Bildersuche mit Hadoop und MapReduce

Verena Dittmer, Mario Graf

10. Juli 2015

1 Umsetzung

Wir haben unsere Implementierung in zwei Module aufgeteilt: `hd-image-search-core` und `hd-image-search-gui`. Als Feature wurde Lire's CEDD Feature verwendet. Um die Distanzen zu berechnen wurde die Euklidische Distanz eingesetzt.

Die Vollständige Implementierung ist unter GitHub erreichbar:

```
https://github.com/hawk23/hd-image-search
```

1.1 hd-image-search-core

Dieses Modul beinhaltet die MapReduce Task um Features zu extrahieren und für die Bildsuche an sich. Das zugehörige jar File `hd-core.jar` bedfindet sich im Ordner `jars`.

1.1.1 Verwendung

Um die MapReduce tasks in einem Hadoop Cluster ausführen zu können muss das entsprechende jar file zuerst bekannt gegeben werden:

```
$ export HADOOP_CLASSPATH=/home/hduser/jars/hd-core.jar
```

Features können mit folgendem Befehl extrahiert werden. Der erste Parameter ist der Ordner im Hadoop Cluster der die Bilder beinhaltet, der zweite Parameter ist der Ordner in dem dann die Indexdatei erstellt werden soll.

```
$ hadoop hdimagesearch.core.FeatureExtract /images/png2 /index
```

Um eine Bildsuche zu starten können folgende Befehle ausgeführt werden. Als erster Parameter muss man den Ordner angeben in dem sich der Index befindet, dann den Ordner in dem das Suchergebnis gespeichert werden soll. Mit `-n` kann man die Anzahl der gewünschten Suchergenisse angeben. Als Suchbild kann man entweder ein bereits im Hadoop Cluster gespeichertes Bild angeben (`-h "/images/png2/1/1_i110.png"`) oder einen Featurestring, wenn man bereits die Features eines Bildes extrahiert hat (`-f "1.0;3.0:1.0;0.0;....."`).

```
$ hadoop hdimagesearch.core.ImageSearch /index /searchResult -n 10
    -h "/images/png2/1/1_i110.png"
$ hadoop hdimagesearch.core.ImageSearch /index /searchResult -n 10
    -f "1.0;3.0:1.0;0.0;..."
```

Anschließend kann man sich das Ergebnis der Suche anzeigen lassen:

```
$ hadoop fs -cat /searchResult/part-r-00000
$
$ 0.0        hdfs://localhost:54310/images/png2/1/1_i110.png
$ 1.41421356 hdfs://localhost:54310/images/png2/1/1_i120.png
$ 1.73205080 hdfs://localhost:54310/images/png2/1/1_i130.png
$ 2.0        hdfs://localhost:54310/images/png2/1/1_i140.png
$ 2.23606797 hdfs://localhost:54310/images/png2/1/1_i150.png
$ 3.0        hdfs://localhost:54310/images/png2/1/1_i160.png
$ 3.60555127 hdfs://localhost:54310/images/png2/1/1_i170.png
$ 3.87298334 hdfs://localhost:54310/images/png2/1/1_i180.png
$ 4.69041575 hdfs://localhost:54310/images/png2/1/1_i230.png
$ 4.69041575 hdfs://localhost:54310/images/png2/1/1_i190.png
```

1.2 hd-image-search-gui

Dieses Modul beinhaltet die grafische Benutzeroberfläche um Features zu extrahieren und um eine Bildsuche durchzuführen. Das Modul baut auf `hd-image-search-core` auf und nutzt die dort definierten MapReduce Tasks. Das jar file, mit dem sich die Benutzeroberfläche starten lässt, findet man unter `jars/hd-gui.jar`.

Gestartet werden kann die GUI einfach mit:

```
$ java -jar hd-gui.jar
```

Zuerst muss man auswählen auf welchem Cluster man die Operationen ausführen mächte. Im oberen Bereich der Benutzeroberfläche findet man dazu eine Auswahlbox um die gewünschte Clusterkonfiguration auszuwählen. Über den Button *Add Config ...* kann man schnell eine neue Konfiguration anlegen.

Um Features extrahieren zu können muss man unter dem Tab *Feature Extract* auf den Button *Extract Features* klicken. Im Log Fenster im unteren Bereich der Oberfläche werden Logeinträge zu Erfolgs- und Fehlerfällen ausgegeben (Abbildung 1.1).

Für die Bildersuche muss zuerst unter dem Tab *Search* mittels *Select Image ...* ein Bild aus dem lokalen Dateisystem ausgewählt werden. Das Bild wird als Vorschau rechts neben den Einstellungen angezeigt. Weiters kann man auch die Anzahl der gewünschten Ergebnisse angeben. Mit einem Klick auf *Start Search* kann die Suche gestartet werden und die Ergebnisse werden angezeigt (Abbildung 1.2).

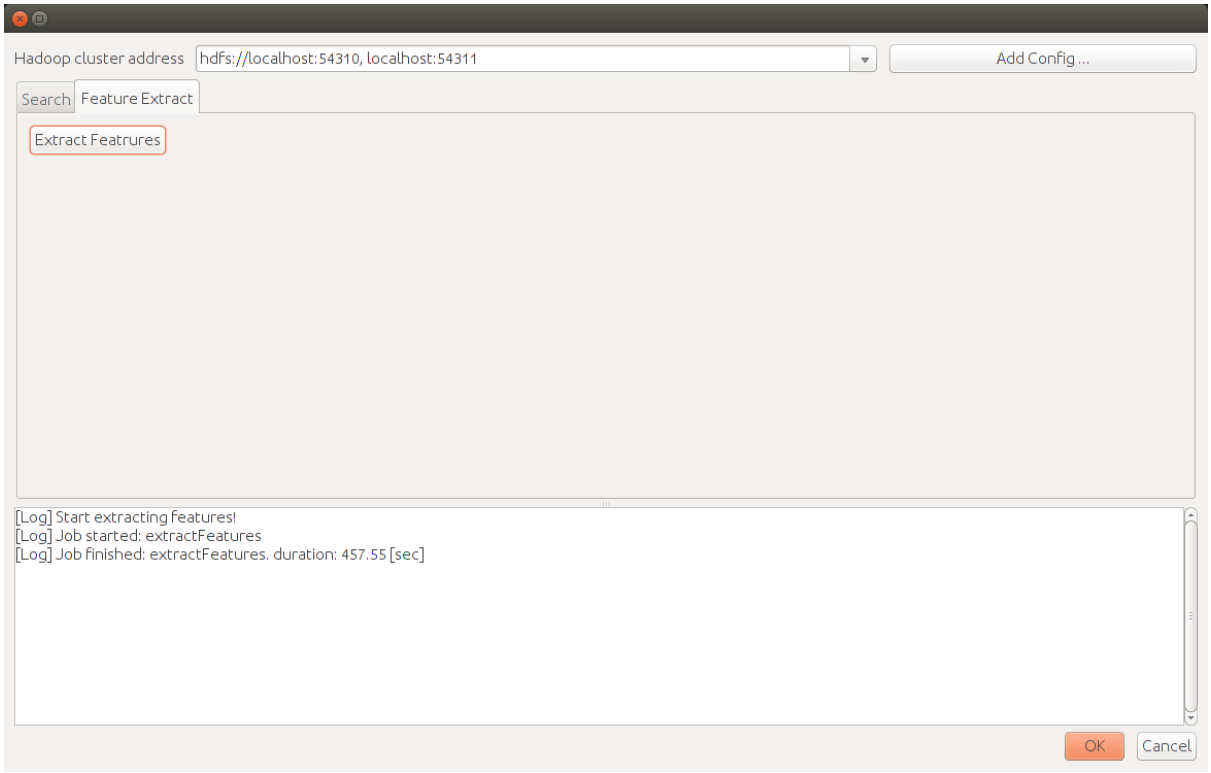

Abbildung 1.1: Eingabemaske um Features zu extrahieren.

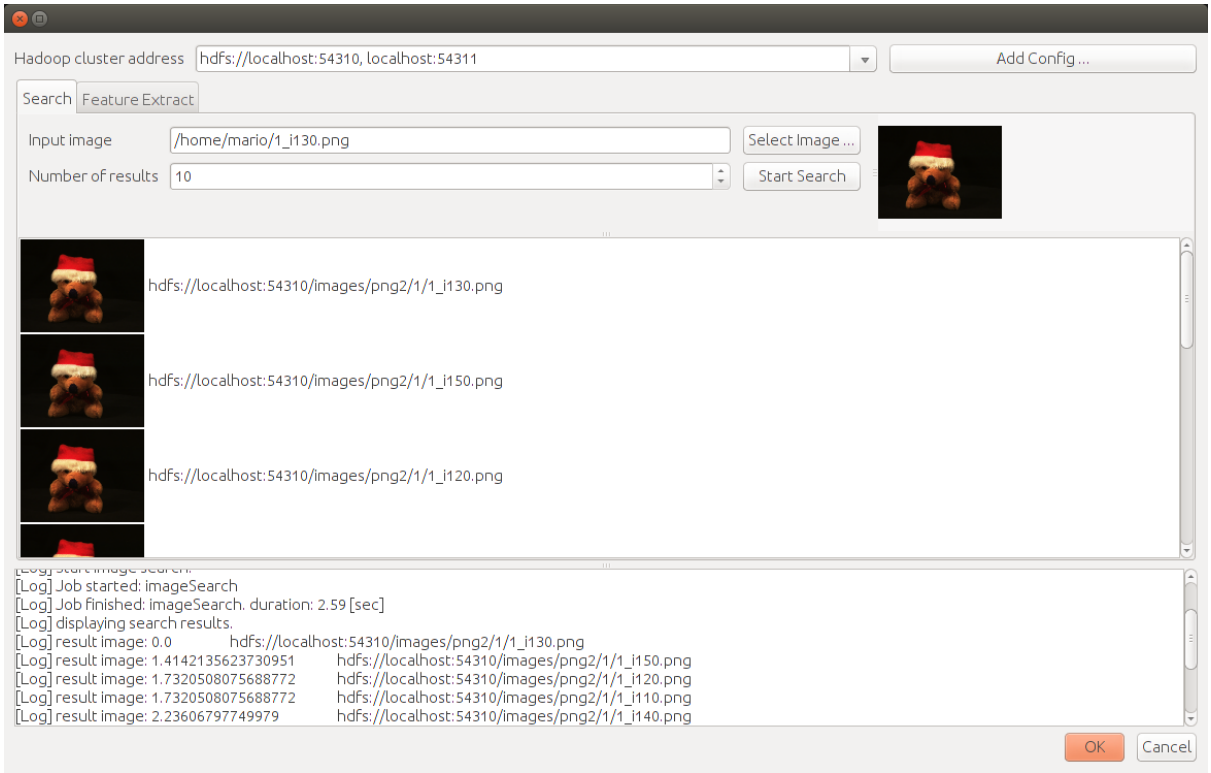

Abbildung 1.2: Eingabemaske um eine Bildersuche zu starten.

1.3 Verwenden des Uni Clusters

Das erzeugte jar File wurde auch auf dem Uni Cluster octopus-itec.aau.at getestet. Die in 1.1.1 beschriebenen Kommandos können dort (nach Anpassung der Dateipfade) ausgeführt werden.

1.4 Aufsetzen eines verteilten Multi Node Clusters

Hadoop wurde zudem auf zwei virtuellen Maschinen auf Virtualbox (jeweils Ubuntu 14.04) aufgesetzt. Dabei ist eine virtuelle Maschine Master und Slave zugleich und die andere nur ein Slave. Damit die beiden virtuellen Maschinen miteinander kommunizieren können, wurde ein Host-Only Network `vboxnet0` aufgesetzt. Die Einstellungen sind in Abbildung 1.3 zu sehen. Den virtuellen Maschinen konnte somit statische IP-Addressen zugeteilt werden.

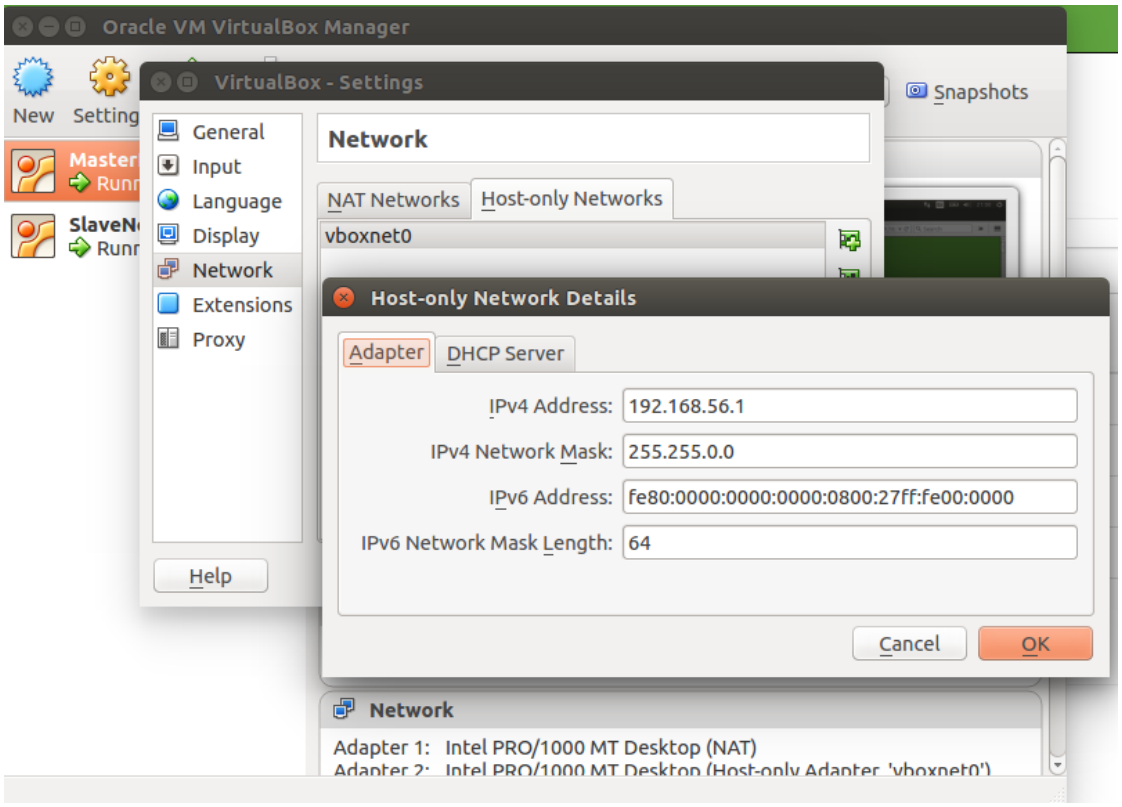

Abbildung 1.3: Einstellungen für das Host-Only Network

Der Master Node hat sowohl den *NameNode* Daemon für das HDFS laufen als auch den *ResourceManager* (früher *JobTracker*) für das MapReduce. Beide Maschinen haben die Slave Daemons laufen: *DataNode* auf dem HDFS Layer und den *NodeManager* auf dem MapReduce Layer.

Dies muss in den Konfigurationsdateien angepasst werden. So müssen die Dateien `core-site.xml`, `mapred-site.xml` and `hdfs-site.xml` auf beiden Knoten verändert werden und die `masters` und `slaves`-Dateien auf dem Master.

In Abbildung 1.4 sind die laufenden Daemons nach dem Ausführen von `start-dfs.sh` und `start-yarn.sh` auf dem Master Node zu sehen.

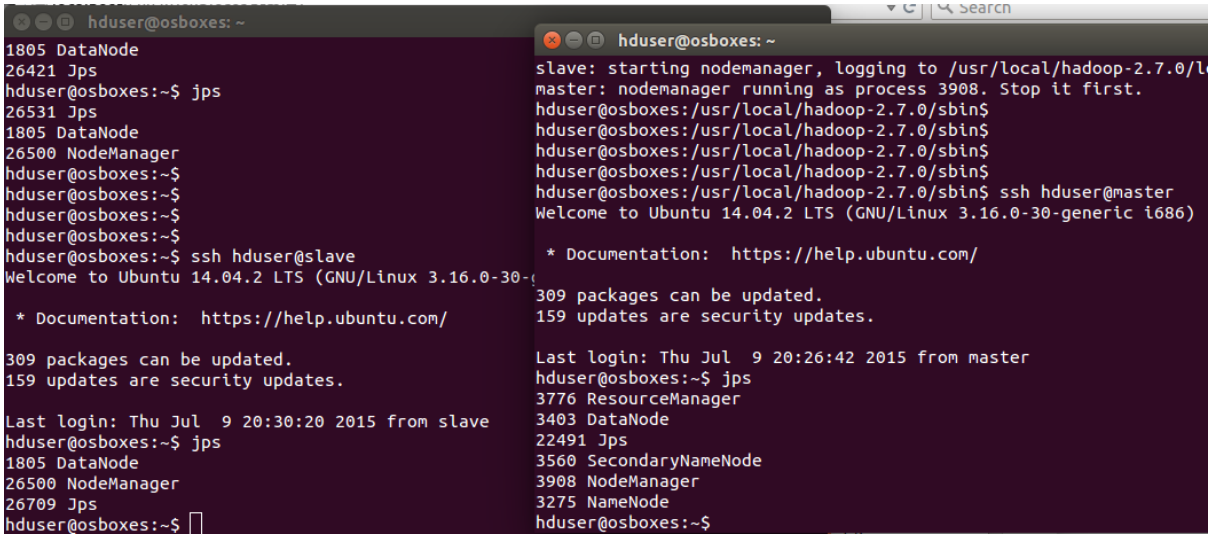

Abbildung 1.4: Daemons auf SlaveNode (links) und MasterNode (rechts)

Nachdem das jar File auf die virtuelle Maschine geladen wurde, kann es wie unter 1.1.1 beschrieben, verwendet werden.

1.4.1 Performance

Getestet wurde der Multi Node Cluster auf einem Intel Core i7(Quad Core) 16 GB RAM.

	1 Node	2 Nodes	4 Nodes
FeatureExtract	79 Min	45 Min	15 Min
ImageSearch	77 Sek	51 Sek	39 Sek

2 Erledigte Aufgaben

Von den zu erledigenden Aufgaben haben wir alle Plichtaufgaben (1. - 4.) erfüllt. Von den Zusatzaufgaben haben wir umgesetzt:

- Modify your code from task 4 so that the returned result contains 10 of the most similar images and their distance index.
- Implement a GUI where the user can search for similar image by loading an image from its local system. Once the search completes the user should be able to see the 10 most similar images.
- Test your code in a fully distributed environment using one of the following techniques.
  - Use our provided servers to test your code in a fully distributed environment.
  - Set up a fully distributed cluster and test your implementation there.
- Increase the number of nodes in the cluster and check the performance of your code with 1, 2, 3, 4 or more different nodes.