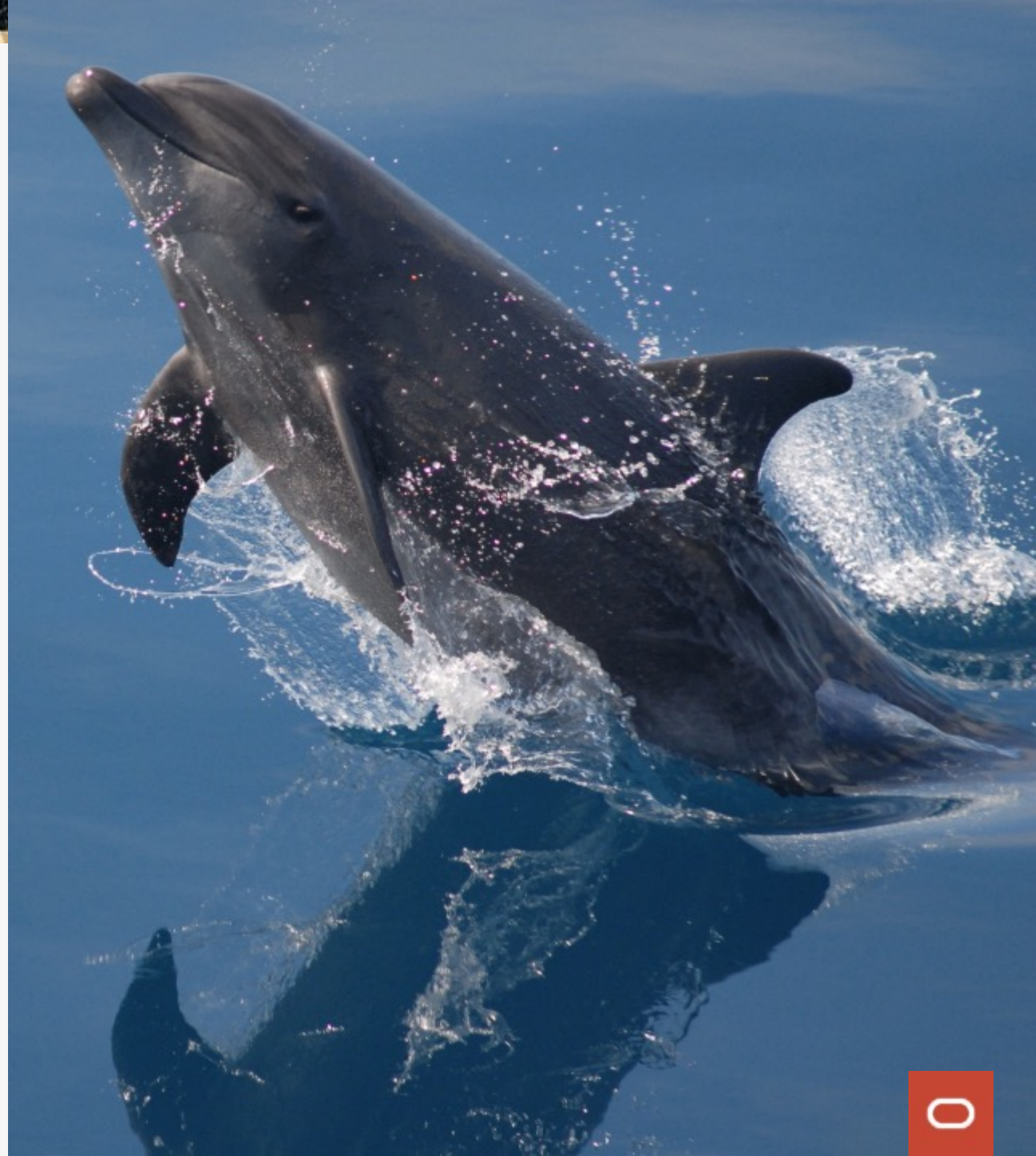
ORACLE

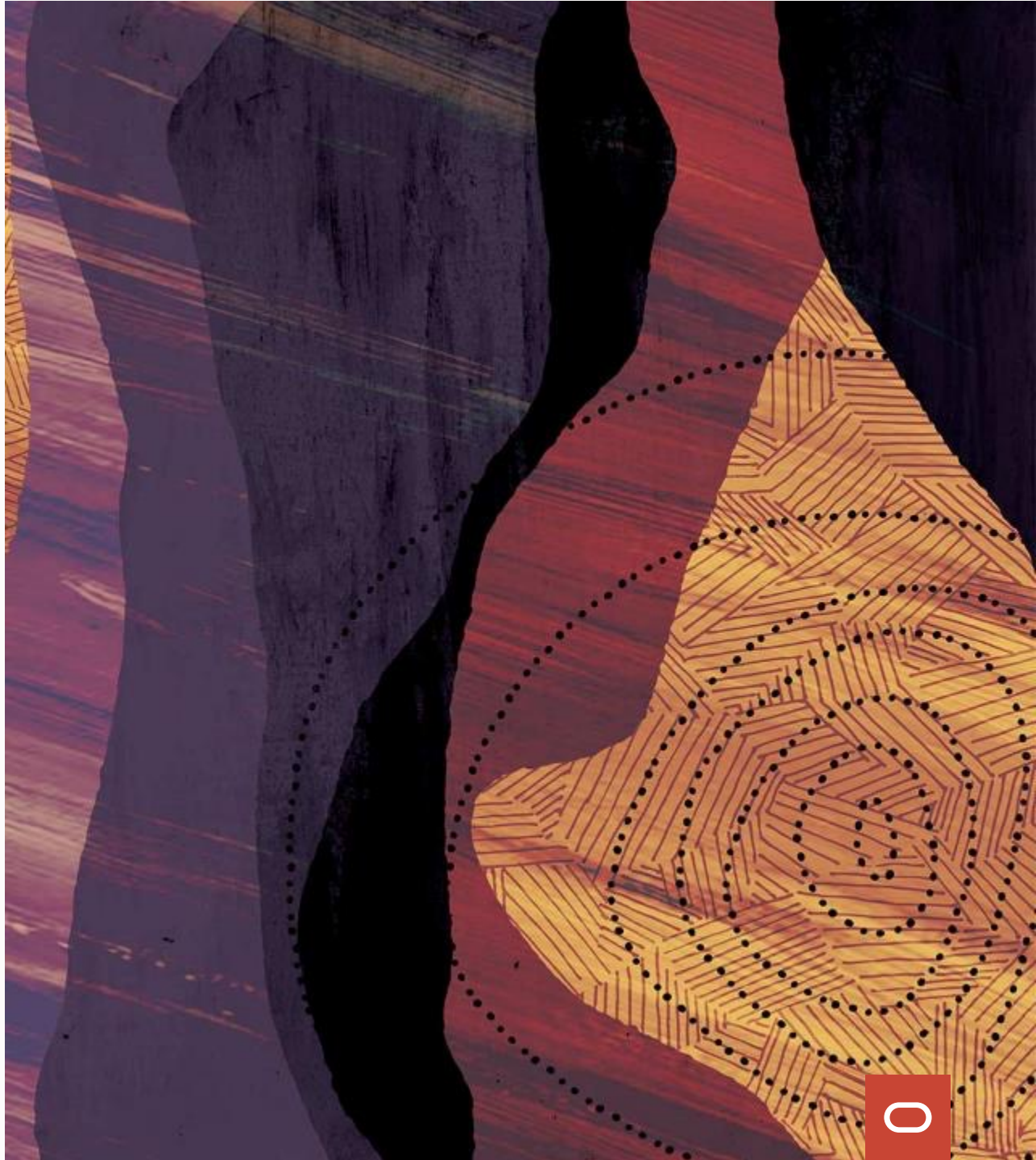
MySQL Performance

Best practices

박혜선(huishan.piao@oracle.com)
MySQL Solution Engineer
Oracle, MySQL GBU



목차



MySQL Innovation: 5.7 -> 8.0

MySQL 5.7

- 3x Better Performance
- Replication Enhancements
- Optimizer Cost Model
- JSON Support
- Improved Security
- Sys & Performance Schema
- GIS

MySQL HA Cluster

- MySQL ReplicaSet
- MySQL InnoDB Cluster
- MySQL Group Replication
- MySQL Router
- MySQL Shell

MySQL 8.0

- 2x Better Performance
- Data Dictionary
- NoSQL Document Store
- JSON
- CTEs
- Window Functions
- InnoDB
- Roles
- Unicode
- GIS

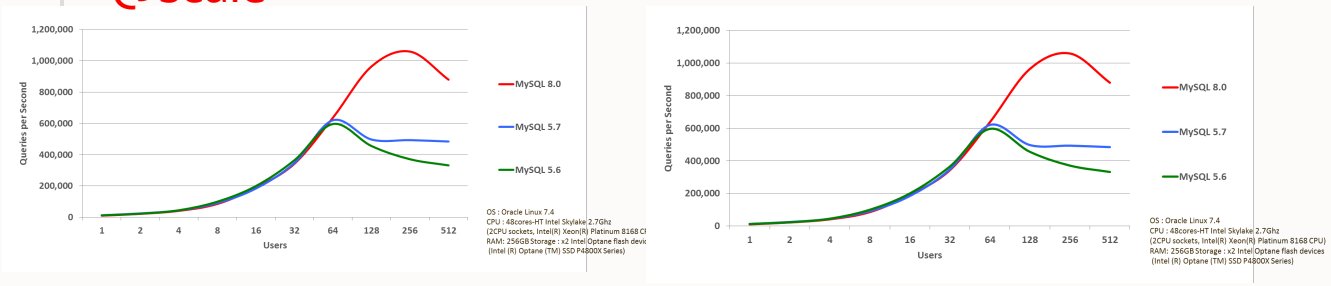
2 Years in Development
400+ Worklogs
5000+ Bugs Fixed
500 New Tests

Performance improvements in MySQL 8.0

- Read/Write 부하 확장
  - Re-designing how InnoDB writes to Redo log
- IO Capacity 충분히 이용
  - Removing file system mutex
- 경쟁이 높은 작업 부하
  - Contention Aware Transaction Scheduling
  - CATS over FIFO
- 자원 그룹
  - Thread–CPU mapping can be managed

- UTF8MB4
- 부분 JSON/BLOB 업데이트
- Information Schema
- Performance Schema
- 향상된 코스트 모델 (mem/disk aware)

2x Faster than MySQL 5.7 for RO and RW @scale



Basic Concepts

Improving Performance

- 응답 시간(response time) 만족:
  - 쿼리 성능 최적화
  - 기존 시스템 부하를 줄임
  - 더 많은 요청을 처리하도록 시스템을 확장
    - Scaling up
    - Scaling out

Performance Tuning Terminology

- 응답 시간(response time)
  - Wait time
    - Waiting for I/O operation to complete
    - Waiting for a lock to be released
  - Service time
    - HW/network
    - Concurrency
    - Query efficiency
- 처리량(Throughput)
  - Transaction per second (TPS)
  - Query per second(QPS)
- 확장성
  - 요청이 많아짐에 따라 일정한 응답 시간 기대치 내에서 처리량이 증가되거나 안정하게 유지

Tuning Steps

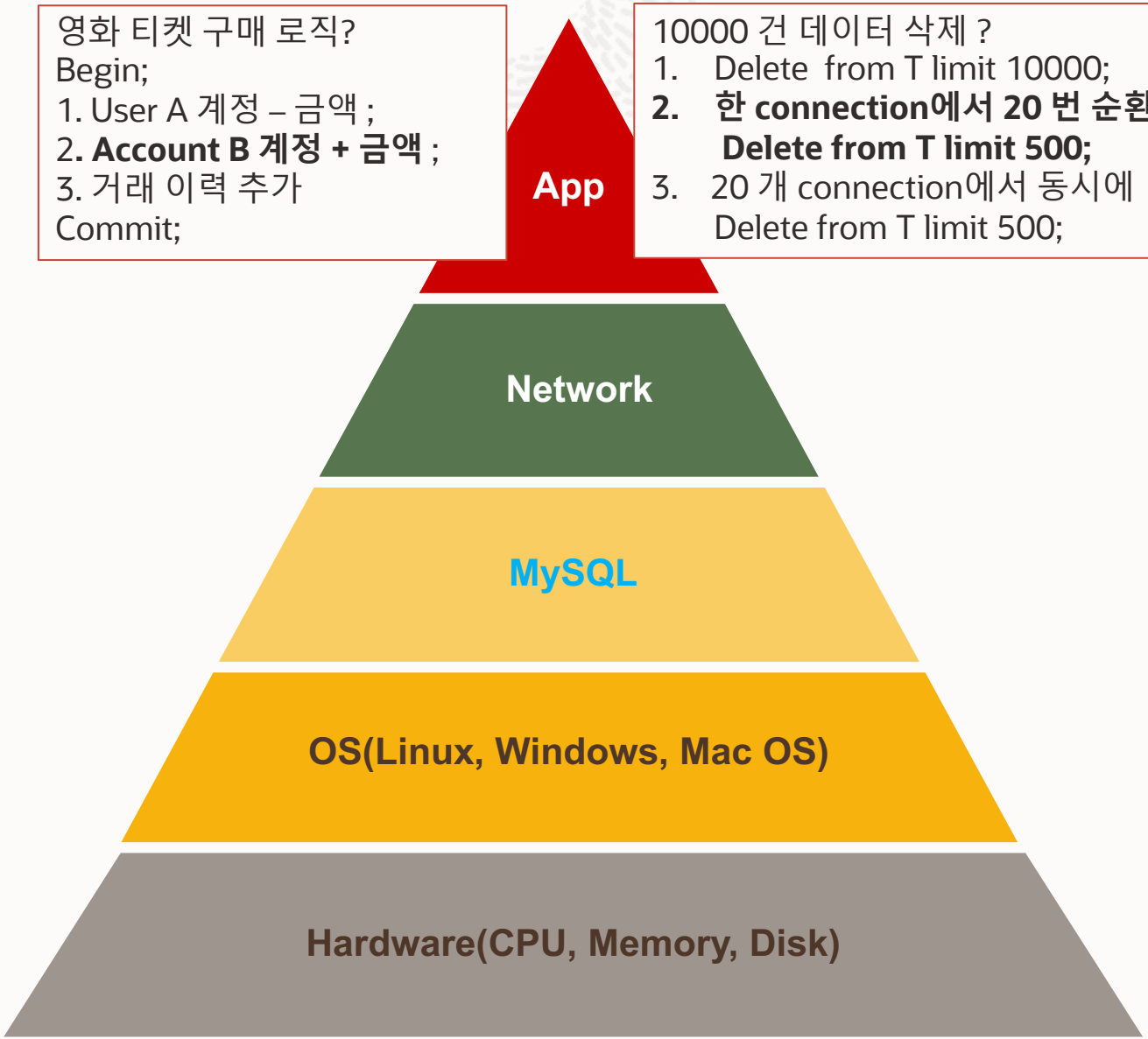
- 시스템 모니터링으로 기준을 찾아내고 이슈 여뷰 판단
- 잠재적으로 가장 유리한 포인트를 가지고 튜닝
  - 대기 시간이 가장 긴 쿼리
  - 서비스 시간이 가장 긴 쿼리
  - 가장 빈번하게 실행되는 쿼리
- 튜닝 목표를 확정
  - 목표에 달성하면 튜닝 멈추기
  - 튜닝 목표는 구체적이고, 측정 가능하고, 달성할 수 있어야 함

Performance Tuning Point

- 애플리케이션
  - 업무 로직 개선
  - 부하 혹은 데이터 사이즈 줄임
- 네트워크
- **데이터베이스**
  - Configuration 튜닝
    - Memory/buffer/connetion/log size
  - Schema 설계 최적화
  - Query 성능 최적화
- OS / HW
  - User limits(such as open files) 등 설정
  - CPU & 메모리 & 디스크 확장 (scale up)
  - 서버 대수 증가(scale out)

영화 티켓 구매 로직?
Begin;
1. User A 계정 – 금액 ;
2. **Account B 계정 + 금액** ;
3. 거래 이력 추가
Commit;

10000 건 데이터 삭제 ?
1. Delete from T limit 10000;
2. **한 connection에서 20 번 순환**
   Delete from T limit 500;
3. 20 개 connection에서 동시에
   Delete from T limit 500;



Performance Tuning
HW and OS

Memory

- 작업 데이터 셋이 innodb buffer pool (RAM)에 적합한지 확인
  - 사용 가능한 메모리의 70-80%를 InnoDB buffer pool에 할당 할 수 있음
- MySQL connection 도 메모리를 소모, 일부 MySQL buffers는 연결 마다 할당 됨
  - max_allowed_packet , thread_stack , net_buffer_length , max_digest_length
- 디스크 접속을 완화 하기 위한 Caching 및 Buffering
  - Redo log/Join/Sorting Buffer 와 Table Cache
  - Replication 관련: max_binlog_cache_size
- 기타 RAM :
  - OS 및 FS cache
  - temporary tables: tmp_table_size
  - Performance Schema
- 모니터링 방법: top 명령어로 확인, 80% 이하

Monitoring MySQL Memory Usage

- 대부분 메모리 지표는 비 활성화됨
  - SELECT * FROM performance_schema.setup_instruments WHERE NAME LIKE '%memory%';
- performance_schema :
  - memory_summary_global_by_event_name
- Sys 스키마 :
  - memory_global_by_current_bytes

- 상태 값 확인:
  - SHOW GLOBAL STATUS LIKE ‘table_open%’ ;
  - SHOW GLOBAL STATUS LIKE ‘binlog%’ ;
  - SHOW GLOBAL STATUS LIKE ‘created_tmp%’;

```
mysql> SELECT SUBSTRING_INDEX(event_name,'/',2) AS
       code_area, FORMAT_BYTES(SUM(current_alloc))
       AS current_alloc
       FROM sys.x$memory_global_by_current_bytes
       GROUP BY SUBSTRING_INDEX(event_name,'/',2)
       ORDER BY SUM(current_alloc) DESC;
+---------------------------+---------------+
| code_area                 | current_alloc |
+---------------------------+---------------+
| memory/innodb             | 843.24 MiB    |
| memory/performance_schema | 81.29 MiB     |
| memory/mysys              | 8.20 MiB      |
| memory/sql                | 2.47 MiB      |
| memory/memory             | 174.01 KiB    |
| memory/myisam             | 46.53 KiB     |
| memory/blackhole          | 512 bytes     |
| memory/federated          | 512 bytes     |
| memory/csv                | 512 bytes     |
| memory/vio                | 496 bytes     |
+---------------------------+---------------+
```

CPU

- 빠른 multi-core processors를 사용할 것 (24-48 cores)
- Hyper-Threading을 활성화
- 일반적으로 더 빠른 코아가 더 많은 느린 코아보다 성능에 유리
- 버전에 따라 코아에 대한 확장성이 향상됨
  - MySQL 5.1 scales to ~4 cores
  - MySQL 5.5 scale to ~16
  - MySQL 5.6 scales to ~36 threads (cores)
  - MySQL 5.7 scales to ~64 threads (32 Core-HT)
  - MySQL 8.0 scales to ~100 threads (test on 48 Cores-HT intel Skylake)

Disk

- 퀄리티가 높고 low-latency SSD 혹은 NVMe 디스크 권장
  - InnoDB datadir, tmp 파일 및 undo 로그는 모두 랜덤 IO, 더 빠른 SSD 권장
  - innodb_page_size = 4K (디폴트는 16K, 디스크 page 사이즈와 매칭)
  - innodb_flush_neighbors = 0
- Spinning 디스크는 “log“ (순서 IO)에 여전히 사용할 수 있음
- RAID 디스크를 사용한다면 성능을 위해 RAID 1+0 권장
- FBWC/BBWC (Flash/Battery-Backed Write Cache) 권장
  - write 성능 향상 및 crash-safe.
- 디스크 유형(rpm지표) 및 부하에 따라 innodb_io_capacity 설정
  - 디폴트값은 200, 느린 ssd 혹은 일반 하드 디스크에 적합

OS

- L of LAMP is Linux ;)
- Unix, Mac, Windows 도 지원
- 성능만 고려하신다면 Linux 권장
  - MySQL 8.0 인 경우 최신 커널 사용
  - 디폴트 파일시스템 사용, ext4 혹은 xfs
  - ulimit 을 사용하여 file/process limits 을 확장
  - MySQL 전용 서버인 경우, innodb_numa_interleave = 1로 권장
  - Swappiness, 권장 옵션은 (1-6): sysctl -w vn.swappinness=1
  - InnoDB 데이터 파일을 FS 캐시를 거치지 않음, innodb_flush_method=O_DIRECT

https://lefred.be/content/mysql-and-memory-a-love-story-part-1/

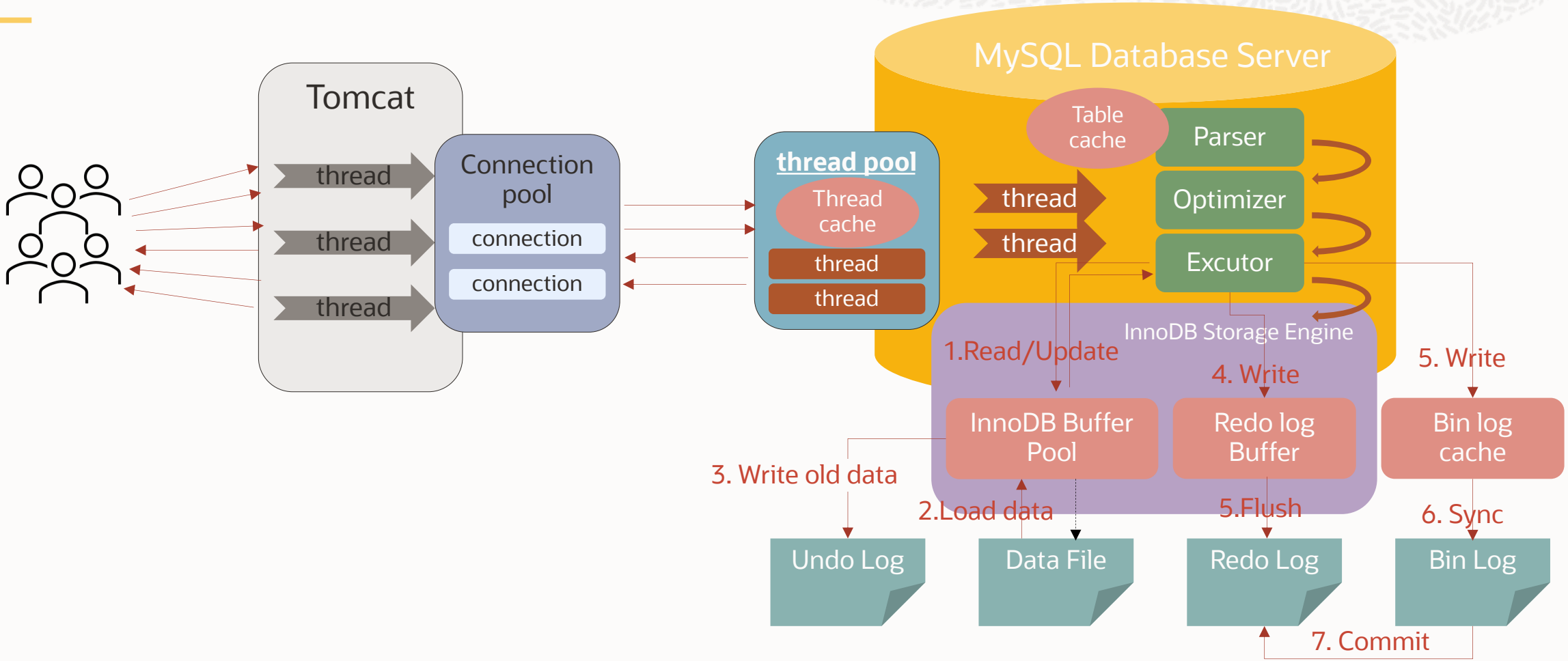
Performance Tuning Configuration

MySQL configuration

- 설정 파일(8.0버전) : my.cnf & mysqld-auto.cnf (예: SET PERSIST max_connections = 1000)
- 확인 방법 :
  - SHOW [SESSION|GLOBAL] VARIABLES LIKE '%expr%' ;
  - performance_schema.global_variables
  - performance_schema.session_variables
  - performance_schema.variables_by_thread;

```
mysql> SELECT t1.VARIABLE_NAME, t1.VARIABLE_SOURCE, t1.VARIABLE_PATH, t1.SET_TIME, t1.SET_USER, t1.SET_HOST, t2.VARIABLE_VALUE
       FROM performance_schema.variables_info t1
       JOIN performance_schema.global_variables t2 ON t2.VARIABLE_NAME=t1.VARIABLE_NAME
       WHERE t1.VARIABLE_SOURCE not like "COMPILED";
+-------------------------------+-----------------+----------------------------------------+----------------------------+----------+-----------+-----------------------------------------------------+
| VARIABLE_NAME                 | VARIABLE_SOURCE | VARIABLE_PATH                          | SET_TIME                   | SET_USER | SET_HOST  | VARIABLE_VALUE                                      |
+-------------------------------+-----------------+----------------------------------------+----------------------------+----------+-----------+-----------------------------------------------------+
| basedir                       | COMMAND_LINE    |                                        | NULL                       | NULL     | NULL      | /home/ted/src/mysql-8.0.21-linux-glibc2.12-x86_64/  |
| datadir                       | COMMAND_LINE    |                                        | NULL                       | NULL     | NULL      | /home/ted/sandboxes/MySQL-HOWTOs/mysqldata/         |
| default_authentication_plugin | EXPLICIT        | /home/ted/sandboxes/MySQL-HOWTOs/my.cnf | NULL                       | NULL     | NULL      | mysql_native_password                               |
| foreign_key_checks            | DYNAMIC         |                                        | 2020-11-18 08:17:26.019090 | NULL     | NULL      | ON                                                  |
| innodb_buffer_pool_size       | EXPLICIT        | /home/ted/sandboxes/MySQL-HOWTOs/my.cnf | NULL                       | NULL     | NULL      | 2147483648                                          |
| innodb_directories            | EXPLICIT        | /home/ted/sandboxes/MySQL-HOWTOs/my.cnf | NULL                       | NULL     | NULL      | /home/ted/sandboxes/MySQL-HOWTOs/slabb2/            |
| innodb_flush_log_at_trx_commit | DYNAMIC         |                                        | 2020-11-18 08:57:12.479082 | ted      | localhost | 1                                                   |
| log_error                     | COMMAND_LINE    |                                        | NULL                       | NULL     | NULL      | ./speedy.err                                        |
| pid_file                      | COMMAND_LINE    |                                        | NULL                       | NULL     | NULL      | speedy.pid                                          |
| plugin_dir                    | COMMAND_LINE    |                                        | NULL                       | NULL     | NULL      | /home/ted/sandboxes/MySQL-HOWTOs/mysqlsrc/lib/plugin/ |
| port                          | COMMAND_LINE    |                                        | NULL                       | NULL     | NULL      | 3306                                                |
| secure_file_priv              | EXPLICIT        | /home/ted/sandboxes/MySQL-HOWTOs/my.cnf | NULL                       | NULL     | NULL      |                                                     |
| socket                        | COMMAND_LINE    |                                        | NULL                       | NULL     | NULL      | /tmp/mysql.sock                                     |
+-------------------------------+-----------------+----------------------------------------+----------------------------+----------+-----------+-----------------------------------------------------+
```

MySQL Database Server 아키텍처



클라이언트 커넥션 관련 설정

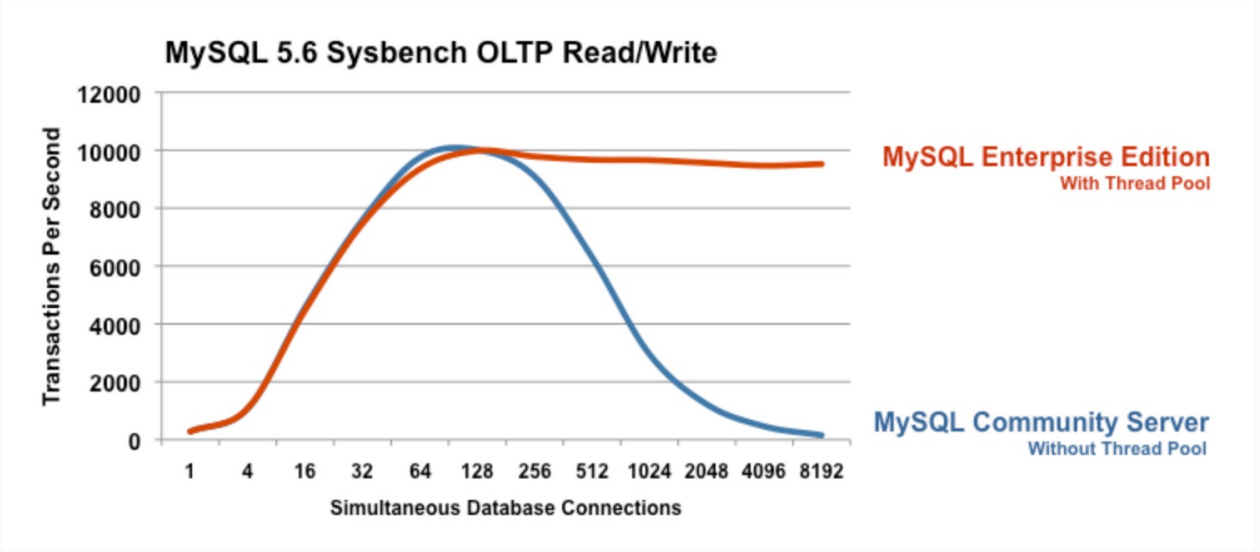
- max_connections 를 필요 이상으로 크게 설정하지 말아야 할 것
  - OS limit on file descriptors
    - Linux/Solaris : Up to 10,000
    - Windwos: Up to 16,384
  - 응답 시간의 기대치 및 워크로드 감안
  - 많은 연결 수가 예상되면 버퍼에 따른 메모리 용량을 확보
  - wait_timeout으로 연결을 닫음
- 상태 값 확인:
  - Connections
  - Max_used_connections
  - Connection_errors_max_connections

- 커넥션별로 할당되는 버퍼 값은 작게 시작:
  - sort_buffer_size
  - binlog_cache_size
  - net_buffer_length ~max_allowed_packet

  - read_buffer_size
  - read_rnd_buffer_size

Reusing Threads

- thread_cache_size : 서버가 스레드를 재 사용할 수 있도록 저장한 스레드 개수
  - 디폴트 값은 자동 설정: 8 + (max_connections/100)
  - Thread cache hit rate % = 100 – ((Threads_created /Connections) * 100)
- 연관된 상태 값 확인:
  - Connections
  - Threads_connected
  - Connections /Uptime
  - Threads_created/Uptime
  - Uptime_since_flush_status
  - Threads_cached
  - Threads_running
- thread_pool_size 스레드 풀 성능 튜닝
  - https://dev.mysql.com/doc/refman/8.0/en/thread-pool-tuning.html



Reusing Tables

- table_open_cache : 모든 스레드를 위해 서버가 캐시한 열린 테이블(디폴트 4000)
  - If too low ? Send the disk I/O request to open required tables
  - If too high? Cache is expensive to maintain
- 관련 상태 값 확인:
  - Open_tables
  - Opened_tables
  - Table_open_cache_hits
  - mysql> SHOW GLOBAL STATUS LIKE 'table_open%' ;
- 캐시 파티셔닝을 통해서 확장성 향상:
  - table_open_cache_instances
  - 각 캐시 사이즈 : table_open_cache/table_open_cache_instances

InnoDB buffer pool

- innodb_buffer_pool_size
  - 테이블 및 인덱스 데이터를 캐시한 InnoDB의 메모리 영역, LRU 알고리즘에 의해 관리
  - 전용 데이터베이스 서버에서 시스템의 실제 메모리 크기의 70%~80%정도로 설정 할 수 있음
  - innodb_buffer_pool_size = N* innodb_buffer_pool_chunk_size * innodb_buffer_pool_instances

- 확인 방법: SHOW ENGINE INNODB STATUS;
  - Total memory allocated
  - Buffer pool size
  - Free buffers
  - Database pages, Old database pages
  - Modified db pages
  - Pending reads, Pending writes
  - Buffer pool hit rate
  - ……

Innodb redo log

- innodb_log_file_size
  - write 성능에 유리하지만 복구 시간에 영향
  - 총 사이즈(~512G) : innodb_log_file_size * innodb_log_files_in_group(디폴트 2)
  - 운영 환경에서 최소 512M로 권장하고, 가능한 충분히 크게 설정할 것 권장
- redo 사용율:

  Used % = (Used log / Total log) * 100

  = (log_lsn_current – log_lsn_last_checkpoint / (innodb_log_file_size * innodb_log_files_in_group)) * 100
  = (47605855 / 100663296) * 100

  = 47.29 %

- SHOW ENGINE INNODB STATUS;

```
---
LOG
---
Log sequence number 602763740
Log flushed up to   602763740
Pages flushed up to 584668961
Last checkpoint at  555157885
```

- Information_schema. INNODB_METRICS:

```
mysql> SELECT NAME, COUNT
         FROM information_schema.INNODB_METRICS
        WHERE NAME IN ('log_lsn_current',
'log_lsn_last_checkpoint');
+--------------------------+-----------+
| NAME                     | COUNT     |
+--------------------------+-----------+
| log_lsn_last_checkpoint  | 555157885 |
| log_lsn_current          | 602763740 |
+--------------------------+-----------+
```

Trading performance over consistency (ACID)

- innodb_log_buffer_size = 16MB
  - Redo로그 I/O를 완화하기 위한 버퍼 영역
  - 트랜젝션 사이즈가 크면 commit 전에 disk IO가 발생되기에, 따라서 크게 조절할 필요가 있음

- innodb_flush_log_at_trx_commit = 1
  - InnoDB가 commit한 트랜젝션에 대해서 언제 flush 할 것인지 결정
  - 디폴트값인 1로 권장하지만 다음 상황에서 제외:
    - InnoDB가 commit한 트랜젝션에 대해서 언제 flush 할 것인지 결정
    - Bulk 데이타 로딩할 경우 2로 설정하거나 8.0 버전에서 redo로그를 비활성화
    - 예측하지 못한 peak 부하로 인해 디스크 I/O부하가 심할 경우
    - 데이터 로스가 발생하여도 괜찮을 경우……

바이너리 로그 성능

- binlog_cache_size에 비해 트랜젝션이 너무 크면, 임시 파일에 저장하기에 성능 영향
- 관련된 상태 변수 확인:
  - Binlog_cache_use
  - Binlog_cache_disk_use
  - 캐시 효율성% = (100 - (Cache Disk Use/Cache Use) * 100)
- 트랜젝션 사이즈를 제어함으로 바이너리 로그 사이즈 제어, 복제 효율성 향상
  - DELETE 대신에 DROP TABLE/PARTITION
  - DELETE FROM … WHERE … LIMIT 1000;
  - binlog_row_image = minimal

- 복제 지연 방지 설정
  - binlog_transaction_dependency_tracking = **WRITESET**
  - replica_parallel_type = LOGICAL_CLOCK
  - replica-parallel-workers = 16
  - binlog_order_commits=OFF
  - replica_preserve_commit_order = 1(optional)

https://dev.mysql.com/blog-archive/improving-the-parallel-applier-with-writeset-based-dependency-tracking/

Performance Tuning Database Design

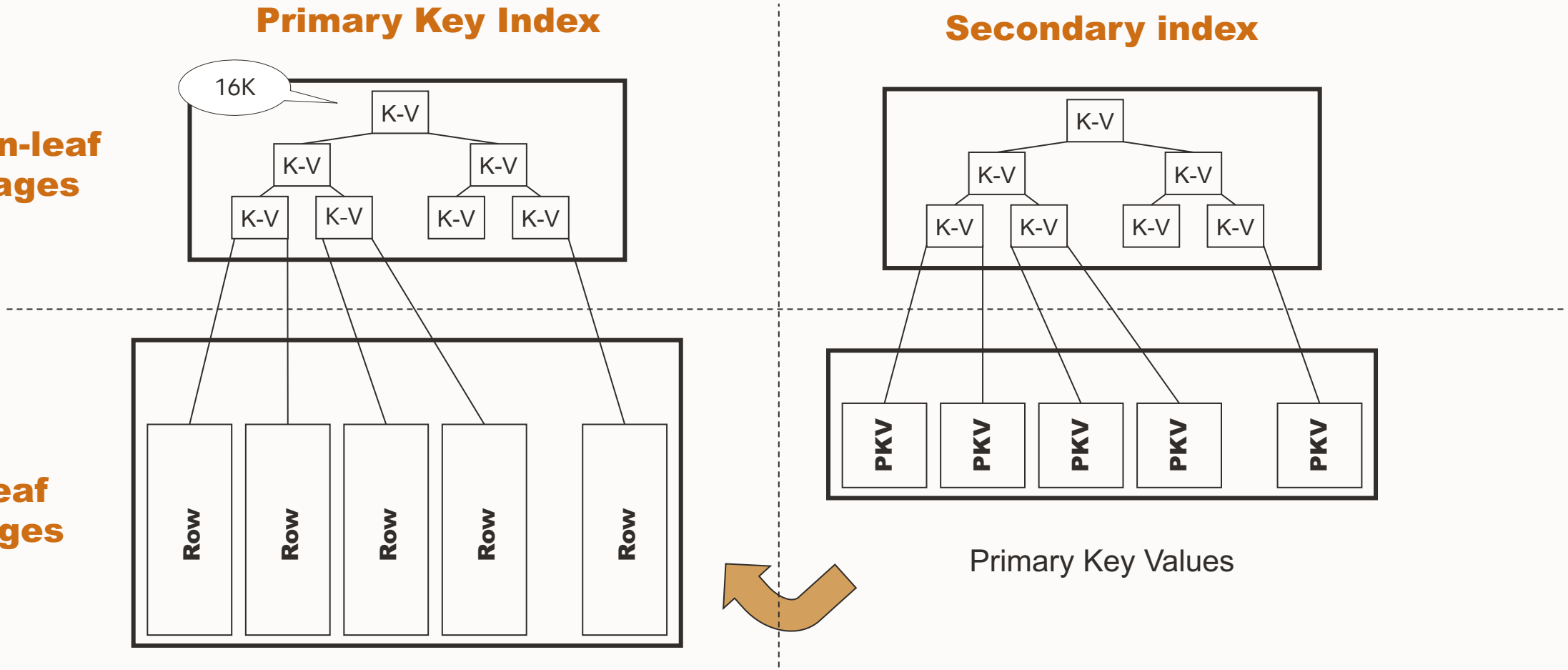
Database Design - Generic

- 효율적인 컬럼 타입을 선택
  - Unsigned 대신에 Signed – 매출 건수
  - DECIMAL(n, m) 대신에 INT – 금액
  - INET_NTOA(), INET_ATON() – IP 주소
  - String타입에 대해서 UTF8MB4 권장
    - ALTER TABLE t CONVERT TO CHARSET utf8mb4;
  - 상태 값은 CHAR(1) + CHECK (8.0.16)
    - CONSTRAINT ‘status_chk’ CHECK (status in (‘S’,’T’,’D’,’U’))
  - JSON 데이터를 TEXT대신에 JSON 타입으로 – 사용자 등록 테이블, 사용자 태그
    - JSOM 함수 지원
    - Functional 인덱스 지원
  - 항상 “NOT NULL” 지정

Database Design - InnoDB

- InnoDB 스토리지 엔진 강추!!!
- PK 선택을 신중하게 선택하고 항상 명시적으로 지정
- Lock 및 meta-data lock을 감소하기 위해 큰 트랜잭션을 피해야 함
  - max_execution_time 로 조회 쿼리 시간을 제어
  - 별도의 replica 서버에서 복잡하고 긴 쿼리 실행
  - OLTP 와 OLAP 워크로드를 분리할 것을 고려
- Foreign keys 는 성능을 저하 시킴
  - 데이터 변경에 대한 FK 검증.
  - FK 로 인한 meta-data locks이 발생
- 압축 옵션으로 공간 절약, I/O 효율성 향상
  - TPC(Transparent Page Compression): COMPRESSION=ZLIB|LZ4|NONE;
  - 이력 및 로그성 데이터, 용량이 크고, 성능에 크게 예민하지 않는 업무
  - https://dev.mysql.com/doc/refman/8.0/en/innodb-page-compression.html

InnoDB Clustered Index



Database Design - Indexing

- 너무 많은 indexes는 inserts/update/delete 작업에 성능 영향
- 복합 인덱스는 중복을 피해야 함
  - index key123 (col1,col2,col3)
  - index key12 (col1,col2) <- 필요 없음
  - index key1 (col1) <-- 필요 없음
- WHERE 절에 functions/expressions 을 피해야 함
- Covering Index 지원:
  - SELECT (a,b) FROM some_table where a=50;
  - 컬럼 (a,b) 에 인덱스를 생성하여 PK B-Tree 조회할 필요 없음(Extra: Using Index)
- 인덱스 컬럼 사이즈를 작게
  - 메모리 절약, 스캔 효율성 향상
  - Prefix 인덱스 활용하거나 MD5로 해시 키 생성
- WHERE/ORDER BY/GROUP BY 조건에 인덱스 컬럼 지정

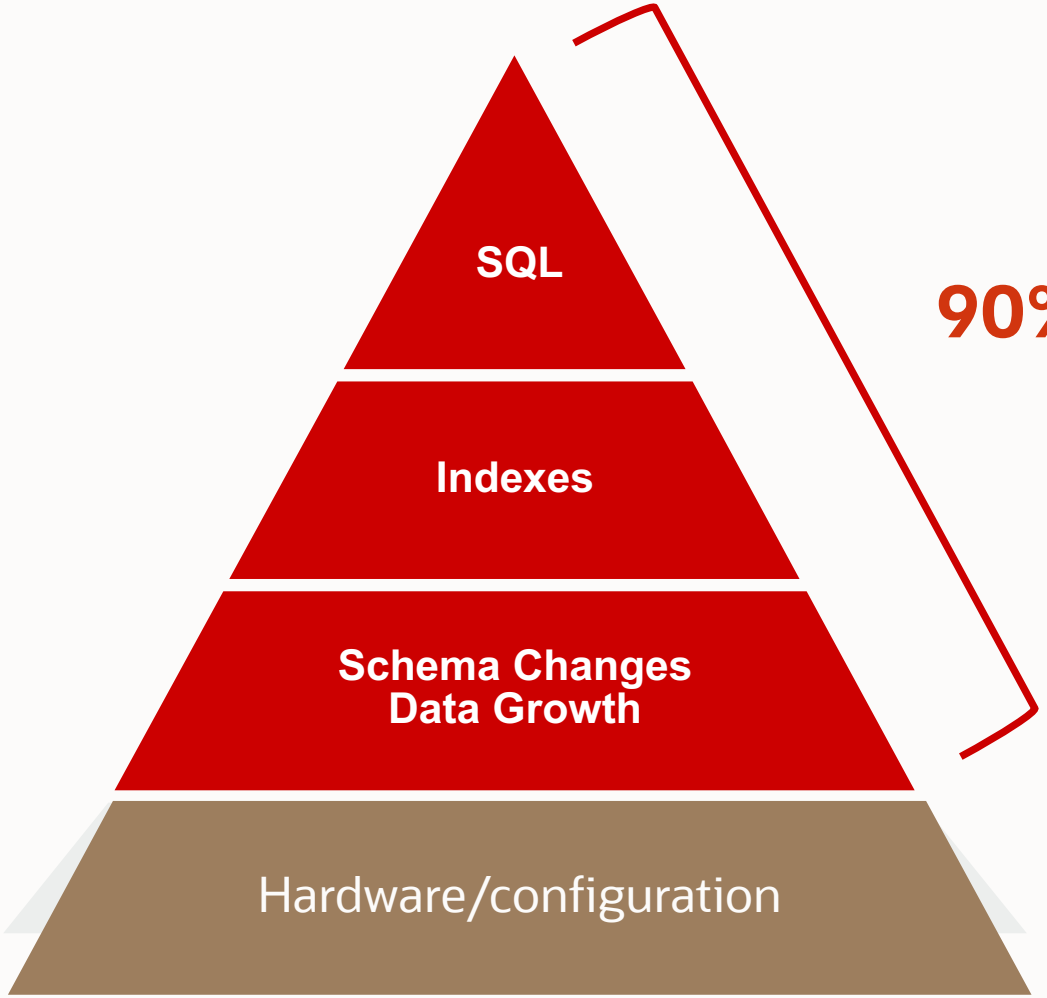
- Select count(*) from tradelog where **moth (t_modified) =7;**
- Select count(*) fro m tradelog where **t_modified >= '2016-7-1' and t_modified < '2016-8-1'**
t_modified >= '2017-7-1' and t_modified < '2017-8-1'

Do not let you database/tables grow out of control!

- 대용량 MySQL instances/schemas/tables을 관리하는 추가 코스트 (> 2TB)
  - 데이터 사이즈가 커짐에 따라 인덱스 구조가 더 넓어지고 깊어짐, 모든 SQL 작업에 영향
  - 사용 하지 않는 이력 데이터에 대해서 삭제하거나 아카이브 성능 저하
  - 신속하게 늘어나는 테이블에 대해서 대책을 세워야 함
  - 파티션 테이블을 사용하면 과거 데이터를 삭제하는데 성능 영향 줄임
    - 파티션 키가 아닌 컬럼에 대한 조회 성능 저하됨
  - Insert 성능을 위해 큰 테이블 (1억건 이상) 은 파티션 테이블로 I/O부하 분산
  - 성능 문제 뿐만 아니라 백업/복구 운영 작업의 문제

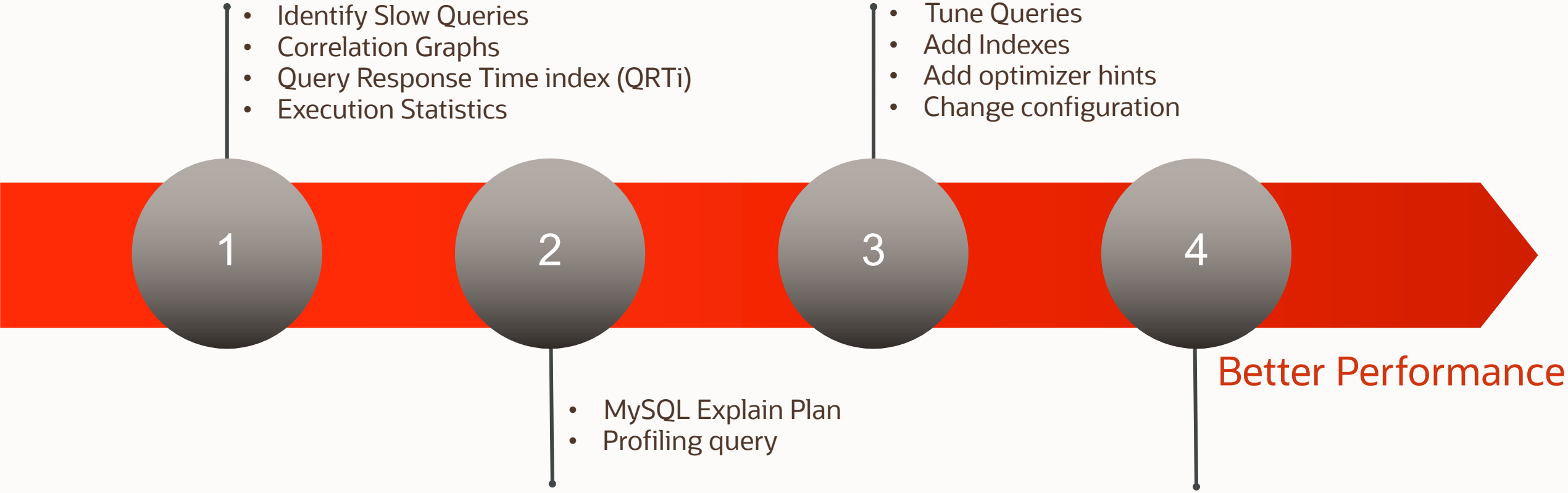
Performance Tuning
Query Tuning

Source of Database Performance Problems



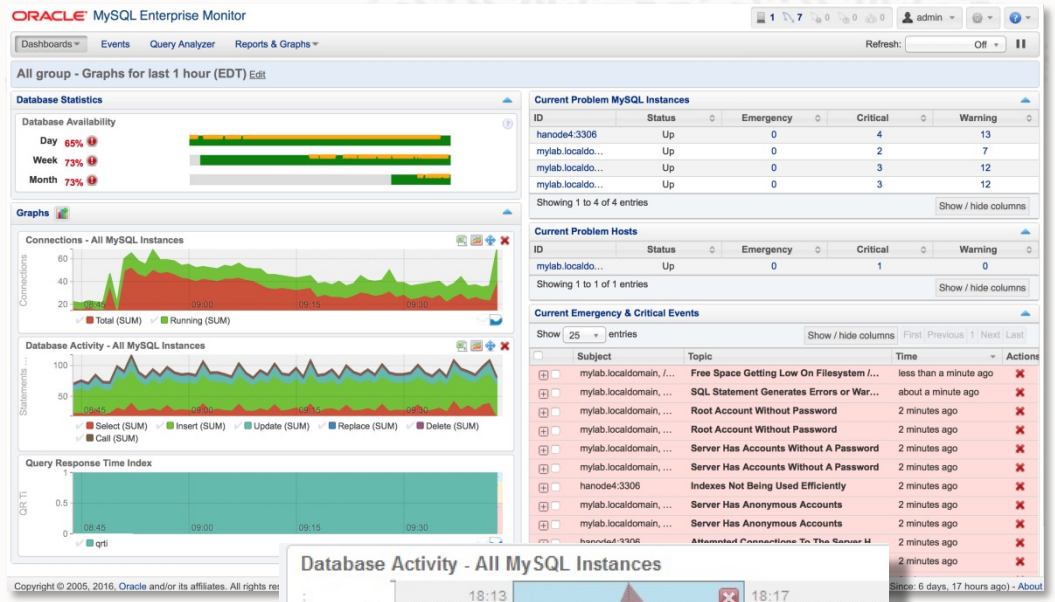
90% of Performance problems

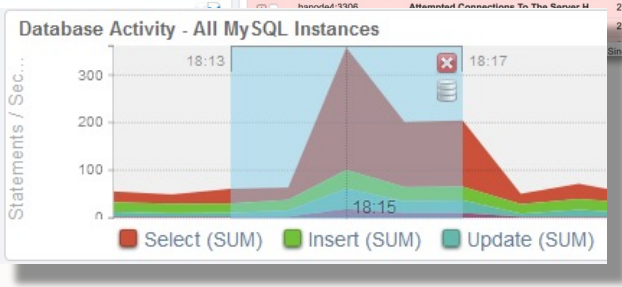
Query tuning from 10.000m

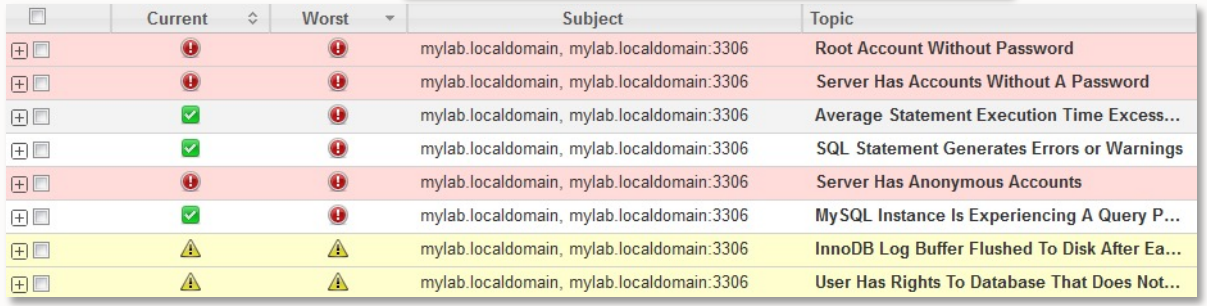


Monitoring

- 애플리케이션 지표를 확인
- Slow Query Log
- SHOW FULL PROCESSLIST;
- **performance_schema**.events_statements_summary_by_digest
- **sys**.statements_with_runtimes_in_95th_percentile
- **MySQL Enterprise Monitor "MEM"**
  - Query Analyzer 기능 : 수행 시간, 실행 통계 기반 이벤트 생성, 샘플 쿼리 및 분석
  - Best practice performance advisors 기능 : 잠재적인 성능 병목에 대한 해결책 제시







Analyzing queries

- EXPLAIN (옵티마이저의 실행 계획 확인)
- EXPLAIN ANALYZE (8.0.19에서 추가)
- Profiling기능으로 쿼리 자원 소모 확인
  - SET profiling=1; SHOW PROFILES;
  - INFORMATION_SCHEMA. PROFILING
- Optimizer trace (모든 가능한 Optimizer 플랜을 확인)
- 쿼리 목적을 분석하고 다시 작성할 수도
  - 예를 들면 subqueries를 joins으로 혹은 OR를 UNION으로

Optimization가이드: https://dev.mysql.com/doc/refman/8.0/en/optimization.html

Explain output

EXPLAIN Columns:

• select_type	SIMPLE,PRIMARY, Union, JOIN , Subquery, Derived, Union_result, Dependent subquery…
• table	테이블 명
• partitions	파티션 명
• **type**	**const(유니크 키), eq_ref(PK), ref(인덱스), range(인덱스),index(인덱스 스캔), all(테이블 스캔)**
• possible Keys	선택 가능한 인덱스
• key	실제 사용한 인덱스
• ref	인덱스와 비교한 컬럼
• rows	예측 건수
• filtered	조회 조건에 의해 필터링된 퍼센트수
• **extra**	**using where, using index, using temporary, using file sort, using join buffer…**

Get total “cost” of query by multiply all row counts in output.

Solving the problem!

- 쿼리 목적을 분석하고 다시 작성할 수 있는지 확인 ?
- 적당한 인덱스를 추가
- 인덱스 통계를 갱신: ANALYZE TABLE
- 옵티마이저 힌트를 사용:
  - Join-Order Optimizer Hints
  - Table-Level Optimizer Hints
  - Index-Level Optimizer Hints and more
  - /*+ ... */
- optimizer_switch
  - SELECT @@optimizer_switch\G

JOIN

- Index Nested-Loop Join

```
mysql> explain select * from t1 straight_join t2 on (t1.a=t2.a);
+----+-------------+-------+------------+------+---------------+------+---------+----------+------+----------+-------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref      | rows | filtered | Extra       |
+----+-------------+-------+------------+------+---------------+------+---------+----------+------+----------+-------------+
|  1 | SIMPLE      | t1    | NULL       | ALL  | a             | NULL | NULL    | NULL     |  100 |   100.00 | Using where |
|  1 | SIMPLE      | t2    | NULL       | ref  | a             | a    | 5       | test.t1.a |    1 |   100.00 | NULL        |
+----+-------------+-------+------------+------+---------------+------+---------+----------+------+----------+-------------+
```

- Block Nested-Loop Join(join_buffer_size)

```
mysql> mysql> explain select * from t1 straight_join t2 on (t1.a=t2.b);
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------------------------------------------+
| id | select_type | table | partitions | type | possible_keys | key  | key_len | ref  | rows | filtered | Extra                                              |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------------------------------------------+
|  1 | SIMPLE      | t1    | NULL       | ALL  | a             | NULL | NULL    | NULL |  100 |   100.00 | NULL                                               |
|  1 | SIMPLE      | t2    | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 1000 |    10.00 | Using where; Using join buffer (Block Nested Loop) |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+----------------------------------------------------+
```

- T2테이블에 인덱스를 추가, NLJ을 BKA(Batched Key Access Join)로 변환
  - set optimizer_switch='mrr=on,mrr_cost_based=off,batched_key_access=on';
  - Using join buffer (Batched Key Access)

Hash JOIN

- Hash Join 지원(8.0.20부터 BNL대체)
  - Inner non-equi-join
  - Semijoin:
  - Antijoin:
  - Left outer join:
  - Right outer join
  - join_buffer_size 로 메모리 설정

```
mysql> EXPLAIN FORMAT=TREE
    -> SELECT * FROM t1
    ->     JOIN t2 ON (t1.c1 = t2.c1)
    ->     JOIN t3 ON (t2.c1 < t3.c1)\G
*************************** 1. row ***************************
EXPLAIN: -> Filter: (t1.c1 < t3.c1)  (cost=1.05 rows=1)
    -> Inner hash join (no condition)  (cost=1.05 rows=1)
        -> Table scan on t3  (cost=0.35 rows=1)
        -> Hash
            -> Inner hash join (t2.c1 = t1.c1)  (cost=0.70 rows=1)
                -> Table scan on t2  (cost=0.35 rows=1)
                -> Hash
                    -> Table scan on t1  (cost=0.35 rows=1)
```

Performance Top 10 tips!

Tip 1: Monitor your databases

Without monitoring you are blind.
Monitor the database and OS.
Monitor over time, monitor query statistics.

Tip 2: Make small incremental changes

Change one parameter, test/monitor,
change the next one.
Document changes in behaviour.

Tip 3: Use InnoDB

MyISAM have table level locks.
InnoDB is ACID.
InnoDB scales on modern HW.
Online DDL

Tip 4: Understand “explain”

Query tuning is the holy grail of performance tuning and explain is the most important tool for query tuning.

Tip 5: Delete/archive old data

Large instances/tables are one of the most common problems for slow queries.

Tip 6: Use SSD/NVMe drives

If your working-set of data does not fit into memory make sure you have a fast disk system (for random IO operations).

Tip 7: Use MySQL 8.0

MySQL 8.0 scales better than older versions of MySQL, the optimizer have also been improved with hash-joins and histograms.

Tip 8: Avoid long running transactions

Huge risk of locking contention, separate your OLTP and OLAP functionality on separate MySQL instances.

Tip 9: Use foreign keys with care

Foreign keys are good for consistency but in general bad for performance and takes more locks.

Tip 10: Avoid complex SQL

Hard to understand, hard to optimize and in many cases slow.

MySQL Enterprise Edition

고급 기능

- 확장성
- 고 가용성
- 고급 보안 기능

관리 툴

- 모니터링 툴
- 백업/복구 툴
- 개발/관리 툴

기술 지원

- 기술 지원
- 컨설팅 서비스
- Oracle 인증

MySQL Enterprise Support

- 최대의 MySQL 엔지니어 및 지원 조직
- MySQL 개발자의 후선 지원
- 한국어 지원
- 핫 픽스 & 유지 보수 릴리스
- 24x7x365
- 제한 없는 SR 건수
- 컨설팅 서비스
- 글로벌 지원 팀

Get immediate help for any MySQL issue, plus expert advice

MySQL Enterprise Consultative Support

- 원격 트러블 슈팅
- 복제 리뷰
- 파티션 리뷰
- 스키마 리뷰
- 쿼리 리뷰
- **성능 튜닝**
- ...and more

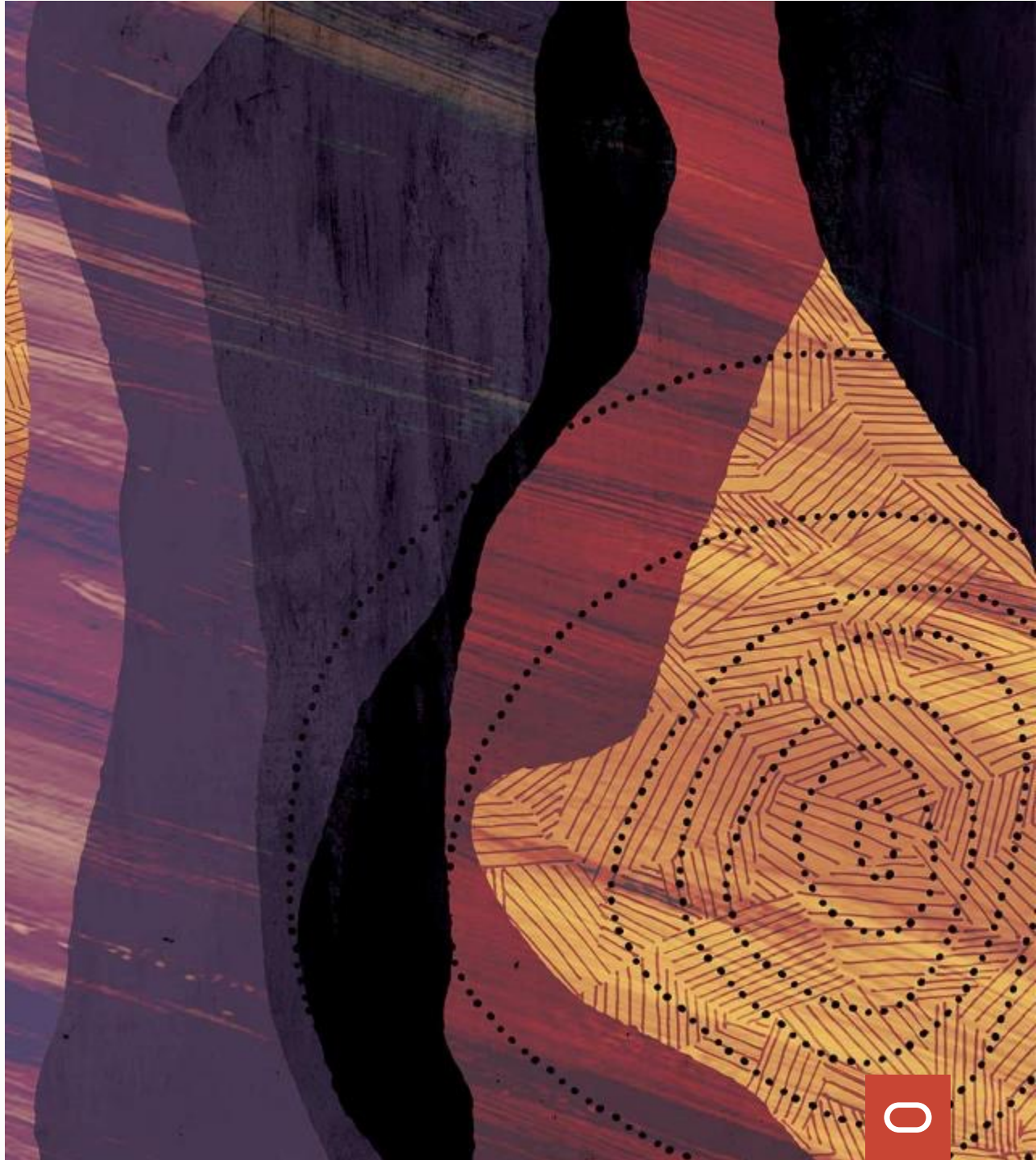
Get immediate help for any MySQL issue, plus expert advice

More information

- https://dev.mysql.com/doc/refman/8.0/en/optimization.html
- https://dev.mysql.com/doc/refman/8.0/en/optimizing-innodb.html
- http://dimitrik.free.fr/blog/
- https://blogs.oracle.com/mysql/
- https://lefred.be/
- 한국 MySQL 사용자 그룹
  - https://www.facebook.com/groups/623067261102382/

감사합니다 !

Huishan.piao@oracle.com



ORACLE