2018

FUNDAMENTAL INTERACTIONS

Neutron

ARCHITECTURE OF A MODERN MULTI ASSET TRADING SYSTEM

ARCHITECTURAL OVEVIEW

Contents

Build Versus Buy a Trading System

When time-to-market and money are top priorities, executives should closely evaluate commercial software prior to building new trading system components. Having the right blend of build versus buy can be as important as the trading strategy. The rule of thumb is to buy applications to the maximum extent possible to cut costs, freeing up resources for the things that really need to be built in-house. Why reinvent the wheel? Commercially available software comes with the benefits of all of the design, development, enhancements and testing history associated with the existing customer base and it can be immediately deployed to meet the majority of requirements out of the box. Important software design to consider for a vendor solution is the following criteria:

- **Reliability** - is the ability of a software platform to perform required functions under stated conditions for a specified period of time. It is critical that the software architecture is fault tolerant and accommodates failures of application and infrastructure. Fault tolerant systems are typically designed with multiple application instances operating simultaneously and in case of failures of one component; the backup component takes over seamlessly.
- **Scalability** - is the capability of a system to increase total throughput under an increased load when resources (typically hardware) are added.
- **Extendibility** - is ability of the system to expand beyond initial functional and operational needs. Systems designed with open architectures are able to expand and integrate additional components without overhaul of core system.
- **Exception Handling** - is ability of the system to handle unexpected circumstances like garble messages, operational failures and continue its usual functions after handling these unexpected scenarios.

Overview of the FIN Trading System

Historically, trading firms have been required to piece together disparate vendor and in house solutions in order to meet business needs; Fundamental Interactions 'Neutron' (FIN) solution offers the alternative of a unified system. The Neutron system consolidates a core set of low latency business functions into single application software that runs on commoditized hardware. Within a single application process, FIN can deliver market data, pre-trade risk, algorithmic routing, and order crossing, allowing new advantages in performance and footprint efficiency.

All of the key trading infrastructure tasks can be accessed within a single server running as a single process rather than across a distributed multi-server environment. This allows FIN clients to achieve unprecedented speed and footprint efficiency. The modular architecture of the FIN system allows clients to deploy only the components they require, ideally accommodating black boxes as well as enterprise trading infrastructure deployments.

Integrated Modular Ultra Low Latency Appliance Functions

- Market Data Ticker plant
- Order Gateway
- Pre-trade compliance
- Algorithmic engine
- Smart Order engine
- Matching engine
- Administration and Monitoring

Neutron is Reliable

- The Neutron system is designed with option to configure backup for all the processes. No single point of failure exists as part of the overall software platform.
- Neutron has implemented end to end hot-hot failover mechanism. Each process is configured with a primary and a backup. All the processes implement internal heartbeat. These heartbeats help identify disconnects, network hung, process hung and similar situations to trigger automatic failover to secondary. The primary process asynchronously updates the backup process with all the necessary data like order information, fix session information etc.
- In case of failure of primary process, the secondary processes will seamlessly take over as primary. The messages transfer between source process and destination process is through a reliable message delivery mechanism.

Neutron is Scalable

- The Neutron software can be configured as a single process handling all the modules or one/multiple process per module. The system is highly scalable. The system can be scaled based on the functional models as well as alpha security range within the module.
- Multiple Inbound Gateway processes can be implemented to handle more clients. Multiple Outbound Gateway processes can be implemented and configured to process symbol ranges. The outbound gateway can be configured as primary/secondary or in peer mode.

Neutron is Extendable

- The Neutron software architecture is very modular and has a high level of flexibility and adaptability. New functional components like new market data feeds, connection to new trade reporting facility, and support for a new application interface can be easily integrated in the system.

Neutron is Proactive

- Neutron has the ability to identify malformed messages and reject those with appropriate reject reasons.
- The system administration monitoring and audit trail has the ability to capture and notify following attributes
  - ✓ Message Rate
  - ✓ System Health
  - ✓ Process Status
  - ✓ Market Data Latency
  - ✓ Order audit trail (combination of the inbound order messages, outbound order messages and final state of the order that is present in the database.)
  - ✓ Exceptions/Alerts
  - ✓ FIX session information (Messages, Session status)
  - ✓ Market Data control messages
  - ✓ Email notifications
  - ✓ Order and Quote management

System Features Overview

An Interdealer Quotation System

The system supports interdealer quotation for market participants to post orders and firm quotes in the in the market making book which can be used internally as well as externally with venues. Two-sided, one-sided or non-priced quotations are accepted and displayed in price/size/time/priority. Externally market making is supported for equities, pinks, options, futures, and FX asset classes.

Automated Execution System

Orders and firm quotes are submitted to this system and the crosser will maintain an electronic book through which orders can be displayed and matched. The order book matching algorithm uses price/size/time priority criteria for crossing the order and also checks other available venues to assure best execution. All orders and market maker quotes at the same price level are filled according to time priority.

Route or Smart Order Router & ALGOs

The Route is a system that maintains an internal book of other available market centers and can route an incoming order to the corresponding venue to satisfy best execution obligations. Orders tagged as routable will be compared to the NBBO and routed to the venue displaying the best price for execution. Subscribers have the option of identifying their quotes and orders as routable. Quotes and orders not tagged as routable will be displayed and crossed on the internal Crosser book only. Other routable options are to use the internal pre-built Algorithms that support TWAP, VWAP, and etc.

Auctions

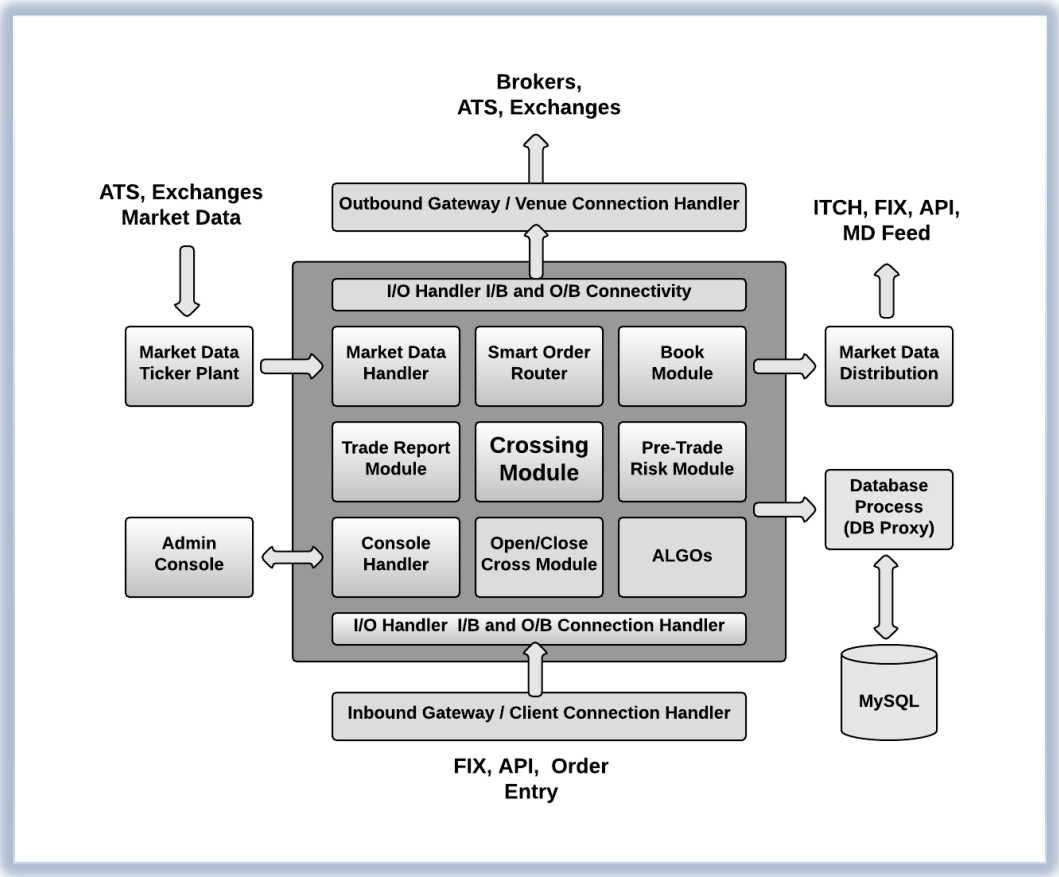
Firms have the opportunity to participate in an auction-based message routing system designed to electronically satisfy a minimum three quotes. Before execution of a crossed order, the market will be checked for two (2) available electronic quotes. If less than two (2) electronic quotes are available, participating firms will be sent an indication of interest (IOI) representing the order. An auction is then held for a specified period of time and the order is executed against the best price received if a total of three (3) quotes/orders are available at the time of execution.

Market Data

The top of book (inside price best bid/offer) is available in crosser book will be disseminated as part of all venue market data distribution. Quote Level I and Level II market data feeds will be available on broadcast via ITCH feed and also TCP broadcast feed. The broadcast will consist of the following:

- Level 1 – best bid and best offer of internal crosser book and all external venues
- Level 2 - Indicative market participant quotations and inside quotations for internal crosser book and all external venues
- Trading halt information for internal crosser book issues and external venues
- Market event control messages and general administrative messages for internal crosser book and external venues
- Symbol Master Feed - Normalized security symbols and suffixes along with Security description
- IOI Feed – During auctions and 3 quote rule validations
- Client Connectivity
- Client firms can connect to the system via FIX or via Open Application Interface (API). The FIX and API will provide access to the crossing platform, smart order router and auction controlled through order types and order attributes.

Illustration – Neutron Modules



Module Details

Market Data Ticker Plant (for external market feeds)

The Market Data Ticker Plant consists of various processes that receive multicast market data feeds from various exchanges. These processes normalize the venue specific messages into the internal binary API and distribute the market data to internal subscribers.

Book

The Book module builds, maintains and distributes internal order book. The main function of the book module is to receive orders, order updates and order executions from the matching engine as well as market data updates from Market Data Ticker Plant and aggregate these messages into a single order

book. The internal book is price aggregated order book sorted with price-time priority. The integrated book includes client orders, client quotes and market data price points received.

Every source is indicated as a separate row in the depth of book. The quote messages received from clients or external destinations are converted internally into order messages before integrating with the book. The Book module receives the order information as a reliable message from the Crossing Module. The Book module receives the market data information against subscription from the Market Data Ticker Plant. Book will maintain cache of the aggregated book information. The Book will not maintain individual order information. The order updates will be applied to the book record and discarded. In case of ATS DISPLAY, the Book module published the book activities (market data) through Market Data Distributor Module which includes all the changes in the internal order book. Clients can subscribe for individual sources or combinations of these sources. Wild card subscriptions from Crosser and Smart Order Router are directly sent to Book to request all the available market data from all available sources.

Market Data Distribution Module

The Market Data Distribution Module subscribes to the Book Module to receive all the order updates and other notifications (security, trader, IOI etc.). This module then distributes these updates as market data feed to the clients. The clients can subscribe through a TCP-IP connection or join multicast distribution to receive the internal book information (market data). The market data distribution engine will also redistribute the market data received from venues. An integrated combination of venues and internal Crosser book will be available as part of market data distribution. A symbol master and trader reference data will be disseminated by the Market Data Distribution Module. The broadcast, except IOI, will be available in multicast as well as TCP-IP with a recovery/retransmission handling through TCP-IP. The market data broadcast implements high performance ITCH protocol. The IOI distribution will be through TCP-IP and will be controlled through entitlements. The platform will make available the services to subscribe for Crosser last sale, Crosser book, ARCA venue books and combination of these books. The administrative messages like Symbol Halt, Market Open/Close, Pre-Opening etc. will be disseminated as part of the market data feeds. Every price aggregated record has an identifier to indicate the book source or combined.

Inbound Gateway

The inbound gateway is the Application's interface with the clients. This module accepts client connections in FIX, API (binary) protocol. The inbound gateway receives orders and quotes from clients and responds back with the order updates and executions. The quote messages are converted to order messages before sending to the Crosser. This module is also responsible for aggregating and scaling the client connections to avoid performance overload on the Crosser. The Inbound FIX Module can be configured to process one client per listen port, or multiple clients per listen port. A thread pool is implemented where one thread can be assigned to multiple client sessions or can be assigned exclusively for a single client session. The inbound gateway has the ability to connect to the Crosser based on symbol range. Thus this module also acts as an aggregator of client connections. All the FIX messages, inbound and outbound, are recorded into database for recovery and records.

Outbound Gateway

The Outbound Gateway is responsible for managing connections with external destinations. It acts as an interface between the Crosser, Smart Order Router and external destinations. The TRF Modules connectivity to the ORF is also facilitated through the Outbound Gateway. Typically one set of outbound gateway processes will be implemented per external connection. The message routing between the Smart Order Router or ORF Module and the outbound gateway can be configured in round robin to gain scalability.

Smart Order Router

The Smart Order Router, SOR, routes orders to pre-determined venues based on the instructions on the order. The Smart Order Router updates the Book module with the parent order and child order updates.

Pre Trade Risk Module

The Pre Trade Risk module will perform all the pre execution and pre validation checks on all the orders received and execution opportunities. The main purpose of this module is to apply compliance checks and pre trade risk checks for all incoming orders and prior to their executions.

Crosser

The Crossing Module is the most important module of the Platform. The Crosser maintains order state, processes market data events, identifies crossing opportunities and makes decisions to execute, cancel, re-route, pullback orders. The Crosser also updates the internal order book and initiates distribution of depth of the order book and top of the order book information. The Crossing Module maintains order information along with the order routes (child orders) and order executions. Depending upon the order attributes, the crosser routes the orders to external venues (as in case of directed orders) or to smart order router (as in case of smart orders). The Crosser sends order updates to the Book Module to participate in the internal depth of book. The Crosser maintains a cache of all the open orders at individual order level. For persistence storage, all the order messages and execution are sent to the Database Engine that further writes it to database. For every market price tick or quote/order received, the Crosser checks the order book for potential cross.

Auction

The Crosser also hosts the Call Auction Module which is responsible for performing Opening and Closing Cross. This Module will monitor order designated for Opening and Closing Cross and calculates cross prices. This Module is responsible for complying with three quote rule and generating IOI.

3 quote rule Module / IOI Module

For a crossing opportunity, if the security is eligible for 3 Quote rule, however it does not satisfy the rule requirements of having at least 3 eligible quotes/orders, this Module is responsible to generate Indication of Interest (IOI). The 3 quote module will also validate the broadcast entitlements before sending IOIs to these clients. The IOI will be disseminated on TCP-IP connection. The clients will need to be configured to participate in receiving the IOI feed. For persistence storage, all the IOI messages are sent to the Database Engine that further writes it to database.

Trade Reporting (TRF) Engine

The Trade Reporting Module receives order execution or trade messages from the Crossing Module and reports these trades to the TRF. The TRF module receives the reporting information against subscription from the Crosser.

Admin Console

The Monitor / Console Handler acts as an interface with the Admin Front End/GUI to facilitate user interaction with Crossing Module, Book module and TRF Module. The Monitor also interfaces with the Inbound Gateway and the Outbound Gateway to support FIX session monitoring and FIX auditing.

The Graphical User Interface allows the support personal to view internal order book, enter orders and indication of interests, cancel orders, and give unsolicited UR-Outs. The front end interface also provides view into the FIX messages and ability to manage FIX sessions. System halt, symbol halt and other administrative messages can also be generated through the admin console

Database Engine

The Database Engine is a process that acts as an interface between the persistence transaction generators or source modules like Crosser, ORF Engine, Inbound Gateway, Outbound Gateway and the Database. The Database Engine implements an efficient mechanism to send batch of insert/update messages to the database. Each batch is considered as a single transaction. Upon successful completion of this transaction, a notification is sent back to the source. The communication between various modules and the Database Engine (DBProxy) is asynchronous. The batch processing mechanism is also implemented for communication between the Crosser and DataBase Engine. MySQL database is used as persistent storage.

Programming Language Benefits

Java sometimes carries the stigma of being slower than C++/C and therefore less appropriate for the design of ultra-low latency trading systems. While Java may have been slow when in 1996 when it first came out, this is no longer the case.

The JIT compiler compiles java interpreted code into native code which executes in a manner identical the C++/C executable. The FIN system runs a warm up script prior to production trading, using the -Xcomp flag to compile as native code, and avoiding the first time compilation latency penalty in production trading.

Java's exception handling is extremely good. It was adapted from ADA, a programming language used by military. Unlike an executable program created in C++/C, which provides programmers with more controls but can make it very difficult to track bugs such as unhandled exception errors, Java code provides very constant output.

Java itself is a mini OS with many features that help to improve program speed. Memory allocation is a good example. In C/C++, all threads allocate from same memory heap, which can become a bottleneck if a program frequently allocates and frees memory. In Java, each thread allocates from its own memory

space. A whitepaper published by IBM and other companies compares the memory allocation time between Java and C++, and shows that while a Java program takes about 10 nanoseconds to allocate, a C/C++ program takes about 100 nanoseconds to allocate. Furthermore Java's memory de-allocation is handled by a separate thread, freeing the active thread from the time spent on memory de-allocation and providing that thread with more time to execute the application code.

Fundamental Interactions Neutron Trading Appliance takes advantage of many features offered by Java. One example is the use of a security symbol as a string object. In C/C++, it may be required to allocate many copies of different instances of a given security symbol so that different threads can be allocated to different objects. In Java however only one instance is needed. One can attempt to write a thread-safe smart pointer for a given security string in C/C++, but the result will be very clumsy, in Java, this is taken care of.

In addition, we do not use the JVM garbage collections; all collection is managed separately by our application using a concurrent collector to manage the allocation and de-allocation of memory. This is more efficient than any C++ application.

Eliminate internal I/O among different modules. Instead of marshall/unmarshall data through IPC protocols, data (of the same memory block) can be passed from one thread to another with almost zero latency.

Data can be shared much more efficiently among different modules. Say pre-trade risk modules needs NBBO to calculate market order margin risk, instead of subscribing for NBBO quote from quote modules, it can share the same current data memory storage that quote modules are using.

Performance Numbers for the Neutron System

Tick to Trade Latency

The industry measures a system performance by testing the time a quote is read from a market data feed handler an order is generate in response to the quote and sent to the market.

The methodology Fundamental Interactions uses for measuring tick-to-trade latency is described below.

The CQS feed handler reads and normalizes the raw inbound feed. The Neutron smart order router receives the normalized data and decides to send a hit/take order. The native message is converted to FIX and is sent to FIN's simulator via TCP.

The measurement is based on the input (the raw live CQS market data in UDP packets), and the output (the FIX message in TCP packets). The Unix utility tool tcpdump is used to record both sides. The two dump files are linked by dynamically binding the unique line sequence number (present in each CQS message) to tag 943 in outbound FIX message. The tick to trade latency is found by comparing the corresponding sequence values.

Testing Results

Testing was conducted on a machine with 2, 6 core processors (12 core total, 3.3G Xeon) running SUSE LINUX.

Percentage	Microseconds
Top 5%	25 µsec
Top 50%	30.7 µsec
Top 90%	32.1 µsec
Top 99%	33.5 µsec
Worst 5%	44 µsec
All Average	32 µsec

- More than 41,000 results
- Avg. 200 - 300 orders per sec
- Best = 5.1 microsecond

The Neutron System Throughput Capacity

The system (depending on the machine and network card configuration) can handle between 100,000 orders per second before the onset of performance degradation.

To understand the number of messages over the course of a trading day a single box can process, we can use the formula orders per second multiple seconds per hour multiply number of hours with the example of 20,000 orders per second. The math works as follows:

(20,000 orders per second) x (3,600 seconds per hour) x (7 hours per day) = 504 million orders per day. Estimating 2-3 associated messages per order, the total comes to 1.5 billion messages per day. FI has clients that exceed 1 billion messages per day per routing machine.

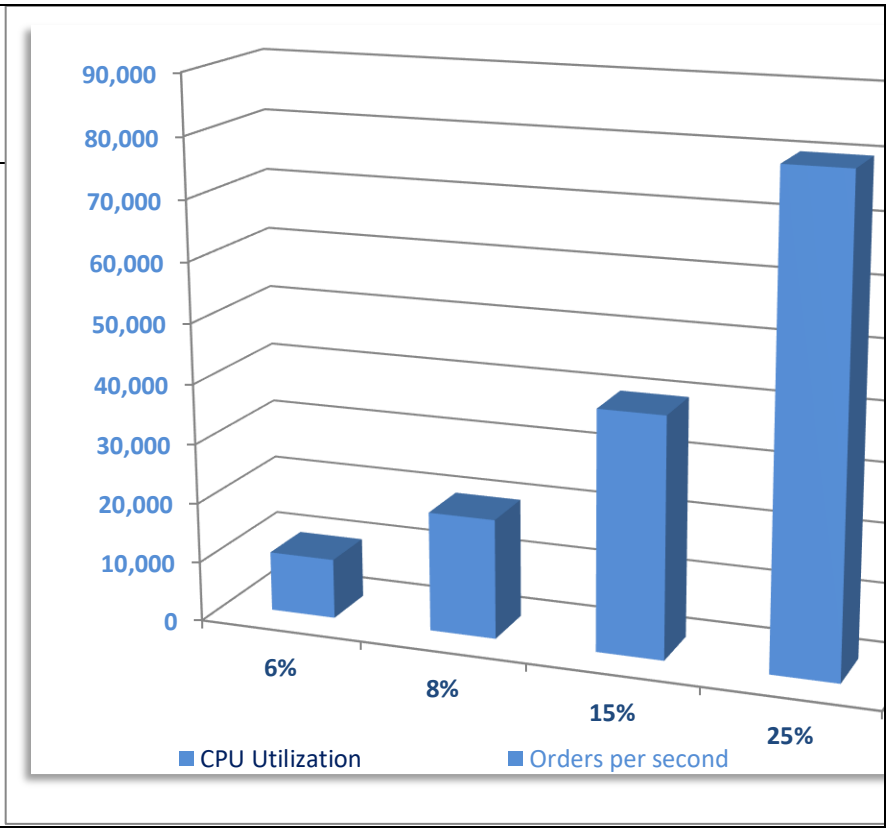
FI client firms with extremely high throughput can split the routing out onto additional machines. The FIN system can load balance by alphabet range or by round robin.

CPU Usage for the FIN Gateway at Different Throughput Rates

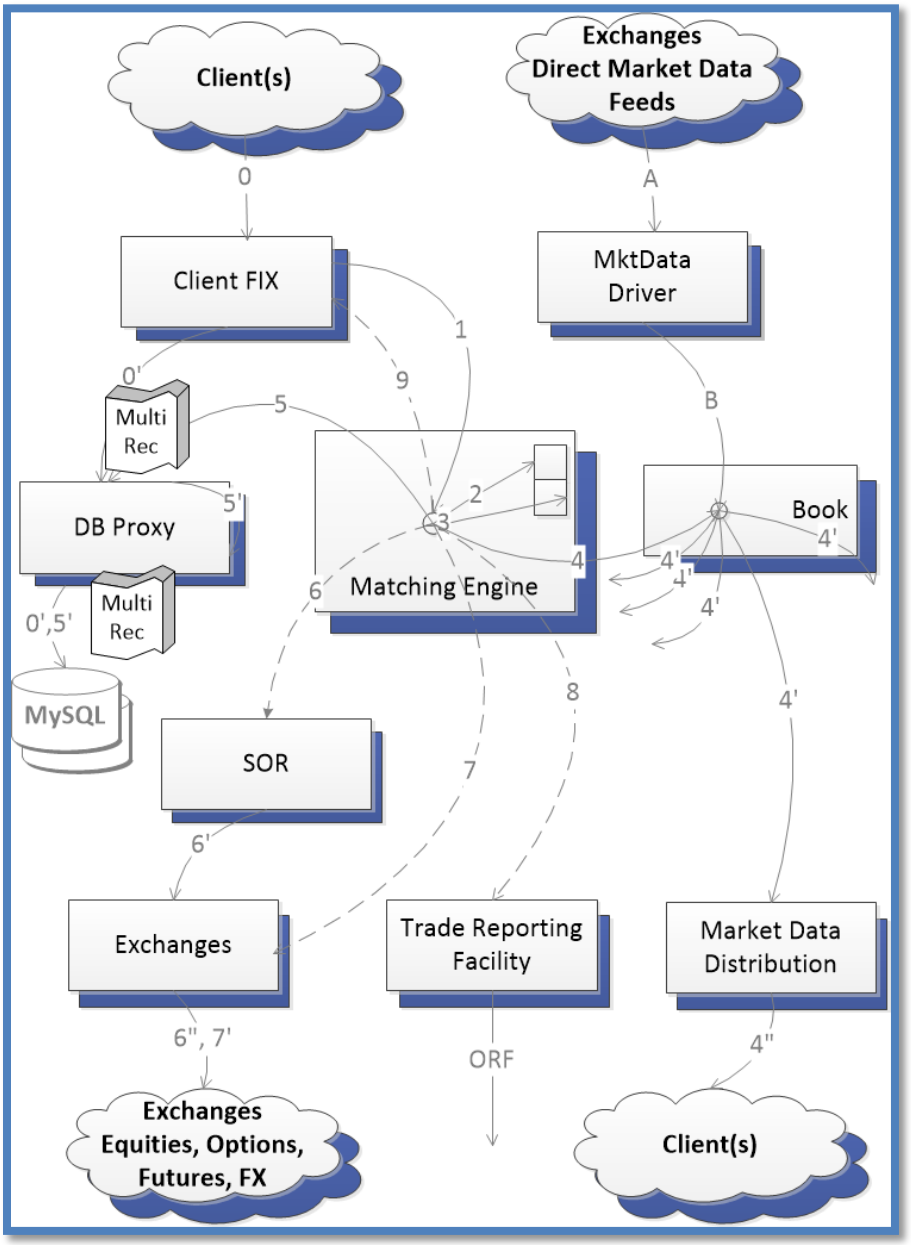
- Control Set / Environment - 12 Core, 2.5 G Intel Xeon, running Linux OS
- Order Flow Characteristics – FIX protocol / mixed profile of Day, IOC orders*

CPU Utilization with Increase of Orders Results

Order Rates	CPU Utilization
10,000 o/p/s	6%
20,000 o/p/s	8%
40,000 o/p/s	15%
80,000 o/p/s	25%



Example – Neutron Deployed as "Virtual Exchange"



Event Flow Sequence

Event	Description
0	Client Order received in FIX module
0'	Client FIX message persisted To Data Base
1	FIX Client Module sends Order to Matching Engine
2	Order is cached in internal memory cache
3	Order is synchronized to secondary
4	Order update sent to Book module
4'	Book Module updates book and broadcast to clients subscribing for book
5	Client Order is persisted to Database
0", 5'	DB Proxy writes transaction to SQL database
6	Smart order is sent to SOR (Smart Order Router)
7	Directed order is sent to Exchange/Venue FIX module
8	If Order is Crossed, execution reported to ORF
8'	Trade Report message sent to FIX module
9	Execution Report sent to Client FIX module
A	Unsolicited Market Data received from Exchange
B	Market Data update sent to Book module
B'	Market data update distributed to modules subscribing for market data (book). (Similar to 4')
7	If B' Market Data is crossable with internal book will result in route to external destination