RC24972 (W1004-025) April 9, 2010
Computer Science

IBM Research Report

Streaming, Low-latency Communication in On-line Trading Systems

Hari Subramoni[1,2], Fabrizio Petrini[1], Virat Agarwal[1], Davide Pasetto[3]

[1]IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598 USA

[2]The Ohio State University
Columbus,OH 43210 USA

[3]IBM Computational Science Center
Dublin, Ireland

Streaming, Low-latency Communication in On-line Trading Systems

Hari Subramoni[1,2] Fabrizio Petrini[1] Virat Agarwal[1]
Davide Pasetto[3]

[1]IBM TJ Watson, Yorktown Heights, NY 10598 USA
[2]The Ohio State University, Columbus, OH 43210 USA
[3]IBM Computational Science Center, Dublin (Ireland)
fpetrin@us.ibm.com, viratagarwal@us.ibm.com, pasetto_davide@ie.ibm.com

Abstract

This paper presents and evaluates the performance of a prototype of an on-line OPRA data feed decoder. Our work demonstrates that, by using best-in-class commodity hardware, algorithmic innovations and careful design, it is possible to obtain the performance of custom-designed hardware solutions.

Our prototype system integrates the latest Intel Nehalem processors and Myricom 10 Gigabit Ethernet technologies with an innovative algorithmic design based on the DotStar compilation tool. The resulting system can provide low latency, high bandwidth and the flexibility of commodity components in a single framework, with an end-to-end latency of less then four microseconds and an OPRA feed processing rate of almost 3 million messages per second per core, with a packet payload of only 256 bytes.

1 Introduction

Financial exchanges communicate with its members through consolidated 'market data feeds'. These feeds are composed of event messages that provide in real time the current status of the market. The event messages contain reports of completed trades, bid and ask prices of various financial instruments, trade correction, trade cancellation, and other status updates. Financial institutions that subscribe to these market data feeds process incoming messages to identify profitable trade opportunities through algorithmic trading, as well as to manage and update risks on trade books. A typical market data processing system at these institutions consists of several functional units that receive event messages, publish financial data of interest to their subscribers (such as traders at workstations), and route trade data to various exchanges and other venues. This system, known as a *ticker plant*, is shown in Figure 1a. With market data feeds already in the gigabit per second range, and growing exponentially, a reliable market data system that can process huge volumes of data at low latency, is becoming increasingly critical to the success of the financial institutions both in the U.S. and abroad, and is the main focus of this paper.

1.1 Skyrocketing Data Rates

U.S. exchanges that allow trading of securities options have been authorized under the Securities Exchange Act of 1934 to agree to a "Plan for Reporting of Consolidated Options Last Sale Reports and Quotation Information". This reporting of consolidated options last sale reports and quotation information is administered by a committee called Options Price Reporting Authority or OPRA [13]. OPRA is the securities information processor that disseminates, in real-time on a current and continuous basis, information about transactions that occurred on a large number of quoted instruments in the options markets. This information is also broadcast by the exchange, and is used by traders at financial institutions to know the state of the market in real time. Based on this information, traders identify 'profitable' opportunities, either through algorithmic trading using analytical models or detecting patterns in the market, through instrument arbitrage, or through managing and reducing risk. Fueled by the growth of algorithmic and electronic trading, the global options and equities markets are expected to produce an average of more than 128 billion messages/day by 2010, rising from an average of more than 7 billion messages a day in 2007 [21]. In the options market, the OPRA consolidated feed for all US derivatives business represents a very significant portion of market data in the national market system. Figure 1b shows that the OPRA market data rates have dramatically increased over the course of the past 4 years, approaching a peak of 1 mil-

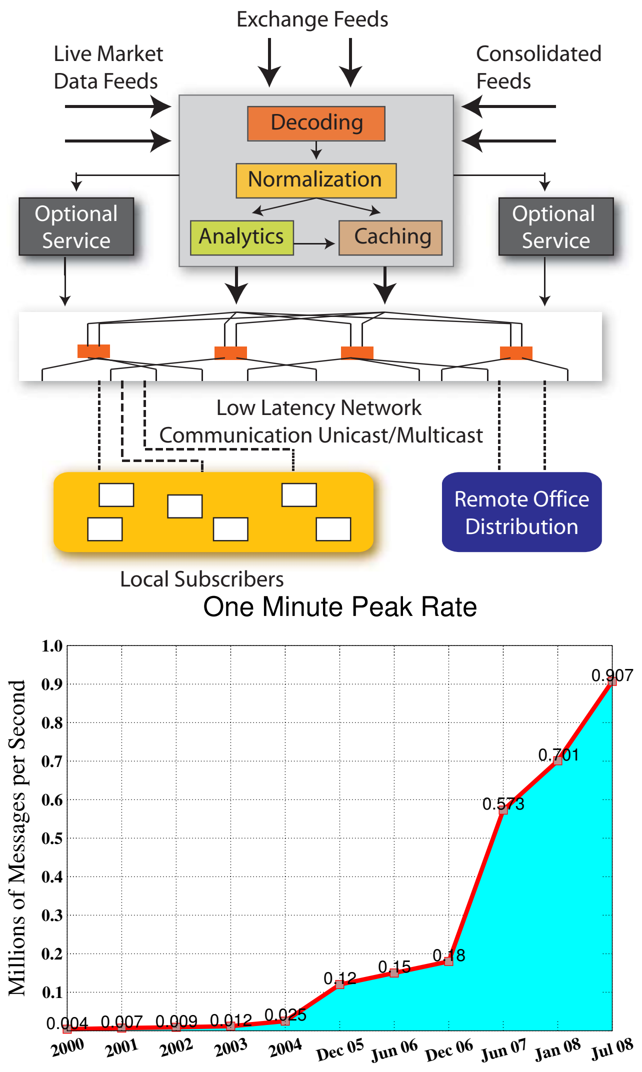


Figure 1: (a) High-Level Overview of a Ticker Plant, (b) OPRA market peak data rates

lion messages per second. The traffic projection for OPRA alone is expected to reach an average of more than 14 billion messages/day at the end of 2010 [13].

1.2 Role of High Performance Computing

The growing message flow is stretching the raw data rates at which the exchange transmits the market feed to its members. This data rate, that was less than a megabit per second a decade ago has already reached hundreds of megabits per second and is rapidly approaching gigabits per second. The increasing data rate creates pressures on the network infrastructure between the exchanges and the financial institutions. This problem is being alleviated by using compressed data formats that are further increasing the processing requirements both at the exchange and receiving institutions. It is important to note that low latency is critical, especially with the increasing competition as well as diminishing profit margin. How fast a trading system can respond to the market will determine who wins and who loses, even a few microseconds gap in latency is enough to make the difference. For instance, the opportunity of an arbitrage (difference in prices between markets) only exists until the first trader is able to exploit it.

Designing a highly responsive trading solution is a challenging problem because it requires both

1. a high-throughput system able to process millions of messages per second, extensible to tens of millions in the future, and

2. a low-latency system with response times of a few microseconds.

We believe that delivering such a solution entails complex research that includes high-level software design, lightweight communication libraries, top-of the line hardware, and an overall design that takes into consideration that each level of the system pipeline processes high bandwidth at low latency. Together with the increased performance, data centers are also encountering typical problems in high performance computing: the complexity of developing, testing and validating parallel software. The need is to reduce power consumption in the processing units that are often "co-located" near the data feeds to minimize the communication latency and reduce floor space requirement, which typically comes at a high premium. For these reasons, the financial community is demanding solutions that are extremely fast, easy to program, and adaptable to dynamically changing requirements.

1.3 Related Work

On-line trading systems pose a unique challenge in terms of computational and communication capabilities. Not surprisingly, many of the technologies that are commonly used in high performance computing are rapidly appearing in the data centers of many financial institutions [9, 8, 3]. Field Programmable Gate Array (FGPA) based solutions [10, 7, 4, 1] employ hardware acceleration to achieve low latency, but are difficult to program and inflexible to dynamically changing environments. For a similar workload, a Cell/B.E. based solution [15] achieved a latency in the order of tens of microseconds. IBM Websphere [6] uses software solutions and commodity hardware and to achieve low latency at the current data rates of the market. Other vendors are designing high performance message routers [20], switches [5, 22], and Network Interface Controllers (NICs) [11] to address the needs of the financial community.

1.4 Commodity vs Special Purpose Solutions

Together with speed, trading systems need to provide a high a degree of interoperability with existing data centers and proprietary software that embody the trading strategy of a financial institution. This extra requirement poses a further design challenge. On the one hand, special purpose systems that rely on the most advanced communication and computation technologies can provide a significant performance advantage. On the other hand, legacy software and the compatibility with various standards require the adoption of a commodity technology.

In this paper we analyze a very specific area of this large design space, trying to push the limit of commodity technology. More specifically, we combine two high-end commodity technologies, 10 Gigabit Myricom Ethernet and the Intel Nehalem processor, to evaluate the performance of a prototype on-line trading system.

1.5 Contributions

The primary contribution of this paper is the design, implementation and evaluation of a trading system that is built out of commodity components, Intel x86 processors and 10 Gigabit Ethernet. The proposed system relies on previous work [2] and integrates the OPRA decoder in a fully functional system that includes the network communication between processing nodes. In fact, the network and the network stack play an important role, and are increasingly becoming the bottleneck in systems that can handle the OPRA protocol operating at the sub-microsecond level.

In our prototype we have used the Myricom Myri-10 GigE Network Interface Cards (NICs) [11] and DBL and MX, optimized libraries that allow OS-bypass data communication, and our novel FAST OPRA feed decoder based on our home-brewed DotStar pattern processing engine [14]. The seamless integration of all these components delivers a prototype trading system that can streams OPRA messages with an end-to-end latency of less than 4 microseconds. The system is able to deliver a high bandwidth, reaching 3 million messages per second per core, with a packet payload size of only 256 bytes.

We also present a preliminary analysis of the Myrinet communication protocols and highlight interesting trade-offs between latency and bandwidth of the Myrinet network interface card. Thanks to our high-level compilative approach based on DotStar our system can be easily retargeted to dynamically changing OPRA formats, as well as other market data feeds [2].

Overall, the paper demonstrates that by using best-in-class commodity hardware, algorithmic innovations and careful algorithmic design it is possible to obtain the performance of custom-designed hardware solutions.

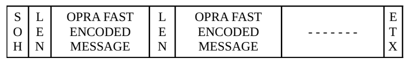
One evident limitation of this work is that our end-to-end latency does not take into account the line arbitration that is performed on the incoming market feeds. Using line arbitration, the redundant input streams from the exchange can be exploited to reconstruct the sequence of message and minimize latency due to packet loss.

2 OPRA feed decoding

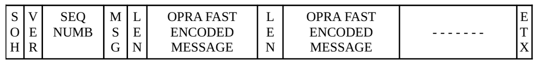
An essential component in ensuring the timely reporting of option equity/index and every other transaction is the OPRA IP multicast data stream. OPRA messages are delivered through the national market system with a UDP-based IP multicast [19]. The options market data generated by each participant is assembled in prescribed message formats and transmitted to the appropriate TCP/IP processor address via participant's private communications facility. As each message is received, it is merged with messages received from other participants, and the consolidated message stream is transmitted simultaneously to all data recipients via their private communications facilities. Each message is duplicated and delivered to two multicast groups. OPRA messages are divided into 24 data lines (48 when counting redundant delivery) based on their underlying symbol. Multiple OPRA messages are encapsulated in a block and then inserted in an Ethernet frame. The original definition of OPRA messages is based on an ASCII format [17], which uses only string based encoding and contains redundant information. With the growth of data volume, a more compact representation for messages was introduced: OPRA FAST (FIX Adapted for STreaming) [16, 18].

The techniques used in the FAST protocol include implicit tagging, field encoding, stop bit, and binary encoding (see Table 1). Implicit tagging eliminates the overhead of field tags transmission. The order of fields in the FAST message is fixed and thus the meaning of each field can be determined by its position in the message. The implicit tagging is usually done through XML-based FAST template. The presence map (PMAP) is a bit pattern at the beginning of each message where each bit is used to indicate whether its corresponding field is present. Field encoding defines each field with a specific action, which is specified in a template file. The final value for a field is the outcome of the action taken for the field. Actions such as "copy code", "increment", and "delta" allow FAST to remove redundancy from the messages. A stop bit is used for variable-length coding, by using the most significant bit in each byte as a delimiter. FAST uses binary representation, instead of text string, to represent field values. OPRA is an early adopter of the FAST protocol for reducing the bandwidth needed for

OPRA messages.



(a) Version 1.03



(b) Version 2.0

Figure 2: OPRA FAST encoded packet format.

Figure 2a shows the format of an encoded OPRA version 1 packet. Start of Header (SOH) and End of Text (ETX) are two control characters that mark the start and end of the packet. One transmission block can contain multiple messages, where a message is a unit of data that can be independently processed by the receiver. In OPRA FAST version 2 (see Figure 2b) there is a header after SOH and before the first message to further reduce redundant information. The first byte of an encoded message contains the length in bytes and is followed by the presence map. For example, presence map 01001010 means field 1, field 4 and field 6 are present. The type of each field is specified by message category and type. Data fields that are not present in an encoded message but that are required by the category will have their value copied from the same field of a previous messages and optionally incremented. OPRA data fields can be either unsigned integer or string.

	Description
Category 'a'	Equity and Index Last Sale
Category 'd'	Open Interest
Category 'f'	Equity and Index End of Day Summary
Category 'k'	Index and Stock Quotes
Category 'C'	Administrative
Category 'F'	FCO End of Day Summary
Category 'H'	Control
Category 'O'	FCO Last Sale
Category 'U'	FCO Quote
Category 'Y'	Underlying Value Message
default	Contains Text value

Table 1: OPRA Message Categories with Description

2.1 A Characterization of the OPRA Protocol

OPRA market data feeds are transmitted in a set of 24 channels. In this experimental section, we assume that each channel can inject messages at full speed by storing the OPRA feeds in main memory, and therefore is not the bottleneck of the decoder. While this hypothesis may not be realistic in practice, it serves the purpose of pushing to the limit the OPRA decoding algorithm and provides an upper bound.

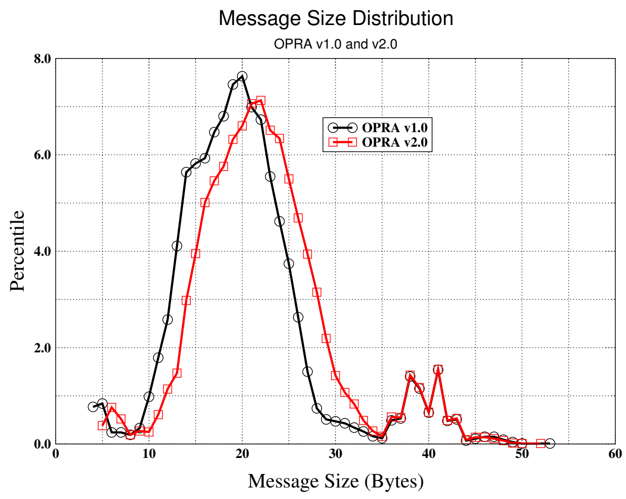


Figure 3: Message Size Distribution

OPRA packets are 400 bytes on average, each containing multiple messages that are encoded using the FAST protocol. We use real market data feeds, obtained by capturing few seconds of the network traffic, totalling one gigabyte of version 1 and 2 OPRA format for experimental analysis throughout this section.

Figure 3 gives the distribution of the message size for both OPRA version 1 and 2 format. The messages are typically distributed across the 10-50 byte range, with an average message size of 21 bytes. Thus each packet contains 19 messages on average.

Under the assumption of full injection, the feeds across the various channels have very similar data pattern and distribution and, as shown in Figure 4, they tend to have the same processing rate. Since, the performance is insensitive to the OPRA protocol version, we will consider only OPRA version 2 traces in the rest of this paper.

Figure 5 gives a distribution of the OPRA messages among 11 categories, as described earlier. We observe that 99% messages flowing in the market are category K equity and index quotes. The first field of each message after the *message length* is the PMAP, containing encoded information about the position and type of the data fields that follow. Figure 6 shows a distribution of the PMAP length, for version 2 OPRA format, and shows that a majority of the messages contain a 5 byte PMAP. The data fields can be either integer or string type, that is given as a part of the protocol specification. Each OPRA message can contain multiple integer fields, with length varying from 1 to 5 bytes. Figure 7 gives a distribution of the encoded integer field length, and shows that more than 80% of encoded integers are less than 2 bytes.

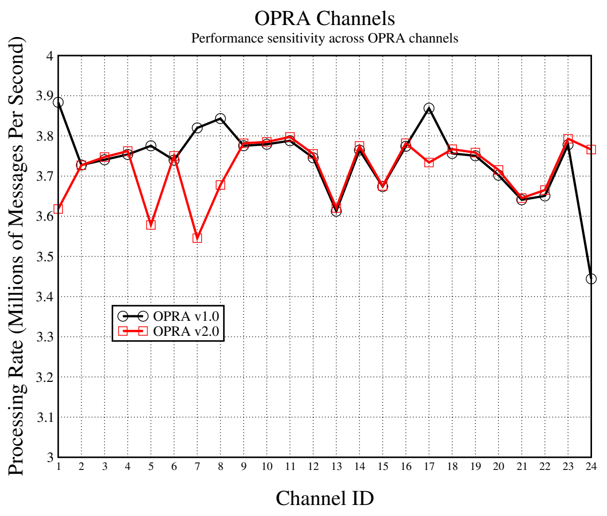


Figure 4: Channel Sensitivity: the processing rate is almost insensitive to the Channel ID and OPRA version.

3 Background

Designing an end-to-end solution to process financial market data feeds at a high bandwidth and low latency requires a state of the art Network Interface Card (NIC) that can interface the OPRA processing system with the network infrastructure. In this paper we evaluate and use the Myricom Myri-10 GigE NICs [12], that promise to deliver the performance of specialized hardware while maintaining a commodity framework. The Myri-10G NIC is internally programmable, for conventional TCP/IP and UDP/IP operation, the processors and firmware in the NIC are used for highly effective stateless offloads, resulting in wire-speed throughput with low host-CPU utilization. In this section, we give some background information about the communication protocols that can be used with these NICs.

3.1 Datagram Bypass Layer

The Datagram Bypass Layer (DBL) is a new communication protocol designed by Myricom [11] to provide low latency communication over a sockets type interface. DBL is based on the UDP/IP transport protocol and hence is not hampered by the additional overhead of flow control imposed by TCP/IP. The DBL software, along with the new generation Myri-10G [12] Ethernet NIC's enables the user to bypass the kernel and communicate directly with peer applications from the user space. The DBL receivers can operate in multiple modes - interrupt based and polling based. As seen with other communication protocols, polling based communication mechanism, though CPU intensive, provides the best communication performance. For applications that have dedicated computing resources, such mode

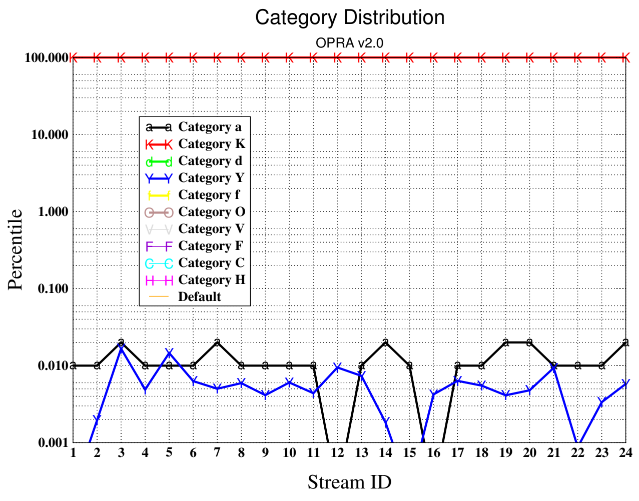


Figure 5: OPRA message category distribution

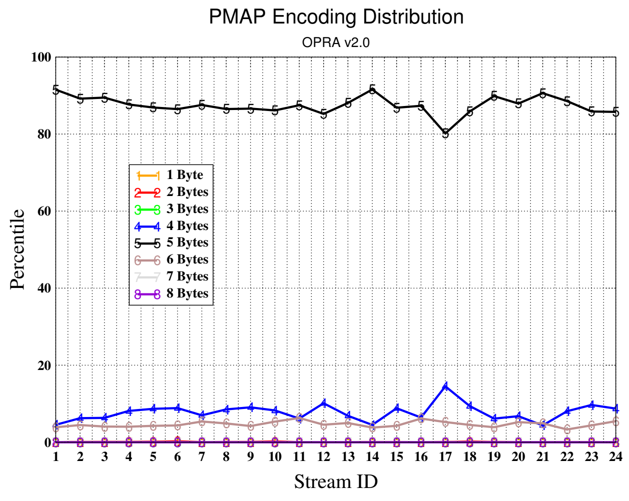


Figure 6: Encoded pmap field length distribution

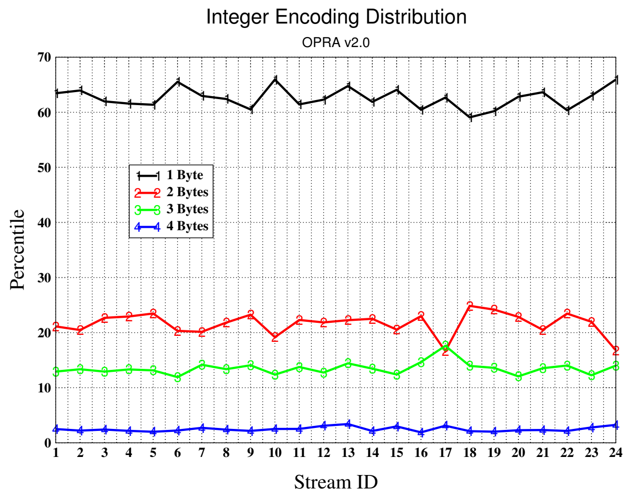


Figure 7: Encoded integer field length distribution

will give the best possible performance in terms of end-to-end latency as well as arrival pattern of packets. As DBL is based on UDP/IP, the overhead of ensuring reliable delivery and flow control falls on the user application.

3.2 Myrinet Express

Myrinet Express (MX), is a low-level communication layer for Myrinet. MX can operate in Ethernet mode (MXoE) as well as native Myrinet mode (MXoM). MX uses a completely different communication semantic as opposed to socket based communication. It also ensures reliable delivery of messages in hardware. Due to the different communication semantic, and hardware offload, MX is capable of providing much lower latencies when compared to the standard sockets style communication.

4 Experimental Results

In this section we describe the setup and evaluate the performance of our test environment. Our experimental setup consists of two hosts equipped with the Intel Nehalem series of processors with dual quad-core processor nodes operating at 2.93GHz with 48 GB RAM and a PCIe 2.0 interface. Red Hat Enterprise Linux Server release 5.2 with Linux Kernel version 2.6.28 was used on the hosts. Each host was equipped with a dual port Myri-10G card capable of operating in both Ethernet as well as the native Myrinet mode. The hosts are connected through a Fujitsu 10 Gigabit Ethernet switch. We have used DBL version 0.4.5 and an experimental version of MX in our experiments.

4.1 Microbenchmark Level Evaluation

We first look at the basic performance of the various communication protocols under evaluation - *DBL*, *MXoE*, *UDP/IP*. We use a standard ping-pong benchmark to perform the evaluation. Figures 10 (a) and (b) shows the communication performance of the various protocols in terms of latency and message rate respectively. The latency measurements include the latency of sending a message from source by reading from the memory of the system, transmitting through the NIC to the network, and receiving it at the destination through the NIC to the memory of the end system. We only look at the performance of small message sizes as that is the target range of most financial applications. We get the best latency of 2.45 microseconds with *MXoE* followed very closely by *DBL* at 3.14 microseconds. The relatively poor performance obtained with *UDP/IP* is due to the effect of interrupts and kernel context switches.

For analyzing the end-to-end system, we parse and normalize OPRA packets at the destination system. We integrate our high speed feed decoder [2] based on the Dot-

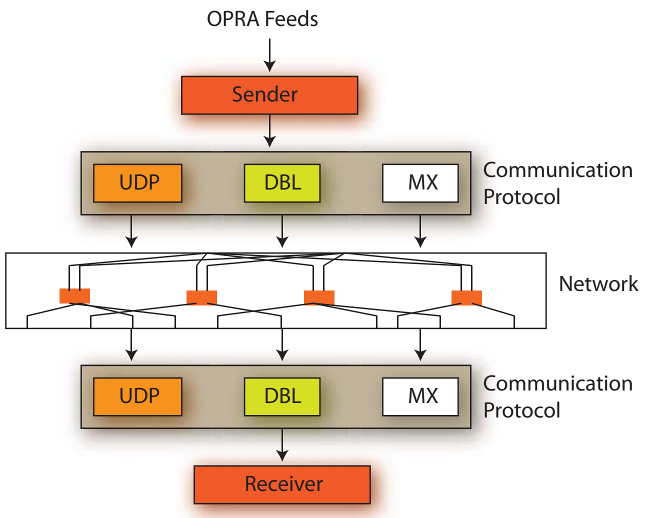


Figure 8: Communication Protocols

Star protocol processing tool, that is able to capture the essence of the OPRA structure in a handful of lines of a high-level description language. This is combined with hand-optimized actions on the various field of the OPRA message, that are triggered by the OPRA scanner. With this optimized implementation, a single Intel Xeon E5472 quad-core is able to achieve a rate of 15 million messages/second. Figure 11 shows the overhead caused by adding OPRA decoding in the flow. As we can see, the performance impact of the message decoding is minimal, between 150 and 300 nanoseconds. While maintaining an end-to-end latency below 4 microseconds both with DBL and MX protocols, our end-to-end system is able to stream, process and parse OPRA feeds at a peak rate of 3.2 million messages per second using the MX protocol, and 2.8 million messages per second using the DBL protocol. It is important to note that ours is the first result that achieves processing rates of millions of messages per second for OPRA at 4 microsecond latency with commodity technology.

4.2 Analysis of Arrival Patterns

For low latency stock market applications, the consistency with which the underlying hardware is able to deliver packets is as important as the lowest latency that can be achieved. The performance of stock market arbitration algorithms is closely tied up with the frequency of arrival of packets. In this context, we evaluate the inter arrival times of messages with the different communication protocols under evaluation. Figures 12 (a), (b) and (c) depict the frequency distribution of the arrival patterns of 99.9% of messages for *MX*, *DBL* and *UDP* respectively. The best case would be when the arrival times of most, if not all, of the

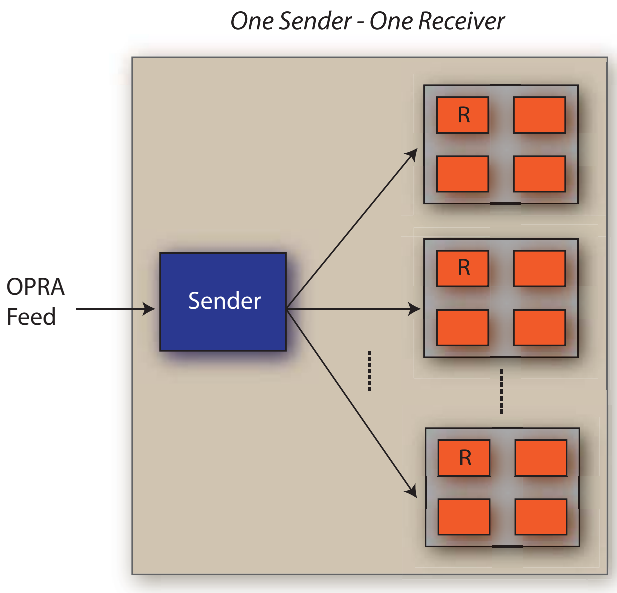
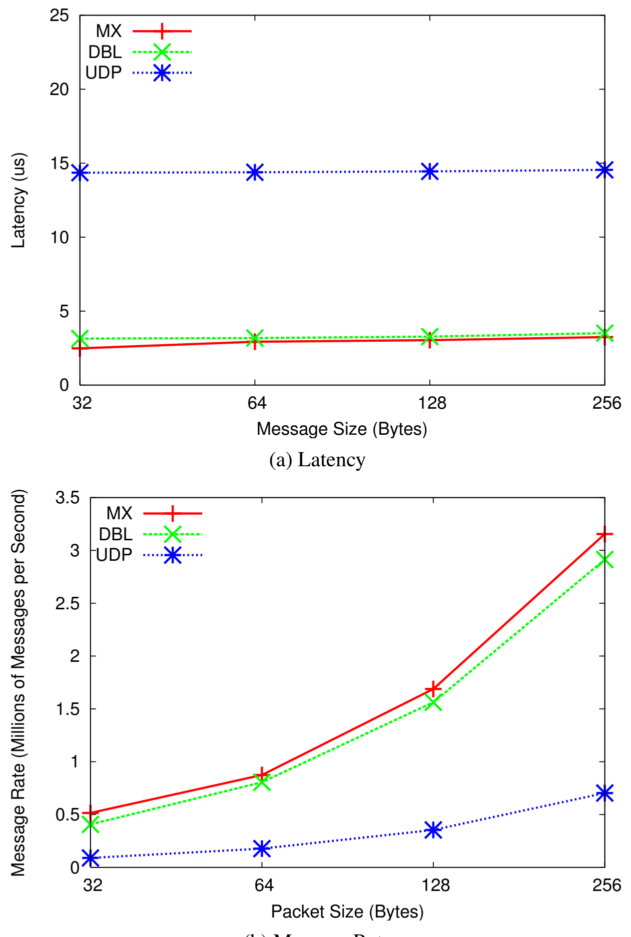


Figure 9: Communication Topology

packets is same. The frequency distribution graph for this ideal case would show up as a sharp Gaussian curve with very few outliers. As we can see, though *MX* gives minimum number of outliers, the curve does not match the ideal scenario we discussed earlier. There are multiple peaks indicating that there is some variation in the arrival pattern of the packets. The curve for *DBL* on the other hand, though has a few more outliers, forms a very sharp Gaussian curve indicating that most of the packets arrive without a lot of variation which is what we want for a stock market type application. Due to the effect of interrupts and context switches, the arrival pattern for *UDP* is not as good.

To understand the effects of load balancing on network performance, we study the performance of the communication protocols with multiple receivers. We only choose *MX* and *DBL* for this experiment as they demonstrated better performance in the previous experiments. Figures 13 (a) and, (b) shows the comparison of arrival patterns with one and two receivers for *MX* and *DBL* respectively. Contrary to our expectations, the graphs show that the arrival patterns for cases with multiple receivers is worse and have more outliers than those with just one receiver. Further investigation showed that we were hitting the NIC firmware's receive limit with just a single receiver and consequently having multiple receivers only degraded the network performance. We plan to conduct more studies on the impact of load balancing by using multiple ports in the same NIC to evaluate their performance.



(a) Latency

(b) Message Rate

Figure 10: Communication Performance

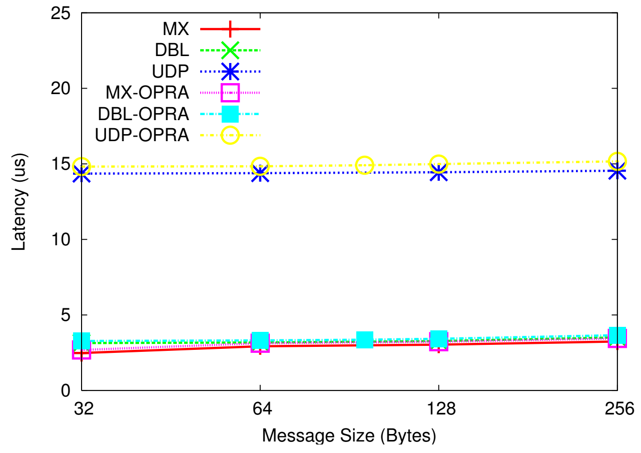
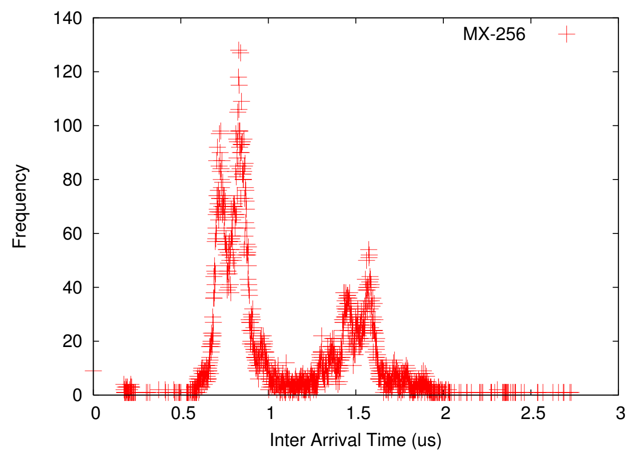
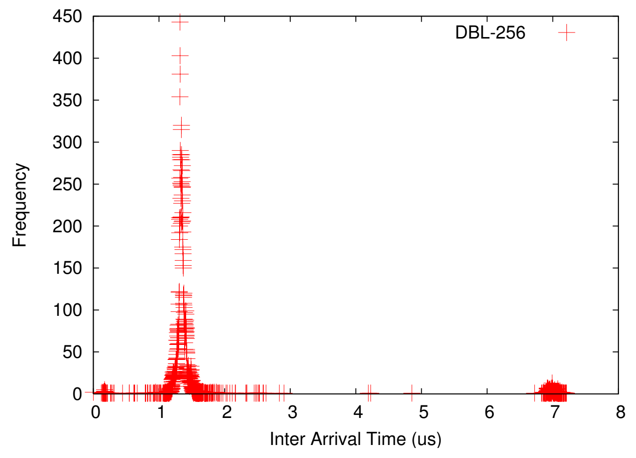


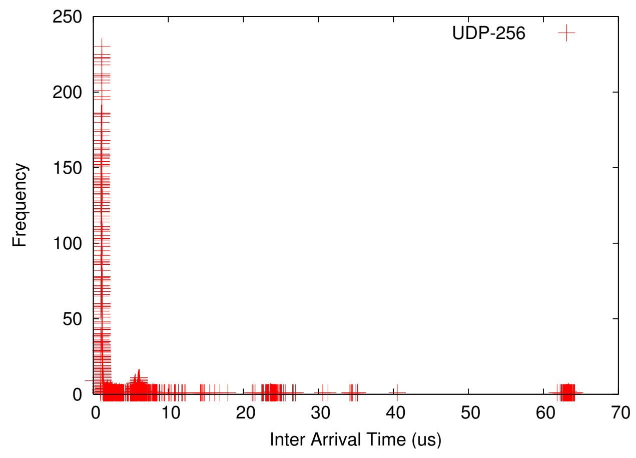
Figure 11: Communication Performance: with and without OPRA parsing: the overall performance is insensitive to the parsing delay



(a) MX



(b) DBL



(c) UDP

Figure 12: Arrival Rates for 99.9% Messages

5 Conclusion

In this paper we present the design, implementation and evaluation of a trading system that is built out commodity components, Intel x86 processors and 10 Gigabit Ethernet. In our prototype we have used the Myricom Myri-10GigE Network Interface Cards (NICs) [11] and DBL and MX, optimized libraries that allow OS-bypass data communication. We integrate our hand optimized OPRA decoder [2] based on our home-brewed DotStar pattern processing engine [14], to process the OPRA feeds at the destination node. The seamless integration of all these components delivers a prototype trading system that can streams OPRA messages with an end-to-end latency of less than 4 microseconds. The system is able to deliver a high bandwidth, reaching 3 million messages per second per core. We also present a preliminary analysis of the Myrinet communication protocols and highlight interesting trade-offs between latency and bandwidth of the Myrinet network interface card.

We observe that the network and the network stack play an important role, and are increasingly becoming the bottleneck in systems that can parse the OPRA protocol in parsing systems that operate at the sub-microsecond level. Overall, the paper demonstrates that by using best-in-class commodity hardware, algorithmic innovations and careful algorithmic design it is possible to obtain the performance of custom-designed hardware solutions.

Acknowledgements

We thank Patrick Geoffray and Christian Bell of Myricom for the insightful technical discussions on low-latency communication and on-line trading.

References

[1] ACTIV Financial Systems. At http://www.activfinancial.com/.

[2] V. Agarwal, D.A. Bader, L. Dan, L.-K. Liu, D. Pasetto, M. Perrone, and F. Petrini. Faster fast: multicore acceleration of streaming financial data. *Computer Science - R&D*, 23(3-4):249–257, 2009.

[3] Virat Agarwal, Lurng-Kuo Liu, and David Bader. Financial Modeling on the Cell Broadband Engine. *IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, pages 1–12, April 2008.

[4] Exegy. At http://www.exegy.com/.

[5] Fulcrum Microsystems. At http://www.fulcrummicro.com/.

[6] IBM. Develop high-volume, low-latency finance solutions with ibm websphere mq low latency messaging. *White Paper*, October 2009.

[7] Impulse Accelerated Technologies. At http://www.impulseaccelerated.com/.

[8] Mike Kistler, Michael Perrone, and Fabrizio Petrini. Cell Processor Interconnection Network: Built for Speed. *IEEE Micro*, 25(3), May/June 2006.

[9] Dinesh Manocha. General-Purpose Computations Using Graphics Procecssors. *IEEE Computer*, 38(8):85–88, Aug. 2005.

[10] G.W. Morris, D.B. Thomas, and W. Luk. Fpga accelerated low-latency market data feed processing. In *HOTI '09: Proceedings of the 2009 17th IEEE Symposium on High Performance Interconnects*, pages 83–89, Washington, DC, USA, 2009. IEEE Computer Society.

[11] Myri. At http://www.myri.com/.

[12] Myri. Myri-10G NICs and Software, 2009.

[13] OPRA. Option Price Reporting Authority. Available at `http://www.opradata.com/`.

[14] Davide Pasetto and Fabrizio Petrini. DotStar: Breaking the Scalability and Performance Barriers in Regular Expression Set Matching. Technical report, IBM T.J. Watson, 2009.

[15] Redline Trading Solutions. InRush Ticker Plant, Low Latency High Throughput Deterministic Solutions. *White Paper*.

[16] SIAC. FAST for OPRA - v1, 2007.

[17] SIAC. Data Recipient Specification, 2008.

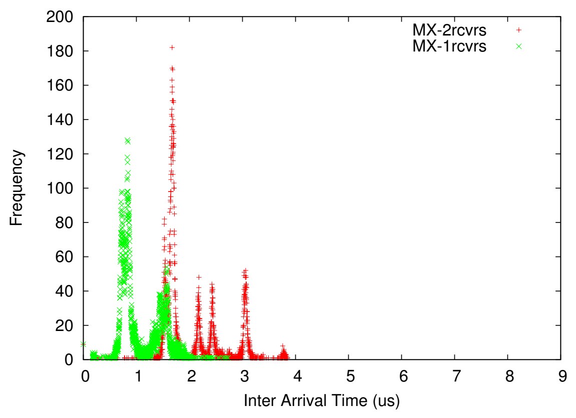
[18] SIAC. FAST for OPRA - v2, 2008.

[19] SIAC. National Market Systems - Common IP Multicast Distribution Network - Recipient Interface Specification, 2008.

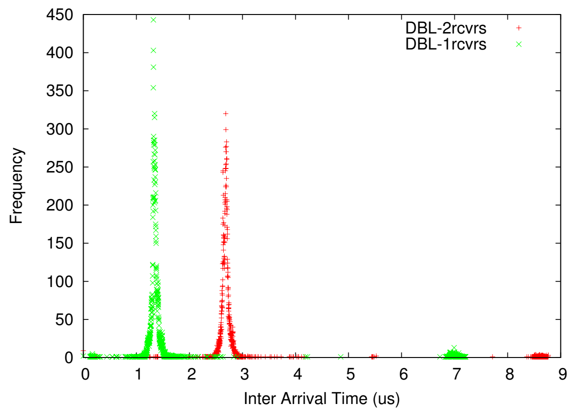
[20] Solace Systems. At http://www.solacesystems.com/.

[21] TABB Group. Trading At Light Speed: Analyzing Low Latency Data Market Data Infrastructure. Available at `http://www.tabbgroup.com/`, 2007.

[22] Tervela. At http://www.tervela.com/.



(a) MX



(b) DBL

Figure 13: Comparison of Arrival Rates with One and Two Receivers for MX and DBL