Performanz Evaluation von Graphdatenbanksystemen versus MySQL am Beispiel von Kookkurrenzgraphen

Martin Junghanns
martin.junghanns@studserv.uni-leipzig.de
Sascha Ludwig
s.ludwig@studserv.uni-leipzig.de
Robert Schulze
robert.schulze@studserv.uni-leipzig.de

11. Januar 2012

Zusammenfassung

Dieser Bericht dient der Zusammenfassung der Ergebnisse eines Praktikums im Rahmen des Moduls "Fortgeschrittene Methoden des Information Retrieval" im WS 2010/11 der Universität Leipzig. Inhalt des Praktikums ist die Evaluation verschiedener Graphdatenbanken und eine Gegenüberstellung mit relationalen Datenbanken. Als Vertreter der Graphdatenbanken werden Neo4j, Dex und OrientDB näher betrachtet, MySQL wird als relationale Datenbank in den Vergleich einbezogen.
Konkrete Ziele des Praktikums sind das Kennenlernen der verschiedenen Graphdatenbanken, der zu Grunde liegenden Datenmodelle und deren Abfragetechniken. Für den experimentellen Vergleich sollen verschiedene Kookkurrenzgraphen des Wortschatz Leipzig Projektes in die Datenbanken importiert und mit den angebotenen Techniken abgefragt werden. Dabei wurden ausgewählte SQL Anfragen in API Calls, aber auch in deklarative Anfragesprachen des jeweiligen Herstellers umformuliert und hinsichtlich ihrer Ausführungszeit verglichen.

1 Neo4j

Neo4j[1] ist eine eingebettete, persistente, transaktionale Graphdatenbank. Sie wird seit 2005 von der Firma Neo Technology, Inc. entwickelt, Version 1.0 erschien Anfang 2010. Die Datenbank wurde vollständig in Java geschrieben und steht quelloffen zur Verfügung.[2]

1.1 Anbindung

Neo4j kann in zwei verschiedenen Szenarien betrieben werden: Die eingebettete Version wird direkt in eine Java Applikation integriert und bietet den performantesten Zugriff auf die Datenbank. Darüber hinaus implementiert Neo4j eine REST Schnittstelle, mit welcher auf die Datenbank via HTTP Requests zugegriffen werden kann. Die Anbindung erfolgt über verschiedene Clients, welche den Zugang aus unterschiedlichen Plattformen und Sprachen her-

[1] http://www.neo4j.org
[2] https://github.com/neo4j/community

aus ermöglichen.
Für die Leistungsmessung wurde die eingebettete Variante der Datenbank genutzt. Sie bietet neben dem Geschwindigkeitsvorteil auch das volle Spektrum an Zugriffsmöglichkeiten an.

1.2 Anfragemöglichkeiten

Neo4j bietet verschiedene Möglichkeiten an, den gespeicherten Graphen zu traversieren bzw. abzufragen. Neben der nativen Java API und der ebenfalls in Java entwickelten Traversal API steht seit Version 1.5 mit Cypher eine deklarative Abfragesprache zur Verfügung. Die drei genannten Abfragemöglichkeiten werden im Folgenden kurz vorgestellt und die Formulierung der vorgegebenen SQL Queries aufgezeigt.

1.2.1 Native API

1.2.2 Traversal API

1.2.3 Cypher

1.3 Einschränkungen

Cypher, Traverser Speicherprobs, keine Optimierung

1.4 Dokumentation

1.5 Aussicht

2 OrientDB

2.1 Anfragemöglichkeiten

OrientDB lässt sich einfach über eine Java API steuern. Das Handling von OrientDB wird, nach einer gewissen Einarbeitungszeit, relativ einfach, was auch der guten Javadoc der API geschuldet ist. Neben einer Java-API bietet OrientDB auch die Möglichkeit der Steuerung des Datenbanksystems über eine Konsole. Dies hat den Vorteil, dass man das Datenbanksystem erkunden und kennen lernen kann. In der Konsolen können Befehle in einer SQL-artigen Syntax an das DBS geschickt und ausgewertet werden.

2.2 Einschränkungen

Ein großes Bottleneck bei OrientDB ist der Import der Daten aus einer SQL-Datenbank. Auch nach dem befolgen der "Performance Hintsünd dem Optimieren diverser JVM Parameter, stieg die Zeit für den Import exponentiell zur Anzahl der einzufügenden Daten an. Dies hat zur Folge, dass große Datenbanken (ab 1M Datensatz) nicht mehr importiert werden konnten. Mittels JProfiler konnte der Persistierungsprozess von OrientDB als HotSpot ausgemacht werden. Ein Kontaktaufnahme mit dem Autor bezüglich dieses Performanzverlustes ist durch uns erfolgt.

2.3 Dokumentation

OrientDB ist gut und ausführlich dokumentiert. Es liegt sowohl eine komplette und ausführliche Javadoc der API vor, als auch Informationsseiten zu grundlegenden und weiterführenden Aspekten vor. Auf diesen Webseiten lassen sich Informationen zu fast allen Fragen finden. Allerdings ist das Finden einer Antwort auf eine konkrete Frage teilweise schwierig. Dies ist unter Umständen dem Fakt geschuldet, dass OrientDB sowohl ein Dokumenten-Datenbanksystem, als auch ein Key-Value-Store, als auch ein Graphdatenbanksystem ist. Oft werden in der Dokumentation diese drei Konzepte vermischt, was

zu Verwirrungen führen kann. Weiterhin hat das Fehlen eines ausführlichen und kompletten Einstiegbeispiels die Einarbeitung erschwert. Auch fehlen Beispiele für komplexe Anwendungsfälle. Allerdings war der Autor bei speziellen Fragen per E-Mail zu erreichen, wobei man auf eine Antwort lediglich eine angemessene Zeit warten musste.

2.4 Aussicht

OrientDB stellt bezüglich seiner Graphdatenbankenfunktionalität einige interssante Features in Aussicht. OrientDB stellt seit Kurzem eine Implementation der Tinkerpop Blueprints APIs zur Verfügung. Somit steht auch die einfache und leicht lesbare Highlevel-Abfragesprache Gremlin zur verfügbar. Mit dieser Abfragesprache ist es möglich auch komplexe und vielseitige Abfragen verständlich zu formulieren.

3 DEX

4 Evaluation

4.1 Testbed

Hardware Fakten Datensätze (Anzahl Knoten, Kanten, evtl. Gradverteilung) cold caches / warm caches JVM settings versionen

4.2 Ergebnisse