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.

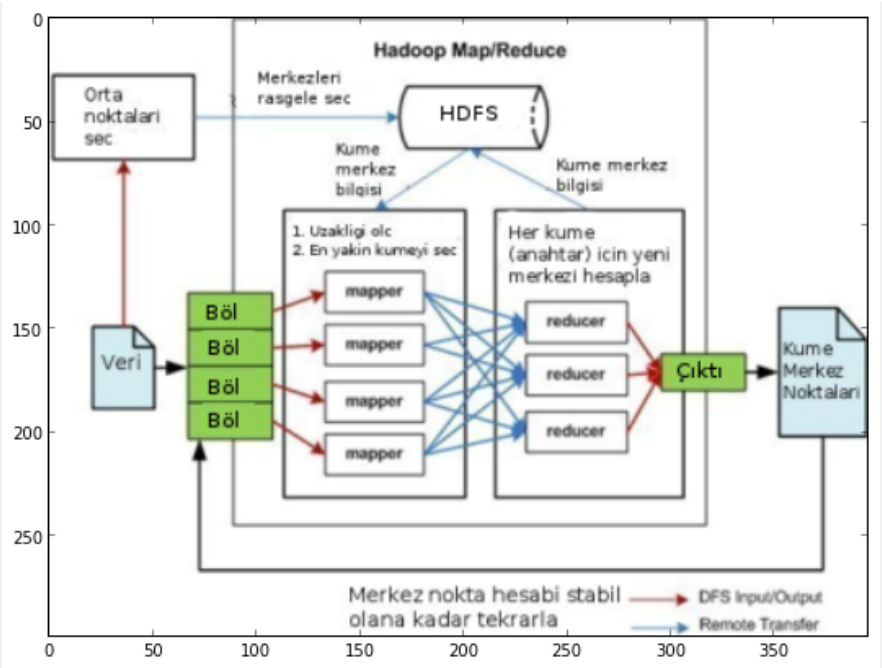
Peki anahtar ile eslenecek "deger" nedir?

Oyle bir deger ariyoruz ki ust uste konulabilecek bir sey olmali, 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, yani basbasyagi veri noktasinin kendisi deger olabilir. Normal K-Means'i hatirlarsak, her nokta icin o noktaya en yakin kumeyi buluyordu ve sonra, atama islemi bitince, her kumenin altindaki noktalarin toparlayip, onlarin ortalamasini alarak yeni kume merkezini hesapliyordu. Bu ortalama islemi ust uste konabilecek bir sey, cunku toplama oyle bir islem, ve / yani farkli makinalarda kume-nokta, eslemelerini uretirsek, indirgeme asamasinda o anahtar icin tum degerleri toplayip, nokta sayisina boleriz ve yeni kume merkezini elde ederiz.

```
figure(figsize=(9,9))
im=imread('kmeans-diag.png'); imshow(im)
```

```
<matplotlib.image.AxesImage at 0x2b25d90>
```



Simdi Hadoop ile ilgili bazi lojistik konulara gelelim:

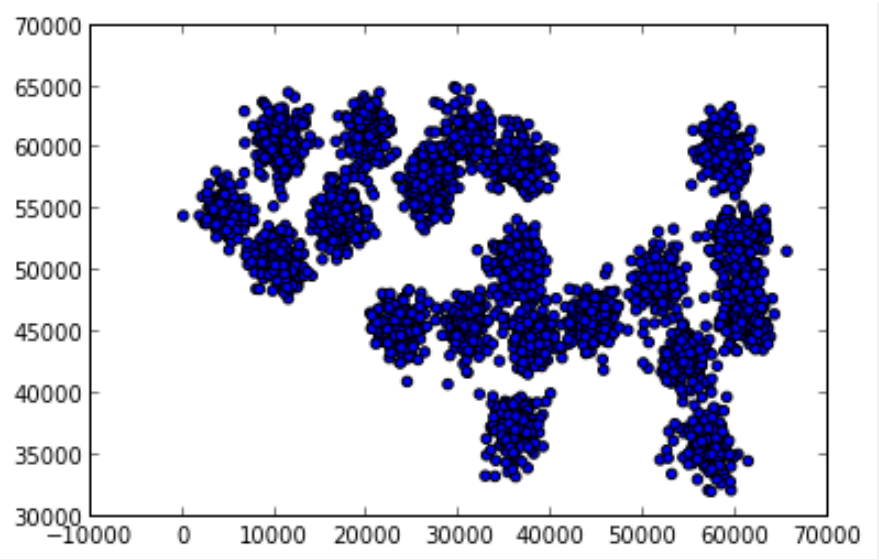
Eger esleme safhasinda her nokta icin en yakin kumeyi bulmak istiyorsak, o zaman (ilk basta rasgele bile olsa) kume merkezlerinin bilgisi tum makinalarin erisebilecegi bir yerde olmali. Biz bu veriyi, centers.csv adli bir dosyaya koymaya karar verdik, bu dosya tek makina ortaminda bilinen bir dizinde (mesela /tmp), cok makinali ortamda ise HDFS uzerinde herkesin erisebilecegi bir yerde olmali.

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

Simdi ham veriyi gosterelim,

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

```
<matplotlib.collections.PathCollection at 0xb3ea22c>
```




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

```
from mrjob.job import MRJob
from mrjob.protocol import PickleProtocol
import numpy as np, sys
import pandas as pd
import os, random

def euc_to_clusters(x,y):
    return np.sqrt(np.sum((x-y)**2, axis=1))

class MRKMeans(MRJob):
    INTERNAL_PROTOCOL = PickleProtocol

    def __init__(self, *args, **kwargs):
        super(MRKMeans, self).__init__(*args, **kwargs)
        self.centers_ = pd.read_csv("/tmp/centers.csv",header=None,sep="   ")
        self.k = 15

    def mapper(self, key, line):
        point = np.array(map(np.float,line.split('   ')))
        c = np.argmin(euc_to_clusters(np.array(self.centers_), point))
        yield(c, point)

    def reducer(self, key, tokens):
        new_centers = np.zeros((1,2))
        counts = 0
        for val in tokens:
            new_centers += val
            counts += 1
        yield('final', (key, new_centers[0] / counts))
```

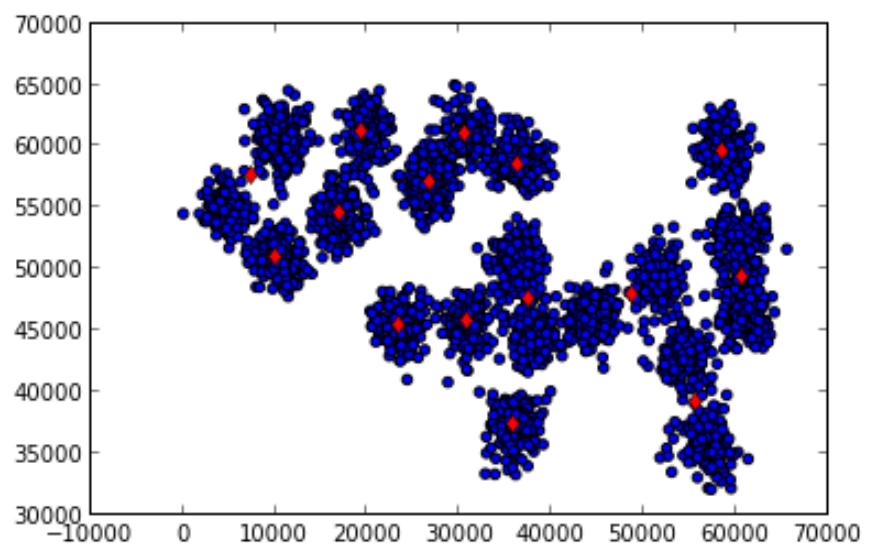
```
    def reduce_all_centers(self, key, values):
        new_centers = np.zeros((self.k,2))
        self.f=open("/tmp/centers.csv","w")
        for (cluster,val) in values:
            print cluster, val
            new_centers[cluster] = val
        for row in new_centers:
            self.f.write("   ".join(map(str,row)))
            self.f.write("\n")
        self.f.close()

    def steps(self):
        return [self.mr(mapper=self.mapper,reducer=self.reducer),
                self.mr(reducer=self.reduce_all_centers)]

if __name__ == '__main__':
    for i in range(15): MRKMeans.run()
```

reduce_all_centers cagrisi tum indirgeyiciler her kume icin yeni orta noktayi hesaplayip onu yayinladiktan (emit) sonra, tum yeni merkezlerin gelecegi yer.

Komut satirindan tek makina icin Hadoop'suz isletelim,

```
!sort --random-sort synthetic.txt > /tmp/synthetic.txt
!head -15 /tmp/synthetic.txt > /tmp/centers.csv
!python kmeans.py synthetic.txt
```

```
import pandas as pd
df1 = pd.read_csv("synthetic.txt",sep=" ",header=None)
plt.scatter(df1.ix[:,0],df1.ix[:,1])
plt.hold(True)
df2 = pd.read_csv("/tmp/centers.csv", sep=" ", header=None)
plt.plot(df2.ix[:,0],df2.ix[:,1],'rd')
plt.show()
```



K-Means'i 20 kere islettik. Eger istenirse (hatta daha iyi olur) dongu bir while icine konur ve bitis icin "stabilite sarti" aranir. Stabilite yeni kume merkezinin eskisinden "cok fazla degisik olup olmadigi" sartidir, degisim yoksa artik sonucu bulmusuz demektir, daha fazla donguye gerek kalmayacaktir. Biz donguyu 20 kere donguyu islettik, (bu problem icin) yeterli oldu.

K-Means isini bitirdikten sonra elde edilen sonuclari okuyabiliriz. Nihai kume merkezleri /tmp/-centers.csv icinde. Bu merkezleri alip, ham veri uzerinde kirmizi nokta olarak gosteriyoruz.

Sonuclar fena degil. Iste bu metotla terabayt olceginde, devasa bir veriyi 20-30 makinaya dagitarak parca parca isleyip kumelemeniz mumkundur. Endustride son zamanlarda habire duyulan Buyuk Veri (Big Data) olayi iste bu.