Cloud Computing

Spring 2017

Project 8*

Harp Mini-Batch Kmeans

Vaishnavi Mukundhan (*vaismuku@indiana.edu)*
Dwayne Dsouza (*dsouzad@indiana.edu*)

* *NOTE: We are using the free one time late submission for this project*

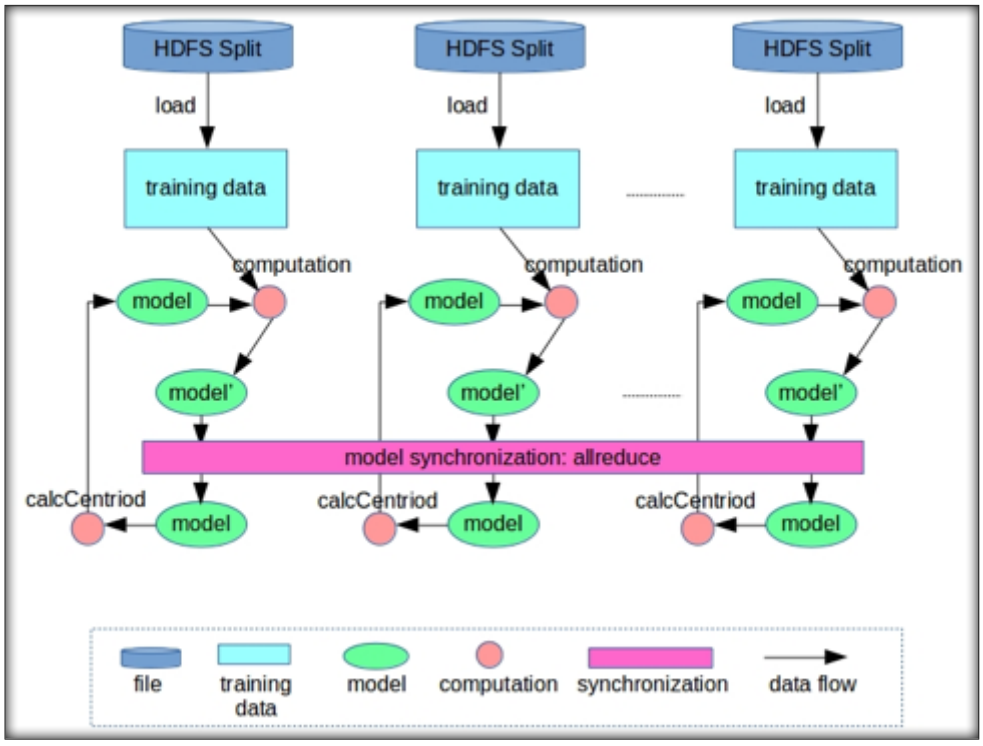
***Goal*:** The goal for this project is to implement Harp[1] Mini-Batch Kmeans (mbkmeans) from scratch.

Working Of the Program

The K-Means algorithm reiterates the following set of steps until there is no change in the partition assignments. In that, it has clarified which data point is assigned to which partition.

1. Choose K points as the initial set of centroids.
2. Assign each data point in the data set to the closest centroid (this is done by calculating the distance between the data point and each centroid).
3. Calculate the new centroids based on the clusters that were generated in step 2. Normally this is done by calculating the mean of each cluster.
4. Repeat step 2 and 3 until data points do not change cluster assignments, which means that their centroids are set. [1]

DATAFLOW



The program includes the following tasks:

1. The Main Method - Configures and runs the job iteratively.
2. The mapCollective function

   Reads data from context and then call runKmeans function to actually run kmeans Mapper task

```
protected void mapCollective( KeyValReader reader, Context context) thro
ws IOException, InterruptedException
```

3. The runKmeans function – We have used the AllReduce collective communication to do synchronization.

```
private void runKmeans(List<String> fileNames, Configuration conf, Conte
xt context) throws IOException
```

4. Computing local centroids

```
private void computation(Table<DoubleArray> cenTable, Table<DoubleArr
ay> previousCenTable,ArrayList<DoubleArray> dataPoints)
```

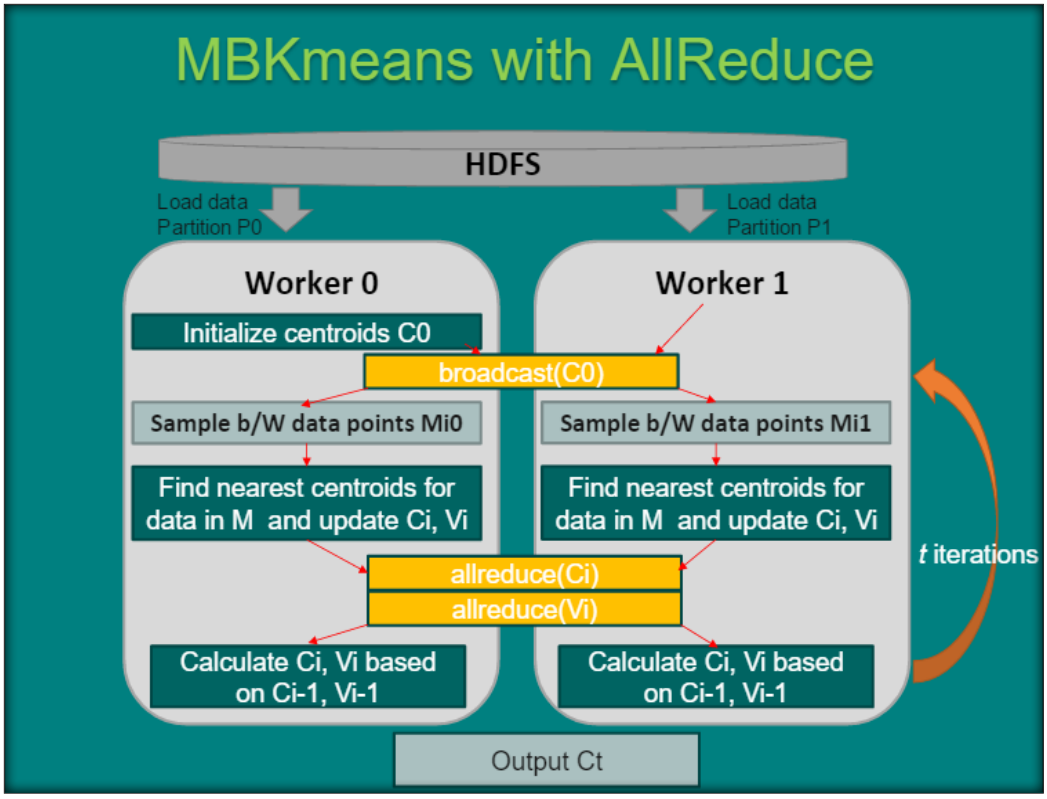
5. Calculate new centroids
   This function calculates the new centroids and prints the same.

```
private void calculateCentroids ( Table<DoubleArray> cenTable, Table<Do
ubleArray> previouscentable)
```

Algorithm:

Algorithm 1 Mini-batch k-Means.

Given: k, mini-batch size b, iterations t, data set X
Initialize each $\mathbf{c} \in C$ with an $\mathbf{x}$ picked randomly from X
$\mathbf{v} \leftarrow 0$
for $i = 1$ to t **do**
  $M \leftarrow b$ examples picked randomly from X
  for $\mathbf{x} \in M$ **do**
    $\mathbf{d}[\mathbf{x}] \leftarrow f(C, \mathbf{x})$ // Cache the center nearest to $\mathbf{x}$
  end for
  for $\mathbf{x} \in M$ **do**
    $\mathbf{c} \leftarrow \mathbf{d}[\mathbf{x}]$ // Get cached center for this $\mathbf{x}$
    $\mathbf{v}[\mathbf{c}] \leftarrow \mathbf{v}[\mathbf{c}] + 1$ // Update per-center counts
    $\eta \leftarrow \frac{1}{\mathbf{v}[\mathbf{c}]}$ // Get per-center learning rate
    $\mathbf{c} \leftarrow (1 - \eta)\mathbf{c} + \eta\mathbf{x}$ // Take gradient step
  end for
end for



Compiling the code:
cd $HARP_ROOT_DIR
mvn clean package
cd $HARP_ROOT_DIR/harp-tutorial-app
cp target/harp-tutorial-app-1.0-SNAPSHOT.jar $HADOOP_HOME
cd $HADOOP_HOME

Run Harp K-Means:

```
hadoop jar harp-tutorial-app-1.0-SNAPSHOT.jar edu.iu.mbkmeans.common.MBKmeansMapCollective <numOfDataPoints> <num of Centroids> <size of vector> <number of map tasks> <number of iteration> <workDir> <localDir> <communication operation>
```

Here,

<workDir>: This is the root directory running in HDFS.
<localDir>: This argument is the local directory where mbkmeans stores the data points that it generates.
<communication operation>: set this to “allreduce”

Output and Experimentations:

Below are the screen shots of our output. For **Bonus Credits,** we experimented with the input size, dimensions or the vector size and the number of Centroids.

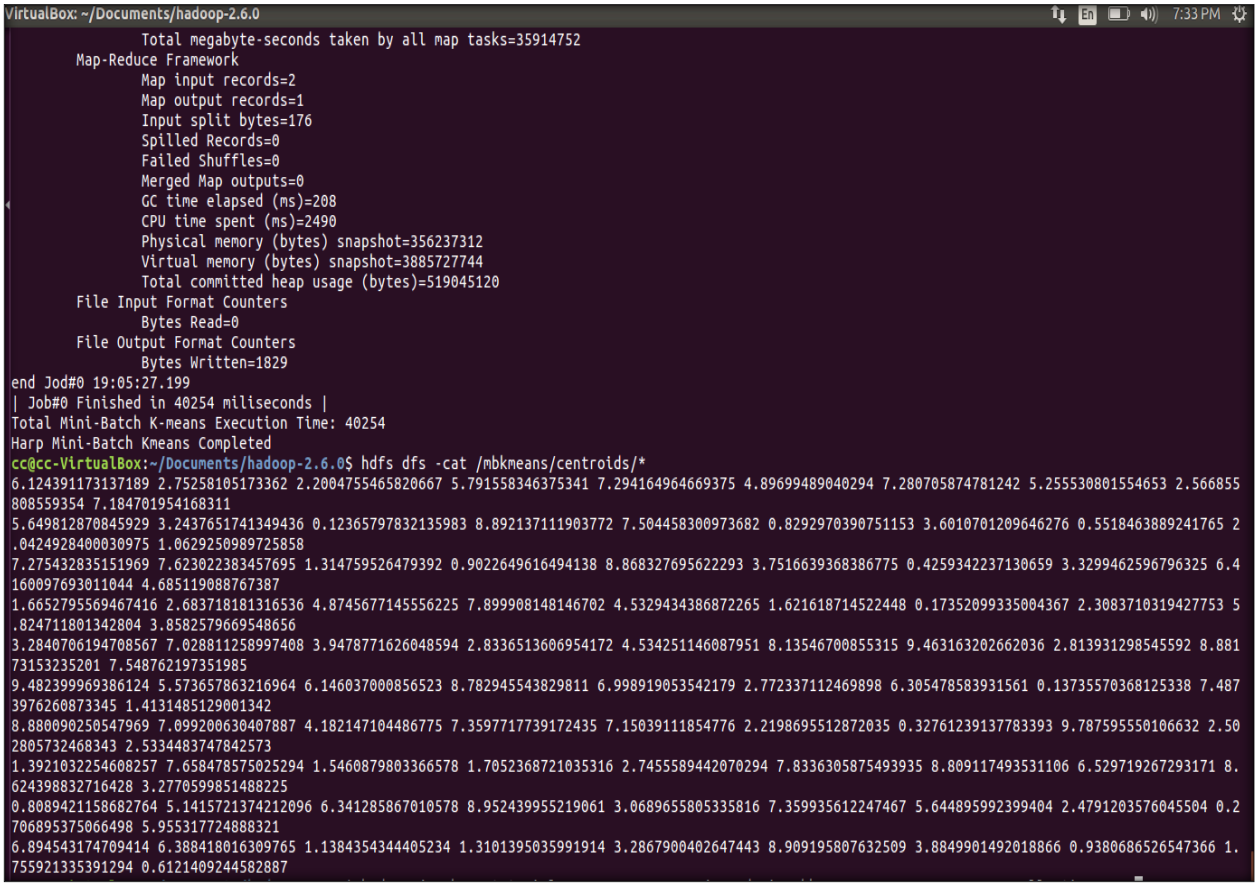
Example command to Run MBKmeans:
hadoop jar harp-tutorial-app-1.0-SNAPSHOT.jar
edu.iu.mbkmeans.common.MBKmeansMapCollective 100 5 10 100 2 10
/mbkmeans /tmp/minibatchkmeans allreduce

1. ***VARYING THE INPUT SIZE:***

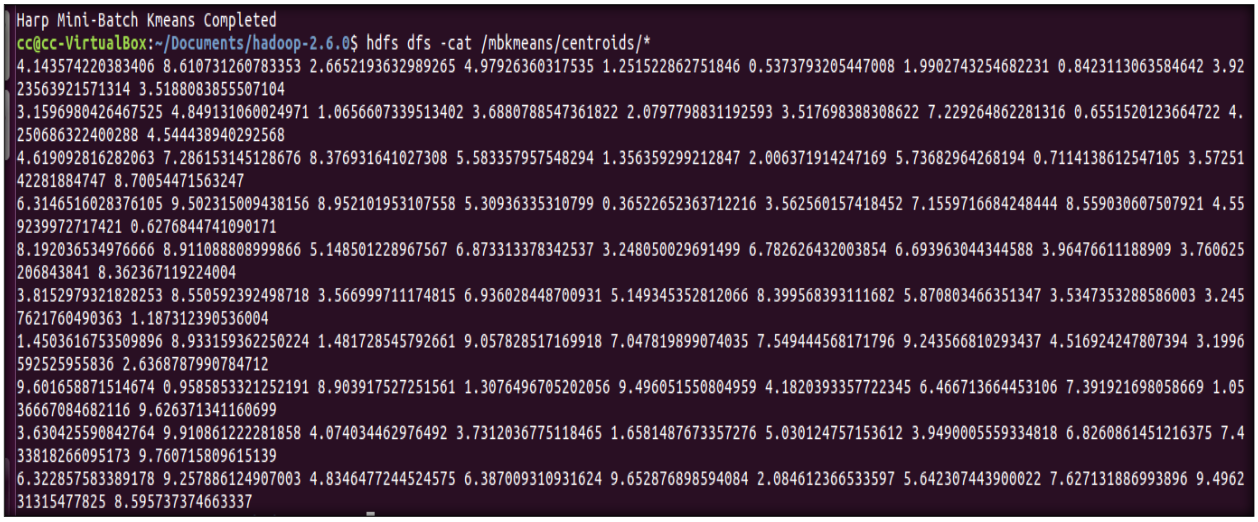
    1. Data-Points: 1000

    6.124391173137189 2.75258105173362 2.2004755465820667
    5.791558346375341 7.294164964669375 4.89699489040294
    7.280705874781242 5.255530801554653 2.566855808559354
    7.184701954168311
    5.649812870845929 3.2437651741349436 0.12365797832135983
    8.892137111903772 7.504458300973682 0.8292970390751153
    3.6010701209646276 0.5518463889241765 2.0424928400030975
    1.0629250989725858
    7.275432835151969 7.623022383457695 1.314759526479392
    0.9022649616494138 8.868327695622293 3.7516639368386775
    0.4259342237130659 3.3299462596796325 6.4160097693011044
    4.685119088767387
    1.6652795569467416 2.683718181316536 4.8745677145556225
    7.899908148146702 4.5329434386872265 1.621618714522448
    0.17352099335004367 2.3083710319427753 5.824711801342804
    3.8582579669548656
    3.2840706194708567 7.028811258997408 3.9478771626048594
    2.8336513606954172 4.534251146087951 8.13546700855315
    9.463163202662036 2.813931298545592 8.88173153235201
    7.548762197351985
    9.482399969386124 5.573657863216964 6.146037000856523
    8.782945543829811 6.998919053542179 2.772337112469898

6.305478583931561 0.13735570368125338 7.4873976260873345
1.4131485129001342
8.880090250547969 7.099200630407887 4.182147104486775
7.3597717739172435 7.15039111854776 2.2198695512872035
0.32761239137783393 9.787595550106632 2.502805732468343
2.5334483747842573
1.3921032254608257 7.658478575025294 1.5460879803366578
1.7052368721035316 2.7455589442070294 7.8336305875493935
8.809117493531106 6.529719267293171 8.624398832716428
3.2770599851488225
0.8089421158682764 5.1415721374212096 6.341285867010578
8.952439955219061 3.0689655805335816 7.359935612247467
5.644895992399404 2.4791203576045504 0.2706895375066498
5.955317724888321
6.894543174709414 6.388418016309765 1.1384354344405234
1.3101395035991914 3.2867900402647443 8.909195807632509
3.8849901492018866 0.9380686526547366 1.755921335391294
0.6121409244582887



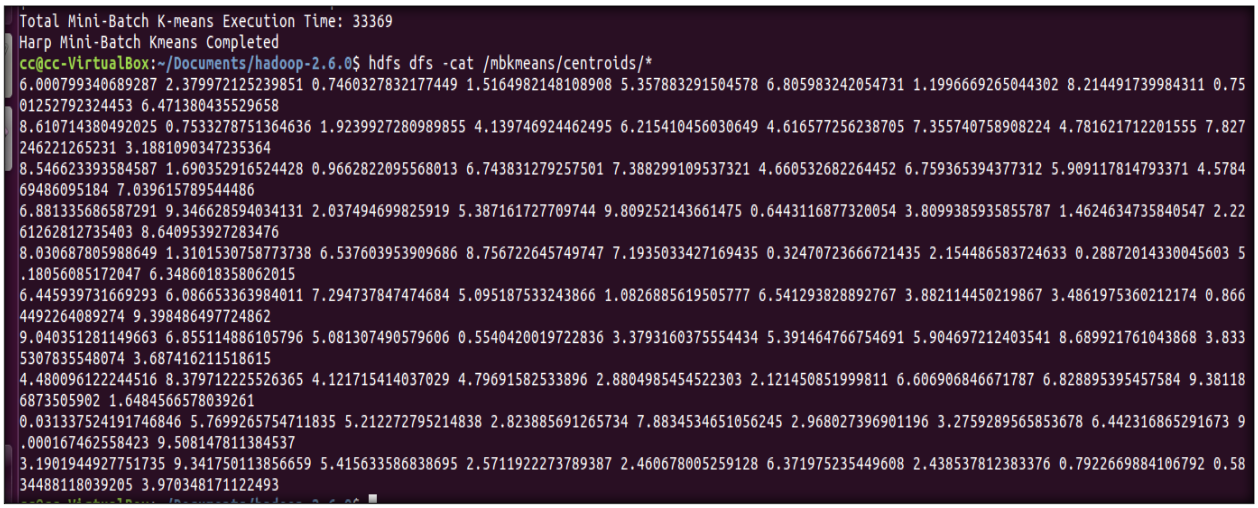
2. Input Size: 10,000



4.143574220383406 8.610731260783353 2.6652193632989265
4.97926360317535 1.251522862751846 0.5373793205447008
1.9902743254682231 0.8423113063584642 3.9223563921571314
3.5188083855507104
3.1596980426467525 4.849131060024971 1.0656607339513402
3.6880788547361822 2.0797798831192593 3.517698388308622
7.229264862281316 0.6551520123664722 4.250686322400288
4.544438940292568
4.619092816282063 7.286153145128676 8.376931641027308
5.583357957548294 1.356359299212847 2.006371914247169
5.73682964268194 0.7114138612547105 3.5725142281884747
8.70054471563247
6.3146516028376105 9.502315009438156 8.952101953107558
5.30936335310799 0.36522652363712216 3.562560157418452
7.1559716684248444 8.559030607507921 4.559239972717421
0.6276844741090171
8.192036534976666 8.911088808999866 5.148501228967567
6.873313378342537 3.248050029691499 6.782626432003854
6.693963044344588 3.96476611188909 3.760625206843841
8.362367119224004
3.8152979321828253 8.550592392498718 3.566999711174815
6.936028448700931 5.149345352812066 8.399568393111682
5.870803466351347 3.5347353288586003 3.2457621760490363

1.187312390536004
1.4503616753509896 8.933159362250224 1.481728545792661
9.057828517169918 7.047819899074035 7.549444568171796
9.243566810293437 4.516924247807394 3.1996592525955836
2.6368787990784712
9.601658871514674 0.9585853321252191 8.903917527251561
1.3076496705202056 9.496051550804959 4.1820393357722345
6.466713664453106 7.391921698058669 1.0536667084682116
9.626371341160699
3.630425590842764 9.910861222281858 4.074034462976492
3.7312036775118465 1.6581487673357276 5.030124757153612
3.9490005559334818 6.8260861451216375 7.433818266095173
9.760715809615139
6.322857583389178 9.257886124907003 4.8346477244524575
6.387009310931624 9.652876898594084 2.084612366533597
5.642307443900022 7.627131886993896 9.496231315477825
8.595737374663337

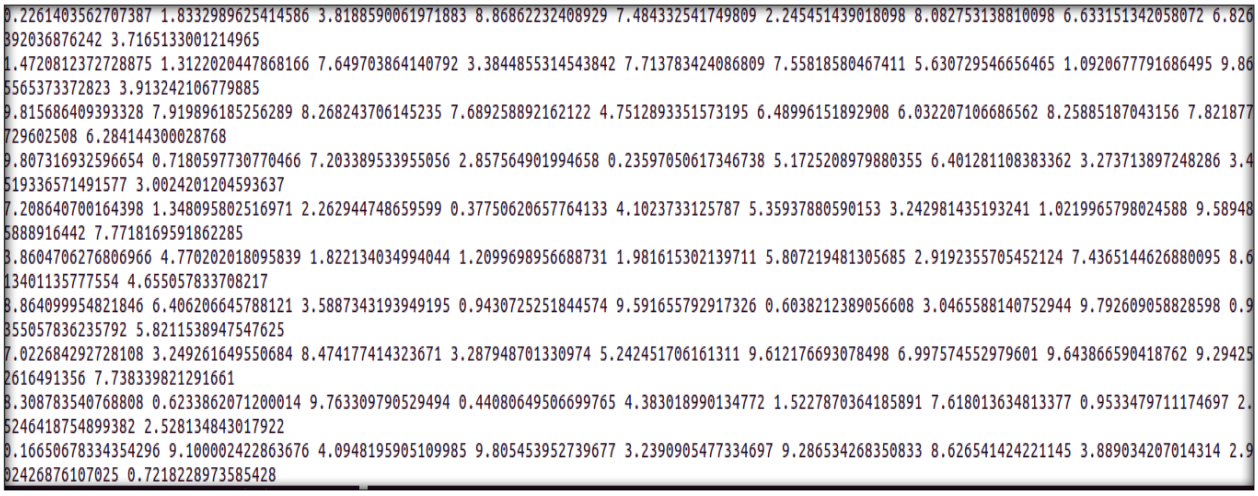
3. Input Size: 100000



6.000799340689287 2.379972125239851 0.7460327832177449
1.5164982148108908 5.357883291504578 6.805983242054731
1.1996669265044302 8.214491739984311 0.7501252792324453
6.471380435529658
8.610714380492025 0.7533278751364636 1.9239927280989855
4.139746924462495 6.215410456030649 4.616577256238705
7.355740758908224 4.781621712201555 7.827246221265231

3.1881090347235364
8.546623393584587 1.690352916524428 0.9662822095568013
6.743831279257501 7.388299109537321 4.660532682264452
6.759365394377312 5.909117814793371 4.578469486095184
7.039615789544486
6.881335686587291 9.346628594034131 2.037494699825919
5.387161727709744 9.809252143661475 0.6443116877320054
3.8099385935855787 1.4624634735840547 2.2261262812735403
8.640953927283476
8.030687805988649 1.3101530758773738 6.537603953909686
8.756722645749747 7.1935033427169435 0.32470723666721435
2.154486583724633 0.28872014330045603 5.18056085172047
6.3486018358062015
6.445939731669293 6.086653363984011 7.294737847474684
5.095187533243866 1.0826885619505777 6.541293828892767
3.882114450219867 3.4861975360212174 0.8664492264089274
9.398486497724862
9.040351281149663 6.855114886105796 5.081307490579606
0.5540420019722836 3.3793160375554434 5.391464766754691
5.904697212403541 8.689921761043868 3.8335307835548074
3.687416211518615
4.480096122244516 8.379712225526365 4.121715414037029
4.79691582533896 2.8804985454522303 2.121450851999811
6.606906846671787 6.828895395457584 9.381186873505902
1.6484566578039261
0.03133752419174684 6 5.7699265754711835 5.212272795214838
2.823885691265734 7.8834534651056245 2.968027396901196
3.2759289565853678 6.442316865291673 9.000167462558423
9.508147811384537
3.1901944927751735 9.341750113856659 5.415633586838695
2.5711922273789387 2.460678005259128 6.371975235449608
2.438537812383376 0.7922669884106792 0.5834488118039205
3.970348171122493

4. Input Size= 1000000



0.2261403562707387 1.8332989625414586 3.8188590061971883
8.86862232408929 7.484332541749809 2.245451439018098
8.082753138810098 6.633151342058072 6.826392036876242
3.7165133001214965
1.4720812372728875 1.3122020447868166 7.649703864140792
3.3844855314543842 7.713783424086809 7.55818580467411
5.630729546656465 1.0920677791686495 9.865565373372823
3.913242106779885
9.815686409393328 7.919896185256289 8.268243706145235
7.689258892162122 4.7512893351573195 6.48996151892908
6.032207106686562 8.25885187043156 7.821877729602508
6.284144300028768
9.807316932596654 0.7180597730770466 7.203389533955056
2.857564901994658 0.23597050617346738 5.1725208979880355
6.401281108383362 3.273713897248286 3.4519336571491577
3.0024201204593637
7.208640700164398 1.348095802516971 2.262944748659599
0.37750620657764133 4.1023733125787 5.35937880590153
3.242981435193241 1.0219965798024588 9.589485888916442
7.7718169591862285
3.8604706276806966 4.770202018095839 1.822134034994044
1.2099698956688731 1.981615302139711 5.807219481305685
2.9192355705452124 7.4365144626880095 8.613401135777554

4.655057833708217
8.864099954821846 6.406206645788121 3.5887343193949195
0.9430725251844574 9.591655792917326 0.6038212389056608
3.0465588140752944 9.792609058828598 0.9355057836235792
5.8211538947547625
7.022684292728108 3.249261649550684 8.474177414323671
3.287948701330974 5.242451706161311 9.612176693078498
6.997574552979601 9.643866590418762 9.294252616491356
7.738339821291661
8.308783540768808 0.6233862071200014 9.763309790529494
0.44080649506699765 4.383018990134772 1.5227870364185891
7.618013634813377 0.9533479711174697 2.5246418754899382
2.528134843017922
0.16650678334354296 9.100002422863676 4.0948195905109985
9.805453952739677 3.2390905477334697 9.286534268350833
8.626541424221145 3.889034207014314 2.902426876107025
0.7218228973585428

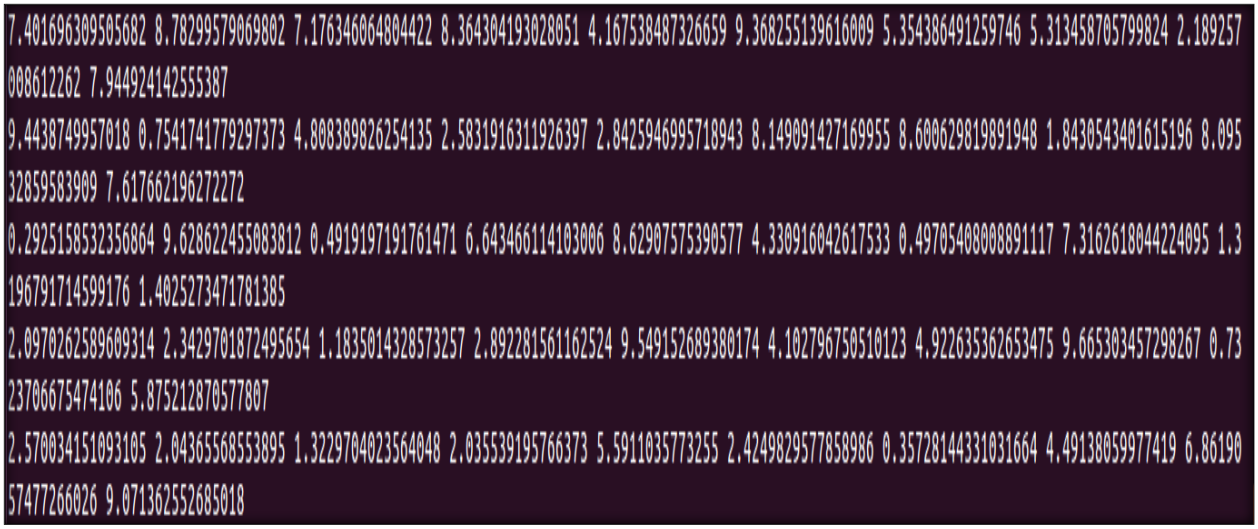
2. ***VARYING THE VECTOR SIZE OR DIMENSION***
dimension/vector size=2
datapoints=10,000

OUTPUT:
8.392628304575984 9.471987035173404
3.1224825151944033 6.375810627068318
2.4895374825214214 9.280271498366771
1.69386427200945 9.901851428995727
5.816333942252282 6.326596059490721
3.0684838481646217 5.054957750535985
5.300759279064402 0.05626807739536388
0.10043666904612003 6.760730164464502
7.324669358508197 9.09265121200024
1.4727927792438955 6.565293806149462

3. VARYING THE NUMBER OF CENTROIDSCENTROIDS

NUMBER OF CENTROIDS =5

INPUT_SIZE=100



OUTPUT:

7.401696309505682 8.78299579069802 7.176346064804422 8.364304193028051
4.167538487326659 9.368255139616009 5.354386491259746
5.313458705799824 2.189257008612262 7.944924142555387
9.4438749957018 0.7541741779297373 4.808389826254135
2.5831916311926397 2.8425946995718943 8.149091427169955
8.600629819891948 1.8430543401615196 8.09532859583909
7.617662196272272
0.2925158532356864 9.628622455083812 0.4919197191761471
6.643466114103006 8.62907575390577 4.330916042617533
0.49705408008891117 7.3162618044224095 1.3196791714599176
1.4025273471781385
2.0970262589609314 2.3429701872495654 1.1835014328573257
2.892281561162524 9.549152689380174 4.102796750510123
4.922635362653475 9.665303457298267 0.7323706675474106
5.875212870577807
2.570034151093105 2.04365568553895 1.3229704023564048
2.035539195766373 5.5911035773255 2.4249829577858986
0.35728144331031664 4.49138059977419 6.8619057477266026
9.071362552685018

REFERENCES:

[1] https://dsc-spidal.github.io/harp/docs/examples/kmeans/