Next-Generation AMQP Messaging Performance, Architectures, and Ecosystems with Red Hat Enterprise MRG

Bryan Che
MRG Product Manager
Red Hat, Inc.

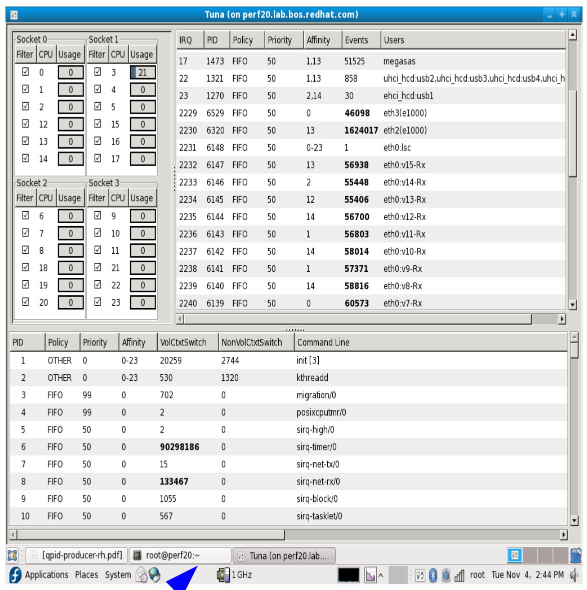
Carl Trieloff
Senior Consulting Software Engineer/ Director MRG
Red Hat, Inc.

Comment from a MRG Market data customer

“After following for few years the progress of the open standard messaging AMQP development, our company was excited to see Red Hat's contribution to the Qpid open source effort in farther developing the messaging product. Their resulting messaging product (MRG) allows our company to deliver a mission critical trading service leveraging messaging features intrinsic to financial workflows and providing outstanding performance.” - MRG customer

View of market data slice with MRG

-- AMQP based trading system deployment --



Tuna (on perf20.lab.bos.redhat.com)

Socket 0

Filter	CPU	Usage
✓	0	0
✓	1	0
✓	2	0
✓	12	0
✓	13	0
✓	14	0

Socket 1

Filter	CPU	Usage
✓	3	21
✓	4	0
✓	5	0
✓	15	0
✓	16	0
✓	17	0

Socket 2

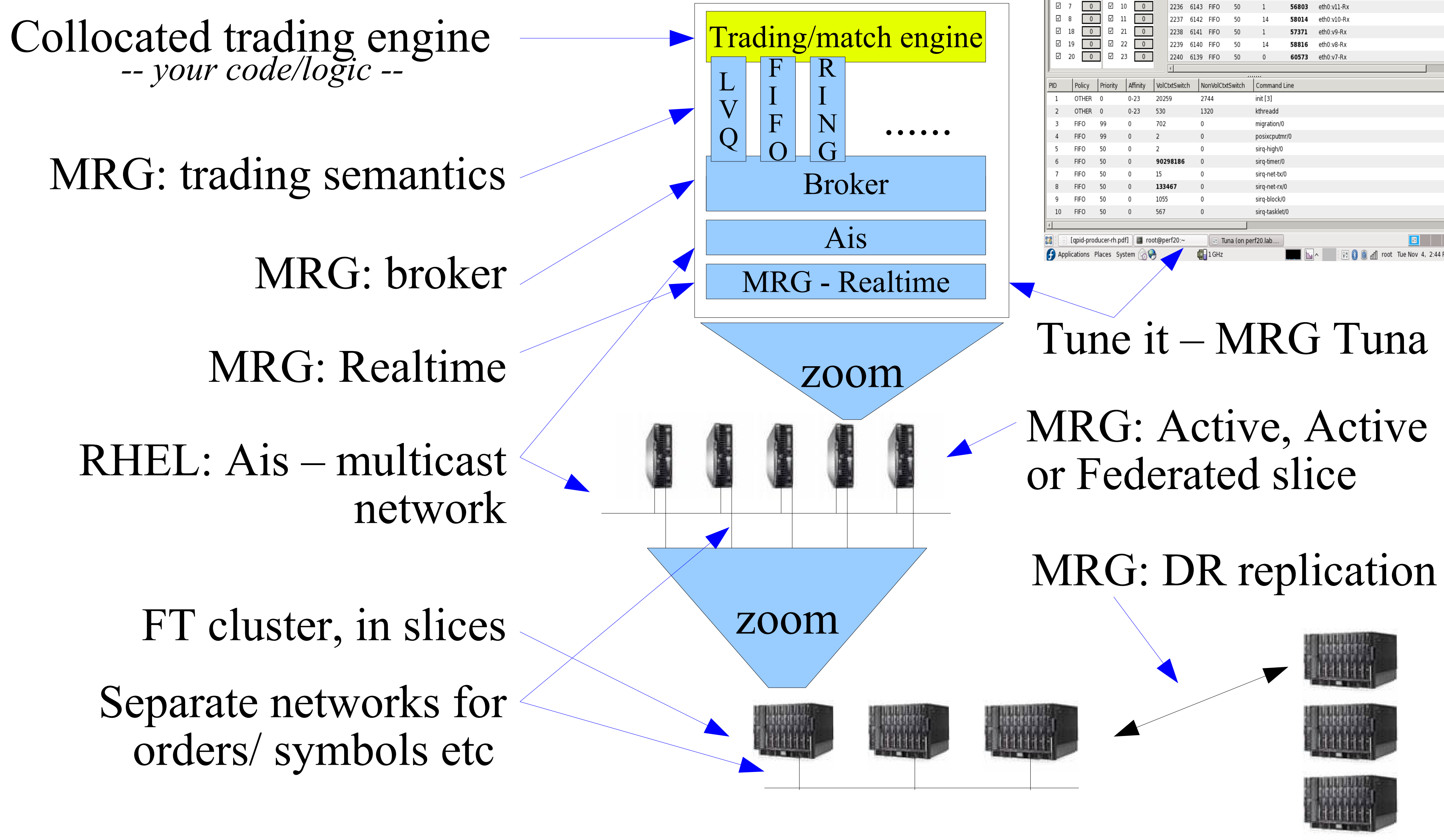
Filter	CPU	Usage
✓	6	0
✓	7	0
✓	8	0
✓	18	0
✓	19	0
✓	20	0

Socket 3

Filter	CPU	Usage
✓	9	0
✓	10	0
✓	11	0
✓	21	0
✓	22	0
✓	23	0

IRQ	PID	Policy	Priority	Affinity	Events	Users
17	1473	FIFO	50	1,13	51525	megasas
22	1321	FIFO	50	1,13	858	uhci_hcd:usb2,uhci_hcd:usb3,uhci_hcd:usb4,uhci_h
23	1270	FIFO	50	2,14	30	ehci_hcd:usb1
2229	6529	FIFO	50	0	**46098**	eth3(e1000)
2230	6320	FIFO	50	13	**1624017**	eth2(e1000)
2231	6148	FIFO	50	0-23	1	eth0:lsc
2232	6147	FIFO	50	13	**56938**	eth0:v15-Rx
2233	6146	FIFO	50	2	**55448**	eth0:v14-Rx
2234	6145	FIFO	50	12	**55406**	eth0:v13-Rx
2235	6144	FIFO	50	14	**56700**	eth0:v12-Rx
2236	6143	FIFO	50	1	**56803**	eth0:v11-Rx
2237	6142	FIFO	50	14	**58014**	eth0:v10-Rx
2238	6141	FIFO	50	1	**57371**	eth0:v9-Rx
2239	6140	FIFO	50	14	**58816**	eth0:v8-Rx
2240	6139	FIFO	50	0	**60573**	eth0:v7-Rx

PID	Policy	Priority	Affinity	VolCtxtSwitch	NonVolCtxtSwitch	Command Line
1	OTHER	0	0-23	20259	2744	init [3]
2	OTHER	0	0-23	530	1320	kthreadd
3	FIFO	99	0	702	0	migration/0
4	FIFO	99	0	2	0	posixcputmr/0
5	FIFO	50	0	2	0	sirq-high/0
6	FIFO	50	0	**90298186**	0	sirq-timer/0
7	FIFO	50	0	15	0	sirq-net-tx/0
8	FIFO	50	0	**133467**	0	sirq-net-rx/0
9	FIFO	50	0	1055	0	sirq-block/0
10	FIFO	50	0	567	0	sirq-tasklet/0



Illustrating trading semantics

-- setting up --

```
connection.open(host, port);
Session session =  connection.newSession();

// Create a queue named "message_queue", and route all messages whose
// routing key is "routing_key" to this FIFO queue.

session.queueDeclare(arg::queue="TICKER.NYSE", arg::exclusive=false);
session.exchangeBind(arg::exchange="amq.topic", arg::queue="TICKER.NYSE",
            arg::bindingKey="TICKER.NYSE.#");

session.queueDeclare(arg::queue="TICKER.NASDAQ", arg::exclusive=false);
session.exchangeBind(arg::exchange="amq.topic", arg::queue="TICKER.NASDAQ",
             arg::bindingKey="TICKER.NASDAQ.#");


// At this point we have two FIFO Queues for NYSE & NASDAQ
```

/* Fully worked example of this located in examples/tradedemo */

Illustrating trading semantics

--receive latest symbols --

```
void Listener::subscribeLVQQueue(std::string queue) {
 // Declare and subscribe to the queue using the subscription manager.
  QueueOptions qo;
  qo.setOrdering(LVQ);
  std::string binding = queue + ".#";
  queue += session.getId().getName();

  session.queueDeclare(arg::queue=queue, arg::exclusive=true, arg::arguments=qo);
  session.exchangeBind(arg::exchange="amq.topic", arg::queue=queue, arg::bindingKey=binding);
  subscriptions.subscribe(*this, queue, SubscriptionSettings(FlowControl::unlimited(), ACCEPT_MODE_NONE));
}

//  Then to subscribe....

        Listener listener(session);

        // Subscribe to messages on the queues we are interested in
           listener.subscribeTTLQueue("TICKER.NASDAQ");
           listener.subscribeTTLQueue("TICKER.NYSE");
           listener.subscribeLVQQueue("MRKT.NASDAQ");
           listener.subscribeLVQQueue("MRKT.NYSE");

        // Give up control and receive messages
        listener.listen();
```

Illustrating trading semantics

-- publish symbol data --

```
Message message;

std::string routing_key = "TICKER." + symbol;
std::cout << "Setting routing key:" << routing_key << std::endl;
message.getDeliveryProperties().setRoutingKey(routing_key);

curr_price = // { update the price ...  }

message.setData(curr_price);

// Set TTL value so that message will timeout after a period and be purged from queues
// This also creates a REPLAY window for late joining subscribers

message.getDeliveryProperties().setTtl(ttl_time);

// Asynchronous transfer sends messages as quickly as possible without waiting for confirmation.
async(session).messageTransfer(arg::content=message, arg::destination="amq.topic");
```

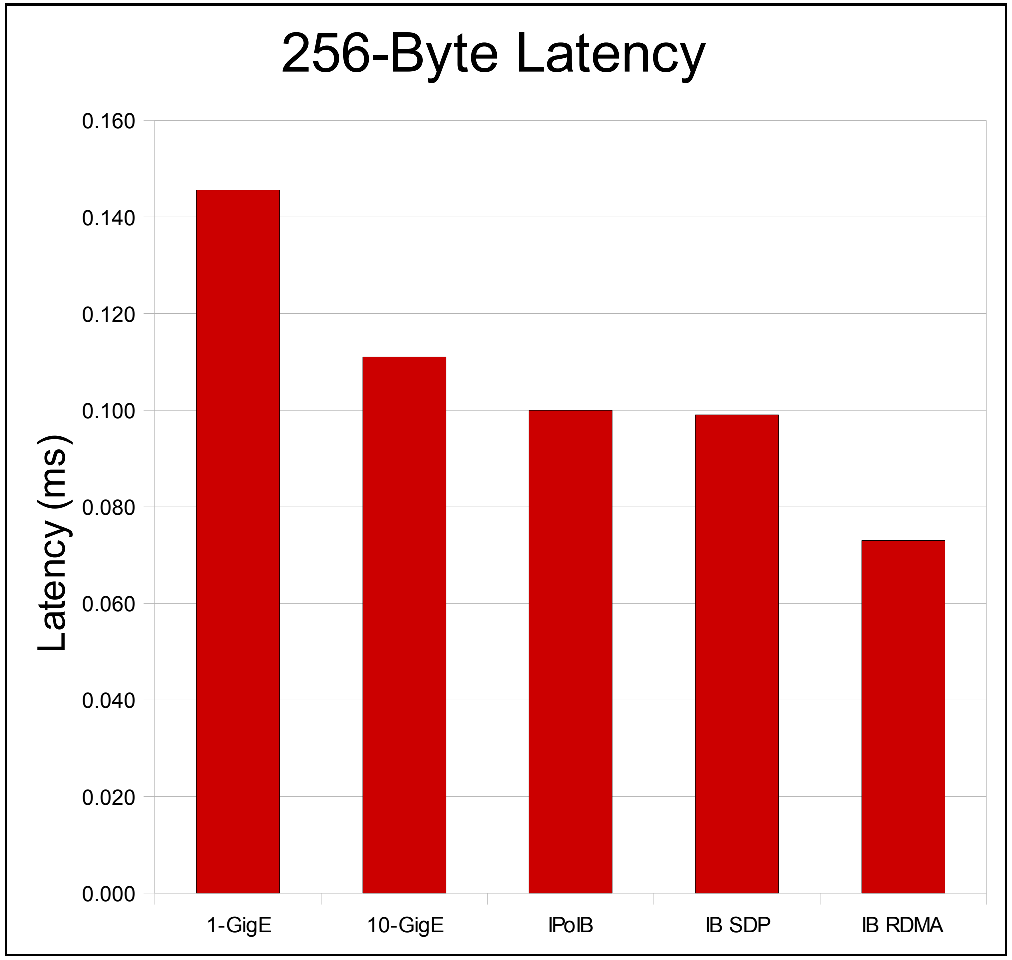
Illustrating trading semantics

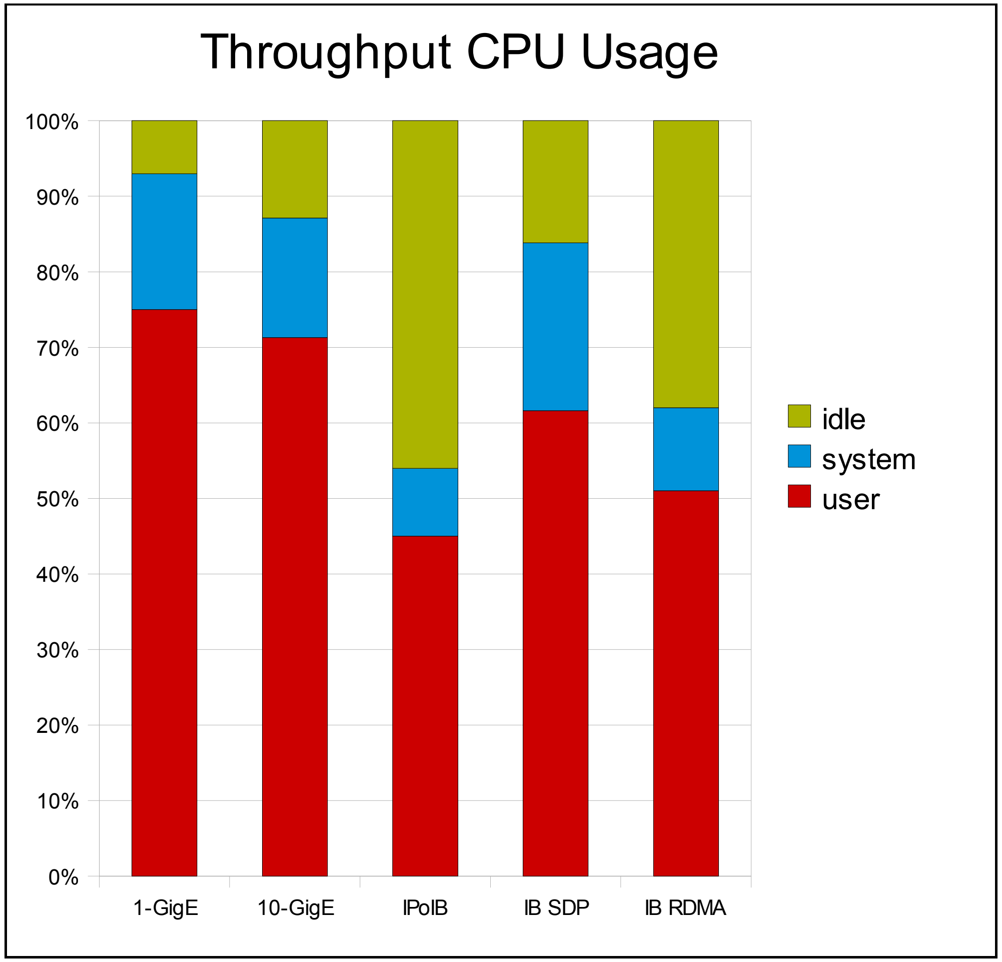
-- example consumer --

```
[MARKET] Symbol:NASDAQ.GOOG     Volume: 39350   Hi:125      Lo:113      MktCap:35796M       SEQ[485]
[TICKER] Symbol:NYSE.RHT        Price[20]    [0] [--]
[MARKET] Symbol:NYSE.RHT        Volume: 43165   Hi:24       Lo:8        MktCap:3800M        SEQ[486]
[TICKER] Symbol:NYSE.IBM        Price[37]    [1] [UP]
[MARKET] Symbol:NYSE.IBM        Volume: 36640   Hi:53       Lo:36       MktCap:49580M       SEQ[487]
[TICKER] Symbol:NASDAQ.MSFT     Price[25]    [1] [UP]
[MARKET] Symbol:NASDAQ.MSFT     Volume: 38089   Hi:26       Lo:8        MktCap:222250M      SEQ[488]
[TICKER] Symbol:NASDAQ.CSCO     Price[35]    [1] [UP]
[MARKET] Symbol:NASDAQ.CSCO     Volume: 39998   Hi:50       Lo:34       MktCap:205100M      SEQ[489]
[TICKER] Symbol:NASDAQ.YHOO     Price[8]     [0] [--]
[MARKET] Symbol:NASDAQ.YHOO     Volume: 38346   Hi:15       Lo:2        MktCap:11120M       SEQ[490]
[TICKER] Symbol:NASDAQ.GOOG     Price[114]   [0] [--]
[MARKET] Symbol:NASDAQ.GOOG     Volume: 40284   Hi:125      Lo:113      MktCap:35796M       SEQ[491]
[MARKET] Symbol:NYSE.RHT        Volume: 43989   Hi:24       Lo:8        MktCap:4180M        SEQ[492]
[TICKER] Symbol:NYSE.RHT        Price[22]    [2] [UP]
[MARKET] Symbol:NASDAQ.MSFT     Volume: 46230   Hi:26       Lo:8        MktCap:151130M      SEQ[596]
[MARKET] Symbol:NYSE.IBM        Volume: 43605   Hi:53       Lo:32       MktCap:42880M       SEQ[595]
[TICKER] Symbol:NASDAQ.MSFT     Price[23]    [2] [DOWN]
[TICKER] Symbol:NYSE.IBM        Price[37]    [0] [--]
[MARKET] Symbol:NASDAQ.CSCO     Volume: 47550   Hi:50       Lo:27       MktCap:158220M      SEQ[597]
[MARKET] Symbol:NYSE.RHT        Volume: 52990   Hi:28       Lo:8        MktCap:5320M        SEQ[594]
[TICKER] Symbol:NASDAQ.CSCO     Price[34]    [1] [DOWN]
[TICKER] Symbol:NYSE.RHT        Price[22]    [0] [--]
[MARKET] Symbol:NASDAQ.YHOO     Volume: 45910   Hi:15       Lo:2        MktCap:8340M        SEQ[598]
[TICKER] Symbol:NASDAQ.YHOO     Price[9]     [1] [UP]
[TICKER] Symbol:NYSE.IBM        Price[37]    [0] [--]
[MARKET] Symbol:NASDAQ.GOOG     Volume: 46082   Hi:125      Lo:111      MktCap:36110M       SEQ[599]
[TICKER] Symbol:NASDAQ.GOOG     Price[112]   [2] [DOWN]
```

Selecting the network fabric:

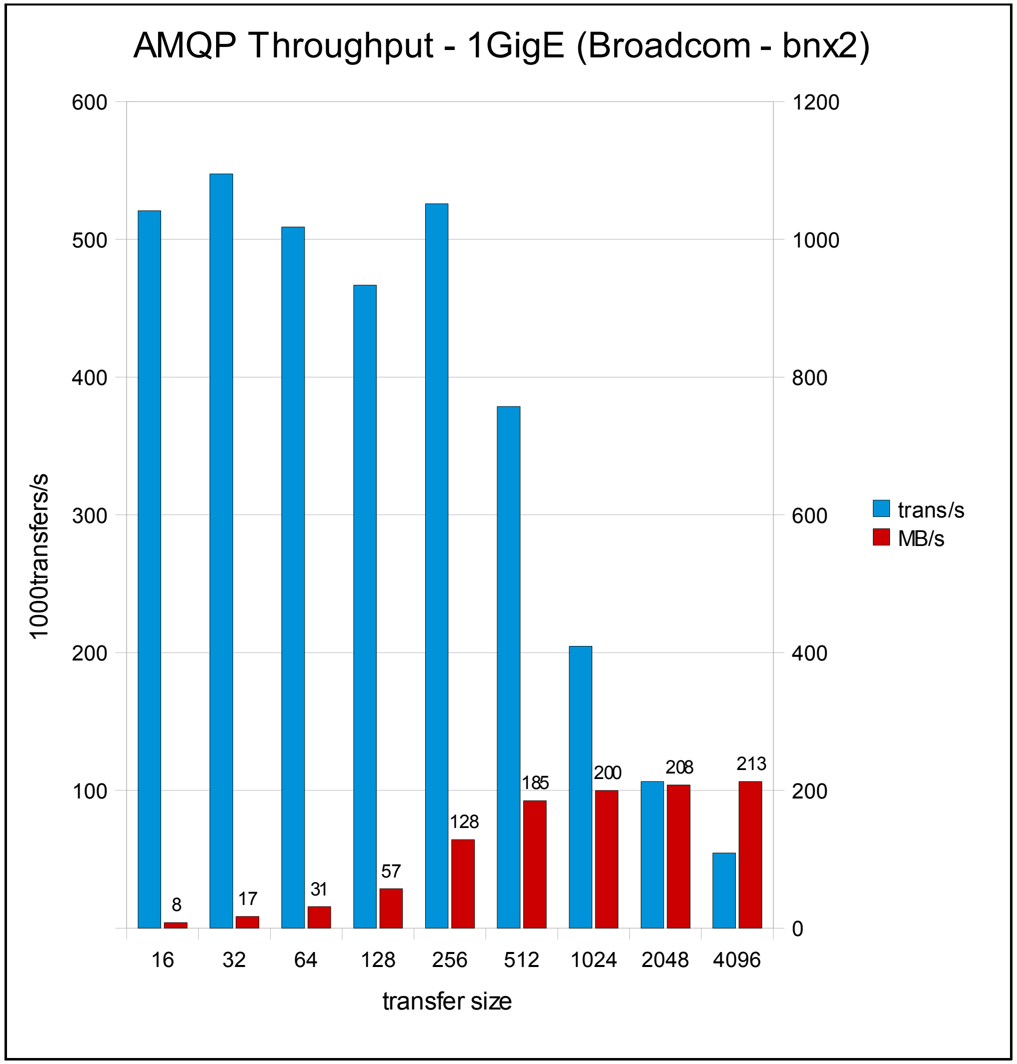
Comparing Latency per technology, per CPU cost at full load.

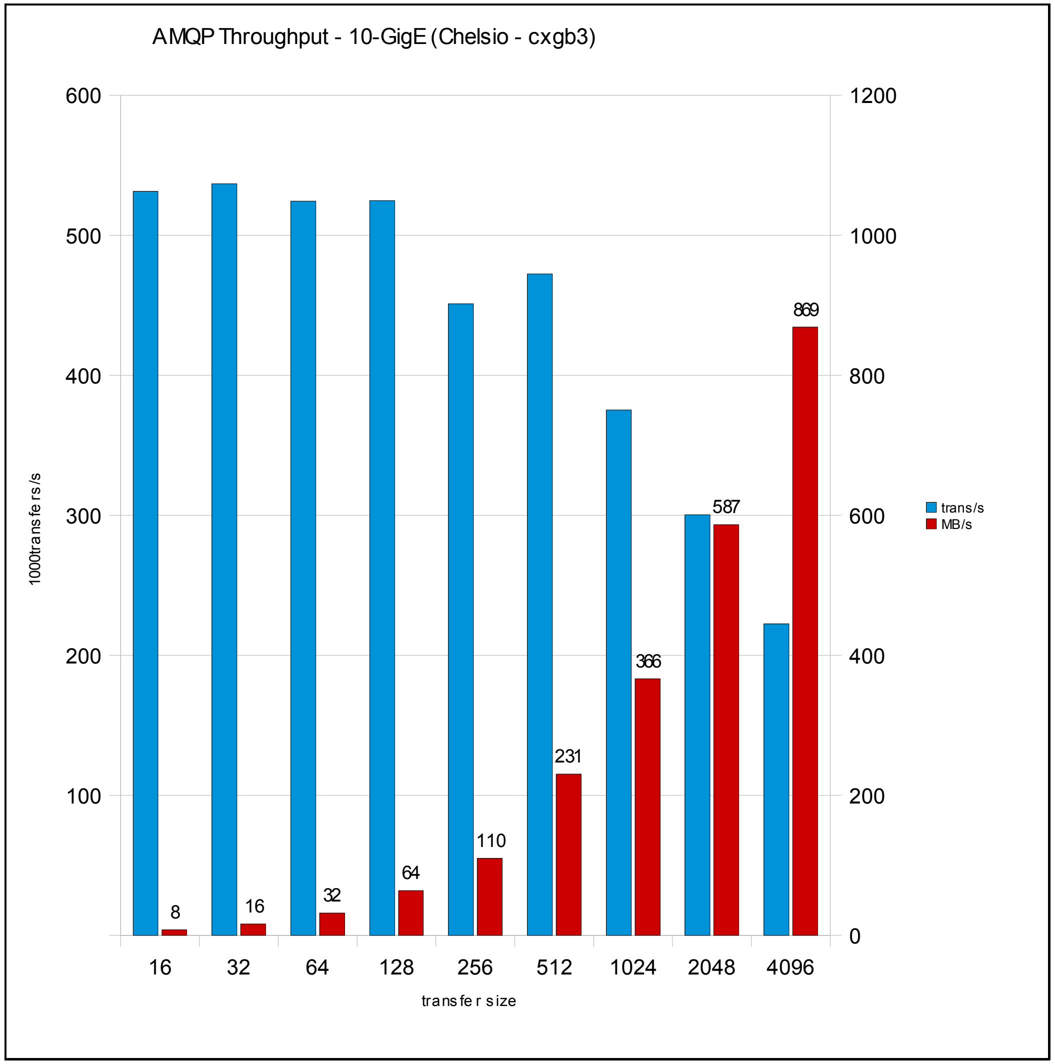




All measurements are AMQP between 3 peers (brokered) and fully reliable

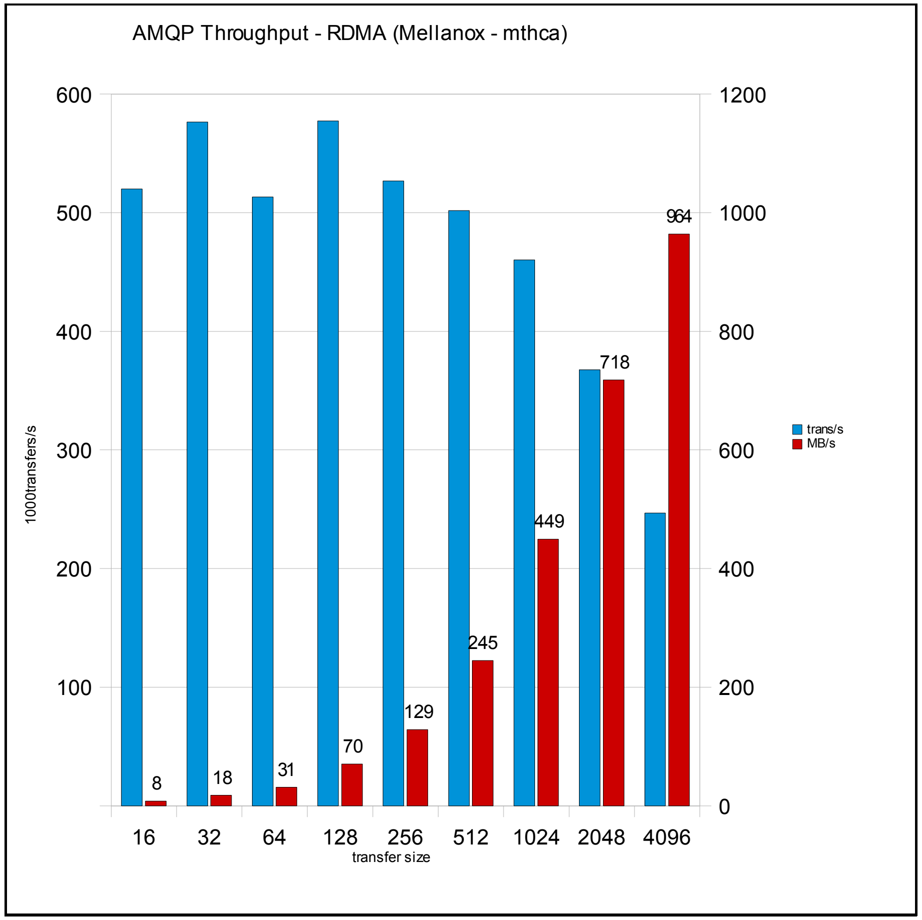
1 Gig versus 10 Gig, non-RDMA

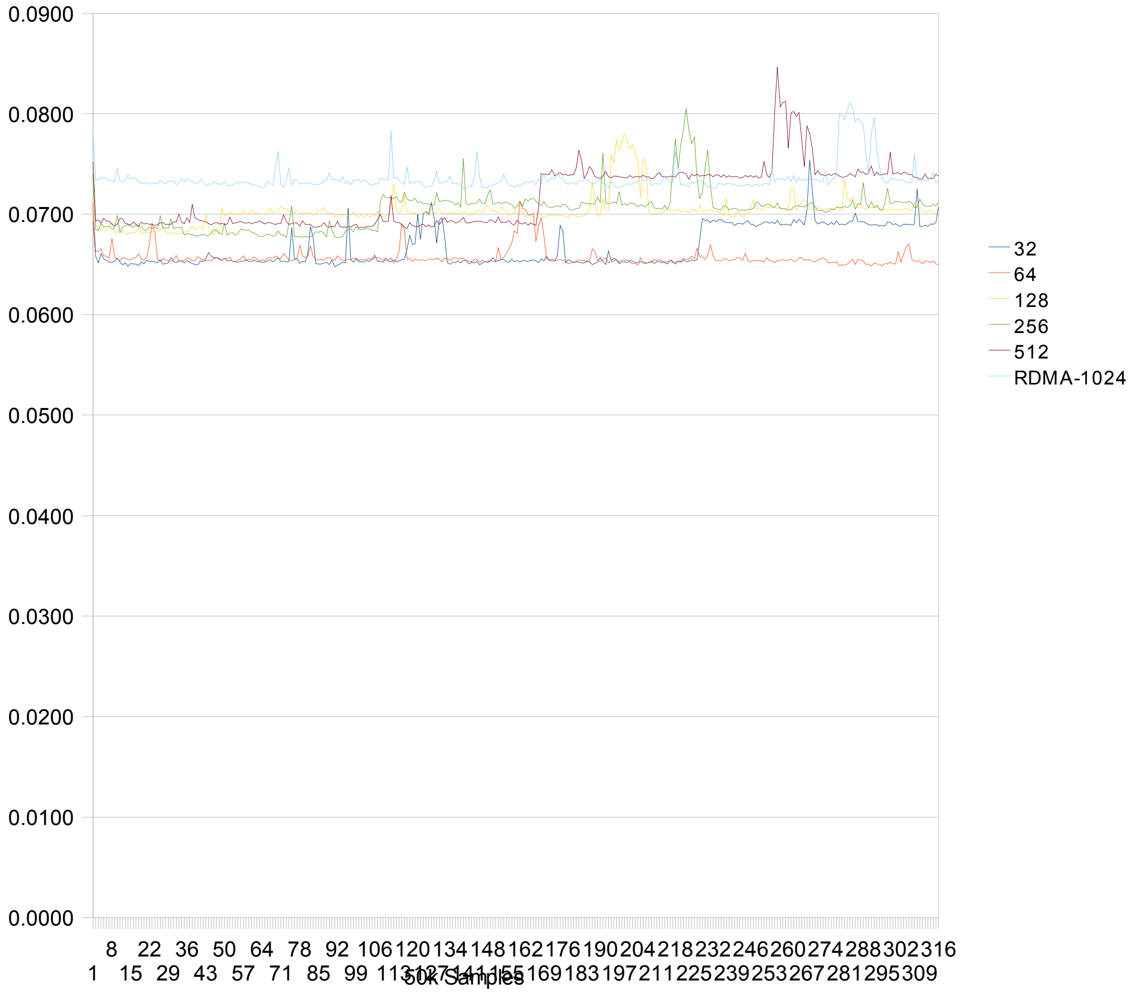




Rates and Throughput for 1 & 10G -- same load for direct comaparison

Messaging with native RDMA transport





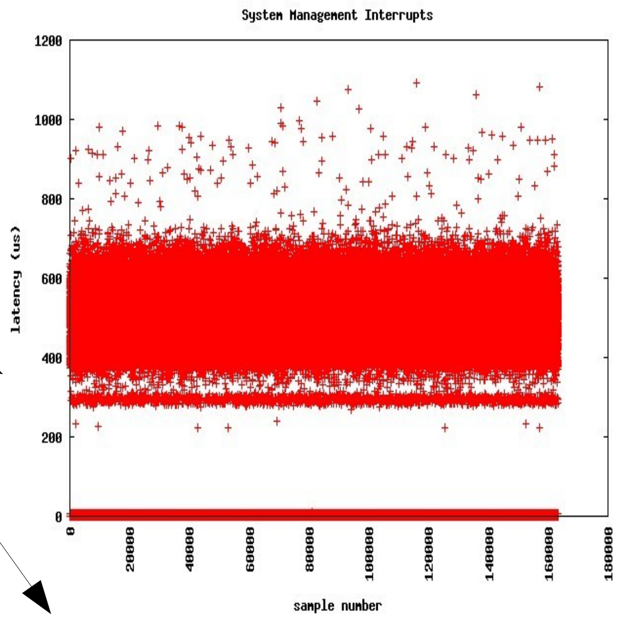
Rates, Throughput & Latency plot

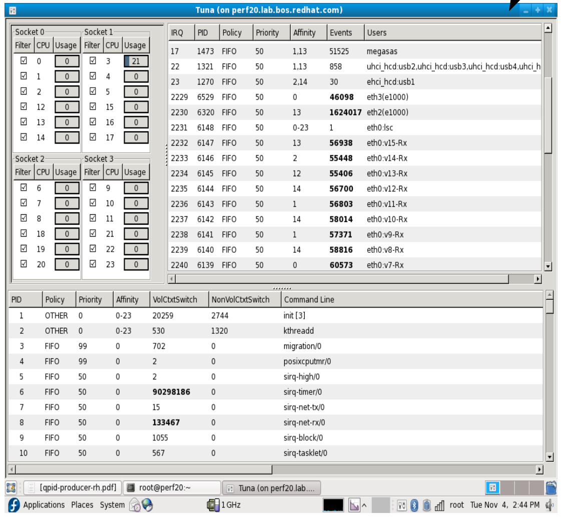
Dealing with other latency factors:

Impact of Realtime, SMIs, NUMA, Tuning, etc

Market Data needs good latency & required determinism, which means each components needs to be able to deliver. (A hardware effect will 'spot' through all the layers for example)

- Two graphs on right show dealing with SMI's on hardware (same box, with and without SMIs)
- Graph center below, contrasts kernel schedule latency from RHEL to MRG-Realtime
- Image left below, MRG-tuna for setting up affinity, memory effects etc





Socket 0

Filter	CPU	Usage
✓	0	0
✓	1	0
✓	2	0
✓	12	0
✓	13	0
✓	14	0

Socket 1

Filter	CPU	Usage
✓	3	21
✓	4	0
✓	5	0
✓	15	0
✓	16	0
✓	17	0

Socket 2

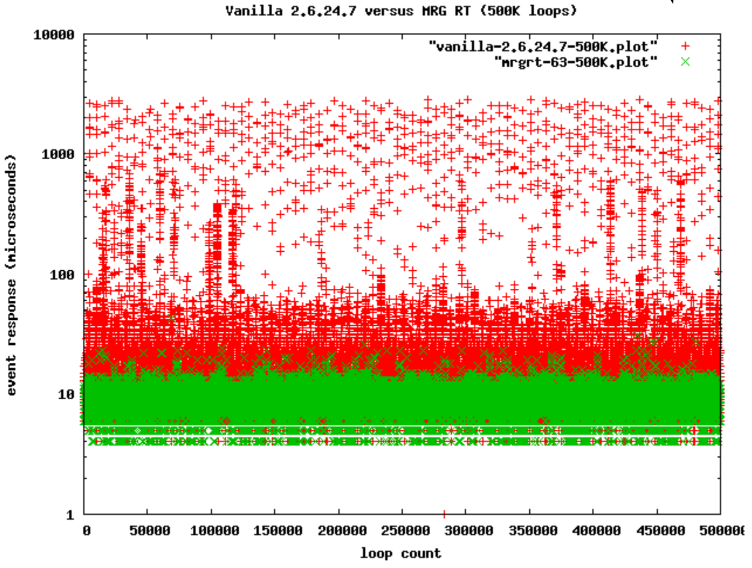
Filter	CPU	Usage
✓	6	0
✓	7	0
✓	8	0
✓	18	0
✓	19	0
✓	20	0

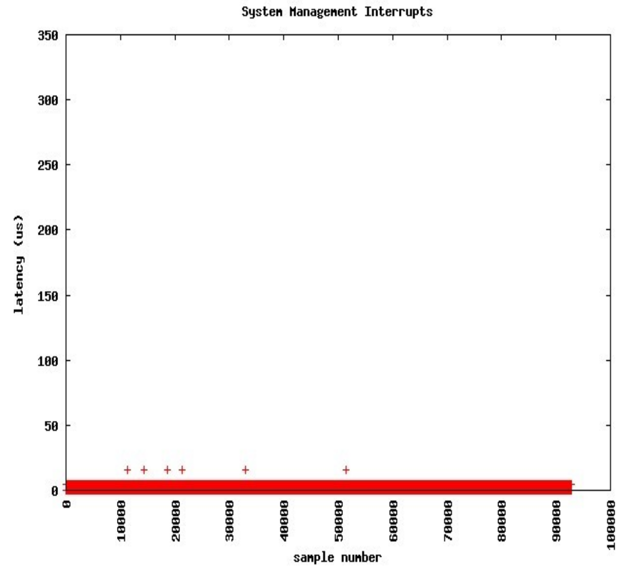
Socket 3

Filter	CPU	Usage
✓	9	0
✓	10	0
✓	11	0
✓	21	0
✓	22	0
✓	23	0

IRQ	PID	Policy	Priority	Affinity	Events	Users
17	1473	FIFO	50	1,13	51525	megasas
22	1321	FIFO	50	1,13	858	uhci_hcd:usb2,uhci_hcd:usb3,uhci_hcd:usb4,uhci_h
23	1270	FIFO	50	2,14	30	ehci_hcd:usb1
2229	6529	FIFO	50	0	**46098**	eth3(e1000)
2230	6320	FIFO	50	13	**1624017**	eth2(e1000)
2231	6148	FIFO	50	0-23	1	eth0:lsc
2232	6147	FIFO	50	13	**56938**	eth0:v15-Rx
2233	6146	FIFO	50	2	**55448**	eth0:v14-Rx
2234	6145	FIFO	50	12	**55406**	eth0:v13-Rx
2235	6144	FIFO	50	14	**56700**	eth0:v12-Rx
2236	6143	FIFO	50	1	**56803**	eth0:v11-Rx
2237	6142	FIFO	50	14	**58014**	eth0:v10-Rx
2238	6141	FIFO	50	1	**57371**	eth0:v9-Rx
2239	6140	FIFO	50	14	**58816**	eth0:v8-Rx
2240	6139	FIFO	50	0	**60573**	eth0:v7-Rx

PID	Policy	Priority	Affinity	VolCtxtSwitch	NonVolCtxtSwitch	Command Line
1	OTHER	0	0-23	20259	2744	init [3]
2	OTHER	0	0-23	530	1320	kthreadd
3	FIFO	99	0	702	0	migration/0
4	FIFO	99	0	2	0	posixcputmr/0
5	FIFO	50	0	2	0	sirq-high/0
6	FIFO	50	0	**90298186**	0	sirq-timer/0
7	FIFO	50	0	15	0	sirq-net-tx/0
8	FIFO	50	0	**133467**	0	sirq-net-rx/0
9	FIFO	50	0	1055	0	sirq-block/0
10	FIFO	50	0	567	0	sirq-tasklet/0





Swapping your transport

-- no code changes --

```
$./qpidd –help
        ...
  -- transport (tcp)       The transport for which to return the port
  -- load-module (file)   Specifies additional module(s) to be loaded
        ...
```

... two of these options allow for the loading of modules and setting a transport, more than one can ran at a time

TIP: *./qpidd –load-module some_module.so –help* will show the help options for the loaded module

Now we start the broker with RDMA module loaded and specified as default.

```
$./qpidd –load-module rdma.so –transport rdma
```

Note: that SSL, clustering, federation, ACL, store, XQuery routing etc can all be loaded in the same way.

There are quite a few interesting modules being build by the community, for example Google ProtocolBuffer support, SELinux based ACL, I have seen a trading engine in an exchange, etc...

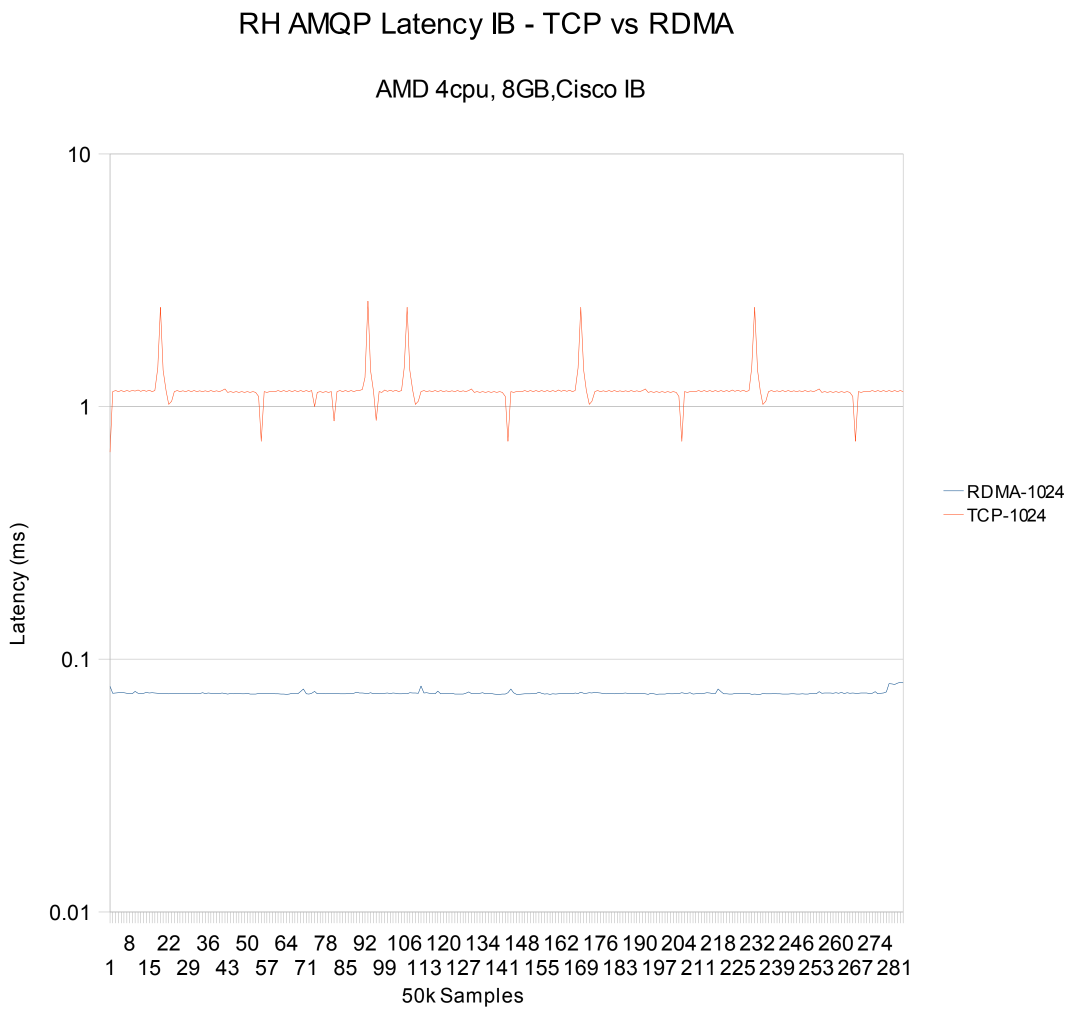
If you need something, come to the qpid project and help add it... qpid.apache.org

So, MRG & AMQP Can Build Stock Exchanges. ...But, Why Should I Care?

- AMQP and Red Hat Enterprise MRG are not just aiming to build next-generation versions of existing messaging-based systems
- Red Hat wants to build a fundamentally new messaging-based ecosystem that will transform the way we build software infrastructure
  - AMQP opens up new hardware ecosystems
  - AMQP and open source open up new software ecosystems and designs
  - AMQP provides true interoperability across ecosystems—even Linux and Windows

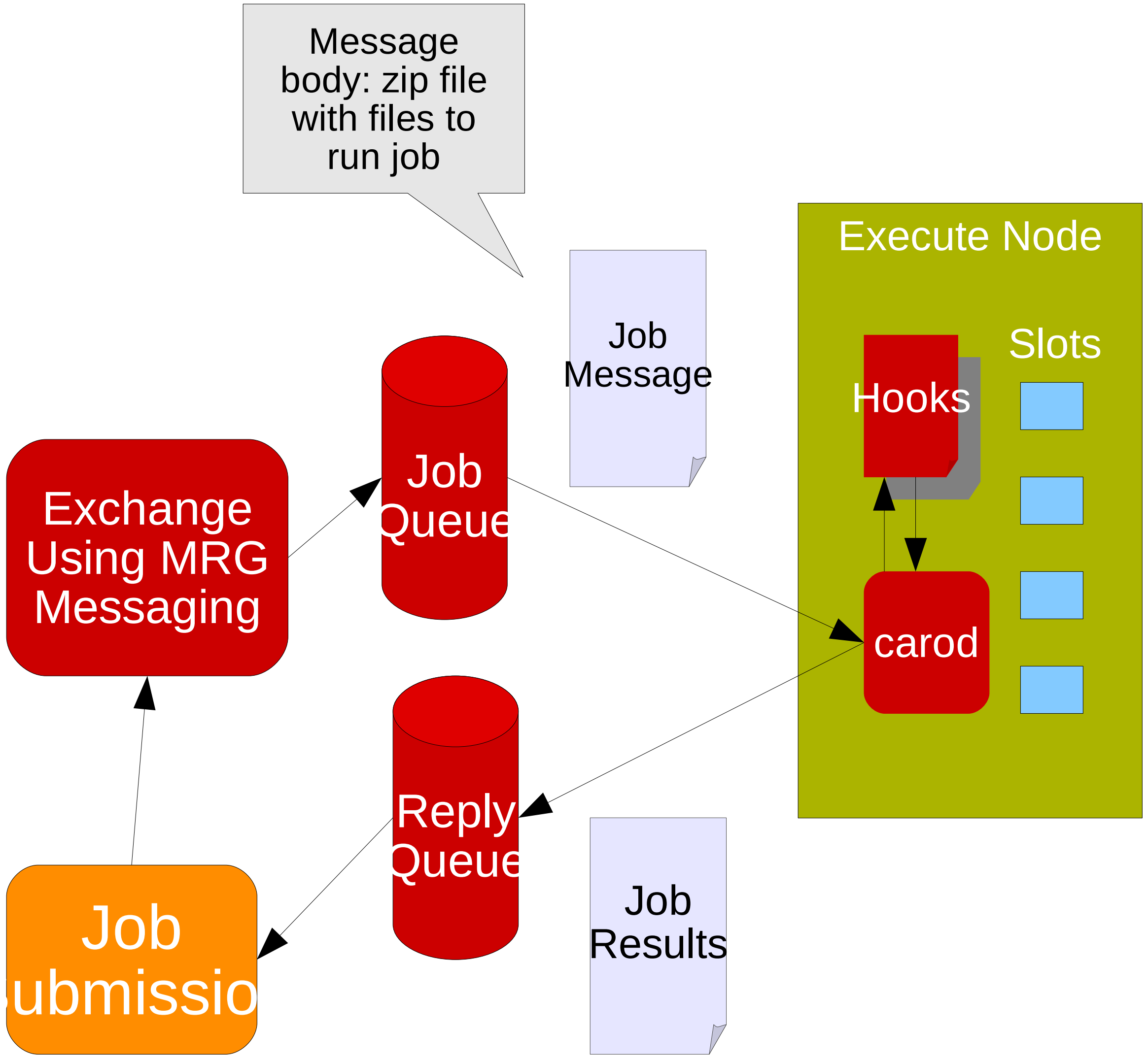
Messaging Hardware Ecosystem Examples

- Red Hat has partnered with hardware manufacturers like Intel and AMD to optimize performance for AMQP and Red Hat Enterprise MRG
- Cisco is an AMQP working group member and has demonstrated in-flight QoS and management for messaging
- Red Hat and Cisco have partnered to bring AMQP compatibility to legacy systems
- Red Hat Enterprise MRG can fully take advantage of modern hardware. Hardware upgrades can yield dramatic performance increases—not just incremental improvements



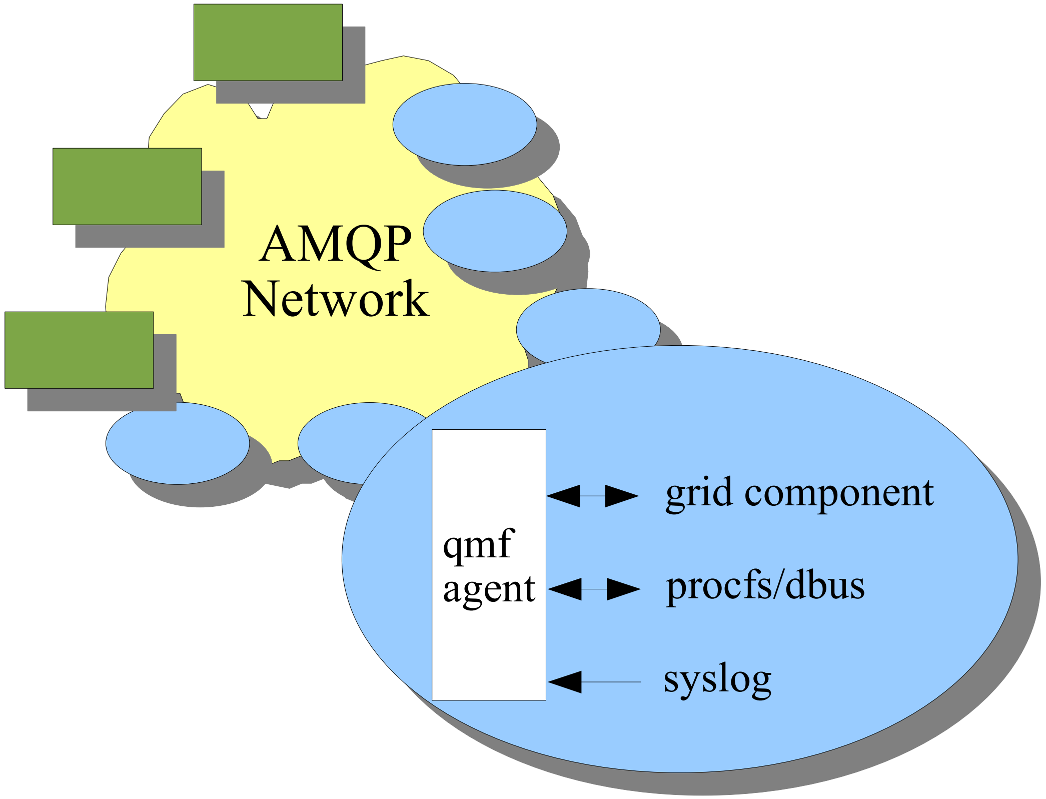
Messaging Software Ecosystem Examples

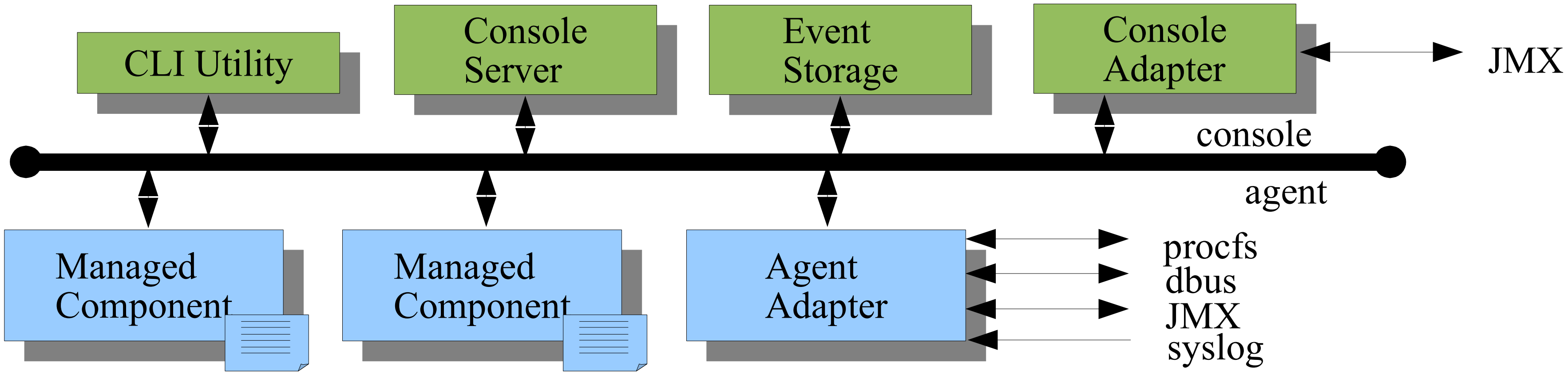
- MRG Grid provides low latency scheduling via messaging
  - Useful pattern for other systems
- MRG/Qpid provides features people often build on top of messaging
  - XML Exchange, LVQ, Ring Queue, TTL, Federation, Management, etc.
- Open Source projects are building on AMQP Messaging
  - OpenIPA project is using AMQP Messaging for management and monitoring of Identity, Policy, Audit systems
  - LibVirt project is using AMQP messaging for management and monitoring
  - Wireshark supports AMQP



QMF: Messaging Management Ecosystem

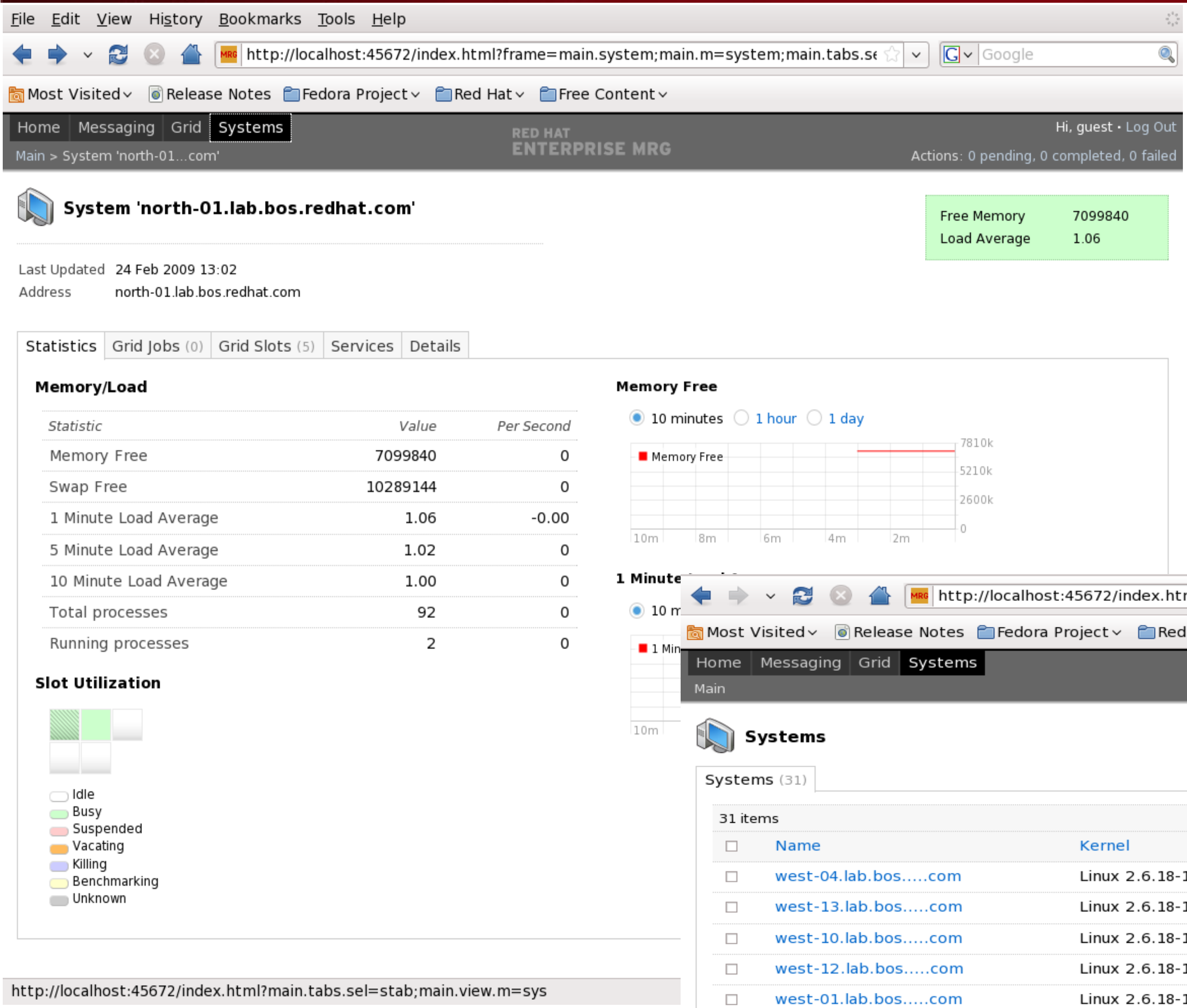
- Red Hat Enterprise MRG's entire management/monitoring system is AMQP messaging-based
  - Asymmetric, Efficient, Scalable, and Secure
  - Any messaging client can manage
- QMF: AMQP Messaging-Based Management Framework
  - Agent-defined management model (self-describing)
  - Objects (properties, statistics, and methods/controls), Events
  - Ease of development and extensibility





RED HAT
ENTERPRISE MRG
Messaging, Realtime, and Grid

Consoles, tools, operational data and control, for infrastructure & your application all using QMF


File Edit View History Bookmarks Tools Help
http://localhost:45672/index.html?frame=main.system;main.m=system;main.tabs.se
Google
Most Visited Release Notes Fedora Project Red Hat Free Content
Home Messaging Grid Systems
RED HAT ENTERPRISE MRG
Hi, guest • Log Out
Main > System 'north-01...com'
Actions: 0 pending, 0 completed, 0 failed

System 'north-01.lab.bos.redhat.com'

Free Memory	7099840
Load Average	1.06

Last Updated	24 Feb 2009 13:02
Address	north-01.lab.bos.redhat.com

Statistics | Grid Jobs (0) | Grid Slots (5) | Services | Details

Memory/Load

Statistic	Value	Per Second
Memory Free	7099840	0
Swap Free	10289144	0
1 Minute Load Average	1.06	-0.00
5 Minute Load Average	1.02	0
10 Minute Load Average	1.00	0
Total processes	92	0
Running processes	2	0

Slot Utilization

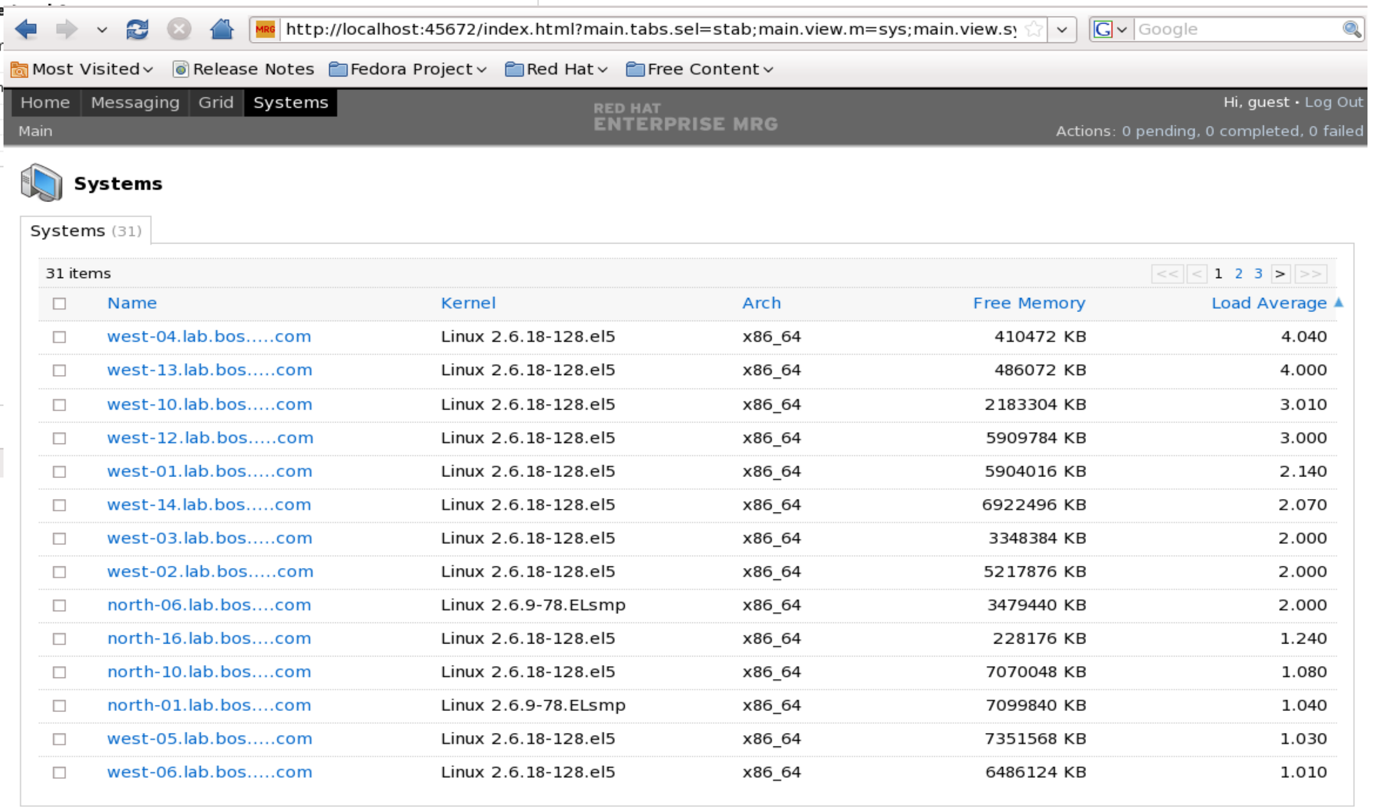
- Idle
- Busy
- Suspended
- Vacating
- Killing
- Benchmarking
- Unknown

Memory Free

[✓ 10 minutes] 1 hour 1 day

Memory Free
7810k
5210k
2600k
0
10m
8m
6m
4m
2m
1 Minute
10 m
1 Min
10m
http://localhost:45672/index.html?main.tabs.sel=stab;main.view.m=sys

System Stats & overview


http://localhost:45672/index.html?main.tabs.sel=stab;main.view.m=sys;main.view.sy
Google
Most Visited Release Notes Fedora Project Red Hat Free Content
Home Messaging Grid Systems
RED HAT ENTERPRISE MRG
Hi, guest • Log Out
Main
Actions: 0 pending, 0 completed, 0 failed

Systems

Systems (31)

31 items << < 1 2 3 > >>

Name	Kernel	Arch	Free Memory	Load Average
west-04.lab.bos.....com	Linux 2.6.18-128.el5	x86_64	410472 KB	4.040
west-13.lab.bos.....com	Linux 2.6.18-128.el5	x86_64	486072 KB	4.000
west-10.lab.bos.....com	Linux 2.6.18-128.el5	x86_64	2183304 KB	3.010
west-12.lab.bos.....com	Linux 2.6.18-128.el5	x86_64	5909784 KB	3.000
west-01.lab.bos.....com	Linux 2.6.18-128.el5	x86_64	5904016 KB	2.140
west-14.lab.bos.....com	Linux 2.6.18-128.el5	x86_64	6922496 KB	2.070
west-03.lab.bos.....com	Linux 2.6.18-128.el5	x86_64	3348384 KB	2.000
west-02.lab.bos.....com	Linux 2.6.18-128.el5	x86_64	5217876 KB	2.000
north-06.lab.bos....com	Linux 2.6.9-78.ELsmp	x86_64	3479440 KB	2.000
north-16.lab.bos....com	Linux 2.6.18-128.el5	x86_64	228176 KB	1.240
north-10.lab.bos....com	Linux 2.6.18-128.el5	x86_64	7070048 KB	1.080
north-01.lab.bos....com	Linux 2.6.9-78.ELsmp	x86_64	7099840 KB	1.040
west-05.lab.bos.....com	Linux 2.6.18-128.el5	x86_64	7351568 KB	1.030
west-06.lab.bos.....com	Linux 2.6.18-128.el5	x86_64	6486124 KB	1.010

MRG & AMQP Provide New Interoperability

- Red Hat provides messaging clients for multiple languages, including Java/JMS, .NET, C++, Python, Ruby, etc
- Red Hat and Microsoft are both members of the AMQP working group
  - Red Hat and Microsoft are both developing in the same upstream open source project: Apache Qpid -- see blog by Mircosoft's Sam Ramsi
  - This will drive significant interoperability between Linux and Windows systems. Both Linux and Windows will gain native AMQP capabilities
  - This will drive significant new interoperability between Java (with Red Hat's JBoss) and .NET
- AMQP will provide you with the confidence that if you build a distributed architecture using AMQP, you can count on its availability and interoperability across platforms
  - This will catapult messaging well beyond its already crucial place in software, just as standards like TCP and HTTP revolutionized networking and the Web

Additional Information

http://www.redhat.com/mrg

40 page report with all the data in it, available by request

Our emails:
cctrieloff@redhat.com
bche@redhat.com