FernUniversität in Hagen

Fakultät für Mathematik und Informatik

Seminar 01912 / 19912

Skalierbare verteilte Datenanalyse

Sommersemester 2017

Prof. Dr. Ralf Hartmut Güting

Dr. Fabio Valdés

Einführung

Die Verarbeitung enorm großer Datenmengen („Big Data Management“) ist ein wichtiges Thema der letzten Jahre. Solche Datenmengen entstehen etwa durch globale Internet-Anwendungen wie Google, Facebook, Twitter oder Amazon; durch das ständige Erzeugen und Protokollieren von Daten durch Smartphones (Fotos, Nachrichtenaustausch, Sport-Apps, Aufzeichnen von Positionsdaten); aber auch durch wissenschaftliche Beobachtungen, Experimente oder Simulationen.

Der grundlegende Ansatz, um dieser Datenmengen Herr zu werden, ist fehlertolerante Parallelisierung. Das heißt, man versucht, solche Aufgaben von Tausenden Rechnern gemeinsam ausführen zu lassen; da dabei Ausfälle unausweichlich sind, entwickelt man Techniken, die Ausfälle ohne größere Verzögerung verkraften. Eine bahnbrechende Technik dieser Art ist das von Google 2004 entwickelte MapReduce-Paradigma, bei dem Programmierer nur gewisse Funktionen schreiben, die dann fehlertolerant auf beliebig vielen Rechnern ausgeführt werden. MapReduce ist als OpenSource-Implementierung Hadoop frei verfügbar.

Thema des Seminars sind Techniken und Systeme für hochskalierbare verteilte Datenverarbeitung mit den Schwerpunkten (i) Verarbeitung im Hauptspeicher und (ii) Darstellung und Analyse von Graphen. Während Hadoop dateibasiert arbeitet, ermöglichen Systeme wie z.B. Spark fehlertolerante Verarbeitung im Hauptspeicher. Graphen spielen eine große Rolle: das World-Wide-Web ist ein Graph, soziale Netze sind Graphen, usw. Systeme wie z.B. Pregel erlauben die skalierbare parallele Manipulation aller Knoten eines Graphen in einem Schritt. Natürlich treten Verwendung von Hauptspeicher und Manipulation von Graphen auch kombiniert auf.

Zugang zu Quellen

Die im Literaturverzeichnis erwähnten Paper finden Sie auf der Internetseite des Seminars im Abschnitt Literatur.

Der Zugriff auf die Seite ist aus urheberrechtlichen Gründen auf Seminarteilnehmer beschränkt.

Themenauswahl

Bitte senden Sie bis zum 12.03.2017 eine Liste mit Ihren Prioritäten für die einzelnen Themen an `fabio.valdes@fernuni-hagen.de`. Die Liste soll 17 Einträge enthalten. Vergeben Sie Priorität 1 für das Thema, welches Sie am liebsten bearbeiten möchten. Markieren Sie das Thema, welches Sie auf keinen Fall bearbeiten möchten, mit Priorität 17. Dazwischenliegende Zahlen werden entsprechend gewertet. Alle Zahlen von 1 bis 17 sollen genau einmal vergeben werden.

Bei der Verteilung der Themen werden Ihre Prioritäten so weit wie möglich berücksichtigt.

Themen

Die Seminarthemen sind in drei unterschiedlich große Themenbereiche gegliedert. Jeder der Abschnitte beschreibt genau ein Vortragsthema.

1 Grundlegende Algorithmen und Datenstrukturen

1.1 MapReduce

Das von Google entwickelte MapReduce-Framework [DG04, DG08, DG10] dient dazu, die Verarbeitung sehr großer Datenmengen effizient auf zahlreiche Rechner zu verteilen. Dieser Vortrag beleuchtet die technischen Hintergründe des MapReduce-Verfahrens. MapReduce ist äußerst populär geworden, vor allem auch durch die frei verfügbare Implementierung Hadoop.

1.2 PageRank

An der Universität Stanford entwickelten die Google-Gründer Page und Brin Ende der 1990er Jahre einen Algorithmus namens PageRank [PBMW99], der auf Basis der Graphstruktur des Internets eine globale Rangfolge für alle Internetseiten berechnet, unabhängig von deren Inhalt. So können Suchergebnisse nach Relevanz sortiert werden.

1.3 Resilient Distributed Datasets

Es existieren zwar bereits fehlertolerante Systeme, die dem Nutzer einen komfortablen Zugriff auf die Hardware eines Clusters ermöglichen. Allerdings sind diese nicht in der Lage, Zwischenergebnisse verschiedener Berechnungen wiederzuverwenden, was in verschiedenen Anwendungen wie iterativem maschinellen Lernen oder Graphalgorithmen von Bedeutung ist. Zu diesem Zweck wurden Resilient Distributed Datasets (RDDs) [ZCD+12] entwickelt. Diese fehlertoleranten parallelen Datenstrukturen unterstützen die Ausführung von iterativen verteilen Algorithmen und werden im System Spark verwendet.

2 Hauptspeicherbasierte Datenbanksysteme

2.1 RAMClouds

Angesichts der Tatsache, dass die Entwicklung hinsichtlich der Übertragungsgeschwindigkeit von Festplatten bei weitem nicht so schnell voranschreitet wie die Vergrößerung der Kapazität, wurde ein verteilter Ansatz entwickelt, der Daten in erster Linie im Hauptspeicher hält und verarbeitet. RAMClouds [OAE+09, OGG+15] unterstützen die Ausführung sowohl einzelner großer als auch zahlreicher kleiner Anwendungen sowie von Anwendungen mit wachsender Datenmenge.

2.2 Shark

Das fehlertolerante System Shark [ELX+12, XRZ+13] verbindet auf effiziente Weise zwei Anwendungen miteinander, die auf grundsätzlich unterschiedlichen Anforderungen hinsichtlich der Unterteilung des Hauptspeichers beruhen. Sowohl SQL-Anfragen als auch Algorithmen für

maschinelles Lernen auf großen Datenmengen lassen sich in einer einzelnen fehlertoleranten Umgebung bis zu 100mal schneller als mit Apache Hive bzw. mit Hadoop ausführen.

2.3 Spark

Typische Anwendungen im Bereich Big Data beinhalten verschiedene Anforderungen wie MapReduce, SQL-ähnliche Anfragen oder maschinelles Lernen. Um zu vermeiden, dass für jede Aufgabe ein anderes System verwendet werden muss, wurde an der Universität in Berkeley seit 2009 das Projekt Spark [ZXW+16, VYL+16, ADD+15, AXL+15] entwickelt. Es führt diese unterschiedlichen Bereiche zusammen, wodurch Effizienzgewinne entstehen, und erleichtert die Anwendungsentwicklung durch eine einheitliche API.

2.4 Flink

Die Verarbeitungsvorgänge von Datenströmen und von statischen Daten werden traditionell getrennt betrachtet. Da heutzutage ein großer Teil der erzeugten Daten (Logdaten von Internetseiten, Anwendungen oder Datenbanktransaktionen, Aufzeichnungen von Sensoren) kontinuierlich produziert werden, wurde mit Flink [CKE+15] ein System entwickelt, das beide Betrachtungsweisen kombiniert und sowohl Daten in Echtzeit als auch historische Daten effizient verarbeitet. Flink betrachtet Stapelverarbeitungsprogramme als Spezialfall von strombasierten Anwendungen mit einem endlichen Strom. Einige auf Flink basierende Forschungsprojekte werden in [RTKM16] vorgestellt.

3 Graphbasierte Datenbanksysteme

3.1 Pregel

Das von Google entwickelte Framework Pregel [MAB+10] ermöglicht dem Nutzer die komfortable Verarbeitung großer Graphen mit Hilfe eines knotenbasierten Ansatzes. Pregel interpretiert Graphalgorithmen als Folgen von Iterationen. In jeder dieser Iterationen kann ein Knoten Nachrichten erhalten, die in der vorherigen Iteration an ihn gesendet wurden, Nachrichten an andere Knoten verschicken sowie seinen eigenen Zustand und die Zustände seiner ausgehenden Kanten verändern.

3.2 Piccolo

Piccolo [PL10] ist ein datenzentrisches Programmiermodell, das die Implementierung von parallelen Hauptspeicheranwendungen erleichtert. Die verteilte Verfügbarkeit von Zwischenzuständen, die für Berechnungen wie k-Means oder PageRank erforderlich ist, kann so deutlich schneller als mit Hadoop realisiert werden.

3.3 HaLoop

MapReduce erfreut sich zwar großer Beliebtheit, bietet jedoch keine integrierte Unterstützung für Analysen oder Anwendungen, die iterative Berechnungen erfordern, beispielsweise PageRank, Graphanalyse in neuronalen oder sozialen Netzen oder rekursive relationale Anfragen. Das System HaLoop [BHBE10, BHBE12] schafft hier Abhilfe. Es handelt sich dabei um eine modifizierte Version des MapReduce-Frameworks auf Basis seiner Implementierung Hadoop

für genau diese Art von Anwendungen, die ein neues Programmiermodell sowie einige Optimierungen enthält.

3.4 PowerGraph

In Graphen, die aus Anwendungen wie sozialen Netzwerken abgeleitet sind, sind die Grade der einzelnen Knoten häufig sehr unterschiedlich, d.h., in vielen Fällen ist eine kleine Teilmenge der Knoten mit einer sehr großen Teilmenge verbunden, was die Partitionierung solcher Graphen erschwert und Anwendungen verlangsamen kann. Im Gegensatz zu Systemen wie Pregel ist PowerGraph [GLG+12] darauf spezialisiert, Knoten mit hohem Grad aufzuteilen und die Verarbeitung von Graphen aus realen Anwendungen zu parallelisieren.

3.5 Neo4j

Einer der Nachteile relationaler Datenbankmanagementsysteme ist die Festlegung auf ein bestimmtes Schema der Datenspeicherung. Der Einsatz graphbasierter Datenbanken wie z.B. Neo4j [Mil13] kann sinnvoll sein, wenn das Hauptaugenmerk auf den Beziehungen zwischen verschiedenen Entitäten liegt. So können Vorhersagemodelle konstruiert und Korrelationen sowie Muster erkannt werden. Anwendungsgebiete für graphbasierte Datenbanksysteme sind soziale Netzwerke [CQPA13], Empfehlungsdienste sowie Bioinformatik.

3.6 GraphX

Im Gegensatz zu anderen graphbasierten Datenbanksystemen zur effizienten Ausführung von Graphalgorithmen liegt das Hauptaugenmerk von GraphX [XGFS13, XCD+14, GXD+14] auf der gesamten parallelen Bearbeitung von Graphen, die auch deren Konstruktion und Transformation – häufig ähnlich komplex wie die anschließende Berechnung – beinhaltet. GraphX wurde auf Basis von Spark entwickelt und bietet einige Operationen zur parallelen Verarbeitung von Graphen sowie Techniken zu deren optimierter Ausführung.

3.7 G*

Mit Hilfe des parallelen Systems G* [LBO+15] lassen sich Folgen von Graphen speichern und verarbeiten. G* komprimiert diese dynamischen Daten mit Hilfe von Übereinstimmungen zwischen einzelnen Graphen und speichert diese duplikatfrei auf zahlreichen Rechnern. Aufgrund der Ähnlichkeit benachbarter Graphen, die beispielsweise die Zustände eines sozialen Netzwerks an zwei aufeinanderfolgenden Tagen repräsentieren können, werden so Effizienzgewinne bezüglich Speicherplatz sowie Lese- und Schreiboperationen erzielt.

3.8 OLAP auf Graphen

OLAP (On-Line Analytical Processing) ist ein zentraler Begriff im Bereich der Datenanalyse. In einem herkömmlichen Datenwürfel können verschiedene Einträge mit Hilfe bestimmter Funktionen aggregiert werden. Dies kann bei Bedarf auch über mehrere Dimensionen und Hierarchieebenen geschehen. Angesichts der heutigen Heterogenität vieler Informationen, deren Verbindungen untereinander häufig relevant sind, wird in [CYZ+08] ein Framework für

OLAP auf Graphen vorgestellt. Ergänzend dazu bietet [THP08] zwei Operationen zur effizienten Aggregierung auf großen Graphen. Außerdem wurde mit Graph Cube [ZLXH11] ein Modell für die multidimensionale Datenanalyse implementiert.

3.9 Distributed Graph Cube

Bei großen Datenmengen zeigt Graph Cube Schwächen durch das sequentielle Lesen der Ausgangsdaten sowie die Beschränkung auf einen zentralisierten Hauptspeicher. Daher wurde mit Distributed Graph Cube [DGS13] ein frei verfügbarer verteilter Ansatz für OLAP auf großen Graphdaten auf Basis des MapReduce-Modells entwickelt und aufsetzend auf Spark implementiert. Ein weiteres paralleles OLAP-System für attributierte Graphen ist Pagrol [WFW+14], das zahlreiche Optimierungsstrategien anwendet.

3.10 Benchmarks

Benchmarks sind unerlässlich, um verschiedene Systeme hinsichtlich ihrer Leistungsfähigkeit miteinander vergleichen zu können. Das sogenannte Linked Data Benchmark Council (LDBC), eine Kooperation von Industrie und Forschung, hat zwei Benchmarks für graphbasierte Datenbanksysteme entwickelt [EAL+15, IHN+16].

Literatur

[ADD+15] Armbrust, M., Das, T., Davidson, A., Ghodsi, A., Or, A., Rosen, J., Stoica, I., Wendell, P., Xin, R., and Zaharia, M. Scaling Spark in the real world: Performance and usability. *PVLDB*, 8(12):1840–1843, 2015.

[AXL+15] Armbrust, M., Xin, R.S., Lian, C., Huai, Y., Liu, D., Bradley, J.K., Meng, X., Kaftan, T., Franklin, M.J., Ghodsi, A., and Zaharia, M. Spark SQL: Relational data processing in Spark. In *ACM SIGMOD*, pages 1383–1394. 2015.

[BHBE10] Bu, Y., Howe, B., Balazinska, M., and Ernst, M.D. HaLoop: Efficient iterative data processing on large clusters. *PVLDB*, 3(1):285–296, 2010.

[BHBE12] Bu, Y., Howe, B., Balazinska, M., and Ernst, M.D. The HaLoop approach to large-scale iterative data analysis. *VLDB J.*, 21(2):169–190, 2012.

[CKE+15] Carbone, P., Katsifodimos, A., Ewen, S., Markl, V., Haridi, S., and Tzoumas, K. Apache Flink: Stream and batch processing in a single engine. *IEEE Data Eng. Bull.*, 38(4):28–38, 2015.

[CQPA13] Cattuto, C., Quaggiotto, M., Panisson, A., and Averbuch, A. Time-varying social networks in a graph database: A Neo4j use case. In *GRADES*, pages 11:1–11:6. 2013.

[CYZ+08] Chen, C., Yan, X., Zhu, F., Han, J., and Yu, P.S. Graph OLAP: Towards online analytical processing on graphs. In *ICDM*, pages 103–112. 2008.

[DG04] Dean, J. and Ghemawat, S. MapReduce: Simplified data processing on large clusters. In *OSDI*, pages 137–150. 2004.

[DG08] Dean, J. and Ghemawat, S. MapReduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008.

[DG10] Dean, J. and Ghemawat, S. MapReduce: A flexible data processing tool. *Commun. ACM*, 53(1):72–77, 2010.

[DGS13] Denis, B., Ghrab, A., and Skhiri, S. A distributed approach for graph-oriented multidimensional analysis. In *International Conference on Big Data*, pages 9–16. 2013.

[EAL+15] Erling, O., Averbuch, A., Larriba-Pey, J., Chafi, H., Gubichev, A., Prat-Pérez, A., Pham, M., and Boncz, P.A. The LDBC social network benchmark: Interactive workload. In *ACM SIGMOD*, pages 619–630. 2015.

[ELX+12] Engle, C., Lupher, A., Xin, R., Zaharia, M., Franklin, M.J., Shenker, S., and Stoica, I. Shark: Fast data analysis using coarse-grained distributed memory. In *ACM SIGMOD*, pages 689–692. 2012.

[GLG+12] Gonzalez, J.E., Low, Y., Gu, H., Bickson, D., and Guestrin, C. PowerGraph: Distributed graph-parallel computation on natural graphs. In *OSDI*, pages 17–30. 2012.

[GXD+14] Gonzalez, J.E., Xin, R.S., Dave, A., Crankshaw, D., Franklin, M.J., and Stoica, I. GraphX: Graph processing in a distributed dataflow framework. In *OSDI*, pages 599–613. 2014.

[IHN+16] Iosup, A., Hegeman, T., Ngai, W.L., Heldens, S., Prat-Pérez, A., Manhardt, T., Chafi, H., Capota, M., Sundaram, N., Anderson, M.J., Tanase, I.G., Xia, Y., Nai, L., and Boncz, P.A. LDBC Graphalytics: A benchmark for large-scale graph analysis on parallel and distributed platforms. *PVLDB*, 9(13):1317–1328, 2016.

[LBO+15] Labouseur, A.G., Birnbaum, J., Olsen, P.W., Spillane, S.R., Vijayan, J., Hwang, J., and Han, W. The G* graph database: Efficiently managing large distributed dynamic graphs. *Distributed and Parallel Databases*, 33(4):479–514, 2015.

[MAB+10] Malewicz, G., Austern, M.H., Bik, A.J.C., Dehnert, J.C., Horn, I., Leiser, N., and Czajkowski, G. Pregel: A system for large-scale graph processing. In *ACM SIGMOD*, pages 135–146. 2010.

[Mil13] Miller, J.J. Graph database applications and concepts with Neo4j. In *Proceedings of the Southern Association for Information Systems Conference*, pages 141–147. 2013.

[OAE+09] Ousterhout, J.K., Agrawal, P., Erickson, D., Kozyrakis, C., Leverich, J., Mazières, D., Mitra, S., Narayanan, A., Parulkar, G.M., Rosenblum, M., Rumble, S.M., Stratmann, E., and Stutsman, R. The case for RAMClouds: Scalable high-performance storage entirely in DRAM. *Operating Systems Review*, 43(4):92–105, 2009.

[OGG+15] Ousterhout, J., Gopalan, A., Gupta, A., Kejriwal, A., Lee, C., Montazeri, B., Ongaro, D., Park, S.J., Qin, H., Rosenblum, M., Rumble, S., Stutsman, R., and Yang, S. The RAMCloud storage system. *ACM Trans. Comput. Syst.*, 33(3):7:1–7:55, 2015.

[PBMW99] Page, L., Brin, S., Motwani, R., and Winograd, T. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[PL10] Power, R. and Li, J. Piccolo: Building fast, distributed programs with partitioned tables. In *OSDI*, pages 293–306. 2010.

[RTKM16] Rabl, T., Traub, J., Katsifodimos, A., and Markl, V. Apache Flink in current research. *it - Information Technology*, 58(4):157–165, 2016.

[THP08] Tian, Y., Hankins, R.A., and Patel, J.M. Efficient aggregation for graph summarization. In *ACM SIGMOD*, pages 567–580. 2008.

[VYL+16] Venkataraman, S., Yang, Z., Liu, D., Liang, E., Falaki, H., Meng, X., Xin, R., Ghodsi, A., Franklin, M.J., Stoica, I., and Zaharia, M. SparkR: Scaling R programs with Spark. In *ACM SIGMOD*, pages 1099–1104. 2016.

[WFW+14] Wang, Z., Fan, Q., Wang, H., Tan, K., Agrawal, D., and El Abbadi, A. Pagrol: Parallel graph OLAP over large-scale attributed graphs. In *IEEE ICDE*, pages 496–507. 2014.

[XCD+14] Xin, R.S., Crankshaw, D., Dave, A., Gonzalez, J.E., Franklin, M.J., and Stoica, I. GraphX: Unifying data-parallel and graph-parallel analytics. *CoRR*, abs/1402.2394, 2014.

[XGFS13] Xin, R.S., Gonzalez, J.E., Franklin, M.J., and Stoica, I. GraphX: A resilient distributed graph system on Spark. In *GRADES*, pages 2:1–2:6. 2013.

[XRZ+13] Xin, R.S., Rosen, J., Zaharia, M., Franklin, M.J., Shenker, S., and Stoica, I. Shark: SQL and rich analytics at scale. In *ACM SIGMOD*, pages 13–24. 2013.

[ZCD+12] Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauly, M., Franklin, M.J., Shenker, S., and Stoica, I. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *NSDI*, pages 15–28. 2012.

[ZLXH11] Zhao, P., Li, X., Xin, D., and Han, J. Graph Cube: On warehousing and OLAP multidimensional networks. In *ACM SIGMOD*, pages 853–864. 2011.

[ZXW+16] Zaharia, M., Xin, R.S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M.J., Ghodsi, A., Gonzalez, J., Shenker, S., and Stoica, I. Apache Spark: A unified engine for big data processing. *Commun. ACM*, 59(11):56–65, 2016.