Big Data Visualisierung

Stephan Lex

4. März 2015

PROBLEMSTELLUNG

- Verwaltung sehr großer Dateimengen
- Analyse dieser Dateimengen
- Visualisierung der Relationen innerhalb der Menge
- Interpretation und Filterung einzelner relevanten Teile
- Unabhängiges Querying
- Simples und ansprechendes Design für den Endnutzer

LÖSUNGSANSÄTZE

Apache Hadoop[1] bietet durch das von Google entwickelte MapReduce-Verfahren[2] einen guten Algorithmus um großes Datenmengen zu crunchen und sortieren. Dabei gilt zu beachten, dass dieses Verfahren vor allem für Big Data entwickelt worden ist. So können die Daten parallel und asynchron auf mehreren Recheneinheiten eines Clusters verarbeitet werden. Dem Entwickler steht es frei zu entscheiden, wie die ankommenden Daten verarbeitet werden, da die Map und Reduce Schnittstellen selbst implementiert werden. Allerdings sind die daraus gewonnenen Daten nur bedingt für den Endverbraucher tauglich, da sie nur relational gespeichert werden.

[1] `http://hadoop.apache.org/`
[2] `http://de.wikipedia.org/wiki/MapReduce`

Um diese Daten nun effizient auswerten und visualisieren zu können bildet sich ein Import in eine NoSQL-Graphdatenbank wie Neo4J[3] an. Diese stellt die Datenvernetzungen als Graphensysteme dar. Durch Knoten und Kanten werden einzelne Objekte und Relationen übersichtlicher und weniger redundant gespeichert. Die Abfrage in Neo4J erfolgt über die eigene, an SQL angelehnte Query Sprache Cypher. Die Umstellung von relationaler auf graphischer Datenbank führt zu größerer Übersichtlichkeit und gewährleistet sowohl einfacheres als auch schnelleres Querying.

Optimal wäre nun ein Joint dieser beiden Systeme, um große Datenmengen zuerst relativ abstrakt zu analysieren und anschließend dem Nutzer adäquat präsentieren zu können[4][5]. Dazu stellen sowohl Hadoop als auch Neo4J umfangreiche APIs und Dokumentationen für Einbindung in andere Systeme bereit.

Das Problem könnte in Java mithilfe der Eclipse IDE[6] als vollwertige Java Applikation oder innerhalb einer RESTful[7] Client-Server Kommunikation implementiert werden. Diese Lösung bezieht sich auf den Fall, dass die Datenmengen noch nicht geordnet in einer relationalen Datenbank vorhanden sind. Sollte dies schon der Fall sein, können auch größere Daten relative einfach mit der embedded-API[8] von Neo4J in die NoSQL Datenbank übertragen werden.

Probleme

Allerdings gilt zu beachten, dass diese Lösung umso komplizierter zu implementieren ist, je abstrakter und ungenauer die Informationen über die Datenmengen sind. Deswegen ist es notwendig sich mit folgenden Fragen auseinanderzusetzen:

- Wie groß sind die Datenmengen?
- Wie sind die Daten beschaffen?
- Wie regelmäßig sind die Feeds?
- Welche Informationen sind für den Endverbraucher wichtig? Welche nicht?
- Sind die Datenmengen so groß, dass ein Server-, Rechnercluster benötigt wird.
- Sind die Daten vertraulich?
- Welches Objektmodell ist am besten geeignet?

[3] http://neo4j.com/
[4] http://blog.xebia.com/2012/11/13/combining-neo4j-and-hadoop-part-i
[5] http://blog.xebia.com/2013/01/17/combining-neo4j-and-hadoop-part-ii/
[6] https://eclipse.org/
[7] http://neo4j.com/docs/milestone/rest-api.html
[8] http://neo4j.com/docs/stable/tutorials-java-embedded.html

ALTERNATIVEN

In diesem Lösungsansatz wurden Hadoop und Neo4J ausgewählt, da sie die momentan am häufigst vertretenen und umfangreichsten Frameworks sind. Allerdings gibt es für die Problemstellung auch einige technische Alternativen, die verschiedenen Vorteile mit sich bringen.

Apache Spark[9] :

- Arbeitet bis zu 100x schneller als Hadoop.
- Ist flexibler als Hadoop im Umgang mit verschiedenen Dateiformaten(Hadoop abhängig vom eigenen HDFS).
- Umfangreiche APIs für den Umgang mit Java, Scala und Python.
- Allerdings schlechtere Dokumentation und geringere Erfahrung als mit Hadoop.
- Weniger umfangreiches Angebot an Tools als Hadoop.

Presto[10] :

- Mehr eine andere Herangehensweise als ein direkter Ersatz für Hadoop.
- Stellt eine Query Engine dar, die variabel große und beschaffene Datensätze in Echtzeit analysieren kann.
- Eine Engine, die sowohl Big Data verwalten kann, sowie diese auch in Echtzeit auswerten kann.

[9] `https://spark.apache.org/`
[10] `https://prestodb.io/`