MINIO

MinIO S3 Throughput Benchmark on Hard Disk Drives

JANUARY 2020

MinIO S3 Throughput Benchmark on Hard Disk Drives

MinIO is a high-performance object storage server designed for AI and ML workloads.

Machine learning, big-data analytics and other AI workloads have traditionally utilized the map-reduce model of computing where data is local to the compute jobs. Modern computing environments have adopted a cloud-native architecture where storage and compute are disaggregated. This enables computing to become stateless, elastic, and scalable independent of storage. Object storage has become the de-facto standard for this architecture.

Applications access data over the network using atomic, immutable object APIs where the data is often in a Binary Large Object (BLOB) format. The performance metrics for object storage are measured in terms of I/O throughput, rather than IOPS.

This document describes benchmarks that MinIO engineering ran to determine the performance of the MinIO Object Storage Server when run on hard disk drives. Specifically, this document shows how to setup the benchmarking environment, how to run the benchmarking tools and reviews the performance results in detail.

Our results running on 24 node Minio cluster can be summarized as follows:

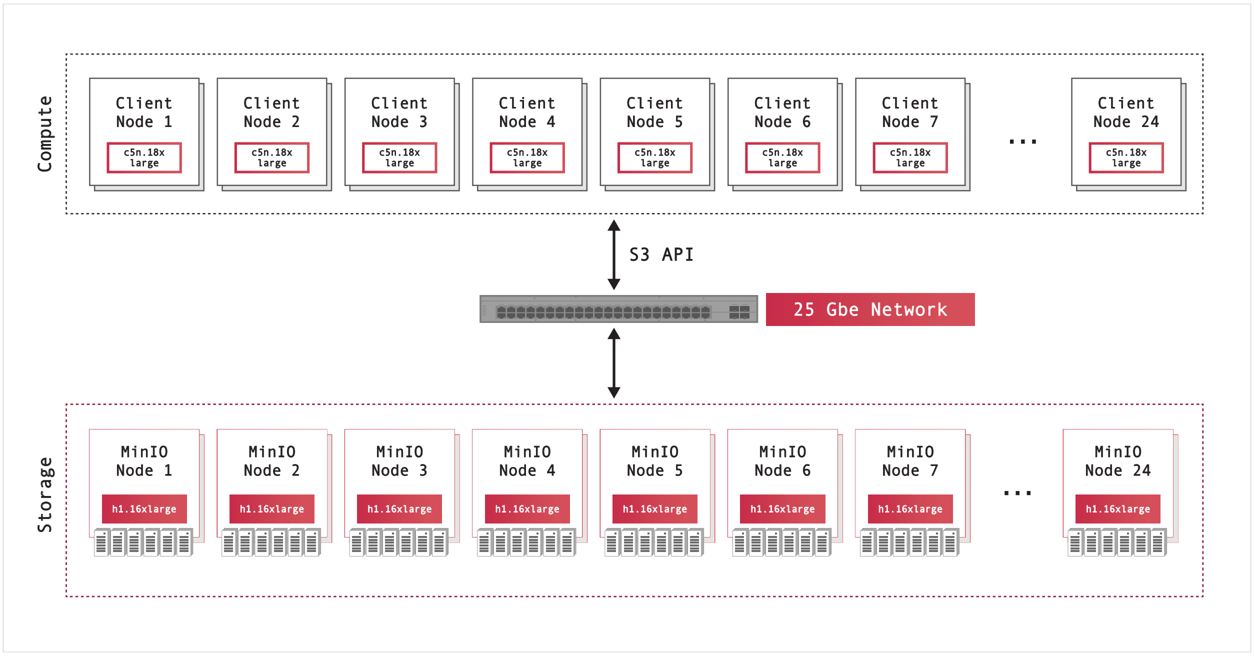
Setup	Avg Read Throughput (GET)	Avg Write Throughput (PUT)
Distributed	16.3 GB/s	9.4 GB/s

1. Benchmark Environment

1.1 Hardware

For the purpose of this benchmark, MinIO utilized AWS bare-metal, storage optimized instances with local hard disk drives and 25 GbE networking.

Instance	# Nodes	AWS Instance type	CPU	MEM	Storage	Network
Client	24	c5d.18xlarge	72	144 GB	2 x 900 GB	25 Gbps
Server	24	h1.16xlarge	64	256 GB	8 x 2000 GB	25 Gbps



1.2 Software

Property	Value
Server OS	Ubuntu 18.04.2 LTS (Bionic Beaver)
MinIO Version	Minio-RELEASE.2019-10-12T01-39-57Z
Benchmark Tool	S3-benchmark

1.3 S3-benchmark

S3-benchmark by wasabi-tech was chosen to perform benchmarking tests. This tool conducts benchmark tests from a single client to a single endpoint. During our evaluation, this simple tool produced consistent and reproducible results over multiple runs.

Minor changes were required in s3-benchmark, such as disabling the client-side md5 generation that hindered the tool from saturating the 25 Gbe network.

1.4 Linux Kernel Performance Tuning

Edit the `/etc/sysctl.conf` file to match the following kernel settings:

```
# maximum number of open files/file descriptors
fs.file-max = 4194303

# use as little swap space as possible
vm.swappiness  =  1

# prioritize application RAM against disk/swap cache
vm.vfs_cache_pressure  =  10

# minimum free memory
vm.min_free_kbytes = 1000000

# maximum receive socket buffer (bytes)
net.core.rmem_max = 268435456

# maximum send buffer socket buffer (bytes)
net.core.wmem_max = 268435456

# default receive buffer socket size (bytes)
net.core.rmem_default = 67108864

# default send buffer socket size (bytes)
net.core.wmem_default = 67108864

# maximum number of packets in one poll cycle
net.core.netdev_budget = 1200

# maximum ancillary buffer size per socket
net.core.optmem_max = 134217728

# maximum number of incoming connections
net.core.somaxconn = 65535

# maximum number of packets queued
net.core.netdev_max_backlog = 250000

# maximum read buffer space
net.ipv4.tcp_rmem = 67108864 134217728 268435456

# maximum write buffer space
net.ipv4.tcp_wmem = 67108864 134217728 268435456

# enable low latency mode
net.ipv4.tcp_low_latency = 1
```

```
# socket buffer portion used for TCP window
net.ipv4.tcp_adv_win_scale = 1

# queue length of completely established sockets waiting for accept
net.ipv4.tcp_max_syn_backlog = 30000

# maximum number of sockets in TIME_WAIT state
net.ipv4.tcp_max_tw_buckets = 2000000

# reuse sockets in TIME_WAIT state when safe
net.ipv4.tcp_tw_reuse = 1

# time to wait (seconds) for FIN packet
net.ipv4.tcp_fin_timeout = 5

# disable icmp send redirects
net.ipv4.conf.all.send_redirects = 0

# disable icmp accept redirect
net.ipv4.conf.all.accept_redirects = 0

# drop packets with LSR or SSR
net.ipv4.conf.all.accept_source_route = 0

# MTU discovery, only enable when ICMP blackhole detected
net.ipv4.tcp_mtu_probing = 1
```

Apply these parameters by calling the command sysctl -p
The MinIO binary was downloaded onto each server node, and started using the following commands:

```
$ export MINIO_STORAGE_CLASS_STANDARD=EC:2
$ export MINIO_ACCESS_KEY=minio
$ export MINIO_SECRET_KEY=minio123
$ minio server http://minio-{1...24}/mnt/drive{1...8}
```

1.5 Client Setup

Each client was provided with a hostname matching the pattern client-{1...24}.

The DNS was configured on the client nodes such that the hostname minio on each client resolved to a unique MinIO server. For instance, the URL http://minio:9000 on the client-1 resolved to the ip address of minio-1, and client-2 resolved to minio-2's ip address etc.

2. Understanding Hardware Performance

2.1 Measuring Single Drive Performance

The performance of each drive was measured using the command dd. DD is a unix tool used to perform bit-by-bit copy of data from one file to another. It provides options to control the block size of each read and write.

Here is the output of a single HDD drive's Write Performance with 16MB block-size using the O_DIRECT option and a total count of 64. Note that we achieved greater than 130 MB/sec of write performance for each drive.

```
$ dd if=/dev/zero of=/mnt/drive/test bs=16M count=64 oflag=direct
64+0 records in
64+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 7.83377 s, 137 MB/s
```

Here is the output of a single HDD drive's Read Performance with 16MB block-size using the O_DIRECT option and a total count of 64. Note that we achieved greater than 200 MB/sec of read performance for each drive.

```
$ dd of=/dev/null if=/mnt/drive/test bs=16M count=64 iflag=direct
64+0 records in
64+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 5.23199 s, 205 MB/s
```

2.2 Measuring JBOD Performance

JBOD performance with O_DIRECT was measured using iozone. Iozone is a filesystem benchmark tool that generates and measures filesystem performance for read, and write among other operations.

Here is the iozone command to test multi-drive performance with 32 parallel threads, 32KB block-size and the O_DIRECT option.

```
$ iozone -t 32 -I -r 32K -s 256M -F /mnt/drive{1..8}/tmp{1..4}
```

Note that we achieved 1.26 GB/sec of read throughput and 655 MB/sec of write throughput for the combination of all the drives in one server.

2.3 Measuring Network Performance

All of the servers used 25GbE networking for both client and internode communications. This network hardware provides a maximum of 25 Gbit/sec, which equates to 3.125 GB/sec (1 Gbyte = 8 Gbit).

Even though the network can support 3.125GB/sec, the JBOD testing showed that the maximum throughput supported by the JBOD on each server is 1.26 GB/sec. Therefore, the maximum server throughput that can be expected from each node would be 1.26 Gbyte/sec.

3. Running the 16-node distributed MinIO benchmark

We measured the performance of the MinIO Object Server by running the s3-benchmark tool in parallel on all 16 clients and aggregating the results. Here is the command that was used:

```
$ parallel-ssh -O "StrictHostKeyChecking=no" --timeout=0
--hosts=hosts.clients -i "./s3-benchmark -a minio -s minio123
-u http://minio:9000 -z 2G -t 32 -b s3bench-`hostname` -d 1"
```

The parameters passed into the S3-benchmark tool are defined as follows:

-a	access key
-s	secret key
-u	endpoint
-z	object size
-t	number of parallel workers
-b	bucket name
-d	duration

3.1 Results

The following table documents the throughput of the MinIO Object Storage servers as measured from the 16 individual clients that ran the s3-benchmark tool.

Node #	GET (MB/sec)	PUT (MB/sec)
1	890.8	304.9
2	581.1	666.6
3	624.2	264.5
4	575.7	482.3
5	716.3	509.6
6	604.1	276.7
7	739.1	326.6
8	696.8	309.2
9	633.7	817.5
10	1008.9	278.2
11	789.2	263.7
12	632.6	287.5
13	557	295.6
14	634.9	308.3
15	552.3	271.5
16	586.1	267.4
17	824.6	435
18	687.5	279.6
19	801.8	269.7
20	780.2	257.7
21	688.6	877.8
22	646.6	263.9
23	546	505.2
24	580.1	579.8
Total	16378.2	9398.8

3.2 Interpretation of Results

The benchmark results demonstrate that MinIO is able to achieve 10.81 GB/sec read performance and 8.57 GB/sec write performance from a 16 node MinIO Object Server.

While reviewing the test runs we determined that the performance of the MinIO Object Server was limited by the performance of the throughput of the hard disk drives. However, the solution was not limited by the network performance of the 25GbE network as the network was not fully saturated.

The bandwidth of the drives was entirely utilized during these tests. Higher throughput can be expected if additional drives were available.

Note that the write benchmark is slower than read because benchmark tools do not account for write amplification (traffic from parity data generated during writes). In this case, the hard disk is the bottleneck as MinIO gets close to hardware performance for both reads and writes.

3.4 CPU and Memory Utilization

Setup	Avg CPU% (millicpu)	Avg Mem (GB)
Distributed	1.4% (670)	11.23

MinIO is optimized using SIMD instructions where special CPU instructions widely available on almost all production grade CPUs were used to accelerate computational steps.

Note that the available CPU for each of the HDD machines was 64000 millicpus. The meaning of millicpu can be found here.

3.5 Network Utilization

Setup	Avg Read Throughput (GET)	Avg Write Throughput (PUT)
Distributed	15.09 Gbps	14.23 Gbps

Note that the available network bandwidth for each of the servers was 25 Gbps. The bandwidth was measured using vnstat on each of the server nodes.

4. Conclusion

Based on the results above, we found that MinIO is only constrained by the underlying hardware available to it. We tested this with the most powerful hardware available on AWS.