MySQL Query Tuning

Olivier DASINI
@freshdaz
http://dasini.net/blog/

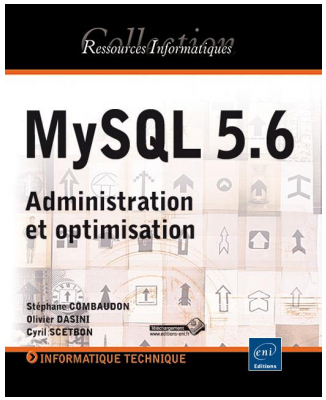
Méthodologie pour améliorer les performances d'une application
AFUP - Paris le 28 février 2013

Me, Myself & I

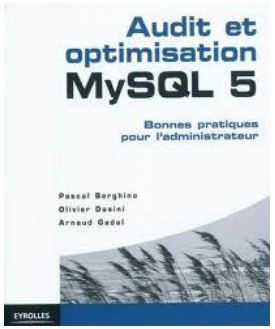
- Olivier DASINI
  - Expert MySQL chez Viadeo
  - Basically a MySQL geek
  - Follow me
    - @freshdaz
  - Read me
    - http://dasini.net/blog/
  - Support us
    - Co-fondateur du **MySQL User Group Francophone** (LeMug.fr)
    - http://lemug.fr
    - Facebook / Linkedin / Google + / Viadeo

Ma vie, mon oeuvre

- Co-auteur :
  - **MySQL 5.6 - Administration et optimisation**
    - ENI, ISBN: 978-2-7460-7864-2



  - **Audit et optimisation - MySQL 5, Bonnes pratiques pour l'administrateur**
    - Eyrolles, ISBN-13: 978-2212126341



  - **MySQL 5 - Administration et optimisation**
    - ENI, ISBN-13: 978-2-7460-5516-2

MySQL Query Tuning - Plan

1/ Récupérer l'information
2/ Extraire les requêtes problématiques
3/ Confirmer
4/ Optimiser
    4.1) L'analyse
    4.2) L'optimisation
5/ Tester

MySQL Query Tuning

Pourquoi optimiser ?

- Pour améliorer l'expérience utilisateur
  - application plus rapide, fluide, plus souvent accessible,...

- Pour gagner (plus) de l'argent => le vrai but
  - en économisant (hard/software, licence, ressources,...)
  - en gagnant de nouveaux clients, nouvelles parts de marché

Comment ? (vision DB)

- Réduire le temps de réponse des requêtes
- Augmenter le nombre de requêtes simultanées (débit)

Mais comment ??

- Réduire les I/O et notamment les I/O disques

Disclaimer

Cette présentation ne traite pas :

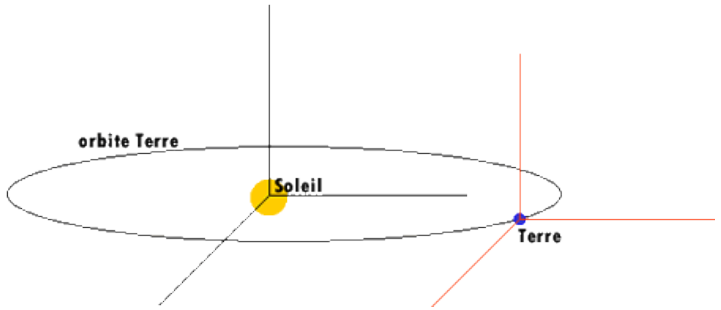
- de l'optimisation niveau hardware
- de l'optimisation du serveur MySQL
- d'Harlem shake

MySQL Query Tuning

Récupérer

Récupérer l'information

Définir un référentiel



Temporel

- Définir une période de temps significative:
  - De façon arbitraire : la journée, ...
  - Par rapport à un contexte :
    - périodes de charge
    - après une MEP, ...

Fonctionnel

- Fonctionnalité de l'application qui pose des problèmes de performance

Récupérer l'information - slow log

Choisir une méthode

Coté MySQL: log de MySQL

- slow query log
  - (Dés)Activable à chaud
  - Contient les informations pertinentes
    - temps d'exécution,
    - durée du verrou, ...
  - Comme si il avait été créé pour ça :)

Récupérer l'information - slow log

Choisir une méthode

```
# head -n 50 mysql-slow.log

/usr/sbin/mysqld, Version: 5.5.24-55-log Percona Server started with:
Tcp port: 3306  Unix socket: /var/run/mysqld/mysqld.sock
Time                 Id Command    Argument
# Time: 130225  7:50:14
# User@Host: my_user[my_user] @  [110.01.22.1]
# Thread_id: 212067  Schema: my_db  Last_errno: 0  Killed: 0
# Query_time: 5.315628  Lock_time: 0.001057  Rows_sent: 1889
Rows_examined: 2514  Rows_affected: 0  Rows_read: 1889
# Bytes_sent: 26417  Tmp_tables: 0  Tmp_disk_tables: 0  Tmp_table_sizes: 0
# InnoDB_trx_id: 3B5830A2
# QC_Hit: No  Full_scan: No  Full_join: No  Tmp_table: No
Tmp_table_on_disk: No
# Filesort: No  Filesort_on_disk: No  Merge_passes: 0
use my_db;
SET timestamp=1361775014;
SELECT col1 FROM My_table WHERE .. .
```

Récupérer l'information

Choisir une méthode

Outils tiers

- **Anemometer** : interface web pour pt-query-digest
  - ...

- **AppDynamics** : client lourd java de monitoring. Pas open source.
  - ...

Récupérer l'information - Anemometer

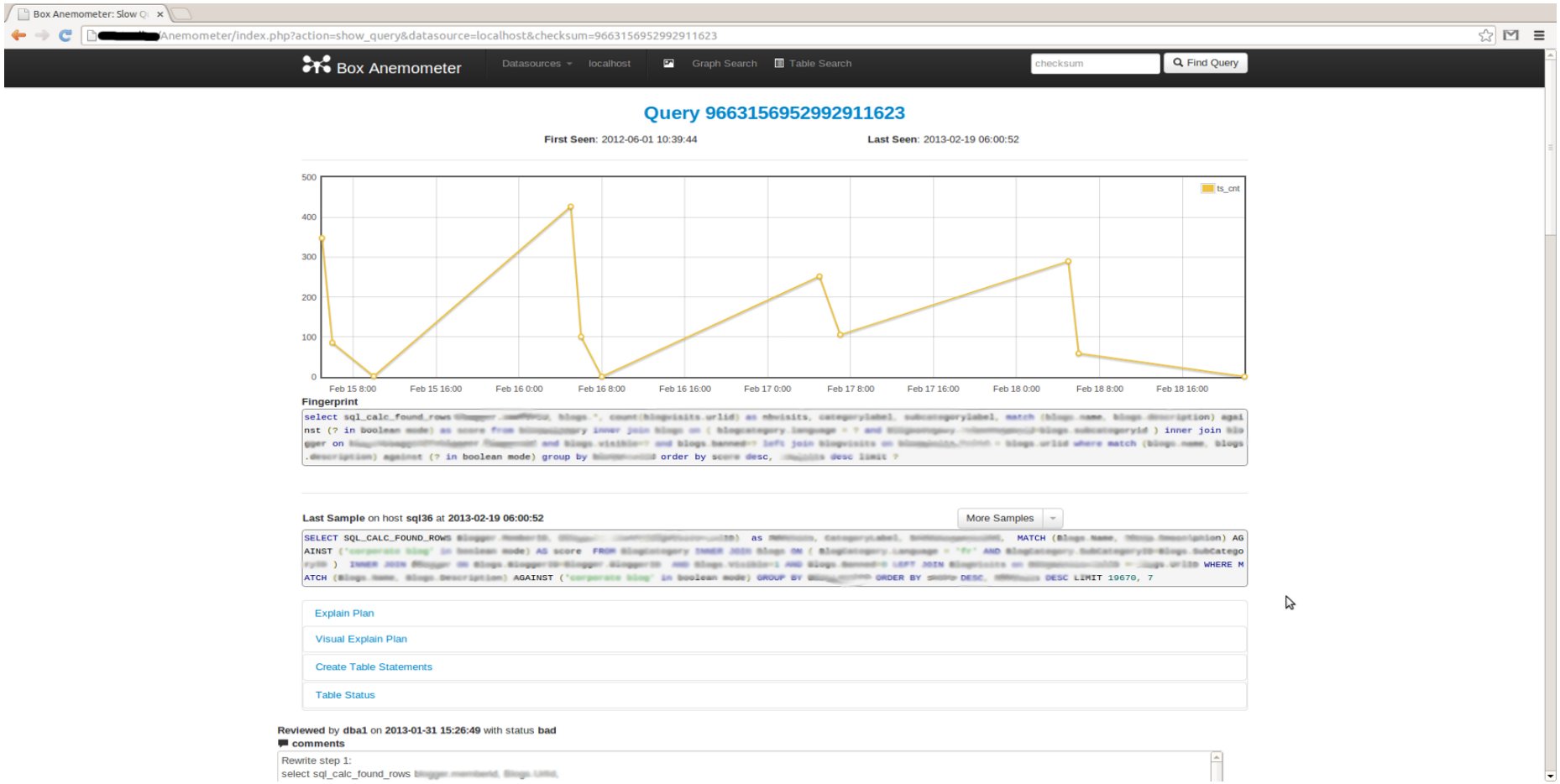
Utiliser les bons outils

Anemometer is a tool for visualizing collected data from the MySQL Slow Query Log.

Anemometer relies on the Percona Toolkit to do the slow query log collection. Specifically you can run pt-query-digest. To parse your slow logs and insert them into a database for reporting.

https://github.com/box/Anemometer/wiki

Récupérer l'information - Anemometer



Récupérer l'information - AppDynamics

Utiliser les bons outils

AppDynamics is an Application Performance Management (APM) solution that monitors, troubleshoots, and diagnoses problems in mission-critical apps.

www.appdynamics.com/

Récupérer l'information - AppDynamics

Récupérer l'information

Choisir une méthode

- **Coté application: logs applicatif**
  - Journalisation des requêtes générée par l'application
  - Permet de ne récupérer que les informations intéressantes
  - Dev maison ou a activer au niveau du framework
  - Pas l'idéal si vous avez de nombreuses applications / équipes différentes

- Outils de monitoring
  - **mytop** : requêtes tournant en live sur le serveur (clone de top)
  - Permet une vision instantanée de la base
  - Peut donner une impression erronée de la cause du problème

- Autres
  - TCP/IP (avec pt-query-digest)
  - Proxy
  - ...

Récupérer l'information

Utiliser les bons outils

Percona Toolkit is a collection of advanced command-line tools used by Percona (http://www.percona.com/) support staff to perform a variety of MySQL and system tasks that are too difficult or complex to perform manually.

http://www.percona.com/doc/percona-toolkit/

pt-query-digest : Analyze query execution logs and generate a query report, filter, replay, or transform queries for MySQL, PostgreSQL, memcached, and more.

http://www.percona.com/doc/percona-toolkit/2.1/pt-query-digest.html

Récupérer l'information

```
# pt-query-digest /var/log/mysql/mysql-slow.log

# 350ms user time, 30ms system time, 21.37M rss, 167.26M vsz
...
# Overall: 232 total, 43 unique, 0.01 QPS, 0.06x concurrency _____________
# Time range: 2013-02-19 06:58:20 to 18:18:57
# Attribute          total     min  max  avg  95%  stddev  median
# ============       ======= ======= ======= ======= ======= ======= =======
# Exec time          2481s     5s   198s      11s  20s  15s       7s
# Lock time          40ms      0    1ms   171us   541us   161us   125us
# Rows sent          1006.66k       0 105.47k   4.34k  34.83k  13.05k  0.99
# Rows examine       67.39M    0  13.93M 297.45k   1.32M   1.00M   49.69
# Bytes sent         100.33M        0  76.45M 442.84k 590.13k   4.98M  964.41
# Merge passes             2    0    1    0.01       0    0.09       0
# Tmp tables         217       0    4    0.94 3.89 0.83 0.99
# Tmp disk tbl       165       0    4    0.71 3.89 0.91 0.99
# Tmp tbl size       1.91G     0 136.15M   8.42M  48.85M  24.42M        0
# Query size         574.19k        6  28.98k   2.47k  11.34k   4.58k  793.42
```

Récupérer l'information

```
# Boolean:
# Filesort     83% yes,  16% no
# Filesort on  0% yes,  99% no
# Full scan    7% yes,  92% no
# Tmp table    76% yes,  23% no
# Tmp table on  54% yes,  45% no

# Profile
# Rank Query ID           Response time  Calls R/Call   Apdx V/M   Item
# ==== ================== ============== ===== ======== ==== ===== =======
#    1 0x0E0BED09B4BC25CE 652.1544 26.3%     90   7.2462 0.00  0.28 SELECT T1 T2 T3 T4
T5
#    2 0xF3611032470604C9 368.4436 14.9%     39   9.4473 0.00  4.41 SELECT T6
#    3 0xDBC752A91C8B5962 234.5549  9.5%     13  18.0427 0.00 22.25 SELECT UNION T7 T8
...
#    9 0x3C71B33274536DA1  60.8141  2.5%     7   8.6877 0.00  1.94 INSERT T9
#   10 0x76BDB9540E306749  52.1852  2.1%     4  13.0463 0.00  7.89 UPDATE T10
```

MySQL Query Tuning

Extraire

Extraire les requêtes problématiques

Stratégies

Requête la plus lente en valeur absolue

- temps d'exécution supérieur à la limite acceptable (>= *long_query_time*)
- attention aux requêtes qui ne *peuvent pas être rapide* (batch)
  - Tips: tagger les requêtes eg SELECT /* B_DELETE_CONTACT */ …

- Anemometer et pt-query-digest font l'affaire

Extraire les requêtes problématiques

Stratégies

Requête la plus lente en temps cumulé

- pas nécessairement la requête la plus lente (parmi toutes les autres)
- mais elle est exécutée un grand nombre fois
- souvent un grand impact sur les perfs de l’appli

- Anemometer et pt-query-digest font l'affaire

Extraire les requêtes problématiques

Stratégies

Requête appartenant à une fonctionnalité qui pause des problèmes de perf

- permet de se concentrer sur un sous ensemble de code / tables / requêtes
  - donne plus de souplesse pour une modification de code / schéma
  - résultat facilement visible
- Les requêtes ne sont pas nécessairement dans le top des requêtes lentes

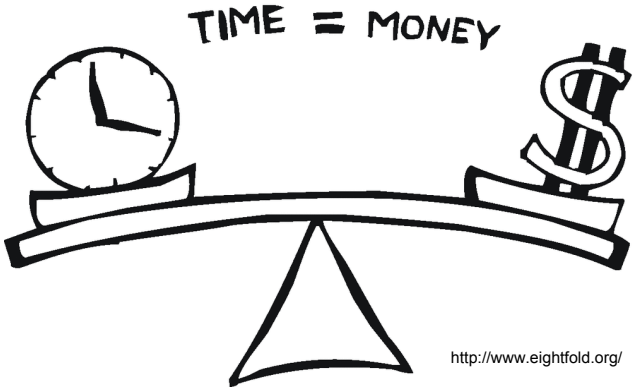
- AppDynamics et pt-query-digest font l'affaire

MySQL Query Tuning

Confirmer

Confirmer

Time is money, don't waste it !



Investiguer pour savoir si:

- Le problème est répétable ?
- La requête sélectionnée est bien la cause (et non la conséquence)

La lenteur peut être générée par:

- un problème de load, CPU, disque, lock...
- un batch/cron applicatif ou système,...
- un maillon de la chaîne applicative défaillant:
  - (mem)cache qui est tombé
  - un bug
  - ...

MySQL Query Tuning

Optimiser

MySQL Query Tuning

Optimiser

zoom sur

L'optimiseur

Optimiser - zoom sur l'optimiseur

Comment fonctionne l'optimiseur

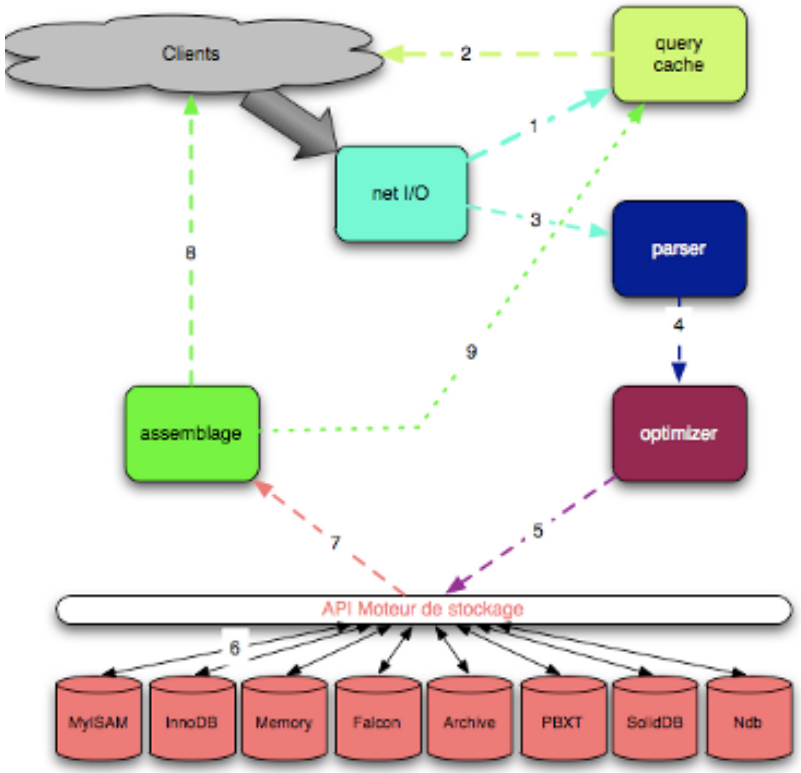


Image tirée du livre "**Audit et optimisation MySQL 5**" aux éditions Eyrolles - p. 50

Optimiser - zoom sur l'optimiseur

Comment fonctionne l'optimiseur

La bête se nourrit (entre autre) :

- d'informations sur les index
  - # valeurs NULL
  - # valeurs uniques
  - cardinalité
  - taille des index
  - ...
- d'informations sur les données
  - # d'enregistrements
  - taille des données
  - ...

- L'optimiseur de MySQL est à base de coût
- Il utilise ces infos pour calculer et choisir le QEP dont le coût est le plus faible

- L'unité du coût:
  - la lecture aléatoire d'une page de 4 ko

Les principaux critères qui influencent le coût :

- Au niveau I/O
  - # pages lu dans la table
  - # pages lu de l'index
- Au niveau CPU
  - coût de l'évaluation de la condition de la requête
  - coût de la comparaison des index et/ou des rows

Optimiser - zoom sur l'optimiseur

Impacts

syndrome du "*Mais ça fonctionne sur ma machine !*"

- En clair, les perfs d'une requête sont sensibles à (aux):
  - la volumétrie
  - la charge
  - la taille des colonnes
  - la valeur des données
  - la présence ou non d'index
  - l'algorithme de l'index
  - contraintes de l'index
  - l'ordre des colonnes dans l'index
  - moteurs de stockage
  - ...

Optimiser - zoom sur l'optimiseur

Impacts

En encore plus clair (ou pas) :

- le dev, ce n'est pas la prod :(
- l'integ, ce n'est pas la prod :(
- la preprod, ce n'est pas la prod :(
- la prod, c'est la prod :). Oui mais... c'est la prod :'(

- Il est impossible de prévenir tous les problèmes en amont
- Une requête problématique peut facilement se révéler en prod

Pour minimiser les risques, il faut

- baser ses choix sur des critères objectifs
  - connaître les règles de bases
  - comprendre le fonctionnement de MySQL
  - utiliser les commandes adéquates

Optimiser - zoom sur l'optimiseur

Ex. différence de temps d'exécutions de 2 requêtes

L'écart de perf entre 1 bonne requête et une mauvaise dépend de la volumétrie

En dev

```
SELECT * FROM Post WHERE k=1;
1 row in set (0.00 sec)
```

avec utilisation de l'index

```
SELECT * FROM Post IGNORE INDEX(k) WHERE k=1;
1 row in set (0.00 sec)
```

sans utilisation de l'index

En prod

```
SELECT * FROM Post WHERE k=1;
1 row in set (7.41 sec)
```

avec utilisation de l'index

```
SELECT * FROM Post IGNORE INDEX(k) WHERE k=1;
1 row in set (18.84 sec)
```

sans utilisation de l'index

MySQL Query Tuning

Optimiser

L'analyse

Optimiser - L'analyse

L'analyse

MySQL dispose des commandes suivantes :

- SHOW [CREATE TABLE / INDEX FROM / TABLE STATUS]
  - ces infos sont dans *information_schema*
- EXPLAIN ***<=> tu connais pas, tu touches pas à la BD !!!***
  - Infos sur le plan d'exécution d'une requête
  - http://dev.mysql.com/doc/refman/5.5/en/explain-output.html
- SHOW PROFILE
  - profilage de requêtes
  - déprécié en 5.6 (remplacé par *performance_schema*)
- Autres (Percona toolkit,...)

Optimiser - L'analyse

SHOW CREATE TABLE

Donne le code sql d'une table

```
CREATE TABLE City (
  CityID int(11) NOT NULL AUTO_INCREMENT,
  Name char(255) NOT NULL DEFAULT '',
  Population int(11) DEFAULT '0',
  PRIMARY KEY (CityID),
  KEY name (name)
) ENGINE=MyISAM
```

CityID INT prend il des valeurs négatives ? => INT UNSIGNED
Name char(255) ? => trop grand char(50)
population INT pour une ville ? => Probable (+ de 15 ville de 16M)
ENGINE=MyISAM ? => InnoDB

Les réponses dépendent du contexte métier

Optimiser - L'analyse

SHOW INDEX FROM

Donne les statistiques des index d'une table

```
******* 1. row *******
     Table: City
   Non_unique: 0
     Key_name: PRIMARY
 Seq_in_index: 1
  Column_name: ID
  Cardinality: 4051
 ...
   Index_type: BTREE
******* 2. row *******
     Table: City
   Non_unique: 1
     Cardinality: 450
...
```

Optimiser - L'analyse

SHOW TABLE STATUS

Donne l'état d'une table

```
           Name: offres
         Engine: InnoDB
        Version: 10
     Row_format: Compact
           Rows: 21521475
 Avg_row_length: 638
    Data_length: 13734248448
Max_data_length: 0
   Index_length: 7987609600
      Data_free: 4194304
 Auto_increment: 20590724
    Create_time: 2012-09-04 21:44:22
    Update_time: NULL
     Check_time: NULL
      Collation: utf8_swedish_ci
```

Optimiser - L'analyse

EXPLAIN

Donnes les infos sur le plan d'exécution d'une requête (SELECT seulement avant 5.6)

```
           id: 1
  select_type: SIMPLE
        table: email
         type: ref
possible_keys: EmailSource
          key: EmailSource
      key_len: 771
          ref: const,const
         rows: 1
        Extra: Using where
```

/!\ A savoir utiliser impérativement !

Optimiser - L'analyse

SHOW PROFILE

Permet de profiler une requête

```
+------------------------------+----------+
| Status | Duration |
+------------------------------+----------+
| starting | 0.000150 |
| checking permissions | 0.000017 |
| Opening tables | 0.000042 |
| System lock | 0.000021 |
| init | 0.000036 |
| optimizing | 0.000010 |
| statistics | 0.000035 |
| Copying to tmp table | 0.107526 |
| converting HEAP to MyISAM | 0.000205 |
| Copying to tmp table on disk | 0.014498 |
| Sorting result | 0.003098 |
...
```

MySQL Query Tuning

Optimiser

L'optimisation

Optimiser - différentes possibilités

Optimisation

- Modifier le schéma
  - ajout / suppression d'index (covering index, suppression du tri/table temporaire,...)
  - ajout / suppression de colonnes (covering index, colonne calculée,...)
  - changer le moteur de stockage (verrous, scalabilité, perfs,...)

- Réécrire la requête
  - l'améliorer
  - la simplifier
  - la découper

- Purger les données
  - supprimer les données inutiles
  - archiver les données les moins utiles

- Supprimer la requête
  - Legacy code is sometime devilish
- ...

Optimiser - le schéma

Ex. ajout d'index pertinents

```
SELECT * FROM Unsubscr WHERE Email IN ('olivier@dasini.net') and Source = 1
```

```
ALTER TABLE IronUnsubscribe ADD KEY EmailSource(Email,Source);
```

Avant : (**13.31 sec**)
Avant : (**1.99 sec**)

Après : (**0.00 sec**)

Avant

```
           id: 1
  select_type: SIMPLE
        table: Unsubscr
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 5018257
        Extra: Using where
```

Après

```
           id: 1
  select_type: SIMPLE
        table: Unsubscr
         type: ref
possible_keys: EmailSource
          key: EmailSource
      key_len: 771
          ref: const,const
         rows: 1
        Extra: Using where
```

Optimiser - réécriture ex1: 1/3

Ex. Attention aux fonctions

```
CREATE TABLE Cust (
 CustID int(11) NOT NULL AUTO_INCREMENT,
 Phone varchar(16) DEFAULT NULL,
 Since timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
 PRIMARY KEY (CustID),KEY idx_Since (Since)
) ENGINE=InnoDB
```

```
SELECT CustID,Phone FROM Cust WHERE  DATE_FORMAT(Since,'%d%m%y')=DATE_FORMAT
(NOW(),'%d%m%y')
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: Cust
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 701514
        Extra: Using where

Minimum number of seconds to run all queries:  0.554 seconds
```

L'utilisation d'une fonction sur une colonne indexée **empêche l'utilisation de l'index** => FTS

Optimiser - réécriture ex1: 2/3

Ex. Attention aux fonctions

```
SELECT CustID,Phone FROM Cust WHERE  DATE_FORMAT(Since,'%d%m%y')=DATE_FORMAT
(NOW(),'%d%m%y')  AND Since >= CURDATE()

*************************** 1. row ***************************
         id: 1
select_type: SIMPLE
     table: Cust
     type: range
possible_keys: idx_Since
     key: idx_Since
     key_len: 4
     ref: NULL
     rows: 1108
     Extra: Using where

Minimum number of seconds to run all queries:  0.002 seconds
```

Ajout (AND) d'un filtre sur la colonne indexée (since)
L'index peut être utilisé

Optimiser - réécriture ex1: 3/3

Ex. Attention aux fonctions

```
SELECT CustID,Phone FROM Cust
WHERE Since BETWEEN '2013-02-26 00:00:00' AND '2013-02-26 23:59:59'
*************************** 1. row ***************************
         id: 1
select_type: SIMPLE
     table: Cust
     type: range
possible_keys: idx_Since
     key: idx_Since
     key_len: 4
     ref: NULL
     rows: 1107
     Extra: Using where

Minimum number of seconds to run all queries:  0.002 seconds
```

La clause WHERE est réécrite afin de supprimer la fonction
On se passe également de la fonction NOW().

Optimiser - réécriture ex2: 1/3

Ex2. Attention aux fonctions

Faire une recherche *datetime* mais avec 1 colonne *date* et 1 colonne *time*

```
CREATE TABLE Cust (
 CustID int(11) NOT NULL AUTO_INCREMENT,
 phone varchar(16) DEFAULT NULL,
 dateCol date NOT NULL default '2013-01-01',
 timeCol time NOT NULL default '00:00:00',
...
 PRIMARY KEY (CustID),
 KEY dateCol (dateCol),
 KEY timeCol (timeCol)
) ENGINE=InnoDB
```

Optimiser - réécriture ex2: 2/3

Ex2. Attention aux fonctions

```
SELECT CustID,phone FROM Cust WHERE
ADDTIME(dateCol, timeCol) > DATE_ADD(NOW(), INTERVAL -1 DAY)
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: Cust
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 644162
        Extra: Using where

Minimum number of seconds to run all queries:  0.736 seconds
```

la fonction ADDTIME() empêche l'utilisation de l'index

Optimiser - réécriture ex2: 3/3

Ex2. Attention aux fonctions

```
SELECT CustID,phone FROM Cust WHERE
ADDTIME(dateCol, timeCol) > DATE_ADD(NOW(), INTERVAL -1 DAY)
AND dateCol >= DATE(DATE_ADD(NOW(), INTERVAL -1 DAY))
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: Cust
         type: range
possible_keys: dateCol
          key: dateCol
      key_len: 3
          ref: NULL
         rows: 2106
        Extra: Using where

Minimum number of seconds to run all queries:  0.005 seconds
```

Ajout (AND) d'un filtre sur la colonne indexée (dateCol)
L'index peut être utilisé

Autre choix, avoir une colonne *datetime* indexée :)

Optimiser - le schéma ex3

Ex3. récupérer seulement les colonnes nécessaires

```
CREATE TABLE sbtest (
 id int(10) unsigned NOT NULL AUTO_INCREMENT,
 k int(10) unsigned NOT NULL DEFAULT '0',
 c char(120) NOT NULL DEFAULT '', pad char(60) NOT NULL DEFAULT '',
 PRIMARY KEY (id),
 KEY k (k)
) ENGINE=InnoDB

SELECT * FROM sbtest WHERE k=1;
    Minimum number of seconds to run all queries:  12.256 seconds

SELECT id, c FROM sbtest WHERE k=1;
    Minimum number of seconds to run all queries:  10.963 seconds

SELECT id FROM sbtest WHERE k=1;"
    Minimum number of seconds to run all queries:  0.810 seconds
```

covering index in da place !

Optimiser - le schéma ex4: 1/4

Ex4. Covering index

```
 CREATE TABLE Cust (
  CustID int(11) NOT NULL AUTO_INCREMENT,
  City varchar(20) DEFAULT NULL,
  Since timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
   ... 7 colonnes de plus ...,
  PRIMARY KEY (CustID), KEY City (City) ) ENGINE=InnoDB

SELECT * FROM Cust WHERE City='Paris';
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: Cust
         type: ref
possible_keys: City
          key: City
      key_len: 63
          ref: const
         rows: 185572
        Extra: Using where
```

Minimum number of seconds to run all queries: **0.955 seconds**

Optimiser - le schéma ex4: 2/4

Ex4. Covering index

```
SELECT Since FROM Cust WHERE City='Paris';
*************************** 1. row ***************************
         id: 1
  select_type: SIMPLE
      table: Cust
      type: ref
possible_keys: City
          key: City
      key_len: 63
          ref: const
      rows: 185572
      Extra: Using where
```

Minimum number of seconds to run all queries: **0.684 seconds**

Optimiser - le schéma ex4: 3/4

Ex4. Covering index

```
ALTER TABLE Cust ADD KEY CitySince(City,Since);

SELECT Since FROM Cust WHERE City='Paris';
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: Cust
         type: ref
possible_keys: City,CitySince
          key: CitySince
      key_len: 63
          ref: const
         rows: 185754
        Extra: Using where; Using index

Minimum number of seconds to run all queries:  0.087 seconds
```

Optimiser - le schéma ex4: 4/4

Ex4. Covering index

```
SELECT CustID, Since FROM Cust WHERE City='Paris';
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: Cust
         type: ref
possible_keys: City,CitySince
          key: CitySince
      key_len: 63
          ref: const
         rows: 185754
        Extra: Using where; Using index

Minimum number of seconds to run all queries:  0.096 seconds
```

Avec InnoDB, la clé primaire fait "automatiquement" partie de l'index covering ie pas besoin de créer l'index sur les colonnes (City, Since, CustID)

MySQL Query Tuning

Tester

Tester

Le moment de vérité

- L'endroit idéal pour tester est... la prod :D

- La preprod est intéressante pour:
  - Tester le nouveau QEP (EXPLAIN)
  - Pour un quick bench (mysqlslap)
  - mysqlslap :
    - a diagnostic program designed to emulate client load for a MySQL server and to report the timing of each stage. It works as if multiple clients are accessing the server.
    - http://dev.mysql.com/doc/refman/5.5/en/mysqlslap.html

Tester

Le moment de vérité

Va quand même falloir passer en prod un jour !

- la réplication est top pour les changement avec 1 gros impact
- avoir plusieurs slaves permet de (in)valider simplement et de façon (quasi) transparente les changements (l'un après l'autre)
- le sort du master est moins simple à gérer
  - faut il lui passer les changements aussi ?
  - switch de master avec MHA
    - http://dasini.net/blog/presentations/?#mha_viadeo

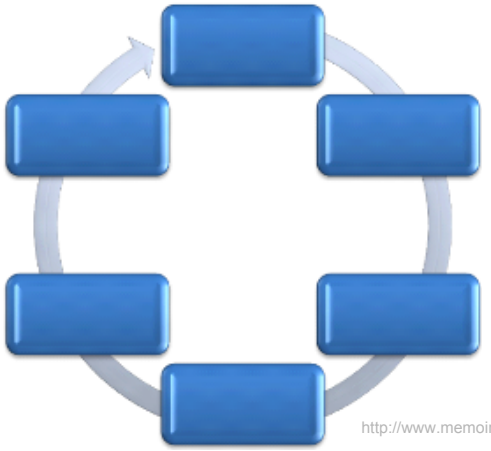
- Percona toolkit: pt-online-schema-change
  - https://www.percona.com/doc/percona-toolkit/2.1/pt-online-schema-change.html

Tester

Le moment de vérité

C’est terminé... pour l’instant

- la base est vivante
- le code est vivant



L’optimisation est un processus itératif

eh ouais baby !!!

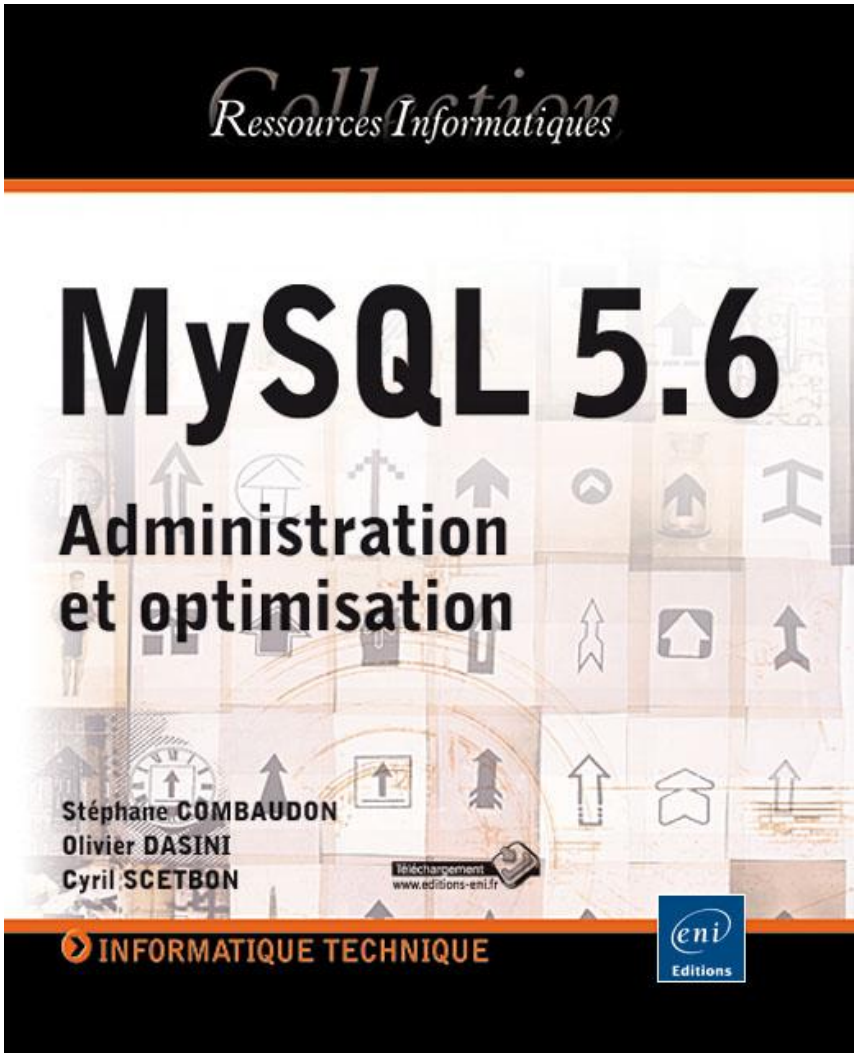
Questions ?

Envie de lire ?

@freshdaz

http://dasini.net/blog/

olivier@dasini.net



Merci !