



Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Daniel Kirchner

Skalierbare Datenanalyse mit Apache Spark

**Architekturanalyse und Performancetests in verschiedenen
Anwendungsfällen**

*Fakultät Technik und Informatik
Studiendepartment Informatik*

*Faculty of Engineering and Computer Science
Department of Computer Science*

Daniel Kirchner

Skalierbare Datenanalyse mit Apache Spark
Architekturanalyse und Performancetests in verschiedenen
Anwendungsfällen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Kahlbrandt
Zweitgutachter: Prof. Dr. Zweitprüfer

Eingereicht am: 1. Januar 2345

Daniel Kirchner

Thema der Arbeit

Skalierbare Datenanalyse mit Apache Spark Architekturanalyse und Performancetests in verschiedenen Anwendungsfällen

Stichworte

Schlüsselwort 1, Schlüsselwort 2

Kurzzusammenfassung

Dieses Dokument ...

Daniel Kirchner

Title of the paper

Scalable Data Analysis with Apache Spark

Keywords

keyword 1, keyword 2

Abstract

This document ...

Inhaltsverzeichnis

1	Einführung	1
1.1	Motivation	1
1.2	Kontextabgrenzung	2
2	Vorstellung von Apache Spark	3
2.1	Überblick	3
2.2	Kernkomponenten	3
2.2.1	Lineage	3
2.2.2	Resilient Distributed Datasets	3
2.3	Standardbibliotheken	3
2.3.1	Spark SQL	3
2.3.2	MLlib	3
2.3.3	Streaming	3
2.3.4	GraphX	3
2.4	Verwandte Produkte	3
2.4.1	YARN	3
2.4.2	Mesos	3
2.4.3	Flink	3
3	Architektur und Skalierungsverhalten	4
3.1	Vorgehen	5
3.2	Versuchsaufbau	5
3.2.1	Hardware Setup	6
3.2.2	Performance-Basisdaten	6
3.2.3	Verteilungsdiagramm	6
3.3	Beispiel 1	6
3.3.1	Problemstellung	6
3.3.2	Lösungsbeschreibung	6
3.3.3	Ergebnisse	6
3.4	Beispiel 2	6
3.4.1	Problemstellung	6
3.4.2	Lösungsbeschreibung	6
3.4.3	Ergebnisse	6
3.5	Beispiel 3	6
3.5.1	Problemstellung	6

3.5.2	Lösungsbeschreibung	6
3.5.3	Ergebnisse	6
4	Schlussbetrachtung	7
4.1	Kritische Würdigung der Ergebnisse	7
4.2	Ausblick und offene Punkte	7

Listings

1 Einführung

1.1 Motivation

Die Entwicklung und Verbesserung von Frameworks zur Verarbeitung großer Datenmengen ist zur Zeit hochaktuell und sehr im Fokus von Medien und Unternehmen [VERWEIS]. Verschiedene Programme und Paradigmen konkurrieren um die schnellste, bequemste und stabilste Art großen Datenmengen einen geschäftsfördernden Nutzen abzurufen.

Unter dem Begriff „große Datenmengen“ oder „Big Data“ werden solche Datenmengen zusammengefasst, die die Kriterien Volume, Velocity, Variety [VERWEIS, Doug Laney] erfüllen oder „Datenmengen, die nicht mehr unter Auflage bestimmter SLAs auf einzelnen Maschinen verarbeitet werden können“ [VERWEIS, Hadoop/Yarn Entwickler].

Als Unternehmen, das früh mit solchen Datenmengen konfrontiert war implementierte Google das Map-Reduce Paradigma [VERWEIS] als Framework zur Ausnutzung vieler „Mittelklasse“-Rechner um Webseiten einzustufen und für andere Aufgaben [VERWEIS].

In Folge der Veröffentlichung ihrer Idee im Jahr 2005 [VERWEIS] wurde Map-Reduce in Form der OpenSource Implementation Hadoop (gemeinsam mit einer Implementation des Google File Systems GFS, u.a.) [VERWEIS] zum de-facto Standard für Big-Data-Analyseaufgaben [VERWEIS?].

Map-Reduce als Programmierparadigma zur effizienten Verarbeitung großer Datenmengen zeigt jedoch in vielen Anwendungsfällen Schwächen:

- Daten, die in hoher Frequenz verändert werden erfordern das ständige Neustarten eines Map-Reduce-Jobs. Iterative Algorithmen sind also nicht vorgesehen.
- Die Anfrage an ein solches System erfolgt in Form von kleinen Programmen. Dieses Verfahren ist offensichtlich nicht so deklarativ und leicht erlernbar wie SQL-Anfragen an klassische Datenbanken.

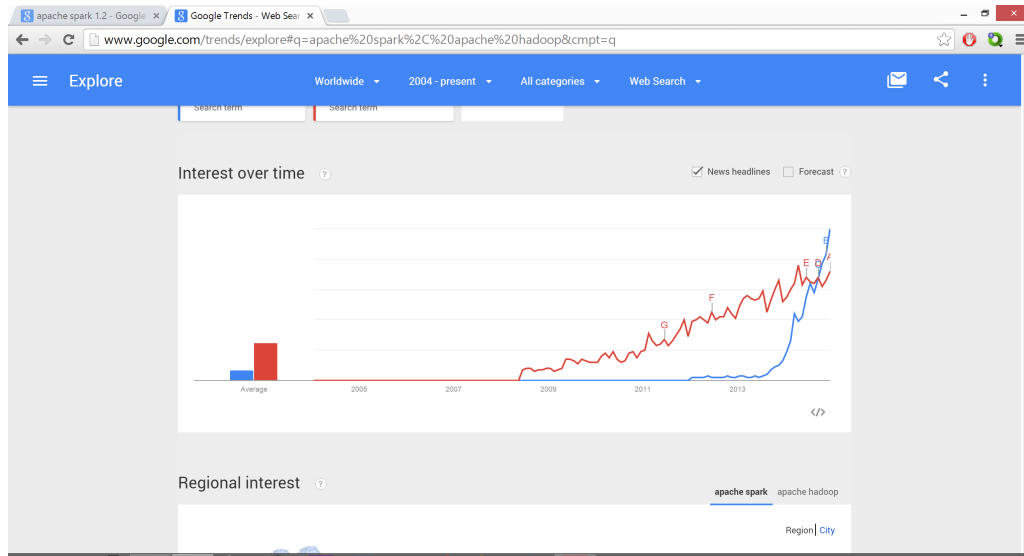
In der Folge entstanden viele Ansätze dieses Paradigma zu ersetzen, zu ergänzen oder durch übergeordnete Ebenen und High-Level-APIs zu vereinfachen.

1 Einführung

- [VERWEIS: A survey of large scale...] oder Aufzählung.

Eine der Alternativen zu der Map-Reduce-Komponente in Hadoop die „general engine for large-scale data processing“ Apache Spark.

Ein Indiz für das steigende Interesse an diesem Produkt liefert unter anderem ein Vergleich des Interesses an Hadoop und Spark auf Google:



1.2 Kontextabgrenzung

2 Vorstellung von Apache Spark

2.1 Überblick

2.2 Kernkomponenten

2.2.1 Lineage

2.2.2 Resilient Distributed Datasets

2.3 Standardbibliotheken

2.3.1 Spark SQL

2.3.2 MLlib

2.3.3 Streaming

2.3.4 GraphX

2.4 Verwandte Produkte

2.4.1 YARN

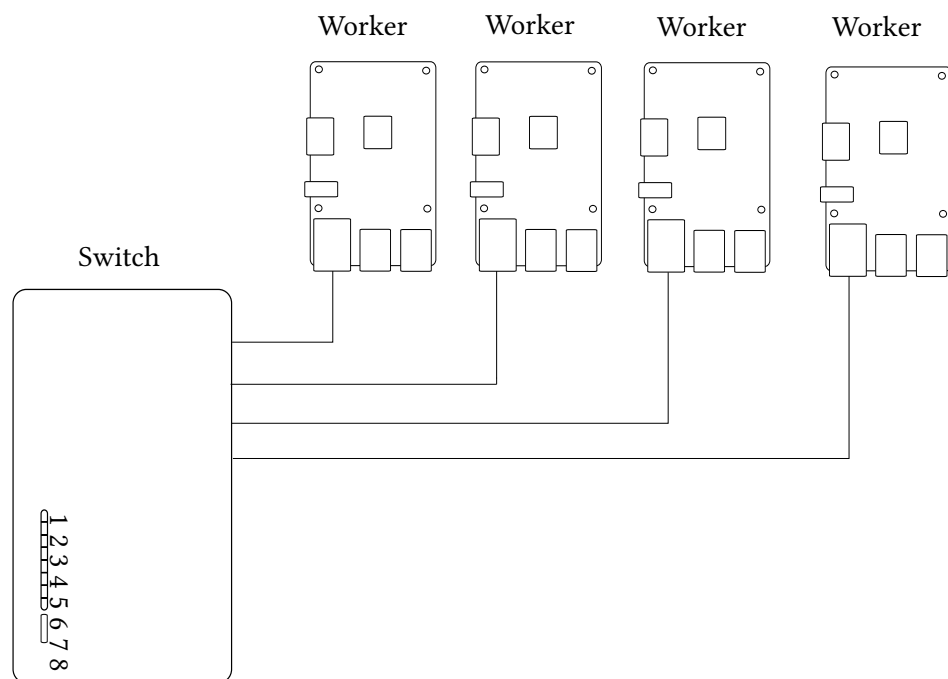
2.4.2 Mesos

2.4.3 Flink

3 Architektur und Skalierungsverhalten

3.1 Vorgehen

3.2 Versuchsaufbau



3.2.1 Hardware Setup

3.2.2 Performance-Basisdaten

3.2.3 Verteilungsdiagramm

3.3 Beispiel 1

3.3.1 Problemstellung

3.3.2 Lösungsbeschreibung

3.3.3 Ergebnisse

3.4 Beispiel 2

3.4.1 Problemstellung

3.4.2 Lösungsbeschreibung

3.4.3 Ergebnisse

3.5 Beispiel 3

3.5.1 Problemstellung

3.5.2 Lösungsbeschreibung

3.5.3 Ergebnisse

4 Schlussbetrachtung

4.1 Kritische Würdigung der Ergebnisse

4.2 Ausblick und offene Punkte

See also [One und Two \(2010\)](#).

Literaturverzeichnis

[One und Two 2010] ONE, Author ; TWO, Author: A Sample Publication. (2010)

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 1. Januar 2345 Daniel Kirchner
