

A Performance Evaluation of OLTP Workloads on Flash-Based SSDs

Rajesh Meena

Abstract—Flash-based Solid State Drives are high-performance data storage device. SSDs deliver high performance than the traditionally rotating storage hard drives. The main objective of this work is to evaluate the Solid State Drive (SSD) performance as a choice for database storage device. Initially SSD out performed in terms of transactions per second as compared to the local disk. After increasing the load, the throughput started flattening. Initial system monitoring has shown some interesting patterns. It was found that soft bottlenecks in the application were chocking the performance lift. By doing the code profiling we overcome with the soft bottlenecks. Later, we observed high performance on flash-based SSD in comparison to local disk.

Index Terms—Benchmark, evaluation, OLTP, SSD and SAS.

I. INTRODUCTION

IOPS (Input/Output Operations per Second) is a common performance measurement used to benchmark computer storage devices like local hard disk drives (HDD), solid state drives (SSD), and storage area networks (SAN). As with any benchmark, IOPS numbers published by storage device manufacturers do not guarantee real-world application performance.

Online transaction processing, or OLTP, is a class of information systems that facilitate and manage transaction-oriented applications, typically for data entry and retrieval transaction processing [1]. OLTP application involves good amount of both reads and writes.

Flash-based solid state disks are rapidly becoming a popular alternative to hard disk drives as permanent storage, because of flash's faster read access, low power consumption, small size, shock resistance and reliability compared to hard disks. SSDs are commercially available in numerous commodity PC models today; they are considered a high-end option due to a price-per-bit that is higher than that of HDDs, but that price gap is closing very quickly. The key benefit of SSD over HDD is a significant reduction in I/O latency for both read and write, while delivering higher IOPS, I/O bandwidth and maintaining low power consumption.

Abbreviated SSD, a solid state disk is a high-performance plug-and-play storage device that contains no moving parts.

SSD components include either DRAM or EEPROM memory boards, a memory bus board, a CPU, or a battery card. Because they contain their own CPUs to manage data storage, they are a lot faster than conventional rotating hard

disks; therefore, they produce highest possible I/O rates. SSDs are most effective for server applications and server systems, where I/O response time is crucial. In this paper we present a case study on performance of SSD storing DB data and accessed by an OLTP application. We have conducted our experiments using MySQL database server with Vehicle Insurance Application on commercial SSD device.

II. RELATED WORK

Cagdas Dirik and Bruce Jacob [2] tried to find few important points about SSD (a) the real limitation to NAND Flash memory performance is not its low per-device bandwidth but its internal core interface; (b) NAND Flash memory media transfer rates do not need to scale up to those of HDDs for good performance; (c) these system- and device-level concurrency mechanisms are, to a significant degree, orthogonal: that is, the performance increase due to one factor does not come at the expense of the other, as each exploits a different facet of concurrency exhibited within the PC workload.

David Bartizal & Thomas Northfield [3] in their whitepaper tried to explain the performance attributes of SSDs and provides a comparison to typical HDD performance. In addition the dependencies on application and some particular hardware are explored [3]. In another whitepaper Thomas Tanaka [3] evaluates Sybase Adaptive Server Enterprise (ASE) 15.5 performance with the inclusion of SSD as a choice for database storage devices.

III. APPLICATION USED FOR TESTING

The Vehicle Insurance (VINS) is a Web-based OLTP application used for benchmarking by Performance Engineering Research Centre (PERC), Mumbai. It has the functionality of maintaining vehicle insurance policies for retail customers. A policy owner (end user) can:-

- Query the policy details.
- Create a new policy for newly added vehicle.
- Renew expired policies.

The web based application is written using Java, and uses MySQL for database support.

The workload had the following composition.

- 1) Read Policy Details – 40%
- 2) New Policy Creation – 30%
- 3) Renew Expired Policy – 30%

In other words, a concurrent user test of 100 users would have 40 users browsing policy details, 30 users creating a new policy and the remaining 30 renew an expired policy. Scripts were run to initialize the database to initial state prior to each test.

In all the given test cases, we have deployed the VINS application and MySQL database on the same server.

IV. METHODOLOGY

The tests were designed in such a manner to evaluate the capability of the SSD [4] device in comparison to local SAS drive. An OLTP application was used to measure the performance comparison between the two. The same set of test cases was executed on both the storage devices. During the evaluation the application and database size was same. We had increased the workload till maximum throughput is reached or one of the physical resource becomes the first hardware bottleneck. Both application and database was deployed on the same server running CentOS Linux 6.

V. TEST ENVIRONMENT

VINS was load tested on Intel Westmere-EP Server; Application & Database was deployed on a server with 2 socket Quad core Intel Xeon E5620 @ 2.40GHz and 16GB memory. 300GB SAS drive was inbuilt and 400GB SSD was installed with the latest Firmware and RPM's. The underlying operating system was CentOS 6.1 x86-64. Load was generated using Grinder-3.2 on an 8CPU Intel server with 16GB of memory. The test was set up on a 1 Gbps network.

Fig. 1 shows the test setup during the evaluation. Application and Database was installed on the same server and SSD card was installed on the PCI-e slot. A load generator was also connected on the same switch 1 Gbps network.

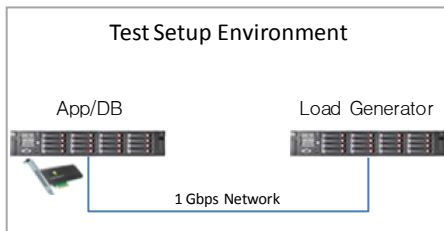


Fig. 1. Hardware setup for the evaluation.

Listed below are some of the conFig.uration details

- MySQL Server:- Version 5.1.66
- Flash-Based PCI-e SSD Drive : Capacity 400 GB
- Sever Hardware:- HP DL380 G7, 2 Socket, Quad core, 16 Gb RAM, 1 Gbps network
- SAS Disk:- 10K RPM, 300GB

The total size of database was 50 GB with approximately 20 million customers in the database.

VI. EVALUATION

VINS was load tested with different concurrent user loads ranging from 5 to 200, with the database on SSD in one conFig.uration and SAS disk on the other. Henceforth the tests will be referred as “SSD” and “SAS” tests. The duration for concurrent user load test was 5 minutes in each run. The throughput and response times reached steady state values in the period for all the tests. Each test was repeated twice to confirm observations. The think time for each page

was zero in all the test cases. We had measured the numbers for SAS disk up to 500 user load, since it becomes the first hardware bottleneck.

Fig. 2 shows that when using the application with SSD, throughput is significantly better with compared to SAS. The SSD was 20% utilized at 2000 user load, it can give better throughput with more number of users. On the other hand, SAS was 70% utilized at 300 user load. We have seen a decrease in the throughput after a certain limit of concurrent users in the system. In contrast to that the CPU utilization of the system was increasing linearly.

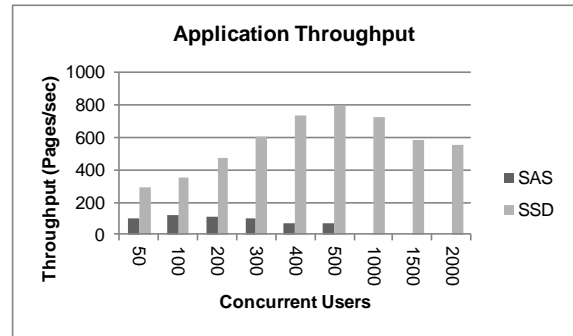


Fig. 2. Throughput comparison between SAS and SSD.

Fig. 3 shows the response time comparison between the SSD and SAS conFig.uration.

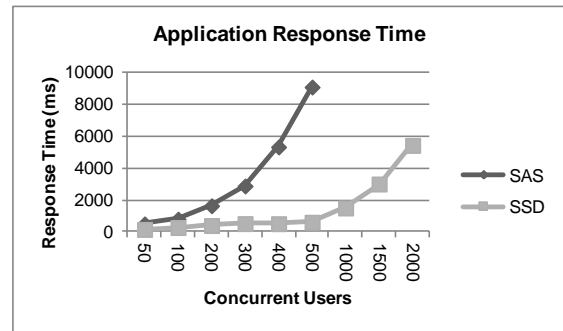


Fig. 3. Response time comparison between SAS and SSD.

From Fig. 4 it can be observed that the linear increase in CPU utilization is there, but throughput is not showing such behavior. This came with some interest to us, why the CPU utilization is kept on increasing when the throughput is showing a decrease in performance. The same behavior is seen for both PCIe SSD and local SAS drives.

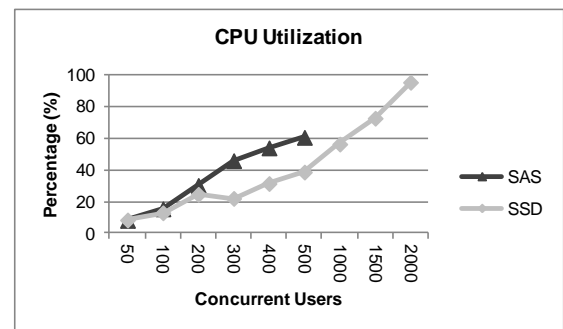


Fig. 4. Comparison of CPU Utilization between SAS and SSD.

The throughput start to decrease after 500 user load for SSD, but the CPU utilization is still increasing linearly with the added load.

From Fig. 5 it is clear that local SAS disk becomes

bottleneck at 300 user load. If we see the CPU utilization graph, system utilization in the case of local disk was increasing linearly. While on the other hand, utilization numbers for SSD is quite low as compare to local disk. The utilization behavior cannot justify clearly the decrease in throughput after 500 user load.

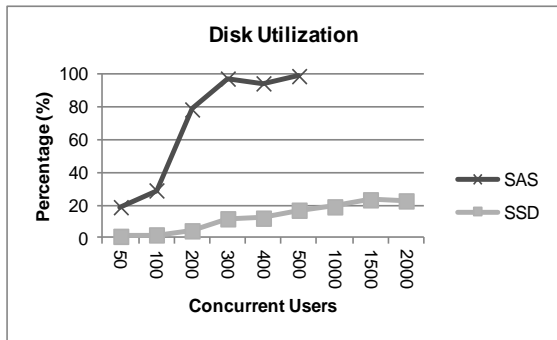


Fig. 5. Comparison of DISK Utilization between SAS and SSD.

Looking at the drop in throughput of application, at the same time utilization numbers are showing a linear increase. We monitored the locking in the application and database side with system monitoring tools like innotop [5], iostat and mpstat. Innotop is a 'top' clone for MySQL with many features and flexibility. It can easily monitors many servers at once and can aggregate across them. We had used innotop to find any locking activity on the tables in case of heavy load. It was learned from the tool that no table is getting locked during the 5 minute duration of the test. We got some confidence now that database is working fine without any locking under load. System tool iostat helped us in capturing the utilization for storage device, which was found to be underutilized even under heavy load. Using mpstat, we came to know that the application and database load is equally distributed across all the cores of the server.

On the other hand, we had used in house application profiling tool Jensor [6]. Jensor is a light-weight; low-overhead Java Profiler built using Java BCI. Jensor provides innovative analysis modules like JVM Replay, Pattern Analysis and Tagging Engine for technical as well as business analysis based on the same underlying data. Initial analysis showed that the time spent by the http request is more than 90 percent on the application server. It was quite clear by now that something is chocking at the application level.

We looked into the insights at the thread level using JConsole. It was observed that everything runs fine; Garbage collector is showing normal behavior, and CPU utilization below 10 percent, JVM memory is also in defined limits. During the course, we observed some unusual behavior in thread dump report. Most of the threads are in waiting or blocked state, in which the thread is waiting to obtain a lock on an object. This usually happen because a new thread is trying to enter a synchronized block or method while another thread already holds the associated lock. To get more confidence on the observation we used VisualVM [7], [8]. VisualVM is a tool that provides a visual interface for viewing detailed information about Java applications while they are running on a Java Virtual Machine (JVM). (See Fig. 6).

We had monitored the threads with visualvm and surprised to find that each thread is spending more than 94

percent of the time in monitor state. "Monitor" is the BLOCKED state, in which the thread is waiting to obtain a lock on an object [8]. Soon we realized this might happen due to overuse of synchronization block used in our application code. This led us to the code profiling of the application written in java. We had found that there were synchronization blocks on database read calls, which was restricting the application performance. We made few changes to the application logic to minimize the use of database calls and synchronization block. Also updated the read calls to non-synchronized blocks. We have repeated the same set of tests after making changes to the application.

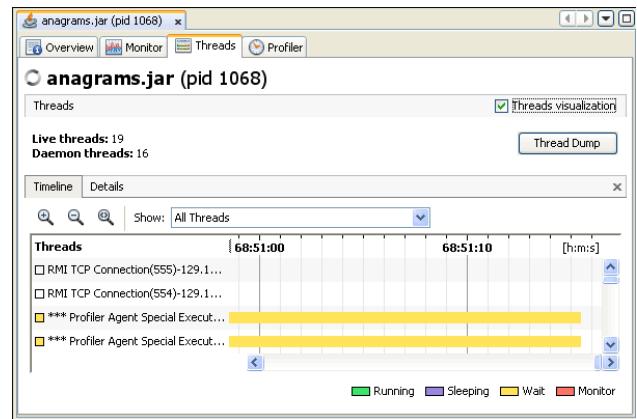


Fig. 6. Sample screen of visualvm monitoring tool.

Fig. 7 shows tremendous increase in the application throughput after removing the synchronization blocks on reads and reducing the number of insert/update database calls. Behavior for the local SSD remains the same as storage becomes the first hardware bottleneck. But, in this test case the storage device chocks at 10 user load only unlike the old test case where SAS device become the bottleneck at 200 user load [9].

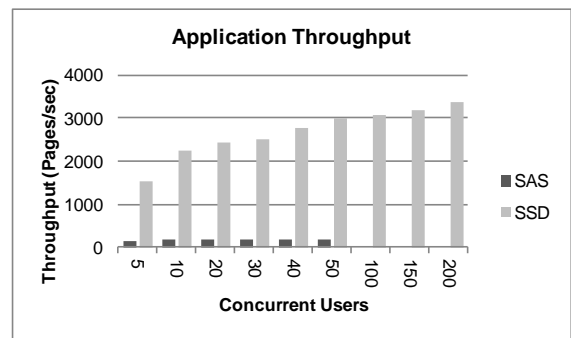


Fig. 7. Throughput comparison between SAS and SSD.

Fig. 8 gives the response time comparison between SAS and SSD storage conFig.uration.

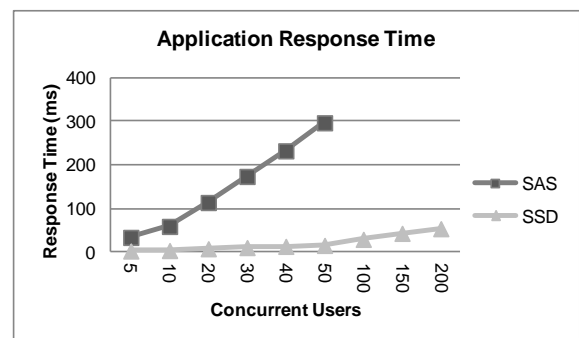


Fig. 8. Response time comparison between SAS and SSD.

Fig. 9 shows that CPU utilization hit 100% for SSD setup at 200 user load and become the first hardware bottleneck for the test case. Whereas, on the other hand SAS drive the utilization is low in comparison to SSD device.

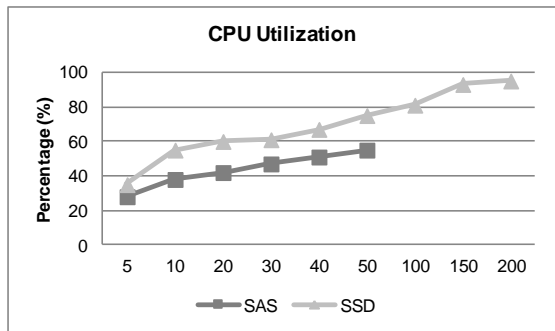


Fig. 9. Comparison of CPU Utilization between SAS and SSD.

Fig. 10 shows the disk utilization of the test case when synchronization block is removed. In this the mechanical SAS device got saturated at 10 user load only. We have taken the numbers only up to 50 concurrent users. While SSD utilization never shown above 50 percent during the complete load testing cycle.

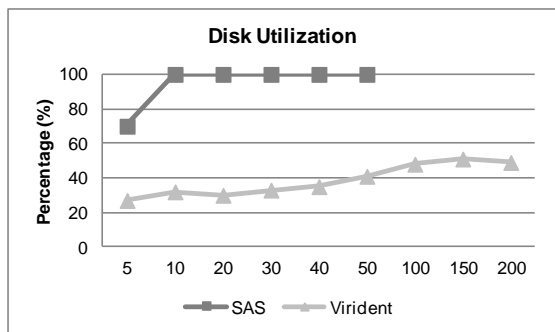


Fig. 10 Comparison of DISK Utilization between SAS and SSD.

Code profiling helped in overcoming the software bottlenecks within the application. With more number of CPU cores we might observe more throughput and better utilization of SSD device. We also expect similar performance results on other platforms as well. We had observed the result of visualvm monitoring tool after making the changes to the code. After the changes, time spent by individual thread in ‘monitor’ state was reduced to less than 15 percent and more than 80 percent in running state. Remaining 5 percent was observed in sleeping state[10], [11].

VII. CONCLUSION

SSD offers low I/O latency, a superior read and write

bandwidth and IOPS when compared to HDD. Achieved higher throughput in terms of transactions per second when SSD is the choice of storage device for the database. In a mixed workload test of an OLTP application SSD outperformed in throughput. The application is using flash-based SSD as a storage device for the database.

With respect to the OLTP benchmark that we ran during our evaluation of SSD we have the following conclusions:

- SSD as a storage device for database delivered 20x better throughput (pages/sec) than SAS disk configuration.
- It is always a good practice to monitor the soft bottlenecks during the load testing.
- SSD can help deliver higher throughput in an OLTP environment with lower response time in not only read-only, but also read-write workloads.

REFERENCES

- [1] G. Satyanarayana *et al.*, “Data warehousing, data mining, olap and oltp technologies are essential elements to support decision-making process in industries,” *Industries, International Journal on Computer Science and Engineering*, vol. 02, no. 09, 2010.
- [2] D. Cagdas and J. Bruce, “The performance of PC solid-state disks (SSDs) as a function of bandwidth, concurrency, device architecture, and system organization,” presented at ISCA’09, Austin, Texas, USA, June 20–24, 2009.
- [3] T. Thomas, “Sybase Adaptive Server Enterprise (ASE) 15.5 performance Evaluation using Fusion-io Solid State Memory”.
- [4] *Solid State Drive Performance*, David Bartizal & Thomas Northfield March 24, 2008.
- [5] Innnotop - A MySQL monitoring tool. [Online]. Available: <https://code.google.com/p/innnotop/>
- [6] Jensor Java Profiler. [Online]. Available: <http://sourceforge.net/projects/jensor/>
- [7] Java VisualVM. [Online]. Available: <http://docs.oracle.com/javase/6/docs/technotes/guides/visualvm/>
- [8] Java JVM profiling, thread status - what does “Monitoring” status mean? [Online]. Available: <http://stackoverflow.com/questions/6705400/java-jvm-profiling-thread-status-what-does-monitoring-status-mean>
- [9] J. Aldrich, C. Chambers, E. G. Sizer, and S. J. Eggers, “Static analyses for eliminating unnecessary synchronization from java programs,” in *Proc. 6th International Static Analysis Symposium*, September 1999.
- [10] M. Ji, E. Felton, and K. Li, “Performance measurements for multithreaded programs,” *ACM SIGMETRICS/Performance*, 1998.
- [11] J. Bartolomé and J. Guitart, “A survey on java profiling tools,” Oct. 2001.



Rajesh Meena was born in Jaipur, India in 1986. He received the B.E. degree in computer science from the Birla Institute of Technology, Mesra, Ranchi, India in 2006.

In 2010, he joined the Performance Engineering Research Center, Mumbai, where he was working as a performance engineer and lab administrator. His current research interests include virtualization, cloud computing and performance modeling. He submitted his contribution in AES-NI technology in a recognized conference ICCTS 2012.