



MEMORY IO BENCHMARK TEST RESULTS

FOR JOYENT
Revision 6

October 13th, 2010

Scope:

This report summarizes Memory IO benchmark testing performed in September of 2010.

References:

[1]: <http://blog.cloudharmony.com/2010/06/cloud-server-benchmarking-part-4-memory.html>

[2]: Svn repository: <https://svn.codespaces.com/ims/joyent-benchmarks>
username: joyent **password:** joyent

[3]: Raw test data: MEMORY-IO_Final_Results.xlsx

[4]: Phoronix Test Suite 2.6.1:
<http://www.phoronix-test-suite.com/download.php?file=phoronix-test-suite-2.6.1>

[5]: <http://www.alasir.com/software/ramspeed/ramspeed-2.6.0.tar.gz>

Joyent Memory IO Benchmark Testing Report

Introduction

The Memory IO testing was performed as part of a larger benchmark effort intended to provide a basis for comparison between Joyent SmartMachines and other virtual servers offered by cloud service providers.

Earlier in 2010, CloudHarmony engaged in an extensive benchmarking effort intended to provide "information and analysis to enable educated decisions pertaining the adoption of, and migration to cloud services". Their results and analysis are presented in a series of articles published online ref[1]. The CloudHarmony blog does not contain results for Joyent SmartMachines. Our testing procedures are intended to follow CloudHarmony's efforts as closely as possible and extend benchmarking for Joyent SmartMachines.

Instead of trying to reproduce all of the CloudHarmony results, we focused on those outlined for the Amazon EC2 servers used in their benchmark tests ref[1]. Our tests closely approximate the methods from CloudHarmony in regards to calculations and tests used. Figures for the Joyent SmartMachines should be a useful addition to the other benchmarks included in CloudHarmony's blog. It should be noted that not all test executables and versions contained in this report are identical to those of CloudHarmony due to differences in operating systems. These results should not be compared side-by-side to those of CloudHarmony. Our mathematical baselines and server instances are however identical.

CloudHarmony standardized on CentOS 64bit as the operating system used for test servers except where it was unavailable. Joyent SmartMachines run an OpenSolaris based operating system. Due to the original tests being Linux based, modifications were necessary to execute similar binaries on the Solaris based system.

SmartMachines provide a "bursting" capability enabled by their OS that allows a service to use more processor and memory resources on a temporary basis than

the guaranteed minimum. This differs from nearly all other cloud providers that provide a fixed processor/memory configuration. While bursting capability can be a tremendous advantage to an operational system, it can complicate benchmark testing by stressing the system to its maximum capacity. On the Joyent SmartMachines the bursting capability allows a process on even the smallest server to potentially use nearly the entire compute and memory capability of the underlying hardware.

Joyent uses large commodity servers with 8 hyper-threaded processors that effectively yield 16 processor cores. This means that Joyent's smallest server, the 1 GB SmartMachine, can in some cases outperform Amazon's largest EC2 instance, m2.4xLarge. Due to this bursting capability, side-by-side comparisons may not be identical in nature between Joyent and other cloud providers.

Benchmark Setup

The baseline used by CloudHarmony for their Memory IO performance benchmark is the NewServers Jumbo server configured with dual Intel E5504 quad core 2.00 GHz processors and 48GB DDR3 ECC ram. NewServers was no longer providing images for CentOS 5.4 at the time of our testing. Since CentOS 5.5 was released in May, 2010, our Jumbo server was setup with CentOS 5.5 instead of CentOS 5.4 used by CloudHarmony.

Amazon EC2 was used as our primary result comparison for Memory IO against those on CloudHarmony. The Amazon servers used consist of: m1.small, c1.medium, m1.large, m1.xlarge, m2.xlarge, c1.xlarge, m2.2xlarge, m2.4xlarge. All Amazon servers – 8 servers in 4 regions, were configured identically in terms of OS, CentOS 5.4 64-bit (or 32-bit in the case of EC2 m1.small and c1.medium where 64-bit is not supported).

All SmartMachines come with Joyent's SmartOS based on OpenSolaris - SunOS 5.11 snv_121. On the Joyent servers, the default gcc compiler is version 3.4.3 and the Amazon instances come equipped with gcc version 4.1.2. To provide comparison with an out-of-box install, the default software versions were used.

To run the majority of benchmark tests, CloudHarmony made use of the Phoronix Test Suite ref[1]. Version 2.6.1 was used for Joyent's OpenSolaris compatibility. There are several differences between version 2.2.0 used by CloudHarmony and 2.6.1 used in this report. These include test versions, source code, and executables.

The Phoronix Test Suite open source framework was originally a Linux specific tool and streamlines the testing procedures. Many of the included tests run natively under Linux, but not on Solaris as noted in the Phoronix documentation. Since the test suite makes use of shell scripts to download, unpack, build, install, and run the various tests, some modifications were necessary to run identical tests on the Joyent SmartMachines. Scripts within the Phoronix Test Suite make use of the Bourne Shell by specifying the `#!/bin/sh` command interpreter. Normally including `#!/bin/sh` works fine on most Linux systems, but under Solaris this is not the case.

Making the minor change from /bin/sh to /bin/ksh on Solaris can be made easily with a global search and replace to make all scripts compatible. Some other modifications required manual inspection and correction – versions and packages of required software are different between Solaris and Linux. There were also path changes needed in the test suite code to link to the appropriate executables in the Joyent SmartMachine OS. All changes to the test suite have been configuration controlled and is available to Joyent via subversion ref[2].

Benchmark Tests

Since the Joyent SmartMachine OS is OpenSolaris based, some tests that rely on Linux specific drivers or executables do not run correctly on a Joyent SmartMachine. The following test falls into this category:

hdparm cached read

The remaining 6 tests were run and compared on the NewServers Jumbo server, Amazon EC2 and Joyent SmartMachine. Default versions in Phoronix 2.6.1 were used for cachebench and stream.

Unixbench 5.1.2, redis-benchmark 2.0.1, ramspeed 2.6.0, cachebench, geekbench and stream

The default versions contained in Phoronix-Test-Suite 2.6.1 were used.

As noted, the benchmark tests on Joyent required script customizations or source code modifications: ref[2] phoronix-test-suite-2.6.1-SolarisPatched.tgz. The benchmark executables test similar functionality on Joyent to the tests CloudHarmony ran on other servers. Calculations from the test results to derive our baselines are of an identical nature between the different Operating Systems.

Testing Procedures

The Phoronix Test Suite 2.6.1 was setup on each server to run cachebench and stream. Unixbench, redis-benchmark and ramspeed required manual installation on the test machines. Phoronix compiles their results in xml files to be displayed in a web browser. The suite also creates image graphs for visual comparison. When not using Phoronix for testing, the output was saved to a flat-file for record keeping.

In order to reproduce our testing procedures on the Joyent SmartMachines see ref[2]: joyent_tests_directory.tgz, phoronix-user-directory.tgz, phoronix-suite-2.6.1-solaris.patch. The following guidelines should produce similar or identical test results:

1. Install the Phoronix Test Suite into a local directory within the user's folder on each server. Tar files for Joyent are included ref[2]: joyent_tests_directory.tgz, phoronix-user-directory.tgz. These two files contain all tests and executables that run on Joyent. If using these tar files, extract them into the user directory and skip to step 5.

2. If installing the default Phoronix Test Suite 2.6.1 ref[4], apply the patch file ref[2] phoronix-suite-2.6.1-solaris.patch to the test suite. This patch makes changes to the installation files and performs all alterations necessary for the Solaris platform. The test suite should be installed to a local user directory labeled 'Tests'. All programs should be installed to this folder.
3. Ramspeed does not have native support for OpenSolaris and requires a manual build and run from command line. Perform the following modifications to install ramspeed:
 - a. Download and extract ref[5] into the Tests/ramspeed-2.6.0 directory. Then open build.sh for editing.
 - b. For lines 111-112, 120-121 modify these commands:
 - a. `CC="cc" >> CC="gcc"`
 - b. `LD="cc" >> LD="gcc"`
 - c. Compile ramspeed with the following commands:
 - a. `cat build.sh | grep -v "read ANS" > build_pts.sh`
 - b. `chmod +x build_pts.sh`
 - c. `./build_pts.sh`
 - d. The run options for ramspeed in the setup_joyent.sh script are:
 - a. `./ramspeed -b 3 >> ramspeed_results.log 2>&1`
 - b. `./ramspeed -b 6 >> ramspeed_results.log 2>&1`
4. Geekbench needs to be manually extracted from the tarball and put into its own directory within the installed-tests/geekbench folder:

```
>> ./phoronix-test-suite/installed-tests/geekbench/geekbench-solaris
>> geekbench-solaris/geekbench_x86_32 (executable)
```

Modify The Phoronix executable – geekbench – in the geekbench folder:

```
>> ./phoronix-test-suite/installed-tests/geekbench/geekbench:
#!/bin/ksh
cd geekbench-solaris/
./geekbench_x86_32 > $LOG_FILE 2>&1
echo $? > ~/test-exit-status
```

If missing, place a similar pts-install.xml file in the geekbench folder.

5. Download and extract the Unixbench source ref[5] into the folder Tests/unixbench-5.1.2. Follow the directions contained in the Unixbench source code to compile and run.
6. Use the test executable script ref[2] setup_joyent.sh to run all tests with specific parameters and execute each Memory IO benchmark. It automatically packs the output files from each test result. Modify the last line to match an available ftp server. Memory IO tests on each server take approximately 4 hours to complete.

Note: If tests fail to run, make the following modifications to the test suite core files to see the full executable outputs for troubleshooting:

```
phoronix-test-suite/pts-core/library/pts-functions_shell.php
```

At line 110 add:

```
echo pts_variables_export_string($extra_vars) . "\n\n";  
echo $exec . "\n\n";
```

This will output the Phoronix variables and executable to the command line.

Baselines

The NewServers Jumbo server was assigned a score of 100. All other servers were assigned a score proportional to the performance of that server, where greater than 100 represents better results and less than 100 represents poorer results. For example, a server with a score of 50 scored 50% lower than the baseline server overall, while a server with a score of 125, scored 25% higher. It should be noted that hdparm-cached-read benchmarks was not used to calculate the MIOP score in this test because it was not run on the Joyent SmartMachine. The MIOP were comparable to the same results posted on the CloudHarmony blog.

Test Results

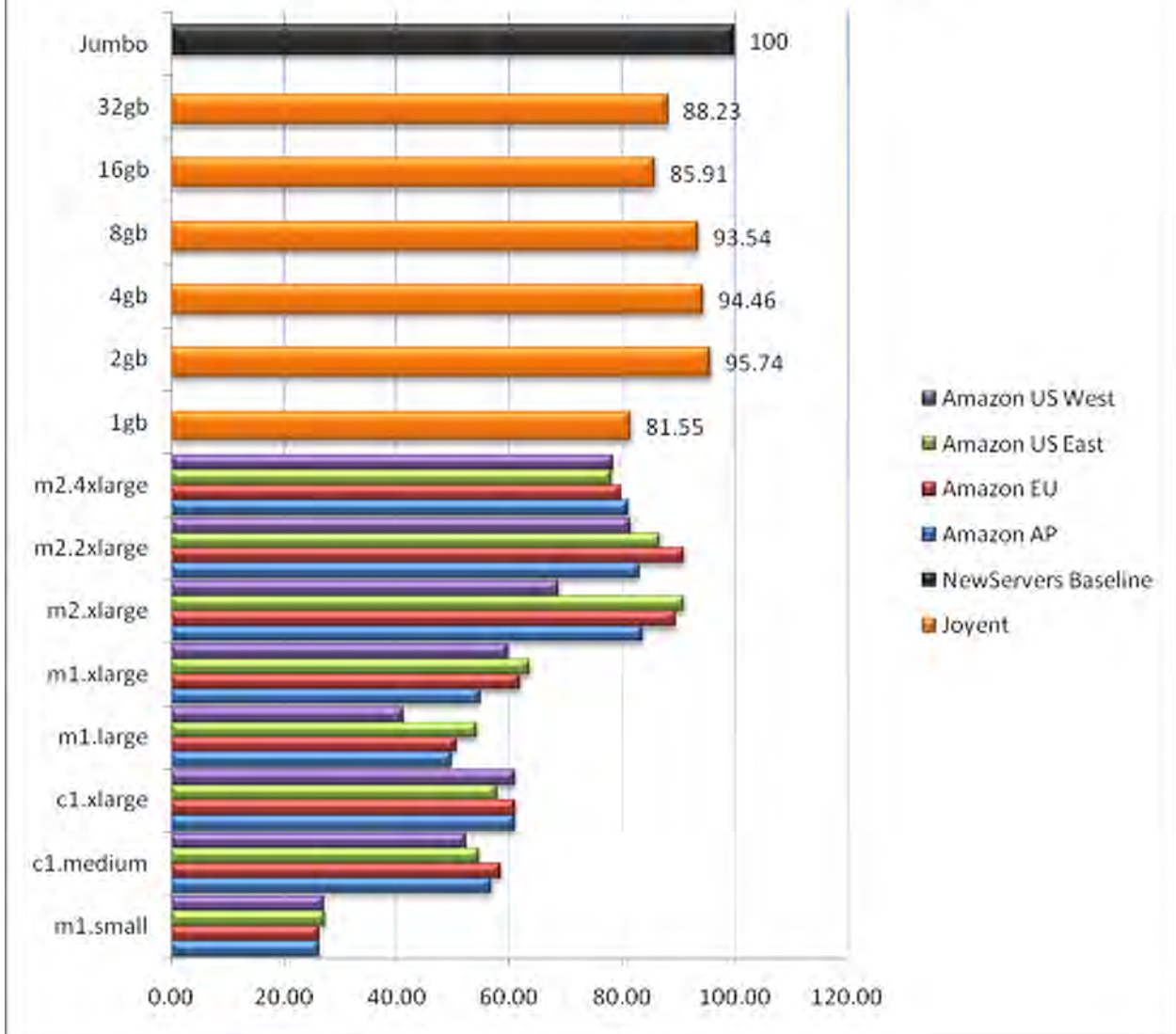
The test scores of MIOP were compared to those of the NewServers baseline and a ratio was created based on a percentage of each result to that of the baseline.

Overall tests show the NewServers baseline to be the highest. Individual tests show similar results except for geekbench and cachebench which range much higher than that of the Jumbo server used for the other cloud providers. See ref[3] for the full raw data reports in the following graph.

Since the majority of our tests are compiled using Gcc, we needed to compare the impact of each operating system's default version. The Gcc version packaged with the Joyent SmartOS is 3.4.3, while the default Amazon CentOS version is 4.1.2. We compiled and ran our Tscp benchmark spot tests with different versions of Gcc and recorded the results. The comparison test scores between Gcc 3.4.3 and 4.2.3 show an approximate 14% increase in performance on both Joyent and Amazon EC2. Gcc 4.1 and 4.2 showed no difference in the results. Should the default Gcc version be upgraded to 4.x, the Joyent SmartMachine scores should be significantly higher.

The two following graphs and data illustrate the above calculations and results:

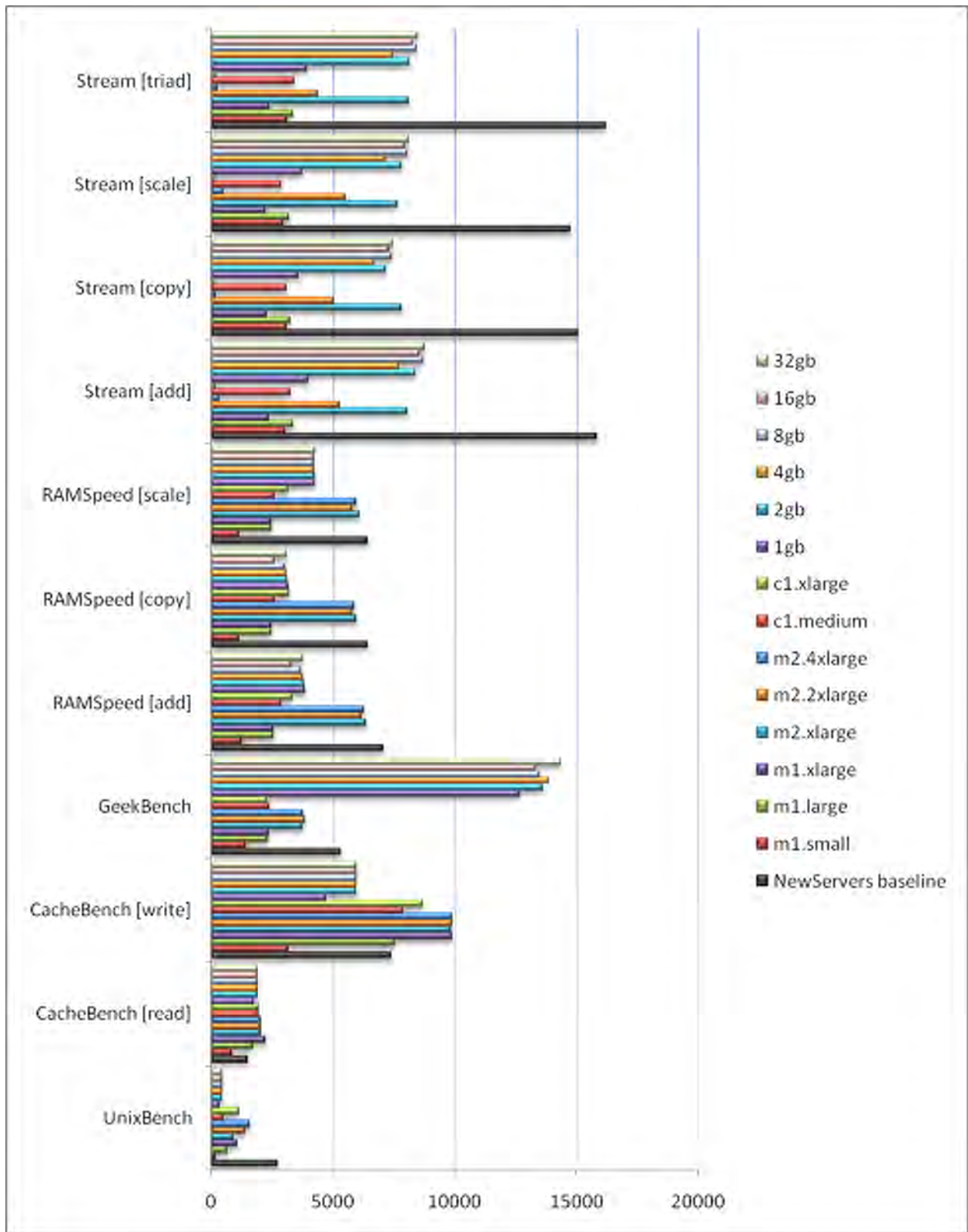
Memory IOP Scores [higher is better]



	m1.small	c1.medium	m1.large	m1.xlarge
Amazon AP	26.19	56.72	49.77	54.94
Amazon EU	26.28	58.44	50.57	61.99
Amazon US East	27.38	54.66	54.31	63.65
Amazon US West	27.11	52.21	41.31	59.76

	m2.xlarge	c1.xlarge	m2.2xlarge	m2.4xlarge
Amazon AP	83.81	61.03	83.19	81.23
Amazon EU	89.48	61.04	91.09	79.71
Amazon US East	91.08	57.85	86.78	78.22
Amazon US West	68.79	61.19	81.63	78.35

	1GB	2GB	4GB	8GB	16GB	32GB
Joyent	81.55	95.74	94.46	93.54	85.91	88.23



See ref[3] for individual statistics of each benchmark test.

Conclusion

The variation in MIOP values between the 1GB and 32GB Joyent SmartMachines did not show a strong linear increase in Memory IO performance with the increase in server capacity. This was expected due to Joyent's Memory bursting capability. Overall, all Joyent servers outperformed the Amazon server instances in our comparisons. The only servers which had higher scores for Amazon are the US East and EU m2.xlarge and m2.2xlarge instances; these are comparable to the Joyent 16GB SmartMachine in terms of underlying hardware and available resources.

While comparing memory throughput to the NewServers baseline, we can see that most of the test results are lower than this established baseline score. This could possibly be due to the CentOS 5.5 operating system upgrade and 48GB of available memory. The only test which did not comply with this ideology is cachebench and geekbench. Joyent performed much higher in these two test categories. Additionally, should the default version of GCC be upgraded to 4.x, the composite Joyent scores could be significantly higher.

Amazon servers, as expected, showed a consistent increase in the MIOP values in the test results from small to large based on server capacity. Individual memory tests however showed variance between the servers which did not follow a linear pattern. These results are comparable to those found on the CloudHarmony blog.