

 FernUniversität in Hagen

-

Seminar 01912 / 19912  
im Sommersemester 2017

„Skallierbare verteilte Datenanalyse“

Thema 2.3

Spark

Referent: Lukas Wappler

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>3</b>
<b>2</b>	<b>Apache Spark</b>	<b>4</b>
2.1	Kern-Bibliotheken / Komponenten . . . . .	4
2.1.1	Grundlage des Systems (Spark-Core & RDD's) . . . . .	4
2.1.2	SQL-Abfragen mit (Spark-SQL & Data Frames) . . . . .	4
2.1.3	Verarbeitung von Datenströmen (Spark-Streaming) . . . . .	5
2.1.4	Berechnungen auf Graphen (GraphX) . . . . .	5
2.1.5	Maschinelles Lernen (MLlib) . . . . .	5
2.1.6	Skalierung von R Programmen (SparkR) . . . . .	5
2.2	Mehrere Komponenten im Verbund . . . . .	6
2.3	Performance . . . . .	7
2.3.1	Besonderheiten bei der Speichernutzung . . . . .	7
2.3.2	Netzwerk und I/O-Traffic . . . . .	7
2.4	Nutzung & Verbreitung . . . . .	8
<b>3</b>	<b>Fazit</b>	<b>9</b>
<b>4</b>	<b>Ausblick &amp; Weiterentwicklung</b>	<b>10</b>
<b>5</b>	<b>Anhang</b>	<b>11</b>
<b>6</b>	<b>Literaturverzeichnis</b>	<b>12</b>

# 1 Einleitung

## 2 Apache Spark

Apache Spark wurde ist ein Open Source Framework, dass ermöglicht Software verteilt über ein Cluster auszuführen. Darüber hinaus ist die Programmier-Modell mit Apache Spark sehr elegant und einfach gehalten. [Owen und Wills 2015]

Im Rahmen eine Forschungsprojekts ist Apache Spark entstanden. Das Forschungsprojekt wurde 2009 in der Universtiy of California in Berkeley im sogenannten AMPLab<sup>1</sup> ins Leben gerufen. Seit 2010 steht es als Open Source Software unter der BSD-Lizenz <sup>2</sup> zur Verfügung. Das Projekt wird seit 2013 von der Apache Software Foundation<sup>3</sup> weitergeführt. Seit 2014 ist es dort als Top Level Projekt eingestuft. Zum aktuellen Zeitpunkt steht Apache Spark unter der Apache 2.0 Lizenz<sup>4</sup> zur Verfügung.

Der Code liegt auf GitHub<sup>5</sup> und ist für jeden zugänglich. Bis zum 10.04.2017 gab es bereits 51 Releases, 19,365 commits und 1,053 contributors<sup>6</sup>.

### 2.1 Kern-Bibliotheken / Komponenten

Apache Spark besteht im wesentlichen aus fünf Modulen: Spark Core, Spark SQL, Spark Streaming, MLlib Machine Learning Library und GraphX. Die Module werden in den folgenden Kapitel näher beleuchtet. Darüber hinaus wird in Kapitel 2.1.6 noch eine weiteres interessantes Modul vorgestellt welches nicht direkt zum Kern gehört, SparkR. Die Aufteilung in die verschiedenen Module macht er sehr gut möglich nur einen Teil der Module zu verwenden.

#### 2.1.1 Grundlage des Systems (Spark-Core & RDD's)

---

<sup>1</sup>AMPLab: Todo

<sup>2</sup>BSD-Lizenz (Berkeley Software Distribution-Lizenz): bezeichnet eine Gruppe von Lizenzen, die eine breitere Wiederverwertung erlaubt.

<sup>3</sup>Apache Software Foundation: TODO

<sup>4</sup>Apache 2.0 Lizenz: Die Software darf frei verwendet und verändert werden. Zusätzlich gibt es nur wenige Auflagen.

<sup>5</sup>GitHub: Todo

<sup>6</sup>contributors: Todo

### 2.1.2 SQL-Abfragen mit (Spark-SQL & Data Frames)

Spark-SQL wurde 2014 veröffentlicht. Die Komponente gehört zu den Komponenten aus der Spark-Familie, die am meisten weiterentwickelt werden.

Dabei kombiniert es zwei wesentliche Dinge. Zum einen ermöglicht es relationale Querys zu schreiben und zum anderen prozedurale Algorithmen einzusetzen. Bisher wurden beide Aktionen nacheinander von verschiedenen Systemen realisiert.

SparkSQL entstammt dem Apache-Shark. Man wollte die Probleme die es in Apache Shark gab lösen.

1. Mit Apache Shark ist es nur möglich auf Daten im Hive Katalog zuzugreifen.
2. Shark lässt sich nur über selbst geschriebene SQL's aufrufen.
3. Hive ist nur für MapReduce optimiert

Mit ApacheSQL hat man erreicht auf relationale Daten zuzugreifen. Es wurde eine hohe Performance aufgrund etablierter DBMS-Techniken erreicht. Neue Datenquellen lassen sich leicht anschließen und integrieren. Zusätzliche Erweiterungen wie Maschine Learning und Graph Processing sind nutzbar.

Todo, auf Dataframe API eingehen. Todo, auf Catalyst eingehen.

Gerade bei SQL ist es enorm wichtig sich für die richtigen Anweisungen zu entscheiden um keine langsamen Operationen zu haben. Hier gibt es sehr große Geschwindigkeitsunterschiede.

**2.1.3 Verarbeitung von Datenströmen (Spark-Streaming)**

**2.1.4 Berechnungen auf Graphen (GraphX)**

**2.1.5 Maschinelles Lernen (MLlib)**

**2.1.6 Skalierung von R Programmen (SparkR)**

## **2.2 Mehrere Komponenten im Verbund**

## 2.3 Performance

Analysen von Performance Probleme erweisen sich mitunter als sehr schwierig. Apache Spark bringt zwar die seiteneffektfreie API mit, jedoch kann trotzdem eine Menge schief gehen. Es ist schwer immer im Hinterkopf zu behalten, dass Operationen auf vielen verteilten Rechnern ablaufen.

Über eine Webbasierte Übersicht ist es Möglich Informationen zu dann aktuell laufenden Auswertungen und Dauer von Ergebnissen etc. zu bekommen.<sup>7</sup>

Todo Beispiel Bild erstellen und erklären. Es gibt ein Live-Dashboard. Es gibt einen Stack-Trace Button

### 2.3.1 Besonderheiten bei der Speichernutzung

Zusätzlich wird oft unterschätzt, dass die Wahl einer geeigneten bzw. speichereffizienten Datenstruktur sehr viel bewirken kann. Spark geht davon aus, eine Datei in Blöcke einer bestimmten Größe geladen wird. In der Regel 128MB. Zu beachten ist jedoch, dass beim dekomprimieren größer Blöcke entstehen können. So können aus 128Mb schnell 3-4GB große Blöcke werden.

Um das Speichermanagement zu verbessern wurde ein per-node allocator implementiert. Dieser verwaltet den Speicher auf einer Node. Der Speicher wird in drei Bereiche geteilt. Speicher zum verarbeiten der Daten. Speicher für die hash-tables bei Joins oder Aggregations Speicher für unrolling Blöcke, um zu prüfen ob die einzulesenden Blöcke nach dem entpacken immer noch klein genug sind damit diese gecached werden können.

Damit läuft das System robust über einen großen Bereich.

### 2.3.2 Netzwerk und I/O-Traffic

Mit Spark wurde schon Operationen bei denen über 8000 Nodes involviert waren und über 1PB an Daten verarbeitet wurden durchgeführt. Das beansprucht natürlich die I/O Schicht enorm.

Um I/O Probleme zu vermeiden, bzw. diese besser in den Griff zu bekommen wurde als Basis das Netty-Framework<sup>8</sup> verwendet.

- Zero-copy I/O:  
Daten werden direkt von der Festplatte zu dem Socket kopiert. Das vermeidet Last an der CPU bei Kontextwechseln und entlastet zusätzlich den JVM<sup>9</sup> garbage collector<sup>10</sup>
- Off-heap network buffer management:  
Todo
- Mehrfache Verbindungen:  
Jeder Spark worker kann mehrere Verbindungen parallel bearbeiten.

---

<sup>7</sup>Vgl. Owen und Wills 2015, S. 12

<sup>8</sup>Netty: High-Performance Netzwerk Framework

<sup>9</sup>JVM: Todo

<sup>10</sup>garbage collector: Todo



## 2.4 Nutzung & Verbreitung

Durch die Unterstützung der drei Programmiersprachen scala, python und java ist arbeit mit Apache Spark einfacher, als wenn es nur einen einzige exotische Programmiersprache zur Nutzung gäbe.

Apache Spark unterstützt zudem noch verschiedene Datenquellen und Dateiformate. Zu den Datenquellen zählen die das Dateisystem S3<sup>11</sup> von Amazon und das HDFS<sup>12</sup>. Die Dateiformate können strukturiert (z.B.: CSV, Object Files), semi-strukturiert (z.B.: JSON) und unstrukturiert (z.B.: Textdatei) sein.

Unter den Mitwirkenden(Contributors) zählen über 400 Entwickler aus über 100 Unternehmen, Stand 2014

Es gibt über 500 produktive Installationen. Seit einigen Jahren finden weltweit jährlich unter dem Namen Spark Summit Konferenzen statt.

Heise.de beauftragte 2015 eine Umfrage in der 2136 Teilnehmer befragt wurden. Diese gaben an, dass 31% Prozent den Einsatz derzeit prüfen. 13% Nutzen bereits Apache Spark und 20% planen den Einsatz noch in dem damaligen Jahr. Scala lag als Programmiersprache mit großen Abstand vorn. Die Nutzung innerhalb verschiedener Berufsgruppen war sehr ähnlich. Mit 16% lag bei den Telekommunikationsunternehmen der Einsatz am höchsten. Eine detaillierte Übersicht ist in Abbildung 2.1 zu sehen.

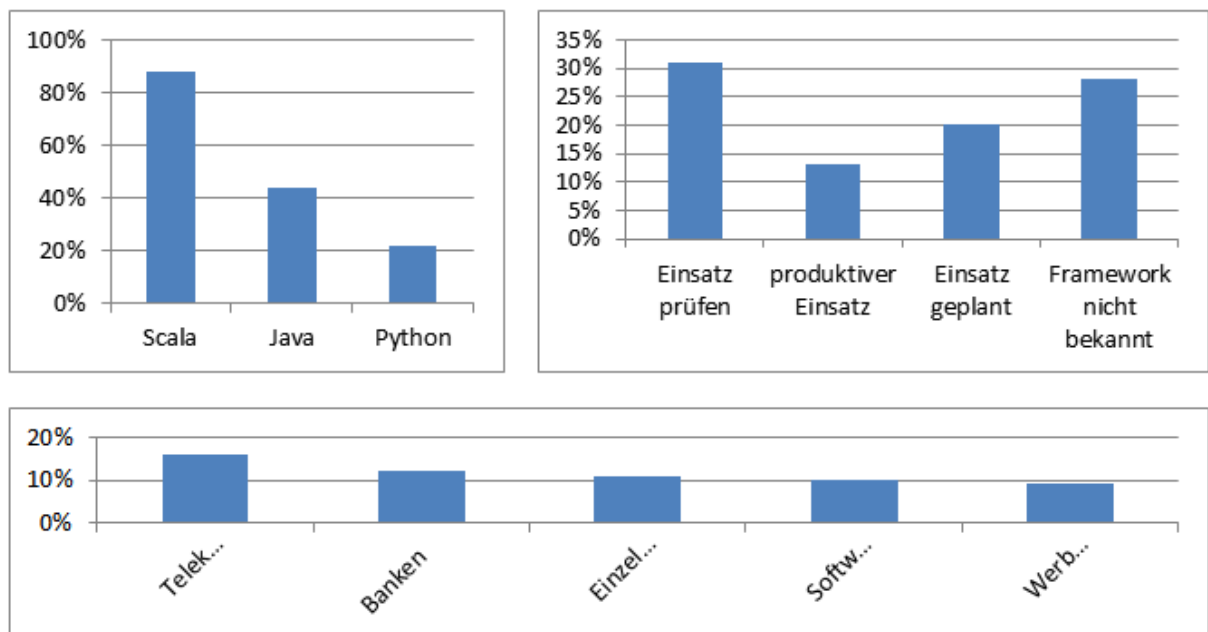


Abbildung 2.1: Einsatz & Verbreitung

<sup>11</sup>S3: Todo

<sup>12</sup>HDFS (Hadoop Distributed File System): Todo

## 3 Fazit

## **4 Ausblick & Weiterentwicklung**

## **5 Anhang**

## 6 Literaturverzeichnis

Owen, Sandy Ryza - Uri Laserson - Sean und Josh Wills (2015). *Advanced Analytics with Spark*. 1005 Gravenstein Highway North: O'Reilly Media, Inc.