Julian Göttmann und Markus Vater

Praktikum "Aufbau eines Clusters"

Blockveranstaltung am DKRZ im WiSe 17/18[1]

- Praktikum "Aufbau eines Clusters"
  - Einleitung
  - Tag 01 (26. Februar) `VirtualBox` `Grundinstallation` `Netzwerk`
    - Arbeitsschritte
    - Bugs und Probleme
    - Ergebnisse
  - Tag 02 (27. Februar) `Dateisystem` `IP-Konfiguration` `Nutzerverwaltung`
    - Arbeitsschritte
    - Bugs und Probleme
    - Ergebnisse
  - Tag 03 (28. Februar) `Nutzerverwaltung` `Konfigurationsmanagement`
    - Arbeitsschritte
    - Bugs und Probleme
    - Ergebnisse
  - Tag 04 (01. März) `Konfigurationsmanagement` `Monitoring`
    - Arbeitsschritte
    - Bugs und Probleme
    - Ergebnisse
  - Wochenende
    - Arbeitsschritte
    - Bugs und Probleme
    - Ergebnisse
  - Tag 05 (05. März) `Konfigurationsmanagement` `Batch Scheduling`
    - Arbeitsschritte
    - Bugs und Probleme
    - Ergebnisse
  - Tag 06 (06. März) `Konfigurationsmanagement` `Batch Scheduling`
    - Arbeitsschritte
    - Bugs und Probleme
    - Ergebnisse
  - Tag 07 (07. März) `Monitoring` `Softwareverwaltung`
    - Arbeitsschritte
    - Bugs und Probleme
    - Ergebnisse
  - Tag 08 (08. März) `Batch Scheduling`
    - Arbeitsschritte
    - Bugs und Probleme
    - Ergebnisse
  - Nach dem Praktikum
    - Arbeitsschritte
    - Bugs und Probleme
    - Ergebnisse
  - Endergebnis

Am Ende des Praktikums

Einleitung

Das Ziel der Veranstaltung bestand darin, ein kleines Cluster aus virtuellen Maschinen aufzusetzten um daran zentrale Probleme und Lösungen kennenzulernen.

Als Virtualisierungslösung haben wir **VirtualBox**[2] (OpenSource) von Oracle verwendet, als Betriebssystem für sämtliche Knoten kommt **CentOS 7**[3] ("DVD-Image") zum Einsatz.

Die genauen Bezeichnungen (Username, Hostname usw.) dürften flexibel wählbar sein, die geschilderten Schritte beschreiben die tatsächlich ausgeführten Tätigkeiten. Aufgrund von nicht direkt lösbaren Konfigurationsproblemen haben wir am Ende der ersten Woche die Knoten erneut von Grund auf neu aufgesetzt, hierbei wurden leichte Änderungen vorgenommen, welche in den Abschnitten nach Tag 04 Erwähnung finden.

Beim Hostsystem handelte es sich um ein Thinkpad T440p mit Haswell Core i5-4210M (2 Kerne + Hyperthreading, Boost bis 3,2 GHz) mit 16 GB Arbeitsspeicher.

Tag 01 (26. Februar) `VirtualBox` `Grundinstallation` `Netzwerk`

Arbeitsschritte

- drei virtuelle Maschinen erstellt
  - Typ `Red Hat (64-bit)`
  - 1 CPU-Kern
  - 2048 MB RAM
  - 10 GB HDD im `.vhd`-Format, dynamisch alloziert
- neues NAT-Netzwerk erstellt
  - `File` > `Preferences` > `Network` > `Adds new NAT Network`
    - Name: clusterNetwork
    - DHCP aktiviert
    - Port Forwarding: node00:22 auf :8022
  - für jede der VMs: `Settings` > `Network` > `Adapter 1`
    - Attached to = "NAT Network"
    - Name = "clusterNetwork"
    - `Advanced` > Promiscous Mode = "allow all" (wahrscheinlich egal)
- CentOS installiert
  - Konfiguration = "Rechenknoten"
  - "nfs-utils" vorinstalliert
  - Hostname = "cent01" bzw. "cent02" bzw. "cent03"
  - User = "userc01" bzw. "userc02" bzw. "userc03"

Bugs und Probleme

Herumprobiert haben wir längere Zeit mit den verfügbaren Netzwerk-Typen, manche ermöglichen ausschließlich Kommunikation `Guest` ↔ `Guest`, andere wiederum nur `Guest` ↔ `Host` usw.

Ergebnisse

Am Ende des Tages liefen die VMs mit der Basisinstallation von **CentOS** und konnten sowohl untereinander als auch auf das Internet zugreifen. Während der erste Knoten als Login-Knoten dienen soll und später auch diverse Server-Dienste im Cluster übernimmt, dienen die anderen beiden als reine Rechenknoten. Zugriff auf die Maschinen vom Host-System ist durch die eingerichtete Portfreigabe ausschließlich zum Login-Knoten möglich, somit ist analog zu existierenden Clustern wie bspw. im DKRZ ein direkter Zugriff auf die Rechenknoten nicht möglich. Letzteres wäre wohl mit der Adaptereinstellung "bridged" nicht möglich, obwohl diese vermutlich auch die anderen Eigenschaften geboten hätte.

Tag 02 (27. Februar) `Dateisystem` `IP-Konfiguration` `Nutzerverwaltung`

Arbeitsschritte

- **Network File System** (NFS) für gemeinsame Nutzerverzeichnisse eingerichtet

  Nach der Einrichtung sollen die Verzeichnisse /home und /var/nfsshare des Login-Knoten von den anderen beiden Knoten gemount werden können.

  - genutztes Tutorial: NFS Server and Client Installation on CentOS 7 unter howtoforge.com
    - abweichend `/home` auf `/home` gemountet, sodass für einen sich erstmalig einloggender Nutzer das Nutzerverzeichnis direkt auf dem NFS-Server gespeichert wird
  - wenn man auf allen Knoten identische Nutzernamen (lokale Benutzer) verwenden wollen würden, müsste vermutlich zumindest die User- und Gruppen-ID der Benutzer angepasst werden, um Zugriff auf ein gemeinsames Nutzerverzeichnis zu erhalten.

- statische IP-Adressen vergeben
  - neue Adressen: `10.0.2.10` bzw. `.11` bzw. `.12`
  - genutztes Tutorial: How to configure a static IP address on CentOS 7 / RHEL 7 unter cyberciti.biz
  - Portweiterleitung entsprechend angepasst
    `/etc/sysconfig/network-scripts/ifcfg-enp0s3` für den ersten Knoten (unvollständig!):

```
DEVICE=enp0s3
BOOTPROTO=none
ONBOOT=yes
PREFIX=24
IPADDR=10.0.2.10
NAME="enp0s3"
```

Anschließend muss die Änderung mit `systemctl restart network.service` angewendet werden.

- vergeblich versucht, **FreeIPA** zu installieren
  - Benutzerverwaltung auf Basis von **LDAP** und **Kerberos** ermöglicht identische Benutzer auf verschiedenen Rechnern und erleichtert deren Verwaltung
    - bei uns an der DNS-Konfiguration und unzureichender Dokumentation gescheitert, eine andere Gruppe erreichte schickes Endergebnis (Verwaltungsinfterface im Browser) mithilfe

kleinerer Workarounds

Bugs und Probleme

Vereinzelt verloren Knoten ihre IPv4-Adresse [4], daher haben wir die IP-Verwaltung von DHCP auf statische Adressen umgestellt. Dabei haben wir allerdings aufgrund fehlender DNS-/Gateway-Konfiguration den Internetzufgriff verloren. Dies haben wir durch Hinzufügen eines weiteres Netzwerkadapters (Typ "NAT") für jede VM unelegant behoben. Die korrekte Konfiguration für statische IP-Adressen findet sich weiter unten beim Abschnitt Wochenende.

Ergebnisse

Das gemeinsames Dateisystem funktionierte tadellos, Nutzerverzeichnisse wurden nun zentral gespeichert. Zusätzlich existierte mit `/var/nfsshare` ein frei nutzbares gemeinsames Verzeichnis, indem bspw. global zu installierende Software-Pakete für alle Knoten zugreifbar gespeichert werden können. **FreeIPA** ließ sich leider noch nicht installieren.

Tag 03 (28. Februar) `Nutzerverwaltung` `Konfigurationsmanagement`

Arbeitsschritte

- **FreeIPA** installiert für clusterweite Authentifizierung mit einem einzelnen Nutzerkonto)
  - genutzte Anleitung für den Server auf dem Login-Knoten: How To Set Up Centralized Linux Authentication with FreeIPA on CentOS 7 unter digitalocean.com
  - genutzte Anleitung für die Clients auf allen Knoten: How To Configure a FreeIPA Client on CentOS 7 unter digitalocean.com
  - DNS-Konfiguration bei der Installation weitestgehend übersprungen, Namensauflösung nur im lokalen Netz durch `/etc/hosts`:

```
127.0.0.1   localhost
10.0.2.10   ipa.node00.local node00
10.0.2.11   ipa-client.node01.local node01
10.0.2.12   ipa-client.node02.local node02
```

  - Probleme bei Kommunikation der Knoten untereinander, die **FreeIPA**-Clients auf den Rechenknoten tauchen nach der Installation kurz in der "Hosts"-Übersicht des Webinterfaces auf, sind anschließend aber wieder verschwunden. Das Einloggen mit den globalen Zugangsdaten ist zu keinem Zeitpunkt auf den Rechenknoten möglich, auf dem Login-Knoten dauert es lange. Wir vermuten DNS-Probleme, welche sich aus einem leicht vom oben beschriebenen abweichenden `/etc/hosts`-File und den zusätzlichen NAT-Adaptern (Workaround von Tag 02) ergeben.
- Auswahl eines geeigneten Programms zum Konfigurationsmanagement. Solche Werkzeuge bieten eine zentralisierte Schnittstelle zur Verteilung von Konfigurationen auf beliebig vielen Knoten im Cluster.
  - **Docker** frühzeitig aussortiert (unserer Meinung nach für Cluster ungeeignet)
    - **Chef** bietet ein umfangreiches Webinterface und neben der Konfigurationsverwaltung viele zusätzliche Funktionen an (übermäßiger Hardware-Hunger, zeitlich stark beschränkte Lizenz)
    - **Puppet**, **Ansible** wurden von anderen Gruppen behandelt

        - **CFEngine** ist eine Puppet-Variante
    - **SaltStack** sieht gut aus, aber an diesem Tag aus Zeitgründen nicht mehr aufgesetzt

Bugs und Probleme

- Login per **FreeIPA** funktioniert bislang nur auf dem Login-Knoten, die anderen Knoten erlauben nur Login über die lokalen Nutzerkonten

Ergebnisse

Die an diesem Tag erstmals abgeschlossene **FreeIPA**-Installation war leider nicht vollständig zum Laufen zu bringen. Immerhin war nach der Fehlersuche noch ausreichend Zeit, die verfügbaren Konfigurationsmanagementwerkzeuge zu vergleichen und bereits eine Entscheidung für den folgenden Tag zu treffen.

Tag 04 (01. März) `Konfigurationsmanagement` `Monitoring`

Arbeitsschritte

- weiterhin vergeblich versucht, **FreeIPA** zu fixen (siehe Bug von Tag 03)
- **SaltStack** auf seperatem Cluster installiert und getestet
    - extrem einfache Konfiguration
    - genutzte Anleitung: SaltStack on CentOS unter centoshowtos.org
- **Munin** auf seperatem Cluster installiert und getestet
    - keine Verbindung zu Rechenknoten möglich (fehlerhafte Konfiguration)

Bugs und Probleme

Identisch zu Tag 03, zusätzlich noch kleinere (vermutlich lösbare) Schwierigkeiten mit den neuen Tools.

Ergebnisse

Am vierten Tag haben wir teilweise erfolgreich parallel zur weiteren Fehlersuche Konfigurationsmanagement- und Monitoringwerkzeuge installiert, getestet und der Gruppe vorgestellt.

Wochenende

Am Wochenende haben wir uns aufgrund der erstmal nicht zu lösenden Probleme mit FreeIPA und anderer knirschenden Kleinigkeiten dazu entschlossen, die drei Knoten von Grund auf neu zu installieren und dabei die Konfiguration geringfügig anzupassen.

Arbeitsschritte

- drei neue Knoten erstellt, alles außer Konfigurationsmanagement- und Monitoringwerkzeuge neu installiert

Bugs und Probleme

- Home-Verzeichnis wird nicht automatisch beim ersten Login erstellt `beherrschbar (nervig)`

Ergebnisse

Unter anderem wurde durch die korrekte Konfiguration der statischen IP-Adressen der NAT-Adapter-Workaround überflüssig (neue Konfiguration weiter unten). Durch die in den letzten Tagen gesammelten Erfahrungen ging das Neuaufsetzen zügiger vonstatten als gedacht und ermöglichte es uns in der zweiten Woche mit einem funktionierenden und verschlankten Cluster weiterzuarbeiten. Zu der lauffähigen Software gehörte das Betriebssystem selber mitsamt Netzwerkkonfiguration, das Dateisystem sowie die Nutzerverwaltung.

`/etc/sysconfig/network-scripts/ifcfg-enp0s3` für den ersten Knoten (fehlerfrei):

```
DEVICE=enp0s3
BOOTPROTO=none
ONBOOT=yes
PREFIX=24
IPADDR=10.0.2.10
GATEWAY=10.0.2.1
DEFROUTE="yes"
NAME="enp0s3"
DNS1=10.0.2.1
DNS2=8.8.8.8
DNS3=8.8.4.4
IPV6INIT=no
```

Tag 05 (05. März) `Konfigurationsmanagement` `Batch Scheduling`

Für den Betrieb eines Clusters werden Dienste zur möglichst optimierten Ausführung von Aufgaben benötigt. Dafür unterschiedlich umfangreich und komplexe Systeme wie **OAR** oder **SLURM**.

OAR bietet neben dem Batch Scheduling auch die Möglichkeit eine Statisik und Visualisierung der Jobs anzufertigen. Wegen der Vielseitigkeit haben wir uns für die Einrichtung von **OAR** entschieden.

Arbeitsschritte

- **SaltStack** aufgesetzt:
  - genutzte Anleitung: SaltStack on CentOS unter centoshowtos.org
    - auf allen Knoten:

      `/etc/hosts`:

      ```
      127.0.0.1   localhost
      10.0.2.10   ipa.node00.local node00 salt
      10.0.2.11   ipa-client.node01.local node01
      10.0.2.12   ipa-client.node02.local node02
      ```

      Durch das Eintragen des "salt"-Hosts kennen alle Clients automatisch die Server-Adresse.
    - auf dem Server (Login-Knoten):

```
yum install epel-release
yum install salt-master salt-minion
systemctl enable salt-master
systemctl enable salt-minion
systemctl start salt-master
systemctl start salt-minion
firewall-cmd --permanent --add-port=4505-4506/tcp
firewall-cmd --reload
```

- auf den Clients (alle Knoten):

```
yum install epel-release
yum install salt-minion
systemctl enable salt-minion
systemctl start salt-minion
```

- auf dem Server können anschließend mit `salt-key -L` die verfügbaren Clients angezeigt und mit `salt-key -A` alle neuen Knoten akzeptiert werden.

- **Munin** aus Zeitgründen verschoben
  - stellt keine Abhängigkeit für andere Installationen dar
- **OAR**[5] versucht zu installieren
  - genutzte Anleitung: OAR 2.5 documentation: Installation unter http://oar.imag.fr
  - unklare Kapitel-Strukturierung der Anleitung: eine Installation ist aus den Paketquellen ("Installation from the packages") *oder* zum Selbstkompilieren aus den Quellen ("Installation from the tarball") möglich, die anschließende Konfiguration wiederum ist immer erforderlich

  Wir haben uns für die Installation aus den Paketquellen entschieden um zusätzliche Probleme zu vermeiden. Dazu muss noch ein eigenes Repository von der Downloadseite des Projektes eingebunden werden.

  > Achtung:
  >
  > Sowohl der zum Signieren der Pakete verwendete Schlüssel als auch die Installationspakete (ebenso die Quellcodearchive) können ausschließlich über eine ungesicherte HTTP-Verbindung bezogen werden. Dies stellt zumindest für Produktivsysteme ein ernstzunehmendes Sicherheitsrisiko dar, weshalb von der Nutzung von OAR zurzeit nur abgeraten werden kann.

  - `oar-node` auf beiden Rechenknoten installiert und per `systemctl enable --now oar-node` gestartet
    - `oar-server` mit der `oar-server-pgsql`-Schnittstelle auf dem Loginknoten installiert
      - `/etc/oar/oar.conf` anpassen
    - PostgreSQL-Datenbank muss separat installiert und initialisiert werden
    - … (abgebrochen)

Bugs und Probleme

Die Installation von **SaltStack** ist fast trivial, dementgegen ist die Installation von **OAR** aufgrund der schlecht strukturierten und mageren Anleitung komplizierter (keine alternativen Anleitungen gefunden!). An diesem Tag steckten wir in der manuellen Initialisierung der Datenbank fest. Überdies wird nicht angegeben, ob Portfreigaben für Server oder Clients erforderlich sind, ob die Clients auf die Datenbank zugreifen können müssen usw.

Ergebnisse

SaltStack läuft, **OAR** mussten wir aufgrund der Schwierigkeiten auf den kommenden Tag verschieben.

Tag 06 (06. März) `Konfigurationsmanagement` `Batch Scheduling`

Softwareverwaltungsprogramme wie **Spack** oder **EasyBuild** erhalten Dateien mit Informationen über Quelltexte, Abhängigkeiten und zu verwendende Compiler, um eine Software zu kompilieren.

Da **Spack** diese Konfigurationsdateien im Gegensatz zu **EasyBuild** direkt mitliefert, haben wir uns letztendlich für **Spack** entschieden.

Arbeitsschritte

- weiterhin vergeblich versucht **OAR** zu installieren (Reset auf alten Snapshot)
  - Wechsel von der empfohlenen PostgreSQL-Variante zur MySQL-/MariaDB-Implementation, da einer von uns bereits Erfahrungen mit dieser Datenbank besaß
  - `oar-server` mit der `oar-server-mysql`-Schnittstelle auf dem Loginknoten installiert
  - `mariadb` installiert und gestartet
  - Datenbank für **OAR** hinzugefügt (siehe Anleitung)
  - `oar-user oar-user-mysql` für Frontend-Tools installiert
- Installation von **Spack** zur Softwareverwaltung:

```
yum install git gcc gcc-gfortran patch gcc-c++ bzip2
git clone https://github.com/spack/spack.git
. spack/share/spack/setup-env.sh
```

Bugs und Probleme

Die für **OAR** verfügbare Dokumentation nach der Grundinstallation ist zu beschränkt. Auch nach Installation der Frontend-Tools haben wir keine Möglichkeit gefunden um uns die erreichbaren Knoten anzeigen zu lassen. Es lassen sich Knoten in **OAR** hinzufügen, welche beliebige sinnlose Hardwareausstattungen besitzen und beim Start eines Jobs nicht gefunden werden können.

Ergebnisse

Triviale Installation von **Spack** erfolgreich, **OAR** läuft immer noch nicht.

Tag 07 (07. März) `Monitoring` `Softwareverwaltung`

Beim High-Performance-Linpack handelt es sich um Software zum Ermitteln der Performance (Benchmarking), welche speziell auf Großrechner mit vielen Kernen ausgerichtet ist. Sie soll uns als Testprogramm dienen, welches die Kommunikation zwischen den Knoten über OpenMPI abwickelt.

Arbeitsschritte

- **Munin** erfolgreich eingerichtet
    - auf dem Server (Login-Knoten):

```
yum install epel-release
yum install httpd munin
systemctl enable httpd
systemctl start httpd
systemctl status httpd
```

`/etc/munin/munin.conf` ergänzt um:

```
dbdir   /var/lib/munin
htmldir /var/www/html/munin
logdir  /var/log/munin
rundir  /var/run/munin

[ipa.node00.local]
    address 10.0.2.10
    use_node_name yes

[ipa-client.node01.local]
    address 10.0.2.11
    use_node_name yes

[ipa-client.node02.local]
    address 10.0.2.12
    use_node_name yes

```

`/etc/httpd/conf.d/munin.conf` ergänzt um:

```
AuthName "admin"
```

Anschließend ausgeführt:

```
htpasswd /etc/munin/munin-htpasswd admin
```

    - auf allen Knoten:

```
yum install epel-release
yum install munin-node
```

`/etc/munin/munin-node.conf` ergänzt um:

```
allow ^10.0.2.10$
```

Anschließend ausgeführt:

```
firewall-cmd --permanent --add-port=4949/tcp
firewall-cmd --reload
systemctl enable munin-node
systemctl start munin-node
```

- **hpl** kompiliert

```
. spack/share/spack/setup-env.sh
spack install hpl
```

Der Kompiliervorgang benötigt relativ viel Zeit. Jedoch ist es möglich, diesen parallel auf einer Kopie des Clusters (zweiter Laptop) laufen zu lassen und das achivierte spack-Verzeichnis (enthält auch die erzeugten Binärdateien) später auf das Zielcluster zu verschieben. Nutzt man eines der NFS-Verzeichnisse, erhalten alle Knoten Zugriff auf die kompilierten Anwendungen.

Bugs und Probleme

keine

Ergebnisse

Das Monitoring mit **Munin** funktionierte gut, da allerdings standardmäßig Daten in 5-Minuten-Intervallen gesammelt werden, hat es einige Zeit gedauert, um anschauliche Daten für die Visualisierung zu sammeln.

Tag 08 (08. März) `Batch Scheduling`

Ziel dieses Tages war es, eine möglichst vollständige Installation eines Clusters für die Endpräsentation zu erreichen um dort **hpl** (High Performance Linpack) verteilt laufen zu lassen.

Arbeitsschritte

- **spack** samt kompilierter Pakete auf das Cluster transplantiert
- aus Zeitgründen OAR zurückgestellt, **SLURM** versucht zu installieren
  - genutzte Anleitungen: How to Install Slurm on CentOS 7 Cluster unter slothparadise.com sowie [] [Slurm on CentOS 7]Slurm on CentOS 7 unter bitsanddragons.wordpress.com

Auf allen Knoten User-Accounts erstellen und Pakete installieren:

```
export MUNGEUSER=991
groupadd -g $MUNGEUSER munge
useradd  -m -c "MUNGE Uid 'N' Gid Emporium" -d /var/lib/munge -u $MUNGEUSER -g
munge -s /sbin/nologin munge
export SLURMUSER=992
groupadd -g $SLURMUSER slurm
useradd  -m -c "SLURM workload manager" -d /var/lib/slurm -u $SLURMUSER -g
slurm  -s /bin/bash slurm

yum install epel-release munge munge-libs munge-devel rng-tools mariadb-server
mariadb-devel -y
```

Auf dem Login-Knoten den **munge**-Key erstellen und verteilen:

```
rngd -r /dev/urandom
/usr/sbin/create-munge-key -r
dd if=/dev/urandom bs=1 count=1024 > /etc/munge/munge.key
chown munge: /etc/munge/munge.key
chmod 400 /etc/munge/munge.key
scp /etc/munge/munge.key root@node01:/etc/munge
scp /etc/munge/munge.key root@node02:/etc/munge
```

Auf jedem Knoten die Berechtigungen anpassen und **munge** starten:

```
chown -R munge: /etc/munge/ /var/log/munge/
chmod 0700 /etc/munge/ /var/log/munge/
systemctl enable munge
systemctl start munge
```

Auf dem Login-Knoten **SLURM** herunterladen, bauen und in ein für alle Knoten zugängliches Verzeichnis verschieben:

```
wget https://download.schedmd.com/slurm/slurm-17.11.4.tar.bz2
yum install openssl openssl-devel pam-devel numactl numactl-devel hwloc hwloc-devel lua lua-devel readline-devel rrdtool-devel ncurses-devel man2html libibmad libibumad wget rpm-build cpanm* -y
rpmbuild -ta slurm-17.11.4.tar.bz2
mv /root/rpmbuild/RPMS/x86_64/* /var/nfsshare/pkg
```

Auf jedem Knoten **SLURM** installieren:

```
yum --nogpgcheck localinstall slurm*.rpm
```

... (abgebrochen, Fortsetzung im folgenden Kapitel)

Bugs und Probleme

Aus unbekannten Gründen ließen sich zumindest Teile von **SLURM** noch nicht starten.

Ergebnisse

SLURM lief noch nicht, allerdings ließ sich der **hpl**-Benchmark lokal ohne Probleme ausführen.

Nach dem Praktikum

Arbeitsschritte

- weitermachen mit der **SLURM**-Installation:

  Mithilfe des Konfigurationstools unter slurm.schedmd.com die **SLURM**-Konfiguration erstellen:

```
# slurm.conf file generated by configurator easy.html.
# Put this file on all nodes of your cluster.
# See the slurm.conf man page for more information.
#
ControlMachine=node00
ControlAddr=10.0.2.10
#
#MailProg=/bin/mail
MpiDefault=none
#MpiParams=ports=#-#
ProctrackType=proctrack/cgroup
ReturnToService=1
SlurmctldPidFile=/var/run/slurmctld.pid
#SlurmctldPort=6817
SlurmdPidFile=/var/run/slurmd.pid
#SlurmdPort=6818
SlurmdSpoolDir=/var/spool/slurmd
SlurmUser=slurm
#SlurmdUser=root
StateSaveLocation=/var/spool/slurmctld
SwitchType=switch/none
TaskPlugin=task/none
#
# TIMERS
#KillWait=30
#MinJobAge=300
#SlurmctldTimeout=120
#SlurmdTimeout=300
#
# SCHEDULING
FastSchedule=1
SchedulerType=sched/backfill
SelectType=select/linear
#SelectTypeParameters=
#
# LOGGING AND ACCOUNTING
AccountingStorageType=accounting_storage/none
ClusterName=cluster
#JobAcctGatherFrequency=30
JobAcctGatherType=jobacct_gather/none
#SlurmctldDebug=3
SlurmctldLogFile=/var/log/slurmctld.log
#SlurmdDebug=3
SlurmdLogFile=/var/log/slurmd.log
#
# COMPUTE NODES
NodeName=node01 NodeAddr=10.0.2.11 CPUs=1 State=UNKNOWN
NodeName=node02 NodeAddr=10.0.2.12 CPUs=1 State=UNKNOWN
PartitionName=debug Nodes=node0[1-2] Default=YES MaxTime=INFINITE State=UP
```

Auf dem Login-Knoten:

```
cd /etc/slurm
nano slurm.conf
[...] # obige Konfiguration eintragen
scp slurm.conf root@node01:/etc/slurm/slurm.conf
scp slurm.conf root@node02:/etc/slurm/slurm.conf

mkdir /var/spool/slurmctld
chown slurm: /var/spool/slurmctld
chmod 755 /var/spool/slurmctld
touch /var/log/slurmctld.log
chown slurm: /var/log/slurmctld.log
touch /var/log/slurm_jobacct.log /var/log/slurm_jobcomp.log
chown slurm: /var/log/slurm_jobacct.log /var/log/slurm_jobcomp.log
```

Auf den Rechenknoten:

```
mkdir /var/spool/slurmd
chown slurm: /var/spool/slurmd
chmod 755 /var/spool/slurmd
touch /var/log/slurmd.log
chown slurm: /var/log/slurmd.log
systemctl stop firewalld
systemctl disable firewalld
```

Auf dem Login-Knoten:

```
firewall-cmd --permanent --zone=public --add-port=6817/udp
firewall-cmd --permanent --zone=public --add-port=6817/tcp
firewall-cmd --permanent --zone=public --add-port=6818/udp
firewall-cmd --permanent --zone=public --add-port=6818/tcp
firewall-cmd --permanent --zone=public --add-port=7321/udp
firewall-cmd --permanent --zone=public --add-port=7321/tcp
firewall-cmd --reload
```

Auf jedem Knoten:

```
yum install ntp -y
chkconfig ntpd on
ntpdate pool.ntp.org
systemctl start ntpd
```

Auf den Rechenknoten:

```
systemctl enable slurmd.service
systemctl start slurmd.service
systemctl status slurmd.service
```

Auf dem Login-Knoten:

```
systemctl enable slurmctld.service
systemctl start slurmctld.service
systemctl status slurmctld.service
```

Bugs und Probleme

Während sich slurmctld auf dem Login-Knoten nach einem Neustart problemlos ausführen ließ, konnte slurmd auf den Rechenknoten nicht gestartet werden, die Ursache bleibt unklar, da u.a. keinerlei Fehlerursache angezeigt wird.

Vermutlich besteht ein Konflikt zwischen den Hostnamen und der Konfiguration, da slurmd "ipa-client" als Hostnamen zu erkennen scheint, der komplette Hostname allerdings "ipa-client.node01.local" lautet.

```
$> slurmd -C
NodeName=ipa-client CPUs=1 Boards=1 SocketsPerBoard=1 CoresPerSocket=1
ThreadsPerCore=1 RealMemory=1839
UpTime=0-00:00:36
```

Ergebnisse

SLURM läuft nach wie vor leider nicht, Ursache unklar.

Endergebnis

Am Ende des Praktikums

Letztendlich hatten wir ein Cluster aus drei Knoten zur Verfügung, an dem man diverse bekannte Verwaltungsprogramme größerer Systeme studieren und administrieren lernen kann. Zur regulären Nutzung fehlt leider der (noch) nicht laufende Batch-Scheduling-Manager, allerdings wäre der Nutzen für echte Rechentasks gering, da alle Knoten auf einem einzelnen Hostsystem laufen. Interessant an der Virtualisierungslösung ist die Möglichkeit mit Snapshots innerhalb von Sekunden die Konfiguration zu verändern und an verschiedenen Punkten andere Programme testen zu können. Weiterhin wäre es vermutlich möglich, ein auf begrenzten Ressourcen aufgesetztes Cluster auf eine oder mehrere Workstations mit deutlich mehr Kernen zu kopieren und dort mehr Ressourcen zuzuteilen.

---

1. https://wr.informatik.uni-hamburg.de/teaching/wintersemester_2017_2018/aufbau_eines_clusters↩

2. https://www.virtualbox.org↩

3. https://www.centos.org/↩

4. RHEL / Centos 7-based instances lose their default IPv4 gateway↩

5. OAR homepage↩