Ceph Distributed File System Benchmarks on an Openstack Cloud

X. Zhang(zxuuzx@gmail.com), S. Gaddam(gcg047@my.utsa.edu), A. T. Chronopoulos(antony.tc@gmail.com)
Department of Computer Science
University of Texas at San Antonio
1 UTSA Circle, San Antonio, Texas 78249, USA

ABSTRACT—Ceph is a distributed file system that provides high performance, reliability, and scalability. Ceph maximizes the separation between data and metadata management by replacing allocation tables with a pseudo-random data distribution function (CRUSH) designed for heterogeneous and dynamic clusters of unreliable object storage devices (OSDs). In this paper, we investigate the performance of Ceph on an Open Stack cloud using well-known benchmarks. Our results show its good performance and scalability.

Keywords- OpenStack Cloud, Ceph distributed file storage, benchmarks

I. INTRODUCTION

Ceph, is a scalable, open source, software-defined storage system that runs on commodity hardware [1-5]. Ceph has been developed from the ground up to deliver object, block, and file system storage in a single software platform that is self-managing, self-healing and has no single point of failure. Because of its highly scalable, software defined storage architecture, Ceph is an ideal replacement for legacy storage systems and a powerful storage solution for object and block storage for cloud computing environments [3].

In this paper, we present a Ceph architecture and map it to an OpenStack cloud. We study the functionality of Ceph using benchmarks such as Bonnie ++, DD, Rados Bench, OSD Tell, IPerf and Netcat with respect to the speed of data being copied and also the read/write performance of Ceph using different benchmarks. Our results show the good performance and scalability of Ceph in terms of increasing clients request and data sizes.

In related work, some results exist on Ceph performance evaluation on clusters [14], [15]. These papers present benchmarks with Ceph installed on standard cluster systems.

The rest of the paper is organized as follows. In section II, the Ceph architecture is presented. Section III contains the benchmarks that we performed and our results. In section IV, we present the installation details for Ceph. In section V, we present conclusions and future work.

II. CEPH ARCHITECTURE

We next outline the Ceph storage cluster. The Ceph storage cluster is made up of several different software daemons. Each of these daemons takes care of unique Ceph functionalities and adds values to its corresponding components [5].

Reliable Autonomic Distributed Object Store (**RADOS**) is the foundation of the Ceph storage cluster. Everything in Ceph is stored in the form of objects, and the RADOS object store is responsible for storing these objects, irrespective of their data type.

Ceph Daemons: Data gets stored in Ceph **Object Storage Device** (**OSD**) in the form of objects. This is the only component of a Ceph cluster where actual user data is stored and the same data is retrieved when a client issues a read operation.

Ceph monitors (**MONs**) track the health of the entire cluster by keeping a map of the cluster state, which includes OSD, MON, PG, and CRUSH maps. All the cluster nodes report to monitor nodes and share information about every change in their state. A monitor maintains a separate map of information for each component.

Librados library is a convenient way to get access to RADOS with the support of the PHP, Ruby, Java, Python, C, and C++ programming languages. It provides a native interface to the Ceph storage cluster, RADOS, and a base for other services such as RBD, RGW, as well as the POSIX interface for Ceph file system.

Ceph Block Device, formerly known as **RADOS block device** (**RBD**), provides block storage, which can be mapped, formatted, and mounted just like any other disk to the server. A Ceph block device is equipped with enterprise storage features such as thin provisioning and snapshots.

978-1-4673-8566-4/16 $31.00 © 2016 IEEE
DOI 10.1109/CCEM.2015.12

Ceph Metadata Server (**MDS**) keeps track of file hierarchy and stores metadata only for CephFS.

Ceph File System (**CephFS**) offers a POSIX-compliant, distributed file system of any size. CephFS relies on Ceph MDS to keep track of file hierarchy.

The architecture layout which for our Ceph installation has the following characteristics and is shown in Figure 1.

- Operating system: Ubuntu Server
- Version: LTS 14.04
- Ceph version: 0.87 Giant
- OSDs number: 3
- MONs number: 3
- Clients number : 4

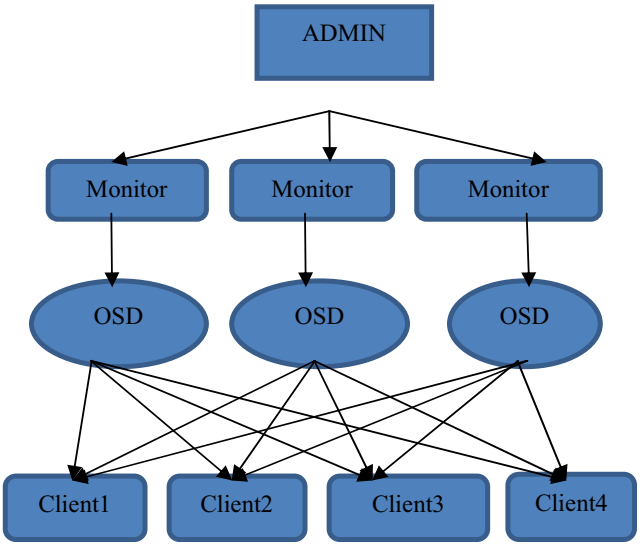

Figure 1 : The ceph architecture

We mapped this Ceph architecture to an OpenStack cloud system operated by the Open Cloud Institute (OCI) at the University of Texas at San Antonio. It offers significant capacity and similar design features found in Cloud Computing providers, including robust compute capability and elastic infrastructure design. We used 8 virtual machines. Each virtual machine has two virtual cpus (VCPUs or cores) , and four GB memory. The underlying physical server in the cloud is a SeaMicro SM15000 system with 64 octal core AMD Opteron processors with 10Gbps network connectivity. Then Threads are created each on separate VCPUs. Thus, for example, 8 threads run on 8 VCPUs and require at least 4 VMs (each one has 2 VCPUs). We performed runs for 1,4,8 clients by using 1,4,8 threads.

III. BENCHMARKING EXPERIMENTS FOR CEPH

Several benchmarks have been proposed for Ceph (e.g. [6, 14-15] and refs therein). We ran the well-known benchmarks (Bonnie ++, DD, Rados Bench, OSD Tell, IPERF and Netcat) that measure the speed of data that is copied and also the read and write performance of Ceph.

A. Bonnie++

Bonnie++ is an open-source file system benchmarking tool for Unix OS developed by Russell Coker [6-10]. Bonnie++ is a benchmark suite that is aimed at performing a number of simple tests of hard drive and file system performance. It allows you to benchmark how your file systems perform with respect to data read and write speed, the number of seeks that can be performed per second, and the number of file metadata operations that can be performed per second.

It is a small utility with the purpose of benchmarking file system I/O performance. It is used to minimize the effect of file caching and tests should be performed on larger data sets than the amount of RAM we have on the test system. Bonnie++ adds the facility to test more than 2G of storage on a 32bit machine, and tests for file creat(), stat(), unlink() operations and it will output in CSV spread-sheet format to standard output.

We show the Time (Latency) and bandwidth results for Bonnie++ in Table 1 and Figures 2-3.

Table 1 : (Bonnie++) Time and bandwidth

	Single thread	4 threads	8 threads
Write Size (MB)	16	16	16
Time (us)	159	180.5	168
Bandwidth (K/s)	18541	10111	2670

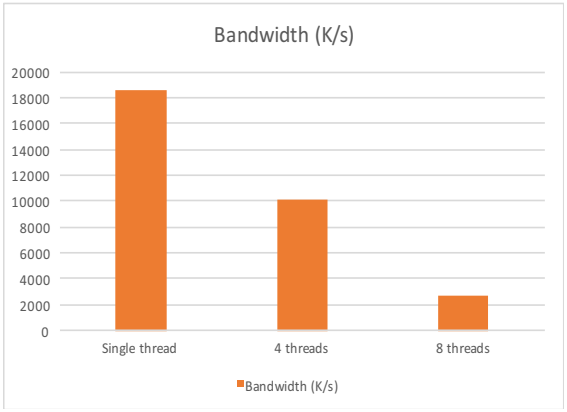

Figure 2 : (Bonnie++) Bandwidth

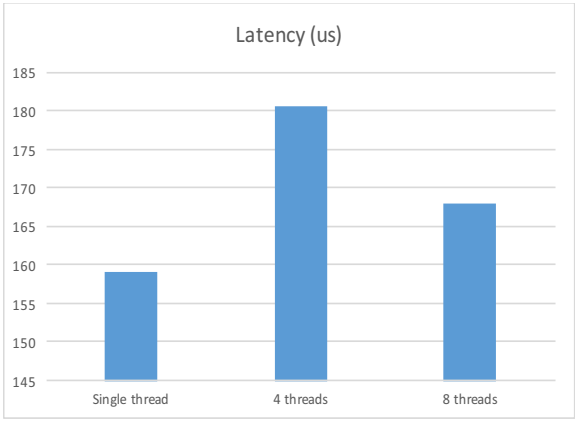

Figure 3 : (Bonnie++) Time

B. DD (Read and Write)

DD is a benchmark to perform a real-world disk test on a Linux system. "DD" stands for data description and it is used for copying data sources. It is the basic test which is not customizable and shows the performance of the file system. This is mostly used in WRITE (copying) into a new file while this is used for READ speed of a disk by reading the file that is created [6-7]. DD helps in testing sequential read and sequential write. We show DD speed in copying data from 1-6 clients in Table 2 and Figure 6. We show DD Read and Write time and bandwidth in Tables 3-4 and Figures 5-8.

DD	128MB	256MB	512MB
client1	38.5	19.6	14.9
client2	29	24.5	18.4
client3	30.8	18.3	18
client4	41.4	21.2	18.3
client5	33.2	17.4	18.6
client6	30.4	21.9	19.2

Table 2 : (DD) Speed in copying data

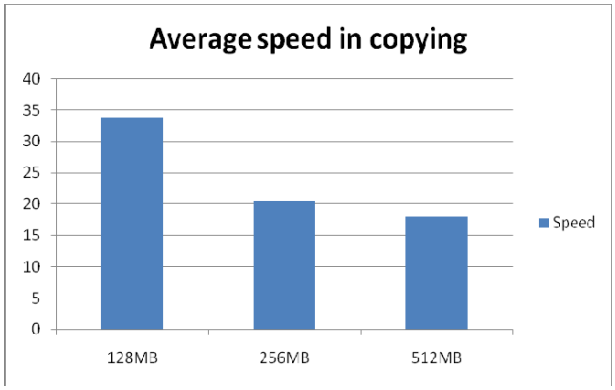

Figure 4 : (DD) Average speed in coping data

DD Read Benchmark:

This benchmark shows the DD performance (time/bandwidth) using read benchmark for 1 ,4 and 8 threads.

	Single thread	4 threads	8 threads
Size(Bytes)	5242880	5242880	5242880
Time (sec)	0.0815217	0.107416	0.113498225
Bandwidth (MB/s)	64.3	50.15	50.725

Table 3 : (DD read) Time/bandwidth

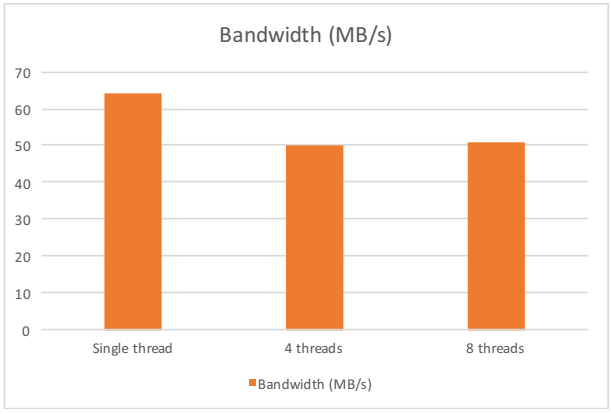

Figure 5 : (DD Read) Bandwidth

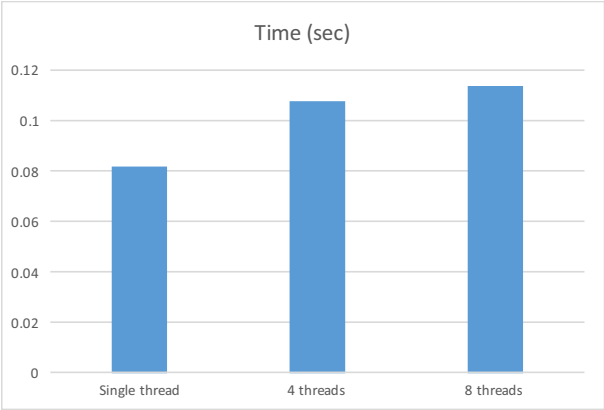

Figure 6 : (DD Read) Time.

DD Write Benchmark:

The write Benchmark shows the DD performance using 1, 4 threads and 8 threads.

	Single thread	4 threads	8 threads
Size(Bytes)	5242880	5242880	5242880
Time (sec)	0.42957	1.533755	1.91441
Bandwidth (MB/s)	12.2	3.425	2.75

Table 4 : (DD Write) Time/bandwidth

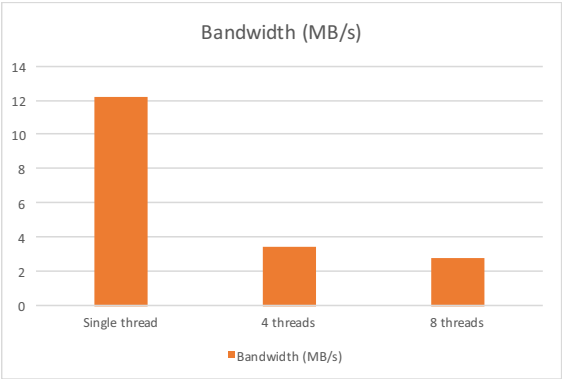

Figure 7 : (DD Write) Bandwidth

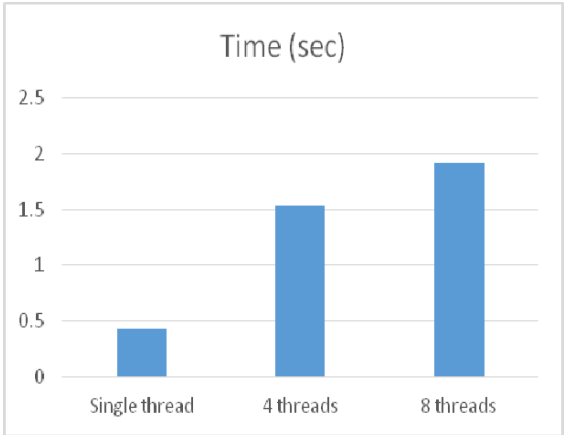

Figure 8 : (DD Write) Bandwidth

C. RADOS Bench (Read/Write)

RADOS is an inbuilt benchmark known as native benchmark, where RADOS is a utility for interacting with Ceph object storage cluster which provides the read and write sequential and random results [6,15]. We show Read and Write time and bandwidth in Tables 5-6 and Figures 9-11, 13-15. We also show the total number of Reads (Writes) per thread in Figure 12 (Figure 16).

RADOS Bench (Read):

We show the performance of RADOS Bench Read for 1, 4 and 8 threads. It shows the bandwidth, total time run, read size, Average and Maximum Latency with respect to threads.

	Single thread	4 threads	8 threads
Time (sec)	30.627255	31.8791815	33.5086398
Total reads	926	290.25	150.875
Read size	4194304	4194304	4194304
Bandwidth (MB/sec)	120.938	36.41875	18.010875
Average Latency (sec)	0.527604	1.75118	3.52563
Max latency (sec)	1.20062	4.81599	11.1616375

Table 5: (RADOS Bench Read) Time/bandwidth

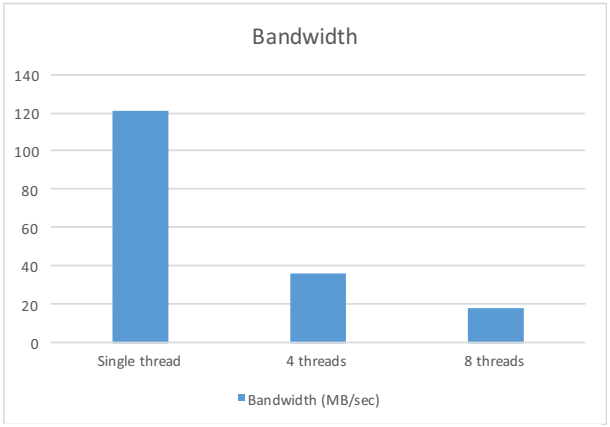

Figure **9** : (RADOS Bench Read) Bandwidth

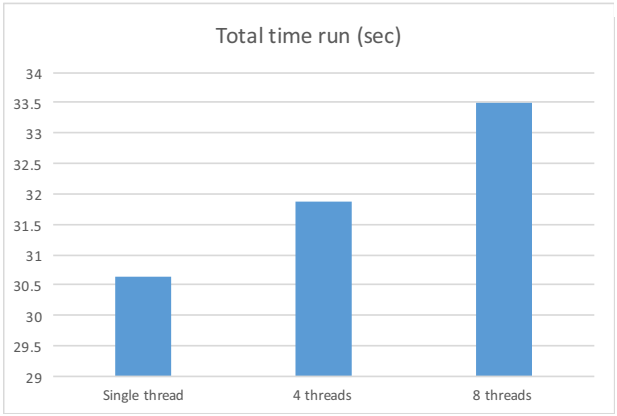

Figure 10 : (RADOS Bench Read) Total time

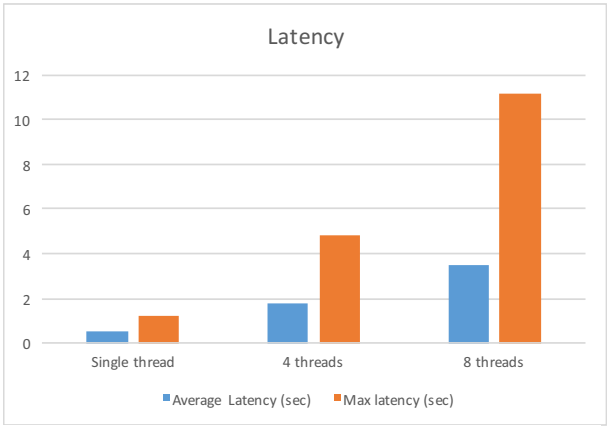

Figure 11 : (RADOS Bench Read) Mean and Max. time

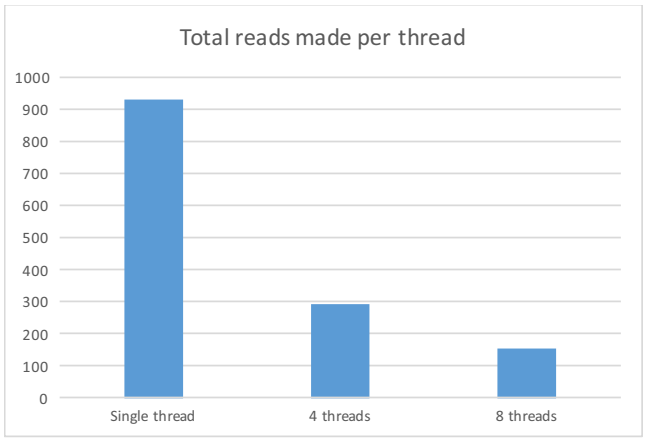

Figure 12 : (RADOS Bench Read) Total reads per thread

RADOS Bench Write:

We show the performance of RADOS Write benchmark for 1, 4 and 8 threads. This shows the bandwidth, total time run, write size, Average and Maximum Latency with respect to threads.

	Single thread	4 threads	8 threads
Time (sec)	34.504556	38.702577	42.630856
Total writes made	140	84	48
Write size	4194304	4194304	4194304
Bandwidth (MB/sec)	16.23	8.682	4.504
Max bandwidth (MB/sec)	32	32	16
Average Latency (sec)	3.89433	7.22118	13.9908
Max latency (sec)	8.3188	19.1852	38.1739

Table 6 : (RADOS Bench Write) Time/bandwidth

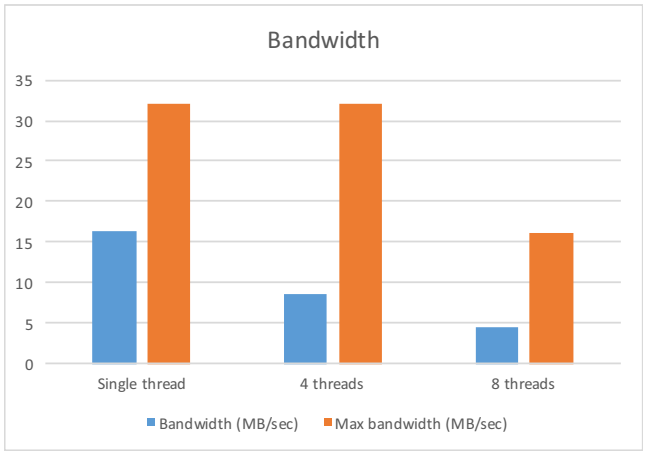

Figure 13 : (RADOS Bench Write) Bandwidth

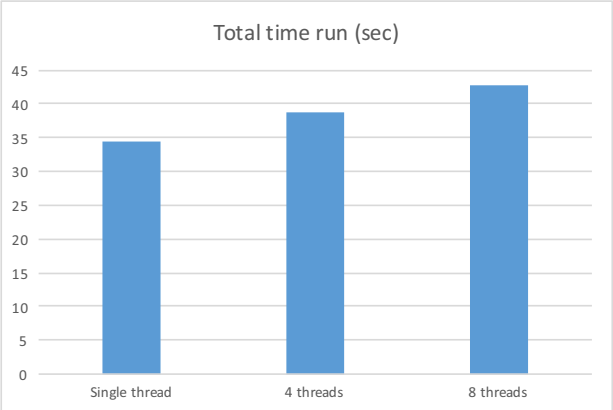

Figure 14 : (RADOS Bench Write) Time

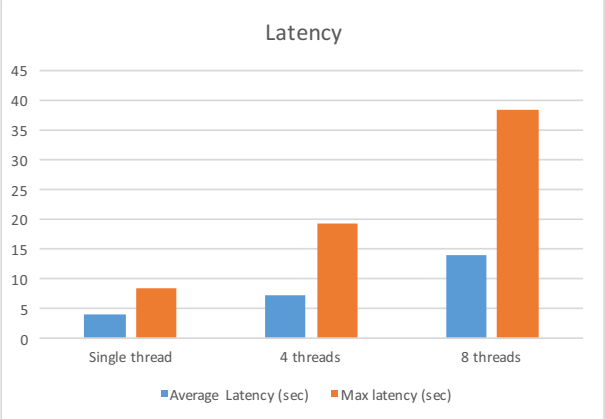

Figure 15: (RADOS Bench Write) Average and maximum time

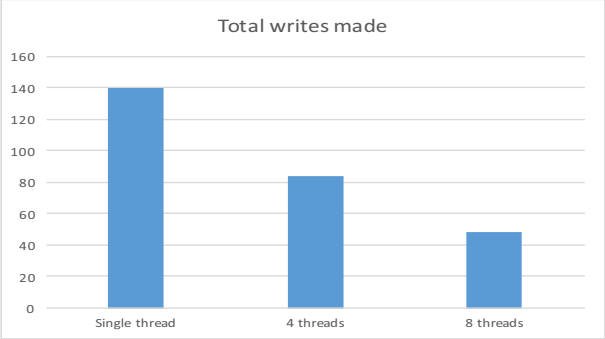

Figure 16 : (RADOS Write) Total writes

D. *OSD Tell benchmark:*

OSD Tell is the native Ceph benchmark, which checks the functioning of the OSDs with respect to the server [4].
We show the bandwidth between the OSD and the server in Table 7 and Figure 17.

	Single thread	4 threads	8 threads
Write Size (MB)	1000	1000	1000
blocksize(MB)	1	1	1
Bandwidth (Byte/s)	828	832.75	828.25

Table 7: (OSD Tell) Bandwidth between OSD and Server

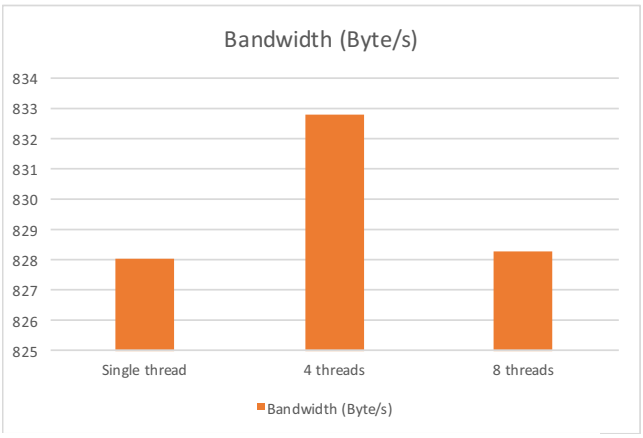

Figure 17: (OSD Tell) Bandwidth between OSD and Server

E. *Iperf Benchmark:*

Iperf is a commonly used network testing tool that can create data streams for Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) and measure the Connection Speed (i.e. Throughput) of the network [12]. We show the bandwidth between the OSD and client in Table 8 and Figure 18.

	Client1	Client2	Client3	Client4
OSD1 (Mb/s)	940	935	932	939
OSD2 (Mb/s)	935	940	940	936
OSD3 (Mb/s)	937	939	938	940

Table 8: Iperf Bandwidth

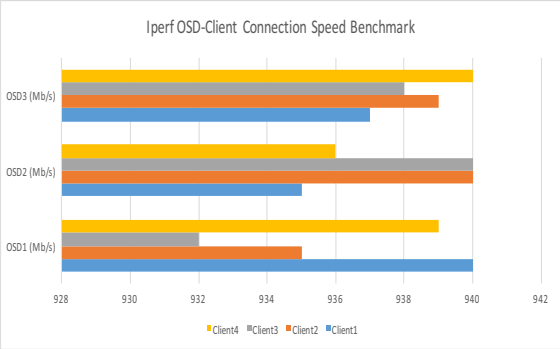

Figure 18: (Iperf) Bandwidth

F. Netcat Benchmark:

Netcat (also known as NC) is a computer networking service for reading from and writing to network connections using TCP or UDP. This is also a connectivity benchmark [11]. We show the bandwidth between the OSD and client in Table 9 and Figure 19.

	Client1	Client2	Client3	Client4
OSD1 (KB/s)	87	62.2	60.3	54.3
OSD2 (KB/s)	38.8	17.4	41.7	68
OSD3 (KB/s)	43.4	15.3	38.3	25.4

Table 9: Netcat Bandwidth

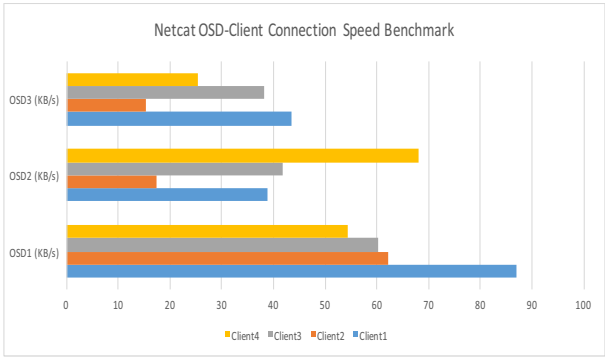

Figure 19: (Netcat) Bandwidth between OSD and Client

IV. CEPH INSTALLATION ON OPENSTACK CLOUD

Ceph installation is clearly described in [13]. Here we summarize the installation steps.

Step 1. Preparation.
Use the latest version of Ceph and also check for the appropriate Release key and install Ceph on the Admin Node
Authenticate SSH key password less authentication and check for the access for all the nodes in the ceph cluster from the admin node.

Step 2. Storage Cluster.
Creating and initialize the cluster. First install ceph from admin node to each and every other nodes in the cluster.
Initialize and create monitor nodes. Then push admin node configuration file to other nodes.
Finally prepare and active the designated osd nodes.
Check for the Ceph Health which states Ceph is ready to use .

Step 3. Ceph Clients.
Install ceph from admin node to designated client nodes.
Using RBD command to create an image of block device and map it appropriate block device.
Create and initialize file system for the block device, after mounting it to Client's directory.
Ceph is ready to use with Clients.

V. CONCLUSIONS AND FUTURE WORK

Ceph, a distributed file system that provides excellent performance, reliability, and scalability. In this paper, we have outlined the functionality of Ceph and we installed it on an OpenStack cloud. We ran well-known benchmarks (Bonnie ++, DD, Rados Bench, OSD Tell, Iperf and Netcat) that measure the speed of data that is copied and also the read and write performance of Ceph. The results show that the performance decreases very slowly as a function of the number of clients requests. This demonstrates the good

performance and (small-scale) scalability of Ceph in the OpenStack cloud.

In future work, we plan to run large-scale experiments to show the scalability of Ceph in a large-scale OpenStack cloud. We also plan to test the reliability of Ceph in the presence of faults.

ACKNOWLEDGMENT

We gratefully acknowledge the following:
(i) Support by NSF grant CNS-1419165 to the University of Texas at San Antonio; and (ii) time grants to access the Facilities of the Open Cloud Institute of University of Texas at San Antonio.

VI. REFERENCES

[1] Weil, Sage A., Scott A. Brandt, Ethan L. Miller, Darrell DE Long, and Carlos Maltzahn. "Ceph: A scalable, high-performance distributed file system." In *Proceedings of the 7th symposium on Operating systems design and implementation*, pp. 307-320. USENIX Association, 2006.

[2] Weil, Sage A., Scott A. Brandt, Ethan L. Miller, and Carlos Maltzahn. "CRUSH: Controlled, scalable, decentralized placement of replicated data." In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, p. 122. ACM, 2006.

[3] Maltzahn, Carlos, Esteban Molina-Estolano, Amandeep Khurana, Alex J. Nelson, Scott A. Brandt, and Sage Weil. "Ceph as a scalable alternative to the Hadoop Distributed File System." *login: The USENIX Magazine* 35 (2010): 38-49.

[4] Ceph Storage, Red Hat, 2015 [Online] Available: http://ceph.com/

[5] Ceph Documentation [Online] Available: http://ceph.com/docs/master/

[6] Ceph Benchmarks, Sebastien Han, [Online] Available: http://www.sebastien-han.fr/blog/2012/08/26/ceph-benchmarks/

[7] Benchmark Disk IO with DD and Bonnie++, James Cole, [Online] Available: http://www.jamescoyle.net/how-to/599-benchmark-disk-io-with-dd-and-bonnie

[8] Bonnie++, Wikipedia [Online] Available: http://en.wikipedia.org/wiki/Bonnie++

[9] Bonnie++, Russel Coker, Webpage [Online] Available: http://www.coker.com.au/bonnie++/

[10] Active Benchmarking [Online] Available: http://www.brendangregg.com/ActiveBenchmarking/bonnie++.html

[11] Netcat, Wikipedia [Online] Available: http://en.wikipedia.org/wiki/Netcat

[12] Iperf, Wikipedia [Online] Available: https://en.wikipedia.org/wiki/Iperf

[13] Singh, Karan. *Learning Ceph*. Packt Publishing Ltd, 2015.

[14] Wang, Feiyi, Mark Nelson, Sarp Oral, Scott Atchley, Sage Weil, Bradley W. Settlemyer, Blake Caldwell, and Jason Hill. "Performance and scalability evaluation of the Ceph parallel file system." In *Proceedings of the 8th Parallel Data Storage Workshop*, pp. 14-19. ACM, 2013.

[15] Gudu, Diana, Marcus Hardt, and Achim Streit. "Evaluating the performance and scalability of the Ceph distributed storage system." In *Big Data (Big Data), 2014 IEEE International Conference on*, pp. 177-182. IEEE, 2014.