WHITE PAPER

Scale Up and Out with Oracle MySQL and SanDisk SSDs

September 2014

SanDisk
a Western Digital brand

Western Digital Technologies, Inc.
951 SanDisk Drive, Milpitas, CA 95035

www.SanDisk.com

Table of Contents

Executive Summary

Oracle's MySQL is a highly popular, widely used open-source relational database . It is commonly used in On Line Transaction Processing (OLTP) applications serving hundreds of users, while delivering thousands of transactions per second . A very important aspect of OLTP system design is to build in the ability to fully utilize available resources—or to add more resources to match a growing load .

In the case of very large system implementations that will require scaling-out, it is very important that the fundamental system building block design fully utilize the computational capability of the individual compute node . This approach will greatly reduce the number of server nodes per cluster (total cluster size) . Accordingly, it will reduce the system cost when scaling-out to achieve performance goals—or to meet growing user demands in the future .

This paper reviews testing results that demonstrate the benefits of using a combination scale-up and scale-out strategy . This approach more fully utilizes the building-block computing resources than by simply adding more computational elements in a scale-out strategy .

Applications such as OLTP workloads require low latency and high IOPS performance to avoid becoming I/O-bound . This is especially the case when attempting scale-up strategies with multiple application instances . Utilizing fast, low latency SSD based storage, such as SanDisk Optimus SAS drives, is a key enabler for scaling-up system performance .

Introduction to SanDisk Optimus Drives

SanDisk is a leader in flash storage solutions offering solid-state drives (SSDs) for existing applications—and for emerging applications such as Cloud Storage, Big Data Analytics and RDBMS (Relational Data Base Management System) applications .

SanDisk Optimus™ SAS SSDs use the Guardian Technology™ Platform, a suite of enterprise class features and write endurance enhancements that make cost-effective MLC (Multi-Level Cell) NAND fully usable for demanding enterprise applications and database workloads . Guardian Technology consists of three major elements:

- FlashGuard™ uses aggregated flash management and advanced signal processing combined with a mix of flash grades to ensure longer usable product life from MLC NAND than other flash products .
- DataGuard™ provides full end-to-end data-path protection (T10 DIF support), ensuring that the data will be safe throughout the entire data path, providing the ability to recover data for failed page and NAND blocks .
- EverGuard™ prevents the loss and corruption of user data in the event of an unexpected power interruption .

SanDisk's vertical integration and ownership of the flash intellectual property (IP) ensures that the Guardian Technology Platform and MLC NAND media work seamlessly together to optimize write endurance, performance, and reliability . Optimus SSDs provide best-in-class performance with a consistent quality of service and with flexibility in storage capacities, ranging from 200GB to 3 .84TB . The MTBF (Mean Time Between Failures) for Optimus SSDs is 2 .5 Million Hours, with a warranty period of five years .

Scale-Up Computing Model

The availability of powerful, inexpensive multi-core processors has enabled the use of the scale-up computing model . This is especially true when supporting multiple instances of applications or virtual machines (VMs) that can run on the same server system without introducing the complexity and software overhead of clustered file systems (CFS) and the introduction of network-related latency .

How well the scale-up approaches work depends upon the performance requirements of the hosted applications and computational capability of the host system .

One obvious example of an easy and transparent scale-up strategy is improving the storage system performance by replacing spinning hard disk drives (HDDs) with solid state drives (SSDs) . Our tests have shown that SSDs can improve peak IOPS performance by a factor of 100, and can improve the average I/O latency by a factor of 30 or more, depending on the number of HDDs drives that are replaced . Some additional examples of scaling-up a server system include: adding more memory, using faster processors, and adding processors that have more cores .

Scale-Out Computing Model

One common method for scaling out systems is to add more processing power in the form of more servers that are linked in a cluster configuration . These additional servers can be physical servers, or virtual servers (VMs)—or a combination of both . Clustered systems, sometimes referred to as distributed computing systems, usually have global name space file system support, and they often have robust high availability (HA) fail-over capabilities .

Because the nodes in a cluster must tightly communicate with each other for database operations, the cluster-aware software introduces a source of latency that limits performance scaling efficiency as additional server nodes are added to the cluster1 . For this reason, customers must seek to reduce the latency that is built into their scale-out clusters .

MySQL Overview

MySQL is an open source relational database, but it is also a database management system, which is a requirement for adding, accessing, and processing data stored in a database . Databases are used to efficiently organize data at a higher level than simple tables, eliminate redundant records and enforce relationships defining how the database will function . MySQL consists of two major modules: MySQL Server and MySQL Client, both of which are necessary to fully utilize and interact with the database functions .

OLTP Simulation with Multiple Database Instances

The test scripts that form the basis for the data presented in this paper create eight MySQL database instances with the purpose of simulating a typical single server computing environment . In this environment, each MySQL client has exclusive use of a database instance: that is, there is a 1:1 correspondence of database “clients” and “servers .” The storage workload is generated using sysbench scripts that initiate database queries to drive the default InnoDB storage engines in the MySQL servers . In this way, the test simulates large numbers of users launching transactions on each of the database instances .

Performance Measurements

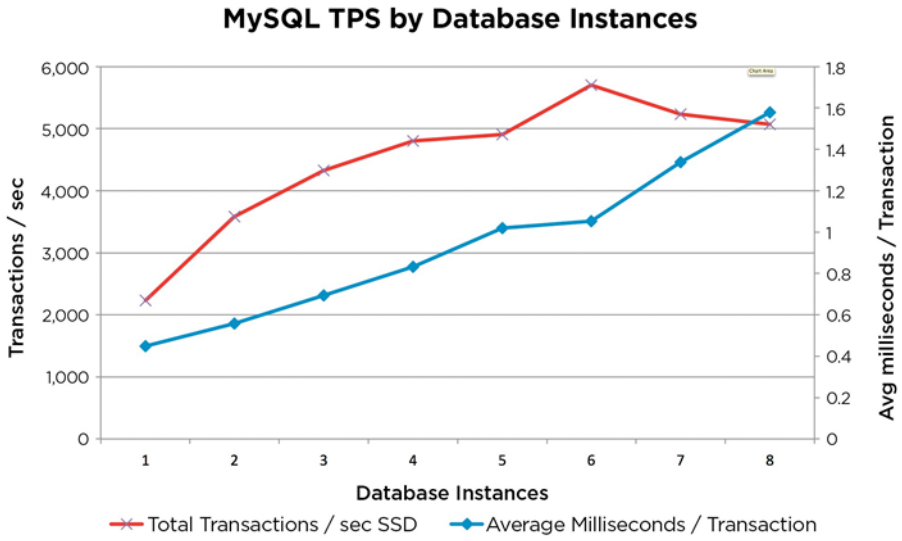

Figure 1: TPS and Latency by Number of Database Instances

Figure 1, above, shows the transactions per second (tps) that were achieved as the workload scaled from one MySQL instance to eight simultaneously running instances . The database buffer pool size was limited to a total of 1GB of DRAM for each of the eight instances . Note that the total number of transactions per second steadily increased from one to six instances, peaking at 5,702 and then tailing off for the instances numbered seven and eight . This established that the peak number of transactions per second was achieved with six simultaneous MySQL instances .

The response time, as measured in the average number of milliseconds (msec) per transaction (shown on the right axis of Figure 1) increased by a factor of 4—as the databases increased from one instance to eight instances . However, the average response time for query completion was still a very low

1 .6 msec in the case of the eight database instances . Note also that the shape of the transactions per second plot is not a straight line, showing that there are definite limitations when adding more database instances to achieve more overall performance .

CPU Utilization

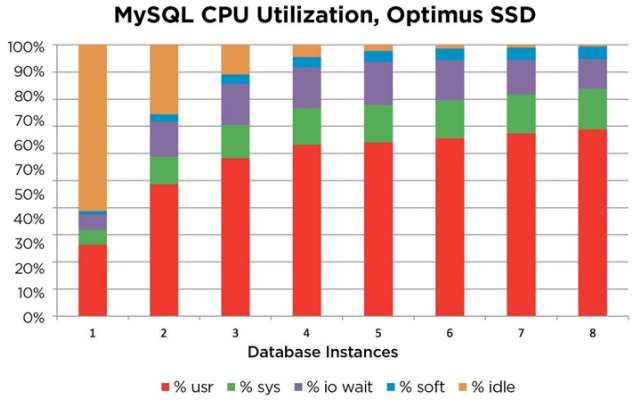

Figure 2: MySQL CPU Utilization, Optimus SSDs

CPU utilization measurement can provide insight into how the system is behaving and provide clues to what changes could be made to improve performance . Figure 2 charts the data collected using mpstat while running the multiple MySQL database instances test . The mpstat data shows how the

CPU utilization varies as the number of MySQL instances increase from one to eight instances . The CPU utilization is segmented into five major categories, as shown in the Figure 2 legend . The “usr” category represents primarily the database application’s use of the available CPU resources, while the “idle” category shows the percent of CPU that is available for other purposes .

The overall trend shown by the CPU utilization data is that the application quickly becomes CPU-bound, with the idle time rising to approximately 2%, by the time the fifth MySQL instance is started . In addition, the io wait and sys categories (I/O driver and operating-system overhead) stayed relatively constant for the instances numbered three and beyond, indicating that providing more CPU resources in the form of higher clock speeds—or more CPU cores—has the potential to improve application performance .

Solid-state drives (SSDs) clearly improved performance . Without the scale-up effect contributed by the high performance SSD-based storage, I/O wait time would have been greatly extended, meaning that the CPUs are idle, waiting more for I/O completion . High idle time translates to getting less meaningful work done, and therefore to fewer database transactions per second .

The test results for multiple database instances show that this approach can significantly improve performance, in terms of transactions per second, by a factor of more than 2 .5:1 . The strategy will reduce the cost per database transaction—and it will also improve server utilization . If the application can be parallelized with multiple simultaneous available instances, then co-locating (on a single server) multiple MySQL server instances using fast storage, such as SanDisk Optimus SAS SSDs, provides a good balance between cost, complexity, and performance on today’s x86 industry-standard server hardware .

Scale It Out

The term “hyperscale” refers to an environment where many instances of applications or virtual machines can run simultaneously on the same server—or on multiple server systems . This general approach is used by cloud service providers (CSPs), enabling them to scale their services as increasing demand is placed on their computing infrastructure . This “hyperscale” environment is really nothing new . It’s an approach that combines the “scale-out” and “scale-up” approaches to add compute, memory, networking, or storage resources to a node, or a set of nodes, in order to improve performance .

Once the scale-up configuration for a single server has been optimized, additional performance can be realized with a scale-out approach . Scaling-out at this point has a multiplying effect on performance as more nodes are added in a clustered configuration .

To illustrate the principle behind scaling-out in a cluster configuration, consider the following example:

- Assume that a single server with six MySQL instances, as established by the performance measurements, achieves a maximum of 5,700 transactions per second .
- The business organization using this server for its database needs has acquired new customers with many users—and must improve response time to handle the increased load .
- Careful analysis shows that, to meet quality of service (QoS) goals, the database system must provide at least 17,000 transactions per second (tps) .
- Now, assume that due to network latency and clustered file system overhead, it is estimated that multiple servers should be able to scale at approximately 90% of the performance per additional server .

Based on these assumptions, it would require at least 17,000/(5,700 x .90) = 3 .31 (rounding to 4) servers needed to match the requirement of 17,000 transactions per second when using six MySQL instances per server . Applying the same calculation to a server hosting a single MySQL instance, at least 17,000/(2,231 x .90) = 8 .5 (rounding to 9) servers would be needed to achieve the 17,000 transactions per second requirement .

Based on these assumptions—and based on the desire to minimize the cost of improving the performance to meet the business goals—the following approach would deliver the best results: The best alternative would be to generate six MySQL instances per server, and to have four of these servers in a clustered configuration . The combined strategies of scaling-up the individual server node first— and then scaling-out with multiple servers—results in a cost reduction of more than 50% . This cost reduction can be achieved, while realizing the database performance that is being demanded by the business need, with four servers versus nine servers .

A much more detailed discussion on parallel computing and the scaling of computing systems, including both scaling-up and scaling-out technologies, can be found under the subjects of Amdahl's Law[1] and Gustafson's Law[2] in the References section and the footnotes of this paper .

Database Sharding

An additional topic related to scaling-out databases, not covered by this paper, is database sharding . Sharding is similar to the use of multiple database instances, but it is sufficiently different such that it requires a brief explanation of the concept . Large databases and high performance applications can exceed the capability of a single server . Sharding is another example of scale-out—and it works by dividing up the data set and distributing the pieces over multiple systems (shards) . Collectively, the shards make up a single logical database[3].

Conclusion

Meeting the quality of service (QoS) needs of a dynamic and growing business is a challenging exercise for IT professionals . This paper presents evidence that architectural choices, such as software parallelism, scale-up, and scale-out approaches can have a large impact on application performance .

Some simple guidelines for optimizing application performance results, based on our testing of SanDisk Optimus SAS SSDs, are as follows:

1. First, scale-up the building-block server (faster storage, more memory, more CPU cores, higher clock rate processors, etc .), before scaling-out with more servers
2. To take full advantage of multiple database instances, use fast SSD storage to provide a good balance between cost, complexity, and performance (scaling-up)
3. Then, once the performance requirements exceed what can be done with a single server, scale- out with multiple building-block servers .

[1] See Amdahl's law at: www.princeton .edu/~achaney/tmve/wiki100k/docs/Amdahl_s_law .html

[2] Reference to Gustafson's Law: www .spe-ed .com/papers/scale04 .pdf

[3] MySQL Sharding: Tools and Best Practices: www .slideshare .net/mkindahl/mysql-sharding-tools-and-best-practices-for-horizontal-scaling

4. Consider task parallelism where possible . Example: Sharding a single database into multiple parallel tasks (scaling-out)

As with any major business decision, careful requirements planning, testing and analysis should be done before committing to production volumes of specific hardware and software purchases . This white paper demonstrates the benefits of using a combination of scale-up and scale-out strategies to improve performance . The use of both approaches more fully utilizes building-block computing resources and reduces cost in the future as business requirements grow, demanding greater system performance .

Appendix A—Test Configuration Details

System/Environment Setup

A Dell PowerEdge 720 server with 16 available drive bays was utilized for this testing . An internal SATA boot drive was utilized for Operating System and MySQL executable files . A MySQL database was created on 8 SanDisk Optimus 400GB SSDs in a 100% redundant RAID10 configuration, with all of the database-related storage, including log files, residing on the RAID10 SSD volume .

Database Software

MySQL Server and Client version 5 .5 .25a, using InnoDB storage engine

Database Configuration

1. The total buffer pool size was purposely limited to 1GB per database instance to maximize stress on the storage system
2. Number of tables: 256 for each MySQL instance
3. Number of rows: 25,0000 for each MySQL instance
4. Test time was 600 seconds (10 minutes) for each set of MySQL instances

SysBench OLTP Simulation Command

```
$SYSBENCH \
--test=$SYSBENCH_TESTS/oltp.lua \
--oltp-tables-count=25 \
--mysql-host=127.0.0.1 \
--mysql-port=3501 \
--mysql-user=ADMIN \
--mysql-password="${PASSWORD}" \
--mysql-db="${DB_NAME}" \
--mysql-table-engine=innodb \
--num-threads=16 \
--max-requests=0 \
--max-time=600 \
--oltp-table-size=25000 \ run >>"${TMPOUTPUT}" 2>&1
```

Operating System

CentOS Linux 6 .5, kernel 2 .6 .32 .431 .5 .1 .el6 .x86_64

Server

Dell® PowerEdge R720, 2 x 8 core Intel® Xeon E5-2700 @ 2 .6 GHz processors, 256GB DRAM Memory

Server Optimization

Following are the server configuration settings:

1. NUMA-enabled .
2. Intel Turbo Boost-disabled
3. I/O Scheduler = noop (meaning none)

Storage

SSD Configuration, 8 Optimus™ SAS SSDs by SanDisk®, 400GB each, configured as a RAID10 volume with a total usable capacity of 1 .6TB .

Storage Topology

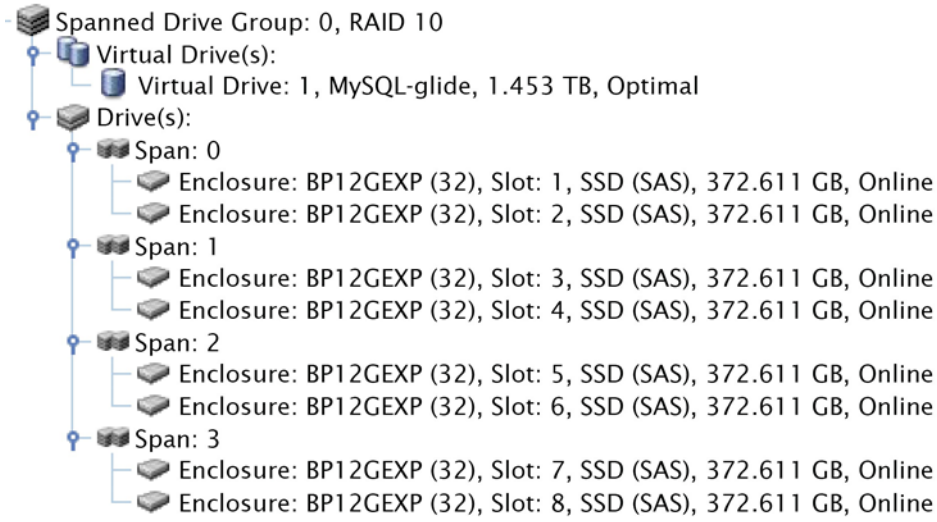

Figure 3: SSD Storage Configuration

Flash Memory Optimization

The following changes are done to optimize SSD performance:

Dell PERC H710P RAID Adapter was used to create and control the 8xSSD RAID10 configuration .

a. Write-back enabled: Write-back mode provides superior performance compared to write- through policy . Meanwhile, SanDisk Optimus SSDs have EverGuard power loss backup protection, providing data persistence in the event of an unexpected power loss .

b. Read-ahead enabled, improving sequential read performance .

c. The Dell R710 RAID controller has battery backed-up cache to prevent data loss due to an unexpected power loss .