SSD 성능시험

작성일 : 2010/07/28

1. 개요

DBMS는 데이터의 저장과 관리를 목적으로 한다. 대용량 DB의 저장매체로서 HDD가 대중적으로 쓰이고 있지만, I/O Bound 워크로드에서 HDD(I/O)에서의 병목현상으로 인해 성능 저하가 발생한다. 본 문서는 새로운 저장매체로 각광받고 있는 SSD를 주 저장매체로 사용할 때 DBMS의 성능향상 정도를 시험한 내용을 기술한다.

1.1. 시험 목적

CUBRID 와 MySQL 을 HDD 와 SSD를 사용하는 장비에 설치하여, 성능 향상 정도(TPS기준)를 알아본다.

1.2. 시험 장비 및 시험 환경

시험에 사용한 HDD장비 및 SSD장비의 스펙은 다음과 같다. HDD vs SSD의 성능향상 정도를 정확하게 확인하기 위해서는 저장매체만 HDD와 SSD로 차이가 있어야 하지만, 시험 장비 확보가 어려워 비교적 스펙이 비슷한 장비 2대를 사용하였다.

	HDD 장비	SSD 장비
OS	CentOS 5.2 x86_64	CentOS 5.3 x86_64
CPU	Xeon 2.5GHz Quad * 2	Xeon 2.26Ghz (L5640) 2way 6core
Memory	2G * 4	8G
Disk	Raid 0+1 SAS 300G * 6	RAID0+1 100G * 6

HDD장비와 SSD장비 각각에 MySQL과 CUBRID를 설치 하였다. 시험에 사용한 CUBRID/MySQL 의 버전은 다음과 같다.

- CUBRID 2008 R3.0
- MySQL 5.1.47 (innoDB)

CUBRID와 MySQL 에 사용된 configuration 은 다음과 같다.(동일하게 4G 데이터 버퍼를 설정함)

- cubrid.conf

```
[service]
service=server,broker,manager

[common]
data_buffer_pages=25000
sort_buffer_pages=16
log_buffer_pages=50
lock_escalation=100000
lock_timeout_in_secs=-1
deadlock_detection_interval_in_secs=1
checkpoint_interval_in_mins=720
isolation_level="TRAN_REP_CLASS_UNCOMMIT_INSTANCE"
cubrid_port_id=15097
max_clients=50
```

auto_restart_server=yes
replication=no
java_stored_procedure=no

checkpoint_every_npages=100000000
data_buffer_pages=262144
error_log_level=notification
communication_histogram=yes
num_LRU_chains=200
async_commit=yes
group_commit_interval_in_msecs=1000

- my.cnf
[client]
socket = /home1/mysql/mysql/tmp/mysql.sock

[mysqld]
user = mysql
port = 3306
basedir = /home1/mysql/mysql
datadir = /home1/mysql/mysql/data
tmpdir = /home1/mysql/mysql/tmp
socket = /home1/mysql/mysql/tmp/mysql.sock

default-character-set = utf8
default_table_type = InnoDB
skip_name_resolve

back_log = 100
max_connections = 500
max_connect_errors = 999999
max_allowed_packet = 16M
max_heap_table_size = 64M
tmp_table_size = 64M
binlog_cache_size = 1M
thread_cache_size = 128

table_cache = 1024
sort_buffer_size = 8M
join_buffer_size = 8M
read_buffer_size = 2M
read_rnd_buffer_size = 16M
query_cache_size = 64M
query_cache_limit = 2M
# MyISAM options
key_buffer_size = 32M
bulk_insert_buffer_size = 64M
myisam_sort_buffer_size = 128M
myisam_max_sort_file_size = 10G
myisam_max_extra_sort_file_size = 10G
myisam_repair_threads = 1

```
myisam_recover
ft_min_word_len = 4

# INNODB options
innodb_buffer_pool_size = 4G                # 50 ~ 70% of main memory
innodb_log_buffer_size = 8M
innodb_additional_mem_pool_size = 16M
innodb_data_file_path = ibdata1:100M:autoextend
innodb_file_per_table
innodb_log_file_size = 256M
innodb_log_files_in_group = 3
innodb_support_xa=0
innodb_thread_concurrency = 16
innodb_lock_wait_timeout = 60
innodb_flush_log_at_trx_commit = 0        #   0 for slave, 1 for master

# Loging Configuration
log-bin=mysql-bin
expire_logs_days=5
log_warnings
log_slow_queries
log_slow_admin_statements
long_query_time = 2
log_long_format

# Replication setting
server-id          = 1
```

1.3. 시험 시나리오

- 시험을 위한 Table Schema

성능 시험을 위해서 아래와 같은 테이블 스키마를 이용하여 tbl_200 ~ tbl_239 까지 40개의 테이블을 생성하였다.

```
CREATE TABLE tbl_200;

ALTER CLASS tbl_200 ADD ATTRIBUTE

        id character varying(20) NOT NULL,

        seq integer NOT NULL,

        col3 character varying(16) NOT NULL,

        col4 character varying(5) NOT NULL,

        col5 character varying(50) NOT NULL,

        col6 character varying(1000),

        col7 character varying(300) NOT NULL,

        col8 character varying(150),

        col9 timestamp NOT NULL,
```

```
        col10 smallint DEFAULT 0 NOT NULL,

        col11 timestamp NOT NULL,

        col12 character varying(15) NOT NULL,

        col13 character(1) NOT NULL,

        col14 character(1) NOT NULL,

        col15 timestamp DEFAULT timestamp '04:25:44 PM 07/30/2009' NOT NULL;


ALTER CLASS tbl_200 ADD ATTRIBUTE

        CONSTRAINT "iuk_tbl" UNIQUE(id, col3, col4, col5),

        CONSTRAINT "ipk_tbl" PRIMARY KEY(id, seq);


CREATE INDEX ink1_tbl ON tbl_200 (id, col9 DESC, col14);
```

위의 테이블 스키마를 사용하여 시험 테이블을 생성하고, HDD/SSD 장비에서 다음 3가지 시험을 진행한다.

- 40개의 테이블에 총 2천5백만건의 데이터를 입력한 DB를 생성한 다음, 30분 동안 Insert Full 부하를 주었을 때 성능측정
- 40개의 테이블에 총 6천4백만건의 데이터를 입력한 DB를 생성한 다음, CPU Bound Select 부하를 주었을 때 성능측정.
- 40개의 테이블에 총 6천4백만건의 데이터를 입력한 DB를 생성한 다음, I/O Bound Select 부하를 주었을 때 성능측정

위 3가지 모두 40개의 쓰레드에서 부하를 주도록 한다. Insert 부하는 1개의 INSERT 질의로 구성된 트랜잭션이고, SELECT 부하는 primary key, unique index, non-unique index 각각을 사용하는 3개의 SELECT 질의로 구성된 트랜잭션이다.

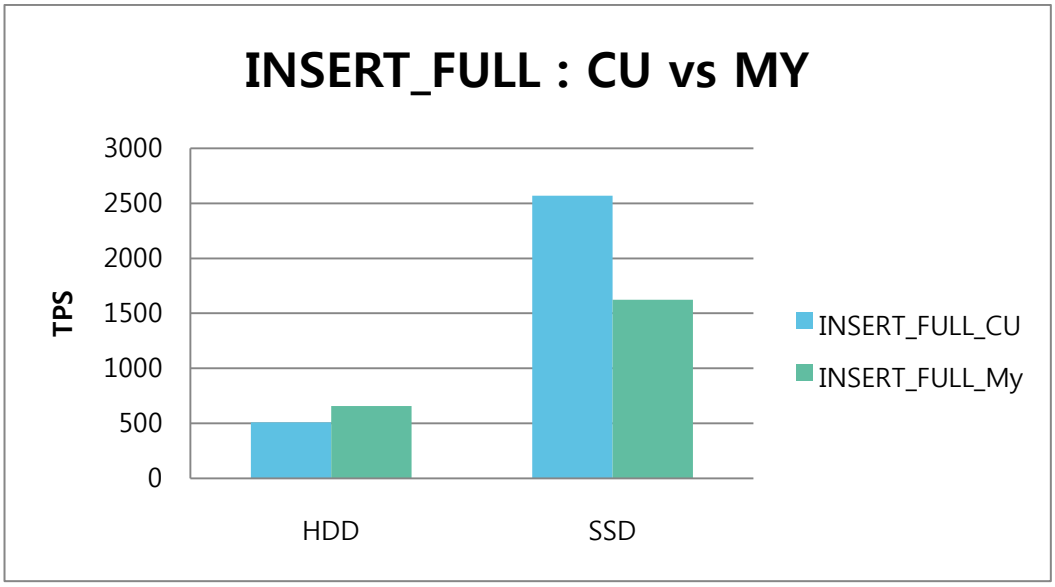
2. 시험 내용 및 결과

2.1. I/O Bound Insert workload 시험

40개의 테이블 각각에 약 625,000건의 데이터가 들어 있는 DB(총 2천5백만건)를 생성한 후 Insert Full 부하를 주는 성능 시험을 HDD/ SSD 장비 각각에서 30분동안 수행하였다. 다음은 수행 결과이다.

	tps		IO Read		IO Write		% UTIL	
	INSERT_FULL_CU	INSERT_FULL_My	INSERT_FULL_CU	INSERT_FULL_My	INSERT_FULL_CU	INSERT_FULL_My	INSERT_FULL_CU	INSERT_FULL_My
HDD	**508**	656	**772**	828	**740**	808	**100%**	100%
SSD	**2569**	1624	**3872**	1968	**4320**	2156	**100%**	75~90%

- TPS 변화



위 Insert Full 부하에서,

- CUBRID 는 SSD장비에서 HDD 대비 약 5배의 TPS 향상이 있었으며,
- MySQL 은 약 2.5 배의 TPS 향상이 있다.(MySQL은 %UTIL이 100%가 아니므로 추가적인 성능향상의 여지가 있어 보인다.)

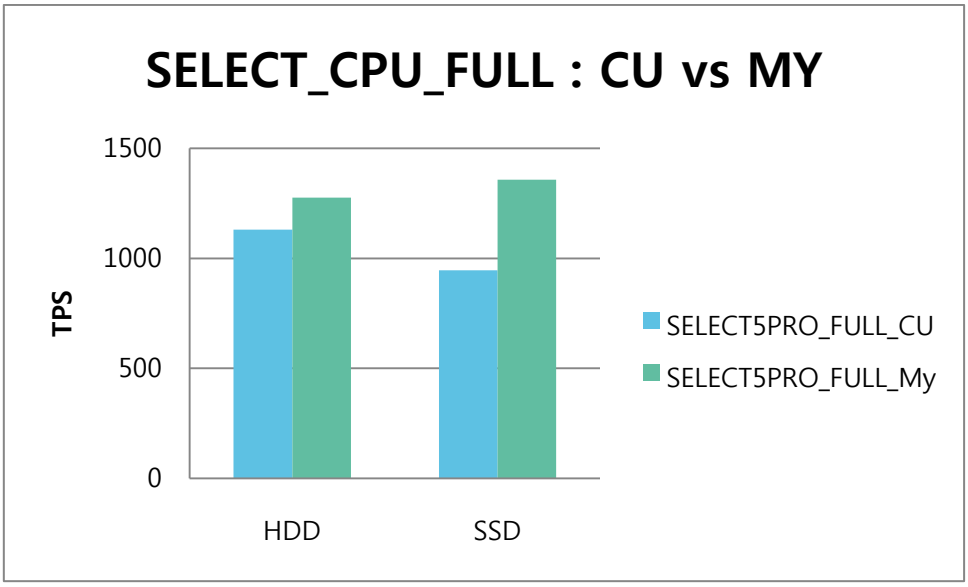
2.2. CPU Bound Select workload 시험

40개의 테이블 각각에 약 1,600,000건의 데이터가 들어 있는 DB(총 6천4백만건)를 생성한 후 CPU Bound 부하를 주는 성능 시험을 HDD/SSD 장비 각각에서 10분 동안 수행하였다. 이 워크로드는 Select 질의를 수행하는데 필요한 모든 Page가 메모리 Buffer에 올라 올 수 있도록 질의문의 검색 범위를 좁혀서 작성하여 Buffer Hit Ratio 가 100%를 유지하도록 하였다. 이 워크로드는 I/O가 발생하지 않기 때문에 HDD vs SSD 장비에서 I/O 성능을 제외한 부분의 성능 차이를 관찰할 목적으로 수행한 것이다.

다음은 시험 결과이다.

	tps		% UTIL	
	SELECT5PRO_FULL_CU	SELECT5PRO_FULL_My	SELECT5PRO_FULL_CU	SELECT5PRO_FULL_My
HDD	**1131**	1276	**0%**	0%
SSD	**945**	1357	**0%**	0%

- TPS 변화



I/O 가 발생하지 않는 상황에서 CUBRID는 약 17%정도의 성능 저하가 있었고, MySQL 은 약 6%정도의 성능 향상이 있었다.
I/O를 제외 부분에서 두 장비 스펙의 성능차이는 이 정도로 보여진다.

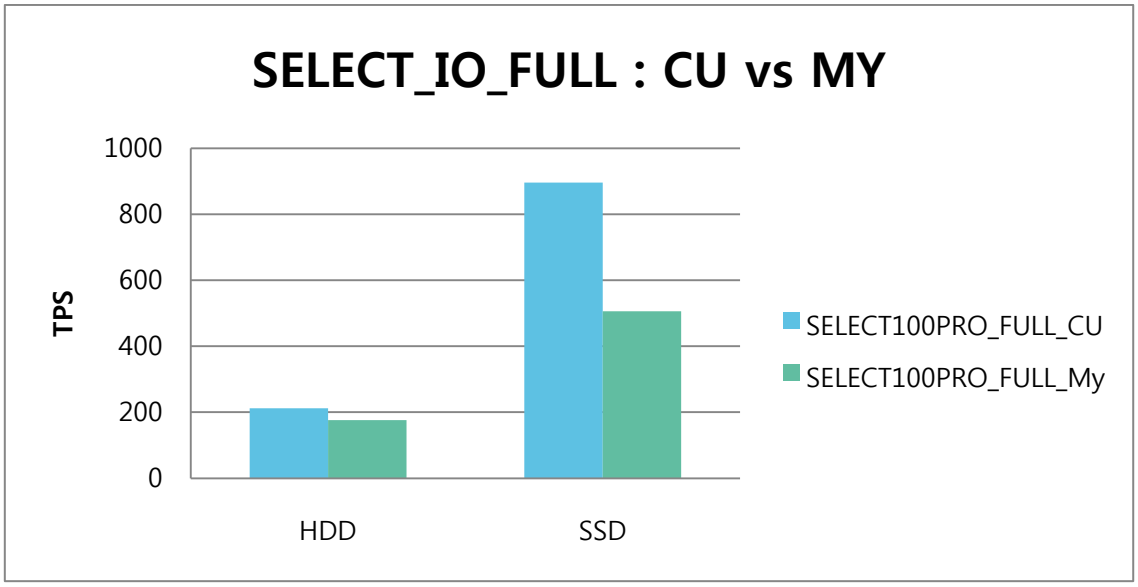
2.3. I/O Bound Select workload 시험

40개의 테이블 각각에 약 1,600,000건의 데이터가 들어 있는 DB(총 6천4백만건)를 생성한 후 I/O Bound 부하를 주는 성능 시험을 HDD/SSD 장비 각각에서 10분동안 수행하였다. 이 워크로드는 Select를 수행하는데 필요한 모든 Page들이 메모리 Buffer에 올라오지 못하도록 질의문의 검색 범위를 넓혀서 빈번한 Page 교체가 발생하도록 작성되었다. 그래서 이 워크로드는 극심한 I/O작업을 발생시키는 것이 특징이다.

다음은 시험 결과이다.

	tps		% UTIL	
	SELECT100PRO_FULL_CU	SELECT100PRO_FULL_My	SELECT100PRO_FULL_CU	SELECT100PRO_FULL_My
HDD	**212**	176	**100%**	100%
SSD	**896**	506	**100%**	100%

- TPS 변화



위 I/O Bound SELECT 부하에서,

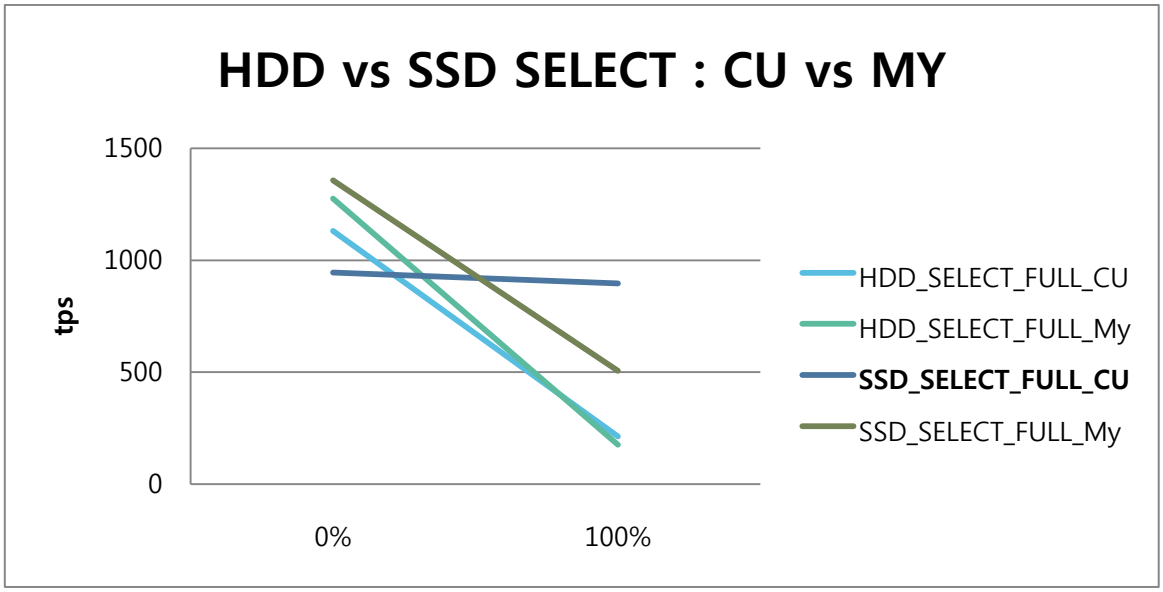
- CUBRID 는 SSD장비에서 HDD 대비 약 4.2배의 TPS 향상이 있었으며,
- MySQL 은 SSD 장비에서 HDD 대비 약 2.8 배의 TPS 향상이 있다.

두 DBMS모두 SSD를 사용할 경우에 성능향상이 있음을 알 수 있다.

2.4. Select 시험의 정리

다음은 위의 두 Select 시험 결과를 하나의 표로 정리한 것이다.

	Workload Type	tps		% UTIL	
		HDD_SELECT_FULL_CU	HDD_SELECT_FULL_My	HDD_SELECT_FULL_CU	HDD_SELECT_FULL_My
HDD	CPU BOUND	**1131**	1276	**0%**	0%
	IO BOUND	**212**	176	**100%**	100%
		SSD_SELECT_FULL_CU	SSD_SELECT_FULL_My	SSD_SELECT_FULL_CU	SSD_SELECT_FULL_My
SSD	CPU BOUND	**945**	1357	**0%**	0%
	IO BOUND	**896**	506	**100%**	100%



위 그래프의 왼쪽은 CPU bound 상황이고 오른쪽은 I/O bound 상황이다. 모든 경우에 CPU Bound 작업이 I/O Bound 보다 TPS 수치가 높다. 이것은 I/O 가 DBMS의 성능을 저하시키는 주요 원인이라는 것을 보여준다. 가장 특이한 점은 CUBRID는 SSD 장비에서 CPU Bound작업과 I/O Bound 작업간의 성능 변화가 크지 않다는 것이다. 즉, SSD 장비의 이점을 최대한 활용하고 있는 것으로 보인다. (시험에 사용된 SSD 장비의 random 엑세스가 매우 빠르기 때문으로 판단된다.)

3. 결론

위 시험에서 MySQL, CUBRID 둘다 SSD에서 TPS 수치가 올라가는 것을 확인 할 수 있다. I/O Bound 워크로드에서 CUBRID는 약 4.2배의 TPS 향상 효과가 있었으며, MySQL 은 2.8배의 TPS 향상 효과가 있다. 이 시험에서는 CUBRID 든 MySQL 이든 SSD 장비의 특성을 고려한 별도의 DB 튜닝을 수행하지 않았다. 때문에 SSD장비에서 어느 DBMS가 더 적합한가는 논의 대상이 아니다. 다만, CUBRID/MySQL 모두 SSD장비를 통해서 I/O bound 작업의 성능 향상이 가능하다는 결론은 얻을 수 있겠다. 향후에 하드웨어 스펙과 OS가 완전히 동일한 장비를 사용하고(저장매체만 HDD와 SSD로 다른 장비를 설정) DB configuration 또한 CUBRID/MySQL 모두 최적으로 설정한 상태에서 보다 다양한 시험을 수행해 볼 수 있다면, 더 많은 흥미로운 결과를 얻을 수 있을 것으로 보인다.