-M.S. THESIS

Characterizing Database Workloads via a Comprehensive I/O Analysis

포괄적인 IO 분석을 통한 데이터베이스의 워크로드 특성 파악

January 2018

DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Joongsuk Park

Characterizing Database Workloads via an Comprehensive I/O Analysis

포괄적인 IO 분석을 통한
데이터베이스의 워크로드 특성 파악

지도교수 염헌영

이 논문을 공학석사 학위논문으로 제출함

2017년 11월

서울대학교 대학원
컴퓨터공학부
박중석

박중석의 공학석사 학위논문을 인준함
2018년 1월

위 원 장 _____(인)
부위원장 _____(인)
위 원 _____(인)

Abstract

Nowadays high-performance storage like NVMe is widely used for various area where massive I/O is required. Although the ideal read or write speed on those high-performance storage is much improved than previous ones, in practical usages there are many factors which are making performance degradation. One of them is occurred when the mixed read and write workload is executed. The read action cannot execute but wait until write action made a lock on same physical device for writing. Many OLTP database workloads are consisted of mixed read and write. So if those cases are happen frequently, high performance device cannot be utilized its capability.

Workloads have each characteristic. TPC-C has a table which is read only. If this table move its table file location to physically separate one, read action to this table doesn't need to wait until the write action's completion. As a result, the entire performance can be increased according to the read only table's query frequency.

In this paper I/O analysis can be used to identify the characteristics of database workloads. After the analysis is completed, the experiments which the separation of read only tables and other read/write mixed ones are executed. Without any other configuration except the location of table file, the performance TpmC from TPC-C will be increased 7% on NVMe storage. So more proactive read/write separation would be effective to improve the entire OLTP database workloads.

Keyword : Database Workload Characterize, I/O Analysis, Mixed Read/Write, I/O Separation, OLTP Workload
Student Number : 2016-21204

Contents

List of Figures

List of Tables

Chapter 1

Introduction

High performance storage like NVMe storage is released for the scenarios which have to do heavy I/O workloads [4]. It is possible to utilize faster speed by using PCIe interface that have much more bandwidth than SATA. The device have specification documents which usually has the benchmarks score that sequential read/write, random read/write, or mixed read/write etc. Generally read speed is much faster than write one. This is because write action need to lock and unlock for writing. But read action don't need to lock for reading. So read might wait until write action finish in the mixed read and write workloads. This makes the entire performance worse.

OLTP (Online Transaction Processing) workloads are frequently used in database system. TPC-C is well-known OLTP workload which emulates online wholesales scenario like order, payment and delivery. TPC-C is used for performance benchmark. Most of database workloads are executed via pre-configured transaction. So it is possible to identify certain tables are used by specific transaction. To optimize the I/O performance, the understanding of the transaction pattern is useful.

In this paper, I/O analysis is used for identify the factors of database performance degradation and find out the solution. We provides the background information on Chapter 2. Three topics will be explained: 1) TPC-C benchmark on Mysql, 2) Mysql data and log structure and 3) Profiling via blktrace. Then on Chapter 3, we define the problem as performance degradation on mixed read/write transaction, and the target environment.

To understand the problem environment, we analysis TPC-C workload on Chapter 4. We find which query is frequently executed per tables when TPC-C is running. And we trace both log and data area via blktrace tool. In the data area, we divide read and write behavior to analyze each detail. Then to find out root-cause of degradation, we analyze the query execution part from Mysql database source code on Chapter 5. Especially differences of read and write actions are focused.

To solve the issue, we design the modification of configuration of Mysql database on Chapter 6. Before doing that, we check the relationship of TPC-C transaction and each tables. And select the tablespaces which are need for optimization. Then evaluation report is written on Chapter 7. Finally we conclude our paper on Chapter 8.

Chapter 2

Background

This chapter explains the background of this paper. First, brief explanation about TPC-C benchmark on Mysql. This includes the internal behavior of TPC-C. And we explain the data and log structure of Mysql for identifying which part can be optimized. Then blktrace for profiling the database I/O is explained.

2.1 TPC-C on Mysql

We use mysql database for TPC-C workloads. Because Mysql is open-source database and very widely used for commercial area.

2.1.1 TPC-C Benchmark

TPC-C is moderately complex OLTP (Online transactional Processing) workload. Figure 2.1 shows the TPC-C workflow. It models the wholesale scenario like order, payment and delivery. TPC-C has total 5 transactions. 88% of total transactions are occurred on both new-order and payment. Others are checking the status of order, processing delivery and checking the stock. To benchmark performance, TpmC metric is used. This metric is calculated by how many new order transactions are occurred within the given time [2].

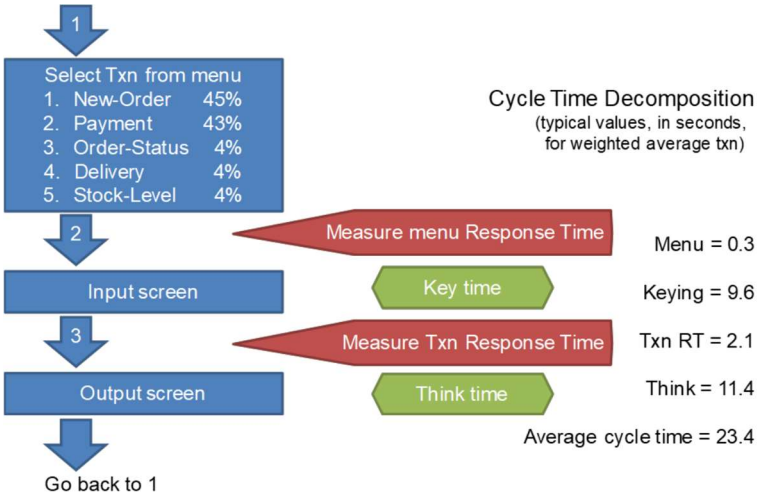


Figure 2.1 TPC-C Workflow

TPC-C has 9 tables including Warehouse, Stock, Item, Distinct, Customer, Order, New-Order, History and Order-Line. Figure 2.2 shows the schema of TPC-C database. The queries of each transaction are execute on these tables.

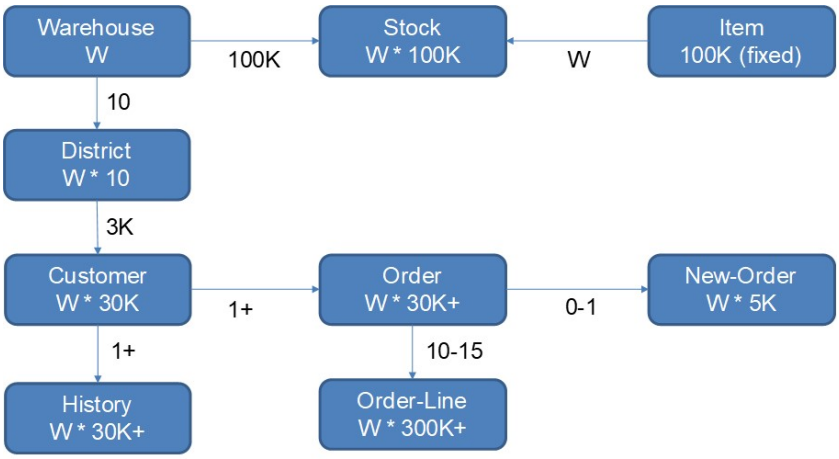


Figure 2.2 TPC-C Database Schema

Those queries mainly consist of SELECT, INSERT, UPDATE, DELETE. Figure 2.3 shows the queries per transaction source code. Both Order-Status and Stock-Level transaction has only select queries. That means these transactions are read only. But other transactions have other queries which execute write operations including read ones.

Neword.c : enter a new order from a customer
- select customer, warehouse, district / update district
- insert orders, new_orders
- select item, stock / update stock
- insert order_line

Payment.c: update customer balance to reflect a payment
- update warehouse / select warehouse
- update district / select district
- select customer.... / update customer..
- insert history

Delivery.c: deliver orders (done as a batch transaction)
- select new_orders / delete new_orders
- select orders / update orders
- update order_line / select order_line
- update customer

Orderstat.c: retrieve status of customer's most recent order
- select customer..., orders, order_line

Slev.c : (stock-level) monitor warehouse inventory
- select district, order_line, stock

Figure 2.3 TPC-C Queries per transaction source code

2.1.2 Running on Mysql

In Mysql, there is a data folder for storing database files. After creating a database, the name of database folder is generated in that data folder. And Mysql manage database's each table as a separate file. Extension ibd file has table's data which is called as tablespace. And extension frm file has table's structure. For instance of TPC-C, new_oder.ibd, new_order.frm, … are generated after TPC-C loading.

When TPC-C is started, TPC-C code makes queries. And these queries were sent to mysqld (Mysql Demon processor). In mysqld, the threads execute queries against mysql (Mysql Server Engine). Internal threads like 1) Innodb_purge_thread, 2) innodb_write_io_threads, 3) innodb_read_io_threads, 4) innodb_page_clean also executed during TPC-C run.

When TPC-C is run, only 4 tables' size are increased. Figure 2.3 shows insert is related with only 4 tables (orders, new_orders, order_line and history). Other tables which execute update queries doesn't change the size but the modification of contents occurred.

2.2 Mysql Data and Log Structure

There are several actions when writing on Mysql. One is for writing on data itself. Another is for logging. Logging is consist of sequential log file data and check-pointing for log file metadata. If the file system is used journaling, JBD2 process writes journaling data. Beside of logging from Mysql, journal is log for file system. Figure 2.4 shows the write action on Mysql. It separates data and log device. And uses EXT4/JBD2.

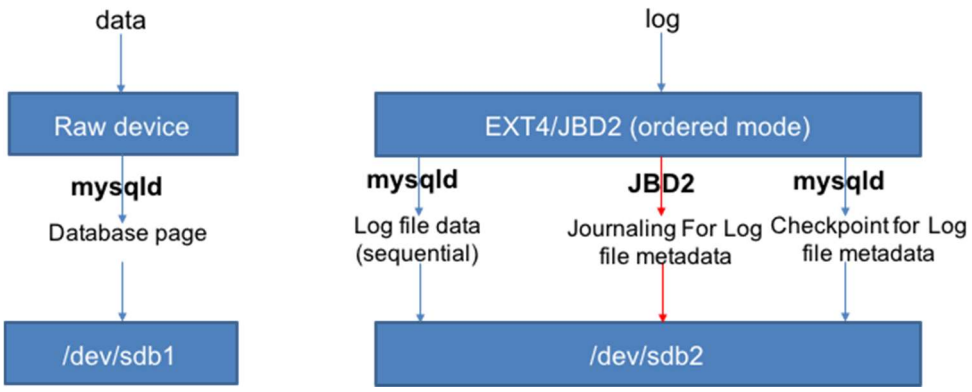


Figure 2.4 Write on mysql(log device is separated)

2.3 Profiling via blktrace

blktrace is a profiling tool for analyzing block I/O layer. It makes possible to show I/O between block device and storage. For example, it is possible to know which action is processed in storage LBA(logical base address) when executing write or read operation. In user space, blkparse is usually used together for parsing the result of blktrace. Figure 2.5 shows the blktrace processing flow for database workloads. We used blktrace for deep understanding of the DB workload in this paper.

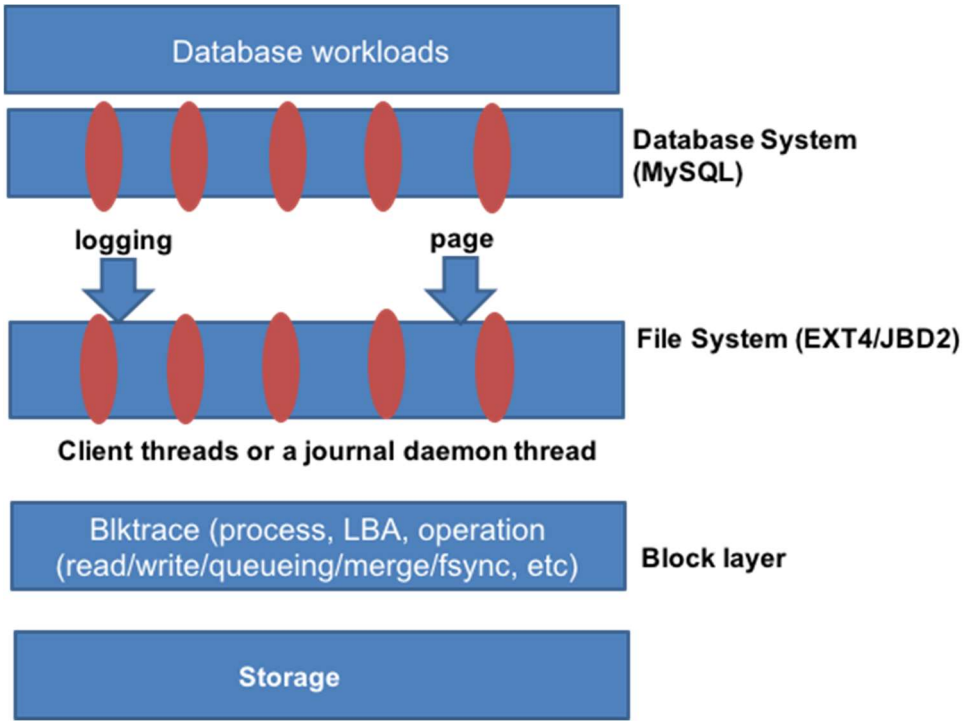


Figure 2.5 Blktrace processing flow for database I/O workloads

Chapter 3

Problem Definition and Target Environment

Now we focus the problem which we want to solve and explain target environment. Although the target environment is limited in the experiments, we believe it would be also applied more general situations.

3.1 Performance Degradation on Mixed Read Write

We found many papers that treated with mixed read write issue on database [3]. The root cause of this issue is the lock mechanism when write operation is occurred. We took the solution as database engine code modification. There are difference between write and read operation. So we measure the performance degradation for write lock. Then we considered the appropriate solution with the result.

3.2 Target Environment

We select TPC-C database workload. Because it is still widely used and simulates real online transactions. And we choose Intel P3700 as NVMe storage device. The device is divided as two part. Each one is 800GB. It can be connected as RAID for increasing performance. But we use each two part as an individual one for separate test. As an experiment with FIO, it is same performance with two physically separated storage devices. It is because PCIe bandwidth is enough for the device I/O performance.

Chapter 4

TPC-C Workload Analysis

Before the drill down the issue, let us analyze the TPC-C workload. We need to check the characteristic like how many percentage of each query during TPC-C running, and how frequently using tables. We checked this by turning on general log in mysql database. After running TPC-C, we analyze the log file. And we analyze profiling data both data and log via blktrace.

4.1 Query and Table Percentage on TPC-C

As you can see in Figure 4.1, TPC-C is SELECT dominant workload. The SELECT query is 62.72% in entire queries. SELECT query don't need to write operation for data. Although it still need write to buffer for caching, but it doesn't need to hold write lock. Other queries is UPDATE 20.44%, INSERT 15.64% and DELETE 1.20%.

We need to find out which queries are called frequently per table. Figure 4.2 shows the result. Item table is 100% SELECT. So we assume if item table placed with other read/write mixed table, the read performance of item table might be degraded.

Additionally stock is most frequent accessed table. But those access consist of both SELECT and UPDATE queries. The percentage of SELECT is much more than UPDATE one. So we remain this to be future works.

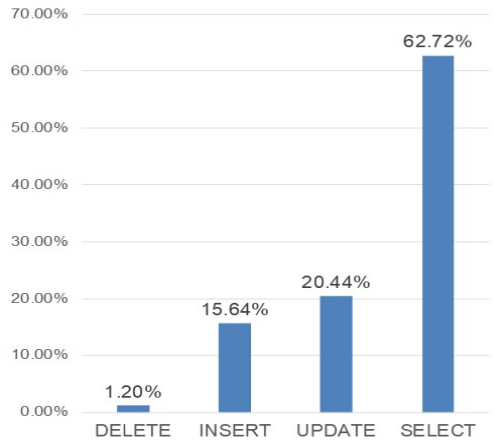


Figure 4.1 Percentage of each query

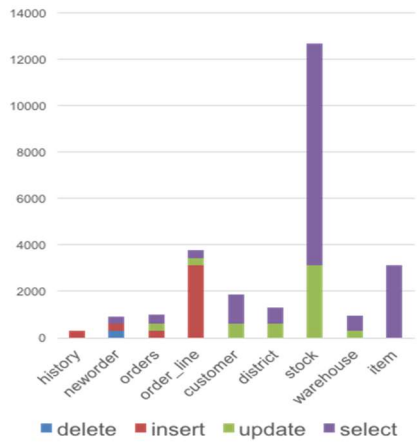


Figure 4.2 Query percentage per table

We also analyze table usage per query. Figure 4.3 shows INSERT query is most frequently occurred in order_line table. Other three tables have same percentages of INSERT. Order_line table is related with order details. So most of insert queries are happened here. And Figure 4.4 shows UPDATE query is most frequently occurred in stock table. Stock table needs to be updated by each order status. So this results were generated.

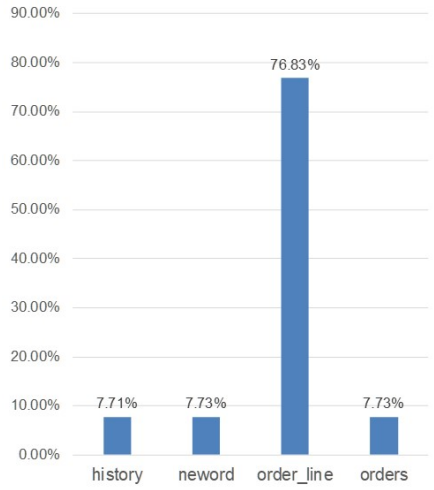


Figure 4.3 INSERT query per table

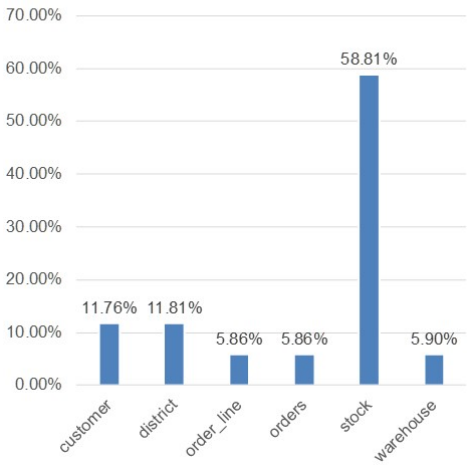


Figure 4.4 UPDATE query per table

And SELECT query is most frequently executed on stock table. And DELETE query is only occurred in neword table. Because neword table contain only the order before delivery.

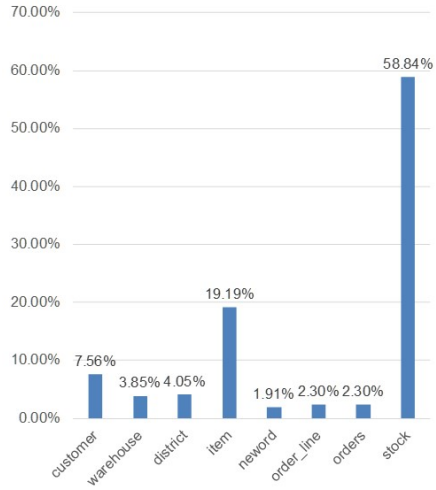


Figure 4.5 SELECT query per table

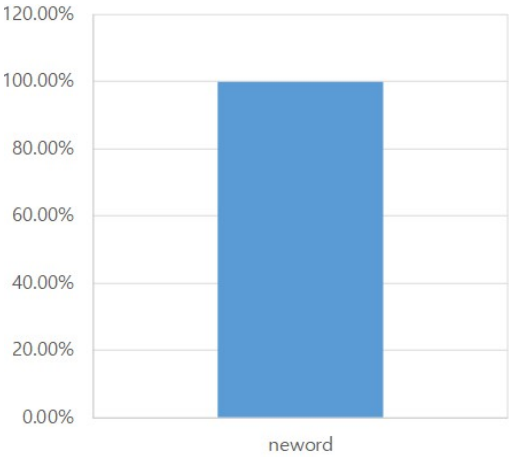


Figure 4.6 DELETE query per table

This analyze help us to decide which tables are good for modification for increasing performance.

4.2 Log Area Trace

If we profile with blktrace, we can find there are different behaviors between journal area and fixed area. We need to consider the logging impact for entire performance. So logging is also analyzed during TPC-C running.

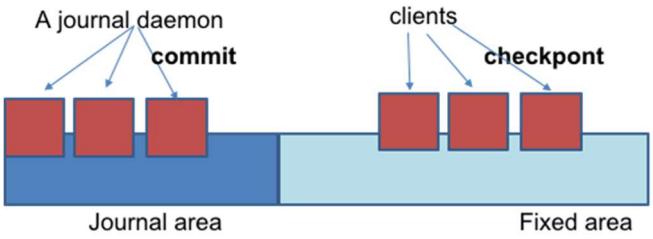


Figure 4.7 Journal daemon thread & client thread

Journal inserts each 8 blocks to queue, and if it approaches the fixed amount, it operates merge and commit. This behavior is like Figure 4.8. WS means Write Synchronous. And FWFS means FUA + Write + Flash + Synchronous.

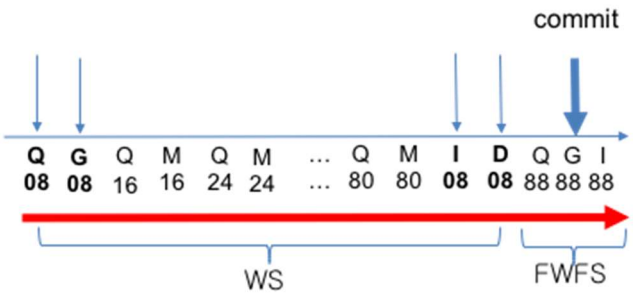
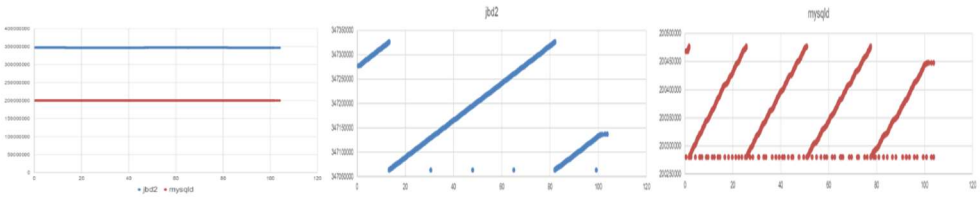


Figure 4.8 Blktrace LBA result for log during TPC-C running

If we convert blktrace result to chart, we can get Figure 4.9. jbd2 and mysqld processes have separate log ranges on each. Mysqld log area can be set via mysql configuration. And both are circular patterns. It means if log is full, previous written log is overwritten.



Jbd2 & mysqld jbd2 mysqld

Figure 4.9 mysqld log and jbd2 log during TPC-C

4.3 Data Area Trace

We did profile entire transaction by dividing read and write. 4.3.1 shows read and 4.3.2 shows write. For checking read, which LBAs are used during TPC-C read operation including entire and all 5 transactions. And for checking write, entire and 3 write related transactions which are occurred. We can compare this figure with each transaction chart.

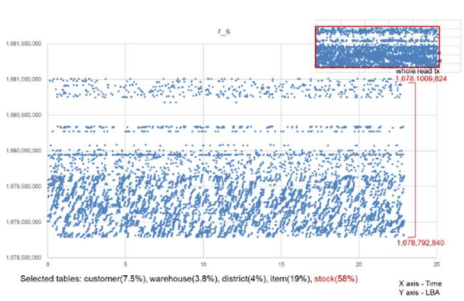
4.3.1 Data Read Behavior

Figure 4.10 shows the entire read transaction. Some LBA seems to be frequently used. We can identify it with each separately transaction views.

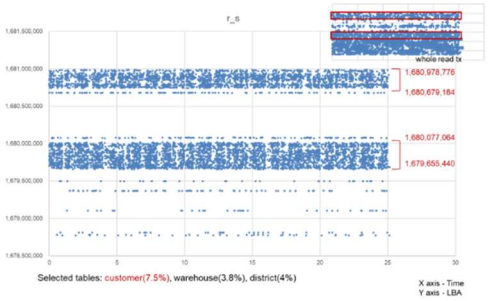
Figure 4.10 Entire TPC-C read

Figure 4.11 shows the 5 transaction's read. Because of executing 1 transaction separately, more actions are occurred. But we can identify each table's location with LBA. For example, we can find customer table from payment and stock table from

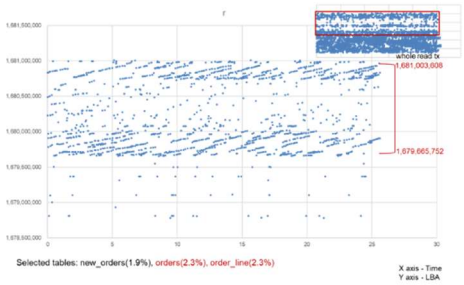
stock level tables are accessed.



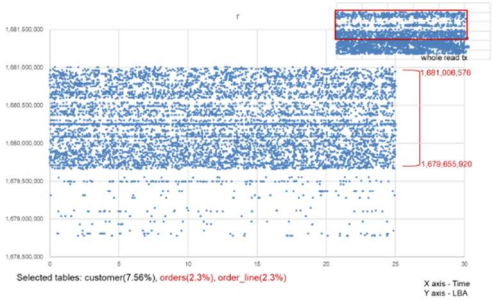
a) new-order



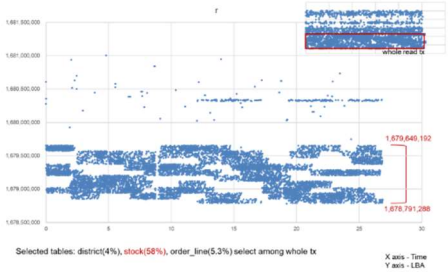
b) payment



c) delivery



d) order_status



e) stock level

Figure 4.11 5 transactions on TPC-C read

4.3.2 Data Write Behavior

Write is not shown as specific patterns in LBA. It is scattered to the entire LBAs. Figure 4.12 shows the results.

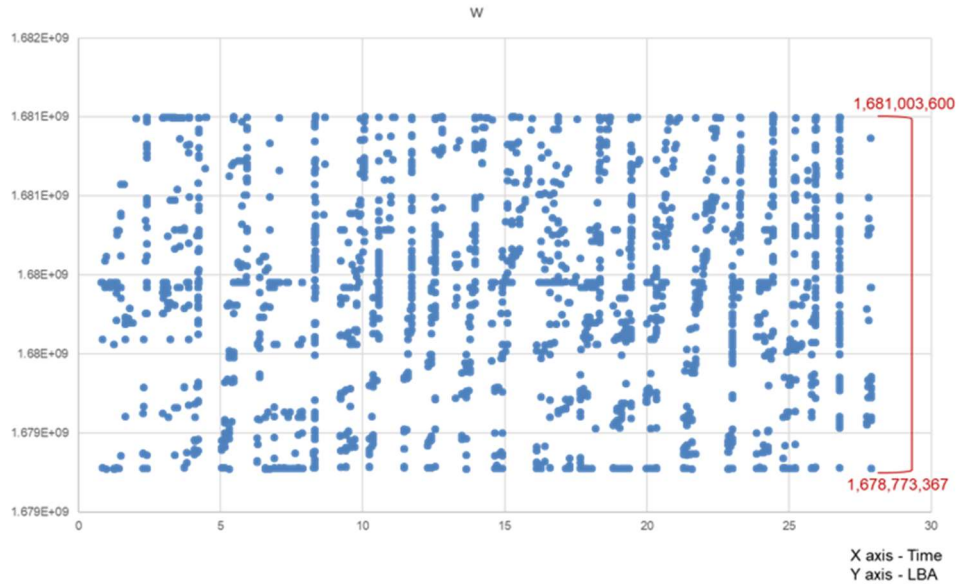
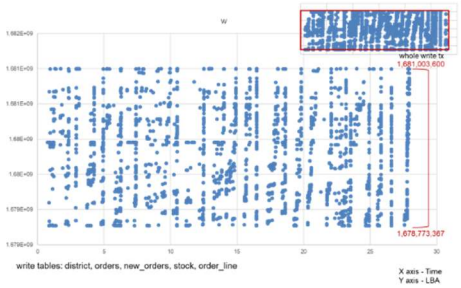
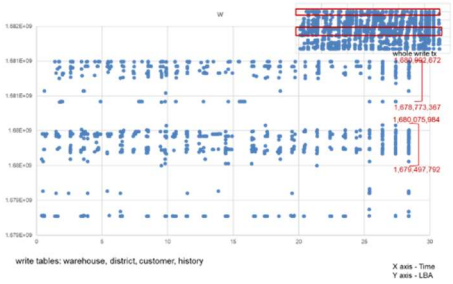


Figure 4.12 Entire TPC-C write

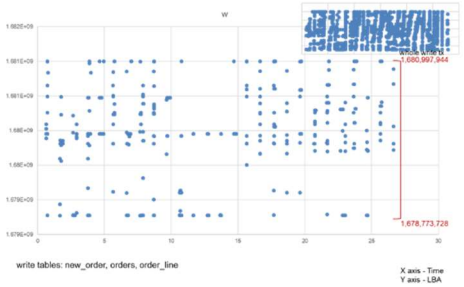
Other transactions has the amount of differences. But it is not easy to find out the specific patterns.



a) new order



b) payment



c) delivery

Figure 4.13 3 transactions on TPC-C write

Chapter 5

Mysql Query Execution Analysis

To drill down root cause of mixed read and write performance degradation, we analyze mysql source code. We focused on read part and write part. Read part is mainly consists of SELECT query execution. And write part is consisted of other queries like INSERT, UPDATE, and DELETE including SELECT.

5.1 Mysql Query Execution Common

Mysql query is initialized by Mysql client and send to Mysql demon. Then after the parsing is done, mysql_execute_command is executed according to query. This is occurred long switch case syntax in source code. Figure 5.1 show the execution flow on Mysql.

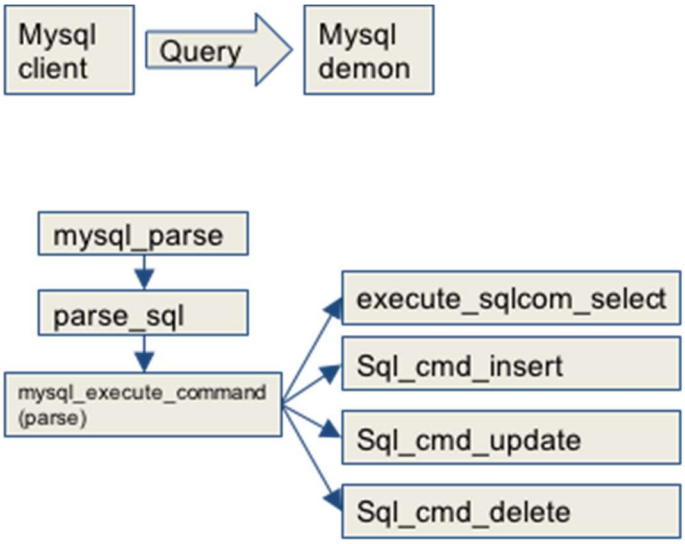


Figure 5.1 Mysql query execution flow via function name

5.2 SELECT Query Execution Flow

execute_sqlcom_select function is initial point for SELECT. Then handle query and execute exec. do_select is actually doing main part in select query. Using rnd_next and row_search_mvcc make search the data location via buf_page_get_gen and get the tables data via fil_io. Figure 5.2 show the key functions and blktrace result. Two LBAs means metadata and actual data for the data.

Figure 5.2 SELECT query flow and blktrace result (select id from tst_tb)

5.3 INSERT, UPDATE, DELETE Query Execution Flow

Key difference of INSERT, UPDATE, DELETE and SELECT is lock. SELECT don't need to lock. INSERT need 'fill_record' to insert and using 'row_insert_for_mysql' fuction make actual write process. Figure 5.3 shows the process and blktrace result.

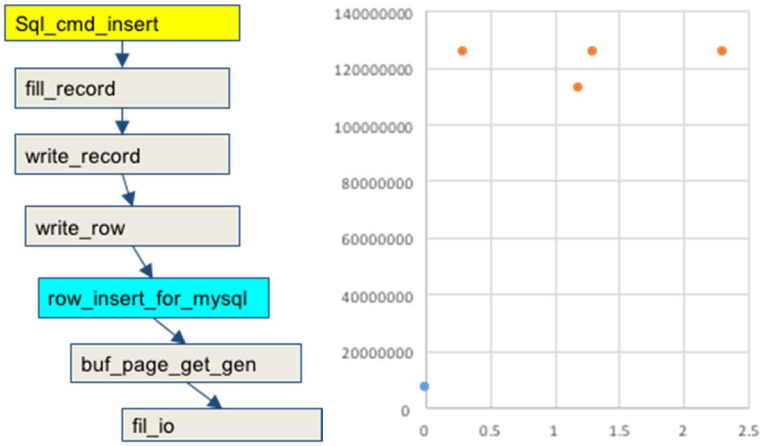


Figure 5.3 INSERT query flow and blktrace result(insert into test_tb (id) values (1))

UPDATE is firstly getting the original location. And write new data and replace the metadata. Figure 5.4 shows the UPDATE query flow and blktrace result.

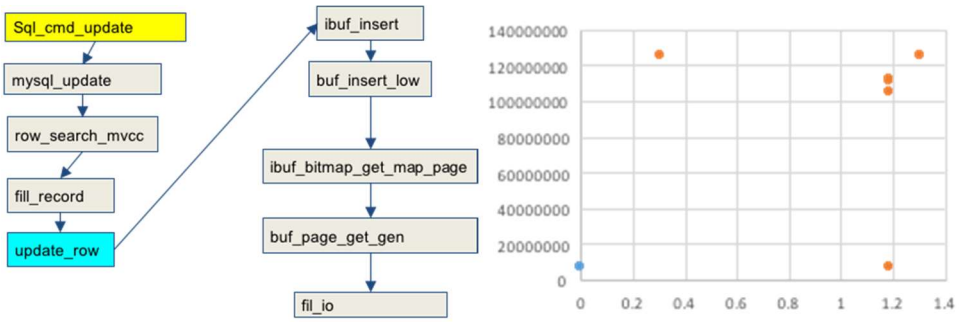


Figure 5.4 UPDATE query flow and blktrace result(update test_tb set id = 2 where id = 1)

DELETE is relatively similar with UPDATE. Because it is also updating metadata. Figure 5.5 show DELETE query flow and blktrace result.

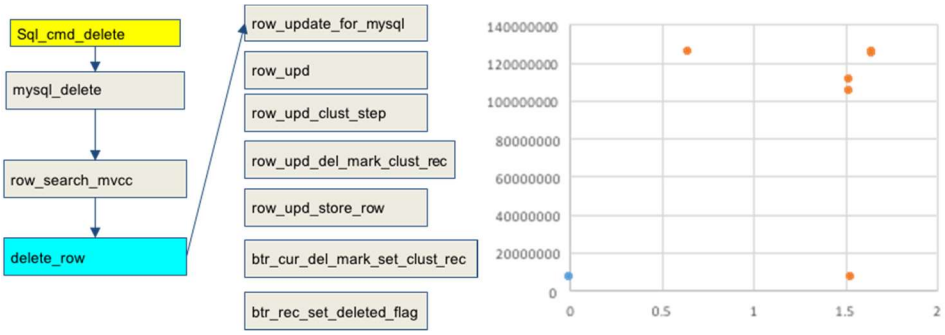


Figure 5.5 DELETE query flow and blktrace result(delete from test_tb where id = 2)

Chapter 6

Design

Until now, we saw the TPC-C workload pattern and mysql read/write flow. And we found the read only table need to be separated for performance.

6.1 Transaction and Table Relation

In TPC-C, the query is executed per transaction. Figure 6.1 shows this as a table. Item table is 100% read. And it is frequently used by new order transaction. So this would be a good candidate for optimization.

	Customer (66.3% sel)	Warehouse (66.7% select)	district	orders	order_line	new_orders	Item (100% sel)	stock	history
neword(45%)	s	s	s/u	I	I	I	s	s/u	
payment(45%)	s/u	u/s	u/s						I
delivery(4%)	u			s/u	u/s	s/d			
ordstat(4%)	s			s	s				
slev(4%)			s		s			s	

Table 6.1 Transaction and Table relation in TPC-C

6.2 Optimization of Tablespace Location

Mysql has separate file per table. So we can easily place item table to physically different storage. But we need to make work together with link file. And to get the result clearly, we edited TPC-C code for executing new order transaction only. New order transaction is about half number of entire transactions. So it can be comparable for unmodified version.

Chapter 7

Evaluation

Evaluation environment is below.
CPU: 4-way, 8 core
RAM: 128GB
Storage: Intel NVMe p3700 (no RAID configured).
Mysql: 5.7 (O_DIRECT: not using cache, Buffer pool: 1G)
TPC-C: the number of warehouse 500, running time 10 minutes

We moved Item.ibd and Item.frm file to a physically separated storage. Both Mysql data files are on NVMe. To work together, we made symbolic link to Item.ibd and Item.frm files.

Figure 7.1 shows the comparison with storage location. TPC-C 'src unmodified' version makes do not noticeable increment for both same and different location. But 'src modified' version which executes only new order transaction, the performance increase about 7%. This is because NVMe is enough fast device to overcome the performance degradation for low rate of mixed read and write frequency [5]. But this kind of method is valid for the excessive I/O for specific read only table with reducing delay from write.

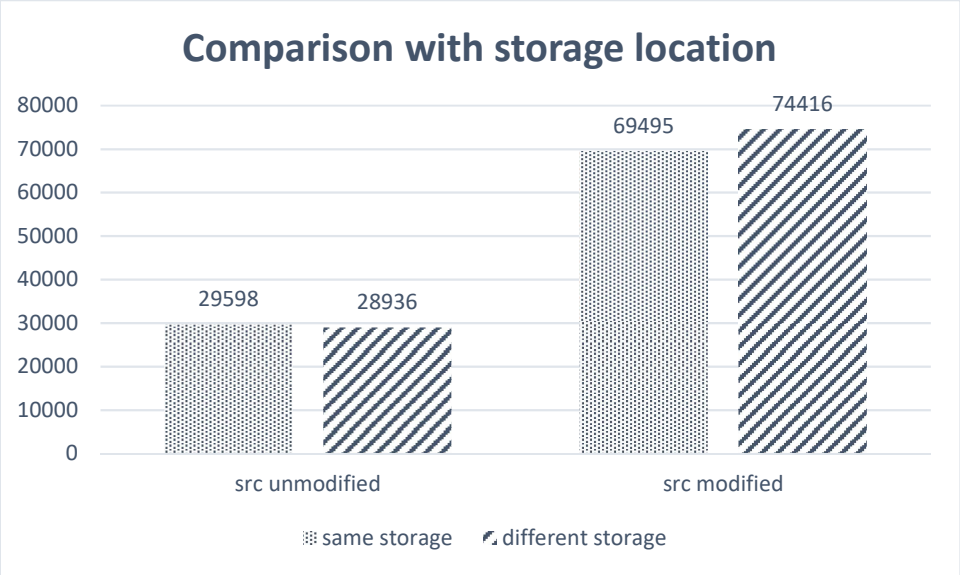


Figure 7.1 Comparison with storage location (TpmC)

Chapter 8

Conclusion

If we already know workload, or identify the characteristic of the workload, we can easily increase the I/O performance via read/write separation. This is still valid on the high performance storage like NVMe. In this paper, we have tested with read only table. But there are mixed read/write tables. We believe read intensive tables also can be get the similar benefits from this separation. Physical separation would be good choice. But if the storage environments are not enough to support separate, partitioning make similar the performance increment.

Bibliography

[1] Chen, S., Ailamaki, A., Athanassoulis, M., Gibbons, P. B., Johnson, R., Pandis, I., & Stoica, R. (2011). "TPC-E vs. TPC-C: characterizing the new TPC-E benchmark via an I/O comparison study" *ACM SIGMOD Record*, 39(3), 5–10.

[2] Leutenegger, S. T., & Dias, D. (1993). "A Modeling Study of the TPC-C Benchmark". *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, 22(2), 22–31.

[3] Nguyen, D. T., Zhou, G., Xing, G., Qi, X., & Hao, Z. (2015). "Reducing Smartphone Application Delay through Read / Write Isolation" *The 13th International Conference on Mobile Systems, Applications, and Services-MobiSys'15*, 287–300.

[4] Chen, F., Chen, F., Koufaty, D. a., Koufaty, D. a., Zhang, X., & Zhang, X. (2009). "Understanding intrinsic characteristics and system implications of flash memory based solid state drives" *ACM SIGMETRICS Performance Evaluation Review*, 37(1), 181–192.

[5] Xu, Q., Siyamwala, H., Ghosh, M., Suri, T., Awasthi, M., Guz, Z., … Balakrishnan, V. (2015). "Performance analysis of NVMe SSDs and their implication on real world databases" *Proceedings of the 8th ACM International Systems and Storage Conference on - SYSTOR '15*, 1–11.

[6] Park, J. Y., Jung, H., & Kim, J. T. (2015). "Exploring SSD Suitable Allocation Schemes Incompliance with Workload Patterns", 9(12), 1390–1393.

[7] Chen, F., Lee, R., & Zhang, X. (2011). "Essential roles of exploiting internal parallelism of flash memory based solid state drives in high-speed data processing" *Proceedings - International Symposium on High-Performance Computer Architecture,* 266–277.

Abstract in Korean

읽기와 쓰기가 섞여 있을 경우 읽기가 쓰기가 완료될 때까지 기다리는 것 때문에 성능이 저하되는 문제는 빠른 저장 장치에서도 여전히 나타나는 문제이다. 이로 인해 고성능 스토리지가 성능을 충분히 발휘하지 못하고 있다.

데이터베이스 워크로드를 분석해보면 읽기만 수행하는 테이블이 있을 수 있다. 따라서 이 같은 테이블의 경우는 쓰기 작업이 일어나는 테이블과 물리적으로 저장소를 분리한다면 읽기만 수행하는 테이블은 쓰기를 기다리지 않고 읽기 작업을 할 수 있기 때문에 전체적인 성능 향상을 가져올 수 있다.

인텔 P3700 NVMe 저장장치에서 오픈소스 데이터베이스인 mysql 을 이용해서 TPC-C 워크로드를 총 32개 스레드로 돌려서 상기한 읽기와 쓰기가 섞여서 발생하는 성능 저하의 원인을 파악하고, 읽기만 하는 테이블을 쓰기와 같이 하는 테이블과 분리해서 저장함으로써 성능이 향상됨을 확인했다.

이 논문에서는 데이터베이스에서 I/O 패턴 프로파일링 등을 통해서 워크로드 자체의 변경 없이 데이터베이스 성능(TpmC)을 7% 향상 시킬 수 있었다.

주요어: Database Workload Characterize, I/O Analysis, Mixed Read/Write, I/O Separation, OLTP Workload
학번: 2016-21204