20：05开始上课

分布式思维概述

T A H N K Y O U F O R W A T C H I N G

分享人：Peter

目 录

CONTENTS

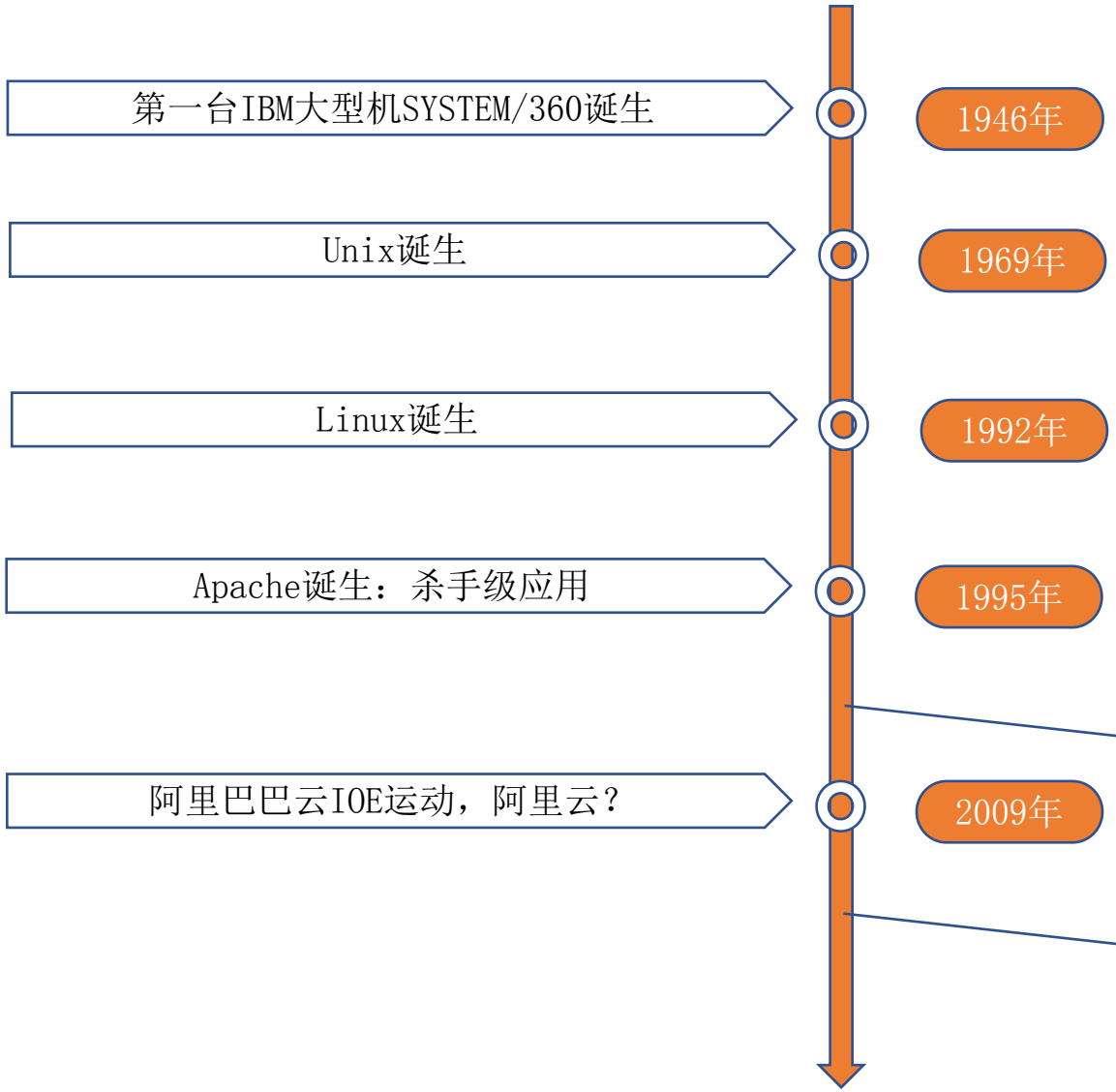
计算机与软件的发展历史

什么是分布式与集群的关系

大型架构演进过程分析

架构演进中的问题挖掘

软件-互联网的发展史



大型主机优点：
集中式的计算机系统，高的稳定性和安全性。

大型主机缺点：

1. 非常贵,一般的小企业用不起。
2. 比较复杂，培养人才的成本比较高。
3. 单点一旦故障,整个系统停转,损失非常大。
4. 个人PC电脑的性能越来越高,成本也越来越低。

Web时代

c/s时代：富客户端方案。卖软件可赚钱。
qq、影音、游戏。

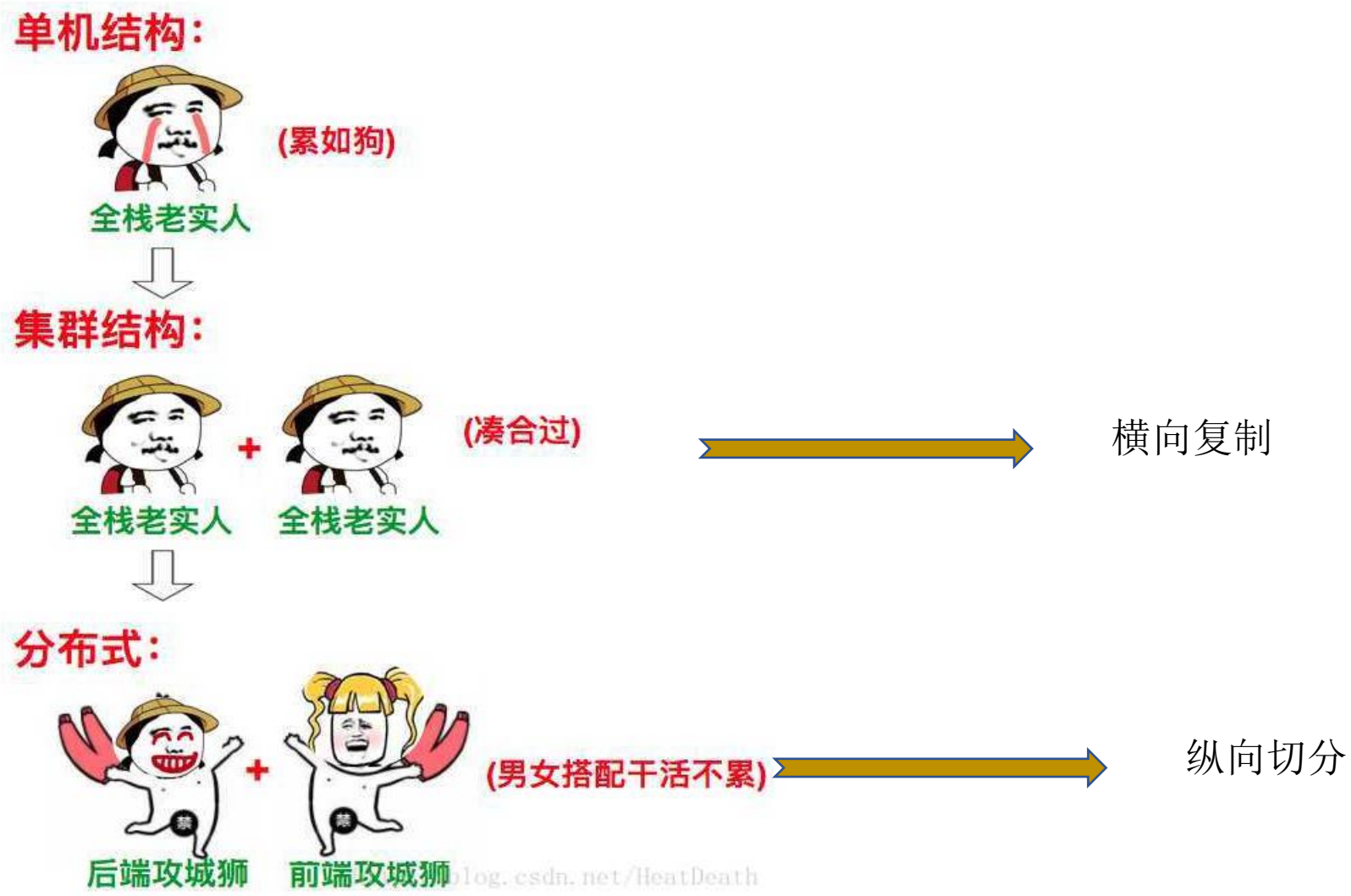
1.0时代：主要是单向信息的发布，即信息门户---→广大浏览器客户端
互联网内容是由少数编辑人员（或站长）定制的。

代表是三大门户，新浪/网易/搜狐。新浪以新闻+广告为主，网易拓展游戏为主，搜狐延伸门户矩阵

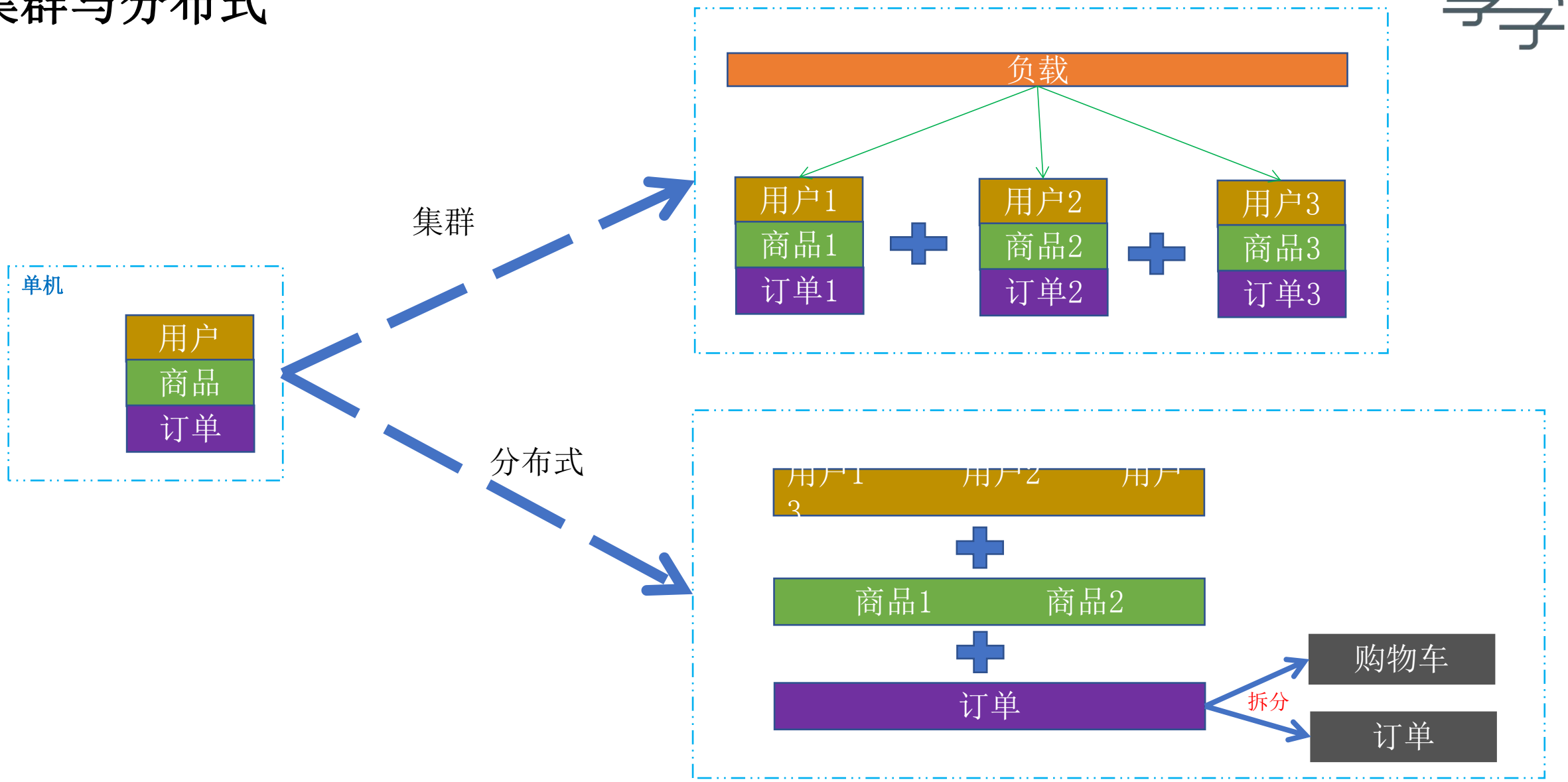
2.0时代：注重用户的交互。每个人都是内容的供稿者。 RSS订阅扮演一个很重要的作用。

代表：
博客、播客、维基、P2P下载、社区、分享服务

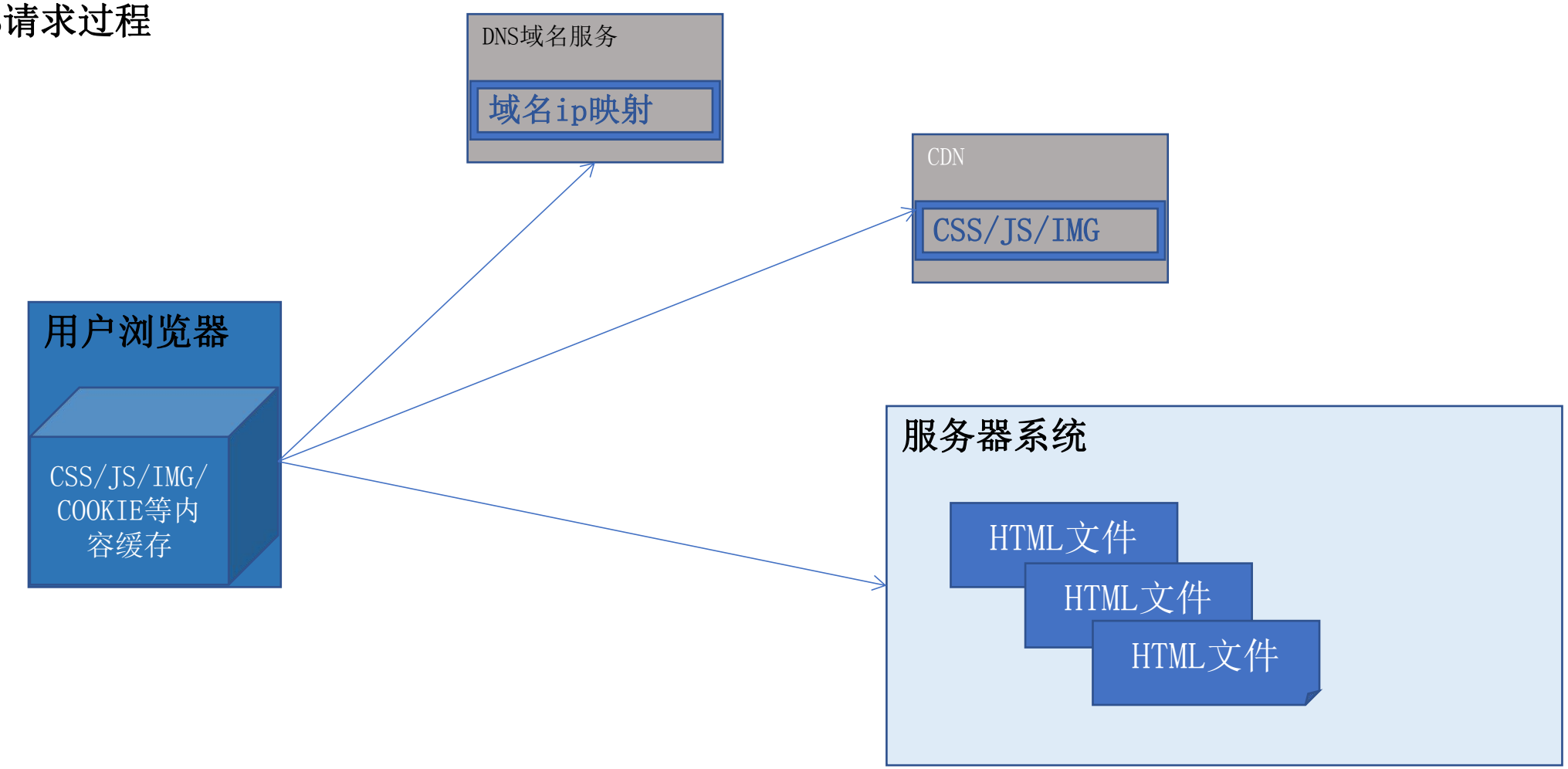
集群与分布式


单机结构：
(累如狗)
全栈老实人
集群结构：
(凑合过)
全栈老实人
全栈老实人
横向复制
分布式：
(男女搭配干活不累)
后端攻城狮
前端攻城狮
纵向切分

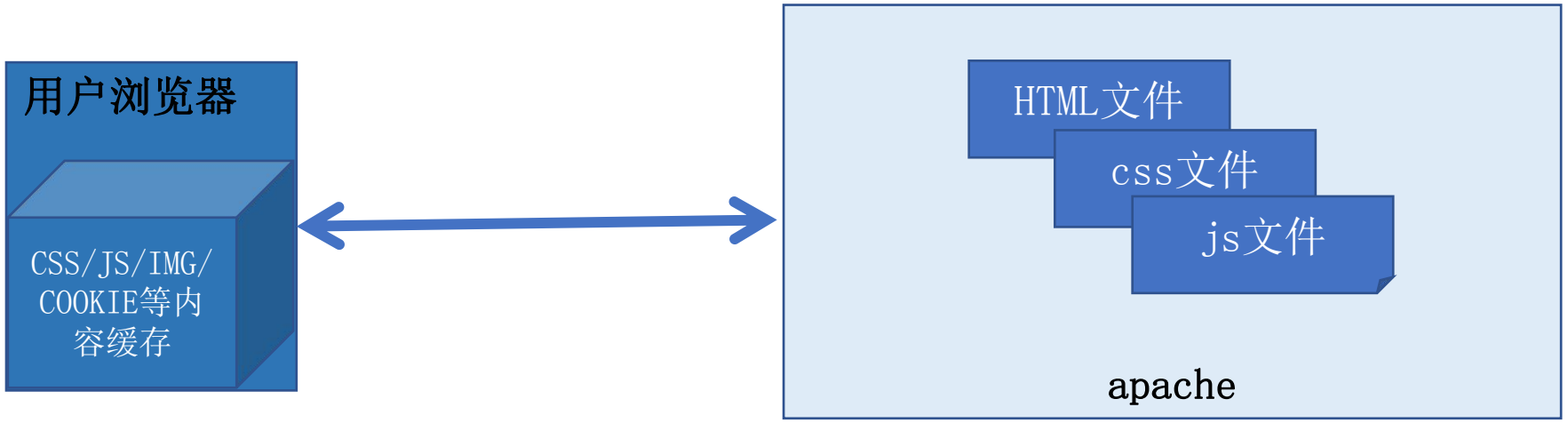
集群与分布式


单机
用户
商品
订单
集群
分布式
负载
用户1
商品1
订单1
用户2
商品2
订单2
用户3
商品3
订单3
商品1
商品2
订单
拆分
购物车
订单

B/S请求过程

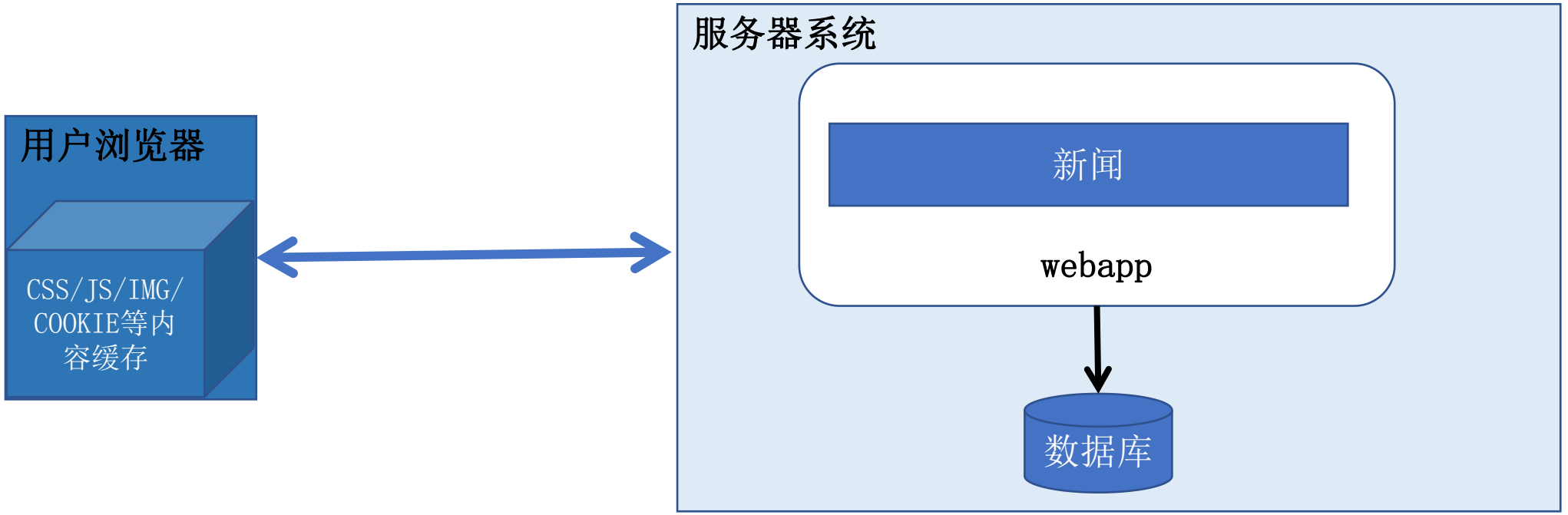

DNS域名服务
域名ip映射
CDN
CSS/JS/IMG
用户浏览器
CSS/JS/IMG/
COOKIE等内
容缓存
服务器系统
HTML文件
HTML文件
HTML文件

架构演进一：早期雏形



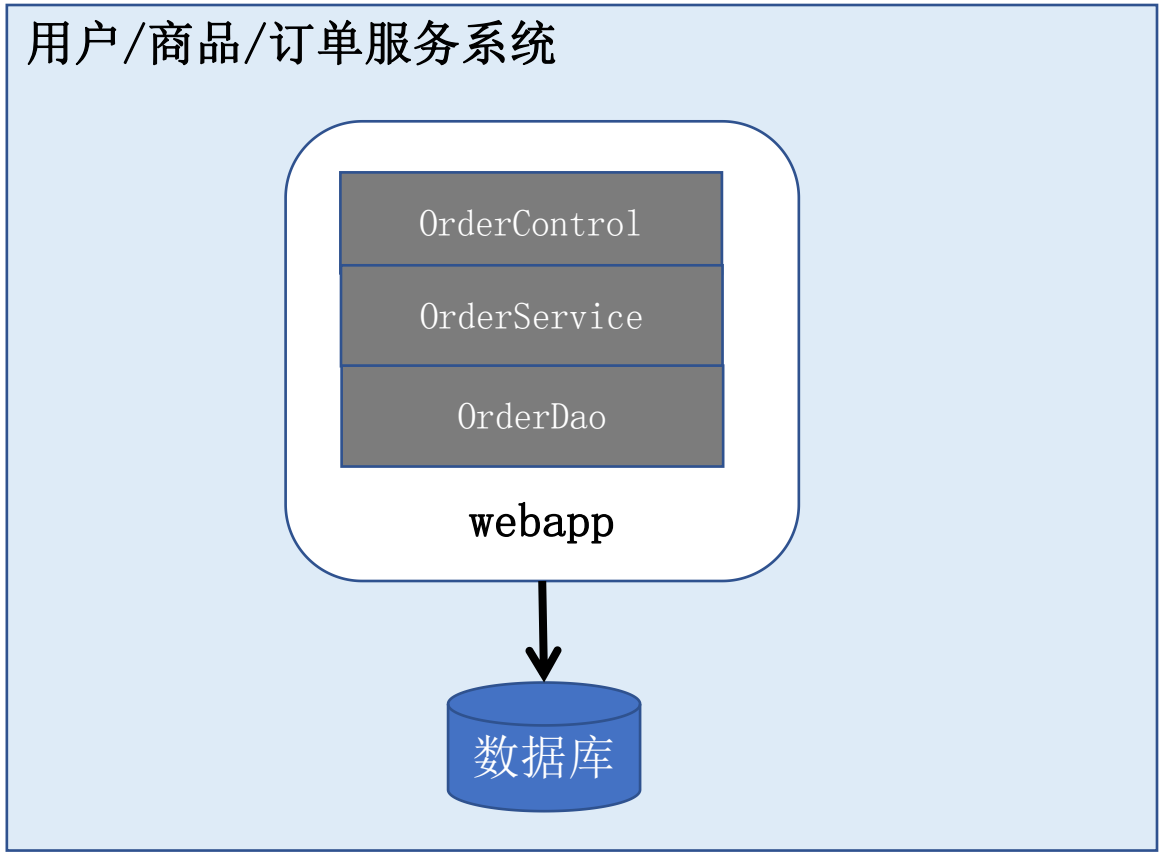
特征：应用程序主要做静态文件读取，返回内容给浏览器。

架构演进二：数据库开发（LAMP特长）



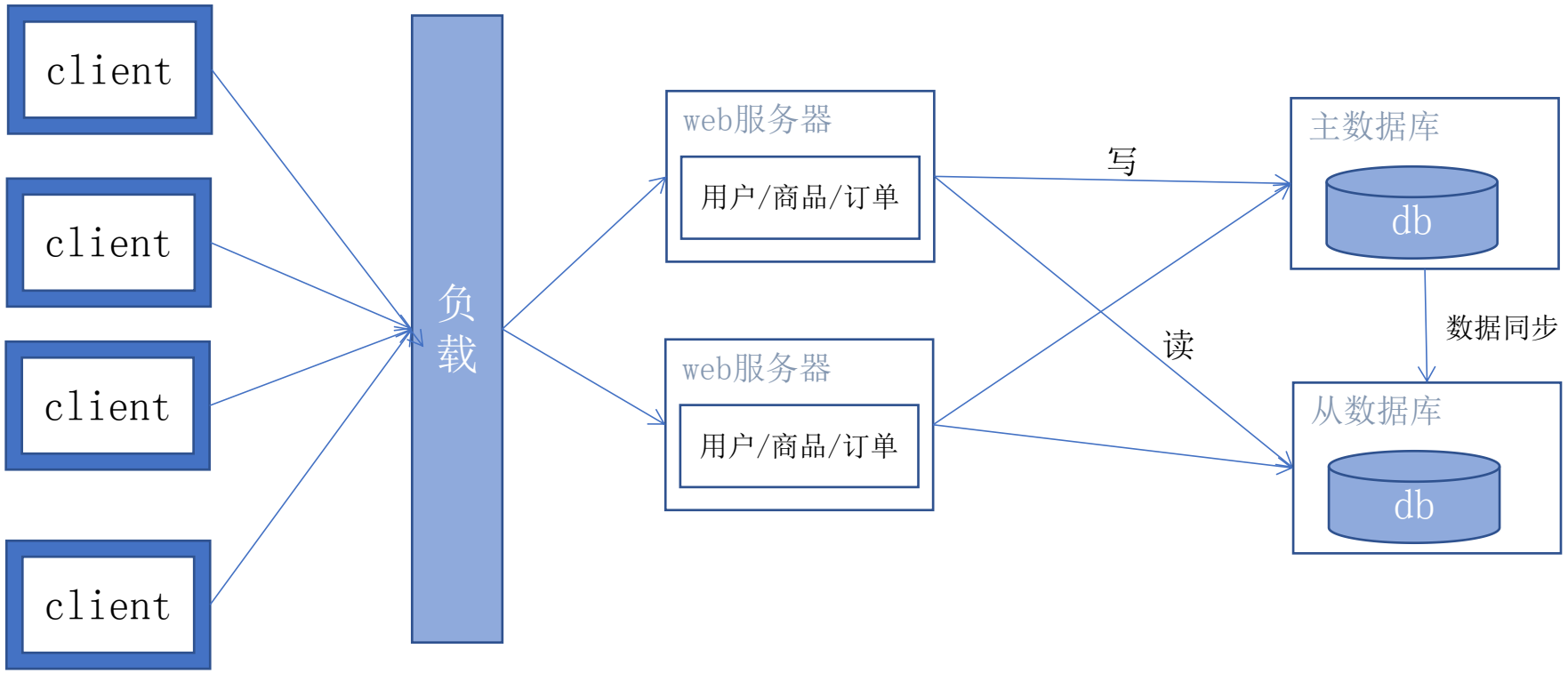
特征：应用程序主要主要读取数据表值，填充html模块。业务逻辑简单，写sql处理。

架构演进三： javaweb的雏形



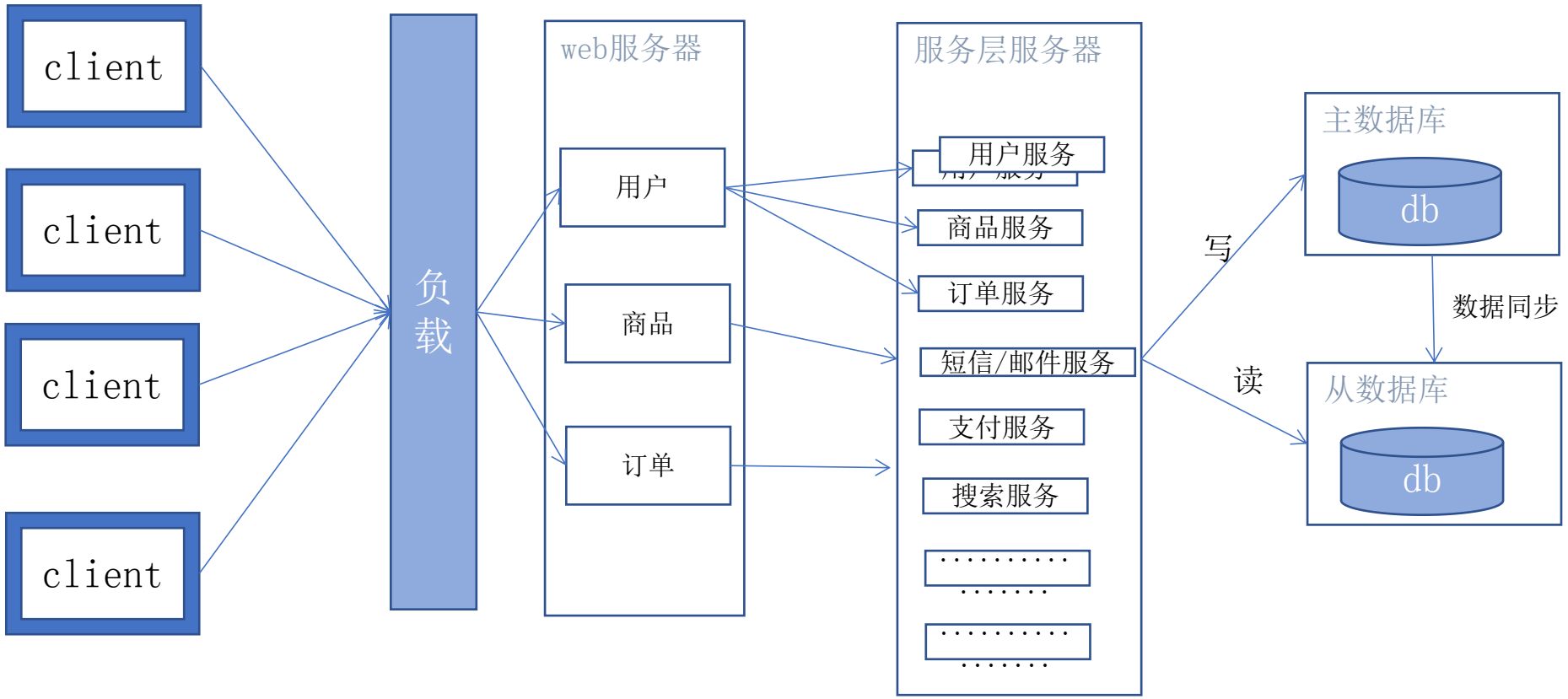
特征： tomcat + servlet + jsp + mysql。一个war包打天下
项目结构：ssh/ssm三层结构。

架构演进四： javaweb的集群发展



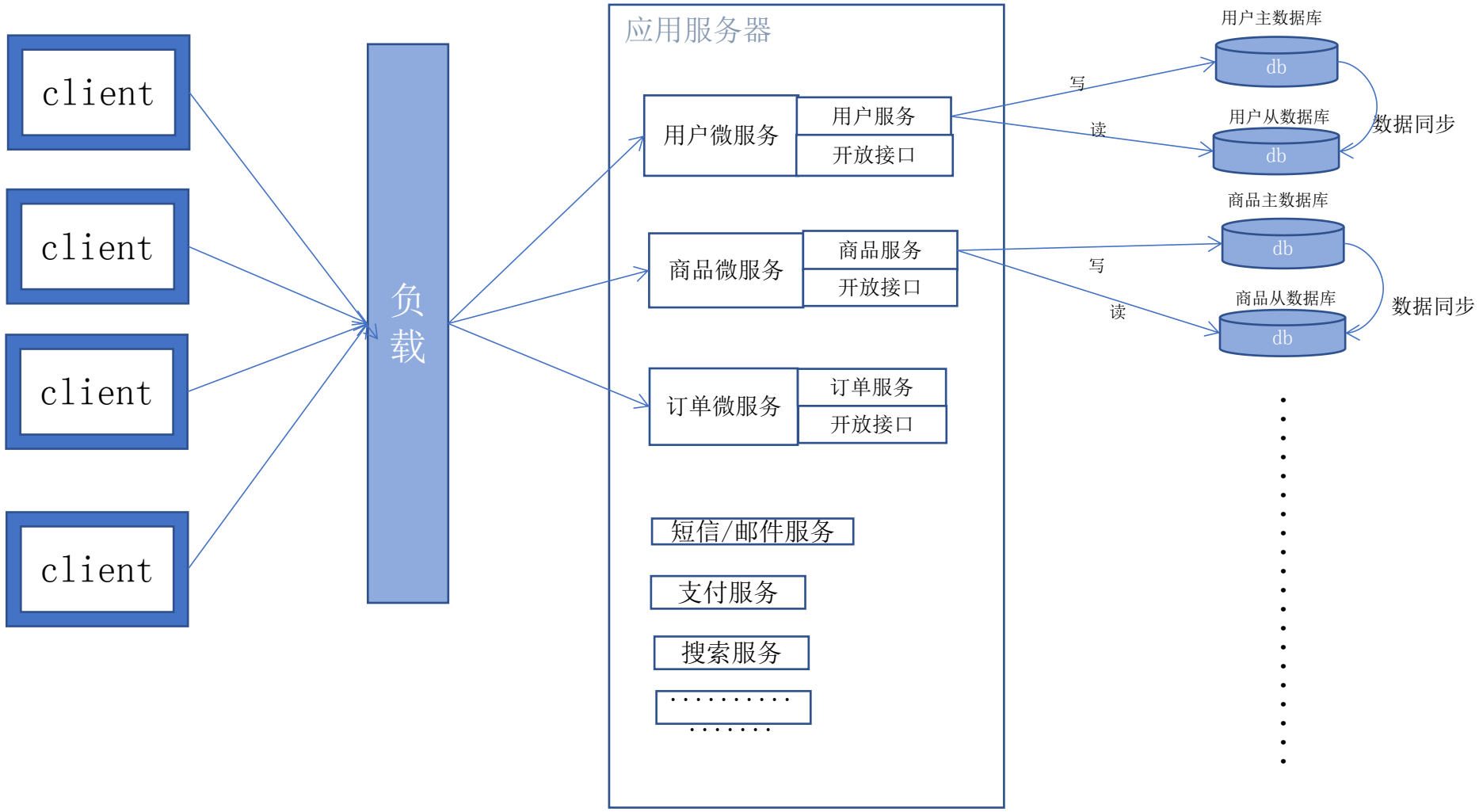
特征：硬件机器的横向复制，对整个项目结构无影响。

架构演进五： javaweb的分布式发展



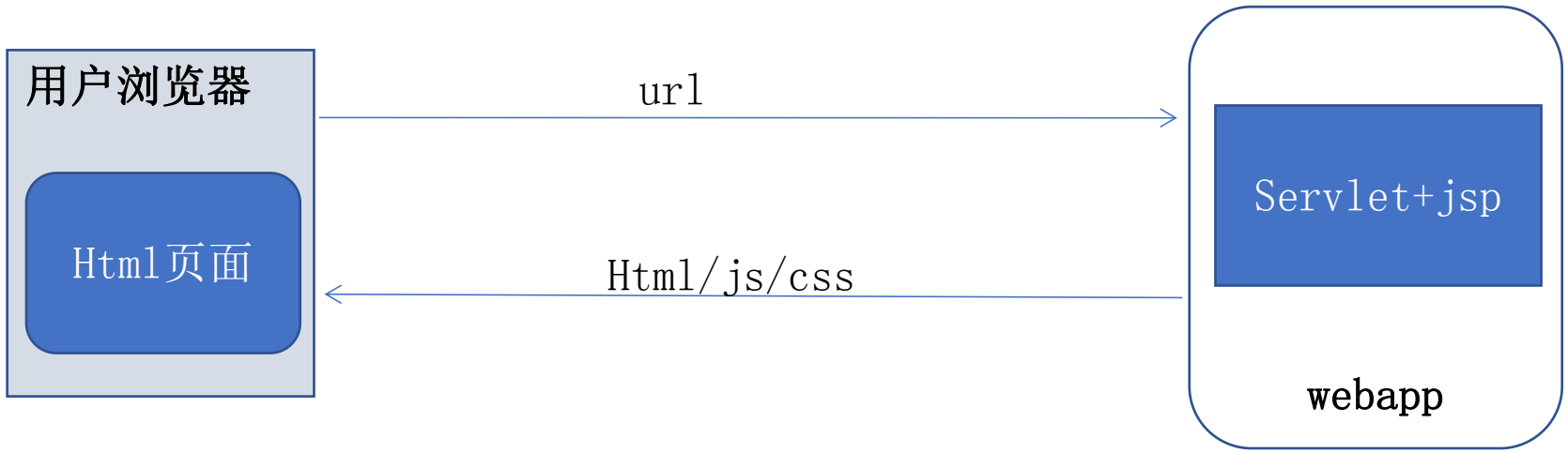
特征：将Service层单独分离出去，成为一个单独的项目jar。单独运行。
Web服务器通过rpc框架，对分离出去的service进行调用。

架构演进六： javaweb的微服务发展



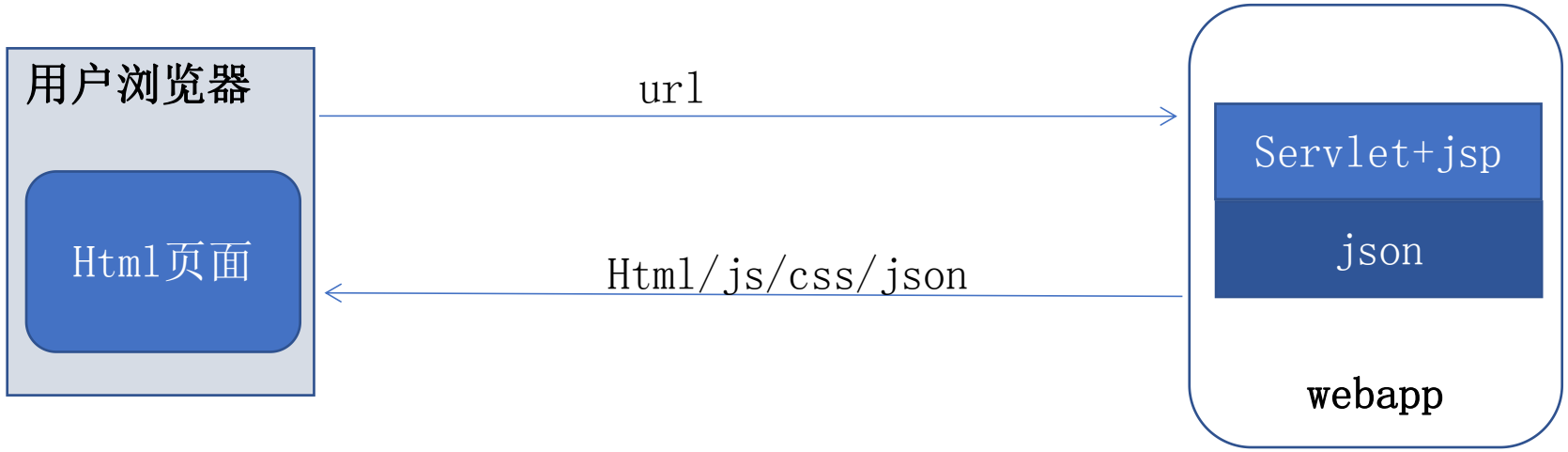
特征：从业务角度，细分业务为微服务，每一个微服务是一个完整的服务（从http请求到返回）。
在微服务内部，将需要对外提供的接口，包装成rpc接口，对外部开放。

前后端交互模式：整页提交



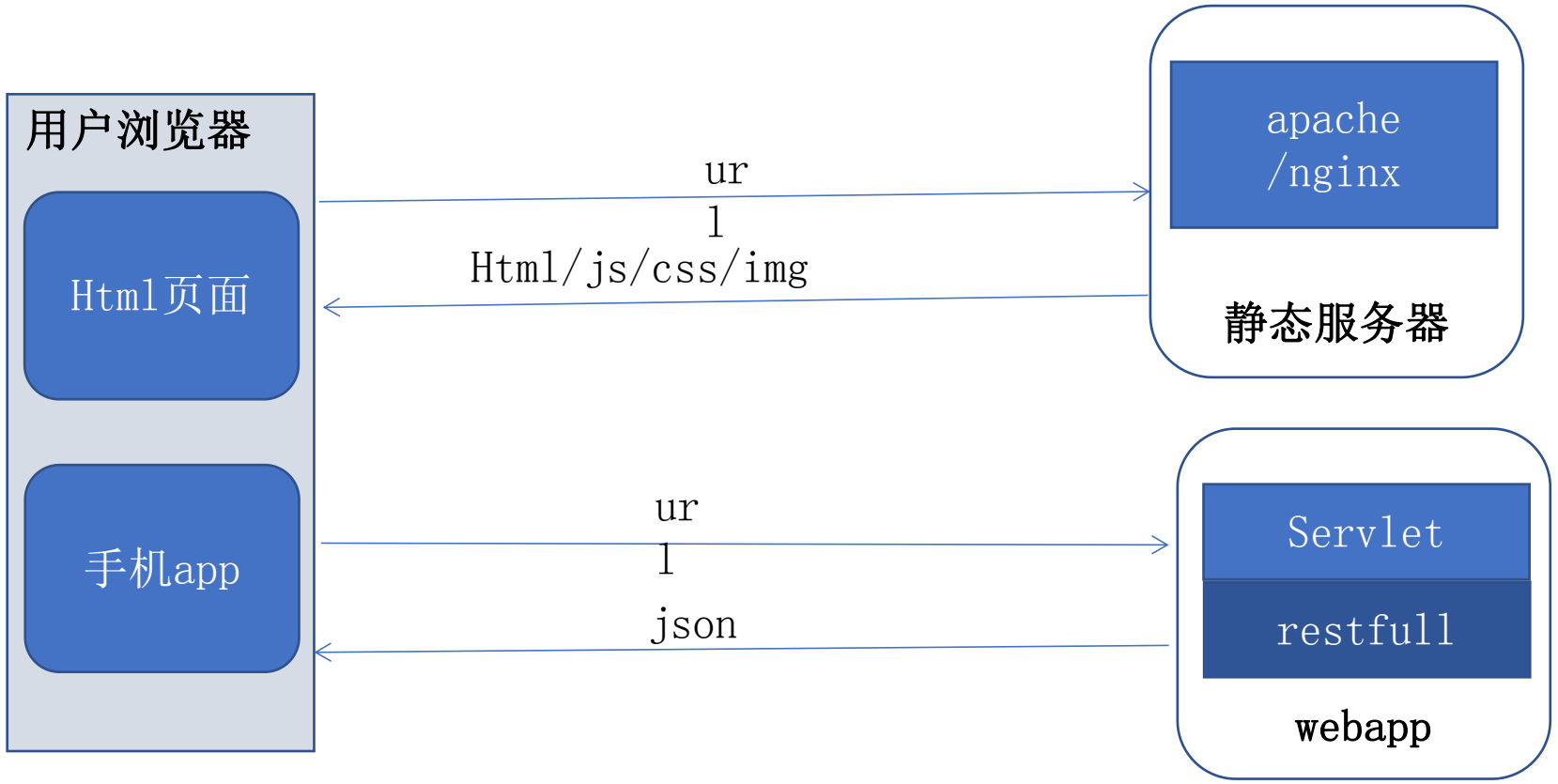
特征：浏览器请求皆为页面级请求，每次请求都是一次页面跳转/刷新。

前后端交互模式：页面+ajax混和



特征：浏览器请求主要为页面级请求，有局部刷新使用ajax刷新，页面体验更好。

前后端交互模式：单页应用mvvm模式

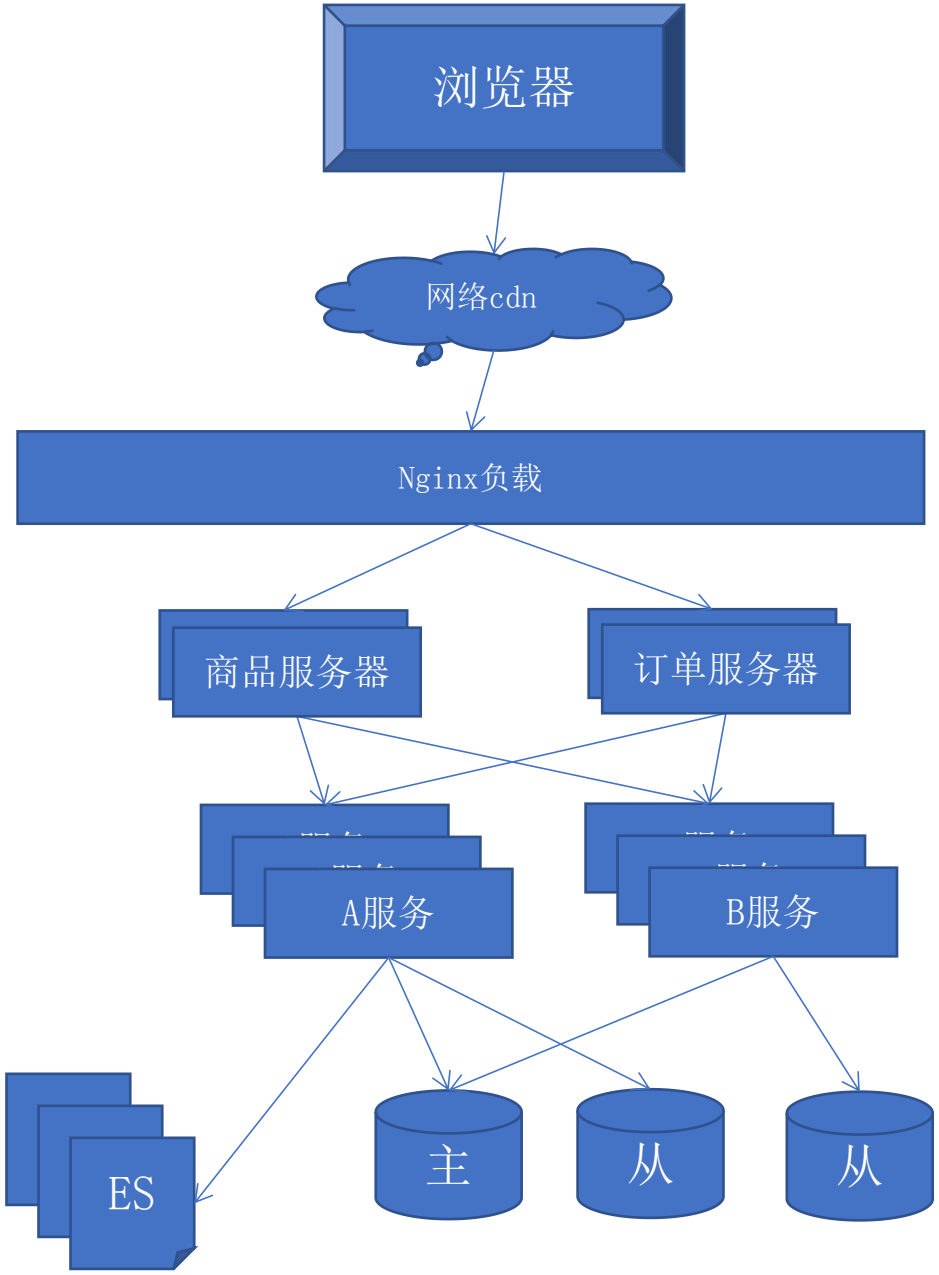


特征：首次请求返回页面html，后续请求皆为restful返回json

架构思路：架构改进中常见解决方案

--------架构的依据，并发数（tps）和数据量级

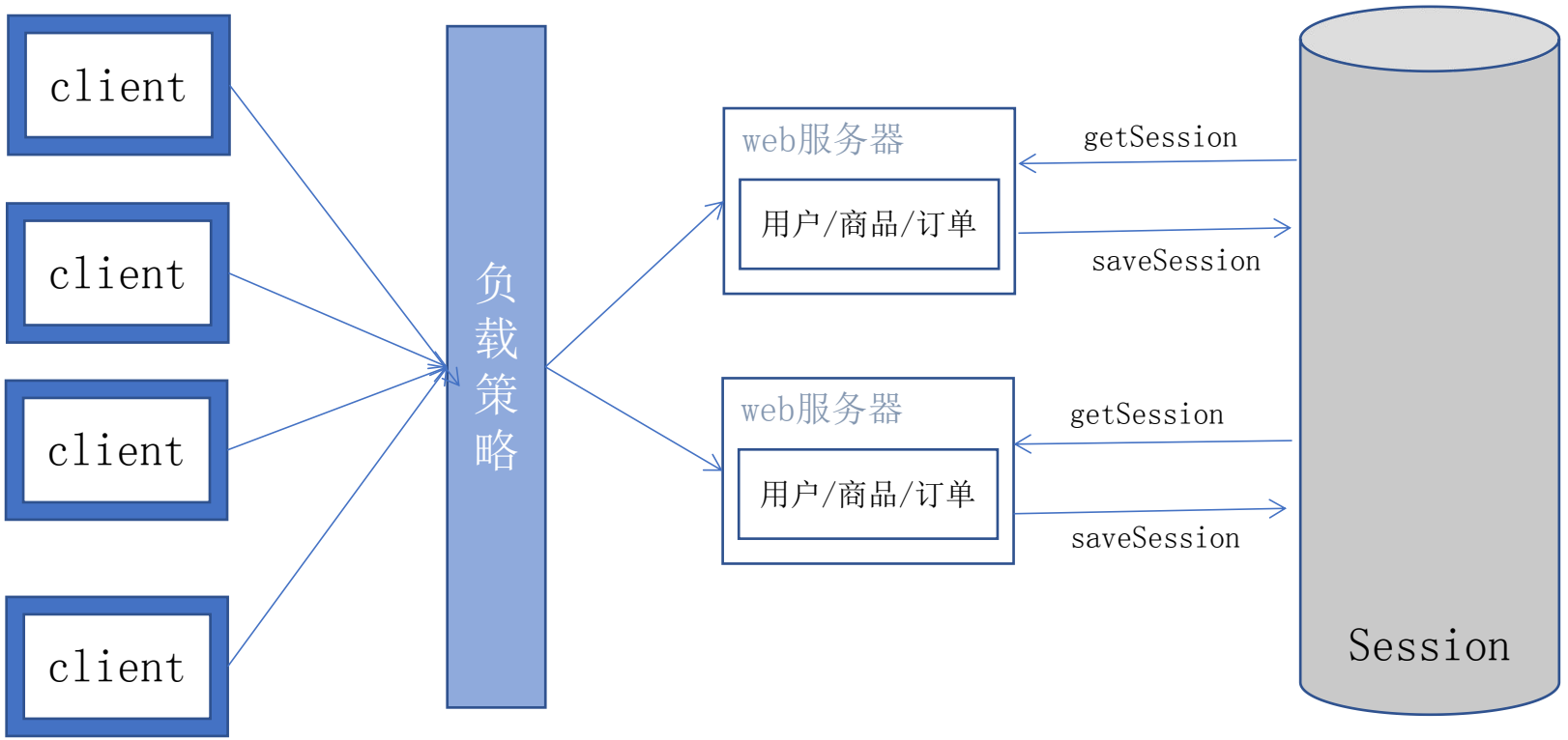
1、缓存（list/redis/memached）
2、横向拓展（集群负载）
3、拆分高负载服务，独立为一模块
4、大表数据切片（ mysql分库分区分表）
5、使用搜索中间件： solr/elasticsearch



架构解决：session跨域共享

方案1：
负载使用 hash（ip）

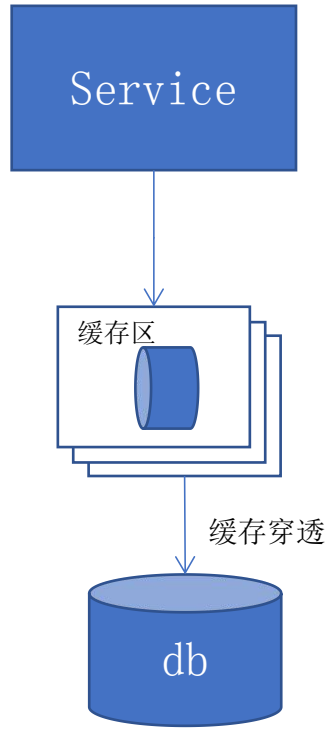
方案2：
使用redis共享session



架构解决：缓存方案

一般缓存方案

1、先到缓存中查，有值直接返回
2、无值（缓存穿透、击穿）则调用接口或者查库，并将值补入缓存区
3、缓存区数据与db中可能不一致，使用过期时间调节
4、若缓存区数据集中在某一短时刻失效，将导致大量的缓存击穿（雪崩）

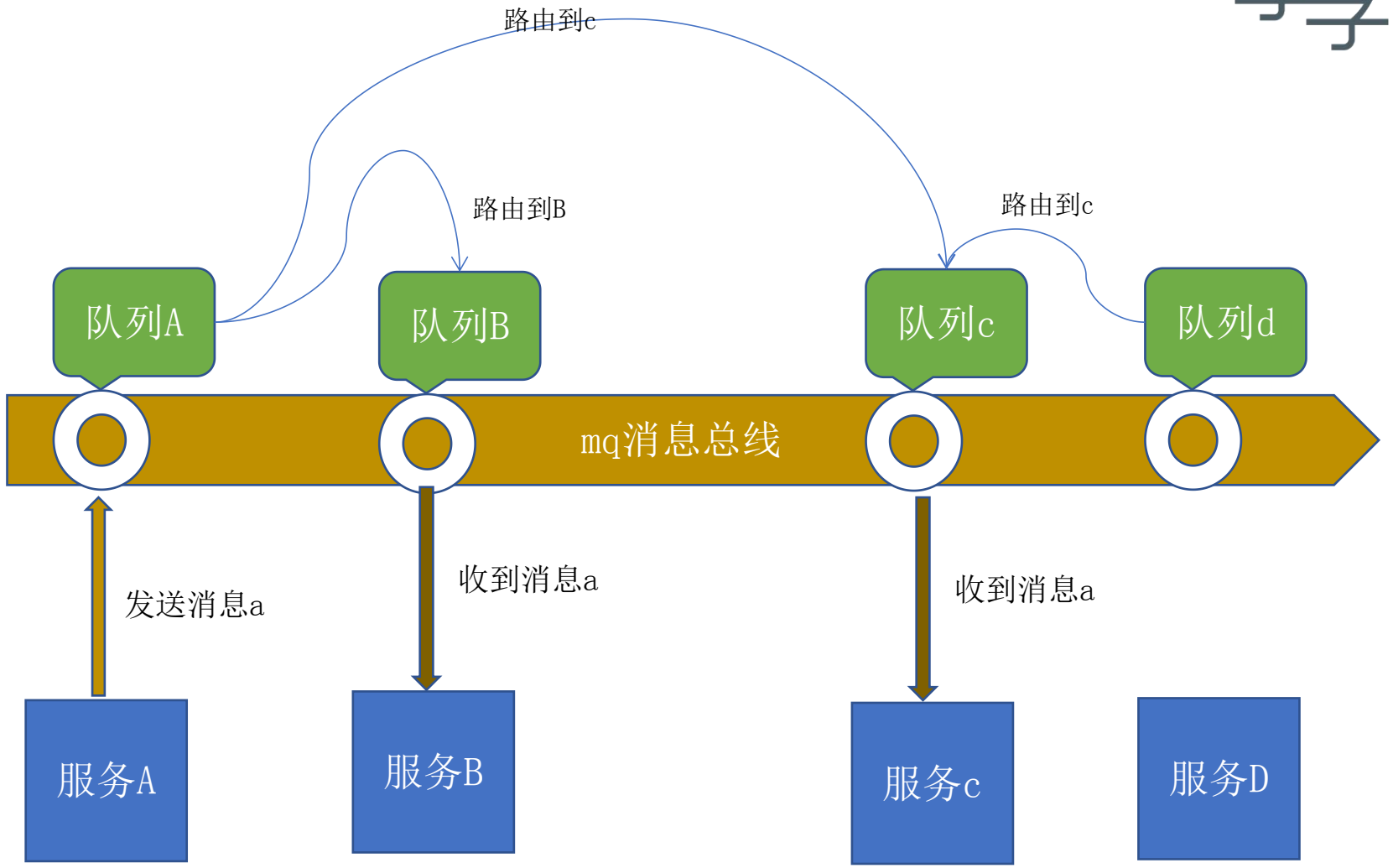


永不过期方案

1、不设置过期时间，数据永久有效，避免雪崩
2、需要额外机制来实现数据的同步更新（参照数据同步）

架构解决：mq方案

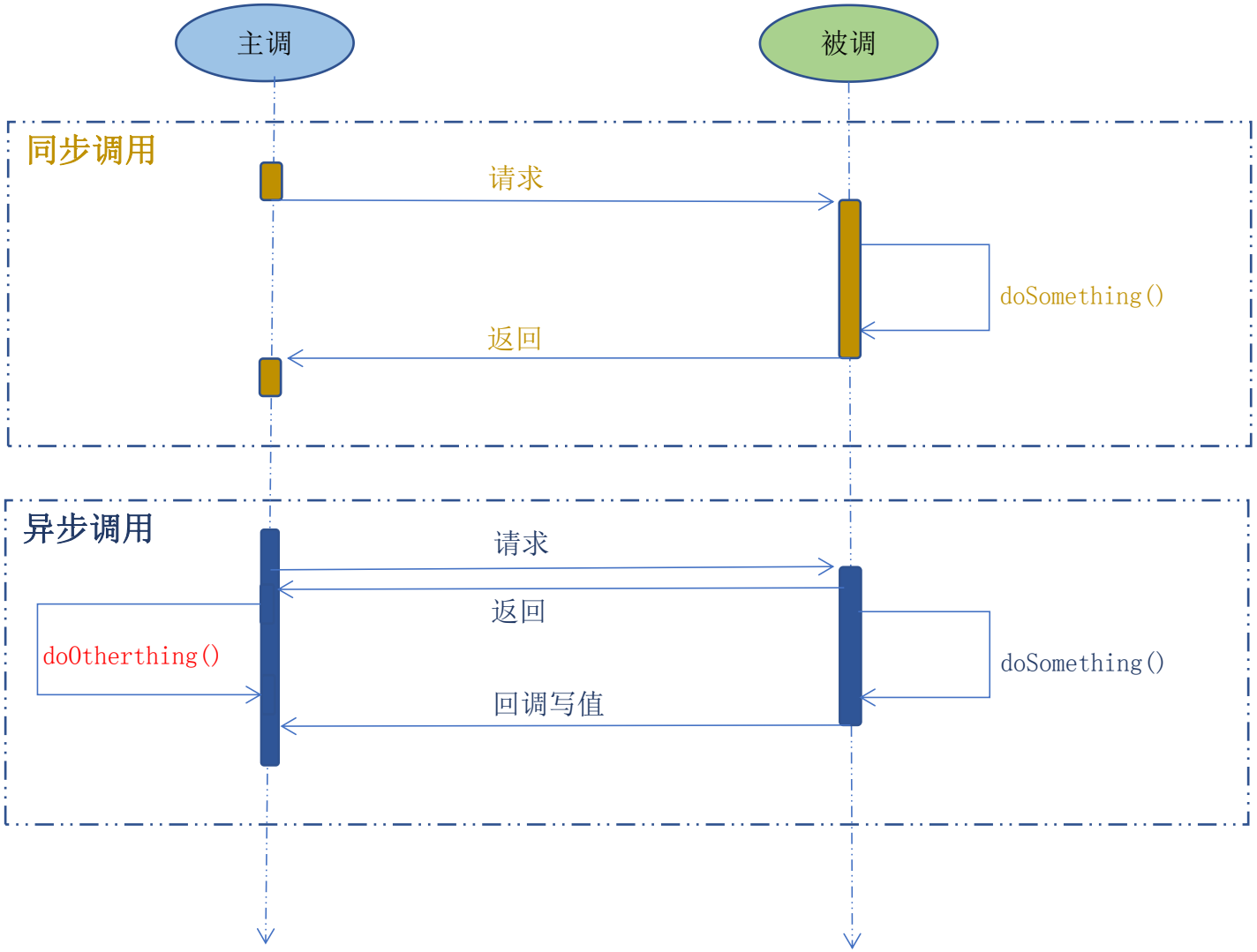
1、每个应用启动时，主动注册队列
2、后续收/发信息，只管收/发队列中数据
3、队列中数据的路由策略，
由mq管理者来配置，跟应用程序无关



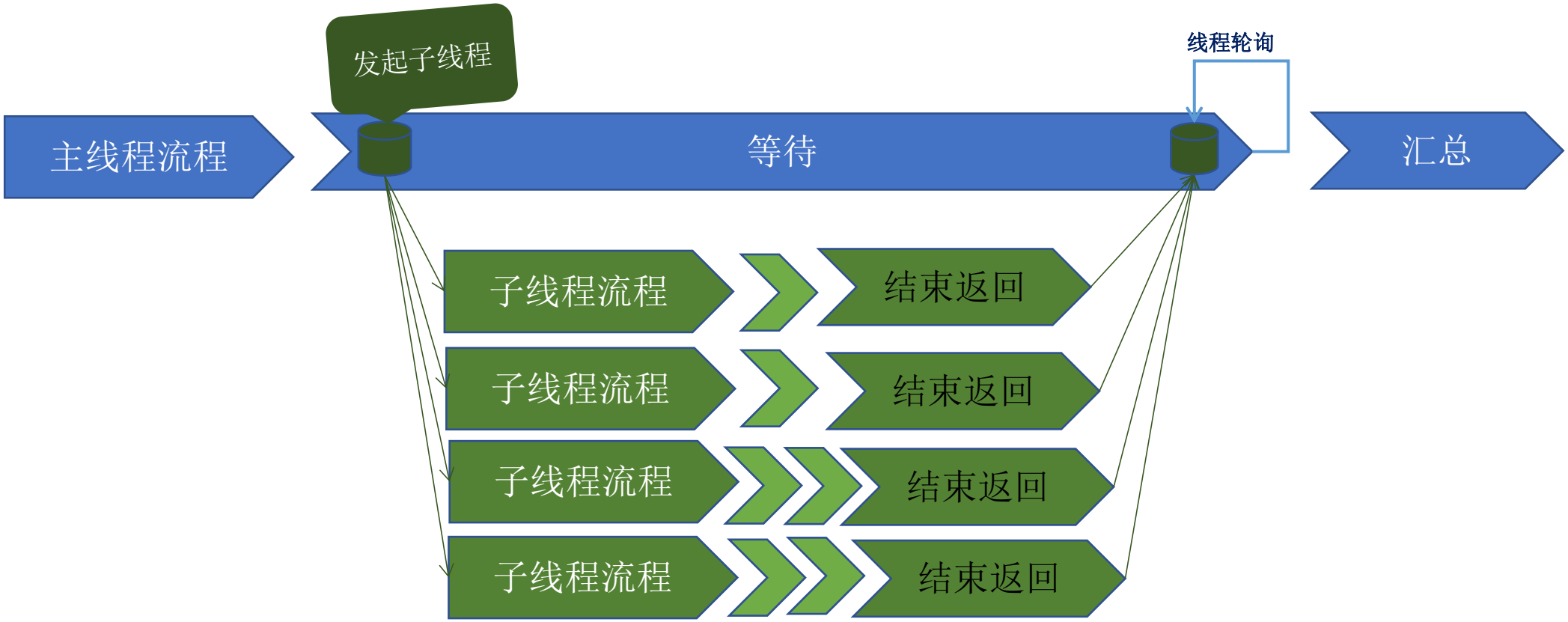
同步、异步：

同步调度：
1、调度期间，主调和被调线程被同时占用。
2、被调执行完成前，主调等待。
3、程序内部的调度，则为一单线程。

异步调度：
1、主调与被调只是一次消息发送，信息到达即返回。
2、被调执行完成后，回调一次主调方，发送结果回来。
3、程序内部的调度，则回调函数是由被调线程执行。



同步转异步：空间换时间


发起子线程
线程轮询
主线程流程
等待
汇总
子线程流程
结束返回
子线程流程
结束返回
子线程流程
结束返回
子线程流程
结束返回

架构解决：数据分片

Redis/es/fastdfs，将数据按片切分：

1、切成6个片，每个片存储总量1/6数据
2、则两个库每个库分担三个片
3、若三个库，则每个库只需要承担两个片
4、路由管理，只记录数据与片柱的关系

此模式实现集群的动态扩容