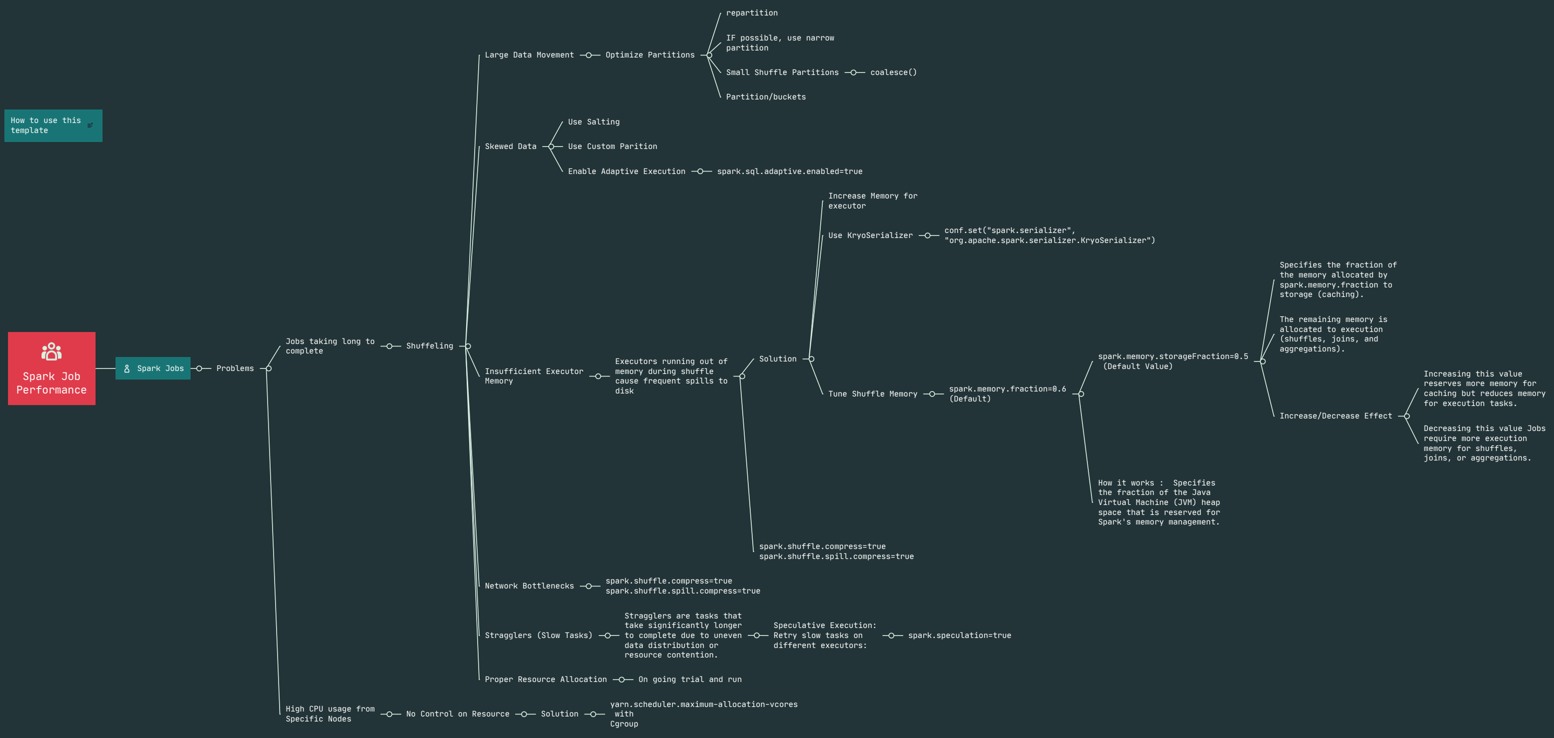

How to use this template

- Spark Job Performance
  - Spark Jobs
    - Problems
      - Jobs taking long to complete
        - Shuffeling
          - Large Data Movement
            - Optimize Partitions
              - repartition
              - IF possible, use narrow partition
              - Small Shuffle Partitions
                - coalesce()
              - Partition/buckets
          - Skewed Data
            - Use Salting
            - Use Custom Parition
            - Enable Adaptive Execution
              - spark.sql.adaptive.enabled=true
          - Insufficient Executor Memory
            - Executors running out of memory during shuffle cause frequent spills to disk
              - Solution
                - Increase Memory for executor
                - Use KryoSerializer
                  - `conf.set("spark.serializer", "org.apache.spark.serializer.KryoSerializer")`
                - Tune Shuffle Memory
                  - spark.memory.fraction=0.6 (Default)
                    - spark.memory.storageFraction=0.5 (Default Value)
                      - Specifies the fraction of the memory allocated by spark.memory.fraction to storage (caching).
                      - The remaining memory is allocated to execution (shuffles, joins, and aggregations).
                      - Increase/Decrease Effect
                        - Increasing this value reserves more memory for caching but reduces memory for execution tasks.
                        - Decreasing this value Jobs require more execution memory for shuffles, joins, or aggregations.
                    - How it works : Specifies the fraction of the Java Virtual Machine (JVM) heap space that is reserved for Spark's memory management.
              - spark.shuffle.compress=true
                spark.shuffle.spill.compress=true
          - Network Bottlenecks
            - spark.shuffle.compress=true
              spark.shuffle.spill.compress=true
          - Stragglers (Slow Tasks)
            - Stragglers are tasks that take significantly longer to complete due to uneven data distribution or resource contention.
              - Speculative Execution: Retry slow tasks on different executors:
                - spark.speculation=true
          - Proper Resource Allocation
            - On going trial and run
      - High CPU usage from Specific Nodes
        - No Control on Resource
          - Solution
            - yarn.scheduler.maximum-allocation-vcores with Cgroup

Spark Job Performance

1. Spark Jobs

- 1.1. Problems
  - 1.1.1. Jobs taking long to complete
    - 1.1.1.1. Shuffeling
      - 1.1.1.1.1. Large Data Movement
        - 1.1.1.1.1.1. Optimize Partitions
          - 1.1.1.1.1.1.1. repartition
          - 1.1.1.1.1.1.2. IF possible, use narrow partition
          - 1.1.1.1.1.1.3. Small Shuffle Partitions
            - 1.1.1.1.1.1.3.1. coalesce()
          - 1.1.1.1.1.1.4. Partition/buckets
      - 1.1.1.1.2. Skewed Data
        - 1.1.1.1.2.1. Use Salting
        - 1.1.1.1.2.2. Use Custom Parition
        - 1.1.1.1.2.3. Enable Adaptive Execution
          - 1.1.1.1.2.3.1. spark.sql.adaptive.enabled=true
      - 1.1.1.1.3. Insufficient Executor Memory
        - 1.1.1.1.3.1. Executors running out of memory during shuffle cause frequent spills to disk
          - 1.1.1.1.3.1.1. Solution
            - 1.1.1.1.3.1.1.1. Increase Memory for executor
            - 1.1.1.1.3.1.1.2. Use KryoSerializer
              - 1.1.1.1.3.1.1.2.1. conf.set("spark.serializer", "org.apache.spark.serializer.KryoSerializer")
            - 1.1.1.1.3.1.1.3. Tune Shuffle Memory
              - 1.1.1.1.3.1.1.3.1. spark.memory.fraction=0.6 (Default)
                - 1.1.1.1.3.1.1.3.1.1. spark.memory.storageFraction=0.5 (Default Value)
                  - 1.1.1.1.3.1.1.3.1.1.1. Specifies the fraction of the memory allocated by spark.memory.fraction to storage (caching).

1.1.1.1.3.1.1.3.1.1.2. The remaining memory is allocated to execution (shuffles, joins, and aggregations).

1.1.1.1.3.1.1.3.1.1.3. Increase/Decrease Effect

1.1.1.1.3.1.1.3.1.1.3.1. Increasing this value reserves more memory for caching but reduces memory for execution tasks.

1.1.1.1.3.1.1.3.1.1.3.2. Decreasing this value Jobs require more execution memory for shuffles, joins, or aggregations.

1.1.1.1.3.1.1.3.1.2. How it works : Specifies the fraction of the Java Virtual Machine (JVM) heap space that is reserved for Spark's memory management.

1.1.1.1.3.1.2. spark.shuffle.compress=true spark.shuffle.spill.compress=true

1.1.1.1.4. Network Bottlenecks

1.1.1.1.4.1. spark.shuffle.compress=true spark.shuffle.spill.compress=true

1.1.1.1.5. Stragglers (Slow Tasks)

1.1.1.1.5.1. Stragglers are tasks that take significantly longer to complete due to uneven data distribution or resource contention.

1.1.1.1.5.1.1. Speculative Execution: Retry slow tasks on different executors:

1.1.1.1.5.1.1.1. spark.speculation=true

1.1.1.1.6. Proper Resource Allocation

1.1.1.1.6.1. On going trial and run

1.1.2. High CPU usage from Specific Nodes

1.1.2.1. No Control on Resource

1.1.2.1.1. Solution

1.1.2.1.1.1. yarn.scheduler.maximum-allocation-vcores with Cgroup

2. How to use this template

Core concept: 6 people write 3 ideas in 5 minutes. Adapt according to the number of people in this exercise.

1. Every participant should make a copy of this map and write down the problem statement.
2. Each participant adds their name to the section "Person 1". They now have 5 minutes to come up with 3 ideas.
3. Each participant should now share their map with another team member. Repeat step 2 for another 5 minutes.
4. Repeat until the map is full.
5. When the brainwriting session is finished, you have 108 ideas (if done with 6 people), ready for assessment.