参考：<http://blog.sina.com.cn/s/blog_62b832910100xok2.html>

目录

[Redis pub/sub(Publish,Subscribe)](http://blog.sina.com.cn/s/blog_62b832910100xok2.html#_Toc312918336)

[1、         Pub/Sub功能](http://blog.sina.com.cn/s/blog_62b832910100xok2.html#_Toc312918337)

[2、         Pub/Sub机制](http://blog.sina.com.cn/s/blog_62b832910100xok2.html#_Toc312918338)

[3、         Pub/Sub 在redis当中的实现](http://blog.sina.com.cn/s/blog_62b832910100xok2.html#_Toc312918339)

[4、         Php-redis扩展测试](http://blog.sina.com.cn/s/blog_62b832910100xok2.html#_Toc312918340)

[5、         Redis pub/sub python客户端测试](http://blog.sina.com.cn/s/blog_62b832910100xok2.html#_Toc312918341)

[6、         Redis pub/sub与node.js结合](http://blog.sina.com.cn/s/blog_62b832910100xok2.html#_Toc312918342)

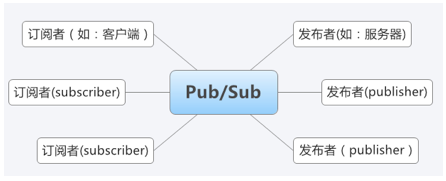
[7、         Redis pub/sub压力测试](http://blog.sina.com.cn/s/blog_62b832910100xok2.html#_Toc312918343)

[8、         Redis pub/sub服务应用场景分析](http://blog.sina.com.cn/s/blog_62b832910100xok2.html#_Toc312918344)

[9、         附件：](http://blog.sina.com.cn/s/blog_62b832910100xok2.html#_Toc312918345)

**1、 Pub/Sub功能**

Pub/Sub功能（means Publish, Subscribe）即发布及订阅功能。基于事件/消息的系统中，Pub/Sub是目前广泛使用的通信模型，它采用事件作为基本的通信机制，提供大规模系统所要求的松散耦合的交互模式：订阅者(如客户端)以事件订阅的方式表达出它有兴趣接收的一个事件或一类事件；发布者(如服务器)可将订阅者感兴趣的事件随时通知相关订阅者。

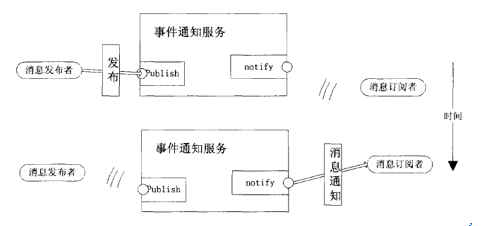
[](http://photo.blog.sina.com.cn/showpic.html#blogid=62b832910100xok2&url=http://s8.sinaimg.cn/orignal/62b83291tbb668f54b337)

Publisher/subscribe 简易模型

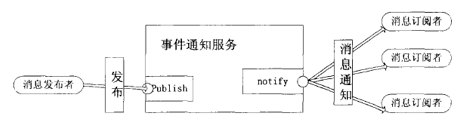
**2、 Pub/Sub机制**

Pub/Sub功能（means Publish,Subscribe）即发布及订阅功能[](http://photo.blog.sina.com.cn/showpic.html#blogid=62b832910100xok2&url=http://s12.sinaimg.cn/orignal/62b83291tbb669066c4bb)

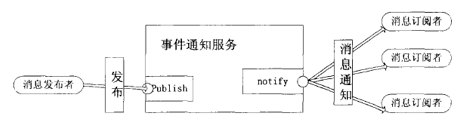
1.       **时间非耦合：**发布者和订阅者不必同时在线，它们不必同时参与交互。

[](http://photo.blog.sina.com.cn/showpic.html#blogid=62b832910100xok2&url=http://s1.sinaimg.cn/orignal/62b83291tbb66916f15a0)

2.       **空间非耦合：**发布者和订阅者不必相互知道对方所在的位置。发布者通过事件服务发布事件，订阅者通过事件服务间接获得事件。发布者和订阅者不需要拥有直接到对方的引用，也不必知道有多少个订阅者或者是发布者参与交互。

[](http://photo.blog.sina.com.cn/showpic.html#blogid=62b832910100xok2&url=http://s16.sinaimg.cn/orignal/62b83291tbb6693c218df)

3.       **同步非耦合：**发布者/订阅者是异步模式。发布者可不断地生产事件，而订阅者(通过一个回调)则可异步地得到产生事件的通知。

[](http://photo.blog.sina.com.cn/showpic.html#blogid=62b832910100xok2&url=http://s8.sinaimg.cn/orignal/62b83291tbb66aba43dc7)

**分类：**

按照订阅方式分为基于主题（topic-based）、基于内容（content-based）、基于类型（type-based）的pub/sub方式。

[](http://photo.blog.sina.com.cn/showpic.html#blogid=62b832910100xok2&url=http://s2.sinaimg.cn/orignal/62b83291tbb66acc67af1)

**总结：**

Pub/Sub是可适用于可扩展要求高、松散耦合系统的分布式交互模型。

在抽象层中，它的时间非耦合、空间非耦合和同步非耦合性可允许参与者不依赖另一个而独立操作，具有一定的可扩展性；然而在实现层，可扩展性仍受其他原因的牵制。

例如：1、灵活的订阅要求复杂的过滤和路由算法；

      2、高可用性开销（事件侦听、日志重传）；

      3、消息认可带来的网络流量消耗；

4、庞大的订阅者数据带来的系统开销；

基于事件的Pub/Sub中间件的开发与利用在一定程度上可以提高系统的效率。

**3、 Pub/Sub 在redis当中的实现**

Redis-cli连接测试命令行参考*附件I*

*协议测试命令*

* [**PSUBSCRIBE**](http://redis.io/commands/psubscribe)
* [PUBLISH](http://redis.io/commands/publish)
* [PUNSUBSCRIBE](http://redis.io/commands/punsubscribe)
* [SUBSCRIBE](http://redis.io/commands/subscribe)
* [UNSUBSCRIBE](http://redis.io/commands/unsubscribe)

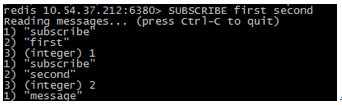
  **订阅者（redis-cli）连接订阅**

/usr/local/bin/redis-cli  -h 10.54.37.212 -p 6380

[Redis <wbr>pub/sub(Publish,Subscribe)](http://photo.blog.sina.com.cn/showpic.html#blogid=62b832910100xok2&url=http://s5.sinaimg.cn/orignal/62b83291tbb66adc93a74)

redis-cli充当订阅者订阅（first 与second）两个topic

SUBSCRIBE first second

[](http://photo.blog.sina.com.cn/showpic.html#blogid=62b832910100xok2&url=http://s15.sinaimg.cn/orignal/62b83291tbb66b250b0fe&690)

  **发布者（redis-cli）连接发布**

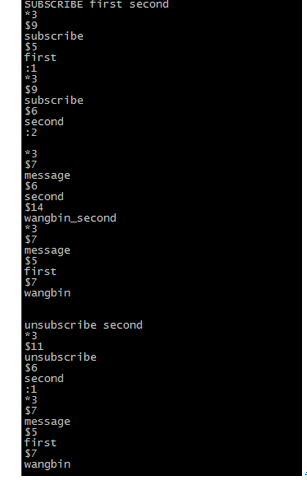
/usr/local/bin/redis-cli  -h 10.54.37.212 -p 6380

Pubish first wangbin

Publish second wangbin\_second

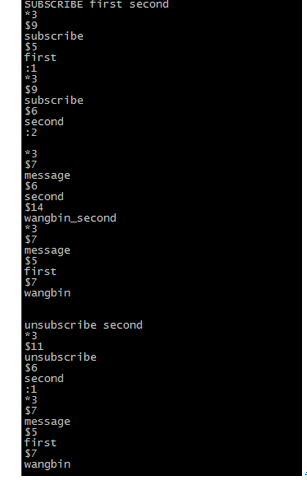


  **订阅者（redis-cli）信息接收**

[](http://photo.blog.sina.com.cn/showpic.html#blogid=62b832910100xok2&url=http://s15.sinaimg.cn/orignal/62b83291tbb66b26fa61e&690)

  **取消订阅UNSUBSCRIBE测试**

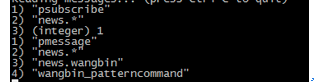
Redis-cli对取消订阅有bug可以使用telnet 进行测试

[](http://photo.blog.sina.com.cn/showpic.html#blogid=62b832910100xok2&url=http://s15.sinaimg.cn/orignal/62b83291tbb66b26fa61e&690)

取消订阅测试成功；

  **匹配模式订阅PSUBSCRIBE测试**

订阅者：

[](http://photo.blog.sina.com.cn/showpic.html#blogid=62b832910100xok2&url=http://s10.sinaimg.cn/orignal/62b83291tbb66b2704fa9&690)

发布者：

publish news.wangbin wangbin\_patterncommand

[Redis <wbr>pub/sub(Publish,Subscribe)](http://photo.blog.sina.com.cn/showpic.html#blogid=62b832910100xok2&url=http://s6.sinaimg.cn/orignal/62b83291tbb66b26cc2f5&690)

  **匹配模式取消订阅PUNSUBSCRIBE测试**

在telnet下测试如下：

PSUBSCRIBE news.\*  wangbin\*

punsubscribe wangbin\*

[](http://photo.blog.sina.com.cn/showpic.html#blogid=62b832910100xok2&url=http://s4.sinaimg.cn/orignal/62b83291tbb66b2702503&690)

测试成功!

**4、 Php-redis扩展测试**

<https://github.com/nicolasff/phpredis> pecl扩展包目前只提供了两个接口 publish  subscribe

phpredis是c写的php模块

<https://github.com/jamm/Memory/blob/master/RedisServer.php>

php这个是php是基于redis protocol的fsocketopen链接后操作的类库，提供的接口比较全面；publish可以进入数据，但是subscrbie没有阻塞；

可以在原类包当中修改其加入对阻塞模形的支持；

发布功能：

$redis = new Redis();

$res = $redis->connect($REDIS\_HOSTS['CACHE']['host'], $REDIS\_HOSTS['CACHE']['port'], 1 );

$res = $redis->publish($key,$value);

定阅功能：

$redis = new Redis();

$res = $redis->pconnect($REDIS\_HOSTS['CACHE']['host'], $REDIS\_HOSTS['CACHE']['port']);

$res = $redis->subscribe(array($key),array('SinaRedis','subscribe\_handler'));

第二个参数为回调方法；

public static function subscribe\_handler($redis, $channel, $msg){

            print\_r($redis);

            echo $chan;

            echo $msg;

            return true;

  }

定阅 redis\_subscriber.php

SinaRedis::subscribe('wangbin\_test');

发布redis\_publisher.php

测试收到订阅者收到发布的内容

[Redis <wbr>pub/sub(Publish,Subscribe)](http://photo.blog.sina.com.cn/showpic.html#blogid=62b832910100xok2&url=http://s2.sinaimg.cn/orignal/62b83291tbb66c2f2d2e1)

**Php-redis扩展bug**

**段错误：**

无论是connect 还是pcconnect 当超时断后，会报错segmentation fault;

超时设置bug

仍会断开解决方法：ini\_set('default\_socket\_timeout', -1);// it works fine

**Web Server端测试**

Web端订阅应用

Apache下ob\_flush flush后可以输出，但是会漏下最后一条；求解决方案

Nginx下 会报504错误；求解决方法

Nginx 模块与redis直接连接呢?

Php-Fpm有没有什么设置呢

**5、 Redis pub/sub python客户端测试**

Wget https://github.com/downloads/andymccurdy/redis-py/redis-2.2.4.tar.gz

tar -xvf   redis-2.2.4.tar.gz

cd redis-2.2.4

python setup.py intall

安装成功

交互模式下测试：

发布者：

Python

Type "help", "copyright", "credits" or "license" for more information.

>>> import redis

>>> r = redis.Redis('10.54.37.212',6380);

>>> r.publish('wangbin\_test','this is a information');

59L

>>>

订阅者：

[root@hadoop-master1 python]# python

Python 2.4.3 (#1, Sep  3 2009, 15:37:12)

[GCC 4.1.2 20080704 (Red Hat 4.1.2-46)] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>> import redis

>>> r = redis.Redis('10.54.37.212',6380);

>>> r.subscribe('wangbin\_test');

>>> for msg in r.listen(): \

...     print msg

{'pattern': None, 'type': 'message', 'channel': 'wangbin\_test', 'data': 'this is a information'}

收到json字符串

Publisher.py

[Redis <wbr>pub/sub(Publish,Subscribe)](http://photo.blog.sina.com.cn/showpic.html#blogid=62b832910100xok2&url=http://s13.sinaimg.cn/orignal/62b83291tbb66c71918cc)

Subscribe.py

[Redis <wbr>pub/sub(Publish,Subscribe)](http://photo.blog.sina.com.cn/showpic.html#blogid=62b832910100xok2&url=http://s6.sinaimg.cn/orignal/62b83291tbb66ca606ec5)

**6、 Redis pub/sub与node.js结合**

Wget http://nodejs.org/dist/v0.6.6/node-v0.6.6.tar.gz

Tar –xvf node-v0.6.6.tar.gz

Cd node-v0.6.6

./configure

Make

Make install

Node 安装在/usr/localbin

添加到环境变量里面：

export  PATH=/usr/local/bin/:$PATH

安装：npm

curl -O <http://npmjs.org/install.sh>

sh install.sh

安装成功

Tar 版本过低，升级下

wget <http://ftp.gnu.org/gnu/tar/tar-1.26.tar.gz>

./configure --prefix=/usr --bindir=/bin --libexecdir=/usr/bin

Make

Make install

Tar –version

tar (GNU tar) 1.26 升级成功

安装socket.io  express hiredis redis

/usr/local/bin/npm  install socket.io

/usr/local/bin/npm  install express

/usr/local/bin/npm  install hiredis redis

**事件驱动用例：**

**Publish.js**

var redis = require("redis");

try{

    var client = redis.createClient(6380,'10.54.37.212');

    client.on(

        "error",

        function(err){

            console.log("err"+err);

            }

    );

    client.on('ready',

        function(){

            client.publish('wangbin\_test',"test,i am test");

            client.publish('wangbin\_test2',"test, i am test2");

            client.end();

            }

    );

}

catch(e){

        console.log("err:"+e);

}

**Subscribe.js**

var redis = require("redis");

try{

    var client = redis.createClient(6380,'10.54.37.212');

    client.on(

        "error",

        function(err){

            console.log("err"+err);

            }

    );

    client.on('ready',

        function(){

            client.subscribe('wangbin\_test');

            client.subscribe('wangbin\_test2');

            //client.end();

            }

    );

    client.on('subscribe',

        function(channel,count){

            console.log("channel:" + channel + ", count:"+count);

            }

    );

    client.on('message',

        function(channel,message){

            console.log("channel:" + channel + ", msg:"+message);

            }

    );

    client.on('unsubscribe',

        function(channel,count){

            console.log("channel:" + channel + ", count:"+count);

            }

    );

}

catch(e){

        console.log("err:"+e);

}

测试结果：

[Redis <wbr>pub/sub(Publish,Subscribe)](http://photo.blog.sina.com.cn/showpic.html#blogid=62b832910100xok2&url=http://s16.sinaimg.cn/orignal/62b83291tbb66cf69347f)

另外还支持：pmessage, psubscribe, punsubscribe 事件监听；

---------------------------------------------------------------------------------------------------------

**Node.js  Socket.io redis pub/sub主动推送数据到浏览器前端**

另外还支持：pmessage, psubscribe, punsubscribe 事件监听；

**Server.js**

var app = require('http').createServer(handler)

  , io = require('socket.io').listen(app)

  , fs = require('fs');

var redis = require('redis');

var redis\_client  = redis.createClient(6380,'10.54.37.212');

app.listen(3000);

function handler (req, res) {

  fs.readFile(\_\_dirname + '/index.html',

  function (err, data) {

    if (err) {

      res.writeHead(500);

      return res.end('Error loading index.html');

    }

    res.writeHead(200);

    res.end(data);

  });

}

io.sockets.on('connection', function (socket) {

  socket.emit('news', { hello: 'world' });

      redis\_client.subscribe('wangbin\_test');

      redis\_client.on('message',

        function(channel,message){

             socket.emit('news', message);

             console.log("channel:" + channel + ", msg:"+message);

         }

         );

  socket.on('my other event', function (data) {

    console.log(data);

  });

});

**Index.html**

<script src="/socket.io/socket.io.js"></script>

<script>

  var socket = io.connect('http://10.54.37.212');

    socket.on('news', function (data) {

        alert(data);

                socket.emit('my other event', { my: 'data' });

                  });

</script>

Redis-cli发送消息：

Publish wangbin\_test test

Server.js订阅监控收到消息后通过socket.io主动push到brower

浏览器收到消息

[](http://photo.blog.sina.com.cn/showpic.html#blogid=62b832910100xok2&url=http://s3.sinaimg.cn/orignal/62b83291tbb66d0a28fa2)

**7、 Redis pub/sub压力测试**

redis-benchmark是redis的读写性能工具

生成测试数据源：

for  i  in `seq 10000000` ;do echo  -ne  'publish   wangbin\_test "'$i'"\r\n' >> publish.data; done;

           发布：

for i in $(seq 10);do (cat publish.data;sleep 100;)  | nc 10.54.37.212  6380  ;done;

订阅（模拟多个客户端同时订阅）

(echo -ne 'subscribe wangbin\_test\r\n';sleep 10000)  | nc 10.54.37.212 6380 | grep  "\".\*\"" > subscribe\_01.txt  &

(echo -ne 'subscribe wangbin\_test\r\n';sleep 10000)  | nc 10.54.37.212 6380 | grep  "\".\*\"" > subscribe\_02.txt  &

(echo -ne 'subscribe wangbin\_test\r\n';sleep 10000)  | nc 10.54.37.212 6380 | grep  "\".\*\"" > subscribe\_03.txt  &

(echo -ne 'subscribe wangbin\_test\r\n';sleep 10000)  | nc 10.54.37.212 6380 | grep  "\".\*\"" > subscribe\_04.txt  &

(echo -ne 'subscribe wangbin\_test\r\n';sleep 10000)  | nc 10.54.37.212 6380 | grep  "\".\*\"" > subscribe\_05.txt  &

(echo -ne 'subscribe wangbin\_test\r\n';sleep 10000)  | nc 10.54.37.212 6380 | grep  "\".\*\"" > subscribe\_06.txt  &  
     性能监控测试

watch -d -n 1 'wc -l  subscribe\_\*.txt '

或者：

for i in $(seq 15); do wc -l subscribe\_\*.txt > result\_$i.txt ;sleep 1;  done;

测试结果为：

每秒返回的数据为4-5w/s

**8、 Redis pub/sub服务应用场景分析**

其他PECL 扩展[SAM]Simple Asynchronous Messaging

**9、 附件：**

**i.**           **Redis-cli命令信息：**

  -h <hostname>    域名或ip

  -p <port>        端口号 (default: 6379)

  -s <socket>      例如：/usr/local/bin/redis-cli  -s  /tmp/redis.sock

  -a <password>   密码

  -r <repeat>      重复次数

  -i <interval>         重复时间隔时间

  -n <db>         数据库个数

  -x              最后的标准输入

  -d <delimiter>    分隔符

  --raw           原始格式回复

  --latency        采集延迟模式

[](http://photo.blog.sina.com.cn/showpic.html#blogid=62b832910100xok2&url=http://s16.sinaimg.cn/orignal/62b83291tbb66d3d02ddf)

**参考：**

[1] http://redis.io/topics/pubsub

[2] http://code.google.com/p/redis/wiki/PublishSubscribe

[3] <http://xmpp.org/extensions/xep-0060.html>

[4] <http://www.php.net/manual/en/book.sam.php>

[5]<https://github.com/mranney/node_redis>重要  
[6] <http://socket.io/>

[7] https://github.com/LearnBoost/socket.io

[8] https://github.com/LearnBoost/socket.io