Reactive App using Actor model & Apache Spark

Rahul Kumar
Software Developer
@rahul_kumarAws
About Sigmoid

We build realtime & big data systems.

OUR CUSTOMERS

Capillary
PubNub
Fusionops
Agenda

- Big Data - Intro
- Distributed Application Design
- Actor Model
- Apache Spark
- Reactive Platform
- Demo
Data Management

Managing data and analysing data have always been the greatest benefit and the greatest challenge for an organization.
Three V’s of Big data

- Volume
- Variety
- Velocity
Scale Vertically (Scale Up)
Scale Horizontally (Scale out)

data files

Processing

database Report
“A distributed system is a software system in which components located on networked computers communicate and coordinate their actions by passing messages.”

Principles Of Distributed Application Design

- Availability
- Performance
- Reliability
- Scalability
- Manageability
- Cost
The fundamental idea of the actor model is to use actors as concurrent primitive that can act upon receiving messages in different ways:

- Send a finite number of messages to other actors
- spawn a finite number of new actors
- change its own internal behavior, taking effect when the next incoming message is handed.
Each actor instance is guaranteed to be run using at most one thread at a time, making concurrency much easier.

Actors can also be deployed remotely.

In Actor Model the basic unit is a message, which can be any object, but it should be serializable as well for remote actors.
Actors

For communication actor uses asynchronous message passing.

Each actor have their own mailbox and can be addressed.

Each actor can have no or more than one address.

Actor can send message to them self.
Akka is a toolkit and runtime for building highly concurrent, distributed, and resilient message-driven applications on the JVM.

- Simple Concurrency & Distribution
- High Performance
- Resilient by design
- Elastic & Decentralized
Akka Modules

**akka-actor** – Classic Actors, Typed Actors, IO Actor etc.

**akka-agent** – Agents, integrated with Scala STM

**akka-camel** – Apache Camel integration

**akka-cluster** – Cluster membership management, elastic routers.

**akka-kernel** – Akka microkernel for running a bare-bones mini application server

**akka-osgi** – base bundle for using Akka in OSGi containers, containing the akka-actor classes

**akka-osgi-aries** – Aries blueprint for provisioning actor systems

**akka-remote** – Remote Actors

**akka-slf4j** – SLF4J Logger (event bus listener)

**akka-testkit** – Toolkit for testing Actor systems

**akka-zeromq** – ZeroMQ integration
Akka Use case - 1

Fully fault-tolerance Text extraction system.

GATE : General architecture for Text processing
Akka Use case - 2

Real time Application Stats

Master Node

Application Logs

Worker Nodes

MySQL

System Logs

Application Stats
Project and libraries build upon Akka
Apache Spark is a fast and general execution engine for large-scale data processing.

- originally developed in the AMPLab at University of California, Berkeley
- Organize computation as concurrent tasks
- schedules tasks to multiple nodes
- Handle fault-tolerance, load balancing
- Developed on Actor Model
Apache Spark

- Speed
- Ease of Use
- Generality
- Run Everywhere

![Diagram showing Apache Spark components: Spark SQL, Spark Streaming, MLlib, GraphX]
We can run Spark using its **standalone cluster** mode, on **EC2**, on **Hadoop YARN**, or on **Apache Mesos**.
Resilient Distributed Datasets (RDDs), a distributed memory abstraction that lets programmers perform in-memory computations on large clusters in a fault-tolerant manner.

RDD shard the data over a cluster, like a virtualized, distributed collection.

Users create RDDs in two ways: by loading an external dataset, or by distributing a collection of objects such as List, Map etc.
RDD Operations

Two Kind of Operations

- Transformation
- Action

Spark computes RDD only in a lazy fashion.

Only computation start when an Action call on RDD.
scala> val lineRDD = sc.textFile("sherlockholmes.txt")


scala> val lowercaseRDD = lineRDD.map(line=> line.toLowerCase)


scala> lowercaseRDD.count()

res2: Long = 13052
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
object SparkWordCount {

  def main(args: Array[String]): Unit = {

    val sc = new SparkContext("local","SparkWordCount")

    val wordsCounted = sc.textFile(args(0)).map(line=> line.toLowerCase)
      .flatMap(line => line.split("\W+"))
      .groupBy(word => word)
      .map{ case(word, group) => (word, group.size)}

    wordsCounted.saveAsTextFile(args(1))
    sc.stop()
  }
}
Spark Cluster

Cluster Manager Can be Spark’s own Standalone Cluster Manager or Mesos or YARN
Spark Cache

pulling data sets into a cluster-wide in-memory

```scala
scala> val textFile = sc.textFile("README.md")


scala> val linesWithSpark = textFile.filter(line => line.contains("Spark"))


scala> linesWithSpark.cache()


scala> linesWithSpark.count()

res12: Long = 19
```
## Storage

<table>
<thead>
<tr>
<th>RDD Name</th>
<th>Storage Level</th>
<th>Cached Partitions</th>
<th>Fraction Cached</th>
<th>Size in Memory</th>
<th>Size in ExternalBlockStore</th>
<th>Size on Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>MapPartitionsRDD</td>
<td>Memory Deserialized 1x Replicated</td>
<td>2</td>
<td>100%</td>
<td>3.3 KB</td>
<td>0.0 B</td>
<td>0.0 B</td>
</tr>
</tbody>
</table>
Spark SQL

Mix SQL queries with Spark programs

Uniform Data Access, Connect to any data source

DataFrames and SQL provide a common way to access a variety of data sources, including Hive, Avro, Parquet, ORC, JSON, and JDBC.

Hive Compatibility Run unmodified Hive queries on existing data.

Connect through JDBC or ODBC.
Spark Streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams.
Reactive Application

Responsive
Resilient
Elastic
Message Driven

http://www.reactivemanifesto.org
Typesafe Reactive Platform
Typesafe Reactive Platform

Diagram showing the integration of various components such as Web, REST, Play, DBs, Akka, Spark, POSIX, HDFS, Mesos, Bare Metal, and IaaS (EC2, ...). The diagram illustrates how these components interconnect, with arrows indicating flow and connections.
Demo
Reference

http://spark.apache.org/docs/latest/quick-start.html
Learning Spark Lightning-Fast Big Data Analysis
  By Holden Karau, Andy Konwinski, Patrick Wendell, Matei Zaharia
https://www.playframework.com/documentation/2.4.x/Home
http://doc.akka.io/docs/akka/2.3.12/scala.html
Thank You