

Mycat

一、什么是 Mycat

基于阿里开源的 Cobar 产品而研发，Cobar 的稳定性、可靠性、优秀的架构和性能以及众多成熟的使用案例使得 MYCAT 一开始就拥有一个很好的起点，站在巨人的肩膀上，我们能看到更远。业界优秀的开源项目和创新思路被广泛融入到 MYCAT 的基因中，使得 MYCAT 在很多方面都领先于目前其他一些同类的开源项目，甚至超越某些商业产品。MYCAT 背后有一支强大的技术团队，其参与者都是 5 年以上资深软件工程师、架构师、DBA 等，优秀的技术团队保证了 MYCAT 的产品质量。MYCAT 并不依托于任何一个商业公司，因此不像某些开源项目，将一些重要的特性封闭在其商业产品中，使得开源项目成了一个摆设。

二、Mycat 的下载与安装

1. 从官网下载 mycat

```
wget http://dl.mycat.io/1.6-RELEASE/Mycat-server-1.6-RELEASE-20161028204710-linux.tar.gz
```

2. 解压 Mycat 压缩包

```
[root@hadoop ~]# ls
mycat  Mycat-server-1.6-RELEASE-20161028204710-linux.tar.gz
```

3. 移动到 usr 目录中

```
[root@hadoop ~]# mv mycat/ /usr/
```

4. 配置 mycat 环境变量

```
[root@hadoop mycat]# vim /etc/profile
```

```
export JAVA_HOME=/usr/java/jdk1.7.0_79
export HADOOP_HOME=/opt/hadoop
export MAVEN_HOME=/opt/maven
export FINDBUGS_HOME=/opt/findbugs-1.3.9
export MYCAT_HOME=/usr/mycat
export PATH=$JAVA_HOME/bin:$MAVEN_HOME/bin:$FINDBUGS_HOME/bin:$MYCAT_HOME/bin:$PATH
```

5. 配置 mycat 中 conf 下的配置 schema.xml

```
[root@hadoop mycat]# vim conf/schema.xml
```

```

<mycat:schema xmlns:mycat="http://io.mycat/">
  <!-- 定义MyCat的逻辑库 -->
  <schema name="test_schema" checkSQLschema="false" sqlMaxLimit="100" dataNode="testNode"></schema>
  <!-- 定义MyCat的数据节点 -->
  <dataNode name="testNode" dataHost="dtHost" database="test" />
  <dataHost name="dtHost" maxCon="1000" minCon="10" balance="1"
    writeType="0" dbType="mysql" dbDriver="native" switchType="-1" slaveThreshold="100">
    <heartbeat>select user()</heartbeat>
    <!-- can have multi write hosts -->
    <writeHost host="hostM1" url="192.168.28.128:3306" user="root"
      password="root">
      <!-- can have multi read hosts -->
      <readHost host="hostS1" url="192.168.28.129:3306" user="root" password="root" />
    </writeHost>
  </dataHost>
</mycat:schema>

```

```

<!-- 定义 MyCat 的逻辑库 -->
  <schema name="test_schema" checkSQLschema="false" sqlMaxLimit="100"
dataNode="testNode"></schema>
  <!-- 定义 MyCat 的数据节点 -->
  <dataNode name="testNode" dataHost="dtHost" database="test" />
  <dataHost name="dtHost" maxCon="1000" minCon="10" balance="1"
    writeType="0" dbType="mysql" dbDriver="native"
switchType="-1" slaveThreshold="100">
    <heartbeat>select user()</heartbeat>
    <!-- can have multi write hosts -->
    <writeHost host="hostM1" url="192.168.28.128:3306" user="root"
      password="root">
      <!-- can have multi read hosts -->
      <readHost host="hostS1" url="192.168.28.129:3306" user="root"
password="root" />
    </writeHost>
  </dataHost>

```

6. 配置登陆 mycat 的权限 server.xml

```
[root@hadoop mycat]# vim conf/server.xml
```

```

<system>
  <!-- 这里配置的都是一些系统属性，可以自己查看mycat文档 -->
  <property name="defaultSqlParser">druidparser</property>
  <property name="charset">utf8mb4</property>
</system>

<user name="root">
  <property name="password">root</property>
  <property name="schemas">test_schema</property>
</user>

```

```

<system>
  <!-- 这里配置的都是一些系统属性，可以自己查看 mycat 文-->
  <property name="defaultSqlParser">druidparser</property>
  <property name="charset">utf8mb4</property>
</system>

```

```
<user name="root">
    <property name="password">root</property>
    <property name="schemas">test_schema</property>
</user>
```

7. 修改日志文件 log4j2.xml

```
[root@hadoop mycat]# vim conf/log4j2.xml
```

```
<asyncRoot level="DEBUG" includeLocation="true">
    <AppenderRef ref="Console" />
    <AppenderRef ref="RollingFile"/>
</asyncRoot>
<AsyncLogger name="io.mycat" level="DEBUG" includeLocation="true" additivity="false">
    <AppenderRef ref="Console"/>
    <AppenderRef ref="RollingFile"/>
</AsyncLogger>
```

```
<asyncRoot level="DEBUG" includeLocation="true">
    <AppenderRef ref="Console" />
    <AppenderRef ref="RollingFile"/>
</asyncRoot>
<AsyncLogger name="io.mycat" level="DEBUG" includeLocation="true"
additivity="false">
    <AppenderRef ref="Console"/>
    <AppenderRef ref="RollingFile"/>
</AsyncLogger>
```

8. 启动 mycat

```
[root@hadoop mycat]# mycat start
```

9. 查看 mycat 日志并测试

```
[root@hadoop conf]# tail -f ../logs/mycat.log
2018-04-06 23:01:09.915 DEBUG release channel
```

Nginx

三、 Nginx 是什么

Nginx 是一款轻量级的 [Web](#) 服务器/[反向代理](#)服务器及[电子邮件](#)（IMAP/POP3）代理服务器，并在一个 BSD-like 协议下发行。由俄罗斯的程序设计师 Igor Sysoev 所开发，供俄国

大型的入口网站及搜索引擎 Rambler（俄文：Рамблер）使用。其特点是占有内存少，[并发](#)能力强，事实上 nginx 的并发能力确实在同类型的网页服务器中表现较好，中国大陆使用 nginx 网站用户有：[京东](#)、[新浪](#)、[网易](#)、[腾讯](#)、[淘宝](#)等。

优点:

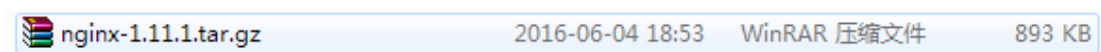
Nginx 可以在大多数 Unix like OS 上编译运行，并有 Windows 移植版。Nginx 的 1.4.0 稳定版已经于 2013 年 4 月 24 日发布，一般情况下，对于新建站点，建议使用最新稳定版作为生产版本，已有站点的升级紧迫性不高。Nginx 的源代码使用 2-clause BSD-like license。

Nginx 是一个很强大的高性能 Web 和反向代理服务器，它具有很很多非常优越的特性：

在高连接并发的情况下，Nginx 是 Apache 服务器不错的替代品：Nginx 在美国是做虚拟主机生意的老板们经常选择的软件平台之一。能够支持高达 50,000 个并发连接数的响应。

四、 Nginx 的安装

10. 从官网下载一个 nginx 的 tar.gz 版。



11. 第一步：解压 tar -zxvf nginx-1.7.4.tar.gz

```
[root@linux ~]# ls
nginx-1.11.1.tar.gz  tomcat
[root@linux ~]# tar -zxvf nginx-1.11.1.tar.gz
[root@linux ~]# ls
nginx-1.11.1  nginx-1.11.1.tar.gz  tomcat
```

12. 第二步：进入 nginx-1.7.4，执行 ./configure --prefix=/usr/nginx 命令

```
[root@linux ~]# ls
nginx-1.11.1  nginx-1.11.1.tar.gz  tomcat
[root@linux ~]# cd nginx-1.11.1
[root@linux nginx-1.11.1]# ./configure
```

13. 出现错误需要安装 gcc

```
checking for OS
+ Linux 2.6.32-431.el6.x86_64 x86_64
checking for C compiler ... not found
./configure: error: C compiler cc is not found
```

出现这个错误。

那么就是gcc 包没有安装。安装gcc 吧，骚年。

```
yum -y install gcc
```

14. 安装 gcc 依赖

```
rpm -Uvh libgcc-4.4.7-17.el6.i686.rpm tzdata-2016d-1.el6.noarch.rpm
rpm -ivh kernel-headers-2.6.32-642.1.1.el6.i686.rpm
rpm -Uvh glibc-common-2.12-1.192.el6.i686.rpm glibc-2.12-1.192.el6.i686.rpm
rpm -ivh glibc-headers-2.12-1.192.el6.i686.rpm
glibc-devel-2.12-1.192.el6.i686.rpm mpfr-2.4.1-6.el6.i686.rpm cpp-4.4.7-17.el6.i686.rpm
ppl-0.10.2-11.el6.i686.rpm cloog-ppl-0.15.7-1.2.el6.i686.rpm
libgomp-4.4.7-17.el6.i686.rpm gcc-4.4.7-17.el6.i686.rpm
```

15. 继续编译报错需要安装 pcre-devel

再次执行./configure

```
./configure: error: the HTTP rewrite module requires the PCRE library.
You can either disable the module by using --without-http_rewrite_module
option, or install the PCRE library into the system, or build the PCRE library
statically from the source with nginx by using --with-pcre=<path> option.
```

```
yum install pcre-devel
```

安装 pcre-devel

```
rpm -Uvh pcre-7.8-7.el6.i686.rpm
rpm -ivh pcre-devel-7.8-7.el6.i686.rpm
```

16. 继续编译报错需要安装 zlib-devel

再次执行./configure

```
./configure: error: the HTTP gzip module requires the zlib library.
```

You can either disable the module by using `--without-http_gzip_module` option, or install the zlib library into the system, or build the zlib library statically from the source with nginx by using `--with-zlib=<path>` option.

```
yum install zlib-devel
```

安装 zlib `rpm -ivh *.rpm`

17. 编译通过执行 `make` 命令

18. 执行 `make install` 命令

五、 Nginx 的负载均衡实现


1. 启动 nginx 服务器

```
./nginx -c /usr/local/nginx/conf/nginx.conf
```

2. 关闭

```
./nginx -s stop
```

3. 准备多台 tomcat



```
linux ~]# ls
tomcat1 tomcat2 tomcat3
linux ~]#
```

4. 分别修改 tomcat 服务器的端口

第一个修改位置端口号

```
Documentation at /docs/config/server.html
-->
<Server port="8004" shutdown="SHUTDOWN">
  <Listener className="org.apache.catalina.startup.VersionLoggerListener" />
  <!-- Security listener. Documentation at /docs/config/listeners.html
  <Listener className="org.apache.catalina.security.SecurityListener" />
-->
```

第二个修改端口位置

```
Define a non-SSL HTTP/1.1 Connector on port 8080
-->
<Connector port="8989" protocol="HTTP/1.1"
  connectionTimeout="20000"
  redirectPort="8443" />
<!-- A "Connector" using the shared thread pool-->
<!--
<Connector executor="tomcatThreadPool"
  port="8080" protocol="HTTP/1.1"
```

第三个修改位置

```
-->
<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector port="10010" protocol="AJP/1.3" redirectPort="8443" />
```

注意:在一台机器上开启多个 tomcat 以上三个端口都不能相同

5. 在 nginx 中配置

```
upstream tomcat-servers {
    #ip_hash;
    server 192.168.1.116:8989;
    server 192.168.1.116:8990;
    server 192.168.1.116:8991;
}
```

```
upstream tomcat-servers {
    #ip_hash;
    server 192.168.1.116:8989;
    server 192.168.1.116:8990;
    server 192.168.1.116:8991;
}
```

跳转路径

location / {

```

proxy_pass http://tomcat-servers;
proxy_redirect    off;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header Host $http_host;
proxy_next_upstream http_502 http_504 error timeout invalid_header;
}

```

```

location ~ .*\. (php|jsp|cgi)?$ {
    proxy_pass http://tomcat-servers;
    proxy_redirect    off;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header Host $http_host;
    proxy_next_upstream http_502 http_504 error timeout invalid_header;
}

```

六、 负载均衡策略

1. 轮询
2. Ip_hash
3. weight

七、 MSM(Memcache Session Manager)session 管理

1. 安装 memcache









名称	修改日期	类型	大小
 libevent-1.4.13-4.el6.i686.rpm	2012-06-25 6:06	媒体文件 (.rpm)	67 KB
 memcached-1.4.4-3.el6.i686.rpm	2011-07-03 12:29	媒体文件 (.rpm)	67 KB

```

[root@linux ~]# rpm -ivh *.rpm
Preparing...                               ##### [100%]
 1:libevent                               ##### [ 50%]
 2:memcached                               ##### [100%]

```

2. 导入 jar

 spymemcached-2.7.1.jar	2014-07-19 12:02	Executable Jar File	386 KB
 reflectasm-1.01.jar	2014-07-19 12:04	Executable Jar File	12 KB
 msm-kryo-serializer-1.6.0.jar	2014-07-19 12:58	Executable Jar File	23 KB
 minlog-1.2.jar	2014-07-19 12:06	Executable Jar File	5 KB
 memcached-session-manager-tc7-1.6.1.jar	2014-07-19 12:51	Executable Jar File	11 KB
 memcached-session-manager-1.6.1.jar	2014-07-19 12:49	Executable Jar File	125 KB
 kryo-1.04.jar	2014-07-19 12:07	Executable Jar File	93 KB
 asm-3.2.jar	2014-07-19 13:00	Executable Jar File	43 KB


```
[root@linux ~]# ls tomcat1/lib/
annotations-api.jar          msm-kryo-serializer-1.6.0.jar
asm-3.2.jar                  reflectasm-1.01.jar
catalina-ant.jar             servlet-api.jar
catalina-ha.jar              spymemcached-2.7.1.jar
catalina.jar                 tomcat7-websocket.jar
catalina-tribes.jar          tomcat-api.jar
ecj-4.4.2.jar                tomcat-coyote.jar
el-api.jar                   tomcat-dbcp.jar
jasper-el.jar                tomcat-i18n-es.jar
jasper.jar                   tomcat-i18n-fr.jar
jsp-api.jar                  tomcat-i18n-ja.jar
kryo-1.04.jar                tomcat-jdbc.jar
memcached-session-manager-1.6.1.jar tomcat-util.jar
memcached-session-manager-tc7-1.6.1.jar websocket-api.jar
minlog-1.2.jar
```

3. 启动 memcache 服务器

Memcached -p 11211 -vvv -u root

```
[root@linux ~]# memcached -p 11211 -vvv -u root
slab class 1: chunk size 80 perslab 13107
slab class 2: chunk size 104 perslab 10082
slab class 3: chunk size 136 perslab 7710
slab class 4: chunk size 176 perslab 5957
slab class 5: chunk size 224 perslab 4681
slab class 6: chunk size 280 perslab 3744
slab class 7: chunk size 352 perslab 2978
slab class 8: chunk size 440 perslab 2383
slab class 9: chunk size 552 perslab 1899
slab class 10: chunk size 696 perslab 1506
```

4. 配置 tomcat 目录中 conf 目录中 context.xml 文件加入如下配置

```
<!--
<Manager className="de.javakaffee.web.msm.MemcachedBackupSessionManager"
          memcachedNodes="nl:localhost:11211"
          sticky="false"
          sessionBackupAsync="false"
          lockingMode="auto"
          requestUriIgnorePattern=".*\.(ico|png|gif|jpg|css|js)$"
          transcoderFactoryClass="de.javakaffee.web.msm.JavaSerializationTranscode
Factory"
/>
</Context>
```

5. 重启 tomcat 测试

八、 Nginx 的动静分离

1. 配置如下:

动态资源应用服务器获取

```
location ~.*\.(php|jsp|cgi)?$ {
    proxy_pass http://tomcat-servers;
    proxy_redirect off;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Real-IP $remote_addr;
```

```

        proxy_set_header Host $http_host;
        proxy_next_upstream http_502 http_504 error timeout invalid_header;
    }
}

```

```

#动态资源请求web服务器
location ~ .*\. (php|jsp|cgi)?$ {
    proxy_pass http://tomcat-servers;
    proxy_redirect off;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header Host $http_host;
    proxy_next_upstream http_502 http_504 error timeout invalid_header;
}

```

静态资源 nginx 获取

#配置 Nginx 动静分离，定义的静态页面直接从 Nginx 发布目录读取。

```

    location ~ .*\. (html|htm|gif|jpg|jpeg|bmp|png|ico|txt|js|css)$ {
        root /usr/local/shop/html;将工程放入文件夹中
        #expires 定义用户浏览器缓存的时间为 3 天，如果静态页面不常更新，可以设置更长，
        #这样可以节省带宽和缓解服务器的压力
        expires 3d;
    }
}

```

```

#配置Nginx动静分离，定义的静态页面直接从Nginx发布目录读取。
location ~ .*\. (html|htm|gif|jpg|jpeg|bmp|png|ico|txt|js|css)$ {
    root shop/html;
    #expires定义用户浏览器缓存的时间为3天，如果静态页面不常更新，可以设置更长，
    #这样可以节省带宽和缓解服务器的压力
    expires 3d;
}

```