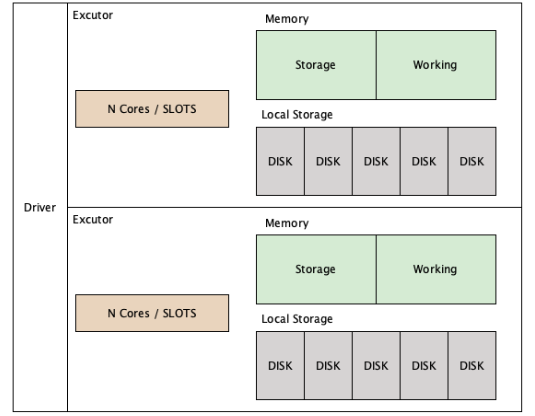
Given our cluster is with **746 cores**, **3 TB memory**, and **5 nodes**, and assuming you're processing **2 TB input data** with each executor having **16 GB memory**, here’s how to configure the spark-submit command and estimate the number of iterations.

Jobs —---------------- Stages—---------------------Tasks

1 **Task = 1 Core = 1 Partition**



### **Step 1: Cluster Configuration Analysis**

#### **1. Memory Per Node**

Total memory: 3 TB = 3072 GB.

Memory per node:

Memory per Node = 3072 GB / 5 Nodes = 614.4 GB/Node.

**2. Executor Memory**

* Each executor is assigned **16 GB memory**, plus overhead memory.
* Overhead memory is typically **10% of executor memory**:
* Overhead Memory=16 GB×0.1=1.6 GB.
* Total memory per executor: 16 GB+1.6 GB=17.6 GB.

**3. Executors Per Node**

Executors per Node = Memory per Node / Memory per Executor = 614.4 / 17.6≈34  Executors per Node

#### **4. Total Executors**

Total Executors = Executors per Node × Nodes = 34 × 5 = 170 Executors.

**5. Executor Cores**

* With **746 cores** across 170 executors:

Cores per Executor = Total Cores / Total Executors = 746/170 ≈ 4

### 

**Data**

### **Partitioning and Parallelism**

* **Partition Types**:
  + **Input Partitions**:
    - Controlled by spark.sql.files.maxPartitionBytes (default: 128 MB).
  + **Shuffle Partitions**:
    - Controlled by spark.sql.shuffle.partitions (default: 200).
    - Ideal partition size: **200 to 250 MB** for shuffle stages.
  + **Output Partitions**:
    - Adjust using coalesce() or repartition().
    - Balance write time vs. read performance.

To process **2 TB of input data**, we need sufficient parallelism:

* Recommended parallelism is **2 - 3 x total cores**:

Parallelism = 746 × 2 = 1492

Set spark.default.parallelism and spark.sql.shuffle.partitions to **1492**.

### **Step 3: Spark-Submit Command**

Here’s the spark-submit command for this configuration:

bash

Copy code

spark-submit \

--master yarn \

--deploy-mode cluster \

--name "Spark Job for 2TB Data" \

--num-executors 170 \ # Total executors (34 per node x 5 nodes)

--executor-cores 4 \ # Cores per executor

--executor-memory 16G \ # Memory per executor

--conf spark.executor.memoryOverhead=1.6G \ # Overhead memory

--conf spark.default.parallelism=1492 \ # Parallelism

--conf spark.sql.shuffle.partitions=1492 \ # Shuffle partitions

/path/to/your\_spark\_application.py

### 

### **Step 4: Estimating Iterations**

#### **1. Tasks per Stage**

With **2 TB data** and **1492 tasks (parallelism)**:

* Each task processes:

Data per Task = 2 TB / 1492 ≈ 1.34 GB/Task

**2. Iterations per Stage**

* **Total Executors**: 170 executors with 4 cores each can run:

Tasks Running Simultaneously = 170 × 4 = 680 Tasks

To complete 1492 tasks:

Iterations per Stage = TasksSimultaneous Tasks = 1492/680 ≈ 2.2  Iterations (round up to 3)

**3. Number of Stages**

The number of stages depends on your Spark job's operations:

* Simple ETL job (e.g., map, filter, reduceByKey): **3-4 stages**.
* Complex jobs (e.g., joins, aggregations): **5-8 stages**.

Let’s say we have 2000 cores available in our cluster, then let’s use 2000 cores.

### **Monitoring Execution**

1. **Spark UI**:
   * Use the **Stages tab** to view the number of stages and tasks executed.
2. **YARN UI**:
   * Monitor resource allocation and task distribution across the cluster.