TECHNISCHE UNIVERSITÄT DRESDEN

Center for Information Services and High Performance Computing (ZIH)

Linux Cluster in Theorie und Praxis

Batchsysteme

19. November 2009

INF 1046
Nöthnitzer Straße 46
01187 Dresden
0351 - 463 38783

Andy Georgi

ZIH
Center for Information Services & High Performance Computing

Foliensatz

Verfügbarkeit der Folien

Vorlesungswebseite:
`http://tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/lehre/ws0910/lctp`

Inhalt

Batchsysteme

Definition

Bei einem **Batchsystem** handelt es sich um eine Softwarelösung, die es ermöglicht, eine Menge von unterschiedlichen Jobs von verschiedenen Nutzern nach dem Stapelverarbeitungsprinzip abzuarbeiten.
Prinzipiell besteht ein Batchsystem aus zwei Komponenten - dem **Job Management System** und dem **Job-Scheduler**. Dabei werden von den Nutzern alle erforderlichen Programme, Daten und Anweisungen zur Ablaufsteuerung zusammengestellt und an das Job Management System übergeben. Dieses gibt anschließend die Informationen über die angeforderten Ressourcen und deren Zustände an den Job-Scheduler, welcher in Abhängigkeit der Scheduling-Kriterien, die Reihenfolge der Ausführung und die dafür notwendigen Betriebsmittel festlegt.

Inhalt

Job Management Systems

Portable Batch System (PBS)

Überblick I

- POSIX Standard 1003.2d [IEEE] konformes Stapelverarbeitungssystem
- Entwicklung von 1993 bis 1997 am NASA Ames Research Center [NARC]
- Ziele:
  - Ablösung des bis dahin verwendeten Network Queueing System (NQS)
  - Abdeckung eines möglichst breiten Spektrums an Computerarchitekturen
- Aktuell von der Firma Altair Grid Technologies, LLC [ALTAIR] weiterentwickelt und vertrieben
- Verfügbare Versionen: OpenPBS (Open Source) und PBS Pro (kommerziell)
- Einschränkungen bei Verwendung von OpenPBS:
  - Geringere Fehlertoleranz
  - Eingeschränktes Job Scheduling
  - Fehlender Support

Überblick II

- Aufgrund der genannten Nachteile von OpenPBS wurde die Original PBS Version von Cluster Resources Inc. [CRI] weiterentwickelt
- Es entstand der **T**era-scale **O**pen Source **R**esource and **Que**ue Manager (TORQUE) [TORQUE], welcher aktuell über 1200 Patches enthält
- Entwicklungsschwerpunkte
  - Erhöhung der Fehlertoleranz
  - Erweitertes Scheduling Interface
  - Skalierbarkeit

Architektur I

- Verteilter Aufbau aus drei Komponenten:
  - Job Server
  - Job Scheduler
  - Job Executor oder MOM (Machine Oriented Miniserver)
- Modulare Implementierung und standardisierte Kommunikation zwischen den Komponenten ermöglichen den Austausch einzelner Module

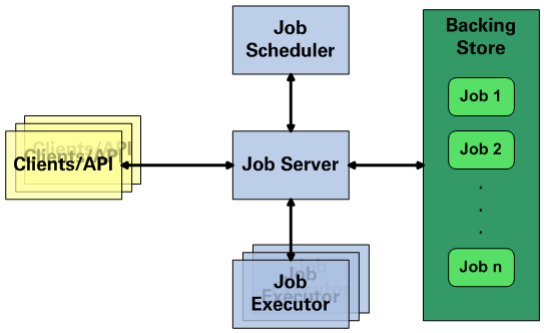


Abbildung: Architektur des Portable Batch Systems

Architektur II

Job Server (`pbs_server`):

- Zentrale Komponente des Systems
- Läuft i.d.R. nur auf einem Knoten, welcher dann der Einzige ist mit dem die Nutzer interagieren können
- Bereitstellung von Informationen über Jobs und Ressourcen
- Aktualisierung mit Hilfe der einzelnen Job Executoren
- Keinerlei Scheduling-Logik implementiert
- Verantwortlich für die Fehlertoleranz durch redundante Speicherung der Jobs bis zur vollständigen Abarbeitung auf einem Backing Store

Architektur III

Job Scheduler (`pbs_sched`):

- Implementiert die Scheduling-Logik
- Scheduling-Entscheidung wird aufgrund der Informationen des Job Servers getroffen
- Job Server wird angewiesen diese auszuführen
- Läuft häufig nur auf einem Knoten, zusammen mit dem Job Server

Architektur IV

Job Executor (`pbs_mom`):

- Verantwortlich für die Ausführung nach der Übermittlung einer Kopie des Jobs durch den Job Server
- Gibt auf Anfrage Informationen über dessen aktuellen Status zurück
- Hohe Plattformverfügbarkeit (z.B. Linux, AIX, SunOS, UniCos, IRIX)
- Läuft auf den Computenodes des Clusters

Kommandos (Auszug)

Kommandos zur Steuerung von Jobs (siehe "man pbs"):

- `qsub` - Batch Job abschicken
- `qalter` - Attribute eines Batch Jobs ändern
- `qdel` - Batch Job löschen
- `qhold` - Batch Job in der Queue anhalten, damit er vom Scheduler nicht zur Ausführung gebracht wird
- `qmove` - Batch Job in eine andere Queue verschieben
- `qmsg` - Einen Kommentar in die Ausgabe eines Batch Jobs schreiben
- `qrerun` - Einen laufenden Job beenden und neu in die Queue schicken
- `qrls` - Gegenteil von qhold
- `qsig` - Signal an laufenden Job senden
- `qstat` - Status von Batch Jobs und Queues abfragen

Jobscript

- Beispiel:

```
1   #!/bin/sh
2   #PBS -V
3   # Name des Jobs
4   #PBS -N myjobname
5   # Ausgabedateien
6   #PBS -j oe
7   # Eine Email an den Benutzer schicken, wenn der Job abgearbeitet ist
8   #PBS -m ae
9   # Email-Adresse des Benutzers
10  #PBS -M benutzer@server.de
11  # Name der Queue
12  #PBS -q queue_name
13  # Knoten anfordern
14  #PBS -l nodes=8,walltime=HH:MM:SS
15  # Das ist das Arbeitsverzeichnis des Jobs
16  echo Working directory is $PBS_O_WORKDIR
17  cd $PBS_O_WORKDIR
18  echo Running on host 'hostname'
19  echo Time is 'date'
20  # Das Programm ausführen
21  myprogram + args
```

- Übergabe des Jobscripts an PBS erfolgt mit Hilfe von `qsub`

xpbs - Ein graphisches Interface für PBS

- Zusammenstellung von Jobscripten und deren Übermittlung mit Hilfe einer graphischen Oberfläche
- Unterstützt nicht alle Konfigurationsparameter

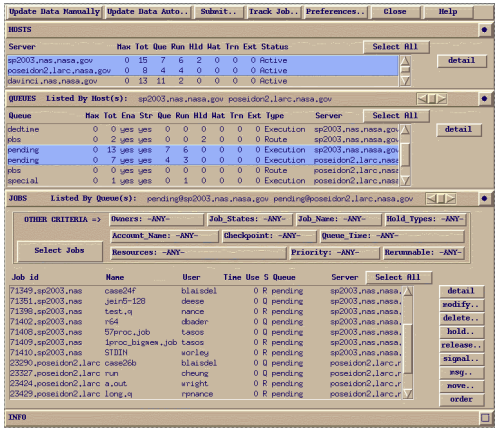


Abbildung: xpbs Display [XPBS]

Darstellung der Statusinformationen mit xpbsmon

- Graphische Darstellung des Clusters
- Bereitstellung von Statusinformationen zu jedem Computenode

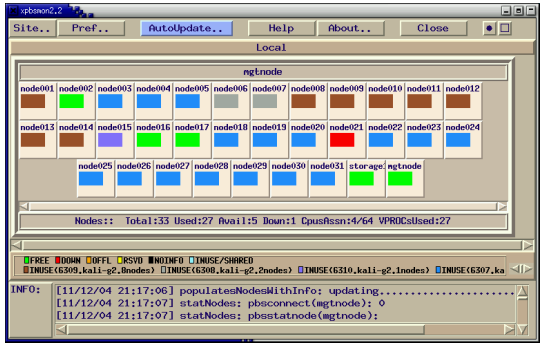


Abbildung: xpbsmon Display [XPBSMon]

Jobverwaltung und Monitoring mit PBSWeb

- Erstellung von Jobscripten und Abfrage von Statusinformationen mit PBSWeb [PBSWeb] über eine HTML-Schnittstelle

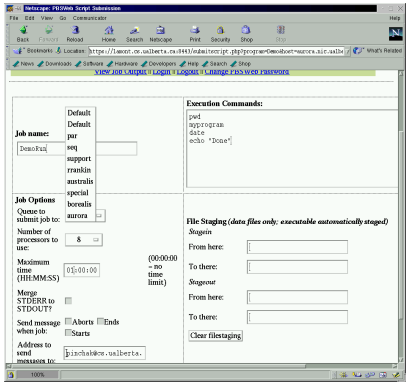


Abbildung: Erstellung eines Jobscriptes

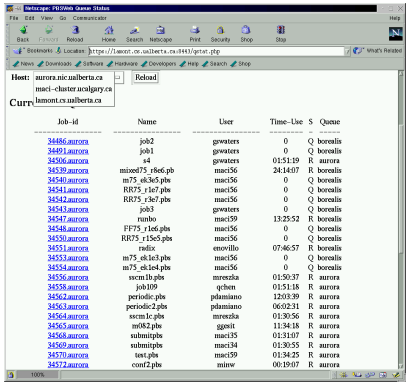


Abbildung: Überwachung der Queues

Quellen I

[IEEE] IEEE Standards Association
IEEE Std 1003.2d-1994, 2004 http://standards.ieee.org/reading/ieee/std_public/description/posix/1003.2d-1994_desc.html

[NARC] NASA Ames Research Center
Webseite, 2009
http://www.nasa.gov/centers/ames/home/index.html

[ALTAIR] Altair Grid Technology, LLC
Webseite, 2009 http://www.altair.com

[CRI] Cluster Resources Inc.
Webseite, 2009 http://www.clusterresources.com

[TORQUE] Cluster Resources Inc.
TORQUE Resource Manager, 2009 http://www.clusterresources.com/products/torque-resource-manager.php

Quellen II

[XPBS] DOE Science Grid
xpbs - Graphical Interface to PBS, 2001
http://www.doesciencegrid.org/public/pbs/xpbs.html

[XPBSMon] Matthias K. Gobbert
Running Jobs using torque and Maui on kali, Sep 2005
http://www.math.umbc.edu/~gobbert/kali/html/runcode.html

[PBSWeb] University of Alberta
Introduction to PBSWeb, Jul 2003
http://www.cs.ualberta.ca/~pinchak/PBSWeb/

Job Management Systems

Simple Linux Utility for Resource Management (SLURM)

Überblick I

- Entwicklung begann 2002 am Lawrence Livermore National Laboratory [LLNL] in Zusammenarbeit mit Linux NetworX
- Produktiver Einsatz am LLNL seit dem 2. Quartal 2003 auf einem Linux Cluster mit Quadrics Switch
- Der Vertrieb begann Anfang 2004 durch Linux NetworX
- Grund für die Entwicklung von SLURM waren - laut LLNC - die Limitierungen der alternativen Batchsysteme:
  - **Quadrics RMS** - Verwendung nur in Verbindung mit einem Quadrics Netzwerk möglich
  - **Portable Batch System (PBS)** - Portier- aber nicht skalierbar
  - **IBM LoadLeveler** - Weder portier- noch skalierbar
  - **Load Sharing Facility** - Portier- und skalierbar, allerdings entstehen hohe Kosten für große Cluster

Überblick II

- Key-Features:
  - Hohe Fehlertoleranz
  - Portierbar
  - Skaliert bis zu 65.536 Knoten und mehr als 100.000 CPU's
  - Frei verfügbarer Source Code (GPL)
  - Schnittstelle für Plugins

Architektur I

- Zwei Daemons:
  - `slurmctld` - Controller (optional existiert zusätzlich ein Backup)
  - `slurmd` - Computenode Daemon

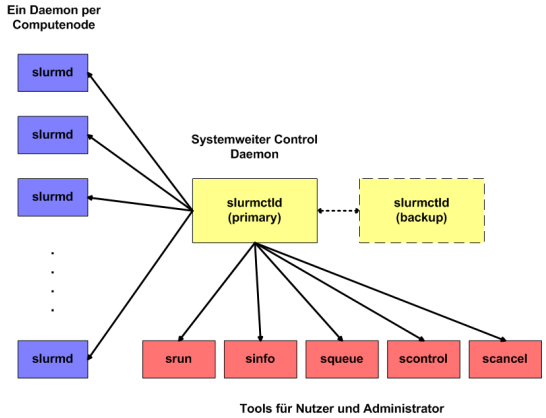


Abbildung: Architektur von SLURM

Architektur II

slurmctld:

- Initialisiert, steuert und protokolliert alle Aktivitäten des Resource Managers auf dem Cluster
- Bestandteile:
  - **Job Manager** - Verwaltet die Queue mit wartenden Jobs
  - **Node Manager** - Hält die Statusinformationen der Nodes
  - **Partition Manager** - Verantwortlich für die Allokierung der Nodes

Architektur III

slurmd:

- Führt von `slurmctld` und `srun` vorgegebene Instruktionen auf den Computenodes aus
- Aufgaben:
  - Remote Execution
  - Rückgabe von Node- oder Jobstatus auf Anfrage
  - Stream Copy (`stdin`, `stdout`, `stderr`)
  - Job Control (über Signal von `slurmctld`)

Kommandos I

`sinfo` - Gibt Informationen über Partitionen und Nodes zurück

```
ageorgi@quad01:~$ sinfo

PARTITION AVAIL  TIMELIMIT NODES  STATE NODELIST
compute*     up   infinite     1  down* quad04
compute*     up   infinite     2   idle quad0[2-3]
compute*     up   infinite     2  alloc quad0[5-6]
debug        up      30:00     1   idle quad01
```

`squeue` - Auflistung von Informationen über laufende und wartende Jobs

```
ageorgi@quad01:~$ squeue

JOBID PARTITION     NAME     USER  ST       TIME  NODES NODELIST(REASON)
16000 compute     spring    alice   R    6:46:04      2 quad0[2-3]
13601 compute     summer    brian   R    4:03:53      2 quad0[5-6]
70573 debug       winter    david   R       6:40      1 quad01
70574 debug         fall    edith  PD       0:00      1
```

`srun` - Starten interaktiver Jobs (impliziert Ressourcenallokierung)

```
ageorgi@quad01:~$ srun --ntasks=4 --partition=compute --label hostname

1: quad02
3: quad03
0: quad02
2: quad03
```


Kommandos II

`sbatch` - Abschicken eines Batch Jobs

```
ageorgi@quad01:~$ cat batch_example

#!/bin/bash
#SBATCH --time=1

hostname
srun --label hostname
srun --ntasks=4 --label hostname

ageorgi@quad01:~$ sbatch --nodes=2 -o batch_example.stdout batch_example

sbatch: Submitted batch job 5478

ageorgi@quad01:~$ cat batch_example.stdout

quad02
0: quad02
1: quad03
1: quad02
2: quad03
3: quad03
0: quad02
```

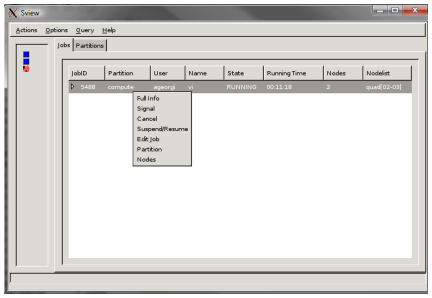

Kommandos III

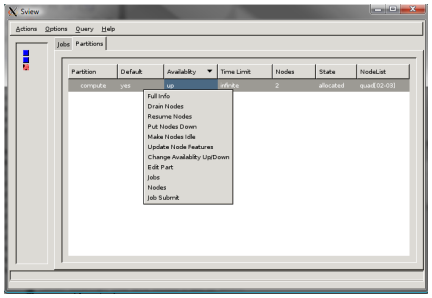
`salloc` - Allokierung von Ressourcen um darin interaktiv zu arbeiten

```
ageorgi@quad01:~$ srun hostname

quad02

ageorgi@quad01:~$ salloc --nodes=2 bash

salloc: Granted job allocation 5483

ageorgi@quad01:~$ srun hostname

quad02
quad03

ageorgi@quad01:~$ exit

exit
salloc: Relinquishing job allocation 5483
```

sview - GTK GUI für Nutzer und Administratoren

- Überwachung und Steuerung laufender oder wartender Jobs
- Statusabfrage oder -änderung der verfügbaren Partitionen





Quellen I

[LLNL] Lawrence Livermore National Laboratory
Webseite, 2009 `https://www.llnl.gov/`

[SDOC] Lawrence Livermore National Laboratory
Documentation, 2009
`https://computing.llnl.gov/linux/slurm/documentation.html`

[SADM] Lawrence Livermore National Laboratory
Quick Start Administrator Guide, 2009 `https://computing.llnl.gov/linux/slurm/quickstart_admin.html`

Inhalt

Job-Scheduler

OpenPBS Scheduler

OpenPBS Scheduler

- In der Open Source Version ist lediglich ein modifizierter FIFO Scheduler implementiert
- Jobs können dabei zusätzlich zur FIFO-Reihenfolge, nach folgenden Kriterien sortiert werden:
  - Verarbeitung aller Jobs in einer Queue, bevor die nächste Queue verarbeitet wird
  - Sortierung der Queues nach Prioritäten
  - Sortierung der Jobs innerhalb einer Queue nach der angeforderten CPU-Zeit (`cput`)
  - Priorisierung von Jobs die sich länger als einen Tag in der Warteschlange befinden
  - Jobs in einer Queue deren Name mit "`ded`" beginnt werden nur gestartet wenn sich das System in der genannten Zeit im dedizierten Zustand befindet
- Schnittstellen zur Definition eigener Scheduling Policies sind vorhanden
- Ein Austausch des Schedulers ist möglich

Job-Scheduler

SLURM Scheduler

Consumable Resources

- Per default werden Nodes von einem Job exklusiv allokiert, auch wenn dieser nicht alle Ressourcen nutzt
- Ab SLURM Version 1.2 können folgende Ressourcen als "consumable" deklariert werden:
  - CPU (CR_CPU)
  - Socket (CR_Socket)
  - Core (CR_Core)
  - Memory (CR_Memory)
  - Socket and Memory (CR_Socket_Memory)
  - Core and Memory (CR_Core_Memory)
  - CPU and Memory (CR_CPU_Memory)

Preemption

- Preemption wird ab SLURM Version 1.3 unterstützt
- Ermöglicht Priorisierung von Jobs oder Queues
- Jobs mit geringer Priorität können auch noch zur Laufzeit suspendiert werden, um Jobs mit höherer Priorität abzuarbeiten

Gang Scheduling

Gang Scheduling:

- Gruppen von verwandten Threads werden als eine Einheit behandelt
- Alle Mitglieder laufen gleichzeitig auf verschiedenen Cores und beginnen bzw. beenden ihre Zeitabschnitte gemeinsam
- Ermöglicht effiziente Kommunikation zwischen Threads eines Prozesses

Timesliced Gang Scheduling:

- Verfügbar ab SLURM Version 1.3
- Allokieren zwei oder mehr Jobs die gleichen Ressourcen werden diese alternierend suspendiert, so dass je ein Job für einen festgelegten Zeitraum dedizierten Zugriff auf diese Ressourcen hat

Weitere konfigurierbare Scheduling Policies I

Ressourcenlimitierung:

- Verfügbar ab SLURM Version 2.0
- Festlegung der maximal allokierbaren Ressourcen
- Mögliche Limitierungen (Auszug):
  - MaxNodesPerJob
  - MaxSubmitJobs
  - MaxWallDurationPerJob

Topologie:

- Verfügbar ab SLURM Version 2.0
- Optimierung der Ausführungszeit laufender Jobs durch Berücksichtigung der Netzwerktopologie
- Aktuell stehen zwei Topologien zur Verfügung - 3D-Torus und hierarchische Verbindungsnetzwerke

Weitere konfigurierbare Scheduling Policies II

Ressourcenreservierung:

- Der Administrator kann Ressourcen für eine definierte Zeitspanne und einen bestimmten Nutzer bzw. eine bestimmte Nutzergruppe reservieren
- Die Reservierungen können mit Hilfe von `scontrol` erstellt, aktualisiert und entfernt werden

Job-Scheduler

Maui Cluster Scheduler

Maui Cluster Scheduler

- Maui [MAUI] ist ein Open Source Job-Scheduler, welcher vorrangig zur Ergänzung vorhandener Stapelverarbeitungssysteme eingesetzt wird
- Maui kann u.a. in folgende Job Management Systeme integriert werden:
  - OpenPBS, PBSPro, TORQUE
  - SLURM
  - LSF
  - IBM LoadLeveler
  - SUN Grid Engine

Maui Scheduling-Zyklus

Der Scheduling-Zyklus besteht in jeder Iteration aus acht Schritten:

1. Abfrage aktueller Informationen über angeschlossene Nodes beim JMS
2. Statistikerhebung über laufende und Generierung von Berichten für abgeschlossene Jobs
3. Starten von Jobs die Reservierungen aufweisen können, die ihnen sofortigen Zugriff auf die benötigten Ressourcen erlauben
4. Auswahl der Jobs welche die Scheduling-Kriterien erfüllen
5. Priorisierung der Jobs in Abhängigkeit ihrer Ressourcenanforderungen und der Verfügbarkeit dieser Ressourcen
6. Starten der Jobs in der Reihenfolge ihrer Prioritäten
7. Ermittlung der aktuell vorhanden Ressourcen-Fenster, die sogenannten Backfill Windows und Füllung dieser mit vorhanden Jobs in der Reihenfolge ihrer Priorität
8. Erneute Ermittlung der verbleibenden Ressourcen-Fenster und Füllung mit vorhandenen Jobs, ohne Beachtung der Prioritäten

Maui Job-Priorisierung

- Prioritäten sind bei Maui keine Direktwerte die durch Nutzer oder Administrator gesetzt werden, sondern setzen sich aus sechs Komponenten zusammen:
  - `Requested Resources` - Priorisierung in Abhängigkeit der Ressourcenanforderung
  - `FairShare` - Differenzierte Priorisierung von Jobs unter Berücksichtigung der Benutzung von Ressourcen in der Vergangenheit
  - `Target` - Bevorzugung von Jobs die sich bereits eine relativ lange Zeit in der Queue befinden
  - `Bypass` - Bevorzugung von Jobs die bereits häufig durch das Starten anderer Jobs in der Backfill-Phase zurückgesetzt wurden
  - `Service` - Gestattet die Anhebung der Prioritäten bestimmter Jobs, die sonst stets am Ende der Prioritätenliste stünden
  - `DirectSpec` - Direktwertangabe durch den Benutzer oder Administrator

Maui Job-Priorisierung

Formel zur Berechnung der Job-Priorität

$$
\begin{aligned}
\text{priority} = \;& \text{resourceweight} * \text{resourcefactor} + \\
& \text{fairshareweight} * \text{fairsharefactor} + \\
& \text{targetweight} * \text{targetfactor} + \\
& \text{bypassweight} * \text{bypassfactor} + \\
& \text{serviceweight} * \text{servicefactor} + \\
& \text{directspecweight} * \text{directspecfactor}
\end{aligned}
$$

Die Gewichtungen der Faktoren können in der Maui-Konfiguration definiert werden. Weitere Informationen zu dem FairShare- und Backfill-Konzept finden sich unter [MSCHED].

Quellen I

- [OPBS] A. Bayucan, R. L. Henderson, L. T. Jasinskyj, C. Lesiak, B. Mann, T. Proett, D. Tweten
  Portable Batch System - Administrator Guide, Aug 1998 `http://www.nas.nasa.gov/Software/PBS/pbsdocs/admin_guide.ps`
- [SLDOC] Lawrence Livermore National Laboratory
  Documentation, 2009
  `https://computing.llnl.gov/linux/slurm/documentation.html`
- [MAUI] Cluster Resources
  Maui Cluster Scheduler, 2009 `http://www.clusterresources.com/products/maui-cluster-scheduler.php/`
- [MSCHED] David Jackson, Quinn Snell, Mark Clement
  Core Algorithms of the Maui Scheduler
  `www.cs.huji.ac.il/~feit/parsched/jsspp01/p-01-6.ps.gz`