Scaling Advanced Message Queuing Protocol (AMQP) Architecture with Broker Federation and InfiniBand [*]

Gregory Marsh, Ajay P. Sampat, Sreeram Potluri, and Dhabaleswar K. Panda
Department of Computer Science and Engineering, The Ohio State University
{marshgr, sampat, potluri, panda}@cse.ohio-state.edu

Abstract

Message Oriented Middleware (MOM) is a key technology in financial market data delivery. In this context we study the Advanced Message Queuing Protocol (AMQP), an emerging open standard for MOM communication. Our previous research has found that the centralized architecture of AMQP presents a considerable bottleneck as far as scalability is concerned. Federating AMQP message Brokers decentralizes this architecture and can potentially help to scale performance across large clusters. In this paper, we examine this potential by analyzing basic federation topologies. We describe a systematic method for configuring Broker federation supported by experiments on a multi-node cluster with an InfiniBand network. To the best of our knowledge, this is the first in-depth study for designing scalable AMQP architectures with Broker federation and InfiniBand.

Keywords: *AMQP, Message Oriented Middleware, Qpid, Publish/Subscribe, InfiniBand*

1 Introduction

Message Oriented Middleware (MOM) plays a key role in financial data generation and delivery. The strength of MOM is that it allows for efficient communication between applications situated on heterogeneous operating systems and networks. MOM allows developers to by-pass the costly process of enabling explicit connections to these varied systems and networks. Instead applications need only communicate with the MOM. Typical MOM implementations feature asynchronous message delivery between unconnected applications via a message queue framework. However, there are prominent MOM implementations that operate without a queue framework [19].

Advanced Message Queue Protocol (AMQP) originated in the financial services industry in 2006 [1] [8] [20]. AMQP is an open standard for MOM communication. AMQP grew out of the need for interaction between MOM systems both within, and between, corporate enterprises. Due to the proliferation of proprietary, closed-standard, messaging systems such integration is considered challenging. As such, the primary goal of AMQP is to enable better interoperability between MOM implementations. Since AMQP's inception, several, open-source, messaging software distributions have emerged.

The three primary components in the AMQP architecture are Publishers (senders), Consumers (receivers), and Brokers (message routing engines). Our previous research has found that the centralized Broker architecture of AMQP presents a considerable bottleneck as far as scalability is concerned [17]. The Apache Qpid AMQP distribution [2] includes a Broker federation option that can decentralize the architecture and share the workload among a group of Brokers linked with each other. However there is very little information available as to how best to set up Broker Federation. For example, what topological layout of federated Brokers yields the best performance? Furthermore, how does expanding or changing a federation topology alter overall system performance?

In this paper we examine the ability of federation to alleviate the centralized broker bottleneck and deliver scalable messaging performance across large clusters. We devise a rigorous, systematic method for configuring Qpid's broker federation to deliver scalability for a large number of message consumers. Our federation methodology focuses on various bi-nomial and k-nomial tree structures. Using experiments on our cluster, we devise federation topologies for varying message sizes and measure performance by total delivered bandwidth, and message rate. Each of our federation schemes varies a tree topology parameter. The parameters that may be varied are: the levels of federated brokers between a Publisher and Consumer, the number of child Consumers serviced by a leaf Broker, and the number of child Brokers federated to a tree root Broker. Optimal values of these parameters may then be used to determine a scalable Broker federation topology for a large number of Consumers on the cluster.

This paper's main contributions are:

- An examination of the variables for Broker federation topology and how they impact messaging performance.
- Recommendations as to how best to setup AMQP Broker federation based on this examination.

[*]This research is supported in part by DOE grants #DE-FC02-06ER25749 and #DE-FC02-06ER25755; NSF grants #CNS-0403342, #CCF-0702675, and #CCF-0833169; and Wright Center for Innovation grant #WCI04-010-OSU-0.

Our performance evaluation shows that Broker federation indeed helps to alleviate the bottlenecks and provide a more scalable architecture. For example, a three level Broker federation supporting 16 consumers delivers 64 byte messages at a rate of 14,342 msgs/sec as compared to 2,570 msgs/sec by a single Broker supporting 8 consumers. We also observe that different topology parameters are favorable for different message sizes. For small messages, trees with Broker fanout limited to 2 and Consumer fanout limited to 2 or 4 give scalable performance while trees with higher fanouts show higher delivered bandwidth numbers for large messages.

The remainder of this paper is organized as follows: Section 2 gives a brief overview of AMQP, Qpid Federation, and InfiniBand technologies. In Section 3, we overview the major variables inherent in constructing a federation topology. Section 4 presents the impact of experiments with these toplogy variables. Using the knowledge gained from the experiments, Section 5 describes design suggestions for scaling Broker federation to a large number of consumers. Section 6 overviews related work. Finally we summarize our conclusions and possible future work in Section 7.

2 Background

In this section we provide a brief overview of Advanced Message Queuing Protocol, Broker federation in Qpid, and InfiniBand.

2.1 Advanced Message Queuing Protocol

Figure 1 shows the general architecture of an AMQP compliant messaging system. An AMQP messaging system consists of three main components: Publisher(s), Consumer(s) and Broker/Server(s). Each component can be multiple in number and be situated on independent hosts. Publishers and Consumers communicate with each other through message queues bound to exchanges within the Brokers. AMQP provides reliable, guaranteed, in-order message delivery. We briefly explain the functionality of each component below.

Broker/Server: A server or daemon program that contains one or more Exchanges and Message Queues.

Consumer: An application that declares one or more Message Queues on the Broker.

Message Queue: A data structure that stores messages in memory or on disk, and delivers these in sequence to the Consumer that declared the queue. Each Message Queue is entirely independent.

Publisher/Producer: An application that constructs messages and sends the messages to an Exchange on a Broker.

Exchange: A matching and routing engine which accepts messages from Publishers and copies the messages into zero or more Message Queues.

2.2 Qpid Broker Federation

Qpid's Broker federation [12] is a mechanism to incorporate more than one broker into a messaging system. The

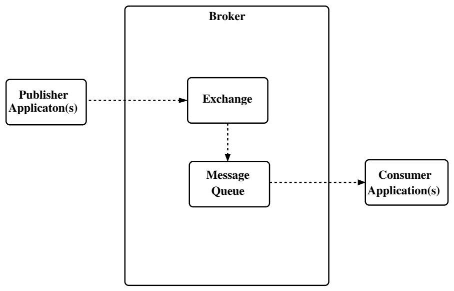


Figure 1. The AMQP Architecture

system administrator uses the "qpid-route" utility to establish links between Brokers. The linked Brokers are then federated. Each Broker may be on an independent computing node and service its own set of connected Publishers and Consumers. The destination Broker in the link acts as a special Consumer of messages from the source Broker. However instead of consuming messages, the destination Broker forwards the messages to its attached Consumers and/or any destination Broker federated to it. From the perspective of the connected Publishers and Consumers, a group of federated Brokers act as one logical Broker, even though each Broker may be on an independent node. Any message published to a federated Broker will be forwarded to all other Brokers in the federation.

2.3 InfiniBand Architecture

InfiniBand Architecture (IB) [4] is an industry standard for low latency, high bandwidth, System Area Networks (SAN). An increasing number of InfiniBand network clusters are being deployed in high performance computing (HPC) systems as well as in E-Commerce-oriented data centers. IB supports two types of communication models: Channel Semantics and Memory Semantics. Channel Semantics involve discrete send and receive commands. Memory Semantics involve Remote Direct Memory Access (RDMA) [14] operations. RDMA allows processes to read or write the memory of processes on a remote computer without interrupting that computer's CPU. Within these two communication semantics, various transport services are available that combine reliable/unreliable, connected/unconnected, and/or datagram mechanisms.

The popular TCP/IP network protocol stack can be adapted for use with InfiniBand by the Internet Protocol over Infiniband (IPoIB) driver [6]. IPoIB is a Linux kernel module that enables InfiniBand hardware devices to encapsulate IP packets into IB datagram or connected transport services. When IPoIB is applied, an InfiniBand device is assigned an IP address and accessed just like any regular TCP/IP hardware device.

3 Major Variables in Federation Topology

In this section we give a high level overview of the concepts involved in building a scalable Broker topology using a k-nomial tree scheme. Such trees may vary in their depth (number of levels - 1 factor) as well as their fanout breadth (k factor). Below we describe depth and breadth individually as applied to AMQP Broker federation. Furthermore we give our general expections as to how each of these variables will impact the interaction of the messaging system with both the network and host node resources.

3.1 Multi-Level Broker to Consumer

When an AMQP system places Brokers on nodes independent of the Producers and Consumers , a message must always traverse the Broker host in route to the destination Consumer. This incorporates at least two network links into the travel path of any message: one from the Producer (P) to the Broker (B) and another from the Broker to the Consumer (C). However the use of Broker federation might add multiple Brokers in the message path. As shown in Figure 2, multi-level Broker federation has the potential to add many network hops into the travel path of any message in the system.

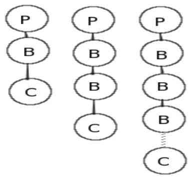


Figure 2. Multi-Level Broker to Consumer

3.2 Single Broker with Consumer Fanout

When a Broker is running on its own host node, a Consumer process must establish a network connection in order to receive messages from the Broker. This connection results in a message queue structure being allocated on the Broker host for each Consumer. Therefore as more Consumers connect to a Broker (Figure 3), there is potentially greater contention for the Broker node's resources. The Broker process must allocate memory for each Consumer's message queue, handle network traffic from multiple connections, and use CPU cycles and bus bandwidth to copy incoming messages to the Consumers' queues.

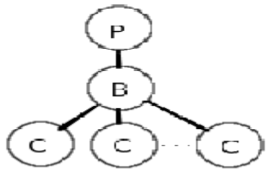


Figure 3. Single Broker with Consumer Fanout

3.3 Federated Brokers Fanout to Consumers

The Broker to Broker links established in federation are similar in operation to that of Consumers connecting to a Broker. The destination Broker in the link is allocated a message queue on the source Broker. In a tree federation with a single root broker (Figure 4), as the number of connecting Brokers increases we expect contention for the source Broker's computing resources. However is this contention of the same nature as that of Consumers connecting? Will the same resources (memory, CPU, bus traffic) be stressed in the same amounts?

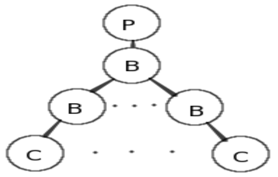


Figure 4. Federated Brokers Fanout to Consumers

3.4 Designing Scalable Federation Topology

In selecting a scalable federation topology the concepts of AMQP Broker federation described above must be combined as shown in Figure 5. A favorable depth of the tree must work in unison with a favorable k1 factor of Brokers fanning out from the root Broker as well as the k2 factor of Consumers fanning out from the leaf Brokers. In the next section we describe an experiment-driven methodology where we test each tree federation variable and select a combined toplogy based on the test results.

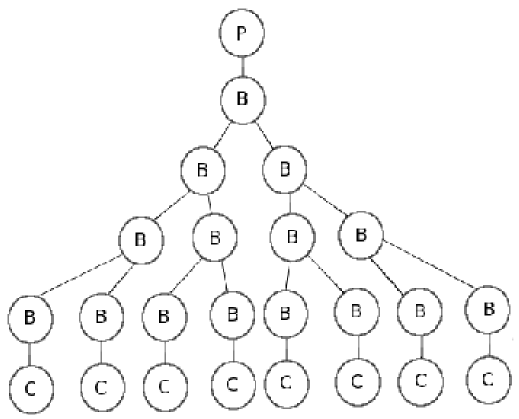


Figure 5. Designing Scalable Federation Topology

4 Understanding the Impact of Major Variables

In this section, we present results of our experiments with the variables in Broker federation. Our goal is to understand how altering federation topology will affect messaging system performance. Since message size impacts performance, we consider the interaction of small (64 bytes to 4 Kilobytes), and large (64KB to 256 Kilobytes) sized messages. Mirroring the topology concepts described in Section 3 we examine how tree dimensions may vary and how this impacts the performance metrics for a Broker federation. By observing the performance differences obtained with controlled, experimental variations, we aim to derive general trends for delivered bandwidth, and message rate performance.

4.1 Setup

Our evaluation uses a 64 node cluster consisting of Intel Xeon Quad dual-core processor hosts for a cluster wide total of 512 cores. Each host node has 6GB RAM and is equipped with a Mellanox Technologies ”MT25208 InfiniHost III Ex” DDR Host Channel Adapter (HCA). This HCA uses Open Fabrics Enterprise Distribution (OFED) version 1.3.1 [9] drivers with IPoIB. The operating system for each node is Red Hat Enterprise Linux Server 5. We conducted all Qpid operations with the M4 release. Our primary evaluation tool was the Qpid benchmarks previously developed and described in [17]. The SYSSTAT sar utility [18] was used in parallel with the benchmarks in order to collect system resource usage.

4.2 Multi-Level Broker to Consumer

As shown in Figure 2 this set of experiments varies the number of Brokers in between one Publisher to one consumer. The goal of these experiments is to determine the performance impact of increasing the Broker federation chain. Adding more Broker levels maintains approximately 25,000 to 30,000 constant message rate for small messages (Figure 16, Table 1). Additional levels beyond 1 Broker actually enhance bandwidth performance for large messages, with 2 Broker levels being the best (Figure 7). However 3, 4, and 5 Broker levels also deliver more bandwidth than 1 Broker (Table 2). Examination of sar data (Figure 8) shows a higher HCA transmission rate for each multi-level Broker as compared to single Broker case. This increased transmission rate is perhaps due to a lower page fault rate on the intermediate Brokers which have no Producers or Consumers attached.

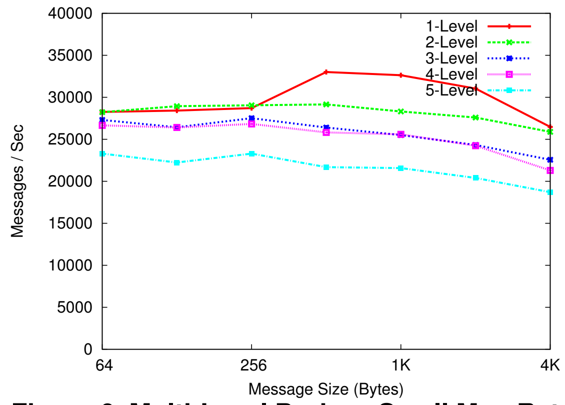


Figure 6. Multi-Level Broker: Small Msg Rate

4.3 Single Broker with Consumer Fanout

As shown in Figure 3 this set of experiments varies the number of Consumers attaching to single Broker and studies the impact. The goal of these experiments is to determine the performance impact of increasing the number of Consumers attached to a leaf Broker. Fanouts of 1, 2, and 4 Consumers maintain an approximately constant message rate of about 20,000 to 30,000 for small messages (Figure 9, Table 3). Furthermore large message delivered bandwidth increases with an increasing number of consumers (Figure 10, Table 4). For small messages, sar shows a progessively increasing transmission rate on the Broker HCA up to 4 Consumers (Figure 11). Also the page fault rate decreases on the Broker with each successive Consumer up to 4. After 4 Consumers transmission is only maintained for large messages.

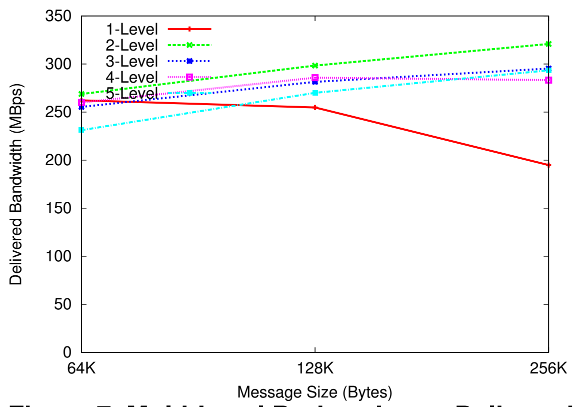


Figure 7. Multi-Level Broker: Large Delivered Bandwith

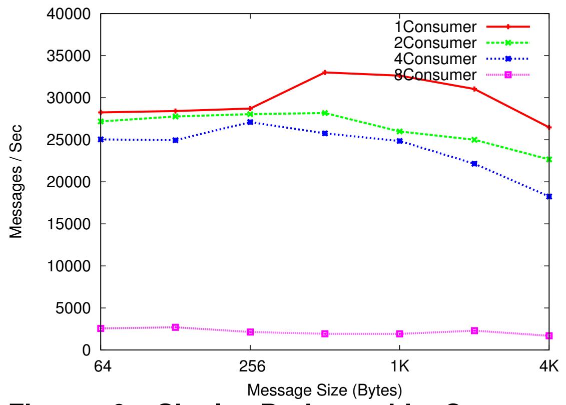
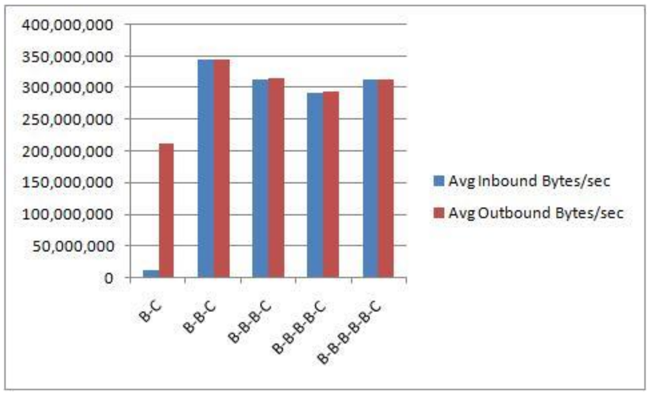


Figure 9. Single Broker with Consumer Fanout: Small Msg Rate

4.4 Federated Brokers Fanout to Consumers

As shown in Figure 4 this set of experiments varies the number of source Brokers federating with a single tree root Broker and studies the performance impact. Each lower level leaf broker has a single Consumer attached to it. A fanout of 2 federated Brokers is favorable for small messages (Figure 12, Table 5). At fanouts of 3 and 4 Brokers sar shows a dramatic increase in context switches and small message performance decreases. However a fanout of 3 and 4 Brokers maintains delivered bandwith for large messages (Figure 13, Table 6) and Figure 14 shows a proportional transmission rate on the Brokers' HCA for such message sizes.



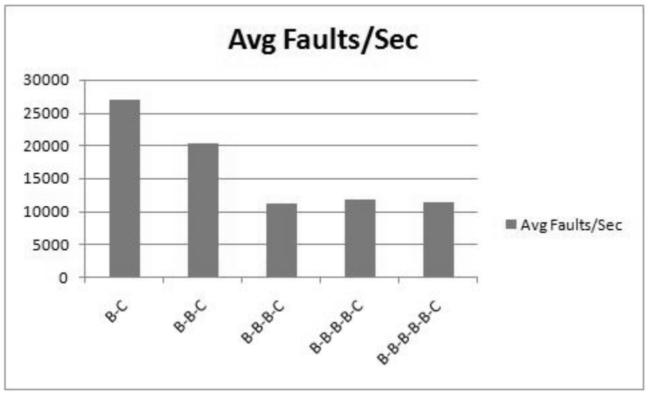


Figure 8. (a) Multi-Level Broker:Inbound/Outbound Bytes/Sec-256KB, (b) Multi-Level Broker:Average Page Faults/Sec -256KB

Table 1. Multi-Level Broker to Consumer: Small Msg Rate (Msgs/Sec)

Model/Msg Size	64B	256B	1KB	4KB
B-C	28,258	28,716	32,628	26,476
B-B-C	28,233	29,046	28,314	25,888
B-B-B-C	27,309	27,510	25,501	22,557
B-B-B-B-C	26,656	26,828	25,589	21,298
B-B-B-B-B-C	23,293	23,296	21,573	18,705

Table 2. Multi-Level Broker to Consumer: Lrg Msg Delivered Bandwidth (MB/Sec)

Model/Msg Size	64KB	128KB	256KB
B-C	262	254	194
B-B-C	268	298	320
B-B-B-C	255	281	295
B-B-B-B-C	259	285	283
B-B-B-B-B-C	231	269	293

Table 3. Single Broker with Consumer Fanout: Small Message Rate

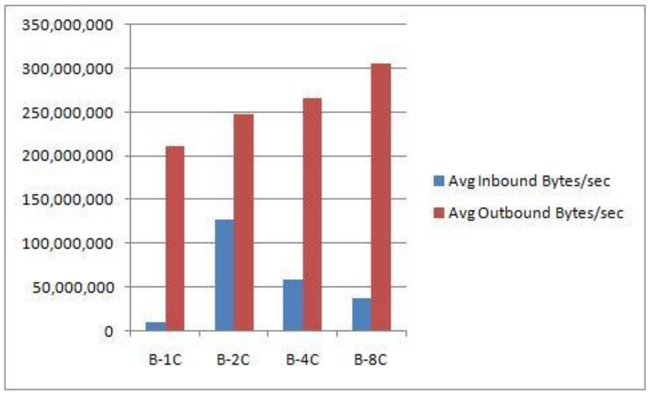
Model/Msg Size	64B	256B	1KB	4KB
B-1C	28,258	28,716	32,628	26,476
B-2C	27,183	28.052	25,991	22,665
B-4C	25,048	27,107	24,854	18,260
B-8C	2,570	2,143	1,925	1,692

Table 4. Single Broker with Consumer Fanout: Lrg Msg Delivered Bandwidth (MB/Sec)

Model/Msg Size	64KB	128KB	256KB
B-1C	262	254	194
B-2C	410	344	264
B-4C	412	300	244
B-8C	532	504	320

Table 5. Federated Brokers Fanout to Consumers: Small Message Rate

Model/Msg Size	64B	256B	1KB	4KB
B-B-C	30,170	28,637	27,812	26,247
B-2B-2C	23,918	25,209	22,595	15,920
B-3B-3C	4,951	4,819	5,321	4,680
B-4B-4C	3,774	3,648	4,058	3,455



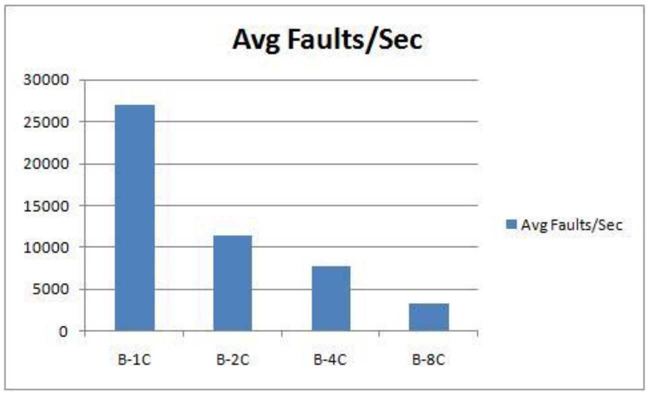
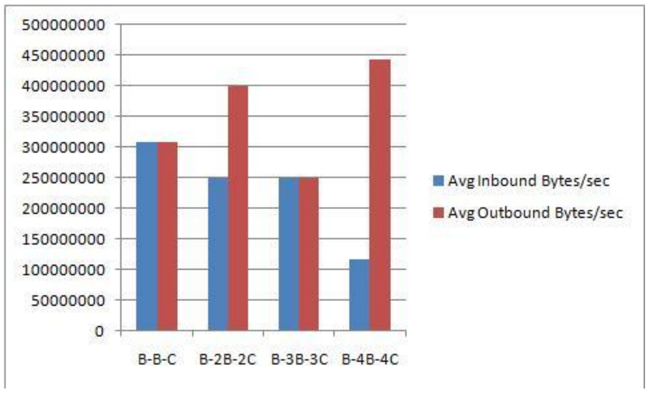


Figure 11. (a) Single Broker Fanout: Inbound/Outbound Bytes/Sec-256KB, (b) Single Broker Fanout: Average Page Faults/Sec -256KB

Table 6. Federated Brokers Fanout to Consumers: Lrg Msg Delivered Bandwidth (MB/Sec)

Model/Msg Size	64KB	128KB	256KB
B-B-C	267	290	292
B-2B-2C	292	282	232
B-3B-3C	474	492	531
B-4B-4C	480	600	464



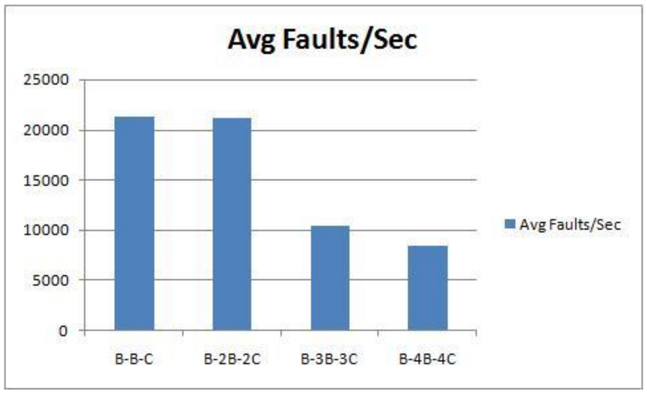


Figure 14. (a) Federated Brokers Fanout to Consumers: Inbound/Outbound Bytes/Sec-256KB, (b)Federated Brokers Fanout to Consumers: Average Page Faults/Sec -256KB

Table 7. Federation to Support 16 Consumers: Lrg Msg Delivered Bandwidth (MB/Sec)

Model/Msg Size	64KB	128KB	256KB
B-2B-16C	704	592	560
B-4B-16C	1296	1264	864
B-2B-4B-16C	1200	1120	1056
B-4B-8B-16C	1744	2192	2544
B-2B-4B-8B-16C	1728	1808	1760

Table 8. Federation to Support 16 Consumers: Small Message Rate

Model/Msg Size	64B	256B	1KB	4KB
B-2B-16C	1,152	1,168	1,140	1,094
B-4B-16C	7,251	4,752	4,049	3,265
B-2B-4B-16C	14,342	11,868	10,786	6,591
B-4B-8B-16C	3,642	3,498	3,373	3,254
B-2B-4B-8B-16C	8,291	8,211	7,618	5,769

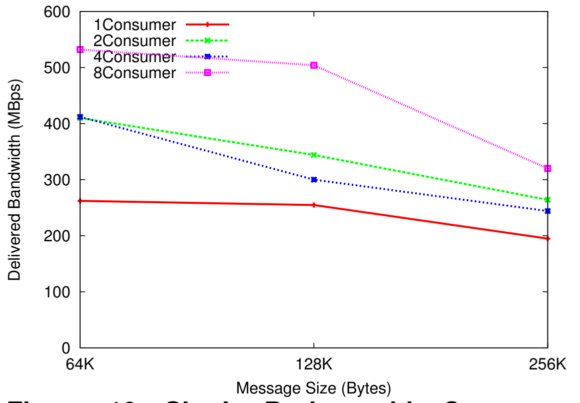


Figure 10. Single Broker with Consumer Fanout: Large Delivered Bandwith

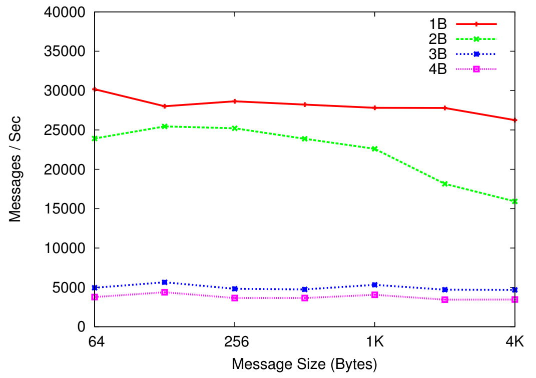


Figure 12. Federated Brokers Fanout to Consumers: Small Msg Rate

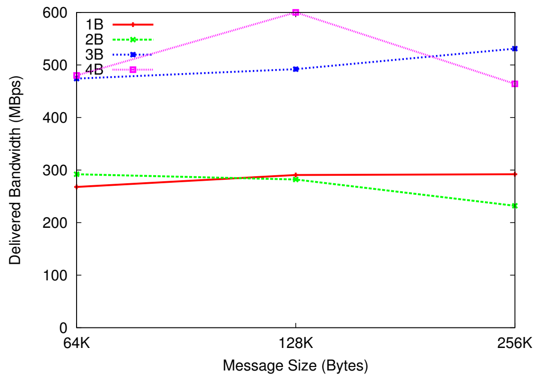


Figure 13. Federated Brokers Fanout to Consumers: Large Delivered Bandwith

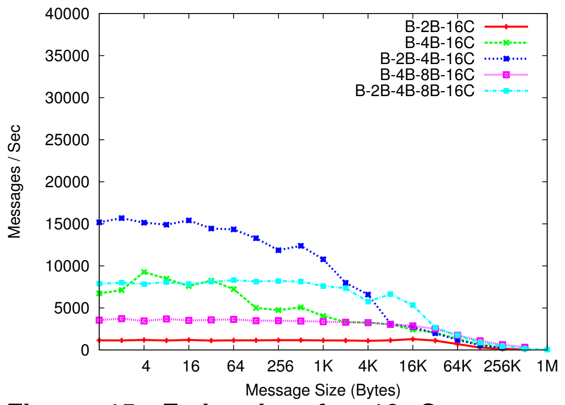


Figure 15. Federation for 16 Consumers: Small Message Rate

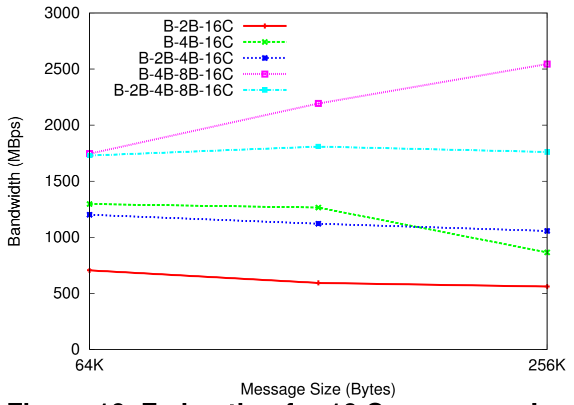


Figure 16. Federation for 16 Consumers: Lrg Msg Delivered Bandwidth

5 Designing Scalable Federation Topology

Using the insights gained from the above experiments, we implemented alternate federation topologies in an attempt to scale performance to 16 Consumers. Table 8 shows the small message results for each attempted alternative. The ”B-2B-4B-16C” toplogy yielded the best performance, with ”B-2B-4B-8B-16C” yielding second best. The parameters of the ”B-2B-4B-16C” topology (3 level, binomial tree with 4 Consumers attached to the leaf brokers) are consisent with the best performing configurations in the experiments of Section 4. As shown in Table 1, fewer levels of Brokers generally produce a higher message rate. Furthermore Table 3 shows small message rate was favorable at a fanout of 4 Consumers to a leaf Broker. Finally a federation fanout of 2 Brokers from a root Broker yields high performance as shown in Table 5.

Table 7 shows large message delivered bandwidth for each 16 Consumer alternative. The ”B-4B-8B-16C” toplogy yielded the best performance, with ”B-2B-4B-8B-16C” yielding second best. Again the tree parameters the ”B-4B-8B-16C” topology (3 level, 4-nomial tree with 2 Consumers at leafs) are consistent with the better performing configurations in our experiments. In Table 2, 3 levels of Brokers maintain better large message performance than 1 or 5 levels. Also a fanout of 2 Consumers to a leaf Broker has good performance as shown in Table 4. Plus a federation fanout of 4 brokers yield high large message performance as shown in Table 6.

6 Related Work

A number of vendors offer proprietary products to meet the demand for MOM applications. Examples include IBM’s Websphere MQ [3], Microsoft Message Queuing Server [7], TIBCO’s Rendezvous and Enterprise Messaging Servers [19], and Progress Software Corporation’s Sonic MQ [11]. Furthermore Sun Microsystems has incorporated the Java Message Service (JMS) API into the Java Platform Enterprise Edition [5]. The Standard Performance Evaluation Corporation (SPEC) has also built a standard MOM benchmark oriented to JMS called SPECjms2007

[16]. Within the AMQP standard, other open source implementations have emerged in addition to Apache Qpid. These implementations include OpenAMQ [10] and RabbitMQ [13]. Furthermore, RedHat Enterprise MRG uses Qpid as it's underlying base [15].

7 Conclusions and Future Work

In this paper we studied the ability of AMQP Broker federation to scale and improve messaging performance across multi-node computing clusters. We devised a methodolgy to determine favorable Broker federation with Apache Qpid. We designed our methodology around the k-nomial tree structure used in network broadcast operations. Our experimental results show that for small message sizes Consumer fanouts of 2 or 4 deliver the best message rate. Similarly for large message sizes, a three level federation with a 4 Broker fanout yields the best delivered bandwidth. As part of our future work we plan to deploy federation topologies supporting 32, 64, and 128 Consumers. Furthermore we would like to study the impact of Quad Data Rate (QDR) rate HCA's and the next generation processor platforms such as Intel Nehalem on AMQP performance.

8 Acknowledgments

We would like to acknowledge the assistance of Rahul Mukhedkar in revising our Qpid benchmark code from the M3 release API to the latest M4 API, as well as revising the scripts to report SYSSTAT/sar data. We would also like to extend thanks to Hari Subramoni for contributing technical support as well as review and comments on this paper's content.

References

[1] Advanced Message Queuing Protocol Website. http://www.amqp.org/.

[2] Apache Qpid. http://qpid.apache.org/.

[3] IBM Websphere MQ. http://www.ibm.com/software/integration/wmq/.

[4] Infiniband Trade Association. http://www.infinibandta.org.

[5] Java Message Service. http://java.sun.com/products/jms/.

[6] Mellanox OFED Stack for Linux Users Manual. http://www.mellanox.com/pdf/products/software/Mellanox_OFED_Linux_User_Manual_1_20.pdf.

[7] Microsoft Message Queuing Server. http://www.microsoft.com/windowsserver2003/technologies/msmq/.

[8] J. O'Hara. Toward a commodity enterprise middleware. *Queue*, 5(4):48–55, 2007.

[9] Open Fabrics Enterprise Distribution. http://www.openfabrics.org/.

[10] OpenAMQ. http://www.openamq.org/.

[11] Progress Software Corporation. http://www.sonicsoftware.com/.

[12] Qpid Broker Federation Website. http://qpid.apache.org/using-broker-federation.html.

[13] RabbitMQ. http://www.rabbitmq.com/.

[14] RDMA Consortium. http://www.rdmaconsortium.org/home/draft-recio-iwarp-rdmap-v1.0.pdf.

[15] RedHat Enterprise MRG. http://www.redhat.com/mrg/.

[16] SPECjms2007 Benchmark. http://www.spec.org/jms2007/.

[17] H. Subramoni, G. Marsh, S. Narravula, P. Lai, and D. Panda. Design and Evaluation of Benchmarks for Financial Applications using Advanced Message Queuing Protocol (AMQP) over InfiniBand. *Workshop on High Performance Computational Finance, 2008. WHPCF 2008.*, pages 1–8, Nov. 2008.

[18] SYSSTAT Utilities Home Page. http://pagesperso-orange.fr/sebastien.godard/.

[19] TIBCO. http://www.tibco.com/.

[20] S. Vinoski. Advanced message queuing protocol. *Internet Computing, IEEE*, 10(6):87–89, Nov.-Dec. 2006.