Vol.2021-IOT-55 No.4
Vol.2021-SPT-44 No.4
2021/9/6

Qualitative and quantitative performance evaluations of relatively inexpensive storage products (3)

Hiroki Kashiwazaki[1,,a)]

Abstract: In 2017 and 2020, respectively, the author reported on his research on qualitative and quantitative performance evaluation of relatively low-cost Network Attached Storage. This report is a continuation of the 2020 report, in which the NIC of Synology's FS6400 all-flash storage was replaced with a 40Gigabit Ethernet NIC, and a 100Gigabit Ethernet Switch was used to connect the 100Gigabit Ethernet NIC to a Linux server. From a Linux server with a 100Gigabit Ethernet NIC connected by a 100Gigabit Ethernet Switch, I prepared multiple values with multiple parameters using the I/O test tool fio, and conducted exhaustive measurements to benchmark all combinations of these values. The measurement results were formatted and visualized in a heat map. This time, I applied this method to disk arrays with a unit price of 100 USD/TB terabytes, all-flash storage with a price of 250 USD/TB, and all-flash storage with a price of 5000 USD/TB.

1. Introduction

A distributed system consists of computers, networks, and power supplies. The computer has a storage device, which is the main memory, and an auxiliary storage device, which is storage. Applications that do not use auxiliary storage, such as on-memory databases, are emerging, but there are still a certain number of applications that read and write through file descriptors.

Some auxiliary storage devices are housed in the computer chassis, while others are installed externally as independent devices and are connected to the computer via a network. FibreChannel and iSCSI are used as interfaces for the OS to access external storage over the network as block devices. There is also the method of mounting as a file system using NFS. The former is sometimes called Storage Area Network (SAN), while the latter is sometimes called Network Attached Storage (NAS).

These network-attached storage products are available from a variety of vendors. Each vendor discloses its product specifications, but only what can be definitively mentioned about the quantitative performance of the product. Therefore, the user cannot know what performance characteristics the product has. The conditions under which the performance is measured may not match the conditions under which the user actually uses the product. There is a need for fair, reproducible, and comprehensive benchmarking.

In 2017, I benchmarked storage products equipped with 10Gigabit Ethernet [1] Network Interface Cards (NICs). In the 2020 report, I benchmarked storage products with 40Gigabit Ethernet [2]. In both cases, I chose products that cost around USD 100 per terabyte. On the other hand, in the 2020 research report, I also benchmarked all-flash storage products that cost around 500 USD per terabyte.

In the 2020 report, I was not able to benchmark storage performance by examining the characteristics of MTU on performance, especially in a 100Gigabit Ethernet environment. In this paper, I propose a method for comprehensive benchmarking using fio, a storage benchmarking software, and propose a method for processing the results.

2. Benchmark

2.1 fio

In the 2020 research report, I described various benchmarking software, but I will not do so this time. In this article, I will perform an comprehensive benchmark using fio, one of the benchmark software mentioned in the 2020 report. fio is a flexibile I/O tester developed by Jens Axboe[*1]. Fio was originally written to save him the hassle of writing special test case programs when he wanted to test a specific workload, either for performance reasons or to find/reproduce a bug. The process of writing such a test app can be tiresome, especially if users have to do it often. Hence he needed a tool that would be able to simulate a given I/O workload without resorting to writing a tailored test case again and again. The newest version of fio is 3.27 (on 9th Aug. 2021).

[1] National Institute of Informatics, Chiyoda, Tokyo 101–8430, Japan
[a)] reo_kashiwazaki@nii.ac.jp

[*1] `https://git.kernel.dk/?p=fio.git;a=summary`

It has support for 19 different types of I/O engines (sync, mmap, libaio, posixaio, SG v3, splice, null, network, syslet, guasi, solarisaio, and more), I/O priorities (for newer Linux kernels), rate I/O, forked or threaded jobs, and much more. It can work on block devices as well as files. fio accepts job descriptions in a simple-to-understand text format. Several example job files are included. fio displays all sorts of I/O performance information, including complete IO latencies and percentiles. Fio is in wide use in many places, for both benchmarking, QA, and verification purposes. It supports Linux, FreeBSD, NetBSD, OpenBSD, OS X, OpenSolaris, AIX, HP-UX, Android, and Windows.

In academic research, fio has been used in various studies: Xianzhang Chen et al. introduced a process-variant wear-leveling mechanism to persistent memory file systems, and fio was used to evaluate the proposed method [3]. Gyusun Lee et al. have proposed a hardware-based demand paging method, and fio is used for this evaluation [4]. Ravi Kiran Boggavarapu et al. proposed a block deduplication-aware page cache management method and perform FIO benchmarking on a dataset containing 25% duplicate data [5]. Rodrigo Leite et al. proposed a hyper-converged system to host Docker containers and investigated the performance of container persistent data storage using various workloads in Microsoft data centers and multiple disk configurations of the hyper-converged system [6]. Ziye Yang et al. propose SPDK-vhost-NVMe, an I/O service target relying on user space NVMe drivers, which can collaborate with hypervisors to accelerate NVMe I/Os inside VMs, and use fio to benchmark against QEMU native NVMe emulation solutions [7].

As related work shows, fio allows benchmarking with specific workloads. fio provides a variety of options, with multiple numbers that can be specified with these options, and exhaustive benchmarking with multiple options specifying all combinations. fio provides various options. The main options of fio are described below.

The `direct` option takes a bool value. If the value is true, use non-buffered I/O. This is usually `O_DIRECT`. Note that OpenBSD and ZFS on Solaris don't support direct I/O. On Windows the synchronous ioengines don't support direct I/O. Default: false.

The `rw` option specifies the Type of I/O pattern. Accepted values are:

read Sequential reads.

write Sequential writes.

trim Sequential trims (Linux block devices and SCSI character devices only).

randread Random reads.

randwrite Random writes.

randtrim Random trims (Linux block devices and SCSI character devices only).

rw,readwrite Sequential mixed reads and writes.

randrw Random mixed reads and writes.

trimwrite Sequential trim+write sequences. Blocks will be trimmed first, then the same blocks will be written to.

Fio defaults to read if the option is not specified. For the mixed I/O types, the default is to split them 50/50. For certain types of I/O the result may still be skewed a bit, since the speed may be different.

The `bs` option can specify the block size in bytes used for I/O units. Default size is 4096. A single value applies to reads, writes, and trims. Comma-separated values may be specified for reads, writes, and trims. A value not terminated in a comma applies to

The `size` option can specify the total size of file I/O for each thread of this job. Fio will run until this many bytes has been transferred, unless runtime is limited by other options (such as runtime, for instance, or increased/decreased by `io_size`). Fio will divide this size between the available files determined by options such as nrfiles, filename, unless filesize is specified by the job. If the result of division happens to be 0, the size is set to the physical size of the given files or devices if they exist. If this option is not specified, fio will use the full size of the given files or devices. If the files do not exist, size must be given. It is also possible to give size as a percentage between 1 and 100. If 'size=20%' is given, fio will use 20% of the full size of the given files or devices. Can be combined with offset to constrain the start and end range that I/O will be done within.

If an integer value is specified in the `numjobs` option, Create the specified number of clones for this job. Each clone of the job is spawned as an independent thread or process. May be used to set up a larger number of threads/processes doing the same thing. Each thread is reported separately. The default value is 1.

If an integer value is specified in the `runtime` option, Tell fio to terminate processing after the specified period of time.

2.2 targets

In 2020, I accidentally met an opportunity to get two storage products. One is Synology FlashStation FS6400[*2] and the other is Synology SA3400[*3]. The unit price per TiB of these two products are shown in Table 1. The unit of price is USD[*4]. I adopted Cisco Nexus 9332C[*5] as SAN 100Gigabit Ethernet Switch. Including the switch, the unit price of FS6400 can be less than 250 USD and the one of SA3400

[*2] Synology FlashStation FS6400 `https://www.synology.com/en-global/products/FS6400`
[*3] Synology SA3400`https://www.synology.com/en-global/products/SA3400`
[*4] USD-JPY currencies rate is 106.9 JPY/USD (18 Juune 2020)
[*5] Cisco Nexus 9332C and 9364C Fixed Spine Switches Data Sheet `https://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/datasheet-c78-739886.html`

Table 1 A comparison of the specifications of each storage product.

	FS6400	SA3400
number of drives	72	36
drive	WDS400T1R0A	WUH721818ALE6L4
total price (USD)	70,000	40,000
total capacity (TiB)	288 TiB	648 TiB
unit price per TiB	243	62

Table 2 A comparison of specifications of each storage product.

	FS6400	SA3400
CPU	Intel Xeon Silver 4110	Intel Xeon D-1541
number of cores	8	8
CPU Frequency (GHz)	2.1 ˜3.0	2.1 ˜2.7
memory (GB)	512	128
number of NIC	40GbE x2	40GbE x2
size (mm)	264 x 482 x 724	264 x 482 x 724
internal file system	Btrfs/EXT4	Btrfs/EXT4

also can be less than 100 USD each. Each total capacity is a number with 2 external units. Each total price includes a cost of the units, 40Gigabit Ethernet and the main memory expansion.

Specification of two products are described in Table 2. Each product certifies 40Gigabit Ethernet NICs such as Mellanox ConnectX series[*6].

2.3 setting up

This paper shows evaluation results of network attached storage with 100GbE/40GbE environment. A diagram of an environment of evaluations is shown in Figure 1. FS6400 and SA3400 are connected to Cisco Nexus 9332C Fixed Spine Switches with 40GBase-SR4. And Nexus 9332C is also connected to Cisco UCS C220M5 with 100GBase-SR4. Ubuntu Linux 20.04 LTS (Focal Fossa)[*7] is running on the C220M5. Benchmark programs are executed on Ubuntu Linux. FS6400 and SA3400 provides block device with iSCSI and NFS service.

Synology products provide a web-based management tool called DiskStation Manager (DSM), which is used to create storage pools using Storage Manager in DSM's suite of management tools. A storage pool can consist of multiple drives, and a RAID configuration can be specified. RAID F1 applies RAID 5 mechanism and provides fault tolerance to increase read performance. However, when RAID F1 is used, more parity information is written to certain drives to accelerate aging. Thus, it prevents all drives from expiring at the same time. This may have a slight performance impact compared to RAID 5. At least three drives are required; RAID F1 will not lose data if one drive fails. In the event of a drive failure, the data on the failed drive is reconstructed from the parity stripped across the remaining drives. For this reason, both read and write performance can be very severely impacted if the RAID F1 array becomes degraded. RAID F1 can be

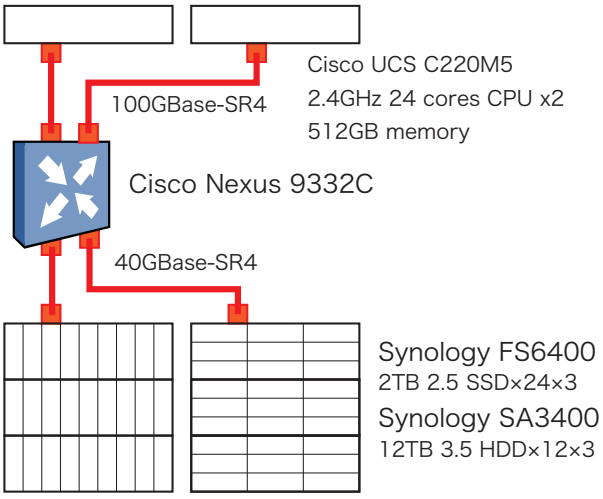

Fig. 1 A diagram of benchmark setup for IOzone

configured with three or more drives[*8].

Creating a storage pool creates a meta-device that can be recognized as a volume, which can be formatted in Linux, the operating system of Synology products. The volume can be formatted in Linux, the OS of Synology products. ext4 and Btrfs are available as file systems. The volume can be used as a block device by client machines using iSCSI, and can also be used for file operations using network file transfer protocols such as NFS and CIFS. In this example, I will use iSCSI as a block device. iSCSI Manager is provided in the DSM. Allocate the specified area from the volume created earlier and assign a Logical Unit Number (LUN). When the target is specified for the allocated LUN, an iSCSI Qualified Name (IQN) is associated with the LUN, and this LUN can be discovered from the client machine by using the iSCSI initiator. In addition to setting up authentication, multiple settings can be configured on the target. in Ubuntu Linux, the iscsiadm command can be used to discover the iSCSI target as a block device.

In this case, I wanted to measure the impact of the number of RAID F1 units on performance, so I created 9 storage pools, ranging from a minimum of 3 units to 11 units, and created iSCSI targets associated with each of them. The SSD in the Synology FS6400 is a Western Digital WDS400T1R0A[*9] with a capacity of 4TB, connected via SATA III. The volume was formatted with Btrfs.

3. Evaluations

In order to perform comprehensive benchmarking, I connected to iSCSI targets associated with the 9 storage pool patterns on FS6400, and benchmarked all iSCSI targets using fio. The `rw` option was specified for read and write re-

[*6] Mellanox ConnectX Ethernet Adapters
`https://www.mellanox.com/products/ethernet/connectx-smartnic`

[*7] Ubuntu 20.04 LTS (Focal Fossa)
`https://releases.ubuntu.com/20.04/`

[*8] Synology RAID F1 White Paper
`https://global.download.synology.com/download/Document/Software/WhitePaper/Firmware/DSM/All/enu/Synology_RAID_F1_WP.pdf`

[*9] WD Red™ SA500 NAS SATA SSD 2.5" /7mm cased — Western Digital Store
`https://shop.westerndigital.com/products/internal-drives/wd-red-sata-2-5-ssd`

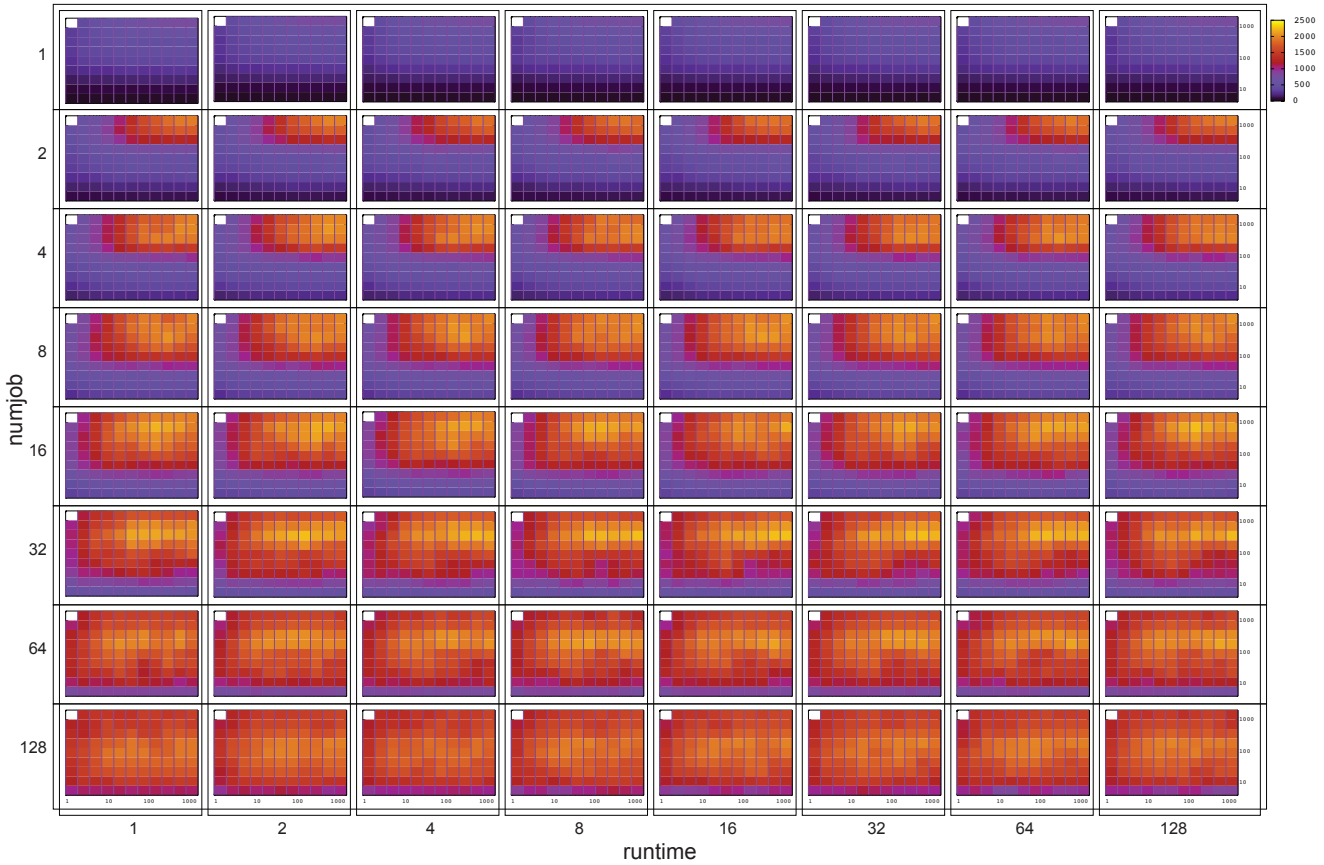

Fig. 2 Heatmaps of Read Throughput Performance of Synology FS6400 (3 SSDs).

spectively. The `bs` option was set to a minimum value of 4 B, and was measured in 10 patterns of doubled values up to 2048 B. The `size` option was set to a minimum value of 1 MB, and was measured in 12 patterns of doubled values up to 2048 MB. The `numjobs` option was set to a minimum value of 1, and was measured in 8 patterns up to 128. The `runtime` option was set to 1 as the minimum value and measured in 8 patterns up to 128. The MTU of the FS6400's 40Gigabit Ethernet NIC is set to 1500 B.

3.1 Results and Discussions

The throughput information was extracted from each benchmark result measured, and a heat map was created by mapping the values of throughput (MB/sec) from 0 to 2500 to the color of the gradient, with the x-axis being the file size and the y-axis being the block size. This heat map was created for each `numjob` value from 1 to 128 and each `runtime` value from 1 to 128, and arranged in an 8x8 tiled format as shown in Figure 2 (the 3 SSDs target). The heatmaps were created using gnuplot[*10], and pm3d was specified. A shell script was prepared to format and align the benchmark result data so that it can be drawn with gnuplot's `splot`. The benchmark was run with 9 storage pool patterns, 2 I/O patterns (`rw`), 10 block size (`bs`) patterns , 12 file `size` patterns, 8 `numjobs` patterns, and 8 `runtime` patterns, for a total of 138,240 benchmark runs, which took 198 hours.

As you can see from the definition of the option runtime, it does not make sense to benchmark runtime in 8 steps. As I understand it at the time of writing, the runtime option was misunderstood as the number of iterations when benchmarking. Therefore, the more iterations, the more the measurements are averaged and outliers are rounded, and I thought it was meaningful to find a runtime value where the mean and deviation converged within a certain value, but I was wrong in my understanding. Therefore, in practice, I can exclude runtime variations and expect to measure in less than 2.75 hours per target.

Since `numjobs` is the number of parallel jobs, the more the number of parallel jobs is, the more the throughput will increase, and finally it will be saturated at the upper limit of IOPS or throughput. In this measurement, the maximum throughput of 2500MB/sec is measured when the value of numjobs is greater than 16 and the block size is 1024KB. 2500MB/sec is equal to 20Gbytes/sec, so it is not a 10Gigabit Ethernet NIC but a 40Gigabit Ethernet NIC. By changing the MTU value, a higher upper limit can be expected. The block size (`bs`) value is changed step by step starting from 4B to 2048B, but the default value is 4096B. I would like to measure it by varying it from 1KB to 1MB. The throughput and IOPS should be doubled as the value of numjobs is doubled until the upper limit of IOPS is reached, so I can calculate the average throughput and IOPS per process by dividing the throughput and IOPS by the value of

[*10] gnuplot homepage `http://www.gnuplot.info`

numjobs. This is the average throughput and IOPS per process. By observing how this value changes as numjobs changes, I can measure the maximum number of connections.

4. Conclusion

In this paper, I used fio, an open-source I/O testing tool, to comprehensively measure and visualize a relatively inexpensive iSCSI storage product. From the generated heatmap, I show that it is possible to intuitively understand at which block size this storage product shows the highest performance. I also presented a visualization method to understand the differences in performance due to changes in parameters by arranging the benchmark results measured with different parameters.

This time, I presented a visualization method to observe the entire throughput comprehensively by arranging heatmaps in 8x8 panels, but this method is still not easy to see. There are many variations of fio options, and by adopting more appropriate options, I can expect to visualize more appropriate and fair benchmark results.

In this paper, I only show the measurement results of Synology FS6400 (with a single MTU value in a single storage pool configuration), but I will release the measurement results of Synology SA3400, a disk array that is already in preparation. In addition, I have installed PureStorage's FlashArray X20, so I plan to perform the same iSCSI configuration here and compare the benchmark results. The price per terabyte is 5000USD/TB, which is 10 times higher than Synology FS6400, so I expect to compare the cost effectiveness of the two.

References

[1] : IEEE Standard for Information technology - Local and metropolitan area networks - Part 3: CSMA/CD Access Method and Physical Layer Specifications - Media Access Control (MAC) Parameters, Physical Layer, and Management Parameters for 10 Gb/s Operation, *IEEE Std 802.3ae-2002 (Amendment to IEEE Std 802.3-2002)*, pp. 1–544 (2002).

[2] : IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 3: CSMA/CD Access Method and Physical Layer Specifications Amendment 4: Media Access Control Parameters, Physical Layers, and Management Parameters for 40 Gb/s and 100 Gb/s Operation, *IEEE Std 802.3ba-2010 (Amendment to IEEE Standard 802.3-2008)*, pp. 1–457 (2010).

[3] Chen, X., Sha, E. H.-M., Wang, X., Yang, C., Jiang, W. and Zhuge, Q.: Contour: A Process Variation Aware Wear-Leveling Mechanism for Inodes of Persistent Memory File Systems, *IEEE Transactions on Computers*, Vol. 70, No. 7, pp. 1034–1045 (online), DOI: 10.1109/TC.2020.3002537 (2021).

[4] Lee, G., Jin, W., Song, W., Gong, J., Bae, J., Ham, T. J., Lee, J. W. and Jeong, J.: A Case for Hardware-Based Demand Paging, *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pp. 1103–1116 (online), DOI: 10.1109/ISCA45697.2020.00093 (2020).

[5] Boggavarapu, R. K. and Jiang, S.: Deduplication-aware I/O Buffer Management in the Linux Kernel for Improved I/O Performance and Memory Utilization, *2020 12th International Conference on Knowledge and Smart Technology (KST)*, pp. 70–74 (online), DOI: 10.1109/KST48564.2020.9059514 (2020).

[6] Leite, R. and Solis, P.: Performance Analysis of Data Storage in a Hyperconverged Infrastructure Using Docker and GlusterFS, *2019 XLV Latin American Computing Conference (CLEI)*, pp. 1–10 (online), DOI: 10.1109/CLEI47609.2019.235108 (2019).

[7] Yang, Z., Liu, C., Zhou, Y., Liu, X. and Cao, G.: SPDK Vhost-NVMe: Accelerating I/Os in Virtual Machines on NVMe SSDs via User Space Vhost Target, *2018 IEEE 8th International Symposium on Cloud and Service Computing (SC2)*, pp. 67–76 (online), DOI: 10.1109/SC2.2018.00016 (2018).