高可用架构设计与实践

讲师：孙玄@58

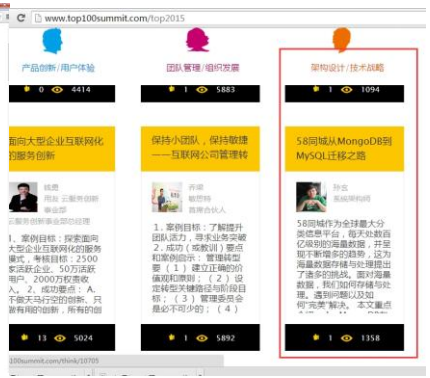
法律声明

关于我

- 58同城高级系统架构师
- 公司技术委员主席
- 即时通讯、转转、C2C技术负责人
- 前百度高级工程师
- 代表58同城对外嘉宾分享
  - QCon
  - SDCC
  - DTCC
  - Top100
  - 程序员
  - UPYUN
  - TINGYUN
  - ……

代表58对外交流

- Qcon(全球软件开发大会)
- SDCC(中国开发者大会)
- Top100(全球案例研究峰会)
- DTCC(中国数据库技术大会)
- 《程序员》撰稿2次
- 58技术发展这10年[计划中]



炼数成金课程

课程

- 《MongoDB实战》
  - 已开课
  - 欢迎大家报名学习
- 《大规模高性能分布式存储系统设计与实现》
  - 已开课
  - 欢迎大家报名学习

上节课回顾

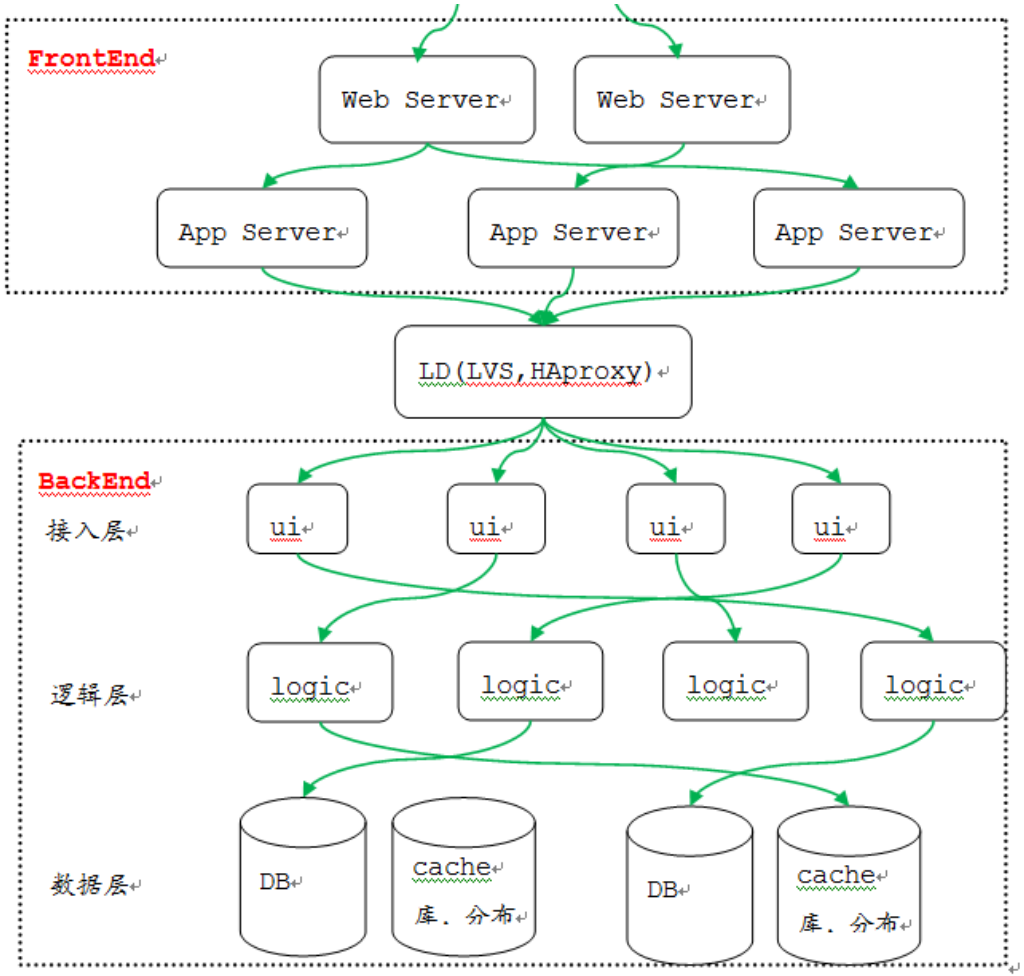
- CDN系统架构高可用涉及技术点都有哪些
- CDN系统为什么要使用
- CDN系统发展进程
- CDN系统国内使用情况
- CDN系统应用领域
- CDN数据一致性如何保证
- 我们的实践案例

OutLine

- 互联网产品通用技术架构
- 接入层的作用是什么
- 接入层Session如何设计?
  - Session复制、Session绑定、Session记录方式、Session高可用等
- 接入层数据安全如何保证?
  - 对称加密、非对称加密、多种方法使用等
- 接入层数据正确性如何保证?
- 接入层高可用设计方案?
  - 无状态、动态扩展
- 接入层高可用设计最佳实践是什么?
- 我们的实践案例;

互联网产品通用技术架构

- Data Flow
  - webServer
  - AppServer
  - LD（LVS,HAProxy…）
  - BackEnd Server
    - 接入层
    - 提交层
    - 业务处理层
    - 数据层
  - 本文重点关注
    - BackEnd Server



58帮帮技术架构

- 线上情况
  - 模块
    - 30+
    - JAVA/CPP
  - 请求
    - 10亿(IM)+30亿(!IM)
    - 同时在线用户数突破100w+
  - 机器
    - 百台+

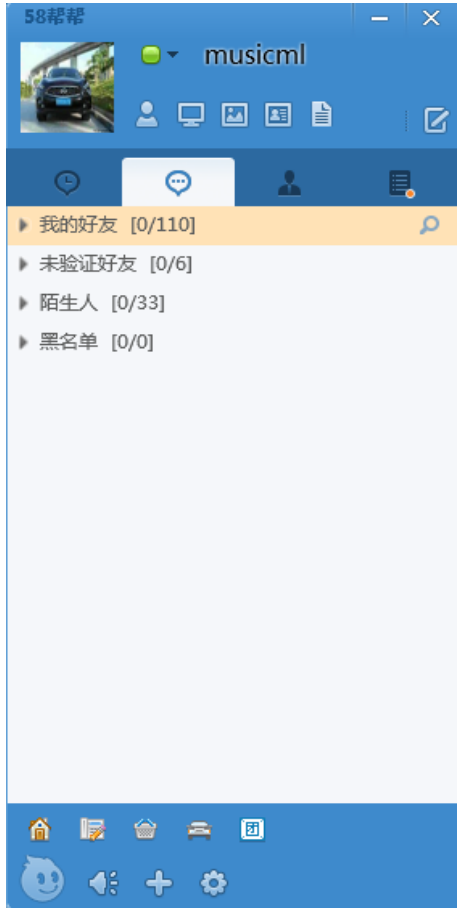
58帮帮技术架构

定位

- 传统IM
- 满足58用户与商户沟通，获取信息

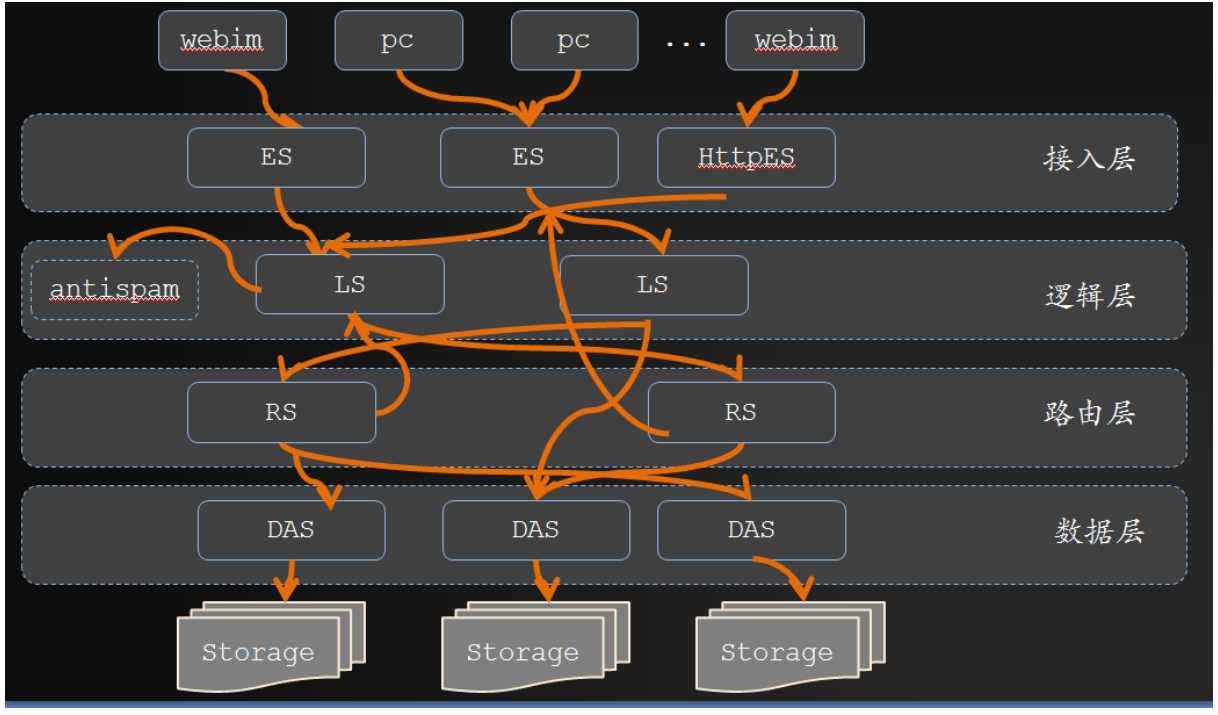
核心功能

- 用户关系
- 添加好友
- 发送消息



58帮帮技术架构

传统IM技术架构



58帮帮技术架构

传统IM架构如何满足千万同时在线性能？

- **接入层**、逻辑层、路由层、数据层
- 无状态设计
- 每层模块动态高扩展
- 模块冗余，高可用性保证
- 动态负载均衡，动态切换可用服务节点
- 优化效果
  - 单机线上支持50W+同时在线
  - 单机线上3w+qps

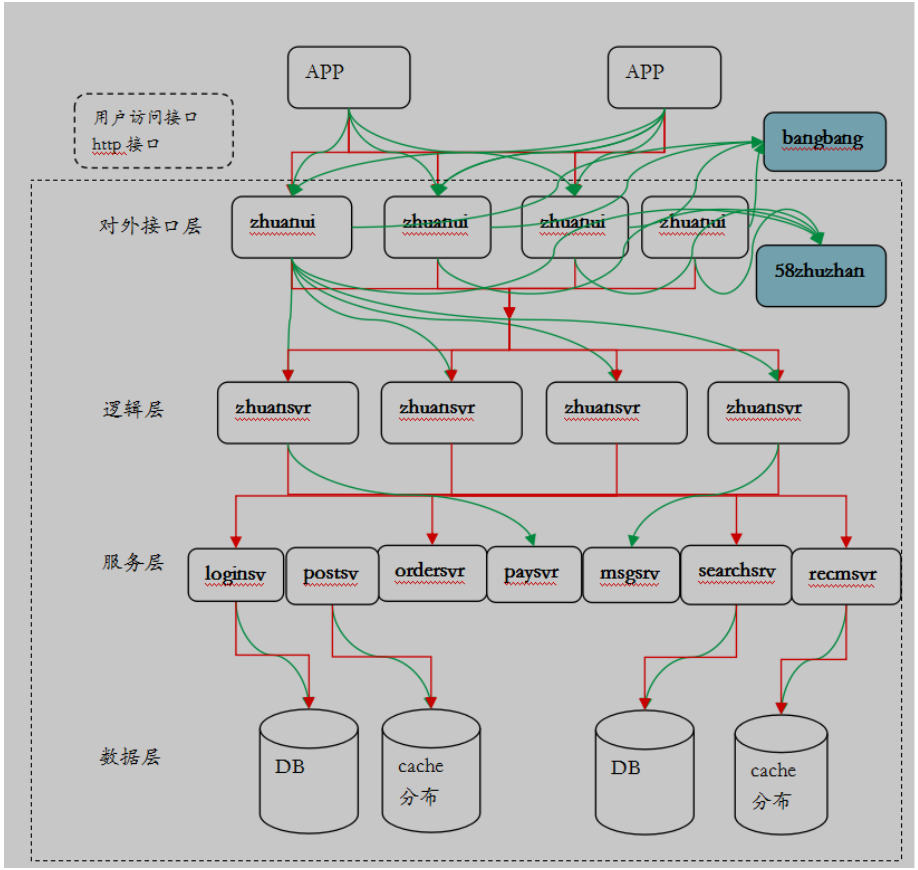
58转转技术架构

- 定位
  - 全国最大的个人真实闲置交易平台
- 功能
  - 用户
  - 商品
  - 社交
  - 交易
  - 圈子
  - 推荐
  - 搜索
  - 运营
  - ……

58转转技术架构

架构如何设计

- 功能多
- 业务复杂
- 高可用性
- 交易高安全
- 未来扩展
- 低耦合分层架构
  - 对外接口层
  - 逻辑层
  - 原子服务层
  - 数据层
- ……



百度空间feed系统架构

Feed系统

百度空间feed系统架构

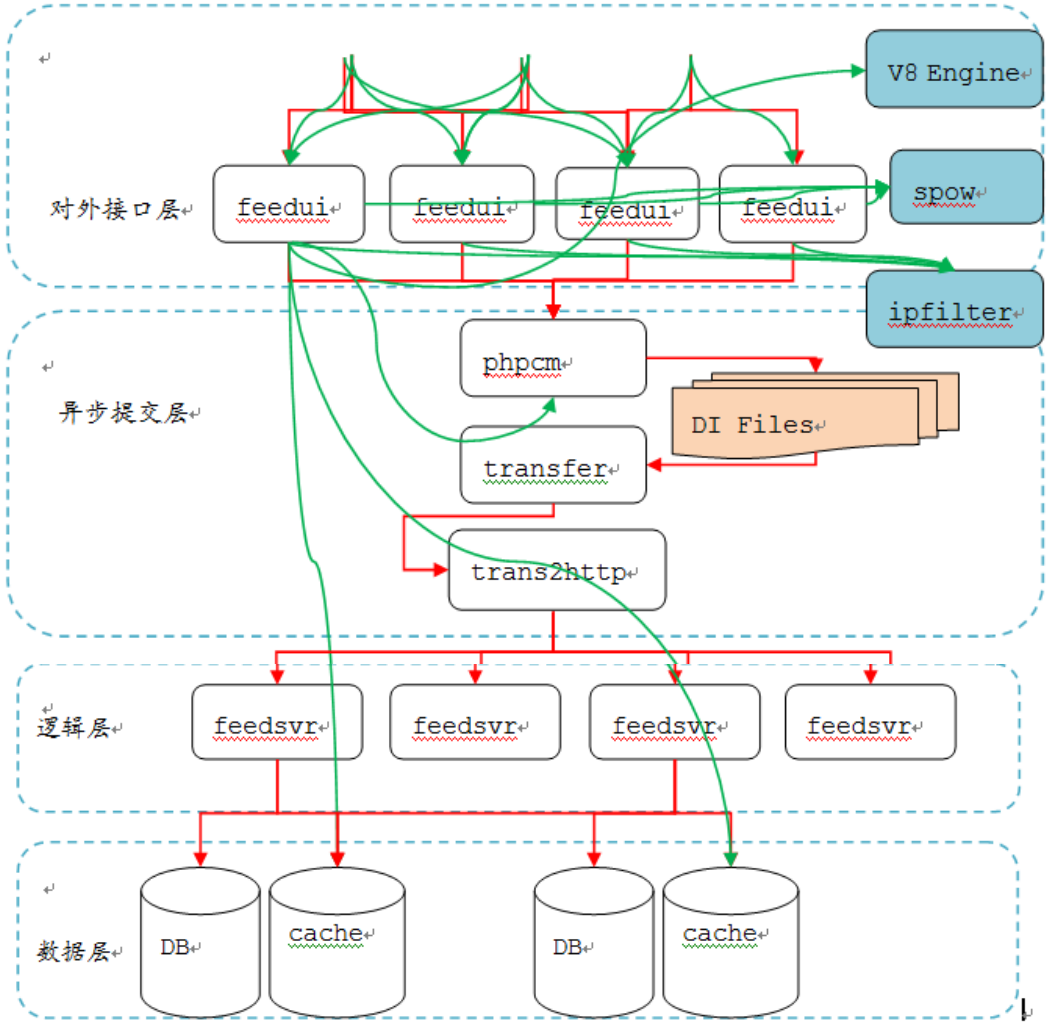
Feed系统关注问题

- 获取好友的feed
- 组合好友的feed聚类展示
- 一般按照feed发布时间倒序展现
- Push or Pull
  - Pull

百度空间feed系统架构

feed系统最终架构

- 对外接口层
- 异步提交层
- 逻辑层
- 数据层



接入层的作用是什么

接入层作用

- 客户端海量长/短连接管理
  - TCP/HTTP[S]
- 建立与客户端通信的加密通道
- 数据合法性、正确性校验
- 整合成内部少量的长连接
- Session的管理
- 实施初步的攻防
- 请求转发到逻辑层

接入层Session如何设计

Session

- Session是什么
  - 读写请求使用的上下文对象，称之会话(Session)
- 高可用主要基于服务无状态
- 事实上业务总是有状态的，为什么？
  - 二手电商网站【转转】，需要记录用户下单购买商品等
  - IM系统中，需要记录用户当前登录状态、好友状态、消息发送情况等
- 这些有状态的信息会随用户操作变化而发生更新

接入层Session如何设计

单机环境设计

- 单机不存在Session共享的问题
- 处理比较简单
- Session放在本机内存
- 高可用无法保证
  - 服务进程挂掉
  - 宕机
  - Session丢失，不可用
- 怎么搞?

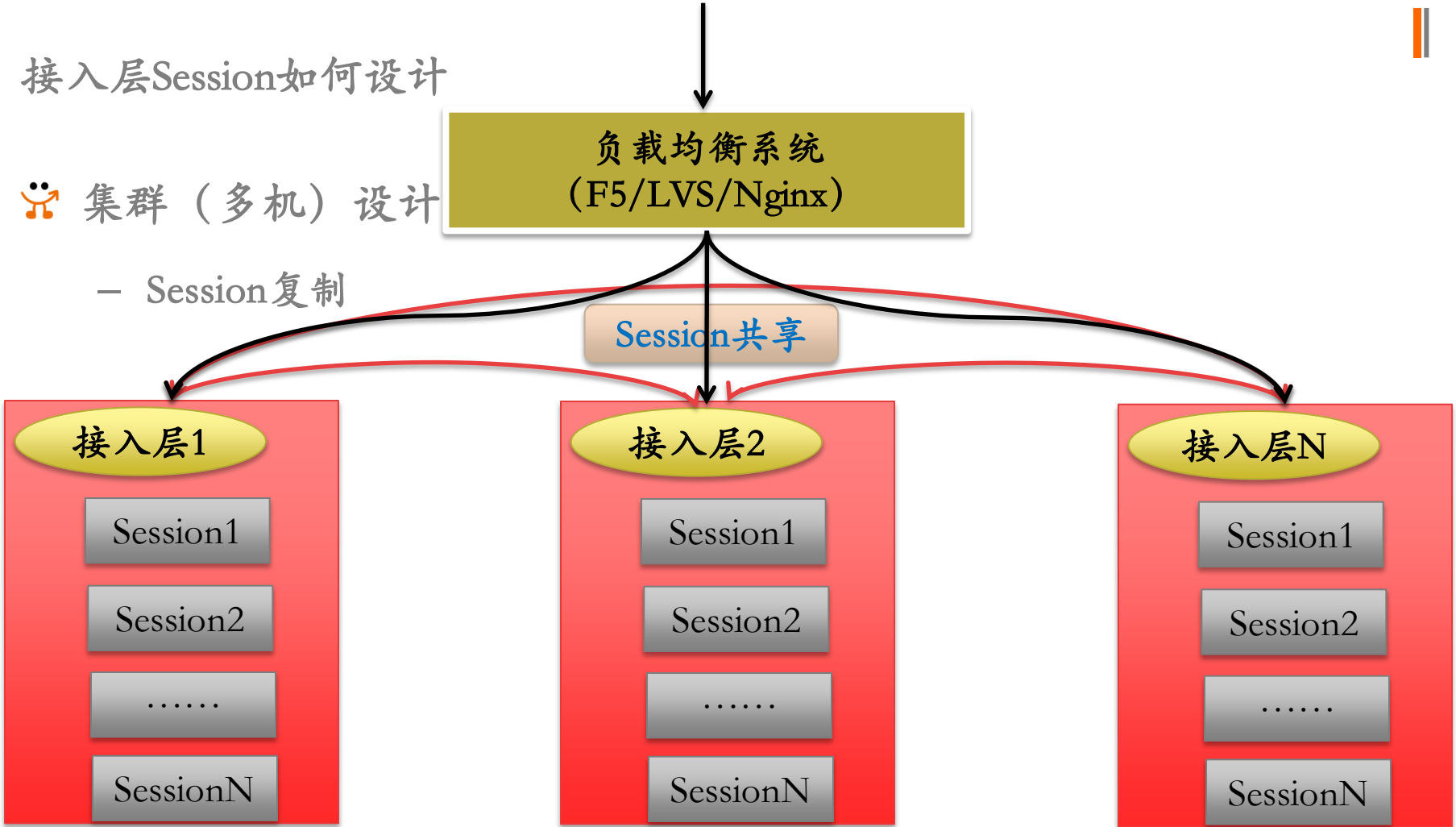
接入层Session如何设计

集群（多机）设计

- Session复制
  - 集群的所有接入层服务器之间同步Session数据
  - 每台接入服务器都保存用户全量的Session数据
  - 用户请求只需要访问其中一台机器，获取速度快
  - 高可用保障
    - 宕机部分机器，没影响

接入层Session如何设计

- 集群（多机）设计
  - Session复制



接入层Session如何设计

集群（多机）设计

- Session复制
  - 存在问题
    - 适用于接入层集群较少
    - 接入层集群量大
      - » 大量的Session复制通信，占用服务器和网络资源
      - » 每台机器存储全量用户Session，内存占用量大，甚至Out Of Memory
      - » 大型网站接入层数千台，同时在线用户达到千万（IM），不适合

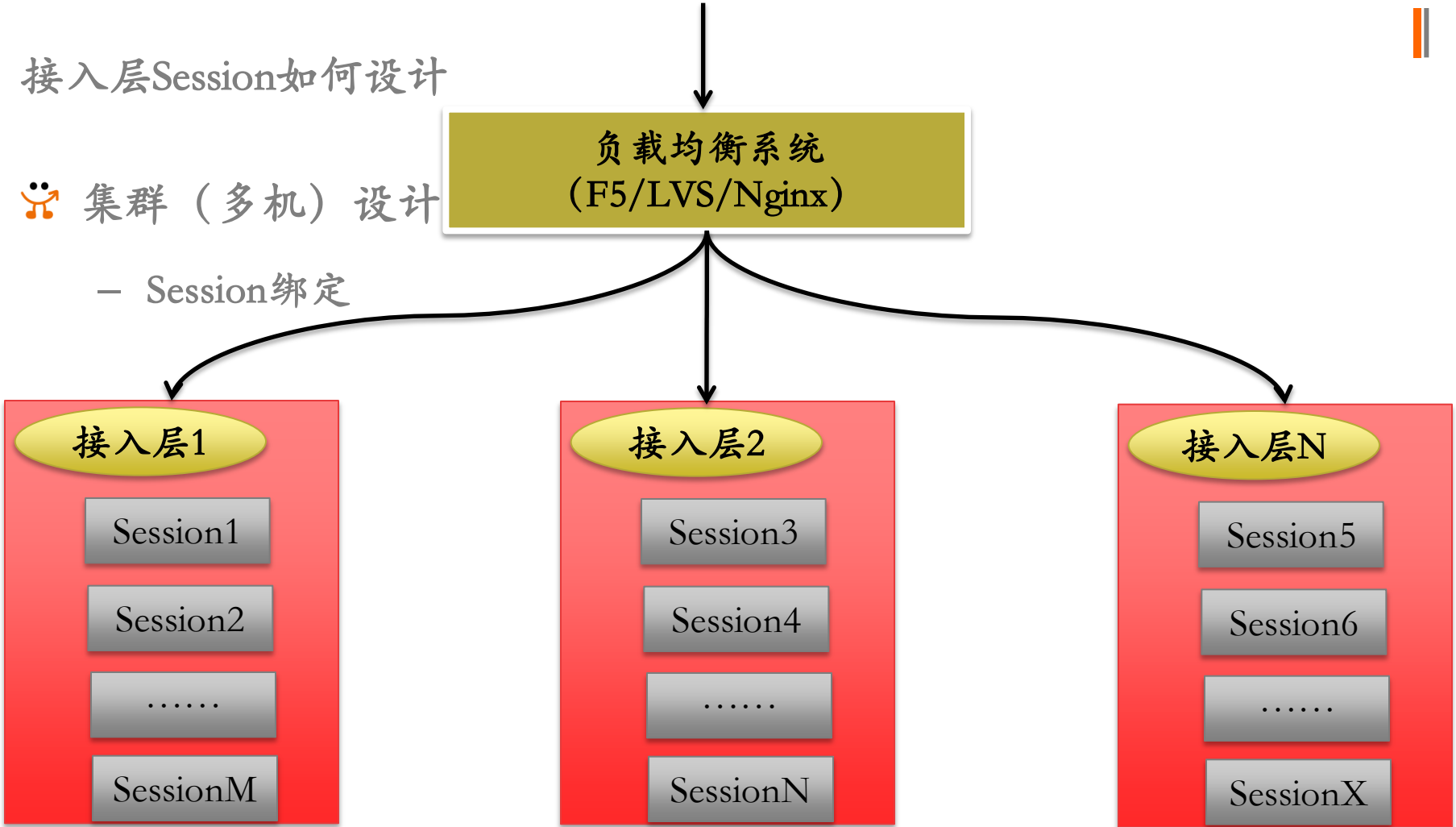
接入层Session如何设计

集群（多机）设计

- Session绑定
  - 根据用户请求（UID、Mac、imei等用户唯一标示)负载均衡到特定接入层
    - HASH(ID)
      - uid%Num
  - 特定用户请求路由到特定接入层服务器
  - 部分网站使用
  - 高可用如何保障
    - 单点问题
    - 复制机制
      - Master-Slave

接入层Session如何设计

- 集群（多机）设计
  - Session绑定



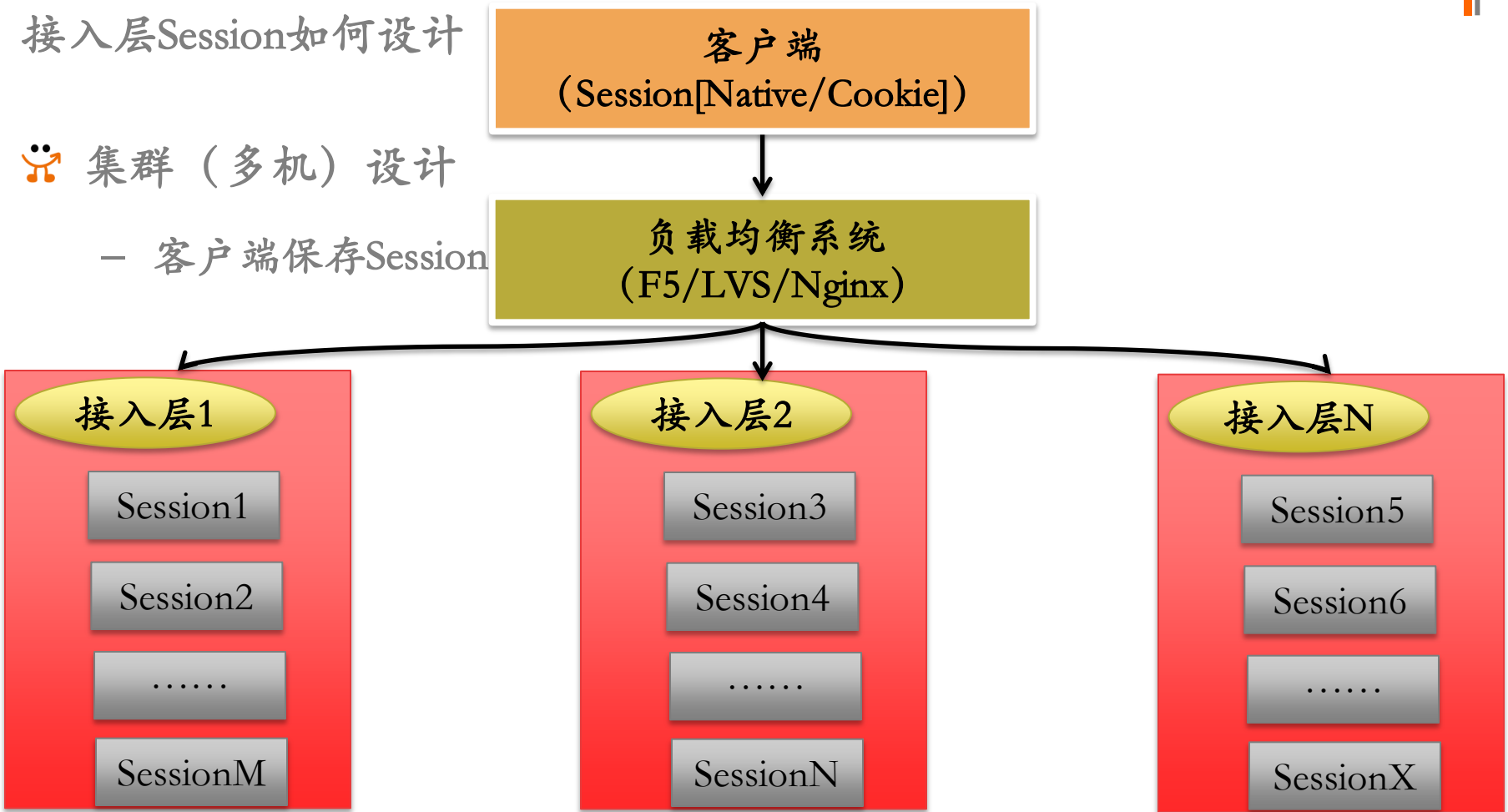
接入层Session如何设计

集群（多机）设计

- 客户端保持Session
  - Session由服务端生成，存储到客户端
  - 每次请求携带客户端Session
  - 服务端若有更新返回给客户端存储
  - C/S
    - Apps
      - » 记录到Native中
  - B/S
    - Web
      - » 记录到Cookie中

接入层Session如何设计

- 集群（多机）设计
  - 客户端保存Session



接入层Session如何设计

集群（多机）设计

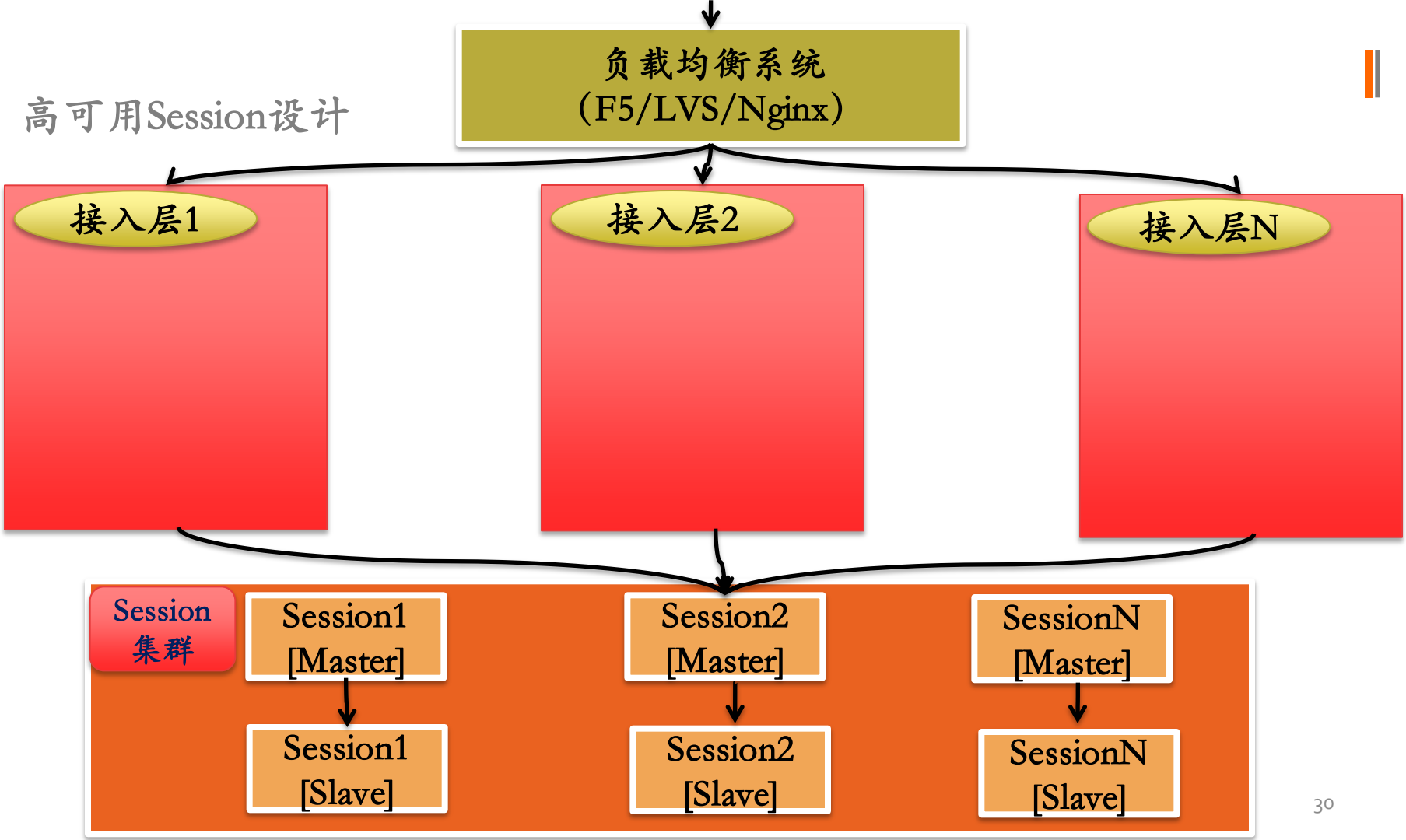
- 客户端Cookie保存Session
  - 缺点
    - Web Cookie中记录信息大小限制
      - 比如：100KB
    - 每次请求都要传输Session
      - 流量、性能受影响
    - 用户关闭、清理掉Session，用户请求不正常
  - 优点
    - 方案简单，支持服务端的无缝伸缩
    - 方案可用性高
    - 较多网站都有使用

接入层Session如何设计

高可用Sesson设计

- Session高可用集群
  - 接入层无状态化
  - 统一的高可用Session服务器
  - 接入层分布式读写Session集群
  - 状态分离
    - 接入层本身无状态
    - Session集群有状态
      - » 分布式缓存
        - √ NoSQL（Memcached/Redis）
        - √ RDBMS（MySQL/MongoDB）

高可用Session设计



接入层数据安全如何保证?

接入层安全性

- 接入层是客户端和服务端的Interface
- 数据安全重要性不言而喻
- 保证数据安全性
  - 连接通道加密
  - 传输数据加密

接入层数据安全如何保证?

- 复杂网络环境下客户端高效与服务端建立安全信道方法
  - 解决客户端与服务端实现加密会话问题
  - 适用于一切客户端
    - 58帮帮
    - 58转转
    - ……

接入层数据安全如何保证?

名称解释

- 对称加密算法:
  - 加密和解码使用同一密钥的加密方案（AES）
- 非对称加密算法
  - 加密和解密使用一对密钥（由两个满足一定关系的密钥组成的密钥对）中不同密钥的加密方法（RSA）
- 公钥
  - 非对称加密算法中公开给大众保密的密钥
- 私钥
  - 非对称加密算法中留给个人保密的密钥
- 会话状态
  - 描述客户端与服务器的一次连接的所有信息集合

接入层数据安全如何保证?

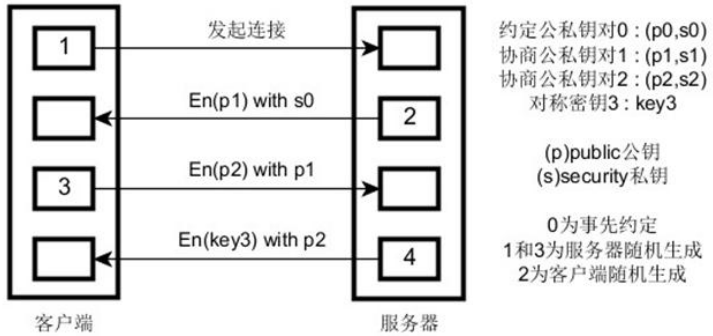
技术实现方案

- 客户端和服务器之间的所有请求(传输数据)都必须加密，提高效率，使用对称加密算法;
- 对称加密密钥使用非对称加密算法经过两次协商确定
- 安全信道的建立必须满足
  - 任何第三方无法伪造服务器
  - 在破解客户端代码的情况下，即使截获其他用户发送的加密请求，也无法解密

接入层数据安全如何保证?

技术实现方案

- 为了满足以上两个条件，客户端和服务器都必须要一个随机生成密钥的过程
- 具体的四步握手（Client->C Server->S)



接入层数据安全如何保证?

技术实现方案

- 约定公私钥对0：写死在代码中的公私钥（公私钥池，服务器每次选一个，并告诉客户端每次选中的是池中的哪一个），用于客户端验证请求的确来自服务器;
- 协商公私钥对1：服务器随机生成的协商密钥;
- 协商公私钥对2：客户端随机生成的协商密钥;
- 对称密钥3：服务器随机生成的对称密钥，用于最终的对称加密，通讯密钥

接入层数据安全如何保证?

- 使用HTTPS
  - HTTPS
    - 提供了数据安全的加密方式
    - 单向加密
    - 双向加密
  - 使用场景
    - 交易/支付
    - 金融
    - 用户信息
    - ……

接入层数据安全如何保证?

- 使用HTTPS
  - HTTPS
    - 单向加密
      - 不安全
      - 中间人攻击

接入层数据安全如何保证?

- 使用HTTPS
  - HTTPS
    - 双向加密
      - 安全
    - 客户端证书
      - 配合
  - 接口分级
    - HTTPS
    - HTTPS+短信验证

接入层数据正确性如何保证?

数据加密

- 解决数据明文的问题
- 即使截获，无法破解明文
- 数据篡改无法避免
- 数据正确性需要保证
  - 如何保证?

接入层数据正确性如何保证?

- 如何保证
  - 数字签名
    - 双方约定规则签名
      - md5sum
      - 其他
  - 过程
    - 客户端按照约定签名
    - 服务端收到数据，按照规则生成md5sum值
    - 和数据包里md5sum值比较是否一致
    - 一致说明没问题
    - 不一致数据被篡改
    - 丢弃策略

接入层数据正确性如何保证?

数字签名举例

- 第一步，设所有发送或者接收到的数据为集合M，将集合M内非空参数值的参数按照参数名ASCII码从小到大排序（字典序），使用URL键值对的格式（即key1=value1&key2=value2…）拼接成字符串stringA。
- 重要规则：
  - 参数名ASCII码从小到大排序（字典序）；
  - 如果参数的值为空不参与签名；
  - 参数名区分大小写；
  - 验证调用返回或微信主动通知签名时，传送的sign参数不参与签名，将生成的签名与该sign值作校验。

接入层数据正确性如何保证?

数字签名举例

- 第二步，在stringA最后拼接上key得到stringSignTemp字符串，并对stringSignTemp进行MD5运算，再将得到的字符串所有字符转换为大写，得到sign值signValue。

接入层数据正确性如何保证?

数字签名举例

- 假设传送的参数如下：

```
appid： wxd930ea5d5a258f4f
mch_id： 10000100
device_info： 1000
body： test
nonce_str： ibuaiVcKdpRxkhJA
```

- 第一步：对参数按照key=value的格式，并按照参数名ASCII字典序排序如下：

```
stringA="appid=wxd930ea5d5a258f4f&body=test&device_info=1000&mch_id=10000100&nonce_str=ibuaiVcKdpRxkhJA";
```

- 第二步：拼接API密钥：

```
stringSignTemp="stringA&key=192006250b4c09247ec02edce69f6a2d"
sign=MD5(stringSignTemp).toUpperCase()="9A0A8659F005D6984697E2CA0A9CF3B7"
```

- 最终得到最终发送的数据：

```
<xml>
<appid>wxd930ea5d5a258f4f</appid>
<mch_id>10000100</mch_id>
<device_info>1000<device_info>
<body>test</body>
<nonce_str>ibuaiVcKdpRxkhJA</nonce_str>
<sign>9A0A8659F005D6984697E2CA0A9CF3B7</sign>
<xml>
```

接入层数据正确性如何保证?

数字签名

- 安全进一提升
  - 约定固定字符串，参与加密
    - securityStr（key）=scry33@#$%3
  - 只有双方知道

高可用接入层如何设计?

模块和数据分离

- 接入层模块无状态
  - 动态线性伸缩
  - 冗余
- Session数据统一分布式存储
  - 数据冗余保证
  - 高可用性保证

高可用接入层最佳实践是什么

- 模块和数据分离
- Session绑定
  - 每个Session同步复制
- 不存储Session
  - 接入层

我们的实践案例

项目背景

- 全国最大的真实C2C交易平台
  - 同之前叙述

我们的实践案例

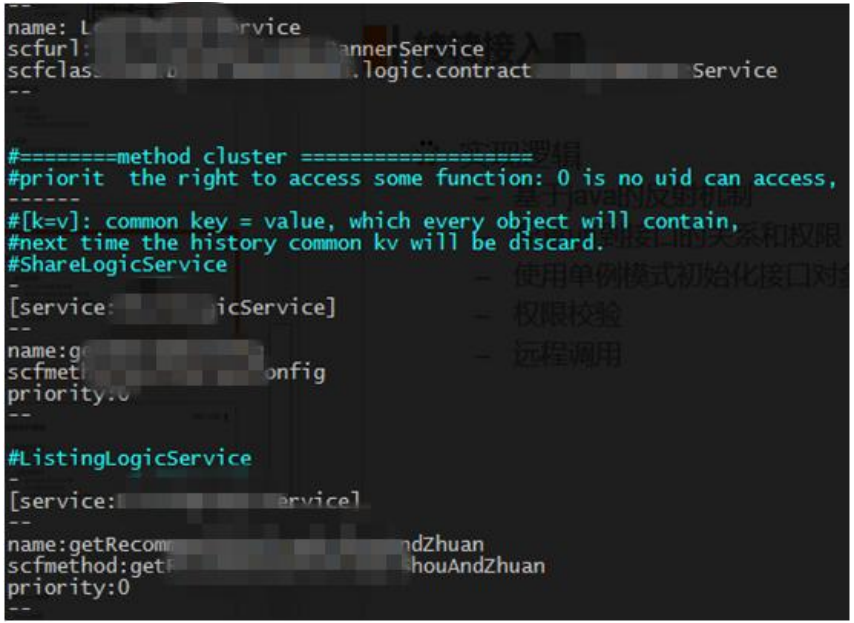
转转接入层设计

- 设计目标
  - 高可用
  - 灵活扩展接口而不修改代码
  - 安全性高
    - 用户、订单、支付
- 实现
  - 负责海量APP端的接入
  - 负责接入请求的合法性校验和安全校验
  - 请求转发Logic层

我们的实践案例

转转接入层设计

- 基于java的反射机制
- 配置uri到接口的关系和权限
- 使用单例模式初始化接口对象
- 权限校验
- 远程调用



安全性

- 双向HTTPS

我们的实践案例二

项目背景

- 58帮帮
  - 58商户和用户沟通平台
  - 海量长连接管理
  - 整流海量长连接
    - 安全通道建立
    - 传输数据加密
  - Session控制
  - 请求转发
  - 反作弊
    - 连接频率、发包频率、发包速率等
    - 对IP/UID等指标实施封禁

本课总结

- 互联网产品通用技术架构
- 接入层的作用是什么
- 接入层Session如何设计?
  - Session复制、Session绑定、Session记录方式、Session高可用等
- 接入层数据安全如何保证?
  - 对称加密、非对称加密、多种方法使用等
- 接入层数据正确性如何保证?
- 接入层高可用设计方案?
  - 无状态、动态扩展
- 接入层高可用设计最佳实践是什么?
- 我们的实践案例;

THANK YOU