JOHANNES KEPLER UNIVERSITÄT LINZ | JKU

Techniken und Trends in Datenbanken
(und Datawarehouses)

Josef Küng

Institut für
Anwendungsorientierte Wissensverarbeitung (FAW)
Johannes Kepler Universität Linz
Altenbergerstraße 69
A-4040 Linz, Austria

FAW

Inhalt

- Einleitung
- Moderne Hardware
- Moderne DBMS-Entwicklungen
  - Scaling up
  - Scaling out
  - Scientific DBs
  - Usability and Maintenance
- Zusammenfassung und Ausblick

Einleitung [1]

- **„Früher (im vorigen Jahrhundert) war die Datenbankwelt noch überschaubar“:**
  - Das relationale Datenmodell war beherrschend
  - Die Anzahl der DBMS-Hersteller war überschaubar
  - Einige, großteils in der Forschung verbliebene Ansätze
    - Deduktiven Datenbanken (erleben derzeit ein Revival)
    - Aktive Datenbanken (Trigger)
    - Objektorientierte Datenbanken
  - Die relationalen Datenbanken übernahmen einige der Konzepte und Techniken (Objektrelationale Datenbanken)

Einleitung [2]

- **Heute**
  - Eine Menge neuer Technologien
  - Eine fast unüberschaubare Anzahl von DBMS-Herstellern

Einleitung [3]

- **Warum?**

Datenmenge

Vernetzung

Peer to Peer (Soziale Netze)

Parallelität

Diversität

Cloud

Moderne Hardware [1]

- **Hauptspeicher entfernt sich von CPU**
  - Performancesteigerung ist höher als die Reduktion der Wartezeit
    eine Lösung: lokale Caches
- **Multicore-Hardware, unter Umständen heterogene Kerne**
  - Die richtigen Daten müssen zur richtigen CPU gebracht werden
    mögliche Lösung „Islands of computation“ → smarter Software
- **Festplattenkapazität (HDD) wächst**
  - aber: Übertragung (Bandbreite, Wartezeit) kommt nicht so gut mit
    Sind Flash-Speicher (oder PRAM) eine Lösung?
- **Neue Speichertechnologien sind im Kommen**
  - Flash-Speicher, PRAM, …

Jim Gray in 2006: **“Tape is Dead, Disk is Tape, Flash is Disk, RAM Locality is**

Moderne Hardware [2]

- **Software Latency wird zunehmend dominieren**

  wenn die Speicher keine beweglichen Teile mehr haben

- **Auch der Beitrag der Software zum Energieverbrauch eines Speichervorganges wird steigen**

Die Zeit eines Speed-Ups, der die Software nicht berücksichtigt, ist vorbei

Gerade DBMS sind davon voll betroffen

Moderne DBMS-Entwicklungen

- **Scaling Up**

  Die DB-Engine wird ‚getuned'

- **Scaling Out**

  Neue, prinzipielle Konzepte werden umgesetzt

- **Scientific DBs**

  Ursprünglich für Aufgabenstellungen aus der Wissenschaft entwickelte Datenbanksystemen

- **Usability and Mainenance**

  Diese Tätigkeiten werden besser unterstützt und in zunehmenden Ausmaß automatisiert

nach Miguel Branco, Data Intensive Applications and Systems Lab
Lousanne, Schweiz

Scaling up [1]

- **Die Transaktionsabwicklung wird effizienter**

  z.B. Data-oriented Transaction Execution Engine

  (Threads werden den Daten zugeordnet, nicht den Benutzern; noch in Forschung

- **Durchführung von Abfragen wird weiter optimiert**

  Verbesserungen an den Query-Optimizern

  Caching von Abfragen und zugehörigen Query-Plans

  Abfragespezifischer Maschinencode

  Einsatz von KI-Methoden, die kommende Abfragen antizipieren und vorbereiten

  Spezielle Hardware, die die Abfragebearbeitung direkt (nahezu ohne Software) durchführen kann

  ….

- **Neue Speichertechnologien werden besser (direkt) unterstützt**

  z.B. SSD

Scaling up [2]

- **Rows, Colums, Hybrid** [1]

Eine typische Tabelle:

Und deren Repräsentation am Permanentspeicher:

Miguel Branco: (Some)
Trends in Database Research

Bei der Berechnung der Summe über die Spalte C3 hat die CPU enorme Wartezeiten.

Scaling up [3]

- **Rows, Colums, Hybrid** [2]

  Die Lösung, Column Store:

  - Die Spalte wird gemeinsam gespeichert.

  - Die Spalte ist wichtiger als die Zeile.

  Beispiele für solche DBMS

Miguel Branco: (Some) Trends in Database Research

- Immer mehr traditionelle DBMS-Anbieter erweitern ihr System um Column-store-features, bleiben aber beim Row-Store-Konzept.
- Es hängt definitiv vom Anwendungsszenario ab, welche Variante (Row-store, Column-Store) angebracht ist.
- Es wird an Hybridsystemen gearbeitet, die zwischen den Repräsentationen wechseln können.

Theorie: CAP-Theorem [1]

- **C – Konsistenz (Consitency)**
- **A – Verfügbarkeit (Availability)**
- **P – Ausfalltoleranz (Partition Tolerance)**

In verteilen Systemen sind die drei Anforderungen nach Konsistenz (**C**onsistency), Verfügbarkeit (**A**vailability), Ausfalltoleranz (**P**artition Tolerance) nicht vollständig vereinbar und nur maximal zwei von dreien sind erreichbar.

(vermutet von Brewer 2000, bewiesen durch Gilbert und Nancy Lynch 2002)

Theorie: ACID - BASE [1]

- **ACID – Atomarität, Konsistenz, Isolation, Dauerhaft**
  - Die Welt der relationalen Datenbanken, Transaktionen (mit Verwendung von Sperren) gewähren diese 4 Eigenschaften.
- **BASE - Basically Available, Soft State, Eventually Consistent**
  - Konsistenz wird als ein Zustand betrachtet, der irgendwann erreicht wird. Es wird in Kauf genommen, dass bis zum Erreichen die Daten in der DB inkonsistent sind.

- ACID und BASE können als zwei Enden eines Konsistenzspektrums betrachtet werden

entnommen aus Datenbanken Online Lexikon, FH Köln, (http://wikis.gm.fh-koeln.de/wiki_db)

Theorie: ACID - BASE [2]

- **Beispiel ACID**:

  TransC und TransD müssen auf das Transaktionsende von TransB (incl. 2-Phase-Commit, wenn Verteilung vorliegt) warten.

ACID-Konsistenz mit Sperren und Wartezeiten

- **Beispiel BASE:**

  Lesezugriff ist immer erlaubt.
  Es wird mit Multiversion Concurrence Control (MVCC) gearbeitet.

BASE-Konsistenz ohne Sperren und ohne Wartezeiten

‚Eventual Consitency' ist unter Umständen nicht genug. Es ist schwer, später die Konsitenz wieder anzuheben.

Beide Graphiken entnommen aus Datenbanken Online Lexikon, FH Köln, (http://wikis.gm.fh-koeln.de/wiki_db)

Scaling out – NoSQL [1]

- An SQL database is a traditional relational database which can be queried using SQL.
- A NoSQL database is a database that is not an SQL database.

  Data is not stored in relations and the main query language to retrieve data is not SQL

> "NoSQL DEFINITION: Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open-source and horizontally scalable. The original intention has been modern web-scale databases. The movement began early 2009 and is growing rapidly. Often more characteristics apply such as: schema-free, easy replication support, simple API, eventually consistent / BASE (not ACID), a huge amount, of data and more. So the misleading term "nosql" (the community now translates it mostly with "not only sql") should be seen as an alias to something like the definition above."

Stefan Edlich, in : S. Edlich, A. Friedland, J. Hampe, B. Brauer und M. Brückner,
NoSQL - Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken, München: Carl Hanser Verlag, 2011.

Scaling out – NoSQL [2]

- NoSQL-Datenbanksysteme sind somit
  - Nicht relational.
  - Verteilt und horizontal skaliert.
  - Schemafrei oder hat nur schwache Schemaausprägungen.
  - Einfach bei der Datenreplikation
  - Durch eine einfache Programmierschnittstelle leicht zu verwenden.
  - Nur eventuell konsistent,
    weil nicht das ACID-Prinzip,
    sondern das eventually consistent/ BASE – Prinzip angewandt wird.

- Sie werden gerne eingeteilt in
  - Wide-Column-Stores / Column-Families
  - Documtent-Stores
  - Key-Value-Stores / Touple-Stores
  - Graph-Databases
  - Object Databases
  - XML-Databases
  - etc.

Beispiele für NoSQL-DBMS:

Scaling out – NoSQL [3]

- NoSQL vereinfacht das Programmiermodell.
- Die Designprinzipien (Strukturen der grundlegenden Datenobjekte aber auch der Abfragetechniken) unterscheiden sich von jenen der relationalen Datenbaken
  - Das relationale Modell unterstützt besonders Joins und Ad-Hoc-Abfragen (zu teuer und unvorhersehbar?)
  - NoSQL vernachlässigt ACID
  - NoSQL unterstützt Schemata eher wenig
- Individuelles Design führte zu einer Vielzahl unterschiedlicher Produkte
- Sie sind ideal für spezielle Aufgabenbereiche und spezielle Hardware-Konfigurationen
- NoSQL-Systeme haben eine niedrige Eintrittsschwelle (besonders im Zusammenhang mit Web- und Mobile-Anwendungen), trotz der vielen Heterogenitäten und Inkompabilitäten.

Scaling out – NewSQL

- ACID-compliant Datenbankmanagementsysteme.
- hoch saklierbar
- Partitionierung der Daten wurde vereinfacht (teilweise ohne Sperren)

- Dennoch ist die Partitionierung und Verteilung unter Beibehaltung von ACID die größte Herausforderung. Es wird nach wie vor daran geforscht, u.a. auch um den 2-Phasen-Commit-Overhead minimieren zu können.

Beispiel für solch ein System: VoltDB

Scaling out – MapReduce [1]

- Ein von Google eingeführtes Framework für nebenläufige Berechnungen über große Datenmengen auf Computerclustern.
- MapReduce-Framework sorgt für die Aufteilung der Berechnungen auf mehrere Recheneinheiten
- Dadurch parallele Ausführung auf mehreren Rechnern
- Nach Beendigung der Berechnungen aggregiert das Framework die Ergebnisse
- Entwickler von verteilten Anwendungen müssen nur das Framework benutzen, keine Codeänderungen bei der Änderung der Client-Anzahl nötig
- Verwendung von handelsüblichen Computern möglich
- Zwei Phasen
  - Map
  - Reduce

  Beide basieren auf dem Key-Value-Pair-Konzept

Scaling out – MapReduce [2]

- Hadoop
  - Eine Implementierung des MapReduce-Konzeptes
  - Open Source Projekt der Apache Software Founation
  - Dient zur Verarbeitung großer Datenmengen
  - Zusammenschluss von Recheneinheiten zu Clustern
  - Parallele Abarbeitung auf den Recheneinheiten
  - Hohe Fehlertoleranz gegenüber Hardwareausfälle
  - Hadoop Distributed File System (HDFS) dient als gemeinsames Dateisystem

Scientific Databases – Array Databases

- RDBMS unterstützen Arrays nicht besonders gut
- Aber, Arrays kommen oft vor
  - Wissenschaft (z.B. Bioinformatik)
  - Finanzwesen
- Array DBMS unterstützen
  - Integrierte Speicherung und Berechnung
  - Deklarative Sprachen (z.B. SciQL)
  - Query Optimisation

Beispiele solcher DBMS:

SciENS SciDB

Relational Database		
I	J	value
0	0	32.5
1	0	90.9
2	0	42.1
3	0	96.7
0	1	46.3
1	1	35.4
2	1	35.7
3	1	41.3
0	2	81.7
1	2	35.9
2	2	35.3
3	2	89.9
0	3	53.6
1	3	86.3
2	3	45.9
3	3	27.6

48 cells

Array Database			
32.5	46	81.7	54
90.9	35	35.9	86
42.1	36	35.3	46
96.7	41	89.9	28

16 cells

Miguel Branco: (Some) Trends in Database Research

Usability, Maintenance [1]

- **Auto-tuning**
  - Tuning-Fragen wie
    - Welche Indices sollen erstellt werden?
    - Wann sollen die Indices erstellt werden?
    - Wie sollen die Daten am besten verteilt werden?
    - etc.

    löst das System von selbst.
  - Das DBS wird zunehmend zur ‚Black Box'
  - Administrationsaufwand und –risiko sinkt.
- ‚Database Cracking' ist in diesem Zusammenhang ein Terminus technicus.

  Physische Datenreorganisation on-the-fly,

  Indices werden automatisch an die Arbeitslast angepasst, Abfragen zerteilen den Datenbestand.

Miguel Branco: (Some) Trends in Database Research

Usability, Maintenance [2]

- **In-Situ Query Processing**
  - In-situ bedeutet, dass man ‚vom Platz aus' auf alle externen Daten direkt zugreifen kann
  - Die Query-Engine hat also direkten Zugriff auf Daten jeder beliebigen Speicherart (File, NoSQL, RDB, …).
  - kein Laden, keine Replikation ist notwendig

Beispiel: PostgresRaw

Figure 7: Comparing the performance of PostgresRaw with other DBMS

I. Alagiannis, R. Borovica, M. Branco, S. Idreos, A. Ailamaki: NoDB: Efficient Query Execution on Raw Data Files

Usability, Maintenance [3]

- **Cloud**
  - Hier treffen dann alle bisher erwähnten Technologien (und weitere) zusammen.
    - RDBMS
    - Speichertechnologien (virtuell)
    - NoSQL
    - MapReduce
  - Jedoch …
    - Stabilität?
    - Sicherheit, Privacy, … ?
    - Wo sind die Daten physisch? – juristische Randbedingungen
  - Beispiele:

amazon webservices™ Azure

Zusammenfassung, Ausblick

- **Es gibt eine Menge von Trends und Richtungen, in die geforscht und entwickelt wird, aber auch Produkte vorhanden sind.**
  - Column stores, Arrays, MapReduce, NoSQL, Query processing, Auto-tuning, …
  - Dabei wurden viele hier nicht erwähnt (Cloud, NoDB, …)
  - kein Laden, keine Replikation ist notwendig
- **Verteilung in Hardware und Software ist im Zunehmen begriffen**
- **Immer mehr Datenbankservices wechseln in die Cloud**
- **Hardware-getriebene Entwicklungen werden zunehmen**
- **Das DBMS-Paradigma ‚Deklarative Data Processing' wird adaptiert werden, um das parallele Verarbeiten zu erleichtern.**

Vielen Dank für Ihre Aufmerksamkeit!

Fragen, Diskussion