica di schedulazione del Parallel Environment (PE) *orte*.

round-robin cerca, invece, di distribuire i processi su più nodi, favorendo la disponibilità di risorse per ogni processo. È la politica di schedulazione del Parallel Environment (PE) *mpirr*.

MPI (FILL-UP) [mpi.qsub]

```
#$ -M mail_utente@mail.com
#$ -m abes
#$ -N mpi_quicksort
#$ -cwd
#$ -j y
#$ -S /bin/bash
#$ -pe orte 96
#$ -q all.q
#$ -l h_rt=05:00:00
#$ -l virtual_free=16G

module load OpenMPI/1.8.4/gcc/module

time /opt/openmpi/bin/mpirun -np 96 mpi_quicksort.x input_21GB.txt output.txt
```

MPI (ROUND-ROBIN) [mpirr.qsub]

```
#!/bin/bash
#$ -M mail_utente@mail.com
#$ -m abes
#$ -N mpi_quicksort
#$ -cwd
#$ -j y
#$ -S /bin/bash
#$ -pe mpirr 16
#$ -q all.q
#$ -l h_rt=05:00:00
#$ -l virtual_free=16G

module load OpenMPI/1.8.4/gcc/module

time /opt/openmpi/bin/mpirun -np 16 mpi_quicksort.x input_21GB.txt output.txt
```

Nel caso in cui si conosca con esattezza il numero di processi MPI che si vuole eseguire su ogni nodo, è possibile utilizzare il Parallel Environmemt (PE) *mpisp*.
(*Attualmente presente solo su CASPER*)

Per utilizzarlo si può utilizzare uno script simile al seguente, avendo cura di:

- passare a *mpirun* la lista degli host disponibili (*-machinefile $JOB_ID.machines*)
  - *$JOB_ID.machines*, è un file generato dal PE contenente la lista degli host disponibili;
- passare a *mpirun* il numero di processi totali (*-np $(($NHOSTS*$NPN))*
  - *$NHOST*, è il numero di host disponibili al momento della sottomissione del job;
  - *$NPN*, è numero di processi che si vogliono eseguire per nodo;
- passare a *mpirun* il numero di processi per nodo (*-npernode $NPN*)

..... MPI (NUMBER OF PROCESSES PER NODE) [mpisp.qsub] .....

```
#!/bin/bash
#$ -S /bin/bash
#$ -pe mpisp 16
#$ -cwd
#$ -V
#$ -N mpitest
#$ -q all.q
#$ -l h_rt=1:00:00

#############################################################
NPN=1 # set here the number of processes per node
#############################################################

module load OpenMPI/1.6.2/gcc/module

# $JOB_ID.machines, file containing the avaible host list, generated by sge
# $NHOSTS, number of avaible hosts passed by sge

mpirun -machinefile $JOB_ID.machines -np $(($NHOSTS*$NPN)) -npernode $NPN mergesort
```

.....

GPU [gpu.qsub]

```
#!/bin/bash
#$ -N cuda_task
#$ -M mail_utente@mail.com
#$ -m abes
#$ -cwd
#$ -S /bin/bash
#$ -pe shared 2
#$ -q cuda.q
#$ -l h_rt=05:00:00
#$ -l virtual_free=16G
#$ -l gpu=1

module load nvidia-cuda/7.5/module

./cuda-task-bin
```

La sottomissione di job, su questa particolare coda (*cuda.q*), permette di sfruttare le risorse di calcolo offerte dal nodo gpu.
-l gpu è una direttiva (vedere "Sottomissione e Controllo di un Job" a pag. 7) che indica il numero di gpu richieste dal job (*al momento 1 o 2*).

APPENDICE B

Per task più complessi che richiedono l'esecuzione dello stesso processo più volte ma con diversi attributi, si consiglia l'uso di questo script che genera diversi file *qsub* partendo da un file csv contente i parametri.

[generate.sh]

```
#!/bin/bash
#
list=$(cat ./parameters.csv)
#
for i in $list
do
parameter1=$(echo $i|cut -f 1 -d ‘,’)
parameter2=$(echo $i|cut -f 2 -d ‘,’)
cat << EOF > ./test-$parameter1-$parameter2.qsub
#/bin/bash
#$ -N task-$parameter1-$parameter2
#$ -M mail_utente@mail.com
#$ -m abes
#$ -cwd
#$ -j y
#$ -S /bin/bash
#$ -pe shared 24
#$ -q all.q
#$ -l h_rt=01:00:00
#$ -l virtual_free=32G
task_di_esempio.bin  $parameter1 $parameter2
EOF
done
```

[parameters.csv]

```
A,1
A,2
B,1
B,2
```