Real Time Fraud Detection in Financial Markets

Kaarthik Elangovan
University Of Stirling
MSc in Advanced Computing

Supervisor: Kevin Swingler

Project Goal

Fraud Detection Real Time in Financial Markets.

Applying Machine Learning techniques to find Patterns in Trading Activity.

Using Big Data/Cluster Computing Technologies.

Why this Project?

Business Application:

Securities/Stock/Investment fraud is a multi-million dollar business/crime and increasing every year.

Technology shift towards the next big thing:

Big Data

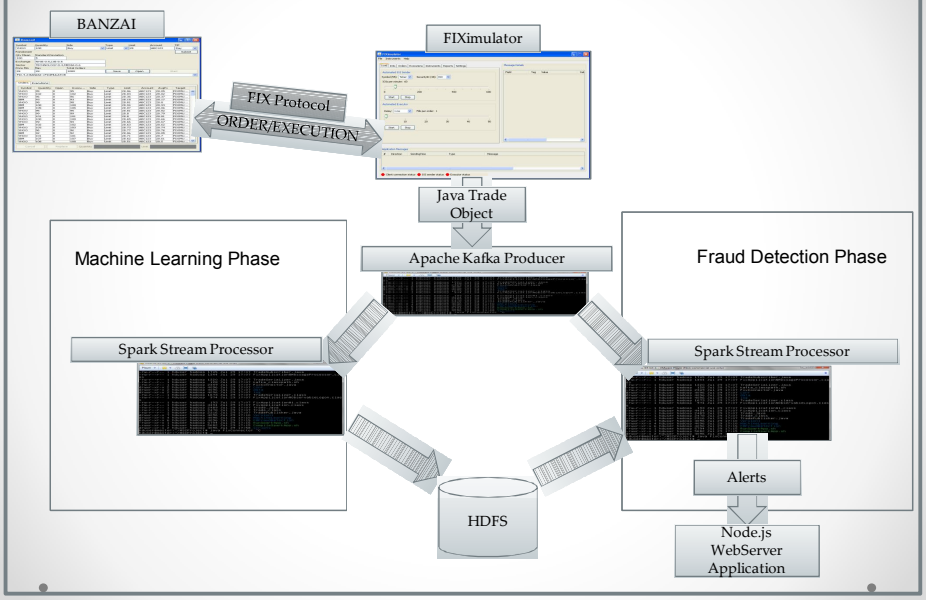
Cluster Computing

Implementation

Key Requirement:

- Data and finding patterns in data.
  - Issue:
    - Production/Real trader activity unavailable.
  - Solution: (Banzai – Fiximulator)
    - Data generator to the rescue. Generate trading activity with some pre-defined patterns. (e.g Quantity distributed normally, Exchange and Sectors split with probability distribution)
- Real-Time trades are published-subscribed using Distributed Message Queues – Apache Kafka
- Using Unsupervised Machine learning algorithm – EM/K-means to cluster/model the historical activity for each trader (assuming trading account as unique id).
- Save the model on a HDFS (Hadoop file system) – Big Data.
- Load the model in memory from HDFS using Spark – In-Memory Cluster Computing Technology
- Process streaming real-time trades in Spark
- Report outliers – email or HTML5/node.js App

Technical Overview



Component Details

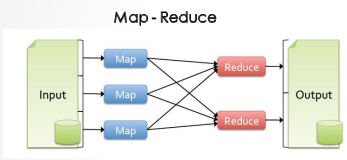
Banzai --- FIXimulator: Trading simulator – Java Swing/QuickfixJ Library (Open source FIX Protocol Implementation).

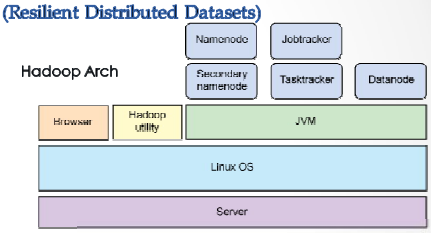
Apache Kafka: Distributed Message Queuing System - Java.

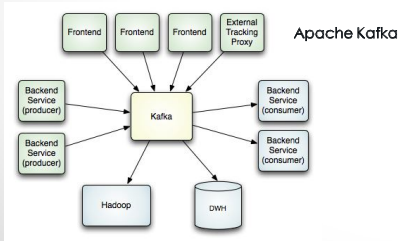
Spark Framework: In-Memory Cluster Computing – Scala/JVM/Java API

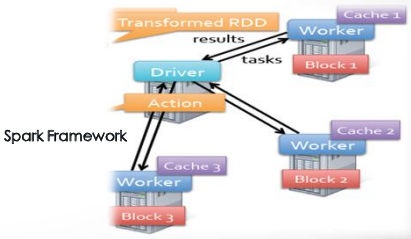
EM /K-Means Clustering Algorithms (Weka Implementation)

- Convert regular Java Objects into Spark Java RDDs (Resilient Distributed Datasets)









Fraudulent Activity

Defining Fraudulent Activity:

For the purpose of this project,

If the new trading activity for any given account doesn't fit into the generated model for that account, then raise an alarm.

** Assumption made to test the framework**

Traders usually follow a particular pattern of trading, manage portfolios with percent split between various exchanges and sectors.

** Known fraudulent activities like, Short Selling Abuses, Insider Trading, Microcap Fraud.

Progress and Challenges

Challenges:

Unavailability of real data and data with some pattern to prove the framework works.

To run the Spark/Hadoop application on a real cluster of nodes. Spark is meant to be run on a cluster, standalone mode performance is bad.

Progress:

Banzai – FIXimulator modified,

* support for account number
* loading stocks xml for randomization
* save/load randomization attributes for each account
* using uncommon math library to randomize order entry –
  Order Qty – Normally distributed using Mean/Standard deviation
  Exchange/Sector – Probability distributed (using %)
* drop copy from fiximulator to the apache kafka producer.

Apache Kafka Producer/Consumer,

* Kafka producer FIX message to Java trade objects
* Java trade objects serialized and published on a message queue.
* Kafka consumer to subscribe from the message queue and persist to file.

EM/K-Means algorithm implementation in Java (Algorithm logic similar to that of Weka)

To Convert the algorithm implementation into Spark implementation – Under development.

Persist data into HDFS, load model from HDFS, find outliers, report alerts – To be done.

Thank You