## Unix

Mac

Ctrl a 移动到行首

ctrl e 移动到行尾

zsh

玩zsh一般都是先玩ohmyzsh

cat /etc/shells 查看系统的shell情况，有没有zsh

Chsh -s /bin/zsh   zsh就像是一种bash切换到zsh

ohmyzsh

 下载之前要把本地的.zshrc删除掉

sh -c "$(curl -fsSL https://raw.github.com/robbyrussell/oh-my-zsh/master/tools/install.sh)"

下载两个插件，这两个必须要下载才行，别的在plugins里改下再source下就可以了

git clone https://github.com/zsh-users/zsh-autosuggestions $ZSH\_CUSTOM/plugins/zsh-autosuggestions

git clone https://github.com/zsh-users/zsh-syntax-highlighting.git $ZSH\_CUSTOM/plugins/zsh-syntax-highlighting

发现安装完ohmyzsh后有些命令用不了了，就去在.zshrc里面写上source .basn\_profile 或则会source .profile或者

PATH=$PATH:/opt/tiger/ss\_bin:/usr/local/jdk/bin:/usr/sbin/:/opt/tiger/ss\_lib/bin

cmoand + h 隐藏很多东西

command + m 最小化

command + ctrl + f 最大化 再按取消

command opiton I 调出谷歌调试

command shrift t 打开已经关闭的浏览器页面

1. Option + ←：光标向左移动一个单词
2. Option + →：光标向右移动一个单词
3. Option + Delete：删除一个单词

选中文件后按空格可以预览

三指，就是按下选中的操作全用三个手指，比如移动窗口，选中文字，拉边框等。

完全键盘控制，弹出选择栏可以用tab和空格来操作

comand ， 调出软件的设置栏

编辑 ~/.ssh/config 文件，添加以下内容，就可以不用输入ip和密码登录了

Host testhost

HostName 1.2.3.4

User root

Port 54321

IdentityFile ~/.ssh/id\_rsa

unix命令

Vim复制不了就用cat

ssh root@c.qingbei.com 远程连接

scp xxx文件路 root@c.qingbei.com:/usr一个路径。文件复制

cat /etc/issue 查看linux版本

Shift e 跳到某个词的最后一个

netstat -nat | grep LISTEN

Grep xxx xx.text

Less 文件名  /xxx 查找什么. 按f （forward）进入下一页。b（backward） 进入上一页   按Q退出

Tail -n 20 文件名。显示文件的末尾20行。tail -f 文件名。动态显示文件的末尾20行

Quota 是用来是查看磁盘使用情况和限额的

wc -l 查看有多少行 这个一般都放在最后咯

Wc 多少行 多少单词 多少字节

Vim

:set filetype=go

:vs 横屏打开文件

:set paste

从外面复制要线set paste一下

w b

E 0

23,43m.   把m换成t就是复制的意思

:Ve 打开文件所在位置

Grep -v xxx  查询不带xxx的 grep mybankedet xxx.log | grep -v err\_no = 0

Set hls 设置高亮.

Ctrl + g 显示当前文件名

光标停在一个单词上 shift \*  就可以全局搜索该单词

Goreferecce 查看被调用的地方

:vs 打开一个新的

 ctrl ww 切换上下

Ctrl o 返回各种编辑点

Ctrl 6 返回上一个编辑点

Vim的配置是不用source加载的

curl -fLo ~/.vim/autoload/plug.vim --create-dirs https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim

下载vimplugin

curl https://j.mp/spf13-vim3 -L > spf13-vim.sh && sh spf13-vim.sh

下载govim

然后操作before

vim ~/.vimrc.before.local

let g:spf13\_bundle\_groups=['general', 'youcompleteme', 'python', 'programming', 'misc', 'writing', 'go']

装完那几个插件后还要去vimrc里面bundleinstall

然后还要去ycm里面

d ~/.vim/bundle/YCM

git submodule update --init --recursive

(install cmake)

./install.py --gocode-completer

然后再goinstall二进制

Vim-plugin只对vimrc这个有用

 gg 跳到第一行

shift+G 跳到最后一行

:n 跳到第n行

/xxx 查找xxx  查找后按n，跳到下一个n，按N，调到上一个

在非插入模式下

H 左移 L右移 J上移 K下移

Ctrl f 下一页 ctrl b 上一页

dd 删除当前行。yy 复制当前行

ndd 删除n行 nyy复制n行

退出编辑模式后点 u 就是把上次修改的内容撤销

非编辑模式 向前删除 X 向后删除 x

 p 将复制来的数据 粘贴到光标的下一行

spf13

zR 展开所有折叠

Ctrl +e 打开关闭浏览器

j k上下移动

o 打开文件或者目录

u 返回上一层

shrift c 选择根目录

Ctrl n

Ctrl p 上下补全

Ctrl ] 跳转

ctrl t 返回

W b 按词移动

ctrl 6 返回上一个文件

行尾 shitf 6    行首 shift 4

在非编辑模式下，选择后，y就是复制  d就是剪切，p就是粘贴， 是在光标处粘贴

在光标处按一下o 就可以在下一行插入，i的话就是在光标当前位置插入

可视化模式和插入模式 按下 > 和 < 都可以 退一个tab的格

:s/vivian/sky/ 替换当前行第一个 vivian 为 sky

:s/vivian/sky/g 替换当前行所有 vivian 为 sky

rm   .vimswap/show\_hub.thrift.swp.  删除.vimswap的东西即可解决

如果修改了东西却没有保存，q的时候就会提示，wq就没有问题了

删除用户根目录中隐藏文件件.vimviews中的所有文件解决高亮块问题

## 路径和密码

路径与账户密码

<https://bytedance.feishu.cn/space/doc/doccntmM3JfMCRHCsejwAK6muRb>

售卖系统。[https://bytedance.feishu.cn/space/doc/doccn5fhB0mpCPi9Xvbiv2QB8Pb#](https://bytedance.feishu.cn/space/doc/doccn5fhB0mpCPi9Xvbiv2QB8Pb)

<https://bytedance.feishu.cn/space/doc/doccnmxsLqsxo89sWsNHLIO28de> elepant

<https://code.byted.org/qingbei>

Houtairongzai

<https://godoc.org/>

godoc

<http://enotify.bytedance.net/eplatform/pigeon/user-groups>

推送中心

https://rocket.bytedance.net/rocket/clientpublish/history/1585/android?biz=1585&filter=&commitId=&emailPrefix=&fiveVersion=&outerVersion=&description=&since=&until=&page=1&levels=&status=

Boeapp

<https://bytedance.feishu.cn/space/doc/doccnziJN5yptIoTRWWaerEM0dc>

<https://github.com/senghoo/golang-design-pattern>

Goshejimoshi

<https://idndx.com/2014/09/01/the-implementation-of-epoll-1/>

Poll

<https://juejin.im/post/5c888a79e51d456ed11955a8>

Goneicunmosixng

<http://jolestar.com/parallel-programming-model-thread-goroutine-actor/>

Gobingfamoxing

<https://mp.weixin.qq.com/s/Rewl0DKnq6CY53m5D3G2qw>

go语言如何跑起来的

location.hash = "#/debug”

git clone gitr:/service\_rpc/idl

https://mermaidjs.github.io/mermaid-live-editor/  在线画图工具

[https://mermaidjs.github.io](https://mermaidjs.github.io/)   文档

https://code.byted.org/ugc/bang 配置开发机

<https://cloud.bytedance.net/ppe/Boe>

boe逃逸分析

<https://light.bytedance.net/docs/cases/rpc.html> 轻服务

<http://rocket.bytedance.net/rocket/mytask?biz=1585>   rocket

[https://bytedance.feishu.cn/space/doc/JyjEm6RBsoCtitY7lS26Ka#lkwDRQ](https://bytedance.feishu.cn/space/doc/JyjEm6RBsoCtitY7lS26Ka" \l "lkwDRQ)  新人必读 代码规范等

<https://bytedance.feishu.cn/space/doc/doccn9KXqTstK9B7zdahgkvg7Ic> vim路径

https://www.processon.com/ 画图

数据分析 <https://data.bytedance.net/tea/app/385/dashboard/6699267122897355272>

/opt/tiger/ss\_conf/ss

sudo /opt/tiger/consul\_devbox/bin/switch.sh boe

http://evboe.bytedance.net/ev/feed/v2/h5\_index。boe环境测试路径的域名

export http\_proxy=10.110.216.52:3128 https\_proxy=10.110.216.52:3128

<https://grafana.byted.org/>  监控路径

https://evboe.bytedance.net/mobile/act/xxxx 投放连接

Cc视频 18534252568    zuguoxiwang2018

C服务器密码：LZKMPFrtY0mX6ICf

E服务器密码：jAu5vYmWYWKZkrvD

mysql -h47.92.202.123 -uroot -pAhuamath#2018 -Dhuamath\_dragon

ping++   15801163287@163.com  ayj123470

http://license.bytedance.net  头条激活jet软件服务器

https://wiki.bytedance.net/pages/viewpage.action?pageId=224479395 产品技术文档

在线课堂系统设计：[https://docs.bytedance.net/doc/70VuLyO4GGbUDmNsIG7NDe](https://docs.bytedance.net/doc/70VuLyO4GGbUDmNsIG7NDe" \t "/Users/yangwenshuo/Documents\\x/_blank)

https://code.byted.org/edu 代码路径

https://docs.google.com/presentation/d/1AQ9ecEW-tnBGN101vDdRk5CkHxkXsDvcC2RWBBJN-uM/edit  api网关的介绍

 sd lookup toutiao.mysql.ev\_class\_write。获得数据库的id，要远程到开发机

<https://blog.csdn.net/jiaobuchong/article/details/80517746> debain thrift

Mac装thrift需要看那个文章的评论

/service\_rpc/idl

[https://bytedance.feishu.cn](https://bytedance.feishu.cn/) 飞书云空间

[https://study.bytedance.net](https://study.bytedance.net/) 学习平台

git push origin HEAD:refs/for/master%r=yangwenshuo.00

[https://lanhuapp.com/web/#/item/project/board?pid=e8209cc5-3be2-4636-95f4-d3e9be11952b](https://lanhuapp.com/web/" \l "/item/project/board?pid=e8209cc5-3be2-4636-95f4-d3e9be11952b) 蓝湖

https://wiki.bytedance.net/pages/viewpage.action?pageId=52897307 Charles调试

## Git gomod vendor

Git gomod govendor

Git branch -r

git show coomitid

Git blame filename

Git clean -n 预删除

git clean -f  删除那些没有被track的文件

git remote update origin —prune

Git rebase master

如果有冲突的话这里就会进入一个新的解决冲突的临时分支，

git status查看文件的状态

解决冲突后，

Git add -all

Git clean -f 删除未track的文件

Git rebase —continue

Git rm xxx

git checkout -b dev（本地分支名） origin/dev（远程分支名）

git stash save “lock"

git stash pop

git stash list

Git stash clear

Go test -run: hanshuming

Git branch -d xxx 删除分支

go get -u [code.byted.org/edu/common/constdef@master](http://code.byted.org/edu/common/constdef@master)

@后面也可以跟上tag的版本,还有就是这里只是拉下来,还有去go,mod文件改一下依赖的版本.

govendor fetch code.byted.org/passport/...

govendor  fetch [code.byted.org/edu/common/errno/](http://code.byted.org/edu/common/errno/)…

Go vendor update +ex.

fetch只是把东西下载到gopath, +ex会把内容移到vendor里面

govendor update +ex

## 英文

英文

integrate 集成，合并

archive 存档，档案

Middleware MW中间件

hub 中心

category 种类

inventory 库存

logic 业务处理单元

mentor 导师

crontable  定时指令

Mesh 网状物 网格

marshal 序列化

promotion 晋级，提拔

RAM 访问控制

adaptive bitrate streaming 自适应比特率流

distribution 分发

intend 打算

customisable 可定制的

Customization 定制

similar 类似的

directory 目录

up to date 现代的

mess 杂乱，脏乱

efficient 效率高的

features 特征，特色

regardless 是 不管的意思，regard就是管，考虑的意思

recommend 推荐

preview 预览

compatible 兼容的

optimize 优化

suit 适合

extremely 极其，非常

section 部件

labeled 标签

comment 注释

option 选项

tier 层

additional 额外的

maintain 维护

bundle 一捆，一包，一扎

optimal 最优的

snippets 代码段

syntaxes 语法

appropriate 适当的

RTT round trip time

SSO  单点登录

Rpc remote procedure call 远程过程调用

Assign 分派

黑话

DAU 日活跃用户

Qos 每秒查询次数

TCE 头条 computer engine 头条电脑发动机

pv pageview 网页访问量

Uv unique vistor 独立访客 同一天的uv只能算一个。

DevOps（Development和Operations的组合词）是一组过程、方法与系统的统称，用于促进开发（应用程序/软件工程）、技术运营和质量保障（QA）部门之间的沟通、协作与整合。

它是一种重视“软件开发人员（Dev）”和“IT运维技术人员（Ops）”之间沟通合作的文化、运动或惯例。透过自动化“软件交付”和“架构变更”的流程，来使得构建、测试、发布软件能够更加地快捷、频繁和可靠。

## 规范

method主流程里面的错误日志级别为error, 内部里面调用的日志级别都是warn

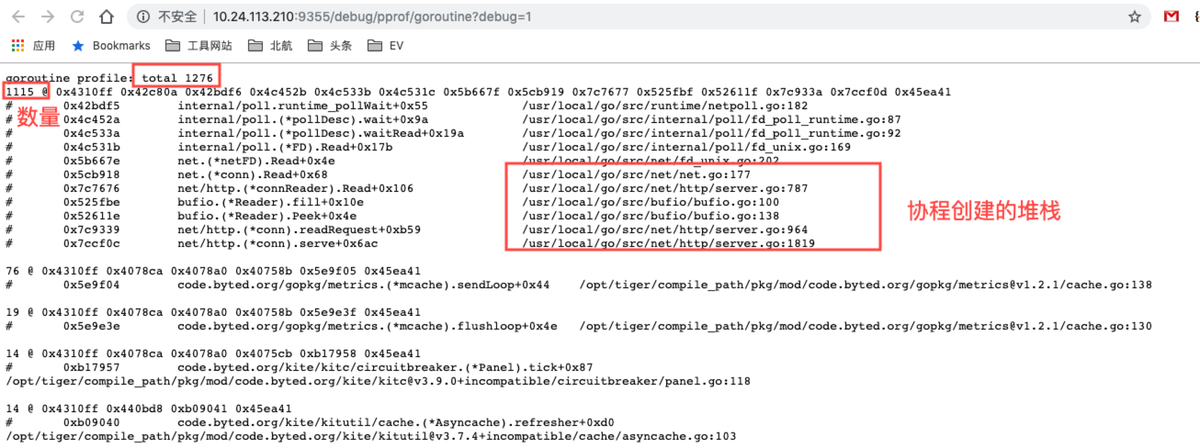
## 协程数过多问题定位

## 通过debug接口查看协程信息

[http://10.24.113.210:9355/debug/pprof/goroutine?debug=1](http://10.24.113.210:9355/debug/pprof/goroutine?debug=1" \t "/Users/yangwenshuo/Documents\\x/_blank)

记得替换ip和端口

长链接



根据堆栈看一下协程是在哪儿创建的，通过代码分析创建这么多协程数是否合理

协程泄漏的例子

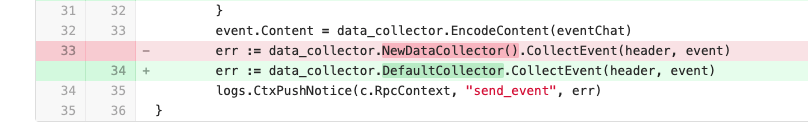
持续创建执行无限循环任务的协程



[https://code.byted.org/data/databus\_client/blob/master/client.go#L686](https://code.byted.org/data/databus_client/blob/master/client.go" \l "L686" \t "/Users/yangwenshuo/Documents\\x/_blank)

DataBusCollector.Consumer是一个无限循环，执行Consumer的协程不会自动结束。上图中显示有466个这样的协程，且还在不断增加。

协程创建的位置是：[https://code.byted.org/data/databus\_client/blob/master/client.go#L162](https://code.byted.org/data/databus_client/blob/master/client.go" \l "L162" \t "/Users/yangwenshuo/Documents\\x/_blank) 每次NewCollectorV1的时候都会创建一个协程，检查业务代码发现每次SendEvent的时候都会调用NewCollectorV1，导致协程一直在创建且不会结束



未close HttpResponse.Body

参考：[https://www.cnblogs.com/cobbliu/p/4517598.html](https://www.cnblogs.com/cobbliu/p/4517598.html" \t "/Users/yangwenshuo/Documents\\x/_blank)

当client接受到整个响应后，如果应用层没有调用response.Body.Close()函数，刚刚传输数据的persistConn就不会被加入到连接缓存中，这样如果您在下次发起HTTP请求的时候，就会重新建立TCP连接，重新分配persistConn结构，这是不调用response.Body.Close()的一个副作用。

      如果不调用response.Body.Close()还存在一个问题。如果请求完成后，对端关闭了连接（对端的HTTP服务器向我发送了FIN），如果这边不调用response.Body.Close()，那么可以看到与这个请求相关的TCP连接的状态一直处于CLOSE\_WAIT状态

忘记截图了，下次碰到再补充。

简单描述下是创建了4w多个persistConn的ReadLoop和WriteLoop

对端连接异常server端还在持续read（压测时会出现）

现象：大量的goroutine处于IO wait状态无法退出

原因：ginex在创建http.Server的时候没有指定conn的读写超时时间：[https://code.byted.org/gin/ginex/blob/master/engine.go#L116](https://code.byted.org/gin/ginex/blob/master/engine.go" \l "L116" \t "/Users/yangwenshuo/Documents\\x/_blank)，对端没有写入的时候可能会造成协程hang住

堆栈：

goroutine 27598861 [IO wait, 761 minutes]:

internal/poll.runtime\_pollWait(0x7fa408fd2790, 0x72, 0xffffffffffffffff)

        /usr/local/go/src/runtime/netpoll.go:182 +0x56

internal/poll.(\*pollDesc).wait(0xc009c6b118, 0x72, 0x1000, 0x1000, 0xffffffffffffffff)

        /usr/local/go/src/internal/poll/fd\_poll\_runtime.go:87 +0x9b

internal/poll.(\*pollDesc).waitRead(...)

        /usr/local/go/src/internal/poll/fd\_poll\_runtime.go:92

internal/poll.(\*FD).Read(0xc009c6b100, 0xc006fc5000, 0x1000, 0x1000, 0x0, 0x0, 0x0)

        /usr/local/go/src/internal/poll/fd\_unix.go:169 +0x19b

net.(\*netFD).Read(0xc009c6b100, 0xc006fc5000, 0x1000, 0x1000, 0xc000048500, 0xc002b35b08, 0x7bf66a)

        /usr/local/go/src/net/fd\_unix.go:202 +0x4f

net.(\*conn).Read(0xc0079d65b8, 0xc006fc5000, 0x1000, 0x1000, 0x0, 0x0, 0x0)

        /usr/local/go/src/net/net.go:177 +0x69

net/http.(\*connReader).Read(0xc00ccad830, 0xc006fc5000, 0x1000, 0x1000, 0x0, 0x0, 0x72)

        /usr/local/go/src/net/http/server.go:787 +0x107

bufio.(\*Reader).fill(0xc00dbc4600)

        /usr/local/go/src/bufio/bufio.go:100 +0x10f

bufio.(\*Reader).Peek(0xc00dbc4600, 0x4, 0x0, 0x0, 0x0, 0x0, 0x48b42a)

        /usr/local/go/src/bufio/bufio.go:138 +0x4f

net/http.(\*conn).readRequest(0xc001c31ae0, 0x1570d40, 0xc0046675c0, 0x0, 0x0, 0x0)

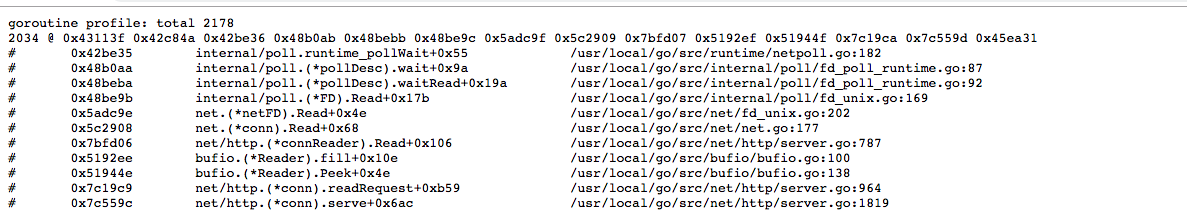
        /usr/local/go/src/net/http/server.go:964 +0xb5a

net/http.(\*conn).serve(0xc001c31ae0, 0x1570d40, 0xc0046675c0)

        /usr/local/go/src/net/http/server.go:1819 +0x6ad

created by net/http.(\*Server).Serve

        /usr/local/go/src/net/http/server.go:2884 +0x2f4



http serve协程数过多问题追查

附录

线上服务debug接口



一般情况下第一个是服务端口，第二个是debug端口。如果没有的话检查一下服务配置：

[https://cloud.bytedance.net/tce/services/16984/update/](https://cloud.bytedance.net/tce/services/16984/update/" \t "/Users/yangwenshuo/Documents\\x/_blank)



## 长链接

长连接框架基本介绍

预备知识

* [Websocket](https://zh.wikipedia.org/wiki/WebSocket" \t "/Users/yangwenshuo/Documents\\x/_blank)
* [HTTP keep-alive](https://zh.wikipedia.org/wiki/HTTP持久连接" \t "/Users/yangwenshuo/Documents\\x/_blank)
* [HTTP/2](https://zh.wikipedia.org/wiki/HTTP/2" \t "/Users/yangwenshuo/Documents\\x/_blank)

扩展知识

[QUIC](https://zh.wikipedia.org/wiki/快速UDP网络连接" \t "/Users/yangwenshuo/Documents\\x/_blank)

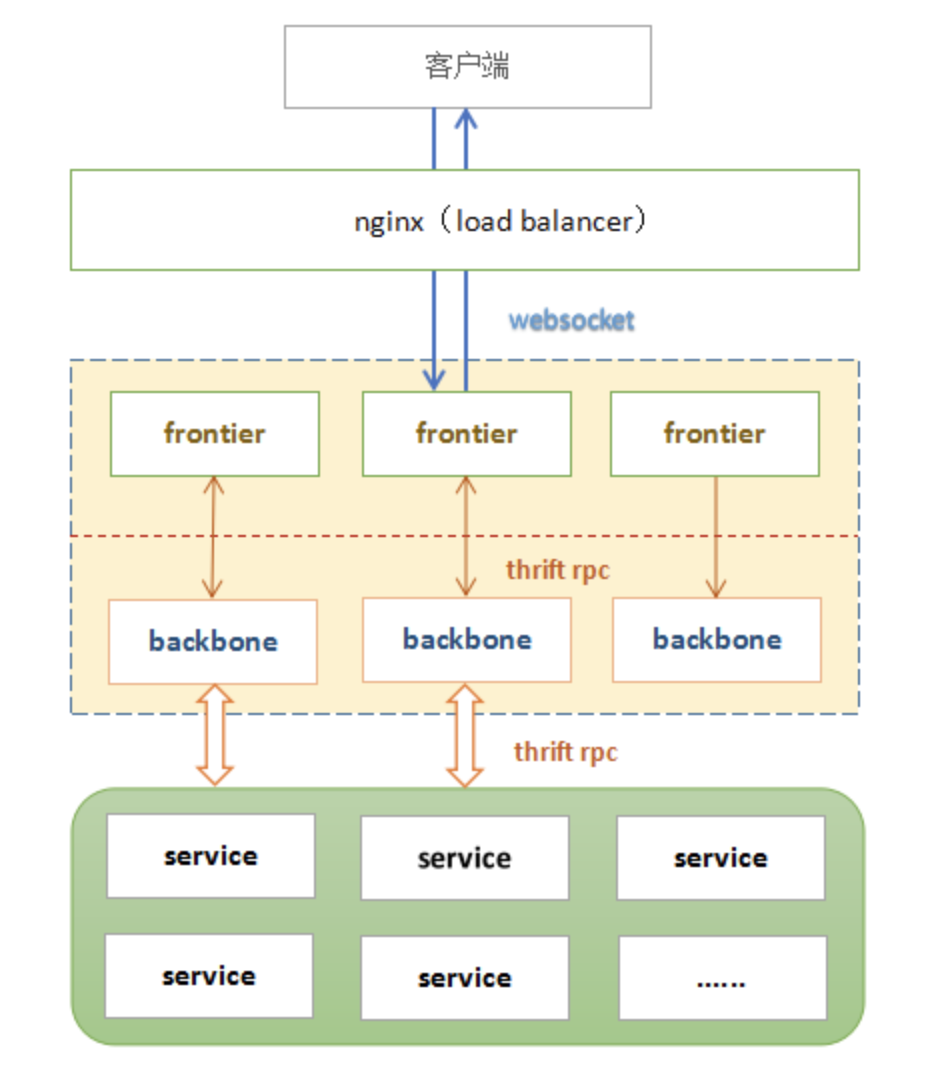
优势

* 多路复用
* ConnectionID
* FEC
* 0RTT
* 安全性 TLS1.3
* 用户态协议
* 在高延迟网络下，性能优秀

问题

* 运营商UDP限制，95.6%的可用性
* ~~QUIC在极高带宽条件下，下载大量数据时，效果不佳~~

整体结构



长连接主要分为**Frontier、BackBone、BackService**三部分，**Frontier**负责接收客户端的长连接，并维持长连接状态；**BackBone**负责为其他服务提供Push的接口，同时维护长连接-Frontier之间的关系；**BackService**负责客户端上传数据时，调用相应的服务端接口。

Frontier

Frontier与客户端通过websocket协议连接，与BackBone通过Thrift协议连接。因此，Frontier有两个主循环——**HTTP循环**和**Thrift循环。**

全局变量

* **gHub**：用来保存于客户端的连接
* gWebsocketServer：全局HTTP服务 /ws/v1 /ws/v2 /ws/echo
* gThriftServer：全局Thrift服务
* gUUIDGenerator：ID生成器
* gConnStats：Metrics服务

Websocket数据帧格式

*message* ***Frame*** *{*

*required uint64 seqid                   = 1;*

*required uint64 logid                   = 2;*

*required int32 service                  = 3;*

*required int32 method                   = 4;*

*message* ***ExtendedEntry*** *{*

*required string key             = 1;*

*required string value           = 2;*

*}*

*repeated* ***ExtendedEntry*** *headers          = 5;*

*optional string        payload\_encoding = 6;*

*optional string        payload\_type     = 7;*

*optional bytes         payload          = 8;*

*}*

1. seqid：主要可以用来进行消息除重以及在这层之上实现RPC的功能
2. logid：这是出于问题定位的考虑，为每个消息增加一个唯一标示，可以方便定位问题，它是一个随机生成的标示
3. service：frontier协议层通过该字段和appid来路由收到的frame，先查本地的路由映射规则表，然后将调用对应的service上的rpc。
4. method: 为某个具体的接口定义的标示，语义上可以理解为“送花”、“充值”、“登陆”对应的远程接口名，但frontier并不需要理解。
5. headers: 这是一个可选的简单的字符串key-value，主要留作后续在协议层的扩展信息，不设置的时候不会增加额外的开销。
6. payload\_type: 和content-type一样，可以对包体的数据格式进行描述，比如取值可以是json、xml、protobuf等
7. payload：传输的业务包

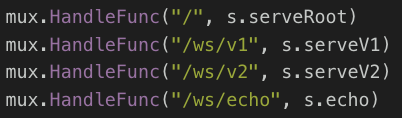
Thrift循环

Frontier服务对外提供两个Thrift接口——PushByUUID、PushByChannel，这两个接口仅由BackBone服务调用，其他服务不能直接调用。

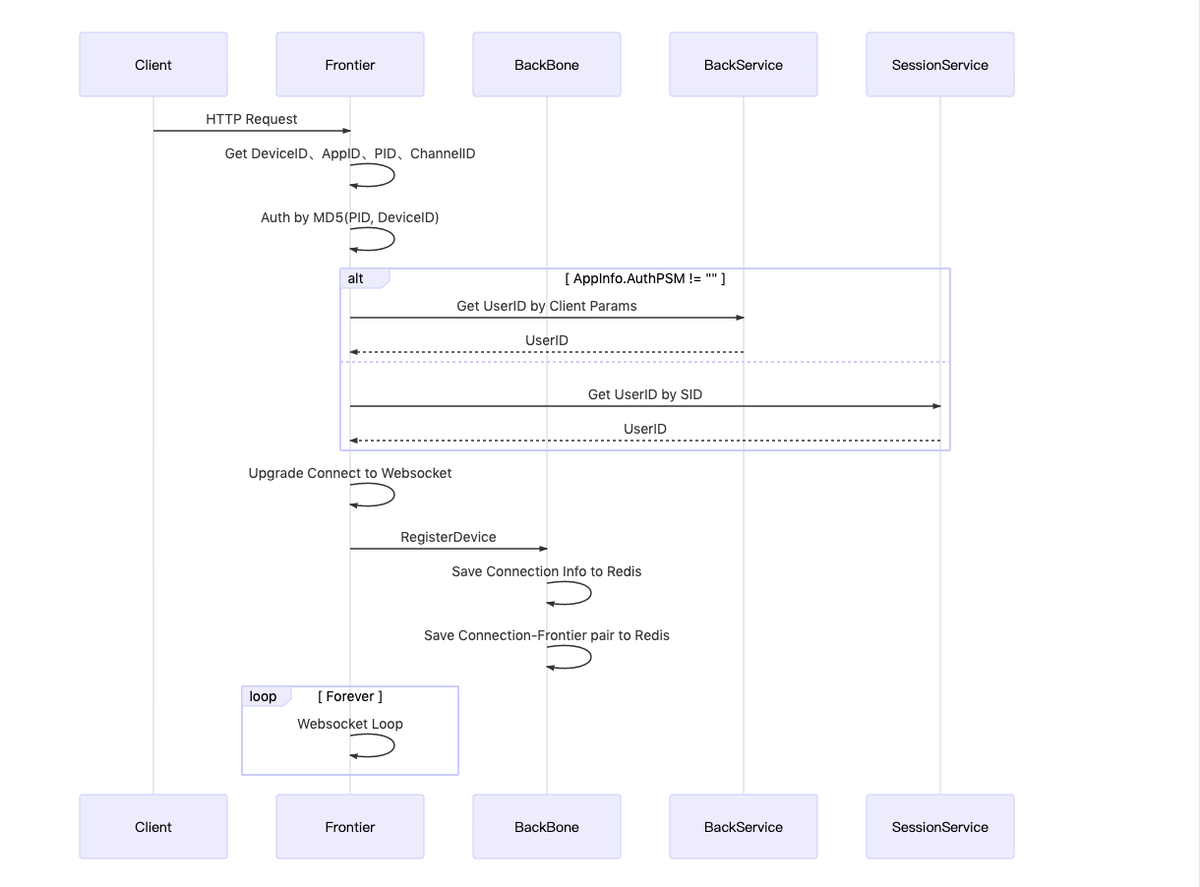
* PushByUUID：数据转成websocket帧格式，从gHub中获取连接，向连接写入数据
* PushByChannel：数据转成websocket帧格式，从gHub.GetByChannelId中获取连接，向连接写入数据

HTTP循环

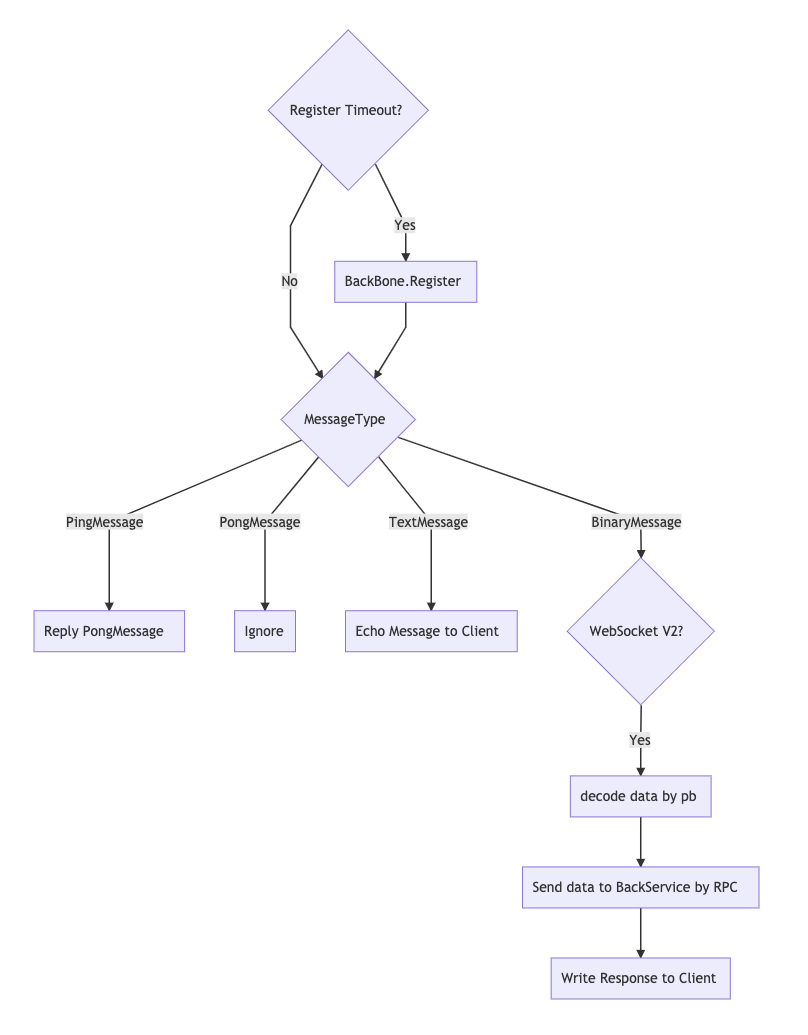
* 连接方式：目前Frontier支持两种连接方式——TCP连接和Unix Socket连接
* 路由：在​gWebsocketServer.ListenAndServe​函数中设置路由及对应的回调函数



* 建立连接的流程



* Websocket轮询



BackBone

接口

* Push：调用Frontier.PushByUUID推送消息
* PushByUUID：调用Frontier.PushByUUID推送消息
* PushByChannel：调用Frontier.PushByChannel推送消息
* QueryOnline：读取Redis中的connection，用于统计在线人数
* RegisterDevice：会将连接的相关信息（产品ID、APPID、用户ID、设备ID、客户端版本、通道ID）存在Redis中，过期时间4小时，并将Channel-Frontier的对应关系存在Redis中，过期时间为6小时
* UnregisterDevice：从Redis中移除该设备的连接

维护连接与Frontier关系

HSET toutiao.ev.backbone.ChannelID address 1

EXPIRE toutiao.ev.backbone.ChannelID 3600 \* 6

HSET UserID/DeviceID:PID:AppID:v2 UUID {PID、APPID、UserID、DeviceID、version、ChnnelID}

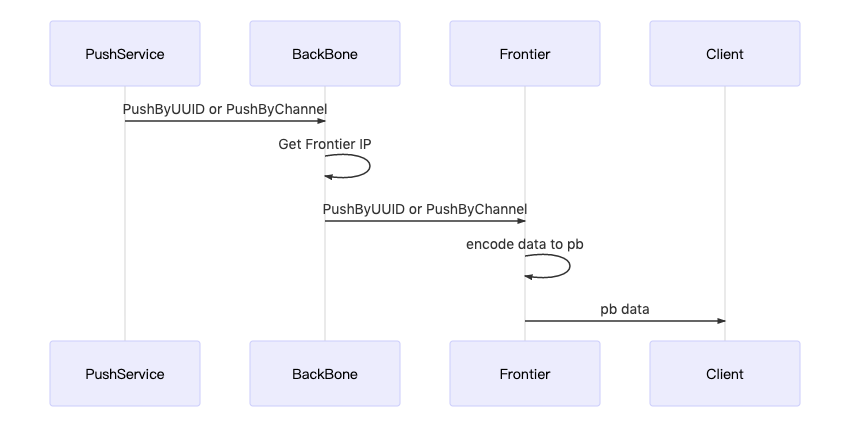
EXPIRE toutiao.ev.backbone.ChannelID 3600 \* 4

BackService

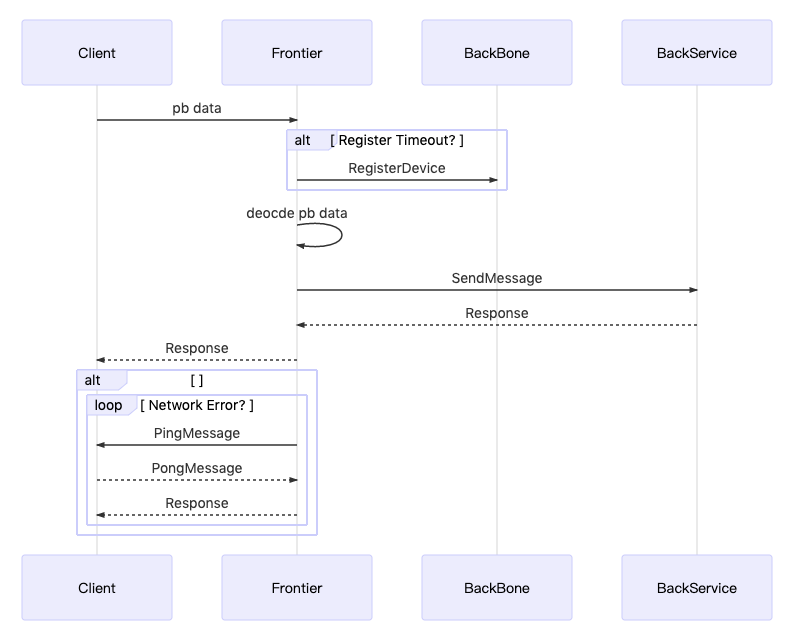
接口

* Auth：连接建立时鉴权
  + SendEvent：连接上下线事件
* SendMessage：客户端上行消息

推送消息过程



接收消息过程



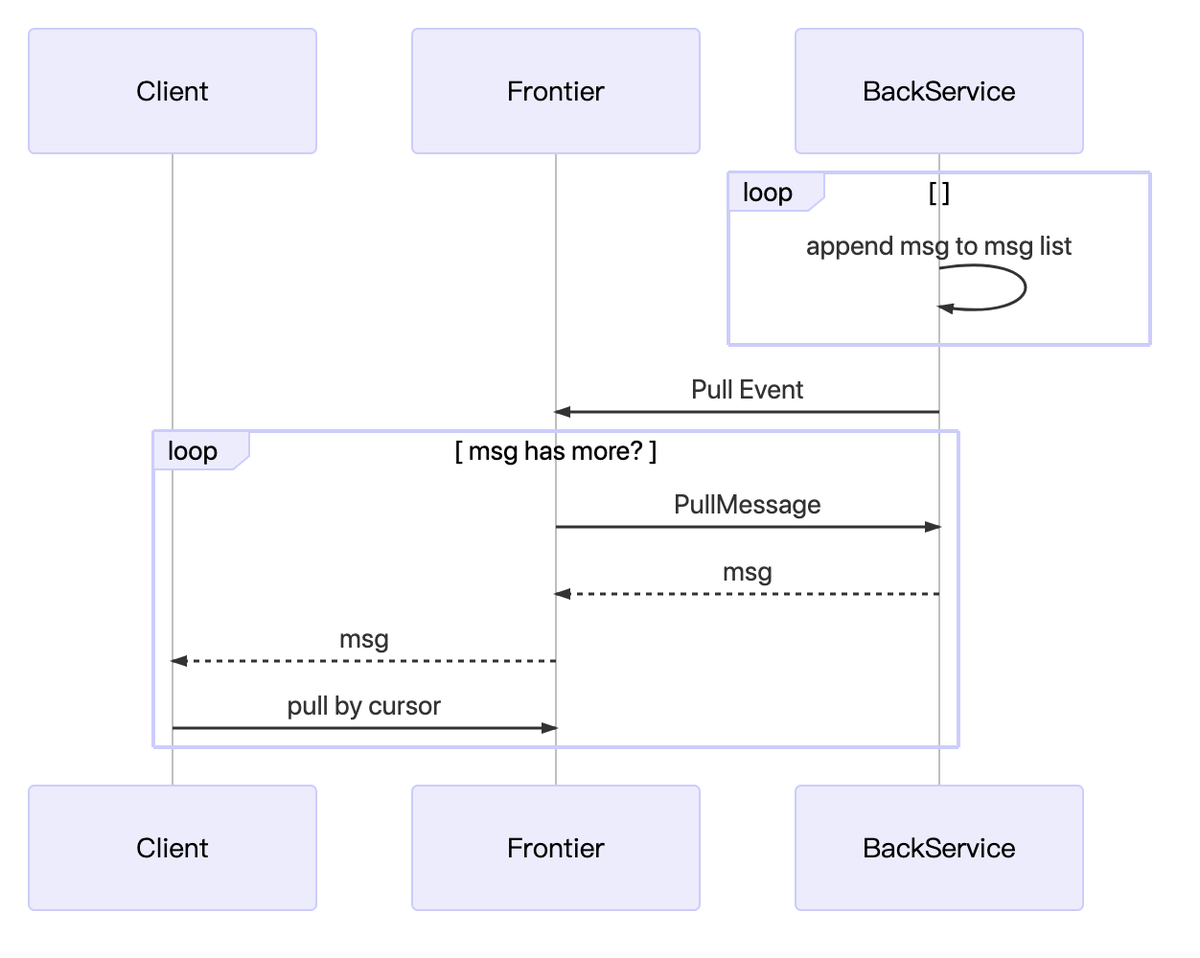
消息有序推送

名词解释

* 消息拉链：一组消息组成的链表，新消息append到链表尾部。
* Cursor：一个消息指针，消息拉链中下一条待发送的消息
* BackService：Push消息的后端服务

实施方案

1. InitPull：通过非长连接的方式Pull所有消息，并同步cursor
2. pull：普通pull



存在的问题

* 服务端不会主动进行ping pong操作，服务端发送消息的时间间隔大约为多少？
* 客户端没有ACK，只依靠TCP协议
* Frontier和BackBone没有消息缓存，如果下发失败，没办法重发
* 缺乏长连接的监控
* HTTP/2没有启用

可能的改进方案

* 服务端固定时间间隔进行ping pong，在ping操作时，可以携带部分状态机核心数据
* 在推送失败和ping pong失败时，需要打点，统计长连接推送成功率
* 启用HTTP/2
* QUIC
* RocketMQ+[MQTT](https://zh.wikipedia.org/wiki/MQTT" \t "/Users/yangwenshuo/Documents\\x/_blank)

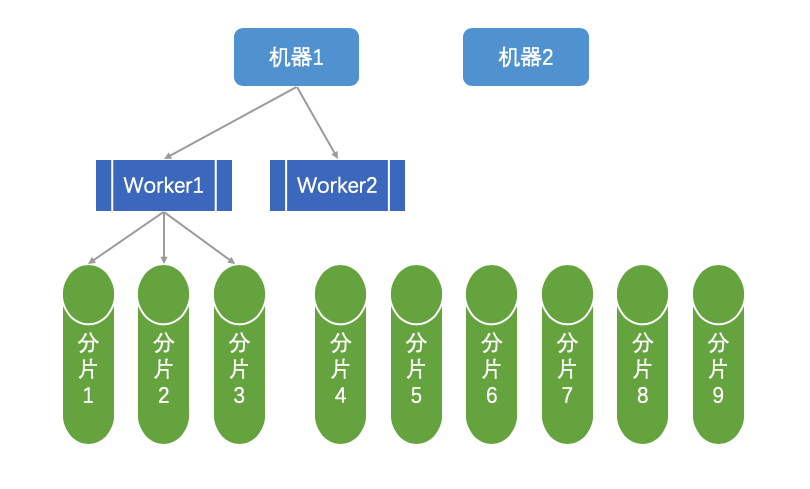
参考资料

1. [鉴权机制设计方案](http://cloud.bytedance.net/tcc/detail/795/config?region=CN&cluster=default" \t "/Users/yangwenshuo/Documents\\x/_blank)
2. [有序推送设计方案](https://wiki.bytedance.net/pages/viewpage.action?pageId=300674971" \t "/Users/yangwenshuo/Documents\\x/_blank)
3. [长连接设计方案](https://docs.google.com/document/d/1IDjl5u1lOEWHtgyHl4NY7hkhVllbKpHqohf0mpm4vms" \t "/Users/yangwenshuo/Documents\\x/_blank)
4. ​[EV 长连接需求整理](https://bytedance.feishu.cn/space/doc/DBfH6ou3DNt6wkrUj4Ddkc" \t "/Users/yangwenshuo/Documents\\x/_blank)​
5. [MQTT](https://wiki.bytedance.net/pages/viewpage.action?pageId=228295427" \t "/Users/yangwenshuo/Documents\\x/_blank)

## kafka队列积压

kafka队列积压问题定位与解决

消费过程说明



名词说明

* 机器：tce上的一个实例，一台机器可以启动多个worker，由MqConfig.WorkerNumber控制
* worker：每个worker是一个协程，从partition中轮训数据，然后串行调用Handler函数
  + 1个worker可以同时处理多个partition的数据
* 分片：partition，分片之间数据独立，所有分片的数据加起来组成队列的全量数据
  + 1个分片最多被一个worker处理数据，假如总共有20个worker但是只有10个分片，那就只有10个worker能处理数据，另外10个worker闲置
    - 极端情况下可能只有部分机器在消费数据，剩余的机器闲置，所以需要合理设置WorkerNumber

消息挤压问题定位

先通过监控确认消费有挤压

[https://grafana.byted.org/d/vEE7GNQmk/data-inf-kafka-user-consumer\_offset?orgId=1&var-consumer\_group=toutiao.ev.slardar\_to\_elk&var-topic=dali\_app\_monitor&var-topic\_related\_metric\_prefix=data.etl.kafka.risc3.monitor&var-broker\_related\_metric\_prefix=inf.kafka.risc3&var-data\_source=bytetsd&var-dc=lf&from=now-3h&to=now](https://grafana.byted.org/d/vEE7GNQmk/data-inf-kafka-user-consumer_offset?orgId=1&var-consumer_group=toutiao.ev.slardar_to_elk&var-topic=dali_app_monitor&var-topic_related_metric_prefix=data.etl.kafka.risc3.monitor&var-broker_related_metric_prefix=inf.kafka.risc3&var-data_source=bytetsd&var-dc=lf&from=now-3h&to=now" \t "/Users/yangwenshuo/Documents\\x/_blank)



点击链接就能进到对应监控页面，重点观察“生产/消费速度对比”和“整个topic的lag情况”，曲线有异常时证明有挤压

问题排查顺序及解决办法

1. 机器性能：扩机器
2. worker性能
   1. 每个worker处理多个partition导致消费能力较低：调高worker数量
   2. 每个worker只处理一个partition仍然消费不过来
      1. 提升Handler函数的性能
      2. 增加partition数量

注意事项

* 机器性能：机器CPU、内存、网卡有瓶颈时，第一时间进行扩容，然后再确认一下机器遇到瓶颈是否正常
  + 每次调整worker数、partition数后要注意观察机器性能
* worker数量的设置方式：一元一次方程：期望每个worker处理的partition数\*机器数\*worker数=总partition数
  + 如：期望每个worker处理一个partition，共有8台机器，40个partition，则方程为：
    - 1\*8\*x=40，x=5
* worker在执行handler函数时是串行执行的，如果handler函数中有较大的io耗时（rpc、读写db等）可以通过调大worker数量提高并发
  + 当每个worker只处理一个partition了消费速度还是跟不上的话，提高worker数量就没有意义了，此时可以先调高partition的数量再相应的调高worker数量，但partition数量不宜太多
* 查看每个机器/worker在处理的partition：[http://kafka-config.byted.org/queryConsumerGroupStatus.do?cluster=kafka\_risc3\_lf&consumergroup=toutiao.ev.slardar\_to\_elk](http://kafka-config.byted.org/queryConsumerGroupStatus.do?cluster=kafka_risc3_lf&consumergroup=toutiao.ev.slardar_to_elk" \t "/Users/yangwenshuo/Documents\\x/_blank)    记得把参数中的集群和group替换成你自己的

使用Python脚本直接拉取kafka数据进行调试

[https://wiki.bytedance.net/pages/viewpage.action?pageId=62433062](https://wiki.bytedance.net/pages/viewpage.action?pageId=62433062" \t "/Users/yangwenshuo/Documents\\x/_blank)

# coding: utf-8

**from** pyutil.program.conf **import** Conf

**from** kafka\_utils.simple\_consumer **import** SimpleConsumerNoGroup

**import** kafka

cluster **=** "kafka\_logcollector"

topic **=** "xxxxx"

conf **=** Conf("/opt/tiger/ss\_conf/ss/kafka.conf")

broker\_list **=** conf.get\_values(cluster)

client **=** kafka.KafkaClient(host**=**broker\_list, timeout**=**60)

partitions **=** [0,1,2]

# 消费个别partiion的数据，如果partitions为none则默认消费所有partition

consumer **=** SimpleConsumerNoGroup(client**=**client, topic**=**topic, partitions**=**partitions)

# 从某个位点偏移读取部分数据，默认为从最新数据小时消费

"""

  def seek(self, offset, whence):

        Alter the current offset in the consumer, similar to fseek

        offset: how much to modify offset

        whence: where to modify it from

            0 is relative to the earliest available offset (head)

            1 is relative to the current offset

            2 is relative to the latest known offset (tail)

"""

#consumer.seek(2, -1000)  #每个partitions消费位点向前移动1000

#指定partition和位点

# consumer.fetch\_offsets = {

#     1: 123,

#     2: 445,

# }

**while** True:

    msgs **=** consumer.get\_messages(count**=**1000, timeout**=**5)

**for** msg **in** msgs:

        print msg.message.value

放到开发机上就可以执行，执行时可能报错：

NotImplementedError: Snappy codec is not available

解决方法：

[https://github.com/dpkp/kafka-python/blob/master/docs/install.rst](https://github.com/dpkp/kafka-python/blob/master/docs/install.rst" \t "/Users/yangwenshuo/Documents\\x/_blank)

apt-get install libsnappy-dev

参考

[https://doc.bytedance.net/docs/](https://doc.bytedance.net/docs/1019/1250/9878/" \t "/Users/yangwenshuo/Documents\\x/_blank)**[1019](https://doc.bytedance.net/docs/1019/1250/9878/" \t "/Users/yangwenshuo/Documents\\x/_blank)**[/1250/9878/](https://doc.bytedance.net/docs/1019/1250/9878/" \t "/Users/yangwenshuo/Documents\\x/_blank)

## 文章

<https://studygolang.com/articles/22681>

<https://mp.weixin.qq.com/s/eDd212DhjIRGpytBkgfzAg>

<https://gobomb.github.io/post/high-concurrency-model/>

<https://juejin.im/post/5c637a35e51d45783211f387>

<https://juejin.im/post/5d7f09e55188251ecc40dcbc>

<https://www.infoq.cn/article/KNMAVdo3jXs3qPKqTZBw>