JUST DO IT
HÖHERE ABTEILUNG FÜR
INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Protokoll

Verteiltes Dateisystem mit OriFS

Systemtechnik-DezSys
5BHIT 2015/16

Thomas Taschner & Michael Weinberger

Version 1.0
Note:
Begonnen am 1. April 2016
Betreuer: Michael Borko
Beendet am 22. April 2016

Inhaltsverzeichnis

1 Installation und Implementierung

"*Ori is a distributed file system built for offline operation and empowers the user with control over synchronization operations and conflict resolution. We provide history through light weight snapshots and allow users to verify the history has not been tampered with. Through the use of replication instances can be resilient and recover damaged data from other nodes.*" [6]

Installieren Sie Ori und testen Sie die oben beschriebenen Eckpunkte dieses verteilten Dateisystems (DFS). Verwenden Sie dabei auf jeden Fall alle Funktionalitäten der API von Ori um die Einsatzmöglichkeiten auszuschöpfen. Halten Sie sich dabei zuallererst an die Beispiele aus dem Paper im Kapitel 2 [7]. Zeigen Sie mögliche Einsatzgebiete für Backups und Roadwarriors (z.B. Laptopbenutzer möchte Daten mit zwei oder mehreren Servern synchronisieren). Führen Sie auch die mitgelieferten Tests aus und kontrollieren Sie deren Ausgaben (Hilfestellung durch Wiki [8]).

1.1 Gegenüberstellung

Wo gibt es Überschneidungen zu anderen Implementierungen von DFS? Listen Sie diese auf und dokumentieren Sie mögliche Entscheidungsgrundlagen für mindestens zwei unterschiedliche Einsatzgebiete. Verwenden Sie dabei zumindest HDFS [9] und GlusterFS [10] als Gegenspieler zu Ori. Weitere Implementierungen sind möglich aber nicht verpflichtend. Um aussagekräftige Vergleiche anstellen zu können, wäre es von Vorteil die anderen Systeme ebenfalls - zumindest oberflächlich - zu testen.

1.2 Info

Gruppengröße: 2 Mitglieder Gesamtpunkte: 16

- Installation und Testdurchlauf von Ori: 2 Punkte
- Einsatz/Dokumentation der Ori API (replicate, snapshot, checkout, graft, filelog, list, log, merge, newfs, pull, remote, removefs, show, status, tip, varlink): 8 Punkte
- Gegenüberstellungstabelle: 4 Punkte
- Einsatz der Gegenspieler: 2 Punkte

2 Ergebnisse

2.1 Installation und Testdurchlauf von Ori

Für die Übungsdurchführung wird mithife von Vagrant eine 'ubuntu/wily64'-Box bereitgestellt. Das dazugehörige Vagrantfile befindet sich in unserem Repository. [11] Zum Ausprobieren wurde dazu noch eine Debian-VM mit LXDE aufgesetzt, um die Funktionalität optimal testen zu können. Auf der Herstellerseite sind auch einige der wichtigsten Kernaspekte von OriFS aufgelistet:

- Peer-to-Peer
  'Ori operates peer-to-peer among your devices and uses existing secure communication channels such as SSH to transfer your data.'
- Work Offline
  'In today's world we often are moving around with intermittent network connectivity and we want to access our data when we board a plane or travel to the office.'
- Secure
  'Ori can verify the authenticity of your data and ensure it has not been tampered with. Data is transfered over SSH. Device discovery and automatic synchronization uses a shared secret to initiate transfers.'
- Instant Access
  'Instantly mount remote file systems and start working while you synchronize data in the background.' [6]

2.1.1 Fehler bei Installation über Ubuntu PPA

Die Installationsanleitung auf der Seite der Hersteller [6] ist falsch.

```
1 add-apt-repository ppa:ezyang/ppa
  apt-get update
  # Hier entsteht der Fehler: 404 Not Found
  # Einige Indexdateien konnten nicht heruntergeladen werden.
  # Sie wurden ignoriert oder alte an ihrer Stelle benutzt.
6 apt-get install ori # Funktioniert nicht!
```

Listing 1: Anleitung zur Installation laut Hersteller - fehlerhaft

Hierbei kommt es zu einem 404-Fehler, die Informationen zur Installation sind nicht mehr aktuell. Diese Paketquelle ist nicht mehr vorhanden, da sie bereits in den Standard-Paketquellen vorhanden sind.

2.1.2 Installation in Ubuntu

Im Zuge dessen ist es regelrecht einfach, OriFS zu installieren. Folgende Befehle sind notwendig:

```
apt-get update
apt-get install ori
```

Listing 2: Anleitung zur Installation

Daraufhin ist OriFS unter Ubuntu vollständig installiert.

2.1.3 Kompilieren und Installation 'from source' mit SCons

OriFS ist laut den Herstellern verfügbar für FreeBSD, OS X, Arch Linux und Ubuntu. Es gibt aber natürlich auch eine andere Art, das Programm zu erlangen: Direkt über den frei zugänglichen Quelltext kompilieren und installieren. Dafür sind folgende Schritte notwendig:

```
apt-get update

apt-get install scons build-essential pkg-config -y
apt-get install libboost-dev uuid-dev libfuse-dev libevent-dev libssl-dev -y

wget https://bitbucket.org/orifs/ori/downloads/ori-0.8.1.tar.xz
tar xvfJ ori-0.8.1.tar.xz
cd ori-0.8.1

scons

scons PREFIX=/usr/local/ install
```

Listing 3: Anleitung zur Installation 'from source'

Hierbei wird unter anderem SCons als Build-Tool verwendet. Der Installationspfad '/usr/local' ist daher vorzuziehen, weil hier vom Administrator Programme und Dateien ablegt werden können, die von der entsprechenden Distribution des jeweiligen Systems unabhängig installiert worden sind, wie etwa selbstkompilierte oder unabhängig von der Distribution heruntergeladene Programme.

2.1.4 OriFS-Testfälle ausführen

Im entpackten Sourcecode-Archiv sind auch die von uns gewünschten Testfälle enthalten. Folgendes Skript führt diese aus:

```
./runtests.sh
```

Listing 4: Erster Versuch Testfälle

Nach nur wenigen Testfällen schlägt der Lauf fehl. Eine Recherche ergibt, dass dies keineswegs ein Einzelfall ist. Das mitgelieferte README-File, zitiert:
'There are multiple unit tests available inside the build directory. The end-to-end tests are in ongoing development and only about half of them are expected to run reliably. In a future release the tests will be improved to make it easier to run.'

Die in der Aufgabenstellung beschriebene Hilfestellung schafft Abhilfe. [8] Im folgenden Schritt wird im extrahierten Archiv die Datei *runtests_ config.sh* angelegt. Das File beinhaltet folgende Zeilen:

```
# Required for Mac OS X and FreeBSD only (comment out on Linux machines)
# export UMOUNT="umount"

# Not updated to new CLI

HTTP_CLONE="no"

HTTP_PULL="no"

MERGE="no"

MOUNT_WRITE="no"

MOUNT_WRITE_PYTHON_MT="no"
```

Listing 5: Inhalt runtests_config.sh

Die Unit-Tests, die nicht funktionieren, werden übersprungen. Bei Fehlern muss der Ordner *tempdir* sowie die Testrepositories in dem im Home-Verzeichnis versteckten Ordner */.ori/<name>.ori* gelöscht werden. Ein SSH-Public Key muss ebenso angelegt werden, um ein passwortloses Anmelden zu ermöglichen. Die Tests mit der Numer 54 und 61 resultieren in einem Syntaxfehler. Das Package ist nicht auf dem Rechner vorhanden, liegt daher nicht an OriFS. Nach ausgedehntem Troubleshooting und Lesen der Fehlermeldung konnte auch dieser Fehler behoben werden. Die Files im Ordner *ori_ tests*, *53-mount-write-wget.sh* sowie *61-mount-write-wget-mt.sh*, müssen ausgebessert werden. Die jeweils erste Zeile muss auf die aktuell installierte Version von *wget* zeigen. In diesem Fall wurde der String auf *wget-1.16.1* ausgebessert. Erst jetzt funktionieren die Tests ohne Probleme. Die Testfälle 01-05, 11-15, 30, 52-53, 60-61 wurden erfolgreich ausgeführt, 21-22, 35, 51, 62 wurden übersprungen.

2.2 Einsatz / Dokumentation der Ori API

Jeder der angeschriebenen Befehle wurde ausgeführt und dokumentiert. Jeglicher Ausgangspunkt in der Konsole ist das Verzeichnis */home/mweinberger*.

2.2.1 newfs - Anlegen eines Filesystems

Das Erstellen eines Dateisystems ist wie anzunehmen der erste Schritt bei OriFS. Mithilfe des Befehls *ori* wird somit eines erstellt, und im Anschluss mit einem leeren, neu erstellten Verzeichnis eingehängt.

```
ori newfs dezsys

mkdir dezsys

orifs /home/mweinberger/dezsys
```

Listing 6: newfs

2.2.2 removefs - Löschen eines Filesystems

Natürlich lässt sich das Dateisystem auch wieder löschen, ebenso mit dem *ori*-Befehl, es muss lediglich ausgehängt sein. Das vorher erstellte Verzeichnis kann auch gelöscht werden, sofern es nicht mehr benötigt wird.

```
umount /home/mweinberger/dezsys

ori removefs dezsys

rm -rf dezsys
```

Listing 7: removefs

2.2.3 list - Lokale Dateisysteme auflisten

Dieser Befehl listet alle lokalen Ori-Dateisysteme auf. Hier werden Remote-Repositories nicht berücksichtigt.

```
root@ubuntu-mweinberger:/home/mweinberger# ori list

Name                                File System ID
dezsys                              0868b64e-d0ec-4c2d-9230-3ce66f301136
```

Listing 8: list

2.2.4 replicate - Eine lokale Repository-Replik anlegen

Der Befehl *replicate* lässt Ori-Dateisysteme auf die lokale Maschine replizieren. So lässt sich ein Repository in dem Sinne 'klonen'. Langsam wird es möglich, Ori effektiv als verteiltes Dateisystem einzusetzen. Es wird nur das Dateisystem erstellt, das Verzeichnis muss händisch erstellt und gemountet werden. Folgende Befehle sind notwendig:

```
ori replicate root@debian:dezsys_remote

mkdir dezsys_remote
orifs dezsys_remote
```

Listing 9: replicate

2.2.5 snapshot - Einen Repository-Snapshot anlegen

In Ori dient ein Snapshot dazu, Veränderungen im Filesystem zu erkennen und diese zu speichern sowie den Commit den Remote Hosts bereitzustellen, welche das Dateisystem replizieren. Snapshots können auch zur lokalen Versionierung verwendet werden, generell vereint der Befehl *commit & push* in einem. Hier ein Beispiel, wie ein Snapshot nach Veränderungen an Dateien aussieht inkl Commit ID.

```
root@debian:/home/mweinberger/dezsys# ori snapshot EINS
Committed 04c457f7e77dc1831eacc564bb2714a34df5b265d46b4f9e2b697564bc3ad160

root@debian:/home/mweinberger/dezsys# echo "Hallo!" > testfile.txt

root@debian:/home/mweinberger/dezsys# ori snapshot ZWEI
Committed 8eeb2869203d968f6ae8eeb28b2714a34df5b265d46b8eeb2869203d968f6ae3
```

Listing 10: snapshot

Die gespeicherten Zustände der Snapshots sind in Unterverzeichnissen (*.snapshot*) abgelegt, weshalb ein 'Rollback' zu früheren Zuständen sehr einfach ist (herauskopieren). Ohne Veränderungen gibt der Befehl nur ein 'No changes' zurück, der restliche Vorgang wird abgebrochen.

2.2.6 log - Eine Commit-History für das Repository

Wie gewohnt sammelt der Log alle Veränderungen, die vom User ausgegangen sind. Ein Beispiel eines Logs wird hier gezeigt:

```
root@debian:/home/mweinberger/dezsys# ori log

Commit:     04c457f7e77dc1831eacc564bb2714a34df5b265d46b4f9e2b697564bc3ad160
Parents:    8df1610b6299e4bf086eae0cf473c12326ef49aeed6a3e4eed4656e3832660307
Tree:       6e9bad8eeb2869203d968f6aed02c4e79e913f6535094d8f512565bf1b8a1b58
Author:     root
Date:       Fri Apr 12 20:20:31 2016

Created snapshot 'Hallo.txt'.
```

Listing 11: log

2.2.7 tip - Gibt den neuesten Commit aus

Mit diesem Befehl lässt sich die aktuellste Commit ID ermitteln. Er muss jedoch in einem gemounteten Ori-Dateisystem ausgeführt werden.

```
root@debian:/home/mweinberger/dezsys# ori tip

04c457f7e77dc1831eacc564bb2714a34df5b265d46b4f9e2b697564bc3ad160
```

Listing 12: tip

2.2.8 pull/checkout - Änderungen von einem Repository beziehen

Dieser Befehl ist nach Git-Ordnung eher als *Fetch* zu verstehen. Alle (entfernten) Veränderungen müssen auch am lokalen Dateisystem sein. Mit *pull* werden sie lediglich heruntergeladen, zurückgeliefert wird eine Commit ID. Solange bis der *checkout*-Befehl mit genannter ID nicht ausgeführt wurde, sind die Files nicht im Verzeichnis vorhanden.

```
ori pull

ori checkout 04c457f7e77dc1831eacc564bb2714a34df5b265d46b4f9e2b697564bc3ad160
```

Listing 13: pull

2.2.9 show - Repository-Informationen ausgeben

show liefert eine Menge an Informationen über das Repository. Eine mögliche Rückgabe, Struktur immer gleichbleibend:

```
root@debian:/home/mweinberger/dezsys# ori show

—— Repository ——
Root: /root/.ori/dezsys.ori
UUID: dc377ee5-79b5-406b-a844-60a8f480c1d3
Version: ORI1.1
HEAD: 04c457f7e77dc1831eacc564bb2714a34df5b265d46b4f9e2b697564bc3ad160
```

Listing 14: show

2.2.10 status - Suche nach Veränderungen seit letztem Commit

Wieder gibt es sehr große Parallelen zu Git: Der Befehl ist im Grunde genommen ein *git status*, Ori hat jedoch eine andere Struktur. Auch dieser Befehl kann nur in einem gemounteten Ori-Dateisystem ausgeführt werden.

```
root@debian:/home/mweinberger/dezsys# ori status

A   /neues-file.txt
D   /geloeschtes-file.txt
M   /geaendertes-file.txt
```

Listing 15: status

2.2.11 filelog - Anzeigen des Logs für ein spezifisches File

ori filelog ist in dem Sinne nur eine Erweiterung für *ori log*. Auch dieser Befehl kann nur in einem gemounteten Ori-Dateisystem ausgeführt werden. Hier kann ein Filename als Parameter mitgegeben werden, welches im Repository mindestens einmal committed wurde. Nun werden alle Commits ausgegeben, bei denen das angegebene File involviert war. Als Beispiel:

```
root@debian:/home/mweinberger/dezsys# ori filelog dezsys.txt

Commit:     04c457f7e77dc1831eacc564bb2714a34df5b265d46b4f9e2b697564bc3ad160
Parents:    8df1610b6299e4bf086eae0cf473c12326ef49aeed6a3e4eed4656e3832660307
Tree:       6e9bad8eeb2869203d968f6aed02c4e79e913f6535094d8f512565bf1b8a1b58
Author:     root
Date:       Fri Apr 12 20:34:45 2016

Created snapshot 'neuen Text eingefuegt'.
```

Listing 16: filelog

2.2.12 graft - Repository auf ein Anderes kopieren

Ori ermöglicht es auch, ein Repository quasi 'hard' auf das andere zu kopieren. Das Verzeichnis wird repliziert in das Verzeichnis des Ziels. Anzugeben sind lediglich Ausgangsrepository, also wovon kopiert werden soll, und das Zielrepository. Um eine aktuelle Version auf dem Zielrepository zu haben, muss nach jeder Veränderung des Ausgangsrepositorys der *graft*-Befehl ausgeführt werden.

```
ori graft dezsys dezsys-neu
```

Listing 17: graft

2.2.13 varlink - Varlink-Variablen verwalten

Mithilfe des Befehls können die Varlink-Variablen ausgelesen, aufgelistet oder auch neu gesetzt werden. Die Ausgabe auf der Debian-VM ist leider weniger repräsentativ.

```
root@debian:/home/mweinberger/dezsys# ori varlink

Variable        Value
machtype        unknown
osname          unknown
domainname      (none)
hostname        debian
```

Listing 18: varlink

2.2.14 merge - Mergen zweier HEADs

Hier gab es leider große Schwierigkeiten. Entweder wurde der Befehl falsch verwendet (obwohl wie richtig angewiesen mit Commit ID) oder er ist in der derzeitigen Version des Programms noch nicht sehr ausgereift. Es scheint so, als wäre das Dateisystem dadurch beschädigt worden, da auch andere Befehle plötzlich nicht mehr einsatzfähig sind und Fehlermeldungen werfen, und das reproduzierbar. Dieser Befehl wurde daher lieber übersprungen.

```
root@ubuntu-mweinberger:/home/mweinberger/dezsys# ori merge 04
    c457f7e77dc1831eacc564bb2714a34df5b265d46b4f9e2b697564bc3ad160
merge failed with an unknown error!
```

Listing 19: merge

2.2.15 remote - Verwalten von Remoteverbindungen

ori remote gibt es, um bei einer Replikation die Verbindungseinstellungen zu verwalten. Es ist ja beispielsweise gut möglich, dass sich eine IP-Adresse ändert. Umso wichtiger ist es, die Remote-Adresse ändern zu können. Es gibt drei Varianten des Befehls:

```
ori remote

ori remote add name root@10.0.0.3:dezsys

ori remote del name
```

Listing 20: remote

Der *ori remote*-Befehl listet lediglich die derzeit abgespeicherten Verbindungsdaten auf. Mithilfe des zweiten Befehls lässt sich eine Remoteverbindung hinzufügen. Mit dem letzten Befehl wird der Remoteaccess wieder gelöscht.

2.2.16 Orisync

Orisync ist ein Kommando von OriFS, der sich um die automatische Synchronisierung zwischen den Teilnehmern kümmert.

```
orisync init
```

Listing 21: orisync init

Wenn Orisync initialisiert wird, wird gefragt, ob die Maschine die Erste des Clusters ist. Diese Frage wird mit 'y' beantwortet, und der Clustername wird eingegeben. In der Ausgabe steht dann nochmals der Clustername sowie und ein Cluster-Key. Wieso dieser immer den gleichen Wert 't8jhfhkm' hat, bleibt bis heute ungeklärt. Im nächsten Schritt wird wie in den vorderen Punkten beschrieben ein Dateisystem erstellt. Bei der Erstellung wird ein Verzeichnis */.ori/dezsys.ori* erstellt, wobei 'dezsys' austauschbar ist mit dem Namen des Dateisystems. Mit dem folgenden Befehl wird das Filesystem zu orisync geadded, damit es automatisch synchronisiert.

```
orisync add ~/.ori/dezsys.ori

orisync
```

Listing 22: orisync add

Mit der nachherigen Eingabe von *orisync* wird die automatische Synchronisierung gestartet. Zum jetzigen Zeitpunkt sollten alle Files automatisch unter den Clients ausgetauscht werden.
Um einem Cluster beizutreten, muss bei *orisync init* 'n' eingegeben werden. Daraufhin müssen Name und Key eingegeben, und das Filesystem vom Remoteserver repliziert werden. (Wie in Anleitung oben beschrieben) Die Prozedur bei *orisync add / orisync* bleibt hier ebenso gleich.

2.3 Einsatz der Gegenspieler

Im nachfolgenden Kapitel sollen Gegenspieler von OriFS, wie beispielsweise HadoopFS und GlusterFS eingesetzt werden.

2.3.1 HadoopFS [1, 2, 3]

"Apache Hadoop ist ein in Java entwickeltes, quelloffenes Framework, welches das Verarbeiten von großen Datenmengen, verteilt über mehrere Cluster, ermöglicht. Es besteht auch mehreren Modulen, wie Hadoop Yarn und Hadoop MapReduce, die zur Verwaltung von Ressourcen eines Clusters dienen. Das Module Hadoop Distributed File System (HDFS) bietet einen Zugriff mit einem hohen Datendurchsatz auf Applikationsdaten an. Andere Untermodule, wie beispielsweise Cassandra, HBase oder Zookeeper sind ebenfalls Bestandteil dieses Frameworks.

Hadoop wird von großen Firmen für ihre Dienste verwendet. Firmen, wie Yahoo! oder Facebook machen intensiv Gebrauch davon. Yahoo! betreibt 42 000 Nodes in seinem Hadoop Cluster (Stand: Juli 2011), während Facebook mehr, als 100 PB an Daten unter Verwendung von HadoopFS speichert. Andere Firmen, wie Amazon, eBay, LinkedIn, ... machen ebenfalls Gebrauch davon." [12]

2.3.2 Installation and Konfiguration von HDFS

Zunächst müssen Java, SSH und rsync installiert werden. HDFS wurde in Java entwickelt, ein SSH Zugriff auf den HDFS Daemon wird ebenfalls benötigt.

```
apt-get install openssh-server rsync openjdk-8-jre
```

Listing 23: Installieren der benötigten Softwarepakete

Nun kann die Installation von HDFS erfolgen.

```
wget http://mirror.klaus-uwe.me/apache/hadoop/common/hadoop-2.7.2/hadoop-2.7.2.tar.gz
tar xf hadoop-2.7.2.tar.gz
```

Listing 24: Herunterladen und Entpacken von HDFS

Als nächstes muss der Pfad zu Java in der Datei *<hadoop-root>/etc/hadoop/hadoop-env.sh* gesetzt werden. Im Rahmen dieser Übung wird die Version 1.8 Build 77 verwendet.

```
export JAVA_HOME=/usr
```

Listing 25: Setzen des Java Pfades in <hadoop-root>/etc/hadoop/hadoop-env.sh

2.3.3 Ausführen eines Examples

Das Example wird in einem Cluster mit nur einer einzigen Node im Pseudo-Distributed Modus ausgeführt. Hierzu wird der Cluster in der Datei textit<hadoop-root>/etc/hadoop/core-site.xml und textit<hadoop-root>/etc/hadoop/hdfs-site.xml konfiguriert.

```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
    </property>
</configuration>
```

Listing 26: Setzen des Hosts in <hadoop-root>/etc/hadoop/core-site.xml

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
</configuration>
```

Listing 27: Setzen der Anzahl der Replikate in <hadoop-root>/etc/hadoop/hdfs-site.xml

It should be able to connect with SSH to localhost without any passphrase, if this is not possible: An dieser Stell sollte die passwortlose Anmeldung via SSH funktionieren. Sollte dies nicht der Fall sein, können folgende Befehle Abhilfe verschaffen.

```
ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
```

Listing 28: SSH Konfiguration

Nun muss das Dateisystem der Node formatiert werden.

```
<hadoop-root>/bin/hdfs namenode -format
```

Listing 29: Formatieren des Dateisystems

Starten des NameNode und DataNode Daemons:

```
<hadoop-root>/sbin/start-dfs.sh
```

Listing 30: Starten des NameNode und DataNode Daemons

Nun sollte das Aufrufen des Webinterfaces unter *http://localhost:50070/* möglich sein.

Als nächstes erstellen wir ein entsprechendes Verzeichnis, um den MapReduce-Job ausführen zu können.

```
<hadoop-root>/bin/hdfs dfs -mkdir /userin
<hadoop-root>/bin/hdfs dfs -mkdir /userout
```

Listing 31: Erstellen eines Verzeichnisses in dem die Daten gespeichert werden sollen

Nun werden die Daten in das HDFS kopiert.

```
<hadoop-root>/bin/hdfs dfs -put etc/hadoop /userin
```

Listing 32: Kopieren von Files ins HDFS

Ausführen eines Hadoop Examples.

```
<hadoop-root>/bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.7.2.jar grep /userin /
    userout 'dfs[a-z.]+'
```

Listing 33: Ausführen eines Hadoop Examples

Copy the output files from the distributed filesystem to the local filesystem and examine them: Kopieren von Daten aus dem HDFS in das lokale Dateisystem mit

```
<hadoop-root>/bin/hdfs dfs -get /userin output
cat output/*
```

Listing 34: Kopieren von Files aus dem HDFS

Das Einsehen von im Dateisystem gespeicherten Dateien ist über ein Webinterfaces möglich. Das Einsehen des Logs ist sowohl über die Konsole, aber auch unter *http://localhost:50070/logs/* möglich.

2.3.4 GlusterFS

GlusterFS ist ein von RedHat entwickeltes quelloffenes verteiltes Dateisystem, welches unter der GNU GPL und anderen Open Source Lizenzen veröffentlicht wird. Es können Dateien im Größenbereich von mehreren Petabyte gespeichert, die von mehreren Tausend Clients verwendet werden können. Es wird ein globaler Namespace verwendet, die Daten werden über storage blocks, welche über TCP/IP oder Infiband RDMA erreichbar sind, bereitgestellt.

Daten, welche in solchen Speicherblöcken abgelegt werden, können vom GlusterFS Server verwaltet und über den GlusterFS Client oder CIFS verwendet werden.

Es ist nicht möglich Dateien zwischen Server Nodes teilen zu lassen. Daten werden hier von mehreren Clients innerhalb des Dateisystems hinzugefügt, verändert und gelöscht. Im Normalfall werden Dateien als Ganzes auf jedem Server Node gespeichert. Die Verwendung von Striping (Dateien werden auf mehrere Teile auf mehrere Nodes aufgeteilt) ist ebenso möglich. Verzeichnisse werden vollständig exportiert, Daten müssen seitens des Clients restrukturiert werden. [13, 14]

2.3.5 Installation und Konfiguration von GlusterFS [4, 5]

Zunächst muss der GlusterFS Server auf beiden Serverinstanzen installiert werden. Hier werden zwei Ubuntu 15.10 Server verwendet.

```
apt-get install glusterfs-server
```

Listing 35: Installation des GlusterFS Servers

Beide Server enthalten einen entsprechenden Eintrag in */etc/hosts*. Dieser Vorgang muss auf beiden Servern durchgeführt werden.

```
192.168.130.128 gluster1.example.com    gluster1
192.168.130.129 gluster2.example.com    gluster2
```

Listing 36: Vorbereiten von /etc/hosts

Beide Server werden als ein Teilnehmer zu dem Cluster hinzugefügt.

```
gluster1> gluster peer probe gluster2
gluster2> gluster peer probe gluster1
```

Listing 37: Hinzufügen der Server zum Cluster

Nun wird der Status der Teilnehmer überprüft.

```
gluster1> gluster peer status

Number of Peers: 1
Hostname: gluster2
Uuid: 1c8c2dfe-fb62-49a9-9e45-a749f3b3c0bf
State: Peer in Cluster (Connected)
```

Listing 38: Überprüfen des Status der Teilnehmer

In diesem Schritt wird ein Verzeichnis angelegt, welches zur Replikation der Clientdaten verwendet werden soll. Dieser Schritt muss auf beiden Servern ausgeführt werden.

```
mkdir -p /mnt/gluster
```

Listing 39: Erstellen eines Replikationsverzeichnisses

Erstellen eines Volumes mit dem Namen *data* und Verbinden mit dem zuvor erstellten Verzeichnis.

```
gluster1> gluster volume create data replica 2 gluster1:/mnt/gluster gluster2:/mnt/gluster force
gluster1> gluster volume start data

volume start: data: success
```

Listing 40: Erstellen des data Volumes

Installation des GlusterFS Clients auf einem beliebigen Debian Testing Client.

```
1 apt-get install glusterfs-client
```

Listing 41: Installation des GlusterFS Clients

Erstellen eines Verzeichnisses, in dem die Daten später auf den Server übertragen werden sollen.

```
mkdir -p /mnt/gluster
```

Listing 42: Erstellen einer Verzeichnisses

Einbinden des zu replizierenden Volumes.

```
mount -t glusterfs 192.168.188.41:/data /mnt/gluster
```

Listing 43: Einbinden des zu replizierenden Volumes

Daraufhin erscheint folgende Fehlermeldung.

```
Mount failed. Please check the log file for more details.

[2015-03-19 16:51:04.426710] I [fuse-bridge.c:3724:fuse_init] 0-glusterfs-fuse: FUSE inited with
    protocol versions: glusterfs 7.13 kernel 7.22
4 [2015-03-19 16:51:04.426878] W [fuse-bridge.c:705:fuse_attr_cbk] 0-glusterfs-fuse: 2: LOOKUP() / => -1 (
    No such file or directory)
[2015-03-19 16:51:04.492092] I [fuse-bridge.c:4628:fuse_thread_proc] 0-fuse: unmounting /mnt/gluster2
[2015-03-19 16:51:04.492589] W [glusterfsd.c:1002:cleanup_and_exit] (-->/lib/x86_64-linux-gnu/libc.so.6(
    clone+0x6d) [0x7fa6a045647d] (-->/lib/x86_64-linux-gnu/libpthread.so.0(+0x8182) [0x7fa6a0729182]
    (-->/usr/sbin/glusterfs(glusterfs_sigwaiter+0xd5) [0x7fa6a1212ef5]))) 0-: received signum (15),
    shutting down
[2015-03-19 16:51:04.492606] I [fuse-bridge.c:5260:fini] 0-fuse: Unmounting '/mnt/gluster2'.
```

Listing 44: Fehler der beim Einbinden des Verzeichnisses auftritt

Da keine Lösung für dieses konkrete Problem gefunden werden konnte, wird gluster1 als Client verwendet und ein neuer Ordner zum Testen angelegt.

```
mkdir -p /mnt/gluster-client
```

Listing 45: Erstellen eines Verzeichnisses welches repliziert wird

2.3.6 Ausführen eines eigenen Examples

Einbinden des zu replizierenden Volumes.

```
mount -t glusterfs gluster1:/data /mnt/gluster-client
```

Listing 46: Einbinden des zu replizierenden Volumes

Nun wird der Inhalt von gluster und gluster-client auf gluster1 überprüft.

```
gluster1> ls -la /mnt/gluster
total 20
drwxr-xr-x 4 root root 4096 Apr 22 03:14 .
drwxr-xr-x 4 root root 4096 Apr 22 03:27 ..
drw------- 6 root root 4096 Apr 22 03:15 .glusterfs
drwxr-xr-x 3 root root 4096 Apr 22 03:14 .trashcan

gluster1> ls -la /mnt/gluster-client/
total 12
drwxr-xr-x 4 root root 4096 Apr 22 03:14 .
drwxr-xr-x 4 root root 4096 Apr 22 03:27 ..
drwxr-xr-x 3 root root 4096 Apr 22 03:14 .trashcan
```

Listing 47: Überprüfen des zu replizierenden und des replizierten Volumes

Abschließend wird eine Datei angelegt und wieder der Inhalt der Verzeichnisse überprüft.

```
gluster1> touch /mnt/gluster-client/test.text

gluster1> ls -la /mnt/gluster-client/
total 13
drwxr-xr-x 4 root root 4096 Apr 22 03:42 .
drwxr-xr-x 4 root root 4096 Apr 22 03:27 ..
-rw-r--r-- 1 root root   10 Apr 22 03:42 test.text
drwxr-xr-x 3 root root 4096 Apr 22 03:14 .trashcan

gluster1> ls -la /mnt/gluster
total 28
drwxr-xr-x 4 root root 4096 Apr 22 03:42 .
drwxr-xr-x 4 root root 4096 Apr 22 03:27 ..
drw------- 9 root root 4096 Apr 22 03:42 .glusterfs
-rw-r--r-- 2 root root   10 Apr 22 03:42 test.text
drwxr-xr-x 3 root root 4096 Apr 22 03:14 .trashcan

gluster2> ls -la /mnt/gluster
total 28
drwxr-xr-x 4 root root 4096 Apr 22 03:42 .
drwxr-xr-x 4 root root 4096 Apr 22 03:27 ..
drw------- 9 root root 4096 Apr 22 03:42 .glusterfs
-rw-r--r-- 2 root root   10 Apr 22 03:42 test.text
drwxr-xr-x 3 root root 4096 Apr 22 03:14 .trashcan
```

Listing 48: Überprüfen des zu replizierenden und des replizierten Volumes

2.4 Gegenüberstellung

Kriterium	**OriFS**	**HDFS**	**GlusterFS**
Dokumentation	wenig	gut	gut
Usability	schlecht	mäßig	schlecht
Community	wenig	aktiv	mäßig aktiv
Fehlertoleranz	gut	gut	gut
Hochverfügbarkeit	gut	gut	mäßig
Replikation	gut	gut	gut
Skalierbarkeit	gut	gut	gut
Ranking	2.	1.	2.

Die Dokumentation der Befehle für OriFS ist in nur mäßigem Umfang enthalten. Die Dokumentation für HDFS und GlusterFS ist vergleichsweise sehr gut geschrieben, da für viele Anwendungsfälle auch kleine Tutorials vorhanden, die sich im Normalfall auch in der Form umsetzen lassen.

Nur ca. die Hälfte der Testfälle für OriFS funktionieren und, selbst, wenn sie funktionieren, dann sind sie stellenweise nicht unbedingt hilfreich. Die Examples für HDFS ließen sich problemlos ausführen. GlusterFS hat schlussendlich auch sehr gut funktioniert, trotz des kleinen Zwischenfalls.

OriFS ist ein neues verteiltes Dateisystem, welches aktuell noch in Entwicklung ist. Infolgedessen ist die Community wahrscheinlich klein. Manche Befehle, wie *merge* und manchmal auch *orisync* wurden noch nicht ausreichend getestet und können einen dauerhaften Schaden am Repository verursachen. Manche Befehle, wie *pull* sollten die Daten herunterladen und anwenden, wie bei git. Stattdessen werden sie aber nur heruntergeladen, was für Verwirrung sorgen kann. Aufgrund dieser Defizite fällt das Fazit für OriFS nur mäßig aus.

Literaturverzeichnis

[1] Hadoop mapreduce next generation - setting up a single node cluster. http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/SingleCluster.html. Zuletzt besucht: 22.04.2016.

[2] Hdfs users guide. http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html. Zuletzt besucht: 22.04.2016.

[3] Hdfs commands. http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HDFSCommands.html. Zuletzt besucht: 22.04.2016.

[4] How to install glusterfs with a replicated volume over 2 nodes on ubuntu 15.10. https://www.howtoforge.com/how-to-install-glusterfs-with-a-replicated-volume-over-2-nodes-on-ubuntu-15.10. Zuletzt besucht: 22.04.2016.

[5] Accessing data - setting up glusterfs client. http://www.gluster.org/wp-content/uploads/2012/05/Gluster_File_System-3.3.0-Administration_Guide-en-US.pdf. Zuletzt besucht: 22.04.2016.

[6] Ori file system, stanford website. http://ori.scs.stanford.edu/. Zuletzt besucht: 01.04.2016.

[7] Yifeng Frang Huang David Mazières Ali José Mashtizadeh, Andrea Bittau. History, and Grafting in the Ori File System. Technical report, Standford University, 11 2013.

[8] Ori file system, bitbucket wiki. https://bitbucket.org/orifs/ori/wiki/Home. Zuletzt besucht: 01.04.2016.

[9] Apache hadoop filesystem. http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html. Zuletzt besucht: 01.04.2016.

[10] Glusterfs. http://gluster.readthedocs.org/en/latest/. Zuletzt besucht: 01.04.2016.

[11] Dezsys13 - repository. https://github.com/tgm-ttaschner/DezSys-13-Verteiltes-Dateisystem-mit-OriFS-.git. Zuletzt besucht: 21.04.2016.

[12] Testing of several distributed file-systems (hdfs, ceph and glusterfs) for supporting the hep experiments analysis. http://iopscience.iop.org/1742-6596/513/4/042014/pdf/1742-6596_513_4_042014.pdf. Zuletzt besucht: 22.04.2016.

[13] About. http://www.gluster.org/documentation/About_Gluster/. Zuletzt besucht: 22.04.2016.

[14] Glusterfs. http://en.wikipedia.org/wiki/GlusterFS. Zuletzt besucht: 22.04.2016.

Listings

