Stonebraker versus Google: 2-0 scores in Rostock - A comparison of big data analytics environments

Daniel Dietrich
Universität Rostock
Albert-Einstein-Str. 22
18059 Rostock
daniel.dietrich@uni-rostock.de

Ole Fenske
Universität Rostock
Albert-Einstein-Str. 22
18059 Rostock
ole.fenske@uni-rostock.de

Stefan Schomacker
Universität Rostock
Albert-Einstein-Str. 22
18059 Rostock
stefan.schomacker@uni-rostock.de

Philipp Schweers
Universität Rostock
Albert-Einstein-Str. 22
18059 Rostock
philipp.schweers@uni-rostock.de

Andreas Heuer
Universität Rostock
Albert-Einstein-Str. 22
18059 Rostock
heuer@informatik.uni-rostock.de

ABSTRACT

In our research project PArADISE, the data-driven development of assistive systems is supported by the highly parallel analysis of large amounts of sensor data. To achieve different aims such as the preservation of privacy, provenance, and sustainability, we stick to SQL as a basis to express the evaluation programs (mining or machine learning algorithms). These SQL queries should then be evaluated by a parallel DBMS. Of course, parallel row store DBMS are competing with column oriented DBMS architectures, as well as with recent big data analytics environments such as map reduce or dataflow programming environments. In a paper of Stonebraker (CACM, 2010), the superiority of row and column stores over a map reduce framework (Hadoop) has been shown several years ago. Years later, we wanted to reconstruct the results of Stonebraker within two studentsÂt' projects at the University of Rostock. Additionally, we wanted to transfer the results to other kinds of tasks and to more recent software environments. The aim of this paper is to present the results of these two studentsÂt' projects.

30^{th} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 22.05.2018 - 25.05.2018, Wuppertal, Germany.

Stonebraker gegen Google: Das 2:0 fällt in Rostock

Ein Vergleich von Big-Data-Analytics-Plattformen

Daniel Dietrich
Universität Rostock
Albert-Einstein-Str. 22
18059 Rostock
daniel.dietrich@uni-rostock.de

Ole Fenske
Universität Rostock
Albert-Einstein-Str. 22
18059 Rostock
ole.fenske@uni-rostock.de

Stefan Schomacker
Universität Rostock
Albert-Einstein-Str. 22
18059 Rostock
stefan.schomacker@uni-rostock.de

Philipp Schweers
Universität Rostock
Albert-Einstein-Str. 22
18059 Rostock
philipp.schweers@uni-rostock.de

Andreas Heuer
Universität Rostock
Albert-Einstein-Str. 22
18059 Rostock
heuer@informatik.uni-rostock.de

ABSTRACT

Im Projekt PArADISE wird die datengetriebene Entwicklung von Assistenzsystemen durch die hochparallele Analyse großer Mengen von Sensordaten unterstützt. Um dabei verschiedene Ziele wie die Sicherung von Privatsphäre, Provenance und Nachhaltigkeit zu erreichen, sind wir darauf angewiesen, die Analyseprogramme (Mining- oder Machine-Learning-Algorithmen) in SQL umzusetzen und dann möglichst mit parallelen DBMS zu realisieren. Dabei stehen diese parallelen DBMS-Lösungen auf zeilenorientierten DBMS-Architekturen natürlicherweise in Konkurrenz zu spaltenorientierten Architekturen, gleichzeitig aber auch zu modernen Big-Data-Analyse-Umgebungen wie MapReduce- oder Datenflussprogrammierungsansätzen. In einem Artikel von Stonebraker [11] wurde die Überlegenheit von zeilen- und spaltenorientierten DBMS gegenüber eines MapReduce-Ansatzes (Hadoop) gezeigt. Die Ergebnisse von Stonebraker sollten nun einige Jahre später in zwei studentischen Projekten an der Universität Rostock nachvollzogen, aber auch auf andere Arten von Problemen und neuere Software-Plattformen übertragen werden. Ziel dieses Artikels ist, die Ergebnisse der beiden studentischen Projekte zu präsentieren.

Categories and Subject Descriptors

Information Systems [**Database Management System Engines**]: MapReduce-based systems; Information Systems [**Database Management System Engines**]: Relational parallel and distributed DBMSs; Information Systems [**Database Administration**]: Database performance evaluation; Computer Systems Organization [**Parallel Architectures**]: Multicore architectures

Keywords

Big Data Analytics, Parallele DBMS, Map-Reduce, Performance, Postgres-XL, Hadoop, Spark, Flink

1. MOTIVATION

Im Rostocker Projekt PArADISE [7, 9] wird die datengetriebene Entwicklung von Assistenzsystemen durch die hochparallele Analyse großer Mengen von Sensordaten unterstützt. Dabei sollen neben der effizienten Datenanalyse zur Unterstützung der Modellbildung in Assistenzsystemen diverse weitere Ziele erreicht werden: die Sicherung der Privatsphäre der das Assistenzsystem nutzenden Personen, das Provenance Management zur Ermittlung von Ursachen bei fehlerhaften Modellbildungen, und die Nachhaltigkeit der Analyseprogramme im Kontext einer Informationssystem-Infrastruktur beim Diensteanbieter des Assistenzsystems. Die Architektur des PArADISE-Frameworks wird in Abschnitt 2 noch genauer beschrieben.

Um die oben genannten Ziele zu erreichen, sind wir darauf angewiesen, die Analyseprogramme (Mining- oder Machine-Learning-Algorithmen) in SQL umzusetzen und dann möglichst mit parallelen DBMS zu realisieren. Dabei stehen diese parallelen DBMS-Lösungen auf zeilenorientierten DBMS-Architekturen natürlicherweise in Konkurrenz zu spaltenorientierten Architekturen, gleichzeitig aber auch zu modernen Big-Data-Analyse-Umgebungen wie MapReduce- oder Datenflussprogrammierungsansätzen.

Die *Communications of the ACM* hatte im Jahre 2010 zwei Artikel mit gegensätzlichen Positionen veröffentlicht: einen Artikel von Google [6] über den Sinn und die Vorteile MapReduce-artiger Lösungen wie Hadoop, dazu einen Artikel von Michael Stonebraker [11] über die Überlegenheit von zeilen- und spaltenorientierten DBMS gegenüber eines MapReduce-Ansatzes, speziell Hadoop. Stonebraker hat dabei neben Hadoop auch sein eigenes spaltenorientiertes DBMS Vertica (heute bei HP) und ein nicht genanntes zeilenorientiertes DBMS genutzt. Die drei Systeme wurden in drei verschiedenen Aufgaben (Tasks) getestet. Die drei Ori-

30^{th} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 22.05.2018 - 25.05.2018, Wuppertal, Germany.

ginalaufgaben von Stonebraker, die Testumgebung und die Testergebnisse stellen wir in Abschnitt 3 noch genauer vor.

Kurz zusammengefasst kann gesagt werden: Stonebraker stellte die Überlegenheit der DBMS-Lösungen gegenüber Hadoop nicht nur in Testfällen heraus, in denen man diese Überlegenheit erwarten konnte (Aufgaben, die einen Verbund großer Datenbestände als Teilproblem hatten), sondern auch in Tasks, für die eine MapReduce-artige Verarbeitung eigentlich eingeführt wurde (Wortsuche in Texten). Nach diesen Tests stand es 1:0 für die von Stonebraker favorisierten parallelen DBMS als Big-Data-Plattformen.

Die Ergebnisse von Stonebraker sollten nun einige Jahre später in zwei studentischen Projekten an der Universität Rostock nachvollzogen, aber auch auf andere Arten von Problemen und neuere Software-Plattformen übertragen werden (siehe Abschnitt 4 für die genauere Aufgabenstellung). Neben Hadoop sollten auch neuere Systeme wie Spark, Flink, Naiad und Tensorflow getestet werden. Und neben den drei Task-Typen von Stonebraker sollte noch mindestens ein Data-Mining-Verfahren als komplexere Task ergänzt werden. Die Testumgebung und die Testfälle, die in Rostock umgesetzt wurden, werden in den Abschnitten 5 und 6 genauer vorgestellt.

Die Ergebnisse der beiden studentischen Projekte werden ebenfalls in Abschnitt 6 genauer vorgestellt. Sie untermauern die These von Stonebraker, dass nicht nur spaltenorientierte DBMS bei vielen Typen von Analysen auf großen Datenmengen einer MapReduce-Lösung überlegen sind, sondern auch zeilenorientierte Architekturen mithalten können. Die Tests haben also für die Stonebraker-Argumentation kein Gegentor beschert, sondern die Argumentation bestätigt: es steht damit 2:0 für die DBMS-Plattformen.

2. DAS PROJEKT PARADISE

PArADISE (Privacy-AwaRe Assistive Distributed Information System Environment) [7, 9] unterstützt die datengetriebene Entwicklung von Assistenzsystemen durch die hochparallele Analyse großer Mengen von Sensordaten. Das in dem Projekt entwickelte Framework besteht aus drei großen Phasen (siehe Abbildung 1):

- In der *Entwicklungsphase* (links im Bild) werden unter Versuchsbedingungen massiv Sensordaten erfasst, um daraus Modelle für Situationen, Handlungen und Intentionen der beteiligten Personen abzuleiten. Die Ableitung der Modelle geschieht über Machine-Learning-Algorithmen (ML), die in SQL umgesetzt werden. Die Umsetzung in SQL wird in [9] näher erläutert.

- In der *Transformationsphase* (im Bild der Pfeil von links nach rechts) wird mit Provenance-Management-Techniken eine kleine, aber aussagefähige Auswahl der Sensoren getroffen, mit der unter genügender Konfidenz ähnliche Modelle hergeleitet werden können. Die Rolle des Provenance Management bei der Entwicklung von Assistenzsystemen skizzieren wir in [8].

- In der *Nutzungsphase* (rechts im Bild) werden beim späteren Einsatz des Assistenzsystems die in SQL vorliegenden Machine-Learning-Algorithmen automatisch auf sensornahe Schichten des gesamten Verarbeitungsnetzwerkes transformiert. Diese Phase wird in [7] vorgestellt und realisiert die datensparsame Analyse der Sensordaten.

	Hadoop	DBMS-X	Vertica
Grep	284 s	194 s	108 s
Web Log	1146 s	740 s	268 s
Join	1158 s	32 s	55 s

Tabelle 1: Ergebnisse der Stonebraker-Tests

Sowohl in der Entwicklungsphase als auch in der Nutzungsphase werden wir auf einem Parallelrechner große Mengen von Sensordaten analysieren müssen. Das Zielsystem für diese Analysen ist ein paralleles DBMS, das SQL-Anfragen parallel und effizient verarbeiten kann (in der Architektur mit PSQL und PDBMS bezeichnet). Um abschätzen zu können, ob (und wenn ja, wie stark) die Performance durch die Nutzung eines SQL-DBMS leidet, testen wir eine solche Lösung gegen spezialisierte Plattformen wie Hadoop, Spark und Flink. Ziel ist dabei nicht unbedingt, in jedem Fall besser zu sein als diese Plattformen, sondern möglichst wenig gegenüber diesen zu verlieren. Die Vorteile der Nutzung von DBMS-Technologien (wie erwähnt: Privacy, Provenance, Nachhaltigkeit) wiegen schwerer als ein geringer Performance-Verlust.

Beim Start des PArADISE-Projektes waren daher die Ergebnisse von Stonebraker [11] ein interessanter Ausgangspunkt, den wir in diesem studentischen Teilprojekt näher untersuchen wollten.

3. DIE ERGEBNISSE VON STONEBRAKER

Stonebraker et al. publizierten 2010 [11] die Ergebnisse eines Vergleichs zwischen Hadoop 0.19.0, DBMS-X und Vertica. DBMS-X ist dabei ein nicht benanntes, kommerzielles, zeilenorientiertes und paralleles DBMS. Vertica hat eine spaltenorientierte und parallele Architektur. Der Test verwendete ein Cluster mit 100 Knoten. Jeder dieser Knoten hatte einen 2.4 GHz Intel Core 2 Duo Prozessor und 4 GB RAM. Die Resultate sind der Tabelle 1 zu entnehmen.

Es wurden die folgenden Szenarien verglichen:

Grep Task. Hierbei werden 10 Milliarden Datensätze (1 TB) nach einer Zeichenkette durchsucht. Bei 100 Knoten ergibt sich eine Datenmenge von 10 GB pro Knoten. Ein Datensatz besteht aus 100 Bytes (10 Bytes Schlüssel, 90 Bytes Wert). Es liegt keine Sortierung vor und es darf kein Index verwendet werden. Dieser Task ist die Basis für Analysen von Webseiten, etwa bei Google. Es ist daher zu erwarten, dass dieser Task sehr gut zu Map-Reduce-Systemen passt.

Web Log Task. Die zweite Aufgabe besteht aus einer Aggregation mit *GROUP BY* auf einem Web Log mit Besuchen. Der Log hat eine Größe von 2 TB und beinhaltet 155 Millionen Datensätze. Bei 100 Knoten ergibt sich eine Datenmenge von 20 GB pro Knoten. Diese Aufgabe wurde ohne Index durchgeführt.

Join Task. Diese Aufgabe beschreibt einen Verbund über zwei Tabellen, eine Selektion und eine Aggregation. Der Web Log wird mit einer PageRank-Tabelle verbunden, die 18 Millionen Datensätze und eine Größe von 100 TB hat. Im ersten Teil muss die IP-Adresse mit der größten Besuchsanzahl (Visits) in einem gewissen Zeitraum gefunden werden. Im zweiten Teil wird daraus der durchschnittliche PageRank berechnet.

Grep Task und Join Task waren dann Ausgangspunkt für die eigenen Untersuchungen, die wir im folgenden Abschnitt

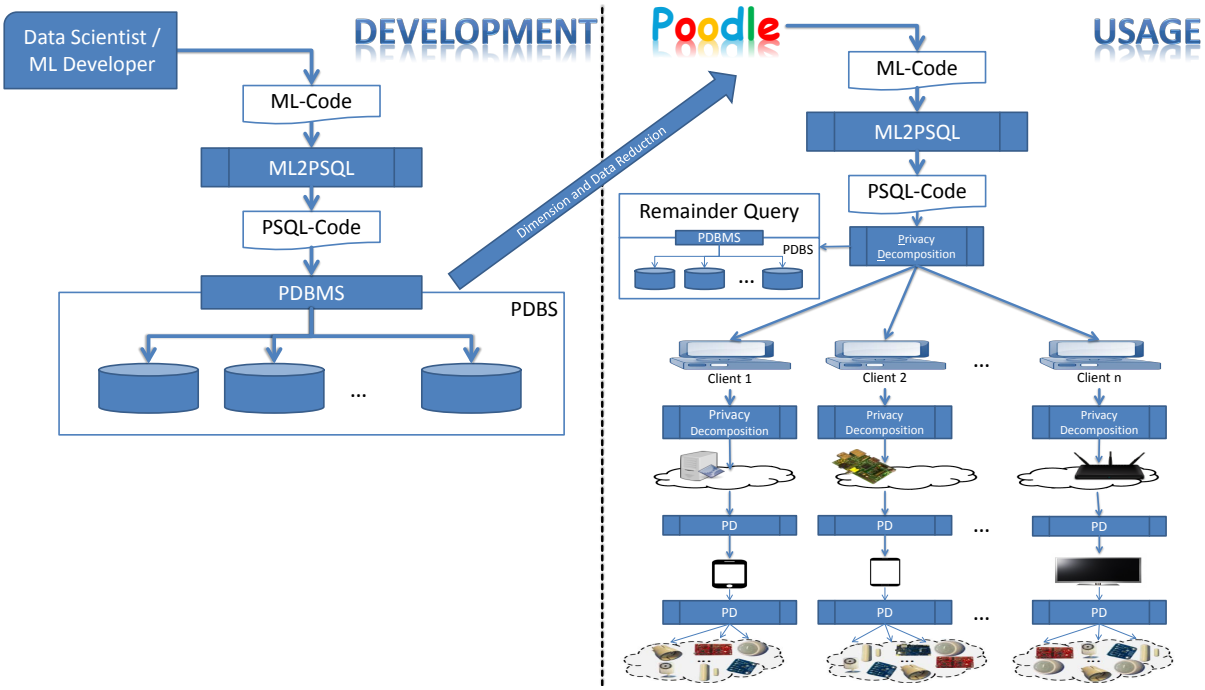


Abbildung 1: Das PArADISE-Projekt (aus [9] und [7])

beschreiben.

Einordnung der Ergebnisse von Stonebraker. Sicher war der Stonebraker-Artikel [11] von den Kriterien her sehr einseitig. So wurden die Effizienzaussagen ohne genauere Details veröffentlicht. Etwa ist nicht klar, welche Join-Implementierungen in welchen DBMS in den internen Anfragepläne generiert bzw. welche Join-Implementierungen in der Hadoop-Umgebung durch das Test-Team umgesetzt wurden. In unserer eigenen Testumgebung wurden daher bei keiner Plattform die Join-Varianten bzw. die Datenpartitionierung auf dem parallelen System von außen beeinflusst. Weiterhin wurden Vorteile der Big-Data-Analytics-Plattformen wie Hadoop (und eventuelle Nachfolger) nicht weiter betrachtet: außer der Effizienzfragestellung gibt es auch Kriterien wie *Cost of Ownership*, Administrationsaufwand und *Ease of Use*, verfügbare Schnittstellen und vor allem die einfache Skalierbarkeit, bei der in vielen Fällen die eingesetzten DBMS-Lösungen schlechter als Hadoop ausgesehen hätten.

Für unsere Originalfragestellung, ob die von uns aus anderen Gründen favorisierten SQL-basierten Systeme zur parallelen Auswertung großer Datenmengen von der Effizienz her mit den spezialisierten Big-Data-Analytics-Frameworks mithalten können, hatten wir nun aber einen interessanten Ausgangspunkt, den wir in eigenen Tests vertiefen wollten.

4. AUFGABENSTELLUNG

Im Sommersemester 2017 wurde in zwei verschiedenen Projektveranstaltungen (eine für den Bachelor Wirtschaftsinformatik, eine für den Bachelor Informatik) die Aufgabe gestellt, die Stonebraker-Ergebnisse nachzuvollziehen, aber auch auf andere Arten von Problemen und neuere Software-Plattformen zu übertragen. Die Wirtschaftsinformatik-Gruppe sollte dabei neben Hadoop auch neuere Systeme wie Spark, Flink, Naiad und Tensorflow untersuchen und drei davon für reale Tests aussuchen. Die Informatik-Gruppe sollte diese Tests parallel auf einem parallelen SQL-DBMS, in diesem Fall Postgres-XL, durchführen.

Der nötige Aufwand für die Realisierung der Tests ist zwar durch die verschiedenen Projektformen im Wirtschaftsinformatik- und Informatik-Studium nicht direkt vergleichbar, die Aufwände wurden aber in den Projekten gemessen und sind von der Größenordnung her vergleichbar:

- Die fünfköpfige Wirtschaftsinformatik-Gruppe wendete nach Abzug von Einarbeitungs- sowie Installations- und Administrations-Tätigkeiten 160 Stunden für die Datenaufbereitung und 170 Stunden für die Implementierung der Tasks in den drei für reale Tests ausgewählten Plattformen auf.

- Von der vierköpfigen Informatik-Gruppe waren drei nur für Installation und Administration des Postgres-XL-Systems zuständig, da dieses auch für mehrere Parallelprojekte benutzt wurde. Nur einer der Informatik-Studenten beschäftigte sich dann mit den Stonebraker-Tests. Für die Datenaufbereitung wurden durch die Vorarbeiten der Wirtschaftsinformatik-Gruppe nur 40 Stunden benötigt, die Implementierung der Tasks erfolgte dann (nach Abzug der Einarbeitung) in 60 Stunden.

Pro System war der Aufwand zur Umsetzung der Tasks mit 50 bis 60 Stunden erstaunlich gleichmäßig verteilt.

Neben den drei Task-Typen von Stonebraker sollte noch mindestens ein Data-Mining-Verfahren als komplexere Task ergänzt werden. In den beiden Projektgruppen wurde dann entschieden, den Web Log Task des Stonebraker-Tests durch

einen einfachen Data-Mining-Algorithmus (Clustering durch k-Means) zu ersetzen. Die Testumgebung und die Testfälle werden in folgenden Abschnitten 5 und 6 noch genauer vorgestellt.

Nach einer Literaturanalyse wurde von der Wirtschaftsinformatik-Gruppe die Auswahl der Systeme auf drei beschränkt: Hadoop [6] als Basis auch für die neueren Systeme Spark [12] und Flink [4] wurden auf der zur Verfügung stehenden Systemumgebung installiert und evaluiert. Zunächst war auch Tensorflow (von Google) [2] ein Kandidat für die realen Evaluierungen: erste Tests ergaben aber, dass Tensorflow in der 2017 zur Verfügung stehenden Fassung auf den großen Datenmengen und den noch wenig auf Machine-Learning-Algorithmen zugeschnittenen Problemen mit deutlichem Abstand nicht konkurrenzfähig war. Das Microsoft-System Naiad [10] wurde schon nach der Literaturanalyse aufgrund von zu spärlichen Informationen und Manuals ausgeschlossen.

Postgres-XL [1] wurde in der Informatik-Gruppe als einziges paralleles SQL-DBMS ausgewählt, da es als Open-Source-System zur Verfügung stand und da auch andere Forschungsprojekte am Lehrstuhl diese Plattform als Basis benutzten. Beachtet werden sollte also bei dieser Auswahl, dass im Gegensatz zum Stonebraker-Test keine spaltenorientierte Architektur zur Verfügung stand.

Wir werden nun die Hardware-Umgebung für die Tests vorstellen und danach die neuen Tasks sowie die Ergebnisse auf den verschiedenen Plattformen. Bei der Hardware-Umgebung muss beachtet werden, dass das RMRDF[1]-Großgerät durch die Projektgruppen nur in kleinerem Maßstab benutzt werden konnte, so dass die Ergebnisse zum Vergleich mit den Stonebraker-Daten skaliert werden müssen.

5. TESTUMGEBUNGEN

Für die Durchführung der Testfälle standen drei Virtual Private Server (VPS) zur Verfügung. Jeder Knoten besteht aus einem Intel Haswell mit 4 Kernen, 64 GB Hauptspeicher und einer durchschnittlichen Festplattengeschwindigkeit von 350 MB/s. Ein VPS diente zusätzlich als zentrale Koordinationseinheit. Die verwendete Linux-Distribution ist CentOS 7.

Hadoop in der Version 2.7.2, inklusive Hadoop Distributed File System (HDFS) und Yet Another Resource Negotiator (YARN), bildet die Grundlage für die Berechnungen auf den Plattformen Flink und Spark. Das HDFS ermöglicht den Zugriff und die verteilte Speicherung von folgenden Testdaten:

- Als hochvernetzte Menge von Dokumenten wurde statt eines Web-Ausschnitts ein Ausschnitt des Twitter-Follower-Graphen gewählt: der Umfang war hier 26 GB.
- Das errechnete PageRank-Ergebnis zu diesem Graphen umfasste 1,15 GB.
- Die zu untersuchenden Web-Log-Einträge hatten einen Umfang von 4,29 GB.

Die Konfigurationsparameter des HDFS können der nachfolgenden Tabelle entnommen werden. Um einen besseren Vergleich der Messergebnisse herstellen zu können, wurde die Konfiguration an den Ausgangsparametern angepasst.

[1]Rostock Massive Data Research Facility, ein Großgerät für datengetriebene Forschung, das vier Lehrstühle im Institut für Informatik der Universität Rostock gemeinsam betreiben.

Parameter	Wert
Speicherblockgröße	256 MB
Heapsize Task Executor	1024 MB
Heapsize History Server	1024 MB
Heapsize DataNode	1024 MB
Rackawareness	deaktiviert
Replikate	3/Block
Kompression	deaktiviert

Das Postgres-XL Cluster besteht aus 9 Koordinatoren und 9 Datenknoten. Jedem Koordinator kann somit genau ein Datenknoten zugewiesen werden. Die Konfigurationsparameter jeder Einheit sind wie folgt:

Parameter	Wert
Effective Cache Size	4 GB
Worker Memory	512 MB
Maintenance Memory	1 GB
Temporary Buffer	64 MB
Shared Buffer	1 GB
Segment Size	1 GB

An den Hardware-Parametern fällt auf, dass sie in Größenordnungen (Faktor 30) von den beim Stonebraker-Test verwendeten Parametern abweichen. Bei den in Rostock verwendeten Umgebungen differierten die Hadoop-, Spark- und Flink-Installationen und die Postgres-XL-Installation um den Faktor drei. Damit die Größenordnungen vergleichbar bleiben, haben wir die Testergebnisse um diese Faktoren skaliert, so dass die Ergebnisse bei Annahme eines linearen Zusammenhangs vergleichbar bleiben.

Wir stellen nun die drei untersuchten Tasks und die Ergebnisse für die drei getesteten Plattformen Flink, Spark und Postgres-XL vor. Hadoop wurde als Basis für Flink und Spark genutzt, nicht jedoch für eigenständige Tests.

6. TESTFÄLLE

Für die Evaluierung der drei Plattformen wurden die Testfälle Grep Task und Join Task des Stonebraker-Vergleichs ausgewählt. Zusätzlich wurde als einfacher Mining-Algorithmus ein Clustering-Verfahren (k-Means) umgesetzt. Wir beschreiben nun die Testfälle und die Ergebnisse auf den drei Systemen.

Grep Task. Diese Aufgabe orientiert sich an dem Original-Google-Map-Reduce-Grep-Task. Dabei sollte die Häufigkeit des Vorkommens einer bestimmte Zeichenkette in dem 26 GB großen Twitter-Follower-Graphen ohne Sortierung und ohne Nutzung eines Indexes ermittelt werden. Im Beispiel werden im Twitter-Follower-Graphen die Zeilen der relationalen Darstellung extrahiert, die eine „4“ enthalten. Die Anzahl dieser Vorkommen im Graphen soll dann ausgegeben werden.

Twitter-Follow-Graph

ID1	ID2
12343454	86968792
29656457	94665834
37695979	81632765

Anzahl Zeilen

die „4“ enthalten

Anzahl
2

Die folgende Tabelle enthält die durchschnittliche Laufzeit des Tests je System:

Flink	Spark	Postgres-XL
100 s	53 s	121 s

Postgres-XL als zeilenorientiertes paralleles DBMS hat hier die schlechteste Laufzeit. Die Unterschiede zwischen Flink und Postgres-XL sind aber nicht so deutlich wie erwartet. Die beste Laufzeit hat Spark.

Join Task. Diese Aufgabe entspricht dem Join Task aus dem Stonebraker-Artikel [11]. Dabei wurde zunächst die IP-Adresse ermittelt, welche die meisten Twitter-Konten in einem bestimmten Zeitraum besucht hat. Anschließend wurden die Zeilen der Weblog-Tabelle selektiert, welche diese IP-Adresse enthalten, um diese mit der PageRank-Tabelle über die ID zu verbinden (natürlicher Verbund). Zum Schluss wurden die PageRank-Werte zu einem Durchschnitt aggregiert. In diesem Fall wurde keine Sortierung benutzt.

PageRank

ID	PageRank
1	0,0016546
2	0,0857657
3	0,4534646

$\bowtie_{\text{ID}}$

Weblog

ID	IP	t
1	10	150
2	08	10
3	10	30

$\Rightarrow$

$\text{Ø}_{IP=10}$ PageRank
0,2275596

Die folgende Tabelle enthält die durchschnittliche Laufzeit des Tests je System:

Flink	Spark	Postgres-XL
121 s	140 s	27 s

Diese Aufgabe eignet sich durch die Verknüpfung von zwei Datenbeständen sehr gut für SQL-DBMS, auch wenn diese nur in einer zeilenorientierten Architektur vorliegen. Flink überholt nun Spark, fällt aber deutlich hinter Postgres-XL zurück. Die Ergebnisse korrespondieren mit den 2010 von Stonebraker veröffentlichten Ergebnissen (Hadoop fiel dort gegen beide DBMS-Lösungen deutlich zurück).

Sicher stellt diese Join-Task ein Heimspiel für die SQL-DBMS wie das von uns verwendete Postgres-XL dar. Allerdings ist ein solches Szenario auch in Data-Science- und Big-Data-Analytics-Anwendungen nicht unüblich. So werden im Bereich des Forschungsdatenmanagements und ihrer Auswerte-Prozeduren und -Workflows Daten verschiedener Messreihen, Projekte und Abteilungen miteinander verknüpft sowie auszuwertende Daten mit Metadaten und weiteren beschreibenden Daten aus anderen Datenbeständen kombiniert (siehe etwa [3]).

k-Means. Diese Aufgabe ist nicht Bestandteil des Stonebraker-Artikels [11], wurde aber in die Tests aufgenommen, weil weder der Grep Task noch der Join Task die Komplexität der in PArADISE umzusetzenden Machine-Learning-Algorithmen (ML) aufweisen. Eine Umsetzung von „echten“ ML-Algorithmen wie dem im PArADISE-Projekt auch verwendeten Hidden-Markov-Modell (siehe [9]) erwies sich für das 14 Wochen andauernde, in der Vorlesungsperiode stattfindende Projekt als zu aufwendig. Daher wurde mit k-Means ein Clustering-Algorithmus als einfacher Vertreter von Data-Mining-Techniken ausgewählt.

Wir stellen hier nun kurz das Prinzip von k-Means dar. Für genauere Informationen verweisen wir etwa auf [5]. Die folgende Erklärung bezieht sich jeweils auf die folgende graphische Darstellung der Datenpunkte und Cluster.

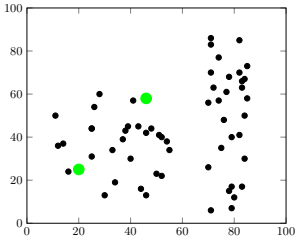
1. Es werden k beliebige Datenpunkte als initiale Clusterzentren (grün) gewählt.

2. Anschließend werden die Distanzen von jedem Datenpunkt zu jedem Clusterzentrum berechnen und der Datenpunkt dem Zentrum minimaler Distanz zugeordnet. Dadurch werden k Cluster (rot und blau) gebildet.

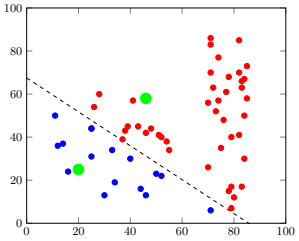
3. In jedem Cluster wird der Durchschnitt aller enthaltener Punkte als neues Clusterzentrum (grün) gewählt.

4. Die Schritte (2) und (3) werden so lange wiederholt, bis sich die Cluster nicht mehr verändern oder die Iterationsobergrenze erreicht ist. Im Folgenden wurde eine Grenze von 10 Wiederholungen verwendet.

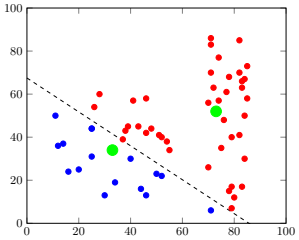
1: Beliebige Punkte als Clusterzentren wählen



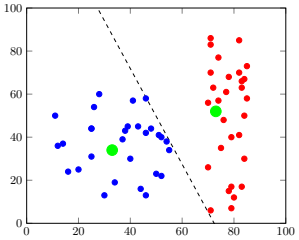
2: Distanzen ermitteln und Cluster bilden



3: Neue Zentren berechnen und Punkt minimaler Distanz wählen



4: Distanzen ermitteln und Cluster bilden



Aus den Daten des Twitter-Follower-Graphen wurde für jede enthaltene Konto-ID die Anzahl der Follower und die Anzahl der Pursuer ermittelt. Über die Menge dieser Datenpunkte wurde dann der k-Means-Algorithmus mit $k = 3$ durchgeführt.

In dieser Aufgabe konnte das parallele Datenbanksystem Postgres-XL wiederum Flink und Spark überflügeln. Die Ergebnisse entnehme man folgender Tabelle:

Flink	Spark	Postgres-XL
705 s	917 s	335 s

Die Ergebnisse entsprechen von der Tendenz her dem des Join Task, auch wenn dort die Abstände zwischen DBMS-Lösung und den Big-Data-Frameworks noch deutlicher waren.

7. ZUSAMMENFASSUNG UND AUSBLICK

Mit diesem Beitrag wollten wir nicht nur die Ergebnisse eines Artikels von Stonebraker [11] nachvollziehen, sondern die Szenarien (Tasks), Plattformen und Hardware-Umgebungen auf die Anforderungen im PArADISE-Projekt hin anpassen sowie gerade die Plattformen auf den heutige Stand aktualisieren.

Während Stonebraker im Jahre 2010 Hadoop, ein unbekanntes zeilenorientiertes DBMS und Vertica als Vertreter spaltenorientierter, paralleler DBMS verglich, haben wir als Plattformen Spark und Flink (sowie Hadoop als grundlegendes System) und als Vertreter zeilenorientierter, paralleler DBMS Postgres-XL ausgewählt.

Bei den Szenarien haben wir ein einfaches Data-Mining-Szenario (Clustering mit k-Means) ergänzt.

Die Tests bestätigen die Tendenz des Stonebraker-Tests: parallele DBMS, selbst in einer eher unpassenden zeilenorientierten Architektur, können im eher Information-Retrieval-artigen Grep Task zumindest größenordnungsmäßig mithalten, hängen aber die Big-Data-Analytics-Plattformen Spark und Flink bei komplexeren Aufgaben (Join Task und k-Means) deutlich ab.

Nimmt man die Ergebnisse von Stonebraker als das 1:0 für parallele DBMS, so konnten die Rostocker Tests nun auf 2:0 erhöhen. Natürlich ist dieses Ergebnis begünstigt durch das Heimspiel, das die Postgres-XL-Gruppe hier absolvieren konnte: zwar war der konkrete Aufwand im Projekt zwischen den Plattformen vergleichbar, allerdings waren die Erfahrungen bei den studentischen Projektteilnehmern in den DBMS-bezogenen Implementierungs- und Tuning-Aspekten deutlich höher als in den neueren Plattformen wie Spark und Flink. Ein weiteres Gegentor konnte nur verhindert werden, weil einige Kriterien wie die Skalierbarkeit ausgeblendet wurden: hier hätten Spark und Flink bei einer Veränderung der Hardware-Umgebung (Erhöhung der Knotenanzahl) gegenüber dem Uminstallationsaufwand bei Postgres-XL einen klaren Vorteil gehabt.

Für einen Heimsieg hoffen wir aber in Zukunft trotzdem auf weitere Tore für die SQL-DBMS-basierten Lösungen, denn die vorgenommenen Tests können nur ein Anfang sein und müssen in folgenden Aspekten erweitert werden:

- Das RMDRF-Großgerät lief derzeit noch in einer fixierten, sehr kleinen Konfiguration (drei Knoten). Hier werden wir in Zukunft die Konfiguration verändern, um Auswirkungen der Hardware-Konfiguration erkennen zu können.
- Bei den Tasks werden wir zusätzliche Mining-Algorithmen und Algorithmen Maschinellen Lernens mit aufnehmen und auf den verschiedenen Plattformen implementieren.
- Bei den Systemen fehlt uns bisher ein Vertreter von spaltenorientierten, parallelen DBMS. Als Ersatz für das von Stonebraker verwendete Vertica haben wir bereits erste Tests auf Actian Vector (früher VectorWise, siehe etwa [13]) durchgeführt, die vielversprechend sind. Vector gibt es auch in einer parallelen Variante als VectorH (Vector in Hadoop).

Die Hoffnung ist, dass sich auch weiterhin DBMS-Lösungen mit SQL als Schnittstelle als konkurrenzfähige Alternative zu MapReduce-Programmierparadigmen und anderen spezialisierten Big-Data-Analytics-Umgebungen erweisen, damit wir die vielfältigen Vorteile einer solchen Lösung in Bezug auf formalisierbare und automatisierbare Anfragetransformationen ohne großen Performance-Verlust ausnutzen können. Diese Anfragetransformationen benötigen wir, um weitere Kernziele des PArADISE-Projektes zu verwirklichen: die Wahrung der Privatsphäre der Nutzer von Assistenzsystemen durch datensparsame Auswertung von Big Data, das Provenance Management zur Ermittlung von Ursachen bei fehlerhaften Modellbildungen, und die Nachhaltigkeit der Analyseprogramme im Kontext einer Informationssystem-Infrastruktur beim Anbieter des Assistenzsystems. Letzteres ist durch die Verwendung von SQL-Basisoperationen als „intergalactic dataspeak“ gegeben.

Literatur

[1] 2ndQuadrant. “Postgres-XL official website”. In: (2018). URL: https://www.postgres-xl.org (besucht am 07. 03. 2018).

[2] Martín Abadi u. a. “TensorFlow: A System for Large-Scale Machine Learning”. In: *OSDI*. USENIX Association, 2016, S. 265–283.

[3] Ilvio Bruder u. a. “Daten wie Sand am Meer - Datenerhebung, -strukturierung, -management und Data Provenance für die Ostseeforschung”. In: *Datenbank-Spektrum* 17.2 (2017), S. 183–196.

[4] Paris Carbone u. a. “Apache Flink™: Stream and Batch Processing in a Single Engine”. In: *IEEE Data Eng. Bull.* 38.4 (2015), S. 28–38. URL: http://sites.computer.org/debull/A15dec/p28.pdf.

[5] Jürgen Cleve und Uwe Lämmel. *Data Mining – 2. Auflage*. De Gruyter, 2016.

[6] Jeffrey Dean und Sanjay Ghemawat. “MapReduce: a flexible data processing tool”. In: *Commununications of the ACM* 53.1 (2010), S. 72–77.

[7] Hannes Grunert und Andreas Heuer. “Datenschutz im PArADISE”. In: *Datenbank-Spektrum* 16.2 (2016), S. 107–117. DOI: 10.1007/s13222-016-0216-7. URL: https://doi.org/10.1007/s13222-016-0216-7.

[8] Andreas Heuer. “METIS in PArADISE: Provenance Management bei der Auswertung von Sensordatenmengen für die Entwicklung von Assistenzsystemen”. In: *BTW Workshops*. Bd. 242. LNI. GI, 2015, S. 131–136.

[9] Dennis Marten und Andreas Heuer. “Machine Learning on Large Databases: Transforming Hidden Markov Models to SQL Statements”. In: *Open Journal of Databases (OJDB)* 4.1 (2017), S. 22–42. ISSN: 2199-3459. URL: https://www.ronpub.com/ojdb/OJDB_2017v4i1n02_Marten.html.

[10] Derek Gordon Murray u. a. “Naiad: a timely dataflow system”. In: *SOSP*. ACM, 2013, S. 439–455.

[11] Michael Stonebraker u. a. “MapReduce and parallel DBMSs: friends or foes?” In: *Communications of the ACM* 53.1 (2010), S. 64–71.

[12] Matei Zaharia u. a. “Apache Spark: A Unified Engine for Big Data Processing”. In: *Communications of the ACM* 59.11 (Okt. 2016), S. 56–65. ISSN: 0001-0782. DOI: 10.1145/2934664. URL: http://doi.acm.org/10.1145/2934664.

[13] Marcin Zukowski und Peter A. Boncz. “Vectorwise: Beyond Column Stores”. In: *IEEE Data Eng. Bull.* 35.1 (2012), S. 21–27.