Allgemeine Informationen zum Compute-Server

Torque - das Stapeljob-System

TORQUE steht für "Terascale Open-Source Resource and QUEue Manager".

Grundsätzlich müssen alle Rechnungen als Jobs über das Stapeljob-System durchgeführt werden, nur die Programm-Entwicklung/Übersetzung darf auf dem Login-Knoten erfolgen. Das Stapeljob-System verteilt dann die Rechenjobs auf die verfügbaren Rechenknoten.

Zur Nutzung von Torque braucht man nur 2 Dateien, das Jobskript (eine Textdatei) und das eigentliche Programm (ein ausführbares Binary). Dabei ist zwischen seriellen (Abarbeitung auf nur einem Knoten) und knoten-übergreifenden Jobs (benötigen mehr als einen Rechenknoten) zu unterscheiden. Der Job wird eingereicht durch den Befehl:

qsub <Jobskript>

bzw.

qsub -I <Jobskript>

(interaktive Version: Input, Output und Fehlermeldungen erfolgen über das Terminal)

Das Jobskript

Mit einer speziellen Syntax wird der Job generiert und zur Abarbeitung eingereicht. Die Möglichkeiten sind (in Abhängigkeit von der konkreten Konfiguration von Torque) sehr groß. Hier werden einige Beispiele angegeben, die für die meisten typischen Jobs als Muster dienen können.

- Ein einfacher **serieller Job** hat im Jobskript 4 Teile, in der ersten Zeile die Angabe der Shell des Jobskriptes (hier bash), dann die Torque-Direktiven (sie beginnen alle mit #PBS), dann die Kommandos. Dazwischen oder dahinter sind Kommentare (beginnend mit #) möglich. Fast alle Direktiven dienen nur der bequemeren und sichereren Job-Steuerung und sind nicht unbedingt erforderlich, da es Default-Einstellungen gibt.

  Beispiel (serieller, d.h. nicht parallel rechnender Job):

  ```
  #!/bin/bash
  #PBS -N Jobname
  #PBS -j oe
  #PBS -l ncpus=1
  #PBS -l nodes=1:ppn=1
  #PBS -l walltime=00:10:00
  ```

```
#PBS -l mem=2gb
cd $PBS_O_WORKDIR
./meine.exe
```

Anmerkungen:

Das Zeichen "#" gehört zur Syntax und bedeutet nicht die Ankündigung eines Kommentars (wie bei Shell-Skripts).
#PBS -N "Jobname" ist ein frei wählbarer Name und dient der besseren Oranisation der Arbeit (optional).
#PBS -j oe legt fest, dass Output und Error in die selbe Datei geschrieben werden.
#PBS -l ... (kleiner Buchstabe L) fordert Ressourcen an oder legt deren obere Grenzen fest, ncpus=z fordert z Kerne an.
#PBS -l walltime fordert die maximale Verweilzeit im Zustand "running" von 10 Minuten und meme=2gb reserviert den maximal benutzten Hauptspeicher von 2 GByte.
Die Größe ncpus steht für die Anzahl der angeforderten Kerne (nicht der Rechenknoten oder der CPUs). Bei seriellen Jobs hat ncpus den Wert 1. Der Wert von ncpus ist wichtig für das "Eintüten" Ihrer Jobs in die vorhandenen Queues.

Es gibt in Torque auch die Ressourcen-Direktive
#PBS -l nodes=x:ppn=y
Diese Direktive steht in Konkurrenz zu ncpus=z. Das liegt an der Entwicklungsgeschichte von Torque (bzw. seinem Vorgänger PBS) in einer Zeit als jede CPU genau einen Rechenkern enthielt. Von einer sich widersprechenden Benutzung von "ncpus" und "nodes" wird im Internet abgeraten. Das ansonsten weitgehend identische Stapeljobsystem PBS Pro benutzt für Cluster mit MPI jetzt sogar neue Steuergrößen, z.B. "select" für die Anzahl der MPI-Prozesse und "mpiprocs" für die Anzahl der openMPI-Threads pro Prozess. Andererseits gibt es selbst in der offiziellen Torque-Dokumentation keine klare Erklärung der Torque-Direktiven bei Nutzung von openMPI bzgl. processes, processors, tasks, threads. **Bitte benutzen Sie beide Direktiven, d.h. auch #PBS -l nodes=1:ppn=1**.
Grundsätzlich sollten bei parallelen Jobs die Größem ncpus und np übereinstimmen.

Der Aufruf des auszuführenden Programms darf nicht mit der eigenständigen Festsetzung einer Priorität durch "nice" verbunden werden. Über die Priorität bei der Abarbeitung entscheidet Torque, niemand sonst. Entsprechende Jobs werden ohne Vorwarnung gelöscht.

ACHTUNG: Bitte starten Sie nicht mehrere Jobs in ganz kurzer Folge hintereinander! Das kann dazu führen, dass der Scheduler alle diese Jobs auf einen Kern legt (oder anderes unverhersagbares Verhalten).

In Torque werden automatisch eine Reihe von verwendbaren Umgebungsvariablen bereit gestellt. Das kann sehr nützlich sein. Im Beispiel bezeichnet $PBS_O_WORKDIR das Verzeichnis, von dem der Job eingereicht (abgeschickt) wird.

Die wichtigsten Direktiven sind:

	Bedeutung
-N name	Gibt als Jobnamen bei Ausgaben "name" an.
-j oe	Vereinige Standard-Output und Standard-Error in ein File.

	Bedeutung
-l nodes=n:ppn=p	Fordert für die Abarbeitung n Rechenknoten und dort je p Prozessorkerne an.
-l nodes=n:ppn=p:property	Fordert für die Abarbeitung n Rechenknoten mit der Eigenschaft "property" und dort je p Prozessorkerne an. Derzeit ist nur die Eigenschaft "bigmem" für die Knoten mit 8 Cores und 256 GB RAM einzutragen.
-l walltime=time	maximale Zeit des Jobs in Zustand "running" einer Queue. Angegeben im Format [hh:]mm:ss
-l mem=wert	maximaler Betrag des Jobs an physikalischem Memory. Die Angabe von "wert" erfolgt in mb oder gb.
-l file=size	Der Gesamtbetrag des Jobs an Festplattenkapazität. Die Angabe von "size" erfolgt in mb oder gb.
-l cput=time	Maximalsumme an CPU-Zeit, von allen Prozessen des Jobs. Angegeben im Format [hh:]mm:ss
-l pcput=time	Maximaler Betrag an CPU-Zeit von jedem einzelnen Prozess des Jobs. Angegeben im Format [hh:]mm:ss
-l pmem=wert	Maximaler Betrag an physikalischem Memory von jedem einzelnen Prozess. Die Angabe von "wert" erfolgt in mb oder gb.
-l ncpus	Die Anzahl der angeforderten Rechenkerne.
-V oder -v	Exportiere alle eigenen Umgebungsvariablen in den Job.
-M mail adresse	Schicke Job-relevante Mails an "mail adresse".
-m a|b|e	Schicke Mail bei: a=Abbruch, b=Beginn oder/und e=Ende
-A project name	Gibt als Projekt-Namen bei Ausgaben "project name" an.

- Ein **Job mit paralleler Abarbeitung** hat die gleiche Grundstruktur, wie ein serieller. Das auszuführende Programm ist jetzt aber der openMPI-Steuerung "mpirun" mit einigen Parameter zu übergeben. Das sieht z.B. so aus:

  ```
  #!/bin/bash
  #PBS -N Jobname
  #PBS -M mustermann@uni-potsdam.de
  #PBS -j oe
  #PBS -l ncpus=96
  #PBS -l nodes=2:ppn=48
  #PBS -l walltime=10:00:00
  #PBS -l mem=128gb
  cd $PBS_O_WORKDIR
  mpirun -np 96 ./meine.exe
  ```

  Anmerkungen:
  Im obigen Beispiel ginge auch:
  #PBS -l nodes=4:ppn=24
  oder
  #PBS -l nodes=8:ppn=12

  Unserem Cluster entsprechend wären der Maximalwert für die Ressourcen-Direktiven:
  #PBS -l ncpus=672
  #PBS -l nodes=14:ppn=48

Das wäre aber das ganze Cluster und ist nicht zugelassen. **Bitte benutzen Sie immer sowohl die Direktive mit ncpus als auch die Direktive mit nodes und ppn!**

Die Binaries für mpirun müssen schon bei der Compilierung auf openMPI vorbereitet werden. Die definitive Dokumentation findet man bei http://www.open-mpi.org/doc/. Man kann auch Nicht-MPI-Programme mit mpirun einreichen. Der aktuelle Pfad zu openMPI lautet /usr/mpi/gcc/openmpi-1.4.2/. Und dort die Verzeichnisse bin und lib64.
ACHTUNG:
Durch die Inbetriebnahme der PGI-Compiler und die Installation der IMSL-Bibliotheken sind zahlreiche neue Umgebungsvariablen wirksam (das können Sie z.B. durch den Befehl "set" überprüfen). Unter anderen haben wir jetzt auch drei (verschiedene!) mpirun und je nach Aufbau der Umgebungsvariablen PATH benutzen Sie verschiedene mpirun:
- /opt/pgi/linux86-64/10.9/mpi/mpich/bin/mpirun zu MPI-Compilationen mit einem PGI-Compiler
- /usr/mpi/gcc/openmpi-1.4.2/bin/mpirun zu openMPI-Compilationen mit einem GNU-Compiler
- /usr/mpi/gcc/mvapich2-1.5.1/bin/mpirun zur MPICH-Implementation von MPI
Das kann gelegentlich zu Verwirrung führen, bitte geben Sie dann in Ihrem Jobskript den vollen Pfad zu mpirun an.

Die wichtigsten Torque-Umgebungsvariablen (die im Jobskript benutzt werden können) sind:

	Bedeutung
PBS_O_WORKDIR	absoluter Pfad des Verzeichnisses, aus dem qsub aufgerufen wird
PBS_NODEFILE	Name der Datei, die die Liste der Knotennummern enthält, auf denen der Job läuft
PBS_O_HOME	Das Home-Verzeichnis des Einreichers
PBS_QUEUE	Name der Job-Queue des Jobs
PBS_JOBNAME	Vom Nutzer angegebener Job-Name
PBS_SHELL	Benutzte Jobskript-Shell
PBS_LOGNAME	Username des einreichenden Nutzers
PBS_JOBID	Die dem Job von Torque vergebene Job-Nummer
PBS_O_PATH	Pfad des Verzeichnisen des ausgeführten Progammes

Sinnvolle Anwendungen von Torque-Umgebungsvariablen (ein Jobskript-Beispiel aus der Literatur):

```
#PBS -N Test
#PBS -l ncpus=16
#PBS -l nodes=2:ppn=8
#PBS -l walltime=0:01:00
#PBS -e error.$PBS_JOBID
#PBS -V
#PBS -m ae
cd $PBS_O_WORKDIR
MYPROG=~/mustermann/bin/meine.exe
#Bestimme die Anzahl an zugewiesenen Kerne:
NCPU=`wc -l < $PBS_NODEFILE`
#Bestimme die Anzahl an freien Knoten
NNODES=`uniq $PBS_NODEFILE | wc -l`
MPIRUN=/usr/mpi/gcc/openmpi-1.4.2/bin/mpirun
export LD_LIBRARY_PATH="/usr/mpi/gcc/openmpi-1.4.2/lib64/"
CMD="$MPIRUN -np $NCPU $MYPROG"
echo "--> Running on nodes " `uniq $PBS_NODEFILE`
echo "--> Number of available cpus " $NCPU
echo "--> Number of available nodes " $NNODES
echo "--> Launch command is " $CMD
echo "--> Running in folder: "$PBS_O_WORKDIR
echo "--> Using mpirun :" $MPIRUN

$CMD > rediroutput.$PBS_JOBID
qstat -f $PBS_JOBID #Ausgabe des Ressourcenverbrauchs nach Beendigung des Progammes (hinter dem Aufruf des Programms)
```

Bitte benutzen Sie immer sowohl die Direktive mit ncpus als auch die Direktive mit nodes und ppn!

Ein Beispiel für einen Gaussian-Job (aus der Literatur, ohne MPI)

```
#PBS -N sample
#PBS -j oe
#PBS -S /bin/tcsh
#PBS -l cput=10:00:00,mem=500mb,ncpus=2
# Initialisiere die Umgebung:
setenv g03root /Pfad/nach/g03
source $g03root/g03/bsd/g03.login
# Kopiere das Input-File nach $TMPDIR
cp sample.com $TMPDIR
# der Aufruf Gaussian 03 mit Daten-Datei:
cd $TMPDIR
g03 < sample.com
# Kopiere das Ausgabe-File in das Home-Verzeichnis:
cp sample.log $HOME
# Schreibe die Jobinformationen in die Job-Ausgabe:
qstat -f $PBS_JOBID
```

Die Jobsteuerung

Die Torque-Kommandos zur Jobsteuerung werden einfach am Prompt eingegeben.

Die wichtigsten Befehle sind:

	Bedeutung
qsub script.pbs	Schickt das Jobskript script.pbs an Torque
qsub -I script.pbs	Schickt das Jobskript script.pbs zur interaktiven Job-Abarbeitung
qstat -a	Listet den Zustand der laufenden Jobs und Queues auf
qstat -n	Listet den Zustand aller Jobs auf, samt Node-Belegung
qstat -f job.ID	Listet die Eigenschaften des Jobs mit der Nummer job.ID auf
qdel job.ID	Beendet einen laufenden Job (Zustand R) mit der Nummer job.ID bzw. löscht ihn aus der Warteschlange (Zustand Q; der Befehl wirkt nicht bei anderen Zuständen des Jobs: C, E, H, S)
qdel -p job.ID	Beendet den laufenden Job mit der Nummer job.ID gewaltsam bzw. löscht ihn aus der Warteschlange
qhold job.ID	Hält den laufenden Job mit der Nummer job.ID an. Dieser Job kann mit dem Befehl qrls weitergeführt werden. Dieses Vorgehen heißt "checkpointing".
qrls job.ID	Freigabe eines angehaltenen Jobs zum weiteren Abarbeiten.

Anmerkungen:
Beim Einreichen eines Jobs mit qsub bekommt man von Torque die vergebene Job-Nummer job.ID mitgeteilt. Das ist eine laufende Nummer, die man zur Beobachtung und Dokumentation des Jobs benutzen kann. Der Hauptbefehl qsub kennt zahlreiche Optionen (Flags). Die sehr nützliche Eigenschaft des "Checkpointens" (Anhalten und Abspeichern eines laufenden Jobs mit allen Ressourcen und also der Möglichkeit des Weiterrechnens auf Befehl) ist vorgesehen, aber noch nicht realisiert.

Es sind noch keine Prioritäten (des Queues) definiert.

Einige praktische Erweiterungen im Jobskript

Im Internet gibt es praktische Templates für Jobskripts. Diese verbessern insbesondere die Möglichkeiten der Ein- und Ausgabe für Jobs. Googlen Sie. Hier ein für alle parallelen Jobs verwendbares Stück (**Bitte benutzen Sie immer sowohl die Direktive mit ncpus als auch die Direktive mit nodes und ppn!)**:

```
#!/bin/bash
#PBS -N batchtest
#PBS -j oe
#PBS -l ncpus=184
#PBS -l nodes=4:ppn=46
#PBS -l walltime=00:20:10
```

```
#PBS -l mem=8gb

cd $PBS_O_WORKDIR
NODES=`cat $PBS_NODEFILE | tr -s '\n' ' '`
NUM_PROCS=`cat $PBS_NODEFILE | wc -l`
NUM_NODES=`cat $PBS_NODEFILE | sort -u | wc -l`
export LOGFILE=batchtest_np${NUM_PROCS}_n${NUM_NODES}_$PBS_JOBID
echo "------------------------------------------------------" > $LOGFILE
echo "Starting at" `date` >> $LOGFILE
echo "" >> $LOGFILE
# Anzahl an Cores
NUMPROCS=`wc -l < $PBS_NODEFILE`
# Anzahl verschiedener Knoten im PBS_NODEFILE:
NODEROWS=`uniq ${PBS_NODEFILE} | wc -l`
# Anzahl eingesetzter Cores (np):
CORES=`cat $PBS_NODEFILE | wc -l` >> $LOGFILE
echo "Job $PBS_JOBID eingereicht von $PBS_O_HOST">> $LOGFILE
echo "" >> $LOGFILE
echo "Aktuelles Verzeichnis: `pwd`" >> $LOGFILE
echo "" >> $LOGFILE
#echo "Die reservierten Knoten:" >> $LOGFILE
#echo " $NODES" >> $LOGFILE
#echo "" >> $LOGFILE
echo "Running auf $NUM_PROCS Kernen auf $NUM_NODES Knoten." >> $LOGFILE
echo "Positionen im File PBS_NODEFILES (zugewiesene Cores) = "$NUMPROCS >> $LOGFILE
echo "Knotenzahl im File PBS_NODEFILES (zugewiesene Nodes) = "$NODEROWS >>
$LOGFILE
echo "CORES:" $CORES >> $LOGFILE
echo "" >> $LOGFILE
echo "Der Job laeuft auf den Kernen: " >> $LOGFILE
echo `cat $PBS_NODEFILE` >> $LOGFILE
echo "" >> $LOGFILE
set| grep PBS >> $LOGFILE
mpirun -np $NUMPROCS --hostfile $PBS_NODEFILE ./helloworld-mpi >> $LOGFILE
echo "Finishing at" `date` >> $LOGFILE
echo "------------------------------------------------------" >> $LOGFILE
```

Die Queues

Das Stapeljobsystem benutzt eine Routing Queue und 5 Execution Queues. Nach Einreichen eines Jobs (durch qsub) in die Routing Queue wird der Job entsprechend seinen Anforderungen automatisch in die passende Execution Queue "eingetütet". Dort steht er in der Warteschlange bis Kapazität zur Abarbeitung frei ist. Wenn ein Job (im Jobskript oder als optionaler Parameter beim Einreichen) unpassende Ressourcen anfordert, wird er nicht abgearbeitet. Jobs mit geringeren Ressourcen-Forderungen haben mehr Chancen, dass ausreichend Resourcen zum Starten frei sind.

Queue	max. Knotenzahl	max. Anzahl an CPU-Kernen	max. Laufzeit	max. Memory
batch (Express- und Test-Queue, für serielle Jobs)	1	1	10 Minuten	1 GByte
short (Standard-Queue für normale parallele Jobs)	1	48	24 Stunden	64 GByte
single (für serielle Jobs mit mittl. Laufzeit)	1	1	24 Tage	8 GByte
long (Standard-Queue für große parallele Jobs)	2	96	180 Tage	128 GByte
huge (Ausnahme, für sehr große parallel-rechnende Jobs) **nur auf Anfrage verfügbar!**	10	480	48 Stunden	640 GByte

Für jede Queue gibt es eine Maximalanzahl an gleichzeitig laufenden Jobs.
Für jede Queue gibt es eine Maximalanzahl an Rechenknoten, die ein Job benuzten darf.
Für jede Queue gibt es eine Maximalanzahl der Summe aller durch die laufenden Jobs benutzten Rechenknoten.
Für jeden Nutzer gibt es in jeder Queue Obergrenzen für die Anzahl gleichzeitig laufender Jobs.
Je nach Auslastung des Rechners kann die Definition von Queues befristet geändert werden.

Bitte steuern Sie die Einordnung Ihrer Jobs in die Queues primär durch die Ressourcen ncpus und mem.

ACHTUNG: In der Queue "huge" darf maximal ein Prozess aktiv sein, da Jobs in dieser Queue praktisch den gesamten Cluster belegen können. Für wirklich große Jobs empfehlen wir den Norddeutschen Verbund für Hoch- und Höchstleistungsrechnen HLRN, dem das Land Brandenburg 2013 beigetreten ist.

Kontakt

Universität Potsdam

ZIM - Zentrum für Informations-
technologie und Medienmanagement
Am Neuen Palais 10
14469 Potsdam

Tel.: +49 331 977-4444
Fax: +49 331 977-1750
E-Mail: zim-service@uni-potsdam.de