Spark Memory Task:

When it comes to Spark executor memory allocation, there are primarily three common types: "Thin," "Balanced," and "Fat." Let me explain each of them theoretically for a cluster with 10 nodes, and I'll also mention a few other common memory allocation strategies.

1. Thin Allocation:
   * In a "Thin" allocation, a smaller portion of the total memory is allocated to the Spark executor heap. This leaves more memory available for the operating system and other processes on the node.
   * For example, if you have 10 nodes, and you allocate 75% of each node's memory to Spark, the remaining 25% can be used by the operating system and other non-Spark processes.
   * This type of allocation is suitable when you want to coexist with other memory-intensive applications on the same cluster nodes.
2. Balanced Allocation:
   * A "Balanced" allocation strikes a balance between Spark and non-Spark memory usage. Typically, you allocate a moderate portion of the memory to Spark, allowing it to perform efficiently without hogging all the resources.
   * For example, you allocate 50% of each node's memory to Spark, leaving the other 50% for the operating system and other processes.
   * This type of allocation is suitable for most use cases and ensures a reasonable balance between Spark performance and cluster stability.
3. Fat Allocation:
   * In a "Fat" allocation, a larger portion of the node's memory is dedicated to Spark, providing maximum resources for Spark to process data.
   * For instance, you allocate 90% of each node's memory to Spark, with only 10% left for the operating system and other processes.
   * This type of allocation is useful when you have a dedicated cluster for Spark and want to maximize its performance.

**Thin Memory Allocation:**

In the "thin" memory allocation approach, we allocate a smaller portion of memory to the heap, leaving more for off-heap storage and other non-heap memory areas.

For the 5 nodes with 64 GB memory:

Heap Memory Per Node: 20% of 64 GB = 12.8 GB

Off-Heap Memory Per Node: 80% of 64 GB = 51.2 GB

Number of Executors Per Node: 2 (as per your assumption)

For the 5 nodes with 32 GB memory:

Heap Memory Per Node: 20% of 32 GB = 6.4 GB

Off-Heap Memory Per Node: 80% of 32 GB = 25.6 GB

Number of Executors Per Node: 5 (as per your assumption)

**Balanced Memory Allocation:**

In the "balanced" memory allocation approach, we allocate roughly equal amounts of memory to both the heap and off-heap storage.

For the 5 nodes with 64 GB memory:

Heap Memory Per Node: 50% of 64 GB = 32 GB

Off-Heap Memory Per Node: 50% of 64 GB = 32 GB

Number of Executors Per Node: 2 (as per your assumption)

For the 5 nodes with 32 GB memory:

Heap Memory Per Node: 50% of 32 GB = 16 GB

Off-Heap Memory Per Node: 50% of 32 GB = 16 GB

Number of Executors Per Node: 5 (as per your assumption)

Aggressive Memory Allocation:

I**n the "fat" memory allocation approach, we allocate a larger portion of memory to the heap.**

For the 5 nodes with 64 GB memory:

Heap Memory Per Node: 70% of 64 GB = 44.8 GB

Off-Heap Memory Per Node: 30% of 64 GB = 19.2 GB

Number of Executors Per Node: 2 (as per your assumption)

For the 5 nodes with 32 GB memory:

Heap Memory Per Node: 70% of 32 GB = 22.4 GB

Off-Heap Memory Per Node: 30% of 32 GB = 9.6 GB

Number of Executors Per Node: 5 (as per your assumption)

Now, let's calculate the total number of executors for each strategy:

**Thin Memory Allocation:**

Total Executors on 64 GB nodes: 5 nodes \* 2 executors/node = 10 executors

Total Executors on 32 GB nodes: 5 nodes \* 5 executors/node = 25 executors

Total Executors for the Cluster: 10 (64 GB nodes) + 25 (32 GB nodes) = 35 executors

**Balanced Memory Allocation:**

Total Executors on 64 GB nodes: 5 nodes \* 2 executors/node = 10 executors

Total Executors on 32 GB nodes: 5 nodes \* 5 executors/node = 25 executors

Total Executors for the Cluster: 10 (64 GB nodes) + 25 (32 GB nodes) = 35 executors

**Aggressive Memory Allocation:**

Total Executors on 64 GB nodes: 5 nodes \* 2 executors/node = 10 executors

Total Executors on 32 GB nodes: 5 nodes \* 5 executors/node = 25 executors

Total Executors for the Cluster: 10 (64 GB nodes) + 25 (32 GB nodes) = 35 executors