

SNS 유형 서비스에서 조회 성능 비교 (CUBRID 8.4.0 vs MySQL 5.1)

DBMS Dev Lab
June 2011



테스트 DB 정보

- 사용하는 테이블
 - friends
 - posts
- 사용하는 인덱스
 - friends 테이블
 - unique(user_id,friend_id)
 - posts 테이블
 - primary key(id)
 - index(reg_date)
 - index(friend_id,reg_date)

테스트 사용자 그룹

- 사용자의 친구 수에 따라 그룹 1, 2, 3으로 나눔.
- DB 테이블
 - user_group_1
 - 친구 수가 50명 이하인 사용자의 ID 1,442,329건 저장
 - user_group_2
 - 친구 수가 51명~2000명인 사용자의 ID 85,568건 저장
 - user_group_3
 - 친구 수가 2001명 이상인 사용자의 ID 836건 저장
- 각 테이블에는 sequential하게 생성된 ID와 사용자 ID가 mapping 되어 있고, sequential 하게 생성한 ID를 랜덤하게 선택해서 사용자 ID를 가져가도록 구성되어 있음.

테스트 사용자 그룹

```
mysql> create table user_group_1 (id int auto_increment primary key, user_id int, num_friends int);  
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> create table user_group_2 (id int auto_increment primary key, user_id int, num_friends int);  
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> create table user_group_3 (id int auto_increment primary key, user_id int, num_friends int);  
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> insert into user_group_1(user_id, num_friends) select user_id, count(friend_id) from friends  
group by user_id having count(friend_id) < 51;  
Query OK, 1442329 rows affected (38.28 sec)  
Records: 1442329 Duplicates: 0 Warnings: 0
```

```
mysql> insert into user_group_2(user_id, num_friends) select user_id, count(friend_id) from friends  
group by user_id having count(friend_id) > 50 and count(friend_id) < 2001;  
Query OK, 85568 rows affected (28.95 sec)  
Records: 85568 Duplicates: 0 Warnings: 0
```

```
mysql> insert into user_group_3(user_id, num_friends) select user_id, count(friend_id) from friends  
group by user_id having count(friend_id) > 2000;  
Query OK, 836 rows affected (27.89 sec)  
Records: 836 Duplicates: 0 Warnings: 0
```

UNION 트랜잭션

1. 사용자 선택

SELECT user_id FROM user_group_1
WHERE id = ? (random!)

user_group_#

2. 사용자의 친구 100명 가져오기

SELECT friend_id FROM friends
WHERE user_id = ? LIMIT 100

friends

3. 친구들의 글 20개에 대한 ID 가져오기

(SELECT id, friend_id, reg_date FROM posts
WHERE friend_id = ?
ORDER BY reg_date DESC LIMIT 20)
UNION (SELECT ...)
ORDER BY reg_date DESC LIMIT 20

posts

4. 친구들의 글 20개 가져오기

SELECT * FROM posts
WHERE id = ? OR id = ? OR ... OR id = ?
ORDER BY reg_date DESC LIMIT 20

posts

IN 트랜잭션

1. 사용자 선택

SELECT user_id FROM user_group_1
WHERE id = ? (random!)

user_group_#

2. 사용자의 친구 100명 가져오기

SELECT friend_id FROM friends
WHERE user_id = ? LIMIT 100

friends

3. 친구들의 글 20개 가져오기

(SELECT * FROM posts
WHERE friend_id IN (?, ?, ..., ?)
ORDER BY reg_date DESC LIMIT 20)

posts

테스트 환경

- CPU: Xeon 2.5 GHz Quad*2
- HDD: RAID 0+1 SAS 200G*6
- Memory: 8G
- OS: Linux CentOS 5.2 x86_64
- MySQL version: 5.1
- CUBRID version: 2008 R4.0
- Buffer configuration: 2G
- Test execution time: 10분
- Test tool: NBench

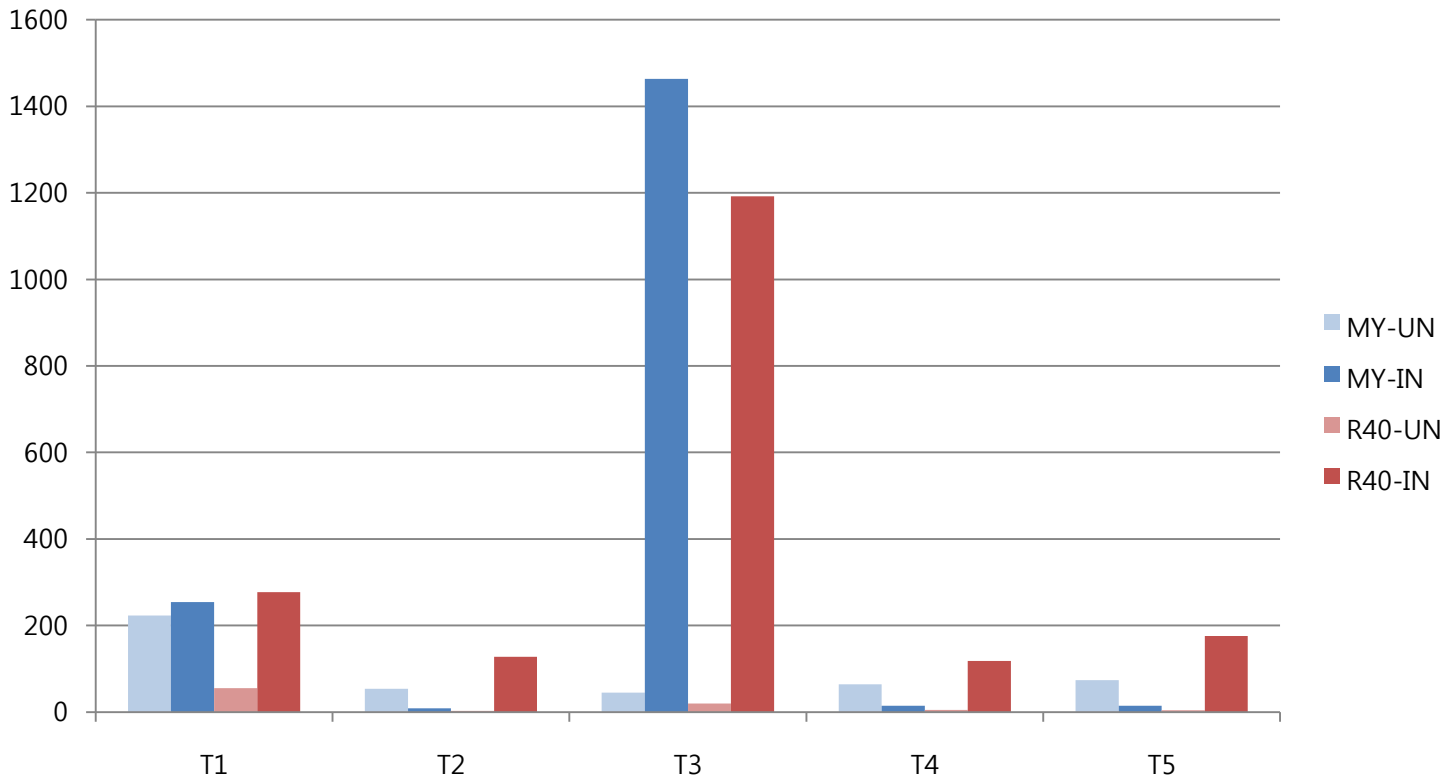
시나리오 요약

- 사용자의 친구 수 기준으로 다음과 같은 시나리오로 구분
 - T1: 친구 수 50명 이하인 사용자들이 “내친구글 전부보기”를 하는 경우
 - T2: 친구 수 51~2000명인 사용자들이 “내친구글 전부보기”를 하는 경우
 - T3: 친구 수 2001명 이상인 사용자들이 “내친구글 전부보기”를 하는 경우
 - T4: T1, T2, T3 시나리오가 40%, 50%, 10%로 mix된 경우(실제 SNS와 유사한 구성 비율)
 - T5: T1, T2, T3 시나리오가 10%, 50%, 40%로 mix된 경우
- 시나리오 수행 시, 한 트랜잭션 내에서 친구 목록 가져오기, 친구 글 가져오기를 모두 수행함
- 각 시나리오에서 사용되는 트랜잭션은 IN 쿼리를 사용하는 것과 UNION 쿼리를 사용하는 것으로 구분되어 있음

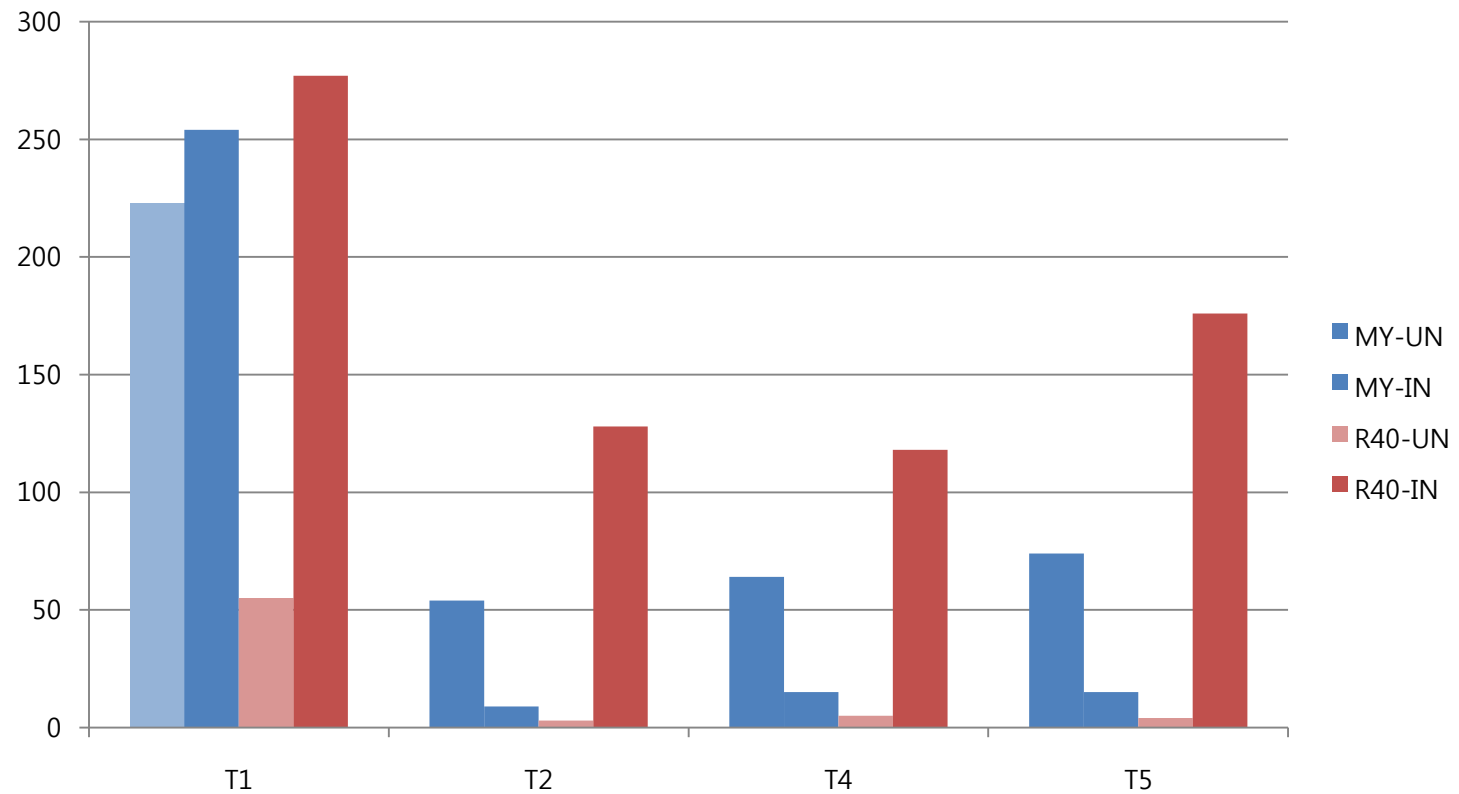
테스트 결과

TPS	MySQL 5.1		R4.0		R3.1	
	UNION	IN	UNION	IN	UNION	IN
T1	223	254	55	277	48	3
T2	54	9	3	128	12	5
T3	45	1463	20	1192	4	0.6
T4	64	15	5	118	0.007	0
T5	74	15	4	176	0.05	0

테스트 결과 비교(T1,T2,T3,T4,T5)



테스트 결과 비교(T1,T2,T4,T5)



Stat 비교

- `SELECT * FROM posts WHERE friend_id IN (?, ?, ..., ?)`
`ORDER BY reg_date DESC LIMIT 20;`
- R3.1
 - Num_data_page_fetches = 229316
 - Num_data_page_dirties = 206182
 - Num_data_page_ioreads = 121851
- R4.0
 - Num_data_page_fetches = 378
 - Num_data_page_dirties = 16
 - Num_data_page_ioreads = 135

결과 분석

- 현실성 있는 시나리오인 T4의 경우 MySQL UNION 테스트에 비해 R4.0 IN이 2배 정도 성능이 좋음
- T3의 경우 buffer hit ratio가 거의 100에 가깝기 때문에 성능이 다른 시나리오 비해 높은 편임
- 2008 R4.0 성능 분석
 - Key limit 효과로 I/O량이 약 50배 정도 감소함
 - Key limit가 적용된 상태에서는 in-place sorting이 약 10배 정도 I/O량 감소를 가져옴
 - in-place sorting을 하지 않으면 T4가 6 TPS 나옴. UNION 테스트 결과와 비슷해짐
 - UNION의 경우 sorting 중에 부분 결과를 반복적으로 temp file로 생성하는 부분이 성능에 영향을 주고 있는 것으로 판단됨
 - dirty page 수가 많음