**Percona-toolkit使用教程**

# 一、percona-toolkit 简介

percona-toolkit 是一组高级命令行工具的集合，用来执行各种通过手工执行非常复  
杂和麻烦的 mysql 任务和系统任务，这些任务包括：  
⚫检查 master 和 slave 数据的一致性  
⚫有效地对记录进行归档  
⚫查找重复的索引  
⚫对服务器信息进行汇总  
⚫分析来自日志和 tcpdump 的查询  
⚫当系统出问题的时候收集重要的系统信息  
percona-toolkit 源自 Maatkit 和 Aspersa 工具，这两个工具是管理 mysql 的最有名的  
工具，现在 Maatkit 工具已经不维护了，请大家还是使用 percona-toolkit 吧！ 这些  
工具主要包括开发、性能、配置、监控、复制、系统、实用六大类，作为一个优秀  
的 DBA，里面有的工具非常有用，如果能掌握并加以灵活应用，将能极大的提高工  
作效率。

# 二、percona-toolkit 工具包安装

|  |  |
| --- | --- |
| 1. | 软件包下载  访 问 http://www.percona.com/software/percona-toolkit/ 下 载 最 新 版 本 的 |

Percona Toolkit 或者通过如下命令行来获取最新的版本：  
wget percona.com/get/percona-toolkit.tar.gz  
wget percona.com/get/percona-toolkit.rpm  
我这里选择直接从网站上找到最新版本下载：  
wget http://www.percona.com/redir/downloads/percona-toolkit/2.1.1/percona-t  
oolkit-2.1.1-1.noarch.rpm  
wget http://www.percona.com/redir/downloads/percona-toolkit/2.1.1/percona-t  
oolkit-2.1.1.tar.gz  
从 http://pkgs.repoforge.org/perl-TermReadKey/下载最新的 TermReadKey 包  
wget http://pkgs.repoforge.org/perl-TermReadKey/perl-TermReadKey-2.30-1.el5.  
rf.x86\_64.rpm  
2. 软件包安装  
我的环境是 Centos 5.5 64 BIT  
A. percona-toolkit 的 rpm 安装方式  
rpm -ivh perl-TermReadKey-2.30-1.el5.rf.x86\_64.rpm  
rpm -ivh percona-toolkit-2.1.1-1.noarch.rpm  
注意：需要安装 Term::ReadKey 包，否则会报 perl(Term::ReadKey) >= 2.10  
is needed by percona-toolkit-2.1.1-1.noarch 错误  
B. percona-toolkit 的编译安装方式  
tar xzvf percona-toolkit-2.1.1.tar.gz  
cd percona-toolkit-2.1.1  
perl Makefile.PL  
make  
make test  
make install

# 三、percona-toolkit 的使用

根据 percona-toolkit 的工具类型可以总结出下面五个类别，方便大家进行学习和实  
践，下面就针对这些不同的类别来分别介绍这些工具的用法。

## (一)开发类工具

1. pt-duplicate-key-checker

|  |  |
| --- | --- |
| ⚫ | 功能介绍： 功能为从 mysql 表中找出重复的索引和外键，这个工具会将重复的索 |

引和外键都列出来，并生成了删除重复索引的语句，非常方便  
⚫用法介绍：  
pt-duplicate-key-checker [OPTION...] [DSN]  
包含比较多的选项，具体的可以通过命令 pt-duplicate-key-checker  
--help 来查看具体支持那些选项，我这里就不一一列举了。 DNS 为数  
据库或者表。  
⚫使用示例：  
查看 test 数据库的重复索引和外键使用情况使用如下命令

|  |  |  |
| --- | --- | --- |
| pt-duplicate-key-checker  --password=zhang@123 | --host=localhost  --databases=test | --user=root |

2. pt-online-schema-change  
⚫功能介绍：  
功能为在 alter 操作更改表结构的时候不用锁定表，也就是说执行  
alter 的时候不会阻塞写和读取操作， 注意执行这个工具的时候必须  
做 好 备 份 ， 操 作 之 前 最 好 详 细 读 一 下 官 方 文 档  
http://www.percona.com/doc/percona-toolkit/2.1/pt-online-schema-ch  
ange.html。  
工作原理是创建一个和你要执行 alter 操作的表一样的空表结构，执  
行表结构修改，然后从原表中 copy 原始数据到表结构修改后的表，  
当数据 copy 完成以后就会将原表移走，用新表代替原表，默认动作  
是将原表 drop 掉。 在 copy 数据的过程中，任何在原表的更新操作都  
会更新到新表，因为这个工具在会在原表上创建触发器，触发器会将  
在原表上更新的内容更新到新表。 如果表中已经定义了触发器这个工  
具就不能工作了。  
⚫用法介绍：  
pt-online-schema-change [OPTIONS] DSN  
options 可以自行查看 help， DNS 为你要操作的数据库和表。  
这里有两个参数需要介绍一下：  
--dry-run 这个参数不建立触发器，不拷贝数据， 也不会替换原表。  
只是创建和更改新表。  
--execute 这个参数的作用和前面工作原理的介绍的一样，会建立触  
发器，来保证最新变更的数据会影响至新表。 注意：如果不加这个参  
数，这个工具会在执行一些检查后退出。这一举措是为了让使用这充  
分了解了这个工具的原理，同时阅读了官方文档。  
⚫使用示例：  
范例 1： 在线更改表的的引擎，这个尤其在整理 innodb 表的时候非  
常有用，示例如下：  
pt-online-schema-change --user=root --password=zhang@123 --host=l  
ocalhost --lock-wait-time=120 --alter="ENGINE=InnoDB" D=test,t=oss\_  
pvinfo2 --execute  
从下面的日志中可以看出它的执行过程：  
Altering `test`.`oss\_pvinfo2`...  
Creating new table...  
Created new table test.\_oss\_pvinfo2\_new OK.  
Altering new table...  
Altered `test`.`\_oss\_pvinfo2\_new` OK.  
Creating triggers...  
Created triggers OK.  
Copying approximately 995696 rows...  
Copied rows OK.  
Swapping tables...  
Swapped original and new tables OK.  
Dropping old table...  
Dropped old table `test`.`\_oss\_pvinfo2\_old` OK.  
Dropping triggers...  
Dropped triggers OK.  
Successfully altered `test`.`oss\_pvinfo2`.  
范例 2： 大表添加字段的，语句如下:  
pt-online-schema-change --user=root --password=zhang@123 --host=l  
ocalhost --lock-wait-time=120 --alter="ADD COLUMN domain\_id INT"  
D=test,t=oss\_pvinfo2 --execute  
3. pt-query-advisor

|  |  |
| --- | --- |
| ⚫ | 功能介绍： |
| 根据一些规则分析查询语句，对可能的问题提出建议， 这些评判规则 |  |
| 大 | 家 | 可 | 以 | 看 | 一 | 下 | 官 | 网 | 的 | 链 | 接 | ： |

http://www.percona.com/doc/percona-toolkit/2.1/pt-query-advisor.ht  
ml，这里就不详细列举了。那些查询语句可以来自慢查询文件、general  
日志文件或者使用 pt-query-digest 截获的查询语句。 目前这个版本有  
bug，当日志文件非常大的时候会需要很长时间甚至进入死循环。  
⚫用法介绍：  
pt-query-advisor /path/to/slow-query.log  
pt-query-advisor --type genlog mysql.log  
pt-query-digest --type tcpdump.txt --print --no-report | pt-query-advisor  
⚫使用示例：  
分析一个语句的例子：  
pt-query-advisor --query "select \* from aaa"  
分析 general log 中的查询语句的例子：  
pt-query-advisor /data/dbdata/general.log  
分析慢查询中的查询语句的例子：  
pt-query-advisor /data/dbdata/localhost-slow.log  
4. pt-show-grants  
⚫功能介绍：  
规范化和打印 mysql 权限，让你在复制、比较 mysql 权限以及进行版  
本控制的时候更有效率！  
⚫用法介绍：  
pt-show-grants [OPTION...] [DSN]  
选项自行用 help 查看， DSN 选项也请查看 help，选项区分大小写。  
⚫使用示例：  
查看指定 mysql 的所有用户权限：  
pt-show-grants --host='localhost' --user='root' --password='zhang@123'  
查看执行数据库的权限：  
pt-show-grants --host='localhost' --user='root' --password='zhang@123'  
--database='hostsops'  
查看每个用户权限生成 revoke 收回权限的语句：  
pt-show-grants --host='localhost' --user='root' --password='zhang@123'  
--revoke  
5. pt-upgrade

|  |  |
| --- | --- |
| ⚫ | 功能介绍： 在多台服务器上执行查询，并比较有什么不同！ 这在升级服务器的时 |

候非常有用，可以先安装并导数据到新的服务器上，然后使用这个工  
具跑一下 sql 看看有什么不同，可以找出不同版本之间的差异。  
⚫用法介绍：  
pt-upgrade [OPTION...] DSN [DSN...] [FILE]  
比较文件中每一个查询语句在两个主机上执行的结果，并检查在每个  
服务器上执行的结果、错误和警告。  
⚫使用示例：  
只查看某个 sql 在两个服务器的运行结果范例：  
pt-upgrade h='localhost' h=192.168.3.92 --user=root --password=zha  
ng@123 --query="select \* from user\_data.collect\_data limit 5"  
查看文件中的对应 sql 在两个服务器的运行结果范例：  
pt-upgrade h='localhost' h=192.168.3.92 --user=root --password=zha  
ng@123 aaa.sql  
查看慢查询中的对应的查询 SQL 在两个服务器的运行结果范例：  
pt-upgrade h='localhost' h=192.168.3.92 --user=root --password=zha  
ng@123 slow.log  
此 外 还 可 以 执 行 compare 的 类 型 ， 主 要 包 含 三 个  
query\_times,results,warnings，比如下面的例子，只比较 sql 的执行时  
间  
pt-upgrade h=192.168.3.91 h=192.168.3.92 --user=root --password=z  
hang@123 --query="select \* from user\_data.collect\_data" --compar  
e query\_times

## (二)性能类工具

1. pt-index-usage  
⚫功能介绍：  
从 log 文件中读取插叙语句，并用 explain 分析他们是如何利用索引。  
完成分析之后会生成一份关于索引没有被查询使用过的报告。  
⚫用法介绍：  
pt-index-usage [OPTION...] [FILE...]  
可以直接从慢查询中获取 sql， FILE 文件中的 sql 格式必须和慢查询中  
个是一致，如果不是一直需要用 pt-query-digest 转换一下。也可以不  
生成报告直接保存到数据库中，具体的见后面的示例  
⚫使用示例：  
从满查询中的 sql 查看索引使用情况范例：  
pt-index-usage /data/dbdata/localhost-slow.log --host=localhost --use  
r=root --password=zhang@123  
将分析结果保存到数据库范例：  
pt-index-usage /data/dbdata/localhost-slow.log --host=localhost --use  
r=root --password=zhang@123 --no-report --create-save-results-data  
base  
使用--create-save-results-database 会自动生成数据库和表来保存结果。  
2. pt-pmp  
⚫功能介绍：  
为查询程序执行聚合的 GDB 堆栈跟踪，先进性堆栈跟踪，然后将跟  
踪信息汇总。  
⚫用法介绍：  
pt-pmp [OPTIONS] [FILES]  
⚫使用示例：  
pt-pmp -p 21933  
pt-pmp -b /usr/local/mysql/bin/mysqld\_safe  
3. pt-visual-explain  
⚫功能介绍：  
格式化 explain 出来的执行计划按照 tree 方式输出，方便阅读。  
⚫用法介绍：  
pt-visual-explain [OPTION...] [FILE...]  
option 请参阅官方网站，这里不一一例举！  
⚫使用示例：  
查看包含 explain 结果的 aaa 文件的范例：  
pt-visual-explain aaa  
查看包含查询语句的 aaa 文件的范例：  
pt-visual-explain --connect aaa --user=root --password=zhang@123  
通过管道直接查看 explain 输出结果的范例：  
mysql -uroot -pzhang@123 -e "explain select email from test.colle  
ct\_data where id=101992419" |pt-visual-explain

## (三)配置类工具

1. pt-config-diff  
⚫功能介绍：  
比较 mysql 配置文件和服务器参数  
⚫用法介绍：  
pt-config-diff [OPTION...] CONFIG CONFIG [CONFIG...]  
CONFIG 可以是文件也可以是数据源名称，最少必须指定两个配置文  
件源，就像 unix 下面的 diff 命令一样，如果配置完全一样就不会输  
出任何东西。  
⚫使用示例：  
范例 1：查看本地和远程服务器的配置文件差异：  
pt-config-diff h=localhost h=192.168.3.92 --user=root --password=zha  
ng@123  
比较出来内容如下：  
22 config differences

|  |  |
| --- | --- |
| Variable  localhost.localdomain =========================  ===================== | localhost.localdomain |
| ===================== |  |
| binlog\_cache\_size 8388608 2097152 |  |
| have\_ndbcluster DISABLED | NO |

innodb\_additional\_mem\_... 16777216 33554432  
innodb\_buffer\_pool\_size 1677721600 1073741824  
范例 2：比较本地配置文件和远程服务器的差异：  
pt-config-diff /etc/my.cnf h=192.168.3.92 --user=root --password=zha  
ng@123  
比较出来内容如下：  
12 config differences  
Variable /etc/my.cnf localhost.localdomain  
========================= =========== =====================  
binlog\_cache\_size 8388608 2097152  
binlog\_format mixed MIXED  
范例 3：比较本地两个配置文件的差异：  
pt-config-diff /usr/local/mysql/share/mysql/my-large.cnf /usr/local/m  
ysql/share/mysql/my-medium.cnf  
2. pt-mysql-summary  
⚫功能介绍：  
精细地对 mysql 的配置和 sataus 信息进行汇总，汇总后你直接看一眼  
就能看明白。  
⚫用法介绍：  
pt-mysql-summary [OPTIONS] [-- MYSQL OPTIONS]  
工作原理： 连接 mysql 后查询出 status 和配置信息保存到临时目录中，  
然后用 awk 和其他的脚本工具进行格式化。 OPTIONS 可以查阅官网  
的相关页面。  
⚫使用示例：  
范例 1：汇总本地 mysql 服务器的 status 和配置信息：  
pt-mysql-summary -- --user=root --password=zhang@123 --host=local  
host  
范例 2：汇总本地 mysql 服务器 192.168.3.92 的 status 和配置信息：  
pt-mysql-summary -- --user=root --password=zhang@123 --host=192.  
168.3.92  
3. pt-variable-advisor

|  |  |
| --- | --- |
| ⚫ | 功能介绍： 分析 mysql 的参数变量，并对可能存在的问题提出建议 |

⚫用法介绍：  
pt-variable-advisor [OPTION...] [DSN]  
原理：根据预先定义的规则检查 show variables 中的配置错误的设置  
和值。

|  |  |
| --- | --- |
| ⚫ | 使用示例： 范例 1：从 localhost 获取变量值 |

pt-variable-advisor --user=root --password=zhang@123 localhost  
范例 2： 从指定的文件中读取配置，这个有格式要求

|  |  |  |
| --- | --- | --- |
| pt-variable-advisor  --source-of-variables my.cnf | --user=root | --password=zhang@123 |

## (四)监控类工具

1. pt-deadlock-logger  
⚫功能介绍：  
提取和记录 mysql 死锁的相关信息  
⚫用法介绍：  
pt-deadlock-logger [OPTION...] SOURCE\_DSN  
收集和保存 mysql 上最近的死锁信息，可以直接打印死锁信息和存储  
死锁信息到数据库中， 死锁信息包括发生死锁的服务器、最近发生死  
锁的时间、死锁线程 id、死锁的事务 id、发生死锁时事务执行了多长  
时间等等非常多的信息。 详情见下面的示例。  
⚫使用示例：  
范例 1： 打印本地 mysql 的死锁信息

|  |  |
| --- | --- |
| pt-deadlock-logger  –print | --user=root --password=zhang@123 h=localhost |

范例 2：将本地的 mysql 死锁信息记录到数据库的表中，也打印出来  
pt-deadlock-logger --user=root --password=zhang@123 h=localhost  
--print D=test,t=deadlocks  
2. pt-fk-error-logger

|  |  |
| --- | --- |
| ⚫ | 功能介绍： 提取和记录 mysql 外键错误信息 |

⚫用法介绍：  
pt-fk-error-logger [OPTION...] SOURCE\_DSN  
通过SHOW INNODB STATUS提取和保存mysql数据库最近发生的外键  
错误信息。 可以通过参数控制直接打印错误信息或者将错误信息存储  
到数据库的表中。  
⚫使用示例：  
我在服务器上运行的时候一直报如下错误：  
Use of uninitialized value in concatenation (.) or string at /usr/bin  
/pt-fk-error-logger line 2045  
我怀疑是这个程序有问题，回头换一个版本试一下或者调试一下那个  
程序。  
3. pt-mext

|  |  |
| --- | --- |
| ⚫ | 功能介绍： 并行查看 SHOW GLOBAL STATUS 的多个样本的信息。 |

⚫用法介绍：  
pt-mext [OPTIONS] -- COMMAND  
原理： pt-mext 执行你指定的 COMMAND，并每次读取一行结果，把  
空行分割的内容保存到一个一个的临时文件中，最后结合这些临时文  
件并行查看结果。  
⚫使用示例：  
范例 1：每隔 10s 执行一次 SHOW GLOBAL STATUS，并将结果合并到  
一起查看  
pt-mext -- mysqladmin ext -uroot -pzhang@123 -i10 -c3  
4. pt-query-digest  
⚫功能介绍：  
分析查询执行日志，并产生一个查询报告，为 MySQL、 PostgreSQL、  
memcached 过滤、重放或者转换语句。  
⚫用法介绍：  
pt-query-digest [OPTION...] [FILE]  
解析和分析 mysql 日志文件  
⚫使用示例：  
范例 1：分析本地的慢查询文件  
pt-query-digest --user=root --password=zhang@123 /data/dbdata/loc  
alhost-slow.log  
范例 2：重新回顾满查询日志，并将结果保存到 query\_review 中，注  
意 query\_review 表的表结构必须先建好，表结构如下：  
CREATE TABLE query\_review (  
checksum BIGINT UNSIGNED NOT NULL PRIMARY KEY,  
fingerprint TEXT NOT NULL,  
sample TEXT NOT NULL,  
first\_seen DATETIME,  
last\_seen DATETIME,  
reviewed\_by VARCHAR(20),  
reviewed\_on DATETIME,  
comments TEXT  
);  
命令如下：  
pt-query-digest --user=root --password=zhang@123 --review h=localh  
ost,D=test,t=query\_review /data/dbdata/localhost-slow.log  
5. pt-trend

|  |  |
| --- | --- |
| ⚫ | 功能介绍： 居于一组时间序列的数据点做统计。 |

⚫用法介绍：  
pt-trend [OPTION...] [FILE ...]  
读取一个慢查询日志，并输出统计信息。 也可以指定多个文件。 如果  
不指定文件的话直接从标准输入中读取信息。  
⚫使用示例：  
范例 1：读取本地慢查询日志并输出统计信息  
pt-trend /data/dbdata/localhost-slow.log  
这里输出的信息没有说明，有点看不明白！

## (五)复制类工具

1. pt-heartbeat  
⚫功能介绍：  
监控 mysql 复制延迟  
⚫用法介绍：  
pt-heartbeat [OPTION...] [DSN] --update|--monitor|--check|--stop  
测量复制落后主 mysql 或者主 PostgreSQL 多少时间，你可以使用这个  
脚本去更新主或者监控复制，具体用法见后面的示例.  
原理： pt-heartbeat 通过真实的复制数据来确认 mysql 和 postgresql  
复制延迟，这个避免了对复制机制的依赖，从而能得出准确的落后复  
制时间，包含两部分：第一部分在主上 pt-heartbeat 的--update 线程  
会在指定的时间间隔更新一个时间戳，第二部分是 pt-heartbeat 的  
--monitor 线程或者--check 线程连接到从上检查复制的心跳记录（前  
面更新的时间戳），并和当前系统时间进行比较，得出时间的差异。  
你可以手工创建 heartbeat 表或者添加-*-create-table* 参数，推荐使用  
MEMORY 引擎。表结构为：  
CREATE TABLE heartbeat (  
ts varchar(26) NOT NULL,  
server\_id int unsigned NOT NULL PRIMARY KEY,  
file varchar(255) DEFAULT NULL, -- SHOW MASTER STATUS  
position bigint unsigned DEFAULT NULL, -- SHOW MASTER STATUS  
relay\_master\_log\_file varchar(255) DEFAULT NULL, -- SHOW SLAVE STATUS  
exec\_master\_log\_pos bigint unsigned DEFAULT NULL -- SHOW SLAVE STATUS  
);  
⚫使用示例：  
范例 1：创建一个后台进程定期更新主上的 test 库的 heartbeat 表（）  
默认是 1s，可以--interval 指定，执行后会成一个 heartbeat 表， test  
库为我监控的同步库：  
pt-heartbeat -D test --update --user=root --password=zhang@123 -h  
192.168.3.135 --create-table –daemonize  
范例 2：监控复制在 slave 上的落后程度（会一直监控）：  
pt-heartbeat -D test --monitor --user=root --password=zhang@123 -  
h192.168.3.92  
监控结果如下：  
0.00s [ 0.00s, 0.00s, 0.00s ]  
0.00s [ 0.00s, 0.00s, 0.00s ]  
0.00s [ 0.00s, 0.00s, 0.00s ]  
0.00s [ 0.00s, 0.00s, 0.00s ]  
0.00s [ 0.00s, 0.00s, 0.00s ]  
0.00s [ 0.00s, 0.00s, 0.00s ]  
范例 3：监控复制在 slave 上的落后程度（监控一次退出）：  
pt-heartbeat -D test --check --user=root --password=zhang@123 -h1  
92.168.3.92  
范例 4： 监控 PostgreSQL 需要添加--dbi-driver Pg：  
pt-heartbeat -D test --check --user=root --password=zhang@123 -h1  
92.168.3.92 --dbi-driver Pg  
2. pt-slave-delay

|  |  |
| --- | --- |
| ⚫ | 功能介绍： 设置从服务器落后于主服务器指定时间。 |

⚫用法介绍：  
pt-slave-delay [OPTION...] SLAVE-HOST [MASTER-HOST]  
原理：通过启动和停止复制 sql 线程来设置从落后于主指定时间。 默  
认是基于从上 relay 日志的二进制日志的位置来判断，因此不需要连  
接到主服务器，如果 IO 进程不落后主服务器太多的话，这个检查方  
式工作很好，如果网络通畅的话，一般 IO 线程落后主通常都是毫秒  
级别。 一般是通过*--delay and --delay"+"--interval* 来控制。 *--interval* 是  
指定检查是否启动或者停止从上 sql 线程的频繁度，默认的是 1 分钟  
检查一次。  
⚫使用示例：  
范例 1：使从落后主 1 分钟，并每隔 1 分钟检测一次，运行 10 分钟  
pt-slave-delay --user=root --password=zhang@123 --delay 1m --run-t  
ime 10m --host=192.168.3.92  
如果不加--run-time 参数会一直执行。  
范例 2： 使从落后主 1 分钟，并每隔 15 秒钟检测一次，运行 10 分钟  
pt-slave-delay --user=root --password=zhang@123 --delay 1m --inter  
val 15s --run-time 10m --host=192.168.3.92  
运行结果如下：  
2012-05-20T16:34:50 slave running 0 seconds behind  
2012-05-20T16:34:50 STOP SLAVE until 2012-05-20T16:35:50 at mas  
ter position mysql-bin.000032/4392054  
2012-05-20T16:35:05 slave stopped at master position mysql-bin.00  
0032/4397124  
2012-05-20T16:35:20 slave stopped at master position mysql-bin.00  
0032/4402194  
2012-05-20T16:35:35 slave stopped at master position mysql-bin.00  
0032/4407264  
2012-05-20T16:35:50 no new binlog events  
2012-05-20T16:36:05 START SLAVE until master 2012-05-20T16:35:0  
5 mysql-bin.000032/4397124  
3. pt-slave-find

|  |  |
| --- | --- |
| ⚫ | 功能介绍： 查找和打印 mysql 所有从服务器复制层级关系 |

⚫用法介绍：  
pt-slave-find [OPTION...] MASTER-HOST  
原理:连接 mysql 主服务器并查找其所有的从，然后打印出所有从服  
务器的层级关系。  
⚫使用示例：  
范例1：查找主服务器为192.168.3.135的mysql有所有从的层级关系：  
pt-slave-find --user=root --password=zhang@123 --host=192.168.3.13  
5  
4. pt-slave-restart  
⚫功能介绍：  
监视 mysql 复制错误，并尝试重启 mysql 复制当复制停止的时候  
⚫用法介绍：  
pt-slave-restart [OPTION...] [DSN]  
监视一个或者多个 mysql 复制错误，当从停止的时候尝试重新启动复  
制。你可以指定跳过的错误并运行从到指定的日志位置。  
⚫使用示例：  
范例 1：监视 192.168.3.92 的从，跳过 1 个错误  
pt-slave-restart --user=root --password=zhang@123 --host=192.168.3.  
92 --skip-count=1  
范例 2：监视 192.168.3.92 的从，跳过错误代码为 1062 的错误。  
pt-slave-restart --user=root --password=zhang@123 --host=192.168.3.  
92 --error-numbers=1062  
5. pt-table-checksum

|  |  |
| --- | --- |
| ⚫ | 功能介绍： 检查 mysql 复制一致性 |

⚫用法介绍：  
pt-table-checksum [OPTION...] [DSN]  
工作原理： pt-table-checksum 在主上执行检查语句在线检查 mysql 复  
制的一致性， 生成 replace 语句，然后通过复制传递到从，再通过  
update 更新 master\_src 的值。 通过检测从上 this\_src 和 master\_src 的  
值从而判断复制是否一致。  
注意： 使用的时候选择业务地峰的时候运行，因为运行的时候会造成  
表的部分记录锁定。 使用*--max-load* 来指定最大的负载情况，如果达  
到那个负载这个暂停运行。 如果发现有不一致的数据，可以使用  
pt-table-sync 工具来修复。  
注意： 和 1.0 版本不同，新版本的 pt-table-checksum 只需要在 master  
上执行即可。  
通过 –explain 参数再结合二进制日志就可以看出脚本的工作原理，  
如我的 test 库有一个名字为 zhang 的表，我们通过抓取二进制日志来  
查看脚本的原理：  
REPLACE INTO `test`.`checksums` (db, tbl, chunk, chunk\_index, lowe  
r\_boundary, upper\_boundary, this\_cnt, this\_crc) SELECT 'test', 'zhan  
g', '1', NULL, NULL, NULL, COUNT(\*) AS cnt, COALESCE(LOWER(CON  
V(BIT\_XOR(CAST(CRC32(CONCAT\_WS('#', `id`, `name`, CONCAT(ISNULL  
(`name`)))) AS UNSIGNED)), 10, 16)), 0) AS crc FROM `test`.`zhang`  
/\*checksum table\*/;  
UPDATE `test`.`checksums` SET chunk\_time = '0.000563', master\_crc  
= '31012777', master\_cnt = '4' WHERE db = 'test' AND tbl = 'zha  
ng' AND chunk = '1'  
从这里可以很明显的看出原理了，前面已经说了，这里就不赘述了。  
⚫使用示例：  
范例 1：比较 test 数据库同步是否一致，结果显示所有的表。  
pt-table-checksum --nocheck-replication-filters --databases=test --re

|  |  |
| --- | --- |
| plicate=test.checksums --create-replicate-table | --host=192.168.3.135 |
| --port 3306 | -uroot -pzhang@123 |

参数说明：第一次运行的时候需要添加--create-replicate-table 参数，  
如果不加这个就需要手工运行添加表结构的 SQL,表结构 SQL 如下：  
CREATE TABLE checksums (  
db char(64) NOT NULL,  
tbl char(64) NOT NULL,  
chunk int NOT NULL,  
chunk\_time float NULL,  
chunk\_index varchar(200) NULL,  
lower\_boundary text NULL,  
upper\_boundary text NULL,  
this\_crc char(40) NOT NULL,  
this\_cnt int NOT NULL,  
master\_crc char(40) NULL,  
master\_cnt int NULL,  
ts timestamp NOT NULL,  
PRIMARY KEY (db, tbl, chunk),  
INDEX ts\_db\_tbl (ts, db, tbl)  
) ENGINE=InnoDB;  
之所以使用--nocheck-replication-filters参数是因为我的my.cnf配置了  
replicate-ignore-db 和 replicate-wild-do-table 等参数。 另外需要特别注  
意执行的 checksums 所在的数据库必须是同步的数据库。我刚开始使  
用的时候摸索的很久，官网也没有范例。呵呵！  
结果如下：

|  |  |
| --- | --- |
| TS | ERRORS DIFFS ROWS CHUNKS SKIPPED TIME TABLE |
| 05-23T16:19:29 0 1 2 1  05-23T16:19:29 0 0 1 1  05-23T16:19:29 0 0 0 1 | 0 0.006 test.aaa 0 0.017 test.bbb 0 0.007 test.category\_part |
| 05-23T16:19:31 0 0 233617 | 6 | 0 1.887 test.collect\_data |
| 05-23T16:19:34 0 0 250346 5  05-23T16:19:34 0 1 1 1  05-23T16:19:39 0 0 1000000 11 | 0 2.709 test.effective\_user 0 0.008 test.heartbeat 0 5.353 test.oss\_pvinfo2 |  |

从结果中，我们可以看到 test.aaa和 test.heartbeat 表的 DIFFS 不为 0，  
那么就是这两个表不同步了。  
范例 2：比较 test 数据库同步是否一致，结果只显示数据不一致的表  
（添加--replicate-check-only 参数即可）。  
pt-table-checksum --nocheck-replication-filters --databases=test --re  
plicate=test.checksums --replicate-check-only --lock-wait-timeout=120  
--host=192.168.3.135 --port 3306 --user=root --password=zhang  
@123  
结果如下：  
Differences on localhost.localdomain  
TABLE CHUNK CNT\_DIFF CRC\_DIFF CHUNK\_INDEX LOWER\_BOUNDAR  
Y UPPER\_BOUNDARY  
test.aaa 1 1 1  
test.heartbeat 1 0 1  
从结果可以看出，只显示了两个不同步的表。  
6. pt-table-sync  
⚫功能介绍：  
高效同步 mysql 表的数据  
⚫用法介绍：  
pt-table-sync [OPTION...] DSN [DSN...]  
原理：总是在主上执行数据的更改，再同步到从上，不会直接更改成  
从的数据，在主上执行更改是基于主上现在的数据，不会更改主上的  
数据。注意使用之前先备份你的数据，避免造成数据的丢失.执行  
execute 之前最好先换成--print 或--dry-run 查看一下会变更哪些数据。  
⚫使用示例：  
范例 1：同步 3.135 的 test 库的 aaa 表到 192.168.3.92，在执行之前  
可以用--execute 参数换成--print 来查看会变更什么东西，后面那个主  
机必须是 master，否则会报错推出。  
pt-table-sync --execute --user=root --password=zhang@123 h=192.16  
8.3.135,D=test,t=aaa h=192.168.3.92  
范例 2： 将主的 test 数据库同步到 192.168.3.92，使从上具有一样的  
数据。  
pt-table-sync --execute --sync-to-master --user=root --password=zhan  
g@123 h=192.168.3.92 --database test  
范例 3：只同步指定的表  
pt-table-sync --execute --sync-to-master --user=root --password=zhan  
g@123 h=192.168.3.92 D=test,t=aaa  
范例 4：根据 pt-table-checksum 的结果进行数据同步  
pt-table-sync --execute --replicate test.checksums --user=root --pass  
word=zhang@123 h=192.168.3.135  
范例 5： 根据 pt-table-checksum 使从的数据和主的数据一致  
pt-table-sync --execute --replicate test.checksums --user=root --pass  
word=zhang@123 --sync-to-master h=192.168.3.92 D=test,t=aaa  
**(**六**)**系统类工具  
1. pt-diskstats  
⚫功能介绍：  
是一个对 GUN/LINUX 的交互式监控工具  
⚫用法介绍：  
pt-diskstats [OPTION...] [FILES]  
为 GUN/LINUX 打印磁盘 io 统计信息，和 iostat 有点像，但是这个工  
具是交互式并且比 iostat 更详细。可以分析从远程机器收集的数据。  
⚫使用示例：  
范例 1：查看本机所有的磁盘的状态情况：  
pt-diskstats  
范例 2：只查看本机 sda2 磁盘的状态情况  
pt-diskstats --devices-regex sda2  
2. pt-fifo-split  
⚫功能介绍：  
模拟切割文件并通过管道传递给先入先出队列而不用真正的切割文  
件  
⚫用法介绍：  
pt-fifo-split [options] [FILE ...]  
pt-fifo-split 读取大文件中的数据并打印到 fifo 文件， 每次达到指定行  
数就往 fifo 文件中打印一个 EOF 字符，读取完成以后，关闭掉 fifo 文  
件并移走，然后重建 fifo 文件，打印更多的行。 这样可以保证你每次  
读取的时候都能读取到制定的行数直到读取完成。注意此工具只能工  
作在类 unix 操作系统。 这个程序对大文件的数据导入数据库非常有  
用， 具体的可以查看 http://www.mysqlperformanceblog.com/2008/07  
/03/how-to-load-large-files-safely-into-innodb-with-load-data-infile/。

|  |  |
| --- | --- |
| ⚫ | 使用示例： 范例 1：一个每次读取一百万行记录的范例： |

pt-fifo-split --lines 1000000 hugefile.txt  
while [ -e /tmp/pt-fifo-split ]; do cat /tmp/pt-fifo-split; done  
范例 2： 一个每次读取一百万行，指定 fifo 文件为/tmp/my-fifo，并使  
用 load data 命令导入到 mysql 中：  
pt-fifo-split infile.txt --fifo /tmp/my-fifo --lines 1000000  
while [ -e /tmp/my-fifo ]; do  
mysql -e "set foreign\_key\_checks=0; set sql\_log\_bin=0; set uniq  
ue\_checks=0; load data local infile '/tmp/my-fifo' into table load\_t  
est fields terminated by '\t' lines terminated by '\n' (col1, col2);"  
sleep 1;  
done  
3. pt-summary  
⚫功能介绍：  
友好地收集和显示系统信息概况，此工具并不是一个调优或者诊断工  
具，这个工具会产生一个很容易进行比较和发送邮件的报告。  
⚫用法介绍：  
pt-summary  
原理： 此工具会运行和多命令去收集系统状态和配置信息，先保存到  
临时目录的文件中去，然后运行一些 unix命令对这些结果做格式化，  
最好是用 root 用户或者有权限的用户运行此命令。  
⚫使用示例：  
范例 1：查看本地系统信息概况  
pt-summary  
4. pt-stalk  
⚫功能介绍：  
出现问题的时候收集 mysql 的用于诊断的数据  
⚫用法介绍：  
pt-stalk [OPTIONS] [-- MYSQL OPTIONS]  
pt-stalk 等待触发条件触发，然后收集数据帮助错误诊断，它被设计  
成使用 root 权限运行的守护进程，因此你可以诊断那些你不能直接  
观察的间歇性问题。默认的诊断触发条件为 SHOW GLOBAL STATUS。  
也可以指定 processlist 为诊断触发条件 ，使用--function 参数指定。  
⚫使用示例：  
范例 1：指定诊断触发条件为 status，同时运行语句超过 20 的时候触  
发，收集的数据存放在/tmp/test 目录下：

|  |  |
| --- | --- |
| pt-stalk | --function status --variable Threads\_running --threshold 20 |
| --dest /tmp/test | -- -uroot -pzhang@123 | -h192.168.3.135 |

范例 2：指定诊断触发条件为 processlist，超过 20 个状态为 statistics  
触发，收集的数据存放在/tmp/test 目录下：  
pt-stalk --function processlist --variable State --match statistics --t  
hreshold 20 --dest /tmp/test -- -uroot -pzhang@123 -h192.168.3.1  
35  
贴一下达到触发条件以后收集的信息：  
2012\_06\_04\_17\_31\_49-df  
2012\_06\_04\_17\_31\_49-disk-space  
2012\_06\_04\_17\_31\_49-diskstats  
2012\_06\_04\_17\_31\_49-hostname  
2012\_06\_04\_17\_31\_49-innodbstatus1  
2012\_06\_04\_17\_31\_49-innodbstatus2  
2012\_06\_04\_17\_31\_49-interrupts  
2012\_06\_04\_17\_31\_49-log\_error  
2012\_06\_04\_17\_31\_49-lsof  
2012\_06\_04\_17\_31\_49-meminfo  
2012\_06\_04\_17\_31\_49-mutex-status1  
2012\_06\_04\_17\_31\_49-mysqladmin  
2012\_06\_04\_17\_31\_49-netstat  
2012\_06\_04\_17\_31\_49-netstat\_s  
2012\_06\_04\_17\_31\_49-opentables1  
2012\_06\_04\_17\_31\_49-opentables2  
2012\_06\_04\_17\_31\_49-output  
2012\_06\_04\_17\_31\_49-pmap  
2012\_06\_04\_17\_31\_49-processlist  
2012\_06\_04\_17\_31\_49-procstat  
2012\_06\_04\_17\_31\_49-procvmstat  
2012\_06\_04\_17\_31\_49-ps  
2012\_06\_04\_17\_31\_49-slabinfo  
2012\_06\_04\_17\_31\_49-sysctl  
2012\_06\_04\_17\_31\_49-top  
2012\_06\_04\_17\_31\_49-trigger  
2012\_06\_04\_17\_31\_49-variables  
2012\_06\_04\_17\_31\_49-vmstat  
2012\_06\_04\_17\_31\_49-vmstat-overall  
**(**七**)**实用类工具  
1. pt-archiver  
⚫功能介绍：  
将 mysql 数据库中表的记录归档到另外一个表或者文件，也可以直接  
进行记录的删除操作。  
⚫用法介绍：  
pt-archiver [OPTION...] --source DSN --where WHERE  
这个工具只是归档旧的数据，不会对线上数据的 OLTP 查询造成太大  
影响，你可以将数据插入另外一台服务器的其他表中，也可以写入到  
一个文件中，方便使用 load data infile 命令导入数据。 另外你还可以  
用它来执行 delete 操作。 这个工具默认的会删除源中的数据。使用  
的时候请注意。  
⚫使用示例：  
范例 1：将 192.168.3.135 上的 sanmao 库的 oss\_log 表 id 小于 100000  
的 记 录 转 移 到 192.168.3.92 上 的 sanmao 库 ， 并 归 档 到  
oss\_log\_archive\_20120605.log 文件中:  
pt-archiver --source h=192.168.3.135,D=sanmao,t=oss\_log --user=root  
--password=zhang@123 --dest h=192.168.3.92,D=sanmao,t=oss\_log -  
-file '/var/log/oss\_log\_archive\_20120605.log' --where "id<=100000"  
--commit-each  
范例 2：将 192.168.3.135 上的 sanmao 库的 oss\_log 小于 160000 的记  
录归档到 oss\_log\_archive\_20120607.log 文件中:  
pt-archiver --source h=192.168.3.135,D=sanmao,t=oss\_log --user=root  
--password=zhang@123 --file '/var/log/oss\_log\_archive\_20120607.lo  
g' --where "id<=160000" --commit-each  
范例 3：删除 192.168.3.135 上的 sanmao 库的 oss\_log 表中 id 小于  
167050 的记录：  
pt-archiver --source h=192.168.3.135,D=sanmao,t=oss\_log --user=root  
--password=zhang@123 --purge --where 'id<=167050'  
注意：如果是字符集是 utf8 的话，需要在 my.cnf 中的[client]下面添  
加 default-character-set = utf8，否则导出的文件内容中文会乱码。  
2. pt-find  
⚫功能介绍：  
查找 mysql 表并执行指定的命令，和 gnu 的 find 命令类似。  
⚫用法介绍：  
pt-find [OPTION...] [DATABASE...]  
默认动作是打印数据库名和表名  
⚫使用示例：  
范例 1：查找 192.168.3.135 中 1 天以前创建的 InnoDB 的表 ,并打印。  
pt-find --ctime +1 --host=192.168.3.135 --engine InnoDB --user=roo  
t --password=zhang@123  
范例 2：查找 192.168.3.135 中 1 天以前更改过的数据库名字匹  
配%hostsops%的并且引擎为 MYISAM 的表，并将表的引擎更改为  
InnoDB 引擎。  
pt-find --mtime +1 --dblike hostsops --engine MyISAM --host=192.1  
68.3.135 --user=root --password=zhang@123 --exec "ALTER TABLE  
%D.%N ENGINE=InnoDB"  
范例 3： 查找 192.168.3.135 中 aaa 库和 zhang 库中的空表，并删除。  
pt-find --empty aaa zhang --host=192.168.3.135 --user=root --passw  
ord=zhang@123 --exec-plus "DROP TABLE %s"  
范例 4：查找 192.168.3.135 中超过 100M 的表：  
pt-find --tablesize +100M --host=192.168.3.135 --user=root --passwo  
rd=zhang@123  
3. pt-kill  
⚫功能介绍：  
Kill 掉符合指定条件 mysql 语句  
⚫用法介绍：  
pt-kill [OPTIONS]  
假如没有指定文件的话pt-kill连接到mysql并通过SHOW PROCESSLIST  
找到指定的语句，反之 pt-kill 从包含 SHOW PROCESSLIST 结果的文件  
中读取 mysql 语句  
⚫使用示例：  
范例 1：查找 192.168.3.135 服务器运行时间超过 60s 的语句，并打印  
pt-kill --busy-time 60 --print --host=192.168.3.135 --user=root --pass  
word=zhang@123  
范例 2： 查找 192.168.3.135 服务器运行时间超过 60s 的语句，并 kill  
pt-kill --busy-time 60 --kill --host=192.168.3.135 --user=root --passw  
ord=zhang@123  
范例 3： 从 proccesslist 文件中查找执行时间超过 60s 的语句  
mysql -uroot -pzhang@123 -h192.168.3.135 -e "show processlist" >  
processlist.txt  
pt-kill --test-matching processlist.txt --busy-time 60 --print