

## Java

Thread-Pools  
Executor-Service  
HttpClient

## Amazon Web Services

Überblick  
Elastic Compute Cloud (EC2)  
Simple Storage Service (S3)  
Elastic Load Balancing  
Amazon Java SDK

## Aufgabe 6



# Runnable

`java.util.concurrent.Runnable`

- Für die abzuarbeitenden Tasks wird eine Klasse benötigt, die das Interface `Runnable` implementiert
- Die Signatur der zu implementierende Methode `run()` hat *keine* Aufrufparameter
- Daten müssen deshalb entweder mit Hilfe des Konstruktors übergeben werden (siehe Beispiel) oder bei der Ausführung von `run()` zur Laufzeit in Erfahrung gebracht werden

```
public class MWRunnable implements Runnable {
    private int id;

    public MWRunnable(int id) {
        this.id = id;
    }

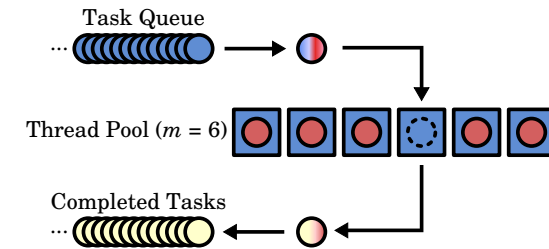
    public void run() {
        System.out.println("ID: " + this.id);
    }
}
```



# Thread-Pools

`java.util.concurrent`

- Ein Thread-Pool ist eine zentral verwaltete Liste von Worker-Threads
- Einem Thread-Pool der Größe  $m$  werden  $n$  Tasks zugeordnet
- Für gewöhnlich gilt:  $n \gg m$



- Für die Abarbeitung der Tasks muss kein eigener Thread gestartet werden → Worker-Thread
- Vorteil: Geringerer Aufwand für die Speicherverwaltung



# Executor-Service

`java.util.concurrent.Executor{S,Service}`

- In Java reduziert der Executor-Service den Programmieraufwand für Thread-Pools
- Erstellung eines Thread-Pool mit einer definierten Größe:

```
ExecutorService svc = Executors.newFixedThreadPool(int m);
```

- Der Parameter `m` spezifiziert die Größe des Thread-Pool
- Für die Ausführung von insgesamt  $n$  Tasks gilt es, die Methode `execute` zu verwenden:

```
for (int id = 0; id < n; id++) {
    svc.execute(new MWRunnable(id));
}
```

- Der Executor-Service garantiert, dass bei der Abarbeitung der  $n$  Tasks, maximal  $m$  Threads gleichzeitig ausgeführt werden



- Mittels der Klasse HttpClient soll festgestellt werden, ob ein Web-Service auf Anfragen antwortet
- Beispiel (Zugriff auf http://www.example.com/foo.bar):

```
HttpClient httpClient = new HttpClient();
String url = "http://www.example.com:80/foo.bar";
GetMethod method = new GetMethod(url);
try {
    int statusCode = httpClient.executeMethod(method);
    if (statusCode == HttpStatus.SC_OK) {
        /* Erfolgreicher Zugriff */
    }
} catch (Exception e) {
    /* Ausnahmebehandlung */
} finally {
    method.releaseConnection();
}
```

- Grundgerüst für eine periodische Abfragemethode, die prüft ob auf einem Server (z. B. VM) ein Web-Service läuft



Java

Thread-Pools  
Executor-Service  
HttpClient

Amazon Web Services

Überblick

Elastic Compute Cloud (EC2)  
Simple Storage Service (S3)  
Elastic Load Balancing  
Amazon Java SDK

Aufgabe 6



## Amazon Web Services (AWS)

- Die Amazon Web Services bestehen aus Diensten, die den Aufbau komplexer Systeme in einer Cloud-Infrastruktur ermöglichen
- Dienste (Auszug):
  - Elastic Compute Cloud (EC2) – Betrieb virtueller Maschinen
  - Simple Storage Service (S3) – Netzwerkbasierter Speicher-Dienst
  - Elastic Load Balancing – Lastverteilung für EC2
  - Elastic Map Reduce – MapReduce-Framework basierend auf EC2 und S3
  - DynamoDB – Key-Value-Store basierend auf Dynamo
- Die Abrechnung erfolgt nach tatsächlichem Verbrauch **und** Standort
  - Betriebsstunden, Speicherbedarf
  - Transfervolumen, Anzahl verarbeiteter Anfragen
  - Standorte in Nord- und Südamerika, Europa und Asien-Pazifik

G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Voshall, and W. Vogels.  
**Dynamo: Amazon's Highly Available Key-value Store.**  
In *Proc. of the 21st Symposium on Operating Systems Principles*. ACM, 2007.



## Amazon Web Services (AWS)



## Amazon Web Services, Betriebsumgebung

- Jede Gruppe hat für die Benutzung der Amazon Web Services eine E-Mail mit Account-Informationen erhalten
- Einrichten der Betriebsumgebung (CIP-Pool)

- Anlegen der privaten Konfigurationsdatei `~/.aws/aws.conf`

```
$ cd ~
$ mkdir .aws
$ touch .aws/aws.conf
```

- Eintragen von `AWS_ACCESS_KEY_ID` und `AWS_SECRET_ACCESS_KEY`

```
export AWS_ACCESS_KEY_ID=<schluessel_id>
export AWS_SECRET_ACCESS_KEY=<privater_schluessel>
```

→ <http://tinyurl.com/access-keys>, Reiter „Access Keys“

- Hinterlegen des privaten Schlüssels und des Zertifikats

- Neues Zertifikat erstellen
- Abspeichern des *privaten* Schlüssels unter `~/.aws/key.pem`
- Abspeichern des Zertifikats unter `~/.aws/cert.pem`

→ <http://tinyurl.com/x509-certificates>, Reiter „X.509 Certificate“

- Zugriffsrechte einschränken

```
$ chmod 600 ~/.aws/*
```



## Betriebsumgebung

- Nach Erstellen von `~/.aws/aws.conf` und dem Hinterlegen des privaten Schlüssels sowie des Zertifikats kann die globale Konfigurationsdatei geladen werden

```
$ source /proj/i4mw/pub/aufgabe6/aws.conf
```

- Diesen Schritt nach jedem Login im CIP-Pool ausführen, damit die EC2-Kommandozeilen-Tools (z. B. `ec2-run-instances`) funktionieren

- Alternative: Obigen Befehl *einmalig* in die Datei `~/.profile` eingetragen um den manuellen Aufruf zu vermeiden.

```
$ echo "source /proj/i4mw/pub/aufgabe6/aws.conf" >> ~/.profile
```

- Liste der verfügbaren EC2-Kommandozeilen-Tools:

```
$ ec2-<TAB><TAB>
```



## Amazon Elastic Compute Cloud (EC2)

- Voraussetzungen für die Instanziierung einer virtuellen Maschine

- Amazon Machine Image (AMI, Liste: `ec2-describe-images -a`)
- EC2-Schlüsselpaar

- Bei der Instanziierung muss die Größe der virtuellen Maschine festgelegt werden

- Instanz-Typen variieren in Anzahl der CPU-Kerne, Speichergröße etc.

→ <http://aws.amazon.com/ec2/instance-types/>

- Für Testzwecke reicht der Betrieb kleiner Instanzen aus

→ API-Name: `m1.small`

- Nutzdatenfeld `user-data`

- Base64-kodierter String
- Maximal 16 kByte
- Optional



## Amazon EC2: Starten einer Instanz

- Einmalig EC2-Schlüsselpaar im Browser generieren

- Schlüsselname wählen (z. B. `gruppe0`)
- Privaten Schlüssel unter `~/.aws/gruppe0.pem` speichern

→ <https://console.aws.amazon.com/ec2/home?region=eu-west-1#s=KeyPairs>

- Zugriffsrechte mit `chmod` setzen

```
$ chmod 600 ~/.aws/gruppe0.pem
```

- Persistente Freigabe von Port 22 (SSH), Starten einer Linux-Instanz

- AMI: `ami-7fd4e10b`
- Instanz-Typ: `m1.small`
- Schlüsselname: `gruppe0`
- Nutzdatenfeld mit einem String füllen: `Hello World.`

```
$ ec2-authorize default -p 22
$ ec2-run-instances --instance-type m1.small --key gruppe0 \
  --user-data="Hello World." ami-7fd4e10b
```

- Überprüfen des Status der Instanz mit `ec2-describe-instances`



## Amazon EC2: Zugriff auf eine Instanz

- Sobald die Instanz den Boot-Vorgang abgeschlossen hat, kann per SSH auf die Maschine zugegriffen werden

→ Öffentlicher Hostnamen der Instanz via `ec2-describe-instances`

```
$ ssh -i ~/.aws/gruppe0.pem -l ec2-user \  
ec2-xxx-xxx-xxx-xxx.eu-west-1.compute.amazonaws.com
```

- Bei Konflikten aufgrund erneuter Adressvergabe, SSH mit dem Parameter `-o StrictHostKeyChecking=no` starten
- Hinweise:
  - In der Betriebsumgebung der virtuellen Maschine werden mit `ec2-metadata` Meta-Informationen über das System angezeigt. Auch das Nutzdatenfeld `user-data` kann so ausgewertet werden.
  - Root-Rechte erhält man mit dem Kommando `sudo su -`
- Bei Zugriffsproblemen: Boot-Meldungen über die Web-Schnittstelle oder mit `ec2-get-console-output` nach Fehlern durchsuchen
- Debugging auf dem Live-System: Prüfen der Log-Dateien (`/var/log/*`)



## Amazon EC2: Beenden einer Instanz

- Für das Terminieren einer im Betrieb befindlichen Instanz ist die eindeutige Instanz-ID notwendig
- Das Kommando `ec2-describe-instances` listet die Instanz-ID in der zweiten Spalte (Format: `i-xxxxxxx`)
- Unter Kenntnis dieser ID kann die Instanz mit `ec2-terminate-instance` beendet werden:

```
$ ec2-describe-instances  
(...)  
$ ec2-terminate-instances i-xxxxxxx
```

- Mit dem Befehl `ec2-terminate-all-instances` (nur im CIP-Pool verfügbar!) werden alle laufenden Instanzen des Benutzers beendet
- Kontrolle: <https://console.aws.amazon.com/ec2/home>
- Bitte sicherstellen, dass **keine unbenutzten** Instanzen laufen!



## Amazon Simple Storage Service (S3)

- Der Simple Storage Service (S3) ist ein Netzwerk-Dateisystem
  - REST-, SOAP- und BitTorrent-Schnittstellen
  - Zugriffskontrolle mittels Zugriffskontrolllisten (Access Control Lists, ACL)
  - Einfache API
- Eindeutige Identifikation von Dateien durch Bucket (Kübel) und Dateiname: `s3://<bucket>/<dateiname>`
- Buckets können *nicht* geschachtelt werden
- Übersetzung der S3-Adressrepräsentation in eine URL
  - S3: `s3://<bucket>/<dateiname>`
  - URL: `http://<bucket>.s3.amazonaws.com/<dateiname>`
- In S3 abgespeicherte Daten werden auf mehrere Standorte repliziert
- Prominente Dienste, die S3 nutzen:
  - Dropbox
  - Twitter (Bildaten)



## Amazon S3: Zugriff auf Daten

- Zugriff auf Daten in S3 im CIP-Pool via `s3cmd`
- Einmalige Konfiguration:

```
$ s3cmd --configure  
Access Key: <schluessel_id>  
Secret Key: <privater_schluessel>  
Encryption password: <leer>  
(...)  
Save settings? [y/N] y
```
- Lediglich „Access Key“ und „Secret Key“ müssen angegeben werden  
→ <http://tinyurl.com/access-keys>, Reiter „Access Keys“

- Erstellen eines Bucket:

```
$ s3cmd mb s3://gruppe0-bucket  
Bucket 's3://gruppe0-bucket/' created
```

- Speichern einer *öffentlichen* Datei im Bucket `gruppe0-bucket`:

```
$ echo "Hello World." > foo.bar  
s3cmd --acl-public put foo.bar s3://gruppe0-bucket/foo.bar  
foo.bar -> s3://gruppe0-bucket/foo.bar [1 of 1]  
Public URL of the object is:  
http://gruppe0-bucket.s3.amazonaws.com/foo.bar
```



## Amazon S3: Zugriff auf Daten

- Laden der Datei `foo.bar` aus dem Bucket `gruppe0-bucket`:

```
$ s3cmd get s3://gruppe0-bucket/foo.bar foo.bar.copy
```

- Löschen der Datei `foo.bar` aus dem Bucket `gruppe0-bucket`:

```
$ s3cmd del s3://gruppe0-bucket/foo.bar
File s3://gruppe0-bucket/foo.bar deleted
```

- Ausführliche Liste der `s3cmd`-Befehle:

```
$ s3cmd --help
```

- Alternative Zugriffsmethoden:

- Browser (Amazon Web Services Console, <https://console.aws.amazon.com/s3/home>)
- Firefox-Plugin (z. B. S3Fox, <http://www.s3fox.net>)
- Einhängen als Dateisystem (s3fs, FUSE-basiert)



## Amazon Elastic Load Balancing

- Mit dem Dienst „Amazon Elastic Load Balancing“ können virtuelle Load-Balancer (Lastverteilungsknoten) in EC2 definiert werden
- Erstellung eines Load-Balancer
  - Symbolischer Name, findet sich im endgültigen Hostnamen wieder
  - Port-Zuweisung(en), Load-Balancer-Port und Ziel-Port der EC2-Instanzen können variieren
  - „Health-Check“ und direkte Zuweisung von EC2-Instanzen sind bei der Erstellung optional
- Einem Load-Balancer können nach seiner Erstellung beliebig viele EC2-Instanzen zugeordnet werden
- Eingehende Anfragen werden im *Round-Robin*-Verfahren an die zugeordneten EC2-Instanzen weitergereicht
- Ansonsten ist der Load-Balancer eine passive Instanz; er instanziiert oder terminiert *keine* Instanzen

→ <https://console.aws.amazon.com/ec2/home?region=eu-west-1#s=LoadBalancers>



## Amazon Java SDK

- Amazon stellt eine Java-Bibliothek für die Verwendung der Amazon Web Services zur Verfügung

→ `/proj/i4mw/pub/aufgabe6/aws-java-sdk-1.1.2.jar`

→ <http://docs.amazonwebservices.com/AWSJavaSDK/latest/javadoc/index.html>

- Relevante Packages für den Betrieb von virtuellen Maschinen und Load-Balancern in EC2:

- `com.amazonaws.services.ec2`
- `com.amazonaws.services.elasticloadbalancing`

- Folgende Objekte sind bei der Instanziierung einer virtuellen Maschine in EC2 beteiligt

1. Anmeldedaten („Credentials“, Typ `BasicAWSCredentials`)
2. Client-Objekt (Typ `AmazonEC2Client`)
3. Instanzierungs-Request (Typ `RunInstancesRequest`)
4. Ergebnis (Typ `RunInstancesResult`)



## Amazon Java SDK: Instanziierung einer VM

- Minimal-Beispiel (analog Kommandozeilen-Beispiel)

```
AWSCredentials credentials = \
    new BasicAWSCredentials(accessKeyID, secretKey);
AmazonEC2Client ec2 = new AmazonEC2Client(credentials);
ec2.setEndpoint("https://ec2.eu-west-1.amazonaws.com");
RunInstancesRequest request = new RunInstancesRequest();
```

```
String userData = "Hello World.";
request.setInstanceType("m1.small");
request.setKeyName("gruppe0");
request.setImageId("ami-7fd4e10b");
request.setMinCount(1);
request.setMaxCount(1);
request.setPlacement(new Placement("eu-west-1a"));
```

```
byte[] userDataBytes = userData.getBytes();
request.withUserData(Base64.encodeBase64String(userDataBytes));
RunInstancesResult result = ec2.runInstances(request);
```

- Hinweise:

- Mittels des Objektes `result` die Instanz-ID in Erfahrung bringen
- Auf die eigentliche Instanziierung prüfen (`DescribeInstancesRequest`)



## Amazon Java SDK: Erstellen eines Load-Balancer

- Entsprechend die Erstellung eines Load-Balancer:
  1. Anmeldedaten („Credentials“, Typ BasicAWSCredentials, unverändert)
  2. Client-Objekt (Typ AmazonElasticLoadBalancingClient)
  3. Portzuweisungs-Objekt (Typ Listener)
  4. Instanzierungs-Request (Typ CreateLoadBalancerRequest)
  5. Ergebnis (Typ CreateLoadBalancerResult)

- Beispiel:

```
AmazonElasticLoadBalancingClient lb = \
    new AmazonElasticLoadBalancingClient(credentials);
lb.setEndpoint("https://elasticloadbalancing.eu-west-1." + \
    "amazonaws.com");
```

```
List<Listener> listeners = new ArrayList<Listener>();
listeners.add(new Listener("tcp", 80, 80));
List<String> zones = new ArrayList<String>();
zones.add(new String("eu-west-1a"));
```

```
CreateLoadBalancerRequest request = \
    new CreateLoadBalancerRequest("lb-name", listeners, zones);
CreateLoadBalancerResult result = \
    lb.createLoadBalancer(request);
```



## Amazon Java SDK: Zuordnung einer Instanz

- Einem Load-Balancer können im Folgenden beliebige EC2-Instanzen zugewiesen werden
- Beispiel:

```
List<Instance> instances = new ArrayList<Instance>();
instances.add(...);
```

```
RegisterInstancesWithLoadBalancerRequest request = \
    new RegisterInstancesWithLoadBalancerRequest();
request.setLoadBalancerName("lb-name");
request.setInstances(instances);
lb.registerInstancesWithLoadBalancer(request);
```

- Nach der Zuordnung der Instanz leitet der Load-Balancer gemäß seiner Portzuweisung Anfragen an die Instanz weiter
- Die Portzuweisung kann zur Laufzeit verändert werden
- Antworten der EC2-Instanz werden durch den Load-Balancer an den anfragenden Host geschickt



## Überblick

### Java

Thread-Pools  
Executor-Service  
HttpClient

### Amazon Web Services

Überblick  
Elastic Compute Cloud (EC2)  
Simple Storage Service (S3)  
Elastic Load Balancing  
Amazon Java SDK

### Aufgabe 6

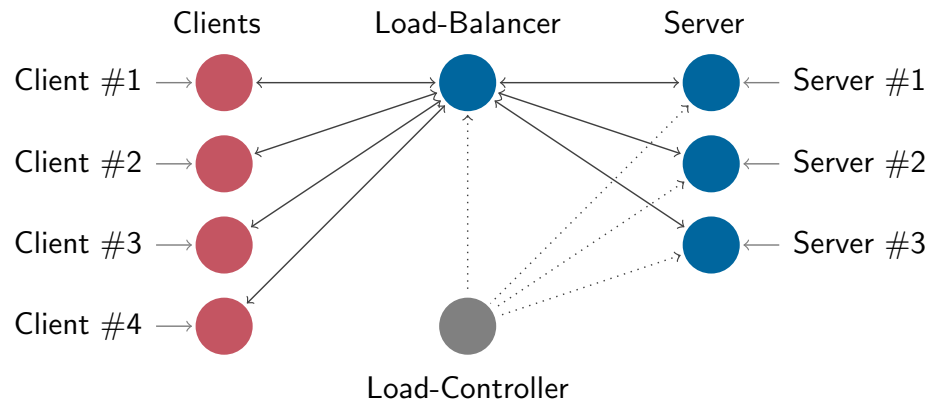


## Aufgabe 6

- Web-Services in der Cloud-Computing-Umgebung von Amazon
  - Erweiterter Web-Service aus der ersten Übungsaufgabe:
    - Lastverteilung
    - Dynamische Skalierung
  - Amazon-Basis-Technologien:
    - Amazon EC2
    - Amazon S3
    - Amazon Elastic Load Balancing
- Amazon Web Services
  - Jede Übungsgruppe verfügt über ein Guthaben von 100 US-Dollar
  - Guthaben kann lediglich für Amazon Web Services verwendet werden
  - Aktuelle AWS-Kosten: <http://aws.amazon.com/pricing/>
- Globaler Systemstatus der Amazon Web Services
  - Bei Störungen können (Teile der) Amazon Web Services ausfallen
  - Aktueller Status: <http://status.aws.amazon.com/>



## Aufgabe 6: Systemübersicht



## Hinweise für Aufgabe 6

- Erweiterung des Web-Service aus der ersten Aufgabe
  - MWFacebookServer: Erfassung der Auslastung des Dienstes  
→ `/proj/i4mw/pub/aufgabe6/src/MWFacebookServer.tar.gz`
  - MWClient: Abfrage der Server-Auslastung, Protokollierung der Antwortzeiten des Server und Erzeugung hoher synthetischer Systemlast
- Betrieb des Dienstes in Amazon EC2
  - Java-Archive (→ Hadoop-Übung vom 14.12.2012) für Server und Client  
**Hinweis:** Die Quellen der Java-Archive müssen mit Java 1.6 übersetzt werden.
  - Hinterlegen der Archive auf S3
  - EC2-Instanzen starten  
**Hinweis:** Das bereitgestellte Amazon-Machine-Image `ami-b36652c7` verfügt über die Daten, die für den Betrieb des Facebook-Service notwendig sind
- Load-Controller
  - Instanziierung und Terminierung von Server-Instanzen mit Java
  - Erstellen eines Load-Balancer in EC2, Zuordnung von Instanzen
  - Lastüberwachung und entsprechende Regulierung des Systems

