HP OpenSource ブループリント

MySQL Server 5.0 サイジング情報

Red Hat Enterprise Linux 4

ver 1.5

invent

はじめに

本書では、MySQL サーバのサイジングを Red Hat Enterprise Linux 4 上で行なうための情報を提示します。
データベースのサイジングには様々な方法がありますが、本書では対象システムと同様のクライアントサーバから負荷テストを実施してサイジングする方法について述べます。

負荷試験を行うツールとして、フリーの Super Smack を用いました。実運用で使用されるような大きなデータを試験するには、Super Smackを一部修正する必要がありますので、修正方法や試験方法についても記述します。また、修正したSupersmackを用いて実機にて基礎的なデータを用いた試験を実施しましたので、サイジングの基礎データとして御活用下さい。

本書の範囲

本書では以下の項目について説明します。

- MySQL サーバのインストール手順の概要
- ベンチマークツール Super Smack の実行環境構築方法および実行手順
- ベンチマーク結果および考察

本書では以下の項目については範囲外です。
- RHEL4 のインストール方法

本書の構成

第 1 章 システム構成
本書でベンチマークを実施したシステムの構成です。

第 2 章 ベンチマーク実行環境のセットアップ
MySQL サーバとベンチマークツール SuperSmack の実行環境構築手順を紹介します。
SuperSmack の修正方法も御紹介します。

第 3 章 ベンチマークの実行手順
SuperSmack を使用して、MySQL サーバに負荷をかける手順をついて紹介します。

第 4 章 MySQL ベンチマーク
ベンチマークを実行し、得られた結果と考察について説明します。

付録 1 file_size_equiv について
Super Smack を使用してベンチマークを実行するにあたり、重要となるパラメータ file_size_equiv について説明します。

付録 2 smack ファイルのサンプル
今回のベンチマークで使用した Super Smack の設定ファイルのサンプルです。

目次

1. システム構成

1.1. システム構成

本書で構築するベンチマーク環境は、2 台の Proliant を使用し、クライアントにベンチマークツールをインストールして、サーバ側の MySQL Enterprise Server に負荷をかけます。

サーバ

Hardware	HP Proliant DL380G5
CPU	Intel Xeon E5320(1.86GHz) 4Core × 1
Memory	4GB
OS	Red Hat Enterprise Linux 4 update 4 (x86)
MySQL	MySQL Enterprise Server 5.0.36 (Linux, i686, glibc-2.3, noRPM 版)
Hostname	dl380g5node1

クライアント

Hardware	HP Proliant DL380G5
CPU	Intel Xeon E5320(1.86GHz) 4Core × 2
Memory	16GB
OS	Red Hat Enterprise Linux 4 update 4 (x86)
Benchmark Tool	super-smack-1.3
Hostname	dl380g5node2

ネットワーク

Network	HP Procurve 2724　ギガビットイーサネットスイッチ

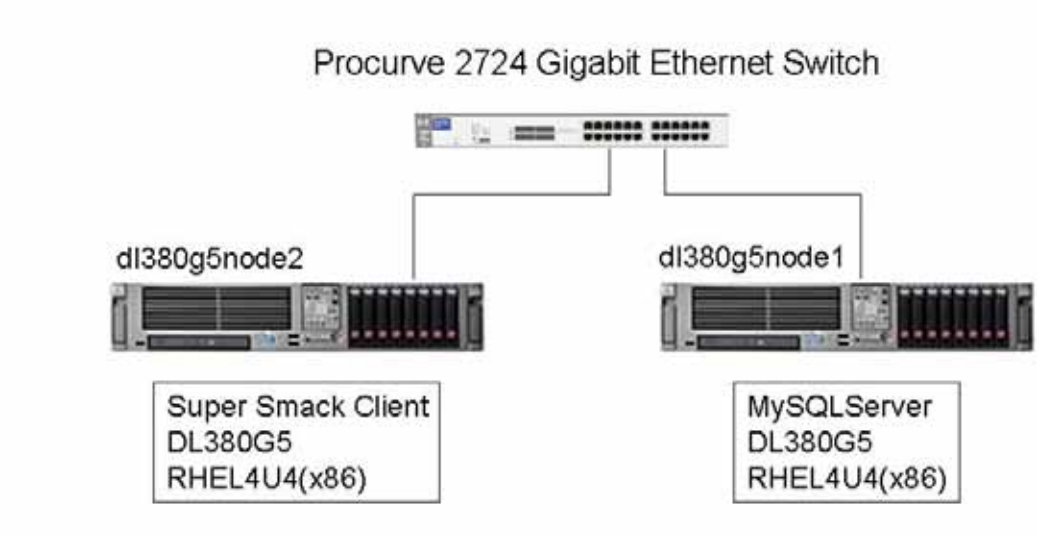

図 1　システム構成図

2．ベンチマーク実行環境のセットアップ

2.1．MySQL Server側ホストの設定手順

以下に MySQL Server を実行するホストの設定手順を示します。

2.1.1． MySQL Server のインストール

ベンチマークのホストとなるサーバに MySQL Enterprise Server をインストールします。（詳細なインストール手順は MySQL Enterprise Server 5.0 インストール手順書を参照してください）
今回は tar.gz 形式の MySQL を使用し、次のような手順で /usr/local/mysql ディレクトリ以下に MySQL Server をインストールしました。

MySQL のインストール手順例

```
$ su -
# /usr/sbin/groupadd mysql
# /usr/sbin/useradd -g mysql -m mysql
# cd /usr/local
# tar zxvf mysql-enterprise-gpl-5.0.36-linux-i686-glibc23.tar.gz
# chown -R mysql:mysql mysql-enterprise-gpl-5.0.36-linux-i686-glibc23
# ln -s mysql-enterprise-gpl-5.0.36-linux-i686-glibc23 mysql
# cd mysql
# cp support-files/my-innodb-heavy-4G.cnf /etc/my.cnf
# ./scripts/mysql_install_db --user=mysql
# ./bin/mysqld_safe --user=mysql &
```

2.1.2． MySQL ユーザのリモートホストからのアクセス設定

今回のベンチマークは 2 台サーバを使用しリモートホストから負荷をかけます。そこで、リモートホストからのアクセスを可能にする設定とパスワードの設定を行います。

```
$ /usr/local/mysql/bin/mysql -u root

mysql> grant all on *.* to 'root'@'%' identified by 'root';
mysql> set password for ‘root’@’localhost’ = password(‘root’);
```

今回は閉ざされた環境でのベンチマークですので、すべてのホストから接続可能な管理者権限を持つユーザ'root'を作成しています。

2.1.3． MySQL の匿名ユーザの削除

最終的に以下のユーザが残る様に匿名ユーザを削除します:

```
mysql> select host, user from mysql.user;
+---------------+------+
| host          | user |
+---------------+------+
| %             | root |
| localhost     | root |
+---------------+------+
2 rows in set (0.00 sec)
```

匿名ユーザの削除方法例:

```
mysql> drop user ''@'localhost';
                 ^^^  ^^^^^^^^^
              ユーザ名    ホスト名
```

2.1.4． システム情報を取得するための設定

ベンチマーク中の CPU 使用率や、IO 情報を取得する必要がある場合は、以下のように sysstat パッケージをインストールすることをお勧めします。

```
# rpm -qa | grep sysstat

# rpm -ivh --test /tmp/sysstat-5.0.5-11.rhel4.i386.rpm
警告: sysstat-5.0.5-11.rhel4.i386.rpm: V3 DSA signature: NOKEY, key ID db42a60e
Preparing...                ########################################### [100%]

# rpm -ivh /tmp/sysstat-5.0.5-11.rhel4.i386.rpm
警告: sysstat-5.0.5-11.rhel4.i386.rpm: V3 DSA signature: NOKEY, key ID db42a60e
```

```
Preparing...                ########################################### [100%]
   1:sysstat                ########################################### [100%]
```

2.2. クライアントホスト側の設定

以下に SuperSmack を実行するホストの設定手順を説明します。

2.2.1. MySQL Server のインストール

Super Smack1.3 はソースで提供されているため、インストールするにはコンパイルが必要です。コンパイルする際に MySQL 本体の共有ライブラリが必要となりますので、クライアントホスト側にも MySQL をインストールします。

```
$ su -
# /usr/sbin/groupadd mysql
# /usr/sbin/useradd -g mysql -m mysql
# cd /usr/local
# tar zxvf mysql-enterprise-gpl-5.0.36-linux-i686-glibc23.tar.gz
# chown -R mysql:mysql mysql-enterprise-gpl-5.0.36-linux-i686-glibc23
# ln -s mysql-enterprise-gpl-5.0.36-linux-i686-glibc23 mysql
# cd mysql
# cp support-files/my-innodb-heavy-4G.cnf /etc/my.cnf
# ./scripts/mysql_install_db --user=mysql
```

2.2.2. Super Smack の入手

今回使用するベンチマークツールは Super Smack の version 1.3 です。以下のサイトよりダウンロードします。

http://vegan.net/tony/supersmack/

2007 年 6 月現在最新の Super Smack は version1.3(super-smack-1.3.tar.gz)です。

2.2.3. SuperSmack のインストール

Super Smackは、通常、データベース(MySQLサーバ)と同一ホスト上で実行させて使用しますが、以下に紹介する負荷試験では、より本格的な運用時の負荷情況を想定しておりますので、MySQLサーバとSuper Smackを別なホストで実行しています。

本書では、以下のような条件でSuperSmackを用いて試験を実施しています。

- MySQL Server とクライアントは別ホストで動作
- ベンチマーク対象の MySQL Server は 1 サーバ(インスタンス)のみ
- ベンチマーク対象のテーブルは 1 つ
- クライアントは 1 ホストから実行
- MySQL Server が listen するポートは固定(3306)
- ベンチマーク対象のデータベースは固定(test データベース)

以下の手順で、クライアントにSuper Smackをインストールします。

```
# su - mysql    (*注意)
$ cd ~/mysql_work
$ tar zxvf /tmp/super-smack-1.3.tar.gz
$ cd super-smack-1.3
```

(*注意: 今回の手順書では mysql ユーザにしていますが、SuperSmack のインストールは必ずしも mysql ユーザである必要はありません)

2.2.4. SuperSmack の修正

今回のベンチマークでは、比較的大規模なデータを使用すること、またベンチマーク後にデータを集計しやすくするために、super-smack のソースコードにいくつか修正を施しています。

・ src/dictionary.cc

試験対象データのレコード長が長い場合には、以下のような変更を実施して下さい。本試験のレコード長は、約1Kbyteです。

修正前

```
#define MAX_FILE_LINE 512
```

修正後

```
#define MAX_FILE_LINE 4096
```

- **src/smack.h**

64bitで表現される大きなサイズのファイルを使用するために、最初の定義部分に以下の定義を追加します。

追加

```
#define __USE_FILE_OFFSET64
#define __USE_LARGEFILE64
```

- **src/query.cc**

Super Smackにて応答時間の集計は、1/1,000秒(msec)単位で行なわれますが、実際の応答時間はこれより小さいので、単位を1/1,000,000秒(usec)に変更します。

修正前

```
void Query_charge::fire(Client* cli)
{
 ...
      int q_time = (end.tv_sec - start.tv_sec)*1000 +
     (end.tv_usec - start.tv_usec)/1000;
 ...
}
```

修正後

```
{
 ...
     int q_time = (end.tv_sec - start.tv_sec)*1000000 +
        (end.tv_usec - start.tv_usec);
 ...
}
```

2.2.5. SuperSmack のコンパイル

以上の変更を加えた後、super-smack **をコンパイルします**

```
$ cd /home/mysql/mysql_work/super-smack-1.3
$ LIBZ_LIB="-lmygcc -lm" ./configure --prefix=/usr/local/supersmack/ \
--with-mysql --with-mysql-lib=/usr/local/mysql/lib \
--with-mysql-include=/usr/local/mysql/include
$ make
$ su
# make install
```

(*注意　紙面の都合上により改行しているところは \ で表現していますが、実際には一行で記述しています)

以上で super-smack **が使用できるようになりました。**

3. ベンチマークの実行手順

3.1. 設定ファイルの記述

Super Smack を使用してベンチマークを行う場合、Smack ファイルと呼ばれるシナリオファイルの設定が必要となります。Smack ファイルは、5種類のブロックと呼ばれる設定のまとまりから構成され、それぞれ設定する内容は次の通りです。

ブロック名	内容
client	MySQL Server に接続するクライアントやクエリの設定
table	ベンチマーク対象テーブルの定義
dictionary	クエリに埋め込む動的なデータの設定
query	クエリの定義
main	ベンチマークの実行手順の設定

Super Smack をインストールすると、インストールしたディレクトリに smack という名前のディレクトリが作成されます。その中に select-key.smack という名前のサンプルファイルがインストールされていますので、ここでは、select-key.smack というファイルを以下のように修正して、SuperSmack が動作することとを確認してみます。

```
$ cat smacks/select-key.smack
client "admin"            // client ブロック
{
 user "root";             // MySQL Server に接続するための MySQL ユーザ名
 host "localhost";        // MySQL Server のホスト名
 db "test";               // ベンチマーク対象のデータベース名'
 pass "root";             // MySQL ユーザのパスワード
 socket "/tmp/mysql.sock";
}

table "http_auth"         // table ブロック
{
  client "admin";         // テーブルを作成する client ブロックの名前

  // ベンチマーク対象テーブルを作成するための DDL(Data Definition Language)
  // ここでは http_auth テーブルの create を定義しています。
  create "create table http_auth
    (username char(25) not null primary key,
     pass char(25),
     uid integer not null,
     gid integer not null
    );
  min_rows "9000";        // ベンチマークを開始する前に最低限必要なレコード数
  data_file "words.dat"; // テーブルが空の場合、テーブルにロードするデータファイルのファイル名

  // data_file で設定したファイルを生成するための設定。
  gen_data_file "gen-data -n 9000 -f %12-12s%n,%25-25s,%n,%d";
}

dictionary "word"         // dictionary ブロック
{
  type "rand";            // データを選択する方法を指定
  source_type "file";     // データソースのタイプ。ここではファイルから読み込むという意味
  source "words.dat";     // 保存されている、もしくは作成したデータファイルのパス
  delim ",";              // データファイルの各列の区切り文字
  file_size_equiv "0";
       // ファイルサイズが指定したサイズより大きい場合に、使用するデータの数を制限します。
       // このオプションを使うことで super-smack が使用するメモリを節約することが出来ます。
       // サンプルファイルでは 45000 に設定されていますが、0 に設定することをお勧めします。
       // 詳細は、「付録 1 file_size_equiv について」を御参照下さい。
}

query "select_by_username"      // query ブロック
{
  // 実際にクライアントから実行されるクエリ
  query "select * from http_auth where username = '$word'";
  type "select_index";    // ベンチマーク終了時に表示されるクエリの名称
  has_result_set "y";     // クエリがセレクト分の場合には y を指定
  parsed "y";             // クエリに$word のような変数を使用する場合は y を指定
}
```

```
client "smacker1"        // client ブロック
{
 user "root";                   // MySQL Server に接続する client ブロック
 pass "root";                   // MySQL ユーザのパスワード
 host "localhost";              // MySQL Server のホスト名
 db "test";                     // ベンチマーク対象のデータベース名
 socket "/tmp/mysql.sock";
 query_barrel "2 select_by_username"; // このクライアントが発行するクエリの集合
                                      // この場合 select_by_username という名前の
                                      // query ブロックで定義されたクエリを2回発行
}

main                    // main ブロック
 {
  smacker1.init();      // smacker1 クライアントの初期化
  smacker1.set_num_rounds($2); // smacker1 の query_barrel を繰り返す回数の設定
  smacker1.create_threads($1); // smacker1 の生成数（スレッド数）を指定
  smacker1.connect();               // MySQL サーバに接続
  smacker1.unload_query_barrel();   // 各 smacker1 クライアントがクエリを発行
  smacker1.collect_threads();       // 各 smacker1 クライアントの終了
  smacker1.disconnect();            // MySQLServer から切断します。
}
```

smack ファイルのサンプル(select-key.smack)では MySQL を実行するサーバは、デフォルトで localhost に設定されていますが、ここを変更することで、SuperSmack を実行するサーバと、MySQL サーバを別々のサーバにする事が可能です。また、MySQL Server に接続するための user 名と password を以下のように設定する必要があります。

修正前

```
    host "localhost";
    user “root”; または “test”;
    pass “”;
```

修正後

```
    host "mysql_server_host";
    user “root”;
    pass “root”;
```

select-key.smackファイルに対して、上記の修正は2箇所(上記の例ではclient "admin"ブロック内とclient "smacker1"ブロック内の2箇所)行なう必要があります。またhostに指定するホスト名で名前解決が出来るよう/etc/hostsを設定しておいてください。

本書では、Super Smack と MySQL サーバを別ホストで実行していますが、この場合、次のことに注意する必要があります。Super Smack ではデータベースにデータをロードさせるためのデータファイルを、gen-data というコマンドを使用して作成するため、このデータファイルは super-smack を実行したサーバ上の指定したディレクトリに作成されます。一方、ベンチマークを実行させるためには、データベースの Table にデータをロードさせる必要があるので、このデータファイルを MySQL が動作しているサーバ上に置く必要があります。このため、MySQL サーバと super-smack を別ホストで実行する場合はデータファイルを MySQL サーバ側の指定されたディレクトリ配下にコピーしておく必要があります。詳細については、次の「3.2ベンチマークの実行方法」を御参照下さい。

ホスト名以外にも、file_size_equiv の値も以下のように0に設定することをお勧めします。詳細については、「付録 1 file_size_equiv について」を御参照下さい。

```
    file_size_equiv "0";
```

3.2. ベンチマークの実行方法

super-smack の実行手順は次の通りです。

パスの設定

```
# export PATH=/usr/local/supersmack/bin:$PATH
```

以下の手順で、データファイルを置くディレクトリ(/var/smack-data)を、SuperSmack を実行するホストとMySQLが動作するホストに作成します。

```
$ hostname
host1
$ mkdir /var/smack-data

$hostname
host2
$ mkdir /var/smack-data
```

準備が整いましたので、以下のように SuperSmack を実行しますが、1度目はリモートの MySQL サーバ側にデータがないため、super-smack の実行は失敗します。

```
$ cd /home/mysql/mysql_work/super-smack-1.3
$ super-smack smacks/select-key.smack 10 1000
Table 'http_auth' does not meet condtions, will be dropped
Creating table 'http_auth'
Populating data file '/var/smack-data/words.dat' with shell command 'gen-data -n
9000 -f %12-12s%n,%25-25s,%n,%d'
Loading data from file '/var/smack-data/words.dat' into table 'http_auth'
Error running query load data infile '/var/smack-data/words.dat' into table
http_auth fields terminated by ',':Can't get stat of '/var/smack-data/words.dat'
(Errcode: 2)
super-smack: aborting on failed query
```

1度目の実行で作成されたデータファイル(words.dat)をリモートの MySQL サーバへコピーします。

```
$ scp /var/smack-data/words.dat [mysql_server_host]:/var/smack-data/
words.dat                                     100%  512KB 511.9KB/s   00:00
```

改めて super-smack を実行します。

```
$ super-smack smacks/select-key.smack 10 1000
Table 'http_auth' does not meet condtions, will be dropped
Creating table 'http_auth'
Loading data from file '/var/smack-data/words.dat' into table 'http_auth'
Table http_auth is now ready for the test
Query Barrel Report for client smacker1
connect: max=2ms  min=1ms avg= 1ms from 10 clients
Query_type      num_queries     max_time        min_time        q_per_s
select_index    20000   0       0       18894.31
```

super-smack コマンドの引数の意味は次の通りです。

第1引数:smack ファイルの指定
第2引数:データベースに接続するクライアント数 (スレッド数)
第3引数:smack ファイルで設定したクエリを実行する回数

実際に MySQL サーバの http_auth テーブル内に、データがロードされているか確認することをお勧めします。

```
# /usr/local/mysql/bin/mysql -u root
mysql> select count(*) from test.http_auth;
+----------+
| count(*) |
+----------+
|     9000 |
+----------+
```

3.3. ベンチマークデータの変更方法

ベンチマーク対象のデータは、words.dat ファイルと MySQL データベース内のテーブルで管理されていますので、ベンチマーク対象のデータを変更する場合は、以下の手順で既存のデータを削除してから新しいデータを作成することをお勧めします。

```
$ hostname
host1
$ rm /var/smack-data/words.dat

$ hostname
host2
$ rm /var/smack-data/words.dat
$ mysql –u root test
mysql> drop table http_auth;
```

4. MySQL ベンチマーク

4.1. 試験環境

今回のベンチマークでは、2 台の PC サーバを使用し、クライアントに Super Smack をインストールして、サーバ側の MySQL Enterprise Server に負荷をかけます。

サーバ	
Hardware	HP Proliant DL380G5
CPU	Intel Xeon E5320(1.86GHz) 4Core × 1
メモリサイズ	4GB
OS	Red Hat Enterprise Linux 4 update 4 (x86)
MySQL	MySQL Enterprise Server 5.0.36 (Linux, i686, glibc-2.3, noRPM 版)
ホスト名	dl380g5node1
ハードディスク	内蔵 DISK 146GB×8 10Krpm

クライアント	
Hardware	HP Proliant DL380G5
CPU	Intel Xeon E5320(1.86GHz) 4Core × 2
メモリサイズ	16GB
OS	Red Hat Enterprise Linux 4 update 4 (x86)
ベンチマークツール	super-smack-1.3
ホスト名	dl380g5node2
ハードディスク	内蔵 DISK 146GB×8 10Krpm

ネットワーク	
ネットワークスイッチ	HP Procurve 2724 ギガビットイーサネットスイッチ

4.2. 試験データ

試験に使用するデータは super-smack を使用してレコード数が異なる 3 つの試験データを作成しました。

データベース名	test
テーブル名	http_auth (1 table)
レコード長	1KB
レコード数	1,000,000 (データサイズ約 1G) 2,000,000 (データサイズ約 2G) 10,000,000 (データサイズ約 10G)

テーブルは test データベースに作成される http_auth テーブル 1 つを使用しました。
データサイズはサーバの搭載メモリ 4G を考慮し、メモリ上にすべてのデータが乗るサイズのもの、メモリの空き容量とほぼ同じサイズのもの、メモリ容量よりも十分に大きいサイズのものの 3 パターンを作成しました。

4.3. queryの種類とシナリオ

MySQL サーバに負荷をかける際の query の種類は次の 2 パターンを作成しました。

select クエリ	select * from table where username = 'xxxxx'; ('xxxxx' は username に登録されているキーの中から任意に選んだ値)
update クエリ	update table set pass= 'xxxxx' where username = 'xxxxx'; ('xxxxx' は username に登録されているキーの中から任意に選んだ値)

また、ベンチマークではこれらのクエリを組み合わせて、次の 5 種類のシナリオを作成しました。

シナリオ1	select だけを実行
シナリオ 2	select と update を 10:1 の割合で実行
シナリオ 3	select と update を 5:1 の割合で実行
シナリオ 4	select と update を 1:1 の割合で実行
シナリオ 5	update だけを実行

4.4. 試験項目

本ベンチマークでは以下の条件を変更し、1 秒当たりの select クエリの実施回数、update クエリの実施回数および MySQL サーバの CPU 使用率について集計しました。

データサイズ	1GB, 2GB, 10GB
シナリオ	シナリオ 1～シナリオ 5
クライアント数(スレッド数)	1，2，4, 8, 16, 32, 64　スレッド

付録に、データサイズ 1GB、シナリオ 4 の smack ファイルを載せますので御参照下さい。

4.5. 試験結果

結果を以下に示します。

4.5.1. スループットの比較

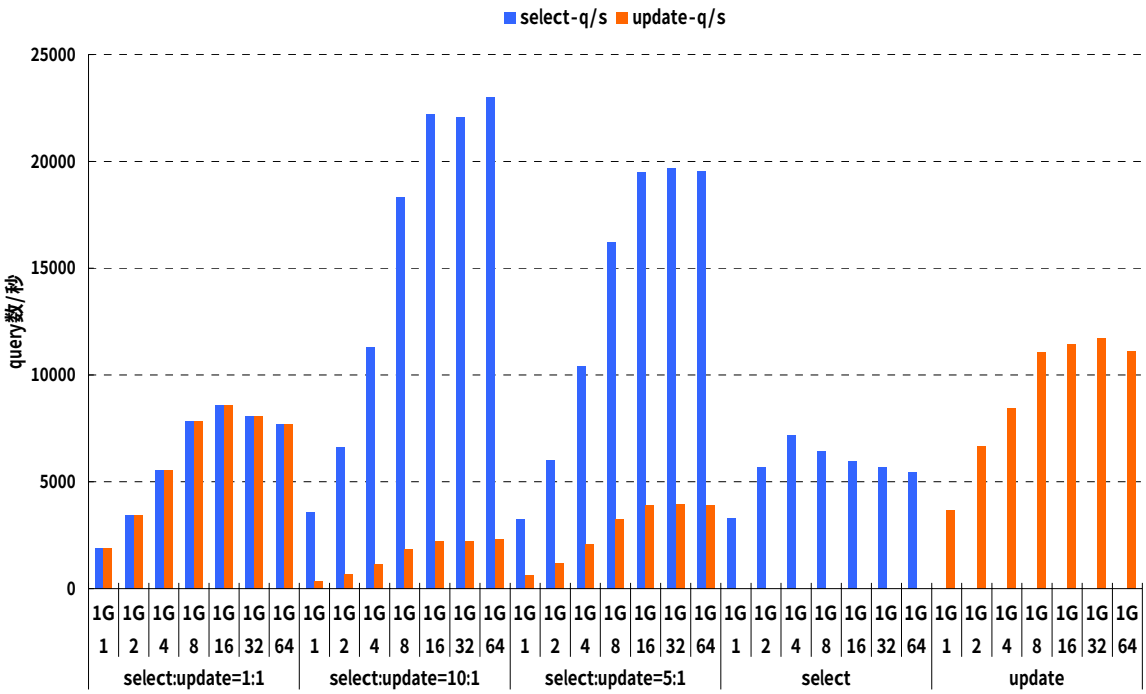

図 2 スループット（query/sec）データサイズ 1G

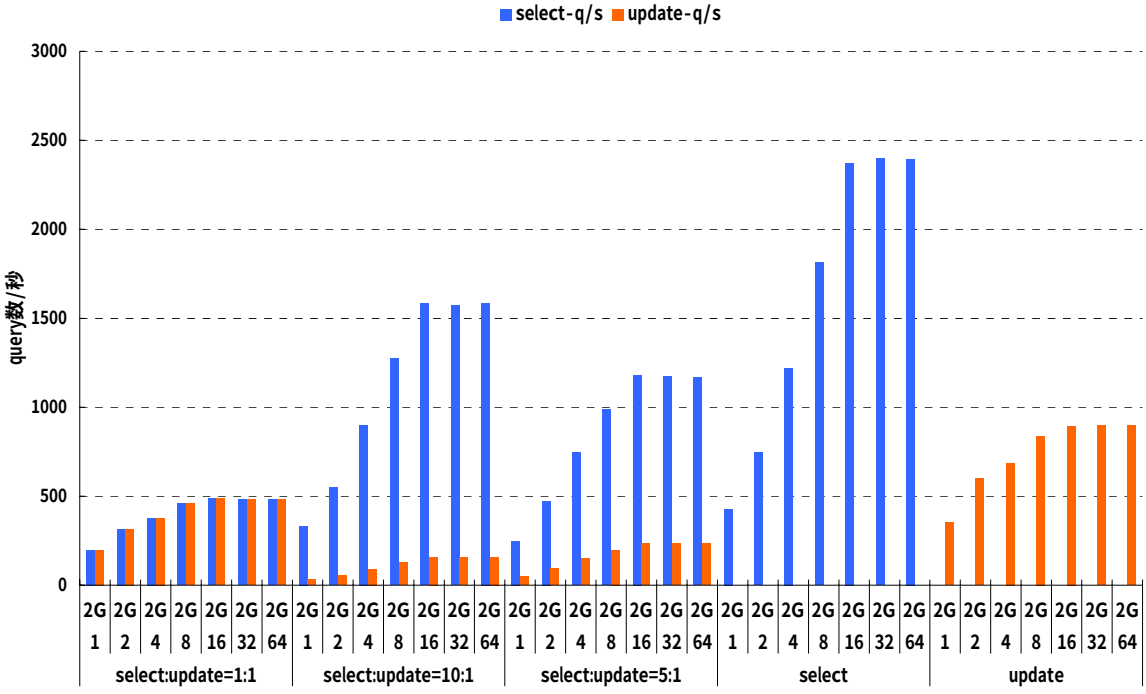

図 3 スループット（query/sec）データサイズ 2G

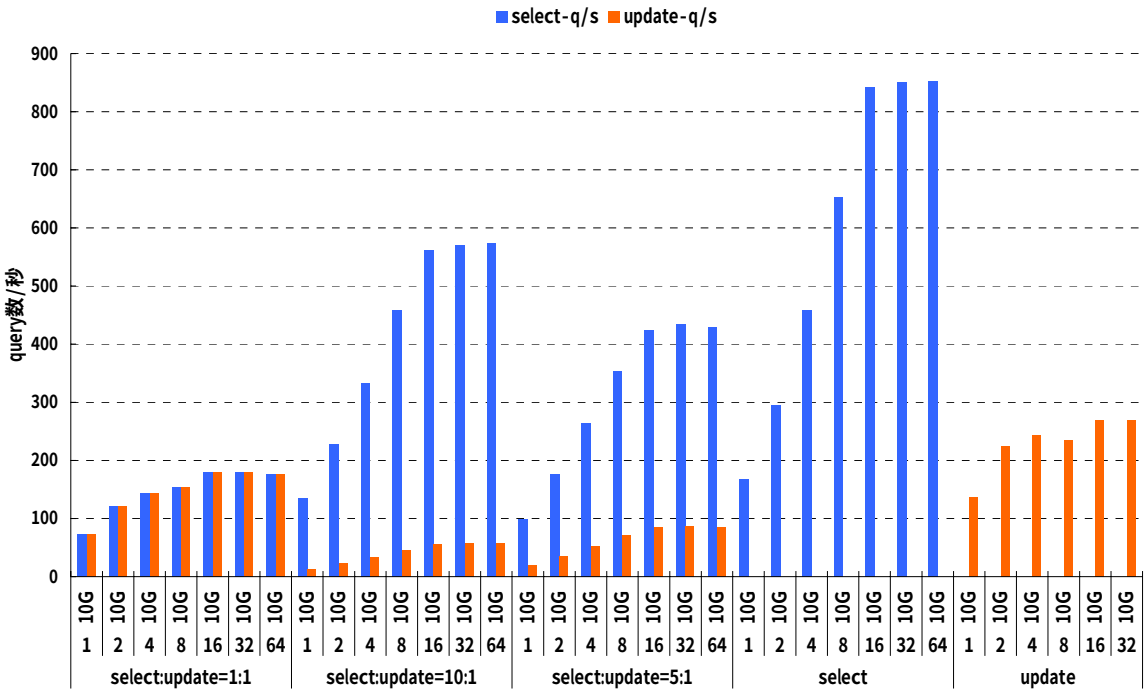

図 4 スループット (query/sec) データサイズ 10G

4.5.2. CPU 使用率の比較

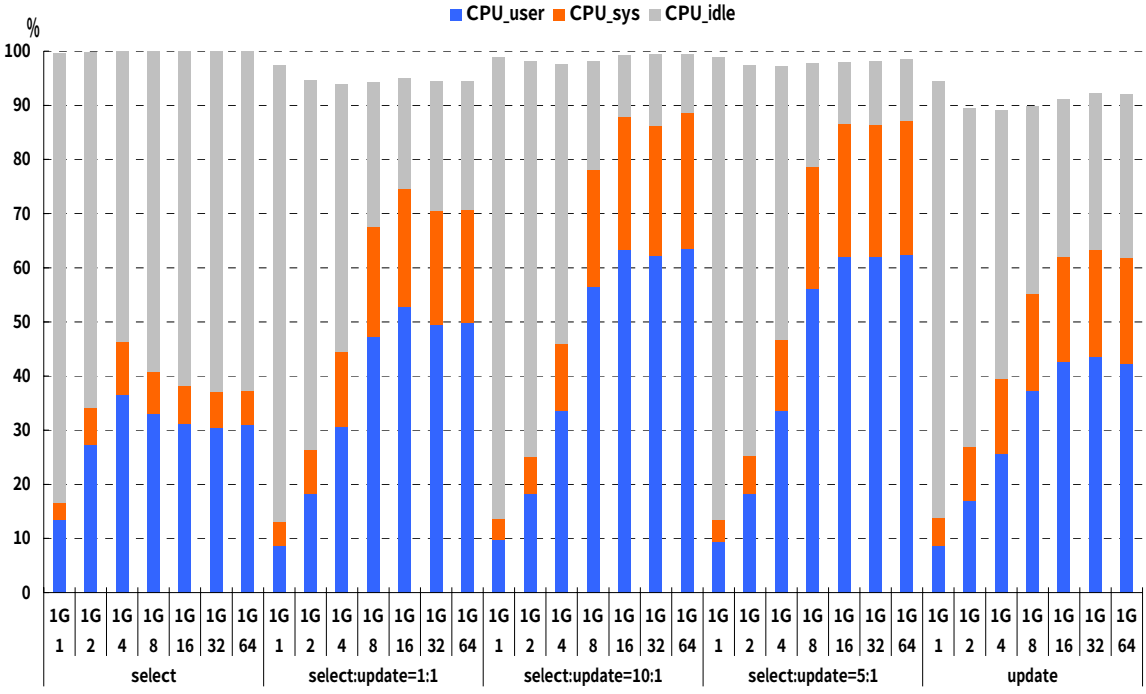

図 5 CPU 使用率 データサイズ 1G

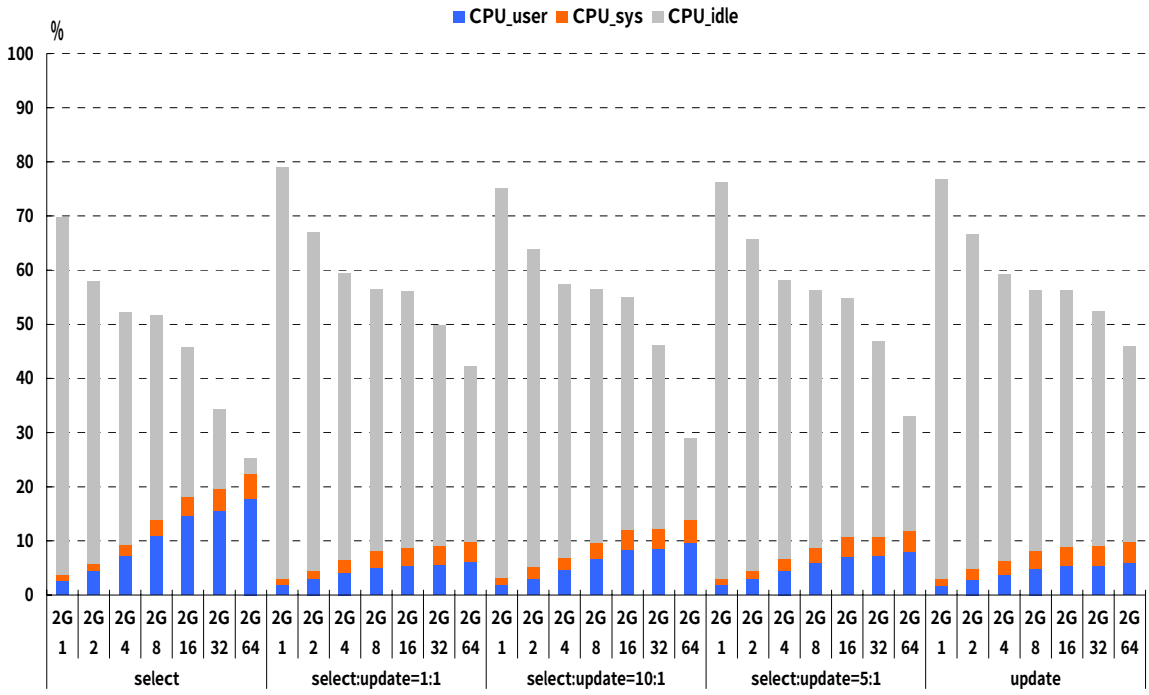

図 6 CPU 使用率　データサイズ 2G

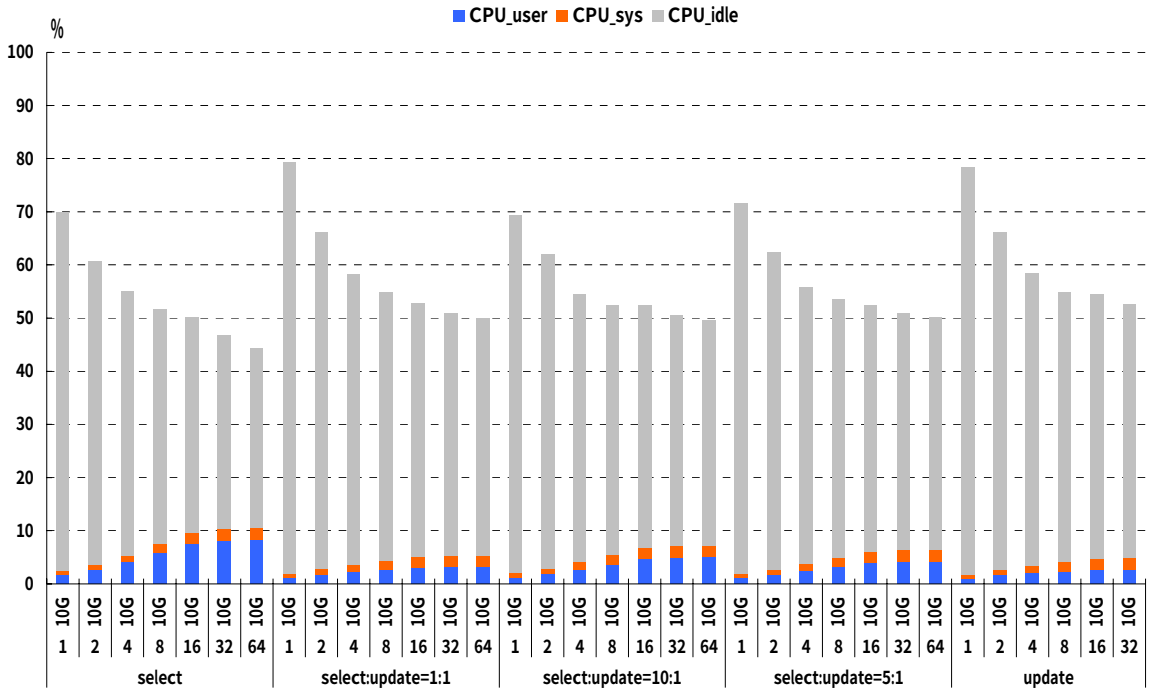

図 7 CPU 使用率　データサイズ 10G

4.5.3. 考察

図 2 より、データサイズ 1G の試験において、select クエリの処理性能は、select クエリだけを実施した場合より select と update を同時に実施した場合の方が高いスループットとなる結果が得られました。図 5 より、このときの CPU 使用率を確認すると、データサイズ 1G の場合は 40～50%となっており、CPU を効率よく使用できていないことがわかります。

CPUを使い切れていない問題を調査したところ、この問題はquery cacheの実装に依存していそうだということが分かりました。そこで、追加検証としてquery cacheを使用しない設定で試験を実施します。

4.6. query cache を0にした場合の結果

/etc/my.cnf より query_cache_size の値を変更して、データサイズ 1G の再試験を行いました。変更箇所は次の通りです。

変更前

```
query_cache_size = 64M
```

変更後

```
query_cache_size = 0
```

4.6.1. スループットの比較

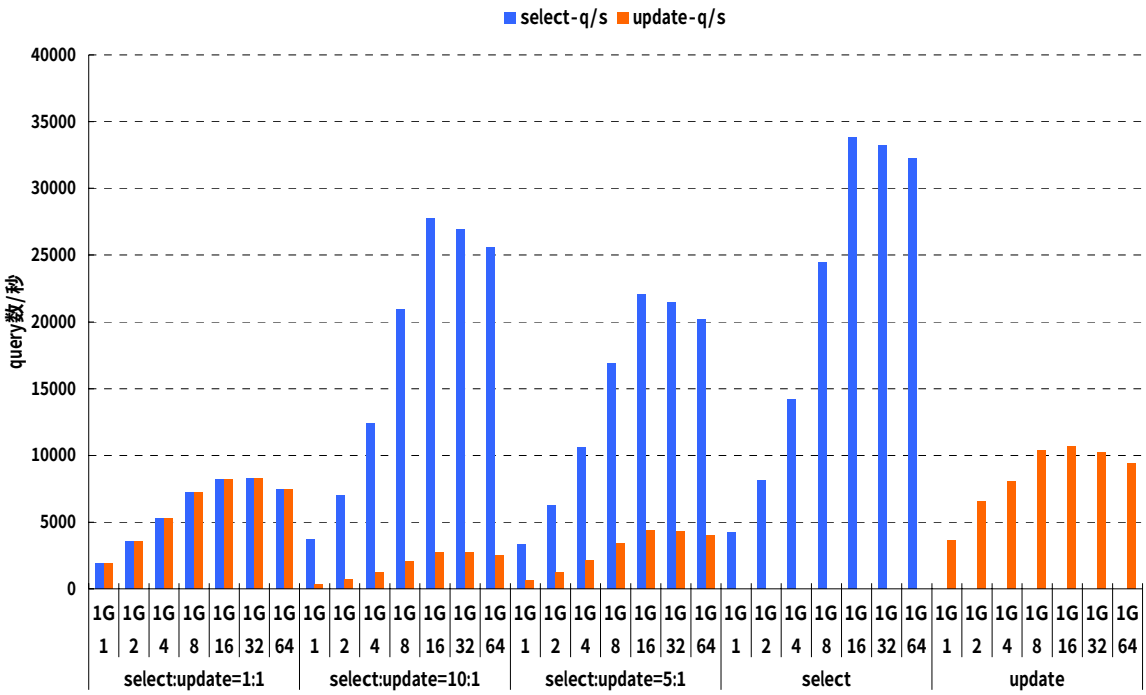

図 8 スループット（query/sec）データサイズ 1G query cache = 0

4.6.2. CPU 使用率の比較

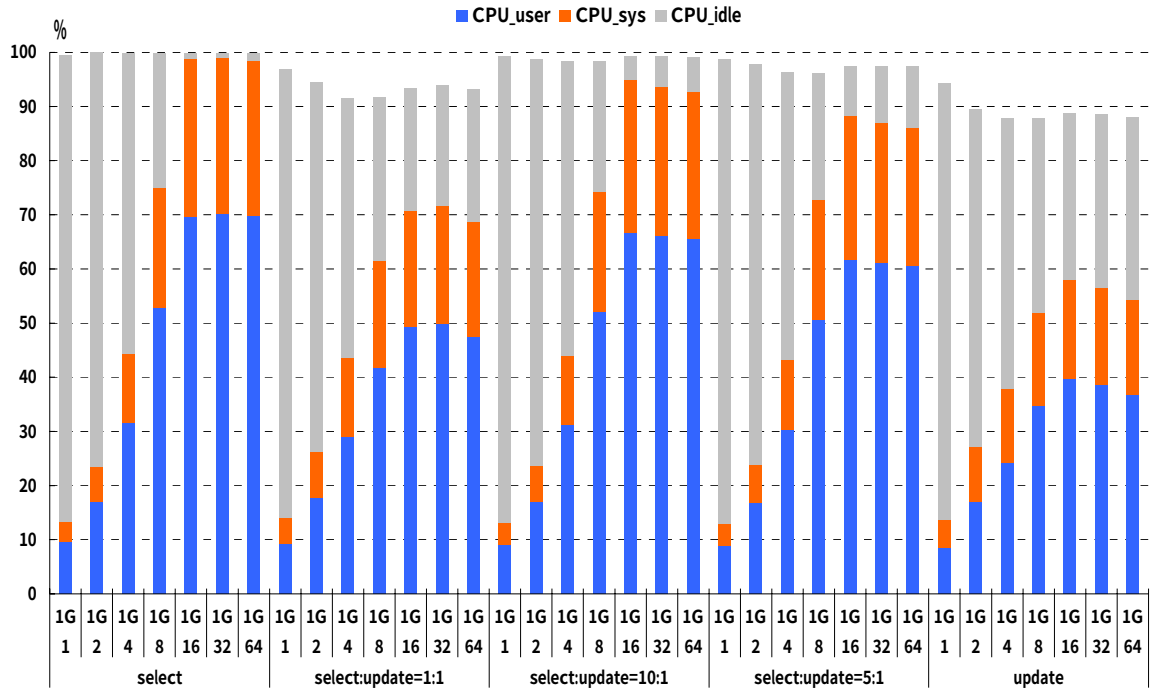

図 9 CPU 使用率　データサイズ 1G query cache = 0

4.6.3. 考察

図 8 より、データサイズ 1G で query cache を使わない設定にして試験を実施したところ、select だけを実施した場合の性能は、約 4 倍の性能が得られた。これは今回実施した試験の条件が以下のような条件であったためと考えられる。

1. データ構造が単純
2. select クエリは、プライマリーキーを用いて 1 つのレコードを選択するような簡単なもの
3. 全部のデータがメモリに入る

このような条件下では、1つのクエリの実行時間がusec単位と非常に短くなるので、複数のスレッドでquery cacheを操作するときに多くの待ちが発生していたと考えられます。このことから、上記のような条件下でMySQLサーバを使用するときは、query cacheを使用しない方が良いと考えられます。

付録 1 file_size_equiv について

ここでは Super Smack の設定ファイル内で指定する file_size_equiv について説明します。

file_size_equiv は、SuperSmack のシナリオを定義するファイルの dictionary 定義の中で使用され、以下のように外部ファイルにあるキーを使用するときに設定するパラメータです。

```
dictionary "word"
{
  type "rand";
  source_type "file";
  source "words.dat";
  delim ",";
  file_size_equiv "45000";
}
```

このような設定の場合、"words.dat"というファイルを読み込み、","で区切られる最初のフィールドの値を使ってランダムなキーを作成して試験を実施します。この例にもあるように、Super Smack に付いているサンプルの smack ファイルでは、file_size_equiv の値が 45000 に設定されています。

この設定の場合、試験データのファイルサイズが45Kbyteを越えた場合に検索などで使用するキーの数を制限していまいます。このため、45Kbyteを超えるデータを用いて試験する場合には、以下のような設定に変更することをお勧めします。

file_size_equiv "0";

このような設定にすることで、データファイルにある全てのキーを使用して試験をすることが出来るようになります。

file_size_equivは、もともとは大きなデータを試験するときに検索などで使用するキーの数を制限することにより、SuperSmackが使用するメモリを制限するために設計されたと考えられます。以下に、file_size_equivの設定方法の詳細を説明します。

file_size_equivは、以下のように使用されます。

データファイルサイズ / file_size_equiv の値で示される行毎にデータファイルを読み込み、
その値を使用してデータを読み込みます。

例えば、データファイルサイズが5 Mbyte、 file_size_equivが、5K byteの場合は、

5M byte / 5K byte = 1,000

となり、1000行おきにkeyとなるデータを読み込んでkeyとして使用しますので、最終的に、読み込まれてkeyとして使用されるのは、1000行目、2000行目、3000行目....のkeyとなります。

一方、全部の行を読み込むには、file_size_equivに対して以下のいずれかの指定を行なう必要があります。

1. file_size_equivの指定を行なわない
2. file_size_equiv = 0 という指定を行なう
3. file_size_equivにデータファイルサイズ以上の値を設定する。

付録 2 性能試験で使用した smack ファイル

今回のベンチマークで使用した smack ファイルのサンプルです。本サンプルは select と update が混在したクエリを実行するための設定となっています。

```
client "admin"
{
 user "root";
 host "mysql_server_host";                    // MySQL サーバのホスト名を記述
 db "test";
 pass "root";
 port "3306";
}

table "http_auth"
{
  client "admin";

  // 試験するテーブルのレコードを定義（レコードサイズは、約 1Kbyte）
  create "create table http_auth
    (username char(25) not null primary key,
     pass char(25),
     address1 char(255),
     address2 char(255),
     address3 char(255),
     address4 char(255),
     uid integer not null,
     gid integer not null
    ) ENGINE=InnoDB";

  min_rows "1000000";
  data_file "words.dat";
  gen_data_file "gen-data -n 1000000 -f %12-12s%n,%25-
25s,%255s,%255s,%255s,%255s,%n,%d";
     // gen-data の行数指定を変更することでデータファイルの大きさを調整
     // ここでは 1000000 を指定しているのでファイルサイズは、約1Gbyte（1KB×1000000）になる。
}

dictionary "word"
{
  type "rand";
  source_type "file";
  source "words.dat";
  delim ",";
  file_size_equiv "0";
}

query "select_by_username"
{
  query "select * from http_auth where username = '$word'";
  type "select_index";
  has_result_set "y";
  parsed "y";
}

query "update_by_username"                     // update の query を追加しています
{
  query "update http_auth set pass='$word' where username = '$word'";
  type "update_index";
  has_result_set "y";
  parsed "y";
}

client "smacker"
{
 user "root";
 pass "root";
 host "mysql_server_host";                    // MySQL サーバのホスト名を記述
 db "test";
 port "3306";
 query_barrel "1 select_by_username 1 update_by_username";
```

```
}

main
{
  smacker.init();
  smacker.create_threads($1);
  smacker.set_num_rounds($2);
  smacker.connect();
  smacker.unload_query_barrel();
  smacker.collect_threads();
  smacker.disconnect();
 }
```

お問い合わせはカスタマー インフォメーションセンターへ
03-5304-6660 **月～金**9:00～19:00 **土**10:00～18:00（**日、祝祭日、年末年始および5/1を除く**）
Linux/**オープンソース製品に関する情報は** http://www.hp.com/jp/linux/

日本ヒューレット・パッカード株式会社
〒102-0076 東京都千代田区五番町 7