AGENZIA NAZIONALE
PER LE NUOVE TECNOLOGIE, L'ENERGIA
E LO SVILUPPO ECONOMICO SOSTENIBILE

Corso di formazione per il progetto TEDAT
ENEA C. R. Brindisi, 24-25 giugno 2013

Utilizzo dei sistemi ENEA-GRID/CRESCO

Agostino Funel
agostino.funel@enea.it
ENEA Centro Ricerche Portici
P.le Enrico Fermi 1, Portici (Napoli)

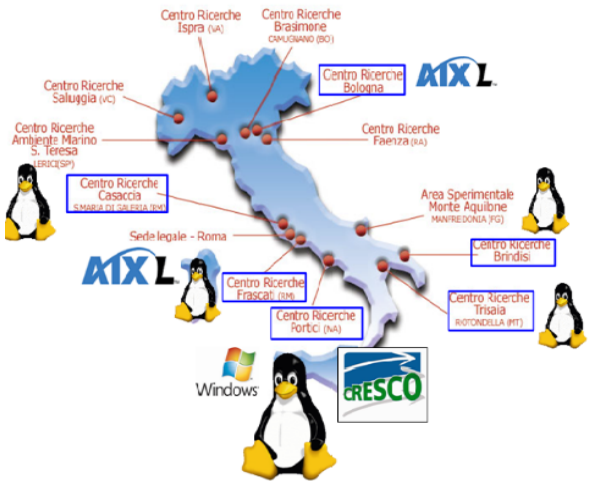
L'infrastruttura ENEA-GRID

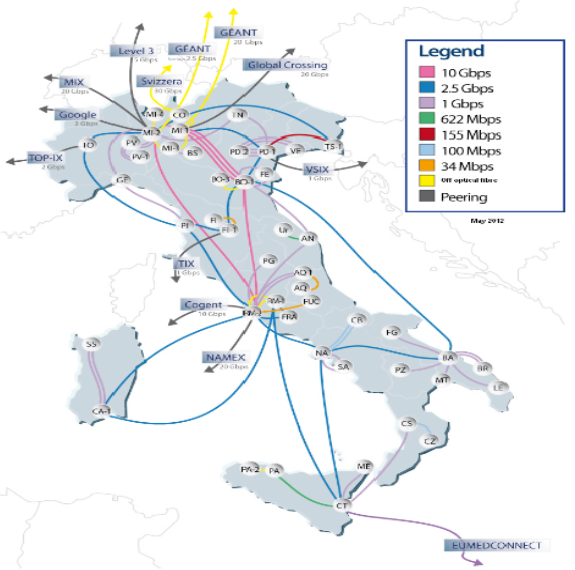
ENEAGRID integra in un'unica infrastruttura l'insieme delle risorse di calcolo scientifico di ENEA, distribuite nei suoi principali Centri di Ricerca.

Le piattaforme di calcolo offerte all'utenza sono attualmente i sistemi Linux x86_64 (i cluster HPC CRESCO ~ 6000 core), i sistemi AIX SP5 (~256 core), sistemi speciali dedicati (ad es. GPU), risorse virtualizzate. ENEA-GRID offre uno spazio dati online di ~500 TB. Le risorse sono distribuite su 6 Centri.

I Centri ENEA-GRID sono connessi dalla rete GARR.

I core HPC CRESCO
Portici: ~5500 (+ 5000)
Frascati: ~500
Casaccia: ~200
Brindisi: ~100

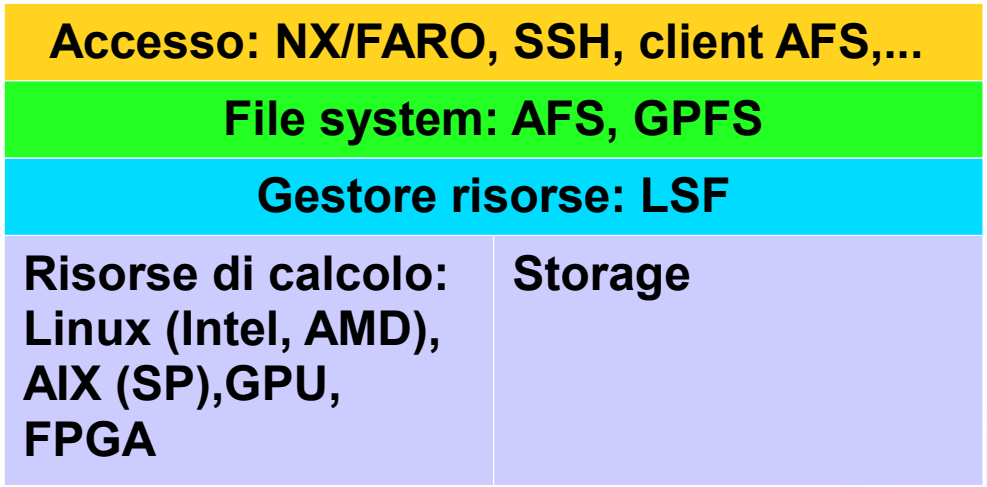




Il middleware ENEA-GRID

Componenti strutturali "maturi" per garanzia di affidabilità e semplicità di gestione, interfaccie Web sviluppate/customizzate per una ambiente utente amichevole:

- **Authenticazione: Kerberos 5**
- **File systems:**
- **AFS/OpenAFS file system geografico**
- **GPFS: file system parallelo**
- **Gestore delle risorse: LSF Multicluster**
- **Interfacce grafiche Web per l'utente:**
- **NX/FARO (accesso)**
- **Jobrama: Stato dei job & Accounting**
- **Sistema di monitoring: Zabbix**
- **Gestione Web utenze e progetti: WARC**



AFS/1 Concetti di base

AFS (Andrew File System) : è un file system distribuito che consente di accedere a file e directory su macchine diverse, dislocate anche su siti remoti, in modo da fornire un ambiente di lavoro uniforme.

Si basa sul modello client/server.

I dati risiedono in volumi che sono partizioni dei dischi dei server.

Client : workstation o nodi di calcolo su cui gira un software in grado di connettersi ai server. Il client chiede i file al server.

Server : provvede l'accesso ai volumi e dà i file richiesti al client.

Cache Manager : processo che gira sui client e fa una copia locale dei file. Una volta finito il lavoro i file vengono salvati sul server.

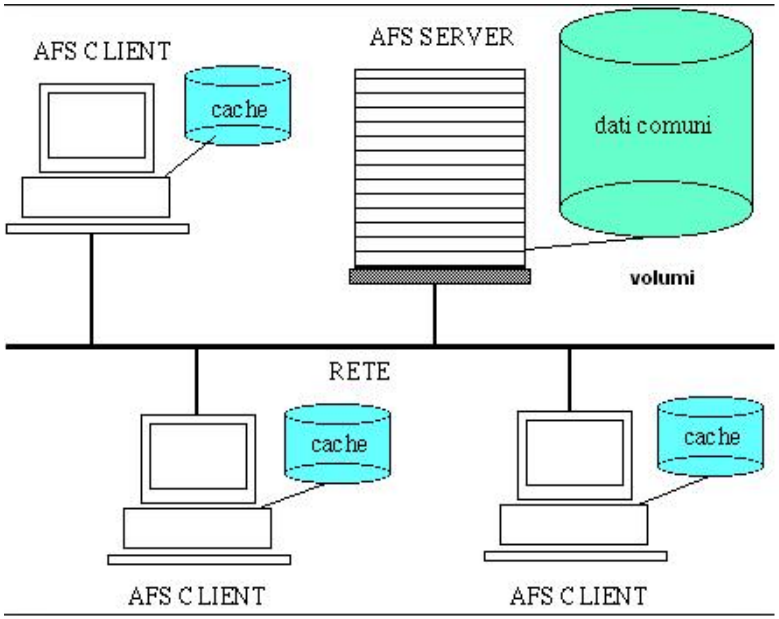
AFS/2 Il File Space

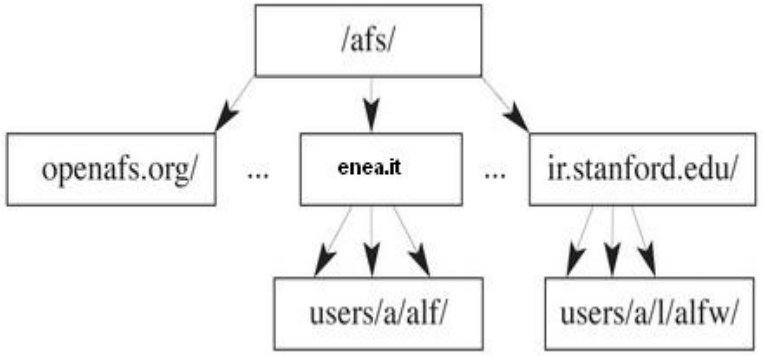
AFS è strutturaro in “celle” (enea.it). Una cella è un insieme di client e server.

Il file sysem ha una struttura ad albero che ha per root il path /afs. Tutte le celle sono visibili al secondo livello dell'albero.

Ogni cella ha il suo proprio File Space e per accedervi bisogna avere l'autorizzazione che avviene mediante l'assegnazione di un “token”. Un utente che non ha un token è considerato anonimo “system:anyuser”.

La struttura ad albero di AFS è vista come un insieme di directory/sotto directory.





AFS/3 Permessi

La protezione dei file è basate sulle ACL (Access Control List). Le ACL agiscono a livello di directory. Le ACL definite per una directory si applicano a tutti i file e sottodirectory creati successivamente all'impostazione delle ACL. Se un file viene spostato da una directory ad un'altra acquisisce le ACL della directory di destinazione.

Permessi di accesso:
- I livello, accesso alla directory: lookup (l), insert (i), delete (d), admin. (a)
-II livello, accesso ai file: read (r), write (w), lock (k)

l: permette di usare il comando ls per vedere i nomi dei file/sottodir. Deve essere impostato per poter usare altre ACL. Per es. rl per leggere/copiare.
a: permette di modificare le ACL
k: per programmi che hanno bisogno di un uso esclusivo della dir. o dei file

AFS consente la creazione di gruppi ai quali è possibile assegnare ACL.

AFS/3 Permessi

La protezione dei file è basata sulle ACL (Access Control List). Le ACL agiscono a livello di directory. Le ACL definite per una directory si applicano a tutti i file e sottodirectory creati successivamente all'impostazione delle precedenti ACL. Se un file viene spostato da una directory ad un'altra acquisisce le ACL della directory di destinazione.

Permessi di accesso:
- I livello, accesso alla directory: lookup (l), insert (i), delete (d), admin. (a)
-II livello, accesso ai file: read (r), write (w), lock (k)

l: permette di usare il comando ls per vedere i nomi dei file/sottodir. Deve essere impostato per poter usare altre ACL. Per es. rl per leggere/copiare.
a: permette di modificare le ACL
k: per programmi che hanno bisogno di un uso esclusivo della dir. o dei file

AFS consente la creazione di gruppi ai quali è possibile assegnare ACL.

AFS/4 Comandi base utente

klog <username>: ottenere un token
unlog <cellname>: distruggere il token
tokens: vedere i token
kpasswd: cambiare password
fs help: online help sui File Server
fs whereis <dir/path>: conoscere la macchina file server della dir. specificata
fs checkservers: riporta lo stato dei server
fs listquota <path>: riporta la quota disco per il path specificato
fs listacl <path>: fa vedere le ACL per il path specificato
fs setacl <path> <ACL entry>: imposta le ACL per il path specificato
fs copyacl <source dir> <dest dir>: copia le ACL da una dir a un'altra
pts help: online help sui Protection Server (gestione gruppi)
pts creategroup <user:group>: crea un gruppo
pts adduser -user <user> -group <group>: aggiunge un utente a un gruppo
pts membership<group> (<user>):fa vedere i membri delgruppo (i gruppi di user)
pts removeuser <user> <group>: rimuove un utente da un gruppo
pts delete <group>: cancella un gruppo

AFS/5 La HOME ENEA-GRID

Tutti gli utenti di ENEA-GRID hanno la stessa struttura di base della HOME. Questa scelta consente di avere un ambiente di lavoro uniforme, indipendentemente dalle risorse hardware/software che si vogliono usare e che potrebbero trovarsi anche su siti diversi.

HOME: /afs/enea.it/site/user/userid

- ~/private di default accessibile solo all'utente
- ~/public di default accessibile ad ogni utente per la condivisione dei dati
- ~/public_html di default accessibile ad ogni utente e il suo contenuto è pubblicato sul web all'indirizzo www.afs.enea.it/userid
- ~/rem è un mount point per volumi che risiedono su siti remoti (da chiedere agli amministratori)

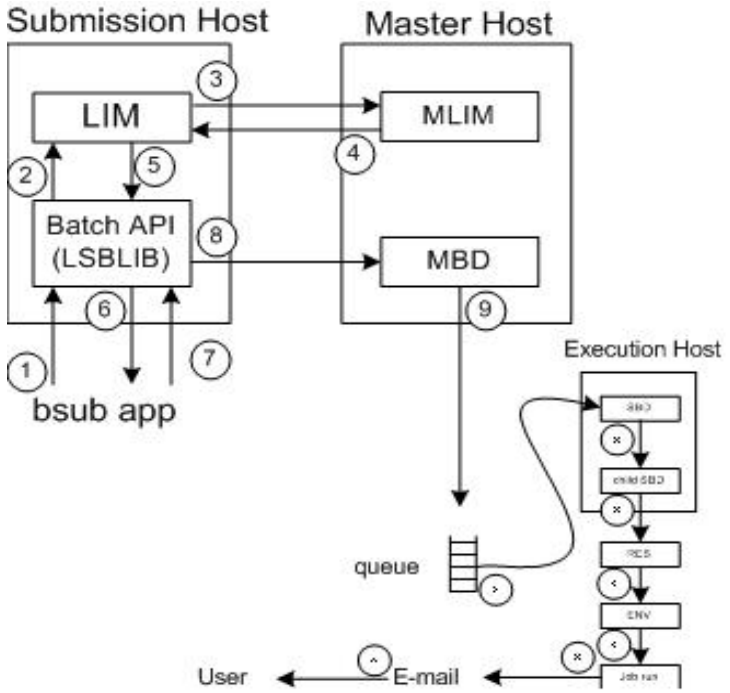
~/PFS link per l'accesso ai file system paralleli GPFS

LFS Concetti di base

LSF (Load Sharing Facility) è un software che si occupa di distribuire i job su ENEA-GRID

Fa vedere le risorse disponibili in maniera astratta nascondendo la complessità architetturale agli utenti.

La architettura software di LSF si basa su programmi che girano in background (demoni). Il Master Batch (MB) viene contattato dall'host da cui si sottomette il job. Il MB schedula il job sulle code in base alle risorse richieste. Le code inviano il job ai nodi di calcolo appena questi sono disponibili.



LFS Risorse/1

Una risorsa è una variabile che identifica ciò che può essere messo a disposizione degli utenti dalla infrastruttura di calcolo.

Esempi di risorse : processori, memoria, disco etc.

In LSF le risorse hanno un tipo :

- boolean : può esistere o meno su una certa macchina

-numeric : risorsa che ha un valore numerico

-string : è usata per assegnare nomi, ma anche elenchi di valori divisi da un separatore

LFS Risorse/2

Oltre al tipo una risorsa può essere :

- statica: se il suo valore è costante nel tempo

-dinamica: se il suo valore cambia nel tempo

Esempio : la memoria installata (RAM) è una risorsa costante. La memoria libera è una risorsa dinamica.

Le risorse numeriche dinamiche si comportano diversamente a seconda della loro “direzione” :

- crescente: il valore aumenta all'aumentare del carico

-decrescente: il valore diminuisce all'aumentare del carico

LFS Risorse/3

Una risorsa locale ha un valore diverso per ogni macchina.

Una risorsa condivisa ha lo stesso valore su tutte le macchine oppure su un gruppo di macchine. Può avere valori diversi su gruppi diversi.

Nome	*Descrizione*	*Tipo*	*Dinamica*	*Condivisa*	*Direzione*
ut	Utilizzo in percentuale della CPU	numeric	si	no	crescente
mem	Memoria libera	numeric	si	no	descrescente
maxmem	Memoria installata	numeric	no	no	-
type	Systema operativo	string	no	no	-
free_slots	Slot liberi su hardware XYZ	string	si	no	-
mpich	Indica che su un host gira mpich	boolean	si	no	-
fluent_lic	Licenze di fluent	numeric	si	si	decrescente
fluent_inst	Licenze di fluent installate	numeric	no	si	-

LFS Comandi base

Ecco alcuni comandi essenziali per l'uso di LSF. I comandi hanno help e man pages.

- **lshosts**: mostra le risorse statiche per host
- **lsload**: risorse dinamiche per host
- **lsinfo**: mostra l'elenco dei nomi di risorsa
- **bhosts**: mostra il numero di job sui vari host
- **bqueues**: informazioni sulle code
- **bsub**: lancia un job
- **bjobs**: controlla lo stato di un job
- **bkill**: termina un job in esecuzione

LFS Comandi : lshosts/1

Per vedere le risorse statiche a disposizione si utilizza lshosts.


```
[lsf@lin4p ~]$ lshosts
HOST_NAME      type    model  cpuf ncpus maxmem maxswp server RESOURCES
graphbri.br     SGI6    ONYX3  30.0     1  1024M  4096M    Yes (sgi telnet)
ercules.bri IBMAIX5   POWER4  60.0     2  8192M  8704M    Yes (sp4 telnet abaqus)
campus03.br   LINUX  PIV3400  55.0     4  4027M 24003M    Yes (linux telnet abaqus)
spartacus.b UNKNOWN UNKNOWN_   1.0     -     -     -     No ()
```

LFS Comandi : lshosts/2

Provare a lanciare i seguenti comandi

lshosts brindisi <tutti gli host di un cluster>

lshosts -l crescob1 <dettaglio di un host>

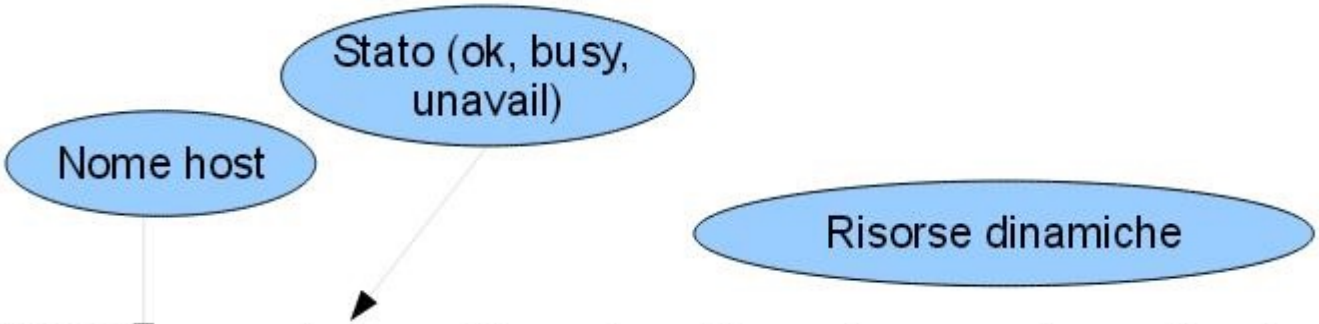
lshosts -R linux <filtro su risorsa boolean linux>

lshosts -R type==generic <filtro su type “generic”>

Nota : -R è usato in molti comandi di LSF e serve a richiedere delle risorse.

LFS Comandi : lsload/1

Il comando lsload è l'analogo di lshosts per le risorse dinamiche.




```
HOST_NAME      status  r15s   r1m  r15m   ut      pg  ls     it   tmp   swp   mem
eurofel16.frasc    ok   0.0   0.0   0.0   0%    0.0   0  10400 6412M 2037M 2634M
eurofel15.frasc    ok   0.0   0.0   0.0   0%    0.0   0    176 5624M 2047M 3294M
sp5-3.frascati.    ok  14.4  14.5  13.5  88%    9.9   0    382  255M   30G 1133M
info-eva.bologn    ok  26.8  26.4  26.5  84%    0.0   1    113  435M   16G  24G
lin4p.frascati.  busy   0.1   0.9   0.1   5%*245.6   6      0 5904M 7148M 7192M
aix42w.frascati unavail
```

LFS Comandi : lsload/2

- Provare a lanciare lsload con le stesse opzioni che abbiamo usato con lshosts. Cosa cambia ?
- Le opzioni dei due comandi, in questo caso sono intercambiabili. L'output dei due comandi rispecchia due aspetti diversi. Il comando lshosts mostra risorse statiche, mentre lsload è focalizzato sulle risorse dinamiche.

Risorse incluse in LSF/1

Risorse statiche:

- **type**, **model**: sistema operativo e modello di macchina
- **cpuf**: è un indicatore della potenza di CPU. L'amministratore può cambiare questo parametro se non fosse accurato
- **ncpus**: indica il numero di cpu (nei processori multicore, dipende dalla configurazione)
- **maxmem**, **maxswp**: memoria e swap installati
- **RESOURCES**: risorse boolean, definite dall'amministratore

Risorse incluse in LSF/2

Risorse dinamiche:

- **status**: **unavail** indica indisponibilità. **busy** indica macchina carica. **ok**, macchina scarica.
- **r15s**, **r1m**, **r15m**: è la run queue length mediata in 15 secondi, 1 minuto, 15 minuti (vedere man uptime).
- **ut**: utilizzo in % di tutte le cpu
- **pg**, **ls**, **it**: paging, login interattivi, idle time
- **tmp**, **swp**, **mem**: rispettivamente lo spazio in tmp, lo swap e la memoria liberi.

La rql dà il valore medio del numero dei processi pronti ad usare la CPU nell'intervallo di tempo specificato. Su Unix rql non è necessariamente uguale al carico medio (uptime) in quanto talvolta si riportano i processi di paging e/o di I/O su disco. Su multiprocessori la rql è opprtunamente scalata.

Pg dà la percentuale (pg/sec) di paging nella memoria virtuale. Se la RAM non è sufficiente allora pg aumenta e il nodo risponde lentamente.

Risorse LSF: esempi/1

Per allenarsi con le risorse senza causare problemi si possono usare lsload ed lshosts.

- lsload -R “r1m<1”
  - Minore, maggiore, uguale: <, >, ==
  - Qui l'uso delle virgolette è necessario
  - r1m si misura in processi
- lsload -R “mem>3000”
  - La memoria si misura in MegaByte
- lsload -R “swp>3000 && mem>4000”
  - Operatori. AND: &&, OR: ||, NOT: !

Risorse LSF: esempi/2

- lsload -R type== generic
  - type è statica, ma si può usare con lsload
- lsload -R "(status==busy || ut>50) && type== generic "
  - L'uso di parentesi è consentito e molto utile
- lshosts -R '!server'
  - “Not server” elenca le macchine solo client
  - L'uso dell'apice singolo è utile se la shell cercasse di interpretare il punto esclamativo.
- lsinfo
  - Vedo l'elenco di tutte le risorse.

LSF : il comando bsub

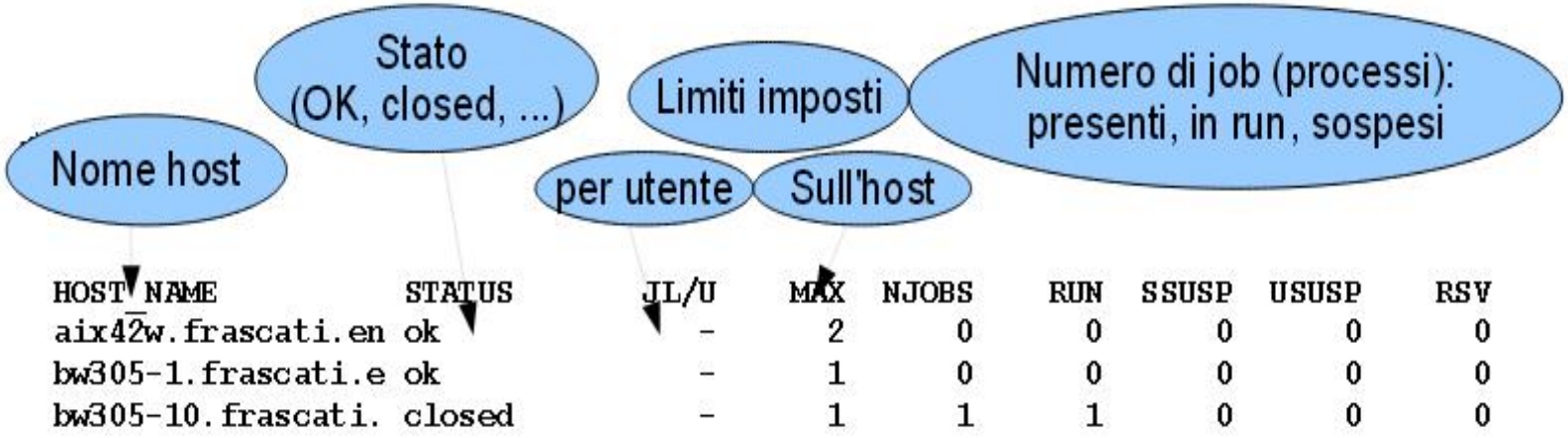
- Per lanciare un job in LSF bisogna trovarsi su un client o server LSF e lanciare:
  - **bsub** <para_bsub> **comando** <para_cmd>
- Primo lancio:
  - bsub sleep 10000
- Che informazioni mi dà il comando bsub in output ?
  - Il jobid, utile per far riferimento al job sottomesso
  - La coda di sottomissione

LSF : le code

- La coda è un contenitore astratto di job che serve a scegliere in modo accurato il server di esecuzione.
- Il comando **bqueues** elenca tutte le code a disposizione. **bqueues -l <coda>**: mostra i dettagli di una coda.
- Usare **bsub -q <coda>** per lanciare il proprio job su una specifica coda. Regole:
  - Tutte le code sono limitate
  - Le code maggiormente limitate hanno maggior priorità.
  - E' molto utile stimare la durata del proprio job
- E' compito dell'amministratore gestire le code

LSF : gli host/1

- Il comando **bhosts** mostra la situazione dei job distribuiti su tutti gli host.
- A differenza di lsload e lshosts, che mostrano la situazione monitorata, bhosts vede il cluster con gli occhi di LSF



LSF : gli host/2

bhosts -l

Per schedulare i job, LSF tiene effettivamente conto di questi parametri, a prescindere dall'output di lsload o lshosts.

le risorse possono essere bloccate

possono essere definite soglie o finestre temporali

```
[lsf@lin4p ~]$ bhosts -l sp5-1
HOST  sp5-1.frascati.enea.it
STATUS           CPUF  JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV DISPATCH_WINDOW
ok              75.00     -     16     13     13      0      0      0      -

 CURRENT LOAD USED FOR SCHEDULING:
                r15s   r1m  r15m    ut    pg    io   ls    it   tmp   swp   mem
 Total           1.0   1.0   1.0   59%  82.7   312    5    54  554M   30G 3200M
 Reserved        0.0   0.0   0.0    0%   0.0     0    0     0    0M    0M    0M

 LOAD THRESHOLD USED FOR SCHEDULING:
            r15s   r1m  r15m    ut      pg    io   ls    it   tmp   swp   mem
 loadSched   -     -     -      -       -     -    -     -    -     -     -
 loadStop    -     -     -      -       -     -    -     -    -     -     -
```

LSF : il job

- Un job è identificato dal jobid
- Per vedere tutti i job attivi sull'utente corrente, uso il comando **bjobs** senza argomenti.
- **bjobs** -l <jobid> dà informazioni utili su un certo job
- Per uccidere un job, si usa **bkill** <jobid>
- I file prodotti dal job rimangono nella directory corrente, ma attenzione a <stdout> e <stderr>.

LSF : stato del job

- bjobs fa vedere gli stati del job.
  - **PEND**: il job è in "pending", ovvero attende risorse dal sistema. **bjobs -pl <jobid>** fa vedere le risorse mancanti.
  - **RUN**: il job sta girando. **bjobs** dice su che server.
  - **DONE**, **EXIT**: Si vedono solo con **bjobs -a** e per un periodo limitato di tempo. Significano "job finito" e "job uscito con errore".
  - **PSUSP**, **USUSP**, **SSUSP**: Job sospeso (in pending, da utente, da sistema).

LSF : output del job

- bsub supporta la ridirezione dell'output tramite i parametri:
  - -i <nomefile>: passa un file sullo <stdin>
  - -o <nomefile>: salva lo <stdout+stderr> del comando dentro ad un file
  - -e <nomefile>: separa <stderr> da <stdout> e lo salva dentro ad un file
- Se l'utente non specifica la loro destinazione, <stdout> ed <stderr> vengono persi.
- Nel nome file si può usare la macro %J per avere in automatico il jobid nel nome del file.
  - **bsub** -o output.%J ls -l

Compilazione e lancio di un job parallelo/1

Gli ambienti paralleli non sono omogenei sui cluster CRESCO di ENEA-GRID (fare riferimento alla documentazione online).

Step fondamentali su CRESCO Por (1/2), Fra, Bri, Cas.

- **Impostare il compilatore**

mpi-selector --list (vedo tutti i compilatori disponibili)

mpi-selector --set [flavour] (scelgo il compilatore)
mpi-selector --openmpi_gcc-1.2.8
exit e di nuovo login
mpi-selector --query
default:openmpi_gcc-1.2.8
level:user

Compilazione e lancio di un job parallelo/2

a questo punto il compilatore è inserito correttamente nel PATH e anche le librerie vengono individuate correttamente:

```
which mpicc
/usr/mpi/gcc/openmpi-1.2.8/bin/mpicc
echo $LD_LIBRARY_PATH
/usr/mpi/gcc/openmpi-1.2.8/lib
```

- Compilazione (programma in C):

```
mpicc mpi_hello.c -o mpi_hello
```

per i programmi paralleli è fortemente consigliato usare la versione parallela del compilatore perché gestisce bene tutte le variabili di ambiente e le flag di compilazione e link delle librerie.

Compilazione e lancio di un job parallelo/3

- Lancio dell'eseguibile:

- bqueues [-w]
  bqueues -l <nome coda> (prima bisogna scegliera la coda!)

bsub -o %J.out -e %J.err -n <n. core> -q <coda> ./sub_lsf.sh

Si è creato lo script (chmod u+x) sub_lsf.sh

```
#!/bin/sh

#Calcola il n. dei core via LSF
PROCS=`cat $LSB_DJOB_HOSTFILE | wc -l`

#Lo script blaunch.h esporta correttamente tutte le variabili di ambiente
# impostate e il token AFS sui nodi remoti

mpirun --mca btl self,sm,tcp --mca pls_rsh_agent "blaunch.sh" \
-np $PROCS  --hostfile $LSB_DJOB_HOSTFILE ./eseguibile
```

Stato e controllo del job sottomesso

Per vedere lo stato del job sottomesso:

bjob -l <JOBID>

- Il sistema dice in che stato si trova il job
- Se il job rimane in PEND per molto tempo bjobs -pl per maggiri info
- Se non si hanno info sul job significa che è terminato

bpeek <JOBID>

bpeek -f <JOBID>

per vedere l'output del job mentre viene prodotto

bkill <JOBID>

per terminare il job

Job multithreading/1

I job multithreading sono stati definiti come job che generano più thread su di un singolo host.

Questi non sono propriamente job paralleli, ma possono sfruttare l'hardware a più processori in modo molto efficiente.

Job multithreading/2

- Il comando **bsub** ha l'opzione **-n <nprocs>** che dice di utilizzare nprocs processori, presi da diversi host.
- Il comando bsub necessita quindi di 2 informazioni:
  - Il numero di thread (1 thread = 1 processore)
  - Il job deve girare su un singolo host
- Soluzione:
  - bsub -n <threads> -R "span[hosts=1]" ./job -i <input_file> -x <threads>
- span[hosts=1] richiede che tutti i processori siano sullo stesso host.

Job PENDING/1

- I job possono rimanere pending per diversi motivi:
  - Mancano CPU
  - Mancano le risorse richieste
  - L'amministratore ha chiuso un certo numero di host per manutenzione
- Capire perché un job è pending, non è semplice all'interno di un cluster.
- Il job rimane pending se il numero di CPU richiesto è minore dell'intersezione tra l'insieme delle cpu che rispondono alla stringa di resource requirement del job (bsub -R) e l'insieme delle CPU selezionate dalla coda (bqueues -l).

Job PENDING/2

- In generale, se un job è in pending per mancanza di risorse, LSF cerca di prevedere il tempo massimo di attesa in coda, all'interno del comando bjobs -l
  - Sun Sep 14 08:58:15: Job will start no sooner than indicated time stamp;
- Questo è comunque solo un'indicazione. Per effetto del gioco delle priorità tra code, il tempo può anche allungarsi.

Job PENDING/3

Strategie per non finire in pending

1. Usare sempre le code corrette: le code di durata minore sono prioritarie rispetto a quelle di durata maggiore.
2. Se la durata del job è minore della durata della coda, specificarlo con bsub -W
3. Se si usano code di durata breve e con meno di 512 processori, si può passare davanti ai job più grandi in attesa, per un tempo limitato, senza penalizzazioni.
4. Non specificare le risorse con bsub -R se non strettamente necessario: le code selezionano già le risorse giuste.

Variabili di ambiente LSF

LSB_JOBID: valore del jobid

LSB_JOBINDEX: indice del job (multi-caso)

LSB_MCPU_HOSTS: nome degli host selezionati e numero di cpu per ogni host

LSB_HOSTS: nome degli host selezionati (lo stesso host viene ripetuto se girano più processi)

LSB_SUB_HOST: host di sottomissione

LSB_QUEUE: coda di sottomissione

LSB_JOBPID: pid padre generato da LSF

LSB_JOBNAME: nome (comando) del job lanciato

LSB_DJOB_HOSTFILE: nome del file di host creato da LSF

Job multicaso/1

In alcuni casi è possibile scomporre job seriali di grandi dimensioni in tanti piccoli job.

- Un job seriale può girare solo su una macchina.
- Tanti piccoli job possono sfruttare tanti processori contemporaneamente e, quindi, ottenere prestazioni di ordini di grandezza superiori

NOTA BENE: è bene parlare con il proprio amministratore di sistema prima di lanciare un numero molto elevato di casi.

Job multicaso/2

- Vantaggi:
  - Risultato in tempi più brevi
  - Controllo centralizzato dei job
  - Miglior utilizzo delle risorse di calcolo
- Svantaggi:
  - Può essere invasivo
  - Ci si deve porre il problema della scalabilità
  - Preparazione del job più difficoltosa

Job multicaso/3

- Nel primo esempio, da una directory vuota, si lancia lo script “hello.sh” per 30 volte per vedere l'output generato

```
#!/bin/sh

echo "Ciao dal caso $LSB_JOBINDEX del job $LSB_JOBID"
echo "mi trovo sull'host `hostname`"
```

- Si lanci ora il job:
  - bsub -J 'hello[1-30]' -o output.%J.%I ./hello.sh

-q parallel per prova su Bri

Job multicaso/4

- Come lanciare 5 script che differiscono solo per il nome indicizzato ?
- launcher.sh:

```
#!/bin/sh

./caso.$LSB_JOBINDEX
```

- caso.1 (caso.2..5)

```
#!/bin/sh

echo "caso 1"
```

- Si lanci:
  - bsub -J 'multiscript[1-5]' ./launcher.sh

-q parallel -o out.%J.%I per prova su Bri

Job multicaso/5

- Un indice non basta… si possono avere 2 indici ?
- Sì. Supponiamo che la matrice sia 4x5.
  - Il numero di casi sarà 4x5=20 (supponiamo di lanciare bsub -J “test[1-20]” …)
  - Il primo indice si calcola come modulo della divisione tra $LSB_JOBINDEX-1 e 4
  - Il secondo indice è la divisione intera tra $LSB_JOBINDEX-1 e 4
- È anche possibile avere indici a più di 2 dimensioni: provare come esercizio.

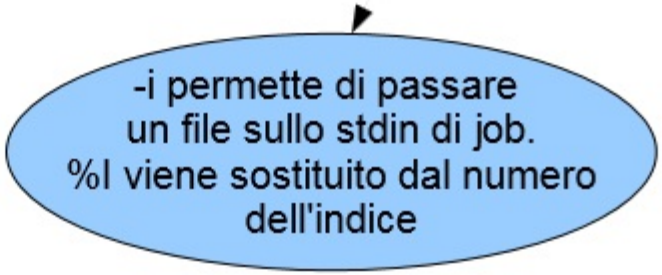
-q parallel -o out.%J.%I per prova su Bri

Job multicaso/6

Un modo per passare ad un eseguibile file di input multipli è quello di usare l'esempio script anziché richiamare n script diversi, si devono passare n file diversi.

Se il job supporta gli input tramite stdin, i può provare con

- bsub -J"multi_inp[1-20]" -i input.%I ./job



-q parallel -o out.%J.%I per prova su Bri

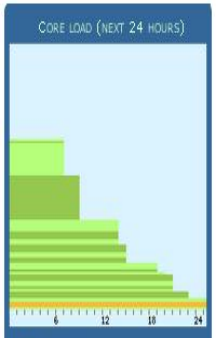
Strumenti online

JobRama

F.A.Q. Logout

CLUSTER: porticiUSER: allSTATUS: ALL

CORE LOAD (NEXT 24 HOURS)



103 total cores used (by jobs running on (= 16 cores)

free CRESCO computability area (currently free: 1757/4884 cores)

JOBID	User	Stat	Queue	CPU #	CpuTime	SubmitTime	Job TTL	JobOutput
143264	aviouser	RUNNING	avio	32		16/04/2013 08:31:32	214.39 h	Job Output
143117	villari	RUNNING	cresco_512h24	600		15/04/2013 13:21:09	6.83 h	Job Output
142925	eugenio	RUNNING	cresco_512h24	800		13/04/2013 17:40:26	8.66 h	Job Output
143197	denicola	RUNNING	cresco_128h24	160		15/04/2013 17:58:44	14.33 h	Job Output
143231	burresi	RUNNING	cresco_128h24	216		15/04/2013 21:15:05	14.33 h	Job Output
142362	sannino2	RUNNING	cresco_16h24	40		11/04/2013 09:21:34	6.83 h	Job Output
142367	sannino2	RUNNING	cresco_16h24	40		11/04/2013 09:25:20	6.83 h	Job Output
142962	grena	RUNNING	cresco_16h24	96		14/04/2013 11:43:56	14.49 h	Job Output
143336	zollo	RUNNING	cresco_16h24	128		16/04/2013 14:52:46	14.78 h	Job Output
142963	grena	RUNNING	cresco_16h24	96		14/04/2013 11:45:37	15.05 h	Job Output
143259	dinardo	RUNNING	cresco_16h24	96		16/04/2013 08:18:05	15.21 h	Job Output
142601	buonocor	RUNNING	cresco_16h24	128		12/04/2013 14:37:23	18.96 h	Job Output
143422	denicola	RUNNING	cresco_16h24	48		17/04/2013 03:29:19	19.03 h	Job Output
143424	denicola	RUNNING	cresco_16h24	48		17/04/2013 03:30:23	20.06 h	Job Output
142605	buonocor	RUNNING	cresco_16h24	128		12/04/2013 14:41:28	20.52 h	Job Output
143434	dinardo	RUNNING	cresco_16h24	88		17/04/2013 08:17:11	21.14 h	Job Output
143426	denicola	RUNNING	cresco_16h24	48		17/04/2013 03:31:18	21.24 h	Job Output
143428	denicola	RUNNING	cresco_16h24	48		17/04/2013 03:32:52	21.26 h	Job Output
143183	zhao	RUNNING	cresco_serh144	1		15/04/2013 17:25:40	102.28 h	Job Output
142551	vitali	RUNNING	cresco_serh144	1		12/04/2013 10:57:18	129.75 h	Job Output
143199	zhao	RUNNING	cresco_serh144	1		15/04/2013 18:02:30	102.97 h	Job Output
143350[1]	stravato	RUNNING	cresco_serh144	1		16/04/2013 15:37:34	126.33 h	Job Output
143350[3]	stravato	RUNNING	cresco_serh144	1		16/04/2013 15:37:34	126.40 h	Job Output
143350[25]	stravato	RUNNING	cresco_serh144	1		16/04/2013 15:37:34	126.62 h	Job Output

ENEA GRID

User Menu
- View Account
- Edit Account
- Notifications
- Inbox (1546)
- Logout

Main Menu
- Home
- xhelp

Summary | Create Ticket | My Profile | View All Tickets | Search | Go!

Filter Tickets:

Department:	State:	Status:	Ownership	Records per page	
Select All	Select All	Select All	Select All	20	Search

All Tickets

ID:	Priority:	Elapsed:	Last Update:	Status:	Subject:	Department:	Owner:	Logged By:
1212		1 day	1 day	Closed	Account req. Sergio Lo Meo ENEA ...	Calcolo scientifico	guarnier	arocchi
1211		4 days	2 days	Closed	Le macchine AIX di Frascati sono...	Calcolo scientifico	podda	kwb
1210		6 days	5 days	Closed	Account req. Adio Miliozzi ENEA ...	Calcolo scientifico	palombi	arocchi
1209		1 week	1 day	In progress	collegamento con SSH da cresco n...	Software di Sistema	guarnier	nunzio

https://jobrama.enea.it/

https://gridticket.enea.it/

Riferimenti

- **www.enea.it**
- **www.cresco.enea.it**
- **www.eneagrid.enea.it**
- **www.afs.enea.it**