微服务架构设计与实践

网关层篇

孙玄@58集团

关于我

- 58集团技 QCon 全球软件开发大会 主席 DTCC 2016中国数据库技术大会 DATABASE TECHNOLOGY CONFERENCE CHINA 2016 WOT World Of Tech
- 58集团高 SDCC 中国软件开发者大会 Software Developer Conference China 构师 TOP1 技术型企业案例研究智库 Strata+Hadoop WORLD
- 转转架构 PROGRAMMER 程序员 责人 ArchSummit 全球架构师峰会
- 百度高级工程师
- 毕业于浙江大学
- 代表公司多次对外分享
- 企业内训&公开课

关于我

企业内训

- ✓ 华为
- ✓ 中航信
- ✓ 平安
- ✓ 银联
- ✓ 华泰证券
- ✓ 思科
- ✓ 云南电力
- ✓ 深信服
- ✓ 新华社
- ✓ 民生银行
- ✓ 招商银行
- ✓ ……

公开课

- ✓ 北京
- ✓ 上海
- ✓ 深圳
- ✓ 广州
- ✓ 成都
- ✓ ……

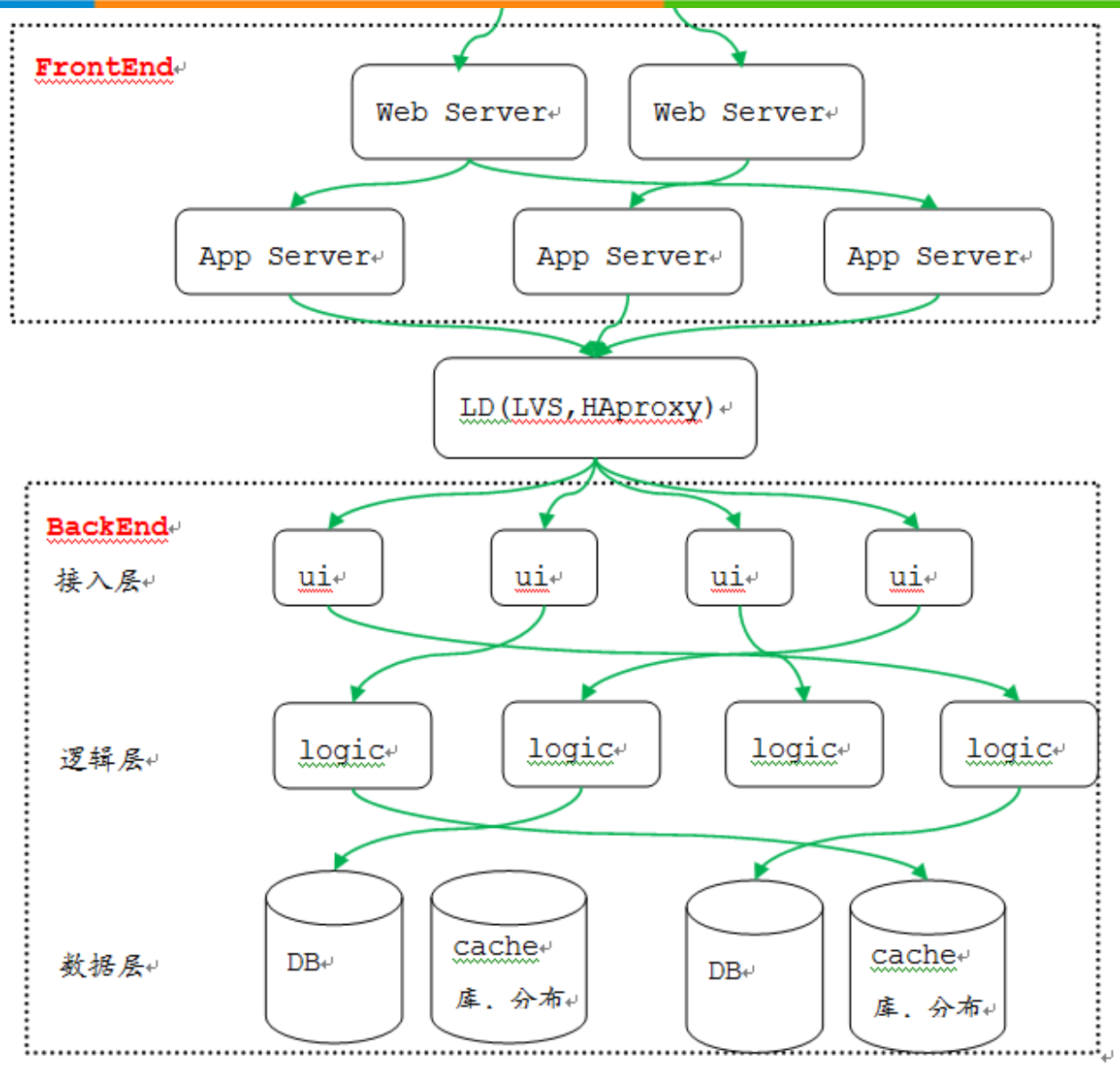
分享要点

微服务架构

1. 互联网产品通用技术架构
2. 网关作用
3. Session设计
4. 安全性设设计
5. 正确性保证
6. 实践案例

互联网产品通用技术架构

- **Data Flow**
  - webServer
  - AppServer
  - LD（LVS,HAProxy...）
  - BackEnd Server
    - 网关层
    - 异步提交层
    - 微服务聚合层
    - 数据层
  - 本文重点关注
    - BackEnd Server

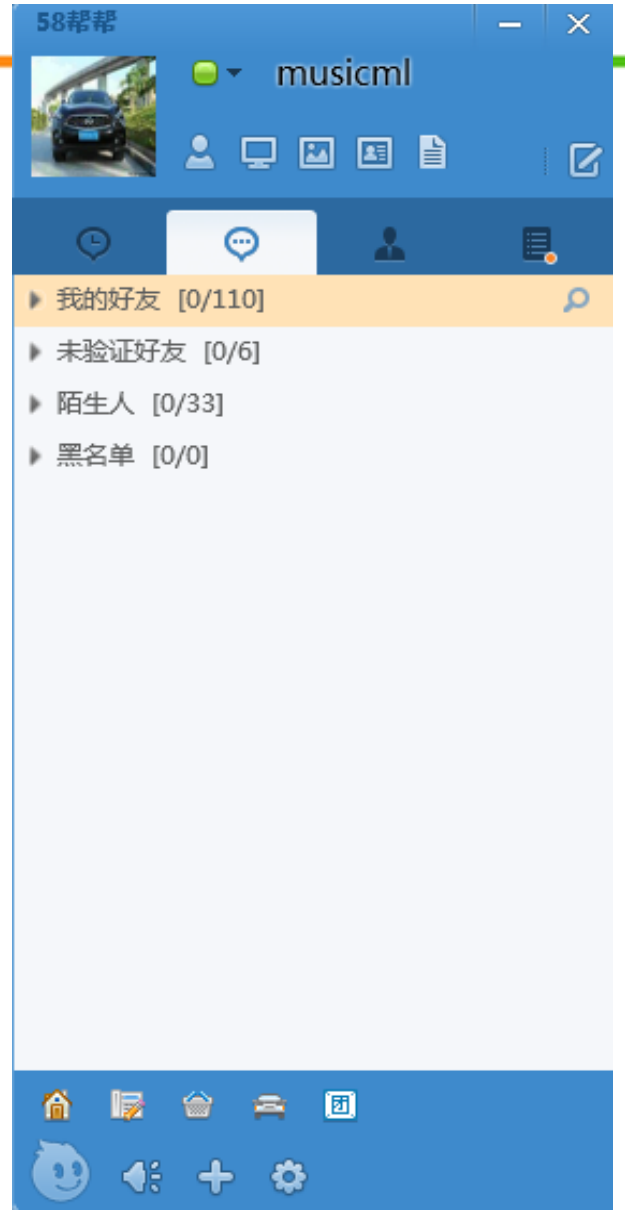


58帮帮技术架构

- **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+

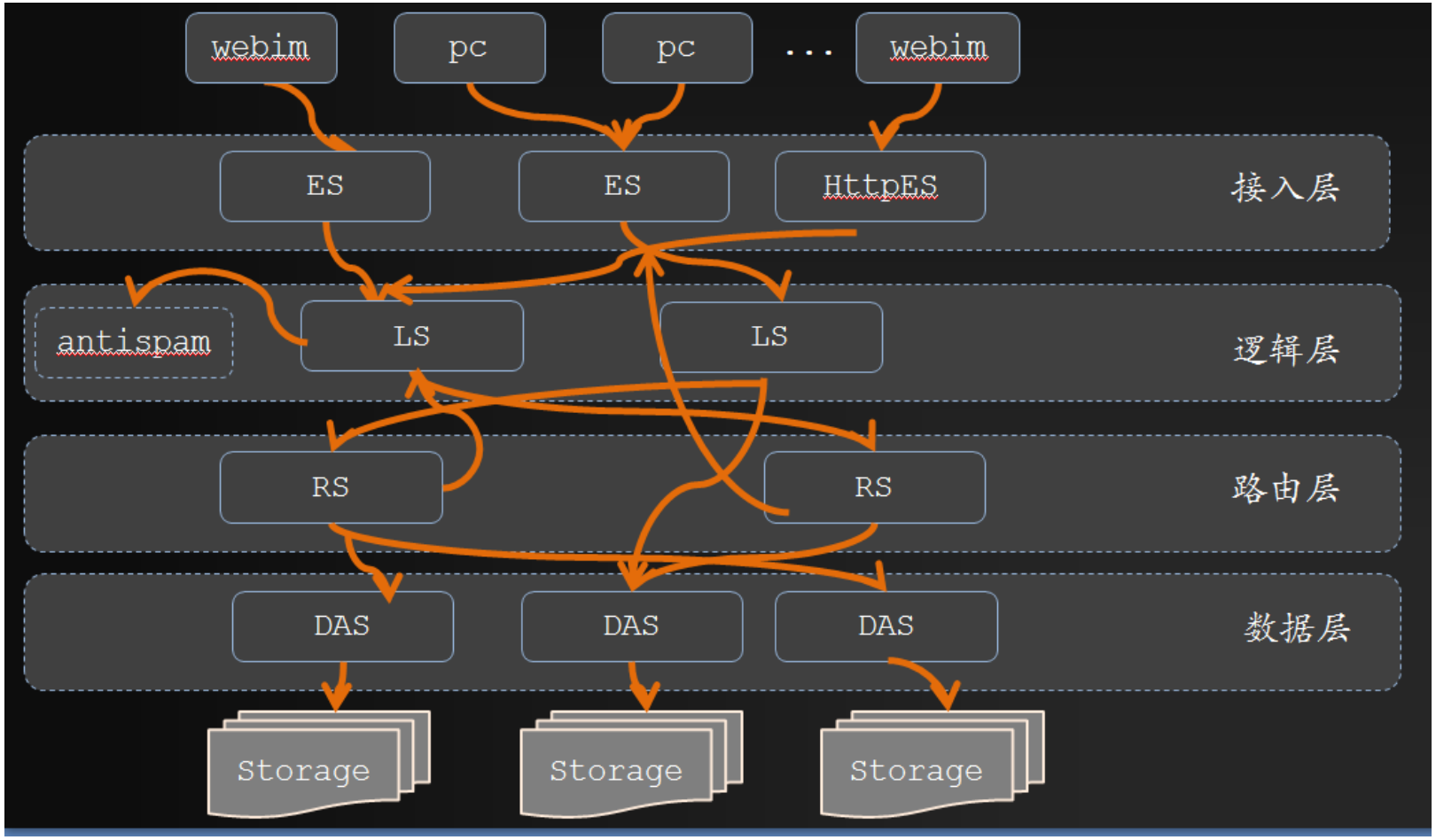
58帮帮技术架构

- **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

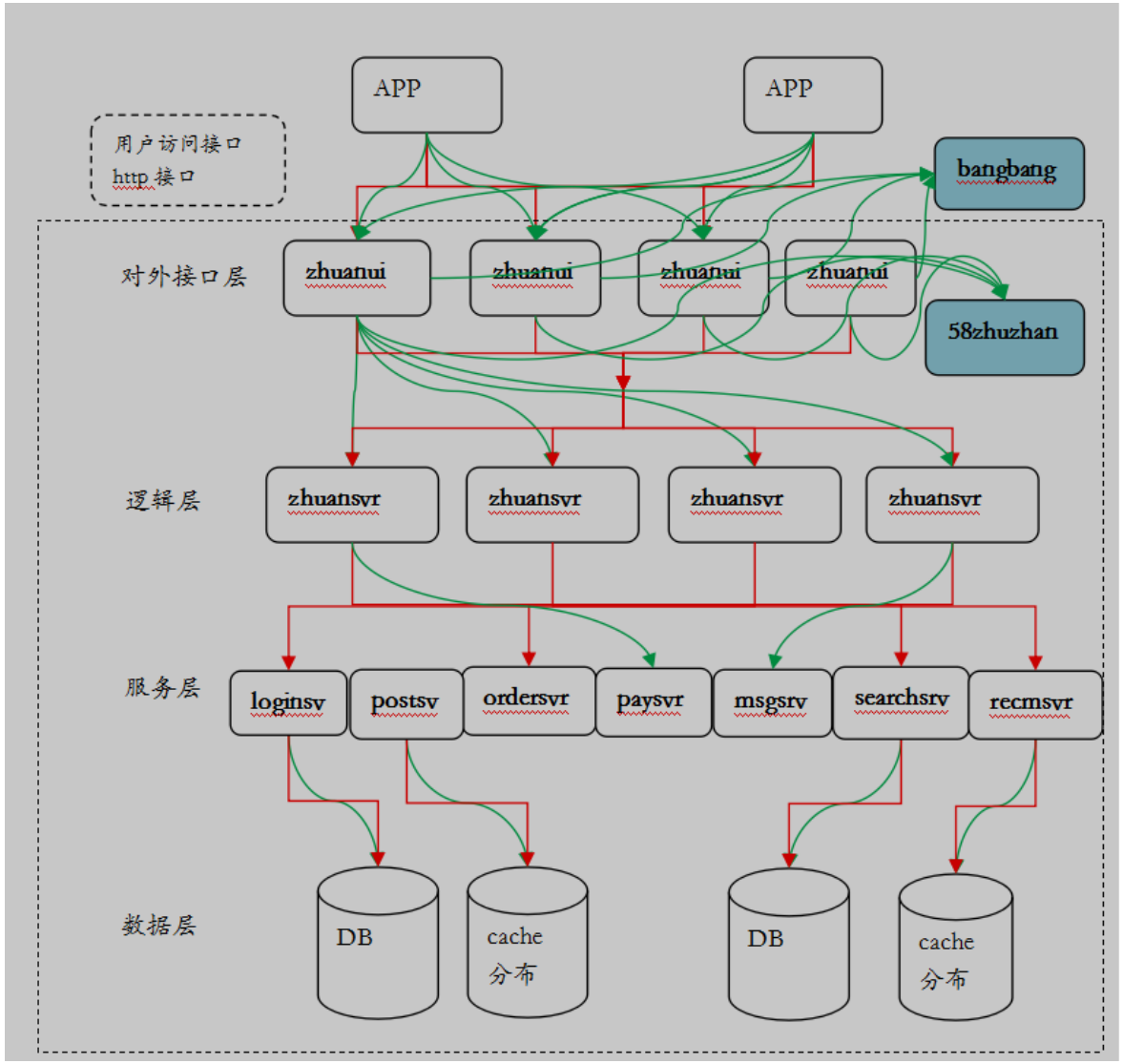
- **传统IM架构如何满足千万同时在线性能？**
  - **网关层、聚合层、路由层、数据层**
  - 无状态设计
  - 每层模块动态高扩展
  - 模块冗余，高可用性保证
  - 动态负载均衡，动态切换可用服务节点
  - 优化效果
    - 单机线上支持50W+同时在线
    - 单机线上3w+qps

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
    - **网关层**
    - **聚合层**
    - **原子微服务层**
    - **数据层**
  - ……



百度空间feed系统架构

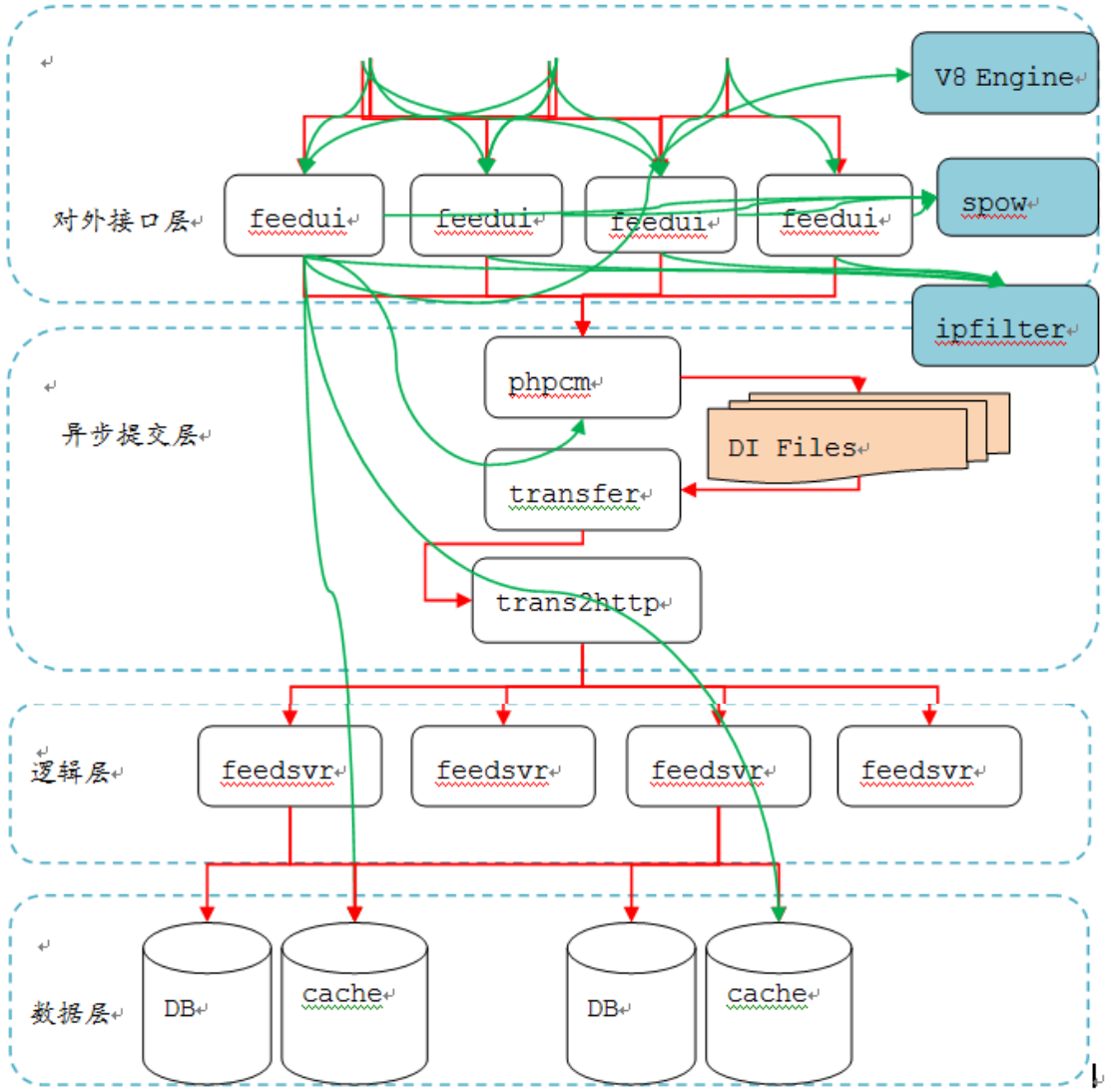
- **Feed系统**

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**
  - 网关层
  - 异步提交层
  - 业务聚合层
  - 数据层



网关作用

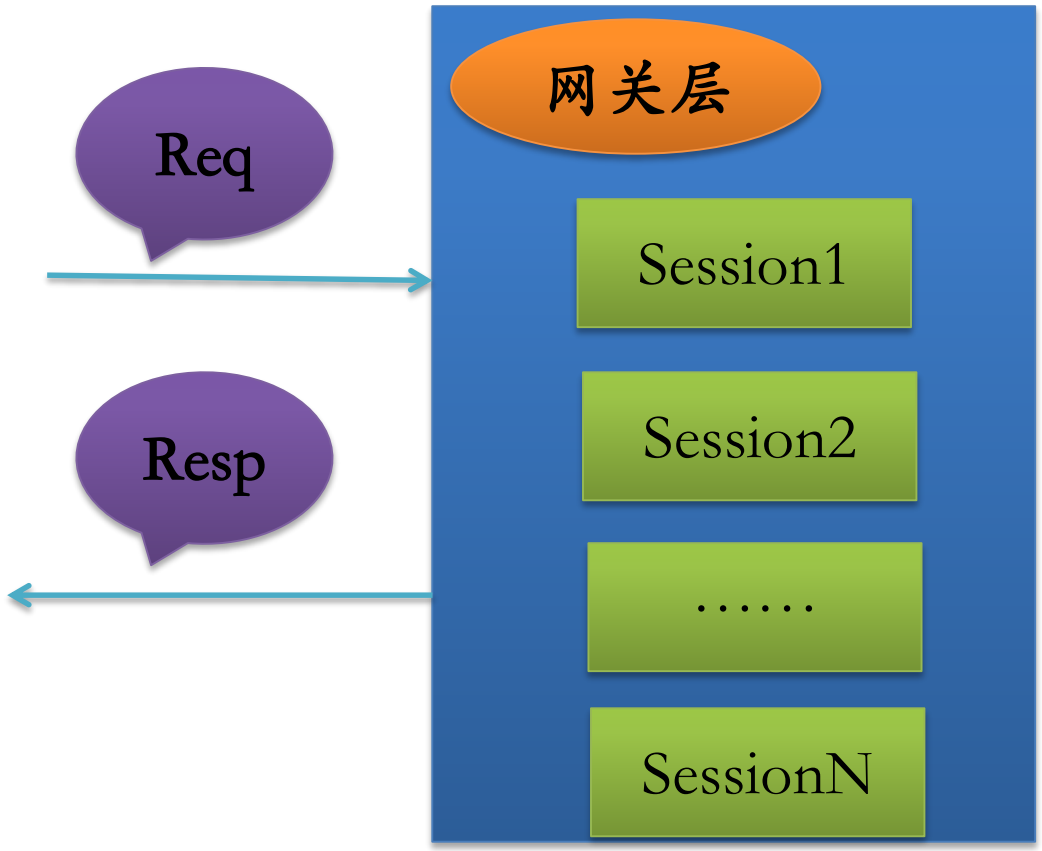
- **网关层作用**
  - 客户端海量长/短连接管理
    - TCP/HTTP[S]
  - 建立与客户端通信的加密通道
  - 数据合法性、正确性校验
  - 整合成内部少量的长连接
  - Session的管理
  - 实施初步的攻防
  - 请求转发到逻辑层

网关层Session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

网关层Session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？

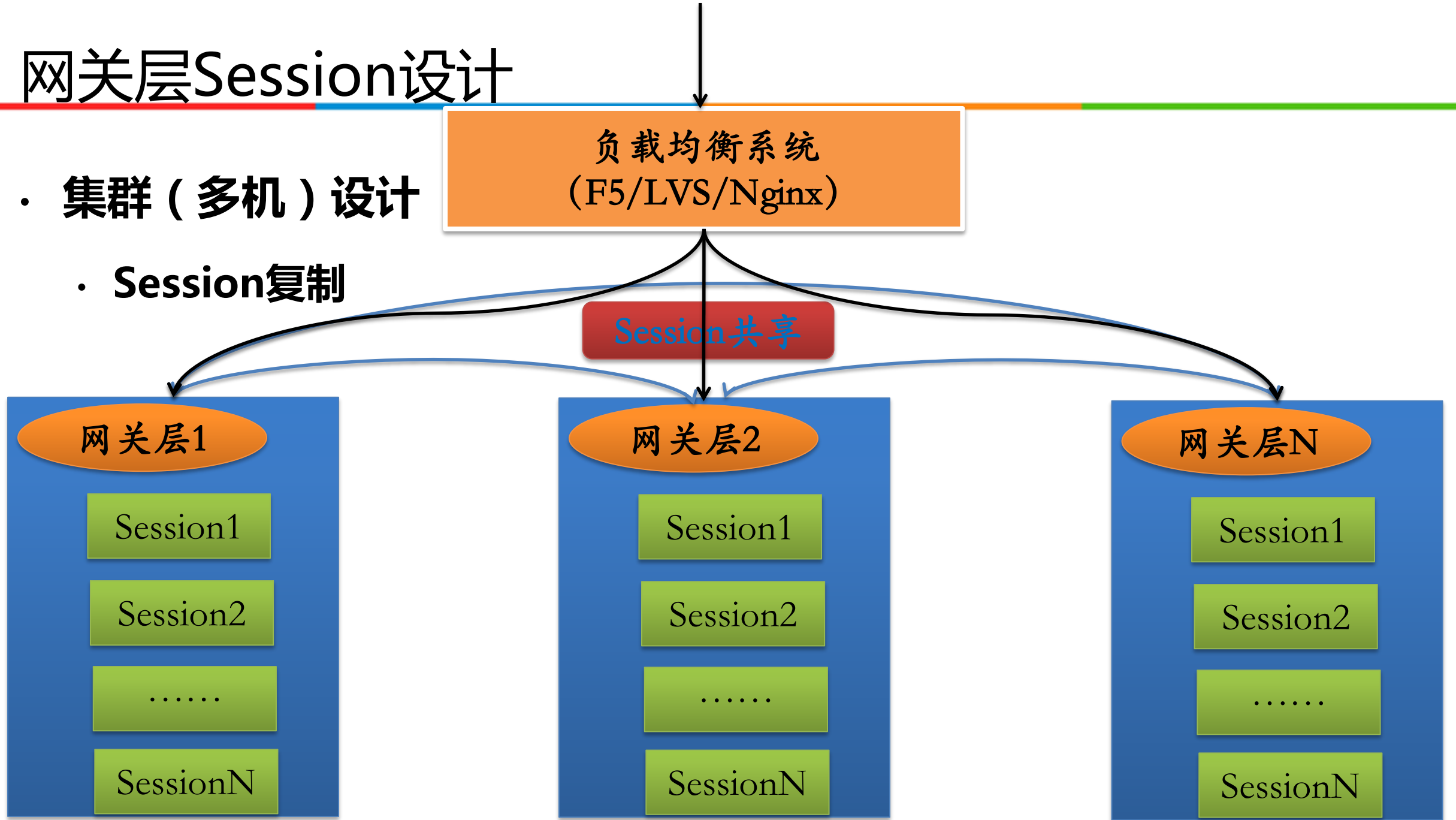


网关层Session设计

- **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

网关层Session设计

- 集群（多机）设计
  - Session复制



网关层Session设计

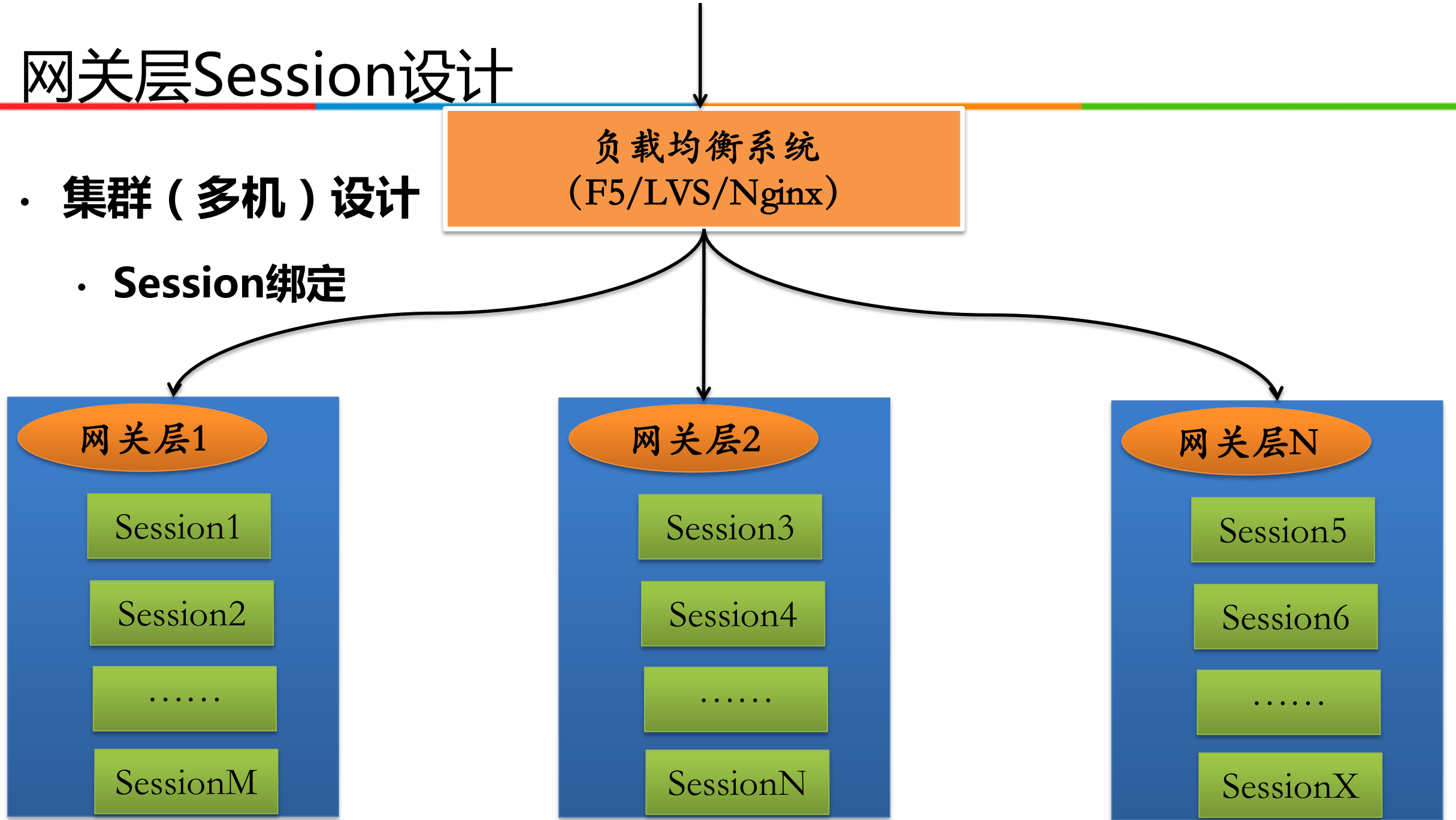
- **集群（多机）设计**
  - **Session复制**
    - 存在问题
      - 适用于网关层集群较少
      - 网关层集群量大
        - 大量的Session复制通信，占用服务器和网络资源
        - 每台机器存储全量用户Session，内存占用量大，甚至Out Of Memory
        - 大型网站接入层数千台，同时在线用户达到千万（IM)，不适合

网关层Session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

网关层Session设计

- 集群（多机）设计
  - Session绑定

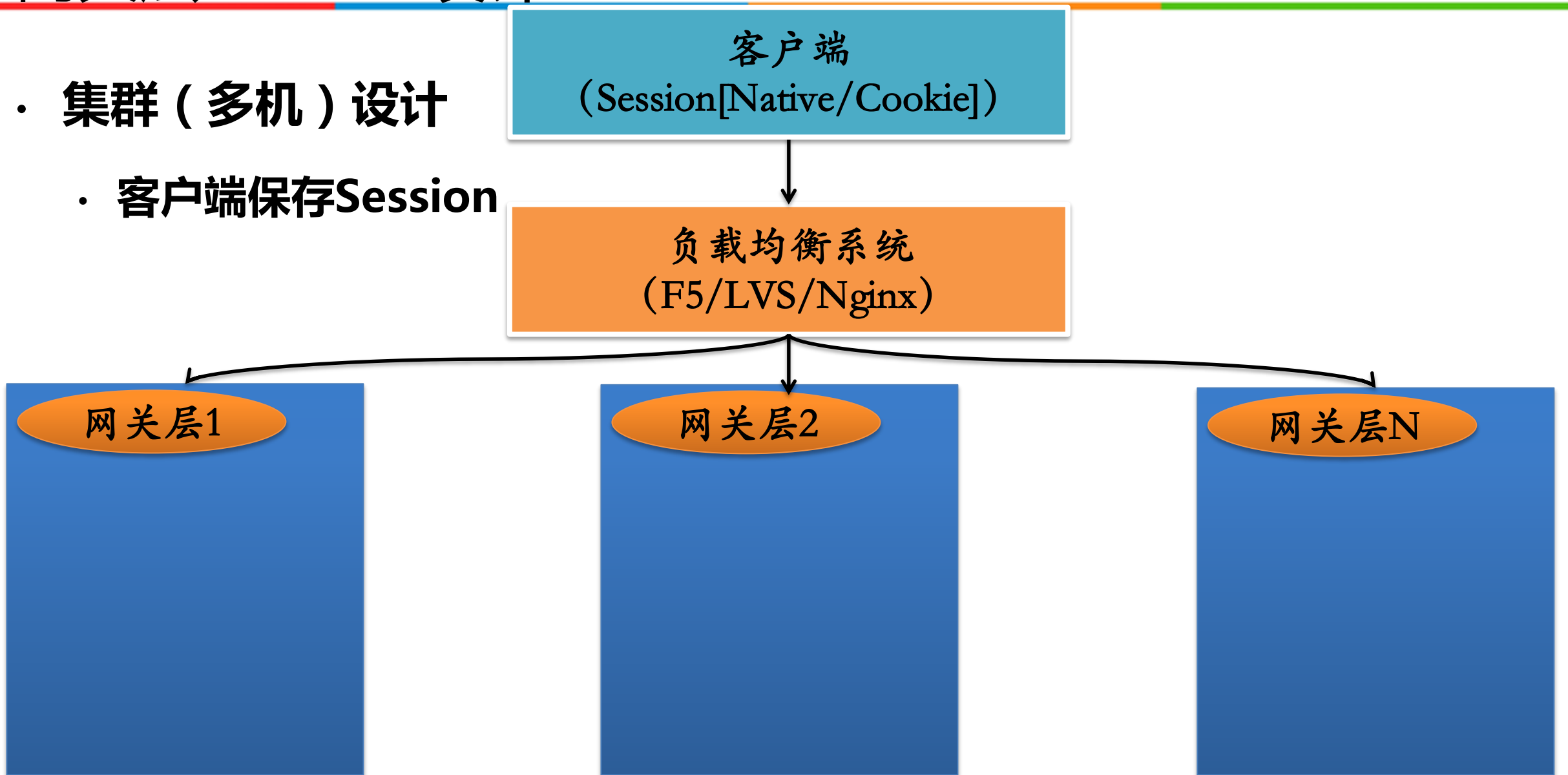


网关层Session设计

- **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

网关层Session设计

- 集群（多机）设计
  - 客户端保存Session



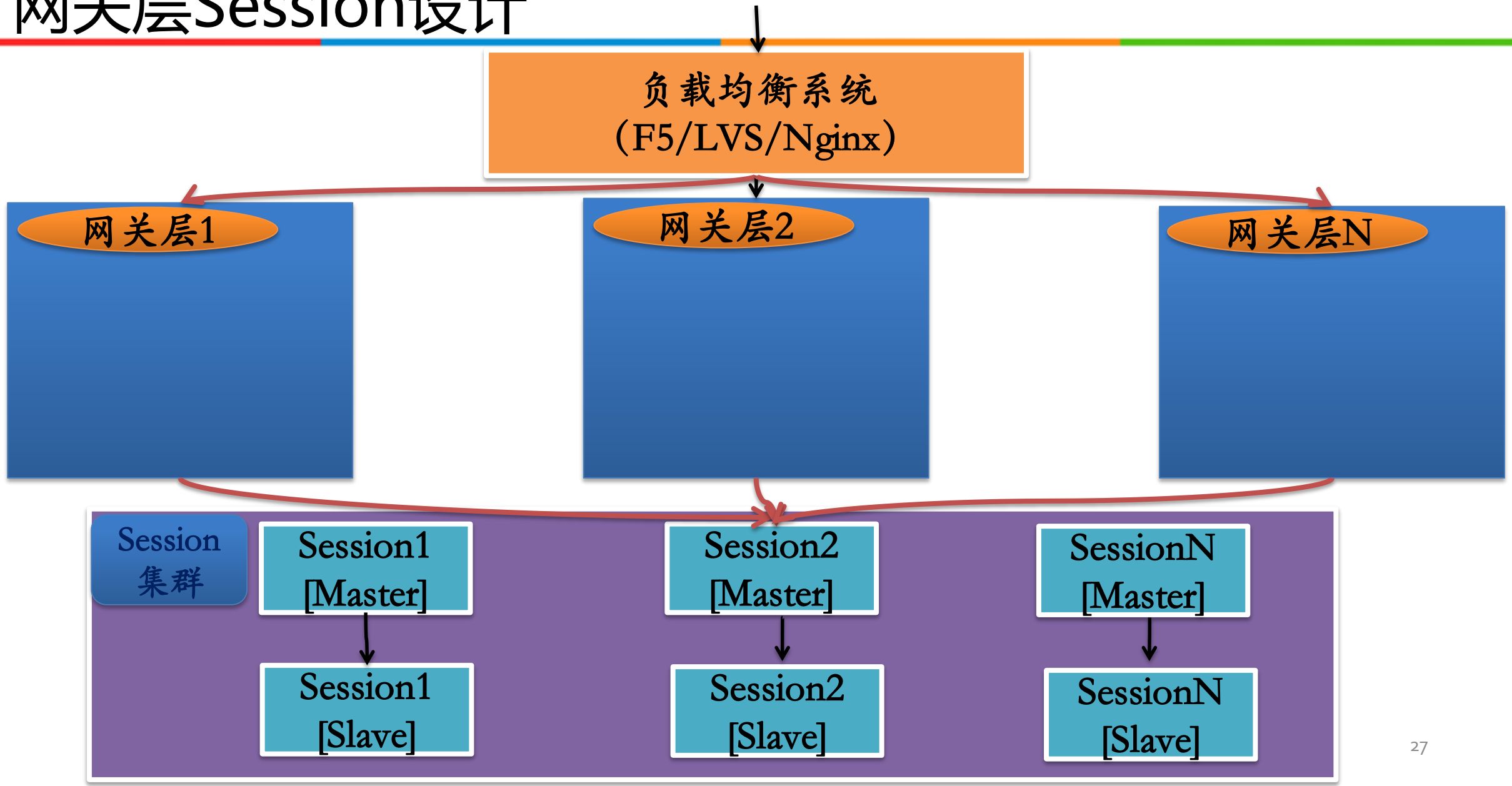
网关层Session设计

- **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

网关层Session设计

- **高可用Sesson设计**
  - **Session高可用集群**
    - 网关层无状态化
    - 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
      - 网关层本身无状态
      - Session集群有状态
        - » 分布式缓存
          - √ NoSQL（Memcached/Redis）
          - √ RDBMS（MySQL/MongoDB）

网关层Session设计



网关层数据安全保证

- **网关层安全性**
  - 网关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

网关层数据安全保证

- **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
    - ……

网关层数据安全保证

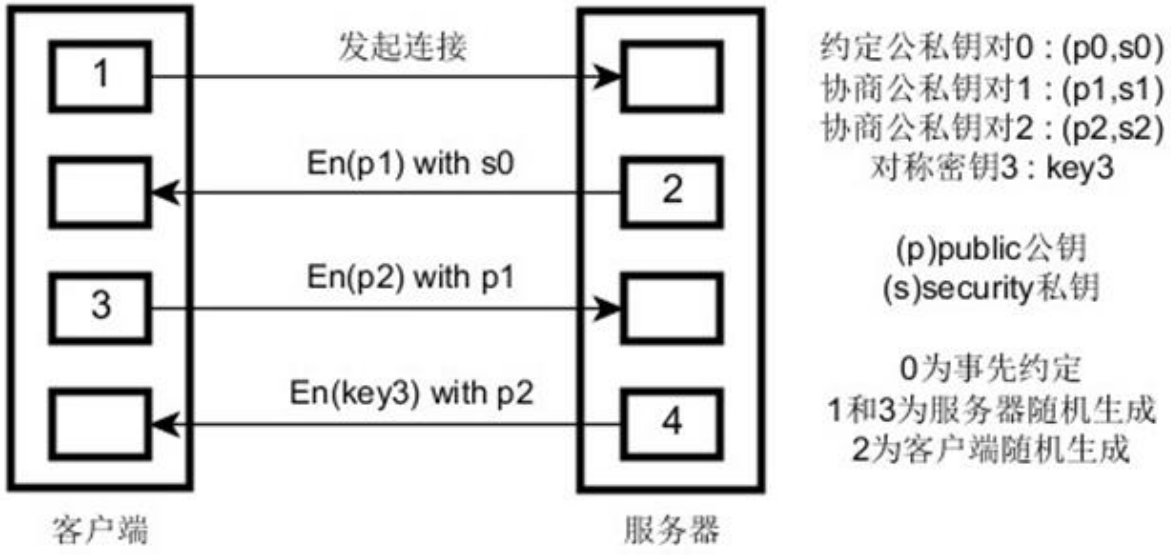
- **名称解释**
  - 对称加密算法：
    - 加密和解码使用同一密钥的加密方案（AES）
  - 非对称加密算法
    - 加密和解密使用一对密钥（由两个满足一定关系的密钥组成的密钥对）中不同密钥的加密方法（RSA）
  - 公钥
    - 非对称加密算法中公开给大众保密的密钥
  - 私钥
    - 非对称加密算法中留给个人保密的密钥
  - 会话状态
    - 描述客户端与服务器的一次连接的所有信息集合

网关层数据安全保证

- **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

网关层数据安全保证

- **技术实现方案**
  - 为满足以上两个条件，客户端和服务器都必须要一个随机生成密钥过程
  - 具体的四步握手（Client->C　Server->S)



网关层数据安全保证

- **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

网关层数据安全保证

- **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
    - ……

网关层数据安全保证

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

网关层数据安全保证

- **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

网关层数据正确性保证

- **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？

网关层数据正确性保证

- **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

网关层数据正确性保证

- **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。

网关层数据正确性保证

- **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。

网关层数据正确性保证

- **数字签名实例**
  - 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>
```

网关层数据正确性保证

- **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

高可用网关层如何设计？

- **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

高可用网关层最佳实践

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

实践案例一

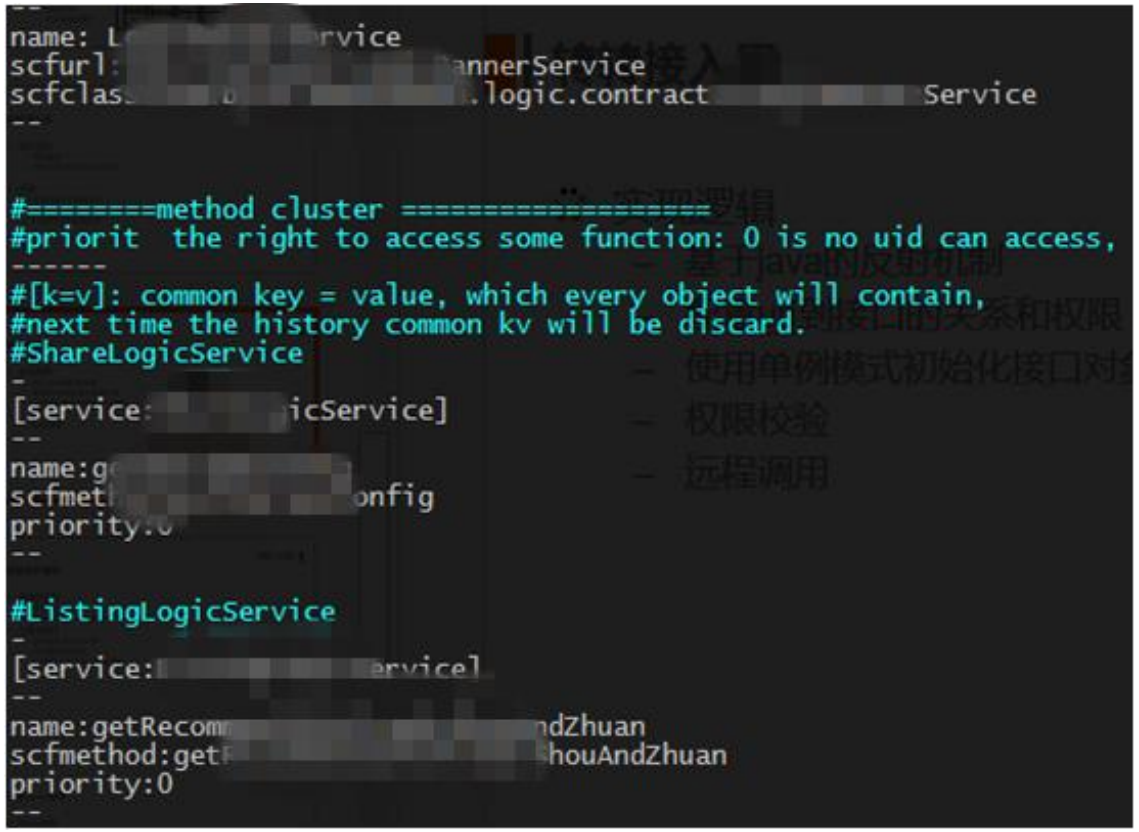
- **项目背景**
  - 全国最大的真实C2C交易平台
    - 同之前叙述

实践案例一

- **转网关层设计**
  - 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
    - 请求转发微服务聚合层

实践案例一

- **转转网关层设计**
  - 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



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

要点回顾

微服务架构

1. 互联网产品通用技术架构
2. 网关作用
3. Session设计
4. 安全性设设计
5. 正确性保证
6. 实践案例

欢迎关注本人公众号 “架构之美”

Thanks!

让 生 活 更 简 单