milkty

芝兰生于深谷，不以无人而不芳。君子修身养德，不以穷困而改志。

Nginx服务状态监控

转自：Nginx服务状态监控

在Nginx的插件模块中有一个模块stub_status可以监控Nginx的一些状态信息，默认安装可能没有这个模块，手动编译的时候加一下即可。

Nginx 还有一个商业版 Nginx Plus，功能更加丰富，对应的监控页面http://demo.nginx.com/status.html

1. 模块安装

先使用命令查看是否已经安装这个模块：

```
[root@ihxb123Z nginx]# ./nginx -V (V大写会显示版本号和模块等信息、v小写仅显示版本信息。

 或用此使用看：nginx -V 2>&1 | grep -o with-http_stub_status_module

如返回 with-http_stub_status_module，则说明该模块已被开放，而什么都不返回的话就是没有被开放。
```

如果已经安装，会在显示的信息中包含 --with-http_stub_status_module信息。如果没有此模块，需要重新安装，编译命令如下：

```
./configure –with-http_stub_status_module
```

2. Nginx配置

安装后只要修改nginx配置即可，配置如下：

```
location /hxbcdnstatus {
            stub_status            on;
            access_log             off;
           allow 127.0.0.1;
            deny all;
            #auth_basic              "NginxStatus";
            #auth_basic_user_file  conf/nginxstaus;
}
```

此处默认只有本地访问，如果远程可以查看需要加相关的IP或者干脆去掉Deny all即可。加密文件可以使用#htpasswd -c /usr/nginx/conf hxb 命令来创建。配置完成后需要重启Nginx服务。

状态配置只能是针对某个Nginx服务。目前Nginx还无法做到针对单个站点进行监控。

3. 状态查看

配置完成后在浏览器中输入http://127.0.0.1/hxbcdnstatus查看（或者用$ curl localhost/hxbcdnstatus），显示信息如下：

```
Active connections: 100
server accepts handled requests
 1075 1064 6253
Reading: 0 Writing: 5 Waiting: 95
```

Accepts（接受）、Handled（已处理）、Requests（请求数）是一直在增加的计数器。Active（活跃）、Waiting（等待）、Reading（读）、Writing（写）随着请求量而增减。

名称	描述	指标类型
Accepts（接受）	NGINX 所接受的客户端连接数	资源: 功能
Handled（已处理）	成功的客户端连接数	资源: 功能
Dropped（已丢弃，计算得出）	丢弃的连接数（接受 - 已处理）	工作：错误*
Requests（请求数）	客户端请求数	工作：吞吐量

4. 参数说明

active connections – 活跃的连接数量

server accepts handled requests — 总共处理了1075个连接 , 成功创建1064次握手, 总共处理了6253个请求

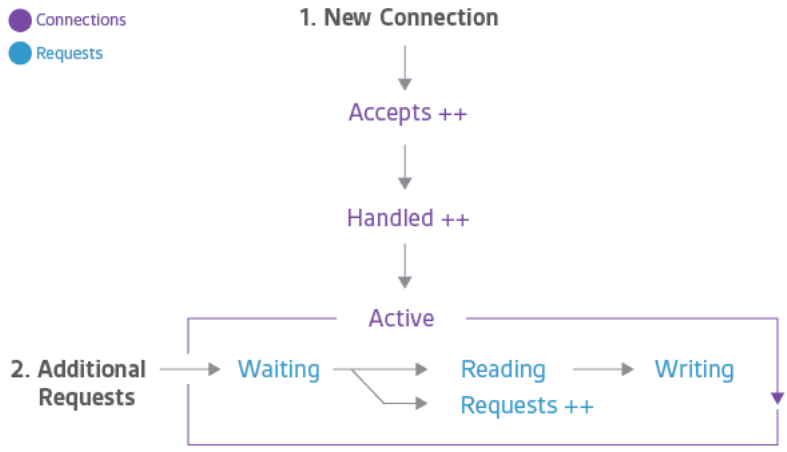
每个连接有三种状态waiting、reading、writing

reading —读取客户端的Header信息数.这个操作只是读取头部信息，读取完后马上进入writing状态，因此时间很短。

writing — 响应数据到客户端的Header信息数.这个操作不仅读取头部，还要等待服务响应，因此时间比较长。

waiting — 开启keep-alive后等候下一次请求指令的驻留连接.

正常情况下waiting数量是比较多的，并不能说明性能差。反而如果reading+writing数量比较多说明服务并发有问题。



当用户请求连接Nginx服务器时，accepts计数器会加一。且当服务器处理该连接请求时，handled计数器同样会加一。一般而言，两者的值是相等的，除非达到了某些资源极限（如worker_connection的限制）。

用户连接请求被处理，就会进入 active 状态。如果该连接没有其他 request，则进入 waiting 的子状态；如果有 request，nginx 会读取 request 的 header，计数器 request 加一，进入 reading 的子状态。 reading 状态持续时间非常短，header 被读取后就会进入 writing 状态。事实上，直到服务器将响应结果返回给用户之前，该连接会一直保持 writing 状态。所以说，writing 状态一般会被长时间占用。

一旦 NGINX 成功处理一个连接时，连接会移动到Active状态，在这里对客户端请求进行处理：

Active状态

> **Waiting**: 活跃的连接也可以处于 Waiting 子状态，如果有在此刻没有活跃请求的话。新连接可以绕过这个状态并直接变为到 Reading 状态，最常见的是在使用“accept filter（接受过滤器）” 和 “deferred accept（延迟接受）”时，在这种情况下，NGINX 不会接收 worker 进程的通知，直到它具有足够的数据才开始响应。如果连接设置为 keep-alive ，那么它在发送响应后将处于等待状态。
> **Reading**: 当接收到请求时，连接离开 Waiting 状态，并且该请求本身使 Reading 状态计数增加。在这种状态下 NGINX 会读取客户端请求首部。请求首部是比较小的，因此这通常是一个快速的操作。
> **Writing**: 请求被读取之后，其使 Writing 状态计数增加，并保持在该状态，直到响应返回给客户端。这意味着，该请求在 Writing 状态时， 一方面 NGINX 等待来自上游系统的结果（系统放在 NGINX “后面”），另外一方面，NGINX 也在同时响应。请求往往会在 Writing 状态花费大量的时间。

通常，一个连接在同一时间只接受一个请求。在这种情况下，Active 连接的数目 == Waiting 的连接 + Reading 请求 + Writing 。

5、怎么利用这些参数

开源的 Nginx 提供的原始参数中，实时性的会比较有用，如 Active connections、Reading、Writing 以及 Waiting。这些数据能够反映当前 Nginx 的负载情况，方便在服务器出现问题时及时发现问题。而另一些数据由于不是状态量，Nginx 无法计算当前的量值而改做其统计数，如 accepts、handled 和 requests。

中间件相关(17)
更多

随笔分类
java(1)
Jmeter(6)
LR(3)
办公(1)

随笔档案
2018年4月 (1)
2018年2月 (2)
2017年11月 (2)
2017年9月 (1)
2017年8月 (2)
2017年7月 (5)
2017年6月 (3)
2017年5月 (2)
2017年4月 (2)
2017年2月 (1)
2016年11月 (1)
2016年10月 (1)
2016年9月 (6)
2016年8月 (3)
2016年6月 (2)
2016年5月 (3)
2016年3月 (2)
2015年11月 (1)

对于维护网站人员，accepts、handled 和 requests 的统计值用处是不大的，值得参考的是短时间内这三者数值的增量。这个短时间可以是一秒，如 accepts_per_second、handled_per_second 和 requests_per_second。一个简单的做法就是每秒都去读取这些参数，返回一个和上一秒的差值就行。当然，handled_per_second 替换成 dropped_per_second=accepts_per_second-handled_per_second 就更完美了。

通过这七个参数，就可以从连接到请求全方位的监控起 Nginx 的运行状态。为了方便检测，对每次获取的参数保留下来，然后按时间展现出来。下图展示了 Nginx 在运行时的参考数据。

nginx折线图

补充：

查看Nginx并发进程数：ps -ef | grep nginx | wc -l

查看Web服务器TCP连接状态：netstat -n | awk '/^tcp/ {++S[$NF]} END {for(a in S) print a, S[a]}'

使用zabbix监控nginx和php-fpm性能

Zabbix监控Nginx基础活跃指标

其中截取了两个shell脚本。

/usr/local/zabbix/bin/nginx_status.sh

```
#!/bin/bash
#check nginx status
ip=127.0.0.1
function ping() {                                #用于检测nginx进程是否存在
    /sbin/pidof nginx | wc -l
}

function active() {                  #用于提取status中的active数值
    /usr/bin/curl http://$ip/nginx_status 2>/dev/null | sed -n '1p' | awk '{print $NF}'
}

function accepts() {             #用于提取status中的accepts数值
    /usr/bin/curl http://$ip/nginx_status 2>/dev/null | sed -n '3p' | awk '{print $1}'
}

function handled() {           #用于提取status中的handled数值
    /usr/bin/curl http://$ip/nginx_status 2>/dev/null | sed -n '3p' | awk '{print $2}'
}

function requests() {         #用于提取status中的request数值
    /usr/bin/curl http://$ip/nginx_status 2>/dev/null | sed -n '3p' | awk '{print $3}'
}

function reading() {         #用于提取status中的reading数值
    /usr/bin/curl http://$ip/nginx_status 2>/dev/null | sed -n '4p' | awk '{print $2}'
}

function writing() {         #用于提取status中的writing数值
```

2015年8月 (2)
2015年6月 (1)
2015年3月 (1)
2014年10月 (1)
2014年9月 (1)
2014年8月 (1)
2014年5月 (1)
2014年3月 (1)
2014年2月 (5)
2014年1月 (13)
2013年12月 (8)

文章分类
android(1)
java web(5)
linux(9)
python(1)
笔记(1)
测试工具(8)
持续集成
数据结构与算法(1)

友情链接
beigai
hualusiyu
itpub
lifetragedy
MKing's Blog

```
    /usr/bin/curl http://$ip/nginx_status 2>/dev/null | sed -n '4p' | awk '{print $4}'
}

function waiting() {     #用于提取status中的waiting数值
    /usr/bin/curl http://$ip/nginx_status 2>/dev/null | sed -n '4p' | awk '{print $6}'
}

$1                          #通过第一个位置参数的值来调用相应的函数
```

类似的

```
touch /home/nginx/ngx-status.sh
vim /home/nginx/ngx-status.sh

#!/bin/bash
############################nginx_check#############################
#                                V20170929                        #
#active|reading|writing|waiting|accepts|handled|requests|dropped#
############################nginx_check#############################

HOST="127.0.0.1"
PORT="80"

# 检测nginx相关参数
case $1 in
    ping)
    result=`/bin/pidof nginx 2>/dev/null| wc -l`
    echo $result
    ;;
    active)
    result=`/usr/bin/curl "http://$HOST:$PORT/ngx_status/" 2>/dev/null| grep 'Active' | awk '{print $NF}'`
    echo $result
    ;;
    reading)
    result=`/usr/bin/curl "http://$HOST:$PORT/ngx_status/" 2>/dev/null| grep 'Reading' | awk '{print $2}'`
    echo $result
    ;;
    writing)
    result=`/usr/bin/curl "http://$HOST:$PORT/ngx_status/" 2>/dev/null| grep 'Writing' | awk '{print $4}'`
    echo $result
    ;;
    waiting)
    result=`/usr/bin/curl "http://$HOST:$PORT/ngx_status/" 2>/dev/null| grep 'Waiting' | awk '{print $6}'`
    echo $result
    ;;
    accepts)
    result=`/usr/bin/curl "http://$HOST:$PORT/ngx_status/" 2>/dev/null| awk NR==3 | awk '{print $1}'`
    echo $result
    ;;
    handled)
    result=`/usr/bin/curl "http://$HOST:$PORT/ngx_status/" 2>/dev/null| awk NR==3 | awk '{print $2}'`
    echo $result
    ;;
    requests)
    result=`/usr/bin/curl "http://$HOST:$PORT/ngx_status/" 2>/dev/null| awk NR==3 | awk '{print $3}'`
    echo $result
    ;;
    dropped)
    result=`/usr/bin/curl "http://$HOST:$PORT/ngx_status/" 2>/dev/null| awk NR==3 | awk '{print $1-$2}'`
    echo $result
    ;;
    *)
     echo "Usage:$0(ping|active|reading|writing|waiting|accepts|handled|requests|dropped)"
    ;;
esac
```

PrefTest性能测试工作室
不胜人生一场醉
郭亨的博客
酷壳CoolShell - 陈皓
蓝冰咖啡
宋沄剑
淘测试
行知

最新评论

1. Re:linux命令——ll

@叶琦彰这个就是文件的
吧...

2. Re:linux命令——ll

应该是user而不是owne

3. Re:java命令--jstack

推荐：

4. Re:jmeter学习记录--(
生成报告

在win7环境下，执行-e
果index.html页面，jmet
x -l result.jtl -e -o /tmp1

5. Re:redis cli命令

十分详细

php-fpm status数值提取脚本为/usr/local/zabbix/bin/php_fpm_status.sh

```
#!/bin/bash
#check php-fpm status
case $1 in
    ping)                           #检测php-fpm进程是否存在
    /sbin/pidof php-fpm | wc -l
    ;;
    start_since)             #提取status中的start since数值
    /usr/bin/curl localhost/php_fpm-status 2>/dev/null  | awk 'NR==4{print $3}'
    ;;
    conn)                     #提取status中的accepted conn数值
    /usr/bin/curl localhost/php_fpm-status 2>/dev/null  | awk 'NR==5{print $3}'
    ;;
    listen_queue)         #提取status中的listen queue数值
    /usr/bin/curl localhost/php_fpm-status 2>/dev/null  | awk 'NR==6{print $3}'
    ;;
     max_listen_queue)  #提取status中的max listen queue数值
    /usr/bin/curl localhost/php_fpm-status 2>/dev/null  | awk 'NR==7{print $4}'
    ;;
    listen_queue_len)    #提取status中的listen queue len
    /usr/bin/curl localhost/php_fpm-status 2>/dev/null  | awk 'NR==8{print $4}'
    ;;
    idle_processes)      #提取status中的idle processes数值
    /usr/bin/curl localhost/php_fpm-status 2>/dev/null  | awk 'NR==9{print $3}'
    ;;
    active_processes)   #提取status中的active processes数值
    /usr/bin/curl localhost/php_fpm-status 2>/dev/null  | awk 'NR==10{print $3}'
    ;;
     total_processes)    #提取status中的total processess数值
    /usr/bin/curl localhost/php_fpm-status 2>/dev/null  | awk 'NR==11{print $3}'
    ;;
    max_active_processes)     #提取status中的max active processes数值
    /usr/bin/curl localhost/php_fpm-status 2>/dev/null  | awk 'NR==12{print $4}'
    ;;
     max_children_reached)    #提取status中的max children reached数值
    /usr/bin/curl localhost/php_fpm-status 2>/dev/null  | awk 'NR==13{print $4}'
    ;;
    slow_requests)   #提取status中的slow requests数值
    /usr/bin/curl localhost/php_fpm-status 2>/dev/null  | awk 'NR==14{print $3}'
    ;;
    *)
    echo "Usage: $0
{conn|listen_queue|max_listen_queue|listen_queue_len|idle_processes|active_processess|total_processes|max_active_pro
cesses|max_children_reached|slow_requests}"
    exit 1
    ;;
esac
```

标签: 中间件相关

milkty
关注 - 81
粉丝 - 91
+加关注

« 上一篇：linux命令--curl命令
» 下一篇：使用nginx作为HTTP负载均衡

posted @ 2017-09-06 09:06 milkty 阅读(1052) 评论(0) 编辑 收藏

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

相关博文：
· zabbix监控nginx服务状态
· nginx状态监控
· 【收录】Nginx 状态监控
· Zabbix 监控 Nginx 状态
· Nginx加状态监控

最新新闻：
· 永不造车？拆解华为拥有的自动驾驶和电动汽车关键技术
· 6天面试、斩获6家硅谷巨头Offer，我是如何做到的？
· IBM成Z代人最青睐科技公司 谷歌和亚马逊分列第二三名
· 为什么我们更像是在为抖音筛选内容，而非消费内容？
· 人人车变脸：曾经想干掉黄牛，如今成了“黄牛公司”
» 更多新闻...