Spark Internals and Performance Tuning

TRAINING

contents

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

Spark Internals and Performance Tuning

TRAINING

course contents

- Distributing Data with HDFS Day1
- Understanding Hadoop I/O
- Hadoop-2
- Spark Introduction Day2
- RDDs
- RDD Internals:Part-1
- RDD Internals:Part-2 Day3
- Data ingress and egress
- Advanced Spark Programming
- Running on a Cluster
- Spark Internals
- High Performance Spark patterns and programming Day4
- Spark SQL , Dataset, Dataframe Internals
- Spark Streaming
- Kafka Internals Day5
- Tuning and Debugging Spark
- Hardware profiling of Spark cluster at Runtime

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

Description:

- Internals of Spark 2.0 is explored in great detail. Programing 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:

- Networking specialists
- Programmers
- Engineers

Key Skills:

- Hadoop-2 Technologies:YARN Detailed Architecture
- How we use Hadoop an spark in practice to solve problems
- Proper Cluster Setup in classroom
- Spark: RDDs as Data Structure
- Spark: SQL
- Spark: Detailed Architecture
- Advanced Map Reduce Concepts & Algorithms
- Spark: In a cluster
- Spark: Streaming
- Advanced Map Reduce Patterns:Cluster utilization
- Spark: RDDs comparison with other techniques
- Advanced Map Reduce Patterns:Compute Intensive tasks
- Spark: Programming Model

Prerequisites:

- Specifically how Comparator and Comparable work in Java.
- For Spark, as it embraces the functional programming, a knowledge of lambda expression in Java 8 would be an added advantage.
- Working knowledge of Java

Instructional Method:

- This is an instructor led course which provides lecture topics and the practical application of Hadoop and the underlying technologies. It pictorially presents most

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

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

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

Spark Internals and Performance Tuning

- ***Distributing Data with HDFS***
  - Interfaces
  - Hadoop Filesystems
  - The Design of HDFS
  - Using Hadoop Archives
    - Limitations
  - Parallel Copying with distcp
    - Keeping an HDFS Cluster Balanced
    - Hadoop Archives
  - Data Flow
    - Anatomy of a File Write
    - Anatomy of a File Read
    - Coherency Model
  - The Command-Line Interface
    - Basic Filesystem Operations
  - The Java Interface
    - Querying the Filesystem
    - Reading Data Using the FileSystem API
    - Directories
    - Deleting Data
    - Reading Data from a Hadoop URL
    - Writing Data
- ***Understanding Hadoop I/O***
  - Data Integrity
    - ChecksumFileSystem
    - LocalFileSystem
    - Data Integrity in HDFS
  - Serialization
    - Implementing a Custom Writable

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

- Serialization Frameworks
- The Writable Interface
- Writable Classes
- Avro

- ORC Files
  - Large size enables efficient read of columns
  - New types (datetime, decimal)
  - Encoding specific to the column type
  - Default stripe size is 250 MB
  - A single file as output of each task
  - Split files without scanning for markers
  - Bound the amount of memory required for reading or writing.
  - Lowers pressure on the NameNode
  - Dramatically simplifies integration with Hive
  - Break file into sets of rows called a stripe
  - Complex types (struct, list, map, union)
  - Support for the Hive type model
  - ORC File:Footer
    - Count, min, max, and sum for each column
    - Types, number of rows
    - Contains list of stripes
  - ORC Files:Index
    - Required for skipping rows
    - Position in each stream
    - Min and max for each column
    - Currently every 10,000 rows
    - Could include bit field or bloom filter
  - ORC Files:Postscript
    - Contains compression parameters
    - Size of compressed footer
  - ORC Files:Data

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

        - Directory of stream locations
        - Required for table scan
    - Parquet
        - Nested Encoding
        - Configurations
        - Error recovery
        - Extensibility
        - Nulls
        - File format
        - Data Pages
        - Motivation
        - Unit of parallelization
        - Logical Types
        - Metadata
        - Modules
        - Column chunks
        - Separating metadata and column data
        - Checksumming
        - Types
    - File-Based Data Structures
        - MapFile
        - SequenceFile
    - Compression
        - Codecs
        - Using Compression in MapReduce
        - Compression and Input Splits
- ***Hadoop-2***
    - Apache Tez
        - Apache Tez: A New Chapter in Hadoop Data Processing
        - Data Processing API in Apache Tez
        - Writing a Tez Input/Processor/Output

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

- Runtime API in Apache Tez
- Apache Tez: Dynamic Graph Reconfiguration

- Apache YARN
  - Agility
  - global ResourceManager
  - per-node slave NodeManager
  - Scalability
  - Support for workloads other than MapReduce
  - Compatibility with MapReduce
  - per-application Container running on a NodeManager
  - Improved cluster utilization
  - per-application ApplicationMaster
- HDFS-2
  - High Availability for HDFS
  - HDFS-append support
  - HDFS Federation
  - HDFS Snapshots

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

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

- Common Transformations and Actions
- Passing Functions to Spark
- RDD Operations
- Creating RDDs
- 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
  - User Applications Built with Spark
  - Google's Pregel
  - Console Log Minning
  - Iterative MapReduce
  - Behavior with Insufficient Memory
  - A Fault-Tolerant Abstraction
  - Support for Checkpointing
  - Evaluation
  - Spark Programming Interface
  - Job Scheduling
  - Advantages of the RDD Model

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

- 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
  - Determining an RDD's Partitioner
  - Operations That Affect Partitioning
  - 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***
  - File Formats
  - Hadoop Input and Output Formats
  - Local/"Regular" FS
  - Text Files
  - Java Database Connectivity
  - Structured Data with Spark SQL
  - Elasticsearch
  - File Compression
  - Apache Hive
  - Cassandra

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

- Object Files
- Comma-Separated Values and Tab-Separated Values
- HBase
- Databases
- Filesystems
- SequenceFiles
- JSON
- HDFS
- Motivation
- JSON
- Amazon S3

- ***Advanced Spark Programming***
  - Working on a Per-Partition Basis
  - Accumulators
  - Optimizing Broadcasts
  - Custom Accumulators
  - Accumulators and Fault Tolerance
  - Numeric RDD Operations
  - Piping to External Programs
  - Broadcast Variables
- ***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

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

- Executors
- Amazon EC2
- Cluster Manager
- Dependency Conflicts
- Apache Mesos
- Which Cluster Manager to Use?

- ***Spark Internals***
  - Spark:YARN Mode
    - Resource Manager
    - Node Manager
    - Workers
    - Containers
    - Threads
    - Task
    - Executers
    - 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

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

- 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
  - Executers
  - Tuning Parameters
  - Workers
  - Threads

- ***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

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

- 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

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

- 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

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

- DataFrame—Dataset of Rows
- SparkSession—The Entry Point to Spark SQL
- RowEncoder—DataFrame Encoder

- DataSource API—Loading and Saving Datasets
  - DDLStrategy
  - Custom Formats
  - 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
  - DataSourceStrategy
  - DataSourceRegister
  - Structured Query Plan
  - 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

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

- 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

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

- 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
    - Logical and Physical Plans
    - Code Generation
  - Data Loading and Saving Functions
    - DataFrameWriter and DataFrameReader
    - Save Modes

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

- 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
  - Interoperability with RDDs, DataFrames, and Local Collections
  - Relational Transformations
- Spark SQL Dependencies
  - Managing Spark Dependencies
  - Basics of Schemas
  - Avoiding Hive JARs
- DataFrame API
  - Multi-DataFrame Transformations
  - Data Representation in DataFrames and Datasets
  - Transformations
  - Plain Old SQL Queries and Interacting with Hive Data
- Getting Started with the SparkSession (or HiveContext or SQLContext)

- ***Spark Streaming***
  - Checkpointing
  - Output Operations
  - Stateless Transformations
  - Receiver Fault Tolerance
  - Core Sources
  - Worker Fault Tolerance
  - Stateful Transformations
  - Batch and Window Sizes

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

- Performance Considerations
- Architecture and Abstraction
- 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

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

- ***Tuning and Debugging Spark***
  - Driver and Executor Logs
  - Memory Management
  - Finding Information
  - Key Performance Considerations
  - Configuring Spark with SparkConf
  - 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
  - Spark Metrics
  - Exploring the Spark Application UI
- Finding Information

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

- Spark Administration & Best Practices
  - Estimating cluster resource requirements
  - Estimating Drive/Executer 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

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io