

Maatkit/Percona Toolkit en pratique

Meetup LeMug/Viadeo

16 Novembre 2011

Paris

Stéphane Combaudon

stephane.combaudon@dailymotion.com

En bref

- Maatkit : + de 30 scripts Perl pour DBA MySQL
- GPL
- Créé par Baron Schwartz
- En 2011, fork de Percona remplace Maatkit
 - Percona Toolkit
- Développement actif
- Voir <http://www.percona.com/software/percona-toolkit>

Maatkit vs Percona Toolkit

- Certains scripts douteux ont été abandonnés
 - `mk-parallel-dump/restore`, `mk-slave-prefetch`...
- Les scripts ont été renommés
 - `s/mk/pt/g`

Une étrange requête lente - 1

```
CREATE TABLE user (  
  user_id int NOT NULL AUTO_INCREMENT PRIMARY KEY,  
  login varchar(30) NOT NULL DEFAULT '',  
  ...  
  UNIQUE KEY (login)  
) ENGINE=InnoDB;
```

- Affichage de mysqldumpslow :

```
Count: 195 Time=33.05s (6445s) Lock=0.00s (0s) Rows=129903.9 (25331256)  
SELECT user_id FROM user WHERE login = N
```

- L'optimiseur pense que l'index unique n'est pas le meilleur choix ?!

Une étrange requête lente - 2

- Les tests avec des logins réels prennent 0s
- Il nous faut plus d'informations

```
mk-query-digest --limit 10 mysql-slow.log
```

```
# Query 2: 0.00 QPS, 0.07x concurrency, ID 0xEB7F16321805BBED at byte 9841263
# Scores: Apdex = 0.00 [1.0], V/M = 0.08
# Query_time sparkline: |      ^|
# Time range: 2010-12-21 06:06:08 to 2010-12-22 05:20:36
# Attribute      pct   total      min      max      avg      95%   stddev  median
# =====      ==  =====  =====  =====  =====  =====  =====  =====
# Count          1    178
# Exec time      3   5879s    31s     35s     33s     35s     2s      32s
# Lock time      0      0        0        0        0        0        0        0
# Rows sent     99 625.36M    0 26.49M  3.51M  21.31M  7.09M    0
# Rows examine  15  1.71G 775.10k 27.14M  9.81M  22.38M  5.99M   8.43M
# Query size    0  27.42k    84     363   157.72 346.17  66.96  118.34
[...]
# EXPLAIN /*!50100 PARTITIONS*/
SELECT user_id FROM user WHERE login = 0\G
```

Une étrange requête lente - 3

- Tout s'éclaircit, non ?
 - login est un VARCHAR, mais la req. utilise un INT
 - MySQL convertit login en un INT
 - L'index ne peut plus être utilisé
 - Un plan d'exécution bcp plus coûteux est choisi
- Résolution : l'application doit s'assurer que login est toujours une chaîne


Voir des requêtes en direct - 1

- DBA : Le serveur exécute bcp de commandes inutiles qui finissent par coûter cher
- Collègue : Impossible, ces requêtes ne coûtent rien, même si on les exécute tout le temps
- mk-query-digest peut traiter la sortie de tcpdump
- On peut donc avoir un rapport en instantanée des requêtes les plus coûteuses

Voir des requêtes en direct - 2

```
tcpdump -i eth0 port 3306 -s 65535 -c 10000 -x -n -q -tttt | mk-  
query-digest --type tcpdump --report-format=profile --limit=10
```

```
# Profile  
# Rank Query ID          Response time Calls R/Call Apdx V/M  Item  
# =====  
# 1 0x04FE01C5B31FD305  1.5284 46.9%   928 0.0016 1.00 0.05 ADMIN PING  
# 2 0x49D343B8B49A7E19  0.4748 14.6%    99 0.0048 1.00 0.36 SET  
# 3 0x348032401B50B8DE  0.4507 13.8%    30 0.0150 1.00 0.02 ADMIN STMT_PREPARE  
# 4 0x6FA12E46AE6F2713  0.0821  2.5%    49 0.0017 1.00 0.07 SELECT video_view_summary  
# 5 0xB35A9E3E7EC24592  0.0811  2.5%     1 0.0811 1.00 0.00 SELECT user  
# 6 0x5D51E5F01B88B79E  0.0799  2.4%    34 0.0023 1.00 0.05 ADMIN CONNECT  
# 7 0x23923E0C446133EF  0.0791  2.4%   134 0.0006 1.00 0.03 SELECT video_view_summary  
# 8 0x4E4572ED44FF00A3  0.0614  1.9%    18 0.0034 1.00 0.05 SELECT contest_has_video  
# 9 0xDC49DE9F32578E81  0.0588  1.8%    82 0.0007 1.00 0.02 SELECT video  
# 10 0x4DDAADBA9959055B 0.0532  1.6%     2 0.0266 1.00 0.05 SELECT skin  
# MISC 0xMISC          0.3123  9.6%  1467 0.0002  NS   0.0 <90 ITEMS>
```



- Ouch! Plus de 60% du temps est perdu...

Des mauvais backups - 1

- Collègue : Un slave dédié fournit les backups
- DBA : Bien ! Tu es sûr que le slave a les mêmes données que le master ?
- Collègue : ??? On a une alerte Nagios si la réplication est cassée
- DBA : Et si la réplication déraile sans erreur ?
- Collègue : ?????? La réplication n'est-elle pas censée fonctionner ?

Des mauvais backups - 2

- mk-table-checksum fournit des contrôles de cohérences des données entre un master et ses slaves
- Concept
 - Exécuter des checksums sur le master et les laisser parvenir aux slaves par la réplication
 - Comparer les checksums quand les slaves sont à jour

Des mauvais backups - 3

- Sur le master, exécution des checksums :

```
mk-table-checksum --chunk-size=1000 --replicate=mydb.chk masterhost
```

- Quand les slave ont rattrapé le master :

```
mk-table-checksum --chunk-size=1000 --replicate=mydb.chk \  
--replicate-check=1 masterhost
```

Differences on P=3306,h=10.8.3.4

DB	TBL	CHUNK	CNT_DIFF	CRC_DIFF	BOUNDARIES
mydb	search	1	-42270	1	`search_query_id` > 0 AND `search_query_id` < '2975138'
mydb	search	2	-2417	1	`search_query_id` >= '2975138' AND `search_query_id` < '5950259'
mydb	search	3	-3236	1	`search_query_id` >= '5950259' AND `search_query_id` < '8925380'
mydb	search	4	-1869	1	`search_query_id` >= '8925380' AND `search_query_id` < '11900501'
mydb	search	5	-877	1	`search_query_id` >= '11900501' AND `search_query_id` < '14875622'
mydb	search	6	-718	1	`search_query_id` >= '14875622' AND `search_query_id` < '17850743'
mydb	search	7	-611	1	`search_query_id` >= '17850743' AND `search_query_id` < '20825864'
mydb	search	8	-551	1	`search_query_id` >= '20825864' AND `search_query_id` < '23800985'

- Oups ! Maintenant, il faut resynchroniser !

Resynchroniser des tables – 1

- Collègue : J'ai dû sauter des événements sur une table, comment je peux la resynchroniser ?
- mk-table-sync peut utiliser les informations de mk-table-checksum
- Il génère des requêtes pour résoudre les différences
- !! Les requêtes doivent toujours être jouées sur le master !!
 - Risque de corruption de données sinon

Resynchroniser des tables – 2

```
mk-table-sync --replicate=mydb.chk --print --sync-to-master \  
slavehost
```

```
REPLACE INTO `mydb`.`search_data_complete` (`search_query_id`, `search_query`, `lan  
g`, `channel_id`, `hit_count`, `result_count`, `first_video_id`, `last_seen`) VALUES ('60  
09', 'hikaru no go special', 0x6A70, '8', '15957', '2440', '6689889', '2011-02-03') /*m  
aatkit src_db: mydb src_dsn:P=3306,h=10.33.1.0,p=...,u=m  
ktablesnc dst_db: mydb dst_dsn:h=10.8.3.4,p=...,u=mkt  
ablesync lock:1 transaction:1 changing_src:dba_checks.checksum replicate:dba_checks.check  
sum bidirectional:0 pid:31925 user:stc host:logstore-04*/;  
REPLACE INTO `mydb`.`search_data_complete` (`search_query_id`, `search_query`, `lan  
g`, `channel_id`, `hit_count`, `result_count`, `first_video_id`, `last_seen`) VALUES ('60  
18', 'clannad after story 21', 0x6A70, '8', '16796', '547', '15032133', '2011-02-03') /*m  
aatkit src_db: mydb src_dsn:P=3306,h=10.33.1.0,p=...,u  
=mktablesync dst_db: mydb dst_dsn:h=10.8.3.4,p=...,u=m  
ktablesync lock:1 transaction:1 changing_src:dba_checks.checksum replicate:dba_checks.che  
cksum bidirectional:0 pid:31925 user:stc host:logstore-04*/;
```

- Utiliser `--execute` au lieu de `--print` pour l'exécution
- L'outil est **très** puissant, il doit être manié avec de grandes précautions!

Mise à jour de grosses tables - 1

- Collègue : Lors des dernières mises à jour massives sur une grosse table, le script a tourné pendant 5 jours. Peut-on trouver mieux ?
- mk-archiver est optimisé pour parcourir de manière efficace des tables volumineuses pour purger ou archiver des données
- On peut étendre l'outil avec des plugins pour exécuter n'importe quel type de tâche

Mise à jour de grosses tables – 2

```
mk-archiver --source D=mydb,t=orig --dest D=mydb,t=dest,m=my_plug \  
--where "... " --no-delete --commit-each --limit=100000 \  
--progress=100000 --bulk-insert
```

TIME	ELAPSED	COUNT
2011-02-03T15:35:05	0	0
2011-02-03T15:35:10	5	100000
2011-02-03T15:35:30	25	200000
2011-02-03T15:35:53	48	300000
2011-02-03T15:36:14	69	400000
2011-02-03T15:36:35	90	500000
2011-02-03T15:36:54	109	600000
2011-02-03T15:37:15	130	700000
2011-02-03T15:37:35	149	800000
2011-02-03T15:37:54	169	900000
2011-02-03T15:38:13	188	1000000
2011-02-03T15:38:32	207	1100000
2011-02-03T15:38:51	226	1200000
2011-02-03T15:39:10	245	1300000
2011-02-03T15:39:29	264	1400000
2011-02-03T15:39:49	284	1500000

- 20s pour mettre à jour chaque morceau
- Temps de traitement stable
- La table a 20M lignes
- Travail terminé en 4000s, un peu plus d'une heure !

Serial killer

- Collègue : Est-il possible de tuer automatiquement les requêtes très lentes ?

```
mk-kill --busy-time=10 --kill --heartbeat --iterations 0 --print
```

```
# 2011-02-02T13:57:34 Checking processlist, iteration 0
# 2011-02-02T13:57:34 Matched 0 queries
# 2011-02-02T13:57:39 Checking processlist, iteration 0
# 2011-02-02T13:57:39 Matched 0 queries
# 2011-02-02T13:57:44 Checking processlist, iteration 0
# 2011-02-02T13:57:44 Matched 0 queries
# 2011-02-02T13:57:49 Checking processlist, iteration 0
# 2011-02-02T13:57:49 Matched 1 queries
# 2011-02-02T13:57:49 KILL 88 (Query 13 sec) select ...
# 2011-02-02T13:57:49 Killed 88
# 2011-02-02T13:57:54 Checking processlist, iteration 0
# 2011-02-02T13:57:54 Matched 0 queries
```

- Attention aux options qui ont changé entre les versions !

Des questions ?