Praktikumsbericht: Betriebsdatenerfassung regenerativer Energieanlagen

Martin Junghanns
martin.junghanns@studserv.uni-leipzig.de

12. Januar 2012

1 Auswahl

Zu Beginn des Praktikums sollten zunächst aus verschiedenen NoSQL Datenbanken geeignete Vertreter zur Erfüllung definierter Anforderungen ausgewählt werden. Im Folgenden werden Graphdatenbanken und Dokumentendatenbanken kurz vorgestellt und anschließend deren Eignung für das Praktikum beurteilt.

1.1 Graphdatenbanken

Graphdatenbanken legen den Fokus auf die Speicherung und Verarbeitung stark vernetzter Daten. Im relationalen Datenbankmodell ist im Rahmen der Normalisierung eine Aufteilung der Daten auf mehrere Tabellen üblich. Diese werden jedoch in aufwendigen Verbundoperationen bei einer Anfrage wieder zusammengeführt. Graphdatenbanken verwalten die Daten in ihrer expliziten Struktur, d.h. Beziehungen zwischen Datensätzen werden direkt am Datensatz modelliert und gespeichert. Dies führt zu einem konstanten Aufwand bei der Abfrage von Beziehungen eines Datensatzes, da nur dieser selbst betrachtet werden muss. In relationalen Datenbanken werden Fremdschlüsselbeziehungen über Indexstrukturen verwaltet, der Aufwand ist dabei abhängig von der Anzahl Einträge innerhalb des Index. Selbst bei logarithmischem Aufwand führt dies bei konstantem Datenwachstum zu erheblichen Nachteilen. Deutlicher wird die Problematik bei der Verkettung mehrerer Verbundoperationen innerhalb einer Anfrage, wie zum Beispiel bei der Frage nach dem kürzesten Pfad zwischen zwei Knoten. Für diese Kategorie von Anfragen ist das Konzept der Graphdatenbanken wesentlich vorteilhafter als das relationale Datenmodell.

Neben der optimierten Speicherung vernetzter Daten bietet ein Großteil der Hersteller die Möglichkeit zur Speicherung semi- bzw. unstrukturierter Daten. Im relationalen Modell wird das Datenschema beim Anlegen der Datenbank definiert, spätere Änderungen sind nur mit erheblichem Aufwand und zusätzlicher Komplexität verbunden. Werden

zum Beispiel neue Attribute an eine Tabelle angefügt, müssen entwededer zusätzliche Standard- oder Null-Werte gespeichert werden, oder Entity-Attribute-Value (EAV) Tabellen verwendet werden, was wiederum die Komplexität der Anfrage durch eine zusätzliche Verbundoperation erhöht.
Ein weiterer nennenswerter Vorteil von Graphdatenbanken ist analog zu Objektdatenbanken die Vermeidung des objektrelationalen Abbildungsproblems.

Das Datenmodell aller Graphdatenbanken basiert auf einem gerichteten Graphen, wobei sowohl Knoten, als auch Kanten durch Attribute (engl. *properties*) erweitert werden können. Dadurch lassen sich zum Einen gewichtete Graphen darstellen und zum Anderen verschiedene Typen von Knoten und Kanten mit unterschiedlichen Eigenschaften definieren. Knoten- und Kantenattribute werden üblicherweise als Schlüssel-Wert-Paare verwaltet. Durch die Verwendung verschiedener Kantentypen besteht die Möglichkeit, Mehrfachkanten zwischen zwei Knoten zu modellieren.

Graphdatenbanken bietet üblicherweise eine native API an, mit der in der jeweiligen Programmiersprache auf die Daten zugegriffen werden kann. Darüber stellen Vertreter wie Neo4j oder AllegroGraph REST Schnittstellen für den entfernten Zugriff zur Verfügung. Insbesondere für Neo4j steht eine umfangreiche Menge von REST-Clients in verschiedenen Programmiersprachen zur Verfügung. Diese werden, wie auch die Datenbank selbst, von der Community unterstützt und mitentwickelt.
Der Zugriff auf die Daten kann wie bereits erwähnt über native Aufrufe erfolgen oder über Anfragesprachen, welche von den Datenbankherstellern zur Verfügung gestellt werden. Neo4j bietet mit Cypher eine deklarative und mit Gremlin eine imperative Anfragesprache zur Traversierung eines Graphen an. Gremlin wird im Blueprints Projekt[1] weiterentwickelt, es handelt sich dabei um ein Projekt, welches den Zugriff auf verschiedene Graphdatenbanken standardisieren will und neben Gremlin weitere Hilfsmittel zur Verwaltung von Graphen zur Verfügung stellt.
OrientDB bietet eine SQL ähnliche Sprache zur Abfrage des Graphen an während InfroGrid als Vertreter der Graph-RDFStores die Sprache SPARQL einsetzt.

Die Einsatzszenarien von Graphdatenbanken sind vielseitig, typische Beispiele sind der Einsatz in Online Sozialen Netzwerken, die Organisation und Optimierung von Transportnetzwerken oder in Empfehlungssystemen. Durch die Möglichkeit zur intuitiven Umsetzung sämtlicher Graphenalgorithmen sind darüber hinaus noch viele weitere Einsatzgebiete denkbar.

Die bekanntesten Vertreter von Graphdatenbanken sind Neo4j[2], OrientDB[3], Allegrograph[4], DEX[5], HypergraphDB[6] und InfiniteGraph[7].

[1] https://github.com/tinkerpop/blueprints/wiki/
[2] `http://neo4j.org/`
[3] `http://www.orientechnologies.com/orient-db.htm`
[4] `http://www.franz.com/agraph/`
[5] `http://www.sparsity-technologies.com/dex`
[6] `http://www.hypergraphdb.org/index`
[7] `http://www.infinitegraph.com/`

1.2 Dokumentendatenbanken

Im Gegensatz zu Graphdatenbanken wurden Dokumentendatenbanken vorrangig für die Verwaltung voneinander unabhängiger, unstrukturierter Daten entwickelt. Daten werden nicht innerhalb von Tabellen in Zeilen und Spalten gespeichert sondern als strukturierter Datentyp. Es handelt sich demzufolge nicht um ein Word- oder Textdokument sondern um Datensammlungen im JSN, YAML oder auch RDF Format. Es gibt kein festgelegtes Schema, jedes Dokument kann einen eigenständigen Aufbau besitzen. Die Schemaverantwortung wurde komplett in die Anwendung verlagert.

Dokumente werden üblicherweise mit einer ID ablegt um sie eindeutig innerhalb der Datenbank zu identifizieren. Fremdschlüsselbeziehungen zwischen einzelnen Dokumenten existieren hingegen nicht. Durch diese Einschränkung eignen sich Dokumentendatenbanken besonders gut für eine Partitionierung auf Ebene der Dokumente und sind somit bei wachsendem Datenvolumen horizontal sowohl für Lese- als auch Schreibzugriffe skalierbar.
Der Großteil der Dokumentendatenbanken bietet neben der Partitionierung die Replikation des Datenbestandes an. Diese sichert Fehlertoleranz und zusätzliche Parallelisierung von Lesezugriffen.

Die bekanntesten Vertreter sind CouchDB[8] und MongoDB[9]. Beide Datenbanken bieten Zugriff über REST Schnittstellen für Clients in verschiedenen Programmiersprachen an. Da es sich um verteilte Datenbanken handelt und somit parallele Anfragen möglich und sinnvoll sind, bieten beide Datenbanken MapReduce zur Formulierung komplexer, verteilter Anfragen an. MongoDB bietet darüber hinaus eine eigene dynamische objektbasierte Anfragesprache.

Dokumentendatenbanken eignen sich insbesondere für den Einsatz in Web 2.0 Anwendungen, bei denen sich das Datenschema kontinuierlich weiterentwickelt bzw. nie fest definiert werden kann. Eine der bekanntesten Anwendungen für Dokumentendatenbanken ist Lotus Notes. Event Logging und Archivierung von Datenbeständen sind neben der Echtzeit-Datenanalyse weitere Einsatzszenarien dieser Systeme.

1.3 Auswahl

Es wurden zwei Vertreter beider Kategorien in Bezug auf die gestellten Anforderungen betrachtet: Neo4j und MongoDB.

Die gestellten Anforderungen beziehen sich zunächst auf die Art der Daten. Es existieren zwei verschiedene Datenarten: Metadaten zu Energieanlagen und Messwerte von Wechselrichtern einer Anlage. Die Messwerte sind einem konkreten Zeitpunkt und einem

[8] `http://couchdb.apache.org/`
[9] `http://www.mongodb.org/`

Wechselrichter einer Anlage zugeordnet. Bis auf diese Zuordnung bestehen keine weiteren Beziehungen zwischen den Daten. Das Format einer Zeitreihe ist festgelegt, variiert jedoch in Abhängigkeit von der Datenart, welche vom Wechselrichter zur Verfügung gestellt wird. Es existieren Datenarten, denen zusätzliche Attribute zugeordnet sind.
Betrachtet man ausschließlich die Art der Daten, bietet sich auf Grund der schwachen Vernetzung eine Dokumentendatenbank an. Graphdatenbanken eignen sich besonders gut zum Traversieren vernetzter Daten, diese Anforderung ist im Praktikum nicht gegeben. Es handelt sich um Daten die einem wechselnden Schema unterliegen, auch diese Situation wird von einer Dokumentendatenbank berücksichtigt.

Spätere Änderung der Daten ist für Messwerte ausgeschlossen, lediglich die Metadaten einer Anlage können aktualisiert werden. Diese Änderungen müssen nachvollziehbar sein, um die Messwerte einem bestimmten Zustand einer Anlage zuzuordnen. Dies erfordert die Möglichkeit zur Versionierung der Daten. Parallele Schreiboperationen einer konkreten Zeitreihe treten nicht auf, d.h. es besteht keine Notwendigkeit ACID konformer Transaktionen.
Eine Versionierung der Daten wird von keinem der Datenbanksysteme angeboten und muss demzufolge über das Datenschema sichergestellt werden. Beide Datenbanksysteme unterstützen zwar konkurrierende Schreibzugriffe, dies ist jedoch nicht erforderlich.

Eine weitere Anforderung betrifft die Datenmenge: Eine Datenbank soll bis zu 10 Milliarden Messpunkte von mehreren Wechselrichtern aufnehmen. Die Frequenz in der eine Anlage Messwerte liefert, liegt im Sekundenbereich. Dies erfordert das parallele Schreiben verschiedener Zeitreihen.
MongoDB eignet sich durch die Möglichkeit zur Partitionierung gut für verteilte Schreib- und Lesezugriffe. Wachsendes Datenvolumen kann durch die Erweiterung eines Clusters abgefangen werden. Keine aktuelle Graphdatenbank unterstützt eine vergleichbare, automatische Partitionierung der Daten. Lesezugriffe können durch Replikation skaliert werden, dem Problem des wachsenden Datenvolumens kann hingegen nur durch vertikale Skalierung begegnet werden.

Mögliche Anfragen auf dem Datenbestand umfassen sowohl rudimentäre Exact- und Range-Match Anfragen in Bezug auf eine konkrete Anlage als auch komplexe, analytische Auswertungen von Messwerten über mehrere Anlagen und Zeitbereiche. Neo4j stellt mit Cypher und Gremlin zwei Anfragesprachen zur Verfügung, diese sind jedoch vorrangig für das Traversieren eines Graphen entwickelt und optimiert wurden. MongoDB hingegen stellt mit der dynamischen objektbasierten Anfragesprache und MapReduce zwei mächtige Werkzeuge zur Formulierung analytischer Anfragen zur Verfügung.

Nach dieser Gegenüberstellung wurde MongoDB als ein Vertreter der Dokumentendatenbanken zur Verwaltung der Betriebsdaten regenerativer Energieanlagen ausgewählt. Informationen über das verwendete Schema, verschiedene Anfragen sowie die Ergebnisse des Benchmarks sind unter `https://github.com/s1ck/MongoDB_eval` abrufbar.