**Starting point**

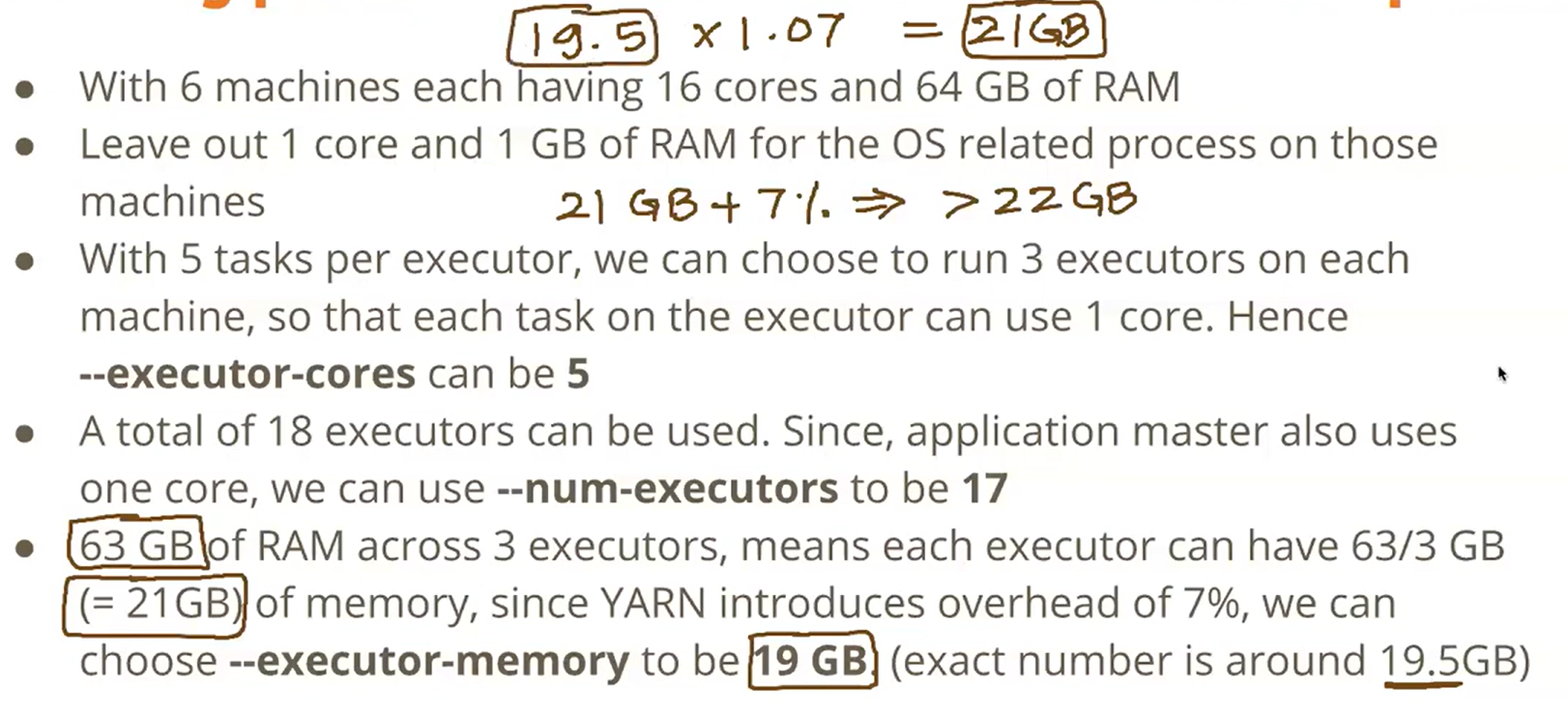
--> num-executors

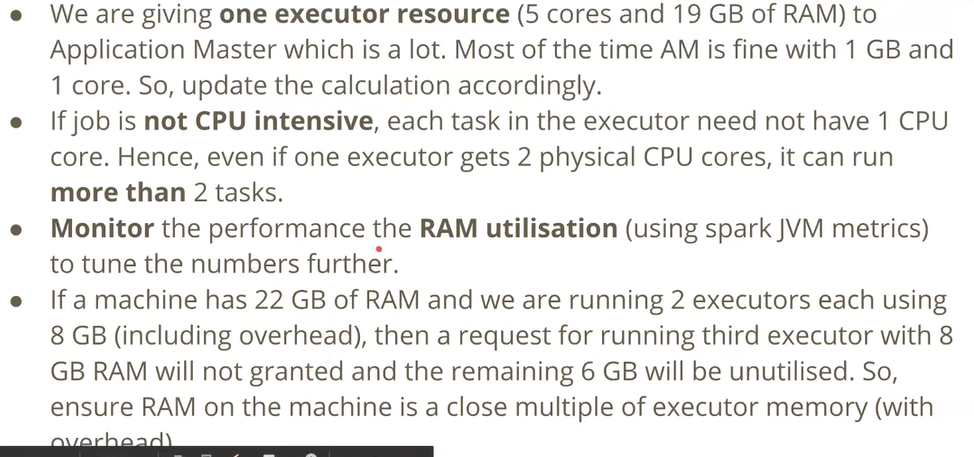
--> executor-cores

--> executor-memory

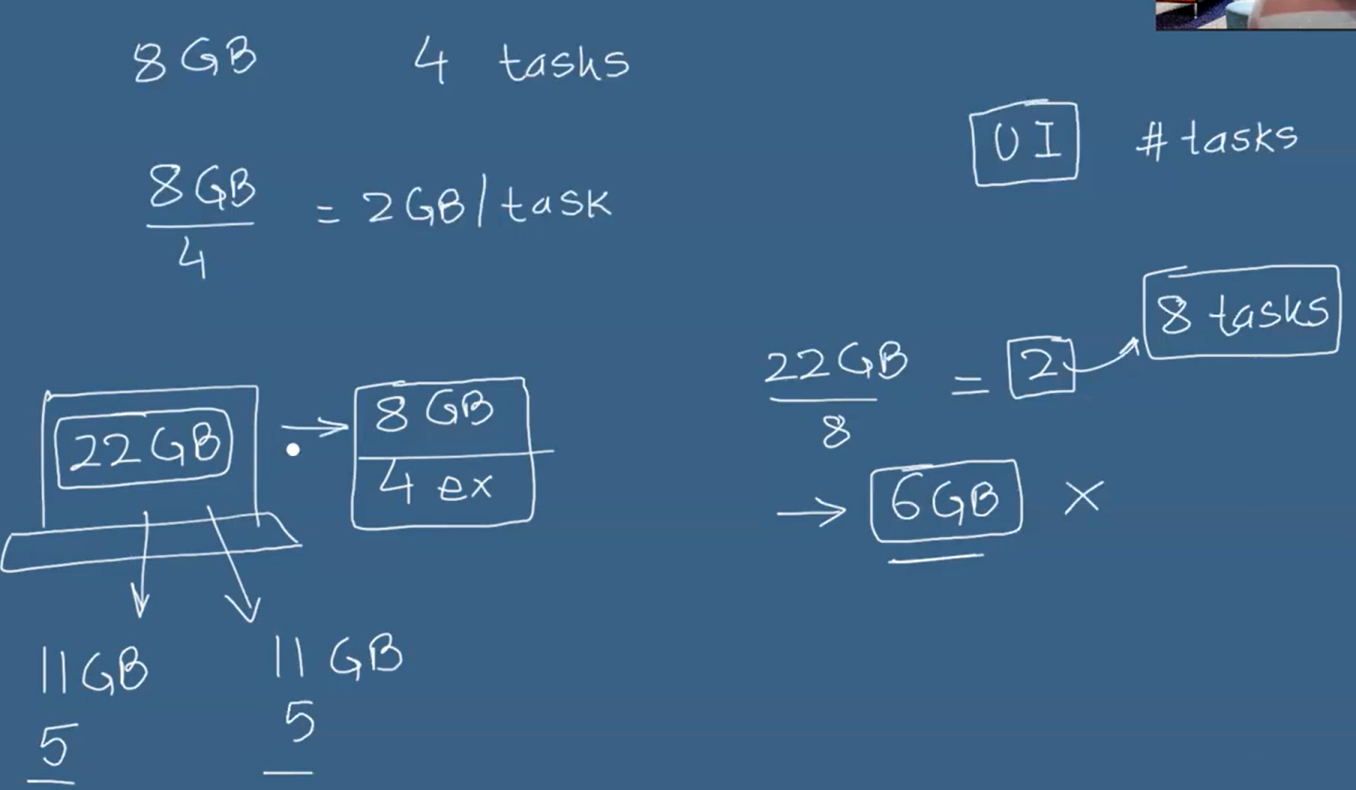
Take care of below points:

----> executor-cores decides number of tasks running in that executor in parallel





Leave out 1 core, 1 GB RAM for Hadoop/Yarn/OS daemons (per node)



==========================================================================

Let’s say you have a 20-node spark cluster Each node is of size - 16 cpu cores / 64 gb RAM

Let's say each node has 3 executors, with each executor of size - 5 cpu cores / 21 GB RAM

1. What's the total capacity of cluster?

We will have 20 \* 3 = 60 executors

Total CPU capacity: 60 \* 5 = 300 CPU Cores

Total Memory capacity: 60 \* 21 = 1260 GB RAM.

1. How many parallel tasks can run on this cluster?

We have 300 CPU cores; we can run 300 parallel tasks on this cluster.

1. Let's say you requested for 4 executors then how many parallel tasks can run on each node?

so, the capacity we got is 20 cpu cores

84 GB RAM

so, a total of 20 parallel tasks can run.

1. Let's say we read a csv file of 10.1 GB stored in datalake and have to do some filtering of data, how many tasks will run?

If we create a dataframe out of 10.1 GB file we will get 81 partitions in our dataframe.

so we have 81 partitions each of size 128 mb, the last partition will be a bit smaller.

so our job will have 81 total tasks. but we have 20 cpu cores

lets say each task takes around 10 second to process 128 mb data. so first 20 tasks run in parallel, once these 20 tasks are done the other 20 tasks are executed and so on...

so totally 5 cycles, if we think the most ideal scenario.

10 sec + 10 sec + 10 sec + 10 sec + 8 sec

first 4 cycle is to process 80 tasks all of 128 mb, last 8 sec is to process just one task of around 100 mb,

so it takes little lesser but 19 cpu cores were free during this time.

1. is there a possibility of, out of memory error in the above scenario?

Each executor has 5 cpu cores and 21 gb ram.

This 21 gb RAM is divided in various parts –

300 mb reserved memory,

40% user memory to store user defined variables/data. example hashmap

60% spark memory - this is divided 50:50 between storage memory and execution memory.

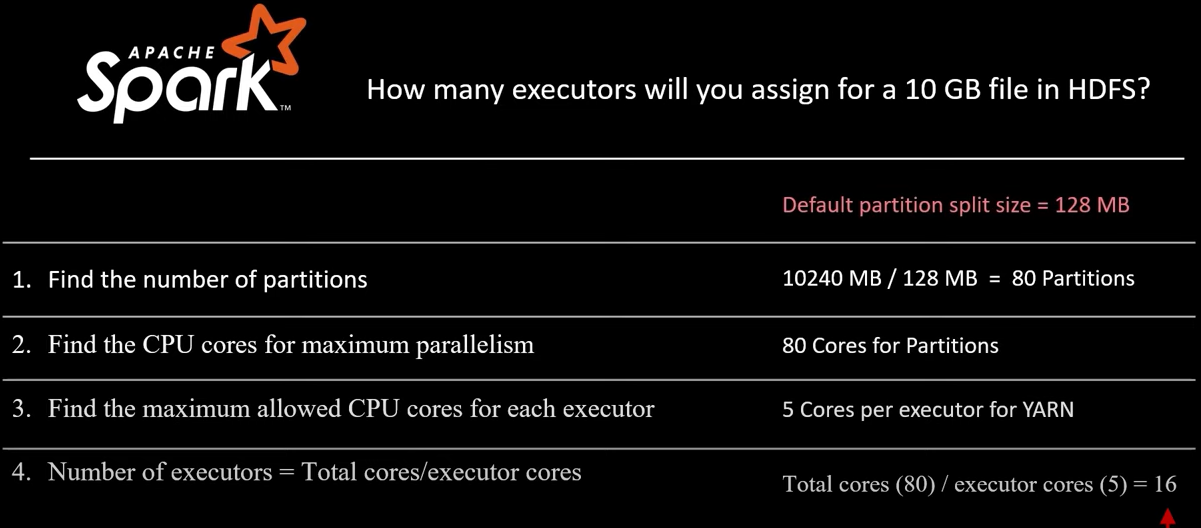
so basically, we are looking at execution memory and it will be around 28% roughly of the total memory allotted.

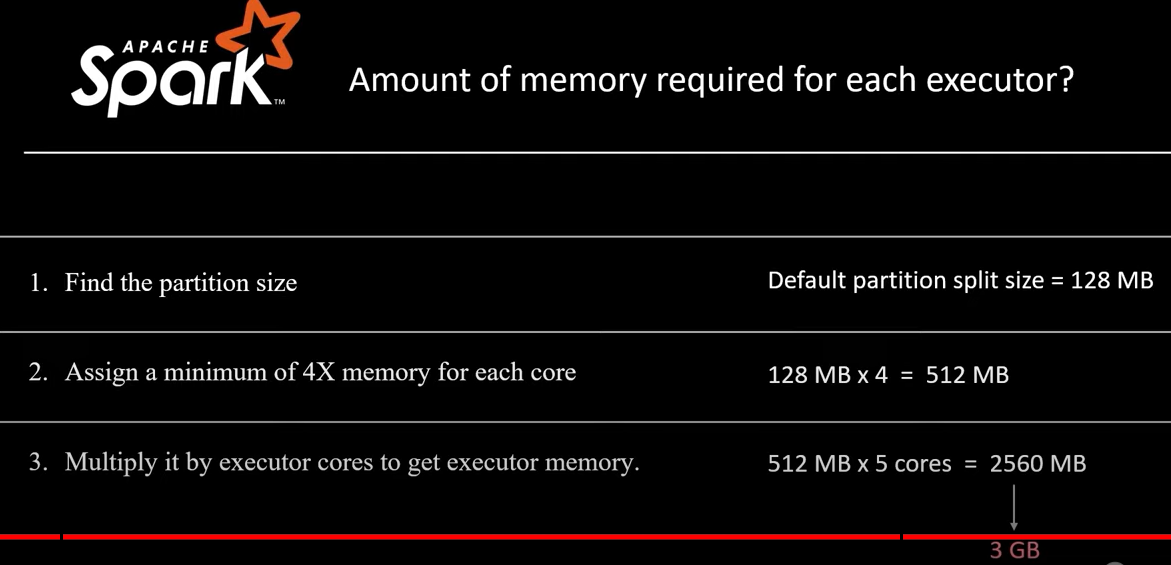
so, consider around 6 GB of 21 GB memory is meant for execution memory.

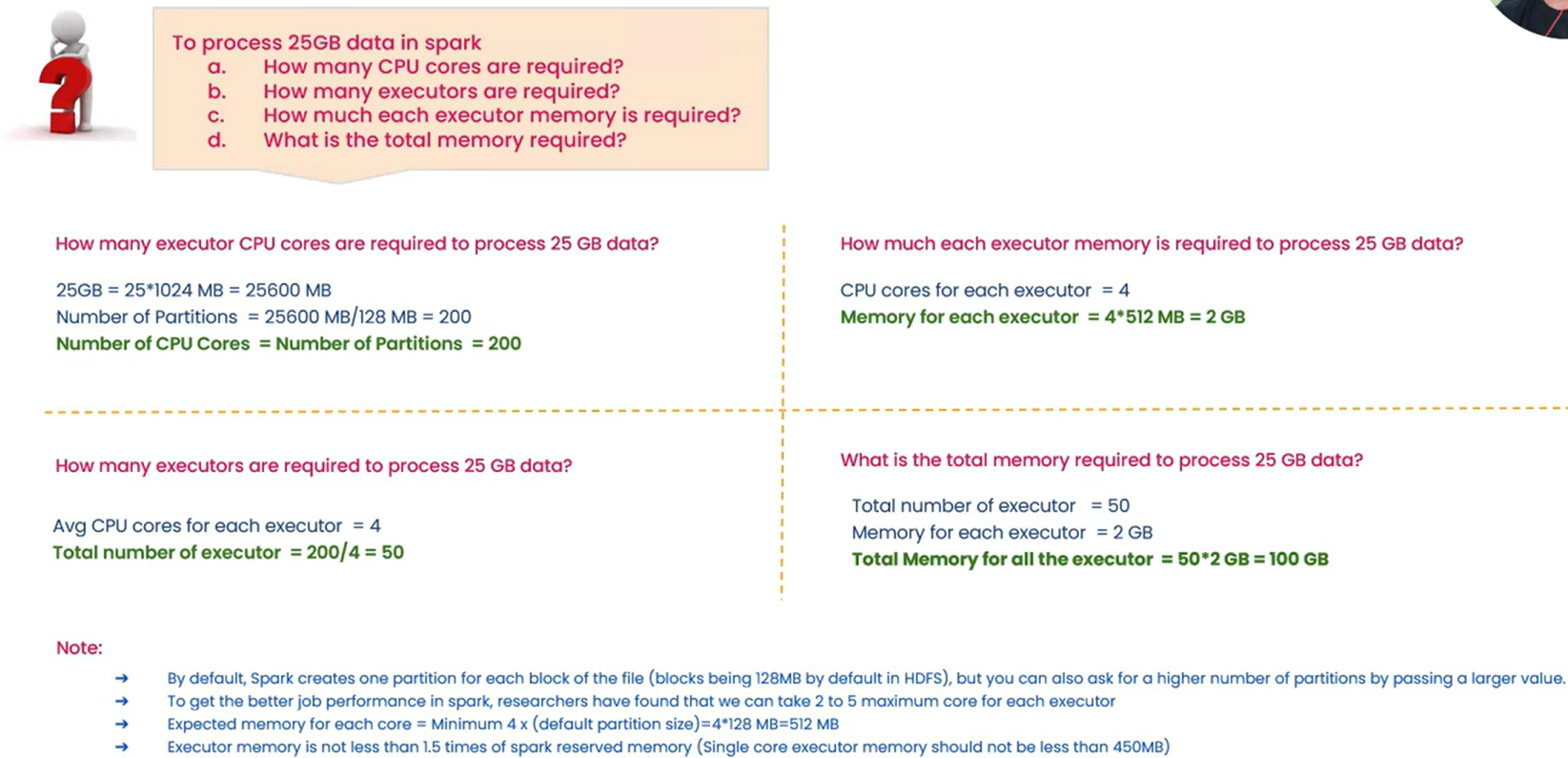
per cpu core we have (6 GB / 5 cores) = 1.2 GB execution memory.

That means our task can roughly handle around 1.2 GB of data.

however, we are handling 128 mb so we are well under this range.







If each(one) task is taking 5 mins to process data, how much time it requires to process 25 GB data.

* In above example one task is 128MB / 200 cores, if it is performing parallel it will take 5 min to process 25GB data.

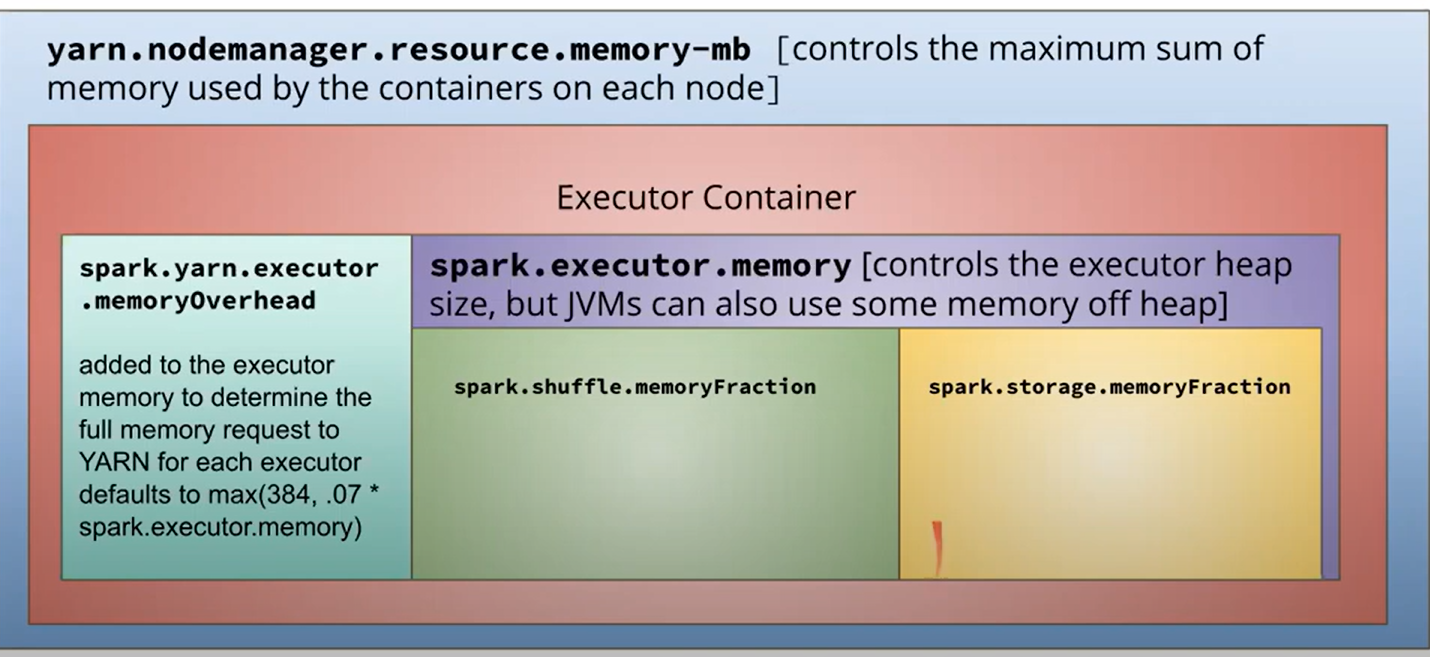
If you have to use only 25 executors for 25GB , how much time will it take?

* 25 \* 4 (cores) = 100 cores
* 5 mins for 200 tasks, it will take 10 for 100 tasks (will execute 100 core parallel in 5 min, next process will be in queue).
* Total memory required 50 GB.

if you are requesting memory--> you will get + 7% more ( min 384 is must)

10 GB --> 10 GB + 700 MB

4 GB --> 4 GB + 384



**In an application**

conf = SparkConf()

# task can fail 2 times

conf.set('spark.task.maxFailures', '2')

# duplicate copy of task run, whichever complete first will take that

conf.set('spark.speculation', 'true')

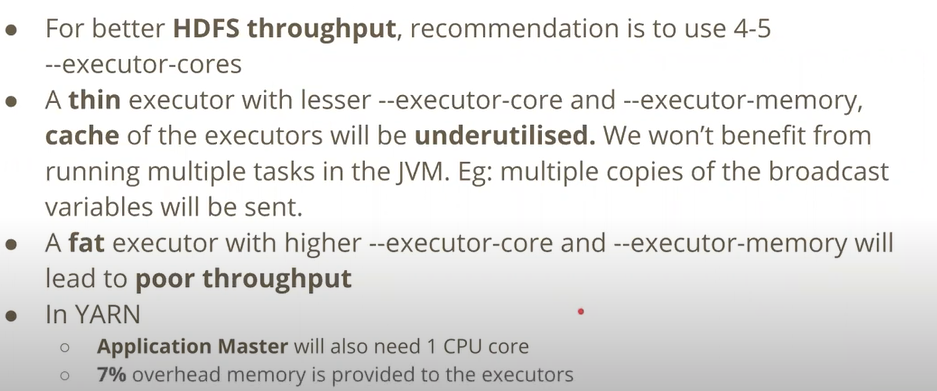
sc = SparkContext(conf)

**At runtime (spark-submit)**

spark-submit \

--conf spark.task.maxFailures=2 \

--conf spark.speculation=true ...



If your application is not CPU bound then, it will share CPU core on executor to run more task than one.

**Fat Executors:** It occupy large portion of the executors.