

OpenResty构建 一站式应用网关实践分享

聂永 王辉 @新浪微博

目录

- 一、网关概述
- 二、问题 & 解决
- 三、总结 & 反思

一、网关概述

概述

- 一站式应用网关服务
- 上行请求
 - 代理
 - 聚合
- 下行推送
- TCP/TLS/HTTP/HTTPS/QUIC



背景

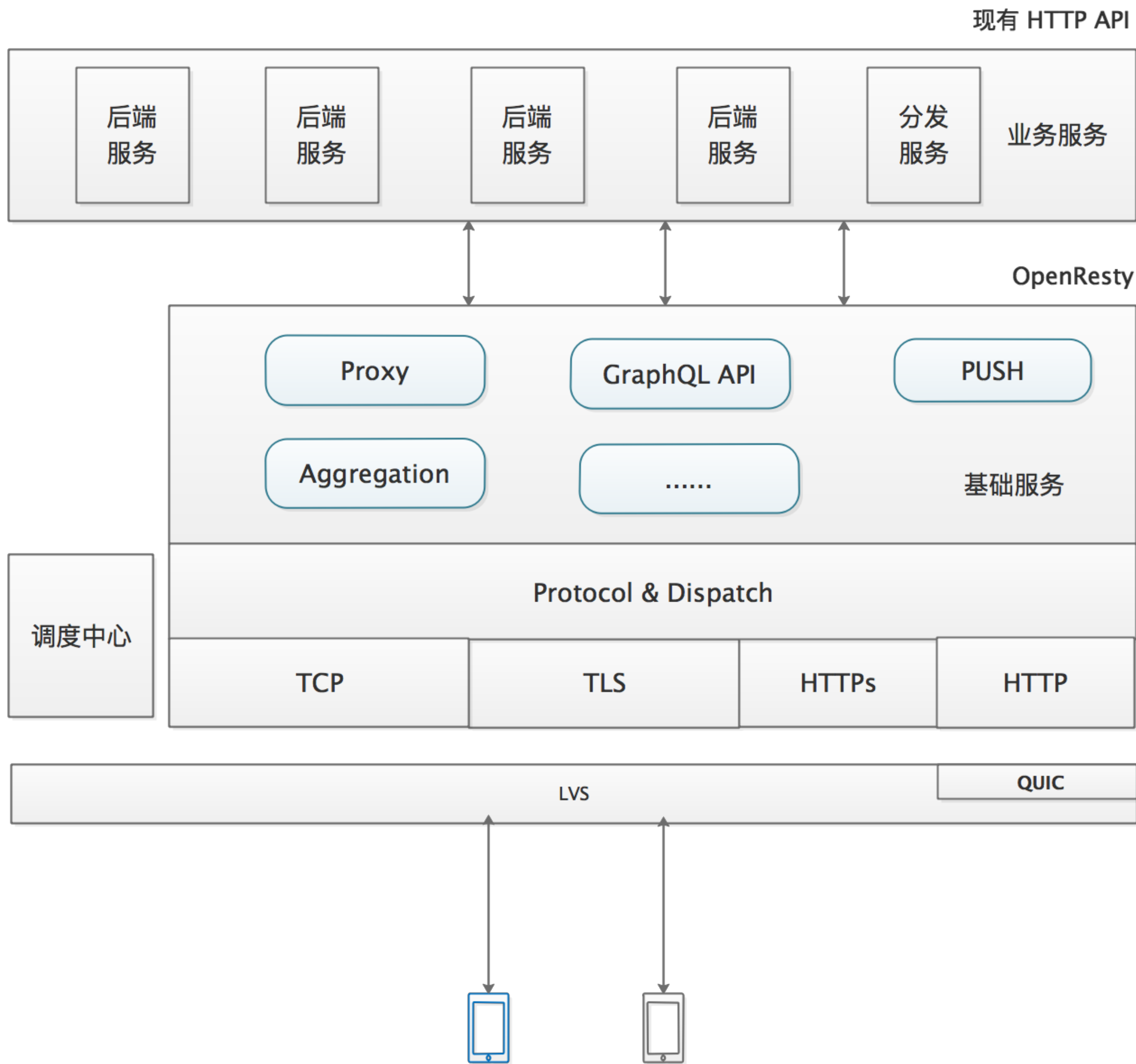
- 网络链路优化，实践见真知
- 新业务，需求、改动频繁
- 人员少，任务重

选择

- Erlang招人难、学习成本高
- Go、Java, 当前业务开发效率问题

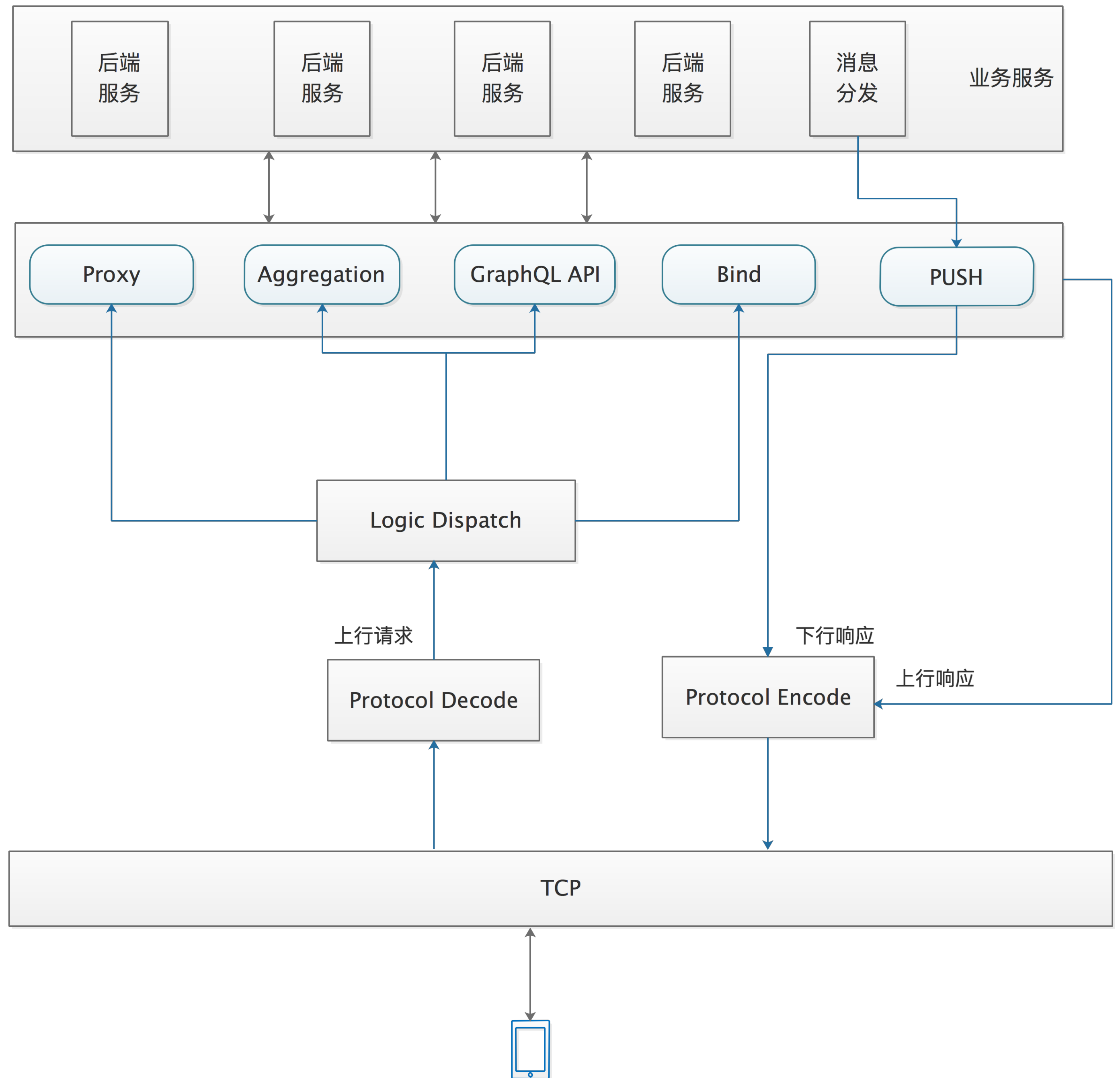
为什么是OpenResty ?

- Nginx+Lua, 速度快性能强
- 学习成本低, 快速开发, 应用广泛
- 支持TCP & HTTP
- 降低运维成本



业务流程

- 所有通道，统一协议
- 代理部分，请求&转发
- 聚合功能，后端多业务聚合，统一输出



二、问题 & 解决

1. TCP不支持Lua Semaphore API

- 目前仅提供HTTP版本
- 长连接下推需要使用到
- 解决方式
 - <https://github.com/openresty/lua-resty-core/issues/94>

2. CJSON-大数

- 使用CJSON进行json的编解码
- API中有64位整数型，会自动被修正为科学计数法

```
Lua 5.1.5 Copyright (C) 1994-2012 Lua.org, PUC-Rio
> num = 232565634353635
> print(num)
2.3256563435364e+14
>
```

- 增加大数支持：<https://github.com/yongboy/lua-cjson>
- `cjson.decode_big_numbers_as_strings(true|false)`

CJSON-数组和对象

- Lua, 数组和对象使用table表示
- table转换JSON
- 数组[]? 结构{}? 同时存在?

```
local cjson = require "cjson"
local ori = {name = "array & object", array = {}, object = {}}

-- 1. 指定某个key为空数组, 推荐
setmetatable(ori.array, cjson.empty_array_mt)
ngx.say(cjson.encode(ori))
-- output: {"array":[],"object":{},"name":"array & object"}

-- 2. 统一全局设置需要编码成数组
cjson.encode_empty_table_as_object(false)
ngx.say(cjson.encode(ori))
-- output: {"array":[],"object":[],"name":"array & object"}

-- 3. 下面情况, 全局设置默认覆盖局部设置
setmetatable(ori.object, cjson.object)
ngx.say(cjson.encode(ori))
-- output: "array":[],"object":[],"name":"array & object"}
```


CJSON Decode性能

- 一定压力下，已暴露 decode 占用CPU过高问题
- 已着手使用 lua-resty-json 替换

Flame Graph

Search



3. 日志传输

- <https://github.com/cloudflare/lua-resty-logger-socket>
- UDP传输消息的上限
 - 警告: **send() failed (90: Message too long)**
 - 单个UDP数据包不能大于65507字节长度!
 - 消息裁剪!
 - `cat /proc/sys/net/core/wmem_max`
- 自动刷新
 - 警告: **logger buffer is full, this log message will be dropped**
 - `periodic_flush`
- `newline`换行符号

```
local logger = require "resty.logger.socket"
local ngx_log = ngx.log

local function do_log(msg)
    if not logger.initted() then
        ngx_log(ngx.INFO, "init logger")
        local ok, err = logger.init{
            host = '10.10.10.10',
            port = 5143,
            flush_limit = 1024,
            -- UDP包最大负荷长度: 2^16 - 1 - 20 - 8 = 65507字节
            drop_limit = 65507,
            -- 每秒钟刷新频率, 适用于日志产生频繁环境
            periodic_flush = 0.001,
            sock_type = 'udp'
        }
        if not ok then
            ngx_log(ngx.ERR, "failed to initialize the logger: ", err)
        end
    end

    if #msg >= 65507 then
        msg = string.sub(msg, 1, 65506)
    end

    msg = msg .. "\n"

    local _, err = logger.log(msg)
    if err then
        ngx_log(ngx.ERR, "failed to log message: ", err)
    end
end
```

日志产生过快

- 压力大，UDP忙不过来了
 - failed to log message: too many pending timers

- 处理方法

- 回退到磁盘日志模式，极可能的保证不丢失

```
error_log syslog:server=10.10.10.10:3514 notice;  
error_log logs/error.log error;
```

- 提高日志等级，减少输出量
 - 合并日志，减少输出次数

4. TCP Server glibc内存不释放问题

- 老问题: https://moonbingbing.gitbooks.io/openresty-best-practices/something/2016_10_1.html
- glibc malloc & free & malloc_trim
- 最偷巧方式
 1. 增加HTTP节点
 2. 配置 `lua_malloc_trim 1;`
 3. 经常周期性访问HTTP接口

Demo

TCP通道压测后

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
10983	nobody	20	0	987m	382m	2300	S	0.0	2.4	2:36.23	nginx: worker process

调用HTTP API接口后

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
10983	nobody	20	0	987m	164m	2472	S	0.3	1.0	2:36.24	nginx: worker process

还可以定时释放

- 请求不均匀，效果不理想
- 基于FFI的glibc.malloc_trim(1)
- Code: [lua_malloc_trim.lua](#)
- 每worker定时触发
- `init_worker_by_lua_file *.lua`

```
local ffi = require "ffi"
local glibc = ffi.load("c")

if ffi.os == "Windows" then
    ngx.log(ngx.ALERT, "DOES NOT SUPPORT windows")
    return
elseif ffi.os == "OSX" then
    ngx.log(ngx.ALERT, "DOES NOT SUPPORT MAC OS X")
    return
end

ffi.cdef[[
    int malloc_trim(size_t pad);
]]

-- eg: glibc.malloc_trim(1)
```

5. GraphQL API



- HTTP API接口兼容版本以及各种特殊业务需求，变得越来越重
- 使用GraphQL API一般可为客户端节省至少50%网络下行流量
- 已经服务于某微博版本

5.1 易用性定制

- 增加GraphQL API Client兼容性，方便调试
- 修复需要时得不到InputObject参数的BUG
- 修复指令和参数不协作的BUG • <https://github.com/yongboy/graphql-lua>

The screenshot displays the GraphQL Playground interface. On the left, a query is defined with two fields: `group` and `user`. The `group` field is queried with an ID, and the `user` field is queried with the ID of the user within that group. The response on the right shows the JSON structure of the data returned, including the group name and the user's details. Below the response, the schema for the `User` type is shown, listing its fields and their types.

```
1 query queryUser($id: String!) {
2   group(id: "3755069097048281") {
3     id
4     avatar
5     name
6   }
7
8   user(id: $id) {
9     id
10    name
11    level
12    remark
13    avatar_large
14    gender
15  }
16 }
```

```
{
  "data": {
    "group": {
      "id": "3755069097048281",
      "name": "微友服务器",
      "avatar": "http://ww4.sinaimg.cn/large/8de561a8jw1f052narbekj2050050wet.jpg",
      "user": {
        "id": "5662498887",
        "name": "不喝酒的小强",
        "avatar_large": "http://tva4.sinaimg.cn/crop.0.0.168.168.180/006bdgDdjw8euhaquhefwj304o04ojr8.jpg",
        "gender": "m"
      }
    }
  }
}
```

weibo user info

FIELDS

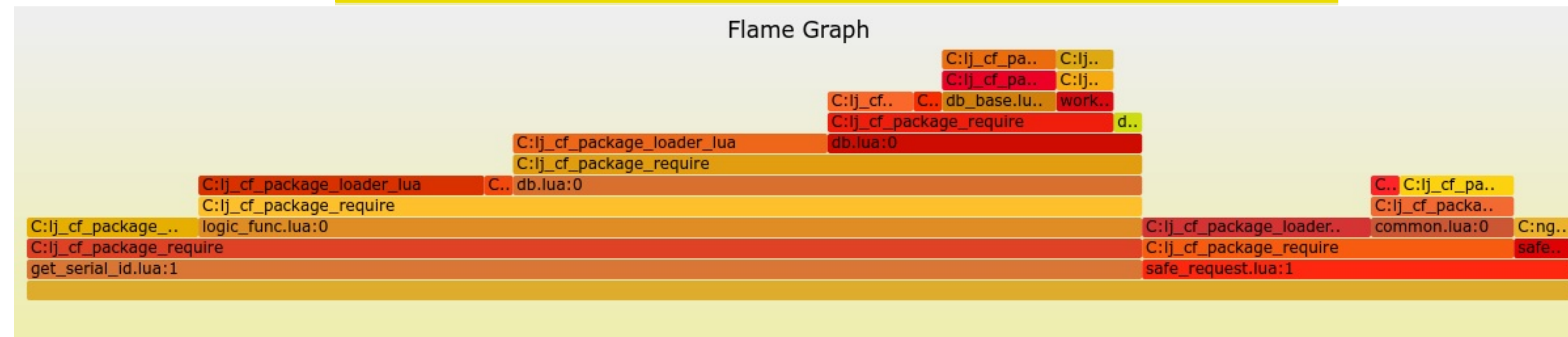
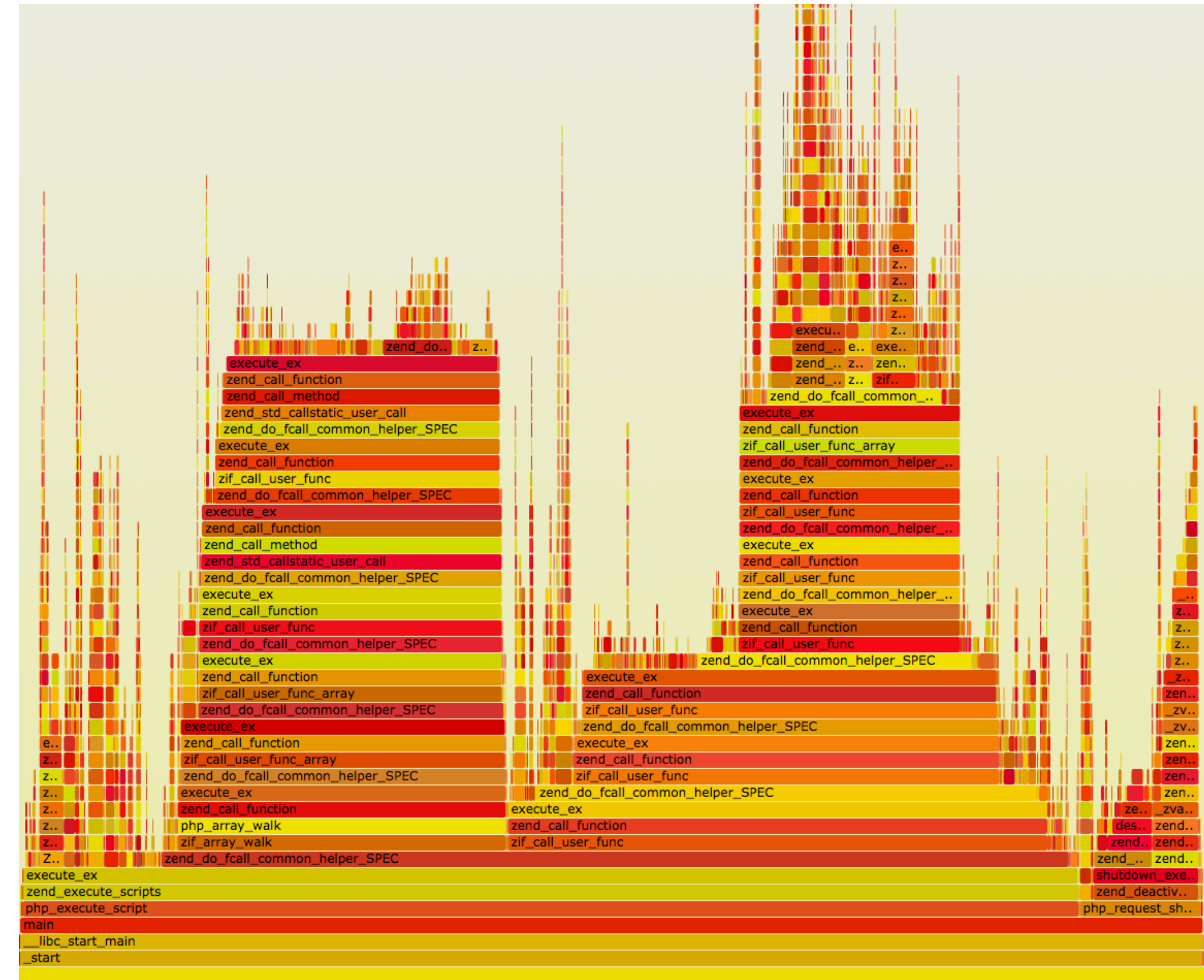
- level: Int
- remark: String
- gender: String
- id: String!
- name: String!
- verified_type: Int
- avatar_large: String!
- verified: Boolean

QUERY VARIABLES

```
1 {"id": "5662498887"}
```

6. CPU过高问题

- 业务处理低效性能出现问题
- 请求高，CPU占用100
- 发现瓶颈
- 横向宽度大平顶
- 纵向调用层级深度



7. 内存火焰图

- 多次采样对比
 - eg: 一次300秒, 一次500秒
 - 小心对比, 寻找增加宽度的可疑点
- 大处着眼, 小处着手, 谨慎寻找异样的输出

Demo

posix_memalign

ngx_memalign

ngx_palloc_block

ngx_palloc_small

ngx_pnalloc

ngx_parse_inet_url

ngx_parse_url

ngx_stream_lua_socket_udp_setpeername

lj_BC_FUNCC

ngx_stream_lua_run_thread

ngx_stream_lua_sleep_resume

ngx_stream_lua_sleep_handler

ngx_event_expire_timers

ngx_process_events_and_timers

ngx_worker_process_cycle

ngx_spawn_process

ngx_start_worker_processes

ngx_master_process_cycle

main

__libc_start_main

_start

all

ngx_stream_lua_socket_udp_setpeername
(220,032 bytes, 6.84%)

Function: ngx_stream_lua_socket_udp_setpeername (220,032 bytes, 6.84%)

8. 调试依赖

- yum install yum-utils
- yum install kernel-devel
- debuginfo-install kernel



- SystemTap 2.6
- GDB 7.11
- Valgrind 3.13.0
- GCC 4.8 ASAN
- stapx
- openresty-gdb-utils
- openresty-systemtap-toolkit

简化调试

- 目标：降低跟踪调试成本，轻松参与其中
- 一个脚本初始化调试环境
- 一个脚本搞定CPU、内存等资源占用跟踪调试
 - 自动输出调试报告并提供火焰图访问地址
- 仓库地址：<https://github.com/yongboy/openresty-debug-tools>

9. 压力测试辅助

- HTTP的压力测试选择很多
 - ab、wrk、WebBench、...
- TCP可以选择Tsung
 - 可以模拟线上复杂业务
 - 压力灵活、可持续

三、总结 & 反思

总结

- 接口请求耗时可减少33.6%
- 接口成功率可提升0.6%
- 极速版采样一分钟内可降低用户72.34%流量
-

 新浪新闻 
5分钟前 来自专业版微博平台

心疼

@当代生活观察家:【女子自称对WiFi等电磁波过敏 逃乡下搭帐篷睡茅屋🤩】英国44岁女子蕾切尔称自己患电磁波过敏症，任何无线电波都让她头痛、头晕、心悸.....为了躲避WiFi，她经常睡在车里，或者在树林里搭帐篷睡觉，甚至睡在茅草屋↓↓生活苦不堪言。但大多数专家却认为↓↓

 女子称对WiFi等电磁波过敏 逃乡下搭帐篷睡茅屋
来源：看看新闻KNEWS 英国44岁女子蕾切尔...

 转发  15  17

 DIY设计家居馆 
10分钟前 来自微博weibo.com

😱😱如果早1个月看到这个，恐怕我早已至少D罩杯了，太给力了!!!

@美胸老师_:#丰胸终极秘诀# 🤩专门帮助21岁以上的菇娘 摆脱平胸，解决胸型问题❤️已有上万女性成功到C，名额有限！要丰胸美胸的菇娘赶紧收了~👉👉👉
[S女神](#) 的秒拍视频

反思

- TCP支持没有HTTP完善
- 针对UDP协议支持还不够充分
 - QUIC, 暂时还看不到希望
- 内存管理方面, 受限于GLIBC内存不及时释放问题
- socket句柄无法在多个轻量级线程中共享
- Timer数量受限, 不要滥用
- 后期学习难度较大

Thank You :))