[GO实现千万级WebSocket消息推送服务 1](#_Toc9822)

[弹幕推送系统的技术挑战 1](#_Toc31238)

[弹幕技术的复杂度 1](#_Toc12913)

[拉模式与推模式的区别 2](#_Toc13755)

[拉模式 2](#_Toc4654)

[推模式 2](#_Toc17936)

[基于WebSocket推送 3](#_Toc6695)

[WebSocket协议与交互 3](#_Toc11359)

[服务端的技术选型与考虑 4](#_Toc28977)

[实现HTTP服务端 4](#_Toc31615)

[完成WebSocket握手 -- upgrade为ws的握手处理 4](#_Toc12471)

[封装WebSocket (对于TCP的封装也是很有作用的) 5](#_Toc14683)

[千万级弹幕系统的架构秘密 9](#_Toc30843)

[方案解决 10](#_Toc31269)

[单机架构 12](#_Toc6520)

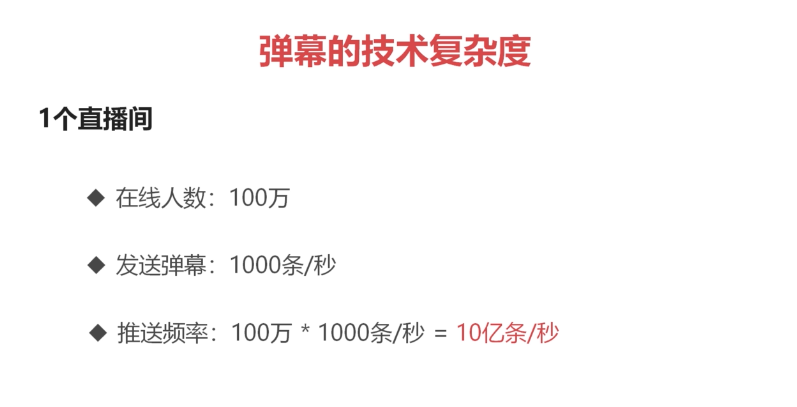
[分布式架构 12](#_Toc27190)

[课程回顾 14](#_Toc22729)

# GO实现千万级WebSocket消息推送服务

## 弹幕推送系统的技术挑战

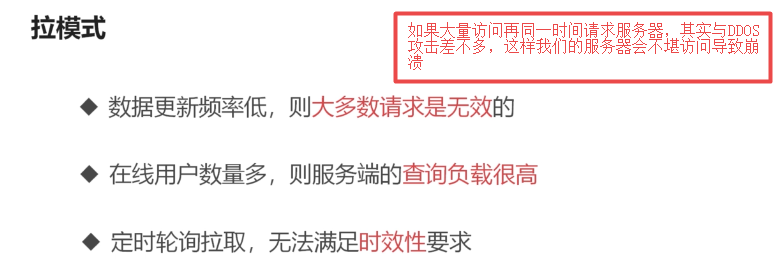
### 弹幕技术的复杂度





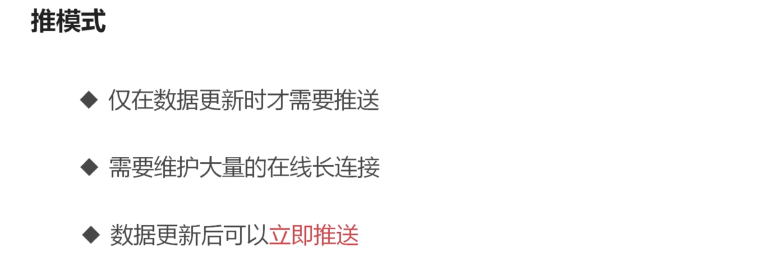
## 拉模式与推模式的区别

### 拉模式

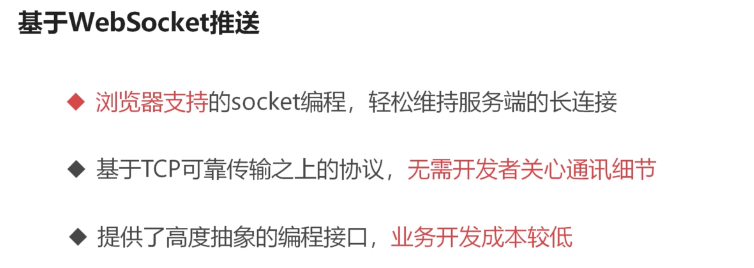


由于弹幕的高并发性和数据访问量过大的特点，所以我们不能直接使用拉模式，而是需要使用推模式

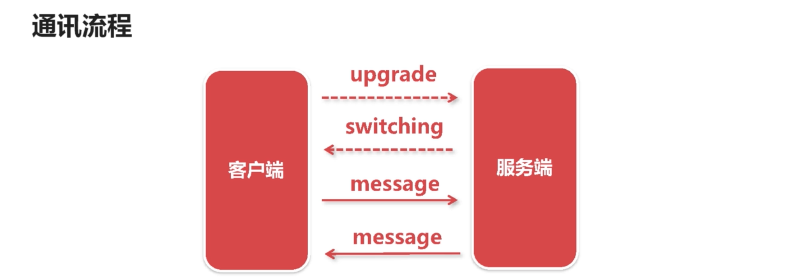
### 推模式

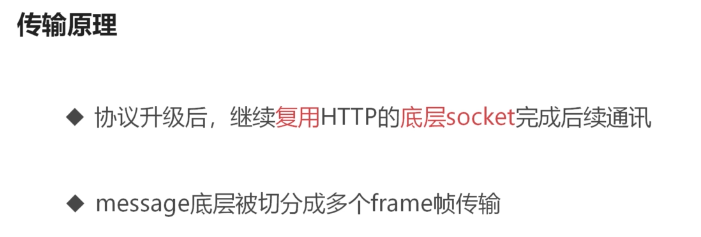


### 基于WebSocket推送



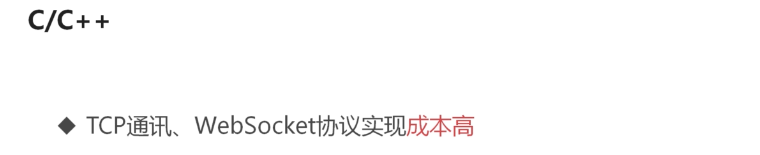
### WebSocket协议与交互

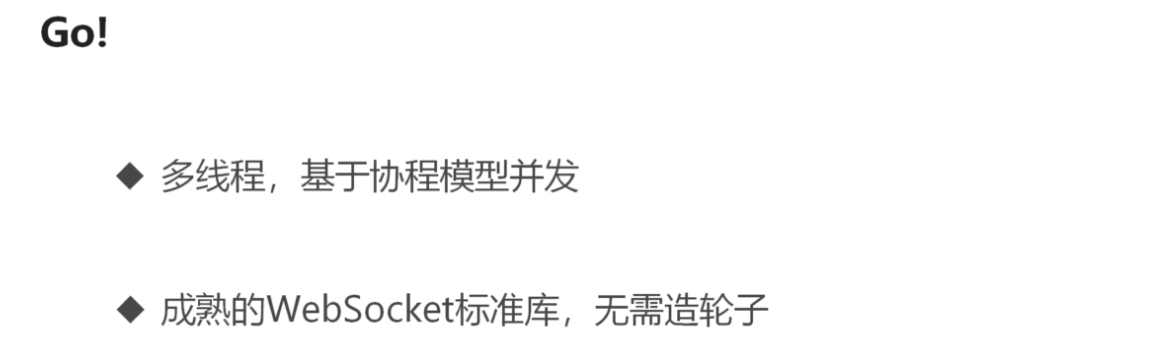




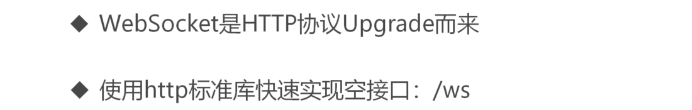
## 服务端的技术选型与考虑



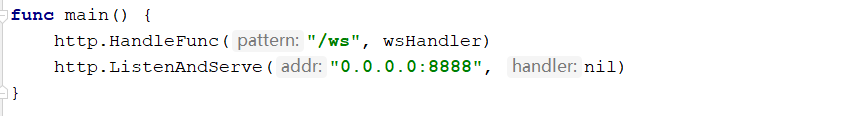
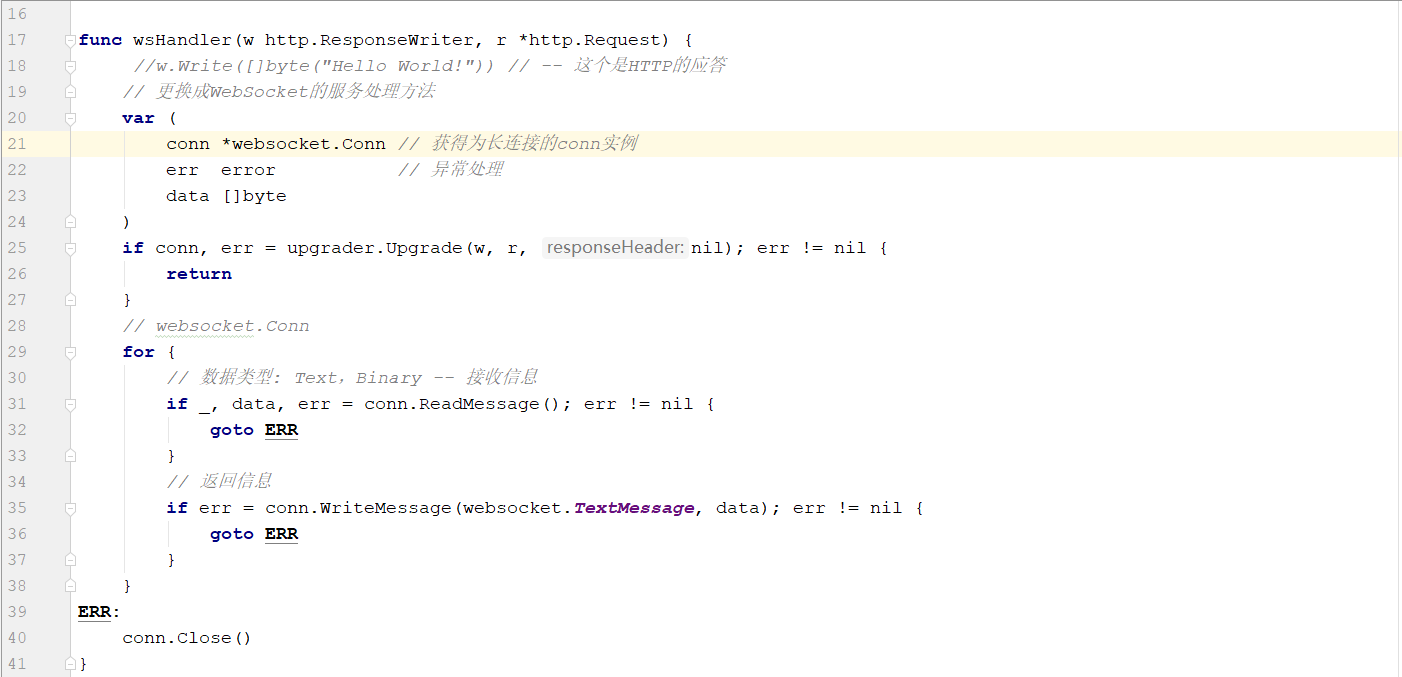
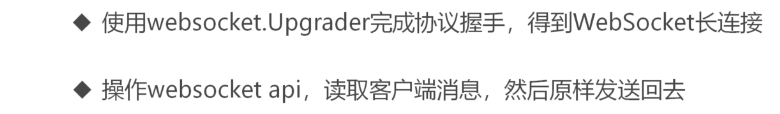




### 实现HTTP服务端



### 完成WebSocket握手 -- upgrade为ws的握手处理



## 封装WebSocket (对于TCP的封装也是很有作用的)

缺乏工程化的设计

1. 其他代码模块，无法直接操作WebSocket连接
2. WebSocket连接非线程安全，并发读/写需要同步手段

隐藏细节，封装API

1. 封装Connnection结构，隐藏WebSocket底层连接
2. 封装Connection的API，提供Send/Read/Close等线程安全接口

API原理

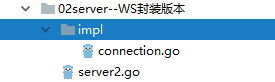
SendMessage将消息投递到out channel

ReadMessage从in channel读取消息

内部原理

启动读协程，循环读取WebSocket，将消息投递到in channel

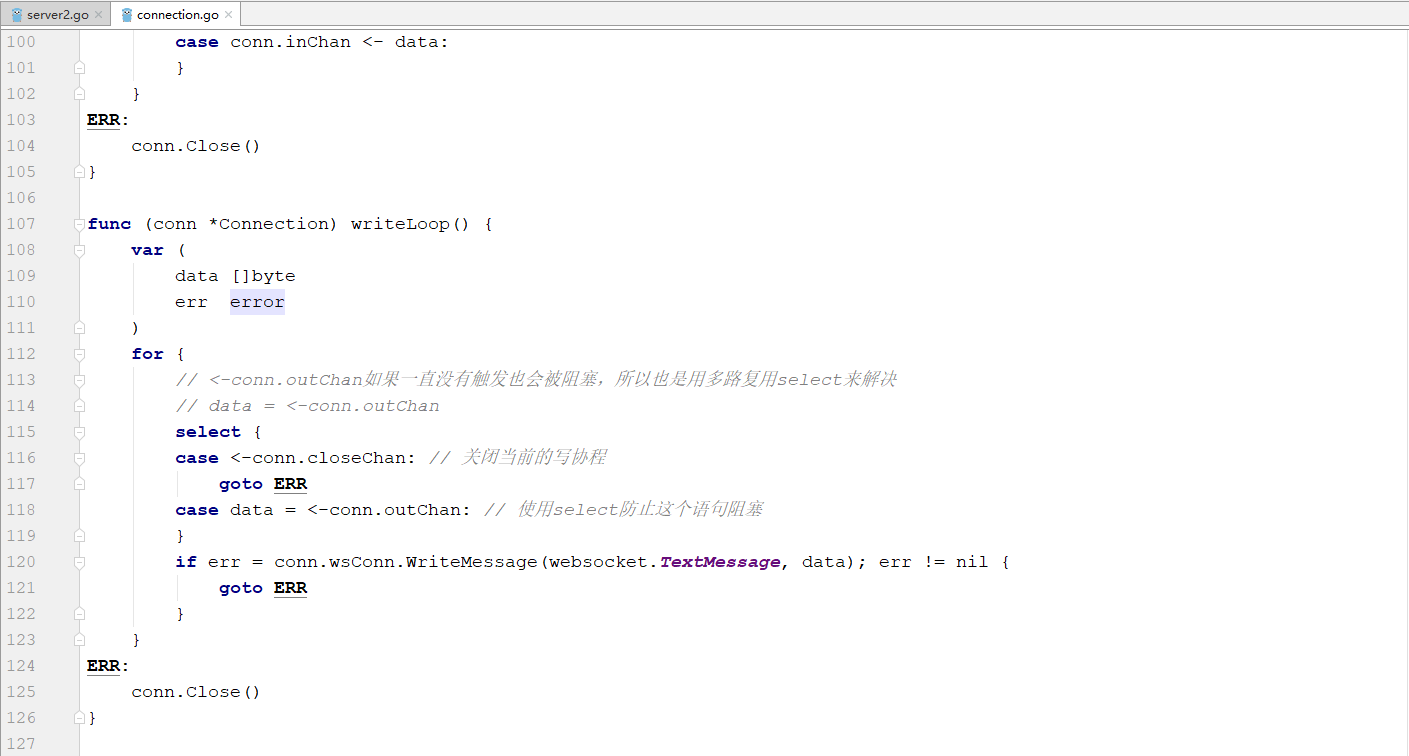
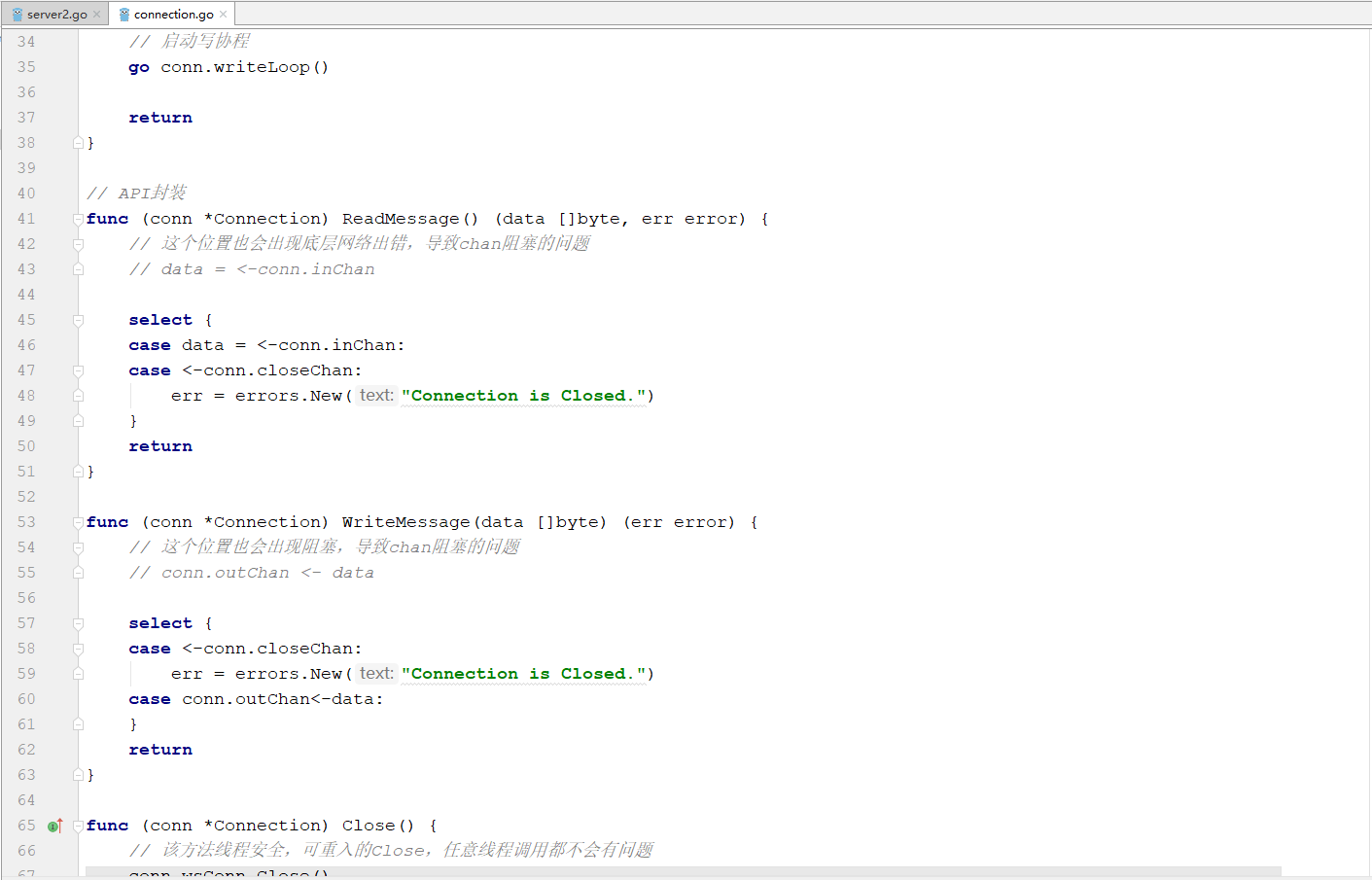
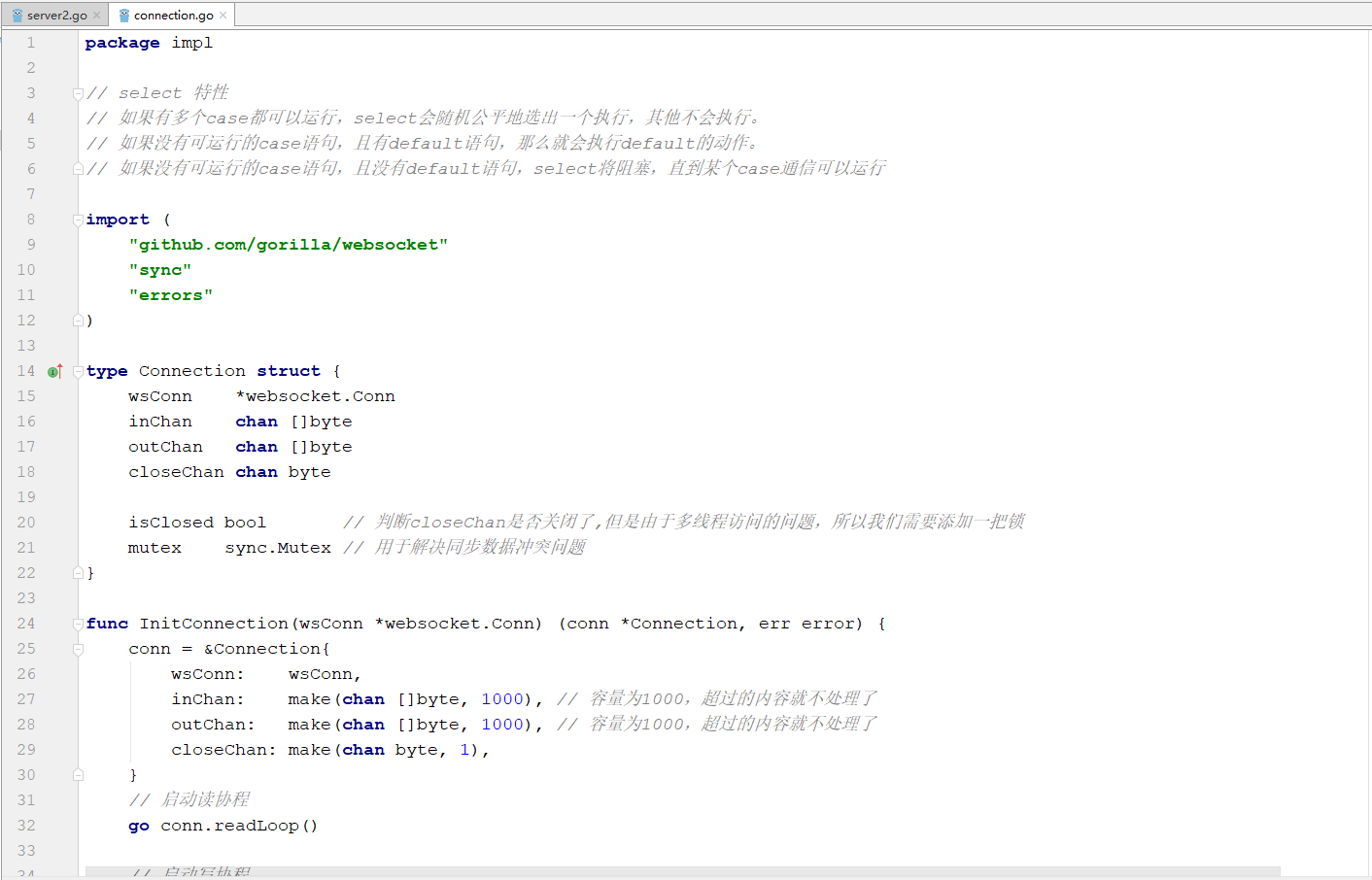
启动写协程，循环读取out channel，将消息写给WebSocket



Server.go



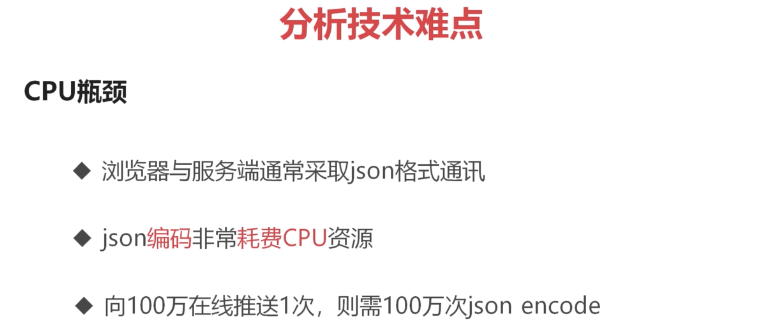
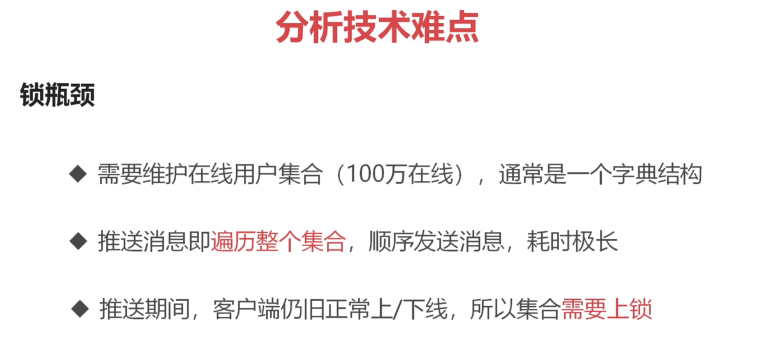
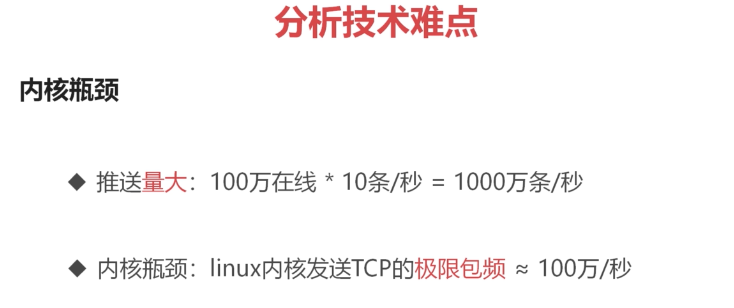
Connection.go



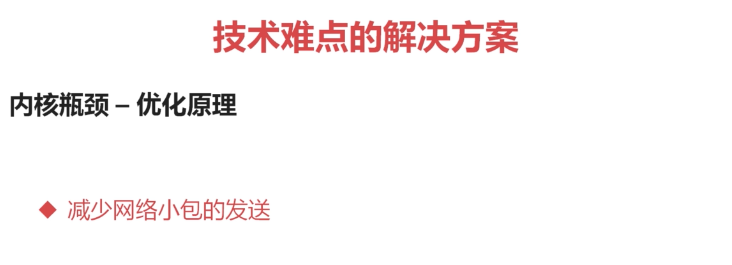


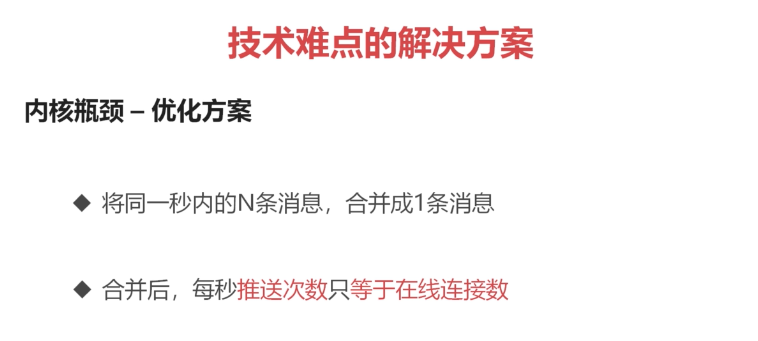
## 千万级弹幕系统的架构秘密



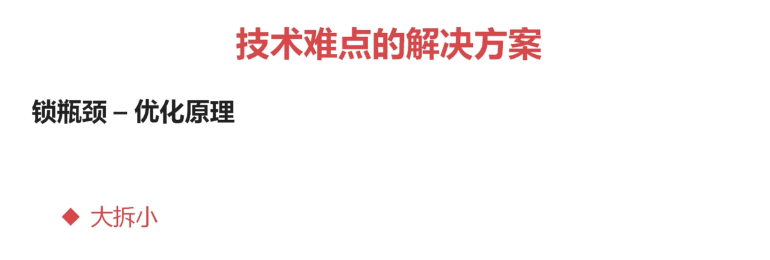


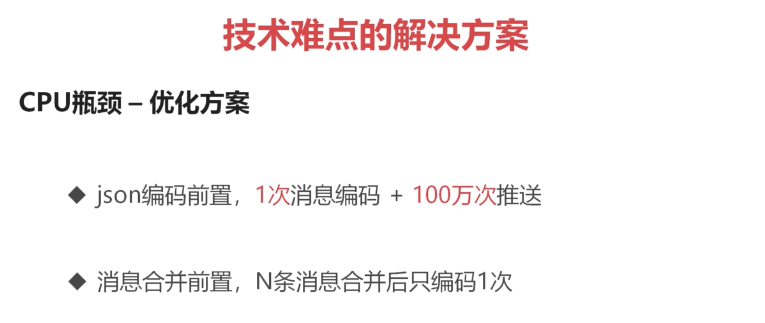
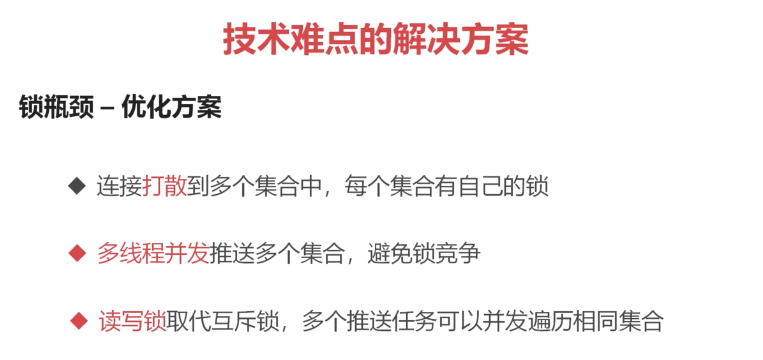
### 方案解决



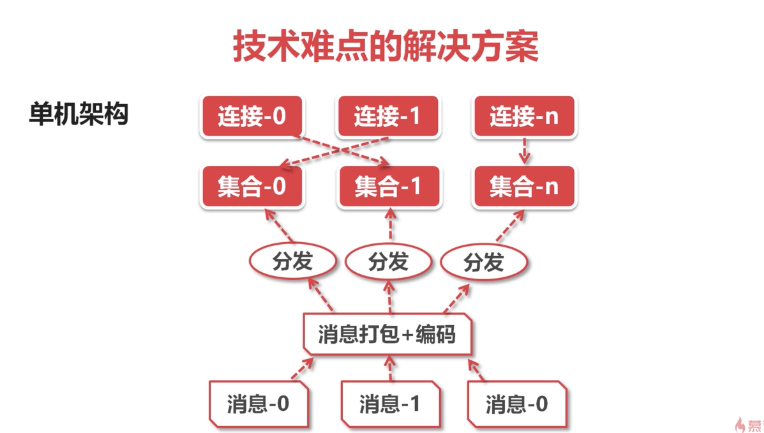


这样的话发一条100w用户，一秒就是100w次，20条一百万用户也是100w次，以此减少io的操作





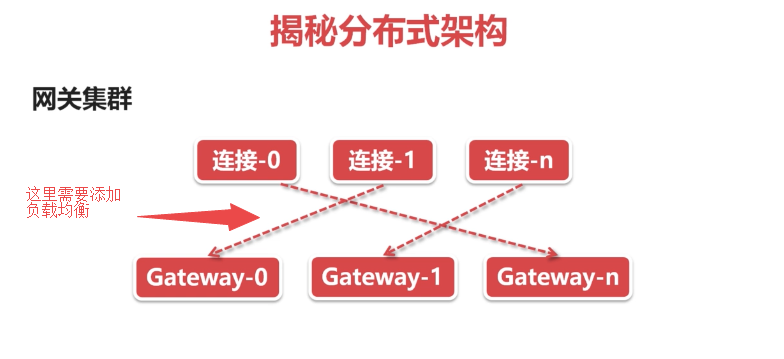
### 单机架构



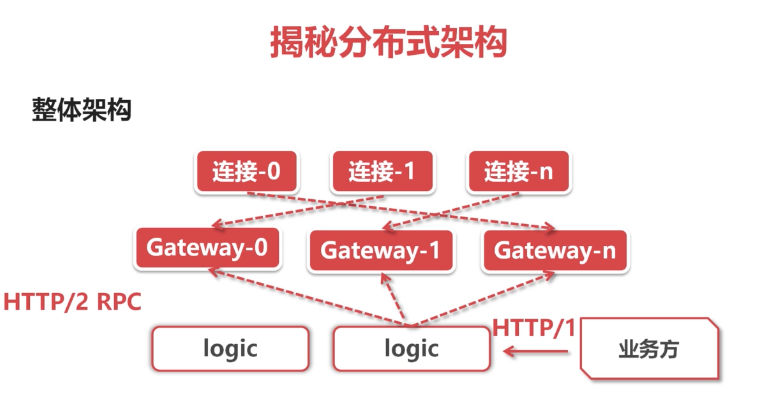
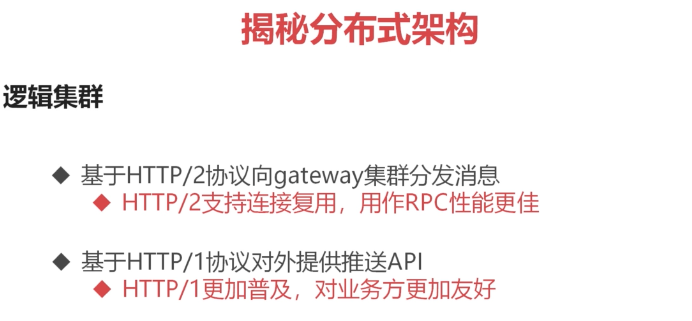
### 分布式架构



主要瓶颈是内存，CPU还有最重要的是带宽,因为一般的带宽仅仅是千兆网卡，远没有达到G级别的网卡，这样单机的主要瓶颈就体现了问题了



网关级别的消息推送，这样才能让消息发送到指定主播的指定弹幕频道上



## 课程回顾





