# spark-notes

View On GitHub

This project is maintained by spoddutur

# Distribution of Executors, Cores and Memory for a Spark Application running in Yarn:

```
spark-submit --class <CLASS_NAME> --num-executors ? --executor-cores ?
```

**Ever wondered how to configure `--num-executors`, `--executor-memory` and `--execuor-cores` spark config params for your cluster?**

# Let's find out how..

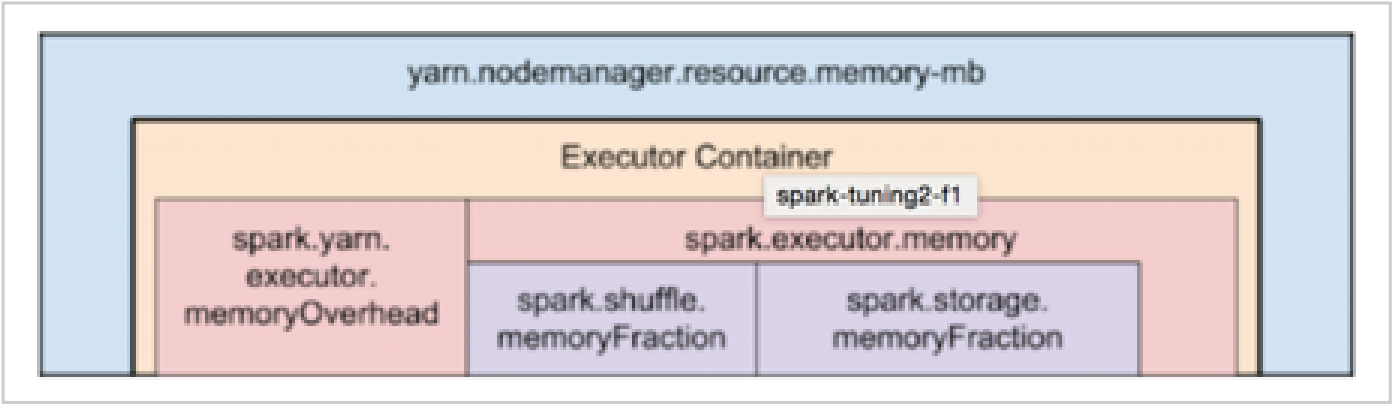
1. **Lil bit theory:** Let's see some key recommendations that will help understand it better
2. **Hands on:** Next, we'll take an example cluster and come up with recommended numbers to these spark params

# Lil bit theory:

**Following list captures some recommendations to keep in mind while configuring them:**

- **Hadoop/Yarn/OS Deamons:** When we run spark application using a cluster manager like Yarn, there'll be several daemons that'll run in the background like NameNode, Secondary NameNode, DataNode, JobTracker and TaskTracker. So, while specifying num-executors, we need to make sure that we leave aside enough cores (~1 core per node) for these daemons to run smoothly.
- **Yarn ApplicationMaster (AM):** ApplicationMaster is responsible for negotiating resources from the ResourceManager and working with the NodeManagers to execute and monitor the containers and their resource consumption. If we are running spark on yarn, then we need to budget in the resources that AM would need (~1024MB and 1 Executor).
- **HDFS Throughput:** HDFS client has trouble with tons of concurrent threads. It was observed that HDFS achieves full write throughput with ~5 tasks per executor . So it's good to keep the number of cores per executor below that number.

- **MemoryOverhead:** Following picture depicts spark-yarn-memory-usage.




Two things to make note of from this picture:

```
Full memory requested to yarn per executor =
        spark-executor-memory + spark.yarn.executor.memoryOverhead.
spark.yarn.executor.memoryOverhead =
                Max(384MB, 7% of spark.executor-memory)
```

So, if we request 20GB per executor, AM will actually get 20GB + memoryOverhead = 20 + 7% of 20GB = ~23GB memory for us.

- Running executors with too much memory often results in excessive garbage collection delays.
- Running tiny executors (with a single core and just enough memory needed to run a single task, for example) throws away the benefits that come from running multiple tasks in a single JVM.

## Enough theory.. Let's go hands-on..

Now, let's consider a 10 node cluster with following config and analyse different possibilities of executors-core-memory distribution:

```
**Cluster Config:**
10 Nodes
```

```
16 cores per Node

64GB RAM per Node
```

## First Approach: Tiny executors [One Executor per core]:

Tiny executors essentially means one executor per core. Following table depicts the values of our spar-config params with this approach:

```
- `--num-executors` = `In this approach, we'll assign one executor per
                     = `total-cores-in-cluster`
                    = `num-cores-per-node * total-nodes-in-cluster`
                    = 16 x 10 = 160
- `--executor-cores` = 1 (one executor per core)
- `--executor-memory` = `amount of memory per executor`
                      = `mem-per-node/num-executors-per-node`
                      = 64GB/16 = 4GB
```

**Analysis:** With only one executor per core, as we discussed above, we'll not be able to take advantage of running multiple tasks in the same JVM. Also, shared/cached variables like broadcast variables and accumulators will be replicated in each core of the nodes which is **16 times**. Also, we are not leaving enough memory overhead for Hadoop/Yarn daemon processes and we are not counting in ApplicationManager. **NOT GOOD!**

## Second Approach: Fat executors (One Executor per node):

Fat executors essentially means one executor per node. Following table depicts the values of our spark-config params with this approach:

```
- `--num-executors` = `In this approach, we'll assign one executor per
                    = `total-nodes-in-cluster`
                   = 10
- `--executor-cores` = `one executor per node means all the cores of t
                     = `total-cores-in-a-node`
```

```
                        = 16
- `--executor-memory` = `amount of memory per executor`
                        = `mem-per-node/num-executors-per-node`
                        = 64GB/1 = 64GB
```

**Analysis:** With all 16 cores per executor, apart from ApplicationManager and daemon processes are not counted for, HDFS throughput will hurt and it'll result in excessive garbage results. Also,**NOT GOOD!**

## Third Approach: Balance between Fat (vs) Tiny

**According to the recommendations which we discussed above:**

- Based on the recommendations mentioned above, Let's assign 5 core per executors => `--executor-cores = 5` (for good HDFS throughput)

- Leave 1 core per node for Hadoop/Yarn daemons => Num cores available per node = 16-1 = 15
- So, Total available of cores in cluster = 15 x 10 = 150
- `Number of available executors = (total cores/num-cores-per-executor) = 150/5 = 30`
- Leaving 1 executor for ApplicationManager => `--num-executors` = 29
- Number of executors per node = 30/10 = 3
- Memory per executor = 64GB/3 = 21GB
- Counting off heap overhead = 7% of 21GB = 3GB. So, actual `--executor-memory` = 21 - 3 = 18GB

**So, recommended config is: 29 executors, 18GB memory each and 5 cores each!!**

**Analysis:** It is obvious as to how this third approach has found right balance between Fat vs Tiny approaches. Needless to say, it achieved parallelism of a fat executor and best throughputs of a tiny executor!!

## Conclusion:

We've seen:

- Couple of recommendations to keep in mind which configuring these params for a spark-application like:
  - Budget in the resources that Yarn's Application Manager would need
  - How we should spare some cores for Hadoop/Yarn/OS deamon processes
  - Learnt about `spark-yarn-memory-usage`

- Also, checked out and analysed three different approaches to configure these params:
  1. Tiny Executors - One Executor per Core
  2. Fat Executors - One executor per Node
  3. Recommended approach - Right balance between Tiny *(Vs)* Fat **coupled** with the recommendations.

`--num-executors`, `--executor-cores` and `--executor-memory`.. these three params play a very important role in spark performance as they control the amount of CPU & memory your spark application gets. This makes it very crucial for users to understand the right way to configure them. Hope this blog helped you in getting that perspective…

My HomePage

---