NVMFS Benchmark

Introduction

Makes MySQL flash-aware and improves latency performance and consistency while increasing storage efficiency of Fusion ioMemory PCIe application accelerators
Replaces traditional double-write operation with single atomic write for lower, more consistent latency and reduced flash wear
New NVM Compression algorithm delivers benefits of compression without performance penalty

Tools：TPC-C, Warehouse numbers：1000,storage：FusionIO SX300 1.6TB，OS：CentOS 6.5 kernel：2.6.32-431.el6.x86_64
InnoDB buffer pool：64G
sync_binlog = 0
innodb_flush_log_at_trx_commit=2
10.0.15-MariaDB-log MariaDB Server(supports atomic writes)
File system: NVMFS ,ext4
warmup time： 120s
runtime： 3600s
Threads： 32 ~ 512
Machine: Dell R720 24core Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60GHz

How to install NVMFS

Step	Action
1	Login as root.
2	Disable `auto_attach` by issuing the following command: `# modprobe iomemory-vsl4 auto_attach=0`
3	Detach the device if it is already attached: `# fio-detach /dev/fctx` (where `fctx` is fct1 or fct2, etc.)
4	Format the device with Fusion ioMemory SDK options enabled use –A for atomic writes –P for Persistent TRIM –e for sparse address `fio-format -APye -b 512 /dev/fctx`
5	Attach the device for I/O operations. `# fio-attach /dev/fctx` This should result in device (fiox) appearing under /dev.

Step	Action
1	Load the kernel module nvmfs.ko using the modprobe command as follows: • `modprobe nvmfs`
2	Verify that the kernel module has loaded properly using the lsmod command as follows: • `lsmod | grep nvmfs`

#mount
/dev/sda2 on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw)
/dev/sda1 on /boot type ext4 (rw)
/dev/sda4 on /home type ext4 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
/dev/fioa on /storage/sas type NVMFS (rw,noatime)

How to configure MariaDB

Partial write operations

When Innodb writes to the filesystem, there is generally no guarantee that a given write operation will be complete (not partial) in cases of a poweroff event, or if the operating system crashes at the exact moment a write is being done.

Without detection or prevention of partial writes, the integrity of the database can be compromised after recovery.

innodb_doublewrite - an imperfect solution

Since its inception, Innodb has had a mechanism to detect and ignore partial writes via the InnoDB Doublewrite Buffer (also innodb_checksum can be used to detect a partial write).

Doublewrites, controlled by the innodb_doublewrite system variable, comes with its own set of problems. Especially on SSD, writing each page twice can have detrimental effects (write leveling).

Atomic write - a faster alternative to innodb_doublewrite

A better solution is to directly ask the filesystem to provide an atomic (all or nothing) write guarantee. Currently this is only available on the NVMFS (previously called directFS) filesystem on FusionIO devices that provide atomic write functionality. This functionality is supported by MariaDB's XtraDB and Innodb storage engines with MDEV-4338.

Enabling Atomic Writes

To use atomic writes instead of the doublewrite buffer, add:

```
innodb_use_atomic_writes = 1
```

to the my.cnf config file.

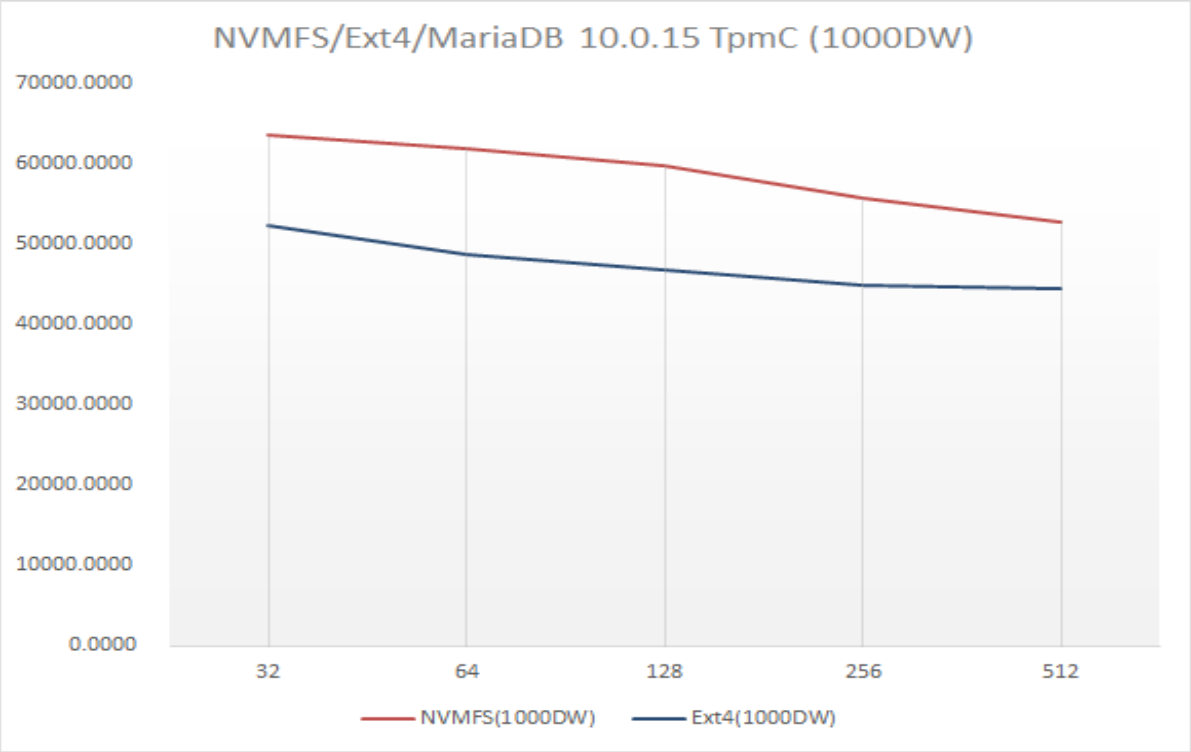
About innodb_use_atomic_writes

The following happens when `innodb_use_atomic_writes` is switched ON

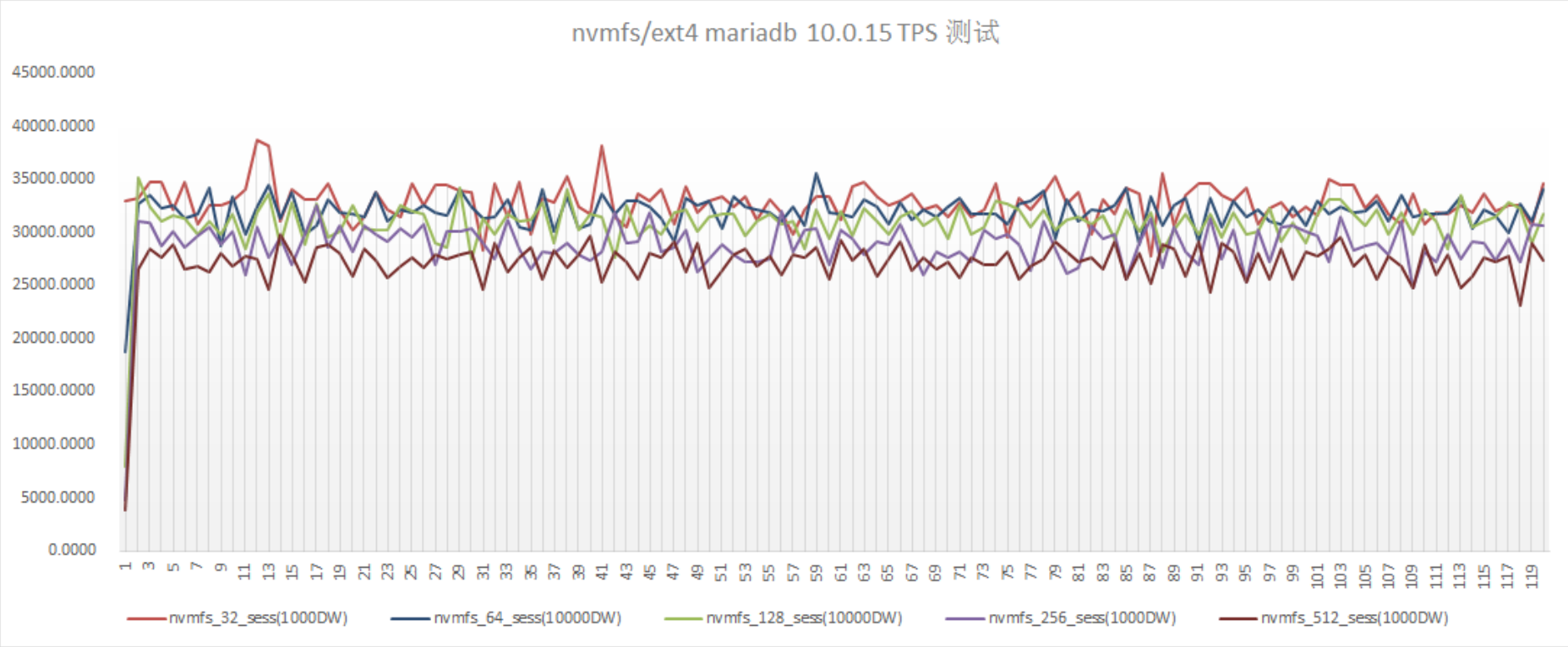
- if innodb_flush_method is neither `O_DIRECT`, `ALL_O_DIRECT`, or `O_DIRECT_NO_FSYNC`, it is switched to `O_DIRECT`
- innodb_use_fallocate is switched `ON` (files are extended using `posix_fallocate` rather than writing zeros behind the end of file)
- Whenever an Innodb datafile is opened, a special `ioctl()` is issued to switch on atomic writes. If the call fails, an error is logged and returned to the caller. This means that if the system tablespace is not located on an atomic write capable device or filesystem, InnoDB/XtraDB will refuse to start.
- if innodb_doublewrite is set to `ON`, `innodb_doublewrite` will be switched `OFF` and a message written to the error log.

Benchmark Result

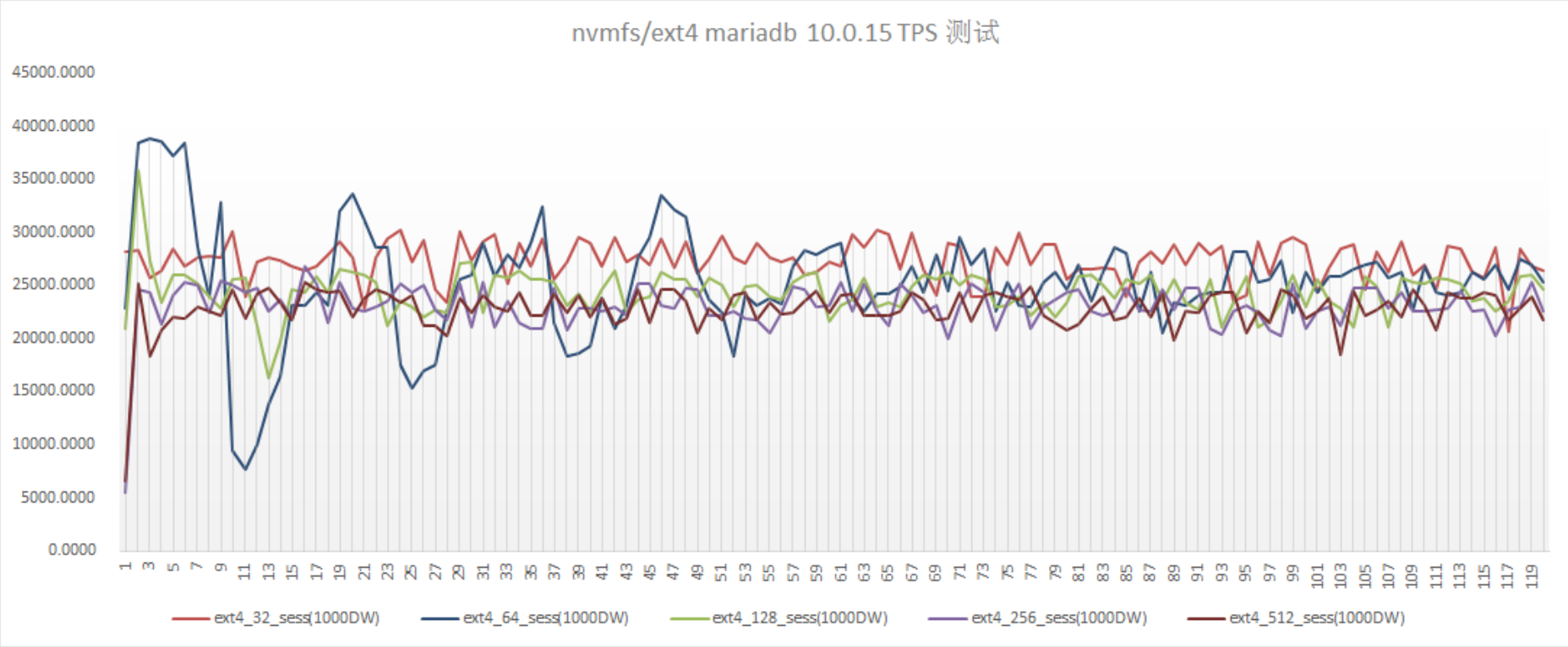
TpmC of all threads



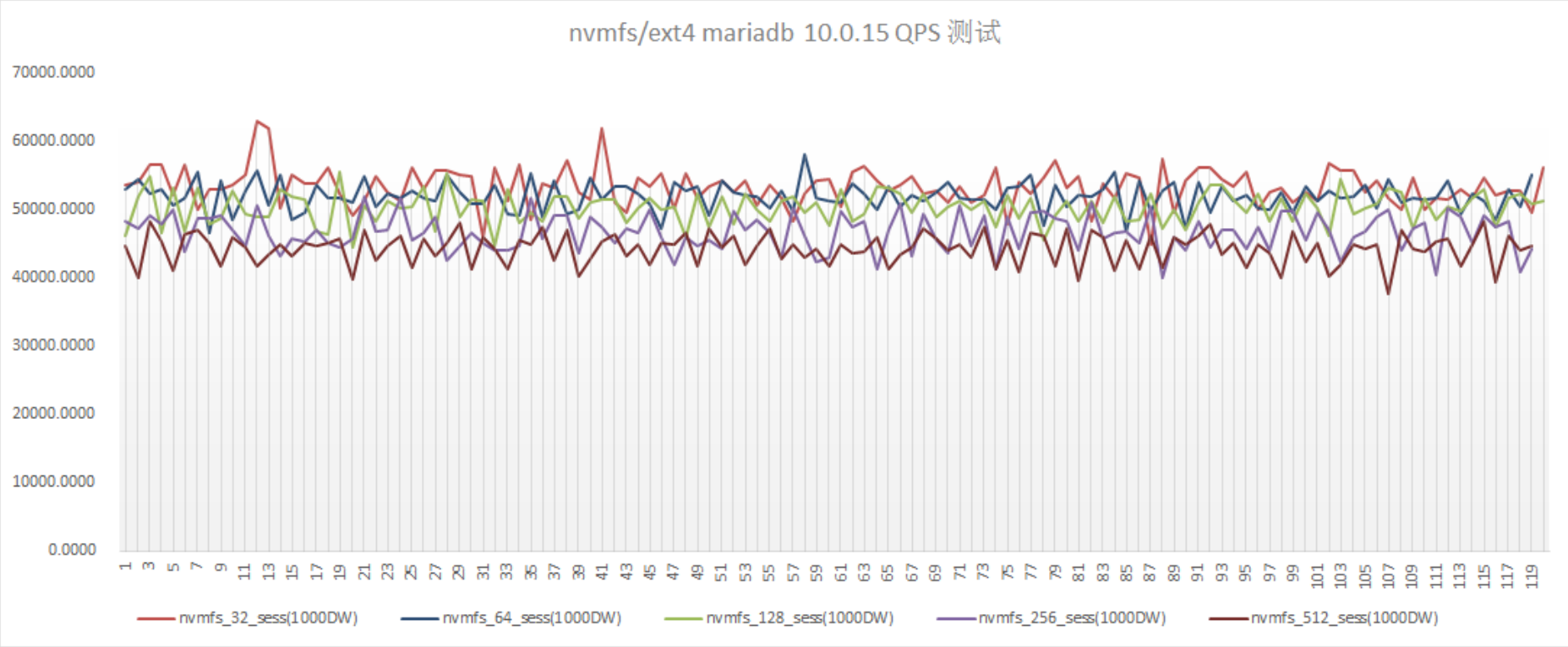
TPS of NVMFS


nvmfs/ext4 mariadb 10.0.15 TPS 测试
45000.0000
40000.0000
35000.0000
30000.0000
25000.0000
20000.0000
15000.0000
10000.0000
5000.0000
0.0000
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99 101 103 105 107 109 111 113 115 117 119
nvmfs_32_sess(1000DW)
nvmfs_64_sess(10000DW)
nvmfs_128_sess(10000DW)
nvmfs_256_sess(1000DW)
nvmfs_512_sess(1000DW)

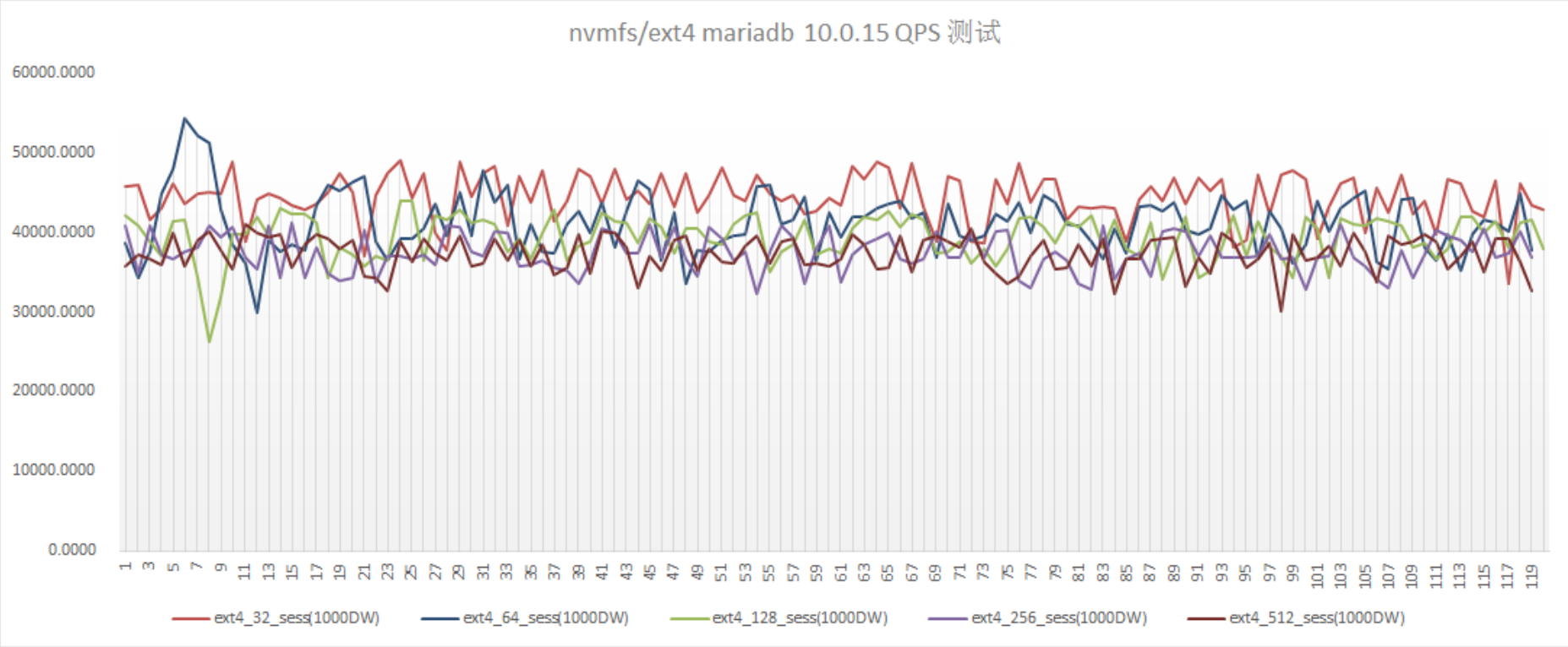
TPS of Ext4


nvmfs/ext4 mariadb 10.0.15 TPS 测试
45000.0000
40000.0000
35000.0000
30000.0000
25000.0000
20000.0000
15000.0000
10000.0000
5000.0000
0.0000
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99 101 103 105 107 109 111 113 115 117 119
ext4_32_sess(1000DW)
ext4_64_sess(1000DW)
ext4_128_sess(1000DW)
ext4_256_sess(1000DW)
ext4_512_sess(1000DW)

QPS of NVMFS


nvmfs/ext4 mariadb 10.0.15 QPS 测试
70000.0000
60000.0000
50000.0000
40000.0000
30000.0000
20000.0000
10000.0000
0.0000
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99 101 103 105 107 109 111 113 115 117 119
nvmfs_32_sess(1000DW)
nvmfs_64_sess(1000DW)
nvmfs_128_sess(1000DW)
nvmfs_256_sess(1000DW)
nvmfs_512_sess(1000DW)

QPS of Ext4


nvmfs/ext4 mariadb 10.0.15 QPS 测试
60000.0000
50000.0000
40000.0000
30000.0000
20000.0000
10000.0000
0.0000
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99 101 103 105 107 109 111 113 115 117 119
ext4_32_sess(1000DW)
ext4_64_sess(1000DW)
ext4_128_sess(1000DW)
ext4_256_sess(1000DW)
ext4_512_sess(1000DW)

Benchmark by SanDisk

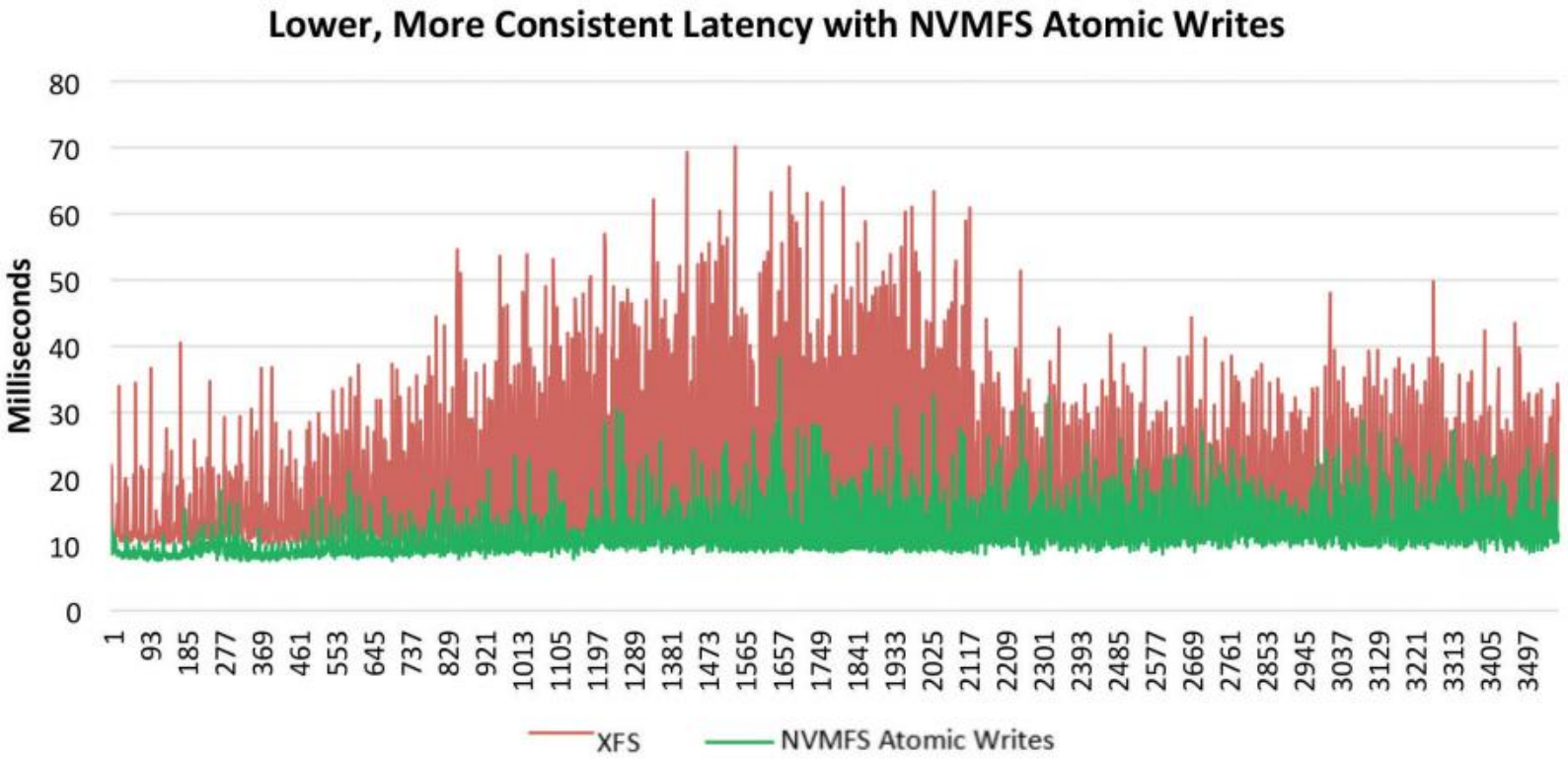


Figure 1: XFS vs. NVMFS Atomic Writes (Sysbench - MariaDB 10.0.15, 4000 OLTP TXN injection/ second, 99% latency, 220 GB data – 10 GB buffer pool)*

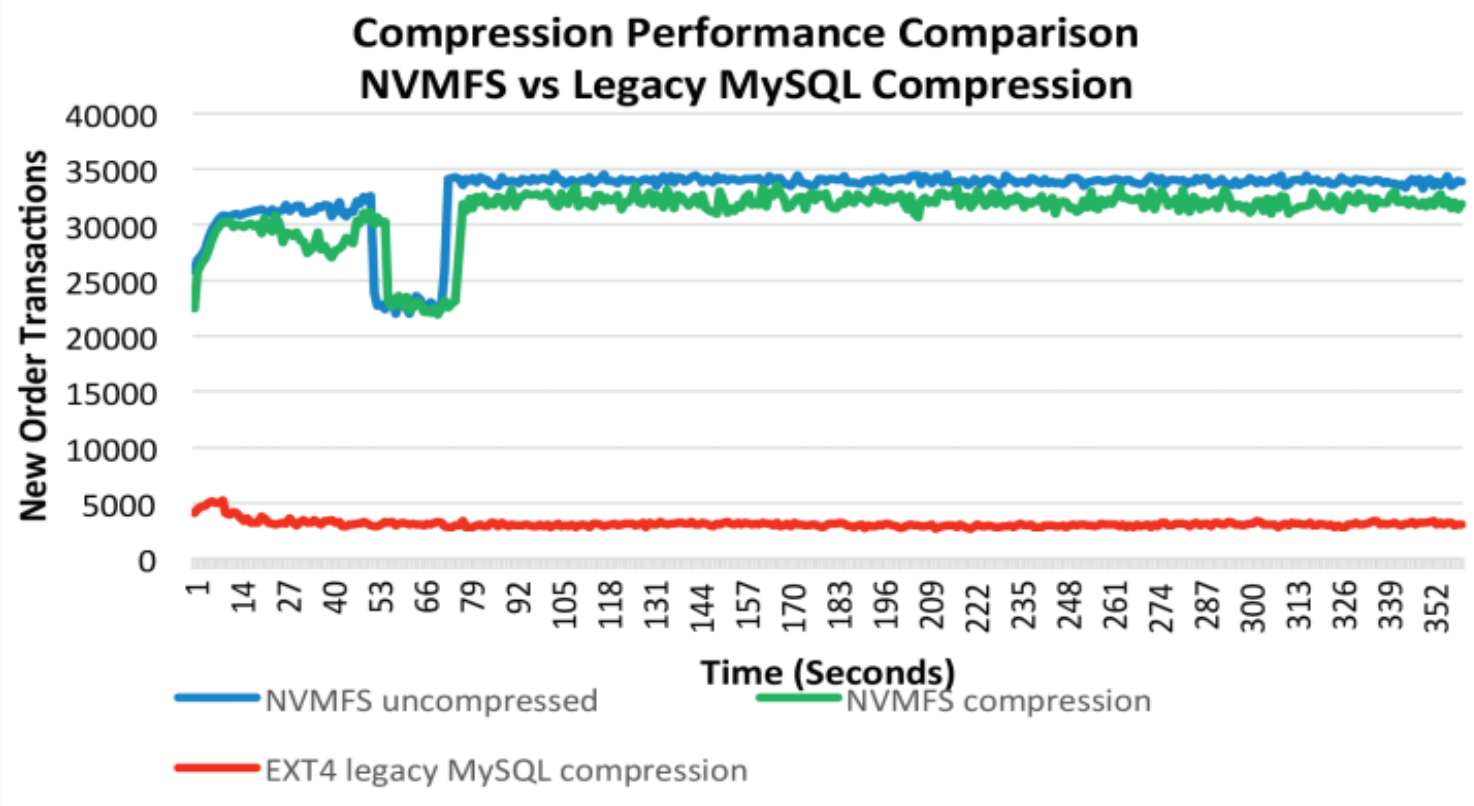


Figure 2: NVMFS compression on MariaDB workload (TPC-C-like benchmark: 1000 warehouses – 75 GB MySQL Buffer pool, MariaDB 10.0.15)*

In My test, MySQL has a better performance on ext4 filesystem which reaches about 5000 TpmC. I wonder if SanDisk really test ext4 filesystem on Flash storage

Details of this Benchmark

Totally NVMFS has a great performance improvement – TPS +20% QPS +25% (and SanDisk's Test shows NVMFS also has a better performance than XFS)

http://www.vmcd.org/docs/NVMFS_INTRO.pdf
http://www.vmcd.org/docs/NVMFS_User_Guide.pdf
http://www.vmcd.org/docs/ext4.log
http://www.vmcd.org/docs/nvmfs.log