Building a package and submit as an application

Right click on the project → Run As → Maven Clean

Right click on the project → Run As → Maven Install

Goto Linux terminal

cd ~

Submit Jar to different clusters:-

Local Mode

spark-submit --class org.inceptez.streaming.filestream --master local[2]
/home/hduser/workspacespark/spark/target/spark-0.0.1-SNAPSHOT-jar-with-dependencies.jar

Standalone Mode

start-master.sh
start-slaves.sh

Submit the jar in standalone mode

spark-submit --class org.inceptez.streaming.filestream --master spark://localhost:7077
/home/hduser/workspacespark/spark/target/spark-0.0.1-SNAPSHOT-jar-with-dependencies.jar

Yarn mode

spark-submit --class org.inceptez.streaming.kafkastream --master yarn
/home/hduser/workspacespark/spark/target/spark-0.0.1-SNAPSHOT-jar-with-dependencies.jar

spark-submit --class org.inceptez.streaming.filestream --master yarn /home/hduser/workspacespark/spark/target/spark-0.0.1-SNAPSHOT.jar --driver-memory 512m --num-executors 1 --executor-cores 2 --executor-memory 512m

spark-submit --class org.inceptez.stream.filestream --master yarn /home/hduser/install/Sparktwitterusecase/target/sparkIncetez-0.0.1-SNAPSHOT.jar --driver-memory 512m --num-executors 1 --executor-cores 1 --executor-memory 1 --spark-shuffle-compress true --spark-speculation true --spark-dynamicAllocation-enabled true --spark-dynamicAllocation-minExecutors 1 --spark-dynamicAllocation-maxExecutors 1 --spark-dynamicAllocation-initialExecutors 1

spark-submit --verbose \
--class org.inceptez.stream.filestream \
--master yarn /home/hduser/install/Sparktwitterusecase/target/sparkIncetez-0.0.1-SNAPSHOT.jar \
--deploy-mode client \
--queue default \
--driver-memory 512m \
--num-executors 1 \
--executor-cores 2 \
--executor-memory 512m \
--spark-shuffle-compress true \
--spark-speculation true \
--spark-dynamicAllocation-enabled true \
--spark-dynamicAllocation-initialExecutors 1 \
--spark-dynamicAllocation-minExecutors 1 \
--spark-dynamicAllocation-maxExecutors 8 \
--spark-dynamicAllocation-executorIdleTimeout 30s \
--conf spark.shuffle.service.enabled true \
--conf spark.sql.shuffle.partitions=4

https://spark.apache.org/docs/2.0.1/configuration.html

To Start spark shell with yarn-client

spark-shell --verbose --master yarn-client --driver-memory 512m --num-executors 1 --executorcores 2 --executor-memory 512m

or

spark-shell --master yarn

Spark Memory Allocation

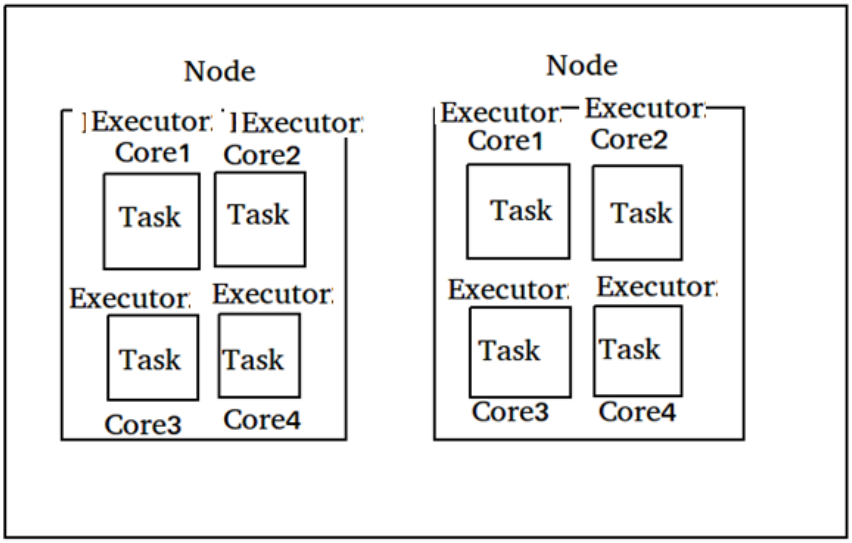
Let's consider a 10 node cluster with following config and analyse different possibilities of executors-core-memory distribution:

****Cluster Config:****

12 node (10 Worker/Data Nodes, 1 Master, 1 Client)
16 cores per Node
64GB RAM per Node

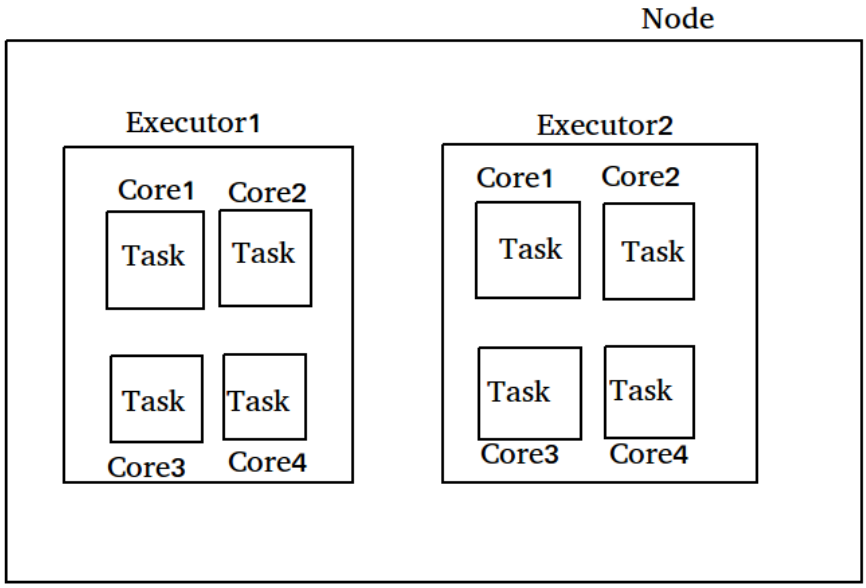
First Approach: Tiny executors [One Executor per core]:

1. **Multiple tasks in the same executor JVM can't be run, huge number of small JVMs start and kill overhead.**
2. **Shared/cached variables like broadcast variables and accumulators will be replicated in each executor of the nodes which is 16 times.**
3. **Not leaving enough memory overhead for Hadoop/Yarn daemon processes and not counting in ApplicationManager.**



Second Approach: Fat executors (One Executor per node)

1. With all 16 cores per executor,
2. ApplicationMaster and daemon processes are not counted
3. HDFS throughput will hurt and it'll result in excessive garbage results.



Third Approach: Balance between Fat (vs) Tiny

With the cluster of 10 Nodes , 16 cores per Node, 64GB RAM per Node

Assign 5 cores per executors => --executor-cores = 5 (for good HDFS throughput)

Leave 1 core per node for DN/Yarn daemons => Num cores available per node = 16-1 = 15

Total available of cores in cluster = 15 cores x 10 nodes = 150 cores

Number of available executors = (total cores/num-cores-per-executor) = 150 total cores/5 executor cores = 30 executors available

Leave 1 executor for ApplicationMaster => 30 – 1 = 29 total executors

Number of executors per node => --num-executors = 30 executors / 10 nodes= 3 executors per node

Memory per executor => 64GB pernode /3 executors = 21GB per executor

Memory per executor after Counting off heap overhead = off heap allocation of ~7% of 21GB = 3GB, hence --executor-memory = 21 - 3 = 18GB Memory per executor

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

Third approach has found right balance between Fat vs Tiny approaches.

Finally it achieved parallelism of a fat executor and best throughputs of a tiny executor!!

Leaving all the above calculations, iterate or benchmark for optimal/better performances considering ondemand/continuous other resources consumption in the cluster for other jobs at that time.