[ThinkPHP项目实现分布式部署（网站集群）](http://blog.csdn.net/cuizm/article/details/51821829)

Thinkphp 版本（3.1.X）

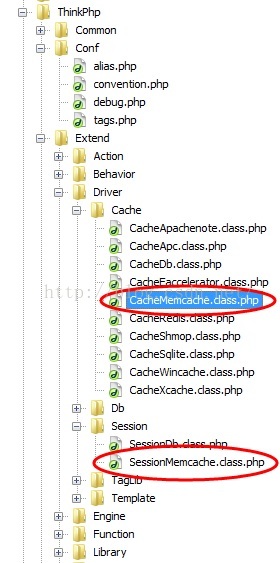
普通的Web开发，常用的模式就是用户登录之后，登录状态信息保存在Session中，用户一些常用的热数据保存在文件缓存中，用户上传的附件信息保存在Web服务器的某个目录上。这种方式对于一般的Web应用，使用很方便，完全能够胜任。但是对于高并发的企业级网站，就应付不了了。需要采用Web集群实现负载均衡。

　　使用Web集群方式部署之后，首要调整的就是用户状态信息与附件信息。用户状态不能再保存到Session中，缓存也不能用本地Web服务器的文件缓存，以及附件，也不能保存在Web服务器上了。因为要保证集群里面的各个Web服务器，状态完全一致。因此，需要将用户状态、缓存等保存到专用的缓存服务器，比如Memcache。附件需要保存到云存储中，比如七牛云存储、阿里云存储、腾讯云存储等。

本文以ThinkPHP开发框架为例，说明如何设置，能够将Session、缓存等保存到Memcache缓存服务器上。

　　下载缓存的Memcache处理类，放到Thinkphp\Extend\Driver\Cache目录中；下载Session的Memcache处理类，放到Thinkphp\Extend\Driver\Session

目录中，如下图所示：



修改配置文件，调整Session与缓存，都记录到Memcache服务器上。打开ThinkPHP\Conf\convention.**[PHP](http://lib.csdn.net/base/php" \o "PHP知识库" \t "http://blog.csdn.net/cuizm/article/details/_blank)**，增加配置项：

1. /\* Memcache缓存设置 \*/
2. 'MEMCACHE\_HOST'         => '192.168.202.20',
3. 'MEMCACHE\_PORT'         => 11211,

修改数据缓存为Memcache：

**[php]** [view plain](http://blog.csdn.net/cuizm/article/details/51821829" \o "view plain) [copy](http://blog.csdn.net/cuizm/article/details/51821829" \o "copy)

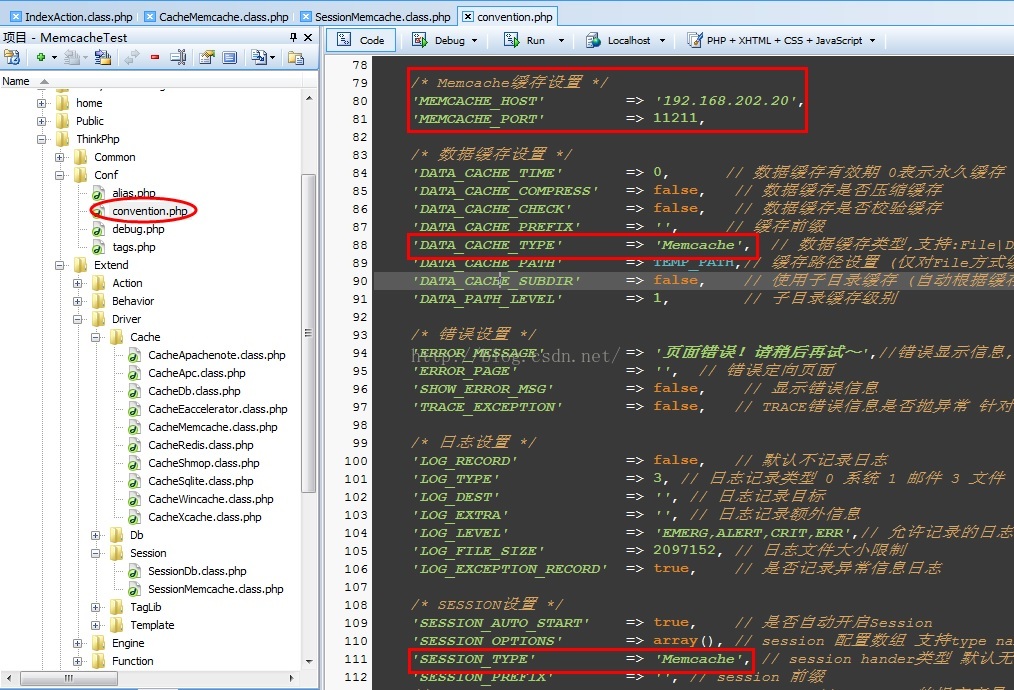
1. 'DATA\_CACHE\_TYPE'       => 'Memcache',

修改Session为Memcache：

**[php]** [view plain](http://blog.csdn.net/cuizm/article/details/51821829" \o "view plain) [copy](http://blog.csdn.net/cuizm/article/details/51821829" \o "copy)

1. 'SESSION\_TYPE'          => 'Memcache',

如下图所示：



因为云存储各类比较多，附件存储到云存储上，就不再赘述，参数各云存储提供的sdk即可。经过以上修改，就可以将Web服务器进行分布式部署了。

　附件1：CacheMemcache.class.php

<?php

// +----------------------------------------------------------------------

// | ThinkPHP [ WE CAN DO IT JUST THINK IT ]

// +----------------------------------------------------------------------

// | Copyright (c) 2006-2012 http://thinkphp.cn All rights reserved.

// +----------------------------------------------------------------------

// | Licensed ( http://www.apache.org/licenses/LICENSE-2.0 )

// +----------------------------------------------------------------------

// | Author: liu21st <liu21st@gmail.com>

// +----------------------------------------------------------------------

defined('THINK\_PATH') or exit();

/\*\*

\* Memcache缓存驱动

\* @category Extend

\* @package Extend

\* @subpackage Driver.Cache

\* @author liu21st <liu21st@gmail.com>

\*/

class CacheMemcache extends Cache {

/\*\*

\* 架构函数

\* @param array $options 缓存参数

\* @access public

\*/

function \_\_construct($options=array()) {

if ( !extension\_loaded('memcache') ) {

throw\_exception(L('\_NOT\_SUPPERT\_').':memcache');

}

$options = array\_merge(array (

'host' => C('MEMCACHE\_HOST') ? C('MEMCACHE\_HOST') : '127.0.0.1',

'port' => C('MEMCACHE\_PORT') ? C('MEMCACHE\_PORT') : 11211,

'timeout' => C('DATA\_CACHE\_TIMEOUT') ? C('DATA\_CACHE\_TIMEOUT') : false,

'persistent' => false,

),$options);

$this->options = $options;

$this->options['expire'] = isset($options['expire'])? $options['expire'] : C('DATA\_CACHE\_TIME');

$this->options['prefix'] = isset($options['prefix'])? $options['prefix'] : C('DATA\_CACHE\_PREFIX');

$this->options['length'] = isset($options['length'])? $options['length'] : 0;

$func = $options['persistent'] ? 'pconnect' : 'connect';

$this->handler = new Memcache;

$options['timeout'] === false ?

$this->handler->$func($options['host'], $options['port']) :

$this->handler->$func($options['host'], $options['port'], $options['timeout']);

}

/\*\*

\* 读取缓存

\* @access public

\* @param string $name 缓存变量名

\* @return mixed

\*/

public function get($name) {

N('cache\_read',1);

return $this->handler->get($this->options['prefix'].$name);

}

/\*\*

\* 写入缓存

\* @access public

\* @param string $name 缓存变量名

\* @param mixed $value 存储数据

\* @param integer $expire 有效时间（秒）

\* @return boolen

\*/

public function set($name, $value, $expire = null) {

N('cache\_write',1);

if(is\_null($expire)) {

$expire = $this->options['expire'];

}

$name = $this->options['prefix'].$name;

if($this->handler->set($name, $value, 0, $expire)) {

if($this->options['length']>0) {

// 记录缓存队列

$this->queue($name);

}

return true;

}

return false;

}

/\*\*

\* 删除缓存

\* @access public

\* @param string $name 缓存变量名

\* @return boolen

\*/

public function rm($name, $ttl = false) {

$name = $this->options['prefix'].$name;

return $ttl === false ?

$this->handler->delete($name) :

$this->handler->delete($name, $ttl);

}

/\*\*

\* 清除缓存

\* @access public

\* @return boolen

\*/

public function clear() {

return $this->handler->flush();

}

}

附件2：SessionMemcache.class.php

<?php

// +----------------------------------------------------------------------

// |

// +----------------------------------------------------------------------

// | Copyright (c) 2013-

// +----------------------------------------------------------------------

// | Licensed ( http://www.apache.org/licenses/LICENSE-2.0 )

// +----------------------------------------------------------------------

// | Author: richievoe <richievoe@163.com>

// +----------------------------------------------------------------------

/\*\*

\* 自定义Memcache来保存session

\*/

Class SessionMemcache{

//memcache对象

private $mem;

//SESSION有效时间

private $expire;

//外部调用的函数

public function execute(){

session\_set\_save\_handler(

array(&$this,'open'),

array(&$this,'close'),

array(&$this,'read'),

array(&$this,'write'),

array(&$this,'destroy'),

array(&$this,'gc')

);

}

//连接memcached和初始化一些数据

public function open($path,$name){

$this->expire = C('SESSION\_EXPIRE') ? C('SESSION\_EXPIRE') :ini\_get('session.gc\_maxlifetime');

$this->mem = new Memcache;

return $this->mem->connect(C('MEMCACHE\_HOST'), C('MEMCACHE\_PORT'));

}

//关闭memcache服务器

public function close(){

return $this->mem->close();

}

//读取数据

public function read($id){

$id = C('SESSION\_PREFIX').$id;

$data = $this->mem->get($id);

return $data ? $data :'';

}

//存入数据

public function write($id,$data){

$id = C('SESSION\_PREFIX').$id;

//$data = addslashes($data);

return $this->mem->set($id,$data,0,$this->expire);

}

//销毁数据

public function destroy($id){

$id = C('SESSION\_PREFIX').$id;

return $this->mem->delete($id);

}

//垃圾销毁

public function gc(){

return true;

}

}

?>

　经过以上配置，就可以将用户状态信息与缓存信息保存到Memcache中。可以使用负载均衡服务器，来实现大规模集群方式架设网站。