ENEA-GRID: LSF

LSF: job paralleli avanzati

Ing. Fiorenzo Ambrosino, PhD
UTICT-HPC C.R. Portici (NA)

Corso ENEA-GRID –
Brindisi 25 giugno 2013

Sommario

Casi particolari di job complessi:

- ✓ Caso 1: Job con processi eterogenei
- ✓ Caso 2: Job con particolari richieste di risorse (licenze)

Flussi di Job

Regole generali

Nello script di lancio, le regole sono:

– Sostituire “ssh” con “blaunch.sh”:

Con openmpi bisogna mettere in testa a $PATH /afs/enea.it/software/bin/support/ssh, o usare l'opzione di mpirun –mca pls_rsh_agent “blaunch.sh” (se possibile)

– Contare i processori passati da LSF

PROCS=`cat $LSB_DJOB_HOSTFILE | wc -l`

– Usare $LSB_DJOB_HOSTFILE come parametro da passare a “hostfile” o “machinefile”.

Caso 1: processi eterogenei

Un utente aveva la necessità di lanciare 3 processi diversi all'interno dello stesso job mpi.
I 3 processi avevano finalità diverse, ma comunicavano tra loro in mpi:

- I processi andavano lanciati tutti all'interno dello stesso job di LSF
- Si doveva trovare un modo per distribuire i processi sui nodi

La soluzione proposta è basata su uno script.

Caso 1: processi eterogenei

```
#!/bin/sh
PROC1=1
NAME1=./a.out
PROC2=40
NAME2=./b.out
PROC3=15
NAME3=./c.out
PROCS=56
PATH=/afs/enea.it/software/bin/support/ssh:$PATH

cat $LSB_DJOB_HOSTFILE | head -$PROC1 > h1
cat $LSB_DJOB_HOSTFILE | head -`expr $PROC2 + $PROC1` | tail -$PROC2 > h2
cat $LSB_DJOB_HOSTFILE | tail -$PROC3 > h3
rm -f launchfile

for i in `cat ./h1`
do
echo "-np 1 --host $i $NAME1" >> launchfile
done

for i in `cat ./h2`
do
echo "-np 1 --host $i $NAME2" >> launchfile
done

for i in `cat ./h3`
do
echo "-np 1 --host $i $NAME3" >> launchfile
done

mpirun -app ./launchfile
```

Caso 2: Fluent

Fluent è un codice commerciale di fluidodinamica.

La sua integrazione con il mondo cresco è stata semplice: è bastato seguire la miniguida:

```
#!/bin/sh
PATH=/afs/enea.it/software/bin/support/ssh:$PATH
export PATH
PROCS=`cat $LSB_DJOB_HOSTFILE | wc -l`
fluent63 -t$PROCS -cnf=$LSB_DJOB_HOSTFILE $*
```

Caso 2: Fluent

Sono state definite le risorse relative alle licenze d'uso di tipo Accademic Research di Ansys:

- aa_r (per ogni istanza del programma)
- aa_r_hpc (per ogni core parallelo utilizzato)

Il comando lsload ci da informazioni sulle risorse dinamiche numeriche tra cui aa_r e aa_r_hpc. Si provi «lsload –s | grep» aa_r.

In fase di sottomissione utilizzare la richiesta:
-R rusage[aa_r=1:aa_r_hpc=1:duration=1]

Flussi di Job

LSF: i flussi di job

- Forse pochi utenti sanno che lsf può gestire i flussi di job.

- I flussi sono utili in molti ambiti:
  - Job dipendente dal file di output di un altro
  - Job che collezioni l'output di diversi job precedenti
  - Sequenze di job automatizzate (loop)

- Per pianificare un flusso di job, è necessario conoscere un po' di scripting, alcune nozioni di LSF ed un minimo di progettazione di algoritmi.

Un utile trucco: recupero del jobID

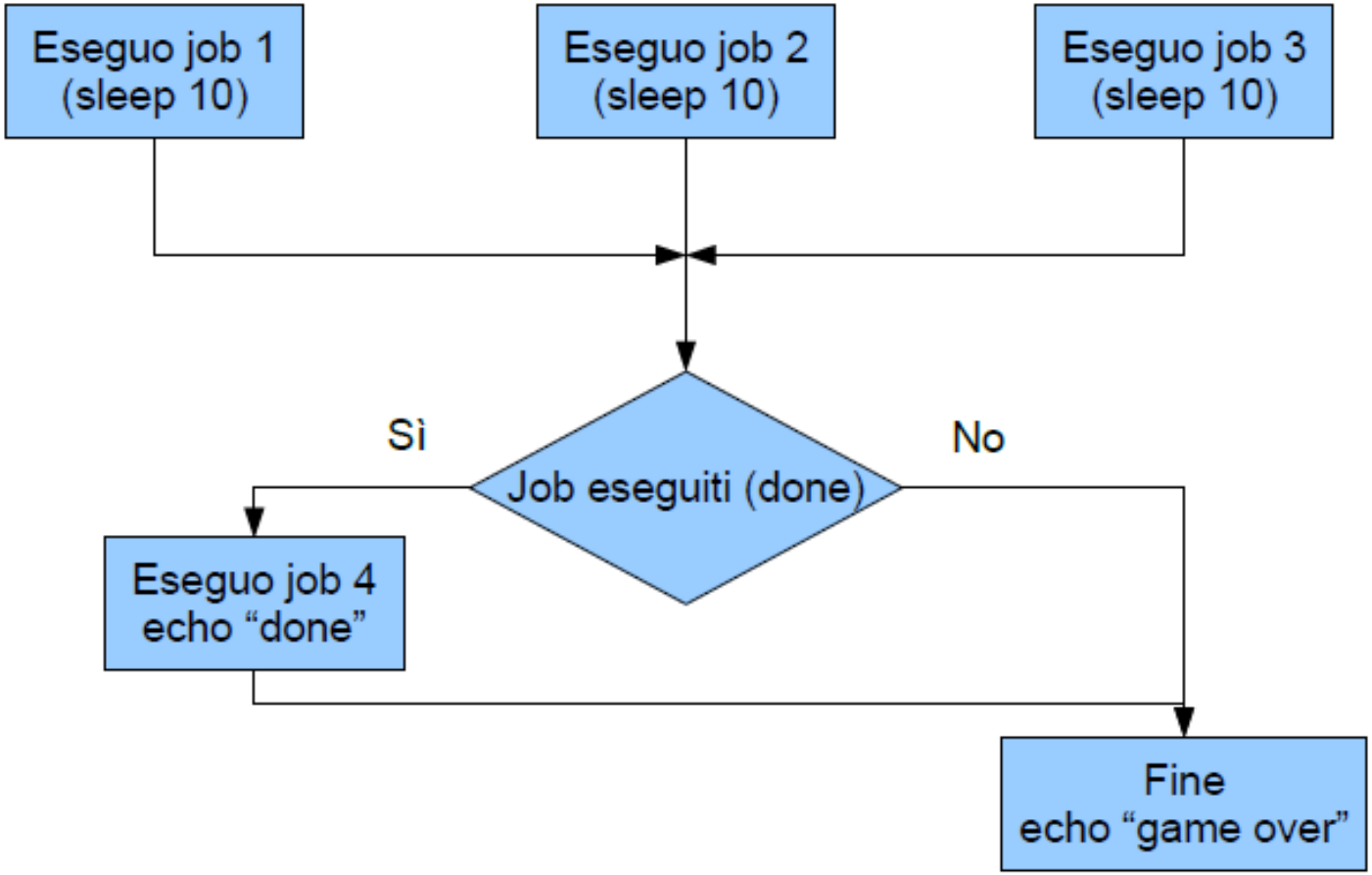
- ✓ In molti casi servirà avere esattamente il jobid come output di bsub:

```
bsub ... | awk -F'<' '{print $2}' | awk -F'>'
'{print $1}‘
```

- ✓ Si potrebbe mettere il filtro in uno script dentro la propria home directory (jobid.sh):

```
#!/bin/sh
awk -F'<' '{print $2}' | awk -F'>' '{print $1}'
```

Esercizio: lancio di 3 job ed attesa



Esercizio: lancio di 3 job ed attesa

```
sub.sh:
#!/bin/sh
JOBID1=`bsub sleep 10 | ./jobid.sh`
JOBID2=`bsub sleep 10 | ./jobid.sh`
JOBID3=`bsub sleep 10 | ./jobid.sh`
export JOBID1 JOBID2 JOBID3
DEPEND1=`echo "done($JOBID1) && done($JOBID2) && done($JOBID3)"`
JOBID4=`bsub -w "$DEPEND1" -o done ./done.sh | ./jobid.sh`
export JOBID4
DEPEND2=`echo "exit($JOBID1) || exit($JOBID2) || exit($JOBID3) ||
ended($JOBID4)"`
JOBID5=`bsub -w "$DEPEND2" -o clean ./clean.sh`

done.sh:
#!/bin/sh
echo “$JOBID1, $JOBID2, $JOBID3 done”

clean.sh:
bkill $JOBID1 $JOBID2 $JOBID3 $JOBID4 >/dev/null 2>&1
echo “game over”
```

Esercizio: lancio di 3 job ed attesa

Esercizio: utilizzo in pratica

- ✓ L'esercizio potrebbe essere utile quando
  - – è necessario aspettare la fine di una serie di job per collezionare gli output tramite uno script
  - – se i job non vanno a buon fine, lanciare uno script di pulizia
- ✓ Questo esempio, unito alla funzionalità di job array, o lancio di job paralleli, permette di creare veri e propri algoritmi di lancio dei job
  - – Un job in attesa potrebbe, al suo interno, lanciare un nuovo bsub, creando una condizione simile ad un ciclo “while”
- ✓ ATTENZIONE: la durata del flusso non può andare oltre la durata del token di AFS.

Grazie per la cortese attenzione!

Ing. Fiorenzo Ambrosino, Ph.D.

ENEA – UTICT-HPC

http://www.afs.enea.it/ambrosin

fiorenzo.ambrosino@enea.it