



© Copyright 2011, The NASDAQ OMX Group, Inc. All rights reserved.

NEXT-GENERATION TRADING SYSTEMS

LOW-LATENCY SUMMIT LONDON 2011

NASDAQ OMX[®]

BJÖRN CARLSON, VP OF SOFTWARE ARCHITECTURE, NASDAQ OMX

SUMMARY OF SOFTWARE EVOLUTION FOR LOW LATENCY



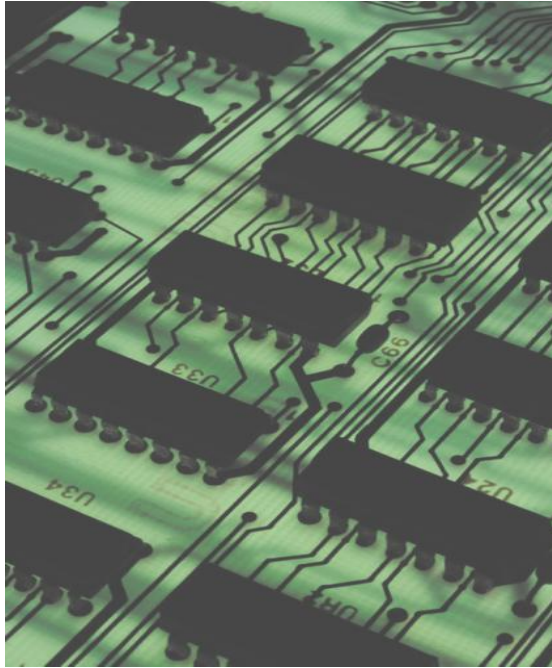
- How to make use of multi-core processors
- What is a reasonable latency floor
- Scalable architecture
- Functionality trends
- Fault-tolerance
- Incremental modernization
- Hosting exchange services
- Remaining vendor-neutral via open-source (build vs buy)
- NASDAQ OMX whereabouts

THE GOAL OF REALTIME



- **Trading system is a realtime message processor**
 - Minimize the jitter of message responses
- **Conflict between throughput and latency and predictability**
 - Throughput = pack as much computation per IO operation as possible
 - Latency = pack as little computation per IO operation as possible
 - Determinism = pack the same amount of computation per IO operation if possible
- **Core of a trading system consists of a matching engine**
 - Each input message triggers matching algorithm and output messaging
 - Ideally same amount of work for each incoming message
 - Limit orders are ideal for deterministic and low-latency computation
 - Complex orders (combinations, triggers, mass-cancels) are far from ideal
- **Conflict between simple and complex matching**
 - One extreme: one engine that rules them all
 - Other extreme: distributed logic that monitors limit-order flow and acts on the conditions (triggers, pegging, etc) in asynch realtime
- ***Does algorithmic trading favor the simple engine?***

MULTICORE PROCESSORS



- **Must thread the application to gain performance**
 - Note that this *does not* imply sharing memory
 - Concurrency is complicated and messes with determinism
- **Must share server among multiple processes**
- **Long-term, programming environments need to better support multicore paradigm**
- **Foundation is a robust and fast messaging system**
 - Build or Buy?
- **Message-passing is a good paradigm for multicore**
- **Event-driven programming good for realtime features**
 - System developers must think like kernel-developers

LATENCY RACE HITS THE FLOOR?



- **At what point do you get decreasing returns?**
- **Core engine does not drive latency**
 - Network does
 - Gateway does
 - Complex func does
- **Don't want the latency race to hollow out the robustness of the architecture**
- **Seems the response times of different systems are converging**
- **Expect gateway latency for basic order types to approach sub-50 micros**
 - Add to this external switching and socket overhead

LATENCY TECHNIQUES



- **Kernel bypass**
 - Buy rather than build
- **Kernel drivers**
 - Hard to maintain and make portable across drivers
- **Hardware acceleration**
 - Inflexible
- **Network improves faster than processor**
 - Software is the bottleneck
 - That is, gateway and engine and backend-server
 - Concurrency/Parallelism is a must
- **Stay predictable, i.e. queues are nasty (but inevitable)**

SCALABLE ARCHITECTURE



- **System scalability is fundamental**
- **Core engine typically capable of managing the whole market in one or two instances**
 - Benefit of partitioned engine is less queuing and hence better and smoother latency
- **Downstream systems less potent**
 - Downstream scalability is critical
 - Downstream processes often include legacy/back-office interfaces
- **Consider architecture where data flows downstream from engine and naturally branches out for parallel processing**
- **Avoid dependency on synchronous replication as this will be a bottleneck**
- **Define clear methods of replay and loosely coupled synchronization (avoid 2PC)**
- **Large-scale data processing is quickly becoming the next challenge**

FUNCTIONALITY TRENDS



- **Multitude of product types**
 - Cash
 - Derivatives
 - Commodities
 - Combinations
 - Currencies
- **Specialized and regulated exchanges**
 - MIFID and SEC regulation
- **More monitoring of order-flow**
 - Pre-trade risk management
 - Market supervision (protect against HFT running wild, unreasonable transaction patterns, evaluate netted risk, cooperate with big firms to avoid UBS-style 'surprises')
- **Over-the-counter negotiation, trade-reporting and clearing**
 - Will it be consolidated/centralized?
- **Large-scale data mining**
 - Realtime monitoring
 - Algorithmic trading

FAULT TOLERANCE



- **Software-based to avoid vendor-lockin**
 - Includes any type of replication
 - Local disk should be fine really
- **In-memory storage more common**
 - Power supply is thus more critical and must be redundant
- **High-speed networks make replication almost gratis**
- **Expect system to support replay of individual components**
 - Loose coupling is good
 - Replay is repeatable
 - Simpler and less intrusive than synchronous checkpoints throughout the day

INCREMENTAL MODERNIZATION



- **Trading systems evolve over time**
 - Hard to replace in one fell swoop
- **Vendors must find incremental upgrade strategy**
 - Those with a scalable architecture from the start will benefit
- **You end up with a mixed-system environment**
 - Look for bottlenecks (most likely downstream)
 - Look for downstream processes that lack strong fault-tolerance
- **Security concerns are entering**
- **Incremental upgrades is a good thing**
 - Increases chance of stability
 - Less member impact
 - Battle-proven code has immense value

HOSTING

- **Hosting of exchange services gives smaller exchanges a chance**
 - Share high-speed networks and servers with others
 - Continuous investment in hardware and networks
 - Cost-efficient way of implementing low-latency and data-mining
 - Easier to enforce regulation and a high-level of security
 - Hence, lowers cost of competition
- **Cloud depends on scalability (partitioning) and a 'leaf' architecture**
 - Virtualization is not an option upstream if performance is critical
 - Downstream this may well be the correct solution
 - Application (exchange) isolation via virtual networks and system partitions



STAYING VENDOR NEUTRAL



- Open source has become a very strong presence within exchange industry
- Linux and x86 is almost the only choice
- Good reasons for this
 - Very lively computing realm
 - Transparency of open-source helps
 - But endorsed editions of critical components are still preferred (kernel, application servers, databases, languages)
- **Is Java it for the next ten years?**
 - Downstream there are other high-level alternatives
 - No need to religiously dictate one language
 - Java not optimal for memory-caches, but future versions may fix this
- **Is SQL it for the next ten years?**
 - Maybe, as the no-SQL systems still require a high level of skills to integrate
 - There is yet to be a clear winner
 - The cloud may help

NASDAQ OMX ROAD MAP TO OPERATIONAL EXCELLENCE

- Operational excellence in next-generation trading systems

- Continue to invest in INET-based systems (NASDAQ, PHX, OMX, Genium INET, X-stream INET)

- Scalability
- Incremental modernization
- Latency
- Cloud services
- Ease of operation
- Market supervision
- Risk management

