链滴

OpenResty + Mysql 实现日志实时存储

作者：fc13240
原文链接：https://ld246.com/article/1543824685202
来源网站：链滴

原文链接：OpenResty + Mysql 实现日志实时存储

应用场景和日志文件解析

本配置主要解决 Nginx 向 mysql 中实时插入日志的问题，采用 OpenResty + Mysql 实现。

1. 刚开始的时候看了Nginx和mysql的连接模块。比如说nginx-mysql-module，可以连接mysql。是插入日志时遇到问题，我们知道nginx的执行过程先是location解析并重写阶段，然后是访问权限制阶段，接着是内容生成阶段，最后是日志记录阶段。mysql访问阶段属于内容生成阶段，所以代理行的时间和状态，mysql都无法获取的到。因此，这种通过nginx直连mysql的方式无法达到我们的要。

2. 通过lua脚本在日志生成阶段获取信息，然后将数据插入mysql。nginx有一个限制，无法在log阶访问socket即无法访问mysql，所以无法在log阶段直接将数据存入mysql。但是可以通过运行包含mysql操作的shell脚本来解决这个问题。但是这个方法有两个弊端：

- 获取到Nginx代理的结果后，每次都要连接mysql并向其插入数据。当并发量大时，mysql端会出问题。
- 不向mysql插入数据，整个时间的消耗大约在0.02-0.04s之间。而向mysql插入数据后，整个时间耗大约在0.4-0.9之间，消耗的时间是原来的10倍。

3. 通过lua + ngx.time.at + lua_mysql + lua.share.dict 解决问。整个过程如下所示：

- 在nginx启动阶段，ngx.time.at启动一个延时任务。在任务中，每隔一段时间取出nginx内存共享的log数据，将数据合并，存入mysql，同时再一个相同的延时任务，递归调用。这样就与crontab命相似。当定时器到期，定时器中的 Lua 代码是在一个“轻线程”中运行的，它与创造它的原始请求是全分离的，因此不存在大量线程同时运行的情况。
- 在日志生成阶段，将数据封装并存入nginx的内存共享区。

Mysql 访问权限的问题

不但访问Mysql的Mysql用户需要有操作对应数据库的权限，还需要调用Mysql命令的用户具有访问msql的权限。授权命令如下：

```
GRANT ALL PRIVILEGES ON *.* to root@xxx IDENTIFIED BY 'password';
```

Mysql 编码类型

总的来说，Mysql的数据库对应三种编码。Mysql客户端显示数据的编码，连接Mysql用的编码（即据存入mysql时，数据的编码），Mysql存储用的编码（字段，表，数据库三种格式可能不同）。不Mysql存储用的编码是什么，只要Mysql客户端显示数据的编码和连接Mysql用的编码相同，数据就通过mysql客户端正确显示。

配置文件

```
user  root;
worker_processes  2;

events {
```

```
worker_connections  1024;
}


http{
lua_package_path "/home/oicq/guomm/nginx_lua/lua-resty-mysql-master/lib/?.lua;;"; --重要
lua_shared_dict logs 10m;

init_worker_by_lua_block {
    local delay = 10
    function put_log_into_mysql(premature)
            local mysql = require "resty.mysql"
            local db, err = mysql:new()
            if not db then
                ngx.log(ngx.ERR,"failed to instantiate mysql: ", err)
                return
            end

            db:set_timeout(1000)
            local ok, err, errcode, sqlstate = db:connect{
                host = "xxx",
                port = 3306,
                database = "database_name",
                user = "username",
                password = "password",
                charset = "utf8",
            }

            if not ok then
                ngx.log(ngx.ERR,"failed to connect: ", err, ": ", errcode, " ", sqlstate)
                return
            end

            -- get data from shared dict and put them into mysql
            local key = "logs"
            local vals = ""
            local temp_val = ngx.shared.logs:lpop(key)
            while (temp_val ~= nil)
            do
                vals = vals .. ","..temp_val
                temp_val = ngx.shared.logs:lpop(key)
            end

            if vals ~= "" then
                vals = string.sub(vals, 2,-1)
                local command = ("insert into es_visit_record(access_ip,server_ip,access_time,run_ti
e,es_response_time,request_body_byte,run_state,url,post_data) values "..vals)
                ngx.log(ngx.ERR,"command is ",command)
                local res, err, errcode, sqlstate = db:query(command)
                if not res then
                    ngx.log(ngx.ERR,"insert error: ", err, ": ", errcode, ": ", sqlstate, ".")
                    return
                end
            end
```

```
        local ok, err = db:close()
        if not ok then
            ngx.log(ngx.ERR,"failed to close: ", err)
            return
        end
        -- decycle call timer to run put_log_into_mysql method, just like crontab
        local ok, err = ngx.timer.at(delay, put_log_into_mysql);
        if not ok then
            ngx.log(ngx.ERR, "failed to create timer: ", err)
            return
        end
    end

    local ok, err = ngx.timer.at(delay, put_log_into_mysql)
    if not ok then
        ngx.log(ngx.ERR, "failed to create timer: ", err)
        return
    end
}


upstream elasticsearch_servers {
    server xxx max_fails=3 fail_timeout=30s;
    server xxx max_fails=3 fail_timeout=30s;
    server xx max_fails=3 fail_timeout=30s;
}

log_format  porxy  '$remote_addr,$upstream_addr,[$time_local],$request,$request_body,$stat
s,$body_bytes_sent,$request_time,$upstream_response_time';

server {
    listen 9202;
    location / {

        proxy_pass http://elasticsearch_servers;

        log_by_lua_block{

            local currentTime = os.date("%Y-%m-%d %H:%M:%S", os.time())
            currentTime = "\"" .. currentTime .. "\""

            local req_body = '-'
            if ngx.var.request_body then
                req_body = ngx.var.request_body
                req_body = string.gsub(req_body,"\n","")
                --req_body = string.gsub(req_body,"\t","")
            end
            req_body = "\"" .. req_body .. "\""

            local req_status = 0
            if ngx.var.status then
                req_status = ngx.var.status
            end
```

```
            local req_time = 0
            if ngx.var.request_time then
                req_time = ngx.var.request_time
            end

            local req_req = "\"" .. ngx.var.request .. "\""
            local remote_addr = "\"" .. ngx.var.http_x_forwarded_for .. "\""
            local server_addr = "\"" .. ngx.var.upstream_addr .. "\""
            local myparams = ("(".."remote_addr..",".. server_addr..",".."currentTime..",".."ngx.var.reque
t_time .. ",".. ngx.var.upstream_response_time..",".."ngx.var.body_bytes_sent..",".."ngx.var.status.."
"..req_req..",".."req_body..")")
            local key = "logs"
            local len,err = ngx.shared.logs:rpush(key, myparams)

            if err then
                ngx.log(ngx.ERR,"failed to put log vals into shared dict")
                return
            end

        }
    }
    access_log logs/es_access.log porxy;
}
}
```

转自：Nginx+lua+mysql实时存日志

原文链接：OpenResty + Mysql 实现日志实时存储