Verteilte Systeme – Übung

Grundlagen: Verteilte Ausführung

Sommersemester 2024

Harald Böhm, Laura Lawniczak, Tobias Distler

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)
Lehrstuhl Informatik 16 (Systemsoftware)

https://sys.cs.fau.de

Lehrstuhl für Verteilte Systeme
und Betriebssysteme

FAU Friedrich-Alexander-Universität
Technische Fakultät

Überblick

Verteilte Ausführung

Verteilte Ausführung

Java-Programme kompilieren und ausführen

- Kompilieren von Java-Programmen

```
> javac -cp 'lib1.jar:libs/*' -d bin File1.java ...
```

  - Klassenpfad (`-cp`) muss verwendete Bibliotheken beinhalten
    - → Besteht aus `jar`-Dateien und Ordnern mit `class`-Dateien
    - → Platzhalter `*` expandiert zu allen `.jar`-Dateien im jeweiligen Ordner
    - → Pfade durch „:“ getrennt
  - Ausgabeverzeichnis `-d bin` für kompilierte `class`-Dateien
  - Quellcodedateien übergeben

- Ausführen von Java-Programmen

```
> java -cp 'bin:lib1.jar:libs/*' [-Dparam=value] package.name.Entrypoint [args ...]
```

  - Klassenpfad um Ausgabeverzeichnis für kompilierte Klassen ergänzen
  - Systemeigenschaften mit `-Dparam=value` übergeben
    - → Abfrage per `System.getProperty("param", "default");`
  - Ausführung startet in der Klasse `package.name.Entrypoint`
  - Restliche Parameter werden an das Java-Programm übergeben

Verteiltes Debugging

- „`printf`“-Debugging
  - An unterschiedlichen Stellen im Programm Debugausgaben erzeugen
  - Zuordnung von Ausgabe zu Programmzeilen sollte möglich sein
  - Bei großen Ausgabemengen in Dateien umleiten
  - Ausgaben mit Zeitstempeln versehen
    Achtung: Uhren der Rechner können im verteilten Fall voneinander abweichen

  Wichtig: Ausgaben verändern ggf. Programmverhalten (*I/O ist langsam!*)

- Debugger
  - Einzelne(n) Java-Prozess(e) im Debugger starten
  - Restliche Prozesse normal starten

  Wichtig: Pausieren im Debugger hält nur den zugehörigen Prozess an. Restliche Prozesse laufen normal weiter.
  → Gefahr von unerwartetem Verhalten durch Timeouts

- Läuft **überall** der aktuelle Programmcode?

- Protokoll für sichere Kommunikation über unsichere Netzwerke
  - SSH-Clients kommunizieren mit SSH-Servern über TCP (meist Port 22)
  - Public-Key-Verfahren für Verschlüsselung und Authentifizierung
- Anwendungen
  - Zugriff auf Rechner `host` unter Benutzernamen `user`

```
> ssh [<user>@]<host>
```

Hinweis: Innerhalb des CIP-Pool-Netzes sind einfache Hostnamen wie `cip2a0` ausreichend. Ansonsten muss der **Domänenname** mit angegeben werden, z. B. `cip2a0.cip.cs.fau.de`.

  - Befehl `cmd` auf Rechner `host` ausführen

```
> ssh [<user>@]<host> <cmd>
```

  - Authentifizierung mit SSH-Schlüssel gegenüber dem entfernten Rechner

```
> ssh [-i <ssh-key>] [<user>@]<host>
```

→ Standard: Verwendung von SSH-Schlüssel unter `~/.ssh/id_rsa`
→ Erstellung des Keys mittels `ssh-keygen`
→ Übermittlung an entfernten Rechner am Besten mit `ssh-copy-id`

- Kopieren von Dateien zwischen Rechnern

```
> scp <path_src> <path_dst>
```

Für entfernte Pfade: `[<user>@]<host>:<path_remote>`, Beispiele:

```
> scp cip2a0:/tmp/srcfile .
> scp /tmp/srcfile user@cip2a0:      # Ziel: Home von user
> scp -r cip2a0:srcdir cip2a0:/tmp  # Rekursiv, Ordner kopieren
```

- **Hinweis:** Die Verzeichnisse `/home` und `/proj` auf CIP-Pool-Rechnern werden per NFS (Network File System) bereitgestellt. Dadurch enthalten diese auf allen Rechner dieselben Dateien

```
> scp README faui00a:
> ssh faui00b cat README
```

- Automatisieren häufiger Vorgänge
  - Skript zum Starten der Anwendung (Dateiname: start-server.sh)

```
#!/bin/bash
echo "Starte␣Anwendung␣mit␣Parametern␣$@"
java -cp <classpath> vs.queue.VSQueueServer "$@"
```

  - Skript ausführen

```
> chmod +x start-server.sh  # einmalig als ausfuehrbar markieren
> ./start-server.sh param1 param2 ...
Starte Anwendung mit Parametern param1 param2 ...
```

- Bash-Skripte debuggen
  - Hinzufügen von `echo`-Anweisungen
  - Starten mit `bash -x`

```
> bash -x start-server.sh param1 param2 ...
```

- Wiki / Tutorialsammlung

The Bash Hackers Wiki

http://wiki.bash-hackers.org/start

- Aus- und wieder einhängbare Terminals
- Programme laufen auch bei getrennter Sitzung weiter
- Verwendung:
  - Starten eines Screens:

```
> screen
```

    Aushängen (*detach*) eines Screens mittels `'Ctrl+a d'`

  - Auflisten aller laufenden Sitzugen

```
> screen -ls
There are screens on:
16656.pts-145.faui48f (25.10.2019 12:10:06) (Attached)
16457.pts-123.faui48f (25.10.2019 12:27:59) (Attached)
2 Sockets in /var/run/screen/S-lawniczak.
```

  - Bestimmte Sitzung fortsetzen

```
> screen -dr 16457.pts-123.faui48f
```

Alternative: `tmux`