International Journal of Knowledge Based Computer Systems
Volume 05 Issue 01, 2017
ISSN.: 2321-5623

Framework to Process High Frequency Trading Using Complex Event Processing

A. Acharya[1], N. S. Sidnal[2]
[1]KLS GIT, Gogte Institute of Technology, Belagavi, Karnataka, India. Email: aacharya@git.edu
[2]KLE MSSCET, Belagavi, Karnataka, India. Email: sidnal.nandini@gmail.com

Abstract: The financial services industry had always been a data intensive industry. From insurance to capital markets the role of data has been pivotal for a lot of applications like financial modeling, portfolio optimization, asset/liability matching, fraud detection and risk modeling. The big data revolution has provided a lot of options for innovation and improved efficiency in this domain. At the same time, a new set of challenges has been thrown up which need to be overcome for future growth and sustainability in the financial services industry. In recent times the securities trading market has undergone dramatic changes resulting in the growth of high velocity data. Velocity being one of the Vs of Big data, presents a unique set of challenges to the capital markets. The tradition approach of using Business Intelligence (BI) is no longer scaling especially in terms of the velocity of data. During the previous decade most of the firms in the capital markets have made significant investments in their ability to collect, store, manage and analyze (to some extent) large amount of data. Based on the benefits offered by big data analytics, financial services firms are now able to provide highly personalized and real time location based services rather than only product-based services which was possible earlier. The rise of electronic trading and the availability of real time stock prices and real time currency trading make it necessary to have real time risk analysis. Market participants who have the ability to analyze the data in real time will be able to garner a disproportionate part of the available profit pool. The availability of huge amounts of financial data, high rate of data generation, and the heterogeneity of financial data make it difficult to capture, process and perform timely analysis of data. Traditional financial systems are not designed to cope with a wide variety of data, especially unstructured data from Twitter, news, social media, blogs etc which affect market dynamics in real time. Traditional data warehousing and BI techniques like extract, transform and load (ETL) take a huge amount of time (often days) to process the large amounts of data and are thus not receptive to real time analytics.

This paper discusses the implication of the rise of big data and especially that of high velocity data in the domain of High Frequency Trading (HFT), a growing niche of securities trading. We first take a brief look at the intricacies of HFT including some of the commonly used strategies used by HFT traders. The technological challenges in processing HFT and responding to the real time changes in the market conditions are also discussed. Some of the potential technological solutions to solve the issues thrown up by HFT are analyzed for their effectiveness to address the real time performance requirements of HFT. We identify Complex Event Processing (CEP) as a candidate to address the HFT problem. The paper is divided into 3 parts; part A deals with understanding HFT and the challenges that it poses to the technological processing. In Part B we look at Complex Event Processing (CEP) and the types of problems it can be applied to. In Part C we show a framework to process HFT using techniques derived from CEP.

Keywords: High frequency trading, Complex event processing, Big data processing.

I. High Frequency Trading

The knowledge economy is in the process of continuous automation. The tasks which were considered specialized and were done by highly skilled people are now increasingly being done by intelligent systems. Dealing with big data and its associated analytics is no longer a niche area; its applications and impact are already being felt in the broader economy. In this paper we take a look at High Frequency Trading (HFT) which has become an important part of the knowledge economy. HFT illustrates a very important V of big data i.e. Velocity. The paper begins with a brief discussion on big data and the challenges in its processing and analytics. This discussion will focus on the aspect of velocity of big data processing.

There has been a trend towards greater transparency and efficiency in stock markets. Over the past few years the trading of financial securities has been significantly impacted by the advent of technology. It started with the replacing of human intermediaries on the stock market floor by electronic trading using limit orders. This led to the development of algorithms to mimic the behavior of human traders. These algorithms have become increasingly sophisticated to take advantage of the improved infrastructure especially at the exchanges. Regulatory changes have driven greater competition among the market participants. The availability of high performance computing and high speed networks in conjunction with the regulatory changes and greater competition has lead to a new paradigm of trading called High Frequency Trading (HFT).

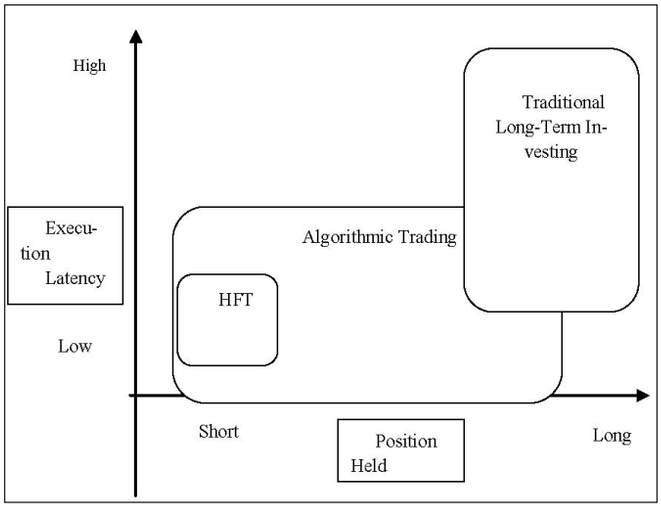

Fig. 1: Latency Vs Position Held

There is no clear distinction between the terms "algorithmic trading" and "high frequency trading" in popular literature. Therefore it is important to differenti-ate between these terms. [1] provides a good discussion on the differences be-tween these terms. The differences are clearly brought about in Fig. 1. Algo-rithmic trading refers to the ability to place and execute orders electronically using computer algorithms as opposed to non-electronic modes like telephone, mail, or in person. Fig. 1 shows the relationship between algorithmic trading, traditional long term investment and HFT in terms of latency and the time for which the positions are held. As depicted, HFT is a subset of algorithmic trad-ing where the positions are held for a small time (typically for a few seconds or even lesser) and the latency of trading is very low with the help of high speed networks and fast computing. On the other hand the traditional long term in-vesting involves holding period of up to several years. For long term investors the speed of execution of orders does not have high priority. HFT is an impre-cise term which has no legal or regulatory definition. The U.S. Securities and Exchange Commission (SEC) which oversees the US capital markets came up with the following characteristics which apply to HFT trading [2]:

1. The use of very fast and sophisticated algorithms for creating, routing and executing orders.
2. The use of data feeds to minimize network and other latencies as well as co-location services offered by the exchanges. Co-located servers are machines which are placed in rented racks which are in the vicinity of the exchange server. This is done with an aim to reduce the network latency.
3. Very short time frames for establishing and liquidating positions are maintained. This results in frequent turnover in the financial instrument being traded.
4. Numerous orders are submitted which are canceled within a short dura-tion. Typically the open orders are canceled within milliseconds [3].
5. The trading day is closed as flat as possible, i.e. unhedged positions are typically not carried overnight.

These characteristics are typical of HFT trading firms but all HFT trading need not exhibit all of these characteristics. HFT volumes have grown substantially over the past few years and therefore have had a major impact on capital markets. HFT trading accounts for about 55% of the US equity market volume and about 40% of the European equity markets. In terms of absolute numbers, it amounts to over 5 billion shares worth daily volumes in the US markets alone [4]. Almost all HFT traders try to make a small profit on a large amount of trades. The typical profit per trade is so small that it would not be worthwhile for regular market participants, but since HFT traders deal with a huge amount of trades, the small profits per trade add up to a good amount. Some strategies may not make profits on all trades but on 51% of trades [5], but the trading volumes may still make it a overall profitable HFT strategy. HFT traders use a variety of strategies to keep ahead of the markets. It is possible to use a combination of strategies to maximize the profits for a particular situation. The strategies can be broadly classified into 2 categories: passive strategies and aggressive strategies [6]. Passive strategies include statistical arbitrage trading and market making. Aggressive strategies include momentum ignition and order anticipation.

Statistical arbitrage trading [7] involves benefiting from the price difference for the same security or related securities. The difference may be in the prices of the same security trading at two or more different exchanges or locations. For example a stock may trade at $50.50 at NYSE and at $50.75 at NASDAQ. So a simple strategy could be to simultaneously put a sell order at NASDQL and a buy order at NYSE and hope to make $0.25. This strategy requires very quick (almost simultaneous) access to both NYSE and NASDAQ. Since there will be many players trying the same strategy the fastest one tends to profit the most and the slower players may find that the market has moved against them resulting in losses. There may be difference in the price of individual securities and in the constituents in an ETF basket. If the S&P 500 based index moves higher, but the underlying stock does not for some reason, one can simultaneously sell the index and buy the stock to profit from the mispricing.

Market making involves posting simultaneous buy and sell orders on the same se-curity with a view to provide liquidity to the other market participants. Market makers sell at the ask price and buy at the bid price, hoping to make the bid-ask spread. The profits tend to increase at times of market volatility resulting in larger spreads. In a lot of markets the market makers also earn liquidity rebates or maker fees [8]. HFT market makers need to adjust their bid as per the price movements and therefore tend to submit a large number of orders and cancel them shortly. It is found that HFT market making reduces the spreads overall resulting in better pricing for the other market participants [9].

Order anticipation, which is also known as liquidity detection is a strategy which is aimed at detecting large open orders

(typically from institutional investors). The strategy involves putting several small orders and checking if they get filled quickly. If they do then it could be concluded that there is a large open order sitting which can allow the HFT player to trade ahead of the detected li-quidity. The consequence of liquidity detection is that the HFT players will profit at the cost of institutional investors who tend to have large orders.

Momentum ignition is a strategy where a HFT player initiates several orders with the aim of causing rapid movements in the price of the securities. The intention of triggering the price movement is to induce other traders (including other al-gorithmic and HFT traders) to participate in the trading of the security causing a buildup of momentum in a particular direction (up or down). After the price has moved sufficiently, the initiator of the momentum can liquidate previously held open position at higher prices. There has been a debate in recent times whether such kind of behavior is unethical or even illegal [10].

These are not the only HFT strategies used; other could be news trading, latency arbitrage etc. Traders are constantly trying different techniques to stay ahead in the HFT game.

As seen in the previous discussion on HFT, the primary challenge is that of han-dling the "velocity" of big data. In summary the main challenges for HFT processing are:

1. Real Time Decision Making
   - The ability to calculate risks and prices and positions at portfolio scale in near real time
2. High Performance Computing
   - The ability to evaluate all the available data from different sources in real time
   - Execution of trading strategies in real time
3. Message Latency
   - Low latency networking
   - Reducing the time between decision making and execution
   - Proximity

Without the ability to detect, analyze and respond in real time, a HFT trader will not have a good chance of surviving in the market. Traditionally, the financial organizations attempted to achieve low latency by utilizing high performance computing infrastructure, especially capable in floating point processing. The drawback of this approach is that of scaling. It is not easy to add more high performance computing nodes as it is to add commodity hardware. Additionally the performance of the storage system will also become a bottleneck when there is storage and transfer of large amounts of data, as is that case with HFT. The storage problem is typically solved using a distributed file system using several nodes which would provide both load balancing and fault tolerance. This architecture also provides the advantage of a high aggregate I/O bandwidth. A typical implementation of such an architecture is the Hadoop Distributed File System (HDFS) [11]. In recent times NoSQL based systems are proving to be popular to store and manage heterogeneous data. Examples of NoSQL databases includes MongoDB and Cassandra [12]. Along with this distributed file system, a new programming model called MapReduce is becoming widespread [13]. MapReduce essentially moves the computing to the location of the data rather than the other way around. MapReduce can process a large amount to data in parallel; however it fails to provide the solution for low latency tasks as it is a batch processing system. Other approaches using in-memory speed ups like that used in Spark [14] improve the performance but do not hold up for real time analytics application like HFT. Some firms are using accelerated CPU and GPUs to speed up computing in conjunction with network technologies like InfiniBand [15] to improve the throughput. Another significant issue is that of tail latency [16]. A computational job with big data is usually split into multiple stages with each stage being pipelined to execute on each node. A slow performing node will block further processing and this cascading will lengthen the tail of the latency distribution.

Due to the lack of well-defined solutions to these problems, organizations are using incremental, exploratory to devise customized solutions for their big data efficiency problems. We shall explore Complex Event Processing (CEP) which promises to be a general purpose solution to the real time analytics problem faced by HFT.

II. Complex Event Processing

A lot of applications require the processing of flowing data from different sources and at different rates to obtain responses to complex queries. Examples of such applications are real time traffic management, wireless sensor networks, click stream analysis, equity trading etc. The cornerstone for the technological success in a low latency environment is the ability to clean, preprocess and analyze the correlated events in real time. Traditional DBMS based approaches will not work for such applications as the DBMS approach requires that the data be first persisted and indexed before any processing. Any process of data is typically user driven and is not related to the arrival of the data. Therefore it is necessary to consider data as a *flow* and process the flow of data using a set of predefined processing rules. In a traditional DBMS setup, the processing happens on stored data while updates to the data are relatively infrequent. The query is run just once to return a complete answer. On the other hand in stream processing, standing queries are run which are executed continuously. As new data arrives the results of the standing queries are updated. The major drawback of generic stream processing systems is that they leave the responsibility of associating the semantics with the data to the clients. Fig. 2 shows a high level schematic representation of a typical stream processing system.

Instead of looking at the incoming information as merely a flow of data, we can also view it as a flow of notifications of events. Historically there have been different models of event processing, the most important of which are publish-subscribe

mechanism and topic based systems. In the "publish-subscribe" mechanism, users interested in a particular type of events would register to receive those events. The event producers will not be aware of the registered subscribers. The topic based systems allow the subscribers to subscribe to specific topics; the publisher will categorize the events into topics. This was used in BPM (Business process monitoring) systems and workflow management systems. This system is based on the human workflow, i.e. process one event at a time. The major drawback of this system is that it does not exploit all the events all the time. Fig. 3 shows a typical representation of an event-processing architecture.

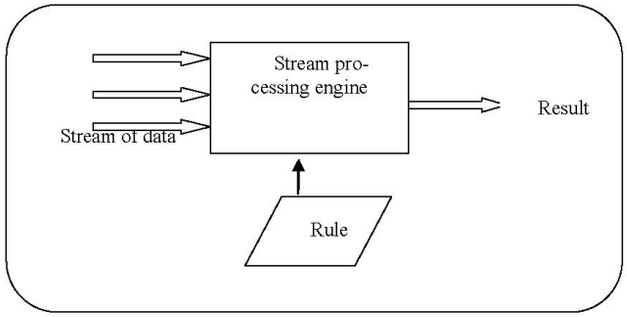

Fig. 2: Typical Stream Processing

Complex event processing systems associate semantics to the information items being processed: the notifications of events observed by sources [17]. These events have to be filtered and combined to understand the information in terms of higher level events. In complex event processing, the aim is to detect patterns of high level events from the incoming stream of low level events. By definition, an event is "anything that happens or is contemplated as happening" [18]. Examples of events can be a key stroke, an earthquake, a financial trade etc. An event may signify a problem, an opportunity, a deviation or a threshold or something else depending on the domain.

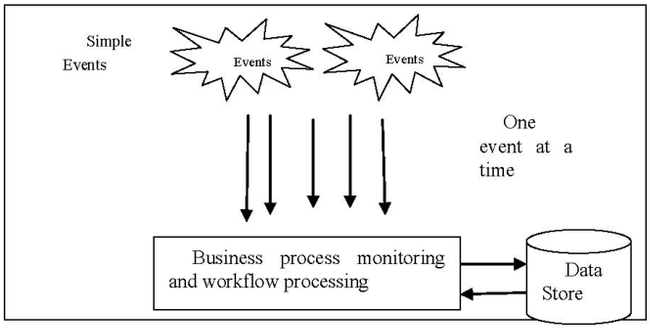

Fig. 3: Conventional Event Processing

Consider the following events: church bells are ringing, the appearance of a man in a suit, a woman in a flowing gown and people throwing confetti. A complex event can be derived from these simple events i.e. a wedding is taking place. The information that a wedding is taking place is not contained in the individual events, but the combination of the events into a complex event enables us to understand the semantics of the complex event.

A major drawback of stream event processing is that it is difficult to detect event pattern across multiple event streams. The complexity increases when there are multiple event types in addition to temporal ordering among the events. This is the area where Complex event processing has a distinct advantage over other techniques. Using computing power to correlate across large amounts of events at high rates enables CEP to identify patterns that are otherwise not apparent. CEP helps to provide solutions by utilizing memory and data grids for analyzing events, trends and patterns in real time and decision making in a matter of milliseconds. This has led CEP to be a matter of choice for typical BAM (Business Activity Monitoring). Fig. 4 shows the different types of delays or latencies that occur during the processing of a typical event. There is a data latency which is the elapsed time between the occurrence of the event and the time it is captured by the system. The decision latency is the time between the capture of the data by the system and the time where a decision is taken on how to respond to the event. The action latency is the time elapsed between the decision making and the actual action taken based on the decision. CEP is designed to minimize the decision latency and the action latency.

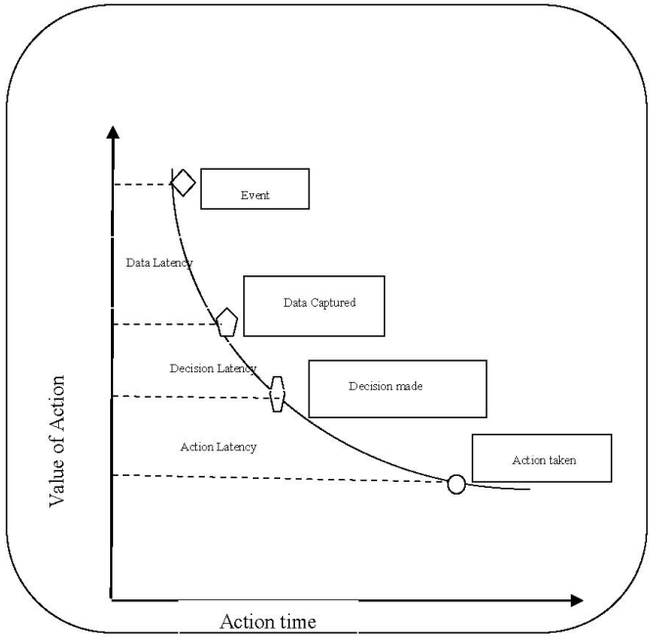

Fig. 4 . Latency Graph

In CEP terminology an "event" is an object that records a piece of activity in a system. An event has three primary features: Form, Significance and Relativity.

Form: The form of an event is the attribute or the set of attributes and data components of the event.

Significance: The significance of the activity to the system.

Relativity: Events are related to other events by time, causality and aggregation.

Event Relationships: The most common relationships between events are time, causality and aggregation.

Time: This relationship allows temporal ordering of events. For example event A happened after event B.

Causality: This defines the dependence of events in the system. For example event B was caused due to event A.

Aggregation: This allows abstraction of relationships. For example if event B signifies the activity comprising of underlying activities A1, A2, A3, then event B is said to aggregate A* events.

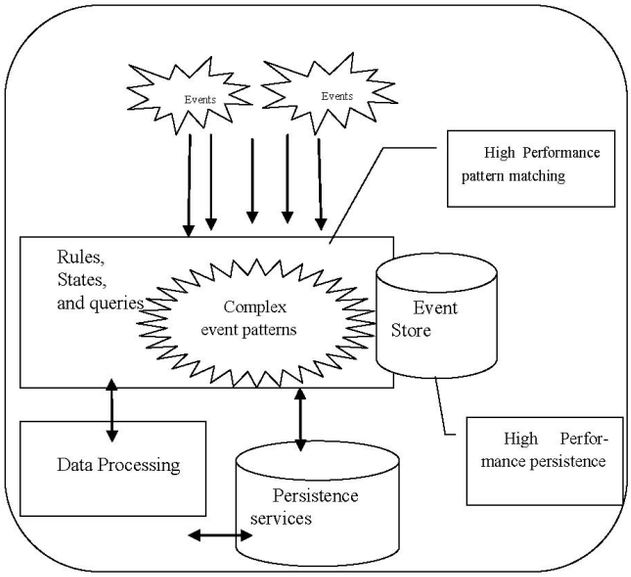

Fig. 5: Complex Event Processing

Fig. 5 provides a schematic high level representation of a CEP processing architecture. Events making up the complex events can come in from different event channels. The key to CEP is the detection of complex event patterns *before* the events get stored to the persistent database. This step enables the real time performance of CEP. A high performance pattern matching engine could comprise of rules, states, queries or any combination of these. Typically a high performance event store using in-memory database is used. A data processing component looks at the complex events and decides to take any further action based on rules defined in a database. The database may also be updated with the results of the pattern matching engine for it to be readily available in future.

A CEP engine is the heart of the CEP system which collects and processes the events and detects the complex events. There are several flavors of CEP engines, the most important of them are state oriented, rule oriented and query oriented CEP engines. Fig. 6 shows a representation of a state oriented CEP engine. Here the data and events are modeled in the form of a state transition machine and the system transitions from one state to another based on the events which are received.

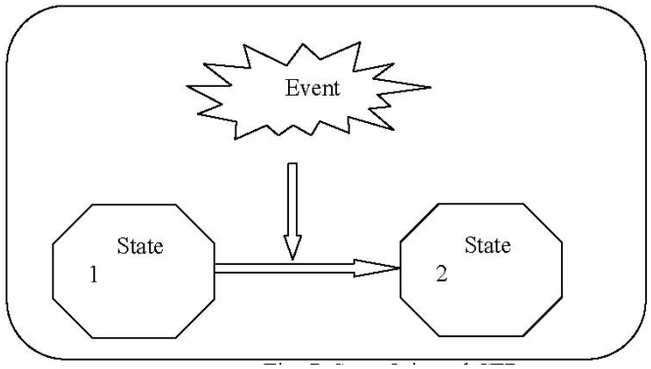

Fig. 6: State Oriented CEP

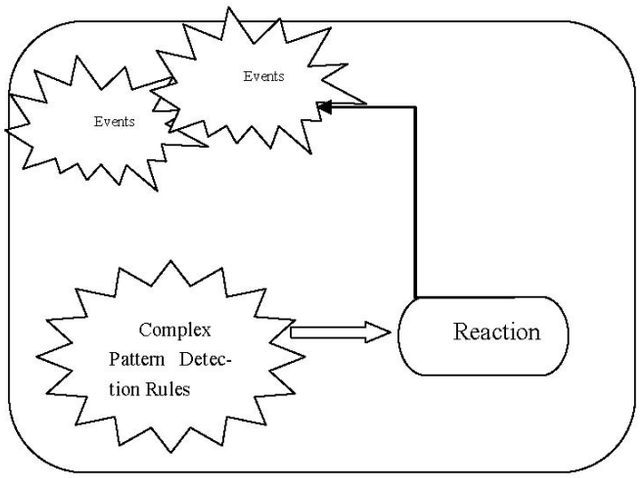

Fig. 7: Rule Oriented CEP

Fig. 7 represents a rule oriented CEP engine. Such an engine utilizes a set of rules to detect complex events and patterns using the complex events. The CEP rule oriented engine enables correlation and aggregation of events over a time window and pattern detection involving multiple events. As shown in the diagram the result of the rule oriented detection of complex events may lead to generation of further events. CEP can express rules that cannot be defined intuitively in other paradigms. Rules could be to detect event logic patterns like events arriving in a certain sequence, or even absence of events. There could be rules with built in temporal awareness like detecting events accruing within a certain moving time frame.

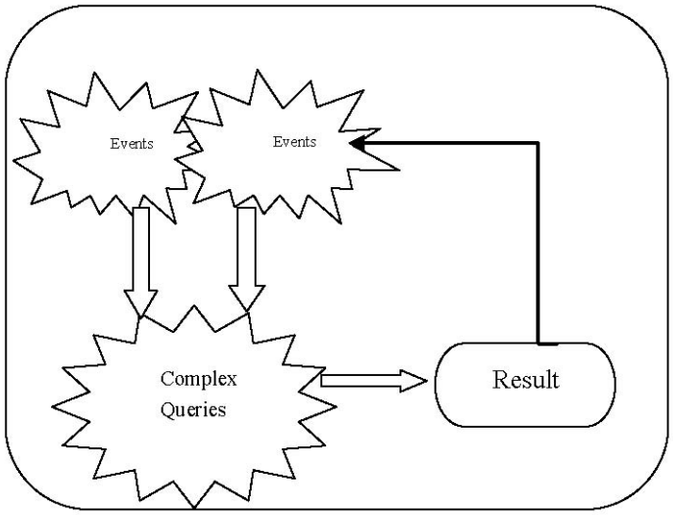

Fig. 8: Query Oriented CEP

Fig. 8 shows a representation of the query based CEP engine. Unlike a RDBMS which store data and queries are run through it, here the queries are stored and data is run through it. It runs in a continuous execution mode than in an ad hoc query mode. The query language is SQL based and typically has an in-memory database to improve performance. Apart for regular operators like joining and counting and logical operators like "ad", "or", "not", the query language will have temporal operators like "within T(Z)" (X and Y within T(20 sec) and "between". In addition it may also have sequence operators "->" (A->B event B follows event A).

III. Framework for High Frequency Trading Using CEP

In this section we discuss a high level architecture for effective processing of HFT. Any HFT processing system should be able to support the following functions.

- Receive incoming market quotes.
- Receive and evaluate news feeds and social media feeds.
- Perform correlation and other econometric analysis.
- Identify and evaluate patterns which could be exploited via HFT.
- Design an algorithm for the opportunity identified.
- Initiate trading signals based on the algorithm.
- Dynamically manage the portfolio based on the risk and market conditions.

We saw the applicability of CEP for real time pattern detection and response to the detected pattern. However a CEP based HFT framework is not sufficient to achieve the speeds required to come on top of other HFT players. To do this we need to focus on every aspect of the processing elements to minimize the latency. As summarized in part A, the main technological challenges in processing HFT are real time decision making and handling message latency.

Message and network latency: The issue of message latency can be handled by utilizing the best of breed network infrastructure. We provide a quick summary of the hardware requirements needed to manage the message and network latency. The section is necessarily short as it only deals with putting together off the shelf hardware components. The HFT infrastructure needs to have the capability to perform super fast message handling, and a low latency network. A common technique used is to make use of collocation services provided by the exchanges to minimize the network round trip between the exchange and the HFT servers. Direct Market Access (DMA) is required to eliminate the latency in the broker's systems. DMA is a electronic facility that allows traders to directly interact with the order book of an exchange [19]. When it comes to network interconnect speed, Infiniband is the interconnect of choice. InfiniBand is a networking and communications standard that features very high throughput and very low latency [15]. It supports a latency of 2 nanoseconds with a very high throughput rate of over 100 GB/s. It is also imperative to use hardware accelerators for market data feed handling, market data line handling, and order access/execution [20]. Such accelerators are based on FPGA and can handle up to 10 million messages per second with a latency of only 15 nanoseconds.

We will focus on the architecture for supporting real time decision making for HFT. As we alluded to in part A, CEP will provide a good platform to support real time event processing. The first thing to consider is the different data feeds into the HFT system. There would be several market feeds from different exchanges. Almost all the exchange feeds would be in the format of (Financial Information eXchange) FIX protocol which is the worldwide standard for international real-time exchange of information related to the securities transactions and markets [21]. In addition there should be a feed from the news agencies like Reuters or Bloomberg to stay up to date with the economic and political happenings around the world and respond appropriately. It would also be beneficial to be also to tap into social networking streams like twitter to gauge the reaction to major news events. Big data analytical approaches like sentiment analysis using machine learning is needed to sort through the vast data coming via the feeds [22]. Finally, feeds from currency, commodity and bond markets may also be needed if HFT algorithms like statistical arbitrage across markets are used. Fig. 9 shows a representation of the incoming and outgoing flows in a HFT system.

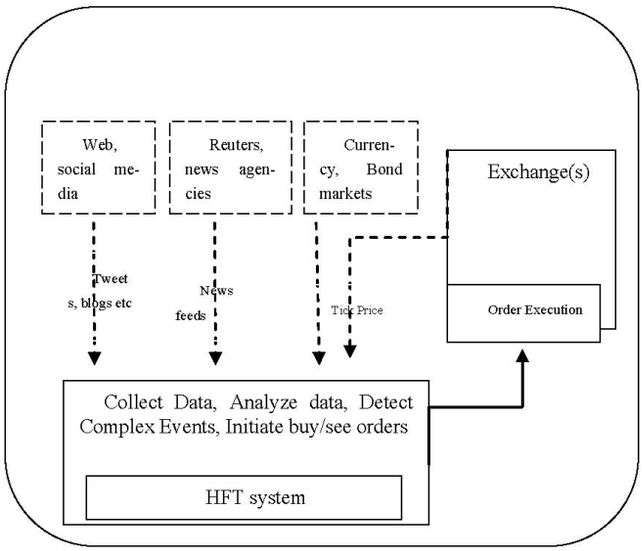

Fig. 9: Data Feeds

Fig. 10 presents a conceptual framework for CEP based HFT processing. The major components are the core CEP engine, a event bus or channel, a distributed event/data cache, a real time decision service and a persistent store.

The event bus or channel is the backbone of the system through which the events and messages are transferred between different components at high speeds. The channel can carry multiple event types. The events carried by a single channel can be consumed by multiple consumers (fan out), or events from multiple channels could go to a single consumer (fan in). A channel is basically a sort of a queue which has an associated thread pool. This allows the upstream and downstream components to operate asynchronously. Channels are useful in increasing the concurrency especially when in incoming data is in a single feed.

The first component of the CEP engine is the Event preprocessing subsystem. Event preprocessing is the process of preparing incoming events and metadata for further stages of complex event processing. It will involve the separation and discarding of unused event data, and the reformatting of the events for downstream event processing. The Event pre processor performs the following functions: Event Identification, Selection, Filtering, Monitoring and Enrichment. Event Identification involves recognizing the events from the raw incoming data. It also involves identifying the event as belonging to a particular event type. Event selection means identifying particular events to be used for further analysis and pattern matching.

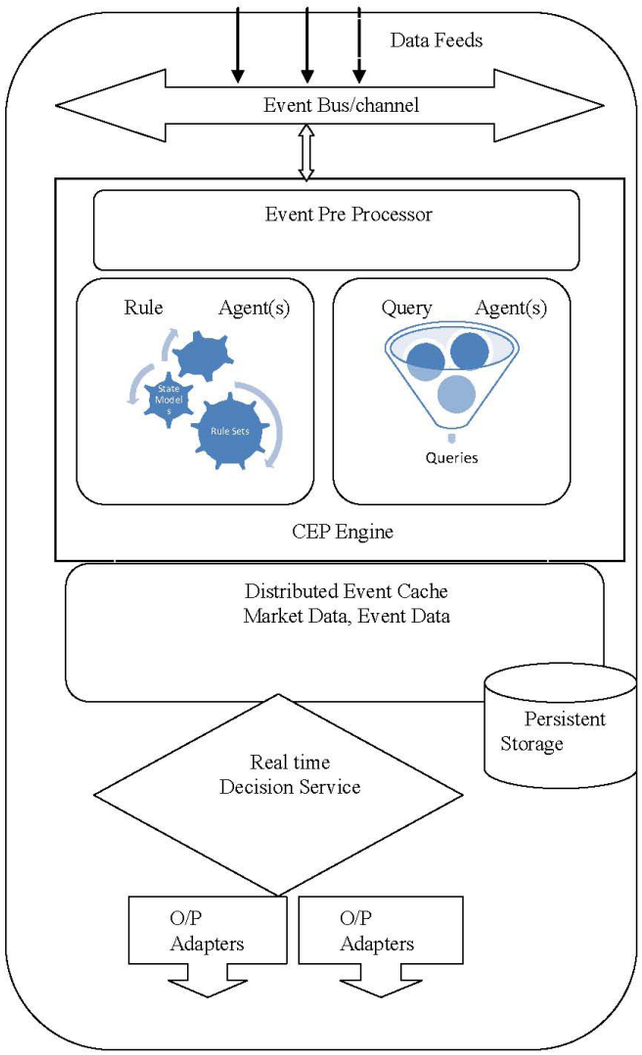

Fig. 10: CEP Engine for HFT

The selected events are then filtered based on some property of the events. Monitoring involves observing particular event channels to identify particular events of interest. The final step in preprocessing is enrichment where the event is augmented with additional information based on prior events.

There can be one or more rule agents at the core of the CEP engine. The main task of the rule engine is to detect situations or conditions based on a combination of events. The rules could be a combination of traditional business rules and inference rules. A rule engine typically has several phases of execution. The first stage is signaling which deals with the detection of an event. The next stage is triggering where the association of an event with the set of rules defined for it is done. After triggering there comes the evaluation phase in which the conditional

part for each triggered rule is evaluated. The next phase is scheduling in which we define an execution order between selected rules. In the final phase called execution, the execution of all the actions associated to selected rules is done. The rule engine will provide an expressive rule language to provide the rules. Complex and intricate rules including temporal and contextual awareness could be easily defines with such a rule language. "The event processing rules may be prescribed in many different ways, including graphical methods, Java code or SQL code. The rules could be also defined by finite state machines, UML diagrams, ECA (event-condition-action) rules or reactive rules that are triggered by event patterns" [18]. The following is an example of a possible rule: "If NASDAQ falls more than 0.21% over its 10 minute moving average && IBM falls less than 0.15% over its 10 minute moving average && ORCL has a positively increasing moving average with delta<0.25 then buy ORCL with a 2% stop loss."

The query agent is the component which enables the CEP to filter, query, and perform pattern matching operations on streams of data using a declarative, SQL-like language. The query language supports filtering, aggregation, pattern matching, and joining of event streams and other data sources. The output of the queries is sent to any downstream listeners. Fig. 11 shows a query defined in Continuous Query Language (CQL), used in Oracle CEP (Oracle 2011).

```
<query id="perc" ordering-constraint="PARTITION_ORDERED" partition-expression="symbol">

<![CDATA[ select symbol, lastPrice, percLastPrice from S MATCH_RECOGNIZE (

PARTITION BY symbol

MEASURES

B.symbol as symbol,

B.lastPrice as lastPrice,

100*(B.lastPrice - A.lastPrice)/A.lastPrice

as percLastPrice

ALL MATCHES PATTERN (A B)

DEFINE B AS

(100*(B.lastPrice - A.lastPrice)/A.lastPrice > 2.0

or 100*(B.lastPrice - A.lastPrice)/A.lastPrice < -2.0)

) as T

]]>

</query>
```

Fig. 11: Sample CQL Query

There are two main stages to the processing done by the query agent. In the first stage, the incoming flow of market data is joined against the symbol watch and an output event is generated for each input event that matches a symbol on the watch list. The output from this initial filtering stage is fed into a subsequent pattern matching stage implemented by the query with id "perc". The pattern matching query produces an output event whenever it detects that the price of a given stock has increased or decreased by more than 2 percent from its immediately previous price. The output from the pattern match query is sent to any downstream listeners which compute aggregate statistics and latency data for the benchmark based on the output events it receives. The query language will support the concept of windows which defines the portions of input flow to be considered during the execution of the operators. Windows are either logical (based on time) or physical (based on count). In the case of logical windows, the bounds are defined as a function of time. For example compute an operation only on the elements that arrived during the last 'n' minutes. In the case of physical windows, the bounds depend on the number of items included in the window. For example limit the scope of an operator to the last 'n' elements.

An important component of the HFT processing architecture is the distributed event and data cache. The cache is used for distributed data access. The data may be written to the cache and written back to the database asynchronously. The cache sits between the processing nodes (rule agent and query agent) and the persistent storage services. As the cache is distributed, one needs to take care to maintain the coherence of data in the cache. An Observer pattern could be implemented to take advantage of the cache. A good example of the data to be cached is the market data. As the market data values from the data feeds come in, it could trigger a subscription and update the cached values. Apart from market data, even event windows and detected complex events could be cached and the cache could be updated as the window slides over. Thus the availability of the cache component will directly support the continuous queries run by the query agent. So having the distributed cache will support scalability as multiple CEP engines could be run I parallel fed by the distributed cache of market data and events. The continuous aggregation will allow the computation of real time risk and pricing.

The final component of the HFT architecture is the Real time Decision service. The purpose of the service is to take real time decisions based on the patterns detected by the CEP engine and respond in a manner which optimizes profitability and minimizes risk. To achieve its objectives, the real time decision service has two major sub-components: a learning service and a decision service. The learning service automatically learns from each detected complex events and discovers important correlations. The learning can then be used to make predictions. The prediction can be based on machine learning techniques, either supervised learning using a training set (of historical data) or unsupervised learning based on the data stream. The predictors used in the prediction will be the key performance indicators of the complex events. The learning component contains a ML part which works on the continuously streaming training set. A large set of predictors is necessary for building the internal machine learning model. The prediction can be

made in the form of an event and the predicted event can be used to define further complex events. Other learning models based on regression and SVM may also be considered. An important consideration to be made is about the performance of the learning algorithm and the number of predictors used. Most HFT algorithms necessarily need to be simple with minimal number of predictors to meet the challenge of extreme performance. The decision service combines rules and automated predictive models to define contextual and optimal decision logic. The decision logic needs to be highly scalable and self-adjusting and agile to the market volatility. The decision service should be capable of rule modification, measurement and analysis in real time. An important aspect of this component is routing, in which based on the event type and data, the appropriate output adapter is called for the execution. For example if an arbitrage opportunity is detected where a security is trading lower at location A compared to location B, a buy order should be sent to location A and a sell order needs to be sent to location B.

IV. Conclusion

HFT processing poses several technological challenges. There are currently being managed by expensive and high end computing infrastructure leading to a quasi monopoly by the large broker-dealers. In this paper we saw some of the challenges of HFT processing and considered a few alternatives towards the solution. In part B we have provided a detailed picture of Complex Event Processing, which is usually used for business activity monitoring and similar applications. In part C we have provided a conceptual architecture using CEP elements and other components like a distributed cache and a real time decision service to address the HFT processing challenges. While this approach still calls for high end networking infrastructure and collocation, the processing elements could be constructed out of (relatively) commodity hardware.

References

[1] I. Aldridge, *High Frequency Trading,* John Wiley & Sons, Inc. 2010.

[2] SEC, "Concept release on equity market structure," no. 34, 2010.

[3] M. Chlistalla, "High frequency trading, better than its reputation?," *Dtsch. Bank Res.*, vol. 8, no. 3, pp. 217-224, 2011.

[4] M. Durbin, *All About High-Frequency Trading,* McGraw Hill Professional, 2010.

[5] C. M. Jones, "What do we know about high-frequency trading?," *Columbia Bus. Sch. Res. Pap. No. 13-11*, pp. 1-56, 2013.

[6] R. S. Miller, and G. Shorter, "High frequency trading Overview of recent developments," *Washingt. Congr. Res. Serv.*, 2016.

[7] M. Avellaneda, and J.-H. Lee, "Statistical arbitrage in the US equities market," *Quant. Financ.*, vol. 10, no. 7, pp. 761-782, 2010.

[8] J. A. Brogaard, T. Hendershott, and R. Riordan, "High-frequency trading and price discovery," *Rev. Financ. Stud.*, vol. 27, no. 8, pp. 2267-2306, 2014.

[9] J. A. Brogaard, "High frequency trading and its impact on market quality," *Management*, p. 66, 2010.

[10] SEC, "Equity market structure literature review part II: High frequency trading," *U.S. Secur. Exch. Comm. Staff Div. Trading Mark.*, vol. 2014, no. March, pp. 1-37, 2014.

[11] A. Hadoop, "Hadoop," 2009-03-06. Available: http//hadoop. apache. org, 2009.

[12] V. Abramova, and J. Bernardino, "NoSQL databases: MongoDB vs cassandra," *Proc. Int. Conf. Comput. Sci. Softw. Eng. ACM 2013*, pp. 14-22, 2013.

[13] J. Dean, and S. Ghemawat, "MapReduce," *Communications of the ACM*, vol. 53, no. 1. p. 72, 2010.

[14] A. Spark, "Apache Spark™ - Lightning-fast cluster computing," *Spark.Apache.Org*, 2015. .

[15] G. F. Pfister, "An introduction to the infiniband architecture," *High Perform. Mass Storage Parallel {I/O} Technol. Appl.*, no. 42, pp. 617-632, 2001.

[16] J. Dean, and L. A. Barroso, "The tail at scale," *Commun. ACM*, vol. 56, no. 2, p. 74, 2013.

[17] G. Cugola, and A. Margara, "Processing flows of information," *ACM Comput. Surv.*, vol. 44, no. 3, pp. 1-62, 2012.

[18] D. Luckham, and R. Schulte, "Event processing glossary - Version 1.1," *Processing*, no. July, pp. 1-19, 2008.

[19] M. Fugazza, and A. Nicita, "The direct and relative effects of preferential market access," *J. Int. Econ.*, vol. 89, no. 2, pp. 357-368, 2013.

[20] P. Grun, "Introduction to InfiniBand ™ for end users," 2010.

[21] S. Swarnkar, and J. Jenq, "Implementation of FIX engine and order management systems using ASP . NET C#,".

[22] S. Kumar, F. Morstatter, and H. Liu, *Twitter Data Analytics,* Springer, p. 89, 2013.