Journal of Theoretical and Applied Information Technology
31st December 2012. Vol. 46 No.2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

RESEARCH ON PERFORMANCE OPTIMIZATION OF OLTP SYSTEMS BASED ON INNODB

[1]XIANGSHENG KONG
[1]Department of Computer & Information, Xinxiang University, Xinxiang, China
E-mail: [1]fallsoft@163.com

ABSTRACT

The technology of flash memory SSDs (solid state drives) which are increasingly adopted in a wide spectrum of storage systems has the potential of changing the database architecture and principles. With the high random access speed and high IOPS of the SSD, this paper describes a secondary buffer pool & read-ahead solution based on OLTP for MySQL InnoDB which can reduce I/O requests & latency and improve flash-based database system performance a lot. Based on the buffer pool & read-ahead solution, this paper provide an experience on running online transaction processing workloads (TPCC) which show that the database performance will achieve up to 120%-320% improvements.

Keywords: *SSD, RAID WRITE BACK MODE, BUFFER POOL & READ-AHEAD, TPC-C, Sysbench*

1. INTRODUCTION

Online Transaction Processing (OLTP), also known as transaction oriented processing, is a technology used to insert data into, or update data in, an operational or production database such as that used by a point-of-sale system. Compared to Online Analytical Processing (OLAP), a typical OLTP system may be updated frequently -- perhaps thousands of times a day. Transaction speed is a critical concern, and the system is designed to handle high numbers of insert and update statements. In OLTP systems, data is arranged in rows that are stored in blocks on hard disks, but cached into main memory on a database server so that they can be retrieved very quickly.

InnoDB provides MySQL with a transaction-safe (ACID-compliant) storage engine that includes commit, rollback, and crash-recovery capabilities. It also implements locking on the row level and provides a consistent non-locking read in SELECT statements.

The InnoDB transaction model is among the best on the market for it combines the best properties of a multi-versioning database along with a traditional two-phase locking. Locking is performed on a row level and queries are run by default as non-locking consistent reads, in the style of Oracle. The lock table in InnoDB is stored so space-efficiently that lock escalation is not needed: typically several users are allowed to lock every row in the database, or any random subset of the rows, without InnoDB running out of memory. MySQL's latest InnoDB engine can now do extensive, high-performance, full text search [1].

The database performance depends on the size of buffer pool and read-ahead algorithm which can reduce I/O requests & latency and improve flash-based database system performance a lot. The rest of this paper will organize as follow: Section 2 will present the structure and the principle of the bottom layer performance optimization of SSD-based MySQL InnoDB. TPC-C benchmark of isolation test will be analyzed and an evaluation of the high performance database system on SSD-based MySQL InnoDB will be provided in Section 3. Finally, the conclusion and the future work will be provided in Section 4.

2. PERFORMANCE OPTIMIZATION OF OLTP SYSTEMS BASED ON INNODB

Performance optimization of OLTP Based on InnoDB is not a simple task, but not the complex difficult task. It will be related to the database server performance (including CPU, Memory, and Hard Disk etc.), file I/O performance, operating system performance and system performance benchmark.

2.1 Solid State Drives (SSD)

Lately there has been a substantial influx of solid state drives (SSD), sometimes called a solid-state

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

disk or electronic disk, in the storage market. Until the last two years, most SSDs use NAND-based flash memory, which retains data without power. Compared to electromechanical disks, SSDs are typically less susceptible to physical shock, are silent, and have lower access time and latency. In particular for OLTP systems requiring fast access, but not necessarily data persistence after power loss, SSDs may be constructed from random-access memory (RAM) [2]. But SSD data is not "updated". That means, the performance of SSD read and SSD write are non-symmetrical, and read performance much faster than write performance. A continued effort to overcome its poor random write performance and to provide stable and sufficient I/O bandwidth havs been made since the market debut of flash memory SSD [3]. Every SSD includes a controller that incorporates the electronics that bridge the NAND memory components to the host computer. The controller is an embedded processor that executes firmware-level code and is one of the most important factors of SSD performance.

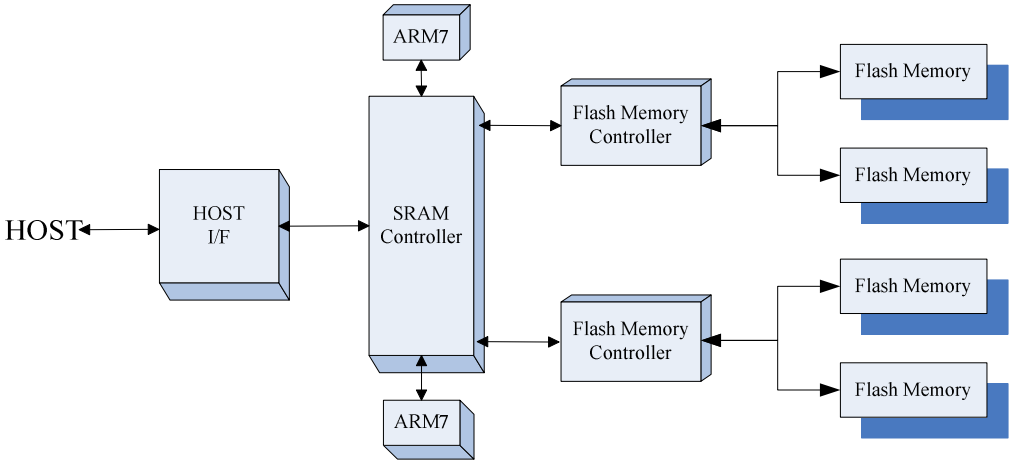


Figure 1 Example Of Dual-Channel SSD Architecture

Figure 1 illustrates how datum are received, transferred and processed in the dual-channel SSD architecture. Static random-access memory (SRAM) is a type of semiconductor memory that uses bistable latching circuitry to store each bit. In order to achieve high bandwidth and better IOPS (IO per second), every modern SSD adopts multi-channel and multi-way architecture and flash memory controller can read and write flash chips in parallel [4]. By increasing the number of channels, the SSD performance can be a linear increase.

In the OLTP systems based on InnoDB, a new variable named innodb_io_capacity was added, which controls the maximum number of IOPS that InnoDB will perform (which includes the flushing rate of dirty pages as well as the insert buffer batch size).

2.2 Buffer Pool & Read-Ahead

A read-ahead request is an I/O request to prefetch multiple pages in the buffer pool where a block is loaded at the midpoint for the first access and then moved immediately to the head of the list as soon as another access occurs asynchronously, in anticipation that these pages will be needed soon. InnoDB uses or has used two read-ahead algorithms to improve I/O performance. Linear read-ahead is based on the access pattern of the pages in the buffer pool, not just their number. If a certain number of pages from the same extent (64 consecutive pages) was found in the buffer pool, InnoDB asynchronously issued a request to prefetch the remaining pages of the extent. Random read-ahead added unnecessary complexity to the InnoDB code and often resulted in performance degradation rather than improvement.

Since InnoDB tries to keep the working set in the InnoDB buffer pool, if the InnoDB buffer pool is set large enough such as 22GB which is much more than the total of data size and index size, the buffer pool hit rate will reach 100%. At this time, the performance is optimal. The following function is usually used to show the buffer pool hit rate.

$$\text{innodb\_buffer\_read\_hits} = \frac{\text{Innodb\_buffer\_pool\_read\_requests}}{(\text{Innodb\_buffer\_pool\_read\_requests} + \text{Innodb\_buffer\_pool\_read\_ahead} + \text{Innodb\_buffer\_pool\_reads})}$$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2.3 RAID Write Back Mode

Redundant array of independent disks (RAID) is a storage technology that combines multiple disk drive components into a logical unit whose operating system will be regarded as a hard disk. Data is distributed across the drives in one of several ways called "RAID levels", depending on what level of redundancy and performance (via parallel communication) is required. Depending on the combination, the most commonly used RAID levels are RAID0, RAID1, RAID5, RAID10 and RAID50 etc. For OLTP systems based on InnoDB that are expected to be exclusively or strongly read-biased, RAID10 maybe is best choice, because it is taking into account the characteristics of RAID 1 and RAID 0 and offers a slight speed improvement over RAID 5 on sustained reads and sustained randomized writes.

Write Back Mode provides better performance in most cases. In Write - Back mode, the RAID controller acknowledges write I/O requests immediately after the data loads into the controller cache. The application can continue working without waiting for the data to be physically written to the hard drives. With Write Back Cache the write operation does not suffer from the Write time delay. The block of data is initially written to the cache, only when the cache is full or required will the data be written to the disk.

For OLTP systems based on InnoDB that are expected to be exclusively or strongly read-biased, Write Back Mode is very important. For example, if the value of sync_binlog is greater than 0, the MySQL server synchronizes its binary log to disk (using fdatasync()) after every sync_binlog writes to the binary log. There is one write to the binary log per statement if autocommit is enabled, and one write per transaction otherwise. A value of 1 is the safest choice. However, it is also the slowest choice (unless the disk has a battery-backed cache, which makes synchronization very fast).

3. SYSTEM PERFORMANCE BENCHMARK TOOL

System performance benchmark is a useful tool for OLTP systems based on InnoDB in developing a scientific methodology for performance evaluation of OLTP systems based on InnoDB. Two excellent tools will be introduced: sysbench and tpcc-mysql. This paper uses two OLTP (online transaction processing) applications (tpcc-mysql and Sysbench-OLTP) as a case study to investigate system performance benchmark.

3.1 Sysbench-OLTP

In computing, a benchmark is the act of running a computer program, a set of programs, or other operations, in order to assess the relative performance of an object, normally by running a number of standard tests and trials against it. Software benchmarks are, for example, run against OLTP systems based on InnoDB. Benchmarks provide a method of comparing the performance of various subsystems across different chip/system architectures.

SysBench is a modular, cross-platform and multi-threaded benchmark tool for evaluating OS parameters that are important for a system to run a database under intensive load. The idea of this benchmark suite is to quickly get an impression about system performance without setting up complex database benchmarks or even without installing a database at all.

For OLTP systems based on InnoDB, disk and OLTP performance is very important. Fileio test mode and OLTP test mode are most concerned. Fileio test mode can be used to produce various kinds of file I/O workloads. At the prepare stage SysBench creates a specified number of files with a specified total size, then at the run stage, each thread performs specified I/O operations on this set of files.

Here is an example:

$ sysbench --num-threads=16 --test=fileio --file-total-size=3G --file-test-mode=rndrw prepare

$ sysbench --num-threads=16 --test=fileio --file-total-size=3G --file-test-mode=rndrw run

$ sysbench --num-threads=16 --test=fileio --file-total-size=3G --file-test-mode=rndrw cleanup

OLTP test mode was written to benchmark a real database performance. At the prepare stage the following table is created in the specified database (sbtest table by default):

```
CREATE TABLE sbtest (
id int(10) unsigned NOT NULL auto_increment,
k int(10) unsigned NOT NULL default '0',
c char(120) NOT NULL default '',
pad char(60) NOT NULL default '',
PRIMARY KEY  (id),
 KEY k (k));
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Then this table is filled with a specified number of rows, and we can run OLTP test. Here is an example:

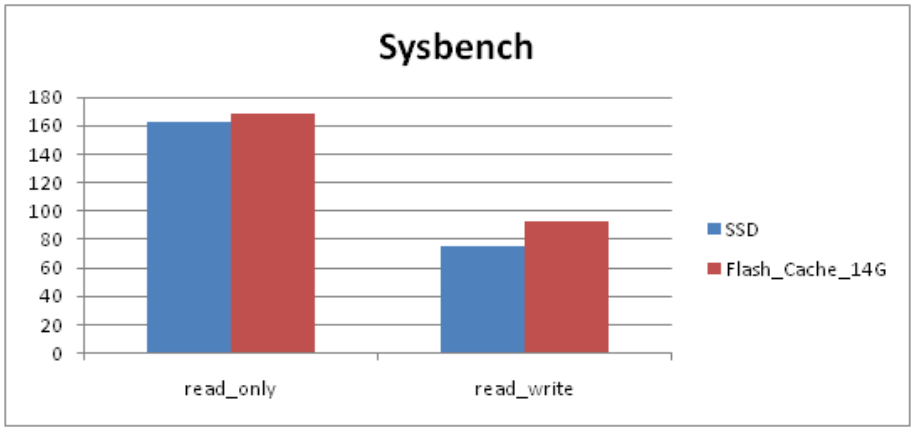
$ sysbench – test = oltp – mysql – table –engine = myisam – oltp – table – size = 1000000 – mysql - socket = /tmp/mysql.sock

$ sysbench – num – threads = 16 – max –requests = 100000 – test = oltp – oltp – table – size = 1000000 – mysql – socket = /tmp/mysql.sock –oltp – read - only

3.2 TPCC-MYSQL

The TPC-C benchmark is a well-known benchmark used to measure the performance of high-end systems. TPC-C simulates the execution of a set of distributed, on-line transactions (OLTP), for a period between two and eight hours. For Oracle Database, we use one of the TPC-C implementations written by Hammerora Project. For Postgres Database, e uses the implementation from TPCC-MYSQL. For mySQL, we use one of the TPC-C implementation from tpcc-mysql.

TPC-C incorporates five types of transactions with different complexity for on-line and deferred execution on a database system. The execution of the benchmark produces a performance parameter, called "TPC-C transactions per minute" (tpmC). This parameter allows comparing the speed of different systems. These transactions perform the basic operations on databases such as inserts, deletes, updates and so on. From data storage point of view, these transactions will generate reads and writes that will change data blocks on disks [5].



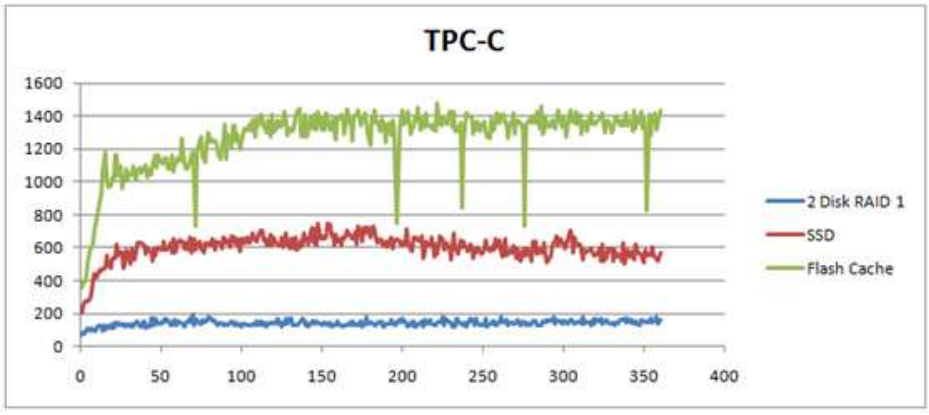


Figure 2 System Performance Benchmark Tools: Sysbench And Tpcc-Mysql

The New Order transaction simulates the entry of a complete order into the system. It is classified as a mid-weight, read-write transaction by the TPC-C standard, involving a mix of INSERT, SELECT and UPDATE statements.

The Payment transaction processes a payment made by a customer, updating their balance and updating the relevant summary statistics on the warehouse and district tables. It is intended to simulate a typical lightweight, frequent transaction where fast response time is important.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The Order Status transaction simulates the querying of a customer's last order, retrieving data from several tables. It is a read-only transaction intended to be run infrequently. It is considered mid-weight by the TPC-C standard.

The Delivery transaction is intended to simulate a typical deferred or "batch" type of transaction that can be run in the background on a given system. It is considered to have the least stringent response requirements and is executed infrequently.

The Stock Level transaction is read-only, querying the number of recently sold items that are below a certain threshold. As a heavyweight transaction, it intended to be run infrequently, with relaxed response requirements [6].

3.3 Performance

For sysbench the environment is (shown in Figure 2):

DB Size = 20G

innodb_buffer_pool = 6G

innodb_log_files_in_group=2

innodb_log_file_size=2000M

innodb_buffer_pool_instances=8

innodb_io_capacity=20000

For tpcc-mysql the environment is (shown in Figure 2):

DB Size = 40G

innodb_buffer_pool = 12G

innodb_log_files_in_group=2

innodb_log_file_size=2000

Minnodb_buffer_pool_instances=8

innodb_io_capacity=20000

4. CONCLUSION AND FUTURE WORK

The objective of this research is to evaluate the distributed database approach that can improve the performance of LTP systems based on InnoDB. We have presented several main optimization approaches that we consider in order to improve the performance of OLTP systems based on InnoDB. In order to better tune on InnoDB, understanding the applications and scope of InnoDB becomes critical. We have introduced a commonly used system performance benchmark tool that can more effectively calculate the load bearing capacity of the current system and analysis results of performance optimization.

As for the future work, we plan to design the experiment system to maximize the performance gain by secondary buffer pool.

REFERENCES

[1] Horikawa, Takashi, "An approach for scalability-bottleneck solution: Identification and elimination of scalability bottlenecks in a DBMS," the 2nd Joint WOSP/SIPEW International Conference on Performance Engineering, pp. 31-41, 2011.

[2] Yongkun Wang, Kazuo Goda, Miyuki Nakano, Masaru Kitsuregawa, "Early Experience and Evaluation of File Systems on SSD with Database Applications," IEEE Fifth International Conference on Networking, Architecture and Storage (NAS), pp. 467- 476, July 2010.

[3] Sang-Won Lee, Bongki Moon, and Chanik Park, "Advances in Flash Memory SSD Technology for Enterprise Database Applications," Proceedings of the ACM SIGMO, pp. 863–870, 2009.

[4] Eun-Mi Lee, Sang-Won Lee, and Sangwon Park, "Optimizing Index Scans on Flash Memory SSDs," Proceedings of the ACM SIGMO, pp. 5–10, December 2011.

[5] Qing Yang, Weijun Xiao, and Jin Ren, "PRINS: Optimizing Performance of Reliable Internet Storages," 26th IEEE International Conference on Distributed Computing Systems, pp. 32, 2006.

[6] Alexander Tomic, "MoSQL: A Relational Database Using NoSQL Technology," Master's Thesis, Faculty of Informatics of the University of Lugano, September 2011.