

## 系统架构设计师考前几页纸

### 1、架构的本质：

- (1) 软件架构为软件系统提供了一个结构、行为和属性的高级抽象。
- (2) 软件架构风格是特定应用领域的惯用模式，架构定义一个词汇表和一组约束。

### 2、数据流风格：适合于分阶段做数据处理，交互性差，包括：批处理序列、管理过滤器。

### 3、调用/返回风格：一般系统都要用到，包括：主程序/子程序，面向对象，层次结构（分层越多，性能越差）。

### 4、独立构件风格：构件是独立的过程，连接件是消息传递。包括：进程通信，事件驱动系统（隐式调用）。应用场景，通过事件触发操作。

### 5、虚拟机风格：包括解释器与基于规则的系统，有自定义场景时使用该风格。

### 6、仓库风格（以数据为中心的风格）：以共享数据源为中心，其它构件围绕中心进行处理。包括：数据库系统、黑板系统（语言处理，信号处理），超文本系统。

### 7、闭环控制架构（过程控制）：定速巡航，空调温控。

### 8、MVC：视图（JSP），控制器（Servlet），模型（EJB）。

### 9、SOA：粗粒度，松耦合，标准化。Webservice 与 ESB 是 SOA 的实现技术。

### 10、ESB：位置透明性、消息路由、服务注册命名、消息转换、多传输协议、日志与监控。

### 11、REST 的 5 大原则：所有事物抽象为资源、资源唯一标识、通过接口操作资源、操作不改变资源标识、操作无状态。

### 12、微服务特点：小，且专注于做一件事情；轻量级的通信机制；松耦合、独立部署。

### 13、微服务优势：技术异构性、弹性、扩展、简化部署、与结构相匹配、可组合性、对可替代性的优化。

### 14、微服务与 SOA 对比：

微服务	SOA
能拆分的就拆分	是整体的，服务能放一起的都放一起
纵向业务划分	是水平分多层
由单一组织负责	按层级划分不同部门的组织负责
细粒度	粗粒度
两句话可以解释明白	几百字只相当于 SOA 的目录

独立的子公司	类似大公司里面划分了一些业务单元（BU）
组件小	存在较复杂的组件
业务逻辑存在于每一个服务中	业务逻辑横跨多个业务领域
使用轻量级的通信方式，如 HTTP	企业服务总线（ESB）充当了服务之间通信的角色

15、MDA 的 3 种核心模型：平台独立模型（PIM），平台相关模型（PSM），代码 Code。

16、ADL 的三个基本元素：构件，连接件，架构配置。

17、DSSA 基本活动：领域分析（建立领域模型），领域设计（获得 DSSA），领域实现（开发和组织可复用信息）。

18、DSSA 角色：领域专家（有经验的用户、分析、设计、实现人员，“给建议”），领域分析人员（有经验的分析师，完成领域模型），领域设计人员（有经验的设计师，完成 DSSA），领域实现人员（有经验的程序员完成代码编写）。

19、DSSA 三层次模型：领域架构师对应领域开发环境，应用工程师对应领域特定的应用开发环境，操作员对应应用执行环境。

20、ABSD 方法是架构驱动，即强调由业务、质量和功能需求的组合驱动架构设计。

21、ABSD 方法有三个基础：功能的分解，通过选择架构风格来实现质量和业务需求，软件模板的使用。

22、ABSD 开发过程：

- (1) 架构需求（需求获取、生成类图、对类进行分组、打包成构件、需求评审）
- (2) 架构设计（提出架构模型、映射构件、分析构件相互作用，产生架构，设计评审）
- (3) 架构文档化：从使用者角度编写，分发给所有相关开发人员，保证开发者手中版本最新。
- (4) 架构复审：标识潜在的风险，及早发现架构设计中的缺陷和错误。
- (5) 架构实现（复审后的文档化架构，分析与设计，构件实现，构件组装，系统测试）
- (6) 架构演化（需求变化归类，架构演化计划，构件变动，更新构件相互作用，构件组装与测试，技术评审，演化后的架构）

23、架构评审四大质量属性：

- (1) 性能：代表参数（响应时间、吞吐量），设计策略（优先级队列、资源调度）。
- (2) 可用性：尽可能少的出错与尽快的恢复。代表参数（故障间隔时间，故障修复时间），设计策略（冗余、心跳线）。
- (3) 安全性：破坏机密性、完整性、不可否认性及可控性等特性。设计策略（追踪审计）
- (4) 可修改性：新增功能多少人月能完成，设计策略（信息隐藏）

24、风险点：系统架构风险是指架构设计中潜在的、存在问题的架构决策所带来的隐患。

非风险点：一般以某种做法，“是可以实现的”、“是可以接受的”方式进行描述。

敏感点：指为了实现某种特定的质量属性，一个或多个构件所具有的特性。

权衡点：影响多个质量属性的特性，是多个质量属性的敏感点。

25、基本场景的评估方法：ATAM，SAAM，CBAM。

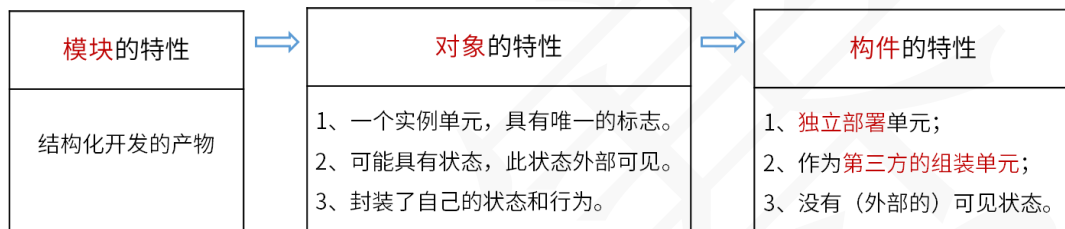
26、SAAM：最初用于分析架构可修改性，后扩展到其他质量属性。

SAAM 五个步骤：即场景开发、体系结构描述、单个场景评估、场景交互和总体评估。

27、ATAM 四大阶段：场景和需求收集、结构视图场景实现、属性模型构造和分析、折中。

ATAM：在 SAAM 的基础上发展起来的，主要针对性能、实用性、安全性和可修改性，在系统开发之前，对这些质量属性进行评价和折中。

28、构件、对象、模块的对比：



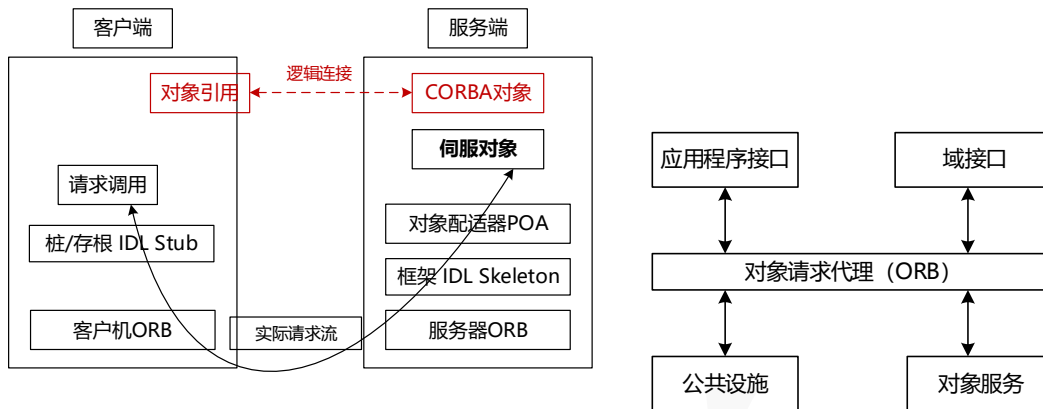
29、中间件：中间件是一种独立的系统软件或服务程序，可以帮助分布式应用软件在不同的技术之间共享资源。

30、中间件功能：客户机与服务器之间的连接和通信，客户机与应用层之间的高效率通信；应用层不同服务之间的互操作，应用层与数据库之间的连接和控制；多层架构的应用开发和运行的平台，应用开发框架，模块化的应用开发；屏蔽硬件、操作系统、网络和数据库的差异；应用的负载均衡和高可用性、安全机制与管理功能，交易管理机制，保证交易的一致性、一组通用的服务去执行不同的功能，避免重复的工作和使应用之间可以协作。

31、采用中间件技术的优点：面向需求；业务的分隔和包容性；设计与实现隔离；隔离复杂的系统资源；符合标准的交互模型；软件复用；提供对应用构件的管理。

32、主要的中间件：远程过程调用；对象请求代理；远程方法调用；面向消息的中间件；事务处理监控器。

33、中间件技术-Corba(公共对象请求代理体系结构)（代理模式）：



**伺服对象 (Servant)：** CORBA 对象的真正实现，负责完成客户端请求。

**对象适配器 (Object Adapter)：** 用于屏蔽 ORB 内核的实现细节，为服务器对象的实现者提供抽象接口，以便他们使用 ORB 内部的某些功能。

**对象请求代理 (Object Request Broker)：** 解释调用并负责查找实现该请求的对象，将参数传给找到的对象，并调用方法返回结果。客户方不需要了解服务对象的位置、通信方式、实现、激活或存储机制。

34、Bean 的分类：

- (1) 会话 Bean：描述了与客户端的一个短暂的会话。
- (2) 实体 Bean：持久化数据，O/R 映射。
- (3) 消息驱动 Bean：会话 Bean+JMS，客户把消息发送给 JMS 目的地，然后，JMS 提供者和 EJB 容器协作，把消息发送给消息驱动 Bean。支持异步消息。

35、负载均衡技术：(1) 应用层负载均衡：http 重定向、反向代理服务器；(2) 传输层负载均衡：DNS 域名解析负载均衡、基于 NAT 的负载均衡；(3) 硬件负载均衡：F5；(4) 软件负载均衡：LVS、Nginx、HAproxy。

36、有状态和无状态：

(1) 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。

(2) 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。

37、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	支持（Redis5.0 及以前版本不支持）
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

### 38、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。

### 39、缓存与数据库的协作

数据读取：根据 key 从缓存读取；若缓存中没有，则根据 key 在数据库中查找；读取到“值”之后，更新缓存。

数据写入：根据 key 值写数据库；根据 key 更新缓存。

40、REST 概念：REST（Representational State Transfer，表述性状态转移）是一种通常使用 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 的五个原则：网络上的所有事物都被抽象为资源；每个资源对应一个唯一的资源标识；通过通用的连接件接口对资源进行操作；对资源的各种操作不会改变资源标识；所有的操作都是无状态的。

41、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：采用流式布局和弹性化设计、响应式图片。

### 42、主从数据库结构特点：

一般：一主多从，也可以多主多从。

主库做写操作，从库做读操作。

主从复制步骤：

主库（Master）更新数据完成前，将操作写 binlog 日志文件。

从库（Slave）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。

从库执行中继日志事件，保持与主库一致。

#### 43、反规范化的技术手段以及优缺点

技术手段	说明
增加派生性冗余列	已有单价和数量列，增加“总价”列
增加冗余列	已有学号列，增加“姓名”列
重新组表	把拆分的表重新组表
分割表	把用户表做水平分割，长沙的用户存在长沙，上海的用户存在上海

反规范化的优点：连接操作少，检索快、统计快；需要查的表减少，检索容易。

反规范化的缺点	解决方案
数据冗余，需要更大存储空间	无解
插入、更新、删除操作开销更大	无解
数据不一致 可能产生添加、修改、删除异常	1、触发器数据同步 2、应用程序数据同步 3、物化视图
更新和插入代码更难写	无解

#### 44、视图的优点：

- (1) 视图能简化用户的操作
- (2) 视图机制可以使用户以不同的方式查询同一数据
- (3) 视图对数据库重构提供了一定程度的逻辑独立性
- (4) 视图可以对机密的数据提供安全保护

其中物化视图：将视图的内容物理存储起来，其数据随原始表变化，同步更新。

#### 45、分表和分区

	分区	分表
共性	1、都针对数据表 3、都提升了查询效率	2、都使用了分布式存储 4、都降低了数据库频繁 I/O 的压力值
差异	逻辑上还是一张表	逻辑上已是多张表。

#### 46、分区的优点：

- (1) 相对于单个文件系统或是硬盘，分区可以存储更多的数据。

- (2) 数据管理比较方便，比如要清理或废弃某年的数据，就可以直接删除该日期的分区数据即可。
- (3) 精准定位分区查询数据，不需要全表扫描查询，大大提高数据检索效率。
- (4) 可跨多个分区磁盘查询，来提高查询的吞吐量。
- (5) 在涉及聚合函数查询时，可以很容易进行数据的合并。

#### 47、关系型数据库和 NoSQL 对比

对比维度	关系数据库	NoSQL
应用领域	面向通用领域	特定应用领域
数据容量	有限数据	海量数据
数据类型	结构化数据【二维表】	非结构化数据
并发支持	支持并发、但性能低	高并发
事务支持	高事务性	弱事务性
扩展方式	向上扩展	向外扩展

#### 48、信息安全整体架构设计

WPDRRC 模型：WPDRRC 信息安全模型是我国“八六三”信息安全专家组提出的适合中国国情的信息系统安全保障体系建设模型。

WPDRRC 模型包括 6 个环节和 3 大要素。

6 个环节包括：预警、保护、检测、响应、恢复和反击。模型蕴涵的网络安全能力主要是预警能力、保护能力、检测能力、响应能力、恢复能力和反击能力。

3 大要素包括人员、策略和技术。

#### 49、区块链技术

(1) 【区块链】≠ 比特币，比特币底层采用了区块链技术。比特币交易在我国定性为【非法运用】。

(2) 区块链的特点：

去中心化：由于使用分布式核算和存储，不存在中心化的硬件或管理机构，任意节点的权利和义务都是均等的，系统中的数据块由整个系统中具有维护功能的节点来共同维护。

开放性：系统是开放的，如：区块链上的【交易信息是公开的】，不过【账户身份信息是高度加密的】。

自治性：区块链采用基于协商一致的规范和协议（比如一套公开透明的算法）使得整个系统中的所有节点能够在去信任的环境自由安全的交换数据，使得对“人”的信任改成了对机器的信任，任何人为的干预不起作用。

安全性（信息不可篡改）：数据在多个节点存储了多份，篡改数据得改掉 51%节点的数据，这太难。



同时，还有其它安全机制，如：比特币的每笔交易，都由付款人用私钥签名，证明确实是他同意向某人付款，其它人无法伪造。

匿名性（去信任）：由于节点之间的交换遵循固定的算法，其数据交互是无需信任的（区块链中的程序规则会自行判断活动是否有效），因此交易对手无需通过公开身份的方式让对方自己产生信任，对信用的累积非常有帮助。

### （3）共识算法（博弈论）/全民记账

一般有：POW（工作量证明）、PoS（权益证明）、DPoS（股份授权证明机制）

比特币采用了 POW（工作量证明）：

争夺记账权 = 挖矿

计算出来的账单节点哈希值前 13 个字符为 0，则符合规则，得到记账权。

有一个节点计算出结果，则广播消息告知其它节点，其它节点更新数据。

50、JWT（JSON Web Token）是一种用于身份验证和授权的开放标准。

它是一种轻量级的、基于 JSON 的令牌，可以在客户端和服务端之间传递信息。

JWT 由三部分组成：

【头部】包含令牌类型和所使用的算法

【载荷】包含用户信息和其他元数据

【签名】用于验证令牌的完整性和真实性

JWT 的应用场景：【信息交换】 【授权】

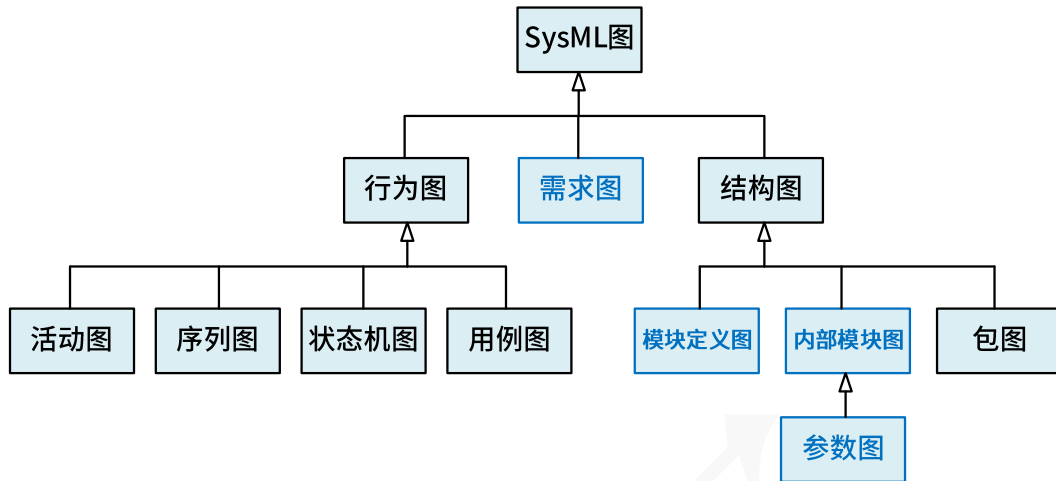
不需要服务器端存储状态，安全地传递【非敏感信息】

	JWT	Cookie	API Key
是否有状态	无	无	无
应用场景	前后端、后端服务之间	前后端	一般后端服务之间
认证对象	主要针对用户	针对用户	主要针对系统、应用
可撤销	不方便	方便	方便
生成方式	认证过程中动态生成	认证过程中动态生成	预先分配

### 51、SysML【系统建模语言】

对象管理组织 OMG 在对 UML 2.0 的子集进行重用和扩展的基础上，提出了一种新的系统建模语言 SysML（Systems Modeling Language），作为系统工程的标准建模语言。SysML 的目的是统一系统工程中使用的建模语言。





52、SysML 中的需求关系：包含、跟踪、继承需求、改善、满足、验证和复制。

包含（Include）：需求可以且只能包含其他需求。

跟踪（Trace）：对提供方元素（位于箭头端）的修改可能会导致对客户端元素（位于尾端）修改的需要。

继承（deriveReq）：一个需求可以继承另一个需求的属性。

改善（refine）：表示一个需求改进了另一个需求的满足程度。

满足（satisfy）：一般是模块满足某种需求。

验证（verify）：表示一个需求验证了另一个需求的正确性。

复制（Copy）：表示一个需求复制了另一个需求的特性。

53、MBSE 是建模方法的形式化应用，以使建模方法支持系统需求、分析、设计、验证和确认等活动，这些活动从概念性设计阶段开始，持续贯穿到设计开发以及后来的所有生命周期阶段。

54、MBSE 的三大支柱：建模语言、建模思路、建模工具。

55、软件复用是一种系统化的软件开发过程，通过识别、分析、分类、获取和修改软件实体，以便在不同软件开发过程中重复使用它们。

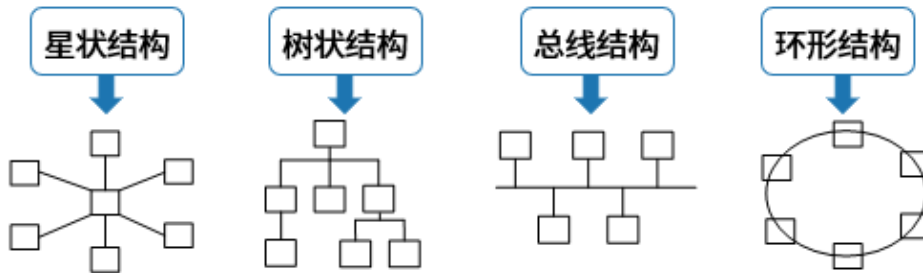
软件开发过程中重复使用相同或相似软件元素的过程。

软件架构	机会复用	开发过程中，只要发现有可复用资产，就对其进行复用
复用类型	系统复用	开发之前，要进行规划，以决定哪些需要复用

56、软件架构复用的基本过程：复用的基本过程主要包括 3 个阶段：首先构造/获取可复用的软件资产，其次管理这些资产，最后针对特定的需求，从这些资产中选择可复用的部分，以开发满足需求的应用系统。

57、局域网是指在有限地理范围内将若干计算机通过传输介质互联成的计算机组（即通信网络），通过网络软件实现计算机之间的文件管理、应用软件共享、打印机共享、工作组内的日程安排、电子邮件和传真通信服务等功能。局域网是封闭型的。

58、局域网专用性非常强，具有比较稳定和规范的拓扑结构。常见的局域网拓扑结构有星状结构、树状结构、总线结构和环形结构。



	特点	缺点
星状结构	网络中的每个结点设备都以中心结点为中心，通过连接线与中心结点相连，如果一个结点设备需要传输数据，它首先必须通过中心结点；中心结点是控制中心，任意两个结点间的通信最多只需两步；传输速度快、网络结构简单、建网容易、便于控制和管理	可靠性低，网络共享能力差，并且一旦中心结点出现故障则导致全网瘫痪。
树状结构	被称为分级的集中式网络；网络成本低，结构简单；在网络中，任意两个结点之间不产生回路，每个链路都支持双向传输，结点扩充方便、灵活，方便寻查链路路径	除叶结点及其相连的链路外，任何一个工作站或链路产生故障都会影响整个网络系统的正常运行
总线结构	网络是将各个结点设备和一根总线相连；网络中所有的结点设备都是通过总线进行信息传输的	作为数据通信必经的总线的负载能力是有限度的，这是由通信媒体本身的物理性能决定的；它的故障将影响总线上每个结点的通信
环状结构	网络中各结点通过一条首尾相连的通信链路连接起来，形成一个闭合环形结构网；网络中各结点设备的地位相同，信息按照固定方向单向流动，两个节点之间仅有一条通路，系统中无信道选择问题	网络不便于扩充，系统响应延时长，且信息传输效率相对较低，任一结点的故障将导致物理瘫痪
网状结构	任何结点彼此之间均存在一条通信链路，任何结点故障不会影响其他结点之间的通信（可靠性高）	网络布线较为繁琐，且建设成本高，控制方法复杂

## 59、嵌入式数据库分类

按照数据库存储位置的不同而进行分类是目前广泛采用的分类方法，它可以划分为三类：

基于内存方式（Main Memory Database System, MMDB）

基于文件方式（File Database, FDB）

基于网络方式（Netware Database，NDB）

60、嵌入式网络数据库主要由三部分组成：客户端、通信协议和远程服务器。客户端主要负责提供接口给嵌入式程序，通信协议负责规范客户端与远程服务器之间的通信，还需要解决多客户端的并发问题，远程服务器负责维护服务器上的数据库数据。

61、大数据特点

5 个 V：大规模（Volume）、高速度（Velocity）、多样化（Variety）、价值密度低（Value）、真实性（Veracity）

62、Kappa 架构与 Lambda 的对比

对比内容	Lambda 架构	Kappa 架构
复杂度与开发、维护成本	需要维护两套系统（引擎），复杂度高，开发、维护成本高	只需要维护一套系统（引擎），复杂度低，开发、维护成本低
计算开销	需要一直运行批处理和实时计算，计算开销大	必要时进行全量计算，计算开销相对较小
实时性	满足实时性	满足实时性
历史数据处理能力	批式全量处理，吞吐量大，历史数据处理能力强	流式全量处理，吞吐量相对较低，历史数据处理能力相对较弱
使用场景	直接支持批处理，更适合对历史数据分析查询的场景，期望尽快得到分析结果，批处理可以更直接高效地满足这些需求。	不是 Lambda 的替代架构，而是简化，Kappa 放弃了对批处理的支持，更擅长业务本身为增量数据写入场景的分析需求
选择依据	根据两种架构对比分析，将业务需求、技术要求、系统复杂度、开发维护成本和历史数据处理能力作为选择考虑因素。 计算开销虽然存在一定差别，但是相差不是很大，所以不作为考虑因素。	

63、使用许可：

按照被许可使用权的排他性强弱不同，可以将使用许可分为以下三种：

独占使用许可-仅 1 个授权对象可用，著作权人不可用

排他使用许可-仅 1 个授权对象和著作权人可用

普通使用许可-多个授权对象和著作权人可用