Spark and Scala Course Content

Course Description:

Apache Spark is a powerful open-source monitoring and distributed general-purpose in-memory computing engine which is written in the Scala programming language. Apache Spark and Scala provide fault-tolerant in-memory data parallelism at a very fast, standard interface and ease of use. Most of the big data developers choosing Scala programming language in the Apache Spark framework to build big data applications with faster data analysis.

Scope of Apache Spark and Scala is immense because of its demandable key features like less complexity, fast memory computations, dynamic programming, inbuilt machine learning libraries, data streaming, real-time analysis, graphs libraries, high productivity, and performance.

Hachion Apache Spark and Scala Online training curated by industry experts from scratch. Our Spark and scala tutorial covers all basic fundamental concepts to updated advanced topics in the course content. Enhance full knowledge with the course that includes Scala programming language, Spark architecture, RDD for creating apps in Apache Spark, SparkSQL, streaming, batch processing, ML programming, and graph analytics. Master your skills and get hands-on experience with given real-time projects to become a big data developer.

Course Content:

Introduction to Data Analysis with Spark

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

Downloading Spark and Getting Started

- Downloading Spark
- Introduction to Spark's Python and Scala Shells
- Introduction to Core Spark Concepts

- Standalone Applications
- Initializing a SparkContext
- Building

Programming with RDDs

- Standalone Applications
- RDD Basics
- Creating RDDs
- RDD Operations
- Transformations
- Actions
- Lazy Evaluation
- Passing Functions to Spark
- Common Transformations and Actions
- Basic RDDs
- Converting Between RDD Types
- Persistence (Caching)

Working with Key/Value Pairs

- Motivation
- Creating Pair RDDs
- Transformations on Pair RDDs
- Aggregations
- Grouping Data
- Joins
- Sorting Data
- Actions Available on Pair RDDs
- Data Partitioning (Advanced)
- Determining an RDD's Partitioner
- Operations That Benefit from Partitioning
- Operations That Affect Partitioning
- Example: PageRank
- Custom Partitioners

Loading and Saving Your Data

- Motivation
- File Formats
- Text Files
- JSON
- Comma-Separated Values and Tab-Separated Values
- SequenceFiles
- Object Files

- Hadoop Input and Output Formats
- File Compression
- Filesystems
- Local/“Regular” FS
- Amazon S
- HDFS
- Structured Data with Spark SQL
- Apache Hive
- JSON
- Databases
- Java Database Connectivity
- Cassandra
- HBase
- Elasticsearch

Advanced Spark Programming

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

Running on a Cluster

- Introduction
- Spark Runtime Architecture
- The Driver
- Executors
- Cluster Manager
- Launching a Program

Deploying Applications with spark-submit

- Packaging Your Code and Dependencies
- A Java Spark Application Built with Maven
- A Scala Spark Application Built with sbt
- Dependency Conflicts
- Scheduling Within and Between Spark Applications
- Cluster Managers
- Standalone Cluster Manager

- Hadoop YARN
- Apache Mesos
- Amazon EC
- Which Cluster Manager to Use?

Tuning and Debugging Spark

- Configuring Spark with SparkConf
- Components of Execution: Jobs, Tasks, and Stages
- Finding Information
- Spark Web UI
- Driver and Executor Logs
- Key Performance Considerations
- Level of Parallelism
- Serialization Format
- Memory Management
- Hardware Provisioning

Spark SQL

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

Spark Streaming

- A Simple Example
- Architecture and Abstraction
- Transformations

- Stateless Transformations
- Stateful Transformations
- Output Operations
- Input Sources
- Core Sources
- Additional Sources
- Multiple Sources and Cluster Sizing / Operation
- Checkpointing
- Driver Fault Tolerance
- Worker Fault Tolerance
- Receiver Fault Tolerance
- Processing Guarantees
- Streaming UI
- Performance Considerations
- Batch and Window Sizes
- Level of Parallelism
- Garbage Collection and Memory Usage

Machine Learning with MLlib

- Overview
- System Requirements
- Machine Learning Basics
- Example: Spam Classification
- Data Types
- Working with Vectors
- Algorithms
- Feature Extraction
- Statistics
- Classification and Regression
- Clustering
- Collaborative Filtering and Recommendation
- Dimensionality Reduction
- Model Evaluation
- Tips and Performance Considerations
- Preparing Features
- Configuring Algorithms
- Table of Contents | vii
- Caching RDDs to Reuse
- Recognizing Sparsity
- Level of Parallelism
- Pipeline API