Paralel KMeans, Hadoop

K-Means algoritmasini nasil paralel sekilde isletiriz? Ozellikle Hadoop gibi bir Esle-Indirge (Map-Reduce) ortamini dusunelim. Veri cok buyuk olcekte olabilir ve bu veriler birden fazla makinaya bolunecektir. Esle-Indirge kavraminda esleme safhasinda "anahtar uretiriz", ve sonra indirgeme safhasinda Hadoop sistemi oyle kurmustur ki ayni anahtarlarlar tek bir makinaya gonderilir, ve bu nihai asamada artik anahtar bazinda indirgeme (ozetleme) yapilir.

Paralel K-Means icin anahtar nedir?

Anahtar, mesela kume olabilir. Yani kume 1, kume 2 gibi kume isaretleri / sayilari anahtar olarak kullanilabilirler.

Peki anahtar ile eslenecek "deger" nedir?

Oyle bir deger ariyoruz ki ust uste konulabilecek bir sey olmali, cunku EI sisteminin kuvveti burada, anahtarlar farkli noktalarda uretilebiliyor, sonra tek noktada ust uste konuyor, o zaman degerler oyle uretilmeli ki bu ust uste koyma, ozetleme islemi yapilabilsin.

Ust uste konabilecek sey kumeye (anahtar) ait olan veri noktasi olabilir. Normal K-Means'i hatirlarsak, her nokta icin o noktaya en yakin kumeyi buluyordu sonra, atama islemi bitince, her kumenin altindaki noktalarin toparlayip, onlarin ortalamasini alarak yeni kume merkezini hesapliyordu. Bu ortalama islemi ust uste konabilecek bir sey, yani farkli makinalarda kume-nokta eslemelerini uretirsek, indirgeme asamasinda o anahtar icin tum degerleri toplayip, nokta sayisina boleriz ve yeni kume merkezini elde ederiz.

```
from IPython.display import Image
Image(filename='kmeans-diag.png')
```

```
<IPython.core.display.Image at 0xc5db98c>
```

Simdi Hadoop ile ilgili bazi lojistik konulara gelelim:

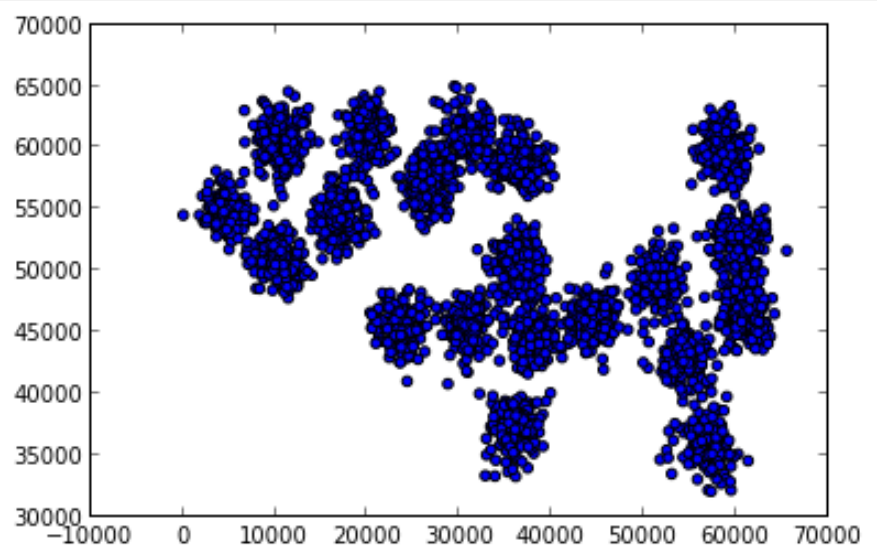
Eger esleme seviyesinde en yakin kumeyi bulmak istiyorsak, o zaman (ilk basta rasgele bile olsa) kume merkezlerinin bilgisi tum makinalarin erisebilecegi bir yerde olmali. Biz bu veriyi HDFS uzerinde tutmaya karar verdik, dosya ismi bilinen (well-known) bir yerde olacak, /tmp/centers.csv.

K-Means tek bir esle-indirge isletimi ile bitmeyecek, bu algoritma dongulu / ozyineli (iterative) bir algoritma, 5,10,20 kez islemesi gerekebilir. Her dongu sonunda yeni kume merkezleri hesaplanacak, bu merkezler eski /tmp/centers.csv yerini alacak ve islem tekrar baslayacak.

Dongu sonundaki merkez bilgisi indirgeyicinin ciktisidir ve bu cikti output/part-00000 dosyasi icinde. Unutmayalim, indirgeyicinin ciktisi demek, tum indirgeyici makinalardan gelen anahtarlarin (ozetleme ardindan) birlestirilmesi sonrasinda demek.

```
from pandas import *
df1 = read_csv("synthetic.txt",sep=" ")
plt.scatter(df1.ix[:,0],df1.ix[:,1])
```

```
<matplotlib.collections.PathCollection at 0xc051e8c>
```




```
!ssh localhost -l hduser /home/hduser/Downloads/hadoop*/bin/stop-all.sh
!ssh localhost -l hduser /home/hduser/Downloads/hadoop*/bin/start-all.sh
```

```
stopping jobtracker

localhost: stopping tasktracker

stopping namenode

localhost: stopping datanode

localhost: stopping secondarynamenode

starting namenode, logging to /home/hduser/Downloads/hadoop-1.0.4/libexec/../logs/hadoop-hduser-namenode

localhost: starting datanode, logging to /home/hduser/Downloads/hadoop-1.0.4/libexec/../logs/hadoop-hdus

localhost: starting secondarynamenode, logging to /home/hduser/Downloads/hadoop-1.0.4/libexec/../logs/ha

starting jobtracker, logging to /home/hduser/Downloads/hadoop-1.0.4/libexec/../logs/hadoop-hduser-jobtra

localhost: starting tasktracker, logging to /home/hduser/Downloads/hadoop-1.0.4/libexec/../logs/hadoop-h
```

```
!ssh localhost -l hduser /home/hduser/Downloads/hadoop*/bin/hadoop dfs -mkdir /tmp/
```

```
!ssh localhost -l hduser /home/hduser/Downloads/hadoop*/bin/hadoop dfs -ls /
```

```
Found 3 items
drwxr-xr-x   - hduser supergroup          0 2013-02-25 17:23 /app
drwxr-xr-x   - hduser supergroup          0 2013-02-26 12:49 /tmp
drwxr-xr-x   - hduser supergroup          0 2013-02-26 11:45 /user
```

```
!ssh localhost -l hduser /home/hduser/Downloads/hadoop*/bin/hadoop dfs -copyFromLocal
    $HOME/Documents/classnotes/stat/stat_hadoop_kmeans/synthetic.txt /user/hduser
```

```
copyFromLocal: Target /user/hduser/synthetic.txt already exists
```

```
print open("mapper.py").read()
```

```
#!/usr/bin/python
import os,sys,itertools
import numpy as np
from numpy import linalg as la
os.environ['MPLCONFIGDIR']='/tmp'
import pandas as pd

centers = pd.read_csv("/tmp/centers.csv",header=None,sep=",")

def dist(vect,x):
    return np.fromiter(itertools.imap(np.linalg.norm, vect-x),dtype=np.float)

def closest(x):
    d = dist(np.array(centers)[:,1:3],np.array(x))
    return np.argmin(d)

comb = lambda x: str(x[0])+":"+str(x[1])

df = pd.read_csv(sys.stdin,header=None,sep="   ")
df['cluster'] = df.apply(closest,axis=1)
df['coord'] = df.apply(comb,axis=1)
df.to_csv(sys.stdout, sep='\t',index=False, cols=['cluster','coord'],
          header=None)
```

```
print open("reducer.py").read()
```

```
#!/usr/bin/python
import os,sys,itertools
import numpy as np
from numpy import linalg as la
os.environ['MPLCONFIGDIR']='/tmp'
import pandas as pd
```

```
def coords(x):
    return np.array(str(x).split(":"),dtype=np.float64)

def my_mean(x):
    return pd.Series(np.mean(x['coord_new']),index=['cluster','coord'])

df = pd.read_csv(sys.stdin,sep="\t",names=['cluster','coord'])
df['coord_new'] = df['coord'].apply(coords)
df2 = df.groupby('cluster').apply(my_mean)
df2.to_csv(sys.stdout, sep=',',header=None)
```

```
import os,sys,itertools
from numpy import linalg as la
import pandas as pd
k = 10
df = read_csv("synthetic.txt",header=None,sep=" ")
centers = df.take(np.random.permutation(len(df))[:k])
centers.to_csv("/tmp/centers.csv",header=None)
```

```
os.system("cp mapper.py /tmp/")
os.system("chmod a+r /tmp/mapper.py")
os.system("chmod a+x /tmp/mapper.py")

os.system("cp reducer.py /tmp/")
os.system("chmod a+r /tmp/reducer.py")
os.system("chmod a+x /tmp/reducer.py")

hp_cmd = "ssh localhost -l hduser /home/hduser/Downloads/hadoop*/bin/hadoop"
streaming_cmd = "/home/hduser/Downloads/hadoop*/contrib/streaming/hadoop-*streaming*.jar
    "

os.system("%s dfs -rm /user/hduser/centers.csv" % hp_cmd)
os.system("%s dfs -copyFromLocal /tmp/centers.csv /user/hduser" % hp_cmd)

for i in range(10):
    os.system("%s dfs -rmr /user/hduser/output" % hp_cmd)
    os.system("%s jar %s -input synthetic.txt -output output -mapper /tmp/mapper.py -
        reducer /tmp/reducer.py -numReduceTasks 1" % (hp_cmd,streaming_cmd))
    os.system("%s dfs -cp /user/hduser/output/part-00000 /user/hduser/centers.csv" %
        hp_cmd)
```

K-Means'i 10 kere islettikten sonra elde edilen sonuclari artik HDFS'ten yerel dizinimize alabiliriz. Nihai kume merkezleri output/part-00000 icinde. Bu merkezleri alip, ham veri uzerinde kirmizi nokta olarak gosteriyoruz.

```
os.system("ssh localhost -l hduser rm /tmp/part-00000")
os.system("ssh localhost -l hduser /home/hduser/Downloads/hadoop*/bin/hadoop dfs -
    copyToLocal /user/hduser/output/part-00000 /tmp/") #
```

```
0
```

```
df1 = read_csv("synthetic.txt",sep=" ")
plt.scatter(df1.ix[:,0],df1.ix[:,1])
plt.hold(True)
df2 = read_csv("/tmp/part-00000",header=None,index_col=0)
plt.plot(df2.ix[:,1],df2.ix[:,2],'rd')
```

```
[<matplotlib.lines.Line2D at 0xb14cf4c>]
```