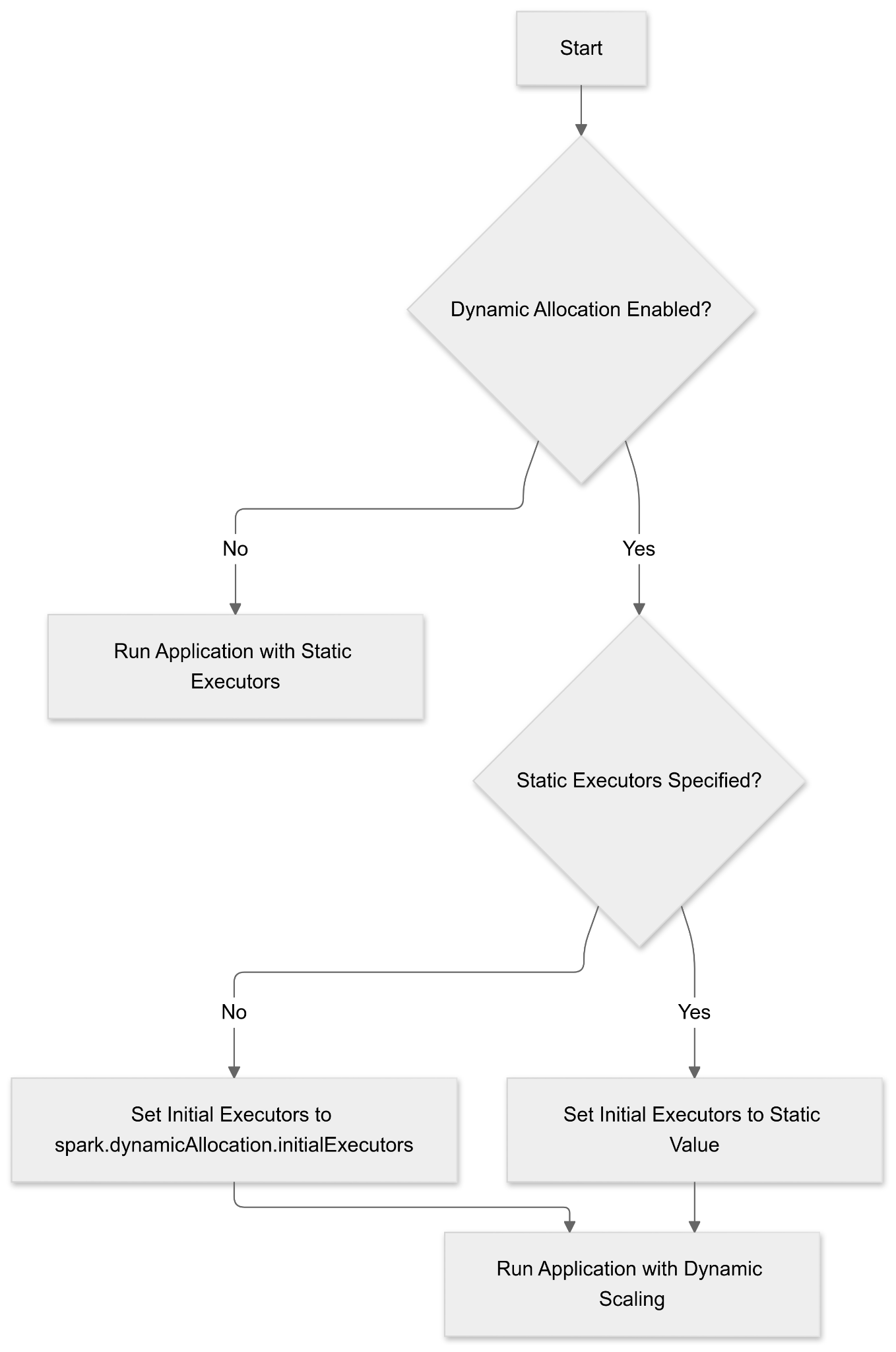
In Apache Spark, configuring executor allocation is crucial for optimizing application performance. When both dynamic allocation and static executor settings are specified, understanding their precedence is essential to ensure the desired behavior.



**Dynamic Allocation Overview**

Dynamic allocation allows Spark to adjust the number of executors during runtime based on the workload. This feature enables Spark to request additional executors when there are pending tasks and release idle ones, promoting efficient resource utilization. To enable dynamic allocation, set the following configurations:

spark.dynamicAllocation.enabled = true

spark.dynamicAllocation.minExecutors = [minimum number of executors]

spark.dynamicAllocation.maxExecutors = [maximum number of executors]

Additionally, ensure that the external shuffle service is enabled to allow executors to be removed without deleting shuffle files:

spark.shuffle.service.enabled = true

**Static Executor Configuration**

Static executor settings define a fixed number of executors for the entire duration of the Spark application. This is typically specified using the --num-executors parameter in the spark-submit command or by setting spark.executor.instances in the configuration:

spark-submit --num-executors [number of executors]

or

spark.executor.instances = [number of executors]

**Precedence and Interaction**

When both dynamic allocation and static executor settings are specified, their interaction is governed by specific precedence rules:

1. **Dynamic Allocation Enabled with spark.executor.instances Set**:
   * If spark.dynamicAllocation.enabled is set to true and spark.executor.instances is also specified, Spark will disable dynamic allocation. This is because specifying a fixed number of executors conflicts with the dynamic nature of executor allocation. A warning message will be logged indicating this configuration conflict.
2. **Dynamic Allocation Enabled without spark.executor.instances**:
   * When dynamic allocation is enabled and spark.executor.instances is not set (or set to 0), Spark manages the number of executors based on the workload, within the bounds defined by spark.dynamicAllocation.minExecutors and spark.dynamicAllocation.maxExecutors. In this scenario, dynamic allocation functions as intended, scaling resources up or down as needed.
3. **Specifying --num-executors with Dynamic Allocation**:
   * Using the --num-executors parameter in the spark-submit command sets the initial number of executors when dynamic allocation is enabled. However, it does not override the dynamic allocation settings. Spark will still adjust the number of executors based on the workload, but it will start with the number specified by --num-executors. It's important to note that if --num-executors is set higher than spark.dynamicAllocation.maxExecutors, the latter will take precedence, and Spark will not exceed the maximum specified by spark.dynamicAllocation.maxExecutors.

**Best Practices**

* **Avoid Conflicting Configurations**: To prevent unintended behavior, avoid setting spark.executor.instances or using the --num-executors parameter when dynamic allocation is enabled. Let Spark manage the executor count within the specified dynamic allocation bounds.
* **Configure Dynamic Allocation Parameters Appropriately**: Set spark.dynamicAllocation.minExecutors, spark.dynamicAllocation.maxExecutors, and spark.dynamicAllocation.initialExecutors according to your application's workload characteristics to ensure optimal performance.
* **Enable External Shuffle Service**: Ensure that the external shuffle service is enabled (spark.shuffle.service.enabled = true) when using dynamic allocation to allow for safe removal of idle executors without affecting shuffle data.