Datenbearbeitung in der Cloud anhand von Apache Hadoop
Hochschule Mannheim

Tobias Neef

Fakultät für Informatik
Hochschule Mannheim
tobnee@gmail.com

18.12.2009

Trends

- Die globalen Datenmengen wachsen
  - Jeder Teilnehmer trägt zum Datenwachstum bei
  - Gesetzmäßigkeiten wie in Moore's Law zeigen sich auch in der Steigerung des Datenverkehrs[1]
- Daten werden stärker zueinander in Verbindung gebracht
  - SocialGraphs (Facebook, Xing, etc.)
  - Suchindices
  - Mashups

[1] Handbook Of Massive Datasets

Anforderungen an Datenverarbeitung der Neuzeit

- Skalierbarkeit
- Verteilbarkeit
- Parallelität

Applikationen auf Basis von relationalen Datenbanken

Die meisten Applikationen werden heute auf Basis von RDS entwickelt.

- Vorteile
  - Datenkonsistenz
  - Speichereffizienz
  - Geringe Einstiegshürde
- Nachteile
  - Skaliert nicht gut über viele Systeme
  - Langsam, wenn viele Daten in Relation gebracht werden müssen (Joins)

Lösung: Hadoop?

- Hadoop ist ein TopLevel Apache Projekt
- Der Core besteht aus:
  - HDFS - Ein verteiltes Dateisystem
  - MapReduce - Ein Programmiermodell für verteilte Systeme
- Auf dem Core setzen diverse andere Projekte auf
- Hadoop ist in Java implementiert

Geschichte

- Unter dem Namen Nutch gestartet als Subprojekt von Apache Lucene mit dem Ziel eine freie Websearchengine zu entwickeln
- Erste Version 2002
  - Nicht skalierbar über Milliarden von Webseiten
- Aktuelle Forschungsergebnisse geben dem Projekte eine neue Richtung
  - Google veröffentlicht ein Paper zum Google File System in 2003
  - Google veröffentlicht ein Paper zu MapReduce in 2004
- Hadoop wird 2006 aus Lucene/Nutch ausgegliedert und wird ein eigenständiges Projekt
- In 2008 gewinnt Hadoop erstmals den Terabyte Sort Benchmark[2] (209 sec / 910 nodes)
- Seit 2008 Yahoo baut seinen Internetindex mit Hadoop

[2] `http://sortbenchmark.org`

Designentscheidung

- Hardwarefehler sind die Regel / Einsatz von Commodity Hardware[3]
- Große Dateien sind die primäre Berechnungsbasis
- Optimierung auf lesenden Dateizugriff
- Netzwerkbandbreite ist der limitierende Faktor

[3] `http://labs.google.com/papers/gfs.html`

HDFS Übersicht

- Ein verteiltes, fehlertolerantes Dateisystem
- OpenSource-Implementierung des Google File Systems
- Das Standarddateisystem von Hadoop
- Implementiert in Java aber zugänglich über diverse Schnittstellen

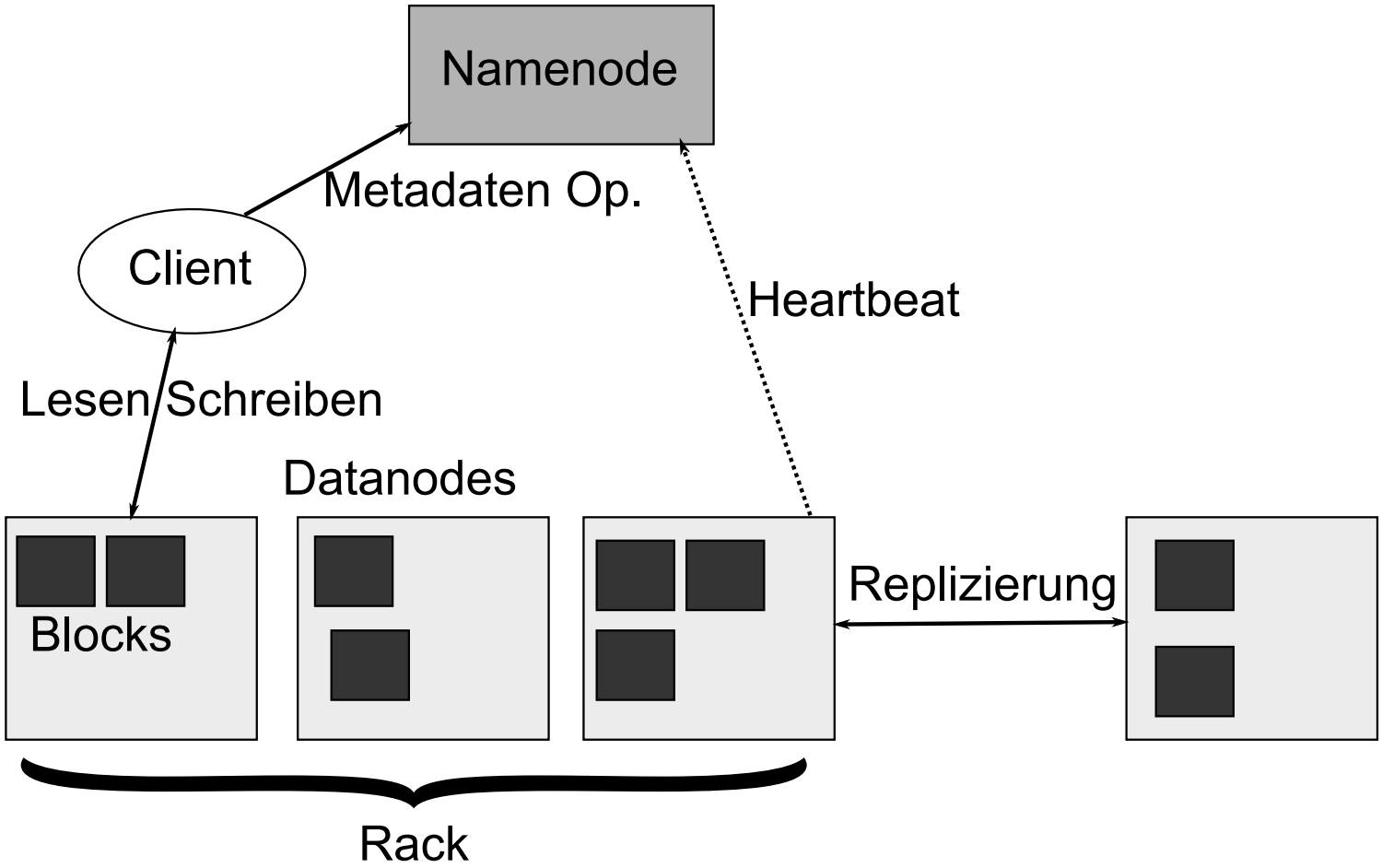
HDFS Architektur: Blocks

- Blocks sind die kleinste Speichereinheit in einem HDFS-Cluster
- Typischerweise sind Blocks 64MB groß
- Die Blockabstraktion hält das Speichersystem einfach
  - Nur der gesamt verfügbare Speicher im Cluster begrenzt die maximale Dateigröße
  - Datensicherheit durch Verteilung der Blocks übers Netzwerk
  - Redundante Vorhaltung von Blocks kompensiert Ausfälle von Nodes und erhöht die Transferraten

HDFS Architektur: Nodes

- Nodes sind in HDFS die Abstraktion eines oder mehreren physikalischen Computern
- Es wird unterschieden zwischen Namenodes und Datanodes
- Der Namenode:
  - Existiert nur einmal in einem HDFS-Cluster
  - Verantwortlich für die Verwaltung von Metainformationen wie Dateinamen oder Berechtigungen
  - Weiß über die Lage der Blocks eines Files Bescheid
  - Fällt der Namenode aus, ist das Dateisystem nicht mehr zugänglich
- Der Datanode:
  - Ist für die eigentliche Datenübertragung verantwortlich
  - Melden sich in einem regelmäßigen Rhythmus (Heartbeat) bei ihrem Namenode mit Informationen über die gespeicherten Blöcke

HDFS Architektur: Übersicht



HDFS Interfaces

- CLI
- Programmatisch
  - Native Java-API
  - Funktional nachstehende C-API
  - Bindings für andere Sprachen
- Dateisystemmapping
  - Kein Linux-Treiber (da nicht voll POSIX-Kompatibel)
  - FUSE-Implementierung
  - WebDav-Schnittstelle

MapReduce Übersicht

- Ein Programmiermodell für die verteilte Datenverarbeitung
- Idee wurde bei Google entwickelt um ihre Datenverarbeitungsaufgaben zu vereinheitlichen
- Benutzer spezifiziert über eine Map- und eine Reducefunktion seine Ablauflogik
- Ist effizient durch Pipelining und streamorientiertes Arbeiten
- Primär ausgerichtet für Stapelverarbeitungsaufgaben
  - Aufbau eines Indexes
  - Sotieren eines großen Datenbestandes
  - Spamfilterung

MapReduce Programmierbeispiel

Beispiel

Zählen der Pagevisits über die Logs aller Webserver in einem Cluster

MapReduce Programmierbeispiel

Eingangsdaten

```
66.249.64.13 - - [18/Sep/2004:11:07:48 +1000]
"GET /robots.txt HTTP/1.0" 200
"Googlebot/2.1"
66.249.64.13 - - [19/Sep/2004:11:07:48 +1000]
"GET /about HTTP/1.0" 200
"Googlebot/2.1
66.249.64.13 - - [19/Sep/2004:11:07:50 +1000]
"GET /web HTTP/1.0" 200
"Googlebot/2.1
66.249.64.13 - - [20/Sep/2004:13:01:00 +1000]
"GET /robots.txt HTTP/1.0" 200
"Googlebot/2.1
```

MapReduce Programmierbeispiel

Map-Funktion

```
map (String schluessel, String wert):
  foreach line z in wert:
    url = getURL(z)
    sammle(url,1)
```

Zwischendaten nach Map-Durchlauf

```
/robots.txt 1
/about 1
/web 1
/robots.txt 1
```

MapReduce Programmierbeispiel

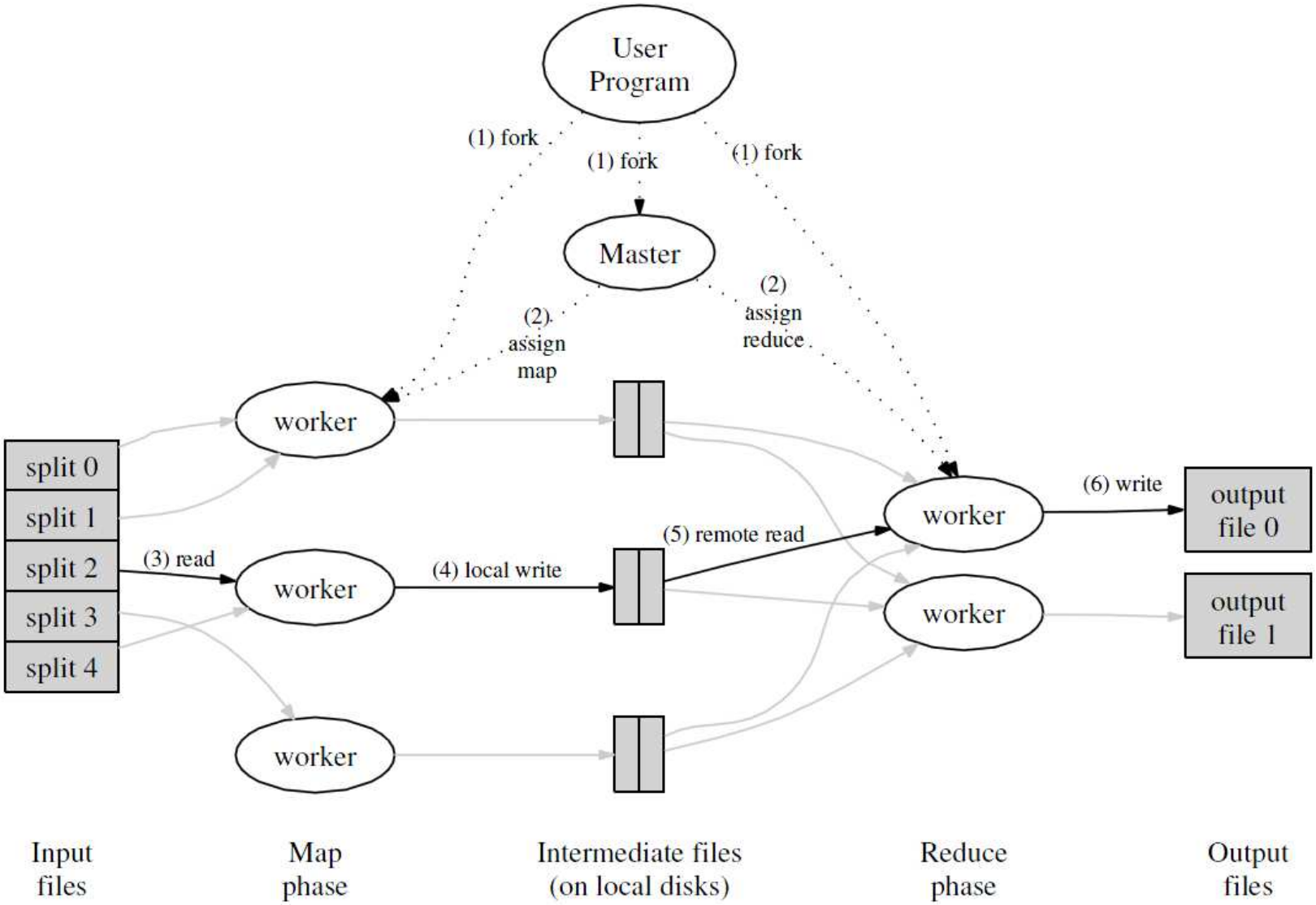
Reduce-Funktion

```
reduce (String schluessel, Iterator werte)
  foreach value v in werte:
    anzahl += v
  sammle(anzahl)
```

Enddaten nach drei Reduce-Durchläufen

```
/robots.txt 2
/about 1
/web 1
```

MapReduce Ablaufbeschreibung



ZooKeeper

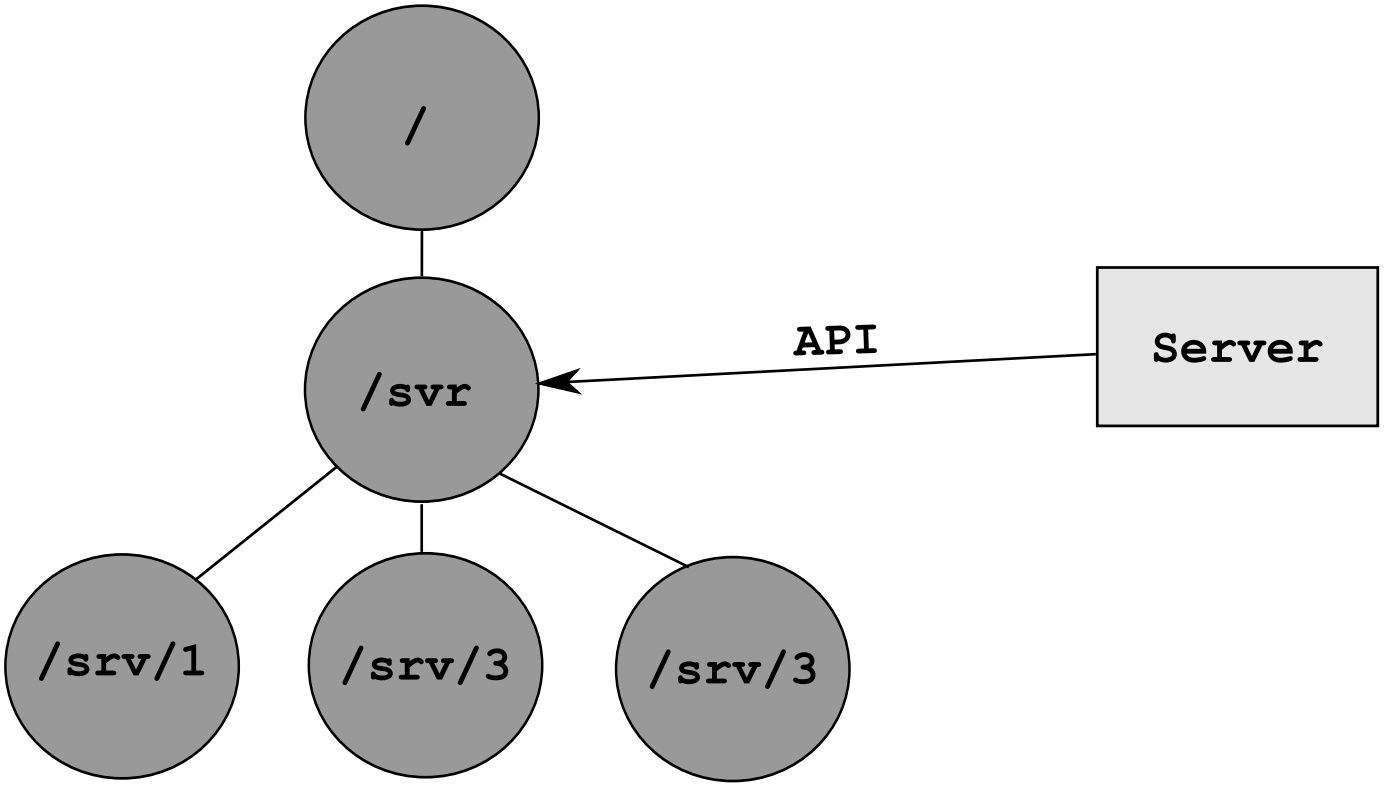
- Koordinationssystem für verteilte Anwendungen
- Ein reduziertes Dateisystem mit einfachen Operationen
- Eingebaute Datenstrukturen für die Koordination von Anwendungen[4]
  - Verteilte Queues
  - Verteilte Locks
  - Auswahl eines Master unter einer Gruppe von gleichgestellten
- ZooKepper ist hoch verfügbar
- Es unterstützt die Entwicklung lose gekoppelter Systeme
- ZooKeeper bietet eine Bibliothek für die Implementierung von gebräuchlichen Koordinationspatterns

[4] Tom White - Hadoop

ZooKeeper

Beispiel

Es soll den Clients eine Liste von Servern bereitgestellt werden, welche Dienste anbieten.



Pig

MapReduce-Probleme

- Die direkte MapReduce Nutzung ist nicht immer unproblematisch
  - Oft werden diverse MapReduce Stages benötigt um eine Aufgabe umzusetzen
  - MapReduce-Definitionen teilweise weit weg von der Problemstellung
  - Transformationen wie Joins sind komplex umzusetzen

- Pig ist ein Lösungsansatz für diese Probleme
  - Pig ist eine Datenflusssprache aus dem Hadoop-Projekt
  - Setzt auf einem höheren Abstraktionslevel auf wie MapReduce
  - Pig kompiliert in MapReduce Tasks
  - Pig bietet eine hohe Wiedernutzbarkeit durch user-defined-functions (UDF)
  - Pig-Abfragen sind im Schnitt etwas langsamer als hand-optimierte MapReduce-Abfragen

Pig

Beispiel

Finde die Wochen, in denen es in einem Wikisystem mehr als 1000 neue Artikel gab.

Pig-Programm

```
table = LOAD 'stats' USING PigStorage('\t')
        AS (week, creations);
heavy_week = FILTER table BY creations > '1000';
```

Ergebnis

```
(10/16/2002, 1102), (6/28/2006, 1005),
(8/23/2006, 1037), (7/18/2007, 1164)
```

Offene Frage

Wo und ob passt Hadoop in die Cloud?