FernUniversität in Hagen

–

Seminar 01912 / 19912
im Sommersemester 2017

„Skallierbare verteilte Datenanalyse“

Thema 2.3

Spark

Referent: Lukas Wappler

Inhaltsverzeichnis

1 Einleitung **3**

2 Apache Spark **4**
2.1 Kern-Bibliotheken / Komponenten 4
2.1.1 Grundlage des Systems (Spark-Core & RDDs) 5
2.1.2 SQL-Abfragen mit (Spark-SQL & Data Frames) 5
2.1.3 Verarbeitung von Datenstrmen (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 5
2.3 Performance 5
2.3.1 Besonderheiten bei der Speichernutzung 5
2.3.2 Netzwerk und I/O-Traffic 5
2.4 Nutzung & Verbreitung 7

3 Fazit **8**

4 Ausblick & Weiterentwicklung **9**

5 Anhang **10**

6 Literaturverzeichnis **11**

1 Einleitung

2 Apache Spark

Apache Spark wurde ist ein Open Source Framework, dass ermglicht Software verteilt über ein Cluster auszufhren. 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[1] ins Leben gerufen. Seit 2010 steht es als Open Source Software unter der BSD-Lizenz [2] zur Verfgung. Das Projekt wird seit 2013 von der Apache Software Foundation[3] weitergeführt. Seit 2014 ist es dort als Top Level Projekt eingestuft. Zum aktuellen Zeitpunkt steht Apache Spark unter der Apache 2.0 Lizenz[4] zur Verfügung.
Der Code liegt auf GitHub[5] und ist für jeden zugänglich. Bis zum 10.04.2017 gab es bereits 51 Releases, 19,365 commits und 1,053 contributors[6].

2.1 Kern-Bibliotheken / Komponenten

Apache Spark besteht im wesentlichen aus 5 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 & RDDs)

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

2.1.3 Verarbeitung von Datenstrmen (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

[1] AMPLab: Todo
[2] BSD-Lizenz (Berkeley Software Distribution-Lizenz): bezeichnet eine Gruppe von Lizenzen, die eine breitere Wiederverwertung erlaubt.
[3] Apache Software Foundation: TODO
[4] Apache 2.0 Lizenz: Die Software darf frei verwendet und verändert werden. Zusätzlich gibt es nur wenige Auflagen.
[5] GitHub: Todo
[6] contributors: Todo

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.

2.3.1 Besonderheiten bei der Speichernutzung

Zustzlich wird oft unterschtzt, dass die Wahl einer geeigneten bzw. speichereffizienten Datenstruktur sehr viel bewirken kann. Spark geht davon aus, eine Datei in Blck einer bestimmten Gre geladen wird. In der Regel 128MB. Zu beachten ist jedoch, dass beim dekomprimieren grer Blcke entstehen knnen. So knnen aus 128Mb schnell 3-4GB groe Blcke werden.

Um das Speichermanagement zu verbessern wurde ein per-node allocator implementiert. Dieser verwaltet den Speicher auf einer Node. Der Speicher wir in drei Breiche geteilt. Speicher zum verarbeiten der Daten. Speicher fr die hash-tables bei Joins oder Aggretaions Speicher fr ünrollingBlcke, um zu prfen ob die einzulesenden Blcke nach dem entpacken immer noch klein genug sind damit diese gecached werden knnen.

Damit luft das System robust ber eine groen Bereich.

2.3.2 Netzwerk und I/O-Traffic

Mit Spark wurde schon Operationen bei denen ber 8000 Nodes involviert waren und ber 1PB an Daten verarbeitet wurden durchgefhrt. Das beansprucht natrlich 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[7] 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 zustzlich den JVM[8] garbage collector[9]
- Off-heap network buffer management:
  Todo
- Mehrfache Verbindungen:
  Jeder Sparker worker kann mehrere Verbdingungen parallel bearbeiten.

[7] Netty: High-Performance Netzwerk Framework
[8] JVM: Todo
[9] garbage collector: Todo

2.4 Nutzung & Verbreitung

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

Unter den Mitwirkenden(Contributors) zhlen ber 400 Entwickler aus ber 100 Unternehmen, Stand 2014
Es gibt ber 500 produktive Installationen. Seit einigen Jahren finden weltweit jhrlich unter dem Namen Spark Summit Konferenzen statt.
Heise.de beauftrage 2015 eine Umfrage in der 2136 Teilnehmer befragt wurden. Diese gaben an, dass 31% Prozent den Einsatz derzeit prfen. 13% Nutzen bereits Apache Spark und 20% planten den Einsatz noch in dem damaligen Jahr. Scala lag als Programmiersprache mit groen Abstand vorn. Die Nutzung innerhalb verschiedener Berufsgruppen war sehr hnlich. Mit 16% lag bei den Telekommunikationsunternehmen der Einsatz am hchsten. Eine detaillierte bersicht ist in Abbildung 2.1 zu sehen.

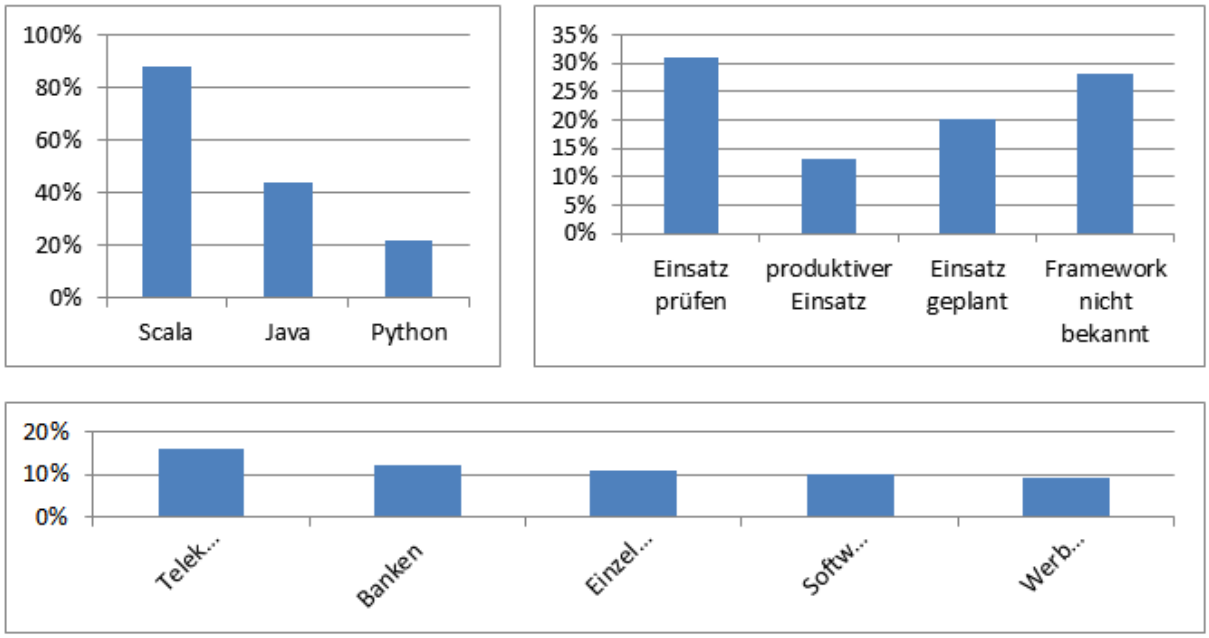

Abbildung 2.1: Einsatz & Verbreitung

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.