成本优先的技术架构

Shopee海量商品系统的治理挑战和应对之策 / 张俊杰

极客邦科技 2024 年会议规划

促进软件开发及相关领域知识与创新的传播

查看会议

参会咨询

4月 — QCon 北京
全球软件开发大会
暨智能软件开发生态展
会议时间：已结束

5月 — AiCon 北京
全球人工智能开发与应用大会
暨大模型应用生态展
会议时间：已结束

6月 — ArchSummit 深圳
全球架构师峰会
暨智能软件开发生态展
会议时间：6月14–15日

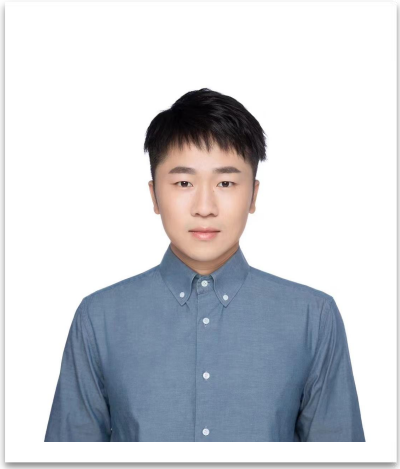
8月 — FCon 上海
全球金融科技大会
暨智能软件开发生态展
会议时间：8月16–17日

8月 — AiCon 上海
全球人工智能开发与应用大会
暨大模型应用生态展
会议时间：8月18–19日

10月 — QCon 上海
全球软件开发大会
暨智能软件开发生态展
会议时间：10月18–19日

12月 — ArchSummit 北京
全球架构师峰会
暨智能软件开发生态展
会议时间：12月20–21日

关于我



张俊杰

- 2019年加入Shopee
- 目前负责**商品系统**后端研发的工作

- 减肥并不难，关键在于找到正确的方法并实践。
- 减肥是一个持续的过程，我们可以借鉴他人的经验，避免走弯路，并增强信心。

目录

1 海量商品系统的背景和挑战

2 海量商品系统的应对 - 缓存篇

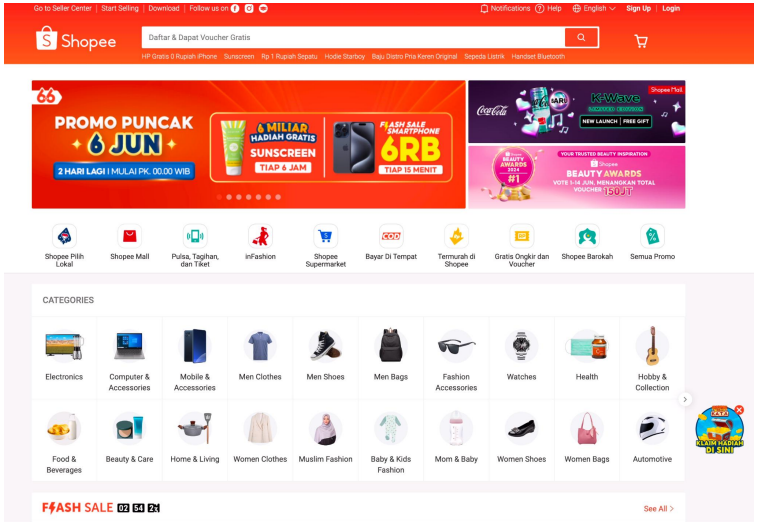
3 海量商品系统的应对 - 存储篇

4 成果和总结

海量商品系统的背景和挑战

海量商品治理 - 背景

Shopee是东南亚领航电商平台，业务范围覆盖新加坡、马来西亚、菲律宾、泰国、越南、巴西等十余个市场，为全球用户提供无缝、有趣且可靠的购物体验。



- 买卖双方通过电商平台促成商品的交易。

海量商品治理 - 现状

东南亚市场份额遥遥领先

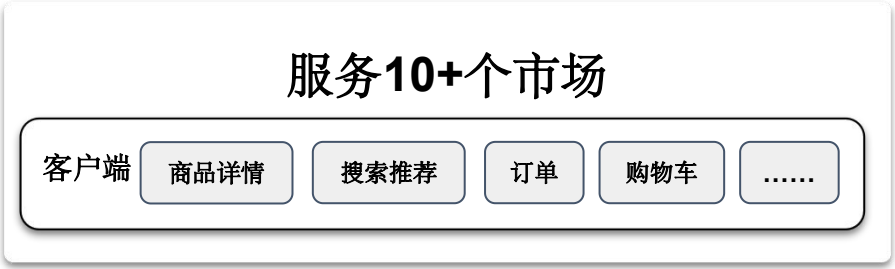
百亿级 商品　　**上千个** 商品字段

服务多市场　十余个市场，卖家体量大，潜在客户人群 **10+ 亿**
不同的文化，具有更多元的商品

玩法多样化　本土和跨境模式使得商品能够同时在多个市场销售。
新兴的全托管等多样化玩法不断吸引商家参与

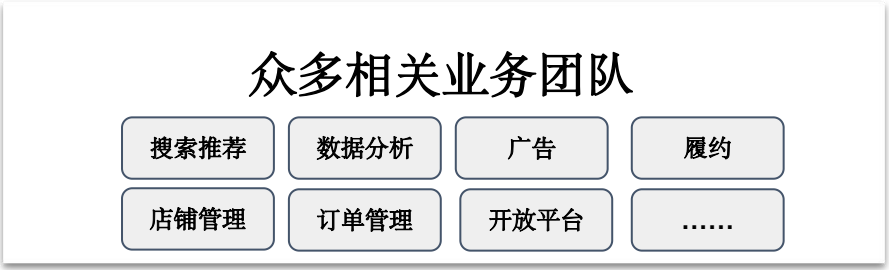
海量商品治理 - 现状

百万级 并发请求



商品系统覆盖了电商平台
绝大部分核心路径

- 商品曝光
- 商品浏览
- 用户下单
- 卖家发货



商品系统的流量来源复杂

- 卖家中心/开放平台/买家
- 内部系统流量
  (业务、算法、数据分析等)

海量商品治理 - 挑战

数据库服务器成本非常高昂

数据库 存储量大

- 百亿级商品存储，数据规模庞大
- 随着业务发展，每个商品数据的体积也不断膨胀

数据量 持续增长

- 每年都需要增加新的数据库来存储更多的数据

数据存储扩散比例大

- 除了从库，还需要考虑离线数据、索引数据等资源的需求

数据库从库 数量多

- 即使有缓存层存在，流量仍然会对数据库施加相当大的压力。

流量 持续增长

- 为了承载持续增加的流量，我们不断增加数据库的从库数量。

应对峰值流量

- 为了避免数据库过载，我们会预留足够的缓冲空间。

海量商品系统的应对 - 缓存篇

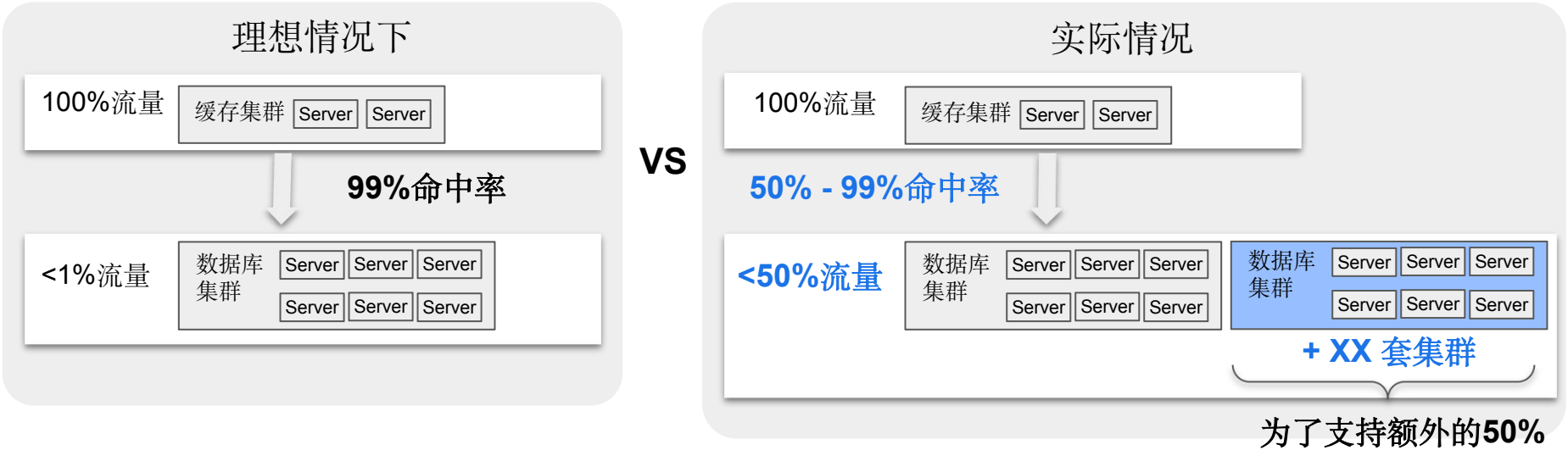
大家项目里面有用缓存?

普遍接受的事实

引入缓存服务可以提升系统整体性能

缓存服务会降低数据库成本吗?

缓存的正确使用对数据库成本影响很大



为了支持额外的**50%**流量，需要大量的额外资源。

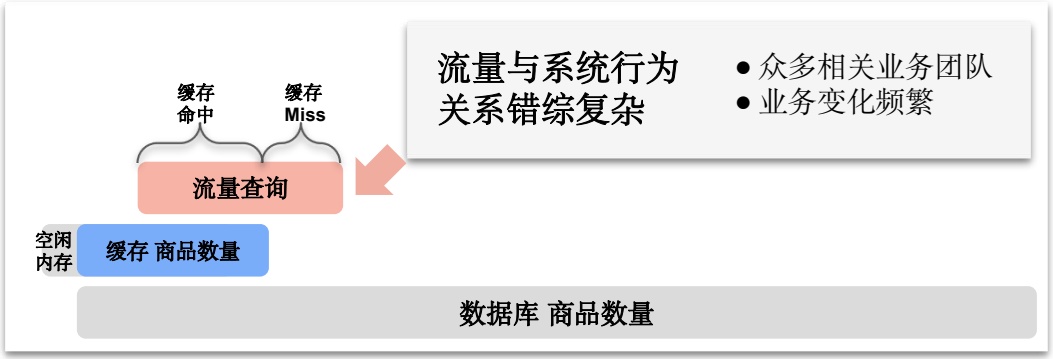
成本：需要更多的数据库来处理由于**缓存不命中**引起的数据库流量（用最坏情况做成本预算）

风险：如果缓存数据**大规模受到污染**，可能引发**系统故障**

从成本和性能的角度考虑，我们需要用缓存服务来支持大部分的读流量。

在这个架构中，缓存命中率对整体成本起到了决定性的作用。

选择最优TTL策略，提升命中率



流量与系统行为关系错综复杂

- 众多相关业务团队
- 业务变化频繁

分析每日缓存数据的特征

- 流量入口过多，非常复杂
- 数据特征不稳定

提高内存利用率

- 内存利用率过高会引起数据驱逐，数据驱逐是否引起系统性风险

对缓存周期做精细化控制

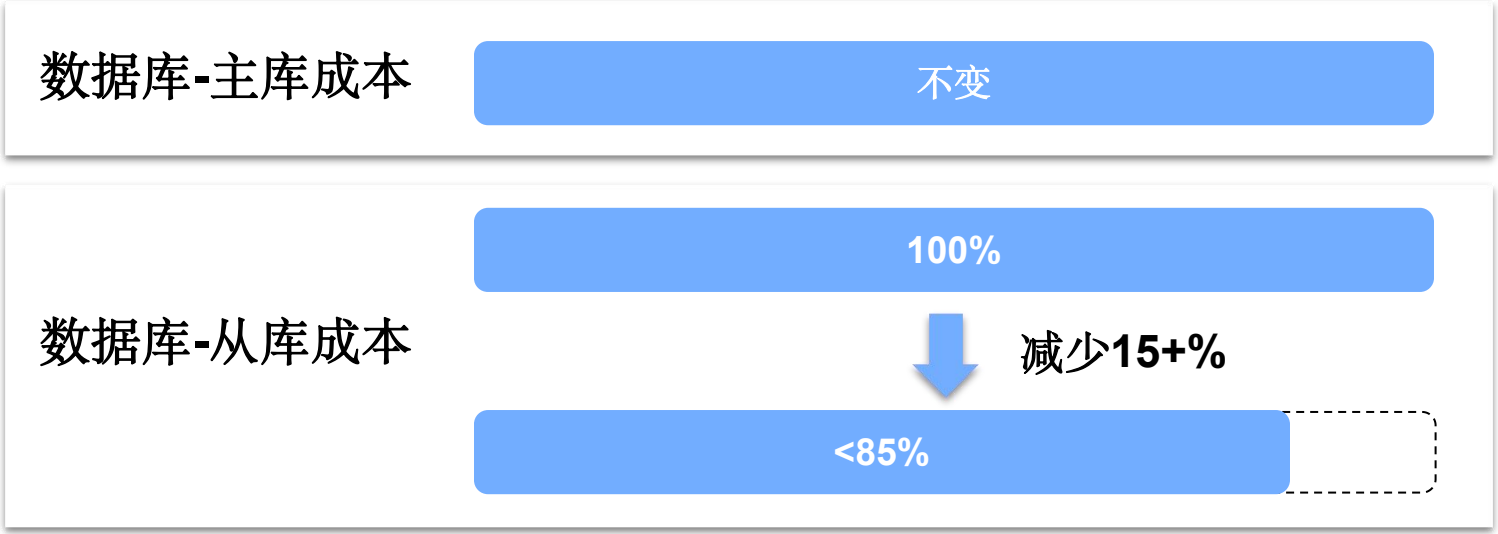
- 怎样的策略是有效的？

仿真验证

访真验证（控制变量）

01	筛选 TTL策略	固定TTL: 3h / 6h /9h…48h 请求次数 大于 某个阈值：2次 -> 24h 按请求次数进行缓存续期: 2..6 -> 3..24h …
02	选择决策因子	缓存命中率、缓存内存使用率 缓存的读写QPS **数据库压力指标** …
03	采集线上流量	用户行为是未知的，我们以线上实际流量为准进行策略验证，避免结果失真
04	搭建仿真环境	参考线上系统环境，保持架构和配置与线上系统一致。 编写脚本，用于执行仿真测试和报告输出
05	运行与调优	执行仿真模拟和输出报告，并优化参数持续迭代

治理进度



关于缓存，除了TTL以外，
还可以通过什么优化来减少数据库压力呢？

➡ 缓存更多的数据

在不扩容的情况下，减少数据体积可以缓存更多的数据

哪些方式可以减少数据体积

减少冗余和重复字段

成本低，收益确定性高

通过简单编码即可实现减少数据体积的目标，无需涉及复杂的技术设计。

数据压缩

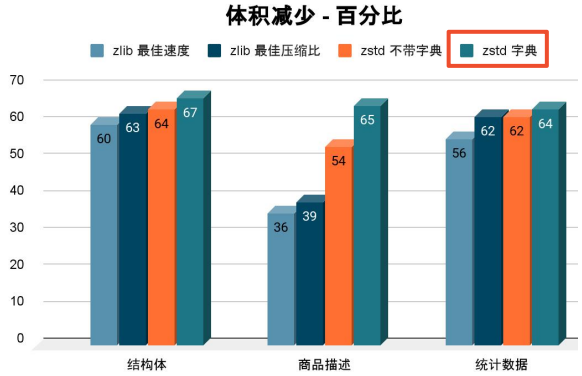
成本中等，收益具有不确定性

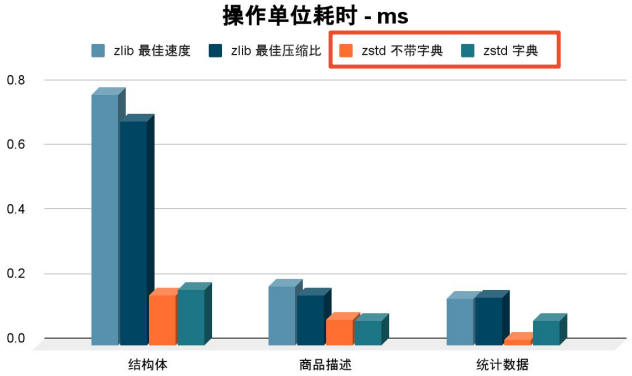
需要找到成本和收益的平衡点

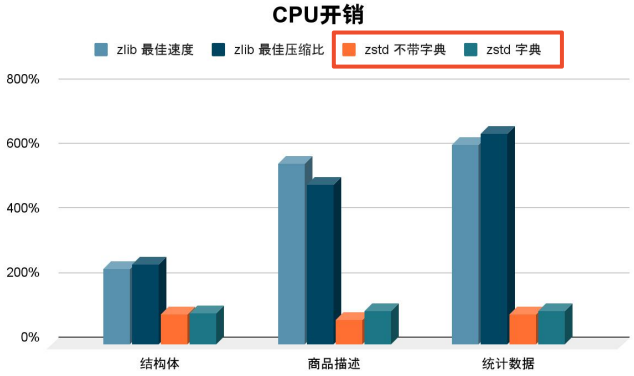
- 寻找业务匹配的压缩算法并进行性能验证
- 考虑算法升级后对系统的兼容性
- 针对不同的缓存数据类型（例如字符串、数字），进行独立验证

通过减小数据体积，可以增加缓存的容量，从而提高缓存命中率。

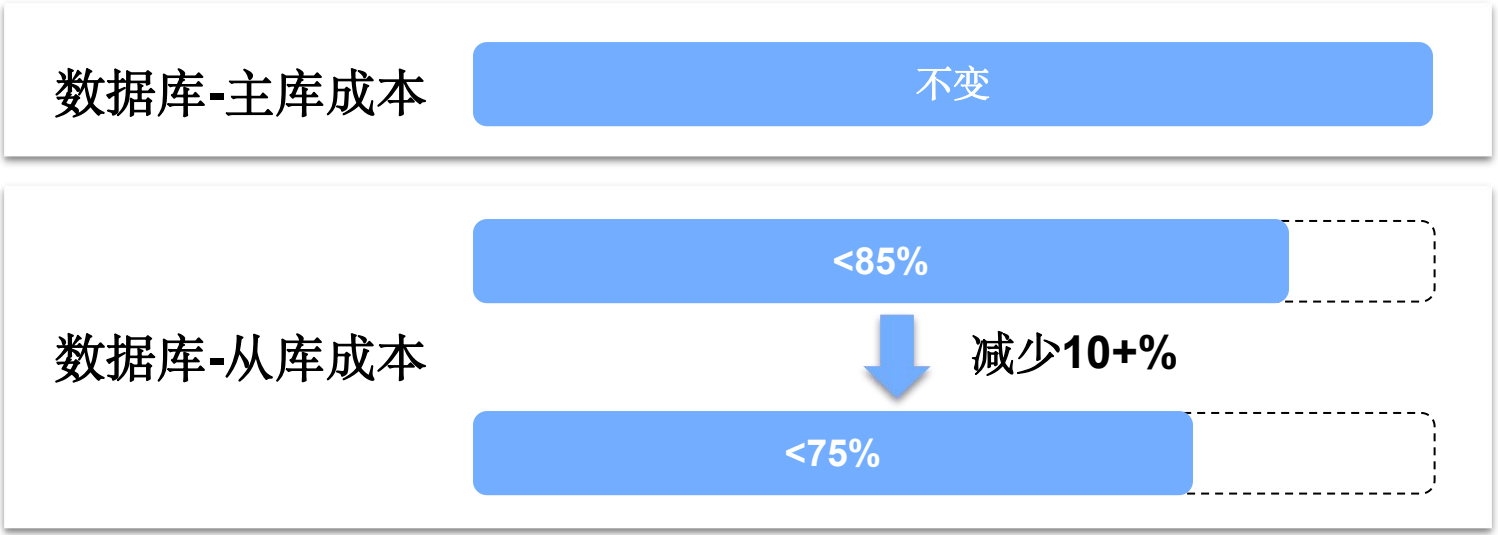
算法性能对比：压缩体积，操作耗时，CPU开销







治理进度

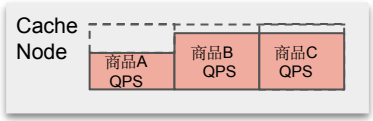


随着缓存服务承载的流量不断增加，风险也会相应增加，例如热点数据的问题。

热点数据问题会影响缓存成本吗？

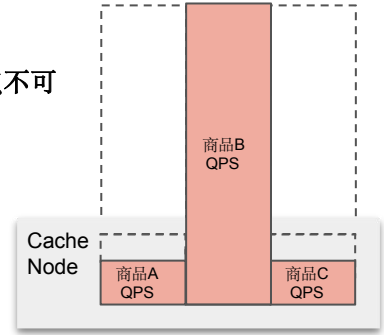
多级缓存减少缓存成本

●**正常场景:**
每个商品有不同的流量，但整体上看不会超过节点承载的水位



●**特殊场景:**

热点数据引起缓存节点不可用



为了解决热点数据导致的缓存节点不可用问题，我们可以考虑提前对缓存集群进行扩容。因为每个节点都可能发生相关的问题，会增加数倍的资源成本

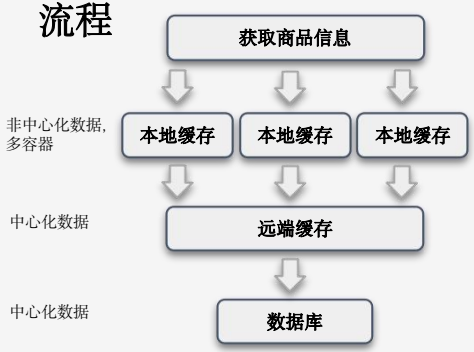
多级缓存: 利用服务的空闲内存，对热点数据做本地缓存

收益

降低系统性风险:
避免热点数据引起缓存节点宕机（CPU过载），用户将无法访问该节点的其他数据

提升性能和吞吐:
本地缓存的数据访问速度远超于远端缓存，数据传输速度是千级的差异。以及减少编解码的性能消耗。

流程



挑战

识别热点数据

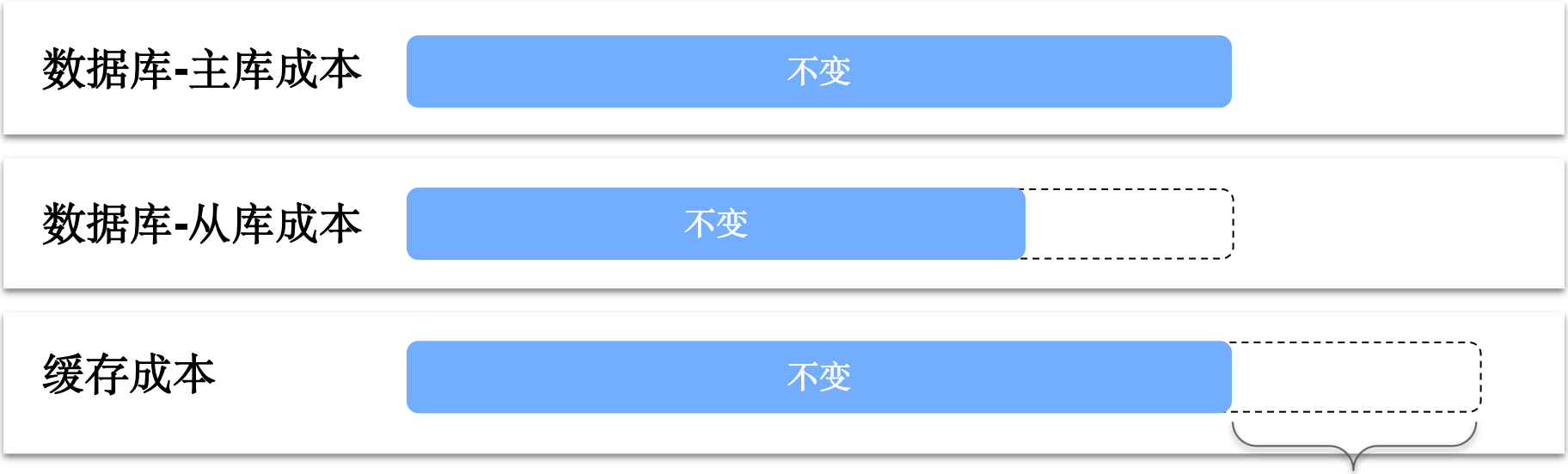
在时间窗口内
QPS超过设定的阈值

数据一致性和更新机制

广播通知&版本对比 vs 短TTL

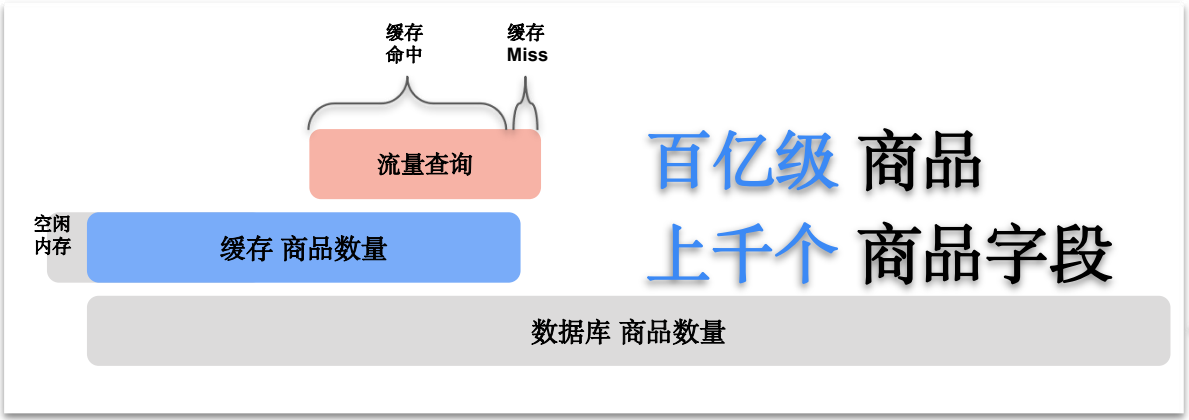
1.引入复杂机制的风险和成本
2.业务思考:
（1）特定时间区间的应用
（2）特定场景的应用

治理进度



海量商品系统的应对 - 存储篇

存储成本居高不下

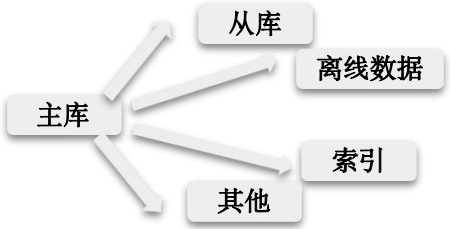


百亿级 商品

上千个 商品字段

数据库已经退化到以存储为主，但是成本还是很高

- 数据规模大
- 资源扩散比例大

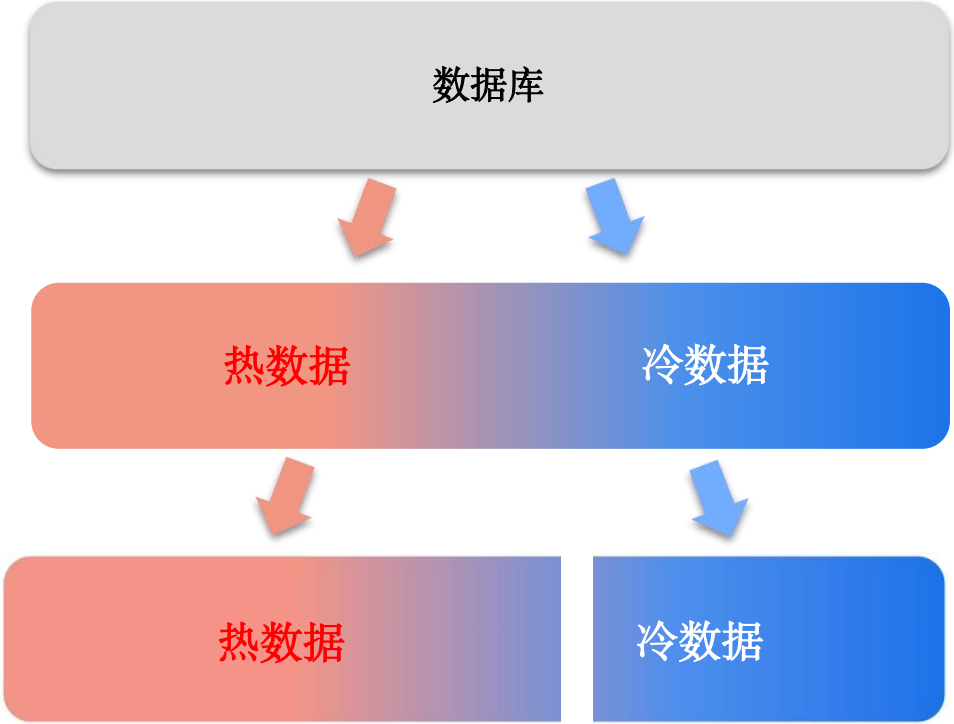


降低存储成本

冷热分离

把冷数据存储在更便宜的资源上

如何落地?



数据归档 - 冷热数据的分类定义

热数据

被在线系统
低频访问的数据

不会被在线系统
访问的数据

用户维度活跃的数据

- 与时间没有强相关的数据，通过流量比例区分，
  例如最近 n 月访问占比超过 x% 的数据。
- 具有时间属性，
  例如 最近n年的商品快照。

低频访问的数据

- 与时间非强相关的数据通过流量比例区分，
  例如最近 n 月访问占比不到 x% 的数据。
- 具有时间属性，距今时间久远，
  例如 n年前的商品快照。

用户主观删除的数据（软删）

* 业务上要求软删除
* 法务问题

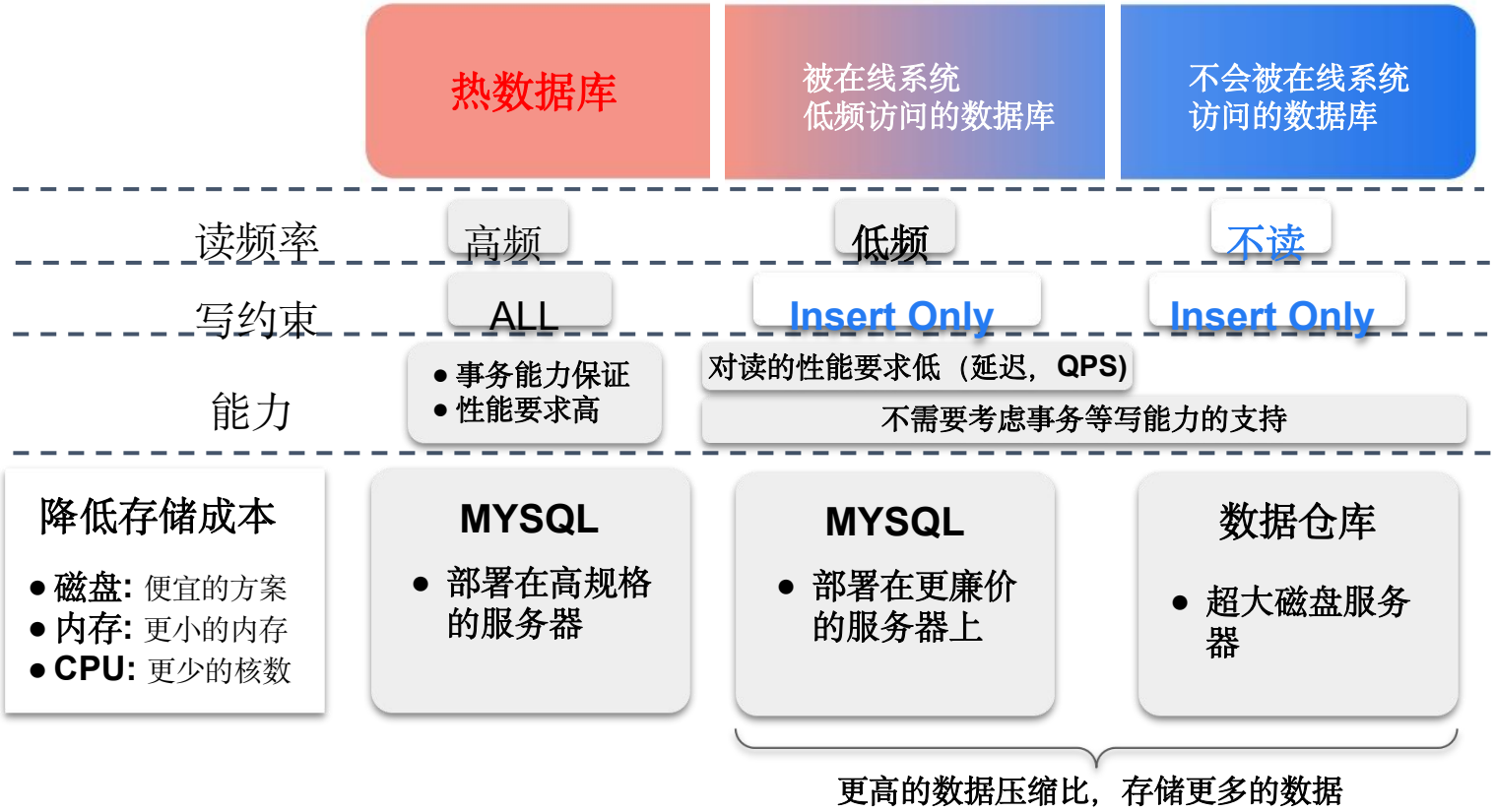
无用的日志数据

日志数据具有时间属性，结合业务上的诉求定义规则，例如n年前的商品变更记录。

没有价值且耗费大量资源的业务数据

对于一些长期不活跃的用户，例如几年都没有登录的卖家的商品数据。

数据归档 - 冷热数据存储成本的对比

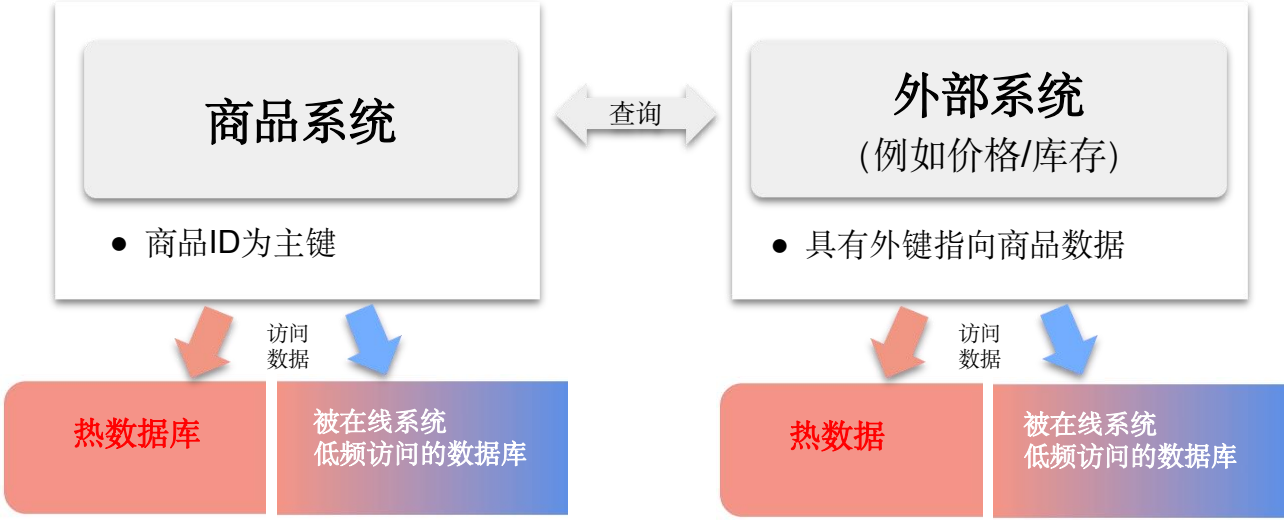


数据归档 - 可能出现的问题

数据实体往往是多张表以及多个数据库，
涉及不同的系统与团队

不同系统之间的兼容性处理

数据归档 - 兼容性（低频访问的冷数据）

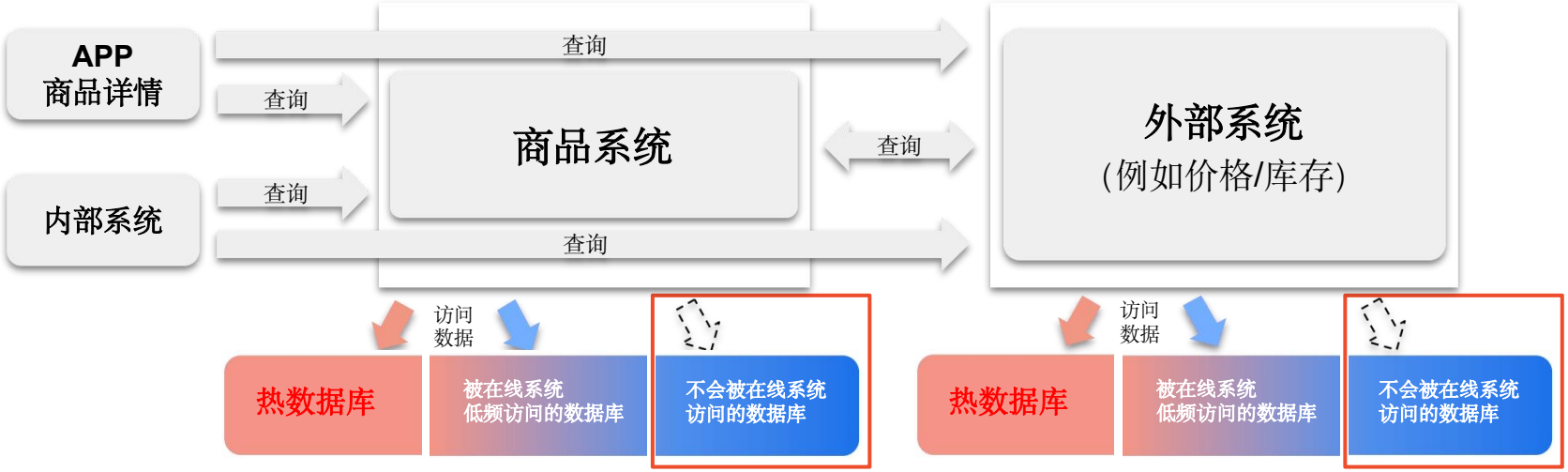


数据归档本质上是把数据从热数据库搬到冷数据库

相关系统需要兼容数据库物理删除消息

- **系统支持查询热库和冷库（一般先热后冷）**
- **支持冷库查询降级**

数据归档 - 兼容性（不被访问的冷数据）



用户体验
- APP不能crash
- 友好的提示

业务流程
- 业务应该正常运行，没有未知的异常

系统服务
- 系统不能出现未知异常
- 全局ID不能被复用
- 避免缓存穿透

数据归档 - 方法

01 **定义规则**
- 深刻理解数据
- 决策存储架构

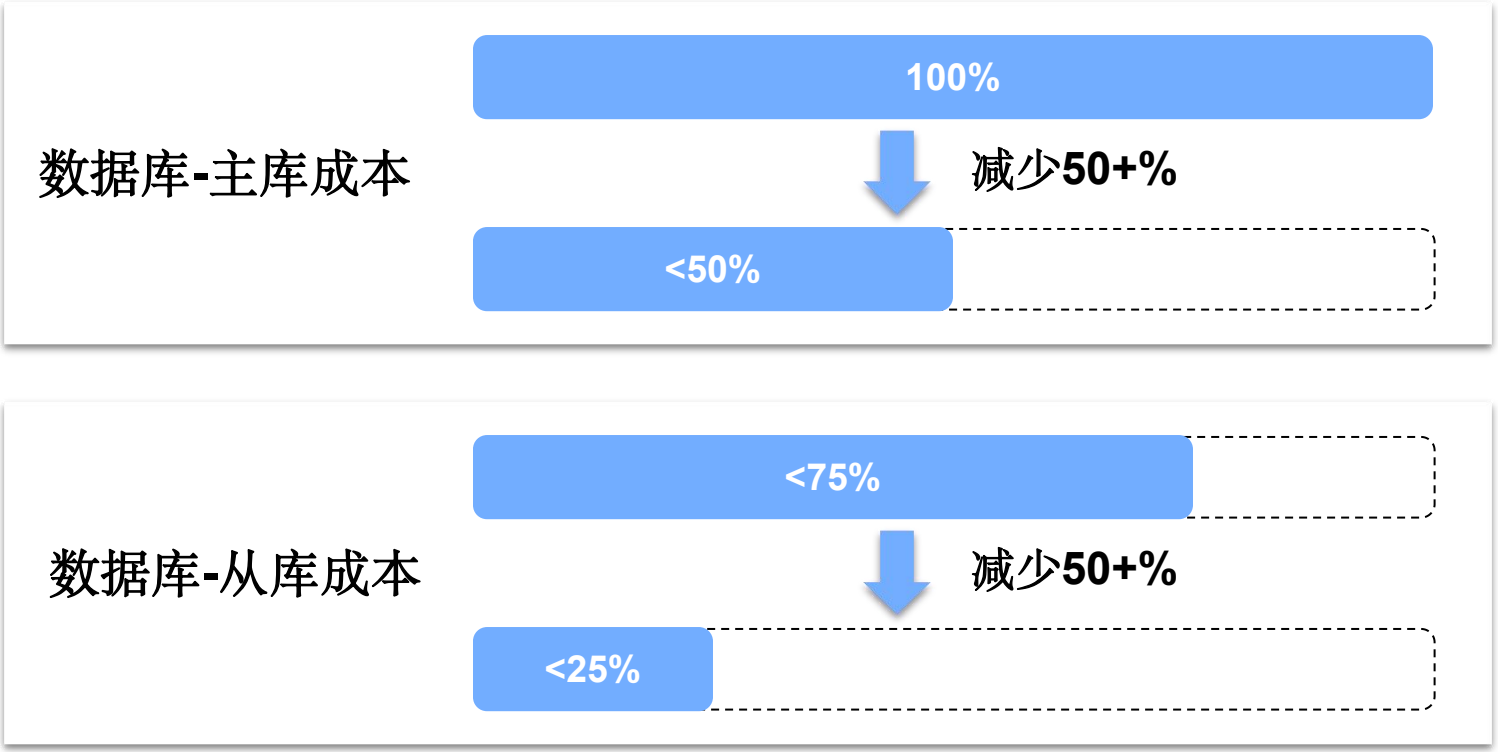
02 **系统兼容**
- 洞悉兼容问题
- 保障用户体验

03 **数据归档**
- 分离冷热数据
- 释放富余资源

业务法规风险

进度管理与沟通

存储应对 - 治理进度



成果与总结

成果与总结

减少 千万级人民币 服务器成本

减少支撑读流量的服务器成本

- 通过仿真验证选择最优TTL，数据库成本减少 **15+%**
- 通过减少缓存对象体积，增加最大缓存数量，数据库成本减少 **10+%**
- 通过多级缓存避免了缓存扩容引起的额外成本开销

减少海量数据存储的服务器成本

- 把“低频访问数据”分离到低规格服务器，把“不被访问的数据”分离到数据仓库，数据库成本减少 **50+%**

1.资源限制是客观存在的。
资源无法无限的增长，我们需要接受它，并**充分利用**已有的资源。

2.规则是可变的。
随着业务的发展，我们可以**重新审视**这些规则，并从成本优先的角度灵活调整它们。

Q&A

想一想，我该如何把这些技术应用在工作实践中？

THANKS