实验4　K-Means聚类算法

> 小组成员：张涵之@191220154 林芳麒@191220057
>
> 完成进度：基础+选做 100%
>
> 日期：2022.5.30

一、实验内容

任务 实现`K - Means`聚类算法

执行命令：`hadoop jar KMeans.jar /data/2022s/kmeans/dataset.data /data/2022s/kmeans/initial_centers /user/2021sg07/KMeansOutput`

自定义聚类中心`(p,1)`类`VectorBean`

```
public class VectorBean implements Writable
{
    double[] vector = new double[15];//向量各分量
    int count = 0;//数据点个数n

    .....
}
```

（一）Map 和 Reduce 的设计思路

Map:

输出`Key`类型为`IntWritable`， `Value`类型为`VectorBean`(聚类中心向量`(p,1)`)

1.在`map`类的初始化方法`setup`中读取全局的聚类中心信息

2.对`map`方法收到的每一个数据点p，计算p与所有聚类中心间的距离，并选择一个距离最小的中心作为p所属的聚类，输出`<ClusterID, (p,1)>`键值对。

Combine:

对每个`Map`节点上即将传递到`Reduce`节点的每一个`<ClusterID, (p,1)>`键值对，用`Combine`进行数据优化，合并相同`Cluster ID`下的所有数据点，并求这些点的均值`pm`以及数据点个数`n`

Reducer:

输入`Key`类型为`IntWritable`(`ClusterID`)，输入`Value`类型为`VectorBean`（以`ClusterID`为分组的的`VectorBean`集合）。

输出`Key`类型为`Text`,`Value`类型为`NullWritable`。

经过`Map`和`Combine`后从Map节点输出的所有`ClusterID`相同的中间结果`<Cluster ID, [(pm1, n1), (pm2, n2) ...]>`，计算新的均值pm，输出`<ClusterID,pm>`

所有输出的`<ClusterID, (pm,n)>`形成新的聚类中心，供下一次迭代计算。

当连续两次的**MapReduce**结果相同，停止迭代，根据需要的精度输出结果。

（二）MapReduce 中 Map 和 Reduce 伪代码

```
/*
Mapper.java
*/
class Mapper{
  void setup(…)
  {
    //读出全局的聚类中心数据 Centers
  }

  void map(key, p) // p为一个数据点
  {
    minDis = Double.MAX VALUE;
    index = -1;
    for i=0 to Centers.length
    {
      dis= ComputeDist(p, Centers[i]);
      if dis < minDis
      {
        minDis = dis;
        index = i;
      }
    }
    emit(Centers[i].ClusterID, (p,1));
  }
}
```

```
/*
Combiner.java
*/
class Combiner {
  reduce(ClusterID, [(p1,1), (p2,1), …])
  {
```

```
    pm = 0.0;
    n = 数据点列表[(p1,1), (p2,1), …]中数据点的总个数;
    for i=0 to n
      pm += p[i];
    pm = pm / n; // 求得这些数据点的平均值
    emit(ClusterID, (pm, n));
  }
}
```

```
/*
Reduce.java
*/
class Reducer {
  reduce(ClusterID, value = [(pm1,n1),(pm2,n2) …])
  {
    pm = 0.0;  n=0;
    k = 数据点列表中数据项的总个数;
    for i=0 to k
    {
      pm += pm[i]*n[i];
      n+= n[i];
    }
    pm = pm / n; // 求得所有属于ClusterID的数据点的均值
    emit(ClusterID, (pm,n)); // 输出新的聚类中心的数据值
  }
}
```

（三）输出结果部分截图

HDFS 上的路径:`/user/2021sg07/KMeansOutput`

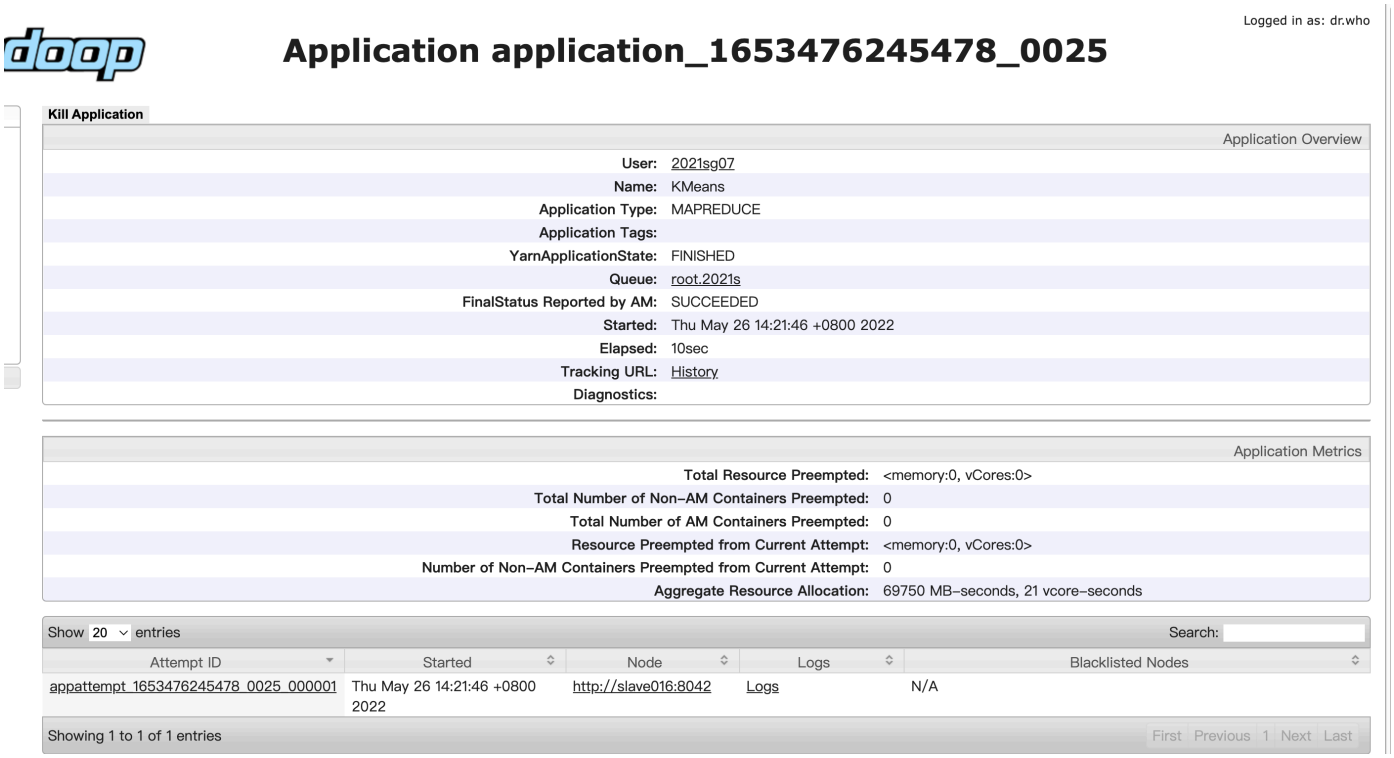
File – /user/2021sg07/KMeansOutput/part...

Page 1 of 1

```
0	4.490,4.501,4.508,4.485,4.501,4.501,4.489,4.493,4.501,4.512,4.492,4.509,4.495,4.492,4.509
1	14.506,14.492,14.504,14.499,14.497,14.501,14.513,14.506,14.509,14.512,14.485,14.520,14.504,14.515,14.502
2	24.507,24.506,24.494,24.498,24.495,24.491,24.502,24.504,24.512,24.499,24.497,24.510,24.512,24.497,24.496
3	34.511,34.495,34.516,34.508,34.514,34.492,34.507,34.508,34.500,34.518,34.496,34.502,34.505,34.511,34.482
4	44.497,44.508,44.498,44.501,44.496,44.503,44.513,44.498,44.504,44.507,44.502,44.502,44.493,44.511,44.500
5	54.501,54.501,54.496,54.504,54.502,54.505,54.499,54.517,54.485,54.510,54.492,54.506,54.488,54.507,54.505
6	64.506,64.516,64.507,64.511,64.493,64.508,64.497,64.493,64.495,64.499,64.510,64.515,64.505,64.490,64.501
7	74.506,74.515,74.509,74.512,74.505,74.499,74.504,74.518,74.515,74.508,74.498,74.512,74.497,74.503,74.492
8	84.500,84.498,84.509,84.479,84.496,84.488,84.496,84.495,84.498,84.497,84.493,84.508,84.494,84.483,84.508
9	94.498,94.498,94.497,94.490,94.512,94.491,94.493,94.495,94.495,94.500,94.505,94.497,94.499,94.501,94.497
10
104.489,104.498,104.481,104.521,104.494,104.498,104.503,104.491,104.500,104.489,104.505,104.512,104.524,104.509,104
.503
11
114.507 114.505 114.501 114.477 114.501 114.509 114.493 114.502 114.516 114.503 114.503 114.499 114.502 114.497 114.503
```

Cancel Download

（四）WebUI 执行报告



（五）算法分析

局限性和不足：

1.对初始cluster centers的选取回影响到最终的聚类结果，因此只能得到局部最优解，不保证得到全局最优解。

2.相似度计算和比较时的计算量较大。

3.数据越大，计算复杂度越高。

优化： 将各个点到cluster center相似度的计算工作分摊到不同的机器上并行地计算，从而大幅提高计算速度。

附加 不同聚类划分

执行命令： `hadoop jar DivideCluster.jar /data/2022s/kmeans/dataset.data /user/2021sg07/KMeansOutput/part-m-00000 /user/2021sg07/ClusterOutput`

1.用基础任务的输出聚类中心作输入。

2.**自定义向量对象 `（index, p）` 类 `VectorBean`**

`index`表示该向量条目的ID

`p`为一个向量的数据，是一个`double`集合

```
public class VectorBean implements Writable
{
    double[] vector = new double[15];
    int count = 0;

    .....
}
```

3.MultipleOutputs 类可以将数据写到多个文件，这些文件的名称源于输出的键和值或者任意字符串。这允许每个 reducer（或者只有 map 作业的 mapper）创建多个文件。

（一）Map和Reduce设计思路

Map:

输出`Key`类型为`IntWritable`， `Value`类型为`VectorBean`(向量对象`(index, p)`)

1.在`Map`类的初始化方法`setup()`中读取全局的聚类中心信息

2.对`map()`方法收到的每一个数据点p，计算p与所有聚类中心间的距离，并选择一个距离最小的中心作为p所属的聚类，输出`<ClusterID, (index, p)>`键值对。

Reducer:

输入`Key`类型为`IntWritable`(`ClusterID`)，输入`Value`类型为`VectorBean` （以`ClusterID`为分组的的`VectorBean`集合）。

输出`Key`类型为`NullWritable`,`Value`类型为`VectorBean`。

1.在`Reduce`类的初始化方法`setup()`中新建`MultipleOutputs`对象

2.在`reduce`方法中，对从Map节点输出的所有`ClusterID`相同的结果`<Cluster ID, [(index1,p1) , (index2,p2), ...]>`，写入同一个文件。

3.在`cleanup`方法中，关闭当前输出结果文件。

（二）Map和Reduce的伪代码

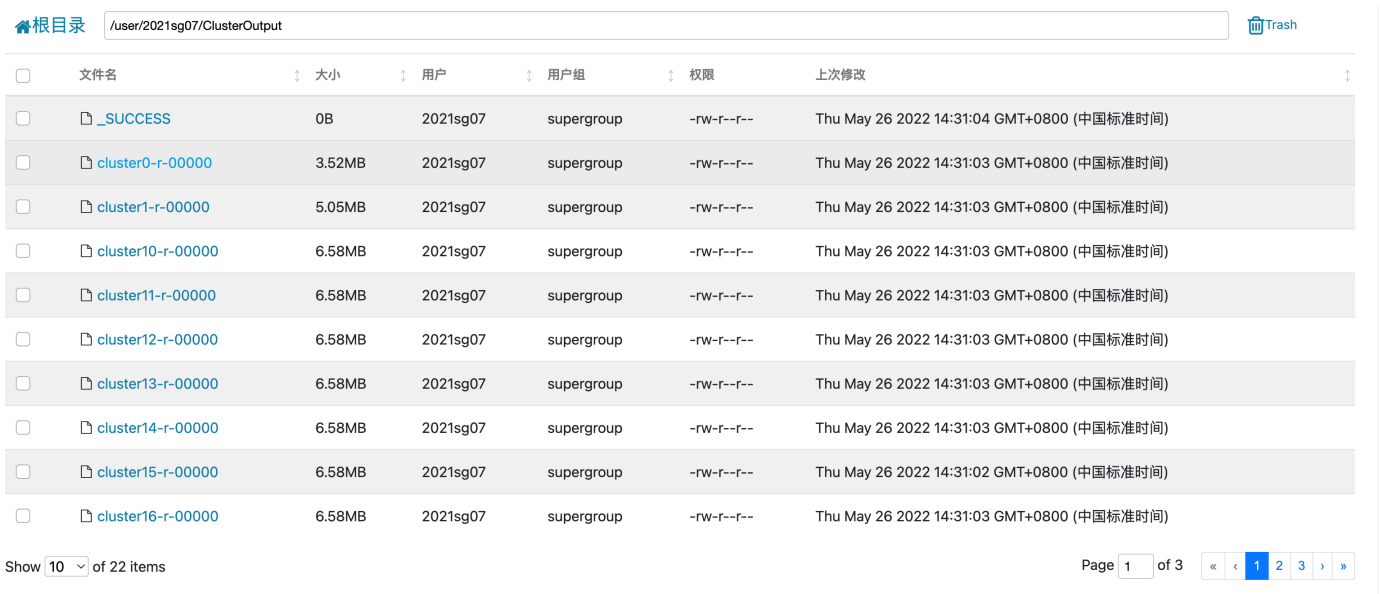
```
/*
DivideMapper.java
*/
class DivideMapper {
    void setup()  {
      //读取全局的聚类中心信息
        centers = DivideUtils.readCenters(centerPath);
    }

    void map(key,p) {// p为一个数据点
        minDis = Double.MAX VALUE;
        index = -1;
        for i=0 to Centers.length
        {
          dis= ComputeDist(p, Centers[i]);
          if dis < minDis
          {
            minDis = dis;
            index = i;
          }
        }
        IntWritable clusterID = new IntWritable(index);//选择一个距离最小的中心作为p所属的聚
类

        VectorBean p = new VectorBean();
        p.setIndex(selfIndex);
        p.setVector(vectorDbl);
        context.write(clusterID, p);//输出<聚类中心ID，向量对象>键值对
    }
}
```

```
/*
DivideReducer.java
*/
class DivideReducer{
    void setup() {
        mos = new MultipleOutputs(context);
    }

    void reduce(key,values) {}
    {
        String clusterID = "cluster" + key.toString();
        for (VectorBean vector : values) {
            mos.write(clusterID, NullWritable.get(), vector);
        }
```

```
    }

    void cleanup() {
        mos.close();
    }
}
```

（三）输出结果部分截图

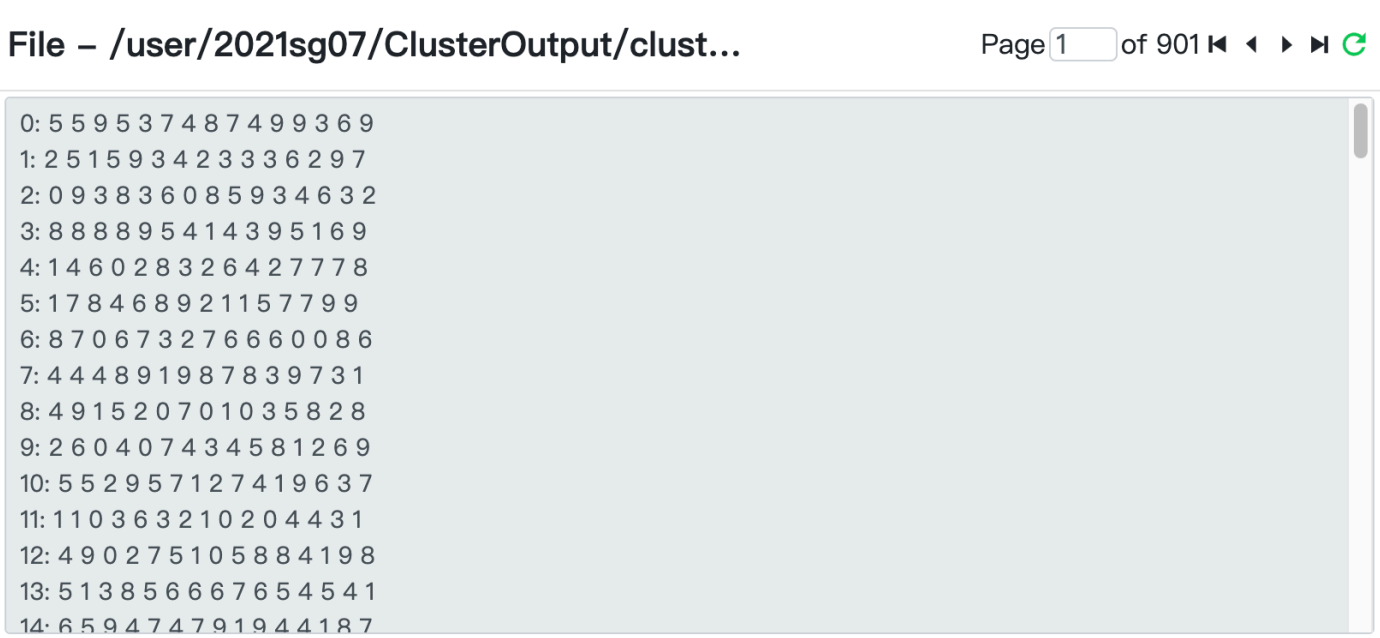
HDFS输出路径：`/user/2021sg07/ClusterOutput`



根目录 /user/2021sg07/ClusterOutput Trash

文件名	大小	用户	用户组	权限	上次修改
_SUCCESS	0B	2021sg07	supergroup	-rw-r--r--	Thu May 26 2022 14:31:04 GMT+0800 (中国标准时间)
cluster0-r-00000	3.52MB	2021sg07	supergroup	-rw-r--r--	Thu May 26 2022 14:31:03 GMT+0800 (中国标准时间)
cluster1-r-00000	5.05MB	2021sg07	supergroup	-rw-r--r--	Thu May 26 2022 14:31:03 GMT+0800 (中国标准时间)
cluster10-r-00000	6.58MB	2021sg07	supergroup	-rw-r--r--	Thu May 26 2022 14:31:03 GMT+0800 (中国标准时间)
cluster11-r-00000	6.58MB	2021sg07	supergroup	-rw-r--r--	Thu May 26 2022 14:31:03 GMT+0800 (中国标准时间)
cluster12-r-00000	6.58MB	2021sg07	supergroup	-rw-r--r--	Thu May 26 2022 14:31:03 GMT+0800 (中国标准时间)
cluster13-r-00000	6.58MB	2021sg07	supergroup	-rw-r--r--	Thu May 26 2022 14:31:03 GMT+0800 (中国标准时间)
cluster14-r-00000	6.58MB	2021sg07	supergroup	-rw-r--r--	Thu May 26 2022 14:31:03 GMT+0800 (中国标准时间)
cluster15-r-00000	6.58MB	2021sg07	supergroup	-rw-r--r--	Thu May 26 2022 14:31:02 GMT+0800 (中国标准时间)
cluster16-r-00000	6.58MB	2021sg07	supergroup	-rw-r--r--	Thu May 26 2022 14:31:03 GMT+0800 (中国标准时间)

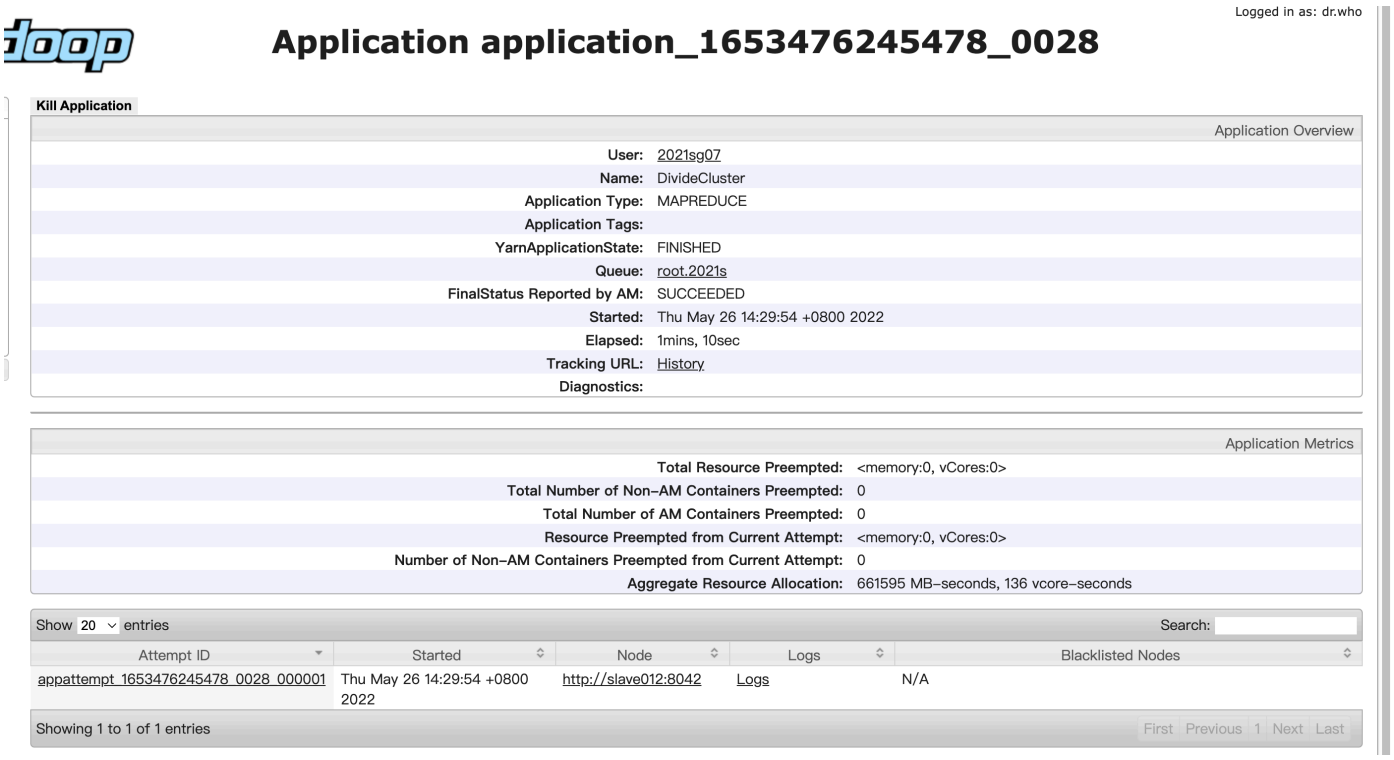
Show 10 of 22 items　　Page 1 of 3　« ‹ 1 2 3 › »



File – /user/2021sg07/ClusterOutput/clust...　　Page 1 of 901

```
0: 5 5 9 5 3 7 4 8 7 4 9 9 3 6 9
1: 2 5 1 5 9 3 4 2 3 3 3 6 2 9 7
2: 0 9 3 8 3 6 0 8 5 9 3 4 6 3 2
3: 8 8 8 8 9 5 4 1 4 3 9 5 1 6 9
4: 1 4 6 0 2 8 3 2 6 4 2 7 7 7 8
5: 1 7 8 4 6 8 9 2 1 1 5 7 7 9 9
6: 8 7 0 6 7 3 2 7 6 6 6 0 0 8 6
7: 4 4 4 8 9 1 9 8 7 8 3 9 7 3 1
8: 4 9 1 5 2 0 7 0 1 0 3 5 8 2 8
9: 2 6 0 4 0 7 4 3 4 5 8 1 2 6 9
10: 5 5 2 9 5 7 1 2 7 4 1 9 6 3 7
11: 1 1 0 3 6 3 2 1 0 2 0 4 4 3 1
12: 4 9 0 2 7 5 1 0 5 8 8 4 1 9 8
13: 5 1 3 8 5 6 6 6 7 6 5 4 5 4 1
14: 6 5 9 4 7 4 7 9 1 9 4 4 1 8 7
```

（四）WebUI 执行报告



Logged in as: dr.who

Application application_1653476245478_0028

Kill Application

Application Overview

User:	2021sg07
Name:	DivideCluster
Application Type:	MAPREDUCE
Application Tags:	
YarnApplicationState:	FINISHED
Queue:	root.2021s
FinalStatus Reported by AM:	SUCCEEDED
Started:	Thu May 26 14:29:54 +0800 2022
Elapsed:	1mins, 10sec
Tracking URL:	History
Diagnostics:	

Application Metrics

Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	661595 MB-seconds, 136 vcore-seconds

Show 20 entries Search:

Attempt ID	Started	Node	Logs	Blacklisted Nodes
appattempt_1653476245478_0028_000001	Thu May 26 14:29:54 +0800 2022	http://slave012:8042	Logs	N/A

Showing 1 to 1 of 1 entries First Previous 1 Next Last