Apache Spark Mindmap

Let's start with **Spark Architecture + Components** mind map in **Mermaid format**.

Here's the code you can paste into any Markdown editor (that supports Mermaid, like Obsidian, Notion, GitHub, or even online Mermaid editors):

```
mindmap
  root((Apache Spark))
    Spark Architecture
      Driver Program
        - Coordinates execution
        - Converts code into DAG
        - Schedules tasks
      Cluster Manager
        - YARN
        - Mesos
        - Standalone
        - Kubernetes
      Executors
        - Run tasks
        - Store data (cache/persist)
        - Report back to Driver
    Core Concepts
      RDD
        - Immutable collection
        - Lazy transformations
        - Actions trigger execution
      DataFrame
        - Schema-based
        - Optimized by Catalyst
        - SQL support
      Dataset
        - Type-safe (Scala/Java)
        - Combines RDD + DataFrame
    Spark Ecosystem
      Spark SQL
      Spark Streaming
      MLlib
      GraphX
```

This will render into a **mind map** with:

- Spark at the center
- Architecture (Driver, Executors, Cluster Manager)
- Core Concepts (RDD, DataFrame, Dataset)
- Ecosystem modules (SQL, Streaming, MLlib, GraphX)

1. Spark RDD Mind Map

```
mindmap
  root((RDD - Resilient Distributed Dataset))
    Properties
      - Immutable
      - Partitioned
      - Fault-tolerant
    Operations
      Transformations
        - map
        - flatMap
        - filter
        - union
        - distinct
        - join
      Actions
        - collect
        - count
        - reduce
        - first
        - take
        - saveAsTextFile
    Persistence
      - cache()
      - persist()
      - unpersist()
    Fault Tolerance
      - Lineage (DAG)
      - Recompute lost partitions
```

---

2. Spark DataFrame & Dataset Mind Map

```
mindmap
  root((DataFrame & Dataset))
    DataFrame
      - Schema-based
      - Optimized by Catalyst
      - Compatible with SQL
    Dataset
      - Type-safe (Scala/Java)
      - Combines RDD + DataFrame
    Transformations
      - select, filter, groupBy
      - withColumn, drop
      - join, union
    Actions
      - show
      - count
      - collect
    Optimization
      - Catalyst Optimizer
      - Tungsten Execution Engine
      - Predicate Pushdown
```

---

3. Spark Streaming Mind Map

```
mindmap
  root((Structured Streaming))
    Sources
      - Kafka
      - Socket
      - Files
      - Rate Generator
    Operations
      - select, filter
      - groupBy, window
      - withWatermark
    Sinks
      - Console
      - File (Parquet/CSV)
      - Kafka
      - Memory
    Triggers
      - microBatch
      - continuous
    Output Modes
      - append
      - update
      - complete
    Fault Tolerance
      - Checkpointing
      - WAL (Write Ahead Log)
```

---

4. Spark SQL Mind Map

```
mindmap
  root((Spark SQL))
    Features
      - Unified query engine
      - Hive integration
      - Works with DataFrames
    Operations
      - createOrReplaceTempView
      - sql("SELECT ...")
      - df.select, df.filter
    Optimizations
      - Catalyst Optimizer
      - Predicate Pushdown
      - Column Pruning
    File Formats
      - Parquet
      - ORC
      - Avro
      - JSON
      - CSV
```

---

5. Spark Performance Tuning Mind Map

```
mindmap
  root((Performance Tuning))
    Serialization
      - KryoSerializer
      - JavaSerializer
    Memory Management
      - Storage vs Execution Memory
      - cache vs persist
    Partitioning
      - repartition
      - coalesce
    Shuffle
      - reduceByKey
      - groupByKey (avoid if possible)
    Resource Tuning
      - executor-memory
      - num-executors
      - cores-per-executor
    Data Skew Handling
      - Salting
      - Broadcast Join
```

With these 5 maps, you'll have a **visual Spark learning framework**:

1. RDD
2. DataFrame/Dataset
3. Streaming
4. SQL
5. Performance Tuning

1. Spark Architecture & Components

Spark consists of a **Driver Program** (coordinates execution, builds DAG, schedules tasks), **Cluster Manager** (e.g., YARN, Mesos, Standalone, Kubernetes), and **Executors** (run tasks, cache data, report back) Wikipedia.

```
import org.apache.spark.sql.SparkSession

object SparkArchitectureExample {
  def main(args: Array[String]): Unit = {
    val spark =
SparkSession.builder.appName("ArchExample").master("local[*]").getOrCr
eate()
    println(s"Spark version: ${spark.version}")
    spark.stop()
  }
}
```

---

2. RDD

RDDs are **immutable**, **partitioned**, and **fault tolerant**. They support lazy **transformations** (e.g., `map`, `filter`) and eager **actions** (e.g., `collect`, `count`). You can `cache()` them and recover partitions via **lineage in DAG** spark.apache.org.

```
val rdd = spark.sparkContext.parallelize(1 to 100)
val evens = rdd.filter(_ % 2 == 0)
println("Even count: " + evens.count())
```

---

3. DataFrame & Dataset

DataFrame is schema-based and optimized by Catalyst; **Dataset** adds type safety. Both support transformations like `select`, `filter`, `groupBy` and optimized via Catalyst/Tungsten MediumWikipedia.

```
case class Person(name: String, age: Int)
val ds = Seq(Person("Alice", 30), Person("Bob", 25)).toDS()
ds.filter(_.age >= 30).show()
```

---

4. Spark SQL

Spark SQL enables querying DataFrames using SQL via temporary views. It benefits from **Catalyst optimizations**, like predicate pushdown and column pruning WikipediaMedium.

```
val df = ds.toDF()
df.createOrReplaceTempView("people")
spark.sql("SELECT name, age FROM people WHERE age > 26").show()
```

---

5. Structured Streaming

Structured Streaming handles continuous data via sources (e.g., Kafka, Socket), operations (`groupBy`, `window`, withWatermark), and sinks (`console`, `Kafka`, Parquet), with triggers controlling micro-batches and guarantees via checkpointing Spark By {Examples}Wikipedia.

```
val lines =
spark.readStream.format("socket").option("host","localhost").option("p
ort",9999).load()
val counts = lines.as[String].groupBy("value").count()
val query =
counts.writeStream.format("console").outputMode("complete").start()
query.awaitTermination()
```

---

6. Performance Tuning

Key strategies include:

- Efficient **serialization** (Kryo),
- Memory management (cache vs persist),
- Partition control (`repartition`, `coalesce`),
- Minimizing shuffle (`reduceByKey` vs `groupByKey`),
- Handling skew (**broadcast joins**, salting) GistWikipedia.

```
val df = spark.read.option("header", "true").parquet("data/*.parquet")
val smallDim = spark.read.parquet("dim/*.parquet")
df.join(broadcast(smallDim), "id").count()
```

Repositories with Examples

- **spark-examples/spark-scala-examples** – A rich collection of Spark Scala snippets across RDD, DataFrame, SQL, Streaming, and more. GitHub
- **spark-examples/pyspark-examples** – Python equivalents for RDD, DataFrame, SQL, SparkSession, broadcast, etc. GitHub

These repos provide concrete code examples to reinforce the topics above and act as templates for experimentation.

Summary

Topic	Explanation Summary	Sample Code Linkage
Spark Architecture	Driver, Cluster Manager, Executors managing distributed workflows	SparkSession builder
RDD	Immutable collections with lazy transforms and lineage-based fault tolerance	RDD filter/count
DataFrame/Dataset	Schema-based data with optimizations via Catalyst and type safety in Dataset	DS filter example
Spark SQL	Query DataFrames using SQL with Catalyst optimizations	`createTempView()` + SQL query
Structured Streaming	Real-time processing with watermarking, sinks, triggers, and fault tolerance	socket stream + console output
Performance Tuning	Techniques like broadcast joins, partition tuning, optimized serialization	Broadcast join sample

---