有一定成功

最佳化部署方案（1）

@CUPDIR Tuesday, October 27, 2009

难度：高级

最佳化部署定义 Optimize The Deployment

- 所谓最佳化部署是我昨天晚上刚想起的一个名字，因为今天不会针对某一方面的课题来讲解，而是多方面的。而今的WEB被互联网所青睐着，多方面的。从静态到动态，从以往的技术层面一直到现在的所谓叫技术瓶颈。所谓瓶颈就是我们在遇到问题一定要处理的。而不是处理不了的。其实这些全部在证明着互联网在成长着。不过做为技术人士的我们，也要不断的去探索追求着。一个WEB开发高手一直沉思着去考虑他的程序能否更好的运行，或者用时间来磨练着它们的意志。今天我们说的就是这个程序员所要干的一些事情。最佳化部署方案，能否让你的程序和思路在这高端互联网中跑的更优美，更顺畅。才是我们需要的。

一段处理问题的代码

- 有时候我们会遇到这样一个问题。数据库读写分离操作问题。这样的问题会促使我们改变我们的代码从而也会考虑如何去部署代码，让它更容易更有效率。
- 不过随着我们的改变也会有些即将到来的问题展现到我们的面。是的，数据同步。下来我们将延伸到热备。数据库小型集群部署。

例子

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+-
 1  <?php
 2
 3  #A 操作
 4  $dbname = 'ucar';
 5  $link_id = array();
 6  $link_id[] = connect('localhost','root','');  //标识链接ID  #2  主  拉
 7  $link_id[] = connect('www.ucarshow.com','haiquan','haiquan');  //标识链接ID #3  从  推
 8
 9  $sql = "SELECT * FROM tb_register LIMIT 10";
10  $result = execute($sql);
11  $row = mysql_fetch_assoc($result);
12  //print_r($row);
13
14  $insert = "INSERT INTO tb_register(nickname) VALUES ('你好')";
15
16  $msg =execute($insert);
17  print_r($msg);
18
19  mysql_free_result($result);
20
21 function &connect($local,$user,$pass) {
22      return mysql_connect($local,$user,$pass);
23  }
24 function &select_db($linkid=0) {
25      if(null === $GLOBALS['dbname']){
26          return false;
27      }else{
28          mysql_query('SET NAMES utf8',$GLOBALS['link_id'][$linkid]);
29          return mysql_select_db($GLOBALS['dbname'],$GLOBALS['link_id'][$linkid]);
30      }
31  }
32
33 function execute($sql) {
34      if(in_array(strtoupper(substr($sql,0,6)),array('INSERT','UPDATE'))) {
35          select_db(0);
36          return mysql_query($sql,$GLOBALS['link_id'][0]);
37
38      }else{
39          select_db(1);
40          return mysql_query($sql,$GLOBALS['link_id'][1]);
41      }
42  }
43  ?>
```

发现的问题将要被处理

- 提高代码的重用型以及效率 OOP
- “slaver”面临着数据不被更新
- 再分析磁盘I/O问题?
- Memcache多线程集群 [2]
- 技术所给我们带来的瓶颈

提高代码的重用型

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8-
 1  <?php
 2
 3  abstract class DMasterAbstract {
 4      public static $m = null;
 5      public function __call($method,$args) {
 6
 7          if(in_array(strtoupper(substr($args[0],0,6)),array('INSERT','UPDATE'))) {
 8              self::$m = 'master';
 9
10
11          }else{
12              self::$m = 'slaver';
13          }
14          call_user_func_array(array('DMaster','connected'),$args);
15          return $this;
16      }
17  }
```

创建主从标识，他可以是个抽象的。回调子类静态方法。

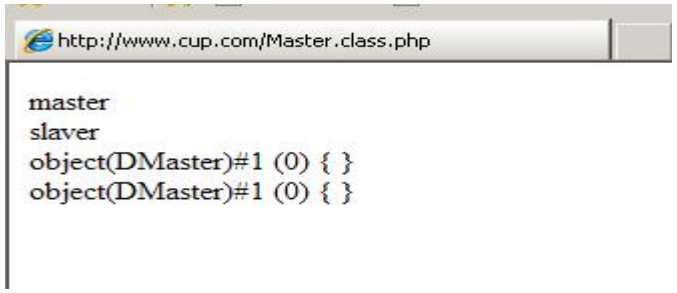
提高代码的重用型

```php
class DMaster  extends DMasterAbstract{

    protected static $_config = array();

    protected static $_linkid = array();

    private static $_charset = 'utf8';

    private static $_instanced = null;

    private static $_linkmarker = null;

    private  function __construct($config =null) {
        self::$_config = (null === $config) ? self::$_config : $config;
    }
    public static function getInstanced($config = null) {
        if( null === self::$_instanced ){
            self::$_instanced = new self($config);
        }
        return self::$_instanced;

    }
    protected static function connected($args=array()) {//回调连接
        if(null === self::$_linkmarker){
            self::$_config = $args;
            self::$_linkmarker = self::connect();
        }
        //foreach($args)
    }

    protected static function connect() {

        echo self::$m;
        //print_r(self::$_config);
        //self::$_linkid[] = mysql_connect(self::$_config['master'])
    }


}
```

- 创建Dmaster类。在本类中为了防止类的资源再生，选择做一个私有架构方法。称为单态。

DMaster类应用

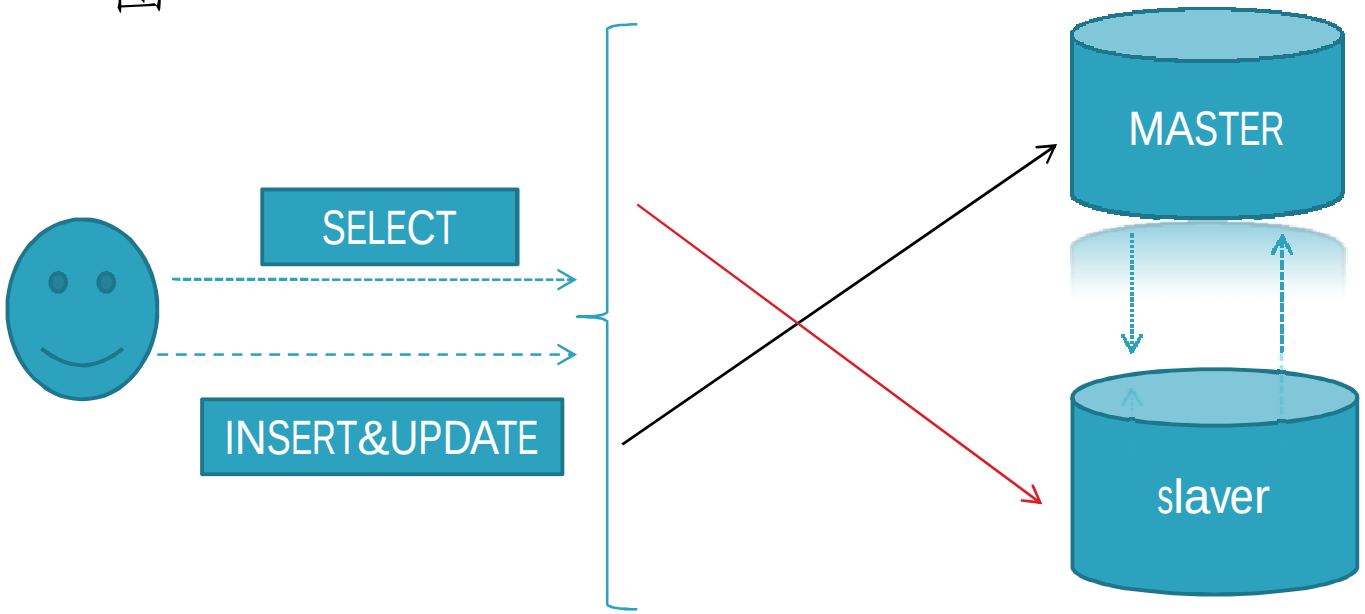
```php
$config = array(

    'master'=>array(
        'DBHOST'=>'localhost',
        'DB_NAME'=>'ucar',
        'DB_USER'=>'root',
        'DB_PASS'=>'',
        'DB_CHARSET'=>'utf8'
        ),
    'slaver'=>array(
        'DBHOST'=>'www.ucarshow.com',
        'DB_NAME'=>'ucar',
        'DB_USER'=>'root',
        'DB_PASS'=>'',
        'DB_CHARSET'=>'utf8'
        ),
);

$a = DMaster::getInstanced($config)->execute("INSERT * FROM tb_register LIMIT 20");

echo "<br />";

$b = DMaster::getInstanced($config)->execute("SELECT * FROM tb_register LIMIT 20");
echo "<br />";
var_dump($a);
echo "<br />";
var_dump($b);
//CODE    master  slaver
?>
```



- 分配一个存放主从配置数组
- 碰到INSERT ,UPDATE我们会让提交给DMasterAbstract类，让它去决断是从还是主。在回调到子类去做一些工作。

“slaver”面临着数据不被更新

- 程序以及开发完毕，省下的工作就是从软件着手去完成一些后续工作。因为两台数据库他们的分工不同，但应用中我们的要求数据一定是及时更新的。看一下图



如何让MYSQL支持数据同步?

- 在MYSQL3以后都支持数据复制功能。这样我们两台MYSQL就可以做到数据同步了。在数据同步模式上又分出两种。
- Master->slaver 方式 单向同步
- Master<->slaver方式 双向同步
- 在LINUX下配置文件一般是你编译的MYSQL目录CONF下
- WINDOWS下可以在安装的MYSQL目录里找到。

Master配置

```
#skip-networking

# Disable Federated by default
#skip-federated

# Replication Master Server (default)
# binary logging is required for replication
log-bin=mysql-bin
binlog-do-db=uchome
# binary logging format - mixed recommended
#binlog_format=mixed
server-id       = 1
```

- 主库必须启用Bin log，主库和从库必须有唯一的Server Id
- 从库必须清楚了解从主库的哪一个Bin log文件的哪一个偏移位置起开始复制
- 从库可以从主库只复制指定的数据库，或者数据库的某些数据表
- 主库和从库的数据库名称可以不一样，不过还是推荐使用一样的名称
- 主库和从库的MySQL版本需保持一致

- 做为主我们只需要配置需要同步到那个库就可以了。我们将要指定一下参数。Log-bin:日志的存放。**Binlog-do-db需要同步的库，server-id 主从标识**
- 注：数据结构一定是相同的。

Slaver

```
server-id=2
master-host=172.19.12.X #主机A的地址
master-user=cupdir #主机A提供给B的用户，该用户中需要包括数据库uchome的权限
master-password=cupdir #访问密码
master-port=3306 #端口，主机的MYSQL端口
master-connect-retry=60 #重试间隔60秒
replicate-do-db=uchome #同步的数据库
```

- 在Slaver配置中加入以下配置项，创建用户。让Master具有访问Slaver数据库的权限。

创建用户

- 将Master数据的权限给Slaver
- 在Master服务器上执行MYSQL命令
- mysql>GRANT FILE ON *.* TO cupdir@‘172.19.12.X’ IDENTIFIED BY ‘haiquan’;
- 重新启动所有MYSQL
- mysql>slave start; 查看同步状态 【Slaver】
- 在Master上查看同步配置情况
- mysql>show master status; 【Msater】
- mysql>show slave status;【Slaver】
- 在主库更新操作，查看从库是否同步。

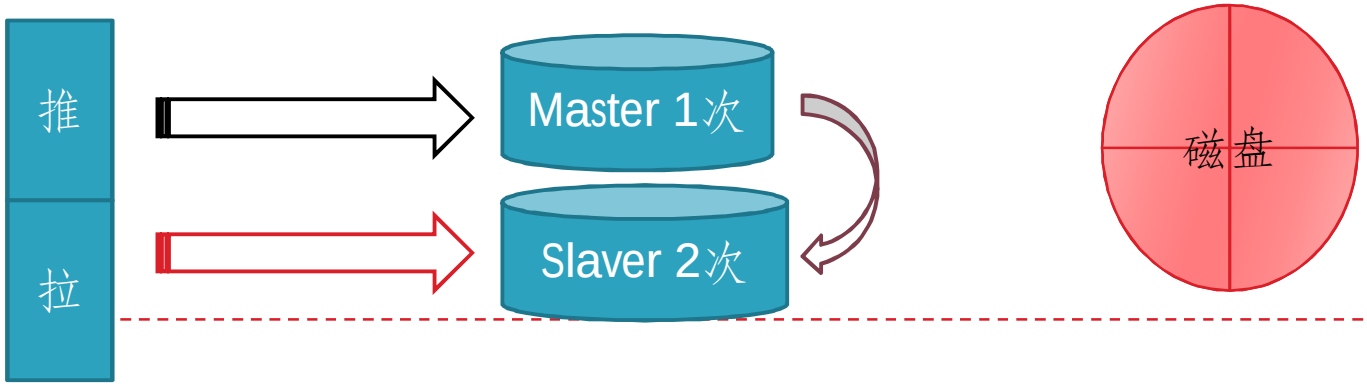
查看状态

```
mysql> slave start
    -> ;
Query OK, 0 rows affected (0.00 sec)

mysql> slave start;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> show master status;
+------------------+----------+---------------+------------------+
| File             | Position | Binlog_Do_DB  | Binlog_Ignore_DB |
+------------------+----------+---------------+------------------+
| mysql-bin.000004 |       98 | uchome,uchome |                  |
+------------------+----------+---------------+------------------+
1 row in set (0.01 sec)
```

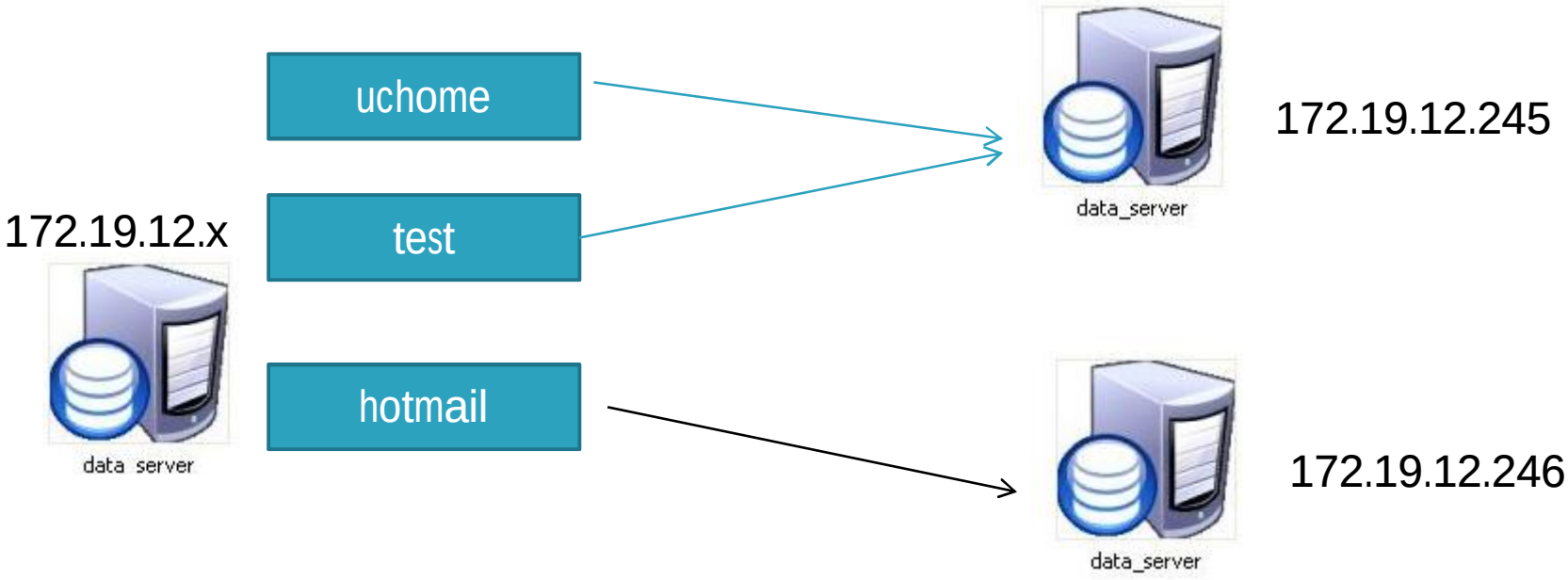
磁盘I/O问题

- 接下来我们把读写分离做到了，确实给MYSQL做了一些减轻压力的工作。但下来面临的挑战是。我们的服务器磁盘问题。因为推数据试我们对Master进行平凡的操作。当谈 这个只是很小的一部分。我们可以不去考虑。但Slaver就倒霉了。因为他身上所承受的压力是双向的。User拉和Mater的推.当Mater操作量大于并发数，也就是5000的时候。服务器会当掉，随之Slaver的数据也不会及时更新。MYSQL是以文件形式存放数据的。所以磁盘的损耗会带来很大的问题，看图



磁盘I/O问题

- 分布式数据库文件 NFS MOUNT
- mount [-afFhnrvVw] [-L] [-o] [-t] [设备名] [加载点]
- 看下图



Thanks~~~

Cupdir@gmail.com