Apache Lucene und Oracle in der Praxis - Volltextsuche in der Cloud

Frank Szilinski, esentri software GmbH
Dominic Weiser, esentri consulting GmbH
Ettlingen

Schlüsselworte:

Apache Lucene, Apache Solr, elasticsearch, Java, Cloud, Volltextsuche, REST, Open-Source, RDBMS

Einleitung

Die zunehmende Dokumentenanzahl welche heute in der Cloud zur Verfügung gestellt werden muss erzeugt immer größere Datenvolumen. Um einzelne Dokumente und Inhalte auffinden zu können, bedient man sich bereits seit einigen Jahren der Suche. Lucene ist ein freies Software Projekt, welches für die Indizierung der Dokumente sorgt und somit dabei hilft, Inhalte schneller auffinden zu können. Auf der Grundlage von Lucene wurden über die Jahre viele unterschiedliche Projekte entwickelt welche die Anwendung von Lucene komfortabler machen sollen. Zwei dieser Projekte sind Solr und elasticsearch. Im Folgenden sollen diese Projekte kurz vorgestellt und miteinander verglichen werden. Dabei liegt der Schwerpunkt der Analyse auf Cloud-Computing und der Suche in relationalen Datenbanksystemen.

Apache Lucene

Apache Lucene ist eine Open-Source Java-Library, die zur Wiederbeschaffung von Informationen entwickelt wurde. Seit 1997 wird Lucene in Java programmiert und gehört seit 2002 zur Apache Software Foundation. Über die Jahre wurde Lucene in viele andere Programmiersprachen, wie C++, C#, Perl und Python, übersetzt. Neue Kern-Features werden aber weiterhin in Java entwickelt.
Lucene besitzt den Anspruch „out-of-the-box“ zu funktionieren, was oft damit verwechselt wird, dass Lucene sofort eine vollumfängliche Suchfunktionalität bereitstellt. Dies erweist sich in der Praxis allerdings als Trugschluss! Lucene dient primär der Informations-Wiederbeschaffung, bei der Dokumente im Vorfeld indiziert werden müssen. Die Suche mit Lucene in einem Index ist schneller als herkömmliche Suchvorgänge, die ohne die Techniken der vollständigen Indexierung auskommen müssen. Für die erfolgreiche Erstellung der Indexe ist es allerdings nötig, dass Dokumente in Textform vorliegen.
Der Index wird hierbei als inverse Liste verwaltet. Dies kann prinzipiell mit dem Aufbau des Stichwortverzeichnisses eines Buchs verglichen werden. Alle Quellen werden unter den entsprechenden Stichworten gruppiert, was den Vorteil bietet, dass bei einem gefundenen Index alle Quellen automatisch zugeordnet sind.

Die Indizierung von Textdokumenten läuft immer nach folgendem Schema ab:

1. **Inhalt auffinden:** Im ersten Schritt müssen die Dokumente an Lucene übermittelt werden. Die Übertragung selbst ist hierbei nicht Bestandteil der Lucene Funktionalität sondern erfolgt mittels Crawler. Es gibt verschiedene auf Lucene aufbauende Projekte welche Crawler implementieren (z.B. Nutch).

2. **Dokumente zusammensetzen:** Es werden die unterschiedlichen Meta-Daten, aus welchen ein Dokument besteht, zusammengetragen. Auch dieser zweite Schritt wird von Lucene nicht proprietär unterstützt. Dokumente bestehen aus vielen Einheiten wie zum Beispiel Inhalt, Autor oder Erstellungsdatum. Viele frei verfügbare Projekte beschäftigen sich mit der Erstellung dieser Metadaten und können mit Lucene verknüpft werden (z.B. Apache Tika).
3. **Analyse:** Um den Index aufbauen zu können, muss der Text in seine Bestandteile zerlegt werden. Diese Aufgabe wird von Lucene übernommen. Dazu werden die Dokumente mittels Tokenizer und Filter auf wichtigen Wortstämme reduziert.
4. **Indizierung:** In diesem Schritt wird das Dokument in den Index aufgenommen. Dabei werden neue Tokens zu bereits vorhanden Tokens in der Liste hinzugefügt oder als neuer Eintrag abgelegt.

Wurde der Index erstellt, kann der Anwender darin suchen. Eine Benutzeroberfläche, wie man sie zum Beispiel von Google kennt, wird von Lucene nicht zur Verfügung gestellt. Erst die Übersetzung der Abfrage wird wieder durch Lucene übernommen. Dies bedeutet, dass die Anwendereingabe so umgestellt werden muss, dass Lucene die Inhalte im Index wieder finden kann. Dabei werden die bereits von anderen Suchmaschinen bekannten Operatoren („+“, „-“, „AND“ oder „AND NOT“) unterstützt. Hat Lucene die Tokens extrahiert, wird der Index nach ihnen durchsucht. Das Ergebnis wird von Lucene in sortierter und gewichteter Reihenfolge zurück gegeben.

Lucene kommt mit vielen Features um eine schnelle Volltextsuche zu implementieren, was einer der Gründe für die große Verbreitung von Lucene ist. Die Liste der ca. 400 registrierten Partner, die Lucene direkt verwenden reicht von Apple bis Zilverline. Betrachtet man die Partner, welche ein auf Lucene aufbauendes Projekt verwenden, ist die Liste noch viel länger.

Apache Solr

Wenn man sich mit Lucene als Suchmaschine beschäftigt, wird man sehr schnell auf das Apache Solr Projekt stoßen. Dieser Zusammenhang geht so weit, dass Lucene und Solr als Geschwister-Projekte behandelt werden. Solr lässt sich als eigenständiger Suchserver beschreiben. Dies bedeutet nicht, dass Solr eine vollständige Suchmaschine ist. Zur funktionstüchtigen Suchmaschine fehlt ein Crawler, welcher Bereiche nach Dokumenten durchsucht und diese dem Indexer übergibt.

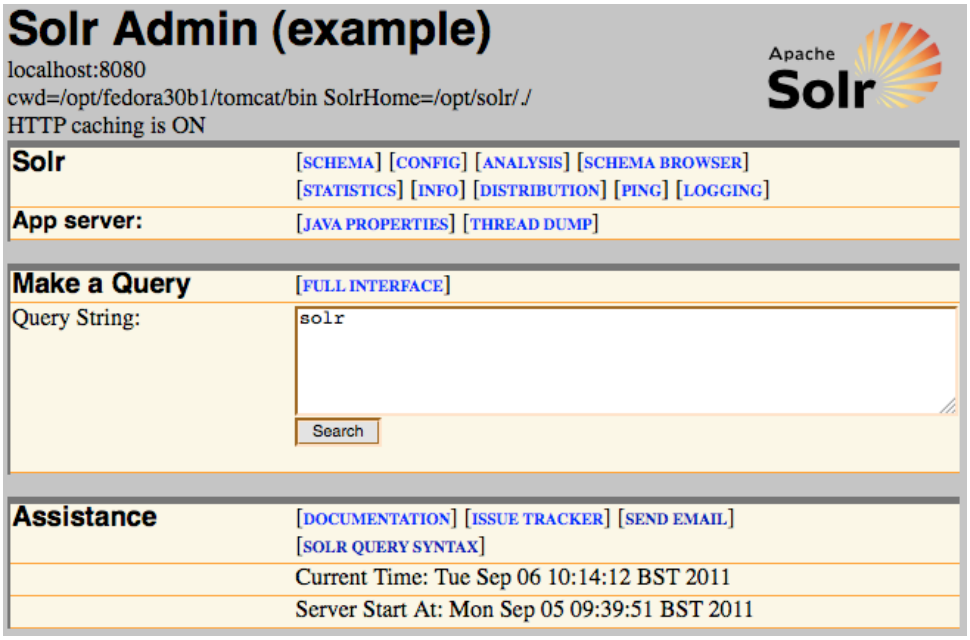

Abb. 1: Solr Administrations-Oberfläche (Quelle: http://ora.ouls.ox.ac.uk:8080/solr/admin/)

Mit dem Empfangen von Daten (Indexer) beginnt das Solr Projekt. Im vergleich zu Lucene liegt der Mehrwert von Solr unter anderem in einer Hight-Level-API. Solr ist mit REST-ähnlichen Schnittstellen über HTTP in XML oder JSON, ausgestattet. Dadurch kann Solr unabhängig von der Programmiersprache eingesetzt werden. Es ist genau diese Unabhängigkeit die Solr so populär macht.
Die einzigen Voraussetzungen die für den Einsatz von Solr erfüllt sein müssen sind ein Application Server, welcher den Servlet 2.4 Standard erfüllt (z.B. Tomcat, Jetty oder Jboss), und ein installiertes Java 5. Ansonsten kommt Solr als fertig zu installierendes .war File. Nach der Installation stellt Solr eine Administrations-Oberfläche zur Verfügung (siehe Abb. 1), in der auch die Konfiguration stattfinden. Außerdem können Testanfragen abgesendet werden um die Antwort im eingestellten Format zu überprüfen.
Leider fehlt die Unterstützung zur Volltextsuche in Rich-Dokumenten wie zum Beispiel Word, PDF oder HTML. Diese Lücke wir vom Apache Projekt Tika gefüllt. Tika ist ein Analyse-Werkzeug, welches Inhalte von verschiedenen Dokumenten analysieren kann. Darüber hinaus kann es die Metadaten eines Dokuments indizieren. Zu den von Tika unterstützen Formaten zählen neben den geläufigen Office Formate (doc, pdf, odt) auch Audio-, Video-, Bild- und Zip-Formate.

Zusammen mit Tika ist Solr eine sehr mächtige Suchmaschine. Wem der Crawler fehlt greift an dieser Stelle auf das Apache Nutch Projekt zurück. Für erweiterte Funktionalität gibt es einige Projekte die auf Solr aufbauen, wie zum Beispiel SolrCloud. Damit lässt sich Solr in der Cloud zu Clustern zusammenfügen.

elasticsearch

elasticsearch ist eine quelloffene und verteilte Suchmaschine. Wie Solr basiert elasticsearch auf Lucene. Dies ist aber nicht die einzige Gemeinsamkeit zwischen den beiden Projekten. Darüber hinaus erweitern beide die Lucene-Schnittstellen in Richtung REST. elasticsearch unterstützt dabei nur die Kommunikation via JSON.

```
$ curl -XPUT http://localhost:9200/twitter/tweet/1 -d'{
    „user“: „esenti“,
    „post_data“: „2011-09-15T15:00:00“,
    „message“: „We look forward to DOAG K+A 2011!“
}
```

In diesem Beispiel wird ein Tweet erstellt. Via curl wird eine JSON Nachricht an den elasticsearch Server gesendet. Dabei ist „twitter“ der Index in dem die Tokens abgespeichert werden sollen. „tweet“ ist der Typ der Nachricht. Neben der eigentlichen Nachricht, enthält jeder Tweet einige Metadaten, wie Username oder Veröffentlichungsdatum.
Bevor man den Index durchsuchen kann muss dieser mit dem Befehl „`_refresh`“ erneuert werden. Man hat die Möglichkeit selbst zu entscheiden wann der Index erneuert wird und muss keinen vorgegebene Zyklus abwarten. Dadurch ist elasticsearch nahezu echtzeitfähig. Möchte man im nächsten Schritt nach einem Tweet suchen, funktioniert dies ebenfalls via JSON:

```
$ curl -XGET http://localhost:9200/twitter/tweet/_search -d'{
    „query“ : {
        „term“ : { „user“: „esentri“}
    }
}'
```

Im Index „twitter“ wird nach einem Eintrag vom Typ „tweet“ gesucht, welcher vom Anwender „esentri“ erstellt wurde. Es ist nicht notwendig den Typ oder den Index mitanzugeben. Via „`_all`“ können alle Indizes durchsucht werden.
elasticsearch ist dafür konzipiert auf Clustern in der Cloud zu laufen. Werden zwei oder mehr elasticsearch Server auf dem gleichen Rechner gestartet, werden diese automatisch miteinander verbunden. Wird elasticsearch in der Cloud verwendet, gibt es Schnittstellen über welche die Adressen der anderen Applikationen übermittelt und die beiden miteinander verbunden werden können. elasticsearch unterstütz seit der Version 0.8 die Amazon EC2 Cloud.

Oracle

Wenn es um die Anbindung einer RDBMS geht, unterscheiden sich die beiden Projekte bereits im Grundsatz. elasticsearch ist darauf ausgelegt die noSQL Bewegung zu unterstützen. Es ist möglich die Quellen, welche einem Index hinzugefügt werden mit anzuhängen. Dadurch können die Daten unabhängig von einer Datenbank gespeichert werden. Mittels der schnellen Suche über Lucene kann man auf die Daten zugreifen.
Solr hingegen implementiert eigene Datenbankhandler und stellt die nötigen Mittel zur Verfügung um mit Datenbanken interagieren zu können. Dadurch ist es möglich die schnelle Suche direkt auf die Datenbank anzuwenden.

Vergleich

	Solr	Elasticsearch
Entwicklung	2004	ca. 2010
Programmiersprache Core	Java	Java
Echtzeitfähig	nein	fast
Cloud	Via SolrCloud	Direkt

Dokumenten-Typen	Via Apache Tika alle gänigen	Schema frei
Schnittstellen	JSON, XML, Java	JSON
Lizenz	Apache 2	Apache 2
RDBMS	Ja	noSQL

Kontaktadresse:

Frank Szilinski, B.Sc.
esentri software GmbH
Pforzheimer Straße 132
D-76275 Ettlingen

Dominic Weiser, B.Sc.
esentri consulting GmbH
Pforzheimer Straße 132
D-76275 Ettlingen

Telefon: +49 (0) 7243-354 90 0
Fax: +49 (0) 7243-354 90 99
E-Mail: frank.szilinski@esentri.com
Internet: www.esentri.com