## 一、MyCat数据库中间件

### 1、简介（此文档搭的是伪集群，只有一个mysql服务器）

1.1、什么是MyCat？

MyCat是一个开源的分布式数据库系统，是一个实现了MySQL协议的服务器，前端用户可以把它看作是一个数据库代理，用MySQL客户端工具和命令行访问，而其后端可以用MySQL原生协议与多个MySQL服务器通信，也可以用JDBC协议与大多数主流数据库服务器通信，其核心功能是分表分库，即将一个大表水平分割为N个小表，存储在后端MySQL服务器里或者其他数据库里。

MyCat发展到目前的版本，已经不是一个单纯的MySQL代理了，它的后端可以支持MySQL、SQL Server、Oracle、DB2、PostgreSQL等主流数据库，也支持MongoDB这种新型NoSQL方式的存储，未。来还会支持更多类型的存储。而在最终用户看来，无论是那种存储方式，在MyCat里，都是一个传统的数据库表，支持标准的SQL语句进行数据的操作，这样一来，对前端业务系统来说，可以大幅降低开发难度，提升开发速度

1.2、那么为什么要用到MyCat呢？

例如操作系统是对各类计算机硬件的抽象。那么我们什么时候需要抽象？假如只有一种硬件的时候，我们需要开发一个操作系统吗？

再比如一个项目只需要一个人完成的时候不需要leader，但是当需要几十人完成时，就应该有一个管理者，发挥沟通协调等作用，而这个管理者对于他的上层来说就是对项目组的抽象。

同样的，当我们的应用只需要一台数据库服务器的时候我们并不需要Mycat，而如果你需要分库甚至分表，这时候应用要面对很多个数据库的时候，这个时候就需要对数据库层做一个抽象，来管理这些数据库，而最上面的应用只需要面对一个数据库层的抽象或者说数据库中间件就好了，这就是Mycat的核心作用。

所以可以这样理解：数据库是对底层存储文件的抽象，而Mycat是对数据库的抽象。

### 2、安装步骤

（1）将MySql的服务端和客户端安装包（RPM）放到服务器

（2）先查看之前是否安装过

rpm -qa|grep -i mysql

（3）若安装过，卸载掉：

rpm -e --nodeps  软件名称

（4）yum方式下载并安装命令：

wget -i -c http://dev.mysql.com/get/mysql57-community-release-el7-10.noarch.rpm

yum -y install mysql57-community-release-el7-10.noarch.rpm

yum -y install mysql-community-server

（5）启动mySql服务

service mysql start

Centos7启动：systemctl start mysqld.service

查看mysql运行状态：systemcrl status mysqld.service

重启命令：systemctl

关闭命令：systemctl stop mysqld.service

查看临时密码：grep “password” /var/log/mysqld.log

（6）登录mysql

mysql -u root -p

（7）修改密码

先修改两个全局参数：

validate\_password\_policy代表密码策略，默认是1：符合长度，且必须含有数字，小写或大写字母，特殊字符。设置为0判断密码的标准就基于密码的长度了。一定要先修改两个参数再修改密码

set global validate\_password\_policy=0;

set global validate\_password\_length=4;

命令：ALTER USER 'root'@'localhost' IDENTIFIED BY '123456';

（8）设置远程登录权限：

GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'%'IDENTIFIED BY '123456' WITH GRANT OPTION;

（9）上传mycat压缩包到服务器并解压（mycat1.4无法看到数据库表，所以要使用1.4以上的版本）

tar zxvf Mycat-server\*\*\*.tar.gz

（10）将解压后的文件移到/usr/local/中

mv mycat /usr/local

（11）进到mycat/bin,可进行如下命令

启动：./mycat start

关闭：./mycat stop

mycat支持的命令[console|start|stop|restart|status|dump]

mycat的默认端口号：8066

### 3、分片配置：（mycat和mysql可以不在一台机器，mycat可以远程连接mysql）

（1）修改mycat/conf/schema.xml文件配置：

schema.xml作为MyCat中重要的配置文件之一，管理着MyCat的逻辑库、逻辑表以及对应的分片规则、DataNode以及DataSource。弄懂这些配置，是正确使用MyCat的前提。这里就一层层对该文件进行解析。

schema 标签用于定义MyCat实例中的逻辑库

Table 标签定义了MyCat中的逻辑表 rule用于指定分片规则，auto-sharding-long的分片规则是按ID值的范围进行分片 1-5000000 为第1片 5000001-10000000 为第2片.... 具体设置我们会在第5小节中讲解。

dataNode 标签定义了MyCat中的数据节点，也就是我们通常说所的数据分片。

dataHost标签在mycat逻辑库中也是作为最底层的标签存在，直接定义了具体的数据库实例、读写分离配置和心跳语句。

原配置如下：

|  |
| --- |
| <!—name是逻辑库的名称-->  <schema name="TESTDB" checkSQLschema="false" sqlMaxLimit="100">  <!-- auto sharding by id (long) -->  <!—逻辑表 name是逻辑表的名称,dataNodes是数据节点,rule是分片规则  auto-sharding-long是按id值划分的分片规则，每500万为一个分片  -->  <table name="travelrecord" dataNode="dn1,dn2,dn3" rule="auto-sharding-long" />  <!-- global table is auto cloned to all defined data nodes ,so can join with any table whose sharding node is in the same data node -->  <table name="company" primaryKey="ID" type="global" dataNode="dn1,dn2,dn3" />  <table name="goods" primaryKey="ID" type="global" dataNode="dn1,dn2" />  <!-- random sharding using mod sharind rule -->  <table name="hotnews" primaryKey="ID" dataNode="dn1,dn2,dn3" rule="mod-long" />  <!-- <table name="dual" primaryKey="ID" dataNode="dnx,dnoracle2" type="global" needAddLimit="false"/>  <table name="worker" primaryKey="ID" dataNode="jdbc\_dn1,jdbc\_dn2,jdbc\_dn3" rule="mod-long" /> -->  <table name="employee" primaryKey="ID" dataNode="dn1,dn2"  rule="sharding-by-intfile" />  <table name="customer" primaryKey="ID" dataNode="dn1,dn2"  rule="sharding-by-intfile">  <childTable name="orders" primaryKey="ID" joinKey="customer\_id"  parentKey="id">  <childTable name="order\_items" joinKey="order\_id" parentKey="id" />  </childTable>  <childTable name="customer\_addr" primaryKey="ID" joinKey="customer\_id"  parentKey="id" />  </table>  <!-- <table name="oc\_call" primaryKey="ID" dataNode="dn1$0-743" rule="latest-month-calldate"  /> -->  </schema>  <!—  <dataNode name="dn1$0-743" dataHost="localhost1" database="db$0-743"/>  -->  <!—dataHost是节点主机，database是物理库的库名-->  <dataNode name="dn1" dataHost="localhost1" database="db1" />  <dataNode name="dn2" dataHost="localhost1" database="db2" />  <dataNode name="dn3" dataHost="localhost1" database="db3" />  <!—  <dataNode name="dn4" dataHost="sequoiadb1" database="SAMPLE" />  <dataNode name="jdbc\_dn1" dataHost="jdbchost" database="db1" />  <dataNode name="jdbc\_dn2" dataHost="jdbchost" database="db2" />  <dataNode name="jdbc\_dn3" dataHost="jdbchost" database="db3" />  -->  <dataHost name="localhost1" maxCon="1000" minCon="10" balance="0"  writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">  <heartbeat>select user()</heartbeat>  <!-- can have multi write hosts -->  <writeHost host="hostM1" url="localhost:3306" user="root"  password="123456">  <!-- can have multi read hosts -->  </writeHost>  <writeHost host="hostS1" url="localhost:3316" user="root"  password="123456" />  <!-- <writeHost host="hostM2" url="localhost:3316" user="root" password="123456"/> -->  </dataHost>  <!--  <dataHost name="sequoiadb1" maxCon="1000" minCon="1" balance="0" dbType="sequoiadb" dbDriver="jdbc">  <heartbeat> </heartbeat>  <writeHost host="hostM1" url="sequoiadb://1426587161.dbaas.sequoialab.net:11920/SAMPLE" user="jifeng" password="jifeng"></writeHost>  </dataHost>    <dataHost name="oracle1" maxCon="1000" minCon="1" balance="0" writeType="0" dbType="oracle" dbDriver="jdbc"> <heartbeat>select 1 from dual</heartbeat>  <connectionInitSql>alter session set nls\_date\_format='yyyy-mm-dd hh24:mi:ss'</connectionInitSql>  <writeHost host="hostM1" url="jdbc:oracle:thin:@127.0.0.1:1521:nange" user="base" password="123456" > </writeHost> </dataHost>    <dataHost name="jdbchost" maxCon="1000" minCon="1" balance="0" writeType="0" dbType="mongodb" dbDriver="jdbc">  <!—心跳机制相关的语句,与主从有关联—>  <heartbeat>select user()</heartbeat>  <writeHost host="hostM" url="mongodb://192.168.0.99/test" user="admin" password="123456" ></writeHost> </dataHost>    <dataHost name="sparksql" maxCon="1000" minCon="1" balance="0" dbType="spark" dbDriver="jdbc">  <heartbeat> </heartbeat>  <!—hostM1是主节点, hostS1是从节点，可以引出主从复制的关系，不需要主从复制的话就可以直接写hostM1-->  <writeHost host="hostM1" url="jdbc:hive2://feng01:10000" user="jifeng" password="jifeng"></writeHost> </dataHost> -->  <!-- <dataHost name="jdbchost" maxCon="1000" minCon="10" balance="0" dbType="mysql"  dbDriver="jdbc"> <heartbeat>select user()</heartbeat> <writeHost host="hostM1"  url="jdbc:mysql://localhost:3306" user="root" password="123456"> </writeHost>  </dataHost> --> |

修改为：

|  |
| --- |
| <schema name="PINYOUGOUDB" checkSQLschema="false" sqlMaxLimit="100">  <!-- auto sharding by id (long) -->  <table name="tb\_test" dataNode="dn1,dn2,dn3" rule="auto-sharding-long" />    </schema>    <dataNode name="dn1" dataHost="localhost1" database="db1" />  <dataNode name="dn2" dataHost="localhost1" database="db2" />  <dataNode name="dn3" dataHost="localhost1" database="db3" />  <dataHost name="localhost1" maxCon="1000" minCon="10" balance="0"  writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">  <heartbeat>select user()</heartbeat>  <!-- can have multi write hosts -->  <writeHost host="hostM1" url="192.168.136.130:3306" user="root"  password="123456">  </writeHost>  </dataHost> |

（2）创建数据库

（3）修改mycat/conf/server.xml配置文件：

server.xml几乎保存了所有mycat需要的系统配置信息。最常用的是在此配置用户名、密码及权限。在system中添加UTF-8字符集设置，否则存储中文会出现问号

原始内容：

|  |
| --- |
| <mycat:server xmlns:mycat="http://org.opencloudb/">  <system>  <property name="defaultSqlParser">druidparser</property>  <!-- <property name="useCompression">1</property>--> <!--1为开启mysql压缩协议-->  <!-- <property name="processorBufferChunk">40960</property> -->  <!--  <property name="processors">1</property>  <property name="processorExecutor">32</property>  -->  <!--默认是65535 64K 用于sql解析时最大文本长度 -->  <!--<property name="maxStringLiteralLength">65535</property>-->  <!--<property name="sequnceHandlerType">0</property>-->  <!--<property name="backSocketNoDelay">1</property>-->  <!--<property name="frontSocketNoDelay">1</property>-->  <!--<property name="processorExecutor">16</property>-->  <!--  <property name="mutiNodeLimitType">1</property> 0：开启小数量级（默认） ；1：开启亿级数据排序  <property name="mutiNodePatchSize">100</property> 亿级数量排序批量  <property name="processors">32</property> <property name="processorExecutor">32</property>  <property name="serverPort">8066</property> <property name="managerPort">9066</property>  <property name="idleTimeout">300000</property> <property name="bindIp">0.0.0.0</property>  <property name="frontWriteQueueSize">4096</property> <property name="processors">32</property> -->  </system>  <user name="test">  <property name="password">test</property>  <property name="schemas">TESTDB</property>  </user>  <user name="user">  <property name="password">user</property>  <property name="schemas">TESTDB</property>  <property name="readOnly">true</property>  </user>  <!-- <cluster> <node name="cobar1"> <property name="host">127.0.0.1</property>  <property name="weight">1</property> </node> </cluster> -->  <!-- <quarantine> <host name="1.2.3.4"> <property name="user">test</property>  </host> </quarantine> -->  </mycat:server> |

修改为：

|  |
| --- |
| <mycat:server xmlns:mycat="http://org.opencloudb/">  <system>  <property name="defaultSqlParser">druidparser</property>  <!—设置编码集-->  <property name="charset">utf8</property>  <!-- <property name="useCompression">1</property>--> <!--1为开启mysql压缩协议-->  <!-- <property name="processorBufferChunk">40960</property> -->  <!--  <property name="processors">1</property>  <property name="processorExecutor">32</property>  -->  <!--默认是65535 64K 用于sql解析时最大文本长度 -->  <!--<property name="maxStringLiteralLength">65535</property>-->  <!--<property name="sequnceHandlerType">0</property>-->  <!--<property name="backSocketNoDelay">1</property>-->  <!--<property name="frontSocketNoDelay">1</property>-->  <!--<property name="processorExecutor">16</property>-->  <!--  <property name="mutiNodeLimitType">1</property> 0：开启小数量级（默认） ；1：开启亿级数据排序  <property name="mutiNodePatchSize">100</property> 亿级数量排序批量  <property name="processors">32</property> <property name="processorExecutor">32</property>  <property name="serverPort">8066</property> <property name="managerPort">9066</property>  <property name="idleTimeout">300000</property> <property name="bindIp">0.0.0.0</property>  <property name="frontWriteQueueSize">4096</property> <property name="processors">32</property> -->  </system>  <user name="test">  <property name="password">test</property>  <!—shcemas的值要与schema.xml中的name一致-->  <property name="schemas">PINYOUGOUDB</property>  </user>  <user name="root">  <property name="password">123456</property>  <property name="schemas">PINYOUGOUDB</property>  <property name="readOnly">true</property>  </user>  <!-- <cluster> <node name="cobar1"> <property name="host">127.0.0.1</property>  <property name="weight">1</property> </node> </cluster> -->  <!-- <quarantine> <host name="1.2.3.4"> <property name="user">test</property>  </host> </quarantine> -->  </mycat:server> |

（4）重启mycat，使配置生效：

./mycat restart

### 4、分片测试：

（1）添加数据后，mycat不承认select \* from 表名中的\*，所以查询时必须要写具体条件

（2）若Id超过1500万，如：15000001，数据不会插入到任意节点中，这时需要新建数据库去扩容。

### 5、分片规则：

在mycat/conf/rule.xml中查看规则内容。

（1）按主键范围分片：

在配置文件中我们找到

|  |
| --- |
| <tableRule name="auto-sharding-long">  <rule>  <columns>id</columns>  <algorithm>rang-long</algorithm>  </rule>  </tableRule> |

tableRule 是定义具体某个表或某一类表的分片规则名称 columns用于定义分片的列 algorithm代表算法名称 我们接着找rang-long的定义

|  |
| --- |
| <function name="rang-long"  class="org.opencloudb.route.function.AutoPartitionByLong">  <property name="mapFile">autopartition-long.txt</property>  </function> |

Function用于定义算法 mapFile 用于定义算法需要的数据，我们打开autopartition-long.txt

|  |
| --- |
| # range start-end ,data node index  # K=1000,M=10000.  0-500M=0  500M-1000M=1  1000M-1500M=2 |

（2）一致性哈希murmur

当我们需要将数据平均分在几个分区中，需要使用一致性hash规则

我们找到function的name为murmur 的定义，将count属性改为3，因为我要将数据分成3片

|  |
| --- |
| <function name="murmur"  class="org.opencloudb.route.function.PartitionByMurmurHash">  <property name="seed">0</property><!-- 默认是0 -->  <property name="count">3</property><!-- 要分片的数据库节点数量，必须指定，否则没法分片 -->  <property name="virtualBucketTimes">160</property><!-- 一个实际的数据库节点被映射为这么多虚拟节点，默认是160倍，也就是虚拟节点数是物理节点数的160倍 -->  <!-- <property name="weightMapFile">weightMapFile</property> 节点的权重，没有指定权重的节点默认是1。以properties文件的格式填写，以从0开始到count-1的整数值也就是节点索引为key，以节点权重值为值。所有权重值必须是正整数，否则以1代替 -->  <!-- <property name="bucketMapPath">/etc/mycat/bucketMapPath</property>  用于测试时观察各物理节点与虚拟节点的分布情况，如果指定了这个属性，会把虚拟节点的murmur hash值与物理节点的映射按行输出到这个文件，没有默认值，如果不指定，就不会输出任何东西 -->  </function> |

我们再配置文件中可以找到表规则定义

|  |
| --- |
| <tableRule name="sharding-by-murmur">  <rule>  <columns>id</columns>  <algorithm>murmur</algorithm>  </rule>  </tableRule> |

但是这个规则指定的列是id ,如果我们的表主键不是id ,而是order\_id ,那么我们应该重新定义一个tableRule：

|  |
| --- |
| <tableRule name="sharding-by-murmur-order">  <rule>  <columns>order\_id</columns>  <algorithm>murmur</algorithm>  </rule>  </tableRule> |

在schema.xml中配置逻辑表时，指定规则为sharding-by-murmur-order

|  |
| --- |
| <table name="tb\_order" dataNode="dn1,dn2,dn3" rule="sharding-by-murmur-order" /> |

# 二、Nginx配置

## 1、安装步骤：（实际生产中，前后端应该是分离的）

（1）安装gcc环境

yum install gcc-c++

（2）安装第三方开发包

PCRE:是一个Perl库，包括perl兼容的正则表达式库。

yum install -y pcre pcre-devel

zlib：提供了压缩和解压缩的方式。

yum install -y zlib zlib-devel

OpenSSL:安全套接字层密码库，囊括囊括饿了密码算法、常用的密钥和证书封装管理功能及SSL协议，并提供丰富的应用程序供测试或其他目的的石勇

yum install -y openssl openssl-devel

（3）上传nginx文件到服务器上，并解压。C++解压的是源码包，需要编译

tar zxvf nginx-\*\*\*.tar.gz

（4）进入nginx目录 使用 configure 命令创建一 makeFile 文件。

\代表换行符

|  |
| --- |
| ./configure \  --prefix=/usr/local/nginx \  --pid-path=/var/run/nginx/nginx.pid \  --lock-path=/var/lock/nginx.lock \  --error-log-path=/var/log/nginx/error.log \  --http-log-path=/var/log/nginx/access.log \  --with-http\_gzip\_static\_module \  --http-client-body-temp-path=/var/temp/nginx/client \  --http-proxy-temp-path=/var/temp/nginx/proxy \  --http-fastcgi-temp-path=/var/temp/nginx/fastcgi \  --http-uwsgi-temp-path=/var/temp/nginx/uwsgi \  --http-scgi-temp-path=/var/temp/nginx/scgi |

（5）编译：根据生成Makefile文件去编译的

make

（6）安装

make install

（7）创建目录

mkdir /var/temp/nginx/client -p

参数：

-p 表示如果nginx目录不存在，则创建目录

（8）在nginx/sbin目录下启动nginx，默认端口80

./nginx

（9）停止nginx

./nginx -s stop

（10）重启nginx

./nginx -s reload

## 2、部署项目

把前端文件放到nginx/html目录下即可

## 3、配置虚拟主机

3.1端口绑定

（1）把文件上传到/nginx下

（2）修改nginx.conf配置：

增加如下内容：（需要几个，增加几个）

|  |
| --- |
| server {  listen 81;  server\_name localhost;  location / {  root cart; #路径名称  index cart.html;  }  error\_page 500 502 503 504 /50x.html  location = /50x.html {  root html;  }  } |

（3）重启nginx

3.2域名绑定

注释：

一级域名：baidu.com

二级域名：map.baidu.com

三级域名：item.map.baidu.com

同上述修改方式，把server\_name修改成自己的域名即可，如果在目录中创建了其他其他配置，如：aa.conf，在nginx.conf的server中使用include aa.conf即可引入，如果有多个，可以用conf/\*.conf全部引入

## 4、反向代理和负载均衡

### 4.1 反向代理的定义：

正向代理就是一些局域网的机器通过代理服务器连接到外网服务器。

 反向代理（Reverse Proxy）方式是指以代理服务器来接受Internet上的连接请求，然后将请求转发给内部网络上的服务器；并将从服务器上得到的结果返回给Internet上请求连接的客户端，此时代理服务器对外就表现为一个服务器。

### 4.2 配置反向代理

1、准备工作：

（1）把tomcat放到服务器上，如果想在域名的根目录下访问页面，直接把ROOT文件夹下的文件清空，放页面进去即可。

（2）创建tomcat集群，并把tomcat复制过去

mkdir /usr/local/tomcat-cluster

cp -r apache-tomcat-7.0.75 /usr/local/tomcat-cluster/tomcat-1

（3）给复制的tomcat赋予权限

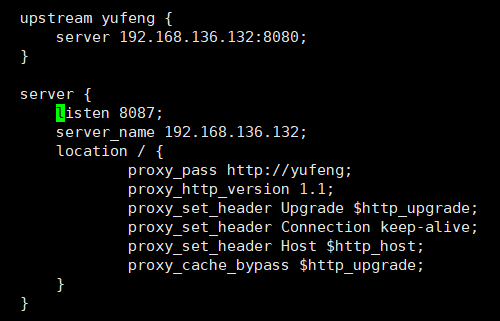
chmod -R 777 tomcat-cluster

（4）启动tomcat

./startup.sh

（5）启动成功后即可在浏览器查看页面，此后将配置反向代理，在nginx.conf中修改：

|  |
| --- |
| upstream 名称 {  server 192.168.136.132:8080;  }  server {  listen 80;  server\_name 域名;  location / {  proxy\_pass http://名称;  index index.html  }  } |

虚拟机配置：

### 4.3负载均衡

（1）修改集群中tomcat配置中的各个端口号

（2）启动所有的tomcat

（3）在nginx.conf的upstream中增加server ip:port；即可

（4）配置权重：（某台机器性能好,优先访问）

在server ip:port后增加weight=2;

### 4.4 高可用

keepalived防止单点故障，集群管理中保证集群高可用的一个服务软件。

## 5、nginx配置详情

下面给大家上一个配置文件，作为理解，同时也配入我搭建的一台测试机中，给大家示例。

[复制代码](javascript:void(0);)

########### 每个指令必须有分号结束。#################

#user administrator administrators; #配置用户或者组，默认为nobody nobody。

#worker\_processes 2; #允许生成的进程数，默认为1

#pid /nginx/pid/nginx.pid; #指定nginx进程运行文件存放地址

error\_log log/error.log debug; #制定日志路径，级别。这个设置可以放入全局块，http块，server块，级别以此为：debug|info|notice|warn|error|crit|alert|emerg

events {

accept\_mutex on; #设置网路连接序列化，防止惊群现象发生，默认为on

multi\_accept on; #设置一个进程是否同时接受多个网络连接，默认为off

#use epoll; #事件驱动模型，select|poll|kqueue|epoll|resig|/dev/poll|eventport

worker\_connections 1024; #最大连接数，默认为512

}

http {

include mime.types; #文件扩展名与文件类型映射表

default\_type application/octet-stream; #默认文件类型，默认为text/plain

#access\_log off; #取消服务日志

log\_format myFormat '$remote\_addr–$remote\_user [$time\_local] $request $status $body\_bytes\_sent $http\_referer $http\_user\_agent $http\_x\_forwarded\_for'; #自定义格式

access\_log log/access.log myFormat; #combined为日志格式的默认值

sendfile on; #允许sendfile方式传输文件，默认为off，可以在http块，server块，location块。

sendfile\_max\_chunk 100k; #每个进程每次调用传输数量不能大于设定的值，默认为0，即不设上限。

keepalive\_timeout 65; #连接超时时间，默认为75s，可以在http，server，location块。

upstream mysvr {

server 127.0.0.1:7878;

server 192.168.10.121:3333 backup; #热备

}

error\_page 404 https://www.baidu.com; #错误页

server {

keepalive\_requests 120; #单连接请求上限次数。

listen 4545; #监听端口

server\_name 127.0.0.1; #监听地址

location ~\*^.+$ { #请求的url过滤，正则匹配，~为区分大小写，~\*为不区分大小写。

#root path; #根目录

#index vv.txt; #设置默认页

proxy\_pass http://mysvr; #请求转向mysvr 定义的服务器列表

deny 127.0.0.1; #拒绝的ip

allow 172.18.5.54; #允许的ip

}

}

}

[复制代码](javascript:void(0);)

上面是nginx的基本配置，需要注意的有以下几点：

1、1.$remote\_addr 与$http\_x\_forwarded\_for 用以记录客户端的ip地址； 2.$remote\_user ：用来记录客户端用户名称； 3.$time\_local ： 用来记录访问时间与时区；4.$request ： 用来记录请求的url与http协议；

  5.$status ： 用来记录请求状态；成功是200， 6.$body\_bytes\_s ent ：记录发送给客户端文件主体内容大小；7.$http\_referer ：用来记录从那个页面链接访问过来的； 8.$http\_user\_agent ：记录客户端浏览器的相关信息；

2、惊群现象：一个网路连接到来，多个睡眠的进程被同事叫醒，但只有一个进程能获得链接，这样会影响系统性能。

3、每个指令必须有分号结束。