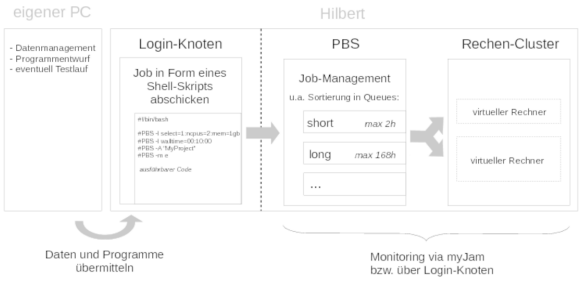
Unser Batchsystem: PBSPro

- Kommunikation mit dem Batchsystem
- Das Shell-Skript
- Ein erweitertes Shell-Skript
- Paralleles Arbeiten im Shell-Skript

Das Batchsystem verwaltet die eingegangenen Arbeitsaufträge und startet deren Ausführung auf den eigentlichen Rechenknoten des Clusters.

Die folgende Abbildung gibt einen Überblick über den gesamten Arbeitsablauf eines Arbeitsauftrags.

1. Die Entwicklung der Programme bzw. Skripte und gegebenenfalls das Datenmanagement findet in der Regel auf dem eigenen Rechner statt.
2. Die Programme/Skripte und Daten werden auf den Festplattenspeicher des Login-Knoten übermittelt.
3. Ein Shell-Skript wird an das Batchsystem geschickt. Dieses Skript enthält sowohl Meta-Informationen zum Auftrag, als auch den eigentlichen Auftrag in Form eines Skripts oder eines Programmaufrufs.
4. Das Batchsystem sortiert es gemäß Eigenschaften des Auftrags in Warteschlangen ein, konfiguriert gemäß der Meta-Informationen den angefragten virtuellen Rechner und startet die Bearbeitung auf diesem.
5. Während der Bearbeitungszeit kann entweder über das Batchsystem oder über das Monitoring-Tool "myJam" der Status des Auftrags überwacht werden.



Kommunikation mit dem Batchsystem

Die Kommunikation geschieht über spezielle Konsolen-Befehle. Die folgende Liste führt häufig verwendeten Befehle auf. Eine vollständige Dokumentation gibt es hier zum Download.

1. Ein Arbeitsauftrag an das Batchsystem senden:

   ```
   qsub SHELL-SCRIPT
   ```

   "SHELL-SCRIPT" bezeichnet hier den Namen des entsprechenden Skripts, dass die Meta-Informationen und die eigentliche Befehle bzw. Programmaufrufe des Arbeitsauftrags umfassen. Die Inhalte und Struktur eines solchen Skripts wird weiter unten besprochen. Als Antwort auf die Übermittlung wir Ihnen die Job-ID mitgeteilt mit deren Hilfe Sie den Status des Arbeitsauftrags überwachen können.
2. Einen bereits übermittelten Arbeitsauftrag (gegebenenfalls anhalten und) löschen:

   ```
   qdel JOB-ID
   ```

   "JOB-ID" bezeichnet die Job-ID Ihres Arbeitsauftrags, die Ihnen beim übermitteln des Arbeitsauftrags zugeteilt wurde (siehe "qsub").
3. Statusinformationen abrufen:

   ```
   qstat -f JOB-ID
   qstat -u ACCOUNT-NAME
   qstat -q
   ```

   In der ersten Zeile werden Informationen zu einem konkreten Arbeitsauftrag abgerufen. In der zweiten Zeile werden Informationen zu allen Arbeitsaufträgen, die unter Ihrem Namen übermittelt wurden, abgefragt. In der letzten Zeile wird der Status (Auslastung, etc) der Warteschlangen des Batchsystems ausgelesen. (Weiterführende Informationen: Kapitel 11 im PBS Pro User's Guide)

Das Shell-Skript

Das folgenden Minimalbeispiel illustriert den Aufbau und den Inhalt eines typischen Shell-Skripts.

startScript1.sh (Minimalbeispiel)

```
#!/bin/bash
#PBS -l select=1:ncpus=2:mem=3gb
#PBS -l walltime=01:59:00
#PBS -A "HPC-Project"

set -e

module load R

R CMD BATCH --slave R-Script.R R-Output.Rout
```

Mit dem Ausdruck in Zeile (1) wird mitgeteilt, dass es sich in der Folge um ein Bash-Skript handelt. (2) -(4) Beinhalten PBS-Direktiven, Anweisungen für das Batchsystem. In Zeile (2) wird mit dem Schlüsselwort "`select`" angegeben wieviele Rechenknoten involviert sein sollen (ein Rechenknoten umfasst i.d.R. mehrere Prozessoren und diese wieder mehrere Kerne). Mit dem Schlüsselwort "`ncpus`" wird die Anzahl an Kernen angegeben die je Knoten benutzt werden sollen. Mit "`mem`" der zu reservierende Arbeitsspeicher je Knoten festgelegt. Zuletzt wird mit "`arch`" die angefragte Rechnerarchitektur angegeben. Wird dieses Attribut nicht gesetzt, so wählt das Batchsystem selbstständig eine passende.

"`walltime`" beschreibt die erwartete Rechendauer. Gemäß der Rechendauer wird der Arbeitsauftrag in eine passende Warteschlange eingruppiert. Unter anderem gibt es die drei Warteschlangen, die sich insbesondere in der maximalen Rechendauer unterscheiden (I) "short" (weniger als 2h), (II) "work" (weniger als 72h bzw. 3 Tage) und (III) "long" (weniger als 168h bzw. 1 Woche).

In der letzten PBS-Direktive in Zeile (4) wird angegeben zu welchem beantragten Projekt dieser Arbeitsauftrag gehört. Das "HPC-Project" ist hierbei der Projekt-Name, der beim Antrag zur Nutzung des HPC-Clusters angegeben wurde.

Nach den PBS-Direktiven kommt der eigentliche Arbeitsauftrag in Form des Bash-Skripts. Mit "module ..." wird dem System mitgeteilt welche Softwareumgebung bereitgestellt werden soll. In diesem Beispiel soll eine Softwareumgebung mit der Statistiksoftware R bereitgestellt werden.

Mit dem Befehl in der letzten Zeile wird R im Batch-Mode gestartet und führt das angegebene R-Skript aus und speichert den Konsolen-Output (Fehlermeldungen, Mitteilungen, Ausgaben, etc.) in die Datei "R-Output.Rout".

Ein solches Skript würde dann mit "`qsub startScript1.sh`" an das Batchsystem übermittelt werden.

Ein erweitertes Shell-Skript

Im folgenden Shell-Skript wird, wie im Minimalbeispiel, R für die Abarbeitung eines R-Skript gestartet. Zudem kommen zwei nützliche Erweiterungen hinzu.

1. Es wird eine Log-Datei angelegt, in die automatisiert Informationen zum Arbeitsauftrag gespeichert werden. Dies erleichtert bei Problemen während der Auftragsausführung die Fehlersuche.
2. Das gesamte Arbeitsverzeichnis wird auf das sehr performante Scratch-Laufwerk (GPFS) kopiert. Dort wird dann die Ausführung gestartet und nach Abschluss wird das gesamte Verzeichnis mitsamt der gespeicherten Resultate zurück kopiert. Diese Ergänzung ist immer dann sinnvoll, wenn innerhalb des Arbeitsauftrag eine intensive Festplattennutzung stattfindet (wenn z.B. viele/große Dateien eingelesen werden sollen oder häufig Resultate gespeichert werden müssen).

startScript2.sh (erweitertes Beispiel)

```
#!/bin/bash
#PBS -l select=1:ncpus=2:mem=3gb
#PBS -l walltime=01:59:00
#PBS -A "HPC-Project"
set -e

## Log-File definieren
export LOGFILE=$PBS_O_WORKDIR/$PBS_JOBNAME"."$PBS_JOBID".log"

##Scratch-Laufwerk definieren und erzeugen
SCRATCHDIR=/gpfs/scratch/$USER/$PBS_JOBID
mkdir -p "$SCRATCHDIR"

##Information zum Start in das Log-File schreiben
cd $PBS_O_WORKDIR
echo "$PBS_JOBID ($PBS_JOBNAME) @ `hostname` at `date` in "$RUNDIR" START"
> $LOGFILE
echo "`date +"%d.%m.%Y-%T"`" >> $LOGFILE

##Software-Umgebung laden
module load R

##Daten vom Arbeitsverzeichnis auf das Scratch-Laufwerk kopieren
cp -r $PBS_O_WORKDIR/* $SCRATCHDIR/.
cd $SCRATCHDIR
rm $PBS_JOBNAME"."$PBS_JOBID".log"

##R-Aufruf
R CMD BATCH --slave R-Script.R R-Output.Rout

##Daten zurück kopieren
cp -r "$SCRATCHDIR"/* $PBS_O_WORKDIR/.
cd $PBS_O_WORKDIR

##Verfügbare Informationen zum Auftrag in das Log-File schreiben
echo >> $LOGFILE
qstat -f $PBS_JOBID >> $LOGFILE

echo "$PBS_JOBID ($PBS_JOBNAME) @ `hostname` at `date` in "$RUNDIR" END"
>> $LOGFILE
echo "`date +"%d.%m.%Y-%T"`" >> $LOGFILE
```

Paralleles Arbeiten im Shell-Skript

Häufig bieten die genutzten Programme wie Matlab, R, usw. Möglichkeiten die gewünschten Berechnungen auf mehrere Prozessoren zu verteilen. Eine weitere Möglichkeit ist direkt in der Shell gegeben. Der folgende Code zeigt ein Skript, in dem zuerst zwei R-Instanzen geöffnet werden, die jeweils ein Skript verarbeiten. Das "&"-Zeichen am Ende der Zeile weist das System an, nicht auf die Fertigstellung zu warten und beginnt dann parallel mit dem nächsten Aufruf. Der Befehl "wait" bewirkt, dass bei der Abarbeitung des Batch-Skripts solange gewarte wird, bis die zuvor begonnenen Programmaufrufe beendet sind. Daran schließen sich zwei weitere Aufrufe von R an. Ein solche Skript muss immer mit einer "wait"-Anweisung enden.

startScript1.sh (Minimalbeispiel)

```
#!/bin/bash
#PBS -l select=1:ncpus=2:mem=3gb
#PBS -l walltime=01:59:00
#PBS -A "HPC-Project"

set -e

module load R

R CMD BATCH --slave R-Script1.R R-Output1.Rout &
R CMD BATCH --slave R-Script2.R R-Output2.Rout &

wait

R CMD BATCH --slave R-Script3.R R-Output3.Rout &
R CMD BATCH --slave R-Script4.R R-Output4.Rout &
wait
```

Das parallele Arbeiten in der Shell kann auch durch Programme wie "parallel" (benötigt "module load Parallel") stark angepasst und feiner gesteuert werden.