

如何设计一个 分布式计数服务

- 系统设计面试案例

架构师 ~ 杨波



扫码试看/订阅

《分布式系统案例课》视频课程

目标



演示一个系统设计面试流程



演示一个端到端系统设计案例



介绍一些分布式系统基本概念

大纲

1. 需求收集和简化架构

2. 存储设计

3. 计数服务设计(上/下)

4. 查询服务设计

5. 技术栈选型

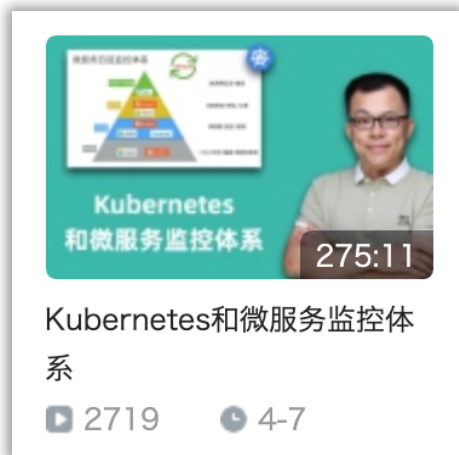
6. 额外考量和总结

第 01 节

需求收集和简化架构

面试题

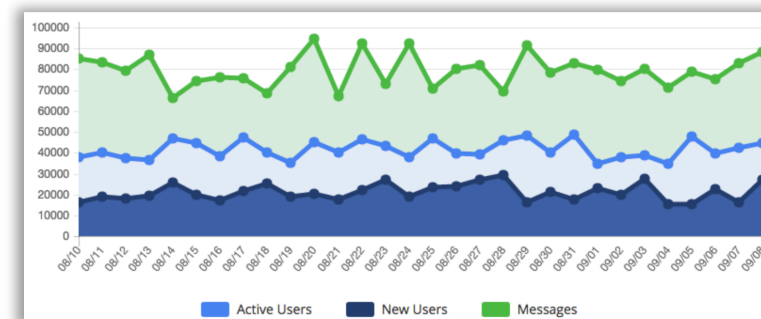
对B站视频观看数计数



对头条关注数计数



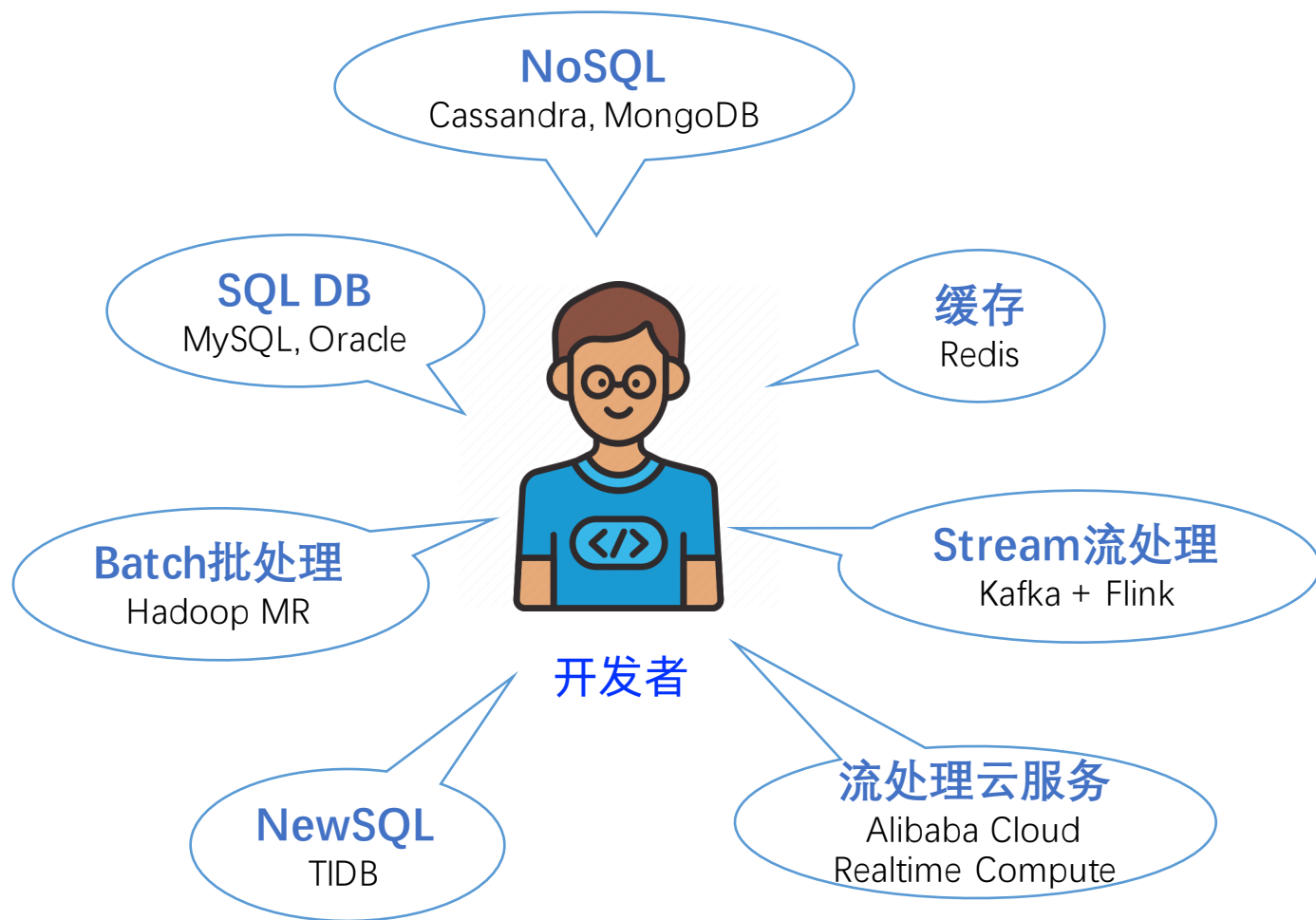
对业务指标/应用性能计数



需求沟通



面试官



需求澄清

场景用例

- 谁用这个系统?
- 用户如何用这个系统?

量级规模(读/写)

- 每秒查询请求?
- 每个请求查询多少数据?
- 每秒处理多少个视频观看记录?
- 流量模式?是否有流量高峰?

性能

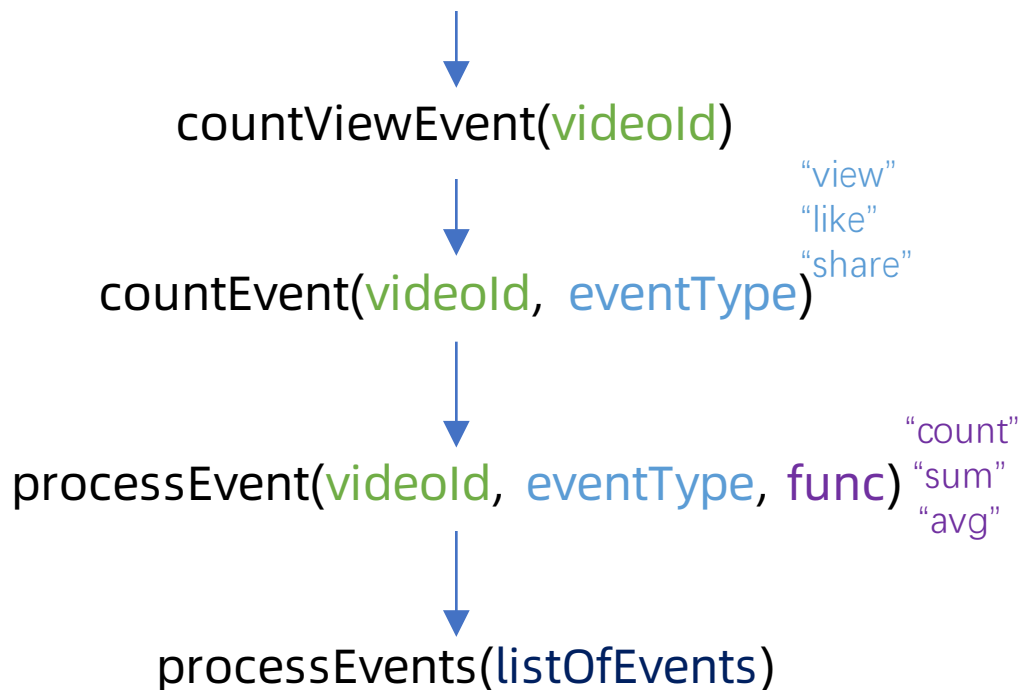
- 预期从写入到读取数据的延迟?
- 预期p99读请求延迟是多少?
- 高可用性(一般隐含)

成本

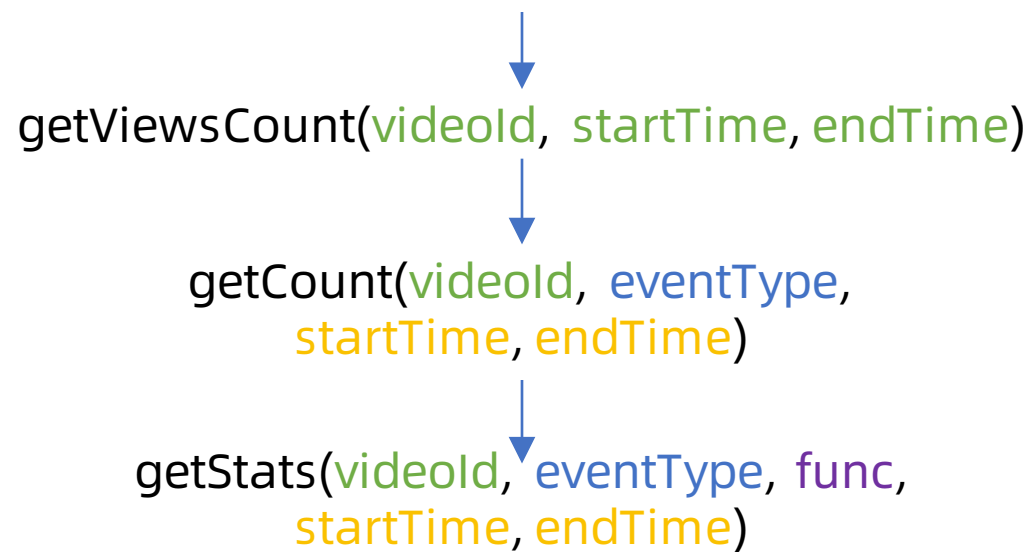
- 开发成本有限制?
- 运维成本有限制?

功能需求 - API

处理需求: 对视频观看数进行计数



查询需求: 根据时间段返回视频观看数量



非功能需求

量级和B站
相当

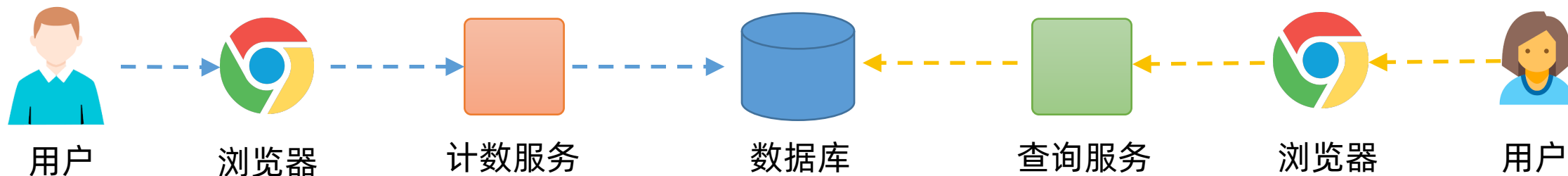
高并发高性能
高可用



面试官

- **规模** ~ 每秒处理1w+视频点击观看记录,
- **性能1** ~ 写入/读取毫秒级延迟
- **性能2** ~ 写入到读取更新分钟级延迟, **近实时流处理**, **最终一致**
- **高可用** ~ 无单点失败
- **水平按需扩展**
- **开源低成本**

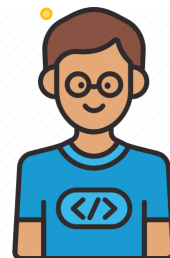
从简化架构开始



我有不少问题，
该从哪里开始呢？



保持主动，不能
被牵着鼻子走



分布式系统的核
心是数据，可以
从存储设计开始

第 02 节

存储设计

存什么?

单个事件(每次点击)

Videoid	Timestamp	...
A	2020-05-21 16:17:21	...
A	2020-05-21 16:17:32	...
A	2020-05-21 16:17:40	...
B	2020-05-21 16:22:47	...

- 可以快速写入
- 按需聚合运算
- 查询慢
- 耗存储

聚合数据(例如每分钟)

Videoid	Timestamp	Count
A	2020-05-21 16:17	3
B	2020-05-21 16:22	2

- 查询快
- 存储少
- 只能按已聚合方式查询
- 需实时聚合运算
- 出错修复无原始数据



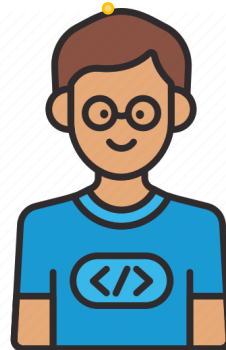
数据库选型

你认为选哪种数据库合适?为啥?



- **可扩展** ~ 根据读写规模按需扩展
- **高性能** ~ 快速读写
- **高可用** ~ 不丢数据, 灾难恢复
- **一致性折衷**
- **数据模型易于升级**
- **成本**
- **开发者学习门槛**

SQL和NoSQL都可以, 我来分析一下

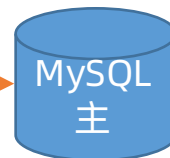


SQL 数据库 + 客户端嵌入代理

计数服务

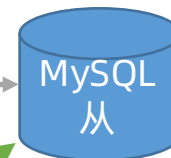


Store



A-M

主从
复制

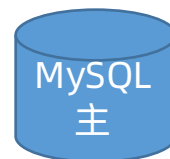


A-M

查询服务

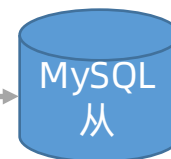


Query



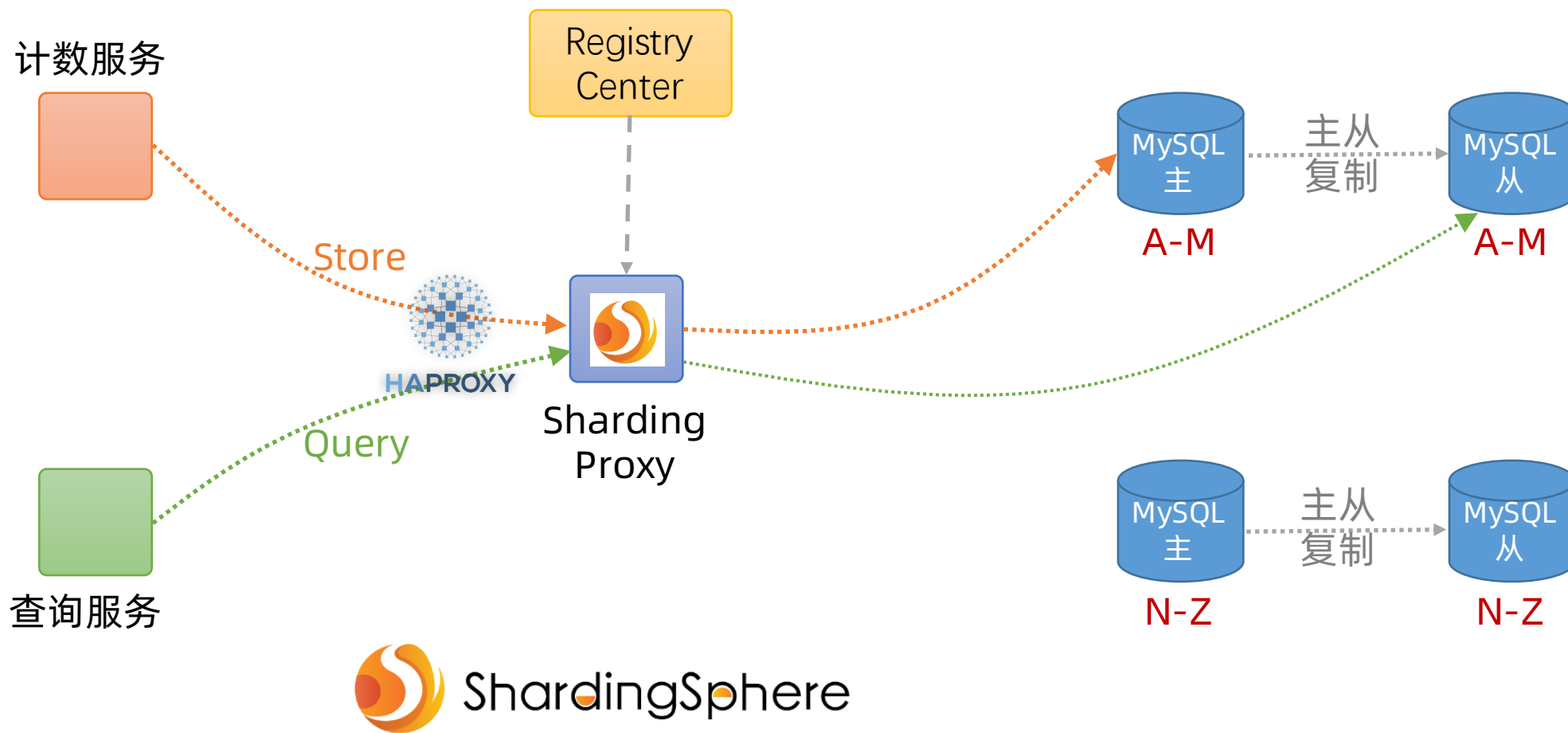
N-Z

主从
复制

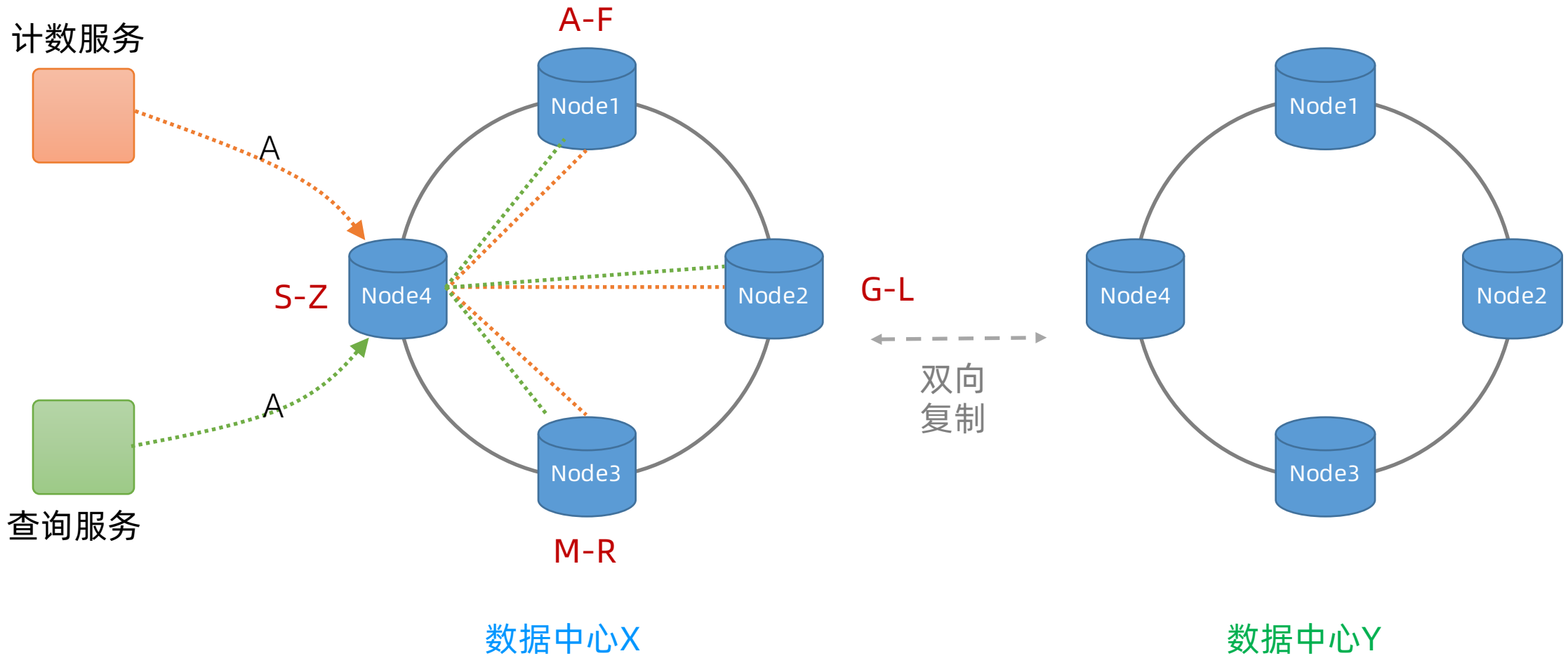


N-Z

SQL 数据库 + 独立部署代理层



NoSQL 数据库(Cassandra)



表设计

Video



深入理解Kubernetes网络

Views

深入理解Kubernetes网络

727

5-1

极客时间
Space



SQL数据库
(MySQL)

Video_Info			
Videoid	Name	...	SpaceId
X	深入理解K8s网络		123

Video_Stats

Videoid	Timestamp	Count
X	15:00	2
X	16:00	3
X	17:00	6

Space_Info

SpaceId	Name	...
123	波波微课	...

NoSQL数据库
(Cassandra)

Videoid+Date	Space Name	Video Name	15:00	16:00	17:00	...
X-2020.05.21	波波微课	深入理解K8s网络	2	3	6	...

第 03 节

计数服务设计

计数服务如何实现

- **可扩展** ~ 可根据写入规模按需扩展
- **高性能** ~ 快速写入，高吞吐
- **高可用** ~ 不丢数据，灾难恢复，数据库慢或者不可用？

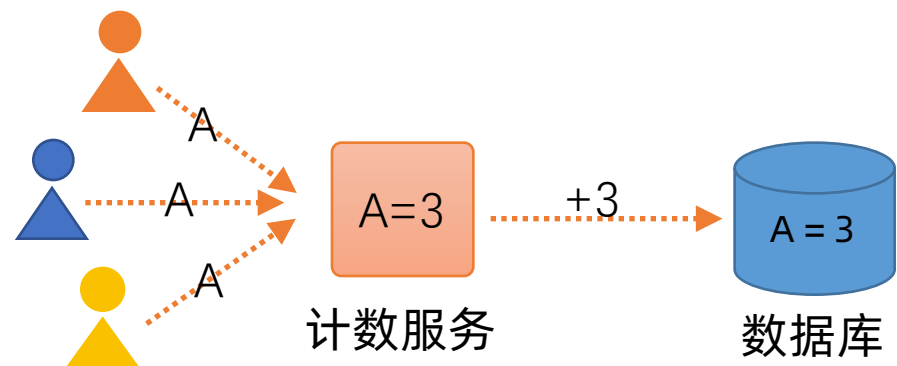
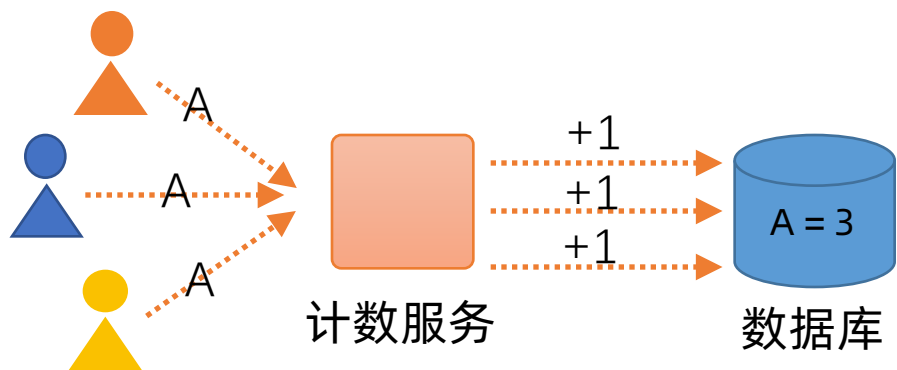
如何实现可扩展、
高性能和高可用
数据处理？



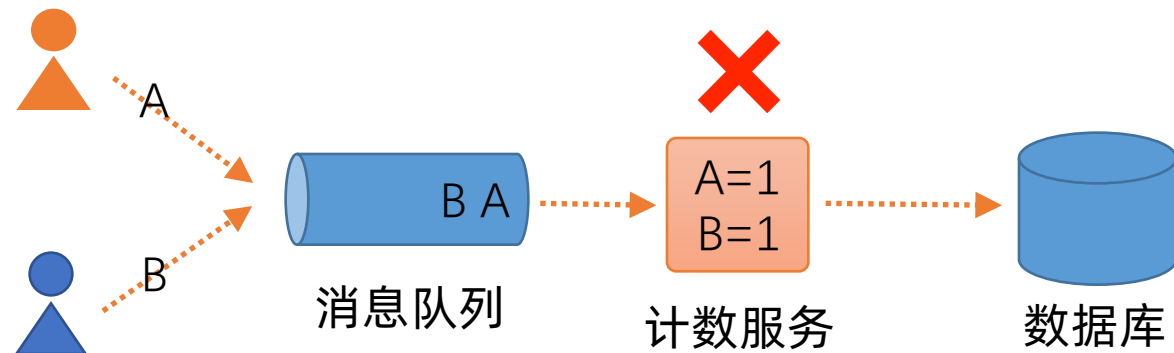
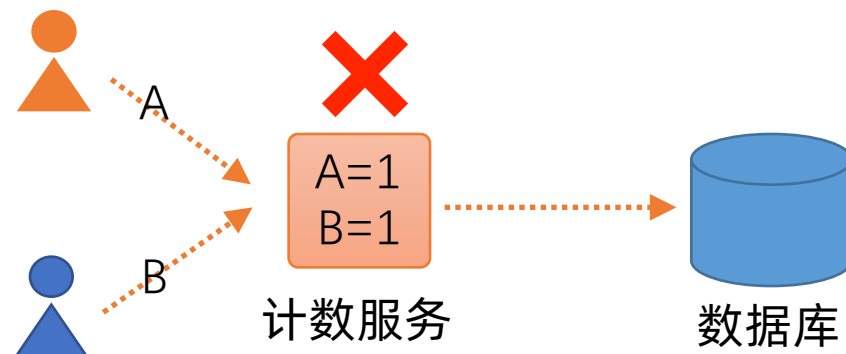
可扩展 = Partitioning/Sharding
高性能吞吐 = 内存计算，Batch批处理
高可靠 = 持久化 & Replication & checkpointing

数据聚合(agggregation)基础

预聚合?

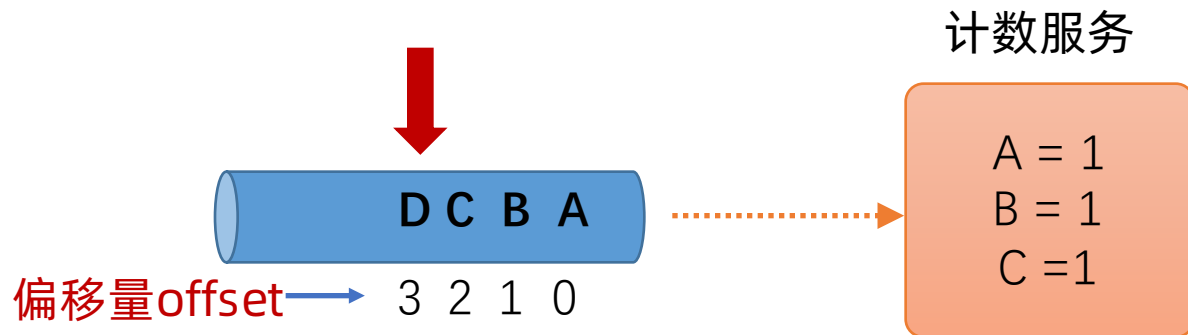


Push vs Pull?



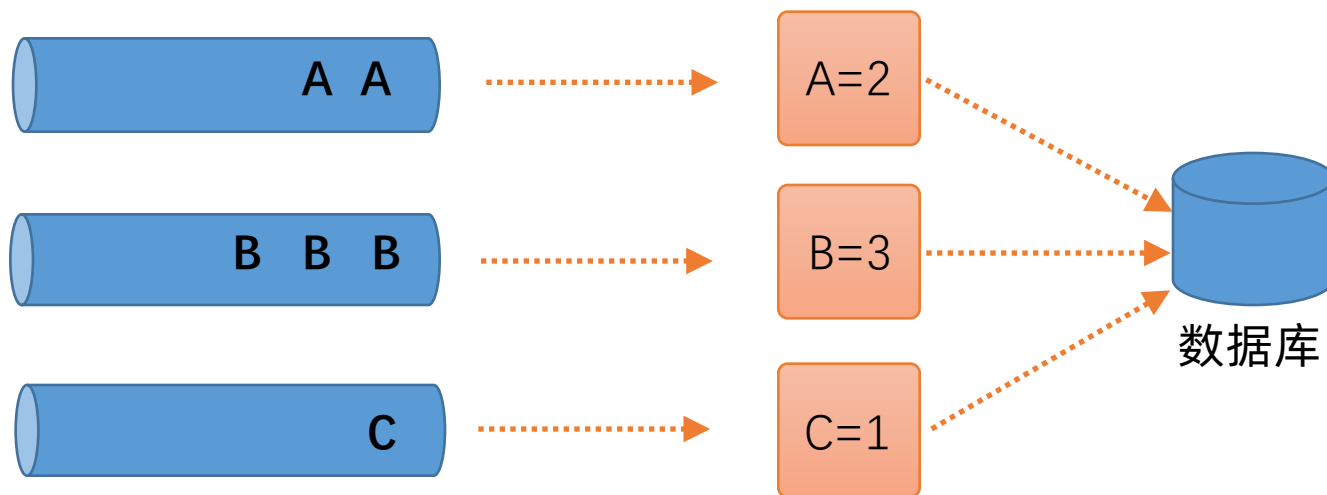
消息队列基础

检查点
Checkpointing

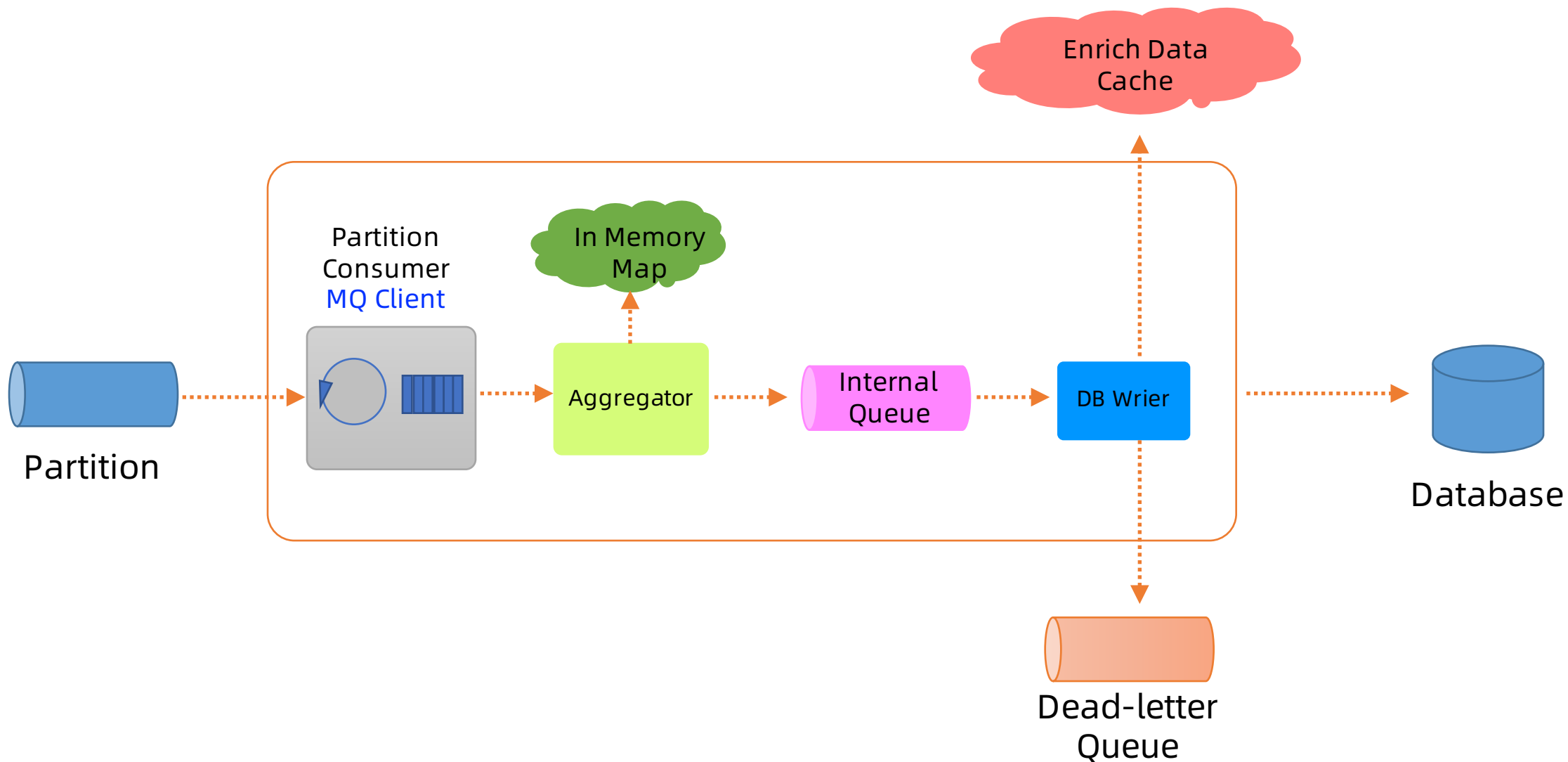


 kafka

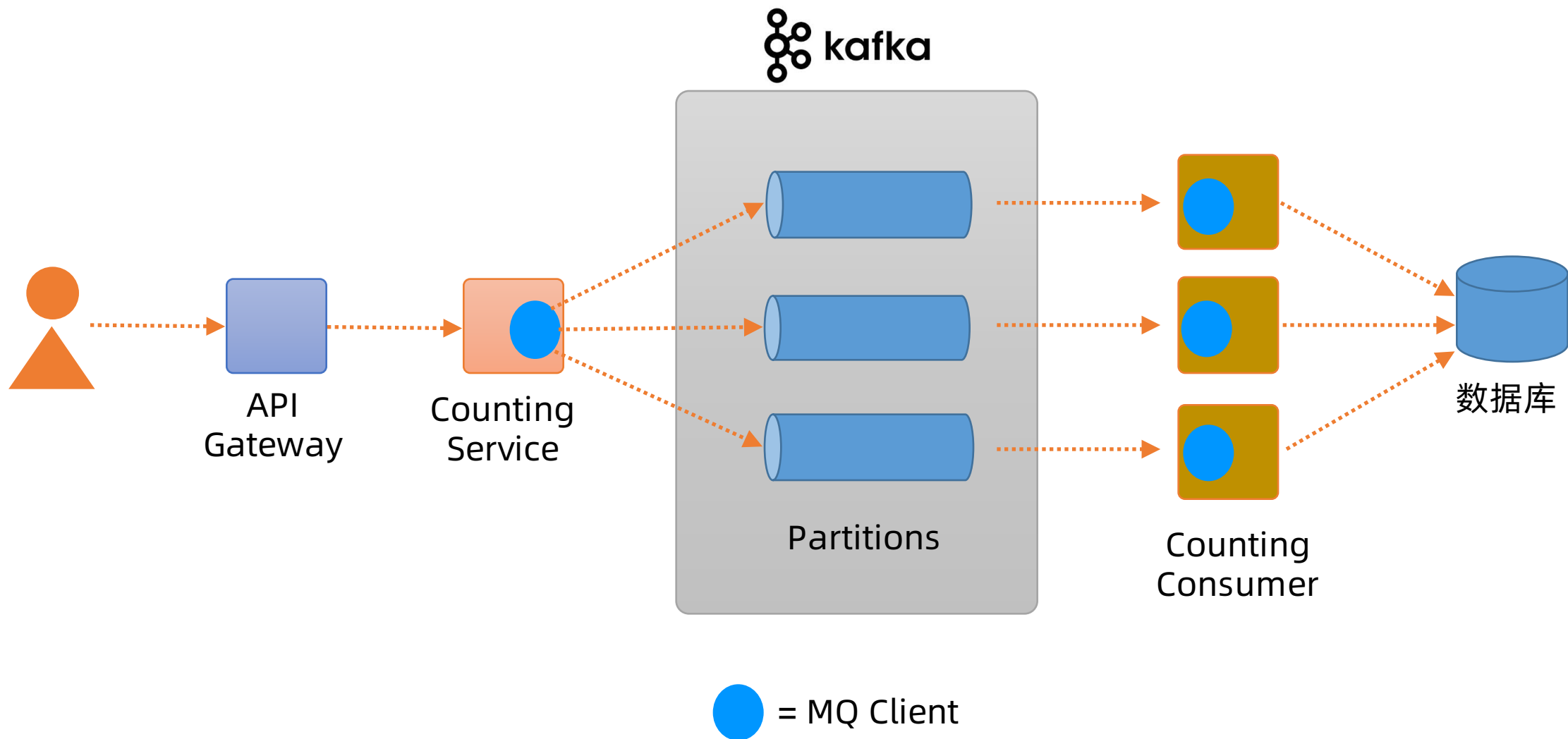
分区
Partitioning



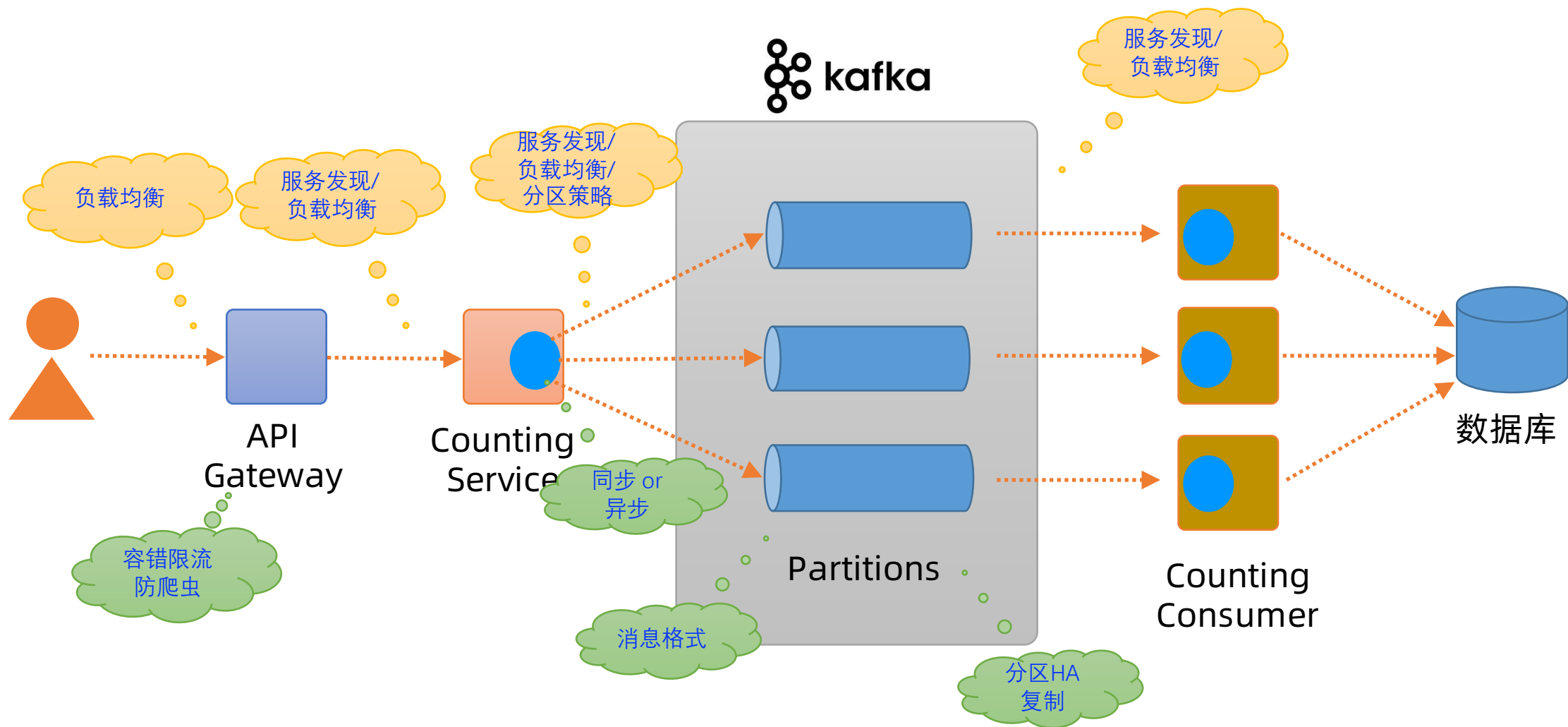
计数消费者(详细设计)



数据接收路径(Data Ingestion Path)



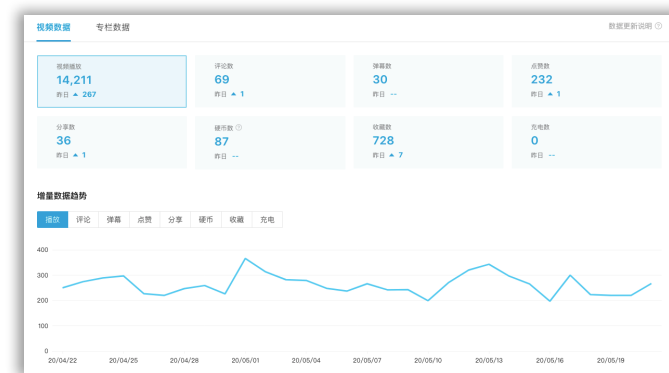
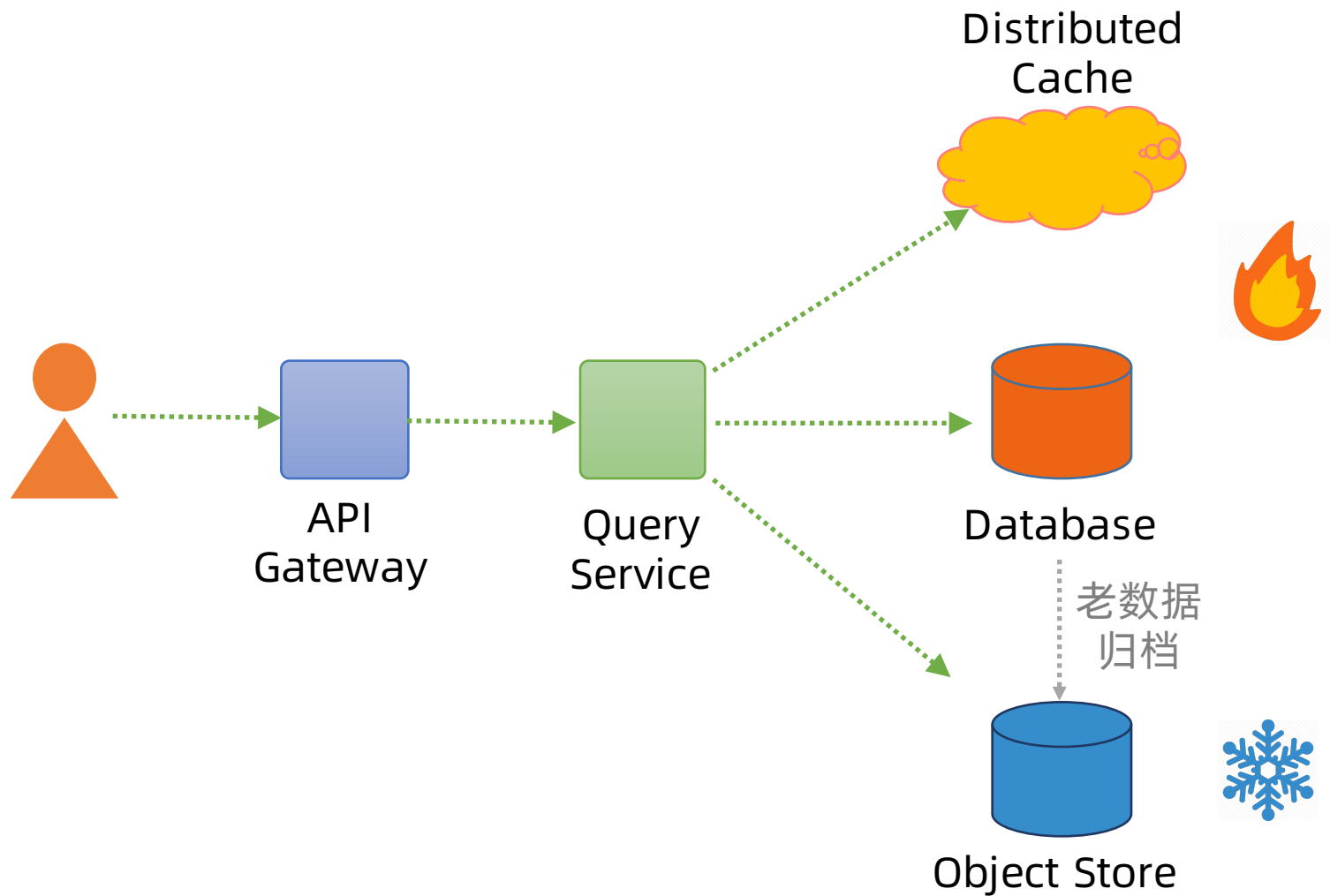
数据接收路径上的面试题



第 04 节

查询服务设计

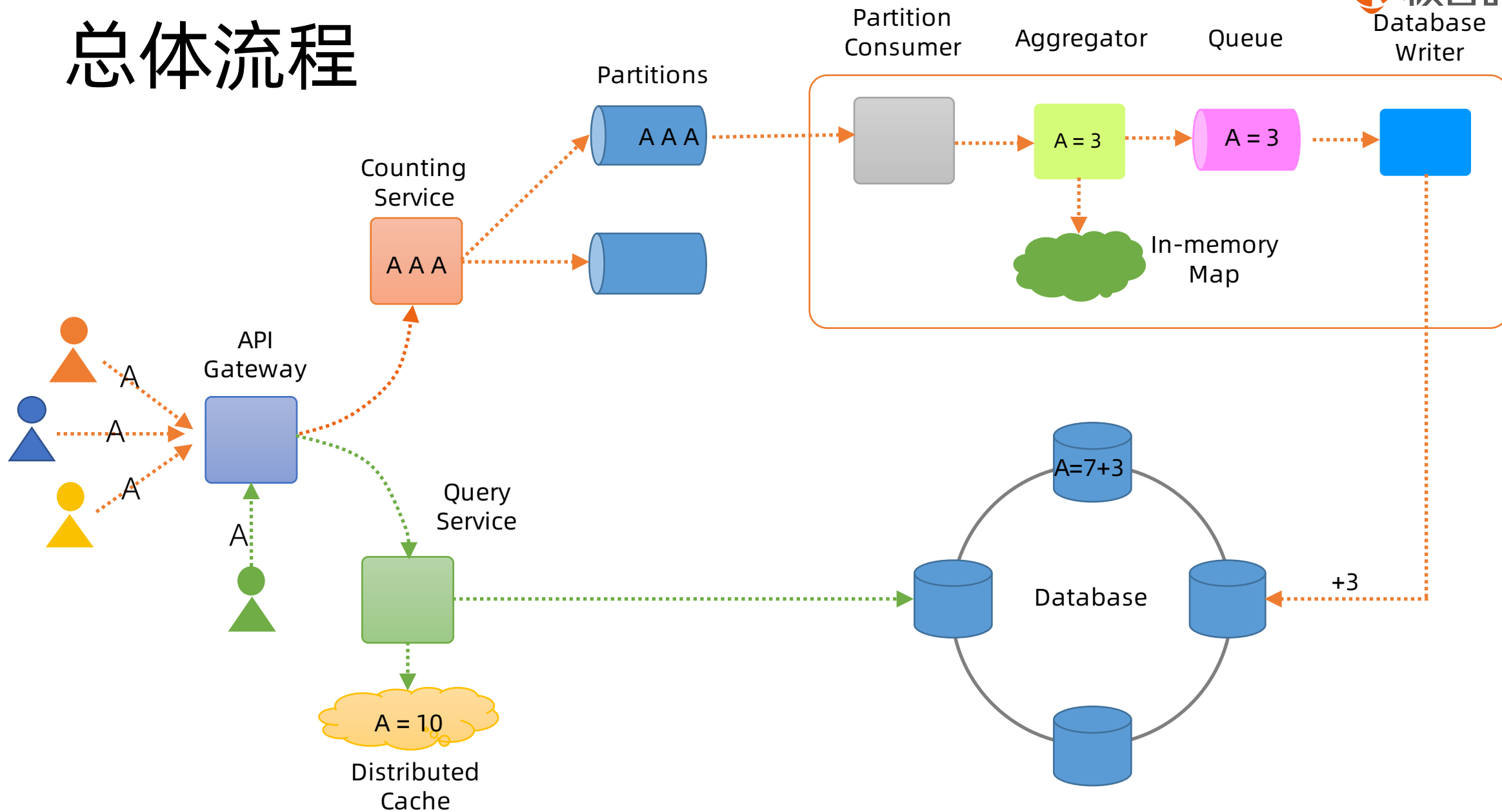
数据获取路径(Data Retrieval Path)



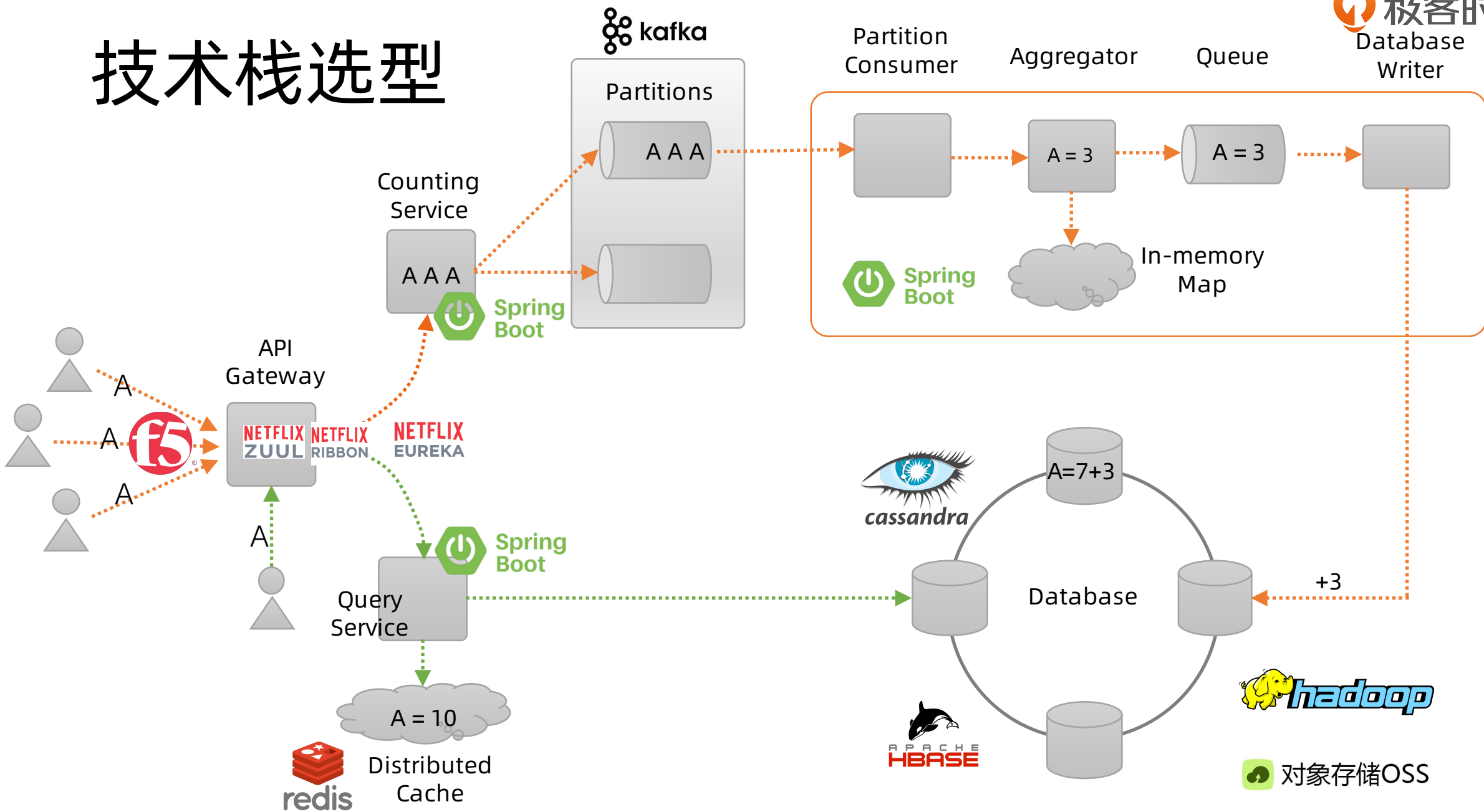
第 05 节

技术栈选型

总体流程



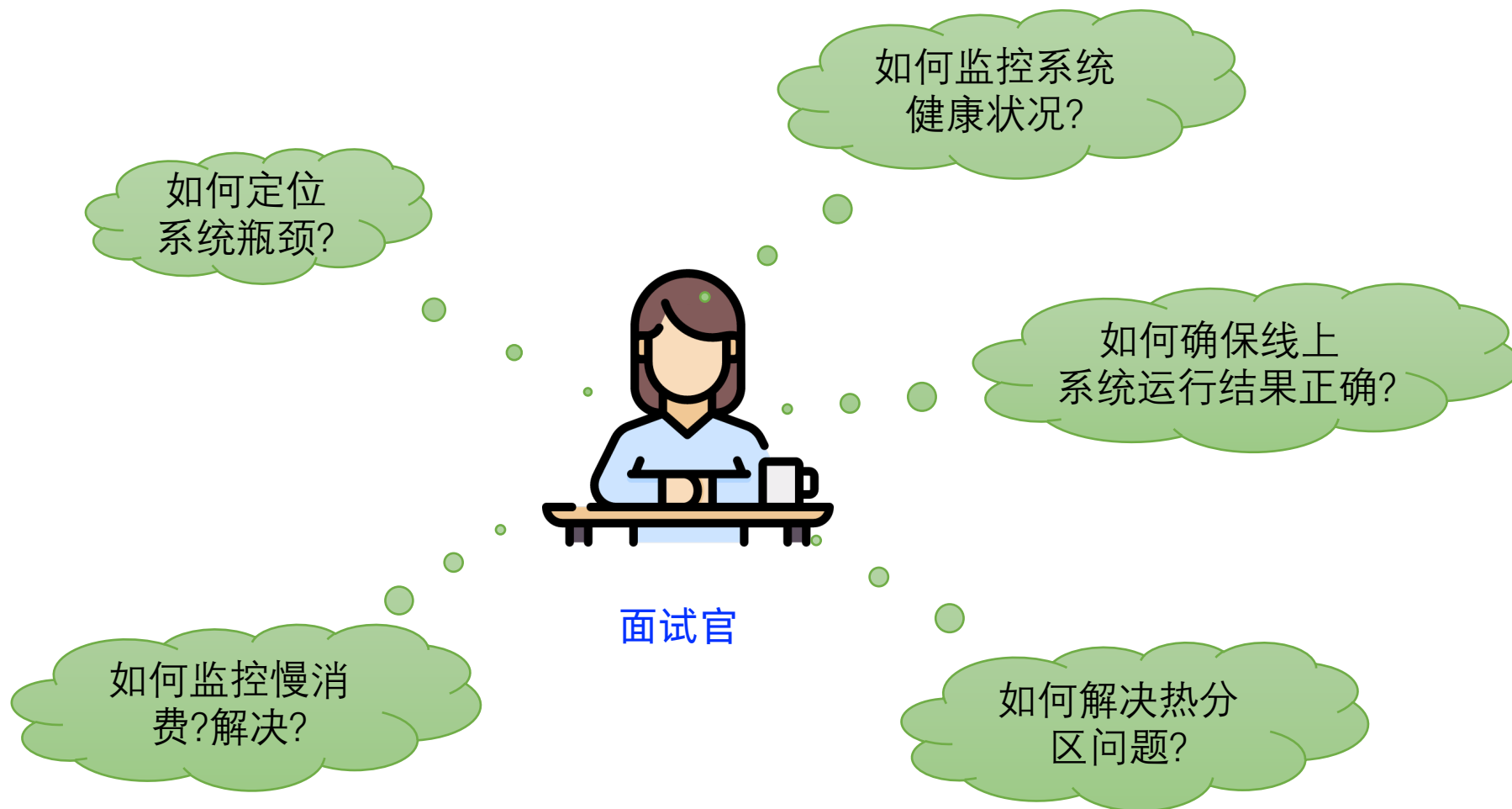
技术栈选型



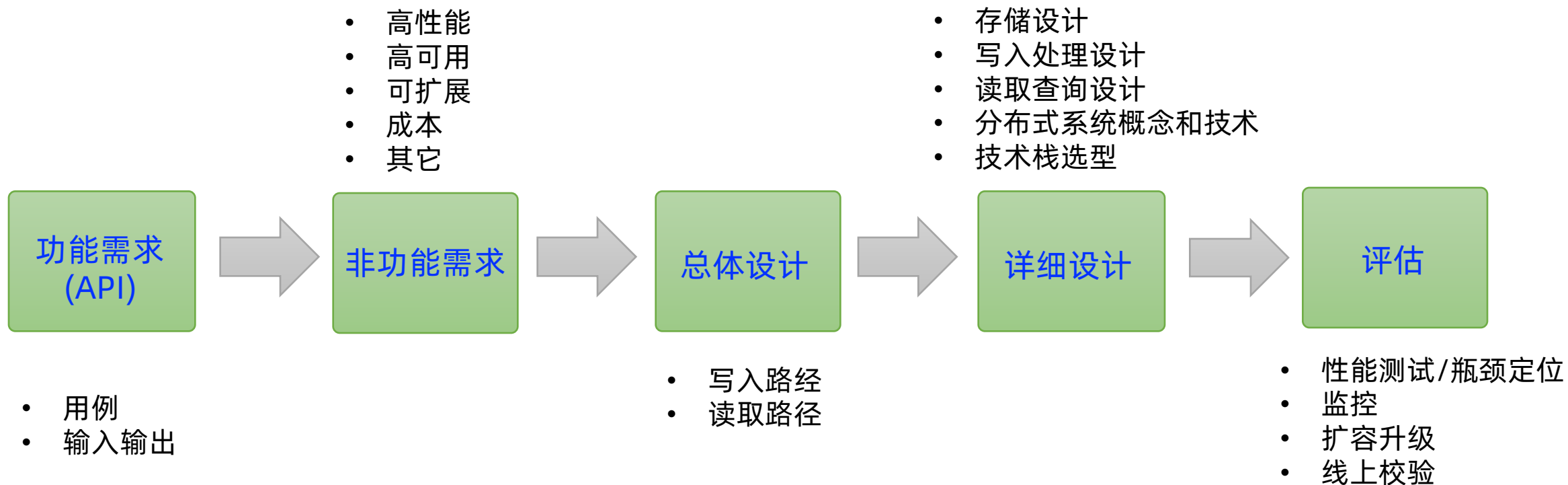
第 06 节

进一步考量和总结

更多面试问题



总结

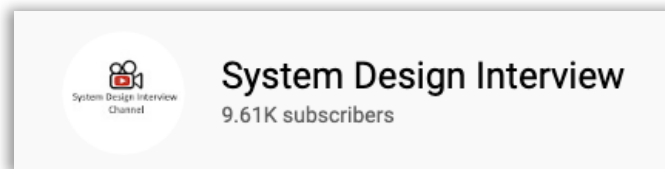


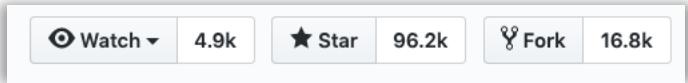
扩展

- 监控系统
- 欺诈检测系统
- 限流系统
- 推荐系统
- 今日热点
- 等等



参考



- System Design Interview – Step by Step Guide
 - <https://www.youtube.com/watch?v=bUHFg8CZFws>
- System Design Primer 
 - <https://github.com/donnemartin/system-design-primer>
- ConsistentHash
 - <https://github.com/Jaskey/ConsistentHash>



扫码试看/订阅

《分布式系统案例课》视频课程