希赛

案例特训专题3

（系统设计）

Web开发

希赛

Web开发涉及技术的综合应用。

高性能　　高可用　　可维护　　应变　　安全

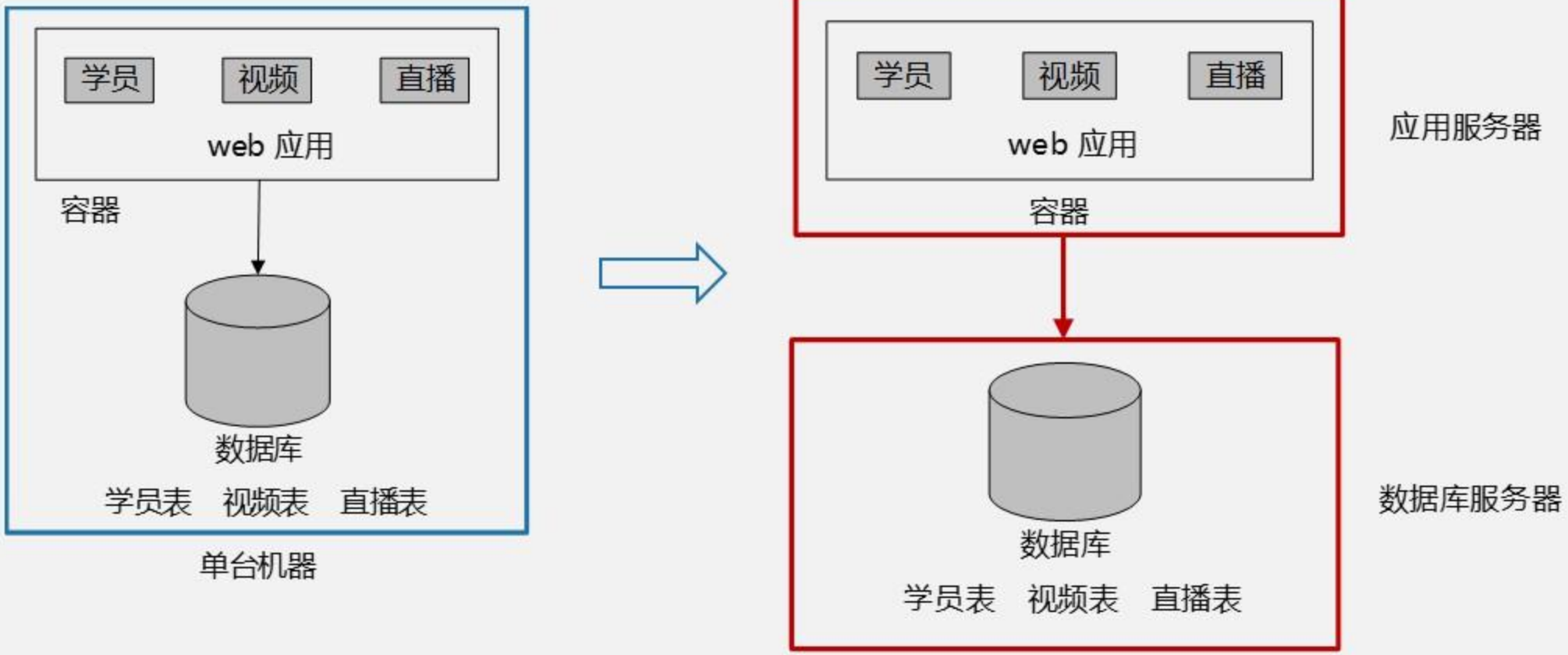
Web开发

希赛

维度	涉及技术内容
从架构来看	MVC，MVP，MVVM，REST，Webservice，微服务
从并发分流来看	集群（负载均衡）、CDN
从缓存来看	MemCache，Redis，Squid
从数据来看	主从库（主从复制），内存数据库，反规范化技术，NoSQL，分区（分表）技术，视图与物化视图
从持久化来看	Hibernate，Mybatis
从分布存储来看	Hadoop，FastDFS，区块链
从数据编码看	XML，JSON
从Web应用服务器来看	Apache，WebSphere，WebLogic，Tomcat，JBOSS，IIS
从安全性看	SQL注入攻击
其它	静态化，有状态与无状态，响应式Web设计，中台

单台机器 到 数据库与Web服务器分离

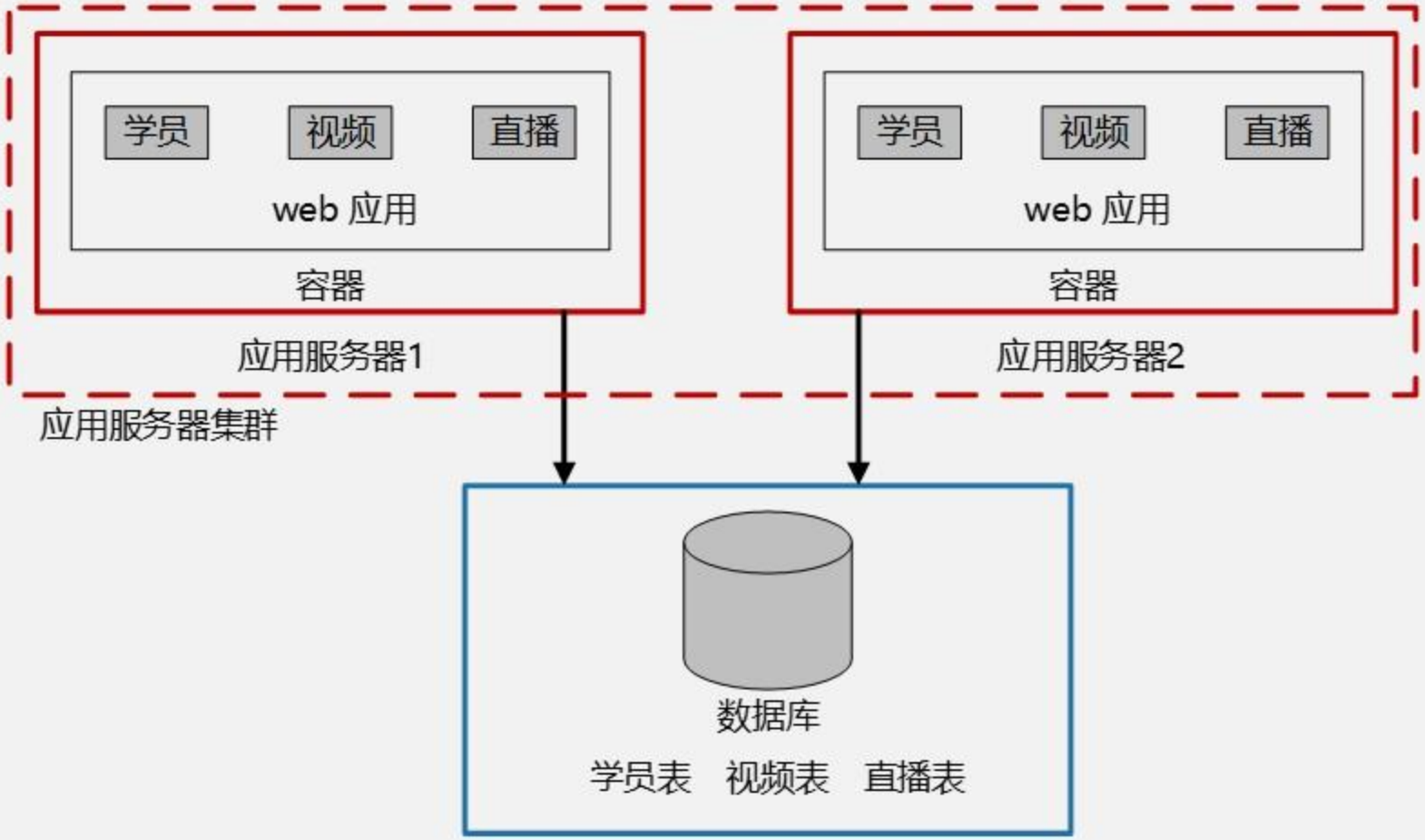
希赛



数据库与web服务器分离

应用服务器集群

希赛



数据库服务器

应用服务器集群

希赛

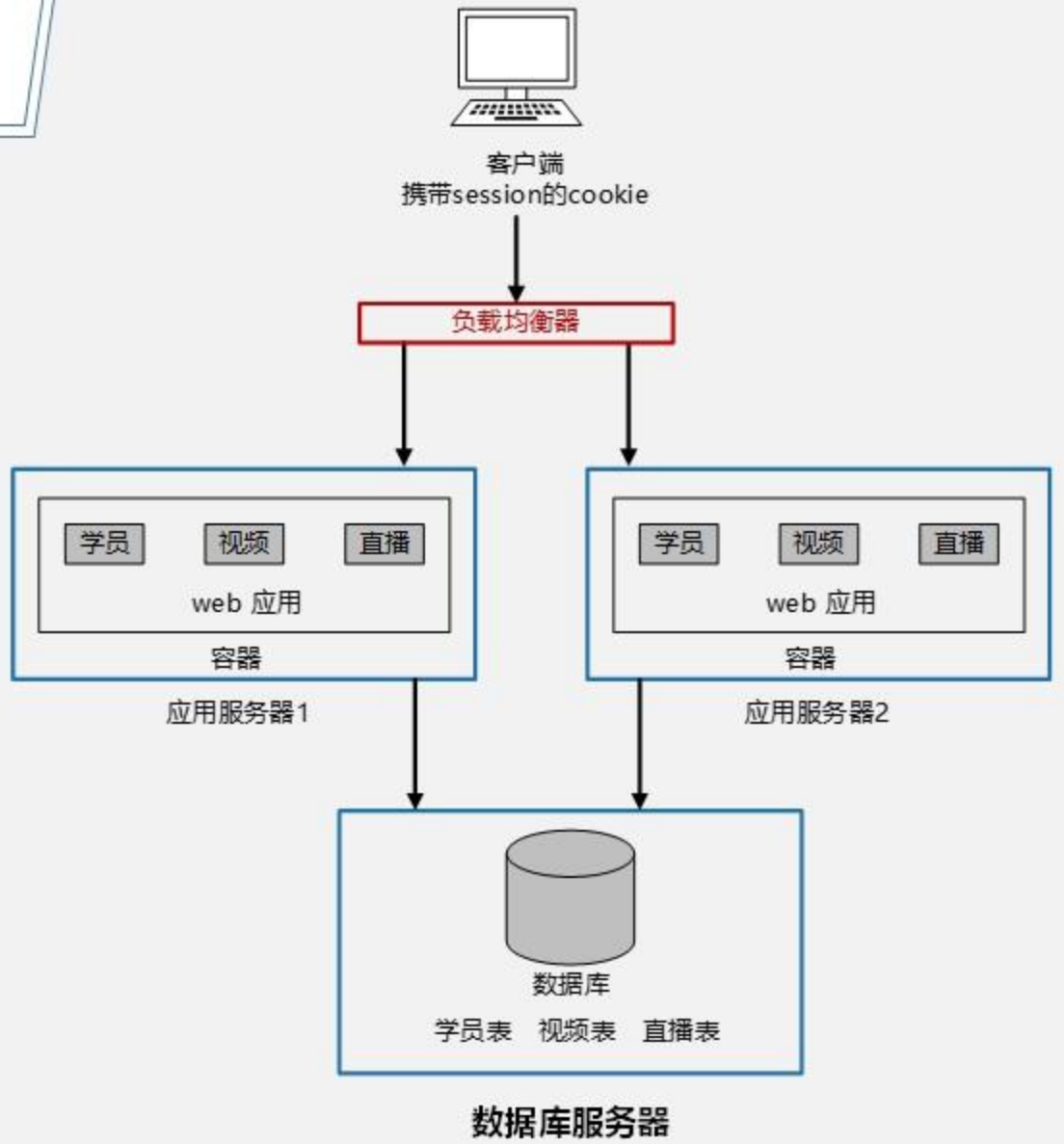
系统演变到这里，将会出现下面几个问题：

1、用户的请求由谁来转发到具体的应用服务器

2、用户如果每次访问到的服务器不一样，那么如何维护session的一致性

1、负载均衡
2、有状态与无状态问题

负载均衡的引入


客户端
携带session的cookie
负载均衡器
学员
视频
直播
web 应用
容器
应用服务器1
学员
视频
直播
web 应用
容器
应用服务器2
数据库
学员表 视频表 直播表
数据库服务器

负载均衡技术

希赛

- ✓ 基于特定软件的负载均衡（HTTP重定向）（应用层）
- ✓ 反向代理负载均衡（应用层）
- ✓ 基于DNS的负载均衡（传输层）
- ✓ 基于NAT的负载均衡（传输层）
- ✓ 混合型负载均衡

负载均衡技术

希赛

应用层负载均衡

1、http重定向。HTTP重定向就是应用层的请求转发。用户的请求其实已经到了HTTP重定向负载均衡服务器，服务器根据算法要求用户重定向，用户收到重定向请求后，再次请求真正的集群。

特点：实现简单，但性能较差。

2、反向代理服务器。在用户的请求到达反向代理服务器时（已经到达网站机房），由反向代理服务器根据算法转发到具体的服务器。常用的apache，nginx都可以充当反向代理服务器。

特点：部署简单，但代理服务器可能成为性能的瓶颈。

负载均衡技术

希赛

传输层负载均衡

1、DNS域名解析负载均衡。DNS域名解析负载均衡就是在用户请求DNS服务器，获取域名对应的IP地址时，DNS服务器直接给出负载均衡后的服务器IP。

特点：效率比HTTP重定向高，减少维护负载均衡服务器成本。但一个应用服务器故障，不能及时通知DNS，而且DNS负载均衡的控制权在域名服务商那里，网站无法做更多的改善和更强大的管理。

2、基于NAT的负载均衡。基于NAT的负载均衡将一个外部IP地址映射为多个IP地址，对每次连接请求动态地转换为一个内部节点的地址。

特点：技术较为成熟，一般在网关位置，可以通过硬件实现。像四层交换机一般就采用了这种技术。

负载均衡技术

希赛

◆ 静态算法（不考虑动态负载）

（1）轮转算法：轮流将服务请求（任务）调度给不同的节点（即：服务器）。

（2）加权轮转算法：考虑不同节点处理能力的差异。

（3）源地址哈希散列算法：根据请求的源IP地址，作为散列键从静态分配的散列表找出对应的节点。

（4）目标地址哈希散列算法：根据请求目标IP做散列找出对应节点。

（5）随机算法：随机分配，简单，但不可控。

◆ 动态算法（考虑动态负载）

（1）最小连接数算法：新请求分配给当前活动请求数量最少的节点，每个节点处理能力相同的情况下。

（2）加权最小连接数算法：考虑节点处理能力不同，按最小连接数分配。

（3）加权百分比算法：考虑了节点的利用率、硬盘速率、进程个数等，使用利用率来表现剩余处理能力。

◆ 硬件负载均衡：F5

◆ 软件负载均衡：LVS、Nginx、HAproxy

Session共享机制

方案1：携带session的cookie

客户端

负载均衡器

方案2：服务器间同步session

学员 视频 直播

web 应用

容器

同步

学员 视频 直播

web 应用

容器

应用服务器1

redis

应用服务器2

方案3：将session存入redis

数据库

学员表 视频表 直播表

数据库服务器

有状态与无状态

希赛

无状态服务（stateless service）对单次请求的处理，不依赖其他请求，也就是说，处理一次请求所需的全部信息，要么都包含在这个请求里，要么可以从外部获取到（比如说数据库），服务器本身不存储任何信息。

有状态服务（stateful service）则相反，它会在自身保存一些数据，先后的请求是有关联的。

判断以下构件是有状态服务还是无状态服务：

（1）Identification Bean（身份认证构件）
（2）ResPublish Bean（资源发布构件）
（3）ResRetrieval Bean（资源检索构件）
（4）OnlineEdit Bean（在线编辑构件）
（5）Statistics Bean（统计分析构件）

持久化技术——ORM

希赛

ORM（Object Relational Mapping）：对象与关系数据之间的映射。

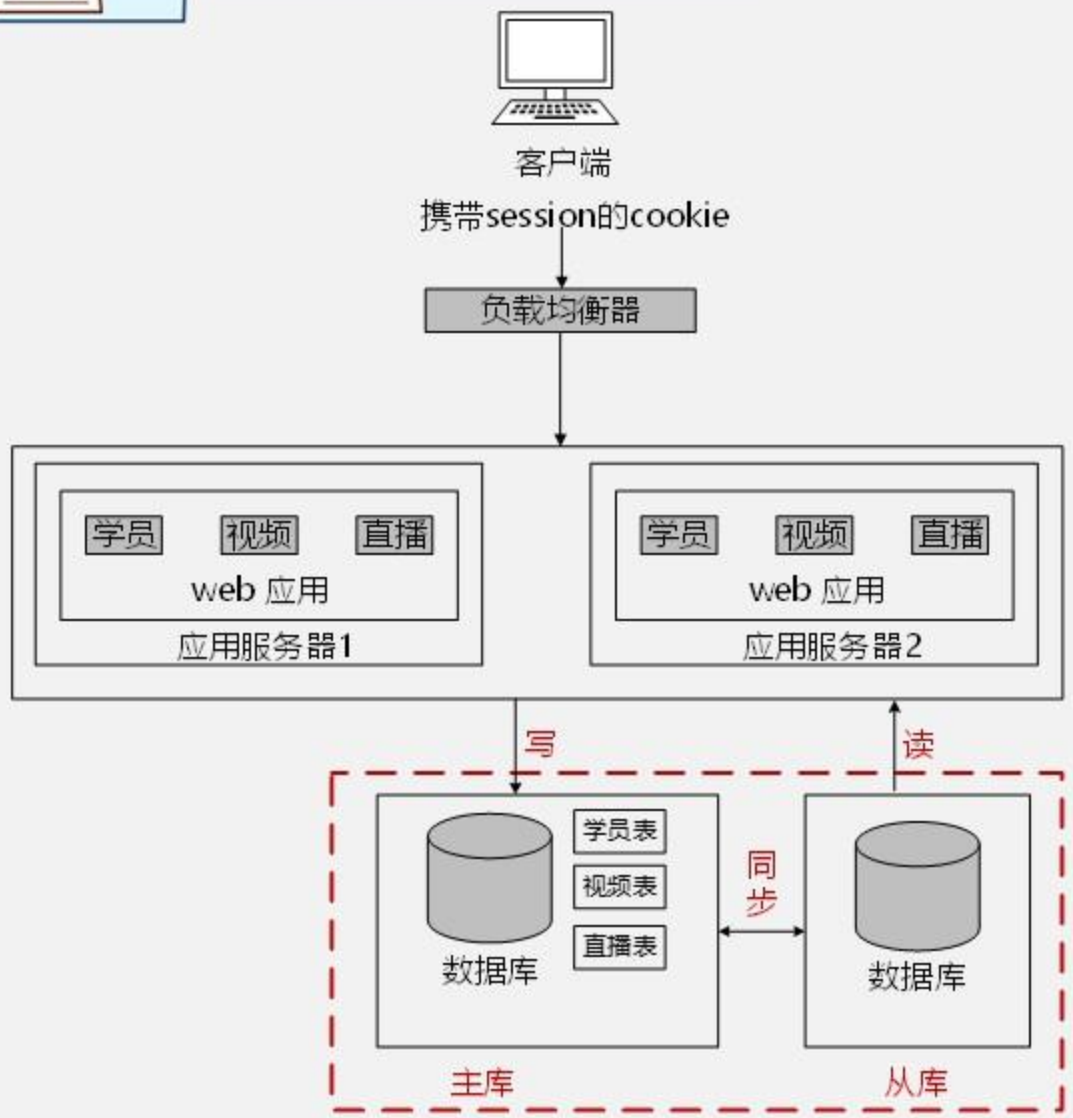
映射关系表

面向对象	关系数据库
类（class）	数据库的表（table）
对象（object）	记录（record，行数据）
对象的属性（attribute）	字段（field）

实现技术对比表

维度	Hibernate	MyBatis
简单对比	强大，复杂，间接，SQL无关	小巧，简单，直接，SQL相关
可移植性	好（不关心具体数据库）	差（根据数据库SQL编写）
复杂多表关联	不支持	支持

数据库读写分离化



主从数据库结构特点：

1、一般：一主多从，也可以多主多从。

2、主库做写操作，从库做读操作。

主从复制步骤：

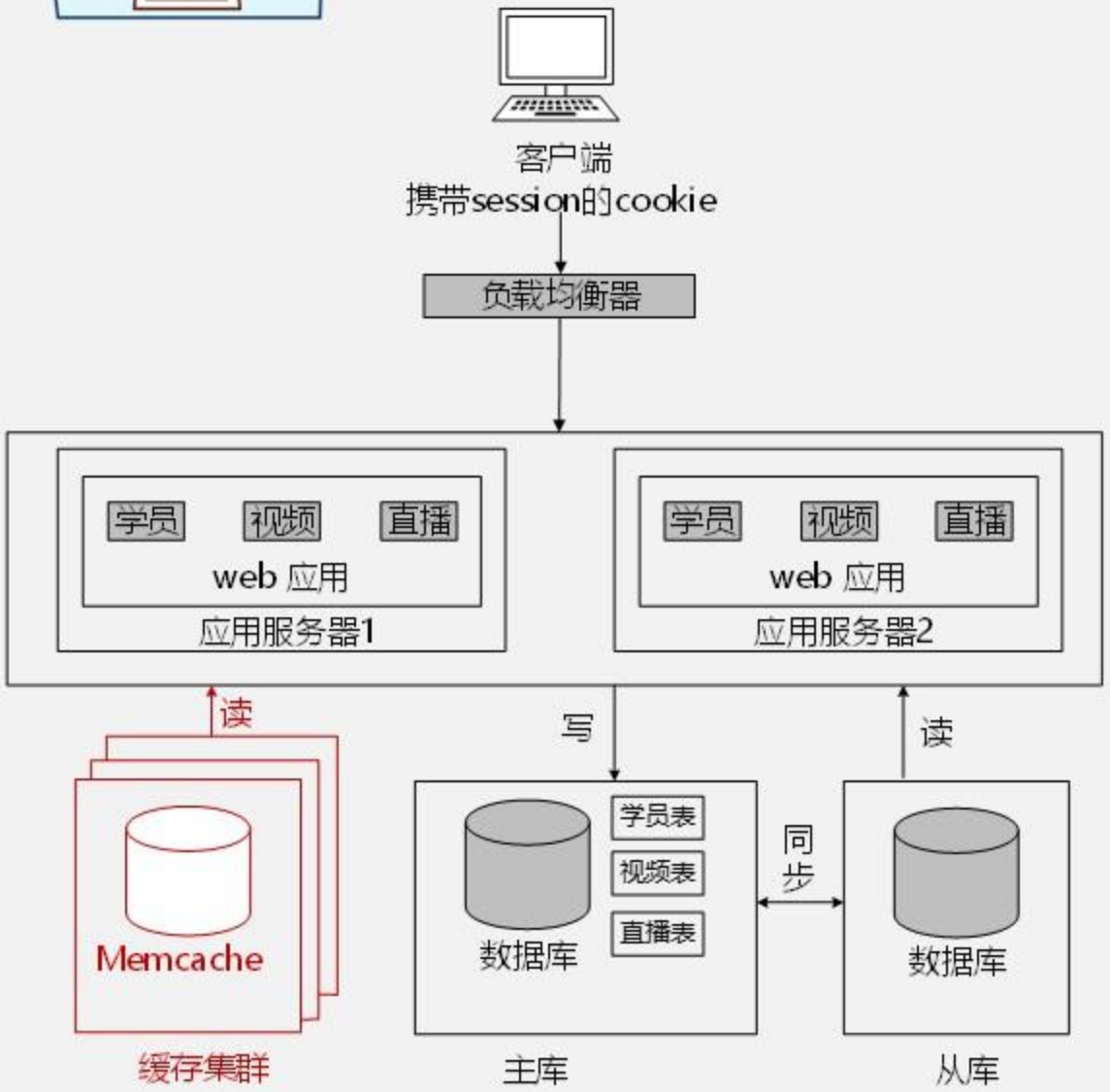
1、主库（Master）更新数据完成前，将操作写binlog日志文件。

2、从库（Salve）打开I/O线程与主库连接，做binlog dump process，并将事件写入中继日志。

3、从库执行中继日志事件，保持与主库一致。

用缓存缓解读库的压力

希赛



常见缓存技术：

MemCache：Memcache是一个高性能的分布式的内存对象缓存系统，用于动态Web应用以减轻数据库负载。Memcache通过在内存里维护一个统一的巨大的hash表，它能够用来存储各种格式的数据，包括图像、视频、文件以及数据库检索的结果等。

Redis：Redis是一个开源的使用ANSI C语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value数据库，并提供多种语言的API。

Squid：Squid是一个高性能的代理缓存服务器，Squid支持FTP、gopher、HTTPS和HTTP协议。

缓存技术

希赛

Redis 与 Memcache 能力比较

工作	MemCache	Redis
数据类型	简单key/value结构	丰富的数据结构
持久性	不支持	支持
分布式存储	客户端哈希分片/一致性哈希	多种方式，主从、Sentinel、Cluster等
多线程支持	支持	不支持（Redis6.0开始支持）
内存管理	私有内存池/内存池	无
事务支持	不支持	有限支持
数据容灾	不支持，不能做数据恢复	支持，可以在灾难发生时恢复数据

内存数据库

希赛

Redis集群切片的常见方式

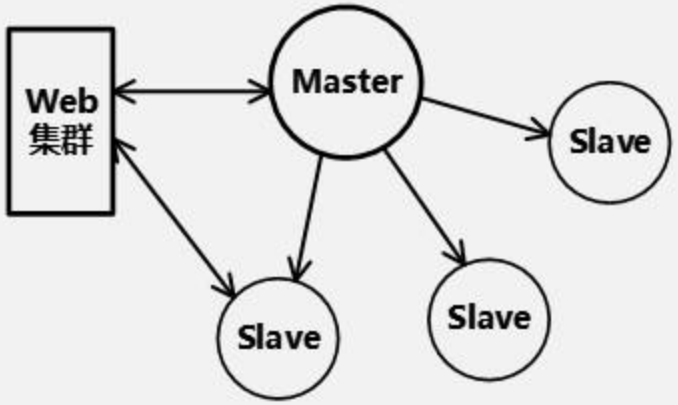
集群切片方式	核心特点
客户端分片	在客户端通过key的hash值对应到不同的服务器。
中间件实现分片	在应用软件和Redis中间，例如：Twemproxy、Codis等，由中间件实现服务到后台Redis节点的路由分派。
客户端服务端协作分片	Redis Cluster模式，客户端可采用一致性哈希，服务端提供错误节点的重定向服务slot上。不同的slot对应到不同服务器。

内存数据库

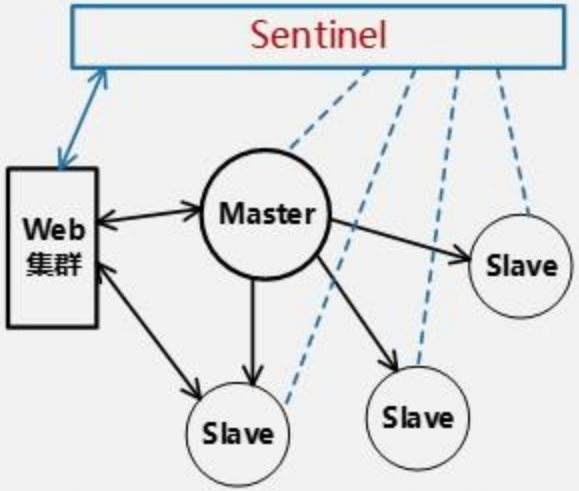
希赛

Redis分布式存储方案

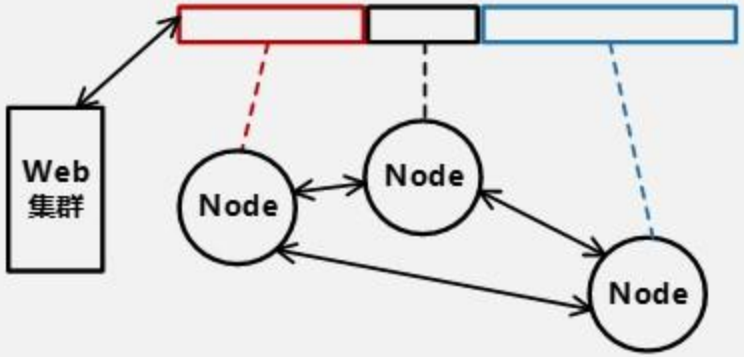
分布式存储方案	核心特点
主从（Master/Slave）模式	一主多从，故障时手动切换
哨兵（Sentinel）模式	有哨兵的一主多从，主节点故障自动选择新的主节点
集群（Cluster）模式	分节点对等集群，分slots，不同slots的信息存储到不同节点



主从（Master/Slave）模式



哨兵（Sentinel）模式



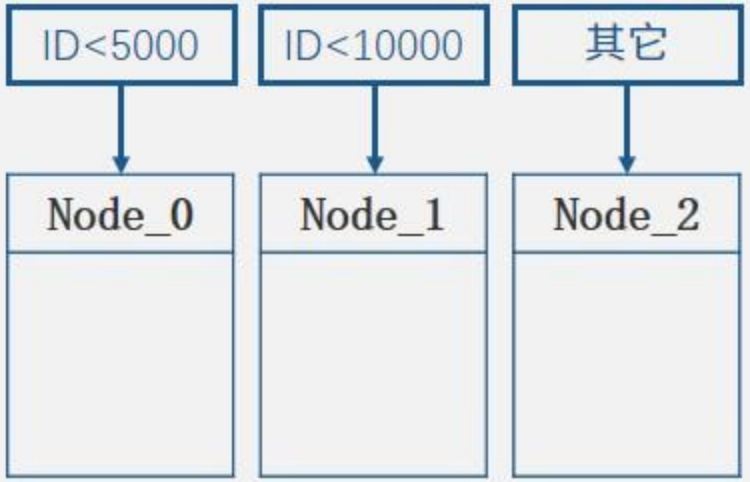
集群（Cluster）模式

缓存技术

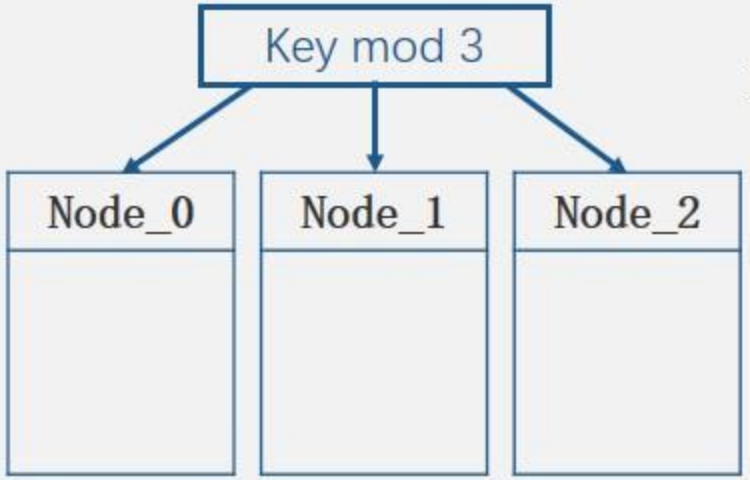
希赛

Redis数据分片方案

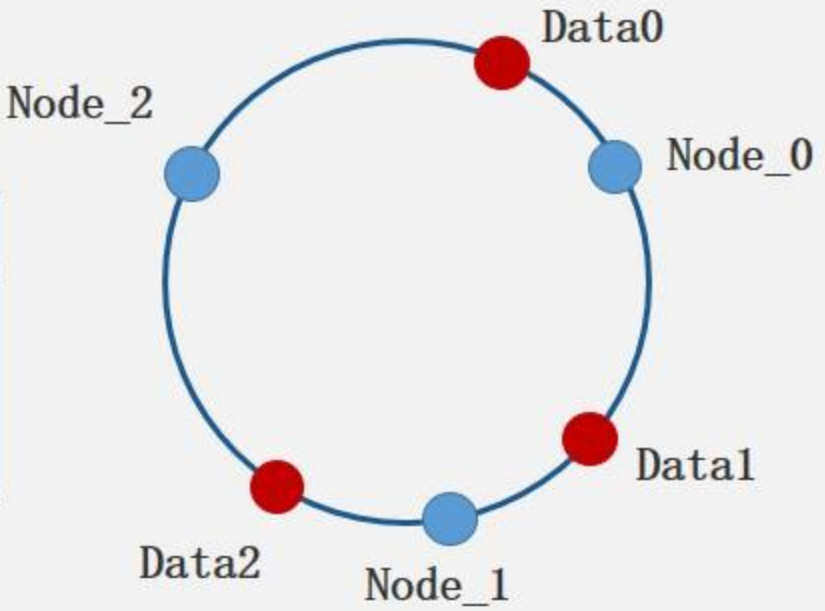
范围分片



哈希分片



一致性哈希分片



缓存技术

希赛

Redis数据分片方案

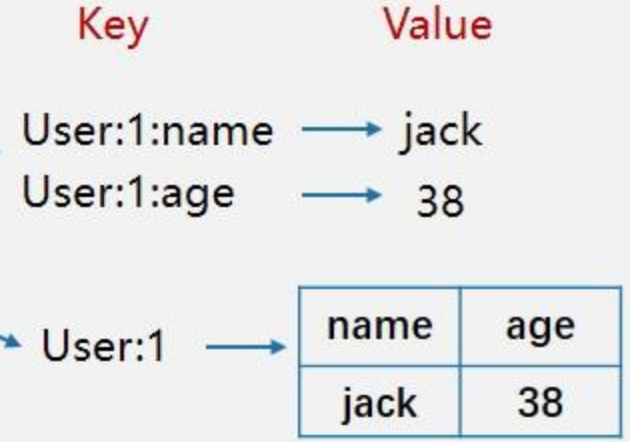
分片方案	分片方式	说明
范围分片	按数据范围值来做分片	例：按用户编号分片，0-999999映射到实例A；1000000-1999999映射到实例B。
哈希分片	通过对key进行hash运算分片	可以把数据分配到不同实例，这类似于取余操作，余数相同的，放在一个实例上。
一致性哈希分片	哈希分片的改进	可以有效解决重新分配节点带来的无法命中问题。

缓存技术

希赛

Redis 数据类型

类型	特点	示例
String（字符串）	存储二进制，任何类型数据，最大512MB	缓存，计数，共享Session
Hash（字典）	无序字典，数组+链表，适合存对象 Key对应一个HashMap。针对一组数据	存储、读取、修改用户属性
List（列表）	双向链表，有序，增删快查询慢	消息队列，文章列表 记录前N个最新登录的用户ID列表
Set（集合）	键值对无序，唯一 增删查复杂度均为O（1），支持交/并/差集操作	独立IP，共同好爱，标签
Sorted Set（有序集合）	键值对有序，唯一，自带按权重排序效果	排行榜



缓存技术

希赛

Redis的持久化主要有两种方式：RDB和AOF。

RDB：传统数据库中快照的思想。指定时间间隔将数据进行快照存储。

AOF：传统数据库中日志的思想，把每条改变数据集的命令追加到AOF文件末尾，这样出问题了，可以重新执行AOF文件中的命令来重建数据集。

对比维度	RDB持久化	AOF持久化
备份量	重量级的全量备份，保存整个数据库	轻量级增量备份，一次只保存一个修改命令
保存间隔时间	保存间隔时间长	保存间隔时间短，默认1秒
还原速度	数据还原速度快	数据还原速度慢
阻塞情况	save会阻塞，但bgsave或者自动不会阻塞	无论是平时还是AOF重写，都不会阻塞
数据体积	同等数据体积：小	同等数据体积 ：大
安全性	数据安全性：低，容易丢数据	数据安全性：高，根据策略决定

缓存技术

希赛

淘汰作用范围	机制名	策略
不淘汰	noeviction	禁止驱逐数据，内存不足以容纳新入数据时，新写入操作就会报错。系统默认的一种淘汰策略。
设置了过期时间的键空间	volatile-random	随机移除某个key
	volatile-lru	优先移除最近未使用的key
	volatile-ttl	ttl值小的key优先移除
全键空间	allkeys-random	随机移除某个key
	allkeys-lru	优先移除最近未使用的key

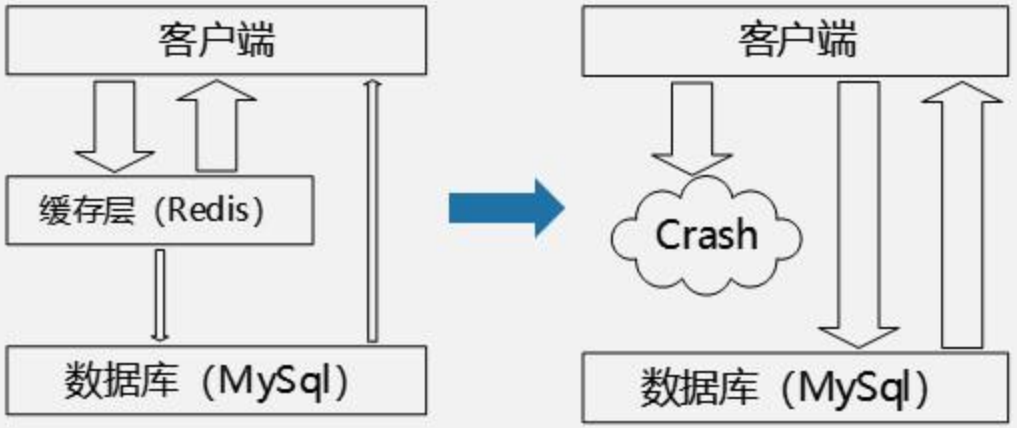
缓存技术

希赛

Redis常见问题

1、缓存雪崩

大部分缓存失效 -> 数据库崩溃



解决方案

1、使用锁或队列：保证不会有大量的线程对数据库一次性进行读写，从而避免失效时大量的并发请求落到底层存储系统上。

2、为key设置不同的缓存失效时间：在固定的一个缓存时间的基础上+随机一个时间作为缓存失效时间。

3、二级缓存：设置一个有时间限制的缓存+一个无时间限制的缓存。避免大规模访问数据库。

缓存技术

希赛

Redis常见问题

2、缓存穿透

查询无数据返回 -> 直接查数据库

解决方案

1、如果查询结果为空，直接设置一个默认值存放到缓存，这样第二次到缓冲中获取就有值了。设置一个不超过5分钟的过期时间，以便能正常更新缓存。
2、设置布隆过滤器，将所有可能存在的数据哈希到一个足够大的bitmap中，一个一定不存在的数据会被这个bitmap拦截掉，从避免了对底层存储系统的查询压力。

3、缓存预热

系统上线后，将相关需要缓存的数据直接加到缓存系统中。

解决方案

1、直接写个缓存刷新页面，上线时手工操作。
2、数据量不大时，可以在项目启动的时候自动进行加载。
3、定时刷新缓存。

缓存技术

希赛

Redis常见问题

4、缓存更新

除Redis系统自带的缓存失效策略，常见采用以下两种：

1、定时清理过期的缓存。

2、当有用户请求过来时，再判断这个请求所用到的缓存是否过期，过期的话就去底层系统得到新数据并更新缓存。

5、缓存降级

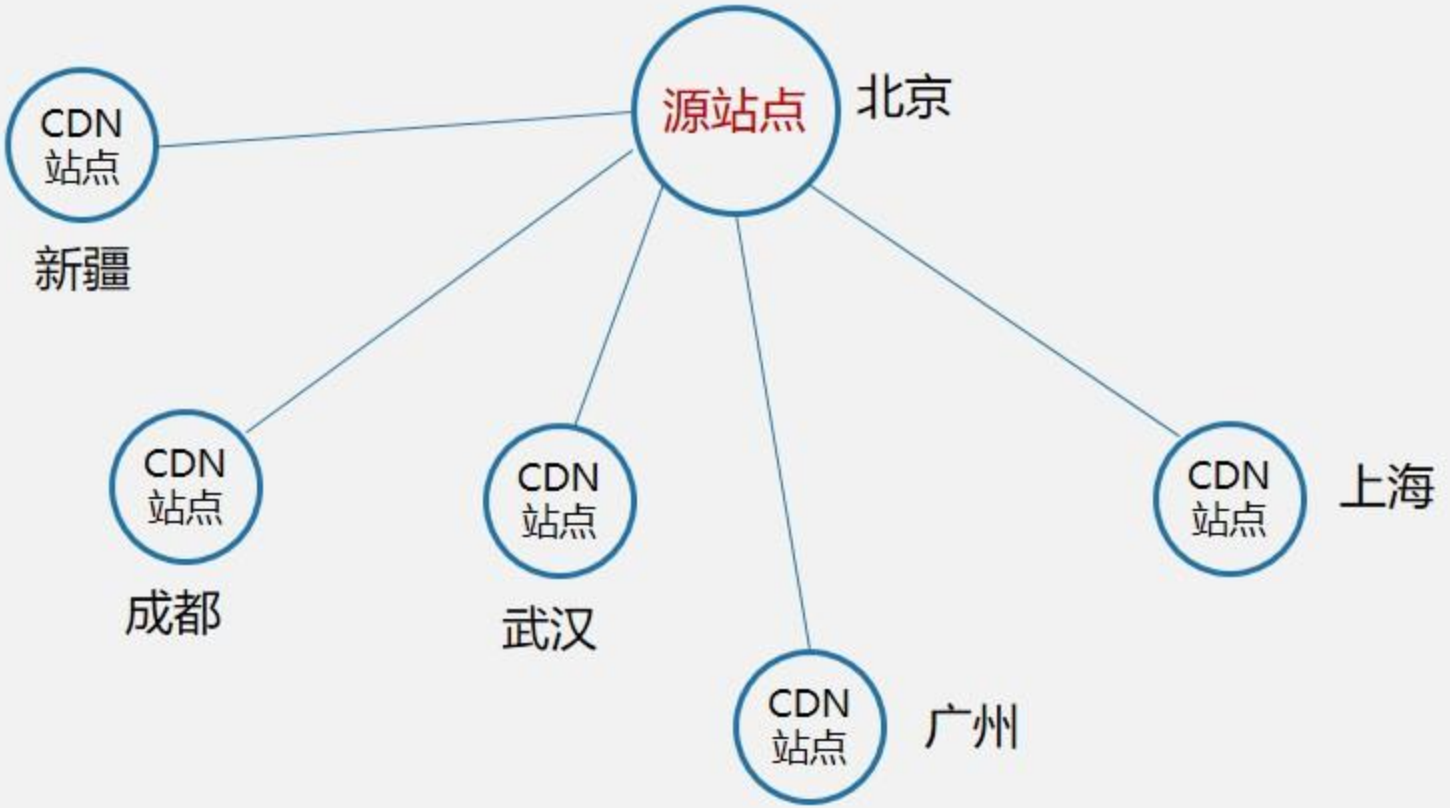
降级的目的是保证核心服务可用，即使是有损的，而且有些服务是无法降级的（如电商的购物流程等）

在进行降级之前要对系统进行梳理，从而梳理出哪些必须保护，哪些可降级。

CDN（内容分发网络）

希赛

CDN的全称是Content Delivery Network，即内容分发网络。其基本思路是尽可能避开互联网上有可能影响数据传输速度和稳定性的瓶颈和环节，使内容传输得更快、更稳定。



XML与JSON

希赛

扩展标记语言（Extensible Markup Language，XML），用于标记电子文件使其具有结构性的标记语言，可以用来标记数据、定义数据类型，是一种允许用户对自己的标记语言进行定义的源语言。

XML的优点

A. 格式统一，符合标准；

B. 容易与其他系统进行远程交互，数据共享比较方便。

XML的缺点

A. XML文件庞大，文件格式复杂，传输占带宽；

B. 服务器端和客户端都需要花费大量代码来解析XML，导致服务器端和客户端代码变得异常复杂且不易维护；

C. 客户端不同浏览器之间解析XML的方式不一致，需要重复编写很多代码；

D. 服务器端和客户端解析XML花费较多的资源和时间。

XML与JSON

希赛

JSON（JavaScript Object Notation）一种轻量级的数据交换格式，具有良好的可读和便于快速编写的特性。可在不同平台之间进行数据交换。

JSON的优点

1、数据格式比较简单，易于读写，格式都是压缩的，占用带宽小；

2、易于解析，客户端JavaScript可以简单的通过eval()进行JSON数据的读取；

3、支持多种语言，包括ActionScript，C，C#，ColdFusion，Java，JavaScript，Perl，PHP，Python，Ruby等服务器端语言，便于服务器端的解析；

4、因为JSON格式能直接为服务器端代码使用，大大简化了服务器端和客户端的代码开发量，且完成任务不变，并且易于维护。

JSON的缺点

没有XML格式推广得这么深入人心和使用广泛，没有XML那么通用。

Web应用服务器

希赛

WEB应用服务器可以理解为两层意思：

（1）WEB服务器：其职能较为单一，就是把浏览器发过来的Request请求返回Html页面。

（2）应用服务器：进行业务逻辑的处理。

Apache：Web服务器，市场占有率60%左右。它可以运行在几乎所有的Unix、Windows、Linux系统平台上。

IIS：早期Web服务器，目前小规模站点仍有应用。

Tomcat：开源、运行servlet和JSP Web应用软件的基于Java的Web应用软件容器。

JBOSS：JBOSS是基于J2EE的开放源代码的应用服务器。一般与Tomcat或Jetty绑定使用。

WebSphere：一种功能完善、开放的Web应用程序服务器，它是基于Java的应用环境，用于建立、部署和管理Internet 和Intranet Web应用程序。

WebLogic：BEA WebLogic Server是一种多功能、基于标准的web应用服务器，为企业构建自己的应用提供了坚实的基础。

Jetty：Jetty是一个开源的servlet容器，它为基于Java的web内容，例如JSP和servlet提供运行环境。

REST（表述性状态传递）

希赛

REST（Representational State Transfer，表述性状态转移）是一种只使用HTTP和XML进行基于Web通信的技术，可以降低开发的复杂性，提高系统的可伸缩性。

REST的5个原则

（1）网络上的所有事物都被抽象为资源。

（2）每个资源对应一个唯一的资源标识。

（3）通过通用的连接件接口对资源进行操作。

（4）对资源的各种操作不会改变资源标识。

（5）所有的操作都是无状态的。

响应式Web设计

响应式WEB设计是一种网络页面设计布局，其理念是：集中创建页面的图片排版大小，可以智能地根据用户行为以及使用的设备环境进行相对应的布局。

方法与策略

（1）采用流式布局和弹性化设计：使用相对单位，设定百分比而非具体值的方式设置页面元素的大小。

（2）响应式图片：不仅要同比的缩放图片，还要在小设备上降低图片自身的分辨率。

关于中台

希赛

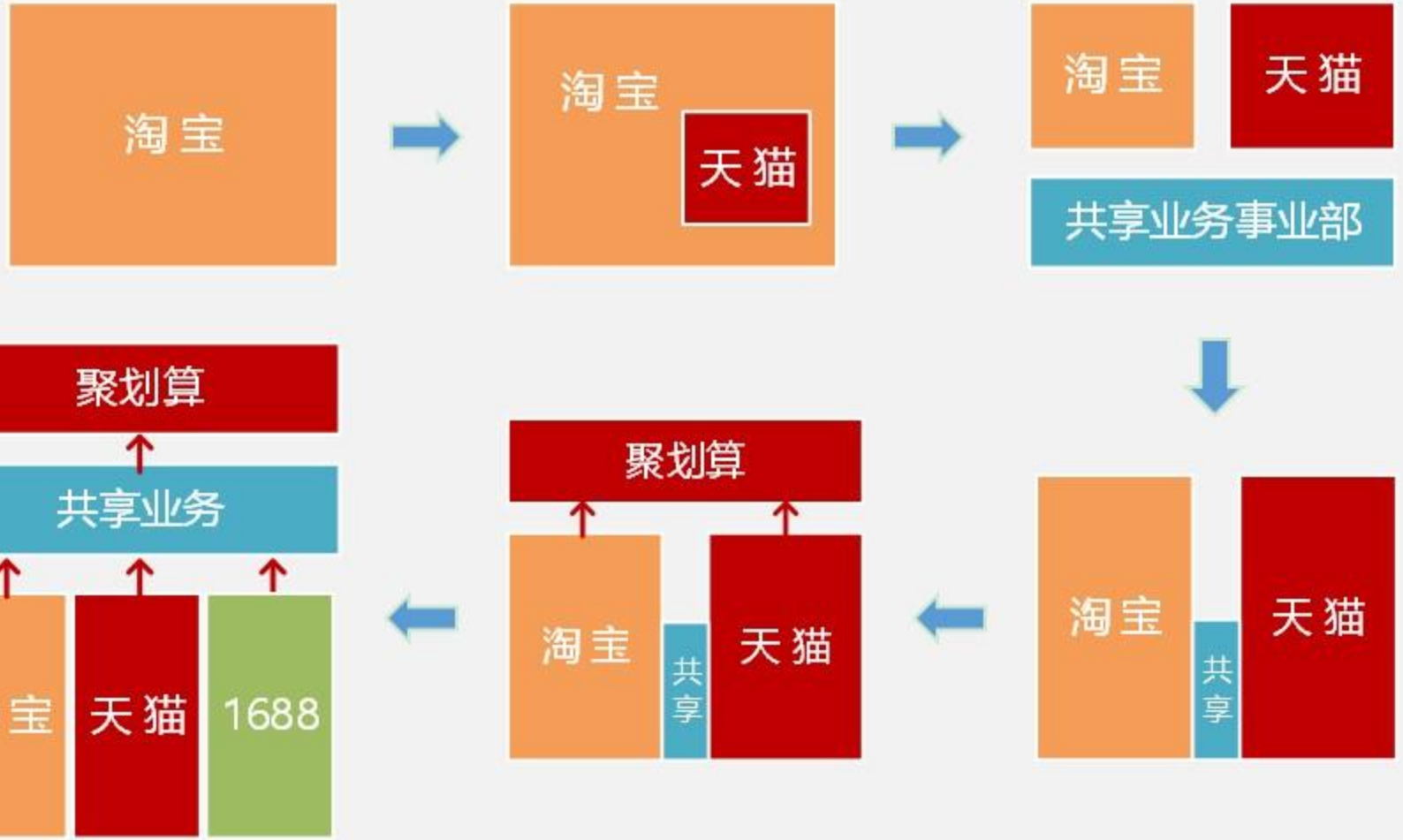
中台是一套结合互联网技术和行业特性，将企业核心能力以共享服务形式沉淀，形成“大中台、小前台”的组织和业务机制，供企业快速低成本的进行业务创新的企业架构。中台又可以进一步细分，比如业务中台，数据中台，XX中台。本质上，都是对企业通用能力在不同层面的沉淀，并对外能力开放。

中台的践行者：

Supercell：芬兰移动游戏巨头，2015年世界游戏前10占5席，员工仅200多人，因使用中台，具有小团队快速开发能力，后被腾讯86亿美金收购。

阿里：2015年参观Supercell，而后推行中台。

关于中台



关于中台

希赛

淘宝 | 天猫 | 1688 | 聚划算 | ...

共享业务（中台）

用户中心 | 商品中心 | 交易中心

阿里云平台

关于中台

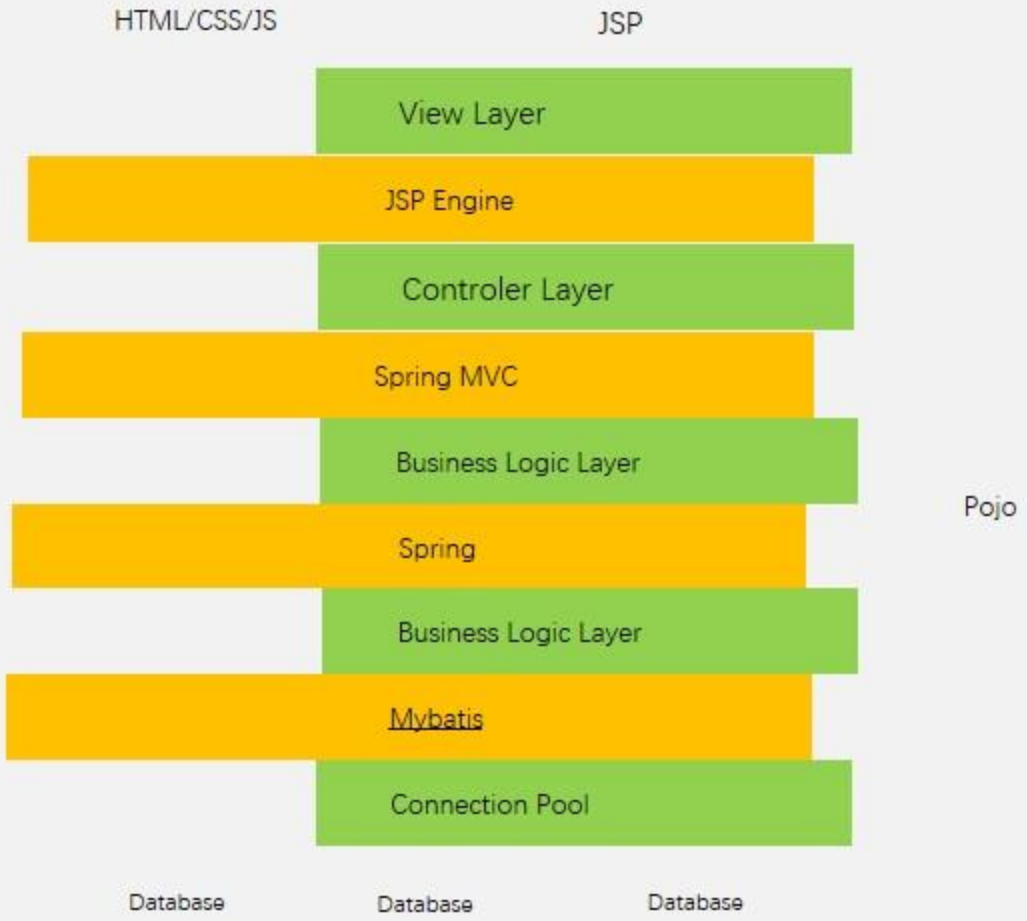
希赛

业务中台 VS 数据中台

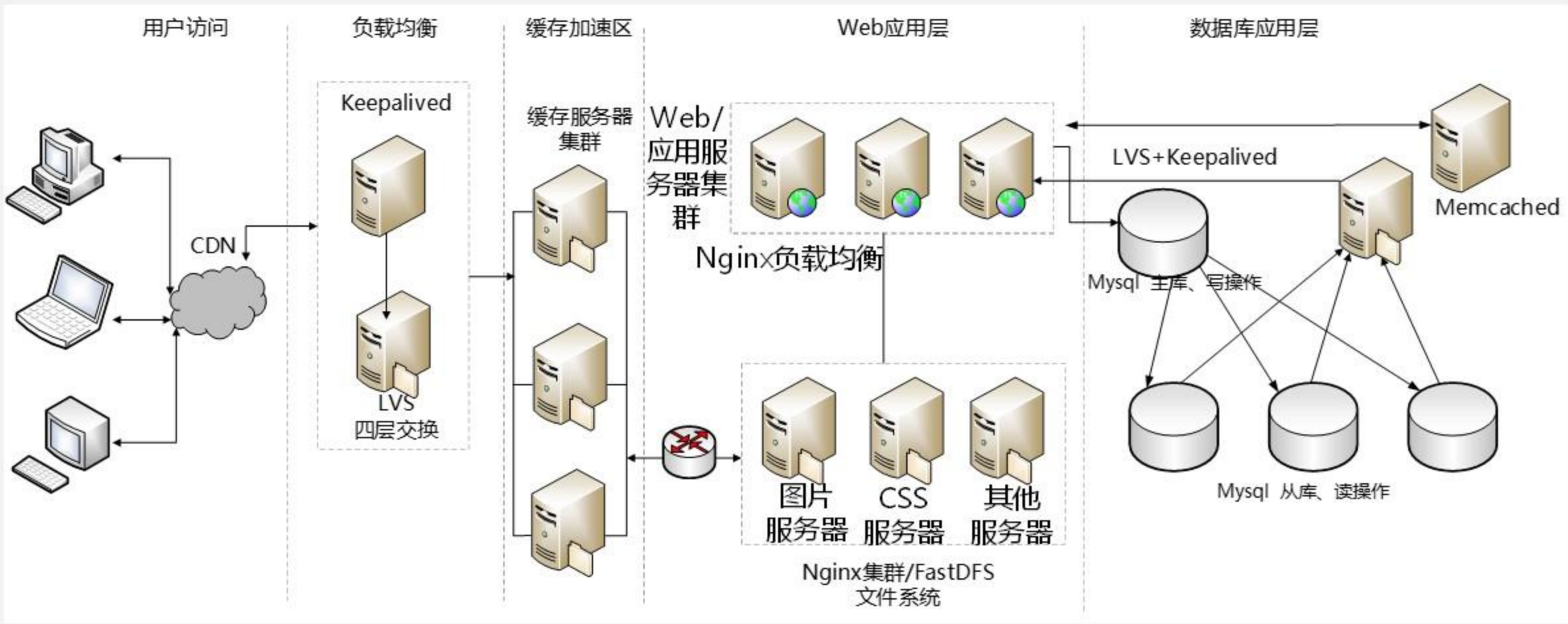
- ✓ 多个电商渠道使用一个下单服务，一个订单接口同时为多个前台系统提供服务。
- ✓ 多个前台系统，根据一个用户的手机号，获取对应的画像，用户的标签。
- ✓ 将多个支付通道，抽象建立成一个支付API，暴露给前台业务系统。
- ✓ 通过一个订单编号，来获取可能的商品推荐清单，从而做到交叉销售。

Web系统分层

希赛


HTML/CSS/JS
JSP
View Layer
JSP Engine
Controler Layer
Spring MVC
Business Logic Layer
Spring
Business Logic Layer
Mybatis
Connection Pool
Pojo
Database
Database
Database

Web系统分层


用户访问
负载均衡
缓存加速区
Web应用层
数据库应用层
Keepalived
CDN
LVS
四层交换
缓存服务器
集群
Web/
应用服
务器集
群
Nginx负载均衡
LVS+Keepalived
Memcached
Mysql 主库、写操作
图片
服务器
CSS
服务器
其他
服务器
Nginx集群/FastDFS
文件系统
Mysql 从库、读操作

案例分析练习题1

希赛

阅读以下关于分布式数据库缓存设计的叙述，在答题纸上回答问题1至问题3。

【说明】

某企业是为城市高端用户提供高品质蔬菜生鲜服务的初创企业，创业初期为快速开展业务，该企业采用轻量型的开发架构（脚本语言+关系型数据库）研制了一套业务系统。业务开展后受到用户普遍欢迎，用户数和业务数量迅速增长，原有的数据库服务器已不能满足高度并发的业务要求。为此，该企业成立了专门的研发团队来解决该问题。

张工建议重新开发整个系统， 采用新的服务器和数据架构，解决当前问题的同时为日后的扩展提供支持。但是，李工认为张工的方案开发周期过长，投入过大，当前应该在改动尽量小的前提下解决该问题。李工认为访问量很大的只是部分数据，建议采用缓存工具MemCache来减轻数据库服务器的压力，这样开发量小，开发周期短，比较适合初创公司，同时将来也可以通过集群进行扩展。然而，刘工又认为李工的方案中存在数据可靠性和一致性问题，在宕机时容易丢失交易数据，建议采用Redis来解决问题。经过充分讨论，该公司最终决定采用刘工的方案。

案例分析练习题1

希赛

【问题1】（9分）

在李工和刘工的方案中，均采用分布式数据库缓存技术来解决问题。请说明分布式数据库缓存的基本概念。

表1中对MemCache和Redis两种工具的优缺点进行了比较，请补充完善表 1中的空（1）~ （6）。

表1 MemCache与Redis能力比较

	MemCache	Redis
数据类型	简单key/value结构	(1)
持久性	(2)	支持
分布式存储	(3)	多种方式，主从、Sentinel、Cluster等
多线程支持	支持	(4)
内存管理	(5)	无
事务支持	(6)	有限支持

【问题2】（8分）

刘工认为李工的方案存在数据可靠性和一致性的问题，请说明原因。

为避免数据可靠性和一致性的问题，刘工的方案采用Redis作为数据库缓存，请说明基本的Redis与原有关系数据库的数据同步方案。

【问题3】（8分）

请给出Redis分布式存储的2种常见方案和Redis集群切片的几种常见方式。

案例分析练习题1

希赛

【问题1】

分布式数据库缓存指的是在高并发环境下，为了减轻数据库压力和提高系统响应时间，在数据库系统和应用系统之间增加的独立缓存系统。

（1）key/value，list，set，hash，sorted 或丰富/多种数据结构

（2）不支持

（3）客户端哈希分片/一致性哈希

（4）不支持

（5）有，私有内存池

（6）不支持

表4-1 MemCache与Redis能力比较

	MemCache	Redis
数据类型	简单key/value结构	（1） 丰富/多种数据结构 (list, key-value, set等)
持久性	（2）不支持	支持
分布式存储	（3）客户端哈希分片/一致性哈希	多种方式，主从、Sentinel、Cluster等
多线程支持	支持	（4）不支持
内存管理	（5）有，私有内存池	无
事务支持	（6）不支持	有限支持

案例分析练习题1 - 参考答案

希赛

【问题2】

（1）Memcache没有持久化功能，所以掉电数据会全部丢失，而且无法直接恢复，这存在可靠性问题。

（2）Memcache不支持事务，所以操作过程中可能产生数据的不一致性。

同步方案：

读取数据时，先读取Redis中的数据，如果Redis没有，则从原数据库中读取，并同步更新Redis中的数据。写回时，写入到原数据库中，并同步更新至Redis中。

【问题3】

Redis分布式存储的常见方案：

1、主从模式（Master/Slave）

2、哨兵模式（Sentinel）

3、集群模式（Cluster）

Redis集群切片的常见方式：

1、客户端分片，即在客户端就通过key的hash值对应到不同的服务器。

2、中间件实现分片。在应用软件和Redis中间，例如：Twemproxy、Codis等，由中间件实现服务到后台Redis节点的路由分派。

3、客户端服务端协作分片。Redis Cluster模式，客户端可采用一致性哈希，服务端提供错误节点的重定向服务slot上。不同的slot对应到不同服务器。

案例分析练习题2

希赛

某电子商务企业因发展良好，客户量逐步增大，企业业务不断扩充，导致其原有的B2C 商品交易平台已不能满足现有业务需求。因此，该企业委托希赛公司重新开发一套商品交易平台。该企业要求新平台应可适应客户从手机、平板设备、电脑等不同终端设备访问系统，同时满足电商定期开展“秒杀”“限时促销”等活动的系统高并发访问量的需求。面对系统需求，希赛公司召开项目组讨论会议，制定系统设计方案。讨论会议上，王工提出可以应用响应式Web设计满足客户从不同设备正确访问系统的需求。 同时，采用增加镜像站点、CDN内容分发等方式解决高并发访问量带来的问题。李工在王工的提议上补充，仅仅依靠上述外网加速技术不能完全解决高用户并发访问问题，如果访问量持续增加，系统仍存在崩溃可能。李工提出应同时结合负载均衡、缓存服务器、Web应用服务器、分布式文件系统、分布式数据库等方法设计系统架构。经过项目组讨论，最终决定综合王王和李工的思路，完成新系统的架构设计。

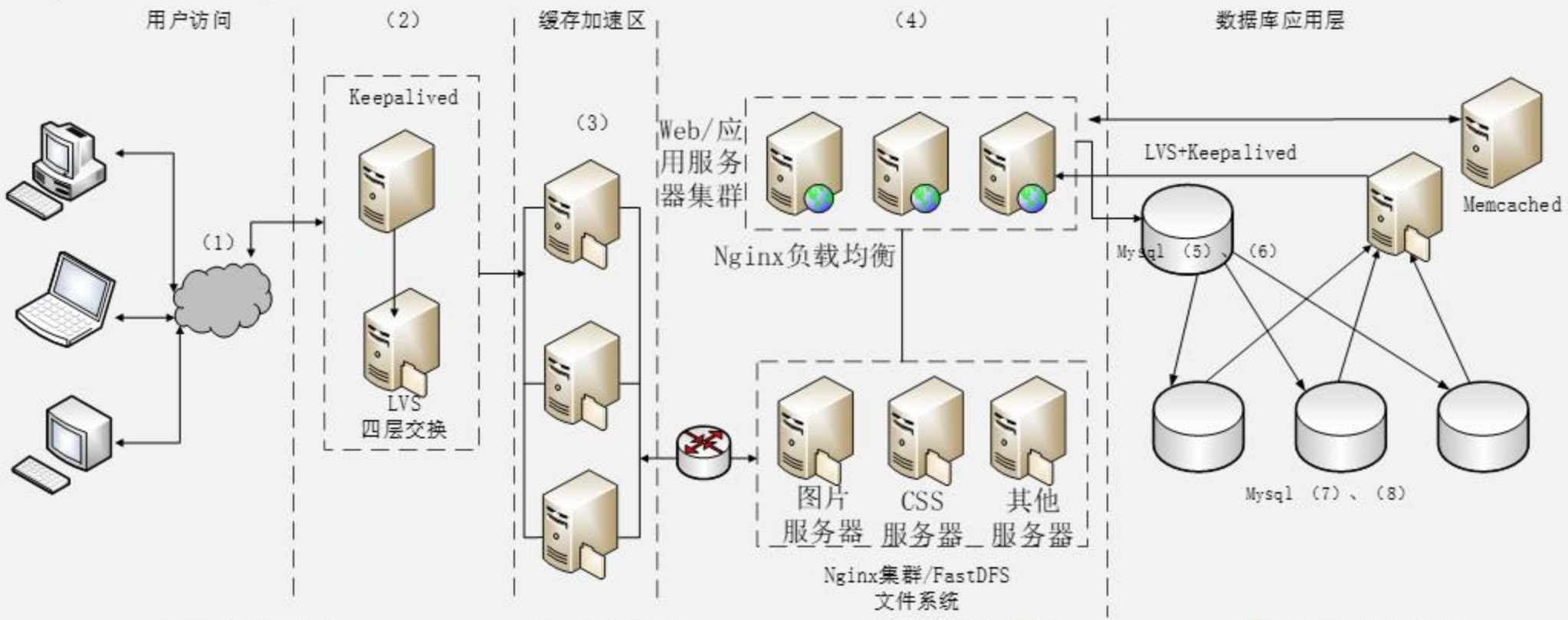
【问题1】（5分）

请用 200字以内的文字描述什么是“响应式Web设计”，并列举 2个响应式 Web 设计的实现方式。

【问题2】（16分）

综合王工和李工的提议，项目组完成了新商品交易平台的系统架构设计方案。新系统架构图如图所示。请从选项（a）-（j）中为架构图中（1）-（8）处空白选择相应的内容，补充支持高并发的Web应用系统架构设计图。

案例分析练习题2



(a) Web应用层　(b) 界面层　(c) 负载均衡层　(d) CDN内容分发

(e) 主数据库　(f) 缓存服务器集群　(g) 从数据库　(h) 写操作

(i) 读操作　(j) 文件服务器集群

【问题3】（4分）

根据李工的提议，新的B2C商品交易平台引入了主从复制机制。请针对B2C商品交易平台的特点，简要叙述引入该机制的好处。

案例分析练习题2 - 参考答案

希赛

【问题1】

响应式web设计是指我们设计与开发的页面可以根据用户的行为和不同的设备环境做出相应的响应来调整页面的布局，以提供用户可感知的、流畅的阅读和操作体验。

实现方式：

（1）流式布局　　（2）弹性布局（如弹性图片）　　（3）媒体查询

【问题2】

（1）（d）　　（2）（c）　　（3）（f）

（4）（a）　　（5）（6）（e）（h）　　（7）（8）（g）（i）

【问题3】

1、提升性能

交易平台要求高并发，主从复制方式一主多从，不同的用户请求可以从不同的从数据库读取数据，提高并发度。

2、可扩展性更优

如果采用单台数据库服务器，则访问量持续增加时，数据库瓶颈暴露，且无法迅速解决问题。而主从结构可以快速增加从服务器数量，以满足需求。

3、提升可用性

一主多从，一台从服务器出现故障不影响整个系统正常工作。

4、相当于负载均衡

一主多从分担任务，相当于负载均衡。

5、提升数据安全性

系统中的数据冗余存放多份，不会因为某台机器硬件故障而导致数据丢失。