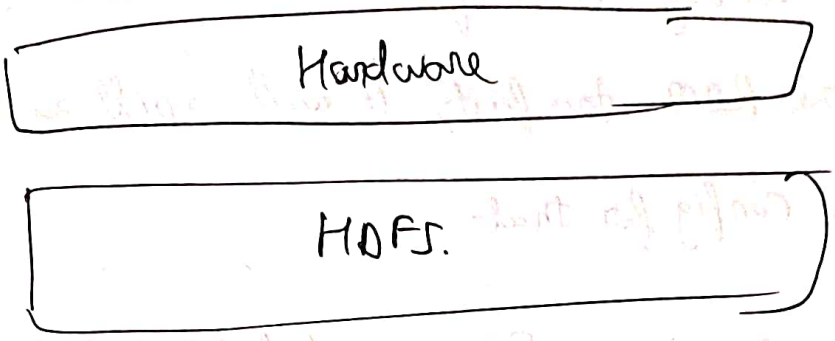
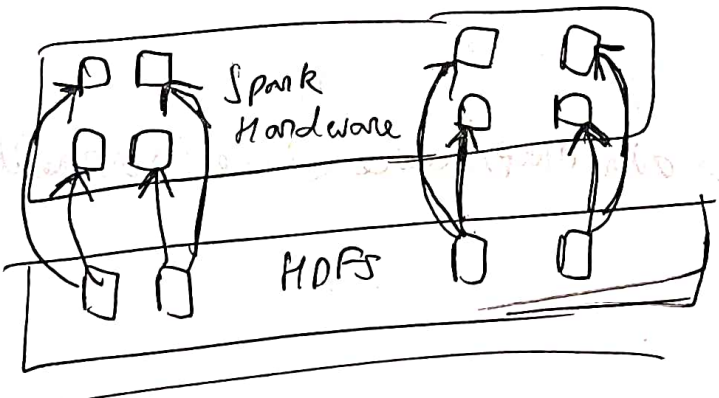
Concept clearing abt Spark



In Map/Reduce functionality the Mapper and the Reducer happens in the HDFS machines itself. But Spark separates this. It has its own Spark Executors which are responsible for doing this computation.



So I had a doubt like the executor machines of Spark are not that big that it can fit the entire data into Spark's executors.

→ To which Chatgpt said, that it was absolutely correct Spark does not load entire data to get into machine. Like it partitions the data and loads into machine. And it loads that much amount, that is feasible to fit

into memory:

Spark does not ~~write~~ write the intermediate state into HDFS again.

↳ It stores it in-memory, and if the intermediate state is filling up the RAM too fast, it will spill ~~to~~ to local disk. There is a config for that

① spark.memory.fraction → {How much of RAM if it gets filled then should it be flushed to disk}.

[usually in /tmp]

→ directory

② (spark.sql.files.maxPartitionBytes = 128MB)

↓ Size of partition to fetch from HDFS.

Now my other doubt was

Q) How is Spark winning over Map/Reduce since it's reading so many ~~Part~~ partitions.

A) Read from local disk is much faster than writing. Since, it not writing intermediate state to disk, Hence, its ~~repeated~~ winning over Map/Reduce.