

Percona Toolkit

利用ガイド

2015年8月21日

第1.5版

株式会社スマートスタイル

[表紙](#)

[目次](#)

[改定履歴](#)

[はじめに](#)

[インストール](#)

[前提](#)

- 1 [pt-align](#)
- 2 [pt-archiver](#)
- 3 [pt-config-diff](#)
- 4 [pt-deadlock-logger](#)
- 5 [pt-diskstats](#)
- 6 [pt-duplicate-key-checker](#)
- 7 [pt-fifo-split](#)
- 8 [pt-find](#)
- 9 [pt-fingerprint](#)
- 10 [pt-fk-error-logger](#)
- 11 [pt-heartbeat](#)
- 12 [pt-index-usage](#)
- 13 [pt-ioprofile](#)
- 14 [pt-kill](#)
- 15 [pt-mext](#)
- 16 [pt-mysql-summary](#)
- 17 [pt-online-schema-change](#)
- 18 [pt-pmp](#)
- 19 [pt-query-digest](#)
- 20 [pt-show-grants](#)
- 21 [pt-sift](#)
- 22 [pt-slave-delay](#)
- 23 [pt-slave-find](#)
- 24 [pt-slave-restart](#)
- 25 [pt-stalk](#)
- 26 [pt-summary](#)
- 27 [pt-table-checksum](#)
- 28 [pt-table-sync](#)
- 29 [pt-table-usage](#)
- 30 [pt-upgrade](#)
- 31 [pt-variable-advisor](#)
- 32 [pt-visual-explain](#)

バージョン	日付	対象ページ	改訂要旨
第0.5版	2014年8月5日	全ページ	レイアウトの調整。各コマンドの説明を修正
第1.0版	2014年8月13日	全ページ	シナリオの追加、各コマンドの説明を修正
第1.1版	2014年8月14日	2, 3, 5, 10, 12, 13, 15, 前提	シナリオの修正、ページの追加、各コマンドの説明の修正
第1.2版	2014年9月10日	7, 9, 11, 22, 23, 25, 27, 30	シナリオの修正、ページの追加、各コマンドの説明の修正
第1.3版	2014年10月28日	全ページ	スクリーンショットの差し替え、本文の加筆・修正
第1.4版	2014年11月25日	全ページ	「はじめに」ページの追加、本文の加筆・修正
第1.5版	2015年8月21日	7, 21, 24, 25, 27, 30	シナリオの追加、各コマンドの説明を修正

■ Percona Toolkit とは

「Percona Toolkit」は、PERCONA社が開発した、MySQLの“運用”、“監視”、“分析”といった複雑な作業を簡単に実施することができるコマンドツール群です。全部で32ツール存在します。(2014年11月19日現在)

PERCONA社は、MySQLのコンサルティング・保守サービス分野において、世界で最も高い評価の得ている企業です。2006年の創業以来、世界各国で1200件以上のコンサルティングおよびサポートの実績をもち、シスコシステムズ、グルーポン、BBCなどはPERCONA社の顧客となっています。

Percona製品は、MySQLと非常に高い互換性を持つ(drop-in replacement)MySQLの派生製品で、MySQLユーザが快適に利用いただけることを開発コンセプトとしております。また、MySQLと同様、オープンソースで提供され、ユーザは無償で利用することができます。そのため、当ツールは日常の業務の中でMySQLを利用しているエンジニアの方は勿論、データベース全般に関するコンサルタントの方にもご活用いただけるツールとなっております。

■ 本書の目的

本書では、「Percona Toolkit」に含まれるツール一つ一つの機能説明と、そのツールを試験的に使用する場合の手順を記述しています。

本書が、複雑な作業に取り組むデータベースエンジニアの一助となると共に、日本において「Percona Toolkit」が普及するきっかけとなっただけであれば幸いに感じます。

■ 注意事項

本書の一部または全部の無断転載を禁じます。

本書の内容に基づく運用結果についての責任は負いかねますので、ご了承ください。

■ Percona Toolkit のインストール

● 環境

CentOS 6.5
MySQL 5.6.19 Enterprise
Percona Toolkit 2.2.10

● Percona Toolkit 2.2 でサポートされているプラットフォーム

Debian 6
RHEL 5 & 6
Ubuntu 10.04 LTS & 12.04 LTS

Percona Server 5.0, 5.1, 5.5 & 5.6 / MySQL 5.0, 5.1, 5.5 & 5.6

Percona XtraDB Cluster 5.5 & 5.6

※ 詳細は下記URL参照してください

<https://www.percona.com/services/support/mysql-support/percona-toolkit-supported-platforms-and-versions>

● インストール文

```
# yum localinstall http://www.percona.com/redirect/downloads/percona-toolkit/LATEST/RPM/percona-toolkit-2.2.10-1.noarch.rpm
```

(バージョンは最新のものに書き換える)

※ 各コマンドは /usr/bin 配下に配置されます

■ Percona Toolkit の設定ファイル

● 設定ファイルを格納するディレクトリ "percona-toolkit" を /etc 配下に作成します

```
# mkdir /etc/percona-toolkit
```

● 設定ファイル "percona-toolkit.conf" を作成します

```
# touch /etc/percona-toolkit/percona-toolkit.conf
```

このファイルには全ツール共通のオプションを記述します

※ 全てのツールは実行時に、まずこの設定ファイルを読み込むため、

実行したツールに存在しないオプションがあるとエラーになります

全てのツールに共通して書けるオプションに、こういったものがあるのかは現在調査中です

個別のツールのオプションは、「〇〇.conf」に記述します。〇〇はツール名です

例えば "pt-duplicate-key-checker" の場合、ファイル名は「pt-duplicate-key-checker.conf」とし、

vi で下記のように記述します

```
# config for pt-duplicate-key-checker
user=root
socket=/var/lib/mysql/mysql.sock
password=password
h=localhost
d=test
t=articles
```

コメント

MySQL のユーザ名

ソケットのパス

MySQL のパスワード

ホスト名

データベース名

テーブル名

ここに記述した内容は、実行時にオプションとして指定する必要がなくなります

■ 使用するテーブルについて

- 本マニュアルのシナリオでは、以下のようなテーブルを使用します

```
root@hoge1:~  
mysql> SHOW CREATE TABLE articles\G  
***** 1. row *****  
Table: articles  
Create Table: CREATE TABLE `articles` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `title` varchar(255) DEFAULT NULL,  
  `body` text,  
  `created_at` datetime DEFAULT NULL,  
  `updated_at` datetime DEFAULT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=501706 DEFAULT CHARSET=latin1  
1 row in set (0.03 sec)  
mysql>
```

- 「テーブルに50万件のデータを取り込む」などの操作には、以下のようなファイルを使用します

```
root@hoge1:~  
[root@hoge1 ~]# head /tmp/mysql_tmp/14_2_articles.txt  
1 title_1 blog_text_1 2006-08-01 00:00:00 2006-08-01 00:00:00  
2 title_2 blog_text_2 2006-08-01 00:00:00 2006-08-01 00:00:00  
3 title_3 blog_text_3 2006-08-01 00:00:00 2006-08-01 00:00:00  
4 title_4 blog_text_4 2006-08-01 00:00:00 2006-08-01 00:00:00  
5 title_5 blog_text_5 2006-08-01 00:00:00 2006-08-01 00:00:00  
6 title_6 blog_text_6 2006-08-01 00:00:00 2006-08-01 00:00:00  
7 title_7 blog_text_7 2006-08-01 00:00:00 2006-08-01 00:00:00  
8 title_8 blog_text_8 2006-08-01 00:00:00 2006-08-01 00:00:00  
9 title_9 blog_text_9 2006-08-01 00:00:00 2006-08-01 00:00:00  
10 title_10 blog_text_10 2006-08-01 00:00:00 2006-08-01 00:00:00  
[root@hoge1 ~]#
```

■ MySQLの設定について

- MySQLの設定ファイルは以下の通り記述します

```
[client]  
port = 3306  
socket = /var/lib/mysql/mysql.sock  
[mysqld]  
port = 3306  
socket = /var/lib/mysql/mysql.sock  
key_buffer_size = 256M  
innodb_file_per_table  
innodb_data_home_dir = /var/lib/mysql/  
innodb_buffer_pool_size = 256M  
innodb_log_file_size = 128M  
character-set-server = utf8  
slow_query_log = 1  
slow_query_log_file = slow_queries.log  
log-error = /var/lib/mysql/mysqld.err
```

■ MySQLの設定について(レプリケーション)

- レプリケーション関連のツールを用いる時は、MySQLの設定ファイルは以下の通り記述します
マスタ側(ホスト名:hogehoge1)

```
[client]
port = 3306
socket = /var/lib/mysql/mysql.sock
[mysqld]
port = 3306
socket = /var/lib/mysql/mysql.sock
key_buffer_size = 256M
innodb_file_per_table
innodb_data_home_dir = /var/lib/mysql/
innodb_log_file_size = 128M
character-set-server = utf8
slow_query_log = 1
slow_query_log_file = slow_queries.log
log-error = /var/lib/mysql/mysql.err

# config for Replication
server-id = 1000
log-bin = /var/lib/mysql/mysql-bin
```

スレーブ側(ホスト名:hogehoge2)

```
[client]
port = 3306
socket = /var/lib/mysql/mysql.sock
[mysqld]
port = 3306
socket = /var/lib/mysql/mysql.sock
key_buffer_size = 256M
innodb_file_per_table
innodb_data_home_dir = /var/lib/mysql/
innodb_log_file_size = 128M
character-set-server = utf8
slow_query_log = 1
slow_query_log_file = slow_queries.log
log-error = /var/lib/mysql/mysql.err

# config for Replication
server-id = 2000
relay-log = /var/lib/mysql/mysql-relay-bin
```

■ コマンド


pt-align

■ 目的

- vmstat コマンドや iostat コマンドなどの出力内容を見やすくします

■ シナリオ

- vmstat コマンドを使用して、CPUやメモリーの負荷率を表示することができますが、ウィンドウサイズが大きくなると非常に見づらい表示となってしまいます

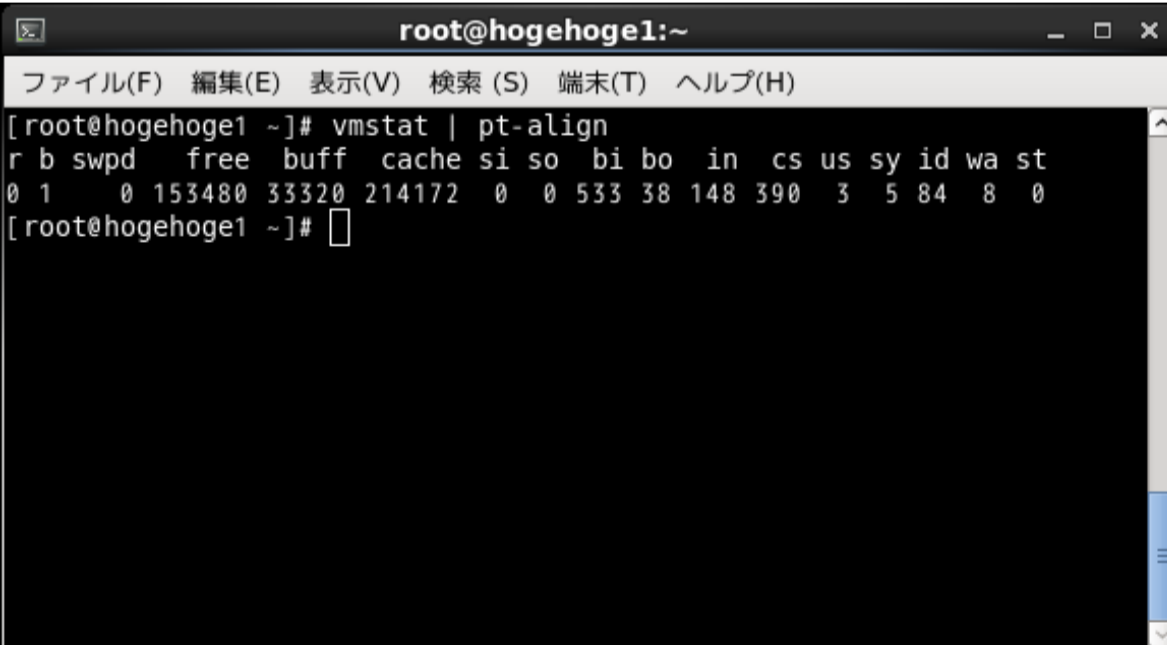


```
root@hoge1:~# vmstat
procs -----memory----- ---swap-- -----io----- --system-- ---
--cpu-----
 r b swpd free buff cache si so bi bo in cs us sy id wa st
0 0 0 162380 33104 209484 0 0 577 39 153 396 3 5 83 9 0
[root@hoge1:~]#
```

パイプを用いてvmstatの標準出力をpt-alignに引き渡すことで、ウィンドウサイズを変えずに出力内容を見やすくすることが出来ます

■ 結果

- 項目を区切るスペース幅などが自動調節され、表形式で整列されて表示されます



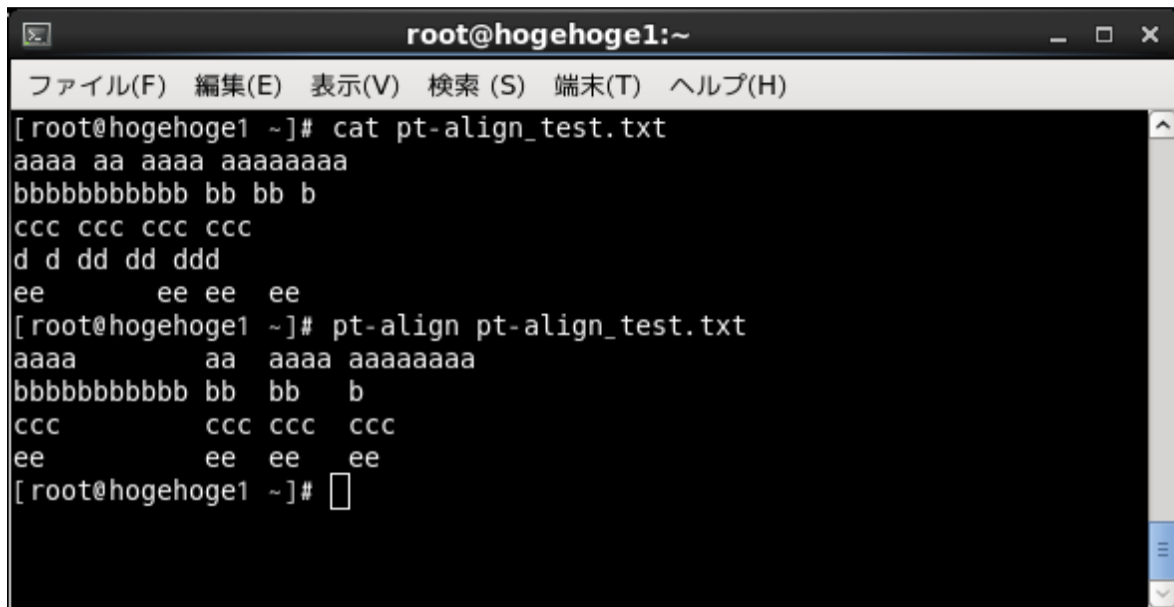
```
root@hoge1:~# vmstat | pt-align
 r b swpd free buff cache si so bi bo in cs us sy id wa st
0 1 0 153480 33320 214172 0 0 533 38 148 390 3 5 84 8 0
[root@hoge1:~]#
```

■ 良い点

- 横に間延びしたステータス表示を自動的に整列して見やすくします

■ その他

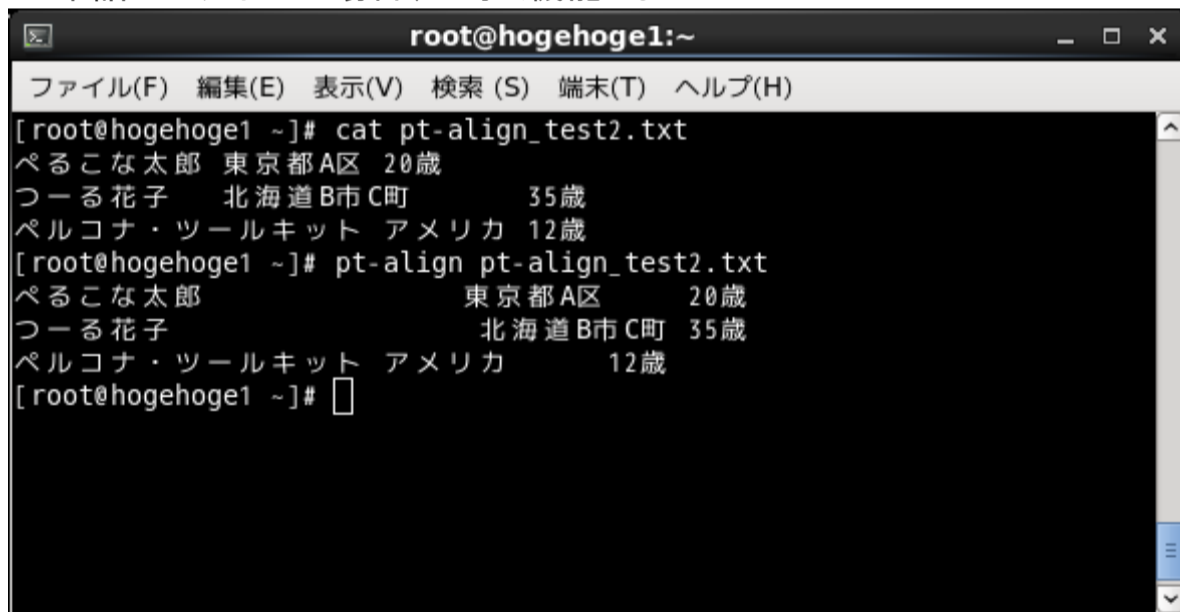
- ファイルを指定して、その内容をスペース区切りで分割して列幅を調整し、標準出力することも出来ます
- 下のように単語の長さがバラバラのテキストファイルも、pt-alignコマンドを使えば見やすく自動調整してくれます



```
root@hoge1:~  
[root@hoge1 ~]# cat pt-align_test.txt  
aaaa aa aaaa aaaaaaaa  
bbbbbbbbbbb bb bb b  
ccc ccc ccc ccc  
d d dd dd ddd  
ee ee ee ee  
[root@hoge1 ~]# pt-align pt-align_test.txt  
aaaa aa aaaa aaaaaaaa  
bbbbbbbbbbb bb bb b  
ccc ccc ccc ccc  
ee ee ee ee  
[root@hoge1 ~]#
```

※ スペースの個数を等しくする必要があります
上の例では、スペースの個数が異なる"d"の行が省略されています

- 日本語のファイルの場合、上手く機能しません



```
root@hoge1:~  
[root@hoge1 ~]# cat pt-align_test2.txt  
ぺるこな太郎 東京都A区 20歳  
つーる花子 北海道B市C町 35歳  
ペルコナ・ツールキット アメリカ 12歳  
[root@hoge1 ~]# pt-align pt-align_test2.txt  
ぺるこな太郎 東京都A区 20歳  
つーる花子 北海道B市C町 35歳  
ペルコナ・ツールキット アメリカ 12歳  
[root@hoge1 ~]#
```

■ コマンド

```
pt-archiver [ オプション ] --source [ DSN ] --file "ファイル名" --where "条件文"
```

【必須項目】

- [オプション] : -p パスワード
- --source [DSN] : h=ホスト名, D=データベース名, t=テーブル名
- --file "ファイル名" : アーカイブした内容を保存するファイルを指定します
- --where "条件文" : アーカイブするレコードの条件を指定します(全レコード取得の場合は "1=1" と指定)

【主なオプション】

- --no-delete : デフォルトではアーカイブしたレコードが削除されるため、それを避ける時はこのオプションを使います
- --no-check-charset : デフォルトではレコードとアーカイブファイルとの間の文字コードの違いをチェックします
このオプションを指定すれば、自動でデータベースの文字コードに合わせてくれます

■ 目的

- 指定したテーブルのレコードを、他のテーブルに取り込める形式のファイルにアーカイブします

■ 設定ファイル

- 上記の必須項目を、設定ファイルにまとめておきます

ただし、--source、--file、--where はコマンドライン上で直接指定する必要があるため、ここでは書きません

```
# touch /etc/percona-toolkit/pt-archiver.conf
```

設定ファイルの作成

```
# vi /etc/percona-toolkit/pt-archiver.conf
```

設定ファイルの編集

```
# config for pt-archiver
```

コメント

```
user=root
```

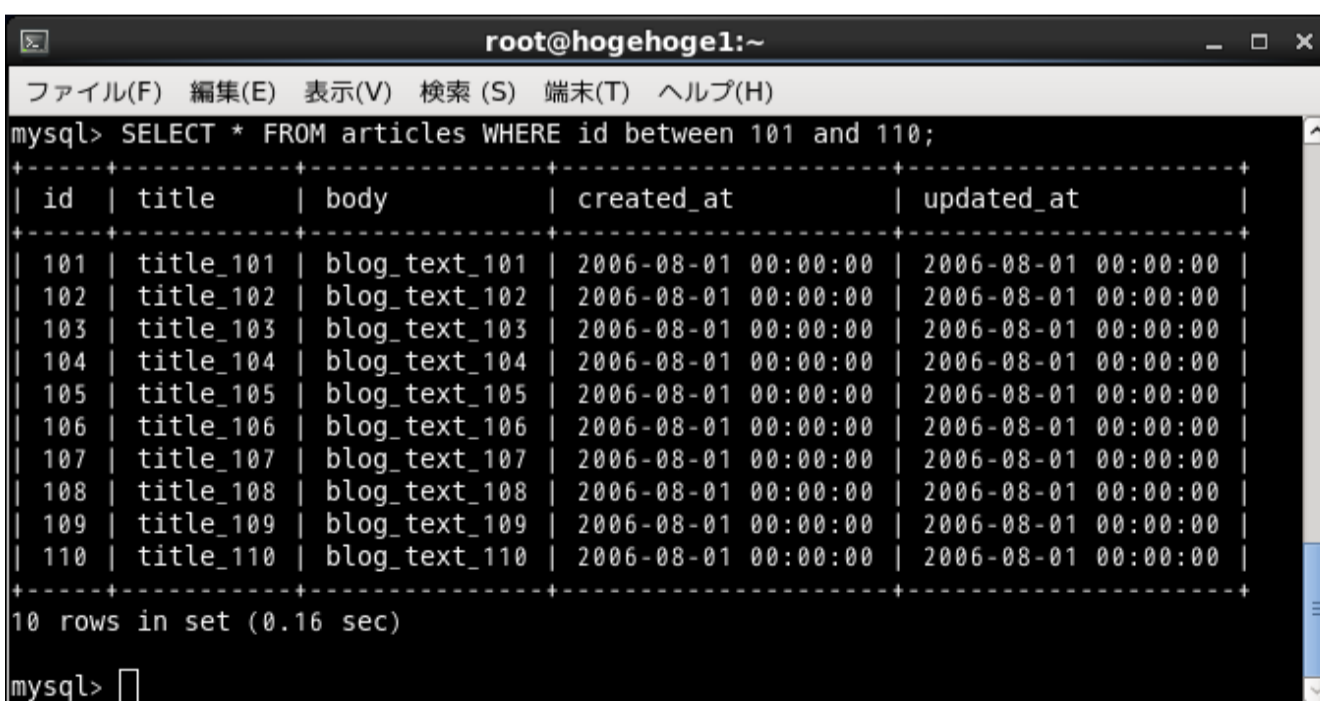
MySQL のユーザ名

```
password=パスワード
```

MySQL のパスワードを記載

■ シナリオ

- 以下のようなテーブルおよびレコードを用意します



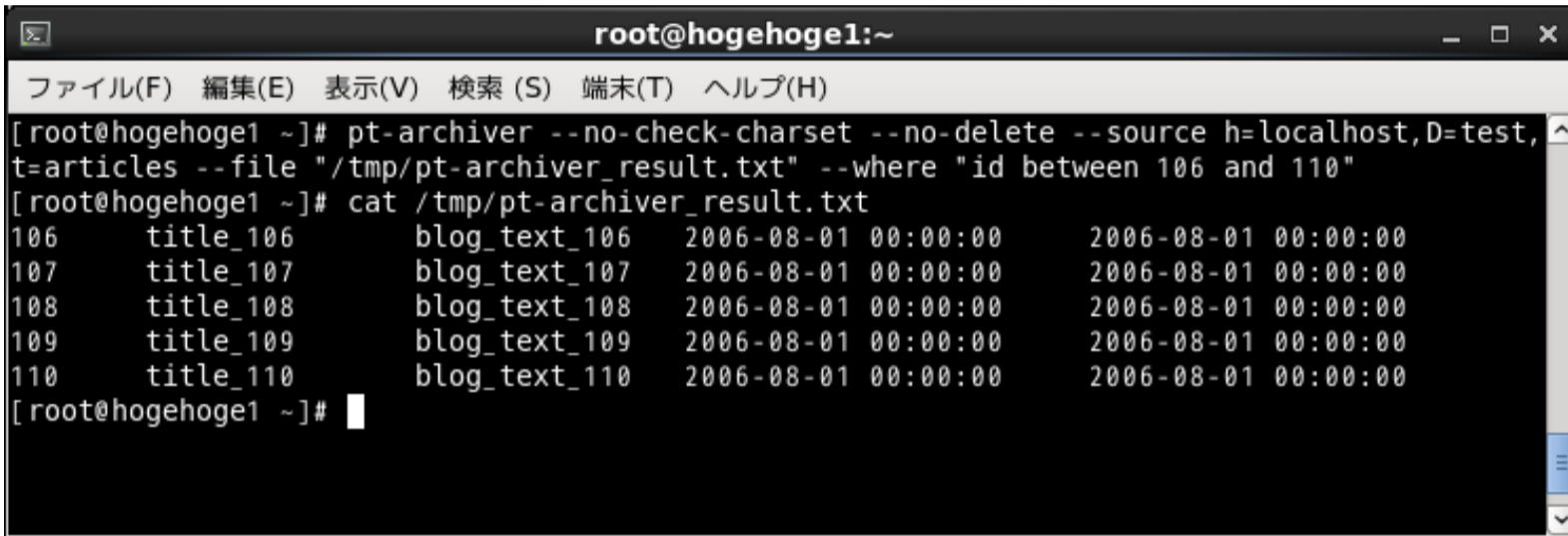
```
root@hoge1:~  
mysql> SELECT * FROM articles WHERE id between 101 and 110;  
+----+-----+-----+-----+-----+  
| id | title | body | created_at | updated_at |  
+----+-----+-----+-----+-----+  
| 101 | title_101 | blog_text_101 | 2006-08-01 00:00:00 | 2006-08-01 00:00:00 |  
| 102 | title_102 | blog_text_102 | 2006-08-01 00:00:00 | 2006-08-01 00:00:00 |  
| 103 | title_103 | blog_text_103 | 2006-08-01 00:00:00 | 2006-08-01 00:00:00 |  
| 104 | title_104 | blog_text_104 | 2006-08-01 00:00:00 | 2006-08-01 00:00:00 |  
| 105 | title_105 | blog_text_105 | 2006-08-01 00:00:00 | 2006-08-01 00:00:00 |  
| 106 | title_106 | blog_text_106 | 2006-08-01 00:00:00 | 2006-08-01 00:00:00 |  
| 107 | title_107 | blog_text_107 | 2006-08-01 00:00:00 | 2006-08-01 00:00:00 |  
| 108 | title_108 | blog_text_108 | 2006-08-01 00:00:00 | 2006-08-01 00:00:00 |  
| 109 | title_109 | blog_text_109 | 2006-08-01 00:00:00 | 2006-08-01 00:00:00 |  
| 110 | title_110 | blog_text_110 | 2006-08-01 00:00:00 | 2006-08-01 00:00:00 |  
+----+-----+-----+-----+-----+  
10 rows in set (0.16 sec)  
mysql>
```

- 以下のコマンドを実行し、上記のテーブルからidが106-110のレコードをアーカイブします。

```
# pt-archiver --no-check-charset --no-delete --source h=localhost,D=test,t=articles
--file "/tmp/pt-archiver_result.txt" --where "id between 106 and 110"
# cat /tmp/pt-archiver_result.txt
```

■ 結果

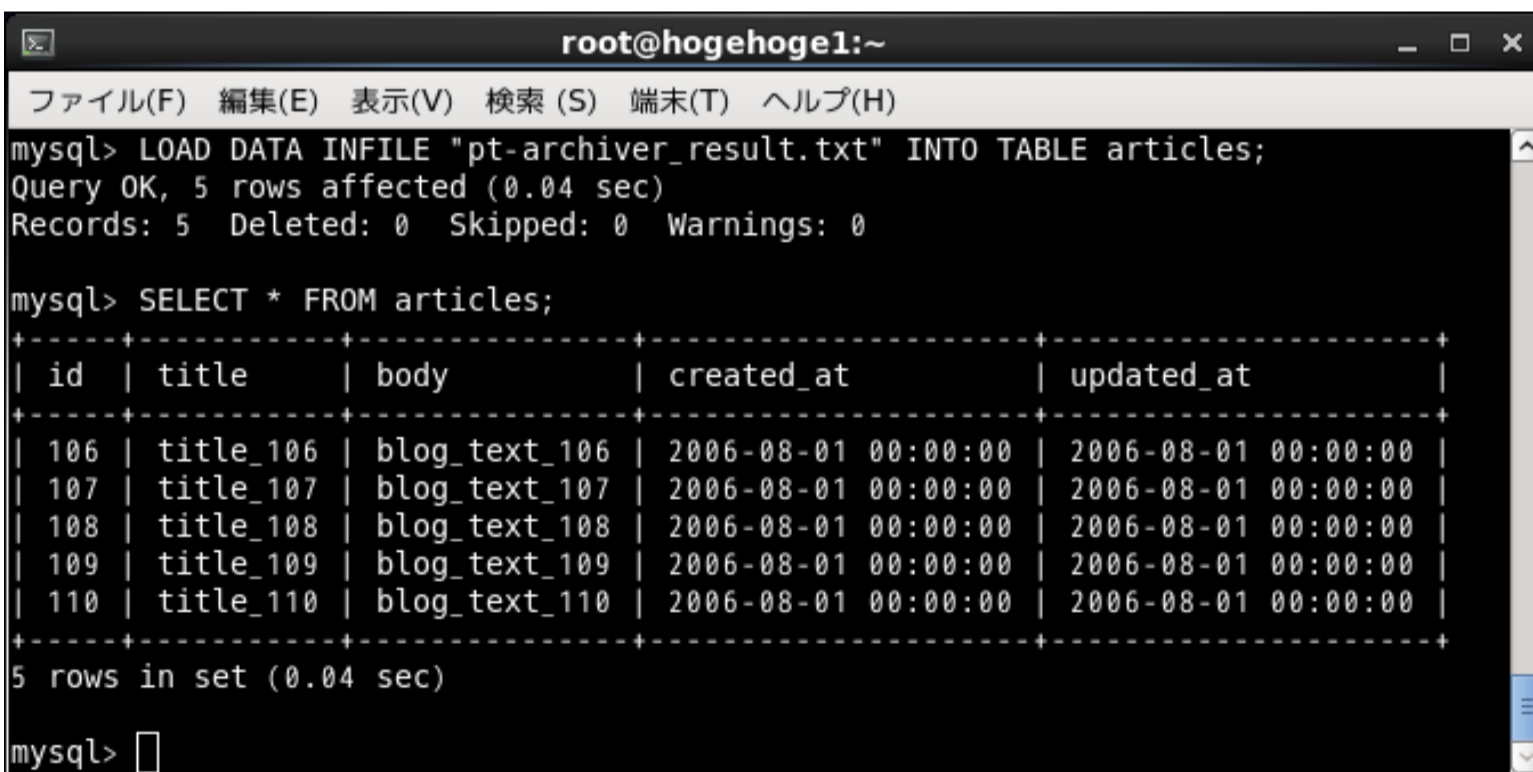
- pt-archiver_result.txt の中に以下のような内容が書き込まれます。



```
root@hoge1:~# pt-archiver --no-check-charset --no-delete --source h=localhost,D=test,t=articles --file "/tmp/pt-archiver_result.txt" --where "id between 106 and 110"
root@hoge1:~# cat /tmp/pt-archiver_result.txt
106 title_106 blog_text_106 2006-08-01 00:00:00 2006-08-01 00:00:00
107 title_107 blog_text_107 2006-08-01 00:00:00 2006-08-01 00:00:00
108 title_108 blog_text_108 2006-08-01 00:00:00 2006-08-01 00:00:00
109 title_109 blog_text_109 2006-08-01 00:00:00 2006-08-01 00:00:00
110 title_110 blog_text_110 2006-08-01 00:00:00 2006-08-01 00:00:00
root@hoge1:~#
```

このファイルを使えば、"LOAD DATA INFILE"文でレコードを取り込むことができます。

```
# cp /tmp/pt-archiver_result.txt /var/lib/mysql/test/
mysql> TRUNCATE articles;
mysql> LOAD DATA INFILE "pt-archiver_result.txt" INTO TABLE articles;
```



```
mysql> LOAD DATA INFILE "pt-archiver_result.txt" INTO TABLE articles;
Query OK, 5 rows affected (0.04 sec)
Records: 5 Deleted: 0 Skipped: 0 Warnings: 0

mysql> SELECT * FROM articles;
+----+-----+-----+-----+-----+
| id | title   | body      | created_at      | updated_at      |
+----+-----+-----+-----+-----+
| 106 | title_106 | blog_text_106 | 2006-08-01 00:00:00 | 2006-08-01 00:00:00 |
| 107 | title_107 | blog_text_107 | 2006-08-01 00:00:00 | 2006-08-01 00:00:00 |
| 108 | title_108 | blog_text_108 | 2006-08-01 00:00:00 | 2006-08-01 00:00:00 |
| 109 | title_109 | blog_text_109 | 2006-08-01 00:00:00 | 2006-08-01 00:00:00 |
| 110 | title_110 | blog_text_110 | 2006-08-01 00:00:00 | 2006-08-01 00:00:00 |
+----+-----+-----+-----+-----+
5 rows in set (0.04 sec)

mysql>
```

■ 良い点

- 行レベルで条件を指定して、別データベースへリストアする場合等に有効です

■ 注意点

- デフォルト設定で動かそうとすると、アーカイブされたレコードがテーブルから削除されてしまいます

■ その他

- pt-archiver で保存されるファイルの形式は、MySQLの"SELECT INTO OUTFILE"と同じです
各フィールドはタブ区切られ、1レコードが改行(LF)で終わります
NULL は"N"で表され、特殊文字は飛ばされます
- --file オプションでは、以下のような変数を用いてファイル名を付けることが可能です

%Y 西暦
%m 月
%d 日付
%H 時
%i 分
%s 秒

%D データベース名
%t テーブル名

今回のシナリオ内で実行したコマンドの --file オプション内を、

```
--file '/tmp/%Y-%m-%d-%D-%t.txt'
```

と設定すると、出力されるファイル名は" 2014-08-14-test-articles.txt "のようになります

■ コマンド

pt-config-diff [オプション] [ファイル名1] [ファイル名2]

■ 目的

- 指定した2つのMySQLの設定ファイルを比較して、その2つの差異を出力します

■ シナリオ

- 以下のような my.cnf と my2.cnf という2つの設定ファイルを用意します

my.cnf

```
[client]
port = 3306
socket = /var/lib/mysql/mysql.sock
[mysqld]
port = 3306
socket = /var/lib/mysql/mysql.sock
key_buffer_size = 256M
innodb_file_per_table
innodb_data_home_dir = /var/lib/mysql
innodb_log_file_size = 128M
character-set-server = utf8
slow_query_log = 1
slow_query_log_file = slow_queries.log
log-error = /var/lib/mysql/mysqld.err
```

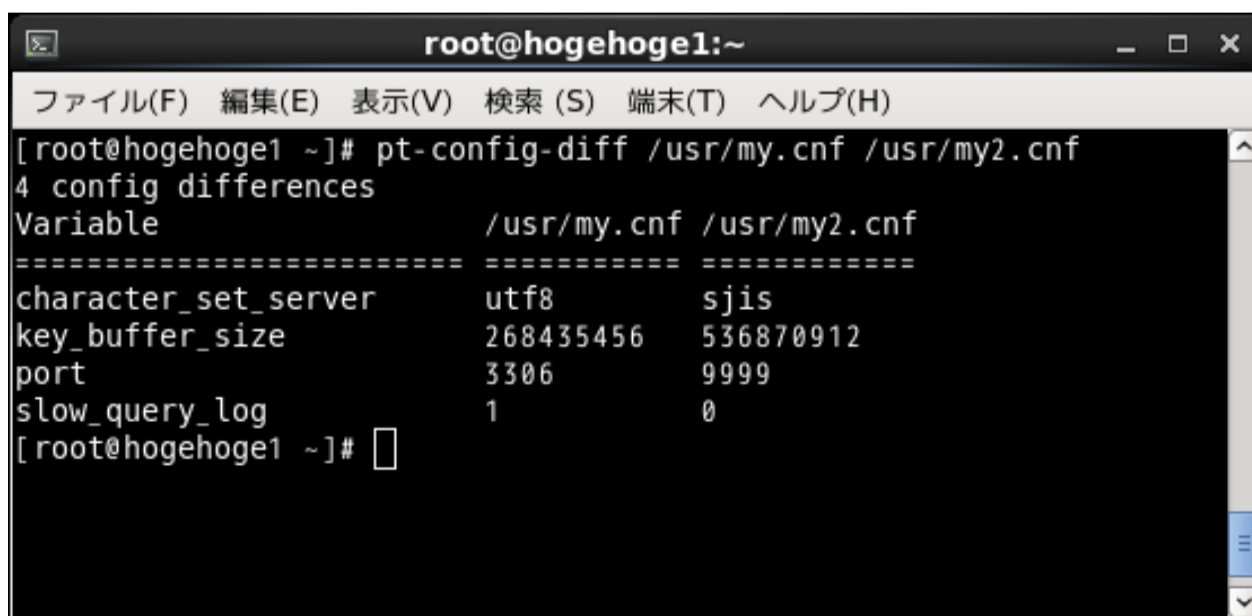
my2.cnf

```
[client]
port = 1000
socket = /var/lib/mysql/mysql2.sock
[mysqld]
port = 9999
socket = /var/lib/mysql/mysql.sock
key_buffer_size = 512M
innodb_data_home_dir = /var/lib/mysql
character-set-server = sjis
slow_query_log = 0
slow_query_log_file = slow_queries.log
log-error = /var/lib/mysql/mysqld.err
```

- pt-config-diff コマンドで両ファイルを比較し、差異を出力させます

■ 結果

- 以下のように、スクリプト上にファイル内で設定が異なる項目とその内容が標準出力されます



```
root@hoge1:~# pt-config-diff /usr/my.cnf /usr/my2.cnf
4 config differences
Variable          /usr/my.cnf /usr/my2.cnf
=====
character_set_server  utf8         sjis
key_buffer_size      268435456   536870912
port                 3306        9999
slow_query_log       1            0
[root@hoge1 ~]#
```

※ [client] の設定は比較の対象外となります

※ 2つの設定ファイルの両方に存在する項目のみが比較されます

(my.cnf のみに存在する innodb_file_per_table と innodb_log_file_size の2項目は比較対象外となります)

■ 良い点

- 異なるMySQL設定ファイルの差異を簡単に検出することができます

■ その他

- 設定ファイルの代わりにホスト名を指定することで、二つのホスト間のサーバ変数の差異を出力することも出来ます
例えば、稼働している二台のサーバを指定すると、「SHOW VARIABLES」の結果を比較して出力します

```
# pt-config-diff h=ホスト名1 h=ホスト名2
```

■ コマンド名

pt-deadlock-logger [オプション] [DSN]

【必須項目】

- [オプション] : -p パスワード

【主なオプション】

- --dest D=データベース名, t=テーブル名 : デッドロック情報を保存するテーブルを指定します
- --create-dest-table : 上記のオプションで指定したテーブルが存在しない場合、新規作成します

■ 目的

- データベースのデッドロック情報をダンプします

■ 設定ファイル

- 上記の必須項目を、設定ファイルにまとめておきます

```
# touch /etc/percona-toolkit/pt-deadlock-logger.conf  
# vi /etc/percona-toolkit/pt-deadlock-logger.conf
```

設定ファイルの作成
設定ファイルの編集

```
# config for pt-deadlock-logger  
# config for pt-deadlock-logger
```

コメント
MySQL のパスワードを記入

■ シナリオ

- デッドロック情報を保存するテーブルを作成するため、初回は保存用テーブル作成オプションをつけて実行します

```
(初回) # pt-deadlock-logger --create-dest-table --dest D=test, t=deadlocks  
(2回目以降) # pt-deadlock-logger --dest D=test, t=deadlocks
```

MySQL内でデッドロックを発生させると、作成したテーブルに情報が書き込まれます
コンソールを2つ起動し、MySQLへログインします

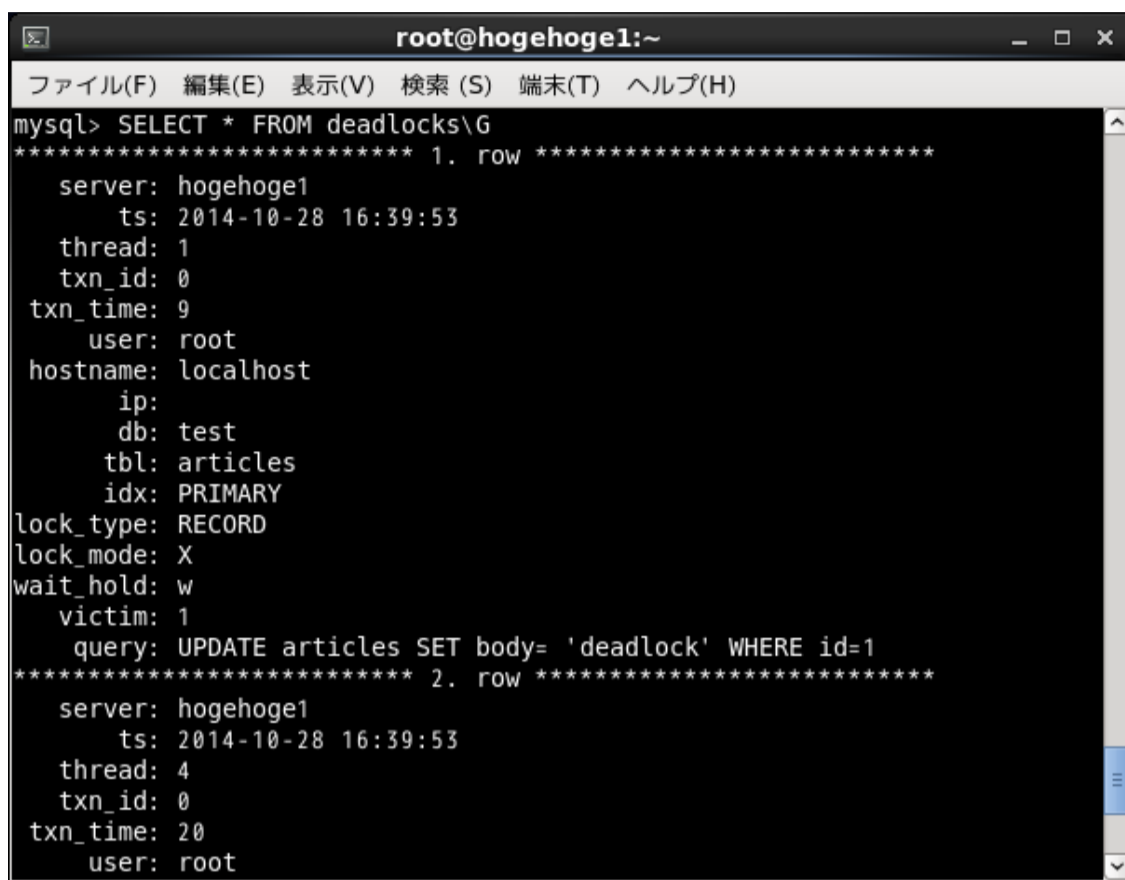
```
コンソール1 mysql> START TRANSACTION;  
コンソール2 mysql> START TRANSACTION;  
コンソール1 mysql> SELECT id, title, body FROM articles WHERE id=1 LOCK IN SHARE MODE;  
コンソール2 mysql> SELECT id, title, body FROM articles WHERE id=1 LOCK IN SHARE MODE;  
コンソール1 mysql> UPDATE articles SET body= 'deadlock' WHERE id=1;  
コンソール2 mysql> UPDATE articles SET body= 'deadlock' WHERE id=1;
```

→ これでデッドロックが発生するので、下記コマンドを実行して書き込まれた情報を確認します

```
mysql> SELECT * FROM deadlocks\G
```

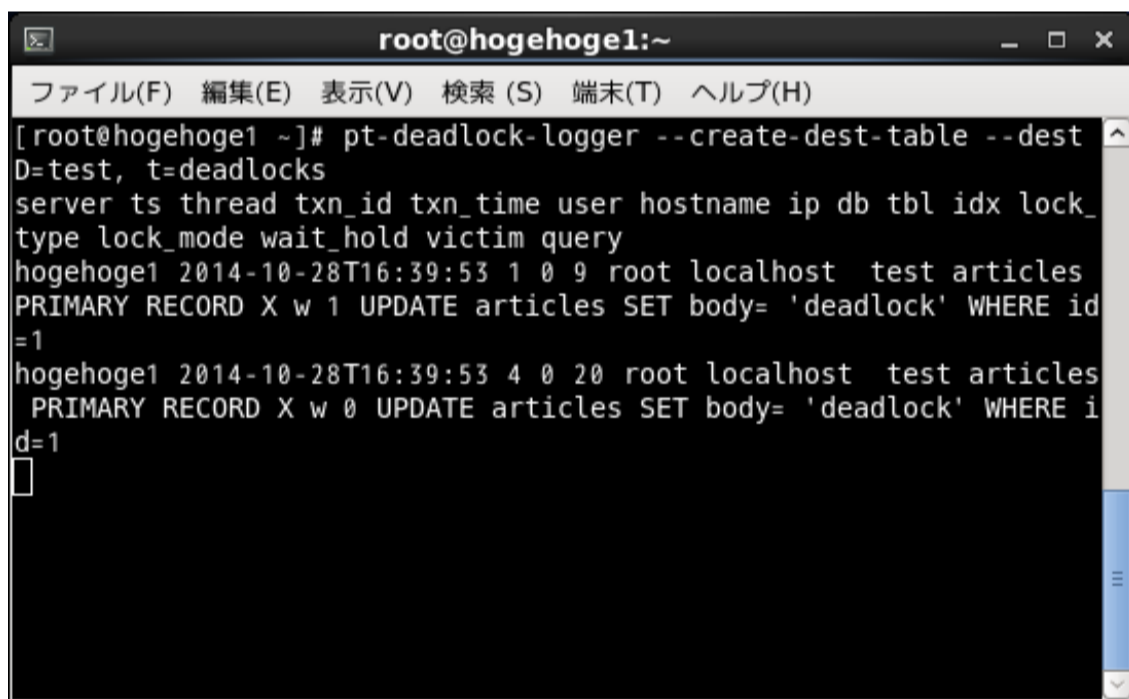
■ 結果

- 以下のように、エラー情報がdeadlocksテーブルに格納されていることが確認できます



```
root@hoge1:~# mysql> SELECT * FROM deadlocks\G
***** 1. row *****
server: hoge1
ts: 2014-10-28 16:39:53
thread: 1
txn_id: 0
txn_time: 9
user: root
hostname: localhost
ip:
db: test
tbl: articles
idx: PRIMARY
lock_type: RECORD
lock_mode: X
wait_hold: w
victim: 1
query: UPDATE articles SET body= 'deadlock' WHERE id=1
***** 2. row *****
server: hoge1
ts: 2014-10-28 16:39:53
thread: 4
txn_id: 0
txn_time: 20
user: root
```

- また、pt-deadlock-logger のコマンドを実行したコンソールにも情報が標準出力されます



```
root@hoge1:~# pt-deadlock-logger --create-dest-table --dest
D=test, t=deadlocks
server ts thread txn_id txn_time user hostname ip db tbl idx lock_
type lock_mode wait_hold victim query
hoge1 2014-10-28T16:39:53 1 0 9 root localhost test articles
PRIMARY RECORD X w 1 UPDATE articles SET body= 'deadlock' WHERE id
=1
hoge1 2014-10-28T16:39:53 4 0 20 root localhost test articles
PRIMARY RECORD X w 0 UPDATE articles SET body= 'deadlock' WHERE i
d=1
```

■ 良い点

- 期間を指定してバックグラウンドで起動させ、指定のデータベーステーブルへデッドロック情報を保存したりする事が可能なので、デッドロックの監視を行う際に有効です

■ その他

- MySQL5.6 以降では、innodb_print_all_deadlocks を有効にすると同様の情報がエラーログへ記録されます

■ コマンド

pt-diskstats [オプション] [ファイル]

【主なオプション】

- --interval 数字 : 統計情報を出力する間隔を指定します(デフォルトでは1秒)
- --iteration 数字 : 動作する期間を指定します(デフォルトでは Ctrl - C するまで動作し続けます)

■ 目的

- ディスクI/Oの統計情報を出力します

■ シナリオ

- 以下の pt-diskstats コマンドを実行します

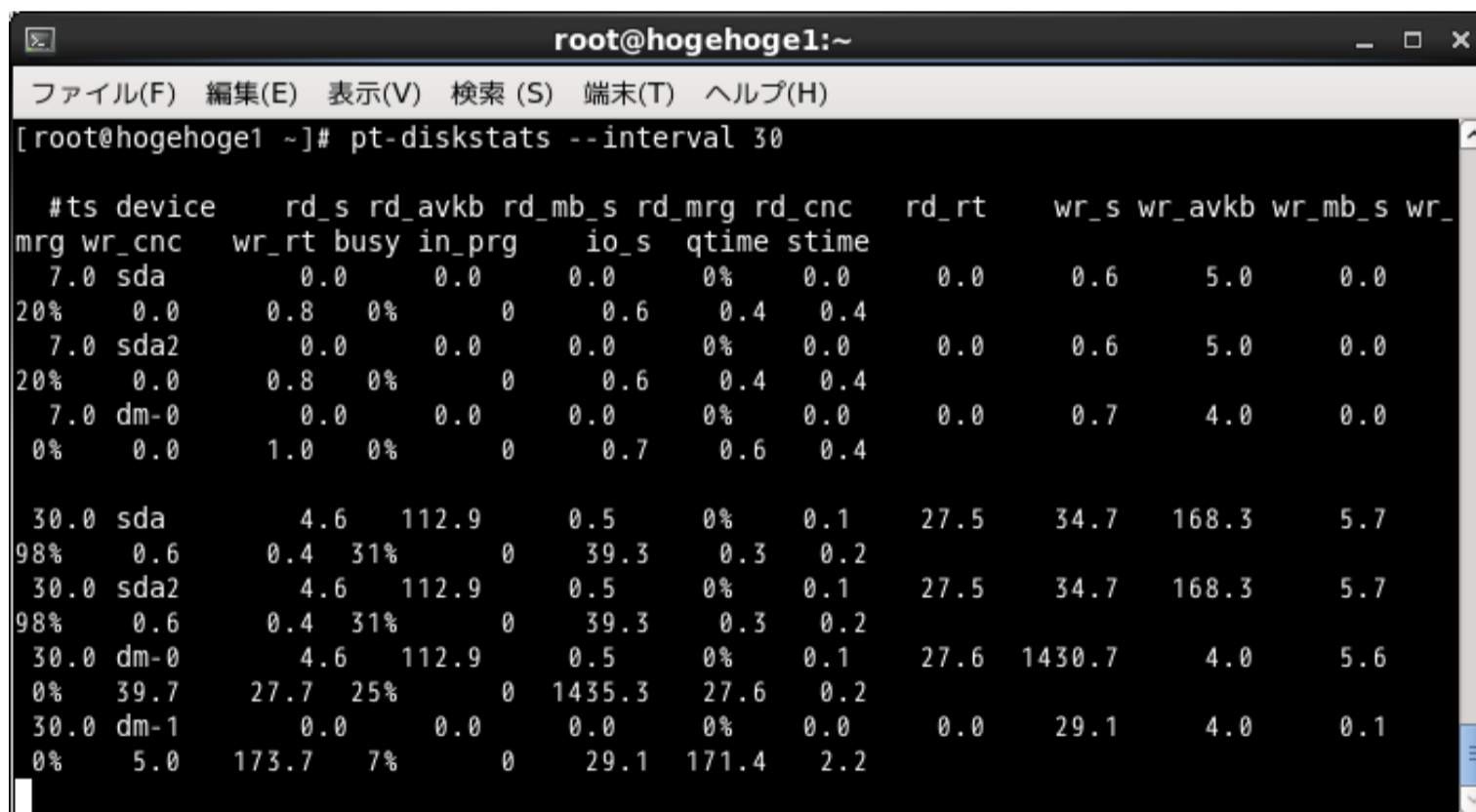
```
# pt-diskstats --interval 30
```

- 30秒後にMySQLで以下のコマンドを実行し、50万件のデータを取り込みます

```
mysql> LOAD DATA INFILE "14_2_articles.txt" INTO TABLE articles;
```

■ 結果

- 以下のように、ディスクI/Oの負荷状況が標準出力されます



```
root@hoge1:~  
[root@hoge1 ~]# pt-diskstats --interval 30  
#ts device      rd_s rd_avkb rd_mb_s rd_mrg rd_cnc      rd_rt      wr_s wr_avkb wr_mb_s wr_  
mrg wr_cnc      wr_rt busy  in_prg      io_s  qtime stime  
7.0 sda         0.0   0.0    0.0    0%    0.0    0.0    0.0   0.6   5.0    0.0  
20%  0.0    0.8   0%    0    0.6    0.4   0.4  
7.0 sda2        0.0   0.0    0.0    0%    0.0    0.0    0.0   0.6   5.0    0.0  
20%  0.0    0.8   0%    0    0.6    0.4   0.4  
7.0 dm-0        0.0   0.0    0.0    0%    0.0    0.0    0.0   0.7   4.0    0.0  
0%   0.0    1.0   0%    0    0.7    0.6   0.4  
  
30.0 sda         4.6  112.9    0.5    0%    0.1   27.5   34.7  168.3   5.7  
98%  0.6    0.4  31%    0   39.3    0.3   0.2  
30.0 sda2        4.6  112.9    0.5    0%    0.1   27.5   34.7  168.3   5.7  
98%  0.6    0.4  31%    0   39.3    0.3   0.2  
30.0 dm-0        4.6  112.9    0.5    0%    0.1   27.6  1430.7  4.0    5.6  
0%  39.7   27.7  25%    0  1435.3  27.6   0.2  
30.0 dm-1        0.0   0.0    0.0    0%    0.0    0.0    0.0   29.1   4.0    0.1  
0%   5.0  173.7   7%    0   29.1  171.4   2.2
```

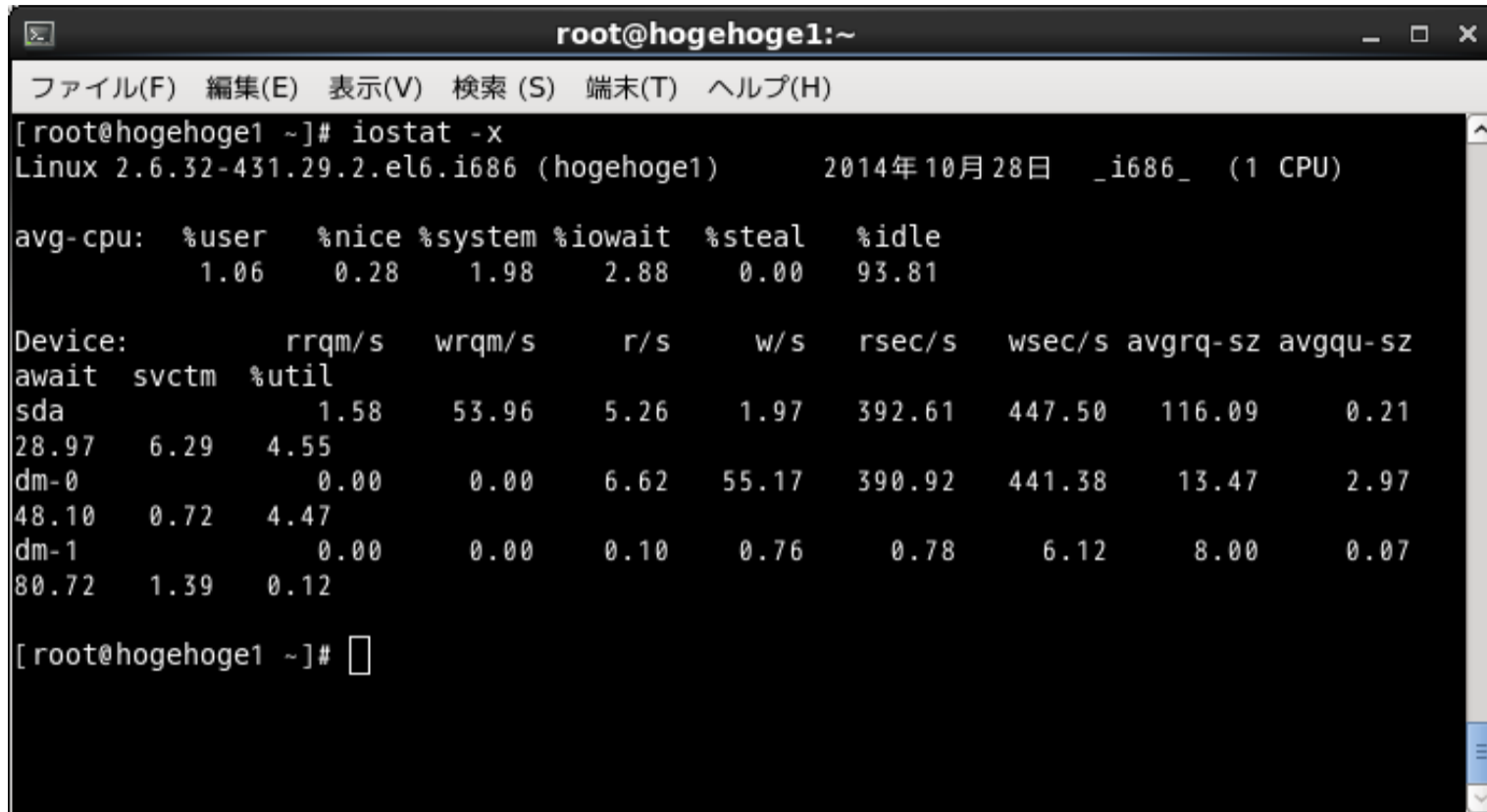
“LOAD DATA”分が実行されている間にディスクI/Oの値が上昇していることが分かります

■ 良い点

- sysstatパッケージをインストールしなくても、iostatコマンドと同様の機能を使うことができます
- オプションを用いれば、定期的にI/O状況を出力することも可能です

■ その他

- 「iostat -x」というコマンドを実行すると、同様の結果が得られます



```
root@hoge1:~  
[root@hoge1 ~]# iostat -x  
Linux 2.6.32-431.29.2.el6.i686 (hoge1)      2014年10月28日  _i686_  (1 CPU)  
  
avg-cpu:  %user   %nice %system %iowait  %steal   %idle  
           1.06    0.28   1.98   2.88    0.00   93.81  
  
Device:            rrqm/s   wrqm/s     r/s     w/s    rsec/s   wsec/s  avgrq-sz  avgqu-sz  
await  svctm  %util  
sda      1.58    53.96     5.26    1.97   392.61   447.50   116.09    0.21  
28.97    6.29    4.55  
dm-0     0.00     0.00     6.62   55.17   390.92   441.38   13.47    2.97  
48.10    0.72    4.47  
dm-1     0.00     0.00     0.10    0.76    0.78    6.12    8.00    0.07  
80.72    1.39    0.12  
  
[root@hoge1 ~]#
```

■ コマンド名

pt-duplicate-key-checker [オプション] [DSN]

【必須項目】

- [DSN] : D=データベース名, t=テーブル名, h=ホスト名
- [オプション] : -u ユーザ名, -p パスワード

■ 目的

- テーブルのインデックスや外部キーの重複チェックをします
- 指定されたデータベース上のテーブルから重複しているインデックスや外部キーを検索し、それを削除するSQL文を表示します

■ 設定ファイル

- 上記の必須項目を、設定ファイルにまとめておきます

```
# touch /etc/percona-toolkit/pt-duplicate-key-checker.conf  
# vi /etc/percona-toolkit/pt-duplicate-key-checker.conf
```

設定ファイルの作成
設定ファイルの編集

```
# config for pt-duplicate-key-checker  
user=ユーザ名  
password=パスワード  
h=ホスト名  
d=データベース名  
t=テーブル名
```

コメント
MySQL のユーザ名
MySQL のパスワード
ホスト名
データベース名
テーブル名

■ シナリオ

- 以下のコマンドで、"test"データベースの"articles"テーブルに、"id"をインデックスとして登録します

```
mysql> CREATE INDEX idx_id ON articles(id);
```

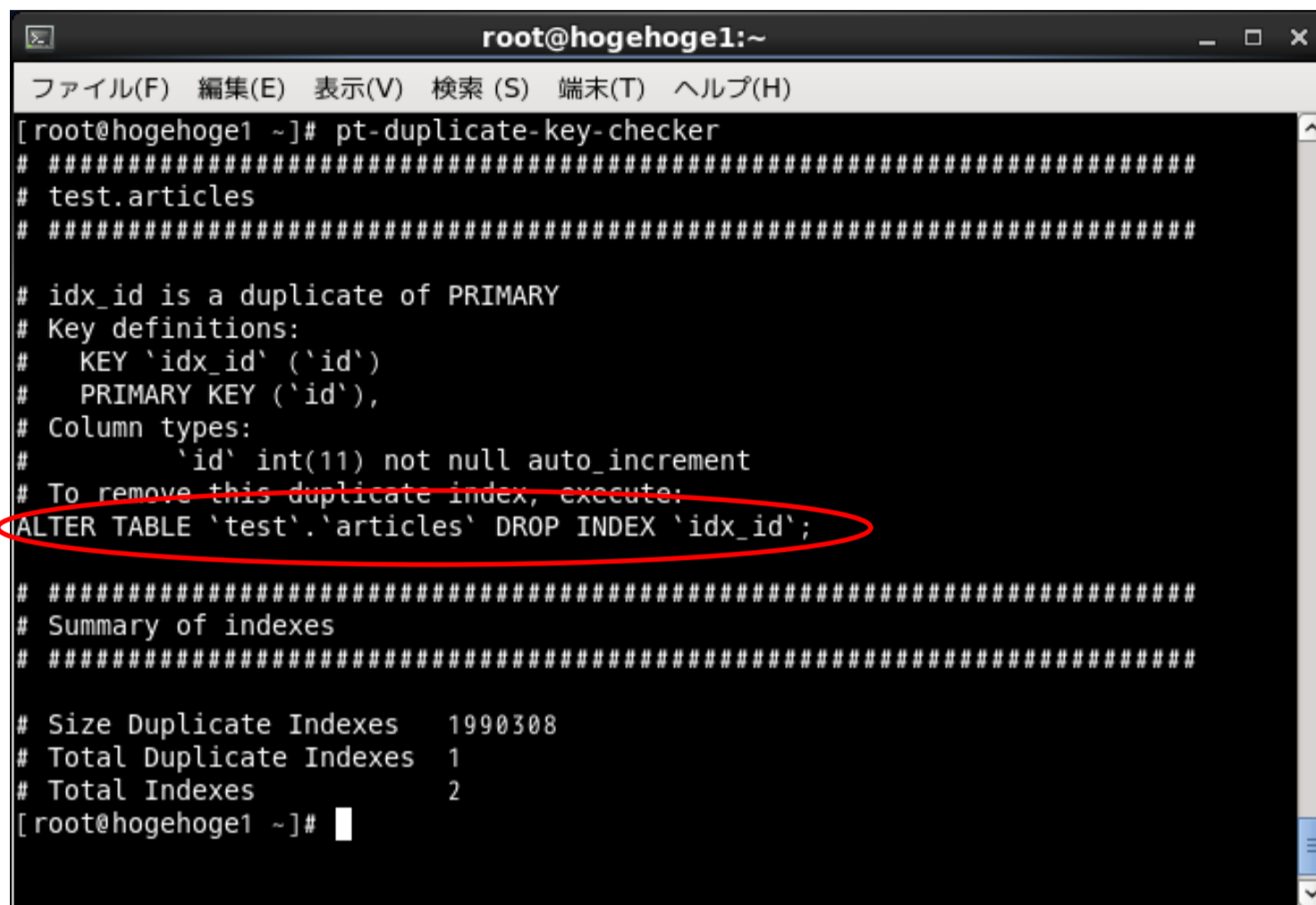
"id"カラムはプライマリーキーであるため、インデックスとして機能します
そのため、"idx_id"は重複したインデックスとして認識されます

- 以下のコマンドで、重複したインデックスをチェックします

```
# pt-duplicate-key-checker
```

■ 結果

- "idx_id"が重複したインデックスとして検知されました
さらに、赤丸の部分に重複しているインデックスを削除する ALTER文が記述されています



```
root@hoge1:~# pt-duplicate-key-checker
#####
# test.articles
# #####

# idx_id is a duplicate of PRIMARY
# Key definitions:
#   KEY `idx_id` (`id`)
#   PRIMARY KEY (`id`),
# Column types:
#   `id` int(11) not null auto_increment
# To remove this duplicate index, execute:
ALTER TABLE `test`.`articles` DROP INDEX `idx_id`;
#####

# Summary of indexes
# #####

# Size Duplicate Indexes    1990308
# Total Duplicate Indexes   1
# Total Indexes             2
[root@hoge1:~]#
```

■ 良い点

- インデックス及び、外部キーに重複がないかを容易に確認する事ができます

■ コマンド

pt-fifo-split [オプション] [ファイル]

【主なオプション】

- --fifo : 作成する一時ファイルの名前を指定します。デフォルトでは /tmp/pt-fifo-split が作られます
- --lines : 一度に読み込む行数を指定します。通常は1000行になっています

■ 目的

- ファイルから指定した行範囲ずつ、内容をFIFO(First In, First Out)で一時ファイルに読み込みます

■ シナリオ

- 読み込むデータファイルを用意し、以下の pt-fifo-split コマンドを実行します

```
# pt-fifo-split --lines 100 /tmp/mysql_tmp/14_2_articles.txt
```

- 別コンソールを開き、一時ファイルの中身を確認します

```
# cat /tmp/pt-fifo-split
```

■ 結果

- 一時ファイルの内容は、ファイルから指定した行数だけ読み込んだものになっています(上のコマンドでは100行)
- ※ 再度ファイルを確認すると、次の指定行数だけ読み込んだ内容になっています
- ※ 最後まで読み込むと、一時ファイルは自動的に削除されます

■ 良い点

- サイズの大きいファイルを物理的に分割することなく、指定行ずつ読み込むことができるので、大量のデータを一度に読み込む際のリスクを減らすことができます

■ コマンド名

pt-find [オプション] [データベース名]

【必須項目】

- [オプション] : -p パスワード

【主なオプション】

- --engine : 検索したいテーブルのエンジンを指定します
- --exec : 指定したテーブルに対して実行したいSQLを指定できます

■ 目的

- MySQLのテーブルを指定したアクションに基づいて検索を行い、該当のデータベース名、テーブル名を出力したり、指定条件に基づくテーブルに対して、SQLを実行する事ができます

■ 設定ファイル

- 上記の必須項目を、設定ファイルにまとめておきます

```
# touch /etc/percona-toolkit/pt-find.conf  
# vi /etc/percona-toolkit/pt-find.conf
```

設定ファイルの作成
設定ファイルの編集

```
# config for pt-deadlock-logger  
password=パスワード
```

コメント
MySQL のパスワード

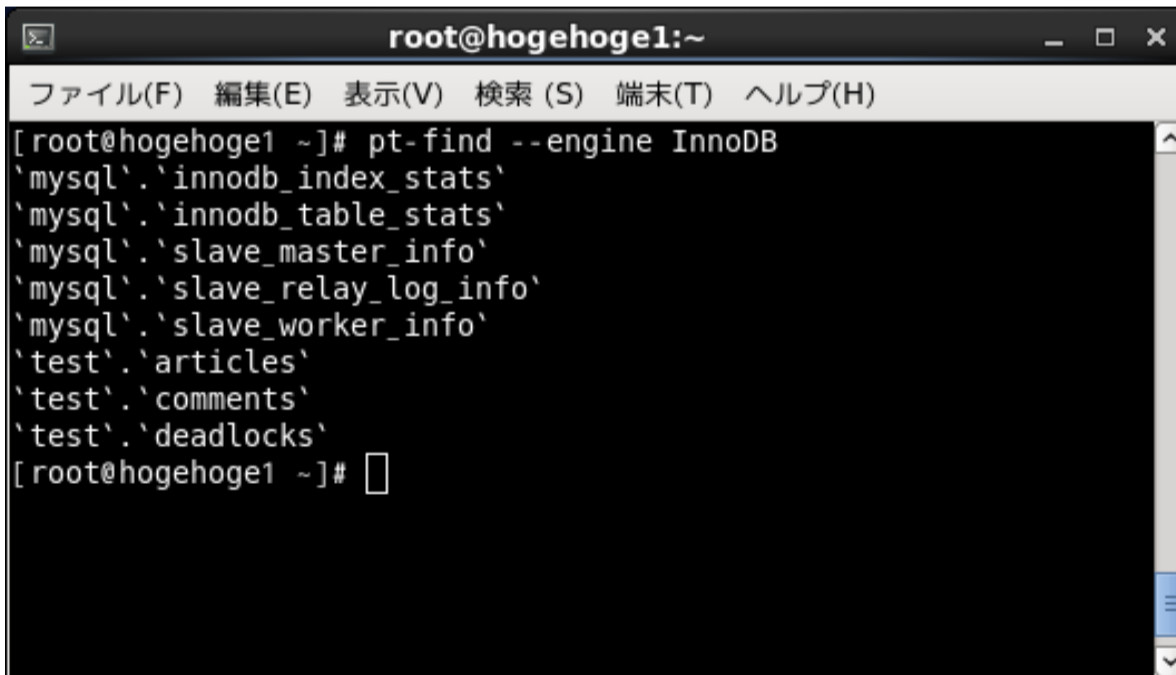
■ シナリオ

- 以下のコマンドを実行し、エンジンがInnoDBであるテーブルを検索します

```
# pt-find --engine InnoDB
```

■ 結果

- エンジンがInnoDBのテーブルが標準出力されます



```
root@hoge1:~  
[root@hoge1 ~]# pt-find --engine InnoDB  
'mysql`.`innodb_index_stats`  
'mysql`.`innodb_table_stats`  
'mysql`.`slave_master_info`  
'mysql`.`slave_relay_log_info`  
'mysql`.`slave_worker_info`  
'test`.`articles`  
'test`.`comments`  
'test`.`deadlocks`  
[root@hoge1 ~]#
```

■ 良い点

- 指定のテーブルサイズを超過しているテーブルを確認したり、エンジンがInnoDBであるテーブル全てに対して、MyISAM エンジンへ変更するALTER文を実行したりする事が可能です

■ その他の実行例

- 1日以上前に作成された、MyISAMエンジンのテーブルを検索します

```
# pt-find --ctime +1 --engine MyISAM
```

- InnoDB のテーブルを検索し、それらを MyISAM に変更します

```
# pt-find --engin InnoDB --exec "ALTER TABLE %D.%N ENGINE=MyISAM"
```

- "test" と "junk" データベースから、空のテーブルを検索して削除します

```
# pt-find --empty junk test --exec-plus "DROP TABLE %s"
```

- トータルで5GB以上のテーブルを検索します

```
# pt-find --totalsize +5G
```

- 全てのテーブルを、そのデータサイズ、インデックスサイズとともに検索し、大きなテーブルから順に表示します

```
# pt-find --printf "%T\t%D.%N\n" | sort -rn
```

■ コマンド

pt-fingerprint [オプション][ファイル]

【主なオプション】

- --query "クエリ" : 指定したクエリを正規化します

■ 目的

改行や書き方の異なる複数のクエリを正規化した同一のクエリになるよう変換して出力します

■ シナリオ

- テストファイルを作成し、以下の内容を書き込みます

```
# touch fingerprint_test.txt  
# vi fingerprint_test.txt
```

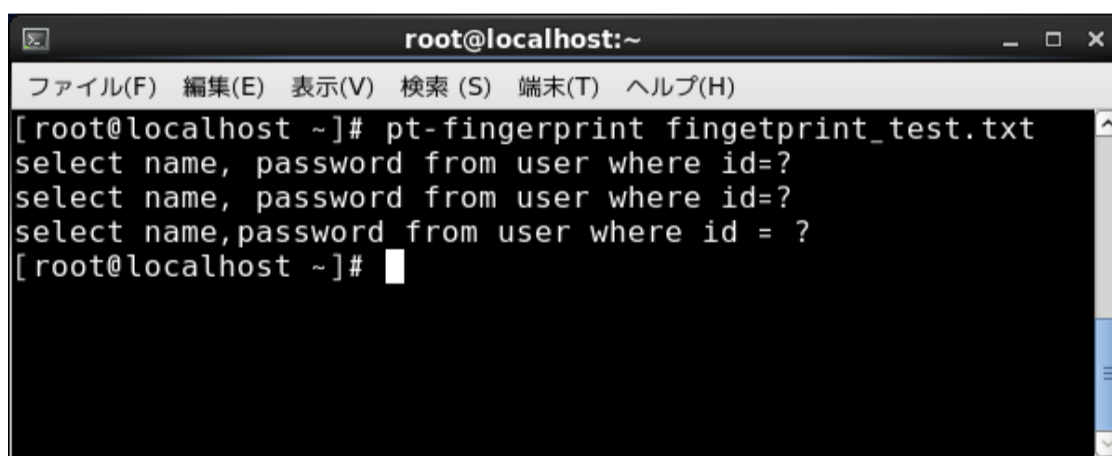
```
SELECT name, password FROM user WHERE id='12823';  
  
select name, password from user  
where id=5;  
  
SELECT name,password FROM user  
WHERE id = 39  
;
```

- 以下のコマンドを実行します

```
# pt-fingerprint fingerprint_test.txt
```

■ 結果

- 各クエリを正規化した結果が出力されます



A terminal window titled 'root@localhost:~' showing the execution of the 'pt-fingerprint' command. The command is 'pt-fingerprint fingerprint_test.txt'. The output shows three normalized SQL queries, each on a new line: 'select name, password from user where id=?', 'select name, password from user where id=?', and 'select name,password from user where id = ?'. The terminal prompt is '[root@localhost ~]#'.

■ 良い点

- 値の違いやスペース、改行等の違いのみである、同一クエリと判断できるクエリを収集する事ができます

■ コマンド

pt-fk-error-logger [オプション] [DSN]

【必須項目】

- [オプション] : -p パスワード
- [DSN] : h=ホスト名, D=データベース名

【主なオプション】

- --dest : 検出したエラー情報を格納するテーブルを指定します(デフォルトでは標準出力となります)
- --run-time 数字 : 動作する期間を指定します(デフォルトでは Ctrl - C するまで動作し続けます)

■ 目的

- 外部キー制約に関するエラー情報を標準出力します

■ 設定ファイル

- 上記の必須項目を、設定ファイルにまとめておきます
ただし、DSNはコマンドライン上で直接指定する必要があるため、ここでは書きません

```
# touch /etc/percona-toolkit/pt-fk-error-logger.conf  
# vi /etc/percona-toolkit/pt-fk-error-logger.conf
```

設定ファイルの作成
設定ファイルの編集

```
# config for pt-fk-error-logger  
user=root  
password=パスワード
```

コメント
MySQL のユーザ名
MySQL のパスワードを記載

■ シナリオ

- 以下のような親テーブルを用意します



```
root@hoge1:~  
mysql> DESC articles;  
+-----+-----+-----+-----+-----+-----+  
| Field      | Type          | Null | Key | Default | Extra          |  
+-----+-----+-----+-----+-----+-----+  
| id         | int(11)       | NO   | PRI | NULL    | auto_increment |  
| title      | varchar(255)  | YES  | MUL | NULL    |                |  
| body       | text          | YES  |     | NULL    |                |  
| created_at | datetime      | YES  |     | NULL    |                |  
| updated_at | datetime      | YES  |     | NULL    |                |  
+-----+-----+-----+-----+-----+-----+  
5 rows in set (0.12 sec)  
mysql>
```

- 外部キーをつけるために titleカラムにインデックスを作成します

```
mysql> CREATE INDEX idx_title ON articles(title);
```

- 以下のCREATE文を実行し、子テーブルを作成します

```
mysql> CREATE TABLE articles_second (id int(11) AUTO_INCREMENT, title varchar(255)  
created_at datetime, updated_at datetime, PRIMARY KEY(id),  
FOREIGN KEY(title) REFERENCES articles(title) ) ENGINE=InnoDB;
```

- pt-fk-error-loggerコマンドを実行します

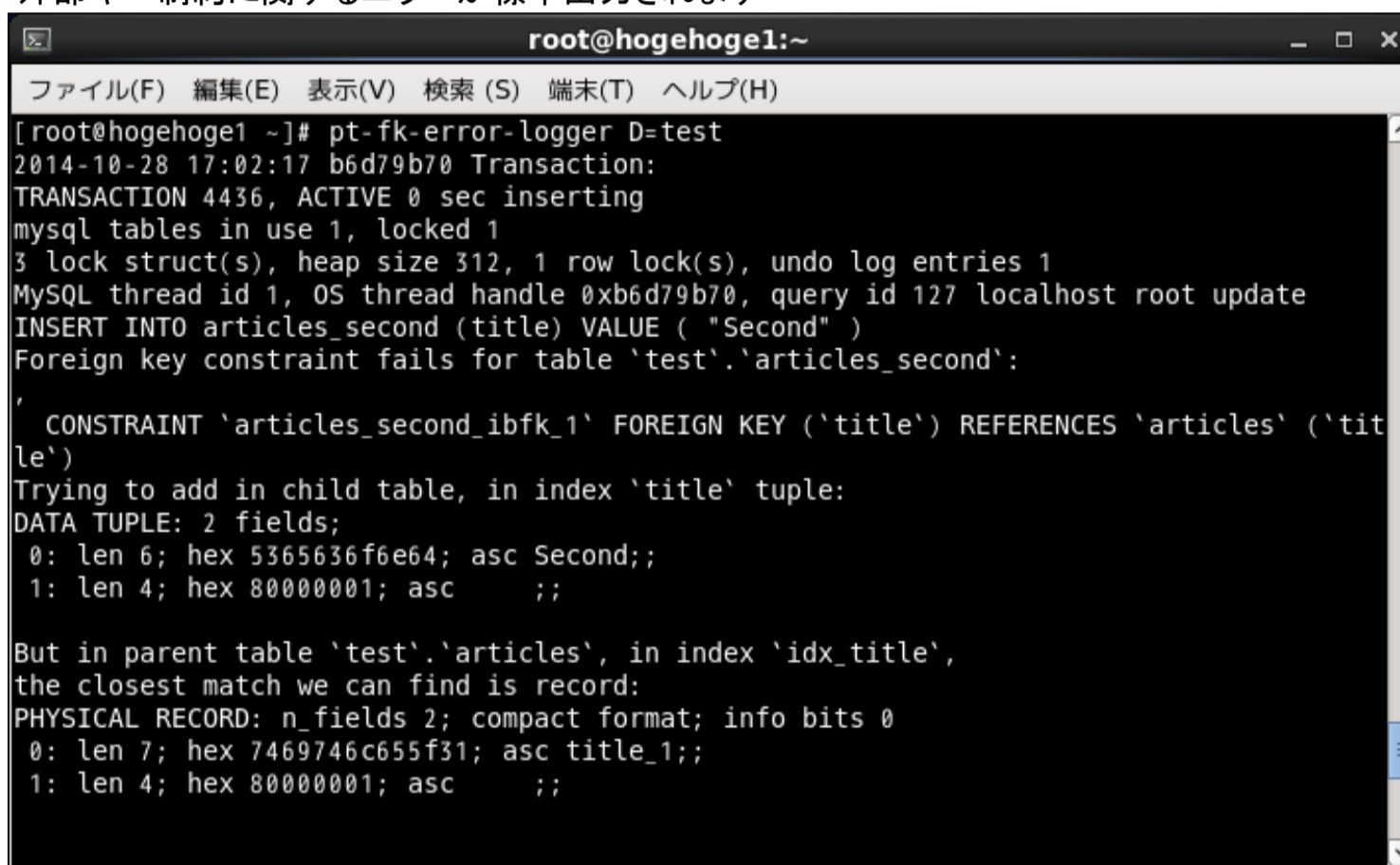
```
# pt-fk-error-logger D=test
```

- 以下のSQL文を実行し、外部キー制約エラーを発生させます

```
mysql> INSERT INTO articles (title) VALUE ( "First" );  
mysql> INSERT INTO articles_second (title) VALUE ( "Second" );
```

■ 結果

- 外部キー制約に関するエラーが標準出力されます



```
root@hoge1:~# pt-fk-error-logger D=test  
2014-10-28 17:02:17 b6d79b70 Transaction:  
TRANSACTION 4436, ACTIVE 0 sec inserting  
mysql tables in use 1, locked 1  
3 lock struct(s), heap size 312, 1 row lock(s), undo log entries 1  
MySQL thread id 1, OS thread handle 0xb6d79b70, query id 127 localhost root update  
INSERT INTO articles_second (title) VALUE ( "Second" )  
Foreign key constraint fails for table `test`.`articles_second`:  
  
  CONSTRAINT `articles_second_ibfk_1` FOREIGN KEY (`title`) REFERENCES `articles` (`title`)  
Trying to add in child table, in index `title` tuple:  
DATA TUPLE: 2 fields;  
 0: len 6; hex 536563666e64; asc Second;;  
 1: len 4; hex 80000001; asc      ;;  
  
But in parent table `test`.`articles`, in index `idx_title`,  
the closest match we can find is record:  
PHYSICAL RECORD: n_fields 2; compact format; info bits 0  
 0: len 7; hex 7469746c651f31; asc title_1;;  
 1: len 4; hex 80000001; asc      ;;
```

■ 良い点

- 外部キー制約に関するエラーを監視することが出来ます

■ その他

- 標準出力の場合、最新のエラーを1件しか表示することが出来ません
そのため、エラー情報を保持するためには出力内容をMySQL内のテーブルに格納する必要があります

以下のようなテーブルを作成します

```
mysql> CREATE TABLE foreign_key_errors (ts datetime NOT NULL,  
    error text NOT NULL, PRIMARY KEY (ts));
```

以下のコマンドを実行すると、エラー情報が foreign_key_errors テーブルに書き込まれます

```
# pt-fk-error-logger h=localhost --dest h=localhost,D=blog_test,t=foreign_key_errors
```

■ コマンド

pt-heartbeat [オプション] [DSN] [--update | --monitor | --check | --stop]

【必須項目】

- [--update | --monitor | --check | --stop] の中から少なくとも一つを指定します
 - update : マスタサーバに接続した時間(タイムスタンプ)を更新します
 - monitor : 指定した間隔でレプリケーションの遅延状況を出力します
 - check : レプリケーションの遅延状況を一度だけ出力します
 - stop : pt-heartbeat-sentinel ファイルを生成します。このファイルが存在するとupdateとmonitorが機能しなくなります
- --master-server-id : マスタサーバのサーバIDを指定します
- --create-table : 遅延状況を記録するheartbeatテーブルを作成します(初めてコマンドを使用する時のみ指定)
- [オプション] : -p パスワード -D データベース名

【主なオプション】

- --interval : 遅延状況を出力する間隔を指定します(デフォルトでは1秒です)

■ 目的

- MySQLサーバのレプリケーションの遅延を測定します

■ 設定ファイル

- 上記の必須項目を、設定ファイルにまとめておきます
ただし、DSNはコマンドライン上で直接指定する必要があるため、ここでは書きません

```
# touch /etc/percona-toolkit/pt-heartbeat.conf  
# vi /etc/percona-toolkit/pt-heartbeat.conf
```

設定ファイルの作成
設定ファイルの編集

```
# config for pt-heartbeat.conf  
D=repl_test  
password=パスワード
```

コメント
測定するデータベース名
マスタ のパスワードを記載

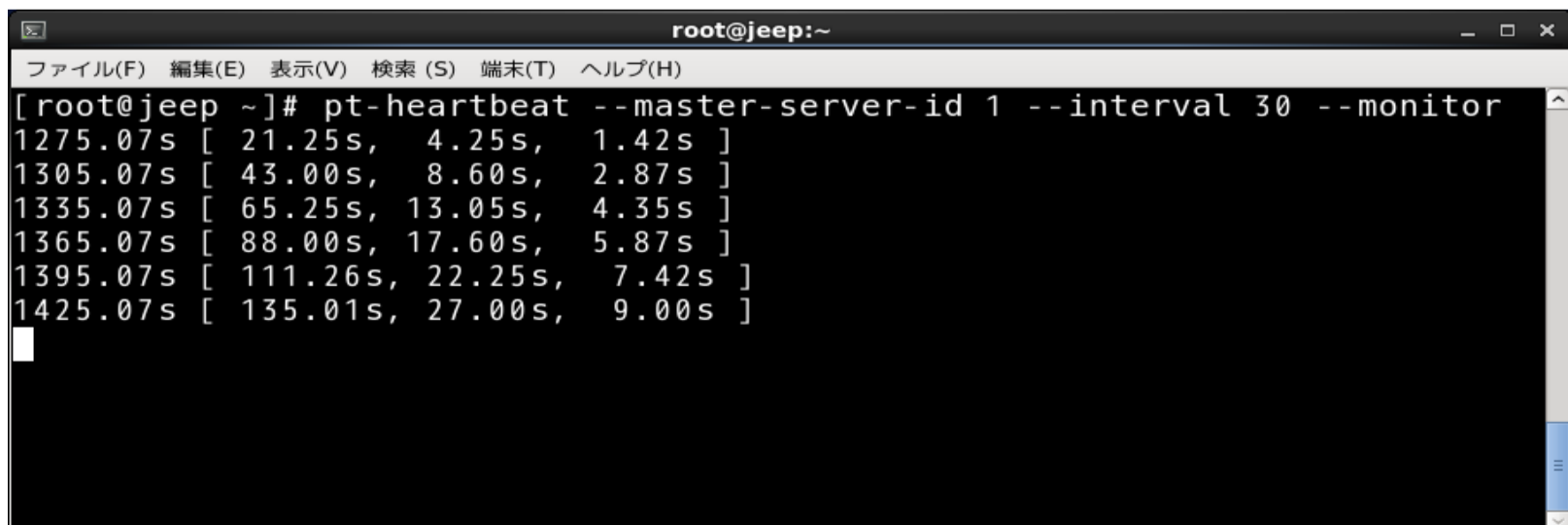
■ シナリオ

- マスタサーバで以下のSQL文を実行し、レプリケーション用のデータベースを作成します

```
mysql> CREATE DATABASE repl_test
```

- マスタサーバで通常時の遅延状況を30秒間隔で測定します

```
# pt-heartbeat --master-server-id 1 --interval 30 --monitor
```



```
root@jeep:~  
[root@jeep ~]# pt-heartbeat --master-server-id 1 --interval 30 --monitor  
1275.07s [ 21.25s, 4.25s, 1.42s ]  
1305.07s [ 43.00s, 8.60s, 2.87s ]  
1335.07s [ 65.25s, 13.05s, 4.35s ]  
1365.07s [ 88.00s, 17.60s, 5.87s ]  
1395.07s [ 111.26s, 22.25s, 7.42s ]  
1425.07s [ 135.01s, 27.00s, 9.00s ]
```

- もう一度同じコマンドを実行し、その後50万件のレコードを読み込むことで遅延を発生させます

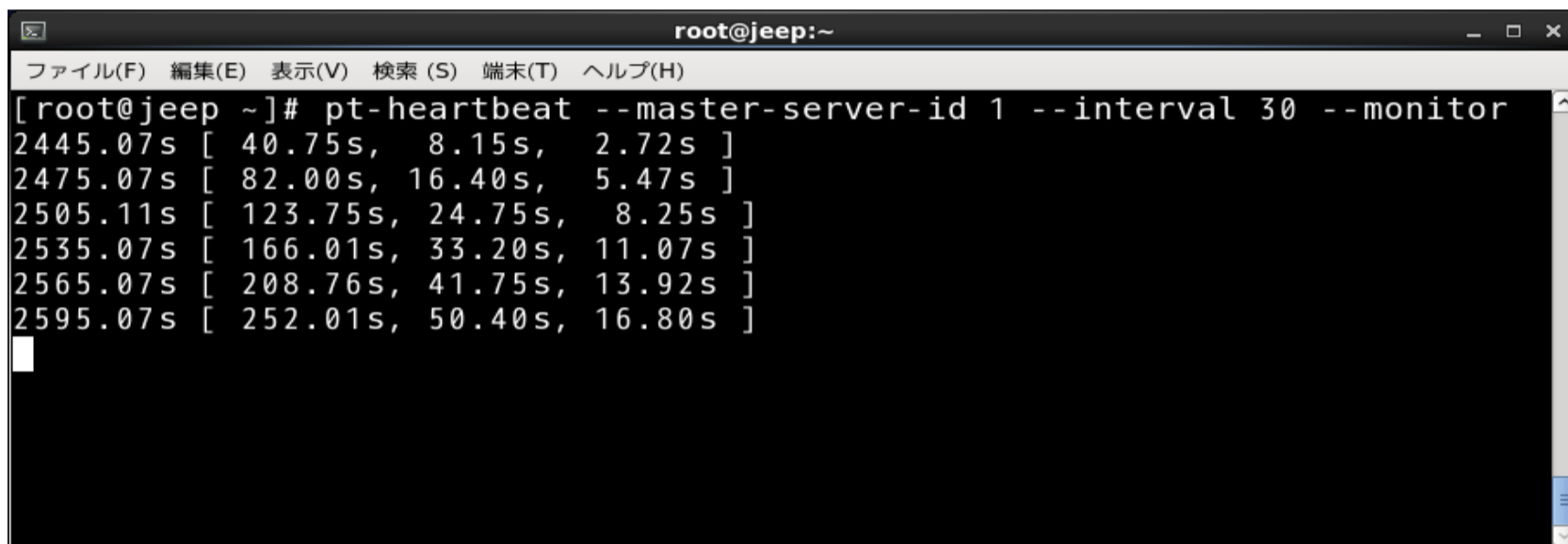
```
# pt-heartbeat --master-server-id 1 --interval 30 --monitor
```

(別コンソールで以下を実行)

```
mysql> LOAD DATA INFILE "14_2_articles.txt" INTO TABLE articles;
```

■ 結果

- 先に測定した時よりもレプリケーションが遅延していることが確認できます



```
root@jeep:~  
[root@jeep ~]# pt-heartbeat --master-server-id 1 --interval 30 --monitor  
2445.07s [ 40.75s, 8.15s, 2.72s ]  
2475.07s [ 82.00s, 16.40s, 5.47s ]  
2505.11s [ 123.75s, 24.75s, 8.25s ]  
2535.07s [ 166.01s, 33.20s, 11.07s ]  
2565.07s [ 208.76s, 41.75s, 13.92s ]  
2595.07s [ 252.01s, 50.40s, 16.80s ]
```


※ 一番左の数値は、MySQLが起動してからの時間、
[]に囲まれた数字は、それぞれ1分、5分、15分間の平均遅延時間を表します

■ 良い点

- レプリケーションの遅延状況を確認する事ができます

■ その他

- --stopオプションを使用した場合は以下のように--updateと--monitorが機能しなくなります



```
root@jeep:~  
[root@jeep ~]# pt-heartbeat --master-server-id 1 --stop  
Successfully created file /tmp/pt-heartbeat-sentinel  
[root@jeep ~]# pt-heartbeat --master-server-id 1 --monitor  
[root@jeep ~]# pt-heartbeat --master-server-id 1 --update  
[root@jeep ~]# pt-heartbeat --master-server-id 1 --check  
5.92  
[root@jeep ~]#
```

※ /tmp/pt-heartbeat-sentinelファイルを削除すれば、通常の状態に戻ります

■ コマンド

pt-index-usage [オプション] [ファイル]

【必須項目】

- [ファイル] : slowクエリログを絶対パスで指定します
- [オプション] : -p パスワード

【主なオプション】

- --drop : ユニークインデックスなども含め、使用されていない全てのインデックスを調査します
(デフォルトでは調査対象となるのはセカンダリインデックスのみです)

■ 目的

- slowクエリログからクエリを読み込み、使用されていないインデックスを調査して、削除する為のクエリを出力します

■ 設定ファイル

- 上記の必須項目を、設定ファイルにまとめておきます

```
# touch /etc/percona-toolkit/pt-index-usage.conf  
# vi /etc/percona-toolkit/pt-index-usage.conf
```

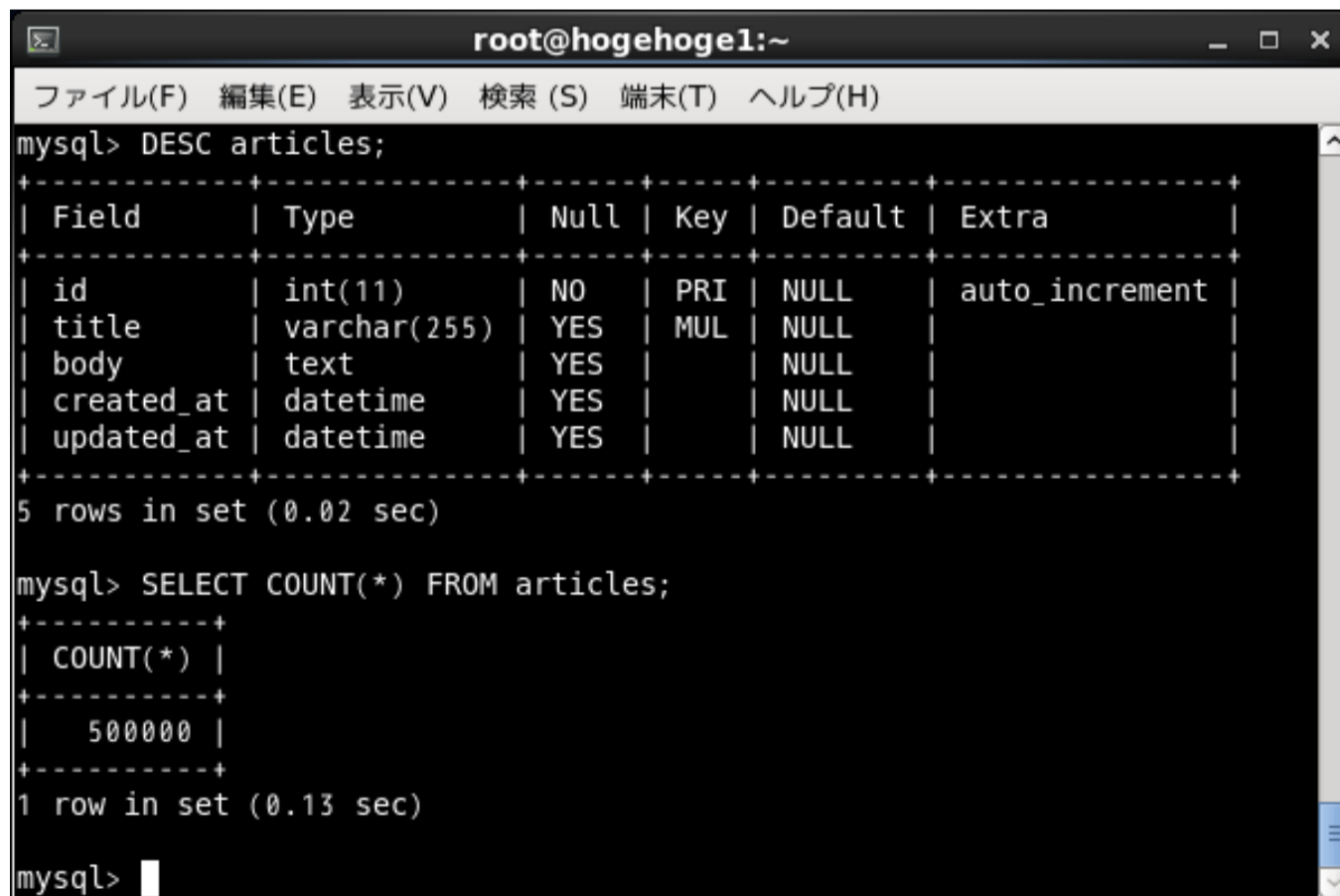
設定ファイルの作成
設定ファイルの編集

```
# config for pt-index-usage  
user=root  
password=パスワード  
D=test
```

コメント
MySQL のユーザ名
MySQL のパスワードを記載
データベース名

■ シナリオ

- 以下のようなテーブルを用意します



```
root@hoge1:~  
mysql> DESC articles;  
+-----+-----+-----+-----+-----+-----+  
| Field      | Type          | Null | Key | Default | Extra          |  
+-----+-----+-----+-----+-----+-----+  
| id         | int(11)       | NO   | PRI | NULL    | auto_increment |  
| title      | varchar(255)  | YES  | MUL | NULL    |                |  
| body       | text          | YES  |     | NULL    |                |  
| created_at | datetime      | YES  |     | NULL    |                |  
| updated_at | datetime      | YES  |     | NULL    |                |  
+-----+-----+-----+-----+-----+-----+  
5 rows in set (0.02 sec)  
  
mysql> SELECT COUNT(*) FROM articles;  
+-----+  
| COUNT(*) |  
+-----+  
| 500000   |  
+-----+  
1 row in set (0.13 sec)  
  
mysql>
```

- 以下のSQL文を実行し、titleカラムに対してインデックスを作成します

```
mysql> CREATE INDEX title_idx ON articles (title);
```

- 以下のSELECT文を実行し、slowクエリログに記録させます

```
mysql> SELECT * FROM articles;
```

- slowクエリログに対して、pt-index-usageコマンドを実行します

```
# pt-index-usage /var/lib/mysql/localhost-slow.log
```

■ 結果

- 使用されていないインデックスを削除するためのALTER文が出力されます



```
root@hoge1:~  
[root@hoge1 ~]# pt-index-usage /var/lib/mysql/slow_queries.log  
  
ALTER TABLE `test`.`articles` DROP KEY `idx_id`, DROP KEY `idx_title`, DROP KEY `title_idx`; -- type:non-unique  
[root@hoge1 ~]#
```

■ 良い点

- 不要なインデックスをすぐに判別することが出来ます

■ コマンド

pt-ioprofile [オプション]

【主なオプション】

- --run-time 数字 : 計測する時間を指定します(デフォルトでは30秒間)
- --cell size : 表示の単位をバイト単位のオペレーション量にします(デフォルトではI/Oの回数)

■ 目的

- mysqldプロセスによる、ファイルへのI/O負荷状況を監視します

■ シナリオ

- 1分間に何バイト分のI/Oが行われたかを見るために、以下のコマンドを実行します

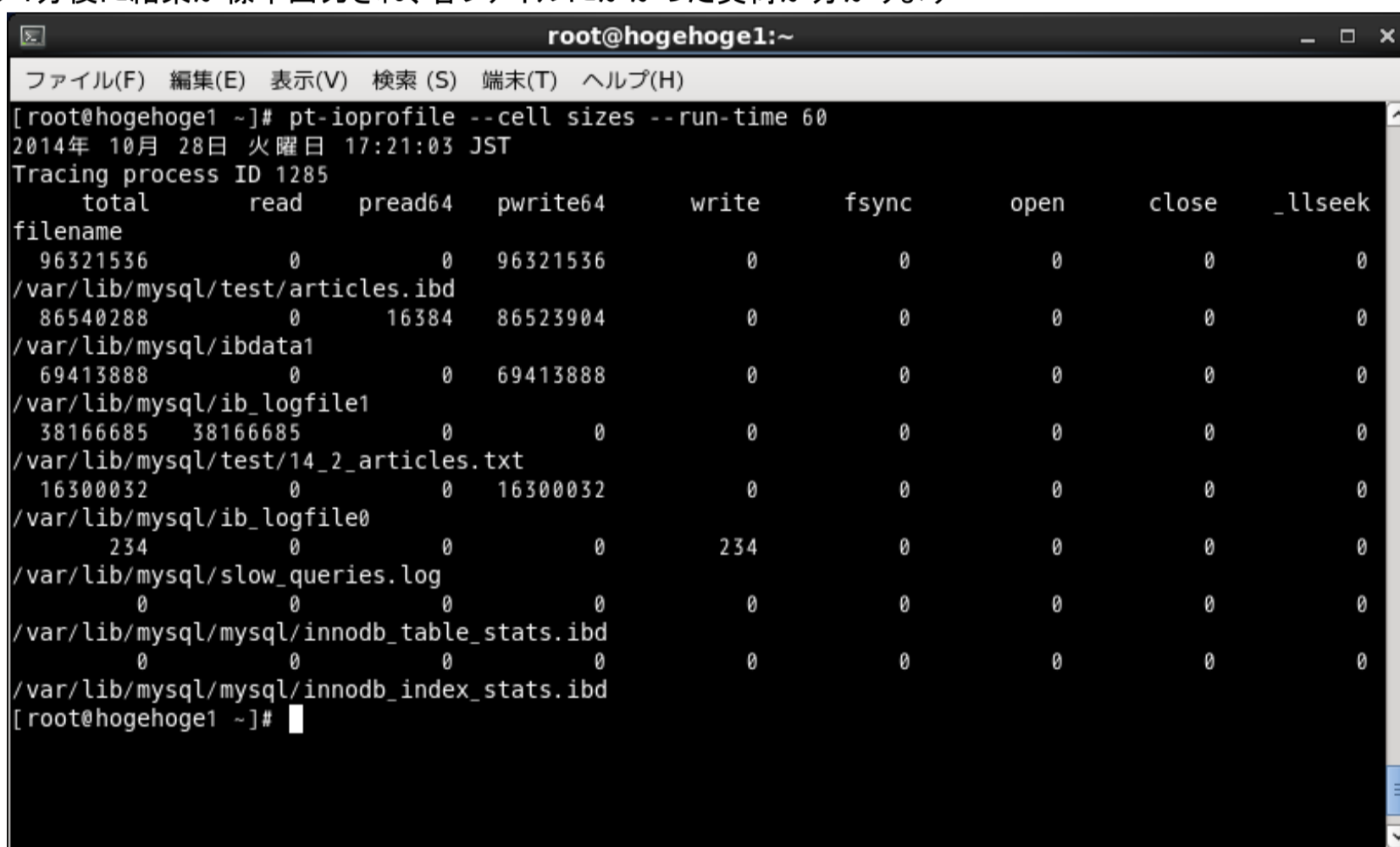
```
# pt-ioprofile --cell sizes --run-time 60
```

- MySQLで以下のコマンドを実行し、50万件のデータを取り込みます

```
mysql> LOAD DATA INFILE "14_2_articles.txt" INTO TABLE articles;
```

■ 結果

- 1分後に結果が標準出力され、各ファイルにかかった負荷が分かります



```
root@hoge1:~# pt-ioprofile --cell sizes --run-time 60
2014年 10月 28日 火曜日 17:21:03 JST
Tracing process ID 1285
total      read      pread64   pwrite64   write      fsync      open      close      _llseek
filename
96321536   0         0         96321536   0         0         0         0         0
/var/lib/mysql/test/articles.ibd
86540288   0         16384     86523904   0         0         0         0         0
/var/lib/mysql/ibdata1
69413888   0         0         69413888   0         0         0         0         0
/var/lib/mysql/ib_logfile1
38166685   38166685  0         0         0         0         0         0         0
/var/lib/mysql/test/14_2_articles.txt
16300032   0         0         16300032   0         0         0         0         0
/var/lib/mysql/ib_logfile0
234        0         0         0         234       0         0         0         0
/var/lib/mysql/slow_queries.log
0          0         0         0         0         0         0         0         0
/var/lib/mysql/mysql/innodb_table_stats.ibd
0          0         0         0         0         0         0         0         0
/var/lib/mysql/mysql/innodb_index_stats.ibd
[root@hoge1 ~]#
```

■ 良い点

- ディスクI/Oの負荷が大きい時に、ファイル単位で負荷の大きさを確認する事が出来ます

■ コマンド名

pt-kill [オプション] [DSN]

【必須項目】

- [オプション] : -p パスワード

【主なオプション】

- --kill : 条件を満たしたクエリを削除します
- --print : 条件を満たしたクエリを表示します
- --busy-time 整数 : 整数の秒数以上かかると、トリガが引かれKILLが実行されます

■ 目的

- 指定した条件を満たすクエリを自動的に取り消すことができます
KILLしたクエリは記録として残すことができます

■ 設定ファイル

- 上記の必須項目を、設定ファイルにまとめておきます

```
# touch /etc/percona-toolkit/pt-kill.conf  
# vi /etc/percona-toolkit/pt-kill.conf
```

設定ファイルの作成
設定ファイルの編集

```
# config for pt-kill  
password=パスワード
```

コメント
MySQL のパスワード

■ シナリオ

- コンソールを開き、以下のコマンドを実行します(実行に2秒以上かかったクエリをKILLし、出力する)

```
# pt-kill --busy-time 1 --kill --print
```

- MySQLにおいて、50万のレコードがあるテーブルに対し、以下の2つのクエリを実行します

```
mysql> SELECT * FROM articles LIMIT 5;  
mysql> SELECT * FROM articles;
```

上のクエリはすぐに実行されますが、下のクエリは1秒以上かかるためKILLされることが期待されます

■ 結果

```
mysql> SELECT * FROM articles LIMIT 5;
```

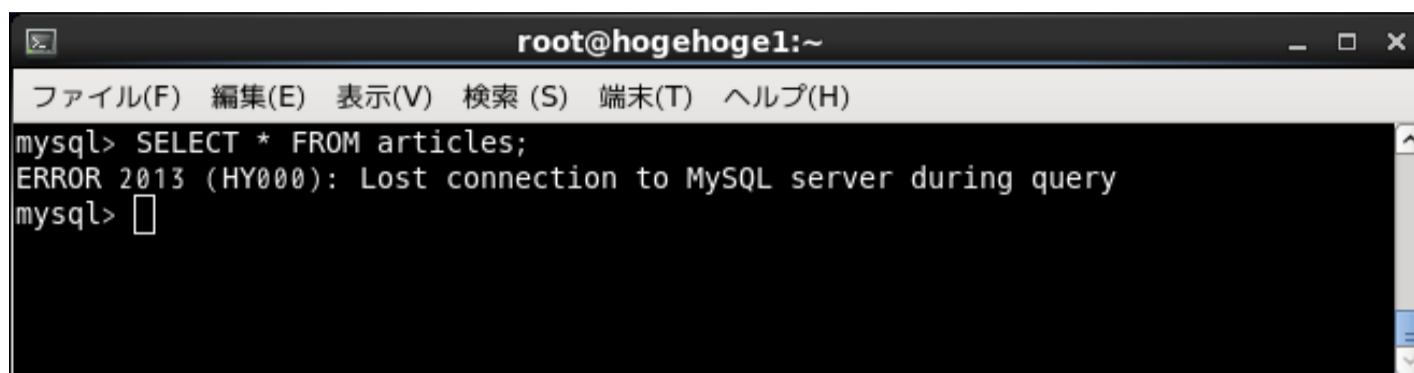
の実行結果

```
root@hogehogel:~  
mysql> SELECT * FROM articles LIMIT 5;  
+----+-----+-----+-----+-----+  
| id | title | body | created_at | updated_at |  
+----+-----+-----+-----+-----+  
| 1 | title_1 | blog_text_1 | 2006-08-01 00:00:00 | 2006-08-01 00:00:00 |  
| 2 | title_2 | blog_text_2 | 2006-08-01 00:00:00 | 2006-08-01 00:00:00 |  
| 3 | title_3 | blog_text_3 | 2006-08-01 00:00:00 | 2006-08-01 00:00:00 |  
| 4 | title_4 | blog_text_4 | 2006-08-01 00:00:00 | 2006-08-01 00:00:00 |  
| 5 | title_5 | blog_text_5 | 2006-08-01 00:00:00 | 2006-08-01 00:00:00 |  
+----+-----+-----+-----+-----+  
5 rows in set (0.00 sec)  
mysql>
```


■ 結果(続き)

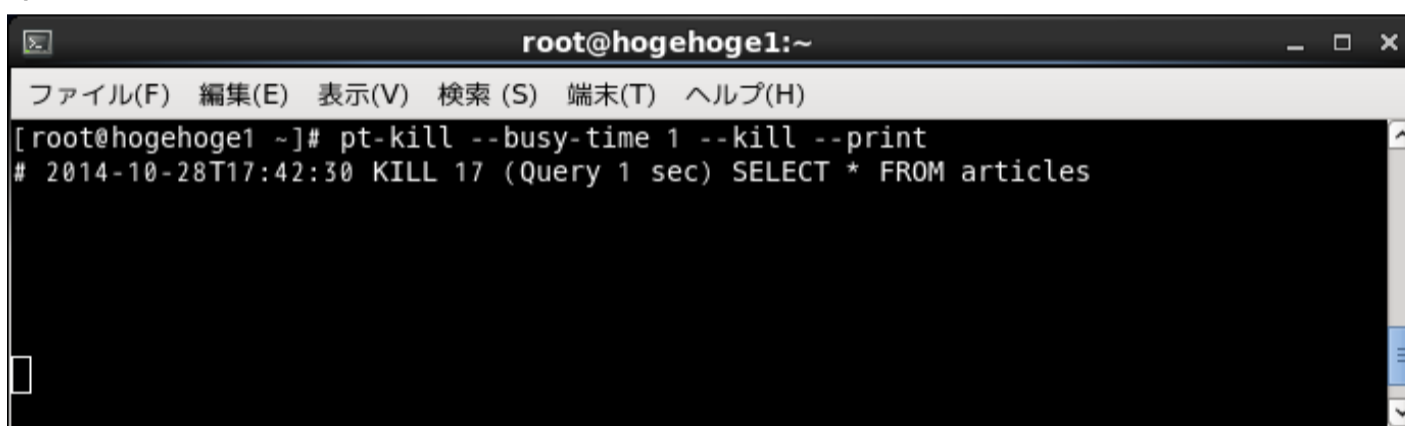
```
mysql> SELECT * FROM articles;
```

の実行結果



```
root@hogehoge1:~  
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)  
mysql> SELECT * FROM articles;  
ERROR 2013 (HY000): Lost connection to MySQL server during query  
mysql> █
```

- 前者は問題なく実行されましたが、後者は1秒以上かかったことでKILLされたことが分かります
pt-kill を実行したコンソールには、以下のような表示がでます



```
root@hogehoge1:~  
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)  
[root@hogehoge1 ~]# pt-kill --busy-time 1 --kill --print  
# 2014-10-28T17:42:30 KILL 17 (Query 1 sec) SELECT * FROM articles  
█
```

■ 良い点

- 指定時間以上の実行時間がかかるクエリを確認したり、該当のクエリを終了させる場合等に有効です

■ コマンド

pt-mext [オプション] -- [コマンド]

【必須項目】

● [コマンド]

- ・ mysqladmin -p exc : mysqladminコマンドを使用します
- ・ -i : 差分検出の間隔を指定します(単位:秒)
- ・ -c : 比較ポイントの個数を指定します(出力される差分は“指定した値-1”となります)

【主なオプション】

- -r : 変化した場合の差分のみを表示します(変化がないステータスについては0と表示されます)

■ 目的

- 指定した間隔で、MySQLの「GLOBAL STATUS」の変化を表示します

■ シナリオ

- 以下のコマンドを実行し、“MySQLへの接続が失敗した回数”を“10秒間隔”で“5回分”表示します

```
# pt-mext -- "mysqladmin -p ext -i 10 -c 6" | grep "Aborted_connects"
```

※ mysqladminコマンドはrootユーザーでMySQLに入るため、rootのパスワードを入力する必要があります

- 別の端末を開いて以下のコマンドを実行し、ログインに失敗します(それを何度か繰り返します)

```
# mysql -u root -p"misssword"
```

■ 結果

- 以下のような結果が標準出力されます

一番左が初期状態で、そこから3回、4回、5回、6回とログイン失敗回数が増加しているのが分かります



```
root@hoge1:~# pt-mext -- "mysqladmin -p ext -i 10 -c 6" | grep "Aborted_connects"
Enter password:
Aborted_connects      0      3      4      5      6
[root@hoge1 ~]#
```

■ 良い点

- 全てのGLOBAL STATUSの変化を監視することが出来るため、不審な動作やエラーをすぐに検知できます
- 任意の回数、間隔で差異を出力できるため、定期的な保守などにも応用できます

■ コマンド名

pt-mysql-summary [オプション]

【必須項目】

- [オプション] : -p パスワード

■ 目的

- MySQLのステータス変数を用途ごとにサマリして出力します

■ 設定ファイル

- 上記の必須項目を、設定ファイルにまとめておきます

```
# touch /etc/percona-toolkit/pt-mysql-summary.conf  
# vi /etc/percona-toolkit/pt-mysql-summary.conf
```

設定ファイルの作成
設定ファイルの編集

```
# config for pt-mysql-summary  
password=パスワード
```

コメント
MySQL のパスワードを記載

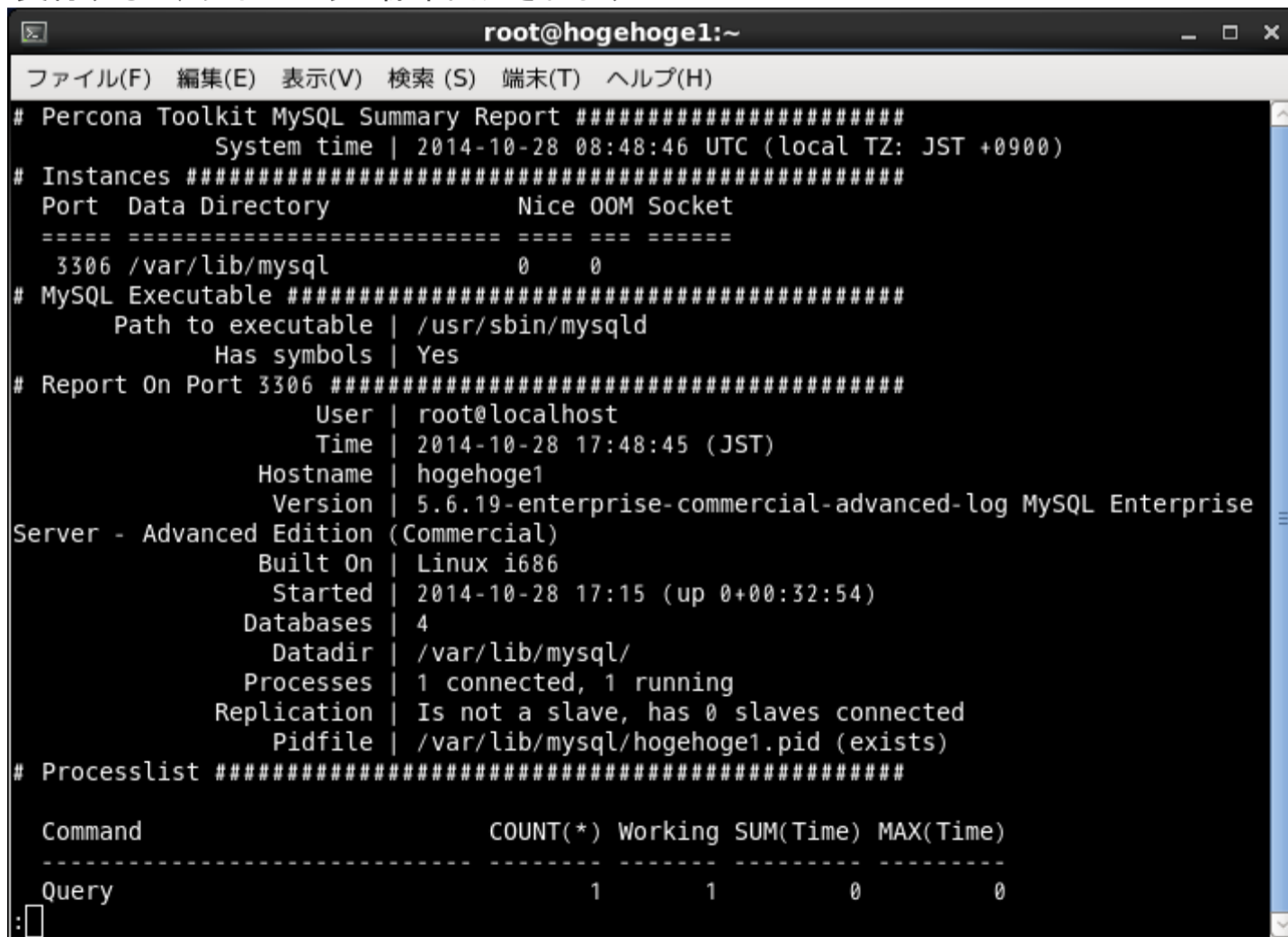
■ シナリオ

- 出力結果が長いので、lessを用いて閲覧(一部省略)

```
# pt-mysql-summary | less
```

■ 結果

- 実行すると、以下のように標準出力されます



■ 結果(続き)

```
root@hoge1:~  
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)  
# Processlist #####  
Command          COUNT(*) Working SUM(Time) MAX(Time)  
-----  
Query            1         1         0         0  
User             COUNT(*) Working SUM(Time) MAX(Time)  
-----  
root             1         1         0         0  
Host             COUNT(*) Working SUM(Time) MAX(Time)  
-----  
localhost        1         1         0         0  
db               COUNT(*) Working SUM(Time) MAX(Time)  
-----  
NULL             1         1         0         0  
State            COUNT(*) Working SUM(Time) MAX(Time)  
-----  
init             1         1         0         0  
# Status Counters (Wait 10 Seconds) #####  
Variable          Per day  Per second  12 secs  
Aborted_clients   450  
Aborted_connects 500  
Bytes_received    1750000  20          300  
:
```

■ 良い点

- テーブルキャッシュの使用率やクエリキャッシュの使用率の確認したり、10秒間にカウントされた接続数やクエリ数のステータス変数を元に、1日あたりの計算値を確認する事が可能です

■ コマンド

pt-online-schema-change [オプション] [--execute または --dry-run] [DSN]

【必須項目】

- [オプション] : -p パスワード
- [DSN] : h=ホスト名, D=データベース名, t=テーブル名
- --execute or --dry-run : --executeを指定すると変更が実行され、--dry-runは変更を実行しません
- --alter "alter文" : 実行したいALTER文を入力します(ただし"ALTER TABLE"という記述は省きます)

【主なオプション】

- --max-load : 大きなデータベースのスキーマ変更の時に、スレッドの閾値を変更します(デフォルトは25です)

■ 目的

- テーブルをロックをせずに、指定のテーブルのスキーマ(構造)を変更する事ができます

■ 設定ファイル

- 上記の必須項目を、設定ファイルにまとめておきます

ただし、[DSN], --execute(--dry-run), --alter はスクリプト上で直接指定する必要があるため、ここでは書きません

```
# touch /etc/percona-toolkit/pt-online-schema-change.conf
# vi /etc/percona-toolkit/pt-online-schema-change.conf
```

設定ファイルの作成
設定ファイルの編集

```
# config for pt-online-schema-change
user=root
password=パスワード
```

コメント
MySQL のユーザ名
MySQL のパスワードを記載

■ シナリオ

- 以下のようなテーブルを用意します

```
root@hoge1:~
mysql> DESC articles;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id         | int(11)       | NO   | PRI | NULL    | auto_increment |
| title      | varchar(255) | YES  | MUL | NULL    |                |
| body       | text          | YES  |     | NULL    |                |
| created_at | datetime      | YES  |     | NULL    |                |
| updated_at | datetime      | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.12 sec)

mysql>
```

- テーブルに "test" というカラムを追加するために、以下のpt-online-schema-changeコマンドを実行します

```
# pt-online-schema-change --execute --alter "add column TEST int" h=localhost,D=test,t=articles
```

- pt-online-schema-changeコマンド実行中に、MySQLで以下のINSERT文・UPDATE文を実行します
通常であれば「カラムの追加」のようなテーブルのスキーマ情報を書き換える動作の場合、テーブルのロックが発生し、その間のINSERT文・UPDATE文はロック待ちとなるため実行に時間がかかります

```
mysql> INSERT INTO articles (title,created_at,updated_at) VALUE ("test", "00000000", "00000000");
mysql> UPDATE articles SET updated_at="11111111111111" WHERE title = "test";
```

■ 結果

- 以下の画面が表示されTESTカラムの追加が行われます
以下のコマンドでINSERT文・UPDATE文も問題なく実行されていることが確認できます

```
mysql> SELECT * FROM articles WHERE title = "test";
```

```
root@hoge1:~# pt-online-schema-change --execute --alter "add column TEST int"
h=localhost,D=test,t=articles
No slaves found. See --recursion-method if host hoge1 has slaves.
Not checking slave lag because no slaves were found and --check-slave-lag was not specified.
Operation, tries, wait:
  copy_rows, 10, 0.25
  create_triggers, 10, 1
  drop_triggers, 10, 1
  swap_tables, 10, 1
  update_foreign_keys, 10, 1
Altering `test`.`articles`...
Creating new table...
Created new table test._articles_new OK.
Altering new table...
Altered `test`.`_articles_new` OK.
2014-10-28T18:10:58 Creating triggers...
2014-10-28T18:10:58 Created triggers OK.
2014-10-28T18:10:58 Copying approximately 497888 rows...
Copying `test`.`articles`: 38% 00:48 remain
Copying `test`.`articles`: 91% 00:05 remain
2014-10-28T18:12:03 Copied rows OK.
2014-10-28T18:12:03 Swapping tables...
2014-10-28T18:12:04 Swapped original and new tables OK.
2014-10-28T18:12:04 Dropping old table...
2014-10-28T18:12:04 Dropped old table `test`.`_articles_old` OK.
2014-10-28T18:12:04 Dropping triggers...
2014-10-28T18:12:04 Dropped triggers OK.
Successfully altered `test`.`articles`.
[root@hoge1 ~]#
```

```
mysql> INSERT INTO articles (title,created_at,updated_at) VALUE ("test", "00000000", "00000000");
Query OK, 1 row affected (0.28 sec)

mysql> UPDATE articles SET updated_at="11111111111111" WHERE title = "test";
Query OK, 1 row affected (0.53 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM articles WHERE title = "test";
+-----+-----+-----+-----+-----+-----+
| id    | title | body | created_at      | updated_at      | TEST |
+-----+-----+-----+-----+-----+-----+
| 501706 | test | NULL | 0000-00-00 00:00:00 | 1111-11-11 11:11:11 | NULL |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.06 sec)

mysql>
```

- コマンドの実行後、articlesテーブルではTESTカラムが追加されたことが確認できます

```
root@hoge1:~# mysql -u root -h localhost -P 3306 -e 'DESC articles;'
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
title	varchar(255)	YES	MUL	NULL	
body	text	YES		NULL	
created_at	datetime	YES		NULL	
updated_at	datetime	YES		NULL	
TEST	int(11)	YES		NULL	

```
mysql>
```

■ 良い点

- 通常のALTER文と異なり、サービスを止める事無く、テーブルへの列追加等が可能となります

■ その他

- --dry-runオプションを指定した場合は以下のようになり、実際にスキーマ変更は実行されません

```
root@hoge1:~# pt-online-schema-change --dry-run --alter "add column TEST int" h=localhost,D=test,t=articles
```

```
Operation, tries, wait:
copy_rows, 10, 0.25
create_triggers, 10, 1
drop_triggers, 10, 1
swap_tables, 10, 1
update_foreign_keys, 10, 1
Starting a dry run. `test`.`articles` will not be altered. Specify --execute instead of --dry-run to alter the table.
Creating new table...
Created new table test._articles_new OK.
Altering new table...
Altered `test`.`_articles_new` OK.
Not creating triggers because this is a dry run.
Not copying rows because this is a dry run.
Not swapping tables because this is a dry run.
Not dropping old table because this is a dry run.
Not dropping triggers because this is a dry run.
2014-10-28T18:16:39 Dropping new table...
2014-10-28T18:16:39 Dropped new table OK.
Dry run complete. `test`.`articles` was not altered.
[root@hoge1 ~]#
```

■ コマンド名

pt-pmp [オプション] [ファイル名]

【主なオプション】

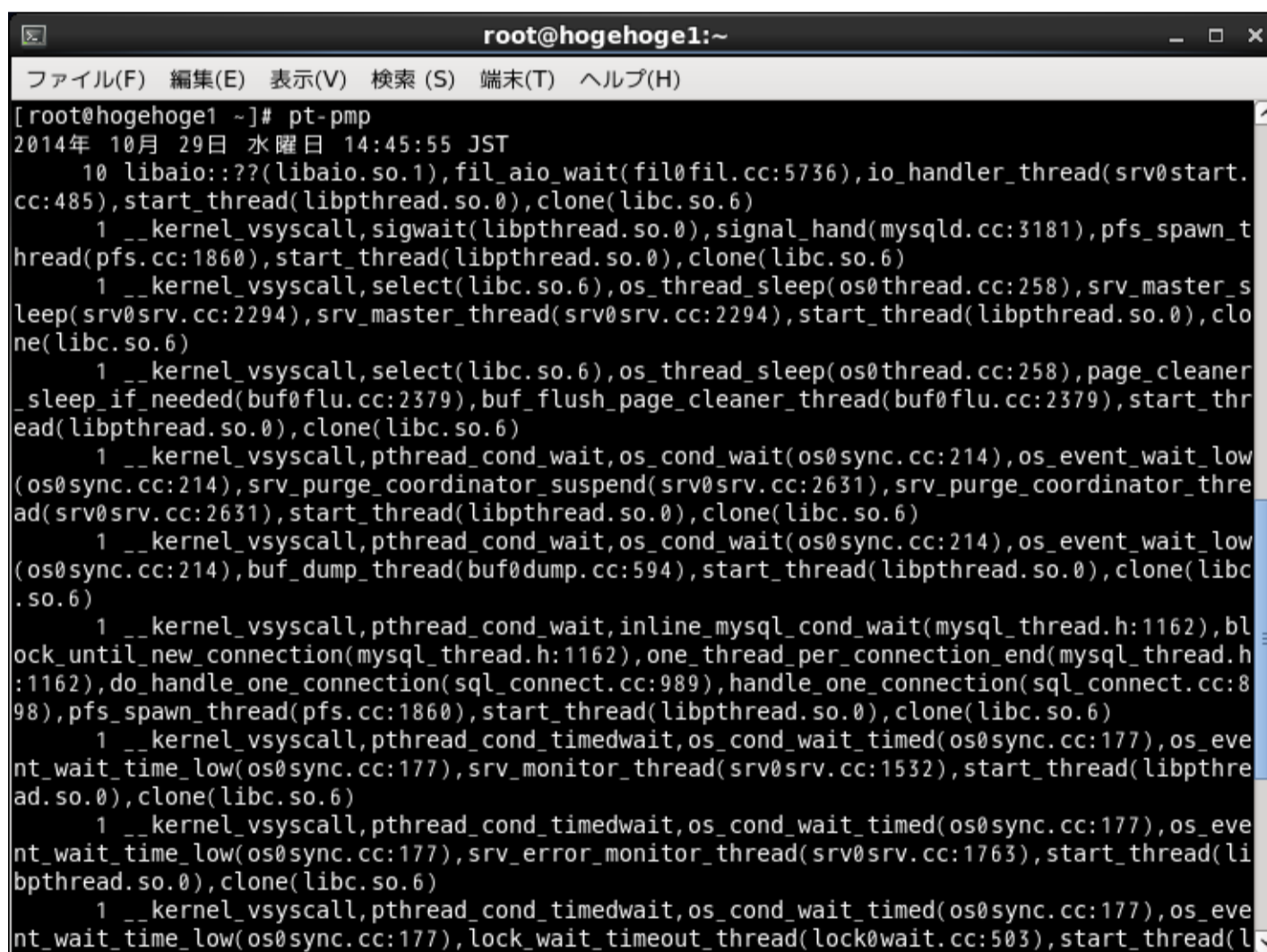
- --binary (-b でも可) : どのバイナリをトレースするか。文字列で入力。デフォルトは mysqld です
- --pid (-p でも可) : トレースするプロセスのID。整数値を入力します

■ 目的

- LINUX上のスタックトレースを表示します

■ 結果

- 実行すると、以下のように標準出力されます



```
root@hogehoge1:~  
[root@hogehoge1 ~]# pt-pmp  
2014年 10月 29日 水曜日 14:45:55 JST  
10 libaio:??(libaio.so.1), fil_aio_wait(fil0fil.cc:5736), io_handler_thread(srv0start.  
cc:485), start_thread(libpthread.so.0), clone(libc.so.6)  
1 __kernel_vsyscall, sigwait(libpthread.so.0), signal_hand(mysql.cc:3181), pfs_spawn_t  
hread(pfs.cc:1860), start_thread(libpthread.so.0), clone(libc.so.6)  
1 __kernel_vsyscall, select(libc.so.6), os_thread_sleep(os0thread.cc:258), srv_master_s  
leep(srv0srv.cc:2294), srv_master_thread(srv0srv.cc:2294), start_thread(libpthread.so.0), clo  
ne(libc.so.6)  
1 __kernel_vsyscall, select(libc.so.6), os_thread_sleep(os0thread.cc:258), page_cleaner  
_sleep_if_needed(buf0flu.cc:2379), buf_flush_page_cleaner_thread(buf0flu.cc:2379), start_thr  
ead(libpthread.so.0), clone(libc.so.6)  
1 __kernel_vsyscall, pthread_cond_wait, os_cond_wait(os0sync.cc:214), os_event_wait_low  
(os0sync.cc:214), srv_purge_coordinator_suspend(srv0srv.cc:2631), srv_purge_coordinator_thre  
ad(srv0srv.cc:2631), start_thread(libpthread.so.0), clone(libc.so.6)  
1 __kernel_vsyscall, pthread_cond_wait, os_cond_wait(os0sync.cc:214), os_event_wait_low  
(os0sync.cc:214), buf_dump_thread(buf0dump.cc:594), start_thread(libpthread.so.0), clone(libc  
.so.6)  
1 __kernel_vsyscall, pthread_cond_wait, inline_mysql_cond_wait(mysql_thread.h:1162), bl  
ock_until_new_connection(mysql_thread.h:1162), one_thread_per_connection_end(mysql_thread.h  
:1162), do_handle_one_connection(sql_connect.cc:989), handle_one_connection(sql_connect.cc:8  
98), pfs_spawn_thread(pfs.cc:1860), start_thread(libpthread.so.0), clone(libc.so.6)  
1 __kernel_vsyscall, pthread_cond_timedwait, os_cond_wait_timed(os0sync.cc:177), os_eve  
nt_wait_time_low(os0sync.cc:177), srv_monitor_thread(srv0srv.cc:1532), start_thread(libpthre  
ad.so.0), clone(libc.so.6)  
1 __kernel_vsyscall, pthread_cond_timedwait, os_cond_wait_timed(os0sync.cc:177), os_eve  
nt_wait_time_low(os0sync.cc:177), srv_error_monitor_thread(srv0srv.cc:1763), start_thread(li  
bpthread.so.0), clone(libc.so.6)  
1 __kernel_vsyscall, pthread_cond_timedwait, os_cond_wait_timed(os0sync.cc:177), os_eve  
nt_wait_time_low(os0sync.cc:177), lock_wait_timeout_thread(lock0wait.cc:503), start_thread(l
```

■ 良い点

- OSレベルでの障害の原因究明や、プロセスが何故待ち状態となっているのか等を調べるのに有効です

■ コマンド名

pt-query-digest [オプション] [ファイル名] [DSN]

【主なオプション】

- --type : どのログを読ませるか。デフォルトは slowlog です。他には binlog や tcpdump 等があります
- --group-by : SQLのものと同じです
- --limit : SQLのものと同じです

■ 目的

- Slowクエリログの集計結果を表示します

■ シナリオ

- Slowクエリログは、デフォルトでは10秒以上のクエリが記録されるようになっていますが、my.cnf を0秒以上で記録するように変更し、いくつかクエリを実行します

```
long_query_time = 0 my.cnf に追記
```

- 記録されたSlowクエリログを pt-query-digest に読み込ませます

■ 結果

- 実行すると、以下のように標準出力されます

```

root@hoge1:~# pt-query-digest /var/lib/mysql/slow_queries.log

# 410ms user time, 100ms system time, 15.93M rss, 24.89M vsz
# Current date: Wed Oct 29 15:08:30 2014
# Hostname: hoge1
# Files: /var/lib/mysql/slow_queries.log
# Overall: 20 total, 14 unique, 0.03 QPS, 0.06x concurrency -----
# Time range: 2014-10-29 14:58:12 to 15:08:13
# Attribute          total      min      max      avg      95%     stddev  median
# =====
# Exec time          38s       276us    11s      2s       8s      3s      100ms
# Lock time          299ms     0        177ms   15ms     71ms    40ms    167us
# Rows sent          958.02k   0        488.28k 47.90k   462.39k 142.21k 0
# Rows examine      1.45M    0        488.28k 74.27k   462.39k 162.40k 0
# Query size         666       11       57      33.30    54.21   17.26   31.70

# Profile
# Rank Query ID      Response time Calls R/Call V/M  Item
# ----
# 1 0x607E11E221CFDFDD 15.8061 41.5% 2 7.9030 2.99 LOAD DATA articles
# 2 0x5F31DF6129A7674A 12.7852 33.6% 2 6.3926 1.24 LOAD DATA comments
# 3 0x6C13F59E762562EE 3.9353 10.3% 2 1.9676 1.25 DELETE comments
# 4 0x0CF8287A02430118 2.8016 7.4% 2 1.4008 2.80 SELECT articles
# 5 0xB9A01D18D0626550 2.2452 5.9% 1 2.2452 0.00 SELECT articles
# MISC 0xMISC         0.5045 1.3% 11 0.0459 0.0 <9 ITEMS>
    
```

- ① Slowクエリの統計情報を見ることができます
- ② Slowクエリのプロファイル情報を見ることができます
ここでは、上位の3つが大きくレスポンスに時間がかかっていることが分かります

■ 結果(続き)

- 以降は、各クエリごとの統計情報を見ることができます

```
root@hogehoge1:~
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
# Query 4: 0.06 QPS, 0.09x concurrency, ID 0x0CF8287A02430118 at byte 2232
# Scores: V/M = 2.80
# Time range: 2014-10-29 14:58:31 to 14:59:02
# Attribute      pct    total      min      max      avg      95%    stddev  median
# =====
# Count          10     2
# Exec time      7      3s    596us    3s      1s      3s     2s      1s
# Lock time      0     490us   165us   325us   245us   325us  113us   245us
# Rows sent      50 488.28k  0 488.28k 244.14k 488.28k 345.27k 244.14k
# Rows examine   32 488.28k  0 488.28k 244.14k 488.28k 345.27k 244.14k
# Query size     6     44     22     22     22     22     0      22
# String:
# Databases      test
# Hosts          localhost
# Users          root
# Query_time distribution
# 1us
# 10us
# 100us #####
# 1ms
# 10ms
# 100ms
# 1s #####
# 10s+
# Tables
# SHOW TABLE STATUS FROM `test` LIKE `articles`\G
# SHOW CREATE TABLE `test`.`articles`\G
# EXPLAIN /*!50100 PARTITIONS*/
SELECT * FROM articles\G
```

■ 良い点

- ユーザや接続元ごとの、時間帯指定による集計を行う事ができるので、Slowクエリログを分析するのに最適です

■ コマンド

pt-show-grants [オプション] [DSN]

【必須項目】

- [オプション] : -p パスワード

■ 目的

- 全ユーザのGRANT情報を出力します

■ 設定ファイル

- 必須オプションを、設定ファイルにまとめておきます

```
# touch /etc/percona-toolkit/pt-show-grants.conf  
# vi /etc/percona-toolkit/pt-show-grants.conf
```

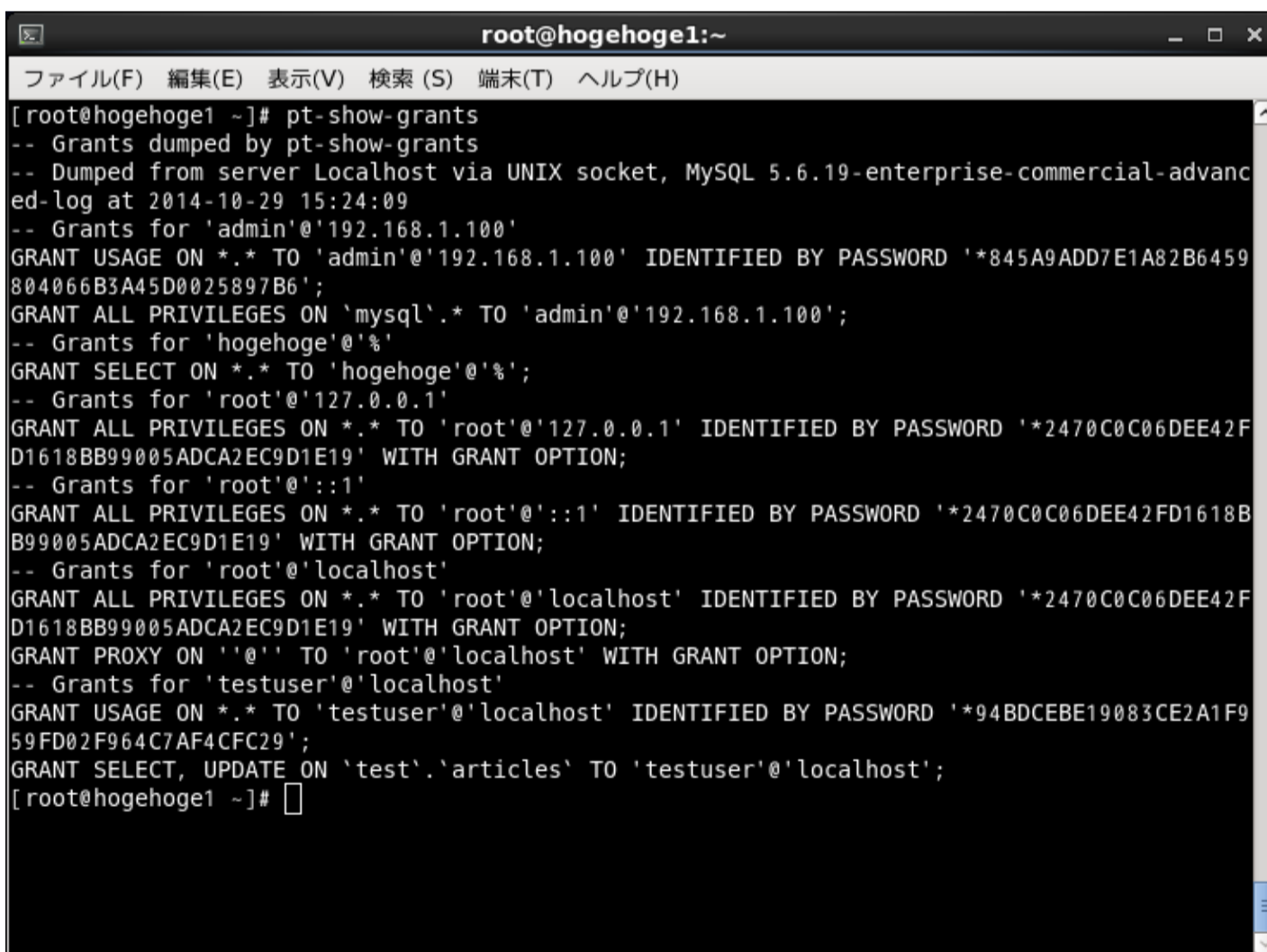
設定ファイルの作成
設定ファイルの編集

```
# config for pt-show-grants  
password=パスワード
```

コメント
MySQL のパスワードを記入

■ 結果

- 実行すると、以下のように標準出力されます



```
root@hoge1:~  
[root@hoge1 ~]# pt-show-grants  
-- Grants dumped by pt-show-grants  
-- Dumped from server Localhost via UNIX socket, MySQL 5.6.19-enterprise-commercial-advanced-log at 2014-10-29 15:24:09  
-- Grants for 'admin'@'192.168.1.100'  
GRANT USAGE ON *.* TO 'admin'@'192.168.1.100' IDENTIFIED BY PASSWORD '*845A9ADD7E1A82B6459804066B3A45D0025897B6';  
GRANT ALL PRIVILEGES ON `mysql`.* TO 'admin'@'192.168.1.100';  
-- Grants for 'hoge1'@'%'  
GRANT SELECT ON *.* TO 'hoge1'@'%';  
-- Grants for 'root'@'127.0.0.1'  
GRANT ALL PRIVILEGES ON *.* TO 'root'@'127.0.0.1' IDENTIFIED BY PASSWORD '*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19' WITH GRANT OPTION;  
-- Grants for 'root'@'::1'  
GRANT ALL PRIVILEGES ON *.* TO 'root'@'::1' IDENTIFIED BY PASSWORD '*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19' WITH GRANT OPTION;  
-- Grants for 'root'@'localhost'  
GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' IDENTIFIED BY PASSWORD '*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19' WITH GRANT OPTION;  
GRANT PROXY ON ''@'' TO 'root'@'localhost' WITH GRANT OPTION;  
-- Grants for 'testuser'@'localhost'  
GRANT USAGE ON *.* TO 'testuser'@'localhost' IDENTIFIED BY PASSWORD '*94BDCEBE19083CE2A1F959FD02F964C7AF4CFC29';  
GRANT SELECT, UPDATE ON `test`.`articles` TO 'testuser'@'localhost';  
[root@hoge1 ~]#
```

■ 良い点

- ユーザの権限情報等を参照するのに有効です

■ コマンド

pt-sift [pt-stalkで作成されたファイルのパス]

■ 目的

- 「pt-stalk」によって作成されたファイルを閲覧します

■ 結果

- 以下のコマンドを実行します

```
# pt-sift /var/lib/pt-stalk/
```

- 実行すると、pt-stalkで収集されたログファイルの一覧が出力されます
次に進むためにEnterキーを押下します

```
[root@localhost ~]# pt-sift /var/lib/pt-stalk/
2015_07_22_16_50_01 2015_07_22_16_57_15 2015_07_22_17_46_56
2015_07_22_17_47_36 2015_07_22_17_48_30 2015_07_22_17_50_33
2015_07_22_17_51_21 2015_07_23_10_12_18 2015_07_23_10_21_45
2015_07_23_11_15_06 2015_07_23_11_24_19 2015_07_23_11_29_23

Select a timestamp from the list [2015_07_23_11_29_23]
```

※ 必ず最新のタイムスタンプが記録されたログが選択されます

- vmstat等の情報を集計した結果が出力されます

```
==== localhost.localdomain at 2015_07_23_11_29_23 DEFAULT (12 of 12) ====
--diskstats--
#ts device rd_s rd_avkb rd_mb_s rd_mrg rd_cnc rd_rt wr_s wr_avkb wr_mb_s wr_mrg w
r_cnc wr_rt busy in_prg io_s qtime stime
[29] dm-0 0.0 0.0 0.0 0% 0.0 0.0 3.4 4.0 0.0 0%
0.0 1.9 0% 0 3.4 0.5 1.3
dm-0 0% . . . . .
--vmstat--
r b swpd free buff cache si so bi bo in cs us sy id wa st
14 0 15344 69152 41304 79572 0 1 21 4 26 38 0 0 99 0 0
0 0 15344 68924 41352 81360 0 0 0 13 178 206 1 4 96 0 0
wa 0% . . . . .
--innodb--
txns: 1xnot (0s)
0 queries inside InnoDB, 0 queries in queue
Main thread: sleeping, pending reads 0, writes 0, flush 0
Log: lsn = 98493547, chkp = 98493547, chkp age = 0
Threads are waiting at:
Threads are waiting on:
```

【主な操作コマンド】

- j : 次のタイムスタンプを表示
- k : 前のタイムスタンプを表示
- q : プログラムの終了

■ 良い点

- 特別なツールを使用しなくても、netstat や vmstat、iostat 等の様々な情報を取得できます

■ コマンド

```
pt-slave-delay [ オプション ] [ スレーブのホスト名 ] [ マスタのホスト名 ]
```

【必須項目】

- [オプション] : -p パスワード
- スレーブのホスト名 : スレーブサーバのホスト名を指定します

【主なオプション】

- --delay : レプリケーションを遅延させる秒数を指定します(デフォルトでは1時間です)
- --interval : スレーブの停止・起動を判断する間隔を指定します(デフォルトで1分です)
- --run-time : コマンドを実行する秒数を指定します(デフォルトでは Ctrl - C するまで動作し続けます)

■ 目的

- 指定した条件でスレーブサーバを停止・起動させ、レプリケーションを意図的に遅延させます

■ 設定ファイル

- 必須オプションを、設定ファイルにまとめておきます

```
# touch /etc/percona-toolkit/pt-slave-delay.conf  
# vi /etc/percona-toolkit/pt-slave-delay.conf
```

設定ファイルの作成
設定ファイルの編集

```
# config for pt-slave-delay  
password=パスワード
```

コメント
スレーブ側のMySQL のパスワード

■ シナリオ

- スレーブ側で以下のコマンドを実行し90秒間、レプリケーションを60秒遅延させます

```
# pt-slave-delay --delay 60 --interval 30 --run-time 90 localhost
```

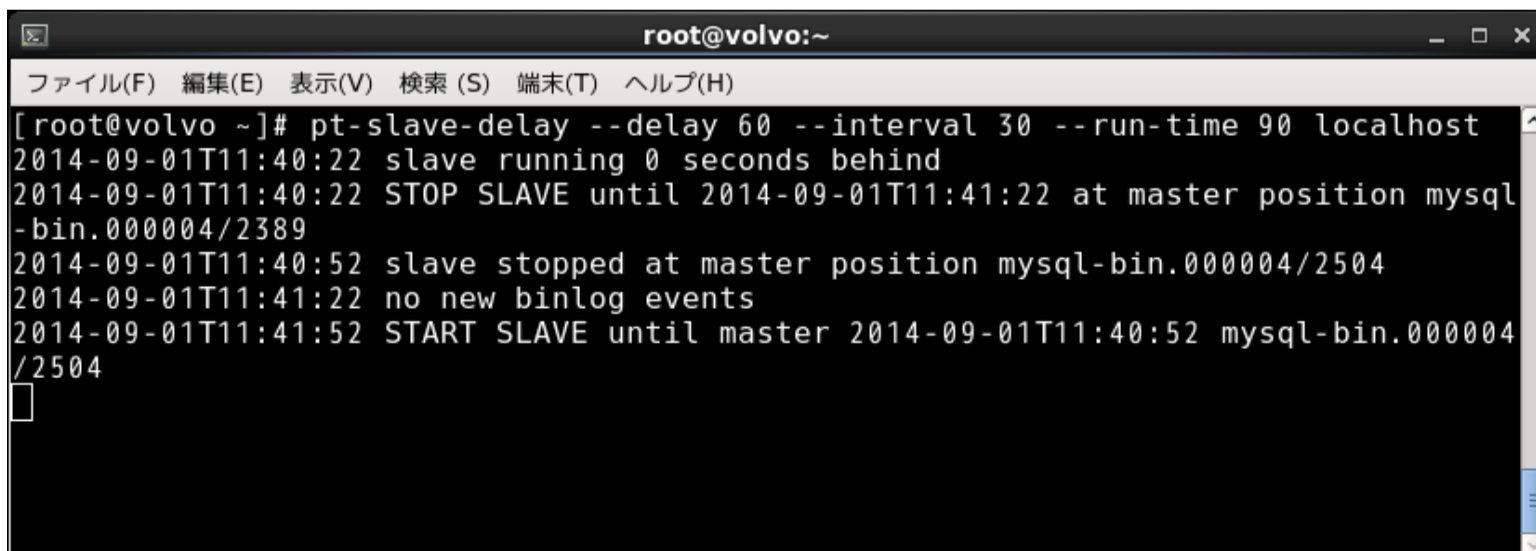
- マスタ側で以下のSQL文を実行します

```
mysql> CREATE TABLE delay_test(id int);
```

- スレーブ側で以下のSQL文を実行します

```
mysql> SHOW TABLES;
```

- コンソールに以下のような文章が表示されたら、もう一度上のSQL文を実行します



```
root@volvo:~  
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)  
[root@volvo ~]# pt-slave-delay --delay 60 --interval 30 --run-time 90 localhost  
2014-09-01T11:40:22 slave running 0 seconds behind  
2014-09-01T11:40:22 STOP SLAVE until 2014-09-01T11:41:22 at master position mysql  
-bin.000004/2389  
2014-09-01T11:40:52 slave stopped at master position mysql-bin.000004/2504  
2014-09-01T11:41:22 no new binlog events  
2014-09-01T11:41:52 START SLAVE until master 2014-09-01T11:40:52 mysql-bin.000004  
/2504  
█
```

■ 結果

- スレーブ側でレプリケーション遅延が発生していることが確認できます



```
root@volvo:~  
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)  
mysql> SHOW TABLES;  
+-----+  
| Tables_in_repl_test |  
+-----+  
| sample  
| test_by_master  
+-----+  
2 rows in set (0.00 sec)  
  
mysql> SHOW TABLES;  
+-----+  
| Tables_in_repl_test |  
+-----+  
| delay_test  
| sample  
| test_by_master  
+-----+  
3 rows in set (0.00 sec)
```

※ コマンド終了時点では、スレーブサーバのスレーブモードは停止した状態になります
そのため、コマンド終了後、以下を実行しスレーブモードを開始する必要があります

```
mysql> SLAVE START;
```

※ 遅延する時間は、「--delayで指定した時間」から「--delay + --interval」の間で推移します

■ 良い点

- マスタサーバのオペレーションミス等によるデータの反映を遅延させる事により、スレーブサーバへの被害を最小限に抑える事ができます

■ その他

- MySQL5.6以降では、以下のSQL文でpt-slave-delayコマンドと同様の機能が実現できます

```
mysql> STOP SLAVE;  
mysql> CHANGE MASTER TO MASTER_DELAY=60;  
mysql> START SLAVE;
```

■ コマンド

pt-slave-find [オプション] [マスタのホスト名]

【必須項目】

- [オプション] : -p パスワード

【主なオプション】

- [DSN] : h=ホスト名, u=ユーザー名 (デフォルトではそれぞれlocalhost, root)
※ ここで指定するDSNはマスタ側のものを使用します

■ 目的

- マスタに対してレプリケーションをしているスレーブをツリー表示します

■ 設定ファイル

- 必須オプションを、設定ファイルにまとめておきます

```
# touch /etc/percona-toolkit/pt-slave-find.conf  
# vi /etc/percona-toolkit/pt-slave-find.conf
```

設定ファイルの作成
設定ファイルの編集

```
# config for pt-slave-find  
h=192.168.2.222  
user=root  
password=password
```

コメント
マスタのIPアドレス
MySQLのユーザー名
MySQLのパスワード

■ シナリオ

- 全てのPCでiptablesがオフになっていることを確認します

```
# service iptables status  
iptables: ファイアウォールが稼働していません。
```

- 全てのサーバのMySQL上で以下のユーザーが存在することを確認します

```
mysql> SELECT user,host FROM mysql.user  
:  
root | 192.168.2.222
```

- マスタ側で以下のコマンドを実行して、マスタに接続されているスレーブの情報を表示します

```
# pt-slave-find
```

■ 結果

- マスタに接続されているスレーブがツリー状に表示されます

```
root@jeep:~  
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)  
[root@jeep ~]# pt-slave-find  
192.168.2.222  
Version          5.6.20-log  
Server ID        101  
Uptime           18:30 (started 2014-09-09T10:15:54)  
Replication      Is not a slave, has 2 slaves connected, is not read_only  
Filters  
Binary logging   STATEMENT  
Slave status  
Slave mode       STRICT  
Auto-increment   increment 1, offset 1  
① InnoDB version 5.6.20  
+- volvo  
Version          5.6.20  
Server ID        102  
Uptime           33:36 (started 2014-09-09T10:00:48)  
Replication      Is a slave, has 0 slaves connected, is not read_only  
Filters  
Binary logging   STATEMENT  
Slave status     0 seconds behind, running, no errors  
Slave mode       STRICT  
Auto-increment   increment 1, offset 1  
② InnoDB version 5.6.20  
+- 192.168.2.226  
Version          5.6.20  
Server ID        103  
Uptime           33:00 (started 2014-09-09T10:01:24)  
Replication      Is a slave, has 0 slaves connected, is not read_only  
Filters  
Binary logging   STATEMENT  
Slave status     seconds behind, not running, error 1396  
Slave mode       STRICT  
Auto-increment   increment 1, offset 1  
InnoDB version   5.6.20  
[root@jeep ~]#
```

■ 良い点

- マスタに接続されているスレーブの情報を簡単に知ることが出来ます

■ その他

- ②のようなIPアドレス表示を、①のようなホスト名表示に変えたい場合は以下のようにします

```
# vi /etc/hosts
```

```
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4  
::1      localhost localhost.localdomain localhost6 localhost6.localdomain6  
192.168.2.224 volvo # スレーブ(ホスト名:volvo)  
192.168.2.226 lancia # スレーブ(ホスト名:lancia)
```

→ 追記
→ 追記

■ コマンド

pt-slave-restart [オプション] [DSN]

【必須項目】

- [オプション] : -p パスワード

【主なオプション】

- [DSN] : h=ホスト名, u=ユーザー名(デフォルトではそれぞれlocalhost, root)
※ ここで指定するDSNはスレーブ側のものを使用します

■ 目的

- レプリケーションの Slave を監視して、エラー等により停止した場合に自動的に再起動を試みます

■ シナリオ

- マスター側に以下のようなテーブルを用意します(スレーブ側にも同テーブルがレプリケーションされます)

```
mysql> CREATE TABLE `test001` (`id` int(11) DEFAULT NULL  
ENGINE=InnoDB DEFAULT CHARSET=utf8
```

- スレーブ側のテーブルにPRIMARY KEY 制約をつけます

```
mysql> ALTER TABLE test001 ADD PRIMARY KEY(id);
```

- スレーブ側のコンソール画面から、以下のコマンドを実行します

```
# pt-slave-restart -u root -ppassword
```

- マスター側で以下のSQLを実行し、意図的にスレーブ側でキー重複エラーを起こします

```
mysql> INSERT INTO test001(id) VALUES(1);  
mysql> INSERT INTO test001(id) VALUES(1);
```

■ 結果

- スレーブ側で、レプリケーションのエラーが発生します

```
mysql> SHOW SLAVE STATUS\G  
***** 1. row *****  
Slave_IO_State: Waiting for master to send event  
Master_Host: 192.168.2.223  
Master_User: master  
Master_Port: 3306  
Connect_Retry: 60  
Master_Log_File: mysql-bin.000001  
Read_Master_Log_Pos: 1009  
Relay_Log_File: mysql-relay-bin.000003  
Relay_Log_Pos: 935  
Relay_Master_Log_File: mysql-bin.000001  
Slave_IO_Running: Yes  
Slave_SQL_Running: No  
Replicate_Do_DB:  
Replicate_Ignore_DB:  
Replicate_Do_Table:  
Replicate_Ignore_Table:  
Replicate_Wild_Do_Table:  
Replicate_Wild_Ignore_Table:  
Last_Errno: 1062  
Last_Error: Error 'Duplicate entry '1' for key '  
PRIMARY'' on query. Default database: 'master001'. Query: 'INSERT I  
NTO test001(id) VALUES(1)'
```

※ スレーブ側のテーブルにのみ、
idカラムに対しPRIMARYキーが
設定されているため、重複した
データをINSERT出来ず、エラーが
発生する

■ 結果(続き)

- その後、pt-slave-restart が動作します

```
[root@localhost ~]# pt-slave-restart -u root -ppassword  
2015-07-27T17:22:47 p=...,u=root mysql-relay-bin.000003          935 1062
```

- エラーの原因となる処理がスキップされ、レプリケーションが正常に継続されます

```
mysql> SHOW SLAVE STATUS\G  
***** 1. row *****  
Slave_IO_State: Waiting for master to send event  
Master_Host: 192.168.2.223  
Master_User: master  
Master_Port: 3306  
Connect_Retry: 60  
Master_Log_File: mysql-bin.000001  
Read_Master_Log_Pos: 1009  
Relay_Log_File: mysql-relay-bin.000003  
Relay_Log_Pos: 1172  
Relay_Master_Log_File: mysql-bin.000001  
Slave_IO_Running: Yes  
Slave_SQL_Running: Yes  
Replicate_Do_DB:  
Replicate_Ignore_DB:  
Replicate_Do_Table:  
Replicate_Ignore_Table:  
Replicate_Wild_Do_Table:  
Replicate_Wild_Ignore_Table:  
Last_Errno: 0  
Last_Error:  
Skip_Counter: 0  
Exec_Master_Log_Pos: 1009
```

■ 良い点

- レプリケーション中にエラーが生じても、その処理をスキップしてくれるので、レプリケーションの停止を防ぐことができます

■ コマンド

pt-stalk

【必須項目】

- [オプション] : -p パスワード

【主なオプション】

- --variable: pt-stalkが実行される時の条件となるステータス変数を指定する
- --threshold: --variableで指定したステータス変数の、実行条件となる値を指定する
- --cycles: ログを集める回数を指定する(デフォルトは5回) ここで指定した回数分ログ収集を実行した後、ログの集計が実行される
(集計前に、上の実行条件を満たさない状態になった場合、pt-stalk は中断される)

■ 目的

- 指定した条件に達した時点の netstat や vmstat、iostat 等の様々な情報を出力します

■ シナリオ

- 以下のコマンドを実行し、MySQLへの接続数が1以上になると起動するようにします

```
# pt-stalk -u =root -p=password --variable=Threads_connected --threshold=
```

- 実行すると、以下のように標準出力されます

```
Warning: Using a password on the command line interface can be insecure.
Overwriting PID file /var/run/pt-stalk.pid because its PID (7949) is not running
2015_07_23_11_23_47 Starting /usr/bin/pt-stalk --function=status --variable=Threads_connected --threshold=1 --match= --cycles=5 --interval=1 --iterations= --run-time=30 --sleep=300 --dest=/var/lib/pt-stalk --prefix= --notify-by-email= --log=/var/log/pt-stalk.log --pid=/var/run/pt-stalk.pid --plugin=
Warning: Using a password on the command line interface can be insecure.
Warning: Using a password on the command line interface can be insecure.
Warning: Using a password on the command line interface can be insecure.
```

以下、警告が続く

- 別画面からMySQLに接続します

```
# mysql -u root -ppassword
```

■ 結果

- MySQLの起動により、コンソール画面が以下のように出力されます

```

Warning: Using a password on the command line interface can be insecure.
Warning: Using a password on the command line interface can be insecure.
① 2015_07_23_11_24_15 Check results: status(Threads_connected)=2, matched=yes, cycles_true=1
Warning: Using a password on the command line interface can be insecure.
2015_07_23_11_24_16 Check results: status(Threads_connected)=2, matched=yes, cycles_true=2
Warning: Using a password on the command line interface can be insecure.
2015_07_23_11_24_17 Check results: status(Threads_connected)=2, matched=yes, cycles_true=3
Warning: Using a password on the command line interface can be insecure.
2015_07_23_11_24_18 Check results: status(Threads_connected)=2, matched=yes, cycles_true=4
Warning: Using a password on the command line interface can be insecure.
2015_07_23_11_24_19 Check results: status(Threads_connected)=2, matched=yes, cycles_true=5
② 2015_07_23_11_24_19 Collect 1 triggered
2015_07_23_11_24_19 Collect 1 PID 10435
2015_07_23_11_24_19 Collect 1 done
2015_07_23_11_24_19 Sleeping 300 seconds after collect
    
```

- ① MySQLの起動により情報(ログ)の収集が繰り返されます
- ② 5回繰り返した後、300秒間休止します ※この際ログファイルが作られます

- 300秒後、再び同じ動作をしてまた休止します (pt-stalkを停止させるまでこの動作の繰り返しとなります)

```

2015_07_23_11_24_19 Collect 1 done
2015_07_23_11_24_19 Sleeping 300 seconds after collect
Warning: Using a password on the command line interface can be insecure.
2015_07_23_11_29_19 Check results: status(Threads_connected)=2, matched=yes, cycles_true=1
Warning: Using a password on the command line interface can be insecure.
2015_07_23_11_29_20 Check results: status(Threads_connected)=2, matched=yes, cycles_true=2
Warning: Using a password on the command line interface can be insecure.
2015_07_23_11_29_21 Check results: status(Threads_connected)=2, matched=yes, cycles_true=3
Warning: Using a password on the command line interface can be insecure.
2015_07_23_11_29_22 Check results: status(Threads_connected)=2, matched=yes, cycles_true=4
Warning: Using a password on the command line interface can be insecure.
2015_07_23_11_29_23 Check results: status(Threads_connected)=2, matched=yes, cycles_true=5
2015_07_23_11_29_23 Collect 2 triggered
2015_07_23_11_29_23 Collect 2 PID 11963
2015_07_23_11_29_23 Collect 2 done
2015_07_23_11_29_23 Sleeping 300 seconds after collect
    
```

- /var/lib/pt-stalk 配下にnetstat や vmstat、iostat 等の様々な情報の入ったファイルが作られます

```

[root@localhost ~]# ls /var/lib/pt-stalk/
2015_07_22_16_50_01-df                2015_07_22_17_51_21-disk-space
2015_07_22_16_50_01-disk-space       2015_07_22_17_51_21-diskstats
2015_07_22_16_50_01-diskstats        2015_07_22_17_51_21-hostname
2015_07_22_16_50_01-hostname         2015_07_22_17_51_21-innodbstatus1
2015_07_22_16_50_01-innodbstatus1    2015_07_22_17_51_21-innodbstatus2
2015_07_22_16_50_01-innodbstatus2    2015_07_22_17_51_21-interrupts
2015_07_22_16_50_01-interrupts       2015_07_22_17_51_21-iostat
2015_07_22_16_50_01-iostat           2015_07_22_17_51_21-iostat-overall
2015_07_22_16_50_01-iostat-overall   2015_07_22_17_51_21-lsof
2015_07_22_16_50_01-lsof            2015_07_22_17_51_21-meminfo
2015_07_22_16_50_01-meminfo          2015_07_22_17_51_21-mpstat
2015_07_22_16_50_01-mpstat           2015_07_22_17_51_21-mpstat-overall
2015_07_22_16_50_01-mpstat-overall   2015_07_22_17_51_21-mutex-status1
2015_07_22_16_50_01-mutex-status1    2015_07_22_17_51_21-mutex-status2
2015_07_22_16_50_01-mutex-status2    2015_07_22_17_51_21-netstat
    
```

※ ファイル名の冒頭部分はコマンドが実行されたときのタイムスタンプです

■ 良い点

- 指定した条件を満たした時に、自動でサーバやMySQLの情報が取得できます

■ コマンド

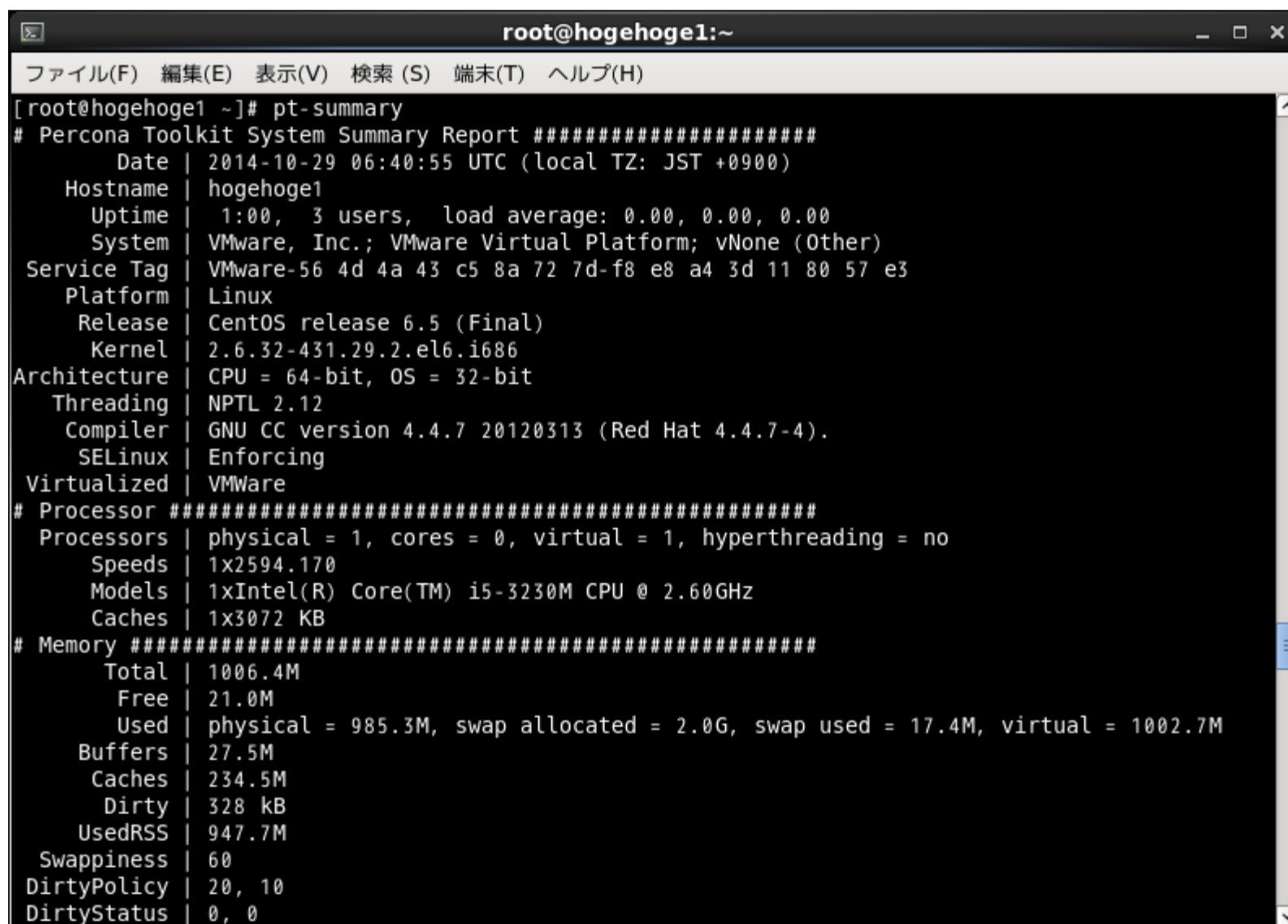
pt-summary

■ 目的

- サーバのOSやメモリ、ボリューム等の物理的な情報と、メモリ使用状況やプロセス情報、ネットワーク接続情報等の状況を出力します

■ 結果

- 実行すると、以下のように標準出力されます
- 出力結果が長いので less で閲覧、2ページ目以降は省略します



```
root@hoge1:~  
[root@hoge1 ~]# pt-summary  
# Percona Toolkit System Summary Report #####  
Date | 2014-10-29 06:40:55 UTC (local TZ: JST +0900)  
Hostname | hoge1  
Uptime | 1:00, 3 users, load average: 0.00, 0.00, 0.00  
System | VMware, Inc.; VMware Virtual Platform; vNone (Other)  
Service Tag | VMware-56 4d 4a 43 c5 8a 72 7d-f8 e8 a4 3d 11 80 57 e3  
Platform | Linux  
Release | CentOS release 6.5 (Final)  
Kernel | 2.6.32-431.29.2.el6.i686  
Architecture | CPU = 64-bit, OS = 32-bit  
Threading | NPTL 2.12  
Compiler | GNU CC version 4.4.7 20120313 (Red Hat 4.4.7-4).  
SELinux | Enforcing  
Virtualized | VMWare  
# Processor #####  
Processors | physical = 1, cores = 0, virtual = 1, hyperthreading = no  
Speeds | 1x2594.170  
Models | 1xIntel(R) Core(TM) i5-3230M CPU @ 2.60GHz  
Caches | 1x3072 KB  
# Memory #####  
Total | 1006.4M  
Free | 21.0M  
Used | physical = 985.3M, swap allocated = 2.0G, swap used = 17.4M, virtual = 1002.7M  
Buffers | 27.5M  
Caches | 234.5M  
Dirty | 328 kB  
UsedRSS | 947.7M  
Swappiness | 60  
DirtyPolicy | 20, 10  
DirtyStatus | 0, 0
```

■ 良い点

- サーバに関する物理的な情報を参照する際に有効です

■ コマンド

pt-table-checksum [オプション] [DSN]

【必須項目】

- [DSN] : h=ホスト名,u=ユーザー名,p=パスワード
※ここで指定するDSNはマスタ側のものを使用します

【主なオプション】

- --no-check-binlog-format : pt-table-checksum を使用する場合はバイナリログの形式を「STATEMENT」にする必要があるため、その他の形式を用いたい場合はこのオプションを指定します

■ 目的

- レプリケーションでの、マスタとスレーブ間のテーブルの整合性をチェックします

■ 設定ファイル

- 必須オプションを、設定ファイルにまとめておきます

```
# touch /etc/percona-toolkit/pt-table-checksum.conf
# vi /etc/percona-toolkit/pt-table-checksum.conf
```

設定ファイルの作成
設定ファイルの編集

```
# config for table-check-sum
h=192.168.2.222
user=root
password=password
databases=test,repl_test
```

コメント
マスタのIPアドレス
MySQLのユーザー名
MySQLのパスワード
MySQLのデータベース

■ 結果

- マスタ側でコマンドを実行すると、以下のようにチェックの結果が標準出力されます

```
root@jeep:~
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@jeep ~]# pt-table-checksum
      TS  ERRORS  DIFFS    ROWS  CHUNKS  SKIPPED    TIME  TABLE
09-08T17:02:45      0      0  500000      7      0    2.155 repl_test.articles
09-08T17:02:45      0      0      0      1      0    0.101 repl_test.comments
09-08T17:02:45      0      0      1      1      0    0.045 repl_test.test
09-08T17:02:47      0      0  500000      5      0    1.993 test.articles
09-08T17:02:49      0      0  500000      5      0    1.993 test.comments
09-08T17:02:49      0      0      1      1      0    0.030 test.heartbeat
09-08T17:02:49      0      0      0      1      0    0.025 test.test_1
[root@jeep ~]#
```

■ 良い点

- 正確にレプリケーションが行われているかを確認することが出来ます

■ コマンド

pt-table-sync [オプション] [同期元DSN] [同期先DSN]

【必須項目】

- --execute or --dry-run : --executeを指定すると変更が実行され、--dry-runは変更を実行しません
- [同期元DSN] : h=ホスト名,u=ユーザー名,p=パスワード (D=データベース名,t=テーブル名)
- [同期先DSN] : h=ホスト名,u=ユーザー名,p=パスワード
※[同期先DSN]にはSUPER権限のあるユーザを指定する必要があります

【主なオプション】

- --sync-to-master : 同期先DSNで指定したホスト(スレーブ)のマスタを同期元DSNとすることができます
※このオプションを使用する場合、[同期元DSN] を書く必要はありません

■ 目的

- ホスト間でテーブルの同期を行うことができます

■ シナリオ

- 今回は、レプリケーション環境で、マスタ(同期元)とスレーブ(同期先)のテーブルを同期させます

- スレーブ側でレプリケーションを停止します

```
mysql> STOP SLAVE;
```

- マスタ側に以下のようなテーブルを用意し、レコードを挿入します

```
mysql> CREATE TABLE test (id INT);  
mysql> INSERT INTO test (id) VALUES(1),(2),(3);  
mysql> SHOW MASTER STATUS;
```

- スレーブ側にもマスタと同じ定義の空テーブルを用意し、改めてマスタとレプリケーションを組みます

```
> RESET SLAVE ALL;  
> CHANGE MASTER TO ...  
> START SLAVE;
```

- マスタ側とスレーブ側でデータが異なっていることを確認します

```
mysql> SELECT * FROM test;
```

マスタ側

```
+-----+  
| id |  
+-----+  
|  1 |  
|  2 |  
|  3 |  
+-----+  
3 rows in set (0.00 sec)
```

スレーブ側

```
Empty set (0.00 sec)
```

- スレーブ側でpt-table-syncを実行します

```
# pt-table-sync --execute --sync-to-master u=root,p=password
```

■ 結果

- スレーブ側のテーブルにマスタのテーブルが同期されます

```
mysql> SELECT * FROM test;
```

マスタ側

```
+-----+
| id |
+-----+
|  1 |
|  2 |
|  3 |
+-----+
3 rows in set (0.00 sec)
```

スレーブ側

```
+-----+
| id |
+-----+
|  1 |
|  2 |
|  3 |
+-----+
3 rows in set (0.00 sec)
```

■ 良い点

- テーブル間の同期を簡単に行うことができます

■ 注意点

- 双方に同じ定義のテーブルが存在している必要があります
- --sync-to-masterを使う場合、ユニークキーの設定されていないテーブルを同期することはできません

■ その他

- ホスト名を指定することで、任意のホスト間でテーブルを同期させることができます

```
# pt-table-sync --execute h=host1,D=database,t=table h=host2
```

※host1(同期元)のテーブルをhost2(同期先)に同期します

■ コマンド

pt-table-usage [オプション] [ファイル名]

■ 目的

- Slowクエリログからクエリを読み込み、どのテーブルへアクセスしたものを分析・表示します

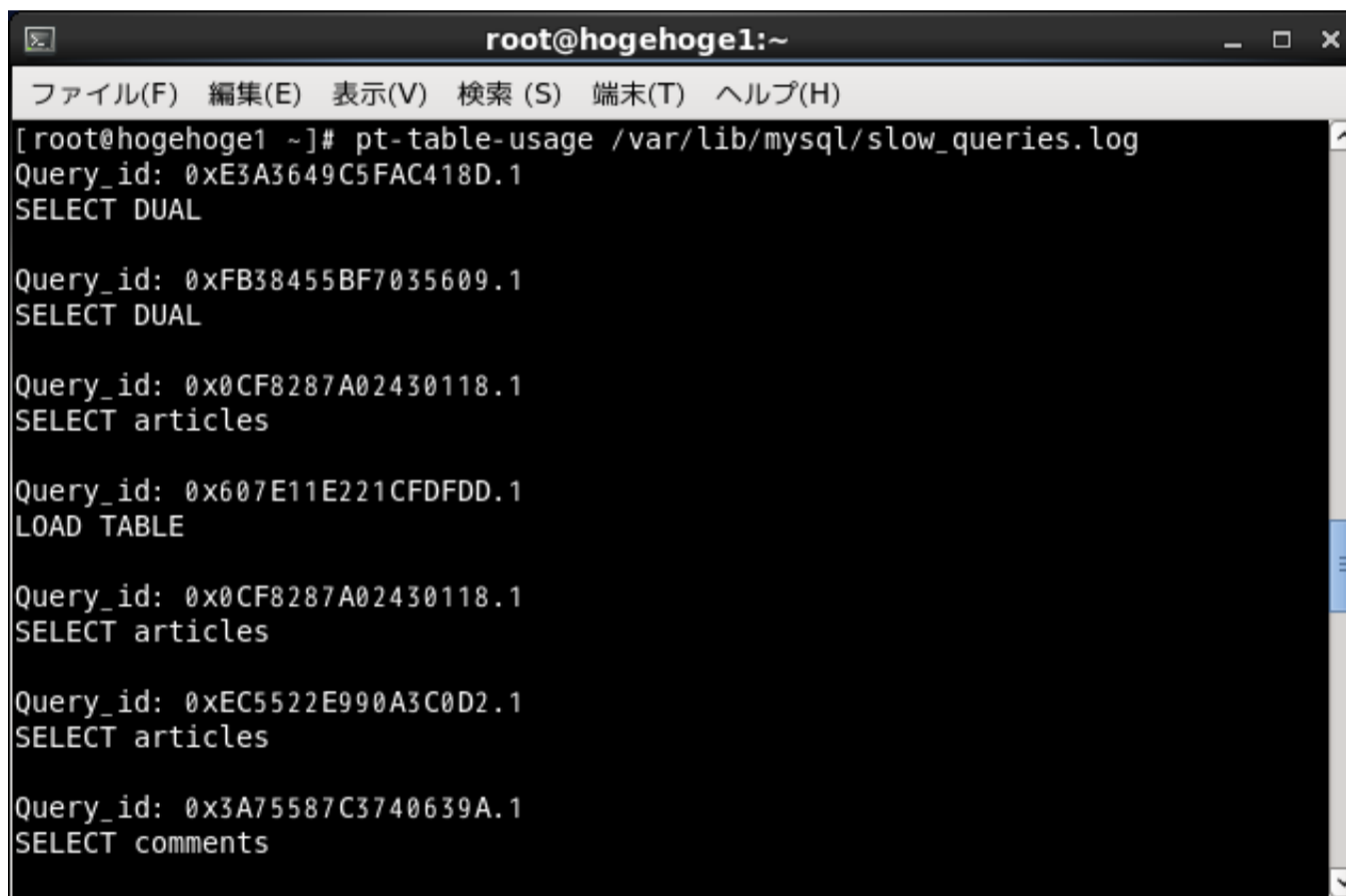
■ シナリオ

- 以下のコマンドを実行し、Slowクエリログを読み込ませます

```
# pt-table-usage /var/lib/mysql/slow_queries.log
```

■ 結果

- 以下のように、それぞれどのテーブルにアクセスされているかを標準出力します



```
root@hogehoge1:~  
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)  
[root@hogehoge1 ~]# pt-table-usage /var/lib/mysql/slow_queries.log  
Query_id: 0xE3A3649C5FAC418D.1  
SELECT DUAL  
  
Query_id: 0xFB38455BF7035609.1  
SELECT DUAL  
  
Query_id: 0x0CF8287A02430118.1  
SELECT articles  
  
Query_id: 0x607E11E221CFDFDD.1  
LOAD TABLE  
  
Query_id: 0x0CF8287A02430118.1  
SELECT articles  
  
Query_id: 0xEC5522E990A3C0D2.1  
SELECT articles  
  
Query_id: 0x3A75587C3740639A.1  
SELECT comments
```

■ 良い点

- どのテーブルアクセスにおいて、Slowクエリログが多発しているか等を簡単に確認する事ができます

■ コマンド

pt-upgrade [オプション] [ログファイル] [DSN1] [DSN2]

【必須項目】

- [DSN] : h=ホスト名,u=ユーザ名,p=パスワード
- [ログファイル] : 比較するログファイルを指定します

【主なオプション】

- --no-read-only : ログファイルの全てのクエリを比較ようになります(通常はSELECT,SET文しか比較しません)
- --type : 比較するログファイルのフォーマットを指定することができます。通常はslowlogになっています

フォーマット名	ログの内容
slowlog	スロークエリログ
genlog	ジェネラルログ
binlog	バイナリログ
rowlog	1行に1つSQL文を書いた、オリジナルファイル

■ 目的

- 異なるサーバ間で、クエリ実行時間の違いや、エラーの有無を比較した結果を出力します

■ シナリオ

- バージョンが異なるサーバを二台用意します

mysql5.6	ホスト名=localhost、DBユーザ名=root、パスワード=password
mysql5.1	ホスト名=192.168.2.222、DBユーザ名=test、パスワード=password

※ デフォルトの設定では、SELECTの権限があればツールは動きます

- Slowクエリログは、デフォルトでは10秒以上のクエリが記録されるようになっていますが、0秒以上で記録するように変更します

```
long_query_time = 0
```

my.cnf に追記

- localhost(mysql5.6)側でいくつかクエリを実行します

SELECT文	10件
INSERT文	5件

※ performance_schemaに対するSELECT文を1件実行しています

- コマンドを実行する前に、my.cnfのlong_query_timeオプションを元に戻し、その後ログを比較します

```
# pt-upgrade /var/lib/mysql/slow_queries.log \  
> h=localhost,p=password h=192.168.2.222,u=test,p=password
```

■ 結果

```
[root@holon4 ~]# pt-upgrade /var/lib/mysql/slow_queries.log h=localhost,p=password
h=192.168.2.222,u=test,p=password

#-----
# Logs
#-----

File: /var/lib/mysql/slow_queries.log
Size: 2518

#-----
# Hosts
#-----

host1:

  DSN:      h=localhost
  hostname: holon4
  MySQL:    MySQL Community Server (GPL) 5.6.25

host2:

  DSN:      h=192.168.2.222
  hostname: localhost.localdomain
  MySQL:    Source distribution 5.1.73

#####
# Query class 16D6F6135AF12700
#####

Reporting class because it has diffs, but hasn't been reported yet.

Total queries      1
Unique queries     1
Discarded queries  0

select count(*) from performance_schema.users

##
## Query errors diffs: 1
##

-- 1.

No error

vs.

DBD::mysql::st execute failed: Table 'performance_schema.users' doesn't exist [for S
tatement "SELECT count(*) FROM performance_schema.users"]

SELECT count(*) FROM performance_schema.users

#-----
# Stats
#-----

failed_queries      0
not_select          5
queries_filtered    0
queries_no_diffs    9
queries_read        15
queries_with_diffs  0
queries_with_errors 1
```

■ 良い点

- 環境の異なるサーバ間で、同じクエリを実行した結果を簡単に確かめることができます
- バージョンアップを行う前に、移行後のデータの安全・整合性を確かめることができます

■ その他

- クエリの実行結果を一度ファイルに保存しておき、そのファイルと比較することもできます

pt-upgrade [オプション] --save-results [保存先ディレクトリ] [ログファイル] [DSN1]

pt-upgrade [オプション] [リザルトディレクトリ] [DSN2]

- ・ --save-results [保存先ディレクトリ] : ログファイルに書かれたクエリと、その実行結果やレコードを保存します
- ・ [リザルトディレクトリ] : リザルトファイルのあるディレクトリを指定します

- --save-results を使用すると、保存先のディレクトリに以下のようなファイルが作成されます

query	実行されたクエリ
results	クエリの実行結果(エラー結果もしくは、警告や実行時間)
rows	抽出されたレコードの内容

- 比較結果が出力されるのは、以下のような場合です

Row Count	抽出されたレコードの行数が違う場合
Row Data	抽出されたレコードの内容が違う場合
Warnings	片方のホストで警告が出た場合
Query time	クエリの実行時間が大幅に異なる場合
Query errors	片方のホストでエラーが起きた場合
SQL errors	両方のホストでエラーが起きた場合(文法エラー)

■ コマンド

```
pt-variable-advisor [ オプション ][ DSN ]
```

【必須項目】

- [オプション] : -p パスワード
- [DSN] : h=ホスト名

■ 目的

- MySQL変数を分析して、問題の可能性のある設定について出力します

■ 設定ファイル

- 必須オプションを、設定ファイルにまとめておきます

```
# touch /etc/percona-toolkit/pt-variable-advisor.conf  
# vi /etc/percona-toolkit/pt-variable-advisor.conf
```

設定ファイルの作成
設定ファイルの編集

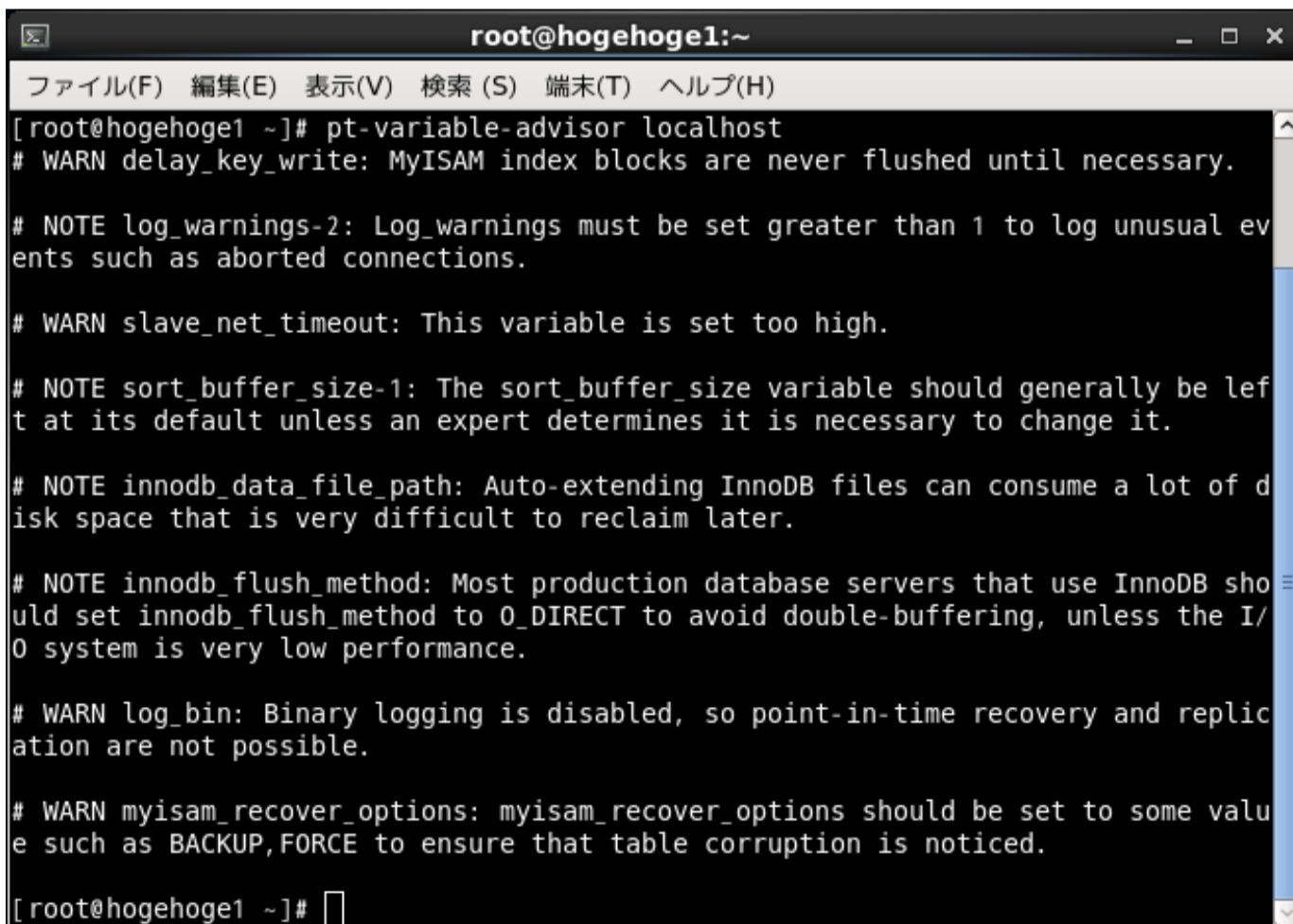
```
# config for pt-variable-advisor  
password=パスワード
```

コメント
MySQL のパスワードを記入

■ 結果

- コマンドを実行すると、変更した方がいい変数が"# WARN"で明示されます

```
# pt-variable-advisor localhost
```



```
root@hoge1:~# pt-variable-advisor localhost  
# WARN delay_key_write: MyISAM index blocks are never flushed until necessary.  
  
# NOTE log_warnings-2: Log_warnings must be set greater than 1 to log unusual ev  
ents such as aborted connections.  
  
# WARN slave_net_timeout: This variable is set too high.  
  
# NOTE sort_buffer_size-1: The sort_buffer_size variable should generally be lef  
t at its default unless an expert determines it is necessary to change it.  
  
# NOTE innodb_data_file_path: Auto-extending InnoDB files can consume a lot of d  
isk space that is very difficult to reclaim later.  
  
# NOTE innodb_flush_method: Most production database servers that use InnoDB sho  
uld set innodb_flush_method to O_DIRECT to avoid double-buffering, unless the I/  
O system is very low performance.  
  
# WARN log_bin: Binary logging is disabled, so point-in-time recovery and replic  
ation are not possible.  
  
# WARN myisam_recover_options: myisam_recover_options should be set to some valu  
e such as BACKUP, FORCE to ensure that table corruption is noticed.  
[root@hoge1 ~]#
```

■ 良い点

- サーバ設定において一般的な問題がないかを確認する際に有効です

■ コマンド

pt-visual-explain [オプション] [ファイル名]

■ 目的

- EXPLAIN文を元に、クエリの構成をツリー構造で出力します

■ シナリオ

- まずはEXPLAIN文の実行結果を入力したファイルを用意します
"test"データベースの"articles"テーブルから、10000 < id < 20000 のレコードをセレクトする、
というクエリのEXPLAIN結果を pt-visual-explain.txt というファイルに出力させます

```
# mysql -uroot -p -e "EXPLAIN SELECT * FROM articles WHERE id > 10000 and id < 20000"
test > /tmp/pt-visual-explain_resu
```

- そのファイルを、pt-visual-explain で読み取ります

```
# pt-visual-explain /tmp/pt-visual-explain_result.txt
```

■ 結果

- 以下のようなツリー構造が標準出力されます

```
root@hoge1:~
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[root@hoge1 ~]# pt-visual-explain /tmp/pt-visual-explain_result.txt
Filter with WHERE
+- Bookmark lookup
  +- Table
    | table      articles
    | possible_keys PRIMARY
  +- Index range scan
    key         articles->PRIMARY
    possible_keys PRIMARY
    key_len     4
    rows        19188
[root@hoge1 ~]#
```

■ 良い点

- EXPLAINの結果を視覚的に分かりやすく表示できます

■ その他

- EXPLAIN文をパイプで引き渡すこともできます

```
# mysql -e "EXPLAIN SELECT * FROM mysql.user" | pt-visual-explain
```