Spark2.0 Internals, Kafka and, NoSQL DBs

contents

TRAINING

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

Spark2.0 Internals, Kafka and, NoSQL DBs

course contents

TRAINING

- Distributing Data with HDFS Day1
- Understanding Hadoop I/O
- Spark Introduction
- RDDs Day2
- RDD Internals:Part-1
- RDD Internals:Part-2
- Data ingress and egress Day3
- Running on a Cluster
- Spark Internals
- Advanced Spark Programming
- Spark Streaming
- Spark SQL Day4
- Tuning and Debugging Spark
- Kafka Internals
- Cassandra Internals Day5
- MongoDB Internals
- Hbase

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

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. Most importantly how does it interact with other components like Cassandra, HBase, Mongo, Kafka , Storm 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:

- 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
mailto:mohit@stillwaters.ai
website:www.stillwaters.ai
blog:slowbreathing.github.io

Spark2.0 Internals, Kafka and, NoSQL DBs

- ***Distributing Data with HDFS***
  - Interfaces
  - Hadoop Filesystems
  - The Design of HDFS
  - Parallel Copying with distcp
    - Keeping an HDFS Cluster Balanced
    - Hadoop Archives
  - Using Hadoop Archives
    - Limitations
  - 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***
  - Serialization
    - Implementing a Custom Writable
    - Serialization Frameworks
    - The Writable Interface

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

- Writable Classes
- Avro

- Data Integrity
  - ChecksumFileSystem
  - LocalFileSystem
  - Data Integrity in HDFS
- 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

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

        - 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

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

- 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

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

- RDD Operations in Spark
- Iterative MapReduce
- Console Log Minning
- 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

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

- 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
- ***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

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

- 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

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

- 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
  - Optimizing Broadcasts
  - Accumulators
  - Custom Accumulators
  - Accumulators and Fault Tolerance
  - Numeric RDD Operations
  - Piping to External Programs
  - Broadcast Variables
- ***Spark Streaming***
  - Stateless Transformations

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

- 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

- ***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

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

- Using Spark SQL in Applications
- Basic Query Example

- ***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
  - 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
  - Hardware Provisioning
  - Level of Parallelism
  - Monitoring Spark
    - Logging in Spark
    - Spark History Server
    - Spark Metrics

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

        - Exploring the Spark Application UI
    - Finding Information
    - Spark Administration & Best Practices
        - Estimating cluster resource requirements
        - Estimating Drive/Executer Memory Sizes
    - 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
- ***Cassandra Internals***
    - Cassandra in a cluster
        - Replication Strategies
        - Seed Nodes

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

- Adding Nodes to a Cluster
- Node Configuration
- Cassandra Cluster Manager
- Creating a Cluster
- Dynamic Ring Participation
- Snitches
- Partitioners

- 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

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

- 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

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

- 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

- ***MongoDB Internals***
  - Indexing and query optimization
  - Replication
  - Sharding

  - Sharding
    - Starting the Servers
    - Adding a Shard from a Replica Set
    - Splitting Chunks
    - Chunk Ranges
    - Configuring Sharding
    - Sharding Data
    - Understanding the Components of a Cluster
    - When to Shard
    - The Balancer
    - Config Servers
    - Adding Capacity
    - The mongos Processes
    - How MongoDB Tracks Cluster Data

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

- Monitoring MongoDB Applications
  - False Positives
  - Seeing the Current Operations
  - Documents
  - Collections
  - Calculating Sizes
  - Finding Problematic Operations
  - Preventing Phantom Operations
  - Using mongotop and mongostat
  - Killing Operations
  - Using the System Profiler
  - Databases
  - Seeing What Your Application Is Doing
- Durability
  - What Journaling Does
  - Replacing Data Files
  - Durability with Replication
  - Sneaky Unclean Shutdowns
  - Planning Commit Batches
  - Checking for Corruption
  - What MongoDB Does Not Guarantee
  - Repairing Data Files
  - Setting Commit Intervals
  - Turning Off Journaling
  - The mongodlock File
- Advanced Sharding
  - Controlling Data Distribution
  - Location-Based Shard Keys
  - Hashed Shard Keys for GridFS
  - The Firehose Strategy
  - Shard Key Limitations
  - Ascending Shard Keys

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

- Shard Key Strategies
- Picturing Distributions
- Randomly Distributed Shard Keys
- Hashed Shard Key
- Choosing a Shard Key
- Shard Key Cardinality
- Manual Sharding
- Shard Key Rules and Guidelines
- Taking Stock of Your Usage
- Multi-Hotspot
- Using a Cluster for Multiple Databases and Collections

Hbase

- Intoduction
- Clients
- Concepts
- Hbase vs RDBMS

Log Structures Merge Trees

- Compaction
- Limitations of B+ Trees
- Limitations of Binary Trees
- LogStructured Merge tree as the back bone of storage

HBase Storage Architecture

- MemStore
- Read and Write Path
- Physical Architecture
- HFile
- WAL
- HFile Format
- HMaster and HRegionServer
- How Data is Store in Hfile
- Root Table and Meta Table
- Key Format

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

- Role ofZookeeper

- Future Directions
  - MMap for bloom filters and Block indexes
  - Exploring OFF-Heap Storage
- Introduction
  - Common Advantages
  - Dynamo and Bigtable
  - Table, Column Families,Rows and Columns
  - Data Mode
  - BigTable and HBase(C + P)
  - What am I giving up?
  - Schemaless
  - Key/Value Stores
- HBase Operations
  - Access Patterns
    - Batching
    - Filters
    - Put
    - Gets
    - Caching
    - Scanning
  - Desiging HBase Tables and Schemas
    - Time-Ordered Relations
    - Pagination
    - Concepts
    - Key Design
    - Partial Key Scans
    - Tall-Narrow Versus Flat-Wide Tables
    - Time Series Data
    - Secondary Indexes
    - Advanced Schemas

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