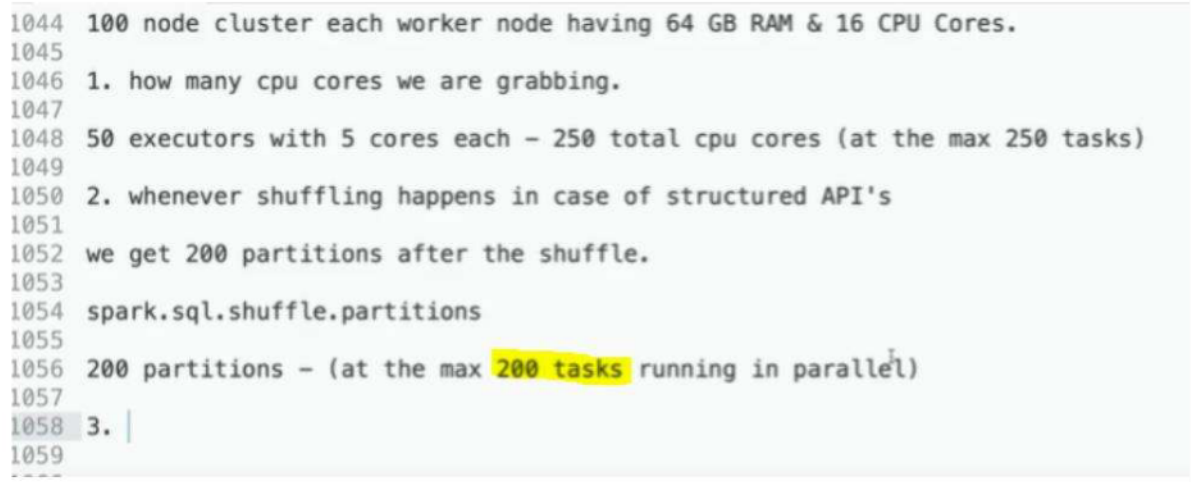
W14:1 if dynamic Allocation is set to True and also we are setting the properties - --driver-memory, --executor-memory, --executor-cores, --num-executors via spark-submit or via SparkConf, in this case how the configurations will work? What parameters are affected by the dynaMicAllocation property?

Ans: In case if we also provide --num-executors along with dynamicAllocation as true then the job will start with greater/max of (--num-executor, initial executors, min executors) number of executors

W14:2 Maeve has grabbed 50 executors with 5 cores each ..so 250 tasks can run max after doing shuffling in case of structured API'S we get 200 partitions 200 partitions means -(at max there will be 200 tasks running in parallel) //here why it can't be 250 tasks ,same 50 executors can work 5 tasks each on 200 partitions right? 50 at a time can someone clear this confusion



Ans: Those 50 executors can still execute 250 tasks(partitions) but the fact is, after shuffling by default you get 200 partitions and in-general 1 CPU core is allocated to one partition that's why it won't use those extra cores even if they are idle and can help speed up the processing

W14:3 Lily doesn't find the dataset customers.csv, orders.csv(2.6 GB) in week 14. These files from the previous week aren't that big. Can somebody please upload these files ?

Ans: Plz download from here:

https://drive.google.com/file/d/18gEpCPPCzebdn9JqwlDhFgFtfGZsewgg/view?usp=sharing
We can create the big file yourself as taught by Sumit and hence it was not uploaded initially I guess

W14:4 why the number of partitions are different when reading a file through the DataSet APIs vs RDD approach. The file has 2.34 GB data and if you read the file through RDD then 76 partitions are created (local machine block size is 32MB) however if the same file is read by DataFrame reader API it creates 19 partitions of 128 MB. Why is there a discrepancy between two approaches? Attached are the screenshot from both approaches

Ans: Olaa would recommend checking through HDFS or any distributed storage, because local systems we won't use in any case.
and this might be as part of internal optimizations I believe, as we majorly use distributed file system so what happens on local does not play much relevance

W14:5 What is a good use case of sending the output to Driver machine using collect() or take(). Adam means after we have increased our Driver memory and made sure that our sample output fits in Driver memory, what would be the use case to why we would want to store this output in the Driver's memory instead of a file system like HDFS?

Ans: Almost all the collect and take can be avoided in production. Normally its used by developers to check the quick output in development environment.

W14:6 How can repartitioning reduce Partition Skew? Why can't Coalesce reduce it?

Ans: When we do repartition the whole data comes to the driver i.e full shuffling happens and then the data gets partitioned. where as in coalesce the partition happens at executor level. The data from each executor gets equally distributed .

Repartition does not bring the data on the driver machine as it is a transformation not an action. Repartition does full shuffling that's why it's an expensive operation however it ensures data of almost the same size in all partitions.
Repartition can effectively reduce partition skew because it involves full shuffling whereas coalesce try to avoid or minimise shuffling whenever feasible so it will try to combine other partitions on the same data node so no shuffling is needed.
However if the other partitions on the same data node are not full you will sort of still have some skewness in your partitions.
But that doesn't completely mean that coalesce can't reduce partition skew, it depends on your data.
coalesce cannot increase the number of partitions unlike repartition...
even after increasing the partitions there is no Guarantee of proper partition... hence salting works better it's almost similar to smb

W14:7 Should Aimee enrol for itversity labs? We have to practice on that platform only? Can you please suggest?

Ans: Yes for optimization programs you need a cluster.

W14:8 the datasets used for the demo aren't present for this week. Could you please upload them at the earliest? Orders, customers, students

Ans:
https://drive.google.com/file/d/18gEpCPPCzebdn9JqwlDhFgFtfGZsewgg/view?usp=sharing

W14:9 Jackson was researching on the topic, how to join 2 rdds or data frames without involving shuffling?

Ans: Found that we can implement it if both the rdds can be co located and co-partitioned. We have to use the same partitioners(hash partitioner)on both the rdds to co-locate them, shuffle would happen one time when the partitioner is applied and then we apply join. Not sure if this a really effective solution that we can apply in production.

W14:10 List out the scenarios where you can use off-heap ?

Ans: scenarios for off-heap

1.Cache a lot of data without GC pausing involved,
2.Share cached data between JVMs
3. to overcome cache data in case of JVM crashes.