Abstract - Wissenschaftliche Datenbanken

-- Oberseminar: “Datenbanksysteme - aktuelle Trends” --

Alexander Twrdik, 62458, 13INM

Motivation

Wissenschaft kann im Allgeminen als die Erweiterung von Wissen durch Forschung und die damit verbundene Lehre von neuen Erkenntnissen verstanden werden[1]. War man vor längerer Zeit möglicherweise noch in der Lage die Ergebnisse und Messungen seiner Experimente per Hand auf Papier festzuhalten, so ist dies in der heutigen Zeit nicht mehr möglich. Den Grund dafür kann man in der weiten Verbreitung von Computern sehen, die in großen Maßen für wissenschaftliche Berechnungen eingesetzt werden. Durch diese Tatsache ist jedoch der Umstand gegeben, dass große Datenmengen in kurzer Zeit entstehen können, wobei der zeitliche Aspekt von der “Rechenkraft” der eingesetzten Maschinen abhängig ist. Solche Datenmengen bewegen sich beispielsweise bei Astro-Daten im Petabyte-Bereich[2]. Zum Vergleich sei der größte Hadoop-Cluster von Facebook herangezogen, dessen *Hadoop Distributed File System (HDFS)* ein Speichervolumen von 100 Petabyte aufweist[3]. Ein weiteres Beispiel für das Anfallen großer Datenmengen im wissenschaftlichen Umfeld, stellt der Teilchenbeschleuniger des CERN dar. Dieser hatte bis zum Jahr 2012 bereits einen Datenbestand von 200 Petabyte angehäuft[4]. Es ist an der Stelle natürlich einfach solche große Datenmengen zu erzeugen, jedoch stellt sich gleich im Anschluss die Frage, wie man diese verwalten und effizient verarbeiten kann.

Arbeiten mit wissenschaftlichen Daten

Durch die oben angesprochenen großen Datenmengen ist es wenig verwunderlich, dass man als Mensch ab einem bestimmten Zeitpunkt den Überblick über die angehäuften Informationen verliert, die gebündelt in sogenannten Daten-Archiven abgelegt werden. Damit das jedoch trotz Maschinenunterstützung nicht passiert, ist es notwendig die erhobenen Daten mit Zusatzinformationen, sogenannten Metadaten, zu bestücken. Solche Metadaten können dabei von Typinformationen bestimmter Messwerte bis zu ganzen Dokumenten, die etwaige Daten genauer erklären, reichen[5][6] und sind damit meist um ein vielfaches größer als die eigentlichen Daten, die gespeichert werden sollen. Metadaten können dabei grob in *Feststellungs-Metadaten*, *Hilfs-Metadaten* und *Herkunfts-Metadaten eingeteilt werden*. Feststellungs-Metadaten dienen in erster Linie dazu Informationen über ein Daten-Archiv zu offerieren, um somit grob filtern zu können, ob ein Daten-Archiv Daten beinhaltet, die benötigt werden. Hilfs-Metadaten sind hingegen dafür nötig, um eine Interpretation von Werten vornehmen zu können. Beispielsweise gibt diese Art von Metadaten diversen Zahlenwerten, die von Maschinen geliefert wurden, eine Bedeutung, indem z.B. Einheiten zugeordnet werden. Herkunfts-Metadaten sind wichtig wenn es darum geht den Ursprung von Informationen herauszufinden bzw. deren Entstehungsweg nachzuvollziehen. Dann ist es wichtig, wie Daten transformiert wurden und welche Eingabedaten an dem Prozess beteiligt gewesen sind. Solche Informationen werden meist als *Name-Wert-Paare* abgespeichert, wobei hier meist das Problem besteht, dass die Namen nicht immer einem bestimmten Vokabular entsprechen, welches allgemeingültig ist, sondern vielmehr von einzelnen Forschern festgelegt wird, die gemeinsam an einem Projekt arbeiten. Deshalb wurde versucht diverse standardisierte Formate zu entwerfen, die eben ein solches Vokabular bieten. Beispiele dafür sind beispielsweise die folgenden Formate: *DIF (Directory Interchange Format), DWC (Darwin Core) oder DDI (Data Documentation Initiative)*. Wie weiter oben bereits erwähnt, wird der Speicherbedarf pro Datensatz durch die Annotation mit Metadaten natürlich um ein vielfaches größer, wobei sich die Frage stellt, wie und wo man diese Daten speichern soll.

Speicherung von wissenschaftlichen Daten

Für die Speicherung von Daten kommen im Allgemeinen zwei verschiedene Varianten in Frage -- die Ablage der Daten in einem Filesystem oder in einer Datenbank. Beide Varianten haben dabei ihre Vor- und Nachteile. In einem Filesystem legt man die Daten als Dateien in einer bestimmten Ordnerstruktur ab, wobei der Einsatz von Meta-Daten in dieser Variante als schwierig einzustufen ist, da diese nicht unbedingt zentralisiert vorliegen muss. Weiterhin ist die Suche nach Informationen teuer, denn es müssen ganze Dateien durchsucht werden, wobei keine parallele Verarbeitung möglich ist und es auch keine spezielle Abfrage-Sprache gibt. Der Vorteil des Einsatzes eines Filesystems besteht darin, dass bereits bestehende Programme der Vergangenheit problemlos daran angebunden werden können. Grund dafür ist, dass Datenbanken damals noch nicht weit verbreitet waren und somit standardmäßig auf Filesysteme zurückgegriffen wurde. Datenbanken auf der anderen Seite besitzen all jene Fähigkeiten an denen es einem Filesystem mangelt. Sie ermöglichen eine parallele Verarbeitung, können Daten für schnelles Auffinden indexieren und besitzen mit SQL eine spezielle Abfrage-Sprace. Der Nachteil ist jedoch, dass es gängigen Datenbanksystemen an wissenschaftlichen Datentypen fehlt. Besonders Arrays werden bisher kaum bis garnicht unterstützt. Als weiteres Vergleichskriterium kann noch die Performance auf lange Sicht angeführt werden, wobei das Hauptaugenmerk dabei auf der Fragmentierung der Daten liegt. Nach dem Ergebnis einer Untersuchung ist dabei festzuhalten, dass die Fragmentierung eines Filesystems nach einer gewissen Zeit geringer ausfällt als bei einer Datenbank[7]. Das hat zur Folge, dass die Performance eines Filesystems, unter gewissen Voraussetzungen, nach einer gewissen Zeit besser ausfällt, als die einer Datenbank. Es kann demnach festgehalten werden, dass der Einsatz einer einzelnen Variante nicht dazu geeignet ist wissenschaftliche Daten in Kombination mit Metadaten zu speichern. Aus diesem Grunde wurden sogenannte *Data-Vaults* entwickelt, die eine Symbiose der beiden Varianten darstellen und somit Vorteile kombinieren und Nachteile kompensieren[8].

RasDaMan Array DBMS

Das RasDaMan *("Raster Data Manager")* Array DBMS stellt eine Art eines Data-Vaults dar, wie sie im vorherigen Abschnitt eingeführt wurden. Dieses Datenbank Management System wurde dabei extra für den wissenschaftlichen Bereich konzipiert. Zu diesem Zwecke setzt es auf bereits existierenden relationalen DBMS auf und erweitert diese um n-dimensionale Arrays als Datentyp, wie er in der Wissenschaft häufig zum Einsatz kommt. Die Arrays selber sind dabei in ihrer größe nicht limitiert und auch die Anzahl der Dimensionen ist nach oben hin offen. Zur Speicherung der Arrays greift dieses DBMS auf das darunter liegende relationale DBMS zurück und legt die Arrays dort als BLOB ab. Zuvor werden die Arrays jedoch noch durch sogenanntes *Tiling* in zusammenhängende und nicht überlappende Sub-Arrays geteilt. Durch diese Teilung können die Daten mit Hilfe von R+-Bäumen multi-indexiert werden, wodurch effektive Abfragen ermöglicht werden[9]. R+-Bäume stellen dabei eine Erweiterung von B-Bäumen in multiple Dimensionen dar[10]. Es ist demnach ersichtlich, dass das RasDaMan Array DBMS als eine Art Middleware zwischen einer Applikation und einem real existierenden relationalen DBMS fungiert. Durch die direkte Anbindung an die Applikation über eine Schnittstelle ist es sogar möglich ganze Objekte eines Programms zu serialisieren und zu persistieren, wodurch innerhalb eines Arrays komplexe Datentypen zum Einsatz kommen können. Dies würde bereits exisitierenden Programmen beim Umstieg auf diese Lösung helfen, die bereits solche komplexen Daten in einem Filesystem speichern.

Scientific Workflows

Um die Akquirierung, Verarbeitung, Speicherung und Annotation von wissenschaftlichen Daten zu vereinfachen bzw. zu automatisieren, wurden sogenannte Scientific Workflows entwickelt. Diese stellen den Wissenschaftlern eine grafische Oberfläche zur Verfügung, über die komplexe Workflows, wie sie beispielsweise beim *"Genome Sequencing"* vorkommen[11], erstellt und nachvollzogen werden können. Solche Workflows werden gemeinhin als Graphen dargestellt, bei denen die Knoten eine Aufgabe widerspiegeln.

Eine solche Aufgabe kann beispielsweise die Transformation oder Speicherung von Datensätzen sein. Die Kanten im Graph stellen meist den Transport von Daten (sogenannten Tokens) oder einen Kommunikationskanal im Allgemeinen dar. Workflows haben dabei drei unterschiedliche Arbeitsweisen. DAG *("Directed Acyclic Graph")* stellt die restriktivste Arbeitsweise dar, denn hier kann jeder Knoten nur 1x aufgerufen werden und auch nur dann, wenn alle ihm vorgelagerten Knoten ihre Ausführung bereits abgeschlossen haben. Bei SDF *("Synchronous Dataflow")* gibt es nur einen Thread für den gesamten Workflow. Das hat den Vorteil, dass der serielle Ablauf vor der Ausführung des Workflows berechnet werden kann und auch die Puffergröße für die Kommunikationskanäle limitiert ist, was besonders für Systeme mit beschränkten Ressourcen wichtig sein kann. PN *("Process Network")* bietet Möglichkeit jede Aufgabe eines Workflows als eigenständigen Thread laufen zu lassen, wobei die Implementierung entweder data-driven ("eager") oder demand-driven ("lazy") sein kann. Neben den eigentlichen Arbeitsweisen gibt es für solche Systeme auch gewisse Designs, vergleichbar mit Design-Patterns aus der Programmierung, die es erlauben gewisse Teile des Workflows wiederverwendbar zu machen, indem gewisse Design-Richtlinien eingehalten werden. Beispiele dafür sind beispielweise BioMoby, Wings und COMAD[11]. Durch Arbeitsweisen wie das PN wird natürlich auch die Parallelisierung gesamter Workflows oder Ausschnitte daraus ermöglicht, wobei zwischen Aufgabe-Parallelisierung, Pipelining und Daten-Parallelisierung unterschieden werden kann[12]. MapReduce kommt dabei als eine spezielle Form der Daten-Parallelisierung besonders im biologischen Bereich des "Genome Sequencing" zum Einsatz[11] und wurde ursprünglich von Google entwickelt, um Arbeiten auf großen Datenmengen auf verschiedene Knoten in einem Cluster aufzuteilen und die Kommunikation und das Datenmanagement hinter einer Schnittstelle zu verstecken[13]. Gerade durch die Parallelisierung und die Komplexität der Abläufe im wissenschaftlichen Umfeld, ist es von höchster Wichtigkeit zu wissen wie Daten entstanden sind und aus welchen anderen Daten diese entstanden sind. Diese Themen werden weitläufig unter dem Begriff Data-Lineage (auch Data-Provenance) zusammengefasst. Data-Lineage hat dabei mit *where-lineage* und *how-lineage* zwei

verschiedene Arten, wobei jede Art nochmals in die zwei Granularitäten *schema-level (grobkörnig)* und *instance-level (feinkörnig)* unterteilt ist[14]. Somit lassen sich folgende Fragen bezüglich der Herkunft von Daten beantworten:

- Welche Datensätze wurden genutzt, um den Output zu generieren?
  (Where-Lineage / schema-level)
- Welche Tupel von welchen Tabellen sind dafür verantwortlich, dass welche Tupel im Output sind?
  (Where-Lineage / instance-level)
- Welche Transformationen wurden genutzt, um den Output zu generieren?
  (How-Lineage / schema-level)
- Welche Transformationen sind auf welchen Tupeln für den Output verantwortlich?
  (How-Lineage / instance-level)

Offene Fragen

Nachdem nun bekannt ist, wie man wissenschaftliche Daten am effektivsten speichert und verarbeitet, bleiben jedoch noch einige Fragen ungeklärt. Beispielsweise werden in Zukunft wahrscheinlich viele wissenschaftliche Aufgaben in die Cloud ausgelagert, denn mit Technologien wie Google's Kubernetes welche von einer breiten Masse von Konzernen eingestzt werden soll[15], sollte es ein leichtes sein Rechenkraft von überall auf der Welt mieten zu können. Die Frage in Verbindung mit wissenschaftlichen Daten ist jedoch folgende: Wie sollen Daten in Petabyte-Bereich an die Cloud gesendet werden bzw. Ergebnisse in diesem Bereich empfangen werden. Die Übertragungsraten stellen in diesem Zusammenhang einen Flaschenhals dar, der umgangen werden muss, wenn Probleme in akzeptabler Zeit gelöst werden sollen.

Quellen:

[1] http://de.wikipedia.org/wiki/Wissenschaft, letzter Aufruf: 06.07.2014

[2] Mwebaze, Johnson, Danny Boxhoorn, and Edwin Valentijn. "Astro-WISE: Tracing and Using Lineage for Scientific Data Processing."

[3] https://www.facebook.com/notes/facebook-engineering/under-the-hood-hadoop-distributed-filesystem-reliability-with-namenode-and-avata/10150888759153920, letzter Zugriff, 10.07.2014

[4] http://www.itbusinessedge.com/cm/blogs/lawson/the-big-data-software-problem-behind-cerns-higgs-boson-hunt/?cs=50736, letzter Zugriff, 10.07.2014

[5] Davenhall, Clive. "DCC Digital Curation Manual: Instalment on Metadata." (2011).

[6] Gray, Jim, et al. "Scientific data management in the coming decade." *ACM SIGMOD Record* 34.4 (2005): 34-41.

[7] Sears, Russell, Catharine van Ingen, and Jim Gray. "To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem?." (2006).

[8] Ivanova, Milena, Martin Kersten, and Stefan Manegold. "Data vaults: a symbiosis between database technology and scientific file repositories."*Scientific and Statistical Database Management*. Springer Berlin Heidelberg, 2012.

[9] Misev, Dimitar, Peter Baumann, and Jürgen Seib. "Towards large-scale meteorological data services: A case study." *Datenbank-Spektrum* 12.3 (2012): 183-192.

[10] Sellis, Timos, Nick Roussopoulos, and Christos Faloutsos. "The R+-tree: A dynamic index for multi-dimensional objects." (1987).

[11] Wandelt, Sebastian, et al. "Data management challenges in next generation sequencing." *Datenbank-Spektrum* 12.3 (2012): 161-171.

[12] Cuevas-Vicenttín, Víctor, et al. "Scientific workflows and provenance: Introduction and research opportunities." *Datenbank-Spektrum* 12.3 (2012): 193-203.

[13] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." *Communications of the ACM* 51.1 (2008): 107-113.

[14] Ikeda, Robert, and Jennifer Widom. "Data lineage: A survey." (2009).

[15] http://googlecloudplatform.blogspot.de/2014/07/welcome-microsoft-redhat-ibm-docker-and-more-to-the-kubernetes-community.html, letzter Zugriff, 11.07.2014