mysql性能优化

1. 为查询缓存优化你的查询

eg :

```
//查询缓存不开启
$r = mysql_query("select username from user where signup_date >= CURDATE()");
//查询缓存开启
$today = date("Y-m-d");
$r = mysql_query("select username from user where signup_date >= '$today'");
```

MySQL的查询缓存对函数 CURDATE()不起作用。so，像NOW()和RAND()或其他的SQL函数都不会开启查询缓存，因为这些函数的返回是会不定的易变的。so，需要用一个变量来代替MySQL的函数来开启缓存。

2. EXPLAIN你的SELECT查询

EXPLAIN让你知道MySQL是如何处理你的SQL语句的，你的索引主键是如何被利用的，你的数据表是如何被搜索和排序的。。等等

查看rows列可以让我们找到潜在的性能问题。

3. 当只要一行数据时使用LIMIT 1

eg :

```
//效率低
$r = mysql_query("select * from user where country = 'China'");
if (mysql_num_rows($r) > 0) {
...
}
//效率高
$r = mysql_q uery("select 1 from user where country = 'China' LIMIT 1");
if (mysql_num_rows($r) > 0) {
...
}
```

4. 为搜索字段建索引

索引不一定就是给主键或唯一的字段。如果在你的表中，有某个字段你总要用来做搜索，那么请为其建立索引吧。

eg :

```
select count(*) from users where last_name like 'a%' /*sql_no_cache*/;
alter table users add index ('last_name');
select count(*) from users where last_name like 'a%' /*sql_no_cache*/;
```

比较以上两个SELECT语句的执行时间

5. 在JOIN表的时候使用相当类型的列，并将其索引

如果你的应用程序有很多JOIN查询，你应该确认两个表中JOIN的字段是被建过索引的。这样MySQL内部会启动为你优化JOIN的SQL语句的机制。

而且，这些被用来JOIN的字段应该是相同的类型。例如：如果你要把DECIMAL字段和一个INT字段JOIN在以前，MySQL就无法使用它们的索引。对于STRING类型，还需要有相同的字符集。（两个表的字符集有可能不一样）

eg :

```
$r = mysql_query("select company_name from users left join companies on (users.state = companies.state) where users.id = $user_id");
//两个STATE字段应该是被建过索引的，而且应该是相当的类型，相同的字符集。
```

6. 千万不要ORDER BY RAND()

想打乱返回的数据行?随机挑一个数据?真不知道谁发明了这种用法，但很多新手很喜欢这样用。但你确不了解这样做有多么可怕的性能问题。

如果你真的想把返回的数据行打乱了，你有N种方法可以达到这个目的。这样使用只让你的数据库的性能呈指数级的下降。这里的问题是：MySQL会不得 不去执行RAND()函数(很耗CPU时间)，而且这是为了每一行记录去记行，然后再对其排序。就算是你用了Limit 1也无济于事(因为要排序)

eg :

```
//千万不要这样做
$r = mysql_query("select username from user order by RAND() LIMIT 1");
//这样会更好
$r = mysql_query("select count(*) from user");
$d = mysql_fetch_row($r);
$rand = mt_rand(0, $d[0] - 1);
```

```
$r = mysql_query("select username from user limit $rand, 1");
```

7. 尽量避免SELECT *

从数据库里读出越多的数据，那么查询就会变得越慢。并且，如果你的数据库服务器和WEB服务器是两台独立的服务器的话，这还会增加网络传输的负载。

所以，你应该养成一个需要什么就取什么的好的习惯。

eg :

```
//不推荐
$r = mysql_query("select * from user where user_id = 1");
$d = mysql_fetch_assoc($r);
echo "Welcome {$d['username']}";
//推荐
$r = mysql_query("select username from user where user_id = 1");
$d = mysql_fetch_assoc($r);
echo "Welcome {$d['username']}":
```

8. 永远为每张表设置一个ID

应该为数据库里的每张表都设置一个ID作为其主键，而且最好是INT类型（UNSIGNED），并设置上自动增长AUTO_INCREMENT标志。

使用VARCHAR类型来当主键会使性能下降。

9. 使用ENUM而不是VARCHAR

ENUM类型是非常快和紧凑的。实际上，其保存的是TINYINT, 但其外表上显示为字符串。

如果你有一个字段，比如“性别”，“国家”，“民族”，“状态”或“部门”，你知道这些字段的取值是有限而且固定的，那么，你应该使用ENUM而不是VARCHAR。

MYSQL也有一个“建议”告诉你怎么去重新组织你的表结构。

当你有一个VARCHAR字段时，这个建议会告诉你将其改成ENUM类型。

使用PROCEDURE ANALYSE()可以得到相关的建议。

10. 从PROCEDURE ANALYSE() 取得建议

11. 尽可能的使用NOT NULL

12. Prepared Statements

是一种运行在后台的SQL语句集合，我们可以从使用prepared statements获得很多好处，无论是性能问题还是安全问题。

prepared statements可以检查一些你绑定好的变量，可以保护你的程序不会收到SQL注入式攻击。

给prepared statements定义一些参数，当一个相同的查询使用多次的时候，MySQL只会解析一次。

eg :

```
//创建prepared statement
if ($stmt = $mysqli->prepare("select username from user where state=?")) {
    //绑定参数
    $stmt->bind_param("s",$state);
    //执行
    $stmt->execute();
    //绑定结果
    $stmt->bind_result($username);
    //移动游标
    ￥stmt->fetch();
    printf("%s is from %s\n", $username, $state);
    $stmt->close();
}
```

13. 无缓冲的查询

正常的情况下，当你在当你在你的脚本中执行一个SQL语句的时候，你的程序会停在那里直到没这个SQL语句返回，然后你的程序再往下继续执行。

你可以使用无缓冲查询来改变这个行为。

mysql_unbuffered_query() 发送一个SQL语句到MySQL而并不像mysql_query()一样去自动fethch和缓存结果。这会相当节约很多可观的内存，尤其是那些会产生大 量结果的查询语句，并且，你不需要等到所有的结果都返回，只需要第一行数据返回的时候，你就可以开始马上开始工作于查询结果了。

然而，这会有一些限制。因为你要么把所有行都读走，或是你要在进行下一次的查询前调用 mysql_free_result() 清除结果。而且，

mysql_num_rows() 或 mysql_data_seek() 将无法使用。所以，是否使用无缓冲的查询你需要仔细考虑。

14. 把IP地址存成UNSIGNED INT

很多程序员都会创建一个VARCHAR(15)字段来存放字符串形式的IP而不是整形的IP。如果你用整形来存放，只需要4个字节，并且你可以有定长的字段。这可以为你带来查询上的有事，尤其是当你需要使用这样的WHERE条件： IP between ip1 and ip2.

我们必须要使用UNSIGNED INT, 因为IP地址会使用整个32为的无符号整形。

而你的查询，你可以使用 INET_ATON() 来把一个字符串IP转成一个整形，并使用 INET_NTOA() 把一个整形转成一个字符串IP。在PHP中，也有这样的函数 ip2long() 和 long2ip()。

eg :

```
$r = "update users set ip = inet_aton('{$_server['remote_addr']}') where user_id = $user_id";
```

15. 固定长度的表会更快

如果表中的所有字段都是固定长度的，整个表会被认为是“static"或”fixed-length".

例如，表中没有如下类型的字段：VARCHAR, TEXT, BLOB。只要你包括了其中一个这些字段，那么这个表就不是“固定长度静态表”了，这样MySQL引擎会用另一种方法来处理。

固定长度的表会提高查询性能，因为MySQL搜寻得会更快一些，因为固定长度是很容易计算下一个数据的偏移量的，所有读取的自然也会很快。而如果字段不是定长的，那么，每一次要找下一条的话，需要程序找到主键。

并且，固定长度的表也更容易被缓存和重建。不过，唯一的副作用是，固定长度的字段会浪费一些空间，因为定长的字段无论你用不用，他都是要分配那么多的空间。

使用“垂直分割”技术(见下一条)，你可以分割你的表成为两个一个是定长的，一个则是不定长的。

16. 垂直分割

垂直分割是一种把数据库中的表按列变成几张表的方法。这样可以降低表的复杂度和字段的数目，从而达到优化的目的。

示例一：在Users表中有一个字段是家庭地址，这个字段是可选字段，相比起，而且你在数据库操作的时候除了个人信息外，你并不需要经常读取或是改 写这个字段。那么，为什么不把他放到另外一张表中呢? 这样会让你的表有更好的性能，大家想想是不是，大量的时候，我对于用户表来说，只有用户ID，用户名，口令，用户角色等会被经常使用。小一点的表总是会有 好的性能。

示例二： 你有一个叫 “last_login” 的字段，它会在每次用户登录时被更新。但是，每次更新时会导致该表的查询缓存被清空。所以，你可以把这个字段放到另一个表中，这样就不会影响你对用户 ID，用户名，用户角色的不停地读取了，因为查询缓存会帮你增加很多性能。

另外，你需要注意的是，这些被分出去的字段所形成的表，你不会经常性地去Join他们，不然的话，这样的性能会比不分割时还要差，而且，会是极数级的下降。

17. 拆分大的DELETE或INSERT语句

如果你需要在一个在线的网站去执行一个大的DELETE或INSERT查询，你需要非常小心，要避免你的操作让你的整个网站停止响应。

因为这两个操作是会锁表的，表一锁住了，别的操作都进不来了。

Apache 会有很多的子进程或线程。所以，其工作起来相当有效率，而我们的服务器也不希望有太多的子进程，线程和数据库链接，这是极大的占服务器资源的事情，尤其是内存。

如果你把你的表锁上一段时间，比如30秒钟，那么对于一个有很高访问量的站点来说，这30秒所积累的访问进程/线程，数据库链接，打开的文件数，可能不仅仅会让你泊WEB服务Crash，还可能会让你的整台服务器马上掛了。

所以，如果你有一个大的处理，你一定要把其拆分，使用LIMIT条件是一个好的方法。

eg :

```
while (1) {
    //每次制作1000条
    mysql_query("delete from logs where log_date <= '2009-11-01' limit 1000");
    if (mysql_affected_rows() ==0) {
        //没得可删了，退出!
        break;
    }
    //每次都要休息一会儿
    usleep(50000);
}
```

18. 越小的列会越快

对于大多数的数据库引擎来说，硬盘操作可能是最重大的瓶颈。所以，把你的数据变得紧凑会对这种情况非常有帮助，因为这减少了对硬盘的访问。

参看 MySQL 的文档 Storage Requirements 查看所有的数据类型。

如果一个表只会有几列罢了(比如说字典表，配置表)，那么，我们就没有理由使用 INT 来做主键，使用 MEDIUMINT, SMALLINT 或是更小的 TINYINT 会更经济一些。如果你不需要记录时间，使用 DATE 要比 DATETIME 好得多。

当然，你也需要留够足够的扩展空间，不然，你日后来干这个事，你会死的很难看，参看Slashdot的例子(2009年11月06日)，一个简单的ALTER

TABLE语句花了3个多小时，因为里面有一千六百万条数据。

19. 选择正确的存储引擎

MyISAM 适合于一些需要大量查询的应用，但其对于有大量写操作并不是很好。甚至你只是需要update一个字段，整个表都会被锁起来，而别的进程，就算是读进程都 无法操作直到读操作完成。另外，MyISAM 对于 SELECT COUNT(*) 这类的计算是超快无比的。

InnoDB 的趋势会是一个非常复杂的存储引擎，对于一些小的应用，它会比 MyISAM 还慢。他是它支持“行锁” ，于是在写操作比较多的时候，会更优秀。并且，他还支持更多的高级应用，比如：事务。

20. 使用一个对象关系映射器（ORM）

Object Relational Mapper

lazy loading, 只有在需要去取值的时候才会去真正的做。

这种机制的副作用 ： 很有可能会因为要去创建很多很多小的查询反而会降低性能。

ORM 还可以把你的SQL语句打包成一个事务，这会比单独执行他们快得多得多。

目前，个人最喜欢的PHP的ORM是：Doctrine。

21. 小心”永久链接“

“永久链接”的目的是用来减少重新创建MySQL链接的次数。当一个链接被创建了，它会永远处在连接的状态，就算是数据库操作已经结束了。而且，自 从我们的Apache开始重用它的子进程后——也就是说，下一次的HTTP请求会重用Apache的子进程，并重用相同的 MySQL 链接。

PHP手册：mysql_pconnect()

在理论上来说，这听起来非常的不错。但是从个人经验(也是大多数人的)上来说，这个功能制造出来的麻烦事更多。因为，你只有有限的链接数，内存问题，文件句柄数，等等。

而且，Apache 运行在极端并行的环境中，会创建很多很多的了进程。这就是为什么这种“永久链接”的机制工作地不好的原因。在你决定要使用“永久链接”之前，你需要好好地考虑一下你的整个系统的架构。

补充 ：

mysql强制索引和禁止某个索引

1. 强制使用索引 ： force index

eg ：

```
select * from table force index(PRI) limit 2;
select * from table force index(test_index) limit 2;
select * from table force index(pri, test_index) limit 2;
```

2. 禁止某个索引 ： ignore idex

```
select * from table ignore index(PRI) limit 2;
select * from table ignore index(test_index) limit 2;
select * from table ignore index(PRI, test_index) limit 2;
```

参考 ： http://www.cnblogs.com/daxian2012/articles/2767989.html