



NEW PLAN

软件高级架构师

• — 段 新 征 程 •





第一部分 案例分析

共 5 题



第二部分 案例分析 · 试题1 题干

【考点：软件架构设计,软件架构风格】

某园区拟建设一套智能入侵检测与告警平台，覆盖办公楼、仓储区和核心机房等重点区域。平台需要7×24小时持续接收门禁、视频、周界和日志等多源数据，对异常行为进行实时识别、分析和告警，并将结果同步给安保人员和运维人员。现整理出如下需求：

- a. 系统发生局部故障时，不应影响核心告警服务继续运行。
- b. 业务高峰期系统需要持续处理大规模实时事件流，并尽快完成分析与告警。
- c. 新增一种传感器接入协议时，应尽量减少对既有模块的修改。
- d. 系统正常运行的情况下，未授权用户越界访问，系统应在1秒内告警并记录在安全审计日志中。
- e. 主分析节点故障后，系统应自动切换到备用节点，保证业务不中断。
- f. 安全管理员需要能够追踪异常访问行为，并保留完整审计记录。
- g. 新增一种检测规则时，应尽量通过扩展方式实现，而不影响已有处理流程。
- h. 对高危入侵事件，系统应在规定时间内完成识别并推送告警结果。

平台需要对连续到达的数据依次完成采集、解析、标准化、特征提取、规则匹配和告警输出。

问题1

12分 根据题干中的 a-h 需求点，在性能、可用性、安全性、可修改性4种质量属性中进行匹配，并对需求 d 进行系统质量属性场景分析。

问题2

6分 针对该智能入侵检测与告警平台，应选择批处理风格还是管道-过滤器风格，并说明理由。

问题3

7分 如果采用问题2中选择的架构风格，请给出该系统的主要构件设计，并说明各构件的大致职责。



- 批处理序列：构件为一系列固定顺序的计算单元，多件事情同步顺序执行，构件之间只通过数据传递交互。每个处理步骤是一个独立的程序，每一步必须在其前一步结束后才能开始，数据必须是完整的，以整体的方式传递。比如批量图像处理：一次性处理大量图像，例如调整大小、添加水印或转换格式。
- 管道-过滤器：每个构件都有一组输入和输出，构件读取输入的数据流，经过内部处理，产生输出数据流。前一个构件的输出作为后一个构件的输入，前后数据流关联。过滤器就是构件，连接件就是管道。比如文本处理管道：在文本分析中，可以将文本处理划分为分词、词干提取、情感分析等多个阶段，每个阶段是一个过滤器。

早期编译器就是采用的这种架构，要一步一步处理的，均可考虑此架构风格。

二者区别：批处理风格通常一次性处理大量数据，离线执行，适用于批量处理任务。管道过滤器风格将任务分解为多个阶段，每个阶段逐个处理数据，通常是实时或近实时执行，适用于可组合和可扩展的任务。



第二部分 案例分析 · 试题1 参考答案

问题1

1. 质量属性匹配如下：

需求场景	质量属性	匹配依据
a	可用性	局部故障不影响核心服务，体现故障下继续提供服务的能力。
b	性能	高峰期持续处理实时事件流并尽快告警，体现吞吐量和响应时间要求。
c	可修改性	新增协议时尽量少改原有模块，体现变更成本低、扩展方便。
d	安全性	未授权越界访问时及时告警并记录审计日志，属于访问控制和安全审计要求。
e	可用性	主节点故障后自动切换到备用节点，体现容错和高可用能力。
f	安全性	异常访问可追踪并保留审计记录，体现安全监控与责任追溯要求。
g	可修改性	新增检测规则尽量通过扩展实现，体现对变化的适应能力强。
h	性能	高危事件需在规定时间内完成识别和推送，体现响应时间约束。

2. d 的系统质量属性场景分析如下：



第二部分 案例分析 · 试题1 参考答案（续）

要素	内容
刺激源	未授权用户
刺激	越界访问
环境	系统正常运行的情况下
制品	访问控制与安全审计机制
响应	在1秒内告警并记录在安全审计日志中
响应度量	告警和日志记录均需在1秒内完成

问题2

1. 应选择管道-过滤器风格。
2. 原因一：系统处理的是持续到达的实时事件流，不适合批量汇总后再统一处理。
3. 原因二：采集、解析、标准化、特征提取、规则匹配本身就是串行转换过程。
4. 原因三：前一环节输出可以直接作为后一环节输入，符合数据流式处理特点。
5. 原因四：该风格便于替换单个过滤器，扩展性和低耦合性更好。

问题3

1. 数据采集过滤器：接收门禁、视频、周界和日志等原始数据。
2. 协议解析过滤器：将异构协议数据解析为统一格式。
3. 数据标准化过滤器：完成字段映射、格式清洗和结构统一。
4. 特征提取过滤器：提取用于识别异常行为的关键特征。
5. 规则匹配过滤器：根据规则库识别入侵或异常事件。
6. 告警生成过滤器：生成告警信息并推送到后续模块。
7. 审计记录过滤器：记录安全日志，形成审计留痕。
8. 连接方式：各过滤器之间通过数据管道顺序连接。



第二部分 案例分析 · 试题2 题干

【考点：数据库,数据库控制功能】

某电商交易平台在大促和秒杀场景下并发量激增，系统既要保证库存扣减、余额更新和订单状态变更的一致性，又要兼顾查询吞吐量和响应时间。平台已使用关系数据库保存核心交易数据，并使用Redis承载热点缓存、计数和分布式协调能力。设计团队拟从数据库锁机制和Redis缓存两个方面进行优化，以降低竞争冲突并提升整体性能。

问题1

12分 悲观锁按行为方式通常可分为哪两种，请分别说明其含义和适用特点。

问题2

6分 比较乐观锁和悲观锁的优缺点，并说明乐观锁在应用层和DBMS层通常分别如何实现。

问题3

7分 在高并发场景下，结合Redis缓存，写出至少3种常见的优化或控制手段，并简要说明其作用。



第二部分 案例分析 · 试题2 参考答案

问题1

1. 悲观锁分类如下：

锁类型	含义	适用特点
共享锁	又称读锁，允许多个事务同时读取同一数据	适合读多写少场景
排它锁	又称写锁，一个事务获得后其他事务不能再加共享锁或排它锁	适合库存扣减、余额更新、转账等独占访问场景

问题2

1. 乐观锁与悲观锁对比如下：

锁机制	优点	缺点
乐观锁	并发度高、阻塞少、系统开销小	冲突频繁时需要不断重试，性能会下降
悲观锁	实现直接，一致性控制较强	阻塞明显，严重时可能导致死锁

2. 乐观锁实现方式如下：

1. 应用层实现：

先读取业务数据及其版本号或时间戳，再在更新时带上原版本条件，例如 `where id=? and version=?`。

若更新成功，则同时把版本号加1或更新时间戳。

若受影响行数为0，说明该数据已被其他事务修改，当前事务应重试、回滚或提示冲突。

2. DBMS层实现：

常见做法是利用数据库的乐观并发控制思想，在提交阶段校验数据版本、时间戳或读写集合是否发生变化。



第二部分 案例分析 · 试题2 参考答案（续）

若校验通过则提交，否则回滚。

一些数据库也会结合MVCC保存多版本数据，使读操作不阻塞写操作，再在更新或提交阶段判断是否存在并发冲突。

问题3

1. 使用Redis分布式锁，控制秒杀、库存扣减等临界资源访问，避免同一资源被并发重复处理。
2. 使用Redis原子计数或Lua脚本，把多步操作变成原子执行，减少并发更新冲突。
3. 对热点数据进行缓存预热、过期时间打散或多级缓存，降低数据库瞬时压力并防止缓存雪崩。
4. 将高频写请求先进入消息队列，再异步落库实现削峰填谷，避免数据库被短时流量压垮。
5. 使用布隆过滤器和空值缓存，拦截不存在的数据访问，防止缓存穿透。



第二部分 案例分析 · 试题3 题干

【考点：扩展：新技术,边缘计算】

某大型园区建设AIoT智能安防系统，用于门禁通行、访客管理、周界防护和异常事件联动处置。系统采用云边协同四层架构，希望在网络正常时实现全局智能分析，在边缘侧实现近实时响应，并支持与现有安保平台和第三方业务系统联动。

问题1

4分 分别说明感知层、边缘层、AI决策层、应用层的主要职责。

问题2

11分 系统候选构件包括：智能门禁控制器、生物识别、身份凭证、周界保护、边缘计算网关、视觉分析引擎、融合认知引擎、模型训练和OTA、事件中心、移动端应用、第三方系统集成，请将题干中的候选构件划分到四层架构中。

问题3

10分 结合云边协同特点，分析该四层架构的5个主要缺点。



第二部分 案例分析 · 试题3 参考答案

问题1

1. 感知层：负责采集门禁、视频、周界等现场数据，并完成基础身份识别与设备接入。
2. 边缘层：负责本地实时计算、视频预处理和快速联动，重点解决低时延响应与断网自治问题。
3. AI决策层：负责多源数据融合、模型训练与推送、事件分析和综合研判，形成全局智能决策能力。
4. 应用层：负责业务应用、可视化展示、告警处置和外部系统对接，向安保人员和管理人员提供服务。

问题2

1. 构件划分如下：

层级	构件
感知层	智能门禁控制器、生物识别、身份凭证、周界保护
边缘层	边缘计算网关、视觉分析引擎
AI决策层	融合认知引擎、模型训练和OTA、事件中心
应用层	移动端应用、第三方系统集成

问题3

1. 边缘设备算力有限，复杂模型难以下沉部署。
2. 模型同步和版本管理复杂，OTA更新存在一致性风险。
3. 边缘节点分散，物理与网络攻击面更大。
4. 异构边缘设备数量多，统一运维和监控难度高。
5. 数据分散在边缘与云端，隐私治理和合规管理更复杂。



第二部分 案例分析 · 试题4 题干

【考点：扩展：新技术,机器学习】

某在线学习平台建设智能辅导与学习路径推荐系统，服务对象既包括基础薄弱学员，也包括有一定基础的进阶学员。平台依据历史学习记录、阶段测评结果、知识点掌握情况和问答行为，为学员推荐后续学习路径，并在学习过程中根据答题情况和提问情况切换到智能辅导模式。回忆版题目显示该题涉及流程或状态图填空、冷启动分析，以及基础知识前置依赖对推荐准确性的影响。

为便于规范化整理，现将流程抽象为以下状态转换：

开始 -- 空1 --> 学习进行中

学习进行中 -- 空2 --> 智能辅导中

智能辅导中 -- 空3 --> 学习进行中

学习进行中 -- 空4 --> 学习完成

智能辅导中 -- 空5 --> 学习完成

开始 -- 空6 --> 智能辅导中

问题1

12分 【缺图】根据上述流程，补充空1至空6最合理的事件或动作名称。

问题2

6分 题干提到新用户推荐正确率不高是因为用户冷启动。请说明用户冷启动的原因，并写出另外两种常见冷启动类型。

问题3

7分 系统对中上水平学员推荐较准确，但对基础薄弱学员推荐偏难。请从路径推理和训练数据两个方面，分析为什么学习路径推荐必须重视基础知识前置依赖梳理。



第二部分 案例分析 · 试题4 参考答案

问题1

1. 一种合理补法如下：

空位	参考答案
空1	开始学习或进入学习
空2	触发答疑或请求辅导
空3	辅导结束返回学习
空4	完成当前学习路径
空5	辅导后完成学习
空6	直接进入智能辅导

问题2

1. 用户冷启动：新用户缺乏学习历史、兴趣偏好、能力水平和行为反馈，系统无法建立稳定画像。
2. 直接影响：推荐模型难以准确判断其当前水平、兴趣方向和适合的学习节奏，因此推荐准确率偏低。

另外两种常见冷启动如下：

1. 物品冷启动：新课程、新题目或新知识点缺少交互数据，系统难以判断其适合推荐给哪些用户。
2. 系统冷启动：平台刚上线时，用户侧和资源侧都缺乏足够数据，模型整体训练基础不足。



第二部分 案例分析 · 试题4 参考答案（续）



知乎知学堂

答案与解析

问题3

1. 从路径推理看：知识学习存在明显前置依赖，后续知识点往往建立在前置知识掌握的基础上。
2. 如果系统未识别基础知识缺口，就可能直接推荐高阶内容，导致学习断层、连续受挫和路径失真。
3. 从训练数据看：如果训练样本主要来自中上水平学员，模型会偏向较快、较跳跃的学习路径。
4. 这样会忽视基础薄弱学员“先补基础、再进阶”的渐进式学习规律，导致推荐难度偏高。
5. 因此需要显式梳理知识图谱、先修关系和能力分层，使推荐既参考相似行为，也校验是否具备进入下一阶段的前置条件。



第二部分 案例分析 · 试题5 题干

【考点：软件架构设计,分布式系统】

某社区建设智能老人居家健康监测系统，需面向老人家庭、社区服务中心和医护人员提供协同服务。系统支持环境监测、老人跌倒识别、一键求助、机器人送药送饭、用电安全告警和设备状态上报等功能，并要求在边缘网关侧具备一定本地处理能力。系统采用分层架构，系统中的消息既包含周期性环境数据，也包含与老人生命安全相关的紧急告警信息。

问题1

12分 【缺图】请将候选技术：FreeRTOS、MQTT、Kafka、TDengine、Flask、Vue.js、MySQL、ROS，与典型功能对应起来，补全设备端OS、通信协议、消息队列、边缘时序存储、后端框架、前端框架、持久化数据库、机器人控制这8项技术选型。

问题2

6分 解释什么是时间序列数据，并说明边缘网关层为什么适合采用时序数据库存储。

问题3

7分 结合以下6类消息，判断更适合采用MQTT的QoS 0、QoS 1还是QoS 2，并说明涉及生命安全场景为什么优先采用QoS 2。

1. 温湿度等周期性环境数据上报
2. 水表、电表等定时抄表数据上报
3. 设备在线状态上报
4. 一般用电异常告警
5. 跌倒检测告警
6. 一键紧急求助



第二部分 案例分析 · 试题5 参考答案

问题1

1. 技术选型对应如下：

层级或功能	技术选型
设备端OS	FreeRTOS
通信协议	MQTT
消息队列	Kafka
边缘时序存储	TDengine
后端框架	Flask
前端框架	Vue.js
持久化数据库	MySQL
机器人控制	ROS

问题2

1. 时间序列数据：按时间顺序记录、带有时间戳的数据点序列。
2. 典型数据：温度、湿度、心率、设备状态等连续变化数据。
3. 采用时序数据库的原因一：写入性能高，适合传感器高频采集和持续上报。
4. 原因二：压缩率高，节省边缘网关有限的本地存储空间。
5. 原因三：支持时间窗口聚合查询，便于统计某一时间段内的平均值、最大值和趋势变化。
6. 原因四：支持降采样和过期清理，适合长期追加写入、很少随机更新的时序数据场景。



第二部分 案例分析 · 试题5 参考答案（续）

问题3

1. QoS映射如下：

消息类型	QoS等级	说明
1. 温湿度等周期性环境数据上报	QoS 0	周期性数据，偶发丢失影响较小
2. 水表、电表等定时抄表数据上报	QoS 0	定时上报数据，可容忍少量丢失
3. 设备在线状态上报	QoS 1	需要保证送达，但允许极少量重复
4. 一般用电异常告警	QoS 1	告警需要可靠送达，但可接受少量重复
5. 跌倒检测告警	QoS 2	生命安全类消息，既不能丢失也不应重复处理
6. 一键紧急求助	QoS 2	生命安全类消息，要求恰好一次

2. 采用QoS 2的原因如下：

1. 跌倒检测和一键求助属于生命安全类消息。
2. QoS 2可以保证既不丢失，也尽量避免重复处理。



第二部分 案例分析模拟题

共 8 题



第二部分 案例分析模拟题 · 试题1 题干

题目



知乎知学堂

【考点：软件架构设计,软件架构质量属性】

【案例背景】

某市拟建设“智慧养老 IoT 平台”，为辖区内 3 万名独居老人提供居家安全监护服务。系统部署于市政政务云，通过老人家中安装的各类传感器（人体红外、门磁、烟感、燃气探测器、智能床垫）采集数据，经家庭网关汇聚后上传。平台需提供跌倒自动告警、异常行为识别（如长时间无活动）、燃气泄漏联动关阀、健康数据统计与家属 App 推送等功能。

项目启动会上，各方诉求如下：

- ① 民政局：系统必须在老人发生意外时可靠告警，全年不可用时间不得超过 5 小时。
- ② 燃气公司：燃气泄漏告警必须在 3 秒内推送到值班人员手机，误报率要低。
- ③ 市大数据局：所有采集的老人健康数据属于敏感个人信息，须满足《个人信息保护法》要求。
- ④ 承建方架构组：平台未来 3 年要逐步接入更多品类设备（跌倒雷达、毫米波），且后续可能扩展智慧社区、居家医疗等新业务。

经过两轮讨论，架构师张工提出平台采用微服务 + 事件驱动架构，设备数据经 MQTT 网关进入 Kafka，由规则引擎实时处理后分发给告警服务与统计服务。



问题1（8 分）

请补充完整质量属性效用树。结合上述场景，在答题纸上填写下表（1）～（6）处内容。

质量属性	场景精化	刺激源	刺激	响应	响应度量
可用性	网络中断时的告警保障	(1)	家庭网关与平台间网络中断	本地缓存并自动重传	(2)
性能	燃气泄漏实时告警	燃气探测器	检测到燃气浓度超标	(3)	(4)
(5)	新设备品类接入	承建方	新增品类设备接入	不影响现有服务，插件化扩展	(6)

问题2（9 分）

请判断：对上述系统而言，下列三条需求分别对应哪种质量属性？（1）（2）（3）

（1）“全年不可用时间不得超过 5 小时” （2）“燃气泄漏告警必须在 3 秒内推送” （3）“健康数据存储与传输必须加密，且用户可撤回授权”

并请进一步指出：（4）为保证可用性采用的“双机房热备 + 本地网关缓存重传”分别属于哪一类可用性战术？（5）上述三条需求之间是否存在哪些质量属性冲突？请举出两对并说明取舍思路。

问题3（8 分）

架构组拟采用 ATAM 对该架构进行评估。

- （1）请写出 ATAM 的四个阶段名称。（4 分）
- （2）请说明什么是“敏感点”“权衡点”“风险点”，并结合本系统各举一例。（4 分）



问题1（8 分）

1. 补充结果如下：

空	答案	说明
(1)	网络运营商 / 网络设备（或“外部环境”）	刺激源是发出刺激的外部实体，此处为家庭网关所在的接入网络
(2)	告警数据零丢失，网络恢复后 5 分钟内全部补传完成	响应度量须可量化，“零丢失 + 时间上限”是典型写法
(3)	3 秒内生成告警并经网关推送至值班人员终端	响应是系统实际采取的动作
(4)	端到端时延 $\leq 3s$ ，误报率 $\leq 0.1\%$	性能类通常用时延 / 吞吐量 / 抖动三项度量
(5)	可修改性（或可扩展性）	“新设备接入不影响现有服务”属典型可修改性场景
(6)	新增设备品类的工作量 ≤ 5 人日，且无需重启现有服务	可修改性度量常用“修改成本（人日）+ 影响范围”

问题2（9 分）

- (1) 可用性（Availability）——“全年不可用时间”直接对应 SLA 可用性指标。5 小时 / 8760 小时 $\approx 99.94\%$ （约三个九半）。
- (2) 性能（Performance）——“3 秒内推送”是时延约束，属性能质量属性。
- (3) 安全性（Security）——涉及数据加密与用户授权撤回，属机密性与隐私合规范畴。
- (4) 战术归属：双机房热备 $\rightarrow$ 可用性战术中的故障恢复类 · 主动冗余（Active Redundancy）；本地网关缓存重传 $\rightarrow$ 故障恢复类中的缓存补传。

系统架构评估方法



289-304页详细描述了ATAM的阶段过程

架构权衡分析法ATAM，是一种系统架构评估方法，主要在系统开发之前，针对性能、可用性、安全性和可修改性等质量属性进行评价和折中，让架构师明确如何权衡多个质量目标，参与者有评估小组、项目决策者和其他项目相关人。

ATAM被分为四个主要的阶段,分别是场景和需求收集、体系结构视图和场景实现、属性模型构造和分析以及架构评审与折中。整个评估过程强调以属性(质量属性)作为架构评估的核心概念。

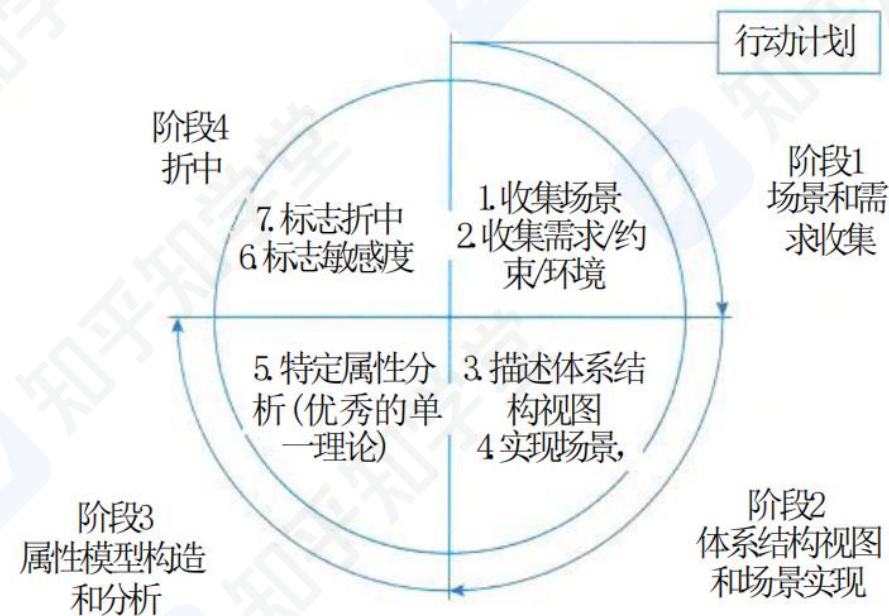


图8-2 ATAM分析评估过程



问题2（9分）（续）

（5）质量属性冲突与取舍（任答两对）：

- ① 性能 vs 安全性：端到端加密（SM4 / TLS）会增加计算与时延，与 3 秒告警目标冲突。取舍思路：告警控制面走轻量化认证 + 短包加密，健康明细这类大体量数据走异步加密通道，把安全开销从关键路径剥离。
- ② 可用性 vs 安全性：为满足合规需做细粒度鉴权与审计，一旦认证中心故障则整体不可用。取舍思路：认证服务本地化缓存 + 降级放行（仅放行告警类只读操作），牺牲部分即时管控能力保住核心可用。
- ③ 可修改性 vs 性能：插件化设备适配层带来抽象开销，与实时性目标冲突。取舍思路：在 hot path（告警链路）做编译期绑定或专用通道，在 cold path（统计、接入）保留插件化。

问题3（8分）

（1）ATAM 四个阶段：① 描述与介绍阶段 ② 调查与分析阶段 ③ 测试阶段 ④ 报告阶段。

（2）三个概念及本系统示例：

概念	定义	本系统示例
敏感点	一个或多个构件（及构件关系）的特性，对实现某个特定质量属性的响应有重大影响，是达成该属性的关键位置	Kafka 集群的分区数与副本因子配置，直接决定告警消息的处理吞吐与丢失率
权衡点	同时影响多个质量属性、且至少对一个属性有利而对另一个不利的架构决策	端到端启用国密加密：提升安全性，但增加时延、损害性能
风险点	已有的、可能带来负面后果的架构决策	依赖单一云厂商的消息中间件且不设降级方案，云侧故障将造成大面积告警失效



第二部分 案例分析模拟题 · 试题2 题干

【考点：扩展：新技术,边缘计算】

【案例背景】

某大型钢铁企业拟建设设备预测性维护平台，覆盖烧结、炼铁、炼钢、轧钢四大工序的关键旋转设备（风机、泵、轧机主传动、齿轮箱）。现状是每台设备加装振动、温度、油液多参数传感器，采样频率最高 10 kHz，全厂测点总量约 12 万个。现有传输方案为全量上传至中心云做分析，导致：

- ① 现场工业环网带宽长期占用超过 85%，影响生产控制系统通信；
- ② 云侧存储成本年增额巨大；
- ③ 设备故障判断时延普遍在分钟级，曾在某次轧机轴承早期故障中因漏报造成非计划停机。

架构师李工提出改用 AIoT 边云协同架构：在各产线部署边缘计算节点，就地完成特征提取与初步诊断，仅将特征数据与疑似异常片段上传云端；云端负责跨产线模型训练、全局数据治理与模型下发。

问题1（8 分）

请说明边缘计算的概念（2 分），并结合本案例指出引入边缘层带来的 3 项以上核心价值（6 分）。



第二部分 案例分析模拟题 · 试题2 题干（续）

问题2（9 分）

请设计该平台的边云协同数据流。

- （1）请说明边缘层应完成哪些处理环节，云端应完成哪些处理环节。（6 分，各列 3 项以上）
- （2）设备振动信号采样率高达 10 kHz，数据量巨大。请从数据存储选型角度说明：为何该类数据更适合采用时序数据库（TSDB）而非关系型数据库？请列出 3 点以上理由。（3 分）

问题3（8 分）

模型需要与云端保持同步，且边缘侧算力受限。

- （1）请说明什么是模型量化（Quantization），并说明它为何适合本边缘推理场景。（4 分）
- （2）边缘节点分布于多个产线，网络条件参差不齐。请列举 3 项保障“模型可靠下发与版本一致”的措施。（4 分）



问题1（8分）

边缘计算概念：在靠近数据源或用户的网络边缘侧，融合网络、计算、存储、应用核心能力的开放平台。它将原本集中在云端的计算任务下沉到距离数据产生地更近的位置执行，就近提供实时处理与本地自治服务，从而缓解带宽压力、降低时延、减少数据外传。

本案例核心价值（答满 3 项即可）：

- ① 带宽大幅减压：10 kHz 振动原始波形在边缘完成 FFT / 包络解调后，仅上传特征值（如 RMS、峭度、故障特征频率幅值），数据量可压缩 1 ~ 2 个数量级，工业环网占用从 85% 降至可控水平，恢复生产控制网络的通信余量。
- ② 诊断时延显著缩短：本地推理无需往返云端，从分钟级降至毫秒 ~ 秒级，使早期故障能够在恶化前被捕获，直接服务于“避免非计划停机”这一核心业务目标。
- ③ 断网自治保障连续性：轧钢等场景对连续性要求极高，边缘节点在云端链路中断时仍可持续监测与本地告警，不会因网络问题产生安全盲区。
- ④ 降低云端存储与算力成本：仅上传特征与疑似异常片段，云端存储年增量大幅下降。
- ⑤ 数据合规与安全：原始工艺数据不出厂区，减少敏感生产数据外泄面，符合数据安全分级要求。



问题2（9 分）

（1）边云协同的数据流分工：

边缘层应完成的处理环节：

- ① 多源传感数据接入与协议转换（Modbus / OPC UA / MQTT）；
- ② 数据清洗、降噪、异常值剔除；
- ③ 信号特征提取（时域 / 频域：RMS、峭度、包络谱、阶次分析）；
- ④ 本地轻量模型实时推理；
- ⑤ 产生告警与本地联动（降速 / 停机建议）；
- ⑥ 数据本地缓存与断点续传；
- ⑦ 仅上传特征值、疑似异常片段与设备状态摘要。

云端应完成的处理环节：

- ① 汇聚全厂多产线历史数据；② 模型离线 / 增量训练与版本管理；
- ③ 跨设备、跨产线的横向对比与根因分析；④ 全局数据治理（元数据、数据质量、血缘）；
- ⑤ 模型下发与边缘节点生命周期管理；⑥ 可视化、报表与统计分析；
- ⑦ 长期趋势预测与备件计划支持。



第二部分 案例分析模拟题 · 试题2 参考答案（续）

（2）时序数据库（TSDB）更适合的理由（答满 3 点）：

- ① 高频写入性能：10 kHz × 12 万测点的持续写入远超关系库的 B+ 树索引写入能力；TSDB 采用 LSM-Tree 类结构 + 追加写，吞吐高一个数量级。
 - ② 高压缩比：时序数据相邻点变化平缓，TSDB 支持差值 / 游程 / Gorilla 等专用压缩算法，压缩比通常 5 ~ 20 倍，直接解决存储成本问题。
 - ③ 按时间窗口的原生能力：故障回溯、趋势分析本质都是“某设备近 N 小时”的时间窗口查询，TSDB 有时间分区与专用聚合算子（降采样、滑动窗口），关系库要靠复杂 SQL 与分区表硬扛。
 - ④ 自动降采样与过期清理：TSDB 内置数据保留策略与连续查询，高频原始数据保留 30 天、降采样数据保留数年，运维成本低。
 - ⑤ 数据特性匹配：振动数据以追加写为主、几乎不更新删除，且总是按“设备 + 时间”查询，与 TSDB 的数据模型天然吻合。
- 可引用产品：InfluxDB、TDengine、IoTDB（国产，Apache 顶级项目，信创场景优先）。



问题3（8分）

（1）模型量化（Quantization）：指将模型的权重与激活值从高精度浮点（FP32 / FP16）转换为低精度整数表示（INT8 / INT4）的技术，同时辅以校准确定缩放因子与零点，必要时结合量化感知训练（QAT）抑制精度损失。

适配边缘推理场景的理由：边缘节点算力与显存受限；INT8 量化可使模型体积与内存占用降至原来的 1/2（INT4 为 1/4），同时整型运算在多数边缘芯片上有专门加速单元，推理延迟显著下降；设备故障检测是判定类任务，对微量精度损失容忍度较高，属“用少量精度换部署可行性”的典型场景。

（2）保障“模型可靠下发与版本一致”的措施（答满 3 项）：

- ① 版本号 + 哈希校验：每个模型包携带语义化版本号与 SHA-256 校验值，边缘节点下载后校验通过才允许切换，杜绝传输损坏。
- ② 灰度发布与分批下发：先向 1 ~ 2 个试点产线下发新模型，观察误报率指标达标后再全量推广，避免一次故障模型影响全厂。
- ③ 双版本并存与一键回滚：边缘本地保留上一稳定版本，新模型连续触发告警异常阈值时自动回滚并记录事件。
- ④ 断点续传 + 离线预置：弱网环境通过 MQTT / 对象存储的断点续传完成下载；新建站点可通过离线介质预置基础版本。
- ⑤ 下发状态统一管理：云端维护每个节点的模型版本视图，定时心跳上报版本，版本漂移自动告警并触发重新同步。



【考点：数据库,数据库控制功能】

【案例背景】

某电商平台的大促秒杀场景，单场活动峰值 QPS 约 50 万，热点商品 SKU 约 200 个。现有架构在大促期间出现以下问题：

- 现象 A：活动开始瞬间数据库 CPU 打满，大量请求超时；
- 现象 B：某次活动后出现“商品库存已扣成负数”的超卖问题；
- 现象 C：有运营反馈个别冷门商品 ID 无论如何查询都很慢，监控显示每次请求都打到数据库，Redis 命中率并未下降。

架构组拟从流量分层拦截与缓存加固两个方向进行优化。

问题1（8 分）

针对上述现象 C，这种现象称为缓存（1）。请说明其产生原因（2 分）、危害（2 分），并给出至少两种解决方案（4 分）。

问题2（9 分）

针对缓存架构，请完成下表（在答题纸填写（1）~（6））：

问题类型	定义	典型成因	常用解决方案
缓存穿透	(1)	查询本就不存在的数据	(2)
缓存击穿	(3)	热点 key 过期瞬间大量并发穿透	(4)
缓存雪崩	(5)	大批 key 在同一时刻集体过期	(6)

问题3（8 分）

- (1) 请分别说明悲观锁与乐观锁的适用场景、优点与缺点。（4 分）
- (2) 李工提出“不直接操作数据库库存，而在 Redis 中用 Lua 脚本原子扣减库存，再异步同步到数据库”。请分析该方案在保证不超卖方面的原理（2 分），并指出仍需注意的 3 个技术问题。（2 分）



问题1（8分）

（1）该现象称为缓存穿透（Cache Penetration）。

产生原因：查询的数据在数据库中本身就不存在（如被恶意构造的非法商品 ID，或已下架删除的商品）。由于数据库查询返回空，缓存层不会写入结果，导致每次同样的请求都会绕过缓存直达数据库。这也是题干中“Redis 命中率并未下降”却依然很慢的原因——这些 key 从未进过缓存，自然不影响命中率统计。

危害：缓存层完全失效，全部压力转嫁给数据库；若被攻击者利用，可用大量随机不存在的 key 发起请求，形成低成本的 DoS 攻击，拖垮整个系统。

解决方案（任选两种）：

- ① 缓存空值：数据库查不到时，仍向 Redis 写入空值或特殊标记并设置较短过期时间（如 60 秒），后续同 key 请求在缓存层被拦截。缺点是占用一定内存，可通过独立的短 TTL 命名空间缓解。
- ② 布隆过滤器（Bloom Filter）：系统启动时将所有合法商品 ID 加入布隆过滤器，请求到达时先经过滤器判断，不存在则直接返回。优点是内存占用极小、查询 O(1)；缺点是存在误判率（只会误判“存在”，不会误判“不存在”），且数据删除后需重建或使用计数布隆过滤器。
- ③ 接口层校验与限流：对 ID 做格式 / 范围合法性校验，对同一 IP 或用户的异常高频请求限流熔断，从入口拦截恶意流量。
- ④ 热点参数风控：结合风控系统识别异常访问模式并降级拦截。

问题	定义	产生原因	解决方案
缓存穿透	查询不存在的数据，请求直达数据库	恶意攻击、业务误删、数据不存在	1. 缓存空值（设置短过期时间） 2. 布隆过滤器（Bloom Filter）拦截
缓存击穿	热点key过期，大量请求打到数据库	高并发+热点key同时过期	1. 热点key永不过期（逻辑过期） 2. 分布式锁互斥重建缓存
缓存雪崩	大量key同时过期，数据库压力激增	批量设置相同过期时间、Redis宕机	1. 过期时间加随机值 2. 多级缓存架构 3. 熔断降级（Hystrix/Sentinel） 4. 集群高可用部署



问题2（9 分）

(1) ~ (6) 填写结果如下：

空	答案
(1)	查询一个根本不存在的数据库，缓存与数据库都不会命中，请求每次都直达数据库
(2)	缓存空值 + 短 TTL；布隆过滤器拦截；接口层参数校验与限流
(3)	某个热点 key 在过期瞬间，同时有大量并发请求到达，全部穿透到数据库重建缓存
(4)	热点 key 永不过期（逻辑过期由业务控制）；互斥锁 / 单飞重建（只有一个线程回源，其余等待）；缓存预热 + 错峰过期
(5)	大量 key 在同一时刻集中失效，或 Redis 集群整体宕机，导致瞬时流量全部涌向数据库
(6)	过期时间加随机抖动（基础 TTL + 随机值），打散失效时刻；多级缓存（本地缓存 + 分布式缓存）；Redis 高可用集群 + 熔断降级；缓存预热

记忆口诀：穿透 = 查不存在的；击穿 = 一个热点过期；雪崩 = 一批同时消失。

问题3（8 分）

(1) 悲观锁 vs 乐观锁：

维度	悲观锁	乐观锁
基本假设	并发冲突经常发生，先加锁再操作	并发冲突很少发生，提交时再校验
典型实现	SELECT ... FOR UPDATE；共享锁 / 排他锁	版本号 version 字段；时间戳；CAS；DBMS 快照隔离
优点	实现简单直观；强一致保证；无脏读	读操作不阻塞，高并发吞吐高；无死锁
缺点	持锁期间阻塞其他事务；高并发下锁竞争激烈、吞吐低；可能死锁	冲突频繁时大量重试浪费资源；需业务配合加版本字段；不适合高冲突场景
适用场景	写多、冲突高、要求强一致的场景	读多写少、冲突低的场景



第二部分 案例分析模拟题 · 试题3 参考答案（续）

问题3（8分）（续）

（2）Redis + Lua 方案保证不超卖的原理：Lua 脚本在 Redis 中以单线程原子方式执行，脚本内的“读库存 → 判断大于 0 → 扣减 → 记录订单”是不可分割的整体，不会被并发请求打断。Redis 的单线程模型天然保证同一时刻只有一个脚本在操作该 key，因此不存在“读到的库存已被他人扣减”的竞态窗口。

仍需注意的技术问题（任答 3 个）：

- ① Redis 与数据库的一致性：Redis 扣减成功后若异步回写数据库失败，会造成库存数据丢失。需引入可靠消息队列 + 对账补偿机制，并要有最终一致性校验任务。
- ② Redis 自身的高可用与持久化：Redis 若故障且未持久化，已扣减的库存会丢失。须启用 AOF + 主从 / 集群，并明确 RPO 目标（异步复制下主库宕机会丢失未同步数据）。
- ③ Lua 脚本的执行性能与阻塞：脚本应尽量精简，避免长时间运行阻塞 Redis 单线程；禁止在脚本内做网络调用或慢查询。
- ④ 库存回滚与防重复提交：用户未支付需回滚库存，要保证幂等（同一订单只能回滚一次），防止重复退库存造成超卖。
- ⑤ 限流兜底：需提前定义 Redis 不可用时的降级方案——是直接拒绝还是走数据库兜底。



第二部分 案例分析模拟题 · 试题4 题干

【考点：企业信息化,数据治理】

【案例背景】

某集团型企业信息化 20 余年，建设了 ERP、MES、SRM、WMS、CRM、OA 等 20 余套异构系统，数据总量约 400 TB。当前存在以下突出问题：

- ① 口径不一：同一“客户”在 CRM 与 ERP 中编码规则不同，“销售额”在财务口径与销售口径计算结果相差 8%；
- ② 孤岛严重：跨系统取数需人工导出 Excel 拼接，月度经营分析报告需 5 人天；
- ③ 质量失控：某次 MES 停机时长字段因单位不统一（分钟 vs 小时），导致全厂 OEE 计算结果失真且未被及时发现；
- ④ 合规压力：集团拟推动数据资产入表，但对全集团敏感数据的分布位置缺乏掌握。

集团决定建设统一的数据治理平台。

问题1（10 分）

- （1）请列举并简要说明数据质量的六个维度（评估维度）。（6 分）
- （2）请说明什么是主数据 MDM，并简述主数据管理与传统业务数据的三点区别。（4 分）

问题2（9 分）

- （1）请给出数据治理平台从数据源到数据服务的分层架构，列出各层名称与职责。（5 分）
- （2）针对上述“停机时长单位不统一导致 OEE 失真”的问题，说明数据治理平台应在哪些环节设置管控措施。（4 分）

问题3（6 分）

- （1）请说明数据网格（Data Mesh）的核心原则。（4 分）
- （2）请结合本集团现状（强管控、多异构系统、缺乏数据人才），判断当前阶段是否适合直接采用数据网格，并说明理由。（2 分）



问题1（10 分）

（1）数据质量六个维度：

维度	说明	本案例对应问题
准确性	数据能否真实反映客观实体的状态	停机时长单位错误导致数据不真实
完整性	关键字段是否有缺失、记录是否齐全	部分系统字段长期无人填报
一致性	同一数据在不同系统间是否保持统一	CRM 与 ERP 客户编码口径不一
及时性	数据是否在需要时可用、更新是否跟上业务节奏	月度报告需 5 人天手工拼接
唯一性	同一实体是否存在重复记录与冲突表示	同一客户多条重复档案
有效性（合乎性）	数据是否符合预定义的业务规则、格式与值域	单位未按标准字典取值

（2）主数据 MDM：主数据是指在企业内被多个业务系统共享使用、描述业务核心实体的高价值、相对稳定的数据，如客户、供应商、物料、设备、组织、会计科目等。主数据管理是一套制度、流程与技术的组合，用于确保在全域范围内主数据的唯一性、一致性与权威性。

与业务数据（交易数据）的区别：

对比维度	主数据	业务 / 交易数据
变化频率	低，相对稳定	高，随每笔业务发生而不断产生
数据量	小（通常万 ~ 百万级）	大（持续累积，可达 TB / PB 级）
生命周期	长期存在，需持续管理	一旦交易完成基本固化
共享范围	跨多个业务系统共享，是全集团的“公共语言”	通常限于单一业务系统内部
管理重点	唯一性、一致性、权威性，需建立分发机制	完整性、及时性，关注采集与处理效率



问题2（9 分）

（1）数据治理平台分层架构（自下而上）：

层级	名称	职责
L1	数据源层	ERP / MES / SRM 等异构业务系统、外部数据与 IoT 设备数据
L2	数据采集层	CDC 增量捕获、批量抽取、日志采集、API 接入；支持实时与离线双通道；协议转换与初筛
L3	数据存储计算层	数据湖（原始层 ODS）+ 数据仓库（明细 DWD、汇总 DWS、应用层 ADS）；湖仓一体架构，存算分离
L4	数据治理层	元数据管理、数据标准管理、主数据管理、数据质量管理规则引擎、数据安全治理（分类分级、脱敏、审计）、数据血缘分析、数据生命周期管理
L5	数据服务层	统一数据服务 API、指标平台、标签体系、即席查询、数据可视化与报表、数据开放与共享交换
L6	应用与运营层	经营分析、风控、数据资产目录、数据资产评估与入表支撑
贯穿	组织与制度保障	数据治理委员会、数据 Owner 制、考核机制、数据标准规范体系

（2）针对“停机时长单位不统一”应在五个环节设置管控：

- ① 标准落地（事前）：建立企业级数据标准字典，明确“停机时长”的标准单位（分钟）、精度与取值范围，所有系统在数据字典层面与该标准对齐。
- ② 元数据注册：新系统接入时强制注册字段单位、业务含义、责任人，未通过元数据采集审核不得入湖。
- ③ 数据质量规则（事中）：在数据集成环节部署质量规则引擎，配置值域校验、单位校验、量纲校验、逻辑一致性校验（如“停机时长不得大于计划工作时长”）；异常数据自动打标并阻断或隔离到异常区，同时触发告警。
- ④ 血缘与影响分析（事后）：通过血缘追踪“停机时长”字段从 MES 原始表到 OEE 指标的完整链路，一旦发现质量事故可快速定位受影响的下游指标与报表范围。
- ⑤ 质量闭环：质量问题自动生成工单派发给数据 Owner，修复后回填结果与根因分析，形成“发现一定位一修复一复盘”的闭环，并沉淀为新的质量规则。



问题3（6分）

（1）数据网格（Data Mesh）四条核心原则：

- ① 领域所有权（Domain Ownership）：数据按业务领域划分，由最懂该领域的团队拥有并对数据质量与可用性负责——谁生产谁负责，去中心化的组织结构。
- ② 数据即产品（Data as a Product）：每个领域把自己的数据作为产品对外提供，须具备可发现、可寻址、可理解、可信赖、自描述语义、安全合规等产品质量特征，并有明确的产品负责人与服务等级承诺。
- ③ 自助式数据平台（Self-serve Data Platform）：由平台团队提供统一的基础设施与工具（存储、计算、流水线、治理、监控），使领域团队无需高门槛专家即可自助发布和使用数据产品，降低认知与运维负担。
- ④ 联邦式计算治理（Federated Computational Governance）：治理策略由领域代表与平台团队共同制定，通过自动化、代码化的策略执行（而非人工审批）在全网统一落地，兼顾自治与全局一致性和合规。

（2）当前阶段不适合直接采用数据网格。

理由：数据网格要求各业务领域都有具备数据工程能力的团队与明确的领域 Owner，属分布式去中心化、成熟度要求较高的治理模式。该集团当前的痛点是标准不统一、孤岛严重、数据人才缺乏，处于从“无序”走向“规范化”的早期阶段。正确路径是先集权：建立统一的数据标准、主数据管理与集中式治理平台，把数据家底摸清、口径统一；待组织认知、人才储备与平台能力成熟后，再逐步向“联邦式”演进。跳过集中治理直接做数据网格，极易演变为新的孤岛叠孤岛。



第二部分 案例分析模拟题 · 试题5 题干

【考点：数据库,数据库设计与优化】

【案例背景】

某跨境电商平台订单库已达单表 8 亿行、日增 600 万行，出现写入锁竞争激烈、大促高峰期数据库 CPU 飙高、历史订单查询超时等问题。业务特征：订单按用户维度访问频繁，商家要求能按订单号 / 时间 / 状态组合查询，运营需要月度、年度销售统计报表；下单强一致、数据不能丢。DBA 与架构团队计划引入分库分表，并配套分布式 ID 与订阅同步等设施。

问题1（7 分）

请依据业务特征选择分库分表的分片键，说明选择依据，并分析“按用户 ID 分片”对商家侧查询带来的问题及应对方案。

问题2（6 分）

分库分表后需要全局唯一 ID。请对比“数据库号段方案”“雪花算法”“UUID”三种方案的优缺点，并给出适用于本场景的选择。

问题3（6 分）

请就“订单全链路查询能力”给出跨分片查询（跨库聚合、排序分页、按订单号直查）的架构方案，避免全分片广播。

问题4（6 分）

为支持月度 / 年度报表，请设计“实库 + 归档库 + 汇总层”的分层数据架构（说明热数据、温数据、统计数据各自的存放方式与调度方式）。



问题1（7 分）

分片键选择用户 ID（buyer_id）。

选择依据：订单的高频访问沿用户维度展开（“我的订单”），按用户 ID 分片可把同一用户的订单聚拢到同一分片，点查命中率高、分片局部性好。

带来的问题：商家按“订单号 / 时间 / 状态”组合查询会跨全部分片，导致全分片广播与深度分页。

应对方案：① 订单号中嵌入分片路由信息，或维护“订单号 → 用户 ID”的索引 / 映射表；② 按商家维度另建宽表或汇总明细（读写分离），商家查询走专用搜索索引（如 ES）而非订单实表。

评分要点：分片键的选择依据、商家侧查询问题与应对方案，三方面均须覆盖。

问题2（6 分）

方案	优点	缺点
数据库号段	批量取号段，性能高；趋势递增，便于索引	依赖发号中心；多实例需协调，存在号段隐患
雪花算法	时间戳 + 机器位 + 序列；趋势递增、无中心依赖、性能好	需处理时钟回拨与机器位收敛
UUID	完全本地生成、无中心依赖	无序，主键索引性能差；长度偏长

本场景选择：雪花算法（可加业务前缀以兼容订单号路由）。

评分要点：三种方案的优缺点对比 + 本场景选型理由。



第二部分 案例分析模拟题 · 试题5 参考答案（续）

问题3（6分）

- ① 按订单号查询：订单号携带分片信息（前几位指向分片号），直连对应分片。
- ② 按用户查询：以用户 ID 为分片键，天然命中单一分片。
- ③ 商家组合查询 / 全维度检索：旁路构建 ES 或宽表搜索引擎，数据通过订阅 binlog 同步；排序分页在搜索层完成并做二次聚合。
- ④ 确有跨分片聚合时：采用“分片并行查询 + 网关汇总 / 归并排序 + 游标式翻页”，控制深度分页开销。

评分要点：四条查询路径各一点，逻辑清晰即可。

问题4（6分）

- ① 热数据（近 3 个月订单）：保留在在线分库分表库，支撑高频读写。
- ② 温数据（3 ~ 24 个月）：定期归档到独立归档库 / 冷存储，支持低频按需查询。
- ③ 统计层：每日 / 每月定时任务（离线数仓或 OLAP 引擎）从实库抽取到汇总表或数仓（如 StarRocks、ClickHouse），支撑运营报表
- ④ 归档调度：按时间分区 + 定时任务（凌晨低峰）批量迁移，同步下线索引以降低存储开销。

评分要点：分层划分合理 + 调度方式说明清楚。



第二部分 案例分析模拟题 · 试题6 题干

【考点：软件架构设计,分布式系统】

【案例背景】

某银行互联网信贷平台将核心拆分为 7 个微服务（账户、授信、放款、还款、风控、通知、征信报送），一次“发放贷款”需同时：账户服务扣减贷款专户、放款服务生成借据、风控服务记录决策快照；一次“还款”需记录还款流水、更新借据余额、调用征信报送服务。业务要求：资金类数据强一致不可错账，征信报送允许异步但最终一致且不可漏报。原单库两阶段提交无法支撑跨服务，团队需在 XA、TCC、Saga、本地消息表 + MQ 中做选型与组合。

问题1（7 分）

对比 XA 两阶段提交与 TCC 补偿事务在一致性强度、性能、侵入性与适用场景上的差异，判断“放款”链路适合哪种。

问题2（6 分）

“还款”链路中征信报送允许异步最终一致，请设计“本地消息表 + 消息队列”的可靠消息最终一致方案，并说明如何解决“消息不丢”与“重复消费”。

问题3（6 分）

请说明 Saga 编排模式与协同模式的实现思路，并指出银行资金场景下选择 Saga 时必须注意的两个约束。

问题4（6 分）

请设计一套分布式事务监控与对账机制，保证资金类交易“绝不不一致终态”，并给出幂等与补偿的统一治理建议。



问题1（7 分）

维度	XA 两阶段提交	TCC 补偿事务
一致性强度	强隔离、强一致（全局事务）	最终一致，靠业务层面保证
性能	资源锁定重、性能差；协调者单点	无长事务锁、性能较好
侵入性	低，对业务基本无侵入	高，需实现 Try / Confirm / Cancel 三个方法
适用场景	低频、强一致的场景（对账、批量）	资金类核心链路，可自定义补偿

选型结论：“放款”链路属资金核心且频率中等，优先采用 TCC 保证资金类强一致，并辅以对账兜底。

评分要点：XA 与 TCC 的差异对比 + 本场景选型理由。

问题2（6 分）

方案设计：

- ① 业务表与消息表同库同事务写入，由本地事务保障“业务 + 消息”的原子性。
- ② 独立事务扫描消息表，把待发消息投递到 MQ，保证消息可靠发出。
- ③ 消费端确认后再修改本地消息状态。
- ④ MQ 至少一次投递，消费端按业务唯一键（还款流水号）做幂等（唯一索引 / 去重表）。

如何做到“消息不丢”：投递失败重试 + 消息表状态机推进；长时间未送达则进死信队列。

如何避免“重复消费”：消费端以业务唯一键做幂等（唯一索引 / 去重表）。

评分要点：方案设计完整，并分别说清“不丢”与“不重”两点。



第二部分 案例分析模拟题 · 试题6 参考答案（续）



知乎知学堂

问题3（6分）

编排模式（Orchestration）：由统一的编排服务按顺序调用各参与服务并跟踪状态，失败时按反向执行补偿；集中可控，利于银行审计。

协同模式（Choreography）：各服务通过事件自发驱动下一步；去中心化，但整体状态难以追踪。

两个约束：① Saga 无隔离性，中间状态对他人可见，资金场景须避免读到未提交的中间态，必要时应做资源预留；② 补偿必须完备且可幂等重试，否则会造成资金不一致。

评分要点：两种模式的实现思路 + 两个必须注意的约束。

问题4（6分）

监控：为每笔交易生成全局事务 ID，收集各参与服务的状态（进行中 / 成功 / 失败 / 补偿中）到事务状态中心；超时未达终态即触发告警与自动补偿。

对账：每日做资金流水与账务余额对账，并与证券 / 结算系统核对，出现差异走异常纠正流程。

幂等治理：资金操作统一以业务请求号（幂等键）做唯一去重；补偿操作同样要求幂等。

治理建议：统一事务框架（如 Seata 的 AT / TCC 模式）、补偿表统一登记、补偿重试与人工介入双通道。

评分要点：监控与对账机制 + 幂等与治理建议。



第二部分 案例分析模拟题 · 试题7 题干

【考点：信息安全,系统可靠性分析】

【案例背景】

某市“一体化政务服务平台”对外提供公积金、社保、不动产、营业执照等 200+ 在线服务，日均请求量 800 万，高峰期窗口服务查询密集。平台已出现多起事件：某核心数据库故障导致 3 小时服务中断；某接口被外部调用方短期高并发压垮；缓存集群单点故障引发雪崩。为达到“政务连续性保障”要求（年度可用性 $\geq 99.95\%$ ， $RTO \leq 30$ 分钟， $RPO \leq 5$ 分钟），需开展系统可靠性架构设计。

问题1（6 分）

请分别阐述冗余、心跳检测、隔离策略在保障服务可用性中的作用，并给出本平台至少三种冗余部署方式。

问题2（6 分）

针对“某接口被并发压垮”，请给出限流、熔断、降级三种保护机制的适用场景与实现要点，并说明三者之间的关系。

问题3（6 分）

结合 $RTO \leq 30\text{min}$ 、 $RPO \leq 5\text{min}$ 的目标，请设计数据库层的高可用与容灾方案，说明主从 / 集群、备份、恢复演练的配置要点。

问题4（7 分）

请设计面向“故障可观测”的可靠性监控体系（覆盖指标、告警、职责与演练），并说明如何持续提升可用性。



第二部分 案例分析模拟题 · 试题7 参考答案

问题1（6分）

冗余：通过多副本 / 多节点消除单点，故障时自动切换（如应用多实例、DB 主备、缓存集群、负载均衡）。

心跳检测：节点间周期性心跳判断存活，驱动故障转移与故障节点剔除。

隔离：把故障限定在局部（舱壁 / 线程池隔离、服务隔离、超时隔离），防止单点故障扩散为雪崩。

本平台可用的冗余部署方式：① 应用多实例 + 负载均衡；② 数据库主从 / 多副本；③ Redis 哨兵或集群；④ 消息集群；⑤ 存储多副本；⑥ 跨机房 / 跨可用区部署。

评分要点：三个概念的作用 + 至少三种冗余部署方式。

问题2（6分）

限流：保护自身不被超额流量打垮，适用于有明确容量上限的场景（令牌桶 / 滑动窗口算法，建议在网关与接口两级实施）。

熔断：下游故障时快速失败、不再发起请求（错误率超过阈值即打开断路器），给下游恢复时间，防止故障扩散。

降级：在压力或故障下主动关闭非核心功能（返回兜底数据或降级文案），保证核心可用。

三者关系：限流控住入口负载 → 熔断保护下游健康 → 降级保障核心体验，常配合形成“入口限流、中间熔断、末端降级”的分层保护。

评分要点：三种机制各自的适用场景与实现要点 + 三者的协同关系。

问题3（6分）

高可用：数据库采用一主多从或高可用集群，主库故障自动切换（MHA / Orchestrator），读写分离分摊压力。

容灾：同城双活 / 异地灾备，用 WAL 日志或同步复制保证数据不丢。

备份：全量备份（每日）+ 增量 / 归档日志（实时），并定期做异机备份。

恢复演练：制定季度或半年的 RTO / RPO 演练计划，验证主备切换时长与数据完整性，形成恢复手册并定责到人。

评分要点：主从 / 集群方案 + 备份策略 + 恢复演练机制。



第二部分 案例分析模拟题 · 试题7 参考答案（续）

问题4（7 分）

指标：采用 Google SRE 的四个黄金信号（延迟、流量、错误、饱和度），并补充核心业务链的可用性指标；RED / USE 方法可作为细分参考。

告警：分级告警（P1 ~ P4）+ 值班轮值 + 超时升级；配合可视化看板与链路追踪辅助定位。

职责：明确 SRE / 运维 / 开发的责任矩阵，重大故障复盘闭环。

演练：定期开展混沌工程 / 故障演练（注入主机宕机、依赖故障），检验架构韧性，并把可用性纳入 OKR 持续改进。

评分要点：指标与告警体系 + 职责与演练机制。



第二部分 案例分析模拟题 · 试题8 题干

【考点：信息安全,安全体系】

【案例背景】

某银行推出移动金融 App（含转账、理财、线上贷款），IT 环境包含私有云、分支行网点、外包合作方调用接口、员工远程办公四大访问域。安全团队识别出典型案例风险：员工账号被钓鱼后横向渗透内网；App 接口被刷 / 被自动化工具攻击；外包接口缺少调用方验证与消息防篡改；内网运维通道暴露面过大。为应对上述风险，安全部门拟采用“零信任”理念重构安全架构，并叠加国密算法与等保 2.0 合规要求。

问题1（6 分）

请解释零信任安全的三大核心原则（永不信任持续验证、最小权限、微隔离），并说明零信任相对于传统边界防御的本质差异。

问题2（6 分）

针对“员工账号被钓鱼后横向渗透”，请从身份认证、持续信任评估、访问控制三个层面给出零信任落地措施。

问题3（6 分）

针对 App 接口与外包接口，请设计 API 安全防护方案（含鉴权、签名防篡改、防重放、风控限流）。

问题4（7 分）

请给出国密算法（SM2 / SM3 / SM4）在本方案中的合理用途配比，并说明合规测评（密评 / 等保）对架构的约束要点。



第二部分 案例分析模拟题 · 试题8 参考答案

问题1（6分）

三大核心原则：

- ① 永不信任、持续验证：每一次访问都先验证身份与设备 / 风险，不因“在内网”而放行。
- ② 最小权限：按需授权、动态收敛，权限随上下文变化。
- ③ 微隔离：东西向流量按业务与身份进一步细分网络与访问控制。

本质差异：传统边界防御假定“内网可信”，一旦边界被突破即可在内网横向漫游；零信任以“身份 + 设备 + 上下文”作为信任依据、默认拒绝，不再区分内外网，构建以身份为中心的动态访问控制。

评分要点：三大核心原则 + 与传统边界防御的本质差异。

问题2（6分）

身份认证：多因素认证（口令 + 短信 / OTP / 生物识别）；统一身份与单点登录（IAM / SSO）。

持续信任评估：结合设备指纹、行为画像、地理位置与异常行为（异地登录、异常时点）做动态风险评分，高风险降权或强制二次认证。

访问控制：基于角色的最小权限 + 动态访问策略（按项目 / 资源细分）；对敏感系统执行“按需申请、审批 + 时长限制”。

评分要点：身份认证、持续信任评估、访问控制三个层面各一点。

问题3（6分）

App 接口：OAuth2 / JWT 鉴权、设备指纹绑定、接口加密（国密 SM4）、提交频率风控限流与设备风控（设备黑名单、指纹识别）。

外包接口：由 API 网关统一接入，为外包方签发应用凭证（AppKey / 密钥）；请求加时间戳 + 随机数 + 签名（SM3 摘要 + SM2 验签）防篡改与防重放；调用额度与频率限制、日志审计留痕。

评分要点：App 侧与外包侧的防护措施各占一半。



第二部分 案例分析模拟题 · 试题8 参考答案（续）

问题4（7分）

用途配比：

SM4：业务敏感数据与链路的对称加密（App 与服务器之间的数据传输、数据库敏感列）。

SM3：消息完整性校验、签名摘要与防篡改。

SM2：数字签名验签、证书体系与通信身份认证（双向认证）。

密钥管理：采用符合国密要求的密钥托管方案。

合规约束：按等级保护与商用密码测评要求，重要系统须通过密评；加密算法、密钥管理、审计留存需符合监管口径，安全架构设计须预留国密改造与合规证明材料。



第三部分 论文范文讲解

之前没讲完的2篇



论集成云服务系统在云平台中的设计与应用（AI生成）

随着企业数字化转型深入，企业内部HR、OA、财务、制造、税务等业务系统独立建设，普遍存在数据孤岛、异构系统难以互通、业务流程无法协同等问题。云平台集成服务系统可通过统一连接器、流程编排、事件驱动、数据集成能力，实现多异构系统的数据互通与业务协同，是企业数字化中台建设的核心基座。在大型SaaS云平台建设中，如何通过合理的架构设计，解决异构适配、数据一致性、多消费者竞争、流量管控等核心问题，成为架构设计的关键。

请围绕“云平台集成云服务系统的架构设计与应用”论题，结合你参与的实际项目，依次完成以下论述：

- 1、简要叙述你参与的云平台集成服务项目以及你承担的核心架构设计工作。
- 2、简述云平台集成系统分层插件架构的核心思想，说明平台基础设施层、消息适配层、连接器扩展层、核心业务层等分层架构的主要职责与协作关系。
- 3、结合项目实践，详细论述你在项目中解决异构系统集成的核心设计方案，重点说明跨系统数据一致性保障、精细化流量治理、分布式多消费者竞争问题的实现思路、技术选型与落地过程。



论集成云服务系统在云平台中的设计与应用

2022 年 3 月至 2023 年 6 月，我参与了某公司产品“集成云服务平台”（以下简称“集成云”）项目建设。该项目是我公司 SaaS 产品和云平台的核心组成部分，服务于数千家企业客户，日均处理数据同步任务超过 10 万次。集成云不但承担着需要为原 ERP 用户逐步升级到 SaaS 云订阅服务的核心任务，同时也需要为企业或集团内外部多个不同异构系统提供进行数据集成的核心要求。项目研发团队超 20 人，项目总投资超 1000 万元，作为平台长期研发项目，一期二期总历时超过 12 个月，我作为核心技术架构之一，负责数据复制和流程编排引擎，及连接器模块的架构设计与实现。项目一期于 2022 年下半年成功上线，并且不断迭代与维护，作为公司云平台的基座组成部分之一，获得众多企业客户和伙伴的良好评价，并成为公司质量金奖项目。本文以该项目为例，论述集成云在微服务架构下的数据协同，流程引擎和事件驱动的核心设计与落地实践。

现代企业的数字化与信息化建设开始逐步走向企业平台管理，随着企业业务和规模的快速扩大，不同业务线和部门之间会创建不同的信息系统，如 HR，公司 OA，制造云，税务云，财务管理等，多个系统之间不能进行数据流通，各自系统数据和服务单独治理形成各自的信息孤岛，也无法形成统一的整体决策和管理。我公司是国内企业级管理软件知名供应商，旨在为客户建立完善的企业管理平台，不仅仅是为企业提供多个业务线上的管理软件产品，同时更要为所有企业内外部系统建立一个统一的企业数字化平台，集成云就是其中作为黏连所有系统的数据通路的至关重要的一个部分。



WPS_266548598

时间、项目介绍、团队介绍、项目金额、岗位、职责，但是最后一句的内容跟后面的论点



WPS_266548598

项目的背景描述



2022 年初，平台决定建设集成云，为我公司云平台下各个云服务产品之间建设数据通路服务，同时也需要和客户公司内其他异构系统形成数据协同。让我公司云平台产品成为企业数字化管理的核心基座。集成云是建立在云平台的微服务架构基础之上的独立应用，所以集成云引入了”基于云平台下的分层插件架构”，建立五层分层架构，包括平台基础设施层、消息队列适配层、连接器扩展层、核心业务层、接入和表现层。其中自研的平台基础设施层为我们的整体架构提供了服务注册，缓存，数据库，动态领域模型管理等所有基础设施。我作为核心架构之一，主导连接器扩展层、消息队列适配层和部分核心业务层的设计与实施工作。重点解决异构系统集成数据一致性保证、流量精细化管理、事件多消费者竞争等核心问题。

在项目实施前中期，我深入研究了集成服务的核心组件和协同机制，其核心设计思想是通过连接契约屏蔽异构，以配置驱动替代编码，以事件编排解耦流程，从而通过产品配置统一解决企业系统集成与数据集成。

集成云主要包括连接器和元数据，集成管理，服务编排三大部分组成，云平台会为所有活动建立监控统计和日志追踪机制。连接器和元数据负责管理所有的内外部连接器配置及来源元数据，通过预先同步外部系统的结构元数据（包括服务接口，数据表，实体模型等），将异构系统的能力描述统一存储在集成云服务内部。实现集成配置的可视化，可验证和低错误率。集成管理面向数据，基于 ETL 管道模型，使用 QuartzCronExpressin 解析定时规则或事件驱动，并通过 Redis 分布式锁保障集群环境下任务互斥执行，他包括数据集成，API 集成，消息集成等多种方案，管理完整的集成方案包的全生命周期。服务编排面向流程实例，通过整合已有的连接器和元数据，通过动作和多步骤编排，彻底解决跨系统的多步骤集成方案问题。



WPS_266548598

回应第一个子问题，并对项目进行详细一些的介绍，有涉及到业务和技术，建议加上一些具体的技术名词



WPS_266548598

这一段按理来说应该是一个过渡段，引入第二个子问题，并且把纯概念性的回答引入到自己的项目业务中去，但是我没看到这一段和下一段的必然关系



WPS_266548598

这一段是回应第二个子问题，根据回答的内容可以看出子问题估计是：集成云有哪些组成部分，这些组成部分的作用是什么，这里对这些子问题的回答缺少跟自己的项目结合



基于上述设计方案,我将重点围绕集成管理和服务编排共同解决企业集成场景下的数据一致性问题,连接器模块深度配置实现流量精细化管理,管理集成方案运行时的 Redis 分布式锁控制多消费者竞争问题三个维度开展工作。

企业集成场景下的数据一致性问题区别于单系统数据库和事务的数据一致性问题,核心挑战是跨系统层面数据一致性问题,由于跨系统无法共享,系统之间调用和业务操作不可逆,网络等原因可能引起系统数据集成失败。我处理的核心思路遵从:用最终一致性替代强一致性,用补偿替代回滚。我构建了 E (Extract 提取) -T (Transform 转换) -L (Load 加载) 三位一体的可观测基础,每一个阶段都存在自己的基于 AtomicLong 的独立计数器,三段计数器把“任务失败”这个模糊的状态,分解成了哪个阶段,失败多少,失败原因的精确信息。建立失败检测机制的观测保障后,在转换阶段和加载阶段采取多种策略保证集成数据一致性,候选键 + 幂等写入保证批次写入的原子性和最终写入一致性;分片 + 断点续传执行批次内多分片并行执行,每个分片独立执行单元有自己组成的独立执行状态,当执行单元发生失败,通过执行重试;指数退避重试保证在任务失败后,采取立方数策略,解决任务瞬时抖动,目标系统瞬时故障等常见原因引起的执行失败,确保重试后的保证执行。

在连接器扩展层,我们通过接口契约把连接配置和连接运行时彻底分离,来屏蔽系统异构性对整个核心集成逻辑引擎的影响,这是一个多对多问题,通过建立统一扩展层,利用策略模式完全隔离内部引擎逻辑和外部连接器配置的耦合。在运行时连接层配置通过构建目标系统的 License 控制准入策略,引入 HikariCP 管理连接数上限控制,设定任务调度线程,利用令牌桶思想管控调用频率、限流、并设定熔断要求,当目标系统故障失败达到阈值时自动停止任务。

由于云平台是一个分布式微服务架构,当完成连接器配置被用于执行数据复制任务时,任务的调度和执行过程中对于多节点集群部署,就会产生多个消费者竞争执行的问题,在这



WPS_266548598

过渡段,下面引出论点,但是这里你要围绕的三个维度不是很容易看出来,因为维度太长了,建议考试的时候加粗显示



WPS_266548598

建议先对论点进行解释,再结合项目进行论述



WPS_266548598

论点的叙述方式是:论点+论点的解释+背景+方案提出+基于业务进行选型+涉及到的技术+实行结果,按照这里来看,这里的论点叙述有背景+方案+实行结果,缺少了涉及的技术以及对过程描述的过于详细



WPS_266548598

一样缺少对论点的解释



由于云平台是一个分布式微服务架构，当完成连接器配置被用于执行数据复制任务时，任务的调度和执行过程中对于多节点集群部署，就会产生多个消费者竞争执行的问题，在这里，我们引入了 Redis 和内部 Rabbit MQ 消息集成来解决问题，虽然使用的是平台提供的自研框架 dlock，但是原理选型上，依然是标准的 Redis 分布式锁实现。通过调用设定 Lua 脚本设计实现了方案级互斥锁和数据级互斥锁，共同作用解决任务队列竞争问题，并通过 RabbitMQ 管理大批量任务入队问题进行高并发状态下的削峰处理，实现任务异步化处理，保证消息只被一个消费者取走。

集成云一期于 2022 年 8 月上线，后不断迭代版本，在 2023 年中完全稳定面对内外部各种集成任务的挑战。于 2023 年中，稳定周期内，为各大企业客户的数据集成任务提供了大量集成标准方案，从原来的专业开发集成数据脚本或程序方案，通过配置集成云方案包，把一个系统的集成工作量大幅度从月周期降低到 2-5 日周期范围内，获得公司内外和客户的一致好评，并获得当年公司内质量金奖项目。

通过该项目，我深刻认识到对于架构过程中，引擎稳定和内外部配置模式的插件化设计思想的作用，并且如何利用平台思维搭建业务服务架构，集成云本身属于单体插件化的应用架构，虽然运行在微服务的架构平台之上，但是利用业务差异化的平台能力，恰好解决了具体的场景问题也是一种成功的架构方法论。

**WPS_266548598**

一样缺少对论点的解释

**WPS_266548598**

这一段还行，有矛盾点，有解决方案，有技术，有结果

**WPS_266548598**

项目统一的结尾，需要背下来

**WPS_266548598**

进行技术或者业务的总结，没问题



论高并发系统的架构设计与性能优化

随着互联网业务的快速发展与用户规模的持续增长，高并发场景已成为企业信息系统面临的核心挑战之一。大促活动、热点事件、爆款商品抢购等场景往往会在短时间内产生千万级甚至亿级的瞬时流量洪峰，对系统造成巨大的准入压力。传统的系统架构受限于纵向扩展的瓶颈、数据库连接池上限以及同步阻塞的I/O模型，极易在流量高峰时出现响应超时、资源耗尽、服务雪崩，甚至全站不可用等问题。如何通过合理的架构设计，在确保数据最终一致性的前提下有效承接瞬时高并发流量，同时保障核心链路的稳定运行，是衡量架构师设计能力的关键标尺。

请围绕“论高并发系统的架构设计与性能优化”论题，依次从以下三个方面进行论述：

1. 概要叙述你参与管理和开发的软件项目以及你在其中所担任的主要工作。
2. 论述高并发系统的核心设计原则与关键技术手段（如弹性伸缩、缓存、异步、限流、降级等），并说明这些技术如何协同实现“高吞吐、低延迟”的核心目标。
3. 结合你具体参与的项目，说明高并发架构方案的选型依据、落地过程中的关键挑战及应对措施，以及最终的实施效果。

2024 年 6 月，我参加了“数据脱敏水印溯源统一管理平台”项目的建设，项目总投资 520 万元，历时 12 个月。该平台是为了解决企业在数据存储、数据共享等场景下面临的敏感数据泄露风险与合规压力问题，提升企业的数据安全合规与溯源能力。我在该项目中担任系统架构师，主要负责系统架构设计和技术选型等工作。

本文结合我的实践，以“数据脱敏水印溯源统一管理平台”项目为例，讨论高并发系统的架构设计与性能优化。接下来我将从弹性扩容、高并发查询优化、系统熔断降级三个维度进行论述。实践证明，通过以上技术的应用，能有效实现高并发系统的高吞吐、低延迟核心目标。最终系统于 2025 年 6 月正式上线，已接入近 100 个业务系统，日均处理脱敏请求 800 万次，获得业主单位年度优秀项目表彰。

随着《数据安全法》《个人信息保护法》等法规的全面实施，企业在数据共享、开发测试及外发等场景中，面临敏感数据泄露风险与合规压力。当前，各业务系统往往独立实现数据脱敏与水印嵌入功能，缺乏统一的技术标准和管控手段，导致脱敏算法不一致、处理结果不可控，水印信息格式各异、难以解析。更为关键的是，一旦发生数据泄露事件，由于缺乏标准化的水印溯源机制，无法追溯泄露源头、定责定损，给企业带来巨大的法律风险与经济损失。为解决上述问题，数据脱敏水印溯源统一管理平台应运而生。该系统有效实现了对敏感数据的统一识别、脱敏处理、水印注入及泄露溯源，全面提升企业数据安全防护能力与合规治理水平。2024 年 6 月，我参加了“数据脱敏水印溯源统一管理平台”项目的建设，项目总投资 520 万元，历时 12 个月。我在该项目中担任系统架构师，负责系统总体的架构设计、技术选型、核心模块技术攻关，带领架构团队共 5 人。系统的主要功能模块包括：涉敏数据治理、脱敏水印策略管理、算法密钥管理、溯源管理、运营监控等。该系统采用微服务架构，使用 Nacos、Redis、Kafka、PostgreSQL、ClickHouse 等技术，采用 DevOps 实现持续集成和交付，结合 Docker 和 Kubernetes 实现容器化部署和编排。该项目目标接入业务系统 500 个，因此存在瞬时流量洪峰，有高吞吐、低延迟的性能需求。

高并发系统的核心设计原则有弹性扩容 异步削峰 缓存分担流量压力 服务熔断降级



無諾

时间、项目介绍、岗位、职责、项目规模、历时都有，语言流畅



無諾

项目背景很完整，交代清楚了项目出现的原因



無諾

回应第一个子问题，对项目功能进行描述，以及对项目所涉及到的技术进行阐述



高并发系统的核心设计原则有弹性扩容、异步削峰、缓存分担流量压力、服务熔断降级等。在面对瞬时流量洪峰时，服务网关通过负载均衡将请求分发至不同的微服务，此时可以通过流量阈值监测，对微服务实例实现弹性扩容，以避免单节点被压垮。还可以引入消息队列，将请求发送至消息队列里，再根据流量情况动态增减分区数量、消费者数量，比如在我的项目里，脱敏日志通过 **Kafka** 异步写入 **ClickHouse**，以此实现了异步削峰。在微服务内部，为了避免数据库连接池资源耗尽和保护数据库不被瞬时流量冲垮，可以加入 **Redis** 作为缓存

来分担大部分的流量，只有极少部分流量才会进行到数据库。如果出现请求超时响应、资源耗尽等问题，可以通过熔断等机制实现服务的主动降级，返回默认值或页面友好提示稍后再试。通过以上技术手段，能在确保数据最终一致性的前提下，有效承接瞬时高并发流量，同时保障核心链路的稳定运行。以下结合“数据脱敏水印溯源统一管理平台”项目实际，详细论述我在实践中的具体做法。

关于弹性扩容的设计原则，我识别到脱敏请求量存在明显的峰谷，工作日白天高、夜间低，如果微服务采用固定副本数会造成资源浪费，高峰期又出现资源不足的情况。我提供了两种解决方案。一种是基于 **Kubernetes HPA** 实现的实时响应方案，根据 CPU、内存、网络带宽、磁盘 IO 等多个核心指标的阈值，实现自动扩缩容，白天高峰期脱敏服务最多有 20 个副本，夜间低谷时期缩至 3 个副本。多副本再配合 **Nacos** 集群的健康检查，自动摘除故障节点。另一种是基于定时的弹性伸缩方案，根据历史流量规律预设扩缩容时间表，8:00-17:00 保持 15 副本，其余时间保持 3 副本。对以上两种方案分析对比，方案一实时响应流量变化，适应突发流量，但存在扩容 1-2 分钟的滞后性，方案二无滞后、资源规划精准，但无法应对突发流量。最终我选择基于 **HPA** 的实时扩缩容方案，因为脱敏业务存在不可预测的突发高峰，实时响应能力比定时更关键。

关于高并发策略查询的性能优化，我引入 **Redis** 作为缓存提前加载热点策略。起因是业务高峰期，脱敏服务出现大量请求超时，服务不可用。经过排查分析，我发现策略查询接口的 QPS 峰值达到 2 万，PostgreSQL 数据库 CPU 100%，PostgreSQL 成为系统的性能瓶颈。于是



無諾

这是一个过渡句，但是略显生硬，应该想办法来通过描述这个项目来引出第二个问题



無諾

回答第二个子问题，在其中引入自己的项目来增加可信度以及让纯概念与你的项目进行结合，此处可以用一个真实例子来说明，更加增强说服力



無諾

论点的叙述，第一句最好是对论点的专业名词做解释，比如这里需要用 1-2 句话解释下什么叫弹性扩容，然后再写弹性扩容跟你项目相关的地方



無諾

后面的内容写的挺好，有业务、有背景、有技术选型、有技术实现



关于高并发策略查询的性能优化，我引入 Redis 作为缓存提前加载热点策略。起因是业务高峰期，脱敏服务出现大量请求超时，服务不可用。经过排查分析，我发现策略查询接口的 QPS 峰值达到 2 万，PostgreSQL 数据库 CPU100%，PostgreSQL 成为系统的性能瓶颈。于是我引入 Redis 作为缓存提前加载热点策略，查询逻辑为优先查缓存，如果未命中再查数据库，同时写回缓存。经过一段时间运行和指标监测，缓存命中率达到 95% 以上，数据库 CPU 降至 20%。但随着 Redis 的引入，随之而来的是数据一致性问题。最初我们采用的是“延迟双删”方案，但难以把握“等待的窗口时间”，等待时间太长影响接口性能，等待时间太短不能解决数据一致性。于是我决定部署 Canal 监听数据库 binlog，异步更新策略到 Redis 缓存。因为 Canal 保证消息的可靠和顺序，且时间延迟很低、对业务代码无入侵，所以不仅实现了数据的最终一致性，还提高了系统性能和可维护性。

系统有一个典型的高并发下的熔断降级场景。脱敏服务自身资源耗尽，如 CPU 满载、内存不足等，脱敏处理能力急剧下降，大量请求积压，可能导致系统雪崩。因此我决定引入熔断降级组件，在进行组件选型时，我对比了 Netflix 开源的 Hystrix 与阿里巴巴开源的 Sentinel。Hystrix 基于线程池隔离，线程切换开销较大，且近年进入维护状态，不再更新版本。Sentinel 以流量控制为切入点，采用轻量级信号量隔离，性能开销低于 Hystrix。Sentinel 还支持动态规则配置，可以通过 Nacos 实现实时推送规则，无需重启。而 Hystrix 的规则配置多为静态，调整后需要重新部署。综合性能、社区活跃度、动态规则配置考虑，最终我选择使用 Sentinel 组件配置熔断降级。具体做法是，针对脱敏服务设置系统负载阈值，CPU 使用率超过 90% 且持续时间 30 秒，触发熔断降级，脱敏服务快速返回预设的默认脱敏结果，对敏感字段统一返回***，保证核心链路的可用性。引入 Sentinel 后，脱敏服务在 CPU 满载时能够自动降级，用户几乎无感知。

整个项目历时 12 个月，于 2025 年 6 月完成交付，截至目前已稳定运行 10 个月，接入的业务系统已经近百个，累计处理脱敏请求 24 亿次，接入业务系统满意度达 96%，获业主单位信息安全管理年度优秀表彰。系统成功支撑了多次流量高峰，验证了高并发架构设计的有效性。经过该项目的实施验证，我深刻认识到，高并发场景下的架构设计需要贯穿需求分析、设计、开发、运维全生命周期。特别是系统可用性不是靠单节点高可靠实现的，而是靠冗余、自动故障转移、快速恢复共同实现。但在微服务治理方面还暴露了一些不足，比如服务拆分粒度难掌握，导致服务数量过多。后续计划引入领域驱动设计，通过划分限界上下文、区分子域类型、定义边界等原则，重新划分微服务，以提高系统的可维护性和可修改性。

后面的内容写的挺好，有业务、型、有技术实现



知乎学堂

無諾

这句结尾的话是对整个文章的总结和收尾，是必须得有的

無諾

进行项目的收获的总结，以业务或者技术的方式进行叙述

無諾

引入一些客观存在的问题，提出对应的解决方案进行收尾



T H E E N D

功不唐捐，玉汝于成！

• 开 启 新 征 程 •

