



Monterey Reference Architecture Performance Test Results:

***Monterey West with BLADE Network Technologies
RackSwitch, Citrix XenServer, and Solarflare
10GbE Server Adapters***

Cloudsoft Technical Report

29 March 2010





Table of Contents

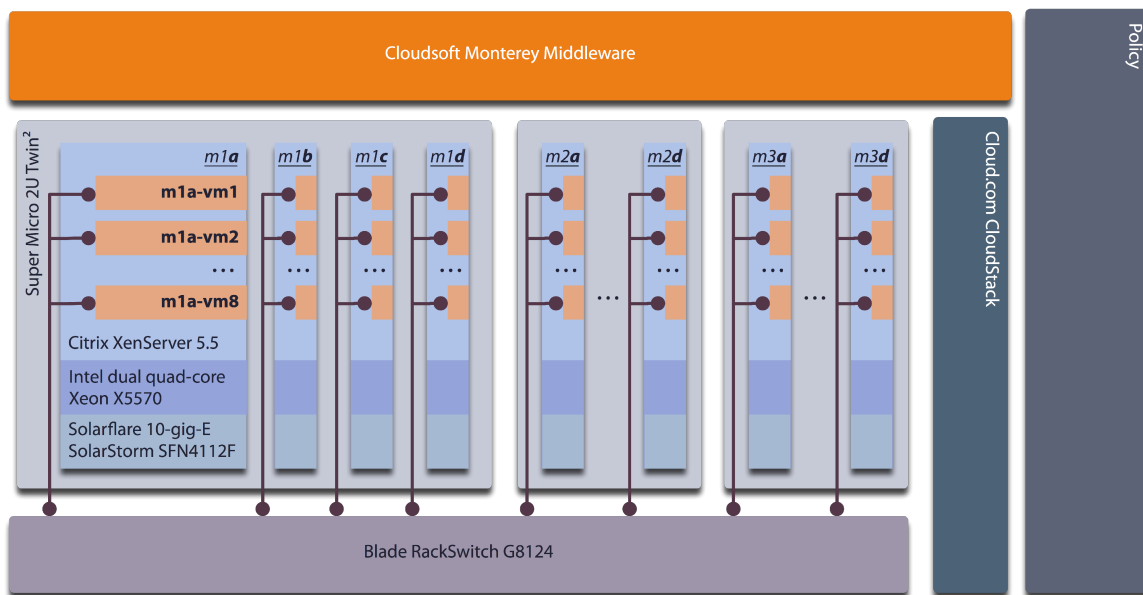
Monterey West Architecture.....	2
Standard Tests: Networking.....	3
Throughput across Two Hosts.....	3
Latency across Two Hosts.....	4
Intra-Host Performance.....	5
Standard Tests: Computation.....	6
Processing Speed.....	6
Hyperthreading.....	6
Monterey Application Tests.....	7
Throughput and Scalability.....	8
Round-Trip Time and Latency.....	9
About the Author.....	10



Monterey West Architecture

This report describes performance tests and their results carried out on the “Monterey West” hosted environment. This infrastructure and stack is hosted by ENKI PrimaCloud and is located in Milpitas, CA, and demonstrates the *Monterey Reference Architecture* for high-performance cloud. The specific components in the configuration tested for this report are:

- 3 x Super Micro 2U Twin²
- 24 x Intel Xeon X5570 Processors
- 12 x Citrix XenServer 5.5 software/OS
- 12 x Solarflare SolarStorm SFN4112F 10GbE server adapters
- 1 x Blade RackSwitch G8124 10GbE switch
- 96 x Debian Etch 32-bit Linux OS (standard XenServer ISO image)
- 96 x Cloudsoft Monterey Middleware software



One modification was made to the XenServer stack, installing the Solarflare SFN4112F drivers and the `sfc_back` module. All guests are standard Debian Etch VMs with a single network interface bridged to the Solarflare device, again with one modification, the `sfc_front` module installed for direct guest-to-hardware access. The hosts are identified as $m\{1, 2, 3\}\{a, b, c, d\}$ with 8 VMs on each, in the standard configuration. Unless otherwise noted each VM has a dedicated 1 GB RAM and 2 hyperthreaded cores (one physical core).

All VMs are on a single sub-net 172.16.7.x, with x equal to 11 through 22 for m1a, m1b, ..., m3d, and x equal to 100 through 195 for m1a-vm1, m1a-vm2, ..., m3d-vm8. All hosts and some guests are exposed externally selected ports on 208.95.232.y, where $y = 100+x$ for hosts and $y = 123 + (\text{int})((x-100)/4) + (x\%2)$ for guests satisfying $((\text{int})(x/2))\%4 = 2$ (corresponding to vm1 and vm2 on each host).



Standard Tests: Networking

Networking tests used one or both of two standard benchmarking tools, netperf and nuttcp, to evaluate throughput and latency at the guest OSs.

Highlights

- 18.5 Gbps bi-directional max throughput of network (shown by 4-bi-4)
- 8.0 Gbps uni-directional max throughput of process (shown by 1-uni-1)
- 26µs latency between guest VMs on different hosts

The latency measurements represent a major milestone, demonstrating near-native performance at guest operating systems. The throughput data replicates the “Killer Performance” results reported at:

<http://community.citrix.com/display/ocb/2009/04/13/Killer+Perf!+XenServer%2C+Nehalem+and+Solarflare>

There are no custom changes to the networking stack on hosts or guests beyond the installation and setup of the Solarflare drivers. (Some experimentation was performed with the networking settings in Linux at the hosts and at the guests but in general these resulted in at best very slight performance gains on some tasks with degradation on others. For this reason the configuration as shipped is used for all tests unless otherwise noted.)

Throughput across Two Hosts

Maximum throughput across a network connection was tested using nuttcp benchmark tool to gauge maximum throughput across a network connection. With this tool, one or more machines are set up as server endpoints (nuttcp -S) to receive data-intensive traffic; the tool is then run as a client from each host which sends data, directed to the target server as a command-line argument (nuttcp hostname). We used the default test parameters sending as much traffic as possible, measuring throughput over a 10s window.

Test	Expected Results	Actual Results
1-uni-1 (nut): one VM sending to one VM on a second host	3 to 10 Gbps	8.09 Gbps <i>mean of 10; SD 2%</i>
2-uni-2 (nut): two VMs on one host, sending to two VMs on a second host	4 to 10 Gbps total (2 to 5 Gbps per VM)	9.38 Gbps total (4.67 per VM) <i>mean of 10; SD <1%</i>
2-bi-2 (nut): two VMs on one host, one sending and one receiving, with respective VMs on a second host	6 to 20 Gbps total (3 to 10 Gbps per VM)	17.20 Gbps total (8.60 per VM) <i>mean of 10; SD 2%</i>
4-bi-4 (nut): four VMs on one host, two sending and two receiving, with respective VMs on a second host	6 to 20 Gbps total (3 to 5 Gbps per VM)	18.52 Gbps total (4.63 per VM) <i>mean of 10; SD <1%</i>



We have also explored a variety of netperf test configurations for throughput. The TCP_STREAM test with its default configuration (comparable to nuttcp above) yields very similar results:

Test	Expected Results	Actual Results
1-uni-1 (np): one VM sending to one VM on a second host	3 to 10 Gbps	7.45 Gbps <i>mean of 10; SD <1%</i>
2-uni-2 (np): two VMs on one host, sending to two VMs on a second host	4 to 10 Gbps total (2 to 5 Gbps per VM)	9.32 Gbps total (4.66 per VM) <i>mean of 10; SD <1%</i>
2-bi-2 (np): two VMs on one host, one sending and one receiving, with respective VMs on a second host	6 to 20 Gbps total (3 to 10 Gbps per VM)	15.31 Gbps total (7.66 per VM) <i>mean of 10; SD <1%</i>
4-bi-4 (np): four VMs on one host, two sending and two receiving, with respective VMs on a second host	6 to 20 Gbps total (2 to 5 Gbps per VM)	18.61 Gbps total (4.65 per VM) <i>mean of 10; SD <1%</i>

Latency across Two Hosts

Latency between VMs was computed both using ping and using the netperf -t TCP_RR benchmark, effectively measuring the amount of time for a single message to travel from one VM to another. (TCP_RR actually reports the number of request-response transactions which can be completed sequentially per second, where mathematically latency works out has half the reciprocal of this number.)

Ping reports latency between VMs on different hosts as 44µs (87µs round-trip, SD 7%, packet size 64B).

TCP_RR reports 19500 round-trip transactions per second with the default 1B packet size, operating across VMs on different hosts (SD 3%), equating to a latency of 26µs. For packets of up to 500B there is no noticeable decrease. Packets of 1kB show a 28µs latency, increasing to 121µs for 100kB packets. The final figure is derived from 4120 transactions per second (SD <1%). This test can be thought of as equivalent to data flowing uni-directionally between VMs at 6.5 Gbps plus processing overhead for each message during which data is not being transferred.

Disabling the Solarflare acceleration modules increases latency by approximately 50%.

Intel power-saving features have a significant effect on latency. These latency results are collected with EIST (SpeedStep), C-State, and Active Power State Management disabled in the BIOS. Enabling EIST and C-State (but not APSM) causes latency figures to increase by about 40%. These power-saving features do not have any noticeable effect on the other tests.



Intra-Host Performance

Throughput as measured with `nuttcp` is bounded by the 9.8 Gbps limit across VMs on a single host:

Test	Actual Results
loopback: one VM sending to itself	25.0 Gbps
1-to-1: one VM sending to VM on same host	9.8 Gbps
2-to-2: two VMs sending to VMs on same host	9.8 Gbps total, 4.91±0.05 Gbps per VM
4-to-4: four VMs sending to VMs on same host	9.8 Gbps total, 2.39±0.02 Gbps per VM

Across VMs on the same host latency is reported as 23±4 μs by `ping` and 18±2 μs by `TCP_RR`. Latency to localhost is reported as 2μs by `ping` and 6μs by `netperf -t TCP_RR`.



Standard Tests: Computation

The computation benchmarks we used rely on the following command:

```
time echo "scale=2000; 4*a(1)" | bc -l > /dev/null
```

to calculate π to 2000 places. The command `bc` is not the most efficient way to compute π , but it is ubiquitous and single-threaded, making it a good general-purpose way to gauge CPU speed. The command above typically takes 3 to 4 seconds on a modern core.

Test configurations explore the effects of running this in parallel within VMs restricted to a fixed number of cores and in multiple VMs.

Highlights

- computation in parallel on 8 VMs shows no degradation in performance
- hyperthreaded “cores” show 50% compute throughput compared with physical cores

Processing Speed

Test	Expected Results	Actual Results
Baseline: one process instance running on a 2-core VM	3 to 4 seconds user time, same for real (elapsed) time	3.01s user 3.02s real
1x8: 8 simultaneous instances, 1 on each of 8 2-core VMs	3 to 4 seconds user time, same for real (elapsed) time	3.01s user 3.02s real
8x1: 8 simultaneous instances on a single 2-core VM	3 to 4 seconds user time, max real time 12 to 16s	3.02s user 11.71s real

All results represent 10 runs, with “user” time (CPU time) being a mean and “real” time (reference clock) a max. Standard deviation is less than 1% on all tests.

Hyperthreading

Test	Expected Results	Actual Results
2x8: 16 simultaneous instances, 2 on each of 8 2-core VMs	3 to 6 seconds user time, max real time 3 to 8 seconds	4.34s user (mean), 5.01s real (max)
4x8: 32 simultaneous instances, 4 on each of 8 2-core VMs	3 to 6 seconds user time, max real time 6 to 12 seconds	4.53s user (mean), 9.20s real (max)



Monterey Application Tests

To demonstrate heterogenous workload, involving both networking and processing, and representative of real-world applications, we have used the EzBrokerage distributed stock exchange designed on the Cloudsoft Distributed Mediation platform-as-a-service layer and running on the Monterey infrastructure.

This application serves as a trading exchange handling bid and ask orders from users in real-time. The imperative in this application is that within each instrument (i.e. for each stock), orders are handled atomically, in the same sequence as they are submitted by a given user, and as quickly as possible. Traditionally, the twin constraints of exactly-once in-order delivery and consistent low-latency high-throughput have ruled out some approaches to scalability (such as optimistic locking) and demanded inflexible, over-provisioned compute environments. Cloudsoft Distributed Mediation offers a platform-level solution to this problem using patented algorithms to dynamically change any aspect of the physical environment hosting the application, and using policies to manage right-sizing (scale-in, -out, and load-balancing) and right-placing (locating data and/or processing in the ideal location at any point in time) the application at runtime.

The EzBrokerage application is written using Monterey Middleware Studio, with every transaction consisting of one upstream message (the request) and at least one downstream message to the sender (the response) message, with more approximately 1% of the time (a broadcast on top-of-book change and a counterparty message when a trade occurs). A transaction round-trip (request and response) on the Monterey Middleware Runtime network passes messages up to layer 7 of the ISO networking stack a total of four times, involving Java processing at one of each of the four node types in the Monterey network. This includes EzBrokerage application-specific processing (the matching algorithm) at one node type called the "mediator" node.

In the analysis below, latency is computed as the round-trip time for a message to pass from the first node, through the three others, and back to the first. Four values are reported: mean, 90th percentile (maximum of the fastest 90% transaction round-trips), "expected shortfall" at 10% (mean of the slowest 10% transaction round-trips), and 99th percentile (maximum of the fastest 99% transaction round-trips). Throughput is computed as the number of messages which can be sent while maintaining a notional SLA where both ES 10% and 99th %ile round-trip times are less than 10ms.

Application messages range in size from 200B to 80kB, with a mean message size of 800B.

Highlights

Running EzBrokerage with the Monterey Middleware Platform, we are able to demonstrate:

- scalable throughput: 64-node network handles 80k messages per second (8 times as much capacity as the "cubic" 8-node network)
- mean latency as low as 1.1 milli-seconds
- extremely consistent low latency: 99% of messages < 10ms RTT (SLA)



Throughput and Scalability

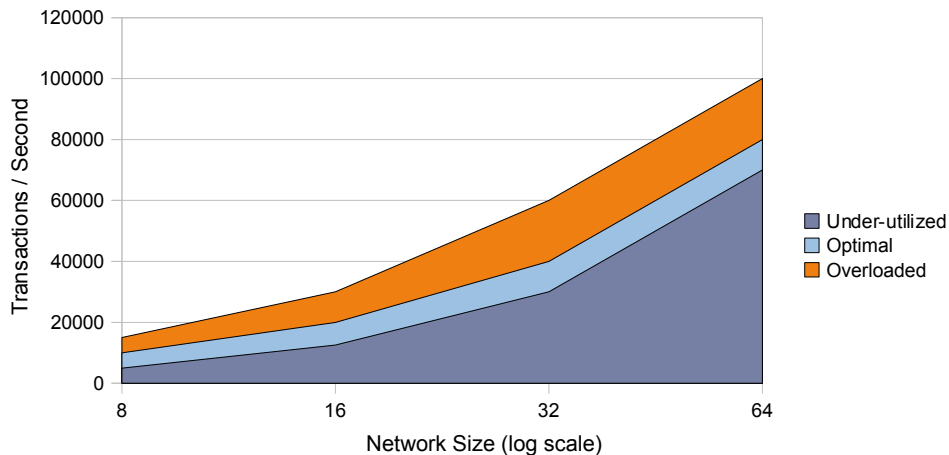
For EzBrokerage on Monterey, the optimal transaction rate has been computed as 5000 transactions per second per mediator node in the Monterey network. At this point capacity is CPU bound by the matching algorithm supplied by EzBrokerage.

The table below summarizes the results of testing various size networks at this rate and a slightly higher rate where eventually processing is expected to become back-logged. On previously used infrastructure, scalability has been limited by network capacity with significant performance degradation well before the 32-node configuration.

Test	Results
8-node network (2-2-2-2) at 10k txns/sec	pass
8-node network (2-2-2-2) at 15k txns/sec	overloaded
16-node network (4-4-4-4) at 20k txns/sec	pass
16-node network (4-4-4-4) at 30k txns/sec	overloaded
32-node network (8-8-8-8) at 40k txns/sec	pass
32-node network (8-8-8-8) at 60k txns/sec	overloaded
64-node network (16-16-16-16) at 80k txns/sec	pass
64-node network (16-16-16-16) at 100k txns/sec	overloaded

The tests demonstrate linear scalability up to and including the case with 16 nodes in each node class (denoted 16-16-16-16), with a total of 80,000 transactions per second. Scaling beyond this point was limited only by the number of CPUs in this Monterey infrastructure.

One of the key features of Monterey is support for application mobility, allowing any network path or processing location to be changed while the application is live. Among other benefits of mobility is the ability to scale-back and scale-out as application load changes in real-time. The graph below indicates the levels where policies can kick in to grow the network (in orange, based on the numbers above) or to shrink the network (in darker blue).





Round-Trip Time and Latency

Test	Mean	90 th %ile	ES 10%	99 th %ile
8-node network (2-2-2-2) at 5k txns/sec	1.46ms	2.21ms	2.57ms	3.04ms
8-node network (2-2-2-2) at 10k txns/sec	2.32ms	4.45ms	5.71ms	7.17ms
16-node network (4-4-4-4) at 10k txns/sec	1.16ms	2.01ms	2.47ms	3.12ms
16-node network (4-4-4-4) at 20k txns/sec	1.92ms	3.44ms	4.53ms	5.39ms
32-node network (8-8-8-8) at 20k txns/sec	1.15ms	1.74ms	4.13ms	3.08ms
32-node network (8-8-8-8) at 40k txns/sec	1.59ms	2.91ms	4.31ms	5.23ms
64-node network (16-16-16-16) at 40k t/s	1.11ms	1.94ms	2.56ms	3.25ms
64-node network (16-16-16-16) at 80k t/s	2.01ms	3.77ms	5.76ms	6.86ms

The latency tests measure the round-trip time for transactions in the Monterey network, including network transmission and processing, with the reference EzBrokerage application. Round-trip time remains roughly constant as network size increases while the transaction rate per node remains unchanged. Increasing transaction rate per node causes an increase in latency, but at a predictable, minor delta up to the optimal level for EzBrokerage, 5000 transactions per second (txns/sec or t/s) per mediator node.

In the largest network configuration, total traffic volume is in excess of 2Gbps traffic, with a varied profile of message size and location. The exceptionally low and consistent latency under high load is the result of extensive integration efforts by Solarflare with other partners. For more information on this initiatives, including discussion of raw latency (without the multiple layer 7 hops, Java, and EzBrokerage execution), please see:

http://www.solarflare.com/technology/tech_papers.php

The latency figures for EzBrokerage reported above have not made use of any of the configuration options available for the components in Monterey. Specifically the Blade switch, Solarflare server adapter and XenServer host all expose tunable parameters which we expect could decrease the latency further. (We were initially surprised by the slower performance at small network sizes, but this is explained by the default configuration of some of the components having been tuned with each other for larger size networks; this is one area which could be optimized.) The remarkable fact is that the Monterey stack delivers the vital requirement of consistently low latency without any customization even under heavy load and large subnet sizes.



About the Author

Alex Heneveld, Cloudsoft's Co-Founder and CTO, is one of the co-inventors of the distributed mediation technology and led the development of Monterey Middleware since 2005 . Previously, he founded PocketWatch Systems, commercializing results from his doctoral research, and worked for many years as a consultant specializing in information access and retrieval. Alex holds a PhD (Informatics) and an MSc (Cognitive Science) from the University of Edinburgh and an AB (Mathematics) from Princeton University.