KAAZING

"FASTEST MILLION" OVER THE WEB WITH KAAZING, DELL, AND TIBCO

DELIVER REAL-TIME DATA TO ONE MILLION CONCURRENT WEB USERS ON A SINGLE RACK

KAAZING

Powered by DELL

TIBCO The Power of Now™

High Performance

Scalability

Security

Table of Contents

"The Fastest Million" over the Web with Kaazing, Dell, and TIBCO

In the 21 years since Tim Berners Lee launched the World Wide Web (WWW) the world is an entirely different place. It is now inconceivable for most people to imagine a time when access to information was not ubiquitous. For the first time in over 20 years, a new innovation has been ratified that will move us from our current capabilities (Legacy Web) to a totally new age entirely, *The Living Web*.

The Living Web will be marked by a renaissance in application development, where a new breed of Web Applications is racing to the Web. Web applications where people or machines, whether stationary or mobile, will be able communicate over the Web in real time, producing, consuming, and processing information simultaneously and instantaneously. These Web applications promise real-time interactivity and collaboration to deliver the user experience expected by a new generation of young people who have grown up with expectations of millisecond response times and no wait. Not something usually associated with traditional Web applications and infrastructure.

To deliver on these new demands companies are faced with significant challenges due to the computing power and data required using today's legacy Web infrastructure solutions. The traditional approach using application servers has led to over-complicated architectures with low developer productivity, long time-to-market delays, large hardware and software requirements, and costly associated ongoing maintenance.

So, how do you share data that is locked up in the enterprise and back-end systems to users on the Web in a way that performs well and is secure?

Kaazing WebSocket Gateway (KWG) is a high-performance Web communication platform built with the HTML5 WebSocket standard as a foundation. It is a platform that enables browsers and mobile devices to participate in an asynchronous event-driven architecture over the Web without sacrificing performance or security.

Without disrupting your existing infrastructure, Kaazing's approach lets you overlay this modern architecture on top of what you already have, preserving your investment. Kaazing WebSocket Gateway extends your back-end systems to the Web, letting browsers and mobile devices participate as first-class citizens in your existing architecture.

Enterprise Communication Over the Web

For many of the World's top 1000 businesses and organizations the stability and reliability of their enterprise communication and data delivery is provided by a common backbone infrastructure, such as TIBCO Enterprise Message Service (TIBCO EMS). Messaging platforms have long been prevalent in the enterprise because it integrates disparate systems in a loosely coupled way, abstracting away their differences, and providing a means to scale these back-end silos independently. Messaging enables service-oriented architectures and serves as the foundation for business processes management thanks to its system integration.

However messaging has traditionally been kept locked in the enterprise, and it's only relatively recently that Web-based messaging has started to proliferate. Due to its infancy, the majority of messaging solutions for the Web are proprietary and don't have the feature richness, capability, performance, security, and robustness of existing standards-based enterprise messaging.

Kaazing WebSocket Gateway - JMS Edition is not a message broker, but extends best-of-breed standard JMS message services to the Web, allowing browsers and mobile devices to participate with high performance while remaining standards-based and preserving the rich feature set and robustness of an enterprise messaging solution.

Benchmark Architecture

This benchmark demonstrates that is feasible to create a Web-based architecture using **Kaazing WebSocket Gateway** to deliver real-time data to one million concurrent connections on a single **Dell PowerEdge R620** rack using standards-based messaging with **TIBCO Enterprise Message Service (EMS)** at 10 messages per second with an average latency of 3.5 milliseconds from message broker to client.

Hardware

Dell PowerEdge R620 server, with:

CPU	Dual 3.3GHz Quad-core Xeon CPUs (Two CPUs, each a Quad-core for a total of 8 cores) Details: Intel Xeon E5-2643 3.30GHz, 10M Cache, 8.0GT/s QPI, Turbo, 4C, 130W
Memory	32 GB RAM Details: 4GB RDIMM, 1600 MT/s, Standard Volt, Dual Rank, x8 Data Width Quantity: 8
Hard Disk	500GB in a RAID-1 mirror configuration Details: 500GB 7.2K RPM SATA 2.5-in HotPlug Hard Drive (Quantity: 2)
Network	Dual 10Gb SPF+ Ethernet cards (Intel and QLogic) In addition: 4 onboard 1Gb Ethernet network ports
Operating System	CentOS 6.1 64-bit (100% binary compatible with RedHat Enterprise Linux Server 6.1)

Software

- Kaazing WebSocket Gateway - JMS Edition
- TIBCO Enterprise Message Service
- KWG is Java-based and runs in a JVM. The version of Java used in this benchmark was Java HotSpot 64 Bit Server 1.6.0_33.

Architecture Diagram

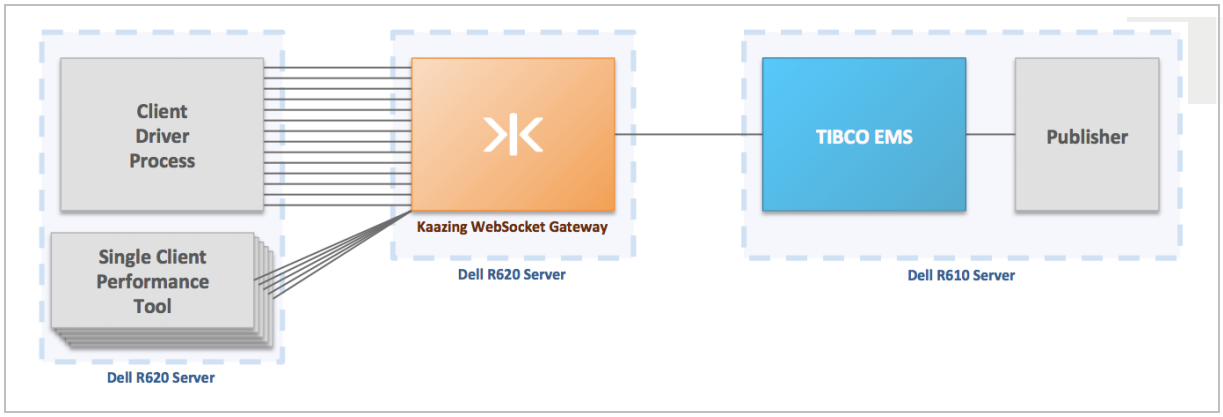

Figure 1: Architecture diagram

Three servers were utilized during the testing.

- As this benchmark was designed to test the performance of Kaazing WebSocket Gateway, it ran isolated on a Dell R620 1U rack server.
- TIBCO EMS and the publisher ran on a separate machine, a Dell R610 1U rack server.
- An additional Dell R620 1U rack server was used to generate the client load.

Isolating the different components in the architecture allowed KWG to be benchmarked without interference from the client driver or TIBCO EMS.

The client driver process generated the client connections, but did not record detailed performance data. The single client performance tool collected the detailed performance data. To ensure a good sample size for data collection, multiple instances of the single client performance tool were run to collect a variety of samples.

Methodology

Scenario

The scenario used mimics a FTSE 100 Index stock streaming scenario.

- The publisher constructs and sends 100 different stock prices on 100 different topics, generating one stock symbol per topic.
- Each stock topic is updated 10 times a second.
- Every client subscribes to topic, and therefore receives 10 messages per second.
- The payload size for each message is quite large at 512 bytes per message.

Latency

Latency in this benchmark is defined from the time the message broker first receives a message until it is finally received on the client, as shown in Figure 2. Thus, the latency reported in this benchmark the summation of all of the following:

- Processing time of the message through TIBCO EMS.
- Two physical network hops meaning that the message was—twice—truly marshalled and unmarshalled across the network as would occur in a real-world scenario.
- Processing time of the message through KWG.
- Time on the client to receive and process the inbound message.

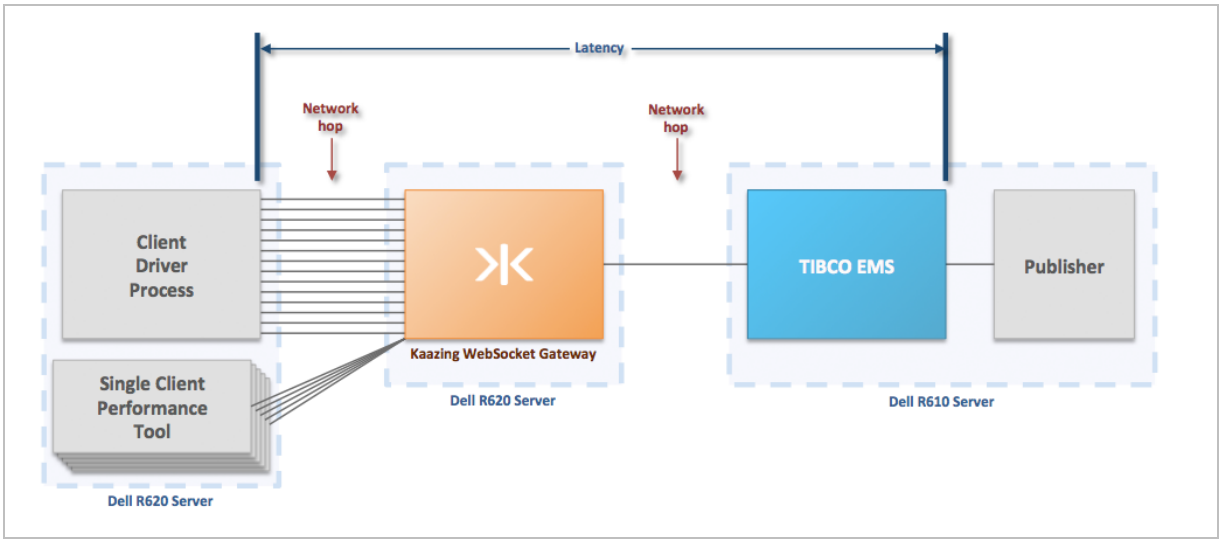

Figure 2: Latency

Fanout

Kaazing WebSocket Gateway - JMS Edition has connection and subscription offloading capability for increased scalability.

Connection Offloading

Figure 2 depicts that there are many client connections to KWG, but only one connection between KWG and TIBCO EMS. The Gateway offloads the cost of terminating many persistent connections from the message broker.

Subscription Offloading

Kaazing WebSocket Gateway detects shared topic subscriptions so that only unique subscriptions are made to the back-end message server. For example, if multiple clients subscribe to Topic A, KWG makes only one subscription to the message system (TIBCO EMS).

This combination of connection offloading and subscription offloading is often referred to as "fanout".

Fanout means that Kaazing WebSocket Gateway can scale up to large numbers of concurrent clients without putting additional pressure on your message service; connections and subscriptions are kept to a minimum. The number of connections between the Gateway and the message service can be as low as one, but depending on the use case may go up to one connection per CPU core for increased parallelism.

The number of connections on the back-end is extremely low and fixed, even as the number of clients increases to Web scale.

Subscriptions to shared topics benefit greatly from fanout. For example, if many clients are subscribed to a topic for a specific stock symbol, say, then the stock price only needs to be sent once from the message service to KWG, where it will then be fanned out to all subscribers for that symbol.

Linear Horizontal Scalability

Kaazing WebSocket Gateway can be clustered for horizontal scalability (and failover). That means you can add more nodes—i.e., servers running KWG—to a cluster in order to scale up. Furthermore, the design of KWG means that you get linear scalability as you add nodes to a cluster. How is this achieved?

Kaazing WebSocket Gateway is neither an application server nor a message broker, and is “stateless” in regard to client application state. Messages sent by the broker flow through the cluster, never across nodes in the cluster.

Cluster members do not share any state and only communicate when it's discovered a cluster member has left the cluster or a new one has been added. This means that once you have determined capacity for a particular scenario and hardware, you can scale horizontally in linear fashion by simply adding more Gateways.

Figure 3 shows an example of a cluster with two nodes, or cluster members. Each node is a Dell R620 running Kaazing WebSocket Gateway. As clients connect they will be balanced to either node-1 or node-2.

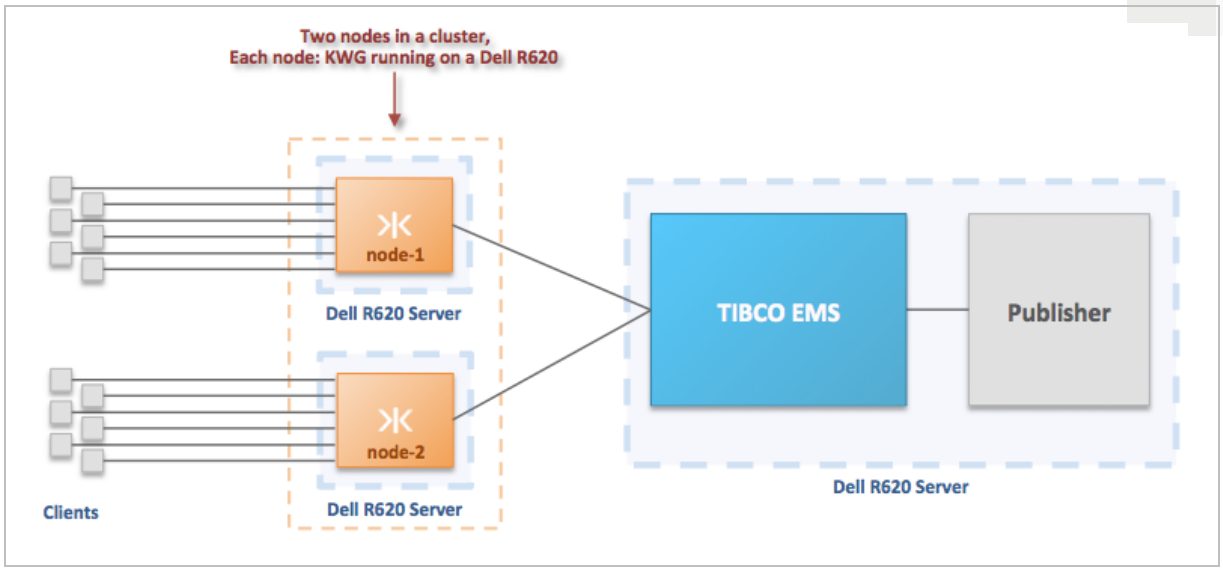

Figure 3: Two-node cluster

Each node is aware of nodes leaving the cluster or nodes joining the cluster for the purpose of load balancing and failover. However those are rare events.

Apart from the cluster being updated when a node leaves or joins the cluster, there is no state shared across the cluster. During a regular steady state, for example, node-1 is not affected by node-2 and vice-versa.

If a third node were added to the cluster, the other two nodes receive an update that a new node has joined. Thereafter the new node has no effect on either node-1 or node-2.

For example, if node-1 supports 24,000 concurrent clients, then you can add node-2 to the cluster and support 48,000 concurrent clients.

Since the nodes do not communicate in a steady state, you can add new nodes without any impact on existing nodes. Each node will have the same capacity and full capacity, resulting in linear scalability.

Benchmark Results

This chapter summarizes the results from the benchmark. See the Analysis of Results chapter on page 15 for a discussion of the results.

Connections

As described in the Benchmark Architecture chapter, three hardware servers were utilized: one to generate client load, one for Kaazing WebSocket Gateway, and one for TIBCO EMS and the publisher. See Figure 4.

On the client driver server, the main client driver process generated a load of 23,970 connections. In addition, 30 instances of the single client performance tool were run to capture detailed performance data. Thus there were a total of 24,000 client connections to Kaazing WebSocket Gateway.

All 24,000 connections went across a physical network and through a network switch, and therefore included the full TCP network stack on both ends.

There was a single connection, also over a physical network, between the Gateway and TIBCO EMS.

The publisher published data into TIBCO EMS over a TCP connection on the same host.

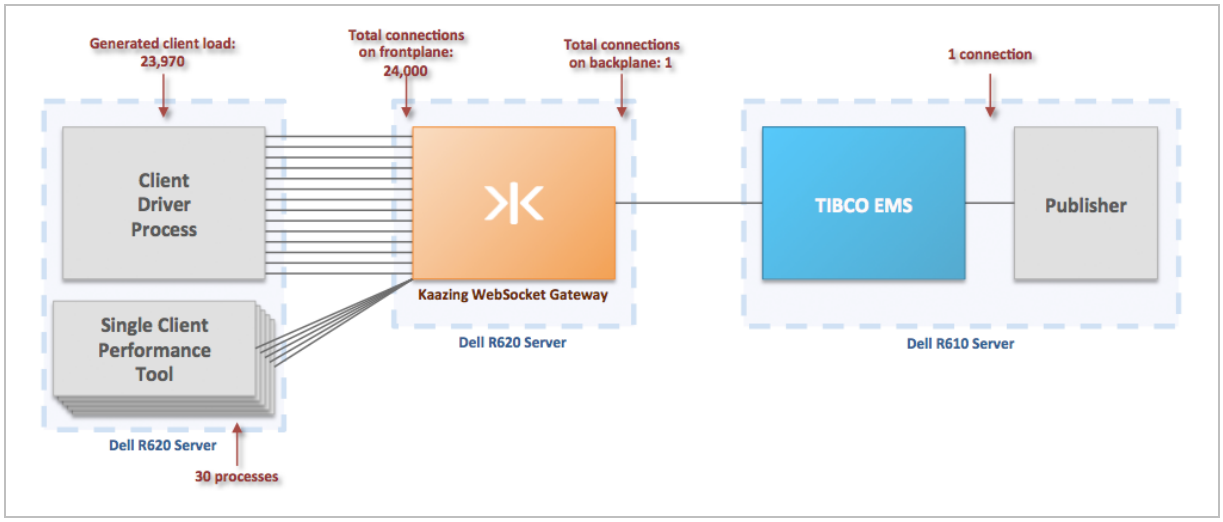

Figure 4: Connections

Messages

24,000 concurrent connections each receiving 10 messages a second results in 240,000 messages per second egress from the Gateway, requiring 1,000 messages per second from TIBCO EMS.

Latency

As described in the **Connections** section, above, there were 30 instances of the single client performance tool used to record performance data. (The Analysis of Results chapter on page 15 will discuss why there were 30 instances.)

Table 1, on page 12, contains the summary information for all 30 instances.

Each row in Table 1 represents the performance data collected for a particular client. For example, client #1 experienced an average latency of 2.776 milliseconds. Client 5 had an average latency of 5.619 milliseconds.

The duration of the test, once steady state was achieved, was 480 seconds.

Latency Summary

From Table 1, the following was calculated:

- The population mean latency is 3.5 +/- 0.35 milliseconds with a confidence interval of 95%
- The population mean 95th percentile is 6.0 +/- 0.61 milliseconds with a confidence interval of 95%
- The population mean 99th percentile is 16.9 +/- 1.55 milliseconds with a confidence interval of 95%

See the Analysis of Results chapter on page 15 for a discussion of how these values were calculated and what they mean.

Client	Average (ms)	95th Percentile (ms)	99th Percentile (ms)
1	2.776	4	12
2	2.754	4	11
3	6.225	9	22
4	2.592	4	15
5	5.619	9	26
6	3.854	9	18
7	3.44	6	14
8	3.31	6	13
9	3.072	6	17
10	2.654	5	14
11	3.287	5	21
12	4.42	7	22
13	5.793	9	26
14	2.575	4	19
15	3.425	6	14
16	3.475	8	14
17	3.971	8	16
18	2.607	4	18
19	3.123	5	19
20	3.028	6	15
21	5.008	8	24
22	2.607	4	16
23	2.842	5	10
24	2.914	5	15
25	3.894	7	20
26	3.24	5	21
27	3.099	4	16
28	3.119	6	17
29	3.478	6	12
30	3.348	5	11

Table 1: Sample averages of latency from 30 clients

Network Utilization

As you can see in Figure 4, the frontplane is the network interface connected to the clients. The backplane is the network interface connected to TIBCO EMS.

Frontplane

Although the payload size to the application in this benchmark was 512 bytes per message, the approximate protocol overhead was 235 bytes. Thus each message averaged 747 bytes in size.

Calculation for NIC utilization:

Earlier we saw that there are 240,000 messages per second, at 747 bytes per message. This results in network utilization on the frontplane of 1.3 Gb/s (Gigabits per second).

Backplane

Due to the fanout capability of KWG, the network traffic from TIBCO EMS to the Gateway is significantly less.

Earlier we saw that there are 1,000 messages per second from TIBCO EMS to KWG. This results in a network utilization of less than 0.008 Gb/s. i.e., Less than 1 MB per second.

CPU Utilization

CPU usage on the Kaazing WebSocket Gateway server was 80%.

Memory Utilization

Kaazing WebSocket Gateway is Java-based, and therefore runs in a JVM. The JVM in this benchmark was configured to to use 2,688 MB of memory (i.e. less than 3 GB RAM).

Message Delivery

There was **zero** message loss, 100% of messages were successfully delivered.

Summary of Results

This is a summary of the key data from the results:

Number of client connections	24,000
Updates to each client	10 messages per second
Total messages throughput	240,000 messages per second
Average latency	3.5 ms
95th percentile latency	6 ms
99th percentile latency	16.9 ms
Hardware	One Dell PowerEdge R620 server
Network utilization	1.3 Gb/s
CPU utilization	80%
Memory utilization	Less than 3 GB RAM

Analysis of Results

This section contains some discussion and analysis of the results.

Latency

Latency in this benchmark is defined from the time the message broker first receives a message until it is finally received on the client. (See the **Latency** section of the Methodology chapter (page 7) for how latency is defined in this benchmark.)

A common approach in benchmarking, as done in this case, is to generate a large number of client connections to drive load while measuring performance using a different client in parallel. However, all too often in benchmarking, only the results of a single client get reported, which can give misleading results.

As an example, look at Table 1 (page 12). If the results from only one client were reported, it may have been for client 4 which would lead you think the average latency was 2.592 milliseconds. Or it could equally have been client 3 which would lead you to think the average latency was 6.225 milliseconds.

Both of those results might appear valid on the surface, but are actually misleading. Using a single client can accidentally yield the wrong results or allow one to cherry pick a good result.

Ideally one would measure all results from all clients but this is impractical in high-load scenarios as processing and storing results can consume resources, possibly influencing the test. A good middle ground is to use a suitable sample size, which should reduce the chance of outliers giving a false impression.

Because we are sampling the population rather than measuring the latency of every message to every client, a statistical level of confidence was calculated. This is the same methodology used whenever sampling from a larger population and is used in many different industries and circumstances from election polling to pharmaceutical drug testing.

As reported in the Benchmark Results chapter, we have the following result:

- The population mean latency is 3.5 +/- 0.35 milliseconds with a confidence interval of 95%.

  This means that if the same test were repeated multiple times, 95% of the time the latency would fall between 3.15 ms and 3.85 ms (which is 3.5 – 0.35 and 3.5 + 0.35 respectively).

Fairness

The concept of fairness is important for many businesses. If you are streaming market data to your customers, for example, you want to ensure that as many of your customers get the same data as close to the same time as possible. One way to measure fairness of distribution is using the 95th and 99th percentiles.

As reported in the Benchmark Results chapter, we have the following results:

- The population mean 95th percentile is 6.0 +/- 0.61 milliseconds with a confidence interval of 95%.

  This means that if the same test were repeated multiple times, 95% of the time the 95th percentile would fall between 5.39 ms and 6.61 ms (which is 6.0 – 0.61 and 6.0 + 0.61 respectively).

- The population mean 99th percentile is 16.9 +/- 1.55 milliseconds with a confidence interval of 95%.

  This means that if the same test were repeated multiple times, 95% of the time the 99th percentile would fall between 14.45 ms and 18.85 ms (which is 16.9 – 1.55 and 16.9 + 1.55 respectively).

Network Utilization

The area of interest for network utilization is on the frontplane of Kaazing WebSocket Gateway since that is where all of the clients are connected and that is where the majority of the data flows (see Figure 4, page 10).

The throughput on the frontplane for this benchmark is 1.3 Gb/s.

That is 130% of a 1 Gb/s NIC card. In a production system, most 1 Gb/s NIC cards won't sustain a throughput of the full 1 Gb/s and will typically run at a lower rate. As an example, if a 1 Gb/s NIC has a sustained throughput of 0.8 Gb/s, say, then this result of 1.3 Gb/s is 163% of that NIC.

Therefore, for a server commonly configured with a 1 Gb/s on the front plane, Kaazing WebSocket Gateway is network-bound for the scenario in this report. Either an extra NIC is required, or a faster network card such as the 10 Gb/s NIC used in this benchmark.

Fanout: Connection Offloading

The overall architectural scalability is greatly enhanced thanks to Kaazing WebSocket Gateway's ability to do connection offloading.

Figure 5 shows that in this benchmark, Kaazing WebSocket Gateway offloaded 24,000 connections, while requiring only a single connection from TIBCO EMS. As more users connect, there is no additional pressure on TIBCO EMS.

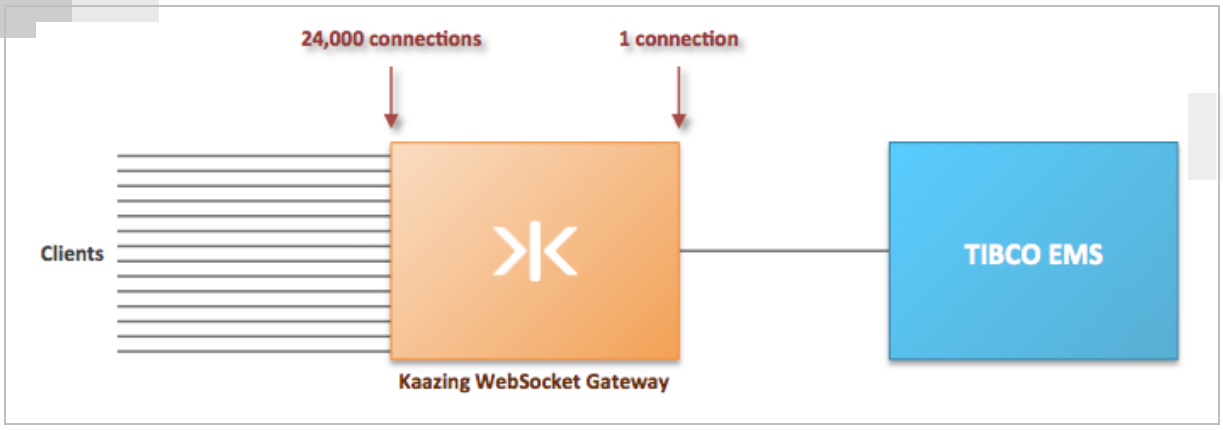

Figure 5: Connection offloading

Fanout: Subscription offloading

The ingress data on the backplane is small compared to the egress data on the frontplane due to the Gateway's subscription offloading capability. Scalability is greatly enhanced because TIBCO EMS only needs to send one message for all subscribed users, which gets fanned out in a highly efficient and performant fashion by KWG.

In this benchmark, the fanout ratio for each topic is the number of concurrent clients divided by 100 (since there are 100 topics shared by all users). Therefore in the case of 24,000 concurrent clients, the fanout ratio per topic is 240:1.

Figure 6 shows a single message sent from TIBCO EMS that is fanned out to the 240 clients, on average, that are subscribed to receive that message.

As more users subscribe to topics which are already subscribed to, no additional pressure is put on TIBCO EMS.

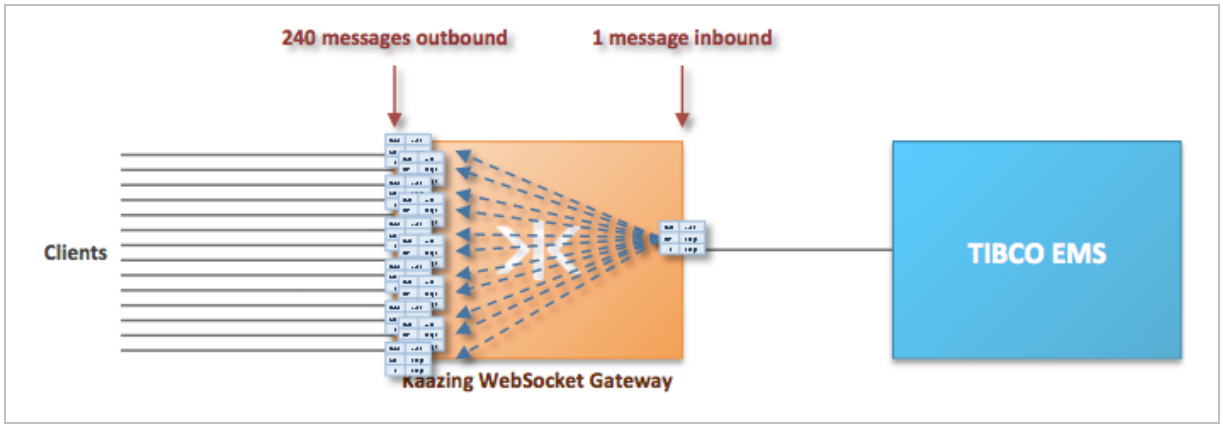

Figure 6: Subscription offloading

Linear Horizontal Scalability

The benchmark results have shown that for the given scenario, Kaazing WebSocket Gateway - JMS Edition running on a single Dell PowerEdge R620 1U rack server can support 24,000 concurrent connections at 10 messages per second with a latency of 3.5 milliseconds.

As discussed in the **Linear Horizontal Scalability** section of the Methodology chapter, KWG does not share client application state across the cluster. Each Gateway running in the cluster is independent of the others and they have no impact on each other. Therefore KWG has linear horizontal scalability.

Once you know a system scales linearly, you can use that to extrapolate and do capacity planning. In this case we can extrapolate to a fully populated rack and see what results can be expected.

A 48U full-height rack enclosure might typically have 4 used for power, networking, or other utilities, leaving 44 units available. One of the 44 units is designated for running TIBCO EMS, which means there is space for 43 1U rack servers to each run Kaazing WebSocket Gateway in a cluster.

Since each rack server can support 24,000 clients, 43 rack servers can support 24,000 * 43 = 1,032,000 concurrent clients (see Figure 7). It is safe to make this extrapolation because as discussed earlier, adding new Gateways to a cluster has no impact or detriment on any other node; they operate independently.

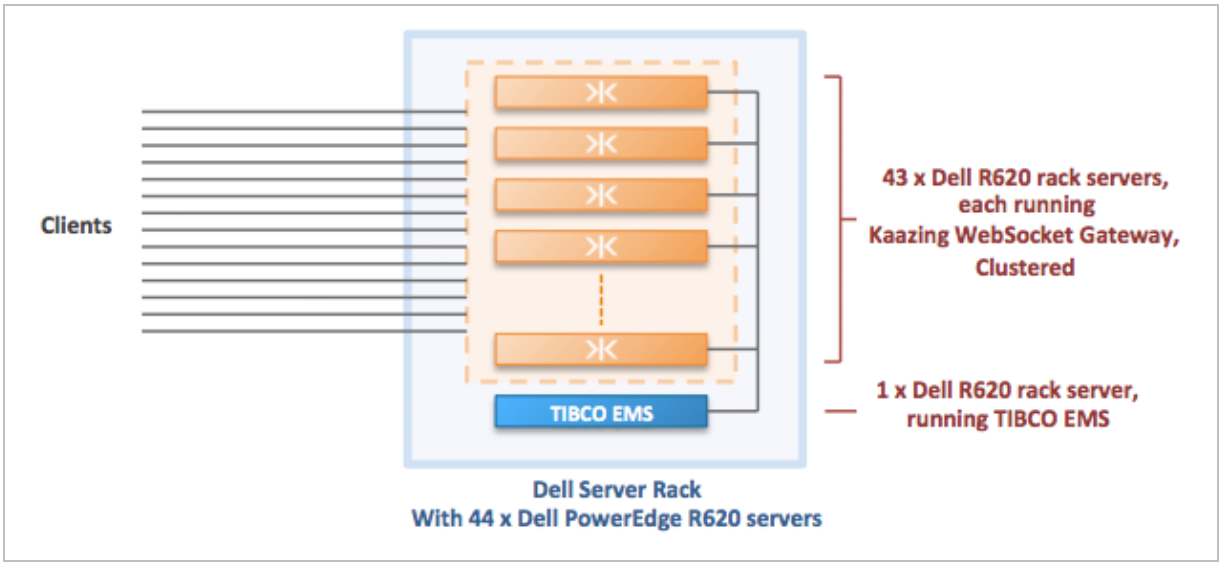

Figure 7: A single Dell PowerEdge R620 rack which can support a million concurrent users

Impact on TIBCO EMS

As the number of KWG cluster members increases, the impact on TIBCO EMS and its

ability to serve all the nodes of the cluster needs to be examined.

From the Results chapter, in the case of a single KWG instance, there was only 1 connection to TIBCO EMS, and the throughput was less than 0.008 Gb/s (or less than 1 MB/s).

Extrapolating that to the fully populated rack scenario, that would mean there are 43 connections to TIBCO EMS with a throughput of 43 * 0.008 = 0.344 Gb/s.

Those metrics are within the capability of both TIBCO EMS and a single 1 Gb/s NIC.

Summary and Conclusion

Here is a summary of the benchmark results for a single instance of Kaazing WebSocket Gateway:

Number of client connections	24,000
Updates to each client	10 messages per second
Total messages throughput	240,000 messages per second
Average latency	3.5 ms
95th percentile latency	6 ms
99th percentile latency	16.9 ms
Hardware	One Dell PowerEdge R620 1U rack server
Network utilization	1.3 Gb/s
CPU utilization	80%
Memory utilization	Less than 3 GB RAM

Due to the linear horizontal scalability of Kaazing WebSocket Gateway, these results can be safely extrapolated to a Dell 48U rack:

Number of client connections	1,032,000
Updates to each client	10 messages per second
Total messages throughput (per server)	240,000 messages per second
Average latency	3.5 ms
95th percentile latency	6 ms
99th percentile latency	16.9 ms
Hardware	One Dell PowerEdge R620 48U rack (With 43 1U rack servers each running KWG, and 1 rack server for TIBCO EMS)
Network utilization (per server)	1.3 Gb/s
CPU utilization (per server)	80%
Memory utilization (per server)	Less than 3 GB RAM

Conclusion

Kaazing began delivering its next generation Web infrastructure, Kaazing WebSocket Gateway, in June 2009, and it is the world's only enterprise solution for full-duplex, high-performance communication over the Web. Since its first release, Kaazing's platform has changed how business and organizations see their online presence and business, and more importantly, has opened ways for customers to pursue new business opportunities.

The simplicity, reliability, and stability of Kaazing's solution in combination with its superior scalability and raw performance makes it the ideal solution for any customer looking at application modernization, mobility, cloud deployment, and overall performance improvements to existing legacy Web solutions.

In addition, Kaazing's WebSocket Gateway delivers a number of technical enterprise features including:

- Full support for the new HTML5 WebSocket Standard
- Full-duplex, Bi-directional Communication over the Web
- Mobile Browser Support
- Enterprise-level Security
- Single Sign-On over the Web, including Kerberos support
- Suitable for the DMZ to Protect Trusted Systems
- Clustering, Load Balancing, and Failover
- Disaster Recovery
- Cross-Origin Connectivity
- WebSocket Emulation for Web Browsers, Mobile Devices, and other Client-side technologies
- JMX Management and Monitoring

For more information or a free trial: http://kaazing.com/about/contact.html

About Kaazing

Kaazing (www.kaazing.com) is the leading company focused on enabling the Living Web(TM) - the nearly ubiquitous real-time, interactive, collaborative applications that have become the norm. Kaazing's founding team played a key role in defining the WebSocketß standard and the Kaazing platform is the world's only enterprise solution for full-duplex, high-performance Web communication supporting the HTML5 WebSocket standard. Based in Mountain View, CA, Kaazing is privately held, with international channels throughout Europe, Asia, and Africa. Some of its customers and partners include Cognizant, Informatica, Intel, TIBCO, USAA, and a variety of financial, ecommerce, transportation and entertainment companies. To learn more about Kaazing, please visit http://www.kaazing.com or http://blog.kaazing.com, and follow @Kaazing on Twitter.