Einführung in Hadoop

- Inhalt / Lern-Ziele
  - Übersicht: Basis-Architektur von Hadoop
  - Einführung in HDFS
  - Einführung in MapReduce
  - Ausblick:
    - Hadoop Ökosystem
    - Optimierungen
    - Versionen

Übersicht: Hadoop

Was ist Hadoop?

- Software Framework für die Entwicklung verteilter Anwendungen
  - Speicherung (Storage)
  - Berechnungen (Computation)
- Apache Top Level Projekt
  - Apache Lizenz
- Entwickelt in Java
- Produktions-Plattform: Linux

Kern von Hadoop

- Verteiltes Dateisystem
  - HDFS (Hadoop Distributed File System)
  - Inspiriert von Google File System (DFS)
    - Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. 2003. The Google file system. SIGOPS Oper. Syst. Rev. 37, 5 (October 2003), 29-43
- Datenverarbeitungs Modell: MapReduce
  - Inspiriert von Google
    - Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51, 1 (January 2008), 107-113.

Ziele

- Skalierbar
  - durch Hinzufügen von Nodes
- Einfach
  - Entwicklung verteilter Anwendungen
- Robust
  - Fehlertoleranz
- Zugäglichkeit (Accessible)
  - Standardhardware oder Cloud Computing Services

Architektur von Hadoop

- Verteilte Master-Slave Architektur für
  - Storage via Data Partitioning: HDFS
  - Parallel Computation: MapReduce

HDFS

HDFS: Ziele und Annahmen

- Streaming Data Access
- Hardware Failure
- Large Data Sets
- Simple Coherency Model
- “Moving Computation is Cheaper than Moving Data”
- Portability Across Heterogeneous Hardware and Software Platforms

HDFS - Eigenschaften

- *data locality*
- *large block size*
- *data replication*
- *fault tolerance* gegenüber Software und Hardware Fehler

Name Nodes

- Single Master
- „Volume Manager“ von HDFS
  - In-Memory mapping der Files zu den Blocks
  - d.h. für jeden Block:
    - auf welchen Nodes ist dieser verfügbar
  - Clients fragen Name Node ab, wenn sie *File System Operations* ausführen wollen, dann direkter Datentransfer zwischen Client und Datanode
- Eigentlichen Daten laufen nie über NameNodes
- Single Point of Failure - außer in “HDFS High-Availability”

Data Nodes

- Verantwortlich für das Speichern der File Blocks
- Pipeline Write möglich
- Kommunikation mit Name Node

HDFS Client

- Anwendung, die eine File System Aktivität ausführen will, wie File schreiben, löschen etc.
- Anfrage an Name Node um Informationen zu erhalten
- direkte Interaktion mit Data Nodes, z.B. für Datei-Lesen und -Schreiben

File System Namespace

- Hierarchisch, wie in anderen Dateisystemen
- Bisher: Keine Hard- oder Softlinks
- Verantwortlich: Name Node

Data Replication - Ziele

- Reliability (Ausfallsicherheit)
- Availability (Verfügbarkeit)
- Performance (Performanz)

Data Replication

- Files bestehen aus Block-Sequenzen
- Blocks werden repliziert
- Replication Factor: Anzahl der Kopien eines Files
- Files sind write-ones
- Name Nodes entscheiden über die Replikation der Blocks
- Name Nodes erhalten Heartbeats und Blockreports von jedem Data Node

Data Replication (cont.)

- Rack Awareness
  - Rechner innerhalb eines Racks haben schnellere Netzverbindung – Hadoop kann dies bei der Replikation berücksichtigen
- Hardware für Hadoop
  - Lokale Festplatten pro Server statt SAN oder NAS
  - Raid-Systeme nicht nötig – ehr hinderlich

Andere Dateisystem mit Hadoop

- Amazon S3
- CloudStore (ex: Kosmos Distributed File Systems)
- FTP Filesystem
- Read-Only HTTP und HTTPS

MapReduce

Eigenschaft von MapReduce

- Programmiermodell für Datenverarbeitung
- Inhärent parallel
- shared nothing
- Zwei Phasen
  - Map-Phase
  - Reduce-Phase
- Algorithmen typischerweise umgesetzt als Sequenz von MapReduce Operationen

Hadoop MapReduce Architektur

- JobTracker: Master der Job von Clients entgegennimmt und überwacht
- TaskTracker: Dämonenprozess, der Kind Prozesse erzeugt, die Teile des MapReduce Jobs ausführen.

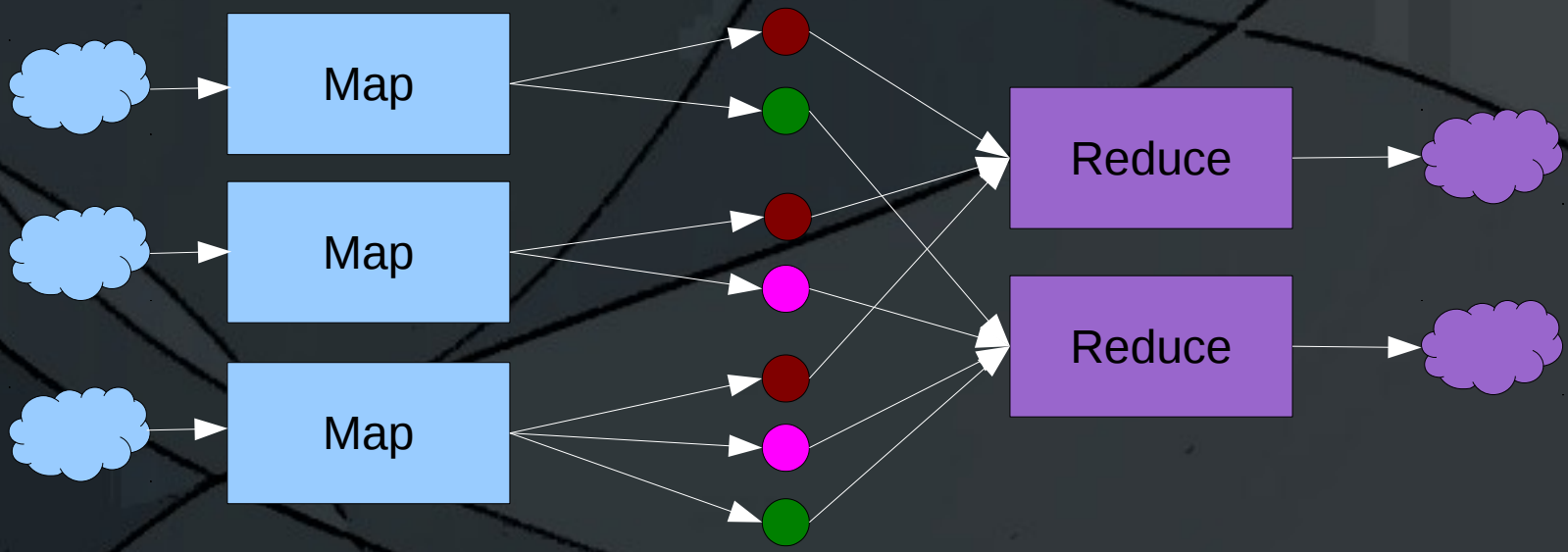
Terminologie

- *Job*: "Arbeitsauftrag" des Clients auf der vollen Datenmenge
- *Task*: Teile des Jobs - auf Teil der Daten
  - *map tasks*
  - *reduce tasks*
- Input wird in Teile fester Größe unterteilt
  - *input splits* – *splits*
  - Ein *map task* für jeden *split*

Key-Value

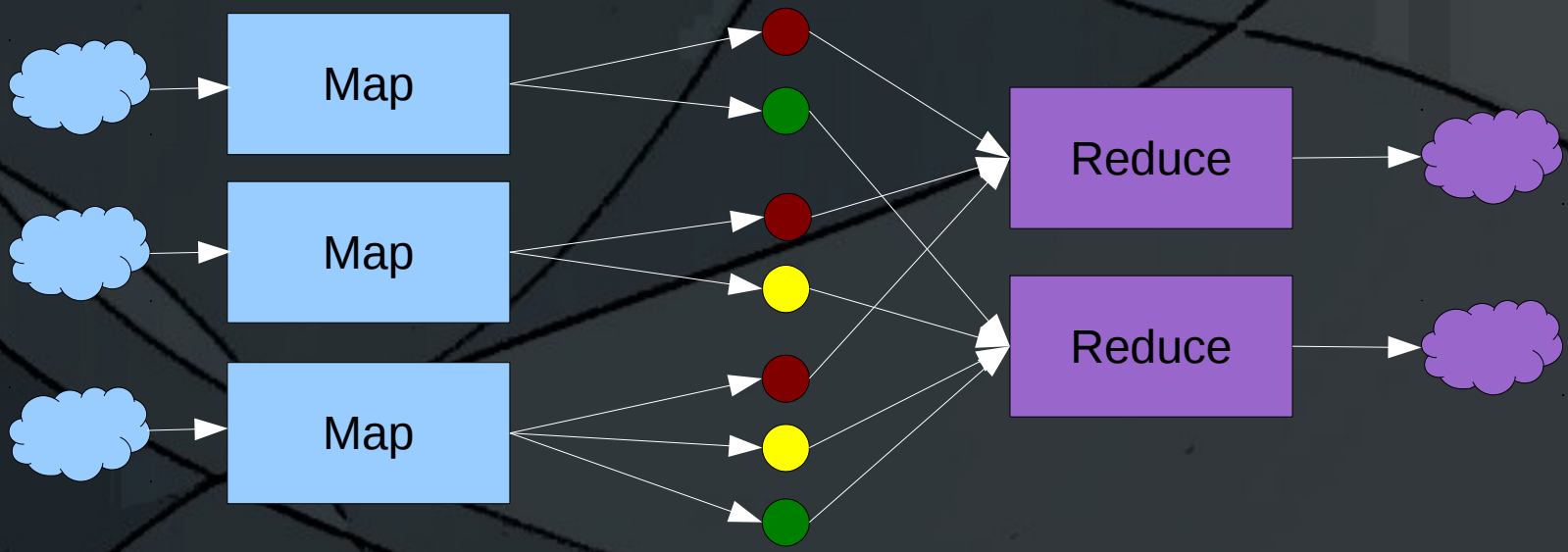
	Input	Output
map	<k1, v1>	list(<k2,v2>)
reduce	<k2,list(v2)>	list(<k3, v3>)

MapReduce



<k1,v1> list<k2,v2> <k2,list(v2)> list<k3,v3>

Wordcount mit MapReduce



Shuffle + Sort

<1, rot grün gelb>
<2, gelb rot rot >….

<rot, 1>
<grün,1>
<gelb,1>

<gelb,1>
..

<rot, (1,1,..)>
<grün,(1,1,..)>
<gelb,(1,1,...>
..

<rot, 101>
<grün,77>
<gelb,98>
..

Ausblick

Weitere Bestandteile

- **Combiner** zur Reduzierung der vom Netzwerk übertragenen Datenmenge
- **Partitioner** zum Aufteilen des Map-Outputs auf die Reducer
- **Suffle&Sort** zwischen Map und Reduce

Hadoop Ökosystem

- Hbase: Spaltendatenbank (Vorbild Googles BigTable)
- Pig: Dataflow Language
- Hive: Data Warehouse (SQL ähnliche Syntax)
- Zookeeper: High availability coordination service
- RHIPE: R - Hadoop Integrated Processing Environment
- Mahout: Distributed Machine Learning
- Sqoop: “SQL-to-Hadoop”
- Oozie (Workflow Engine)

Literatur

- C. Lam, J. Warren, „Green Paper from Hadoop in Action“, 2009 http://manning.com/lam/
- A. Holmes: „What is Hadoop“ („Hadoop in Practice“), 2012 http://www.manning.com/holmes/
- Hadoop Wiki http://wiki.apache.org/hadoop/
- Hadoop Dokumentation http://hadoop.apache.org/common/docs/current/index.html