Accelerating OLTP performance with NVMe SSDs

Veronica Lagrange
Changho Choi
Vijay Balakrishnan

Memory Solutions Lab.
Samsung Semiconductor, Inc.
9/01/2016
Version 3.0

Table of Contents

1. Introduction

Today's datacenter transition to high performance, open-source and hyper scale architectures continues its relentless acceleration. In the hardware domain, *the use of NAND flash-based SSDs in enterprises has grown at a ~33% compound annual growth rate worldwide for the past three years* [3]. In the software domain, MySQL Server is the world's second most used Relational Database Management System (RDBMS), and the most used open-source, client-server RDBMS [4]. This white paper contends that it is only natural for today's datacenters to combine NVMe SSDs and MySQL to achieve an unprecedented business advantage.

Going one step further, we find *Percona Server* [5], a free, fully compatible, open source MySQL Server enhancement. Because Percona Server is especially optimized for the I/O subsystem, we were able to extract more throughput out of it when using fast storage devices, and therefore use it for the final experiments reported here.

An RDBMS typically supports high performance Online Transaction Processing (OLTP) applications that require transactional integrity. Examples include ATM withdrawals, online purchases, stock buy/sell orders, etc. TPC-C is a 23-year-old OLTP benchmark industry standard that simulates an online store. "Users" place orders, make payments, and check the status of their orders while "warehouses" make deliveries and check their stock levels. TPC-C is readily understood, which enables database administrators to correlate quoted TPC-C results to their specific application.

We experimented with Percona's implementation of the TPC-C standard, also known as *tpcc-mysql*. We ran tpcc-mysql using a Percona Server version 5.7.11 running on a Linux box (for full configuration settings, see Table 8 in Appendix A and Table 10 in Appendix B). This report presents current results for four storage devices: PM1725 (NVMe), PM1633 (SAS), 850Pro (SATA), and a Seagate 15Krpm HDD (HDD). We show how faster storage devices revolutionize typical OLTP applications that are traditionally I/O bound, thereby increasing throughput by orders of magnitude.

The report is organized as follows. Section 2 describes TPC-C, tpcc-mysql and how it differs from the standard TPC-C. Section 3 discusses MySQL Server optimizations and how that knowledge can be leveraged for applications running both on MySQL and Percona Servers. Section 4 explains our final configuration and testing details, while presenting results, and analyzing them. Section 5 summarizes the key takeaway points from this exercise. Section 6 presents conclusions and future directions.

2. 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]. TPC-C simulates a wholesale supplier and is centered on processing orders. It is a mixture of read-only and update-intensive transactions that simulate complex OLTP application activities. It implements five transaction types (new order, payment, delivery, order status and stock level), and reports performance by measuring the number of new orders processed per minute (tpmC) [1]. It contains strict guidelines that must be adhered to in any official implementation. Furthermore, in spite of its complexity, TPC-C is easily scaled up or down to appropriately 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 standard specification [1]. It does not, however, implement the following features:

- Rollbacks: 1% of New Order transactions are supposed to be rolled back due to "user error."
- "thinking" or "keying" times. Herein, "user" connections will submit another transaction request as soon as their current one completes, while the standard calls for a pre-determined delay to be added between user requests.
- Each connection sends requests evenly to all warehouses, while the standard calls for each connection to be assigned to one warehouse. That is, each connection should submit requests to its "home warehouse" 99% of the time and to some other, random generated warehouse, 1% of the time.
- Other minor bugs with Payment and Load codes.

We tested with tpcc-mysql "out-of-the box" code, downloaded from its GitHub site [2] on Feb 17, 2016. Because TPC did not certify these tests, we are reporting our findings in "Percona new orders processed per minute" (tpmP).

3. Optimizing MySQL Server and Percona Server for SSDs

Optimizing a complex system is a balancing act, one further complicated by the presence of varying hardware. For example, the "best" configuration for disk type *A* could be really problematic for disk type *B*, or for a major component change, such as CPU or Software, it may be necessary to reevaluate previously optimized and "guaranteed" setups. Yet, because choosing the best configuration will invariably yield significant performance gains, this is usually time well spent. In this section, we describe our MySQL and Percona optimization steps for using faster storage devices with OLTP applications.

We started by reviewing both external and Samsung internal literature and compiled a list of configuration parameters others have experimented with and deemed important [6, 7, 8, 9]. The MySQL Open Source community recommends reports [8, 9] which can be found under the "MYSQL Performance Tuning and Optimization Resources" section on MySQL's site [10]. These reports used different MySQL versions, and different TPC-C implementations. They all used SSDs, or both SSDs and HDDs. Most used the InnoDB storage engine. Table 5 (see Appendix A: MySQL and Percona configurations), summarizes the relevant configuration parameters mentioned.

Before we go any further, it is important to understand that MySQL (and Percona) are built for stability. They contain multiple gatekeepers and constraints to guarantee that important components are not overwhelmed beyond their capacity. This is particularly true for the I/O subsystem.

We start our tests with Config#1 (see Table 6), followed by Config#2. Here, we used a dual-socket server (see Table 9 in Appendix B), MySQL Server and a TPC-C database with 1,000 warehouses. For scalability, we ran tpcc-mysql with different user loads, also known as "connections", and quickly realized that while Config#2 is better for NVMe SSDs, it is worse for both SATA[1] and SAS HDDs. Figure 1 compares Configs #1 and #2 using SATA and NVMe for two workloads (200 and 250 connections). Figure 2 compares Configs #1 and #2 for SAS HDD and SATA for a 50-connection workload.

From Figure 1, one could say that NVMe storage delivers either 200% or 700% more performance than SATA storage, depending on the selected configuration. With that in mind, we proceeded to obtain the best performance from each device. To guide optimization efforts, we also heavily relied on system performance data, such as CPU, memory, network, and disk I/O, collected using *collectl* and other internal tools. Besides MySQL configuration parameters, we also tried increasing and decreasing the database size, as well as partitioning (*sharding*) the database across multiple storage volumes. We settled on a 500-warehouse database size. Partitioning MySQL tables didn't work for our test case because MySQL does not support foreign key constraints on partitioned tables.

1,000-wh; 200-connections	SATA-SSD	NVMe	pct. diff
Config #1	7,366.55	24,440.57	232%
Config #2	4,478.28	37,802.13	744%
pct. diff	-39%	55%	
1,000-wh; 250-connections	SATA-SSD	NVMe	pct. diff
Config #1	6,668.33	23,175.19	248%

[1] For brevity, when we omit the storage type, we mean SSD. For example, when we say SATA, we mean SATA SSD. When we say NVMe, we mean NVMe SSD.

Config #2	**4,457.47**	**33,857.05**	660%
pct. diff	**-33%**	**46%**	

Figure 1. MySQL Server throughput (tpmP) for 200 and 250 connections – SATA-SSD versus NVMe

1,000-wh; 50-connections	SAS-HDD	SATA-SSD	pct. diff
Config #1	**599.70**	**11,376.09**	1797%
Config #2	**380.80**	**6,725.16**	1666%
pct. diff	**-37%**	**-41%**	

Figure 2. MySQL Server throughput (tpmP) for 50 connections - SAS-HDD versus SATA-SSD

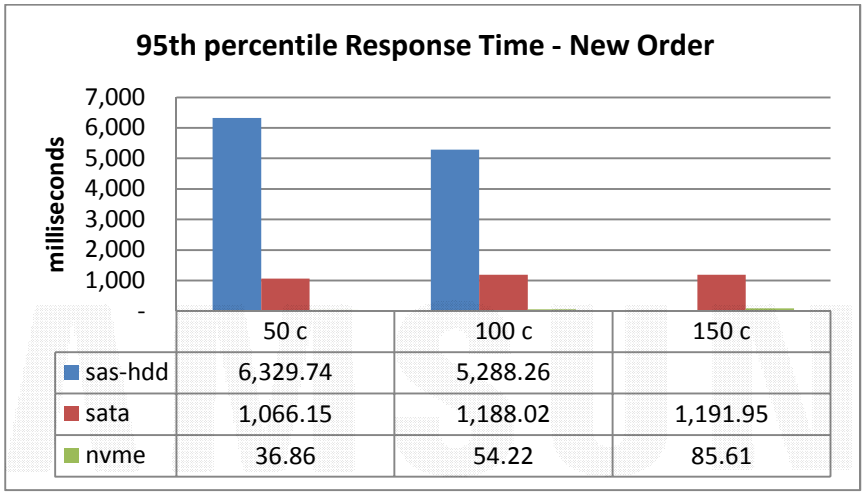
After much experimentation and analysis, we determined the issue with Config#2: when we switched the flush_method to 'O_DIRECT', MySQL stopped using Linux's filesystem buffer space and started using its buffer pool as disk cache (on top of its other functions). Configurations #1 and #2 have only 3GB of buffer pool, and therefore very little real estate to be spared as disk cache. Only a very fast device, such as an NVMe SSD, could benefit from that. In Config#3 (see Table 6), we increased the buffer pool to 12GB, which benefits all three storage types. Figure 3 summarizes comparisons between configurations #1 and #3. Tests in Figure 3 use a 500-warehouse database because it yields better performance than the 1,000-warehouse database. Furthermore, we use the 100-connections workload because it is both a middle of the ground load for all devices, and it is near the knee of the performance curve for NVMe storage. The **NA** for SAS-HDD in Config#1 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 Config#3, we get a significant boost for all devices, especially for SAS-HDD, which is now able to complete the test without errors.

500-wh; 100-connections	SAS-HDD	SATA-SSD	NVMe	HDD->SATA	SATA->NVMe	HDD->NVMe
Config #1	NA	3,043.33	42,127.02		1284%	
Config #3	1,191.47	4,775.19	**79,337.96**	301%	1561%	6559%
pct. diff		57%	88%			

Figure 3. MySQL Server throughput (tpmP) for 100 connections

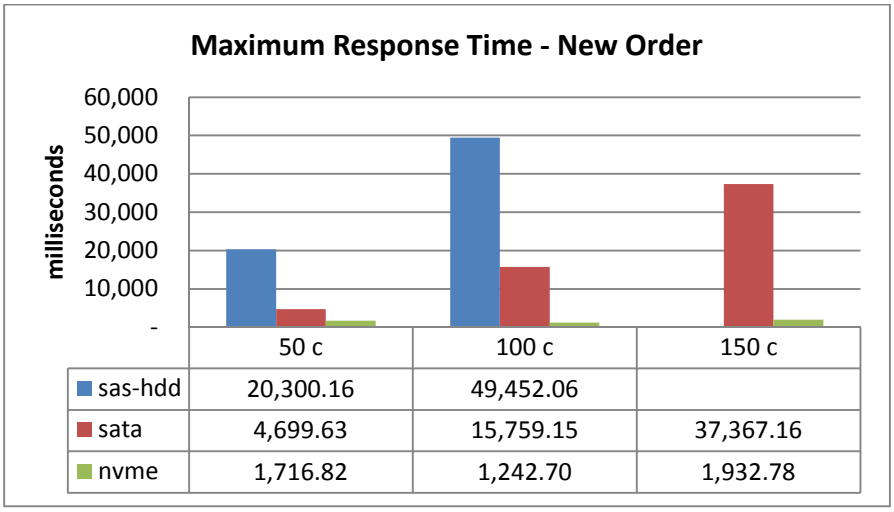
Even though all storage types benefitted from Config#3, SAS-HDD benefitted the most (notice how the percentage differences between types changed from Figure 2 to Figure 3). The SAS-HDD to SATA difference, which used to be over 1,000%, is now only 301%. And the SATA to NVMe difference decreased slightly. This seems to indicate that Config #3 benefits slower devices more than faster devices.

Another very important result for OLTP applications is response time, or how long it takes for transactions to complete. Figure 4 shows the 95th percentile response times for the *New Order* transactions for all storage devices when running with Config#3. Figure 5 shows the Maximum response times for the same tests. This is when we can really appreciate the benefits of faster storage devices. When everything else is the same, most users who connect to an "NVMe system" will see their transactions complete in less than 90 milliseconds – two orders of magnitude faster than the 5,000+ milliseconds users connected to the "HDD system" will experience.



	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

Figure 4. MySQL Server 95th percentile response times for Config#3



	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

Figure 5. MySQL Server maximum response times for Config#3

Our next move was to install the Percona Server in our dual socket server. The Percona Server was expected to be more performant than the MySQL Server. To our surprise, results with Config #3 were worse than the results with MySQL Server. We achieved 53% less throughput for NVMe SSDs, 47% less throughput for SATA, and a mere 4% throughput improvement for SAS-HDD. Percona Server has a few new parameters to note, a subset of which is listed in Table 7. Moving the innodb_parallel_doublewrite_path to the log device decreased the gap by 50%, but Percona was still less performant than MySQL. Again, after some investigation as well as consulting with Percona [13], we arrived at Config #6 (Table 6 and Table 7), which is still slower. Finally, turning innodb_use_native_io back ON (a.k.a. Config#A) produced the best results yet. Figure 6 compares the best MySQL Server versus the best Percona Server results on the same system.

tpmP	MySQL Config#3	Percona Config#A	pct. diff
SAS-HDD (50 c)	766.36	778.68	2%
SATA (50c)	2,891.30	3,257.02	13%
NVMe (100c)	79,337.96	103,683.58	31%

Figure 6. Dual-core results - MySQL versus Percona Server

In Figure 6, notice that there is not much difference for SAS-HDD, because that test is performance-bound by the HDD storage device. There is a small improvement for SATA and a strong 31% improvement for NVMe. Looking at system performance data, we see that the Percona Server uses CPU more efficiently than the MySQL Server (Figure 7). Furthermore, by externalizing the xb_doublewrite file, and allowing the user to move it to another location, Percona Server better distributes the write load between the available data and log disks.

	MySQL Server	Percona Server
Mean CPU (User+Sys) %	81	65
Mean CPU Wait %	10	15
Mean Data Disk Reads IOPS	19,919	30,291
Mean Data Disk Writes IOPS	79,274	31,933
Mean Log Disk Writes IOPS	3,541	32,634

Figure 7 System metrics for NVMe on Dual-core – MySQL versus Percona Server

Next, we moved to a more powerful quad socket server (see Table 10), and repeated the Percona Server experiments. We now have updated the NVMe and SAS-SSD devices that are currently shipping. The HDD and SATA-SSD remain the same (see Table 9 and Table 10).

Notice that with Percona Server quad-socket server, SATA-SSD results are 700+% better because these tests were CPU bound on the smaller system (Figure 8 and Figure 9). Observe two important findings:

1. We cannot scale beyond 100 connections (Figure 10);
2. Even though we are not exhausting the available resources, increasing innodb_io_capacity beyond 15,000 will hurt throughput (Figure 11).

The current understanding is that we have reached the Server's limit and will not achieve better performance without some coding improvement. Still, we have come a long way, from the initial results around 23K tpmP (for NVMe and the MySQL Server on the dual-socket) to 136K tpmP (for NVMe storage and the Percona Server on the quad-socket).

Because the NVMe and SAS devices used in the quad-socket tests are not present in the dual-socket system, we did not directly compare them. In section 4, we present and analyze the final results using the Percona Server on the quad-socket server.

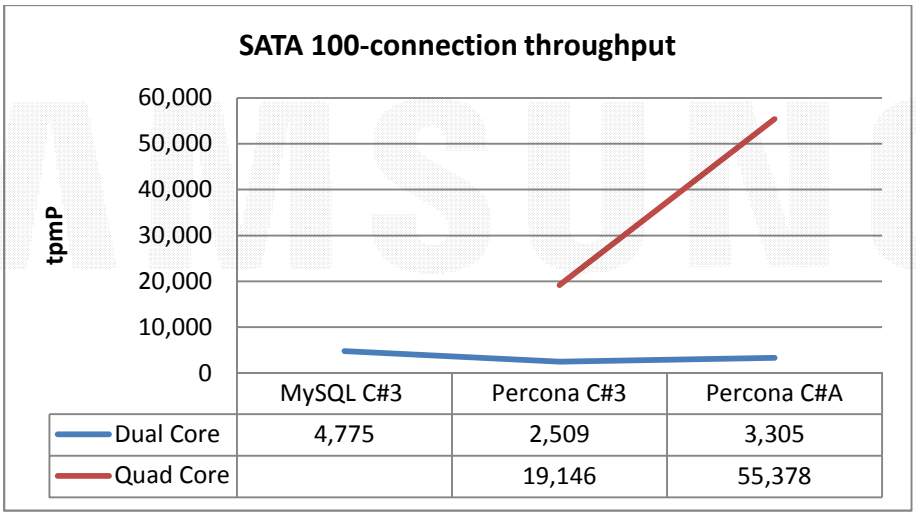


Figure 8. Dual-socket versus Quad-socket SATA results

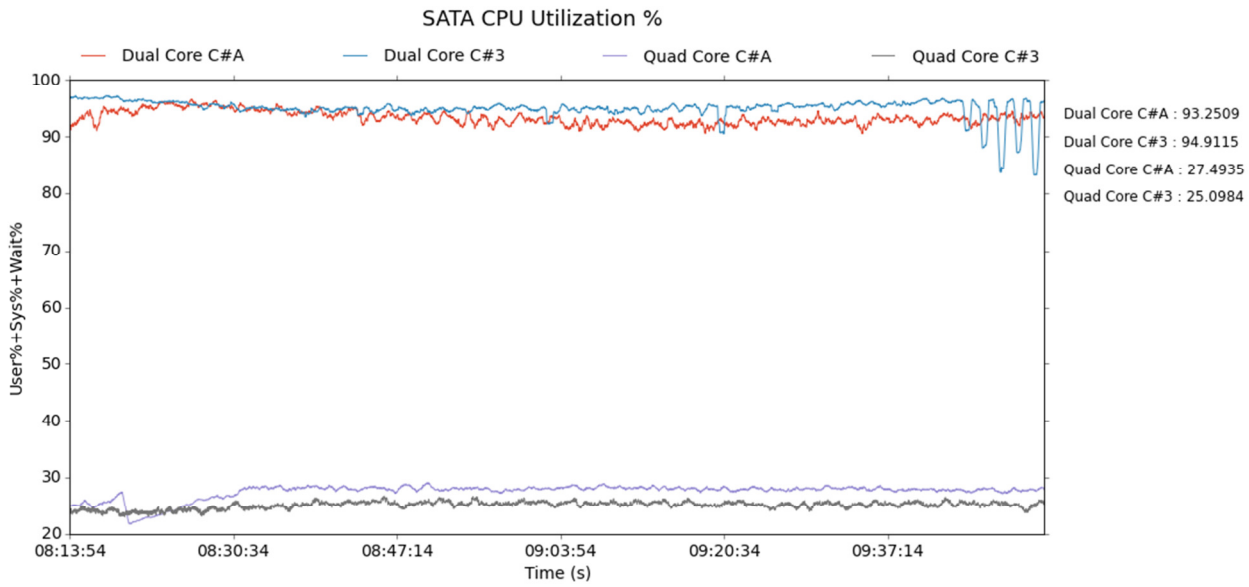
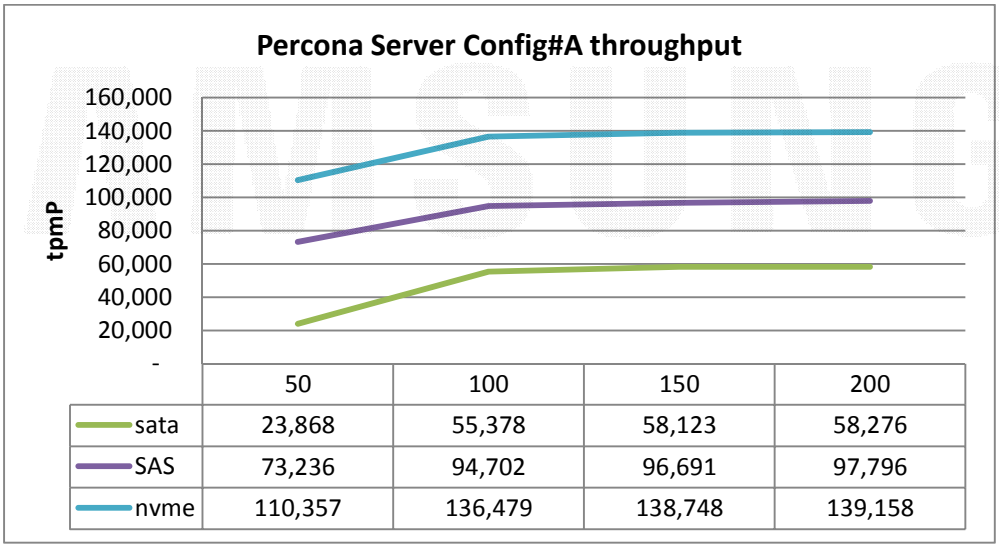
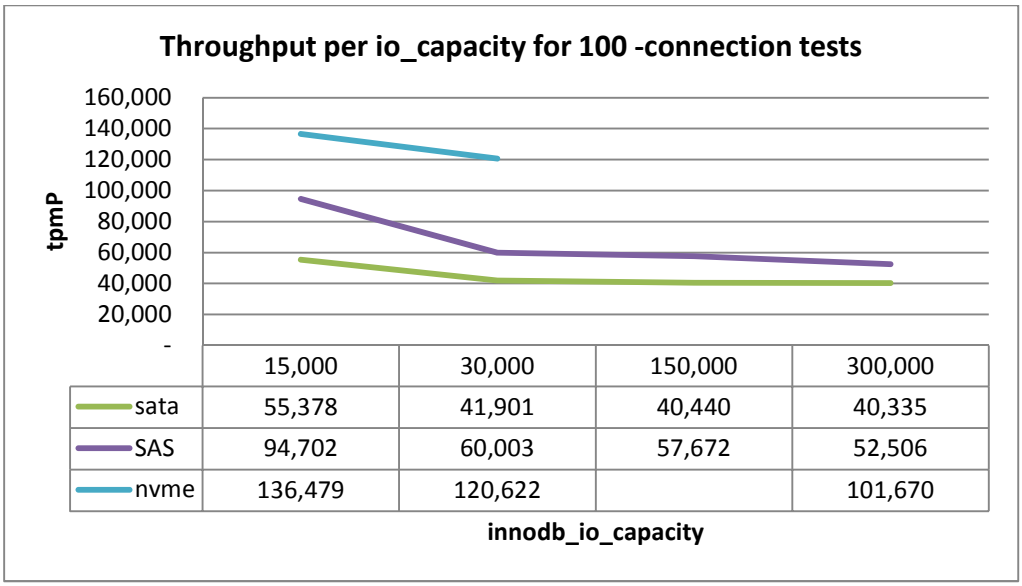


Figure 9. Dual-socket versus Quad-socket SATA-SSD CPU utilization



	50	100	150	200
sata	23,868	55,378	58,123	58,276
SAS	73,236	94,702	96,691	97,796
nvme	110,357	136,479	138,748	139,158

Figure 10. Quad-socket scalability with Config #A



	15,000	30,000	150,000	300,000
sata	55,378	41,901	40,440	40,335
SAS	94,702	60,003	57,672	52,506
nvme	136,479	120,622		101,670

Figure 11. Experimenting with Percona Server and innodb_io_capacity

4. Final Performance Evaluation

Using Configuration Table 10, Table 8, and Table 7, we have run tpcc-mysql on SAS HDD, SATA SSD, SAS SSD, and NVMe. Section 4.2 describes and analyses throughput results (tpmP = Qualified transactions per minute for the Percona server, similar to tpmC defined by the TPC-C Standard [1]). Section 4.3 describes and analyses results in terms of response times. In Section 4.4, we go over CPU and disk utilization for all storage types.

4.1. System and Test configuration

We ran tpcc-mysql with a 500-warehouse database. Transactions are executed on a Dell PowerEdge R930 Linux server, driven from another Dell PowerEdge R730xd Linux box. See Table 10 for their system configuration.

All reported measurements ran on either MySQL server 5.7.11 or Percona Server 5.7.11. Table 8 details the final Percona Server configuration used. This configuration is the result of the combined expertise of others, as well as experiments described in Section 3. A significant part of this measurement effort was spent on carefully calibrating the settings Table 8 presents.

The following parameters are especially important for NVMe:

- **innodb_io_capacity**. The default (200) is meant to throttle disk I/O on slower devices. In order for MySQL to benefit from faster devices, one must increase this parameter. We used 300,000 for the MySQL Server and 15,000 for the Percona Server.

- **innodb_buffer_pool_size**. This must be carefully balanced. If O_DIRECT is used, then we will need more buffer_pool and have used 12GB. When the default flush_method is used, then the ideal MySQL Server buffer_pool_size for our configuration is 3GB.
- **innodb_flush_method**. Using O_DIRECT, this means that we use innodb_buffer_pool to directly store I/O buffers, to bypass the filesystem cache. The faster the device, the more benefit we see from O_DIRECT.

The following parameters are especially important for TPCC:

- **innodb_thread_concurrency**. We tried different values, but found the best performance with the default 0 (unlimited thread_concurrency).
- **innodb_adaptive_hash_index**. We turned this 'OFF', since OLTP workloads typically do not reuse data from previous queries.
- **innodb_fill_factor**. 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.
- If possible, separate log_dir and datadir. All storage types benefit from this. For both Percona and MySQL Servers, it means setting up the parameters from Table 8 marked with either **<data storage>** or **<log storage>** to the appropriate storage location.

4.2. tpcc-mysql throughput

Figure 10 and Table 1 present throughput results for all storage types. Observe that SATA's result is 32+ times better than with the HDD, and that NVMe's throughput is 2 to 5 times better than SATA's, and up to 51% better than that of SAS.

Table 1 Throughput in tpmP

Connections	SAS-HDD	SATA-SSD	SAS-SSD	NVMe	HDD-> SATA	SATA -> SAS	SAS-> NVMe	SATA -> NVMe	HDD -> NVMe
50	741.99	23,868	73,236	110,357	32.17	3.07	1.51	4.62	148.73
100	756.73	55,378	94,702	136,479	73.18	1.71	1.44	2.46	180.35
150	1,089.78	58,123	96,691	138,748	53.34	1.66	1.43	2.39	127.32
200	N A	58,276	97,796	139,158		1.68	1.42	2.39	

The best tpcc-mysql throughput was obtained with NVMe storage using 200 connections (Table 1). That result, though, is only 2% better than the 100 connection case: we reached the knee of the curve with 100 connections. The 100-connection result, 136K tpmP, is 180+ times better

than with the HDD. Figure 12 shows *New Order* transactions executed over time for all four storage types for the 100-connection case.

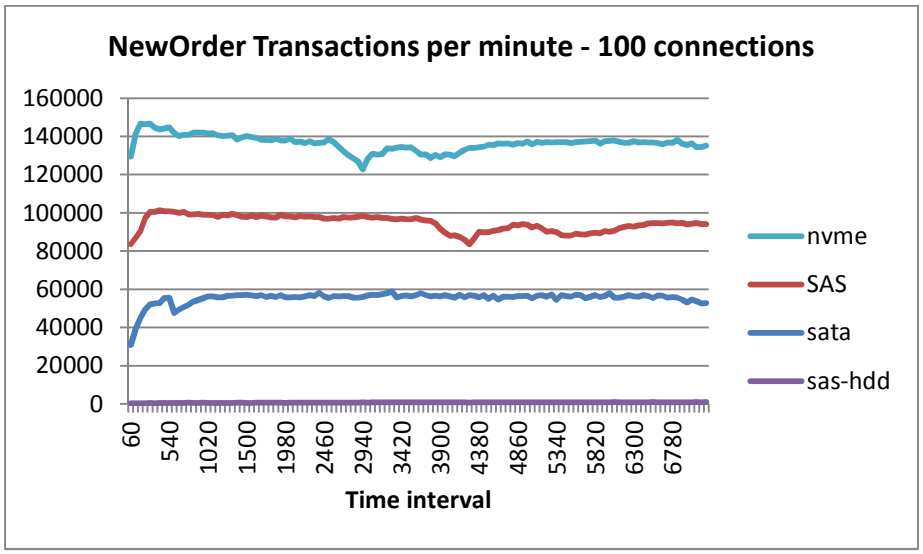


Figure 12. New Order transactions over time

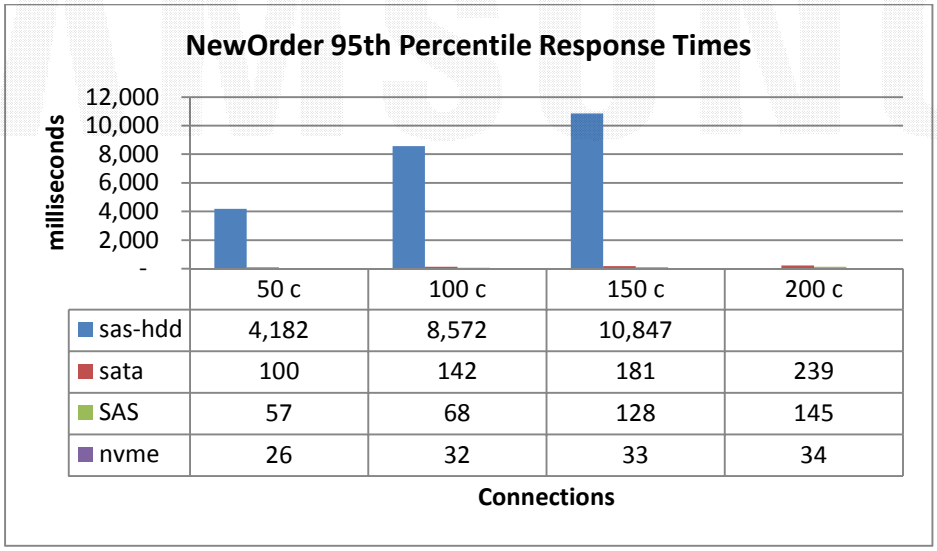


Figure 13. New Order 95th percentile response times

4.3. *tpcc-mysql response times*

OLTP application latencies (response times) are also very important. So we measured response time for each device. Figure 13 compares *New Order* transaction 95th percentile response times

for all four storage types. SATA-SSD response times are about 5 times higher (slower) than NVMe storage. HDD response times are up to 280+ times slower than the NVMe.

Figure 14 displays maximum response times for *New Order* transactions according to the number of connections for all four storage types. In this case, compared to NVMe, SATA-SSD can be 2 to 3 times slower, SAS-SSD up to 65% slower, and HDD 100+ times slower. All maximum SSD response times are much quicker than the 95th percentile HDD response times.

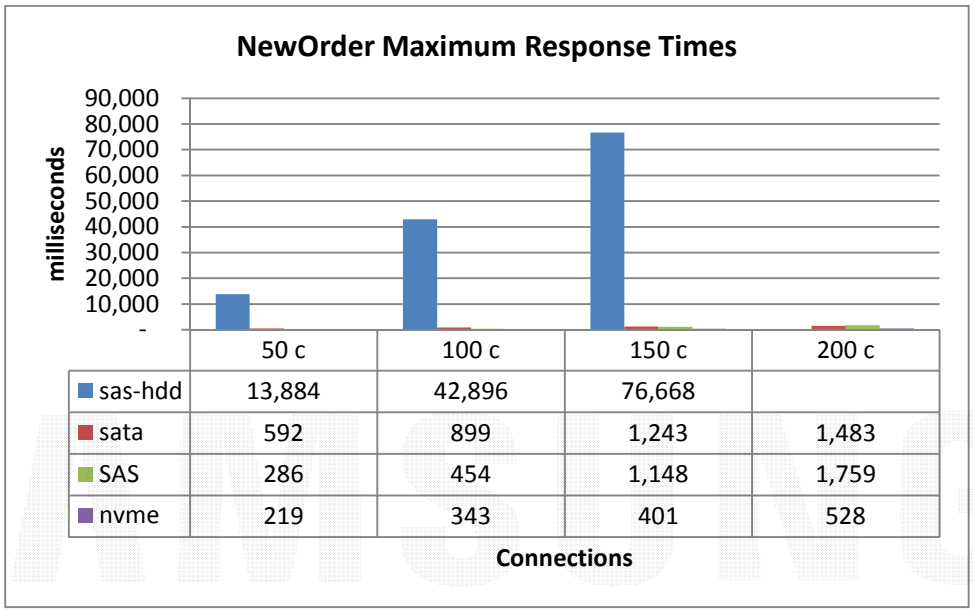


Figure 14. New Order maximum response times

Figure 15 show 95th percentile response times for *New Order* transactions over the measurement duration. Figure 15 compares a SATA SSD versus a SAS SSD versus an NVM storage. Observe that NVMe devices are consistently faster than other devices. NVMe's latency is about one fourth that of SATA-SSD and one half of an SAS-SSD.

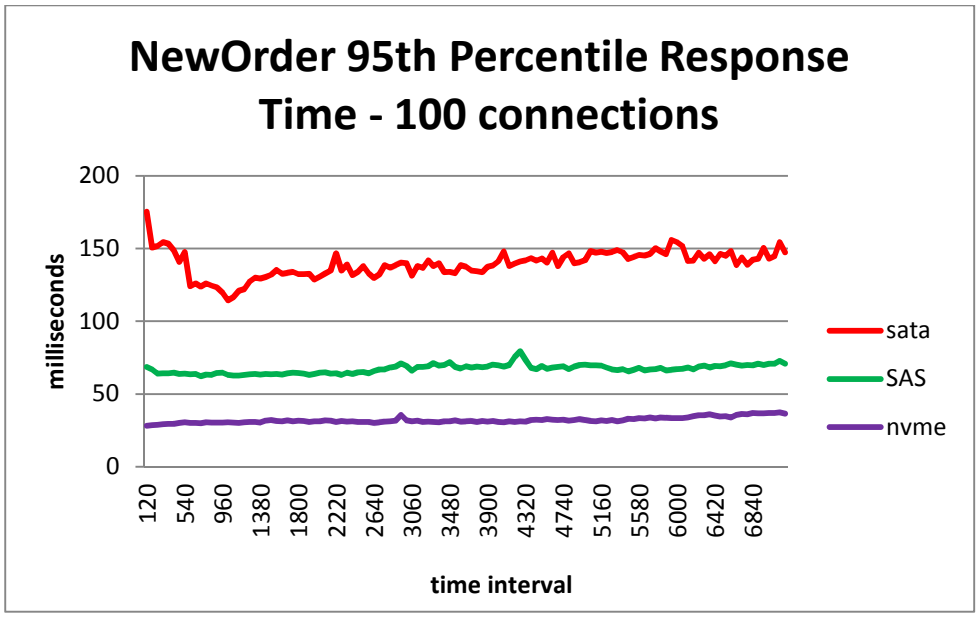


Figure 15. SATA-SSD, SAS-SSD, and NVMe latencies

4.4. *tpcc-mysql system resource utilization*

This section compares CPU utilization and disk utilization for all storage types.

It is generally accepted that OLTP database applications are I/O bound. They typically do not exhaust CPU capacity. While this is true for HDDs, we see a paradigm shift with NVMe devices. Table 2 shows tpcc-mysql mean CPU utilization running different loads on different devices. With HDDs, CPUs are always less than 1% busy, while with NVMe, CPU utilization goes up to 30+% with 100+ connections. This 30 times CPU increase translates into 110+ times more transactions per minute (tpmP), making the case for better server resource utilization when using NVMe devices.

Table 2. Mean CPU (User+Sys) percentage utilization

Mean CPU %	SAS-HDD	SATA-SSD	SAS-SSD	NVMe
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%

Another way to gauge the benefits of faster storage in terms of CPU consumption is to normalize it per transaction. This approach is known as CPU path length. CPU path length is the average number of CPU cycles it takes to complete one transaction:

*CPU PATHL = (CPU frequency * cores * total average CPU utilization) / (transactions per second)*

It is a measure of how much work CPUs need to do to execute a single transaction. Because our workload and our software are constant, CPU PATHL variations gauge the extra, bookkeeping work that needs to be done by the Server to manage queues, buffers, context switches, etc. Looking at Figure 16 and Figure 17, we notice that CPU PATHLs for faster devices are much smaller than the CPU PATHL for HDDs. In particular, the CPU PATHL for NVMe tests is 50 to 60% smaller than the equivalent HDD CPU PATHL.

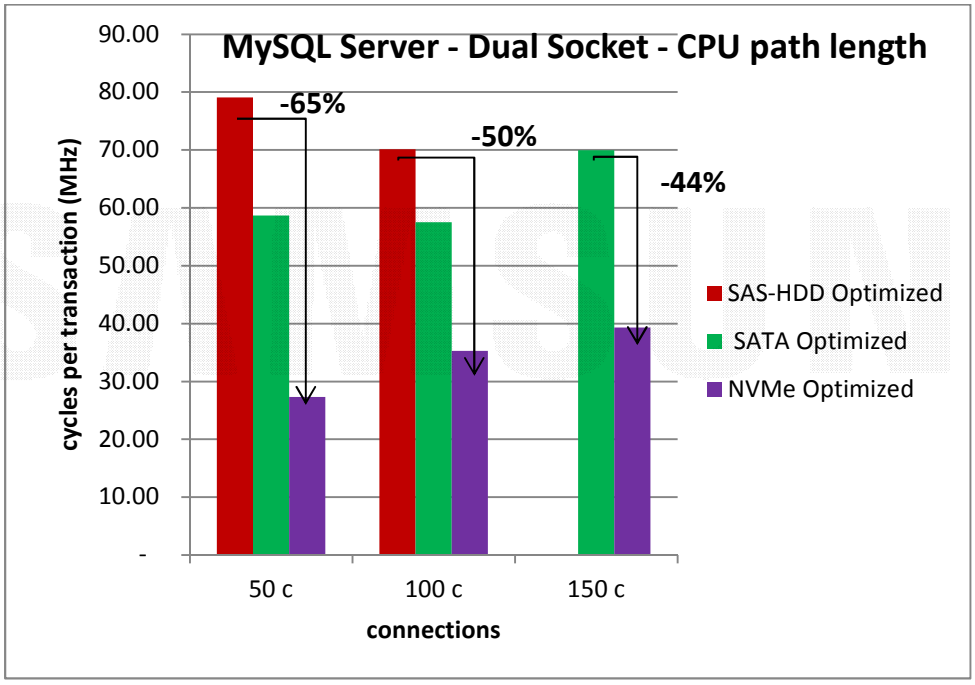


Figure 16. CPU path length for MySQL Server Optimized Results

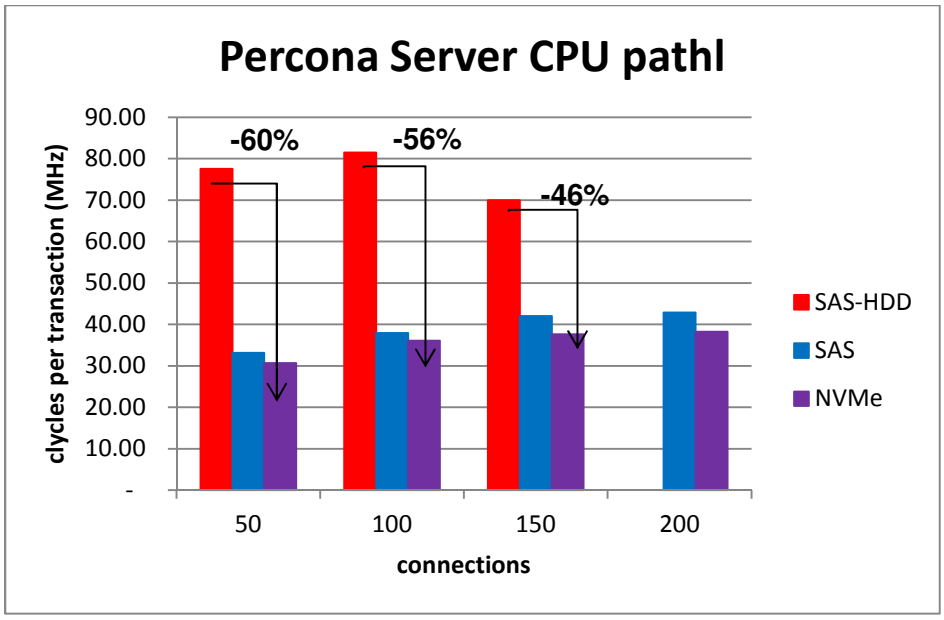


Figure 17. CPU path length for Percona Server optimized results

In Table 3 we see mean CPU I/O waits for all device types. Here the situation between HDDs and NVMe devices is reversed. Notice that systems using NVMe devices waste less than 4% of their time waiting on I/O, while a system using HDDs may spend more than 20% of its time waiting on I/O. From Table 3, we also see that I/O wait times for SATA-SSD and SAS-SSD are somewhat close. In Table 4, we list each SSD's specification from Samsung's website [14]. Even considering that the data on consumer devices (SATA-SSD) is at peak performance and enterprise (SAS-SSD and NVMe) is sustaining, we see that SATA-SSD is expected to perform better than SAS-SSD for random writes. Note that OLTP applications perform many random writes.

Table 3. Mean CPU I/O wait percentages

Mean CPU Wait %	SAS-HDD	SATA-SSD	SAS-SSD	NVMe
50 connections	13.2%	3.8%	8.4%	3.0%
100 connections	17.7%	14.1%	12.8%	3.5%
150 connections	22.2%	17.9%	13.8%	3.0%
200 connections	N A	14.7%	14.7%	3.5%

Table 4. SSD's used in this experiment and their specifications [14]

		Sequential Reads	Sequential Writes	Random Reads	Random Writes	price	size
SATA-SSD	850Pro	550MB/s	520MB/s	100K IOPS	90K IOPS	$215	512GB
SAS-SSD	PM1633	1,350MB/s	750MB/s	190K IOPS	30K IOPS	$702	960GB

NVMe	PM1725	6,000MB/s	2,000MB/s	1 M IOPS	120K IOPS	$1,881	1.5T

Figure 18 shows Read IOPS, and Figure 19 shows Write IOPS. They complete the story: because NVMe can deliver two orders of magnitude more IOPS than an HDD, for both Reads and Writes, they are able to keep CPUs two orders of magnitude busier (Table 2), yielding two orders of magnitude more transactions per minute (tpmP). We verified a similar story for both SATA-SSD and SAS-SSD.

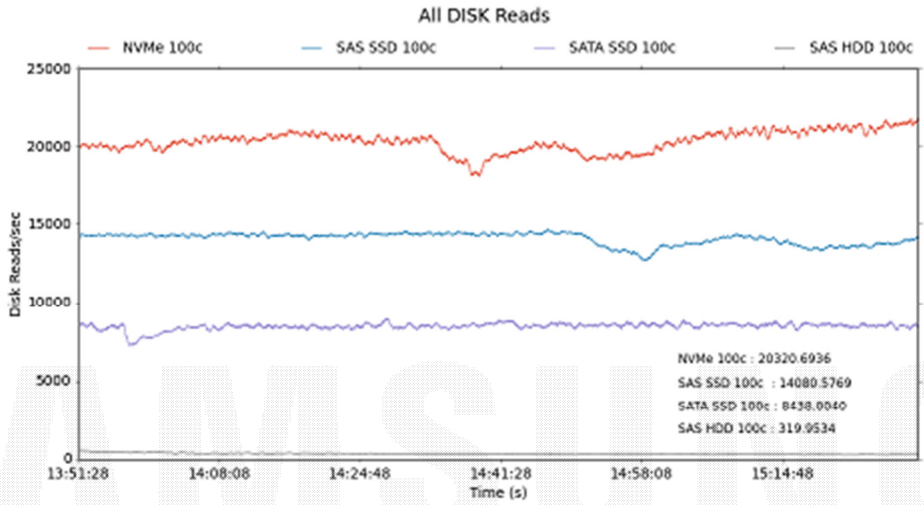


Figure 18. All Disk Reads at 100 connections

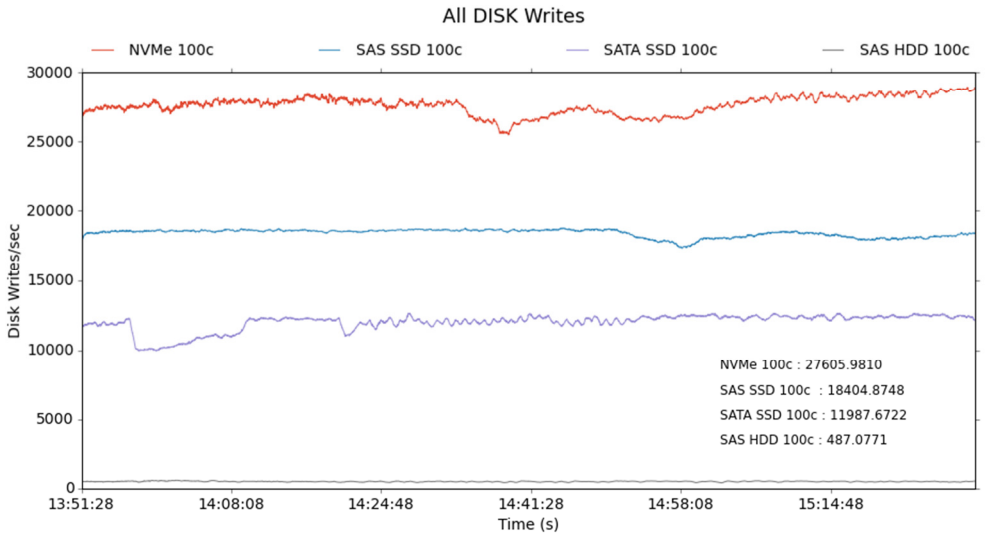


Figure 19. All Disk Writes at 100 connections

5. Key Takeaways

- With HDDs, CPUs are always less than 1% busy, while with NVMe CPU utilization goes up to 30+% with 100+ connections. This 30 times CPU increase translates into 110+ times more transactions per minute (tpmP), making the case for **much better server resource utilization** when using NVMe storage devices.
- OLTP database applications are I/O bound when using HDDs. They are **not** I/O bound when using NVMe devices. This is a paradigm shift for OLTP applications.
- All maximum SSD response times are **much smaller** than the 95th percentile HDD response times.
- NVMe throughput (tpmP) can be up to 180+ times better than that of an HDD.
- NVMe throughput (tpmP) is 2-5 times better than that of SATA-SSDs, and up to 51% better than SAS-SSDs.
- On configuration tuning for SSDs: the parameters listed are very important. If any of the numbers quoted do not seem ideal for your environment, we recommend that you do a little experimentation for possible fine-tuning.

6. Conclusions

NVMe storage can be up to 5 times more performant than SATA-SSDs, and 180 times more performant than HDDs (Table 1).

One may be able to add more SATA-SSDs or HDDs to a system to match the throughput of a high performance NVMe. However, it will not be possible to achieve the response times observed with NVMe storage when using the other storage types, due to their inherent latencies. We saw, for example, that the maximum response time for NVMe storage can be one tenth of the HDD 95th percentile response times (Figure 13).

With a faster storage device, we have the added benefit of more server capacity in terms of CPU cost per transaction (path length). That is, the server will deliver more transactions per CPU cycle. With NVMe storage, we can get considerably more work done, to the otherwise same hardware, beyond the SATA and SAS SSDs saturation point, and way beyond the HDD saturation point.

Appreciating the substantive additional server capacity available with the switch to SSDs, our next step will be to scale out. That is, run multiple MySQL instances on the same hardware (server and SSD storage) to evaluate how much more performance our configuration may yield for OLTP workloads.

7. References

[1] http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-c_v5.11.0.pdf Last accessed 02/24/2016.

[2] https://github.com/Percona-Lab/tpcc-mysql Last accessed 02/17/2016.

[3] Akin, W. "Understanding NAND's intrinsic characteristics critical role in Solid State Drive (SSD) design" IEEE International Memory Workshop 2015.

[4] https://en.wikipedia.org/wiki/MySQL Last accessed 4/1/2016.

[5] https://www.percona.com/software/mysql-database/percona-server Last accessed 4/1/2016.

[6] http://sharepoint.ssi.samsung.com/labs/MSL/DCP/SitePages/tpcc.aspx Last accessed 10/29/2015.

[7] https://www.percona.com/files/presentations/percona-live/nyc-2011/PerconaLiveNYC2011-Optimizing-MySQL-for-Solid-State-Storage.pdf Last accessed 10/29/2015.

[8] https://www.percona.com/about-us/mysql-white-paper/scaling-mysql-with-virident-flash-drives-and-multiple-instances-of-percona-server Last accessed 10/29/2015.

[9]http://cdn.oreillystatic.com/en/assets/1/event/56/Linux%20and%20H_W%20optimizations%20for%20MySQL%20Presentation.pdf Last accessed 10/30/2015.

[10] Burton, Kevin. "*Final Thoughts on SSD and MySQL AKA Battleship Spinn3r*" https://burtonator.wordpress.com/2008/02/22/final-thoughts-on-ssd-and-mysql-aka-battleship-spinn3r/ Last accessed 11/02/2015.

[11] MySQL Performance Tuning and Optimization Resources https://www.mysql.com/why-mysql/performance/ Last accessed 11/02/2015.

[13] Tkachenko, Vadim. *E-mail communications*. March 2016.

[14] http://www.samsung.com/semiconductor/products/flash-storage/ Last accessed 04/13/2016.

Appendix A: MySQL and Percona configurations

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

Table 5 Summary of MySQL parameters mentioned in the literature

	MySQL 5.7 default	Siyawala[6]	Percona[7]	Matsunobu[8]	Burton[9]
innodb_buffer_pool_size	(128MB)	(2GB,16GB,40GB)		(1G, 2G, 5G, 30G)	
innodb_doublewrite	ON		?		
innodb_flush_neighbors	1	0	(ON,OFF)		
innodb_io_capacity	200	(200, 100000)			
innodb_log_file_size	(48MB)	(5MB, 3GB)	> 4GB		
innodb_log_files_in_group	2	(2, 3)			
innodb_page_size	(16K)		(4K, 8K, 16K)	(4k, 8k)	8k
innodb_thread_concurrency	0	(8,20,32,50)			
innodb_flush_log_at_trx_commit	1			(1,2)	
innodb_flush_method				O_DIRECT	
read_ahead_buffer	?	disable			disable
innodb_log_block_size	?		(512, 4096)		
innodb_adaptive_checkpoint	?		keep_average		

Table 6 Early testing configurations - MySQL and Percona parameters.

Parameter Name	Config #1	Config #2	Config #3	Config #6 / #A
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/ibtmp1:72M: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'			
## The following settings are sugested by Percona for TPC-C:				
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 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_max_purge_lag_delay	30000000			0
# Monitoring				
innodb_monitor_enable	'%'			<empty>
performance_schema	OFF			

Table 7 Subset of parameters unique to Percona Server

default	Config#6 and Config#A
innodb_cleaner_lsn_age_factor = high_checkpoint	
innodb_corrupt_table_action = assert	
innodb_empty_free_list_algorithm=backoff	
innodb_kill_idle_transaction = 0	
innodb_max_bitmap_file_size = 104857600	
innodb_max_changed_pages = 1000000	
innodb_parallel_doublewrite_path='xb_doublewrite'	/<log_storage>/mysql_log/xb_doublewrite'
innodb_show_locks_held = 10	
innodb_show_verbose_locks = 0	
innodb_track_changed_pages = OFF	
innodb_use_global_flush_log_at_trx_commit = ON	
max_binlog_files = 0	
max_slowlog_files = 0	
max_slowlog_size = 0	
query_cache_strip_comments = OFF	

Table 8 MySQL Server and Percona Server optimal configurations

Parameter Name	MySQL (Config #3)	Percona (Config#A)
datadir	/<data storage>/mysql_data/mysql	
innodb_log_group_home_dir	/<log storage>/mysql_log	
innodb_undo_directory	/<log storage>/mysql_log	
innodb_buffer_pool_size	12GB	
innodb_thread_concurrency	0	
innodb_temp_data_file_path	'../../../<log storage>/mysql_log/ibtmp1:72M:autoextend'	
innodb_page_cleaners	32	8
innodb_buffer_pool_instances	32	8
innodb_io_capacity	300000	15000
innodb_adaptive_hash_index	'OFF'	
innodb_fill_factor	50	100
innodb_write_io_threads	16	
innodb_read_io_threads = 16	16	
innodb_flush_method	'O_DIRECT'	
innodb_flush_neighbors	1	0
query_cache_size	0	

log_timestamps	'SYSTEM'	
## The following settings are sugested by Percona for TPC-C:		
table_open_cache	8000	
table_open_cache_instances	16	
back_log=	1500	
max_connections	4000	
## To use many connections in TPC-C:		
max_prepared_stmt_count	64000	
# files		
innodb_log_files_in_group	3	
innodb_log_file_size	1G	
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_max_purge_lag_delay	30000000	0
# Monitoring		
innodb_monitor_enable	'%'	<empty>
performance_schema	OFF	

Appendix B: Software and Hardware used.

This appendix contains the two hardware and software setups discussed in this report.

Table 9 Early testing (dual-core) hardware configuration

	Dell PowerEdge R730xd (Server)	Dell PowerEdge R730xd (Driver)
CPU		
Model Name	Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz	Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz
Cores	12	8
Processors	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		
HDD	SEAGATE ST600MP0005 15K rpm	
SATA	Samsung 850 PRO	
NVMe	Samsung XS1715	

Table 10 Quad-core Configuration

	Dell PowerEdge R930 (Server)	Dell PowerEdge R730xd (Driver)
CPU		
Model Name	Intel(R) Xeon(R) CPU E7-4850 v3 @ 2.20GHz	Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz
Cores	28	8
Processors	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	SEAGATE ST600MP0005 15K rpm	
SATA SSD	Samsung 850 Pro	
SAS SSD	Samsung PM1633	
NVMe	Samsung PM1725	

###