Often running a HQL query you may notice that it progresses to 99% reduce stage quite fast and then stucks:

...

2014-10-07 08:46:01,149 Stage-8 map = 100%, reduce = 99%, Cumulative CPU 6905.85 sec

2014-10-07 08:47:01,361 Stage-8 map = 100%, reduce = 99%, Cumulative CPU 6999.45 sec

2014-10-07 08:48:01,441 Stage-8 map = 100%, reduce = 99%, Cumulative CPU 7065.59 sec

2014-10-07 08:49:01,670 Stage-8 map = 100%, reduce = 99%, Cumulative CPU 7125.26 sec

2014-10-07 08:50:01,808 Stage-8 map = 100%, reduce = 99%, Cumulative CPU 7188.12 sec

The problem is that Hive estimates the progress depending on the number of reducers completed, and this does not always relevant to the actual execution progress. It is possible that a query can reach 99% in 1 minute and then execute remaining 1% during 1 hour.

The most typical reason of this behavior is skewed data. For example, assume that you have a table that tracks all visits to the specific sites and SITE.COM has 100M rows while there are a dozen of other sites SITE.ORG, SITE.NET etc. that have just 10K visitors each.

Then when you join this table with another by site name, one reducer has to process 100M rows while other reducers process just 10K rows each.

So if you have 99 sites having 10K visitors, single site having 100M visitors and specify 100 reducers then 99% of reducers will finish their work very quickly and you have to wait for a long time when the last reducer terminates.

**Not only joins**

Data skew issue can arise not only in joins. For example, if you perform a GROUP BY SITE\_NAME in our example then a single reducer has to deal with 100M rows while others have to process much smaller number of rows.

How to avoid skew on reducer for "Group-By" on Hive

A "Group-By" query has heavy skew on one reducer.

For example, even if we set reducer number to 100 using below commands, one reducer takes hours to finish while other reducers only take seconds or minutes to finish.  
MRv1:

|  |  |
| --- | --- |
| 1 | set mapred.reduce.tasks=100; |

MRv2:

|  |  |
| --- | --- |
| 1 | set mapreduce.job.reduces=100; |

Note that on Hadoop 2 (YARN), the mapred.map.tasks and mapred.reduce.tasks are deprecated and are replaced by other variables:

mapred.map.tasks (MRv1) --> mapreduce.job.maps (MRv2)

mapred.reduce.tasks (MRv1) --> mapreduce.job.reduces (MRv2)

## Root Cause:

By default Hive puts the data with the same group-by keys to the same reducer.   
If the distinct value of the group-by columns has data skew, one reducer may get most of the shuffled data.  
So that reducer takes much longer time to finish than other reducers.

## Solution:

Set below configuration so that Hive will trigger an additional MapReduce job whose map output will randomly distribute to the reducer to avoid data skew.

|  |  |
| --- | --- |
| 1 | set hive.groupby.skewindata=true; |

After setting it, the reducers' statistics should show data is evenly distributed to each reducer.