“Aggregation of an FX order book based on complex event processing”

AUTHORS	Barret Shao Greg Frank

ARTICLE INFO	Barret Shao and Greg Frank (2012). Aggregation of an FX order book based on complex event processing. *Investment Management and Financial Innovations*, *9*(1)
RELEASED ON	Friday, 30 March 2012
JOURNAL	"Investment Management and Financial Innovations"
FOUNDER	LLC “Consulting Publishing Company “Business Perspectives”

NUMBER OF REFERENCES	NUMBER OF FIGURES	NUMBER OF TABLES
0	0	0

Barret Pengyuan Shao (USA), Greg Frank (USA)

Aggregation of an FX order book based on complex event processing

Abstract

Aggregating liquidity across diverse trading venues into a single consolidated order book is important for financial institutions that trade foreign exchange. But doing so poses several challenges, including stable latency performance under spurious bursts in message rate. Complex event processing offers an approach to this problem that yields performance and maintenance advantages over thread-based approaches.

Keywords: aggregate order book, foreign exchange rate, complex event processing.
JEL Classification: C61, F31.

Introduction

A centralized electronic exchange has a certain book to record all quoted bid and ask prices and sizes. Without considering the commission fee required by brokers, the difference between optimal bid and ask price, or spread, is a major part of the transaction cost for taking market orders. Reducing the spread leads to lower transaction cost. Unlike a centralized electronic exchange, various trading venues and brokers erase the uniqueness of the order book in a centralized electronic exchange. The interest of aggregating an order book comes from the need to lower transaction cost, which is very important for high frequency trading.

1. Aggregated order book for FX

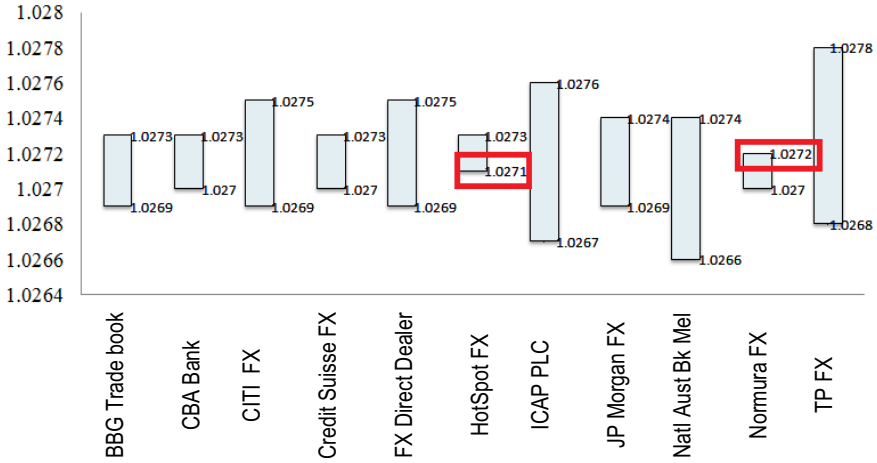
1.1. Properties of FX trading. Foreign exchange trading in the interbank market is quite different from trading exchange-traded instruments such as equities or futures. Instead of one central electronic exchange, many different trading venues exist for the same currency, such as Reuters, Hotspot, Currenex, and single-bank e-commerce platforms such as BARX. Financial institutions are becoming increasingly interested in FX trading and volumes are increasing in a market that is already the largest in the world, with an estimated volume of over $3 trillion per day just in spot FX.

1.1.1. Complication of trading environment. The FX trading environment is much more complicated than trading on centralized electronic exchanges such as CME or NASDAQ. In addition to electronic clearing networks that offer a standard limit order book, there are single bank platforms that offer trading. Oftentimes, each trading venue provides a unique quote stream for investors with different spread and skew characteristics depending on that customer's credit profile and style of trading. Different trading venues mean that it is possible to have multiple simultaneous quotes for the same instrument. The FX trader's challenge is to trade with the venue that offers the most attractive quote at that moment for that currency, when consuming liquidity. When providing liquidity, the challenge is to publish quotes on the specific venues where counterparties are likely to deal in a particular currency and price level at that moment.

1.1.2. 24-hour trading. FX is traded 24 hours per day, in contrast to exchange traded instruments, that are only offered during exchange opening hours. The ability to trade FX 24 hours per day increases the diversity of liquidity to FX traders.

1.2. Aggregated order book. Typically, sell-side institutions have built their own aggregated FX order book for their own use or as price offerings to their customers. Buy-side institutions such as many hedge funds are typically more interested in minimizing cost and market impact by sourcing the best pricing for their order, even if it has to be broken into parts and routed to several venues, rather than paying the spread offered by a single trading venue for every trade. Transaction costs have been decreased in aggregate by the increasing prevalence of high frequency algorithms employed by buy-side firms. The largest components of transaction cost are usually bid-ask spread and slippage due to market impact.

Due to the decentralized nature of interbank FX trading, different trading venues provide different bid and ask prices for the same currency pair. Even though some trade venues mostly may offer tighter spreads in specific currency pairs than others, this is not always the case. Liquidity characteristics and spreads vary throughout the day for each trading venue and currency pair. The tightest spread and least concentrated market impact can usually be obtained by using an aggregated order book. For example, the following graph is a snapshot of quotes from different trading venues for the AUD/CAD. In this hypothetical example, we see that even though HotSpot provides the tightest spread, the optimal trade decision would be to use the higher bid price from HotSpot when selling AUD/CAD, and the lower ask price from Nomura when buying. This combination would lower bid-ask spread by 50% compared with trading on the venue with the tightest spread.



Source: Bloomberg.

Fig. 1. Snapshot of different quotes for AUD/CAD

1.3. The challenge of aggregating an FX order book. *1.3.1. Low latency.* Reducing the latency of high frequency trading requires a capital-intensive infrastructure, including server hardware, collocation with trading counterparties, networking equipment, and a trading platform. FX rates change in the millisecond time range, requiring a high standard for latency of message transfer and trade calculation. If the latency is high, the delayed FX rates streaming from trading venues or banks may be less competitive and subsequent orders may even be rejected due to the market pricing having changed in the time between the original quote having been generated and the order having been placed. Latency is introduced by message transport as well as computational latency in the trading platform. For example, assume an aggregated order book consists of four trading venues, each with different message transport latency. If the optimal offer price currently comes from BARX, it is still possible that an order may not be filled because of latency. Hence, when aggregating an order book of different venues, varying message transport latencies need to be considered in the order routing methodology.

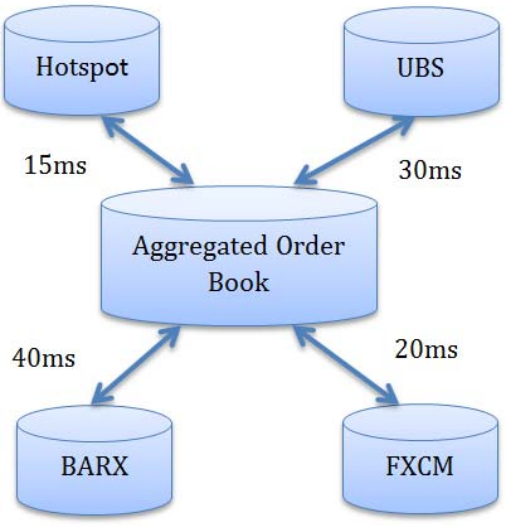


Fig. 2. Different pipe lengths for trading venues

1.3.2. Distributed and varying market depth. Large orders usually exceed the amount available to trade at the optimal quote price in a single trading venue. An aggregated order book requires an intelligent way to split a large parent order into subsidiary orders, each of which get routed to different trading venues. The following is an example of the various market depths for an aggregated order book for AUD/CAD. The red prices represent the optimal bid and offer prices at each time. If we want to execute an order smaller than 1M, it will be easy to take the optimal bid and offer prices from the composite order book. However, the specific market depth for the optimal prices may not be sufficient for a larger order. For example, an order to sell 3M AUD/CAD needs to be executed at time 1 (the first column of the order book), but the available liquidity in Nomura is only 1M. Rather than executing the full 3M in a single venue, the aggregate fill price will be improved by splitting the order into different parts and allocating them to different trading venues.

Table 1. Snapshot of a simple order book for AUD/CAD

EBS	UBS	NORMURA	BARX
$1.02_{70(1M)/73(1.1M)}$	$1.02_{69(3M)/73(2M)}$	$1.02_{71(1M)/73(1.5M)}$	$1.02_{70(2M)/72(2M)}$
$1.02_{72(1.2M)/76(1.3M)}$	$1.02_{71(2M)/74(2.5M)}$	$1.02_{73(1.2M)/75(1.6M)}$	$1.02_{72(1.5M)/74(1.5M)}$
$1.02_{74(1.2M)/77(1.4M)}$	$1.02_{73(2M)/75(2.5M)}$	$1.02_{75(1.2M)/77(1.5M)}$	$1.02_{74(1.5M)/76(1.6M)}$
$1.02_{73(1M)/76(1.1M)}$	$1.02_{71(2M)/74(2M)}$	$1.02_{72(1.3M)/75(1.3M)}$	$1.02_{72(1M)/75(1.3M)}$

Another phenomenon is that each trading counterparty changes its quote book in response to the liquidity being consumed at that instant. If a large order placed on Nomura consumes a lot of the AUD/CAD liquidity available on that platform, it is likely that the remaining AUD/CAD quotes on Nomura would be changed in response to hedge Nomura's resultant change in exposure to the opposite side of this large AUD/CAD trade. Over time, these quotes may revert to their original position as Nomura hedges its exposure and replenishes its AUD/CAD inventory from other available sources. These sources may be the same venues that the original trader also connects to and can trade on, creating a potentially complex series of market responses to large orders. The algorithms used by a buy-side aggregated order book need to take this into account so as not to concentrate market impact or create conditions that make it difficult for the counterparties to hedge their positions.

1.3.3. No standardization of messaging protocol. Despite the FIX protocol offers a broad standard for financial trading message interchange, every FX trading venue has different message formats, and different rules. For example, Electronic Clearing Networks (ECNs) – such as Reuters and HotSpot – allow traders to post limit orders but do not guarantee execution. Some single-bank trading venues do not allow limit orders, only Fill-or-Kill orders. These are executed against a stream of quotes from that bank, and those quotes are in turn generated in response to a request for quote message specifying a size band and duration for the quote stream to be valid.

1.3.4. No standardization of prices. ECNs usually control which market makers' prices are visible to specific customers depending on that customer's trading styles and needs. A trader usually does not get to see all quotes contributed by all market makers. For single-bank trading platforms, different customers get differing amounts of spread and skew applied by the bank depending on their trading style and credit profile.

1.3.5. Only quotes visible, not trades. In contrast to exchange-traded assets such as equities or futures, the interbank FX market typically does not allow traders to see other market participants' trades. Traders can only observe quotes. Traders can see how those quotes change, and then infer what trades or cancellations created those changes, but this problem does not have a unique solution. As a result, it is difficult to reconstruct the order flow that led to a particular venue's order book variations over time. This makes it difficult to use traditional algorithmic execution approaches employed in the equity market.

2. Complex event processing

2.1. The concept of CEP. The concept of "Event-Condition-Action" came about in database research in the 1990s as a way to describe the composite event processing logic of "active databases". The structure of traditional database architectures that use a "store-index-query" model is limited when one confronts a problem in which there are fast-moving updates. The challenge is multiplied multifold when events are derived from distributed sources (e.g., network delays, out-of-order events) and when performance is critical (e.g., when there are many event queries operating on a large number of events, only a few of which are of interest).

2.2. Difference between event-based and thread-based programming. Imperative thread-based programming languages such as C++, Java or Python that run process step-by-step in a series of threads have been the most common way for programmers to realize a project with transactional logic.

In thread-based programming, execution continues sequentially until that code thread is blocked by an I/O operation. At that point, execution in that thread is suspended pending the I/O completion, and the CPU core switches another (non-blocked) thread. This approach enables simultaneous I/O and computation, while still offering the predictability and coding simplicity of serial programming. However, this concurrency requires the programmer to deal with thread synchronization. The programmer has to ensure the protection of shared data spaces with locks and condition variables. In I/O intensive applications such as FX trading in an aggregated order book, this can lead to latent data races and deadlocks. The following figure is the basic structure of imperative programming.

Fig. 4. Basic structure of imperative programming

Compared with imperative programming, event programming uses a different flow structure. The order of execution is not determined by the order of statements in the code – it is determined by the arrival and processing of events.

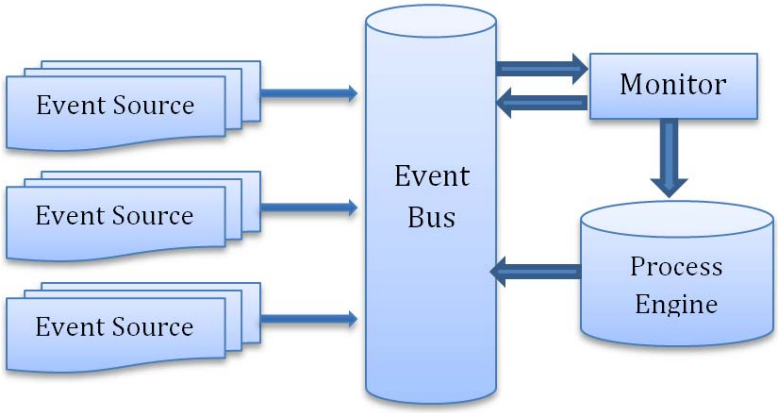


Fig. 5. Basic structure of CEP

In event programming, when a sequence of code cannot continue because it has to wait for an I/O event to complete, it registers a "callback" – a pointer to code that is to be called when the I/O event is complete. A callback executes linearly until it encounters a blocking operation, at which point it registers a new callback and returns execution to its originating point.

2.2.1. Computational efficiency. Traditional transactional architectures take data, store it to some static entity like memory or disk, index this data, and pass queries over the data to get results. A CEP architecture takes queries (named "listeners") and passes streams of data over those queries to trigger results, called "events". Those results can in turn trigger other queries (hence, the "Complex" name of event processing – the results of queries can create other queries. This allows one to define events that are aggregates or combinations of fundamental data events.)

One reason why CEP is computationally efficient (particularly for finance) is that a program sets up listeners only for events that are of interest at that particular time. It does not have to listen for all possible events all the time. There is no concept like a "main loop" that controls the flow of execution in a sequential manner. This allows a program to discard events that are not of interest. Only computational resources for the events that are of interest in a specific context are processed.

2.2.2. Loose coupling. CEP is more efficient for creating code for transactional problems in that the structure of code often matches the structure of the problem. The programmer defines events that match an event in the real world, or an event of interest that is derived from a collection of other events. Blocks of code communicate only by passing events to one another, rather than by using shared resources. This makes code more modular, easier to structure for parallel execution across numerous cores or machines, and more robust against failures. We call this "loosely coupled code".

This is an increasingly important consideration in event-driven applications that are diverse and distributed such as FX trading where there are multiple event sources and trading destinations, rather than "closed loop" environments.

2.3. CEP in FX order book aggregation. Because of the architectural differences described above, event-based approaches can produce stable latency performance that does not increase linearly in the face of increasing data throughput. This is in contrast to thread-based approaches, where aggregate latency is directly dependent on data throughput.

In FX trading where event rates are often many thousand per second and can be very "bursty" in the periods immediately following economic data releases, an event-based architecture can offer important advantages. The response to events in FX trading needs to be nearly instantaneous. This poses a challenge to traditional transactional architectures because the need for stable latency performance under bursty loads was not foreseen when these approaches were conceived.

FX aggregation is a computationally complex task and it scales significantly as more liquidity pools are added into the mix. Every time one of the pools changes, the aggregation algorithm must detect this, consider whether the change impacts the aggregated market view for a particular currency pair, and then make any necessary changes. This must be done with the minimum possible latency. Event-based rules can be used to instantly detect and act on FX market changes that require fine-grain reorganization of the aggregated view. For example:

1. Quote initiations, amends and cancellations from banks or ECNs are treated as events.
2. A CEP approach enables easy normalization of the different messaging and quote structures from each venue into a common event for representing market depth.

3. Orders can be normalized into a common event definition that may have a different semantic structure and set of rules for each liquidity venue.
4. CEP code is natively asynchronous. For example, one might receive a fill against an order before receiving the order acknowledgement. This might confuse some algorithms. By not requiring any synchronous structure, the code operates the way the problem does − by passing events or messages. Those events can have rules defined for them that address issues like reconstituting the structure of a complex event that has been implied by underlying events (in this case, a filled order).

Conclusion

FX order book aggregation poses several challenges, including differing transport latency between different trading venues, varying market depth, lack of standardization of messaging protocols, and stability of latency performance. In this analysis, we argue that complex event processing deals with the difficulties of FX aggregation better than alternative approaches. This is primarily due to the fact that: (1) computational resources are used only to process events that are of interest at that particular time; (2) code is more maintainable because it is built around events that are defined to match real-world events or events derived from an aggregate or sequence of other events; and (3) aggregate latency does not increase linearly as a function of event rate as it does with thread-based approaches, and this is important for maintaining latency stability during the bursty periods that are characteristic of FX trading.

References

1. Dabek, F., N. Zeldovich et al. (2002). *Event-driven programming for robust software*, ACM.
2. Etzion, O. and P. Niblett (2010). *Event Processing in Action*, Manning Publications Co.
3. Luckham, D.C. (2001). *The power of events: an introduction to complex event processing in distributed enterprise systems*, Addison-Wesley Longman Publishing Co., Inc.
4. Biais, B., P. Hillion et al. (1995). "An empirical analysis of the limit order book and the order flow in the Paris Bourse", Journal of Finance, 50, pp. 1655-1689.
5. Flood, M.D. (1991). "Microstructure theory and the foreign exchange market", *Federal Reserve Bank of St. Louis Review*, 73 (6), pp. 52-70.
6. Progress Software (2010). Apama technical white paper.