# Big Data and Market Surveillance

April 28, 2014

# Table of contents

# Introduction

Big data is a collection of data so large and complex that it becomes difficult to capture, store, search, analyze and visualize using traditional data processing methods. Big data is becoming more and more important in all industries, but none more so than in the financial sector.

Modern market surveillance applications process billions of transactions in real-time every business day. The current trend is that new regulatory and compliance requirements result in that the market data sets get larger and larger and the need to process longer time periods increases. Also the interest to use unstructured data, like news and sentiment, has increased lately.

In recent years a number of Big Data tools have arisen to handle these massive quantities of data. The tools parallelize large data sets across shared clusters built on low-cost commodity hardware and provide easy scaling while dramatically reducing the cost of environments compared to using traditional large servers.

This makes Big Data an interesting complement to more traditional means of implementing market surveillance functionality.

# Introduction to the technology of Big Data

This chapter introduces some of the techniques and concepts used in conjunction with Big Data. Readers familiar with Big Data are encouraged to skip this section.

## Distributed File System

File systems that manage the storage across a network of machines are called distributed file systems. A distributed file system aims to solve problems like fault tolerance and performance bottlenecks. Basically files are split into chunks and stored in a redundant fashion. Distributed file systems makes it theoretically possible to read thousands of gigabytes of data in a few seconds. They also enable cost efficient storage of very big amounts of data. There are currently systems in production with more than 100 petabytes of data.

## MapReduce

MapReduce is a programming model for processing large sets of data, typically used together with a distributed file system. The MapReduce pattern works very well if the algorithm is operating on data that can be isolated (no concurrent usage).

The algorithm has two steps that need to be implemented. The Wikipedia description of the steps are:

*"**Map step**: The master node takes the input, divides it into smaller sub-problems, and distributes them to worker nodes. A worker node may do this again in turn, leading to a multi-level tree structure. The worker node processes the smaller problem, and passes the answer back to its master node.*

***Reduce step**: The master node then collects the answers to all the sub-problems and combines them in some way to form the output – the answer to the problem it was originally trying to solve."*

A simple example of the algorithm: The problem is to count the times a member has been involved in a trade with a size larger than a threshold value. In this example the TradeId is the key and the Value is a set of Trade attributes, including Member and Trade size.

**MAP INPUT:** Map<TradeId, List<TradeAttributes>>

The Map step in Map Reduce will produce a new Map with Member as Key and Trade size as value:

**MAP OUTPUT:** Map<Member, TradeSize>

The next step is shuffle, typically handled by the MapReduce framework. The shuffle task will group all the values by Key.

**SHUFFLE OUTPUT:** Map<Member, List<TradeSize>>

This is also the input to the final task; the reduce. The reduce task is responsible for the logic on the data, in this case calculating the number of occurrences of a trade size larger than the threshold.

**REDUCE OUTPUT:** Map<Member, Number>

MapReduce works very well when the data is already available. It is essentially batch-oriented. Support for real-time or near real-time, such as stream processing has lately become a requirement to address, more on this in chapter 1.6.

## Ad hoc Querying

While implementing MapReduce for querying data is possible, there are also tools that provide data access using simpler methods. Typically they provide a query language to operate on the data for easy data summarization and ad-hoc queries.

## Visualization

While the tools for ad hoc querying provides simpler access, there are also tools available that out of the box provide rich visualization of big data including advanced graphics.

## Machine Learning

Implementing machine learning on large data requires scalable solutions. There are tools available, built upon tools for big data that deliver the scalability and enables classification, recommendation mining, and clustering on large data sets.

# Areas of functionality where Big Data can be applied

This chapter introduces functional areas where Big Data can be used to to provide cost efficient implementations.

## Back testing

One fact that is likely to remain true for a foreseeable future is that new alert rules, i.e. suspicious patterns to search for will continue to be invented. In a lot of cases it is desirable to test these newly invented alert rules against archived data, possibly against years worth of data.

Since major markets produce massive amounts of messages per day, US financial markets for example produce around 50 billions messages per day, this is an extremely computationally intensive task.

While back testing is a computationally intensive task it is also easy to parallelize. Multiple trading days can be back tested simultaneously making it an ideal candidate for big data techniques.

In order to complete back testing within a reasonable time in an environment like the mentioned example of US markets, Big Data techniques are an invaluable tool.

## Reconciliation and validation

While many types of validation and reconciliation can be done at the time of collection other types can only be performed later. This can have several reasons:

- The data needed to perform validation is not available until at a later time.
- The task of validation is too computationally intensive to perform in real-time.
- Re-validation is done as a result of changed conditions

An example is the data that is collected as a part of many financial authorities transaction-reporting programs. To properly validate the reported data it is necessary perform tasks like comparing reported trades with publically reported trades and spreads etc.

Another example is cross-referencing orders/trades routed over several execution venues, making sure that the entire trail of of an order can be tracked even if it has executions on multiple venues.

The activities listed above can be time consuming and lends itself well to parallelization offered by Big Data techniques.

## Searching (non-indexable conditions)

A core task of any market surveillance system is to search and filter orders/trades/quotes based on various criteria. While certain of these criteria are easy to support using different types of indexing schemes, others are not. An example is when the amount of potential criteria to search for is so large that it is simply not feasible from a resource perspective to create index for all of them.

Using Big Data technology searches can be implemented by brute force techniques giving a very high degree of search criteria flexibility while still completing within reasonable time frame.

## Reports

Complex reports can be designed and executed with good performance. Many of the Big Data technical solutions include integration with standard reporting tools and data formats minimizing the threshold to create new reports.

## Ad-hoc / Data Warehouse

Maybe the most important usage of Big Data is the ad-hoc usage where large amounts of data can be examined and mined using reporting and query facilities.

Most of the standard Big Data technical solutions include ad-hoc query facilities.

Big Data Ad-hoc reporting and querying have the potential to unlock the value of information previously hidden in massive amount of data that conventional tools were not able to process.

# Leveraging open source solutions

While there are proprietary Big Data technical solutions, the most innovative and most widely accepted base technologies are open source. A few examples include:

- Apache Hadoop – big data framework, includes MapReduce and Hadoop Distributed File System (HDFS). Hadoop is an open source framework for scalable, distributed computing that enables processing on large clusters of commodity hardware. The first version was released in 2005 and the framework is now used widely by major companies worldwide.
- Apache Hive – data warehouse system for Hadoop with query support using a SQL-like language. Hive is an open source data warehouse solution that makes it possible to query data stored in a Hadoop cluster using a SQL like syntax. Initially developed by Facebook, but is now used and developed also by other large companies. Query times are often measured in minutes and hours, which have resulted in development of alternatives more targeted for real-time queries.

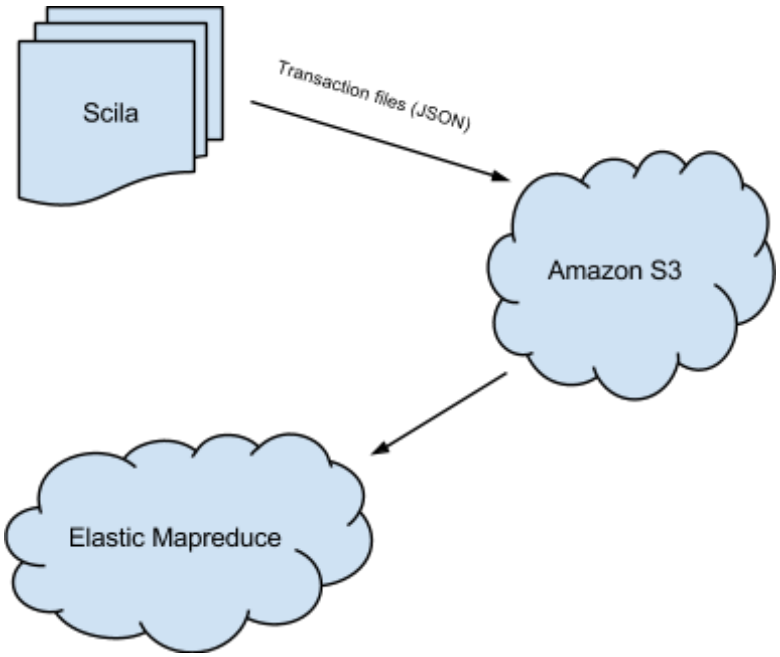
- Cloudera Hadoop distribution (CDH) - A distribution with bundled Hadoop and related tools. Offers offer unified batch processing, interactive SQL, and interactive search. CDH is free of charge, but it is possible to buy support and enterprise features from Cloudera.
- Pivotal - Big data suite, a bundle of commercial big data products including a Hadoop distribution and products for both real-time as well as interactive and batch analysis. Pivotal now works with a simplified licensing model, including per-core pricing for all the products in the suite, which the company says will allow customers to store an unlimited amount of data in Hadoop.
- Amazon Elastic MapReduce – Amazon's hosted Hadoop solution. Also includes additional software from the big data ecosystem such as Hive and HBase.
- Amazon S3 - Amazon's storage solution for any amount of data. Data uploaded to S3 can be accessed by all applications deployed in the Amazon cloud.
- Redshift - Column database hosted by Amazon. Uses standard JDBC and ODBC drivers, which makes it easy to integrate with existing Business intelligence solutions.

Since the dominating technologies in the field are based on open source technologies and furthermore provide good integration with existing standards and packages they constitute a good alternative to base Big Data applications on. They have also been tried in some very large installations; Hadoop for example is used by Facebook to handle more than 100 PB of data.

# Case Study

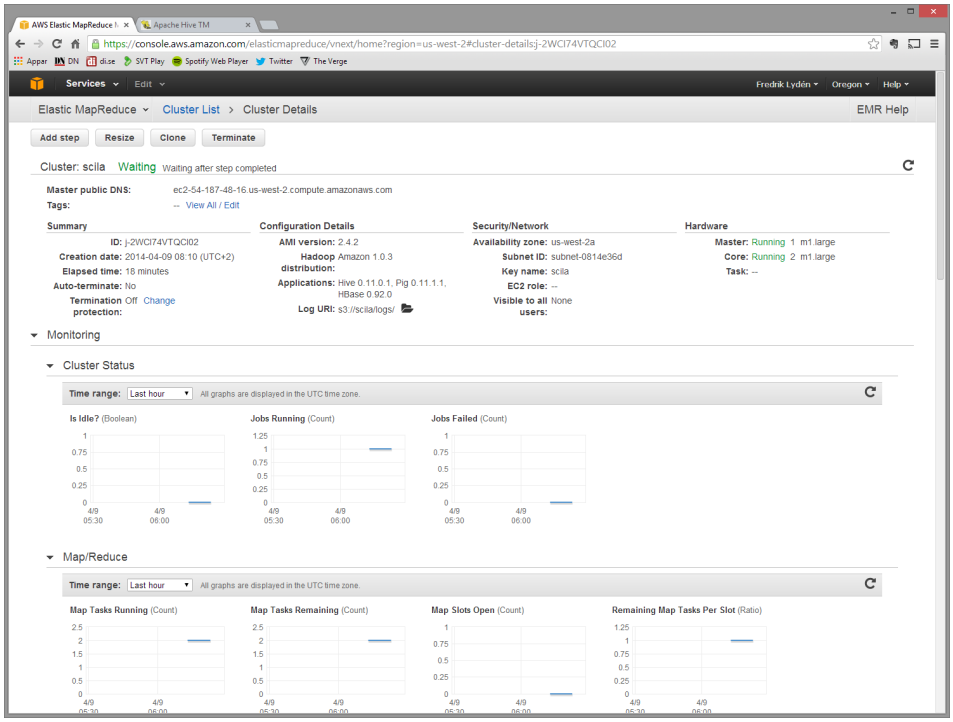
The transaction file format written by Scila is based on the industry standard JSON format. This makes it easy to integrate with other systems. In this case study we want to show how to easily integrate Scila with Hadoop and Hive. The query we used for our tests was to find out the trading participants that have traded more than a specified amount in an order book.

Hive has support for pluggable input format modules called SerDe's (Serializer / Deserializer). Since the Scila transaction files are JSON, we used a JSON SerDe.




*Using Scila Transaction Files with Amazon Elastic MapReduce.*

We tested this solution by deploying a Hadoop cluster using Amazon's cloud services. The Scila transaction files were compressed and uploaded to a S3 bucket. Using Hive we were able to query the transaction files we uploaded to the cluster. By adding more computing nodes to the cluster the query performance improved almost linearly.

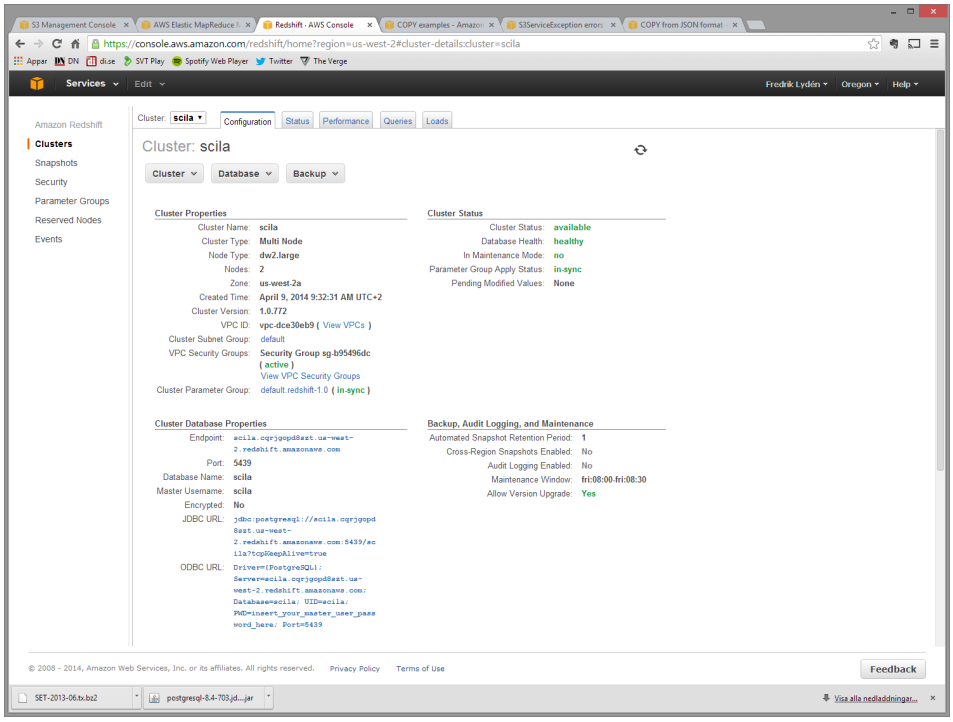



*The Administration page for Amazon's Elastic MapReduce service.*

Our investigation confirms that it is possible to use software from the Hadoop ecosystem to add powerful offline querying capabilities to the Scila surveillance system. This can be used for ad-hoc queries as well as for reporting and statistics.

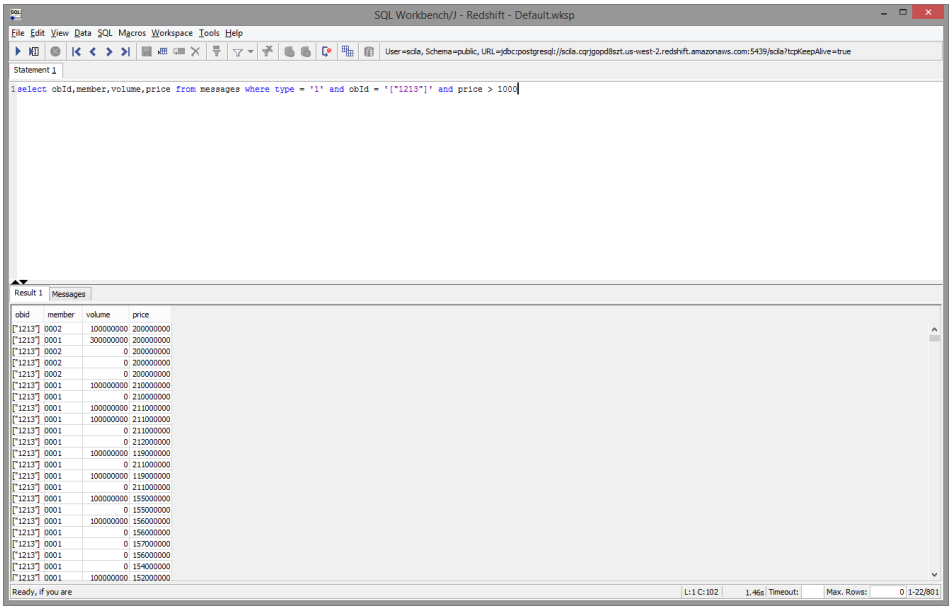
The drawback of using Hive is the performance. Queries on large data sets will take a long time (minutes or hours rather than seconds) to complete, so this is a solution that is more suited for batch-oriented queries. There are alternatives to Hive such as Shark, Apache Drill and Impala. All of these offer more or less the same functionality as Hive but are targeted for lower response times and may offer better performance in certain use cases.

For queries where instant responses are needed, there are also other alternatives such as Amazon Redshift.

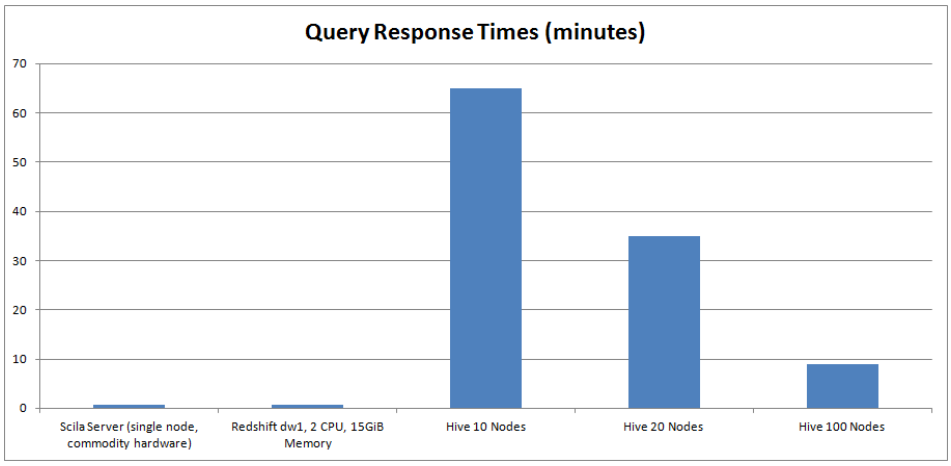



*The Amazon Redshift Administration Page.*

Redshift cannot read JSON data directly, but using a simple mapping file we were able to import transaction data from Scila into Redshift. Then we were able to query the database using a database client using standard SQL syntax. Since Redshift uses a standard JDBC / ODBC driver it is possible to use Redshift as a backend for all reporting engines and business intelligence solutions that supports external data sources.




*Querying Redshift with a standard SQL client.*




*Response time when running the test case query*

During the case study we also tested HP Vertica. Vertica is proprietary software, and is possible to deploy both on Amazon or inhouse. This may be a good alternative for companies that is not yet ready to copy their data to an external cloud service.

Just as Redshift, Vertica is a column database, with standard JDBC/ODBC capabilities. During the case study we successfully loaded Scila JSON data into a Vertica "Flex Table". Vertica automatically converts the JSON tags to columns during the import process, and it is then possible to query the data with standard SQL.

# Using the cloud

## Cost advantages

Setting up a data center/cluster for big data processing can be very expensive, and it may be more cost efficient to use a cloud provider. This may be even more true if the utilization degree of the cluster is uneven (for example if big queries/processing is done at the end of each month and fewer queries are done otherwise).
Scaling up a cloud cluster from 0 to 100 machines can be done in an instant, and when the computation is done the servers can be closed down and the cost for processing will go down to zero. One will probably keep the data in the cloud between processing runs though, so the cost of storage can not be avoided.
Storing 50TB of data in Amazon S3 costs around 1800$/month (March 2014).

## Security aspects

There are many concerns about the security of cloud providers and if your data is safe with them. But, as of this date, there doesn't seem to have been any big security breach at a cloud provider, where customer data has been compromised. There have been quite a few incidents where services has become unavailable, and that may be bad enough. But at least your data is safe (for now).
There are a few things that can be considered though, if you want to make your cloud experience as safe as possible:

- Private vs Public: If the cloud provider offers private servers, that is to prefer over public ones. These will give you more control over who can access the servers etc. Example: Amazon Virtual Private Cloud (VPC).

- Location: Even the most secure data center will most likely have to release data if there is a request from the local government. European companies may be under regulation that prevents them from storing data in another continent, and vice versa.
  The big cloud providers will let you choose the data center location. Usually there is at least one in USA, Europe and Asia. Or a more local cloud provider can be chosen, instead of one of the big international ones.
- Anonymize: Another way to add a bit of extra security is to anonymize the data before it is uploaded to the cloud. Then the data can be processed as usual, and when the result is downloaded to the corporate network the anonymous values can be translated back to the original ones.

### Compliance

Many cloud providers are ISO27001 certified, and at least Amazon and Rackspace are PCI DSS (handle cardholder information for debit and credit cards) compliant.
Amazon even says that they have customers that have built HIPAA compliant (medical data) system using their services. Amazon also has authorization from DoD to host government data.

## Conclusion

Using the Scila server for real time surveillance combined with a big data solution such as Hadoop or Redshift for data warehousing creates a powerful and versatile solution for market surveillance.

By using an external cloud provider such as Amazon it is possible to deploy hardware on the fly, making it painless to scale up or down as requirements change.

Security and redundancy in cloud based solutions have improved and maybe more importantly gained enough public credibility to constitute a serious alternative even for mission critical applications.

There are established Big Data technologies that have matured to the point where it can be safely assumed that they will be around and continuously developed for a foreseeable future, implying that it is worthwhile investing time in developing applications based on them.

# About Scila AB

Scila provides trading surveillance products built on many years of experience from both market surveillance and systems design. Scila Surveillance uses modern technology to give the customer a seamless route from detection of market abuse to presentable evidence. Scila delivers the future in modern market surveillance technology by offering trading venues, regulators and market participants the most competitive solution available.

For more information or details of our surveillance offerings, please contact Executive Chairman Lars-Ivar Sellberg at lars-ivar.sellberg@scila.se or +46 733 47 87 10.

*Company Address*

Scila AB
Kåkbrinken 11 A
111 27 Stockholm, SWEDEN
+46 8 546 402 90
www.scila.se