

# Spark2 Internals and Performance Tuning

## *contents*

## TRAINING

- ☞ Description
- ☞ Intended Audience
- ☞ Key Skills
- ☞ Prerequisites
- ☞ Instructional Method
- ☞ course contents

# Spark2 Internals and Performance Tuning

## *course contents*

TRAINING

- ☞ Spark Introduction Day1
- ☞ RDDs
- ☞ RDD Internals:Part-1
- ☞ RDD Internals:Part-2
- ☞ Data ingress and egress
- ☞ Spark Internals Day2
- ☞ Advanced Spark Programming
- ☞ High Performance Spark patterns and programming Day3
- ☞ Spark SQL , Dataset, Dataframe Internals
- ☞ Spark Streaming
- ☞ Kafka Internals
- ☞ Tuning and Debugging Spark Day4
- ☞ Hardware profiling of Spark cluster at Runtime

### ***Description:***

- Internals of Spark 2.0 is explored in great detail. Programming paradigm of spark is given due importance. RDDs are compared to Dataframes and Data set. High performance programming patterns are explored with complex problems. For example, how to take advantage of copartition and colocation. Another example is against the grain Spark programming using the powerful Iterator to Iterator Spark Programming. All of this on a proper cluster on cloud. Most importantly how does it interact with other components like Cassandra, HBase, Mongo, Kafka etc. For example how do we perform a distributed join with Spark with data in multiple join tables. Or How we process real-time events from Kafka and then store processed data in Mongo for online consumption.

### ***Intended Audience:***

- Spark Programmers
- Programmers
- Spark Admin

### ***Key Skills:***

- Spark: In a cluster
- Spark: Streaming
- Spark: Tuning and Debugging
- Spark: SparkSQL
- Spark: RDDs comparison with other techniques
- Kafka
- Spark: High Performance Patterns
- Spark: Advanced Programming Model
- Spark: Detailed Architecture
- Spark: Dataframes
- Spark: Datasets

### ***Prerequisites:***

- Good knowledge of Java8 Lambda expressions
- Good understanding of map-reduce
- Good understanding of Java
- Good Knowledge of Spark Programming
- Good understanding of HDFS
- Linux would help as it would be the platform

### ***Instructional Method:***

- This is an instructor led course which provides lecture topics and the practical application of Spark, Spark Streaming and the underlying technologies. It pictorially

presents most concepts and there is a detailed case study that strings together the technologies, patterns and design.

# Spark2 Internals and Performance Tuning

## ■ *Spark Introduction*

- GraphX
- MLlib
- Spark SQL
- Data Processing Applications
- Spark Streaming
- What Is Apache Spark?
- Data Science Tasks
- Spark Core
- Storage Layers for Spark
- Who Uses Spark, and for What?
- A Unified Stack
- Cluster Managers

## ■ *RDDs*

- Lazy Evaluation
- Common Transformations and Actions
- Passing Functions to Spark
- Creating RDDs
- RDD Operations
- Actions
- Transformations
- Scala
- Java
- Persistence
- Python
- Converting Between RDD Types
- RDD Basics
- Basic RDDs

## ■ *RDD Internals:Part-1*

- Expressing Existing Programming Models

- Fault Recovery
- Interpreter Integration
- Memory Management
- Implementation
- MapReduce
- RDD Operations in Spark
- Iterative MapReduce
- Console Log Mining
- Google's Pregel
- User Applications Built with Spark
- Behavior with Insufficient Memory
- Support for Checkpointing
- A Fault-Tolerant Abstraction
- Evaluation
- Job Scheduling
- Spark Programming Interface
- Advantages of the RDD Model
- Understanding the Speedup
- Leveraging RDDs for Debugging
- Iterative Machine Learning Applications
- Explaining the Expressivity of RDDs
- Representing RDDs
- Applications Not Suitable for RDDs

## ■ ***RDD Internals:Part-2***

- Sorting Data
- Operations That Affect Partitioning
- Determining an RDD's Partitioner
- Grouping Data
- Motivation
- Aggregations
- Data Partitioning (Advanced)
- Actions Available on Pair RDDs

- Joins
- Creating Pair RDDs
- Operations That Benefit from Partitioning
- Transformations on Pair RDDs
- Example: PageRank
- Custom Partitioners

## ■ ***Data ingress and egress***

- Hadoop Input and Output Formats
- File Formats
- Local/“Regular” FS
- Text Files
- Java Database Connectivity
- Structured Data with Spark SQL
- Elasticsearch
- File Compression
- Apache Hive
- Cassandra
- Object Files
- Comma-Separated Values and Tab-Separated Values
- HBase
- Databases
- Filesystems
- SequenceFiles
- JSON
- HDFS
- Motivation
- JSON
- Amazon S3

## ■ ***Spark Internals***

- Spark:YARN Mode
  - Resource Manager
  - Node Manager

- Workers
- Containers
- Threads
- Task
- Executors
- Application Master
- Multiple Applications
- Tuning Parameters
- Spark:LocalMode
- Spark Caching
  - With Serialization
  - Off-heap
  - In Memory
- Running on a Cluster
  - Scheduling Within and Between Spark Applications
  - Spark Runtime Architecture
  - A Scala Spark Application Built with sbt
  - Packaging Your Code and Dependencies
  - Launching a Program
  - A Java Spark Application Built with Maven
  - Hadoop YARN
  - Deploying Applications with spark-submit
  - The Driver
  - Standalone Cluster Manager
  - Cluster Managers
  - Executors
  - Amazon EC2
  - Cluster Manager
  - Dependency Conflicts
  - Apache Mesos
  - Which Cluster Manager to Use?



- Spark Serialization
- StandAlone Mode
  - Task
  - Multiple Applications
  - Executors
  - Tuning Parameters
  - Workers
  - Threads
- ***Advanced Spark Programming***
  - Working on a Per-Partition Basis
  - Optimizing Broadcasts
  - Accumulators
  - Custom Accumulators
  - Accumulators and Fault Tolerance
  - Numeric RDD Operations
  - Piping to External Programs
  - Broadcast Variables
- ***High Performance Spark patterns and programming***
  - Effective Transformations
    - Shared Variables
    - Using Smaller Data Structures
    - Narrow Versus Wide Transformations
    - Reducing Setup Overhead
    - Deciding if Recompute Is Inexpensive Enough
    - Accumulators
    - Implications for Fault Tolerance
    - LRU Caching
    - Space and Time Advantages
    - What Type of RDD Does Your Transformation Return?
    - Cases for Reuse
    - Implications for Performance

- Minimizing Object Creation
- Alluxio (nee Tachyon)
- Interaction with Accumulators
- Noisy Cluster Considerations
- Reusing Existing Objects
- Types of Reuse: Cache, Persist, Checkpoint, Shuffle Files
- Broadcast Variables
- Set Operations
- Reusing RDDs
- The Special Case of coalesce
- Working with Key/Value Data
  - The Powerful Iterator-to-Iterator transformation
    - Reduce to Distinct on Each Partition
    - A Different Approach to Key/Value
    - Straggler Detection and Unbalanced
    - Sort on Cell Values
  - Secondary Sort Solution
  - groupByKey Solution
    - Multiple RDD Operations
    - Choosing an Aggregation Operation
    - Sorting by Two Keys with SortByKey
    - Preserving Partitioning Information Across Transformations
    - Dictionary of OrderedRDDOperations
    - Leveraging Co-Located and Co-Partitioned RDDs
    - Leveraging repartitionAndSortWithinPartitions for a Group by Key and
    - Range Partitioning
    - Dictionary of Mapping and Partitioning Functions
    - PairRDDFunctions
    - Sort Values Function
    - Partitioners and Key/Value Data
    - Co-Grouping

- Custom Partitioning
- Secondary Sort and repartitionAndSortWithinPartitions
- Using the Spark Partitioner Object
- Hash Partitioning
- Dictionary of Aggregation Operations with Performance Considerations
- Iterative Solution
  - What's So Dangerous About the groupByKey Function
  - How to Use PairRDDFunctions and OrderedRDDFunctions
  - Actions on Key/Value Pairs
- How Not to Sort by Two Orderings
- ***Spark SQL , Dataset, Dataframe Internals***
  - Spark SQL—Structured Queries on Large Scale
    - UserDefinedAggregateFunction—User-Defined Aggregate Functions (UDAFs)
    - User-Defined Functions (UDFs)
    - Schema—Structure of Data
    - Dataset Operators
    - Encoders—Internal Row Converters
    - Joins
    - StructField
    - StructType
    - InternalRow—Internal Binary Row Format
    - Builder—Building SparkSession with Fluent API
    - Aggregation—Typed and Untyped Grouping
    - Column Operators
    - Standard Functions—functions object
    - Window Aggregate Operators—Windows
    - Caching
    - Data Types
    - Row
    - Datasets—Strongly-Typed DataFrames with Encoders

- DataFrame—Dataset of Rows
- SparkSession—The Entry Point to Spark SQL
- RowEncoder—DataFrame Encoder
- DataSource API—Loading and Saving Datasets
  - Custom Formats
  - DDLStrategy
  - CSVFileFormat
  - BaseRelation
  - DataFrameWriter
  - JoinSelection
  - Query Execution
  - DataSource—Pluggable Data Sources
  - SparkPlanner—Default Query Planner (with no Hive Support)
  - ParquetFileFormat
  - QueryPlanner—From Logical to Physical Plans
  - Structured Query Plan
  - DataSourceRegister
  - DataSourceStrategy
  - BasicOperators
  - FileSourceStrategy
- EnsureRequirements Physical Plan Optimization
  - AlterViewAsCommand Runnable Command
  - SparkPlan—Physical Execution Plan
  - Join Logical Operator
  - CoalesceExec Physical Operator
  - LocalTableScanExec Physical Operator
  - ExplainCommand Logical Command
  - ClearCacheCommand Runnable Command
  - ExecutedCommandExec Physical Operator
  - CreateViewCommand Runnable Command
  - BroadcastHashJoinExec Physical Operator
  - ShuffledRowRDD

- ExchangeCoordinator and Adaptive Query Execution
- Debugging Query Execution
- LocalRelation Logical Operator
- CheckAnalysis
- BroadcastNestedLoopJoinExec Physical Operator
- Logical Query Plan Analyzer
- InMemoryRelation Logical Operator
- InMemoryTableScanExec Physical Operator
- LogicalPlan—Logical Query Plan
- ShuffleExchange Physical Operator
- DeserializeToObject Logical Operator
- WindowExec Physical Operator
- Joins (SQL & Core)
  - Core Spark Joins
  - Spark SQL Joins
  - Choosing a Join Type
  - DataFrame Joins
  - Choosing an Execution Plan
- Datasets vs DataFrames vs RDDs
  - Catalog
  - SQL Parser Framework
  - Eliminate Serialization
  - Combine Typed Filters
  - Predicate Pushdown / Filter Pushdown
  - CatalystSerde
  - Nullability (NULL Value) Propagation
  - SessionState
  - Logical Query Plan Optimizer
  - Column Pruning
  - Constant Folding
  - Vectorized Parquet Decoder
  - SessionCatalog

- Propagate Empty Relation
- SQLExecution Helper Object
- Simplify Casts
- SQLConf
- GetCurrentDatabase / ComputeCurrentTime
- SparkSqlAstBuilder
- ExternalCatalog—System Catalog of Permanent Entities
- Tungsten Execution Backend (aka Project Tungsten)
  - Catalyst—Tree Manipulation Framework
  - Spark SQL CLI - spark-sql
  - Whole-Stage Code Generation (CodeGen)
  - Attribute Expression
  - Hive Integration
  - Expression TreeNode
  - SparkSQLEnv
  - CacheManager—In-Memory Cache for Cached Tables
  - Settings
  - (obsolete) SQLContext
  - TreeNode
  - Generator
  - DataSinks Strategy
  - Thrift JDBC/ODBC Server—Spark Thrift Server (STS)
- DataFrames, Datasets & Spark SQL
  - Query Optimize
    - Debugging Spark SQL Queries
    - Large Query Plans and Iterative Algorithms
    - JDBC/ODBC Server
    - Code Generation
    - Logical and Physical Plans
  - Data Loading and Saving Functions
    - DataFrameWriter and DataFrameReader
    - Save Modes

- Partitions (Discovery and Writing)
- Formats
- Datasets
  - Easier Functional (RDD “like”) Transformations
  - Compile-Time Strong Typing
  - Grouped Operations on Datasets
  - Extending with User-Defined Functions and Aggregate Functions (UDFs, UDAFs)
  - Multi-Dataset Relational Transformations
  - Relational Transformations
  - Interoperability with RDDs, DataFrames, and Local Collections
- DataFrame API
  - Multi-DataFrame Transformations
  - Data Representation in DataFrames and Datasets
  - Transformations
  - Plain Old SQL Queries and Interacting with Hive Data
- Spark SQL Dependencies
  - Managing Spark Dependencies
  - Basics of Schemas
  - Avoiding Hive JARs
- Getting Started with the SparkSession (or HiveContext or SQLContext)
- ***Spark Streaming***
  - Stateless Transformations
  - Output Operations
  - Checkpointing
  - Core Sources
  - Receiver Fault Tolerance
  - Worker Fault Tolerance
  - Stateful Transformations
  - Batch and Window Sizes

- Architecture and Abstraction
- Performance Considerations
- Streaming UI
- Driver Fault Tolerance
- Multiple Sources and Cluster Sizing
- Processing Guarantees
- A Simple Example
- Input Sources
- Additional Sources
- Transformations
- ***Kafka Internals***
  - Kafka Core Concepts
    - brokers
    - Topics
    - producers
    - replicas
    - Partitions
    - consumers
  - Operating Kafka
    - P&S tuning
    - monitoring
    - deploying
    - Architecture
    - hardware specs
  - Developing Kafka apps
    - serialization
    - compression
    - testing
    - Case Study
    - reading from Kafka
    - Writing to Kafka



- ***Tuning and Debugging Spark***
  - Driver and Executor Logs
  - Memory Management
  - Finding Information
  - Configuring Spark with SparkConf
  - Key Performance Considerations
  - Components of Execution: Jobs, Tasks, and Stages
  - Spark Web UI
  - Hardware Provisioning
  - Level of Parallelism
  - Serialization Format
- Memory Management
- Driver and Executor Logs
- Components of Execution: Jobs, Tasks, and Stages
- Key Performance Considerations
- Hardware Provisioning
- Metrics and Debugging
  - Evaluating spark jobs
  - Monitoring tool for spark
  - Spark WebUI
  - Memory consumption and resource allocation
  - Job metrics
  - Debugging & troubleshooting spark jobs
  - Monitoring Spark jobs
- Level of Parallelism
- Monitoring Spark
  - Logging in Spark
  - Spark History Server
  - Exploring the Spark Application UI
  - Spark Metrics
- Finding Information

- Spark Administration & Best Practices
  - Estimating cluster resource requirements
  - Estimating Drive/Executor Memory Sizes
- Serialization Format
- ***Hardware profiling of Spark cluster at Runtime***
  - Adapting Perf for cluster with NOSQL
  - Understanding Hardware performance counters
  - Making perf JVM aware
  - Understanding Perf