Best Practices for MySQL with SSDs

A whitepaper by:

Vijay Balakrishnan
Changho Choi
Veronica Lagrange
Hubbert Smith

Samsung Semiconductor, Inc.

Table of Contents

1. Introduction

Imagine a world where networking speeds became mired at 1MB/s, stayed "stuck" at that speed for 20 years and then became "unstuck," suddenly jumping to 1GB/s (1000X). For storage, something similar has indeed happened with the transition from Hard Disk Drives (HDDs) to Solid State Devices (SSDs). In the past, storage administrators sized systems by estimating the IOPS needed, then buying a quantity of 15K rpm HDDs to provide those IOPS. Often, this resulted in buying more HDD capacity than was necessary. Now, storage administrators typically size a system based on the needed capacity, then buy a quantity of SSDs to provide that capacity, and are pleasantly surprised with the performance that comes along with the capacity. This paper underscores that all SSDs are not alike and that specific configurations and settings need to be considered to maximize performance and system value.

In the software domain, MySQL Server is a commonly used Relational Database Management System (RDBMS).This whitepaper clarifies how today's data centers combine SSDs and MySQL to achieve a substantial business advantage. The paper focuses on *Percona Server* – a free, fully compatible, open source MySQL Server enhancement. Because Percona Server is optimized for the I/O subsystem, it was selected for the experiments herein described, with the best known methods reported on the following pages.

About the Benchmark: TPC-C and tpcc-mysql

TPC-C is an OLTP industry standard benchmark developed and maintained by the Transaction Processing Performance Council (TPC) [www.tpc.org]. It is widely used and well understood, which enables database administrators to quickly correlate quoted TPC-C results to their specific application. TPC-C simulates a wholesale supplier and is centered on processing orders. It is a mixture of read-only and update-intensive transactions that represent complex OLTP application activities. The benchmark implements five types of transactions (new order, payment, delivery, order status and stock level), and reports performance by measuring the number of new orders processed per minute (tpmC). It contains strict guidelines that must be observed in any official implementation. Furthermore, in spite of its complexity, TPC-C is easily scaled up or down to fit the system under test.

Tpcc-mysql is Percona's TPC-C implementation. It is written in C and follows Revision 5.11 of the MySQL standard specification. We tested with tpcc-mysql "out-of-the box" code, downloaded from the Percona GitHub site.

2. Hardware Configurations

In the past, OLTP systems were frequently bottlenecked by IO, meaning CPUs were constantly waiting on HDDs to respond. But when we replace HDDs with NVMe SSDs on a Dual Socket

Server, the CPU becomes the bottleneck. In an attempt to minimize the CPU bottleneck, we tested two types of database servers: Dual Socket (12 core) and Quad socket (28 core).

Dual-socket (12 Core) configuration

	Database Server	(Client) Load Generation Server
CPU		
Model Name	Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz	Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz
Processors	12	8
Cores	24	32
Memory	64GB	128GB
OS version	Linux 4.4.0-040400-generic	Linux 3.19.0-14-generic
MySQL Server	**5.7.11**	
Percona Server	**5.7.10-3**	
Storage		
SAS HDD	2x SEAGATE ST600MP0005 15K rpm	
SATA SSD	2x Samsung 850 PRO	
NVMe SSD	2x Samsung XS1715	

Quad-socket (28 Core) Configuration

	Database Server	(Client) Load Generation Server
CPU		
Model Name	Intel(R) Xeon(R) CPU E7-4850 v3 @ 2.20GHz	Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz
Processors	28	8
Cores	112	32
Memory	124GB	126GB
OS version	Linux 4.4.0-040400-generic	Linux 3.19.0-14-generic
Percona Server	**5.7.11-4**	
Storage		
SAS HDD	2x SEAGATE ST600MP0005 15K rpm	
SATA SSD	2x Samsung 850 Pro	
SAS SSD	2x Samsung PM1633	
NVMe	2x Samsung PM1725	

It is generally accepted that OLTP database applications have been I/O bound. They typically have not exhausted CPU capacity. While this is true for HDDs, we see a paradigm shift with NVMe SSDs.

With HDDs, CPUs are always less than 1% busy, while with NVMe SSDs CPU utilization goes up to more than 30% with 100+ connections. This 30 times CPU increase translates into 110+ times more transactions per minute (tpmP), making the case for better server utilization when using NVMe SSDs.

Mean CPU % utilization (User+Sys) on the Quad Socked Server

Mean CPU %	15K rpm SAS-HDD	SATA SSD	SAS SSD	NVMe SSD
50 connections	0.4%	5.8%	15.9%	22.3%
100 connections	0.4%	13.4%	23.3%	32.1%
150 connections	0.5%	15.0%	26.4%	28.9%
200 connections	N A	16.4%	27.2%	33.8%

This report presents current results for four storage devices: PM1725 (NVMe SSD), PM1633 (SAS SSD), 850Pro (SATA SSD), and a Seagate 15Krpm HDD (HDD). We show how faster SSD storage is revolutionizing typical OLTP applications that have been traditionally I/O bound, thereby increasing throughput by orders of magnitude, measured with *tpcc-mysql benchmark.*

3. Optimizing MySQL Server and Percona Server for SSDs

We observed that the NVMe SSDs performed either 200% or 700% better than SATA SSDs, depending on the selected configuration. With that in mind, we then proceeded to determine the best performance level for each device. In addition to adjusting MySQL configuration parameters, we tried increasing and decreasing the database size[i], as well as partitioning (sharding) the database across multiple storage volumes.

MySQL Server throughput (tpmP[1]) for 200 and 250 connections - SATA versus NVMe on the Dual Socket Server

1,000-wh[2,3]; 200-connections	SATA SSD	NVMe SSD	pct diff
Config #1 – MySQL baseline	7,366.55	24,440.57	232%
Config #2 – sub-optimal config	4,478.28	37,802.13	744%
pct diff	-39%	55%	
1,000-wh; 250-connections	SATA	NVMe	pct diff
Config #1 – MySQL baseline	6,668.33	23,175.19	248%
Config #2- sub-optimal config	4,457.47	33,857.05	660%
pct diff	-33%	46%	

[1] tpmP= New Order transactions per minute
[2] TPC-C database size is defined by number of warehouses (wh). All tables are scaled up or down based on the number of warehouses defined. A 10-wh database occupies roughly 1GB of disk space.

BKM	Use SSDs instead of HDDs; there is no viable reason to use HDDs. First preference: NVMe SSD Second preference: SAS SSD

4. Flush Method and Buffer Pool

BKM	Switch the flush_method to 'O_DIRECT' MySQL uses its buffer pool as disk cache, rather than using Linux filesystem as buffer space.

BKM	Increase the buffer pool from 3GB to 12GB Larger buffer pool improves performance for all storage types.

After considerable experimentation and analysis, we switched the flush_method to 'O_DIRECT', and then stopped using Linux's filesystem buffer space and instead began using MySQL buffer pool as the disk cache. Configuration #2 (sub-optimal, see appendix A) has 3GB of buffer pool, and therefore very little buffer space to be spared as disk cache.

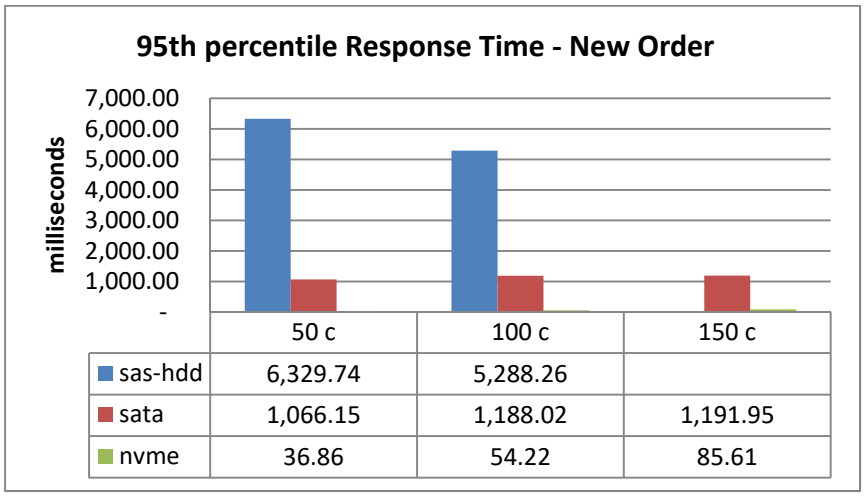
In Configuration #3 (MySQL Optimal, see appendix A) we increased the buffer pool to 12GB, which benefitted all three storage types.

The **NA** for 15K rpm SAS-HDD in Configuration #2 (sub-optimal) indicates we cannot run tpcc-mysql with that setup (we get "lock wait timeout" error messages, meaning that the system is not able to complete transactions within the expected time). When we move to Configuration #3 (MySQL optimal), we get a significant boost for all devices, especially for 15K rpm SAS-HDD, which is now able to complete the test without errors.

5. Impact of Latency

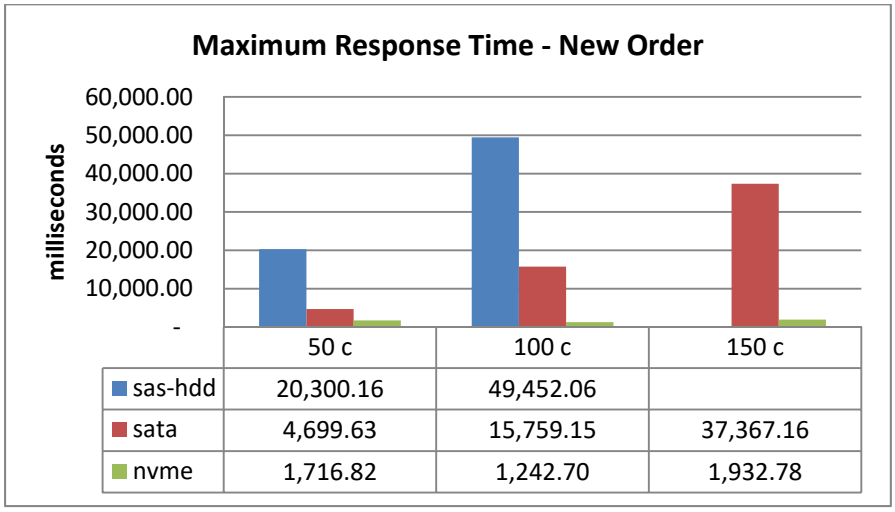
Think of latency as a traffic signal on a popular roadway. If the roadway has few cars, it's an inconvenience. If the roadway has too many cars, traffic backs up which can be a disaster; this is true in I/O. This is when data center managers can really appreciate the benefits of faster storage devices. When everything else is the same, most users connected to an "NVMe system" will see their transactions completed in less than 90 milliseconds – two orders of magnitude faster than the 5,000+ milliseconds users connected to the "HDD system" experience.

MySQL Server 95th percentile response times for Config #3 (dual socket)



	50 c	100 c	150 c
sas-hdd	6,329.74	5,288.26	
sata	1,066.15	1,188.02	1,191.95
nvme	36.86	54.22	85.61

MySQL Server maximum response times for Config #3 (dual socket)



	50 c	100 c	150 c
sas-hdd	20,300.16	49,452.06	
sata	4,699.63	15,759.15	37,367.16
nvme	1,716.82	1,242.70	1,932.78

In the case of SAS-HDD with 150 connections, the I/O traffic backs up and the system becomes unresponsive. If this happens in an actual market environment, anyone using the database (bank tellers, cash registers, RFID checkers) will see their transactions cease to operate and their business grind to a halt.

6. Optimization Guidelines

BKM	**innodb_thread_concurrency**. Different values were tried; the best performance came with the default 0 (unlimited thread_concurrency).

BKM	**innodb_adaptive_hash_index**. Turned 'OFF', since OLTP workloads typically do not reuse data from previous queries.

BKM	**innodb_fill_factor**. Indicates the percentage of space on each B-tree page that is filled during a sorted index build. 50 works best for workloads with lots of INSERTs.

BKM	**Separate log_dir and datadir**. All storage types benefit from this. For both Percona and MySQL Server, it means setting up the parameters from Appendix A marked with either **<data storage>** or **<log storage>** to the appropriate storage location.

7. Performance Results Comparison

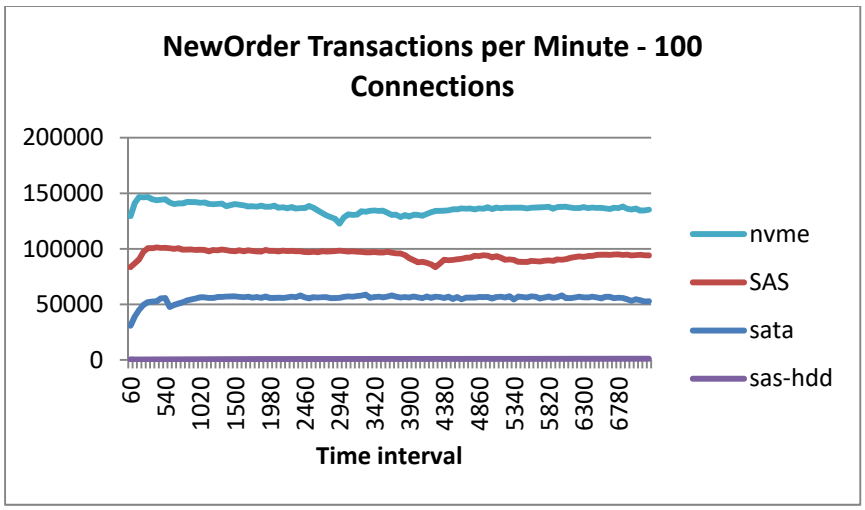
Here throughput results for all storage types were reviewed. Observe that the SATA SSD result is 32+ times better than that of the HDD, and that NVMe SSD throughput is two to five times better than SATA, and up to 51% better than SAS SSD.

Throughput in tpmP – Quad Socket

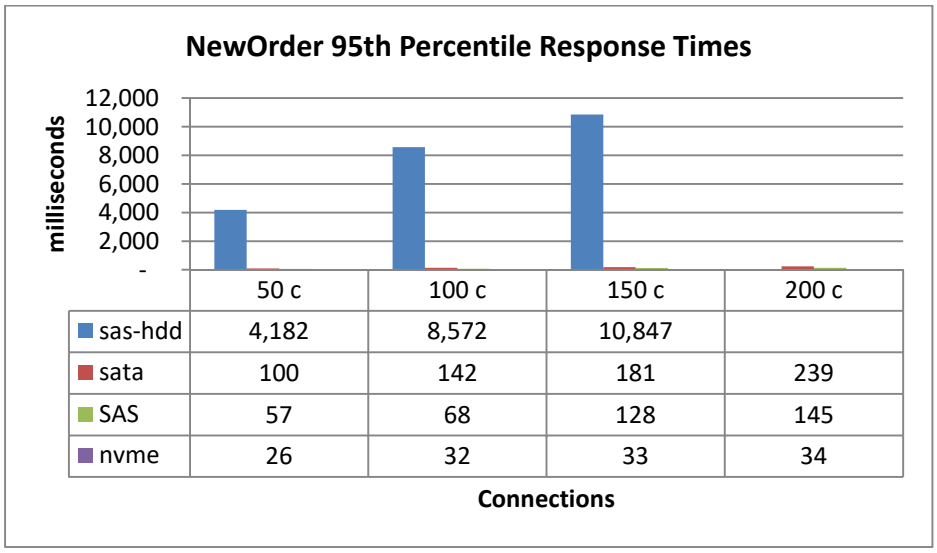
Connections	SAS-HDD	SATA SSD	SAS SSD	NVMe SSD	SAS_HDD -> SATA	SAS_HDD -> SAS-SSD	SAS_HDD -> NVMe	SATA -> SAS	SATA -> NVMe	SAS-> NVMe
50	742	23,868	73,236	110,357	3117%	9770%	14773%	207%	362%	51%
100	757	55,378	94,702	136,479	7218%	12415%	17935%	71%	146%	44%
150	1,090	58,123	96,691	138,748	5234%	8773%	12632%	66%	139%	43%
200	N A	58,276	97,796	139,158				68%	139%	42%

The best tpcc-mysql throughput, 139K tpmP, is obtained using NVMe SSDs running with 200 connections. The 100 connection result, 136K tpmP, is 180+ times better than HDD. Next figure shows *New Order* transactions executed over time for all four storage types for the 100-connection case.

New Order transactions over time (Higher is better)

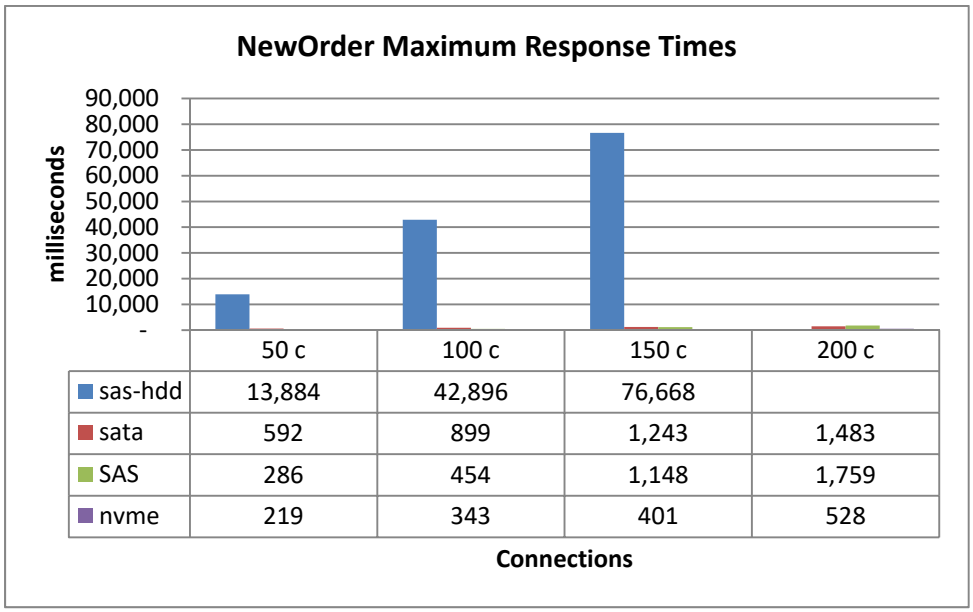


New Order 95th percentile response times (Lower is better)



Notice for 15K SAS-HDD 200c, the result took too long to obtain (failed test). In an actual market environment, this represents a non-responsive system impacting business operations.

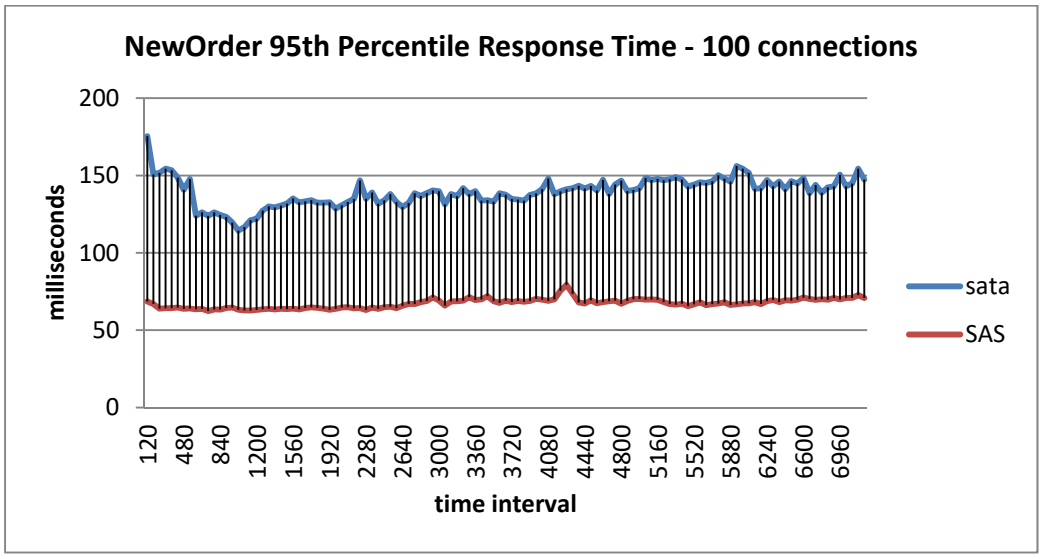
New Order maximum response times (Lower is better)



	50 c	100 c	150 c	200 c
sas-hdd	13,884	42,896	76,668	
sata	592	899	1,243	1,483
SAS	286	454	1,148	1,759
nvme	219	343	401	528

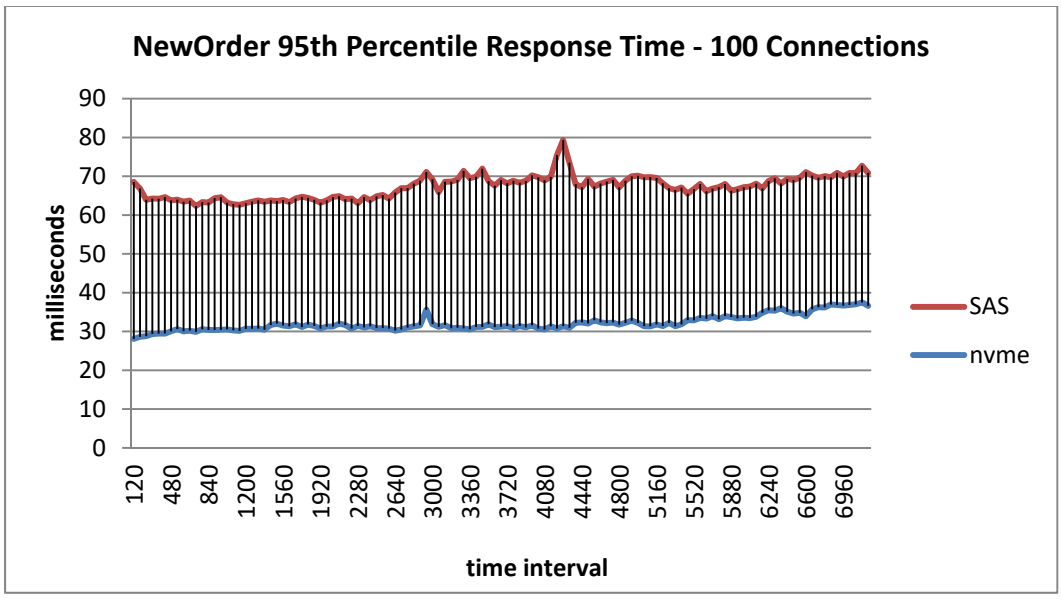
The 95th percentile response times for *New Order* transactions over the duration of each measurement.

SATA SSD and SAS SSD latencies (Lower is better)



SAS SSDs demonstrate significantly lower latencies than SATA SSD for this OLTP workload.

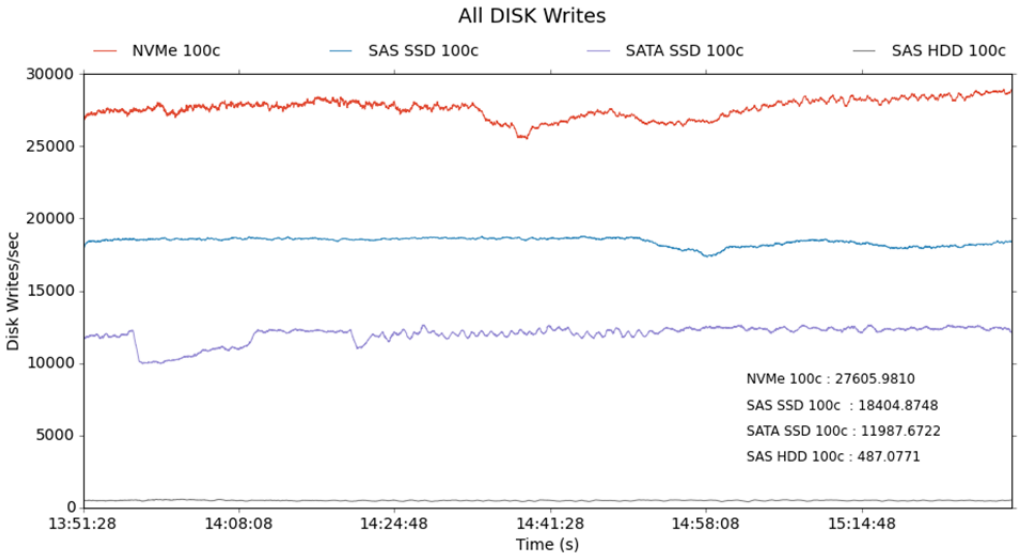
SAS and NVMe latencies (Lower is better)



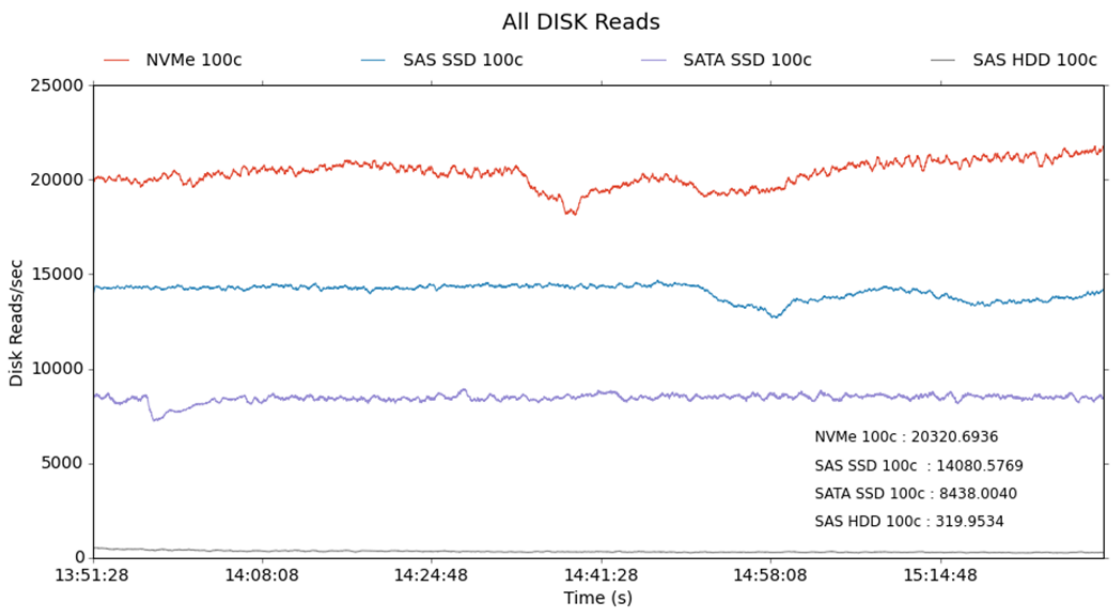
NVMe SSDs demonstrate significantly lower latencies than SAS SSDs for this OLTP workload.

8. tpcc-mysql System Resource Utilization

All Disk Writes 100 connections (Higher is better)



All Disk Reads 100 connections (Higher is better)



9. Conclusions

Not all SSDs are the same but all SSDs perform substantially better than HDDs. NVMe SSDs can be up to five times more performant than SATA SSDs, as tested with Percona. SAS-SSD can be up to 125 more performant than SAS-HDD. NVMe SSDs are 180 times more performant than 15K rpm HDDs.

10. Appendix A: MySQL and Percona Configurations

This appendix contains the many configuration settings for both MySQL Server and Percona Server discussed in this report.

MySQL Server and Percona Server configurations

Parameter Name	Config #1 MySQL Initial	Config #2 MySQL Suboptimal	Config #3 MySQL Optimal	Config #4 Percona Optimal
datadir	/<data storage>/**mysql_data/mysql**			
tmpdir	**/tmp**			/<log storage>/mysql_log
lc-messages-dir	**/usr/share/mysql**			
explicit_defaults_for_timestamp				
innodb_log_group_home_dir	/<log storage>/**mysql_log**			

innodb_undo_directory	/<log storage>/mysql_log			
innodb_buffer_pool_size	3GB		12GB	
innodb_thread_concurrency	0			
innodb_temp_data_file_path	'../../..</log storage>/mysql_log/ibtmp1:72 M:autoextend'			
innodb_page_cleaners	32			8
innodb_buffer_pool_instances	32			8
innodb_io_capacity	300000			15,000
innodb_io_capacity_max	600000			25,000
innodb_adaptive_hash_index	OFF'			0
innodb_fill_factor	50			100
innodb_write_io_threads	16			
innodb_read_io_threads	16			
innodb_flush_method	<empty>	O_DIRECT	O_DIRECT	O_DIRECT
innodb_flush_neighbors	1			0
query_cache_size	0			
log_timestamps	'SYSTEM'			
table_open_cache	8000			
table_open_cache_instances	16			64
back_log	1500			
max_connections	4000			
innodb_use_native_io	ON (default)			OFF / ON
## To be able to use many connections in TPC-C:				
max_prepared_stmt_count	64000			
# files				
innodb_log_files_in_group	3			
innodb_log_file_size	48MB	1G	1G	10G
innodb_open_files	4000			
# tune	=			
innodb_checksum_algorithm	NONE			crc32
innodb_max_dirty_pages_pct	90			
innodb_max_dirty_pages_pct_lwm	10			
innodb_lru_scan_depth	4000			8192
join_buffer_size	32K			
sort_buffer_size	32K			
innodb_spin_wait_delay	96			6
# perf special				
innodb_flush_neighbors	0			

innodb_max_purge_lag_delay	**30000000**			0
# Monitoring				
innodb_monitor_enable	**'%'**			<empty>
performance_schema	**OFF**			

..oo0oo..

[i] TPC-C database size is defined by number of warehouses (wh). All tables are scaled up or down based on the number of warehouses defined. A 10-wh database occupies roughly 1GB of disk space.