Freitag: Arbeiten auf dem Cluster

10 Arbeiten auf dem Cluster

Arbeiten auf dem Cluster

Arbeiten auf dem Cluster

- Nicht alle Prozesse können auf den Servern oder privaten Rechnern ausgeführt werden
- Parallele Verarbeitung von großen Daten auf vielen Rechnern
- Bedarf zusätzlicher Ressourcen
  - (Mehrere) Grafikkarten
  - Große Arbeitsspeicher
  - Mehrere Prozessoren

Cluster

- Cluster: Ansammlung von Rechner, die gemeinsam Rechenressourcen zur Verfügung stellen
- Das ICL hat ein Cluster aus 54 Maschinen
  - 45 Rechner für normale compute-Aufgaben (ohne GPU)
  - 9 GPU Rechner mit bis zu 8 GPUs
- Das Cluster ist die einzige Variante, auf die Instituts-CPUs zuzugreifen
- Wichtige Mailingliste: `http://lists.cl.uni-heidelberg.de:8080/mailman/listinfo/cluster-users`
- **Viele Infos im Wiki** `https://wiki.cl.uni-heidelberg.de/bin/view/Main/FaQ/Tutorials/SlurmTutorial`

Anmeldung

```
:~$ ssh cluster
```

- Zugriff muss von der Technik freigeschaltet werden
- Loggt sich auf Verwaltungsknoten ein $\rightarrow$ Hier keine Aufwändigen Tasks starten

Arbeiten auf dem Cluster

Cluster Ressourcenallocation

- Das Cluster wird von vielen Menschen gleichzeitig genutzt
- Effiziente Ressourcenallokation entscheidend
  - Welche Prozesse laufen wo?
  - Wie viel Speicher bekommt der Prozess?
  - Wer braucht Zugriff auf exlusive Ressourcen Grafikkarten)?
- ICL-Cluster verwendet Slurm zur Verwaltung

Slurm

- *Simple Linux Utility for Resource Management*
- Sog. Workload Manager, verwaltet Clusterressourcen
- Terminologie

Node	Ein einzelner Computer
Partition	Gruppierung von Computern (zu Verwaltungszwecken)
Job	Ansammlung von Ressourcen (CPUs, Speicher, ...), die einem Benutzer zur Ausführung seiner Programme freigegeben wurden
Job Step	Einzelner Prozess, der als Teil eines Jobs ausgeführt wird

- Dokumentation: `https://slurm.schedmd.com`

Slurm - Ablauf

1. Benutzer bittet Slurm um bestimmte Ressourcen aus einer Partition
2. Ressourcen werden dem Benutzer zugewiesen (man sagt *alloziert*)
   - Jede Partition hat eine Warteschlange (queue)
   - Wenn die Ressourcen belegt sind, müssen wir warten, bis sie frei sind
3. Benutzer kann einzelne Job-Steps starten
   - Steps können Ressourcen belegen, die uns alloziert wurden
   - Wir können mehrere Steps nebeneinander ausführen
4. Nach einem Zeitlimit, oder nach Ende aller Schritte, werden die Ressourcen freigegeben

Arbeiten auf dem Cluster

Slurm bedienen

- Slurm Commands liegen in `/opt/slurm/bin`
- Sollten in die Pfadvariable gelegt werden
- Einfachste Variante: Befehl in `~/.profile` schreiben
  - `~/.profile` wird bei Login ausgeführt

```
if [ -d "/opt/slurm/bin" ] ; then
    PATH="/opt/slurm/bin:$PATH"
fi
```

Slurm bedienen - Partition wählen

- Zu Beginn müssen wir die passende Partition wählen
- Ergibt sich aus unseren Anforderung
- Unterschiedliche Partition haben verschiedene Nodes und Timelimits

Slurm bedienen - Partition wählen

Name	Zeitlimit	Zweck
main	3 Tage	30 Ältere CPU-Nodes
compute	3 Tage	15 Moderne CPU-Nodes
gpushort	8 Stunden	GPU-Nodes für kurze Tests
gpulong	3 Tage	GPU-Nodes
students	3 Tage	GPU für Studierende

Slurm bedienen - Infos einholen

- Bevor wir anfangen, sollten wir uns den Zustand des Clusters anschauen

  sinfo Zeigt Infos über Cluster-Knoten an
  - Welche Nodes gibt es in welchen Partitionen?
  - Sind Nodes nicht verfügbar (down)?
  - Welche Nodes werden benutzt?

  squeue Zeigt Infos über laufende und wartende Jobs an
  - Zeigt Job-Status in Spalte `ST` an
  - Zentrale Statuscodes: `R` für laufende Jobs, `PD` (pending) für Jobs, die auf Ressourcen warten
  - Sind viele Jobs für eine Partition pending, werden wir potenziell lange warten müssen

Interaktiver Modus

- Wir können auf dem Cluster (fast) wie auf einem lokalen Computer arbeiten
- Das nennen wir "Interaktiven Modus"
- Das besteht aus zwei Schritten:
  1. `salloc` alloziert Ressourcen
  2. `srun` führt Kommandos aus (quasi wie Bash)
- Geeignet für Konfiguration und Tests

Interaktiver Modus - `salloc`

- `salloc` benötigt eine Spezifikation unserer Anforderung an das Cluster
- `-p <partition>` wählt Partition aus
- `--time hh:mm:ss` legt maximale Laufzeit fest
- `--cpus-per-task <n>` legt Anzahl der CPUs fest, die pro Task alloziert weden
- `-N<n>` legt minimale Anzahl der Nodes fest, die reserviert werden
- `-n<n>` legt Anzahl der Prozessoren fest, die wir brauchen
- `--gres=gpu:<n>` reserviert *n* GPUs
- Wartet, bis Anforderung Erfolg hatte

Interaktiver Modus - `srun`

- Nachdem `salloc` fertig ist, können wir mit srun Befehle ausführen

```
:~$ srun <command>
```

- `-n` erlaubt festzulegen, wie Instanzen des Programms gleichzeitig ausgeführt werden (Normalerweise benötigen wir nur eine Instanz)
- Achtung: Wir befinden uns weiterhin auf unserem ursprünglichen (Cluster-Login-)Computer, erst `srun` startet Befehle auf dem Cluster

Interaktiver Modus - `scancel`

- Nachdem wir fertig sind, sollten wir unsere Ressourcen freigeben

```
:~$ scancel <job-id>
```

- Job-ID wird von `salloc` zurückgegeben
- Wenn vergessen: `sacct` listet Jobs
  - Listet unsere ausgeführten Jobs
  - Laufende Jobs haben den Status `RUNNING`
  - Job-ID ist in der ersten Spalte

Batch Modus

- Interaktiver Modus ist nur zum Testen da
- Für den Produktiveinsatz: `sbatch`
- Erwartet als Eingabe Batch-file

Batch-Files

- Beschreibt einen Job für das Cluster
- Besteht aus Befehlen und `#SBATCH`-Direktiven
  - `#SBATCH` beschreibt `salloc`-Parameter
  - Befehle werden wie ein (Bash-)Script ausgeführt
- `#SBATCH` Direktiven können einfach von salloc übernommen werden
- Ausgabe wandert per default in `slurm-<jobid>.out`

Batch-Files - Beispiel

```
#!/bin/bash
#SBATCH --job-name=my_long_batch_job
#SBATCH --output=long_batch_result.txt
#SBATCH --mail-user=YOUR_USERNAME@cl.uni-heidelberg.de
#SBATCH --mail-type=ALL
#SBATCH --partition=compute
#SBATCH --time=1-0:00:00

# JOB STEPS
srun hostname # example job step
srun echo $CUDA_VISIBLE_DEVICES # another example job st
```

siehe Wiki

Warum Batch-Files?

- Beenden automatisch, sobald alle Schritte ausgeführt wurden
- Befehle werden ausgeführt, sobald Ressourcen alloziert wurden
- Fortschritt kann per Mail an uns gesendet werden

```
#SBATCH    --mail-type=ALL
#SBATCH    --mail-user=<USERNAME>@cl.uni-heidelberg.de
```

Arbeiten auf dem Cluster

Daten auf dem Cluster

- Cluster hat kein Ressourcen Verzeichnis
- Home-Verzeichnisse auf dem Cluster werden nicht mit Servern/Rechnern synchronisiert
- Daten müssen manuell transferiert werden

Werkzeuge zum Datentransport

- `scp <src-host>:<src> <dest-host>:<dest>`
  - Kopiert Datei `src` auf `src-host` nach `dest` auf `<dest-host>`
  - Wenn `src` sich auf dem lokalen Rechner befindet, kann `src-host` weggelassen werden (genauso für `dest-host`)
- `rsync`
  - Grundlegende Nutzung wie `scp`
  - Erkennt, wenn Daten im Ziel schon vorhanden sind
  - Oft schneller

Daten löschen

- Auf dem Cluster ist nicht unbegrenzt viel Platz
- Daten sollten gelöscht werden, wenn man sie nicht mehr braucht

Arbeiten auf dem Cluster

Slurm Array Jobs I

- Cluster ist nicht nur gut, um GPUs zu benutzen
- Nützlich, wenn man viele Inputdaten im selben Format hat, die auf gleiche Art und Weise verarbeitet werden
- Beispiel: Parsing von großen Corpora
- Slurm hat einen eingebauten Mechanismus: Job Arrays

Slurm Array Jobs I

```
:~$ sbatch --array <start-id>-<end-id> <script>
```

- Benötigt Batch-Script
- Startet übergebenes Batch-Script einmal für jede Zahl zwischen start-id und end-id
- Umgebungsvariable `$SLURM_ARRAY_TASK_ID` wird auf die momentane ID gesetzt

Beispiel I

- Wir nehmen an, wir haben einen Corpus in den Dateien `corpus-1`, `corpus-2`, ..., `corpus-20`
- Fiktives Programm `parse` soll Dokumente verarbeiten
- Jeder Split wird unabhängig von den anderen bearbeitet

Batch-Datei

```
#!/bin/bash
#SBATCH --ntasks=1
#SBATCH --array=1-20

srun ./parse corpus-$SLURM_ARRAY_TASK_ID
```

Arbeiten auf dem Cluster

Virtualenv

- Cluster stellt `python 2 + 3` zur Verfügung
- Bibliotheksauswahl ist aber begrenzt
- Lösung: Virtual Environment
  - Kapselt packages in lokalem Verzeichnis
  - Isoliert Installation vom Rest des Systems

Installation und Aktivierung

- Python2:

```
:~$ virtualenv --system-site-packages <env-name>
```

- Python3:

```
:~$ python3 -m venv --without-pip <env-name>
```

- oder

```
:~$  virtualenv -p python3 --system-site-packages <
env-name>
```

- Anschließend:

```
:~$ source <env-name>/bin/activate
```

- Neue Pakete installieren (nach Aktivierung):

```
:~$ pip install <package-name>
```

Python3 venv

- `--without-pip` ist ein Workaround für problematisches Debian Verhalten
- Nach Installation fehlt `pip` in der Environment
- Lösung: **Nach** Aktivierung `pip` installieren

```
:~$ curl https://bootstrap.pypa.io/get-pip.py | python
```

Übung 17

Arbeiten auf dem Cluster

GPU -CUDA I

- Installationsprozeduren für verschiedene GPU-Werkzeuge sind unterschiedlich
- Meistens müssen wir CUDA einbinden
  - Werkzeug-Kasten für Programmierung auf GPU
- Dafür müssen wir ein paar Variablen richtig setzen
  - Wiki enthält passendes Script
  - `https://wiki.cl.uni-heidelberg.de/bin/view/Main/FaQ/Tutorials/GpuHowto`

GPU - Bibliotheken I

- Da CUDA global installiert ist, müssen wir die passenden Versionen unserer Lieblingsbibliothek suchen, die darauf passt
- `https://wiki.cl.uni-heidelberg.de/bin/view/Main/FaQ/Tutorials/GpuHowto` gibt Hinweise
- Python-Bibliotheken müssen in Virtual Environment installiert werden
- Um GPUs nutzen zu können, müssen sie angefragt werden

```
#SBATCH --gres gpu:1
```

Cluster Etikette I

- Cluster wird von vielen Menschen benutzt
- Vielen brauchen schnelle Ergebnisse
- Viele Nutzer sorgen für schnelle Verstopfung und lange Queues
- Deswegen sollte man die Ressourcen so wenig wie möglich beanspruchen (v.a. Grafikkarten)

Cluster Etikette II

- Jobs beenden, sobald fertig (Batch-Files)
- Nicht mehr Ressourcen allozieren, als man braucht
- Ressourcenschonender Arbeitsablauf
  1. Erst lokal testen (z.B. ohne GPU) und debuggen
  2. Dann erst z.B. auf students ausführen