Middleware – Cloud Computing – Übung

Verteilte Dateisysteme & Container: Aufgabe 3

Wintersemester 2020/21

Michael Eischer, Laura Lawniczak, Tobias Distler

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)
www4.cs.fau.de

Lehrstuhl für Verteilte Systeme und Betriebssysteme

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Überblick

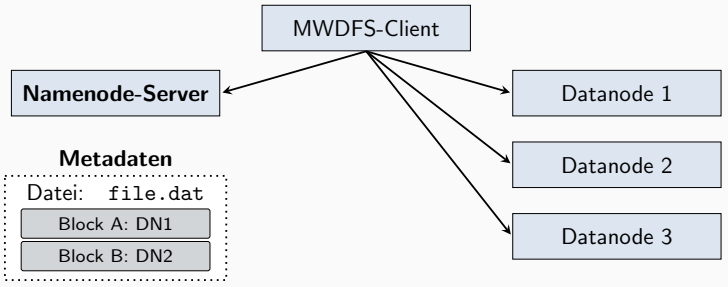
Aufgabe 3

- Übersicht
- Hinweise zu Java

Aufgabe 3

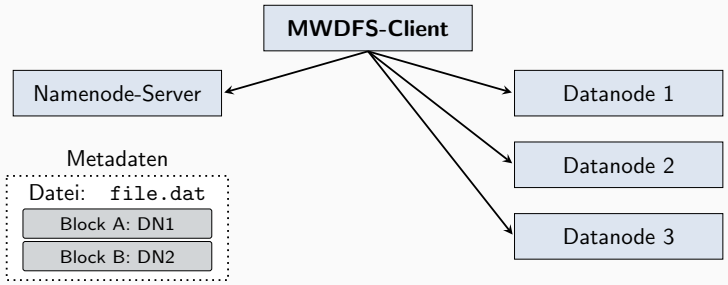
Übersicht

Übersicht — Alle



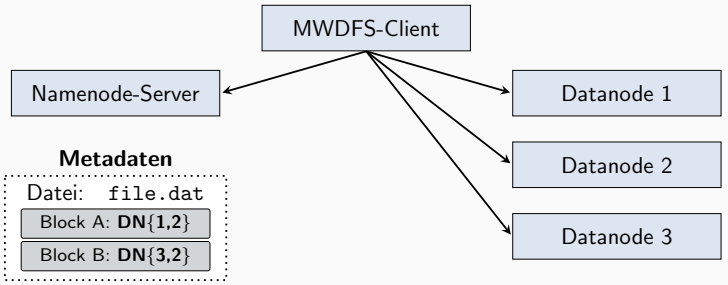
- **Namenode-Server**
  - **Metadaten**
  - Datei-Operationen (Anlegen, Anzeigen, Löschen)
  - Leases für Schreibzugriffe

Übersicht — Alle



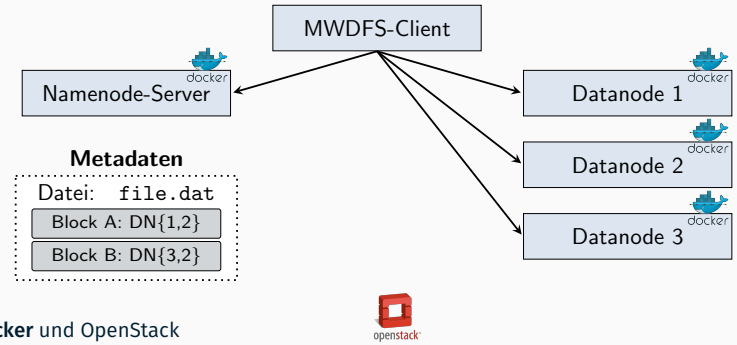
- **MWDFS-Client**
  - Datenzugriff
  - Datei-Operationen (Anlegen, Anzeigen, Löschen)

Übersicht — 7.5 ECTS



- **Replikation** (optional für 5,0 ECTS)
  - Datenblöcke redundant auf mehreren Datanodes speichern
  - Erweiterung der serverseitigen **Metadaten**
- **Zustandspersistierung** (optional für 5,0 ECTS)
  - Effizientes Schreiben der Dateimetadaten bzw. Operationen
  - Wiederherstellung des Zustands nach Namenode-{Absturz,Neustart}

Übersicht — Optional



- **Docker** und OpenStack
  - Docker-Images erstellen
  - Betrieb von Namenode-Server und drei Datanodes als **Docker-Container**
    → OpenStack-Cloud
  - Zugriff auf das System über MWDFS-Client
    → CIP-Pool

Aufgabe 3

Hinweise zu Java

{S,Des}erialisierung in Java

- {S,Des}erialisierung mittels `{Data,Buffered,File}{Output,Input}Stream`
- Öffnen der Ströme zum Schreiben und Lesen

```
// Holen der Ausgabestroeme (Schreiben in Datei 'journal')
FileOutputStream fos = new FileOutputStream("journal");
DataOutputStream dos = new DataOutputStream(new BufferedOutputStream(fos));

// Holen der Eingabestroeme (Lesen aus Datei 'journal')
FileInputStream fis = new FileInputStream("journal");
DataInputStream dis = new DataInputStream(new BufferedInputStream(fis));
```

- Schreiben und Lesen von Daten
  - `write`- und `read`-Methoden für unterschiedliche Datentypen (z.B. `writeInt()`, `writeBytes()`)
  - Erzwingen des Schreibvorgangs auf Datenträger mittels Aufruf von `force()` am `FileChannel`-Objekt
    → `boolean`-Parameter von `force`: 'true' := Dateiinhalt **und** -metadaten schreiben

```
dos.writeLong(42);
dos.flush(); // Puffer leeren
fos.getChannel().force(true);
```


JAX-RS: Übertragung von Binärdaten

- Datanodes empfangen (`POST`) und senden (`GET`) Blockdaten als Binärdaten
- Client-Zugriffe zum Senden und Empfangen eines Datenblocks
  - Für `POST`-Anfrage `Entity`-Objekt mit geeignetem MIME-Type wählen:
    „`application/octet-stream`“ → `MediaType.APPLICATION_OCTET_STREAM`

```
// WebTarget datanode zeigt auf http://<server>/datablock/<blockid>
public void sendBlockToDatanode(byte[] block, WebTarget datanode) {
    try {
        Response r = datanode.request()
                             .post(Entity.entity(block, MediaType.APPLICATION_OCTET_STREAM));
    } [...] // Fehlerbehandlung
}
```

  - Für `GET`-Anfrage Response-Type auf `byte[]` setzen

```
public byte[] receiveBlockFromDatanode(WebTarget datanode) {
   byte[] block = null;
   try {
       block = datanode.request().get(byte[].class);
   } [...] // Fehlerbehandlung
   return block;
}
```