

# Middleware – Cloud Computing – Übung

## Grundlagen: Verteilte Ausführung

---

Wintersemester 2023/24

Laura Lawniczak, Tobias Distler, Harald Böhm

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

<https://sys.cs.fau.de>



Lehrstuhl für Verteilte Systeme  
und Betriebssysteme



FRIEDRICH-ALEXANDER  
UNIVERSITÄT  
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Verteilte Ausführung

## Verteilte Ausführung

---

## ■ 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

## ■ „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 `fau100a` ausreichend. Ansonsten muss der **Domänenname** mit angegeben werden, z. B. `fau100a.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 faui00a:/tmp/srcfile .
> scp /tmp/srcfile user@faui00a:      # Ziel: Home von user
> scp -r faui00a:srcdir faui00a:/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 $@"
java -cp <classpath> mw.queue.MWQueueServer "$@"
```

- 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 Sitzungen

```
> screen -ls
There are screens on:
16656.pts-145.faii48f (25.10.2019 12:10:06) (Attached)
16457.pts-123.faii48f (25.10.2019 12:27:59) (Attached)
2 Sockets in /var/run/screen/S-lawniczak.
```

- Bestimmte Sitzung fortsetzen

```
> screen -dr 16457.pts-123.faii48f
```

**Alternative:** tmux