Spark, Kafka and Cassandra Internals

TRAINING

contents

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
website:www.movaatechnologies.com

Spark, Kafka and Cassandra Internals

TRAINING

course contents

- Distributing Data with HDFS Day1
- Understanding Hadoop I/O
- Spark Introduction
- RDDs
- RDD Internals:Part-1
- RDD Internals:Part-2 Day2
- Data ingress and egress
- Running on a Cluster
- Spark Internals
- Advanced Spark Programming
- Spark Streaming
- Spark SQL Day3
- Tuning and Debugging Spark
- Kafka Internals
- Storm Internals
- Cassandra Internals Day4

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
website:www.movaatechnologies.com

Description:

- Spark is explored in great detail. Programing paradigm of spark is given due importance. RDDs is explored as data structures. The novelty of RDDs is brought into focus. Contrast is done with Hadoop Map reduce. All of this on a proper cluster.

Intended Audience:

- Programmers
- Engineers

Key Skills:

- Spark: In a cluster
- Spark: Streaming
- Spark: Programming Model
- Spark: RDDs comparison with other techniques
- Spark: Detailed Architecture
- Spark: RDDs as Data Structure

Prerequisites:

- Good knowledge of Java8 Lambda expressions
- Good understanding of Hadoop
- Good understanding of Java
- 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.

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
website:www.movaatechnologies.com

Spark, Kafka and Cassandra Internals

- ***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
website:www.movaatechnologies.com

- 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

- 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

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

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 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

- 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

- ***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
- ***Spark Streaming***
  - Checkpointing
  - Output Operations
  - Stateless Transformations
  - Receiver Fault Tolerance

- Core Sources
- Worker Fault Tolerance
- Stateful Transformations
- Batch and Window Sizes
- 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

- ***Spark SQL***
  - User-Defined Functions
  - Long-Lived Tables and Queries
  - Spark SQL Performance
  - Apache Hive
  - Loading and Saving Data
  - Performance Tuning Options
  - Parquet
  - Initializing Spark SQL
  - Caching
  - SchemaRDDs
  - JSON
  - From RDDs
  - Linking with Spark SQL
  - Spark SQL UDFs
  - Using Spark SQL in Applications
  - Basic Query Example
- ***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

- ***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

- ***Storm Internals***
  - Developing Storm apps
    - Case Studies
    - Bolts and topologies
    - P&S tuning
    - serialization
    - testing
    - Kafka integration
  - Storm core concepts
    - spouts
    - groupings
    - tuples
    - bolts
    - parallelism
    - Topologies
  - Operating Storm
    - monitoring
    - Architecture
    - deploying
    - hardware specs
- ***Cassandra Internals***
  - Cassandra in a cluster
    - Replication Strategies
    - Seed Nodes
    - Adding Nodes to a Cluster
    - Node Configuration
    - Cassandra Cluster Manager
    - Creating a Cluster
    - Dynamic Ring Participation
    - Snitches
    - Partitioners

mobile:+91.9880951838
mailto:mohit.riverstone@gmail.com
website:www.movaatechnologies.com

- The Cassandra Query Language
  - Data Types
  - CQL1
  - The Relational Data Model
  - CQL3
  - CQL Types
  - Cassandra's Data Model
  - Secondary Indexes
  - CQL2
- Performance Tuning
  - Memtables
  - Commit Logs
  - Caching
  - Compaction
  - Hinted Handoff
  - JVM Settings
  - Concurrency and Threading
  - SSTables
  - Networking and Timeouts
  - Managing Performance
  - Using cassandra-stress
- Cassandra Introduction
  - A Quick Review of Relational Databases
  - Beyond Relational Databases
  - Web Scale
  - What's Wrong with Relational Databases?
  - The Cassandra Elevator Pitch
  - The Rise of NoSQL
  - Where Did Cassandra Come From?
  - Is Cassandra a Good Fit for My Project?
- The Cassandra Architecture
  - System Keyspaces

- Partitioners
- Data Centers and Racks
- Staged Event-Driven Architecture (SEDA)
- Lightweight Transactions and Paxos
- Rings and Tokens
- Compaction
- Queries and Coordinator Nodes
- Caching
- Consistency Levels
- Hinted Handoff
- Bloom Filters
- Gossip and Failure Detection
- Anti-Entropy, Repair, and Merkle Trees
- Snitches
- Virtual Nodes
- Managers and Services
- Replication Strategies
- Memtables, SSTables, and Commit Logs
- Tombstones

Data Modeling

- Evaluating and Refining
- Conceptual Data Modeling
- Defining Database Schema
- Defining Application Queries
- Logical Data Modeling
- RDBMS Design
- Physical Data Modeling

Monitoring and Maintenance

- Logging
- Cassandra’s MBeans
- Backup and Recovery
- Maintenance Tools

- SSTable Utilities
- Basic Maintenance
- Adding Nodes
- Handling Node Failure
- Health Check
- Monitoring with nodetool
- Monitoring Cassandra with JMX