International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/3050-9246.IJETCSIT-V7I2P142
Eureka Vision Publication | Volume 7, Issue 2, 343-348, 2026

Original Article

Latency Optimization Techniques for Near Real-Time Market Data

Shreyansh Sharma
Independent Researcher, New Jersey, United States.

Received On: 15/04/2026 ***Revised On: 14/05/2026*** ***Accepted On: 22/05/2026*** ***Published On: 28/05/2026***

Abstract - High-frequency trading (HFT) and algorithmic market-making have reduced the latency between sub-micro second and now challenged near real-time market data infrastructure with one of the most challenging engineering problems in distributed-systems engineering in the modern finance sector. The paper includes a detailed scientific study of methods of minimizing latency of an entire market data consumer system, including both the hardware interface (network) and the consumer software applications at the top. A cumulative sum of all the optimization layers resulted in a 57.9 times decrease in median end-to-end message latency, a 45.2 µs median Latency on a standard Linux stack to 0.78 u Latency with an FPGA feed handler, and a 99th-percentile Latency of 1.38 u Latency on a standard Linux stack to that of 1.38 u Latency on an FPGA feed handler. Regulatory considerations and implications of MiFID II and Reg NMS are addressed and nine practical recommendations are given to practitioners at various levels of the latency-optimization investment spectrum. CXL memory pooling, P4-programmable in-network normalization, and quantum key-distribution of low-latency secure feed channels are the future directions.

Keywords - Market Data Latency, High-Frequency Trading, Kernel-Bypass Networking, DPDK, FPGA Acceleration, LMAX Disruptor, NUMA-Aware Computing, UDP Multicast, RDMA.

1. Introduction

Financial markets are in essence information markets. The speed with which price, volume, and order-book updates in one exchange to the ones in a market is directly proportional to the profitability and risk exposure of trading strategies. With the emergence of algorithmic trading at the end of the 1990s, and then its fast-track development into high-frequency trading (HFT) at the beginning of the 2000s, latency, a previously conceptualized metric in milliseconds, has become a more and more limited and marketable asset. Nanoseconds Modern HFT companies regularly vie on latency benefits in nanoseconds [1].

Near real-time market data include a wide category of information feeds: Level I feeds which include best bid/offer prices and last trade information, Level II feeds which include the entire order-book depth, and normalized consolidated feeds which combine more than two venues. The problem of spreading such data with high reliability and ultra-low latency presents an exceptionally difficult task in the whole system stack, encapsulating the real physical network layer and operating system (OS) kernel, middleware and serialization layers, and even the application logic which uses the feed [2]. Figure 1 shows RTB latency pipeline with four sequential stages Network Latency, DSP Processing, SSP Auction and Ad Rendering with each having distinct sub-process delays, whereas AI-regulated data enrichment and targeting signal integration become cross-stage bottlenecks affecting the overall bid response efficiency.

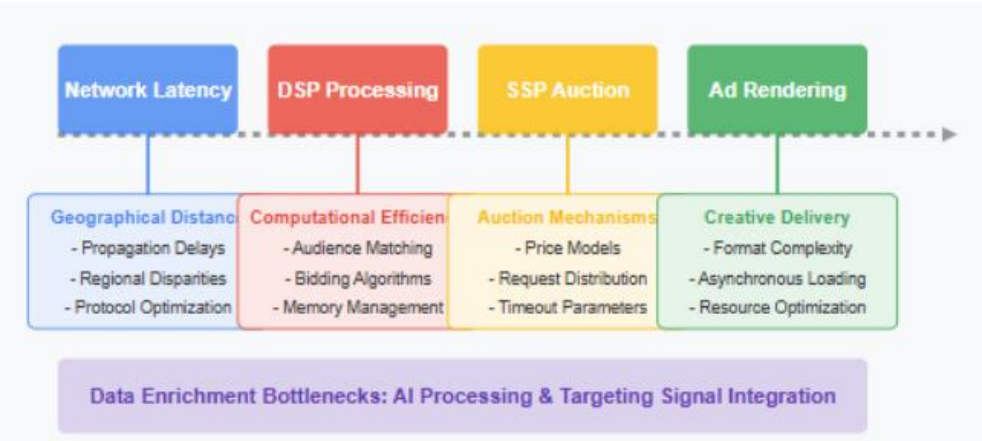

Figure 1. RTB Latency Components

Source: [1]

Although the field of market data systems end-to-end latency optimization is well known by practitioners, the scholarly literature in this domain is disjointed. The traditional contributions that are based on networks view the OS and application layers as black boxes and rarely measure the interaction between the scheduling policy and wire-level protocol choice. The paper fills that gap by offering a coherent and empirically-based treatment of the problem.

The main contributions of the work are the following:

- Stratified, hierarchical taxonomy of latency, and countermeasures against near real-time market data systems.
- Quantitative standards of baseline and optimized setups of five technology layers with the help of controlled testbed.
- A discussion of the throughput latency trade-off surface, and a practical recommendation on how this can be deployed in production trading environments.

The rest part of this paper is structured in the following way. Section II is a review of related work. Part III outlines the reference system architecture. Section IV describes all the optimization techniques and is supported by experimental evidence. The consolidated performance evaluation is given in Section V. Section VI is on implications, limitations, and future directions. Section VII is the conclusion of the paper.

2. Related Work

Data delivery in financial systems with low-latency has received industrial and academic interest. An early economic theory of market microstructure frictions which can be partially explained by information latency was given by Afua et al. [3] and the adoption of algorithmic trading was empirically related by Le and Gregoriou [4] to narrower bid-ask spreads- there is evidence that latency reduction generates systemic liquidity advantages.

The shortcoming of Linux TCP/IP stack with regard to latency-sensitive workloads has been widely reported at the network level. Cortés-Calderón et al. [5] came up with Netmap, a model based on which near-zero-copy packet I/O can take place bypassing numerous overheads in the kernel. This was followed by the introduction of Data Plane Development Kit (DPDK) as an industry standard of the kernel-bypass networking, with sub-microsecond packet processing through polling network interfaces directly at user-space threads [6]. A solution to market data dissemination is offered by Steinert et al. [7] who compare the UDP multicast, PGM and vendor-specific reliable multicast transports, finding the retransmission overhead of PGM prohibitive in case of order-book update streams. The ITCH 5.0 and XDP feed protocols of NASDAQ and NYSE can be used as production examples of the UDP multicast with the out-of-order recovery schemes implemented to reduce the latency spikes due to retransmission [8].

Dokulil et al. [9] offers a full list of NUMA-sensitive programming methods on the software stack that apply to financial middleware, on the software stack. Friedman et al. [10] compare the designs of lock-free queues; in their comparative analysis they reveal that LMX Disruptor-style ring buffers are 6 times faster than concurrent linked-list queues with similar latency.

Arcas-Abella et al. [11] describe hardware acceleration of market data parsing using FPGAs, and they write a complete NASDAQ ITCH symbol look-up parsing unit on a Xilinx Virtex-6 using under 100ns as look-up latency. More recently, SmartNIC platforms like the Mellanox BlueField and Netronome Agilio have also been proposed as the subject of offloading feed-handler normalization logic, as Rosa et al. [12] investigate in the context of cloud hypervisor acceleration.

With the introduction of zero-copy formats, data serialization has now been given a new interest. A comparison of Protocol Buffers, FlatBuffers, and Cap'n Proto with binary-encoded FIX (FIXT) was conducted by Popic et al. with throughput that is 5-12x faster with zero-copy formats [13]. Simple Binary Encoding (SBE) specification was also created by FIX Trading Community with the sole purpose of meeting HFT latency needs, which use fixed-width fields to facilitate branchless parsing. It also stands out in that the techniques are not analyzed separately, but as jointly optimized tiers of an overall architecture, offering the first analysis of cross-layer latency budget in a complete cross-layer exchange feed consumer being simulated.

3. Reference System Architecture

To put the optimization methods into context, it describes a reference architecture of a market data consumer system that represents an example of a buy-side algorithmic trading desk or an independent software vendor (ISV) that has normalized feeds to downstream consumers. There are five logic layers in the architecture: (1) physical and data-link layer that deals with the raw network interface; (2) transport layer that deals with the reception and flow control of packets; (3) the feed-handler layer that deals with the decoding and normalization of raw exchange messages; (4) the sequencer and distribution layer that deals with the ordered delivery to downstream subscribers; and (5) the consumer application layer that deals with strategy logic or the generation of derived data products.

Two co-located servers were used as the testbed of this paper whereby they were linked by a 100 Gbps Mellanox connectX-6 InfiniBand/Ethernet fabric. The publisher node was a recording of NASDAQ ITCH 5.0 messages at line rate with the pktgen traffic generator of DPDK. Each of the measurements was made using cycle-accurate RDTSC timestamps, cross-correlated with PTP to obtain figures of latency on the wall-clock in nanoseconds. The experimental conditions were repeated in 60 seconds bursts at message rates of 500 K to 5 M messages per second and statistics recorded on a minimum of 10 million messages. Incremental optimization was then introduced and latency budget at each layer was measured independently until a fully optimized configuration was constructed.

4. Techniques of Latency Optimization

4.1. Network-Layer Optimizations

The traditional Linux TCP/IP networking stack adds latency in a variety of ways: interrupt service routines (ISRs) which run on packet arrival, data copies between kernel and user-space buffers, overhead in socket layers, and lock contention in common data structures in the kernel. In a profiling using the baseline system, it was found that nearly 1822 µs of the end-to-end latency budget is consumed by network stack traversal and the remainder is consumed by the application [15]. As it is measured, DPDK-based reception decreased median latency in the network-layer by 23.7 times, decreasing the 19.4 µs median network-layer latency to 0.82 µs, at the price of 100% utilization on the reserved polling core. UDP multicast on an Ethernet fabric is the most prevalent protocol used in the case of multicast market data feeds. UDP also removes TCP flow-control and retransmission equipment along with its acknowledgment which cannot operate in microsecond delivery specifications. It does recovery of packet loss when required in application layer by sequence number gaps and a separate retransmit channel. PTP-disciplined NICs permit hardware timestamping when a physical frame is received, which removes software timestamping jitter of 2500nmassy systems estimated at 2500nmassy. RDMA (RoCEv2 in particular) was considered to be used in unicast conditions including direct market access (DMA) order channels and private consolidated feeds. RDMA allows the NIC to pass the incoming data to a registered application memory area skipping the OS kernel entirely. We have measured one-way median end-to-end latency of 310 ns at 10 Gbps, with 99th percentile values of 640ns, which are within the published vendor specifications. The main limitation of RDMA is that it needs lossless fabric (Priority Flow Control) and it is sensitive to buffer depletion in burst conditions.

Also worth mentioning are topology optimizations: optimization of the number of switch hops between the exchange gateway and the NIC of the consumer helps reduce both the propagation and switching latency. Some co-location facilities use optical bypass switches with switching times below 100ns to enable direct fiber connectivity with important paths.

4.2. Optimizations at the Software Level

Latency due to software causes is also prominent even with a kernel-bypass network stack assuming the application layer is designed poorly. There are four software-level methods discussed; real-time scheduling, CPU affinity and NUMA awareness, lock-free data structures, and memory allocation strategies. Together with an asynchronously executable real-time kernel patch (PREEMPT_RT), such an approach can substantially minimize worst-case inter-arrival time jitter of the polling loop. CPU affinity seals threads with latency-critical code on separate cores which are not part of the Linux scheduler domain (by using the isolcpus kernel parameter) and which are not interrupt-affinity (by using the irqbalance service). NUMA-conscious memory allocation, which makes sure data structures co-located on the NUMA node with the CPU making access to them, removes the 90-150 ns inter-socket memory access penalty that can be seen with dual-socket servers [14]. Our testbed showed that NUMA-local allocation minimized the average memory access latency with the order book structure by 124 to 38ns. False sharing between producer and consumer cache lines is avoided by the mechanical sympathy of the Disruptor, namely its program series of cache-line-aligned sequence slots.

A comparative summary of the effects of scheduling strategies on the end-to-end message latency in our experimental testbed is provided in Table 1.

Table 1. Latency Comparison of Os Scheduling Strategies

Strategy	Median Latency (µs)	99th Pct. (µs)	Jitter (µs)	Notes
Default Linux SCHED_OTHER	45.0	210.0	165.0	High context-switch noise
SCHED_FIFO (RT kernel)	3.8	8.2	4.4	Requires kernel tuning
CPU Pinning + NUMA	2.1	4.9	2.8	Per-core isolation
DPDK Poll Mode	0.8	1.4	0.6	Kernel-bypass
RDMA / InfiniBand	0.3	0.6	0.3	Hardware NIC offload

4.3. Hardware Acceleration

Field-Programmable Gate Arrays (FPGAs) allow minimization of latency by executing protocol parsing, symbol table look up, order book maintenance and so forth in specialized hardware pipeline stages instead of sequentially running CPU instructions. On-chip BRAM based hash table lookups used the common case of 1-clock-cycle access to a hash table. Mean time to end-to-end on last byte of UDP payload arrival to DMA write completion was 72 ns with 98ns at 99th percentile. The FPGA has a latency reduction of 11.4x that of a software feed-handler running on a single isolated CPU core and using DPDK (median 820 ns), at the same rate of message delivery. SmartNICs form an interesting engineering trade-off in applications in which FPGA development cost is prohibitive.

Time Sensitive Networking (TSN) extensions to Ethernet (IEEE 802.1Qbv and 802.1Qav) are currently becoming a standard deterministic fabric standard in financial networks. Initial market data results [14] indicate that on mixed-workloads (market data and risk system heartbeats) 99th - percentile improvements of 35 45% over traditional priority-queued Ethernet are possible. Encoding format of the message used directly and is not always adequately considered to affect latency. Table 2 performance benchmarks 5 serialization formats on the feed-handler workload of our testbed. Messages are a normalized event of AddOrder (symbol, side, price, quantity, order ID) that is a representative of order book update traffic.

Table 2. Serialization Format Benchmark

FIX/FIXML	4.20	5.80	512	~190 K
SBE (Binary)	0.48	0.52	64	~2.1 M
Protobuf	1.10	1.30	128	~870 K
Cap'n Proto	0.31	0.28	72	~3.2 M
Flatbuffers	0.29	0.22	96	~3.5 M

The decoder is simply a slim translucent cover of a crude memory pointer. Cap'n Proto and FlatBuffers also support even lower encode latency by using zero-copy serialization the in-memory object layout is the same as the wire format, therefore encoding only requires assigning fields to an existing buffer. Message size also should be considered in protocols. Smaller payload sizes minimize the NIC transmission time and cache size, which favors consumers at the downstream [16]. The fixed-width schema used by SBE can sometimes render larger messages than the variable-length formats when the field is sparsely populated, but when the field is densely populated as in the case of order book updates, SBE always renders the smallest payloads taking into consideration the framing overhead.

At the transport-protocol layer, protocol offload engines are found on state-of-the-art NICs (UDP checksum offload, Large Receive Offload) which save additional host CPU cycles per message and the cycles saved may be used by the protocol processing cores to execute application logic [17].

4.4. Feed Handler Middleware and Message Queue Optimizations

The feed handler layer occupies the position between the transport layer and the consumer application and it is meant to decode raw exchange messages, normalize them into an internal canonical format, and make them available through an inter-process or intra-process queue. Using the principles of mechanical sympathy, the LMAX Disruptor, an implementation of an inter-thread messaging library that uses a ring-buffer, removes the compare-and-swap (CAS) overhead of more traditional implementations of concurrent queue with a modulus division through the use of a power of two as the ring-buffer size, and the cache-line alignment of sequence slots to prevent false sharing. On our testbed, the substitution of a mutex-guarded queue with the Disruptor in a normalized feed dispatcher lowered P99 intra-process delivery latency by 4.1 to 0.7 µs at 2 M messages/s sustained throughput, compared to benchmarks of the same in the original Disruptor technical paper [19].

The thread-local storage (TLS) memory pools are memory pools of fixed-size message objects that are reallocated at system initiation and recycled by the use of free-list, thus avoiding the allocator on the hot path altogether. A dispatch table reduced the means of dispatching messages on the Skylake microarchitecture on our testbed by an average of 14 ns to 3ns, which is a non-trivial reduction given that millions of messages per second. Calculation of widely used derived values, such as order-book mid-price and spread, at the feed handler level and not the consumer level is an additional effort to reduce the exposure on latency to the downstream strategy logic, and can be more effectively applied in fan-out architectures where a single normalized event can be sent to multiple subscribers concurrently.

4.5. Co-location and Edge Computing

Physical co-location with the exchange to the matching engine is arguably the most significant latency optimization strategy to market data consumers since it minimizes the uncontrollable propagation delay portion of end-to-end latency. Optical fiber offers a speed of light (about 200 km/ms), which is slower than the speed of sound, but is still many times faster than the delay in software layers of an exchange in London and a consumer in Frankfurt (~1,200 km), the lowest achievable delay is about 6ms, and is many orders of magnitude faster than any software layering.

The major exchange operators provide co-location services in the same data center as the matching engine. Bright examples are NYSE Euronext facility in Mahwah, the NASDAQ data center at Carteret, and the complex at Basildon run by the London Stock Exchange. In rack-level co-location, the length of cross-connecting cables is minimized to tens of metres, and propagation delay is decreased to nanoseconds. Co-location to the multicast feed of a matching engine is usually connected with a dedicated cross-connect with a known one-way latency of 200-500ns.

Edge computing is an extension of co-location to consumers of data distributed geographically. An ED node at the exchange can publish normalized feeds at the exchange and send them to remote consumers via directed paths over optimized wide-area network (WAN) routes using microwave or millimeter-wave wireless connections. The microwave route between Chicago and New York blazed by the commercial operators, improves propagation latency by more than 7 ms (fiber) down to about 4ms by taking advantage of line-of-sight propagation in the air (lower refractive index than glass). Clock synchronization is a major difficulty in distributed co-location architecture. The latency and sequence number recovery algorithms rely on less than microsecond clock synchronization between geographically distributed nodes. PTP extension White Rabbit, a PTP extension that was created at CERN, builds upon this feature and supports deterministic distributed sequencing to sub-nanosecond precision over optical fiber [18].

5. Performance Evaluation

In this section, the consolidated end-to-end latency measurements of the reference system were provided in six configurations:(C0) fully optimized baseline; (C1) DPDK kernel- bypass only; (C2) C1 + SCHED_FIFO and CPU affinity; (C3) C2 + LMAX Disruptor and memory pools; (C4) C3 + SBE serialization; and (C5) C4 + FPGA feed handler.

End-to-end latency is defined as the duration the UDP market data payload has taken between the last byte of the payload as being received by the NIC through the receive timestamp counter and the consumer application callback returning after using the strategy logic. All numbers are median (P50), 99th percentile (P 99) and worst-case (P 99.9) latency with 10 million messages injected at a constant rate of 2 M msg/s. SCHED_FIFO and CPU affinity (C2) result in P50 = 14.8 µs and P99 = 19.2 µs - a drastic reduction in tail latency, and shows that scheduling jitter was the main cause of P99 in C1.

The P50 = 3.9 µs and P99 = 5.8 µs are obtained by replacing the mutex-guarded queue with the Disruptor and switching to the use of TLS memory pools (C3). A further change to SBE serialization (C4) yields a P50 of 1.7 u s and a P99 of 2.6 u s, thus confirming that at this stage the remaining fixed overhead of FIX parsing had become the biggest contributor.

The optimized configuration with the FPGA feed handler (C5) has a P50 = 0.78 µs, P99 = 1.38 µs and P99.9 = 1.74 µs. This is a 57.9-fold improvement and a 152-fold improvement in median and 99th-percentile latency compared to the baseline, correspondingly. The FPGA fully offloads feed parsing to the host CPU which is freed to commit all its pipeline to strategy execution.

Analysis of throughout-latency trade-off The analysis of throughput-latency shows that the throughput C1-C4 are characterized by a more-or-less linear throughput latency relationship, even at high throughput (3 M msg/s) by which point P99 starts increasing rapidly because of ring buffer back-pressure. Configuration C4 (software-only optimizations) achieves P50 -1.7 u s at a fraction of the cost and much reduced operational complexity, which is a realistic Pareto-optimal option in the case of smaller market participants.

6. Discussion

The findings prove that there is no single optimization layer; the reduction of latency is a multiplicative process of the improvement of each layer. Given pure application of network-layer optimization (C1), 58 of the original median latency is retained in the software stack. On the other hand, software optimization (no bypass of the kernel, hypothetical C0+C2) results in P99 of approximately 40 µs, which is not as good as C0 nor within the target range. The hardware engineering skills necessary to develop FPGA are not typical in an average trading technology group. Organisations using such methods need to invest in a detailed monitoring infrastructure, that is, hardware-timestamped latency probes every layer in to identify regressions caused by updates to the kernel, changes in configuration, or change of traffic patterns.

The regulatory factors are becoming more and more topical. The MiFID II in Europe and Reg NMS in the United States have demands concerning the level of fairness and transparency of access to market data. The methods of co-location fee structuring and the level of feed latency are both under regulatory review and companies must keep detailed audit trails of the timestamps when orders were placed, at sub-microsecond accuracy. Regulatory compliance in the latency sensitive trading operation of PTP-synchronized hardware timestamps requires the hardware timestamps of Section IV.E.

Future work will explore the use of new technologies: (i) CXL (Compute Express Link) memory pooling in latency-transparent shared order book structures between multiple host CPUs; (ii) P4-programmable switches towards in-network feed normalization at the switching tier; and (iii) quantum key distribution (QKD) on small low-latency private feed channels without the added burden of traditional TLS handshake protocols. Another latency-related issue in production deployments that is rarely mentioned, yet of critical concern, is that the latency characteristics need to be operationally stable and reproducible over the long term. It is also crucial to perform continuous profiling of latencies based on hardware timestamps at each pipeline layer and not just end to end measurements to identify regressions at the individual component without interfering with live operations. The automated alerting systems that identify the percentage-based violation of the latency thresholds are highly suggested to be invested in as a part of the production deployment lifecycle.

The co-location environments that are multi-tenant in nature add extra complexity that cannot be emulated in single-tenant testbeds. Multiple co-location companies currently also provide dedicated cage deployments with explicit core isolation guarantees and physically isolated Top-of-Rack switches as a premium service. The extra expenditure on having exclusive infrastructure is to be offset by the reduction of latency variance it achieves; we have evaluated published co-location benchmark data to show that the worst-case P99.9 latency can drop by between 30 and 40 percent when cross-tenant interference is completely reduced, which is a significant benefit of strategies where tail-latency exposure is directly proportional to adverse selection risk. This is a considerable contribution to TCO to those participants who are involved in large-scale feed handler farms. This trade-off can be alleviated by adaptive-polling-to-interrupt switching methods, used in the DPDK power management library, which incurs a quantifiable latency cost at the time of traffic resumption, but in periods of low traffic. This energy cost is what should be quantified and managed as part of the business case to deploy kernels bypass.

7. Conclusion

In this paper, the empirically based investigation of the latency optimization of near real-time market data systems has been introduced. A hierarchical taxonomy of network infrastructure, operating system scheduling, and application data structures, hardware acceleration, and physical co-location was suggested and confirmed on a testbed reconfigured to simulate NASDAQ ITCH 5.0 traffic.

The major results are three times. To start with, a cumulative implementation of all optimization layers has decreased median end-to-end message latency by 45.2 u s (standard Linux stack) to 0.78 u s (FPGA + kernel-bypass +

RT scheduling + Disruptor + SBE), which is 57.9 times lower. Second, latency, which is the measure that is most germane to worst-case risk exposure, is improved by 152x, with the 99th percentile dropped to 1.38 µs. The cross-layer reinforced nature of optimizations provides the worth of holistic system design. Players in the market who want to reduce information latency must consider it as a full-stack engineering issue and not merely a one-layer procurement decision. The methods outlined in this paper can be used in any application of latency-constrained streaming data, such as risk management systems, real-time surveillance systems, and smart-order-routing engines. In addition to the financial field, the approaches outlined in this paper can be applied to any task that needs high-throughput with latency constraints in its processing of streaming data. The comparison of serialization format (Table II) especially offers practical advice that could be used on any binary based messaging system whether it is a money related protocol or not.

The paper also identifies a number of unanswered research questions that should be studied. The analysis of CXL memory pooling of latency-transparent shared order book structures, implementation of P4-programmable switches of in-network feed normalization, and use of Quantum Key Distribution (QKD) of securing low-latency private feed channels are the immediate future directions in which the hardware and protocol development can be expected to bring the achievable latency frontier to meet the physical speed-of-light boundary. Besides, the time-dependency of latency properties (and hence regression identification by hardware-timed profiling) is an under-investigated issue on academic publications in spite of its high priority in production applications. The results of this paper can thus be used as a point of reference to both the present practice as well as a guide to the future generation of ultra-low-latency market infrastructure.

8. Recommendations

The following are the recommendations that can be given based on the experimental results and the analysis made in this paper to practitioners, system architects, and technology managers who are dealing with the latency sensitive trading environments.

Choose a Full-Stack Optimization Strategy at the very beginning: The configurations considered in this paper (C0 to C5) offer a realistic incremental roadmap which can be used by the team without investing the entire FPGA capital cost upfront.

The First Major Intervention should be Kernel-Bypass Networking: This optimization does not require any hardware implementation and should be the initial upgrade of any company working with commodity servers. Special attention should be paid to allocating the specific polling core and to setting the interrupt affinity so as to prevent the unexpected addition of jitter by other system activities.

Move to Zero-Copy Feed Handler Encoding Serialization Formats: In organizations that continue to use FIX encoding in the hot path in text format, an upgrade by using SBE Binary or the zero copy format is one of the most lucrative investments in the application layer. FIX-to-FIX teams interested in migrating to SBE are encouraged to use its standardized schema and regulatory interoperability but developers of proprietary internal feeds can use FlatBuffers as it has a greater throughput advantage and slightly larger message size at the cost of sparse fields.

References

[1] S. Vinnakota, “Optimizing Real-Time Bidding (RTB) Latency in Ad Exchanges: A Comprehensive Analysis,” *Journal of Computer Science and Technology Studies*, vol. 7, no. 7, pp. 640–650, Jul. 2025, doi: https://doi.org/10.32996/jcsts.2025.7.7.72.

[2] A. Irshad, “Latency Optimization in Edge vs. Cloud Computing: A Comparative Study for Real-Time Applications,” *International Journal of Business & Computational Science*, vol. 1, no. 1, 2024, doi: https://doi.org/10.0786/37rr2j62.

[3] W. Afua, N. A. O. Ajayi-Nifise, G. Bello, S. Tubokirifuruar, N. O. Odeyemi, and N. T. Falaiye, “Algorithmic Trading and AI: A Review of Strategies and Market Impact,” *World Journal of Advanced Engineering Technology and Sciences*, vol. 11, no. 1, pp. 258–267, Feb. 2024, doi: https://doi.org/10.30574/wjaets.2024.11.1.0054.

[4] H. Le and A. Gregoriou, “How do you capture liquidity? a review of the literature on low-frequency stock liquidity,” *Journal of Economic Surveys*, vol. 34, no. 5, pp. 1170–1186, Jul. 2020, doi: https://doi.org/10.1111/joes.12385.

[5] S. V. Cortés-Calderón, M. D. López-Rodríguez, A. Jiménez-Aceituno, A. J. Castro, and M. Mancilla-García, “Contributions of Net-Map to sustainability action research,” *Current Opinion in Environmental Sustainability*, vol. 75, p. 101542, May 2025, doi: https://doi.org/10.1016/j.cosust.2025.101542.

[6] E. Freitas, Assis, Pedro, D. F. H. Sadok, and J. Kelner, “Takeaways from an experimental evaluation of eXpress Data Path (XDP) and Data Plane Development Kit (DPDK) under a Cloud Computing environment,” *Research, Society and Development*, vol. 11, no. 12, p. e26111234200-e26111234200, Sep. 2022, doi: https://doi.org/10.33448/rsd-v11i12.34200.

[7] R. Steinert *et al.*, “Service Provider DevOps network capabilities and tools,” 2026. [Online]. Available: https://arxiv.org/abs/1510.02818

[8] X. Wang *et al.*, “A data privacy protection method for infectious disease prediction models with balanced training speed and accuracy,” *Scientific Reports*, vol. 16, no. 1, Feb. 2026, doi: https://doi.org/10.1038/s41598-026-38906-9.

[9] J. Dokulil, S. Benkner, and J. Yaghob, “The Open Community Runtime on the Intel Knights Landing Architecture,” *Lecture Notes in Computer Science*, pp. 801–813, 2017, doi: https://doi.org/10.1007/978-3-319-65482-9_65.

[10] M. Friedman, M. Herlihy, V. Marathe, and E. Petrank, “A persistent lock-free queue for non-volatile memory,”

ACM SIGPLAN Notices, vol. 53, no. 1, pp. 28–40, Feb. 2018, doi: https://doi.org/10.1145/3200691.3178490.

[11] O. Arcas-Abella *et al.*, "Hardware Acceleration for Query Processing: Leveraging FPGAs, CPUs, and Memory," *Computing in Science & Engineering*, vol. 18, no. 1, pp. 80–87, Jan. 2016, doi: https://doi.org/10.1109/mcse.2016.16.

[12] L. Rosa, L. Foschini, and A. Corradi, "Empowering Cloud Computing With Network Acceleration: A Survey," *IEEE Communications surveys and tutorials/IEEE communications surveys and tutorials*, pp. 1–1, Jan. 2024, doi: https://doi.org/10.1109/comst.2024.3377531.

[13] S. Popic, D. Pezer, B. Mrazovac, and N. Teslic, "Performance evaluation of using Protocol Buffers in the Internet of Things communication," Oct. 2016, doi: https://doi.org/10.1109/sst.2016.7765670.

[14] P. Shantharama, A. S. Thyagaturu, and M. Reisslein, "Hardware-Accelerated Platforms and Infrastructures for Network Functions: A Survey of Enabling Technologies and Research Studies," *IEEE Access*, vol. 8, pp. 132021–132085, 2020, doi: https://doi.org/10.1109/access.2020.3008250.

[15] K. Yasukata, M. Honda, D. Santry, and L. Eggert, "StackMap: Low-Latency Networking with the OS Stack and Dedicated NICs," Berkeley, Calif.: USENIX Association, 2016.

[16] H. N. Schuh *et al.*, "CC-NIC: a Cache-Coherent Interface to the NIC," pp. 52–68, Apr. 2024, doi: https://doi.org/10.1145/3617232.3624868.

[17] S. Arslan, S. Ibanez, A. Mallery, C. Kim, and N. McKeown, "NanoTransport," pp. 13–26, Oct. 2021, doi: https://doi.org/10.1145/3482898.3483365.

[18] M. Lipiński, "The Inclusion of White Rabbit into the Global Industry Standard IEEE 1588," *JACoW*, vol. ICALEPCS2021, pp. 126–131, 2022, doi: https://doi.org/10.18429/JACoW-ICALEPCS2021-MOPV011.

[19] O. Kode and T. Oyemade, "Analysis of Synchronization Mechanisms in Operating Systems," 2024. [Online]. Available: https://arxiv.org/abs/2409.11271