



Hochschule für Angewandte Wissenschaften Hamburg  
*Hamburg University of Applied Sciences*

# Bachelorarbeit

**Daniel Kirchner**

## **Skalierbare Datenanalyse mit Apache Spark**

**Evaluation von Anwendungsfällen aus Klimaforschung und Text-Mining**

*Fakultät Technik und Informatik  
Studiendepartment Informatik*

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

Daniel Kirchner

## **Skalierbare Datenanalyse mit Apache Spark**

**Evaluation von Anwendungsfällen aus Klimaforschung und Text-Mining**

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 Evaluation von Anwendungsfällen aus Klimaforschung und Text-Mining

**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

<b>1</b>	<b>Einführung</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Kontextabgrenzung . . . . .	2
<b>2</b>	<b>Vorstellung von Apache Spark</b>	<b>4</b>
2.1	Überblick . . . . .	4
2.2	Kernkonzepte . . . . .	4
2.2.1	Resilient Distributed Datasets . . . . .	4
2.2.2	Lineage . . . . .	4
2.2.3	DAG Scheduler . . . . .	4
2.3	Standardbibliotheken . . . . .	4
2.3.1	Spark SQL . . . . .	4
2.3.2	MLlib . . . . .	4
2.3.3	Streaming . . . . .	4
2.3.4	GraphX . . . . .	4
2.4	Entwicklergemeinschaft . . . . .	4
2.5	Verwandte Produkte . . . . .	4
2.5.1	YARN . . . . .	5
2.5.2	Mesos . . . . .	5
2.5.3	Flink . . . . .	5
<b>3</b>	<b>Untersuchung von Anwendungsfällen</b>	<b>6</b>
3.1	Identifikation von Hot Topics in der Spark Community . . . . .	6
3.1.1	Beschreibung des Problems . . . . .	6
3.1.2	Hardwarekontext und Performance-Basisdaten . . . . .	6
3.1.3	Architekturübersicht . . . . .	8
3.1.4	Detaillierte Lösungsbeschreibung . . . . .	8
3.1.5	Ergebnisse . . . . .	8
3.2	Evaluierung einer spark-basierten Implementation von CDOs auf einem HPC Cluster mit nicht-lokalem Storage . . . . .	8
3.2.1	Beschreibung des Problems . . . . .	8
3.2.2	Hardwarekontext und Performance-Basisdaten . . . . .	8
3.2.3	Architekturübersicht . . . . .	8
3.2.4	Detaillierte Lösungsbeschreibung . . . . .	8
3.2.5	Ergebnisse . . . . .	8

<b>4</b>	<b>Schlussbetrachtung</b>	<b>9</b>
4.1	Kritische Würdigung der Ergebnisse . . . . .	9
4.2	Ausblick und offene Punkte . . . . .	9

# 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 abzuringen.

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 kostengünstiger 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?].

Reines Map-Reduce (nach Art von Hadoop) als Programmierparadigma zur Verarbeitung großer Datenmengen zeigt jedoch in vielen Anwendungsfällen Schwächen:

- Daten, die in hoher Frequenz entstehen und schnell verarbeitet werden sollen erfordern häufiges Neustarten von Map-Reduce-Jobs.
- Algorithmen die während ihrer Ausführung iterativ Zwischenergebnisse erzeugen und auf vorherige angewiesen sind (typischerweise Maschinenlernalgorithmen) können nur durch persistentes Speichern der Daten und wiederholtes Auslesen zwischen allen Iterationsschritten implementiert werden.

## 1 Einführung

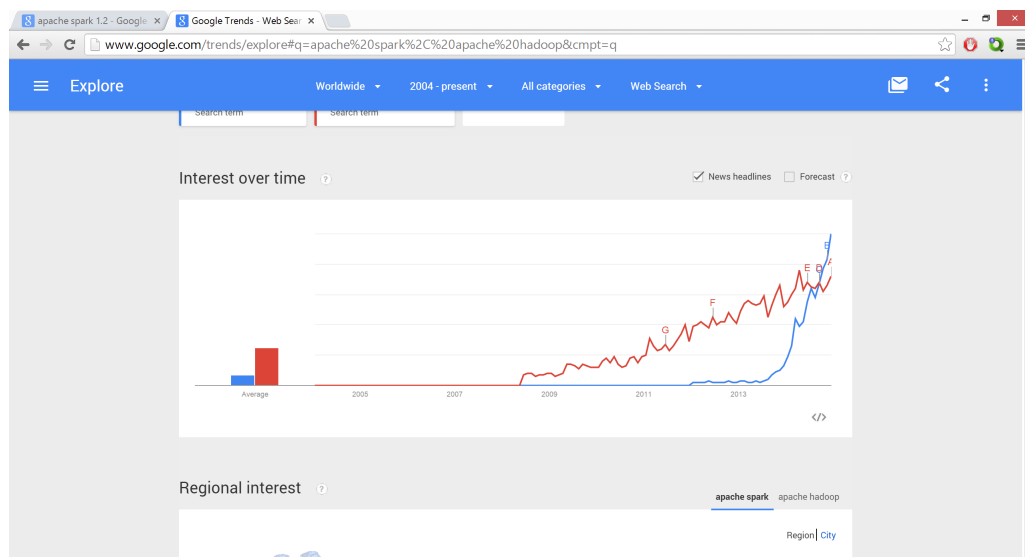
- Generell erfolgen Anfragen an ein solches System immer in Form von kleinen Programmen. Dieses Verfahren ist offensichtlich nicht so deklarativ und leicht erlernbar wie beispielsweise SQL-Anfragen an klassische Datenbanken.

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

- [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

Das Ziel dieser Arbeit ist es einen Einblick in die grundlegenden Konzepte und Anwendungsmöglichkeiten von Apache Spark zu vermitteln.

Für ein tieferes Verständnis werden zwei Anwendungsfälle untersucht und deren Lösung detailliert dokumentiert und bewertet.

Nur am Rande wird betrachtet:

- Vergleich mit ähnlichen Produkten



- Empirische Messung des Skalierungsverhaltens
- Konkrete Hinweise zu Installation und Nutzung

## **2 Vorstellung von Apache Spark**

### **2.1 Überblick**

### **2.2 Kernkonzepte**

— Warum ist Spark so schnell (und wo vielleicht nicht)? —

#### **2.2.1 Resilient Distributed Datasets**

#### **2.2.2 Lineage**

#### **2.2.3 DAG Scheduler**

### **2.3 Standardbibliotheken**

— Warum ist Spark so einfach (und wo vielleicht nicht)? —

#### **2.3.1 Spark SQL**

#### **2.3.2 MLlib**

#### **2.3.3 Streaming**

#### **2.3.4 GraphX**

### **2.4 Entwicklergemeinschaft**

### **2.5 Verwandte Produkte**

— Ergänzende oder konkurrierende Produkte —

**2.5.1 YARN**

**2.5.2 Mesos**

**2.5.3 Flink**

## 3 Untersuchung von Anwendungsfällen

Im Folgenden wird Apache Spark im Rahmen zweier grundsätzlich verschiedener Anwendungsfälle betrachtet.

Beispiel 1: Eine typische Anwendung mit verteiltem lokalem Storage (HDFS) und Spark als „Client“ eines bestehenden Yarn Clustermanagers. — Commodity Hardware (Raspberry Pi Cluster). —

Beispiel 2: Eine untypische Anwendung mit verteiltem entfernten Storage und dem Spark Standalone Clustermanager. — HPC Hardware („Thunder“ des Hamburger KlimaCampus). —

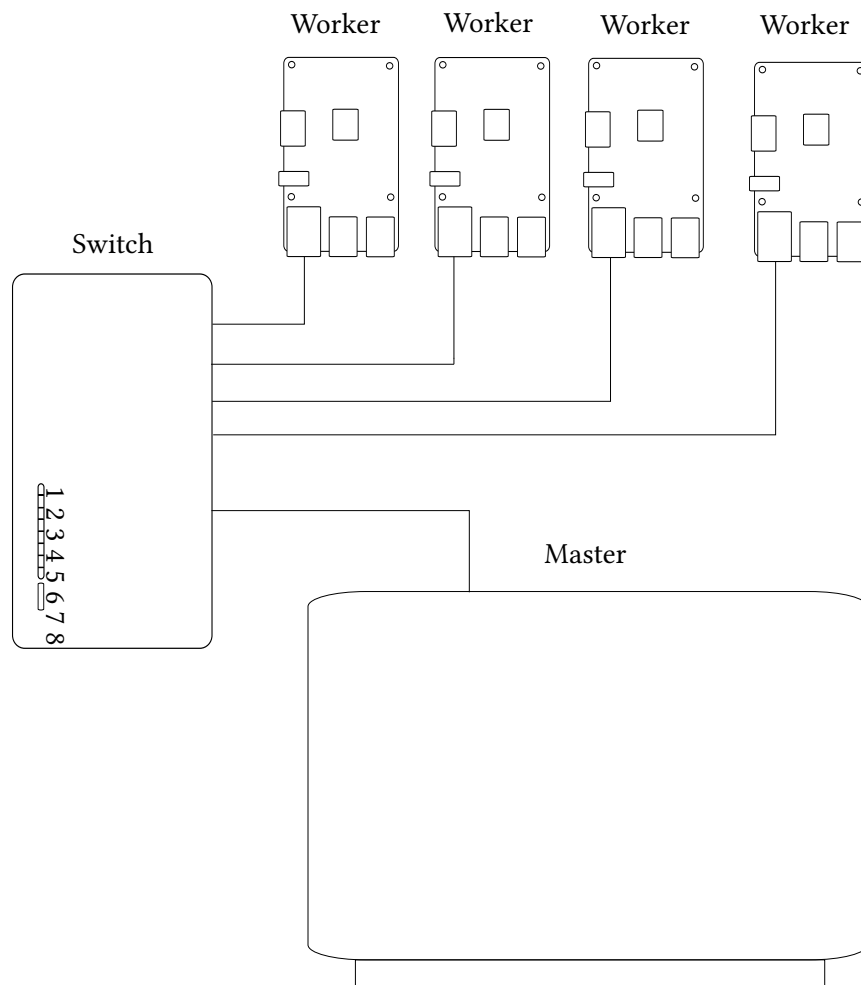
### 3.1 Identifikation von Hot Topics in der Spark Community

— Fusion von Tweets und Mailinglisten <https://spark.apache.org/docs/1.3.0/mllib-feature-extraction.html>  
Implementation auf einem Raspberry Pi Cluster mit HDFS und Yarn Clustermanager —

#### 3.1.1 Beschreibung des Problems

#### 3.1.2 Hardwarekontext und Performance-Basisdaten

— hier kommen die eingesetzten systeme, und relevante laufzeitmessungen (netzwerk, storage, cpu) hin —



### 3.1.3 Architekturübersicht

— hier kommen Verteilungs- und Komponentendiagramm hin —

### 3.1.4 Detaillierte Lösungsbeschreibung

— hier kommen laufzeitdiagramme und codeschnipsel hin —

### 3.1.5 Ergebnisse

— Tabellen und Diagramme Ergebnissen, evt. Skalierungsverhalten — — Bewertung —

## 3.2 Evaluierung einer spark-basierten Implementation von CDOs auf einem HPC Cluster mit nicht-lokalem Storage

— Implementation ausgewählter CDOs (sehr wenige, möglicherweise nur 1-2) mit der Core-API von Spark. Testlauf auf einem HPC Cluster mit nicht-lokalem, allerdings per Infiniband angeschlossenen Storage. Insbesondere Betrachtung des Skalierungsverhaltens und der „Sinnhaftigkeit“. —

### 3.2.1 Beschreibung des Problems

— Erläuterung von CDOs (Climate Data Operators). —

### 3.2.2 Hardwarekontext und Performance-Basisdaten

— hier kommen die eingesetzten systeme, und relevante laufzeitmessungen (netzwerk, storage, cpu) hin —

### 3.2.3 Architekturübersicht

— hier kommen Verteilungs- und Komponentendiagramm hin —

### 3.2.4 Detaillierte Lösungsbeschreibung

— hier kommen laufzeitdiagramme und codeschnipsel hin —

### 3.2.5 Ergebnisse

— Tabellen und Diagramme Ergebnissen, evt. Skalierungsverhalten — — Bewertung —

## **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

---