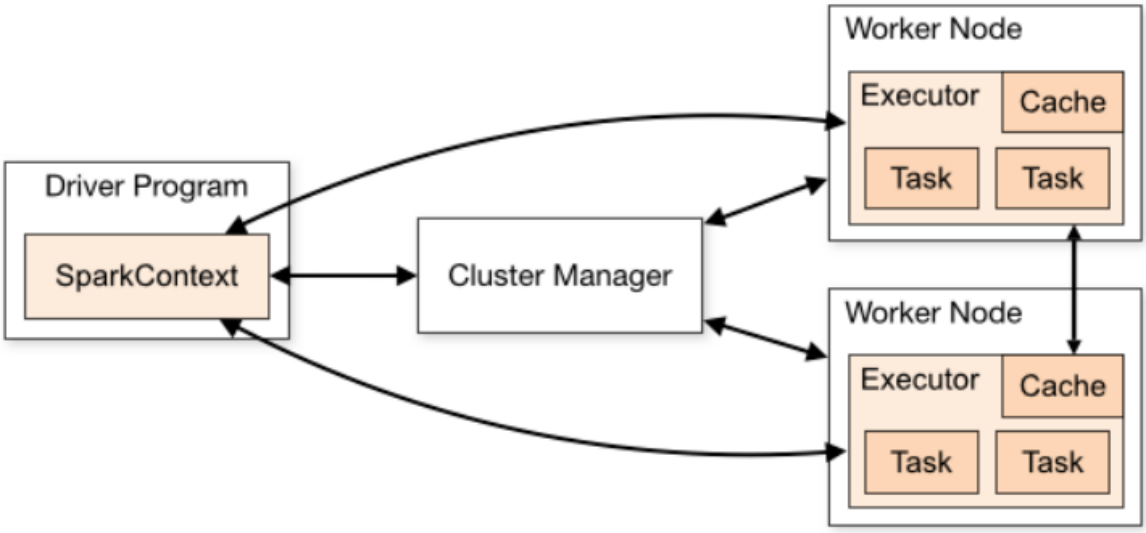
PySpark—The Cluster Configuration

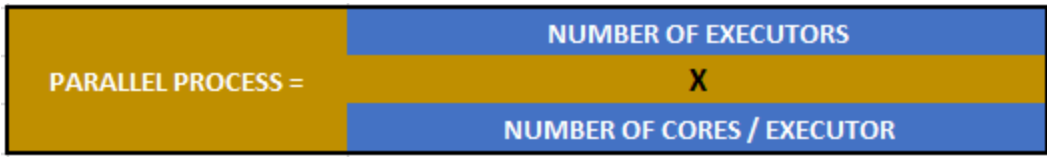
Spark Applications hugely relies on the cluster configuration used for execution. It is important to know the cluster sizing beforehand to increase the efficiency providing enough resource for processing.



Spark Cluster Representation (source: Spark documentation)

Spark achieves its power through the distributed parallel processing capability. For better results we always have to tune and estimate the correct configuration from all possibilities.

Spark determines the ***degree of parallelism = number of executors X number of cores per executor***



Degree of parallelism

Lets consider the following example: We have a cluster of ***10 nodes***, ***26 cores/node*** and ***256GB memory/node***.

Top Five

Following are the **top five articles** as per views. Don't forget check them out:

- PySpark - Count(1) vs Count(*) vs Count(col_name) - 1259 views
- PySpark—The Cluster Configuration - 1121 views
- PySpark - The Famous Salting Technique - 945 views
- PySpark - Merge Data Frames with different columns - 824 views
- PySpark - Create Spark Data Frame from API - 789 views

Buy me a Coffee

If you like my content and wish to buy me a COFFEE. Click the link below or Scan the QR.
Buy Subham a Coffee
*All Payments are secured through Stripe.

Scan the QR to Pay

TOTAL NODES	10
CORES/NODE	26
MEMORY/NODE	256GB

Available configuration

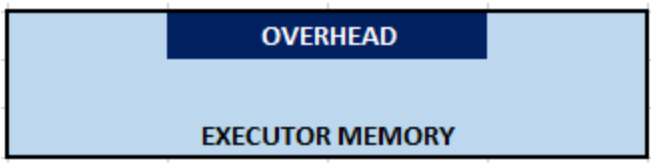
Fat Executors: In case we assign all cores to create a single executor per node i.e. 1 executor/node with 26 cores/node. Downside—It will create a lot of Garbage Collection (GC) issues leading to slow performance.

Tiny/Slim Executors: In case we assign 1 core/executor and create 26 executor/node from the above configuration. Downside—There will be too much of data movement between executors keeping the application busy throughout leading to performance degradation.

So, what can be a possible better configuration? Its seems from several tests that 4 to 5 cores/executors provide the optimum performance and avoids issue of Fat or Tiny Executor.

Lets consider the overheads as well before we run for the final configuration.

Overhead 1: Total Memory/Executor = Memory Overhead + Executor Memory



Total Memory of Executor

Memory Overhead for Yarn = Max (384MB or 7% of spark.executor-memory)

> *Parameter **spark.yarn.executor.memoryOverhead** can be checked for Yarn configurations.*

Overhead 2: 1 core and 1 GB RAM at least for Hadoop daemon and OS per node.

Overhead 3: 0.1% of efficiency loss

Overhead 4: 1 executor for Yarn Application Manager

Now lets finalize our cluster configuration:

1. ***Number of cores/executor*** = 5
2. ***Number of executor/node*** = (26—1) = 25/5 = 5 (*1 core for Overhead Hadoop and OS)
3. ***Memory/executor*** = (256–1)GB = (255/5)GB = 51GB – (7% of 51GB) = 46GB ~ 45GB (*excluding all overheads and efficiency loss)

4. ***Total Executors*** = 10 nodes x 5 executor/node = 50 = (50–1) = 49 executors
   (*exclude 1 executor for Application Manager)

So, we have to request ***49 executors*** with ***5 core/executor*** and ***45GB per executor memory*** from the Resource Manager for efficient cluster configuration.

NUMBER OF CORE/EXECUTOR		5 CORES
NUMBER OF EXECUTOR/NODE	(26 - 1) = 25/5 = 5	5 EXECUTORS/NODE
MEMORY/EXECUTOR	(256–1)GB = (255/5)GB = 51GB – (7% of 51GB) = 46GB ~ 45GB	45 GB/EXECUTOR
TOTAL EXECUTORS	10 X 5 = 50 = (50–1) = 49	49 EXECUTORS

The final configuration

Conclusion: There is a famous saying "*Same size doesn't fit all*". The same is for cluster configurations, sometimes same configuration cannot work for every workload. We might need to adjust according to the requirements. And, now we know how to do it.

Checkout my personal blog—https://urlit.me/blog/

Checkout the PySpark Medium Series—
https://subhamkharwal.medium.com/learnbigdata101-spark-series-940160ff4d30

Wish to Buy me a Coffee: Buy Subham a Coffee

About the Author

Subham is working as Senior Data Engineer at a Data Analytics and Artificial Intelligence multinational organization.
Checkout portfolio: Subham Khandelwal