1. Observation: In most time, only 50% of CPU is used. There are three peaks in CPU utilization, reaching around 70% to 90%.

Analyse: The potential activities in the peaks are collectAsMap and mapPartitions. There is still free CPU to be used. Therefore, I may increase ‘executor-cores’ and ‘spark.default.parallelism’ to increase the number of task threads and tasks. I will also try model parallelism on the cloud to check whether it helps.

1. Observation: The memory usage is stable at around 2300MB.

Analyse: Seems good.

1. Observation: The net utilisation is concentrated on several phases, but stable.

Analyse: Data locality might be taken into consideration. Currently there are ‘PROCESS\_LOCAL’ (best), ‘NODE\_LOCAL’ and ‘RACK\_LOCAL’ from the log file. The configuration of ‘spark.locality.wait’ is related with this.

1. Observation: The disk utilisation, I/O and block I/O process has five peaks.

Analyse: The peaks appear near the start of each circle (take -> count -> flatMap …). I will try Kryo serialisation on the cloud to check whether it helps.

Experiment Result: The following table shows the comparison between using Kyro serialisation and Java serialisation.



The rows with grey background are mean values of resource utilisation in three workers. It can be observed that

1. the amount of disk read and write is reduced with Kyro serialisation;
2. Kyro serialisation consumes more cpu;
3. Kyro serialisation reduces training time.

Note: The above observations are only based on single experiment, which means the outcome might be wrong. Therefore, more repetitive experiments are required.

1. Observation: The data per task is quite balanced, about 7000 to 9000 bytes per task. Therefore, there might not be the problem of data skew.
2. Probably useful configurations

* spark.shuffle.sort.bypassMergeThreshold
* spark.io.compression.codec (codec options)
* spark.rdd.compress

1. Burden brought by compression on CPU vs. Net burden

Analysis: To relieve the Map time, I will try to set some configurations related with compression to false. (e.g.: spark.shuffle.compress, which is about the map output files)

To relieve the burden on net, I will try to set some configurations related with compression to true. (e.g.: spark.rdd.compress)

spark.shuffle.spill.compress