Thin Executer :- Intension is to create more Executors with each Executor holding minimum possible resource

Total of 16 Executors, with each Executor holding each Executor - 1 core, 4 gb ram

Draw Back :-

1. In this Scenario we will be losing the benefits of multithreading
2. A lot of copies of broadcast variable are required each Executer should receive it's own copy.

Fat Executor :- Intension is to give maximum resources each Executor

16 cores, 64 gb ram

you can create a Executer which can hold 16 CPU cores and 64 gb ram

Drawbacks :-

1. It is observed that if the Executer holds more tha 5 CPU cores then the hdfs throuput Subbers.
2. If the Executer holds very huge amount of memory then the garbage collection takes a lot of time.

garbage collection means removing unused objects from memory.

Here we got to know that Both of them had their own limitations and challenges.

what is the right approach → Should go with balanced approach

16 cores, 64 gb ram

1 core is given for other background activities
1 gb ram is given for operating system.

in each node we are now left with 15 cores, 63gb ram

⇒ we want multithreading within a Executor (> 1 CPU core per Executor)

⇒ we, want do not want our hdfs throughput to suffer (it suffers when we use more than 5 cores per Executor)

→ 5 is the right choice of number of CPU cores in each Executor.

→ Now we have 15 cores, 63 gb Ram → each machine

→ we can have 3 Executors running on each worker node

→ each Executor will contain 5 CPU cores and 21 gb ram

→ of this 21 GiB Ram some of it will go as part of overhead (off heap memory) ——— It takes
max (384 mB, 7% of Executer memory)

= 1.5 GiB (overhead / off heap memory) — this is not part of containers

21 − 1.5 = ~19 GiB

→ That mean in each Executer - 5 CPU cores, 19 GiB Ram.

→ we have a 10 node cluster / worker nodes

10 x 3 = 30 (Executors across the cluster)

→ 30 Executors with each Executor holding — 5 CPU cores
19 GiB RAM.

→ 1 Executor out of these 30 will be given for YARN application manager

30 − 1 = 29 Executers.