HPC-Scheduling mit PBS Pro

Altair

Summer 2010

PBS Professional – New in 10.x!

OS Provisioning	Hooks		Eligible time	Fail-over
Scheduling Formula	Policy-based Scheduling		$restrict_user	Checkpoint / Restart
Fairshare	Standing Reservations		Preemption	Job Arrays
High-availability Reservations	Kerberos	Estimated Job Start Times	Dynamic Resources	Topology Placement Sets
Web Services	Age-based Scheduling	Huge Scalability	Interactive Jobs	Job History (qstat –x)
Job Dependencies	License Scheduling	Green Provisioning	User/Group VIP Limits	Multi-core
24x7 On-line Community	EAL3+ Security	Over-subscription	Meta-scheduling	Peer Scheduling
On-demand Licensing	GPU Scheduling	Script-less Job Submission	MPI Integrations	Backfill TopN

Job Management

Job Submit mittels qsub:	Erläuterung:
`#PBS -N MyJob`	Name
`#PBS -l mppwidth=128`	Anzahl MPI-Prozesse
`#PBS -l mppnppn=4`	Anzahl MPI-Prozesse pro Node (opt)
`#PBS -l mppdepth=6`	Anzahl OpenMP-Threads pro Prozess
`#PBS -l walltime=01:00:00`	Laufzeit
`#PBS -j oe`	Behandlung von stdout/stderr (opt)
`export OMP_NUM_THREADS=6`	Für OpenMP Runtime
`aprun -B a.out`	aprun -B: aprun-Parameter werden aus der Batch-Umgebung ausgelesen

Weitere User-Kommandos:

Anzeigen der Jobs bzw der Queues: qstat [-a|-f|-Q]

Löschen von Jobs: qdel

Job vom Scheduling ausnehmen und rückgängig machen: qhold und qrls

Vorschlag zur Queue-Struktur

- User müssen keine Queue angeben
  - Jobs werden nach Größe (in Core-Anzahl) und Laufzeit automatisch in die richtige Queue geleitet
  - Passt der Job in keine Queue, wird er nicht angenommen
- Vorhandene Queues:
  - testq: bis zu 96 Cores, max. 2 Stunden Walltime
  - c1kt48h: bis zu 1.024 Cores, max. 48 Stunden Walltime
  - c2kt8h: bis zu 2.048 Cores, max. 8 Stunden Walltime
  - c4kt2h: bis zu 4.096 Cores, max. 2 Stunden Walltime
- Bei Erreichen des Walltime-Limit werden Jobs abgebrochen
- Weitere Einstellungen:
  - Zahl der laufenden Jobs pro Queue
  - Zahl der laufenden Jobs pro User pro Queue
  - Zugang mittels User Id

Flexible Jobprioritäten mittels Formel: Ansatz

- Queues zur Definition von Jobklassen
  - Limits für Zahl der laufenden Jobs oder belegten Nodes
  - ACLs für User und Gruppen
- Formel zur Berechnung der Jobpriorität und damit der Startreihenfolge
  - Reihenfolge strikt einhalten (strict_ordering: true)
  - Verbesserung der Auslastung mittels Backfill (backfill_depth)
  - Jede Ressource (vordefiniert oder selbst hinzugefügt) kann in der Formel verwendet werden plus einige spezielle Ressourcen (queue_priority, eligible_time)
  - Verfolgung für Admins (Scheduler-Log) und für User (est_start_time_freq, qstat –T, erst mit PBS 10.4)

Flexible Jobprioritäten mittels Formel: Vorschlag

```
job_sort_formula =

Wsize * mppwidth *
mppdepth +

Wwait * eligible_time /
3600.0 +

Wqueue * queue_priority +

Wuser * user_p +

special_p
```

Bevorzugung:

Großer Jobs

Lange wartender Jobs

Jobs in High Prio Queues

Bestimmter User

„Admin-Knopf“

Flexible Jobprioritäten mittels Formel: Selbstdefinierte Ressourcen

```
job_sort_formula =

Wsize * mppwidth
*mppdepth +

Wwait * eligible_time /
3600.0 +

Wqueue * queue_priority +

Wuser * user_p +

special_p
```

Berücksichtigung selbst-definierter Ressourcen (server_priv/resourcedef)

- Flags r und i für read-only und invisible Ressourcen
- Setzen per Queue- oder Server-Defaults
- Setzen per qsub-Hook (Python): Nachschlagen von user_p, z.B. aus Allocation Management-System

Admin-Knopf:

qalter -l special_p=1000000 jobid

Flexible Jobprioritäten mittels Formel: Information

Information für Admins und User:

```
% qstat -T

                                                          Req'd Req'd     Est
Job ID  Username Queue Jobname  SessID NDS TSK Memory Time  S Start
------- -------- ----- -------- ------ --- --- ------ ----- - -----
5.quark  bill    workq  foo      12345  1   1   128mb 00:10 R   -
9.quark  bill    workq  bar        --   1   1   128mb 00:10 Q 11:30
10.quark bill    workq  gril       --   1   1   128mb 00:10 Q 11:40
7.quark  bill    workq  baz        --   1   1   128mb 00:10 Q Tu 18
```

```
% qstat -f
     . . .
     estimated.exec_vnode = (pepsi:ncpus=1)
     estimated.start_time = Tue Apr 27 11:52:44 2010
```

Wenn die Startzeit nicht ausreicht, kann mittels der Gewichte oder special_p nachjustiert werden.

Flexible Jobprioritäten mittels Formel: Weitere Konfiguration

Weitere Server- und Scheduler-Konfiguration unterstützt die Formel

- Ressourcen-Definition in server_priv/resourcedef
- Queue-Definition mit Limits, ACLs, Defaults, Queue-Priorität
- Scheduler-Konfiguration
  - Wartezeit (eligible_time_enable)
  - Backfill (backfill_depth)
  - Startzeit-Vorhersage (est_start_time_freq)
  - Gleitende Berücksichtigung der Queue-Priorität (round_robin: false und by_queue: false und sort_queues: false)
- qsub/qalter-Hook: Abfrage eines externen Tools für Allocation Management o.ä. zur Ermittlung der User-Priorität

HPC-Scheduling mit PBS Pro

- Ressourcen-Engine
- Scheduling-Optionen
- Numerische Formel für Jobprioritäten
- Python-Hooks

erlauben flexibles und verlässliches Scheduling mit PBS Pro