百度搜索中台检索系统的弹性化演进

——从存储到计算的极致弹性化

百度搜索中台检索架构负责人／ 刘伟

（排名不分先后）

“我们在 DIVE 全球基础软件创新大会上等你”

DIVE

全球基础软件创新大会

议/题/提/交

大/会/官/网

深入基础软件，打造新型数字底座

2021.11.26-27 / 北京·悠唐皇冠假日酒店

个人介绍

刘伟　　搜索中台检索系统技术负责人

2014年加入百度后，一直从事搜索架构相关工作

在集群管理、在离线混布、检索引擎、成本优化方面有非常多的积累和实践

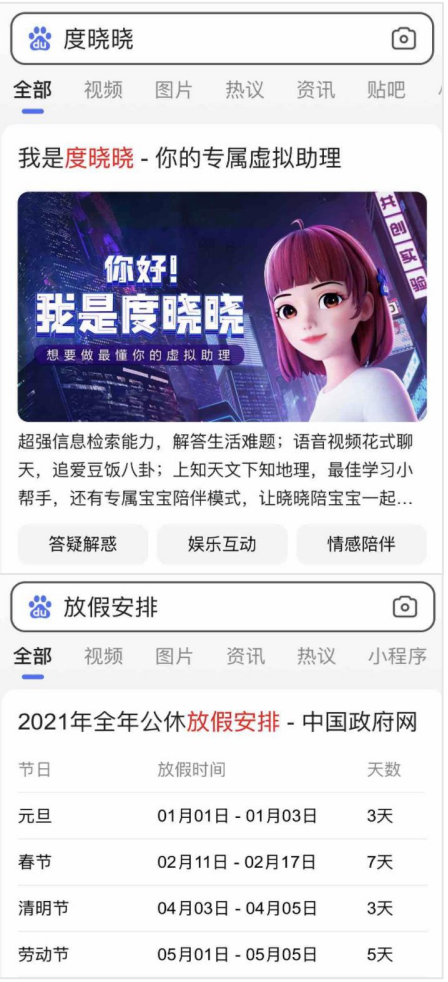
工作方向：检索系统智能化架构，超自动化交付，检索低代码

目录

百度搜索中台检索系统背景

百度搜索中台检索系统背景——业务背景

①配置化阿拉丁



②医疗阿拉丁

③有驾APP

④知了好学小程序

⑤爱采购独立站

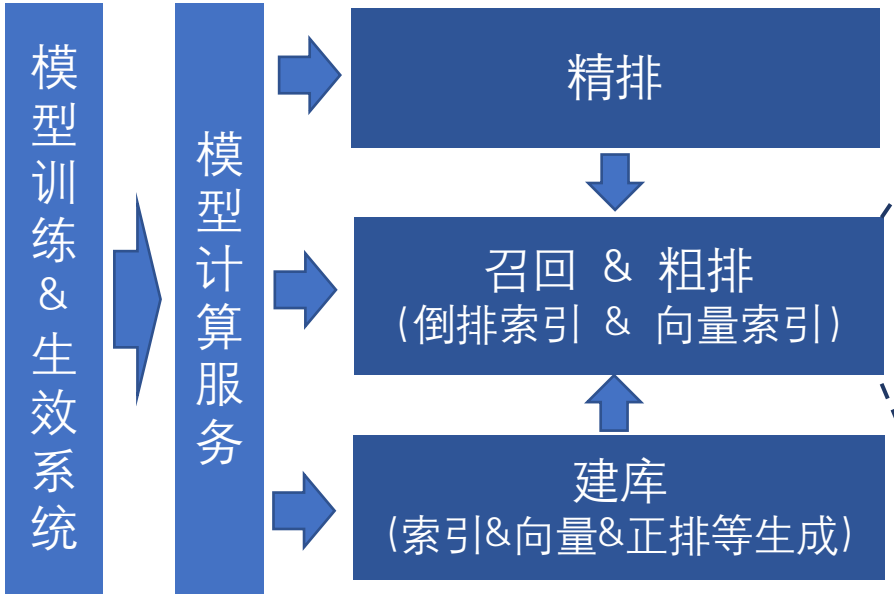
支撑多元业务场景，满足不同形态/规模/发展阶段的业务要求

百度搜索中台检索系统背景——系统特点与技术挑战

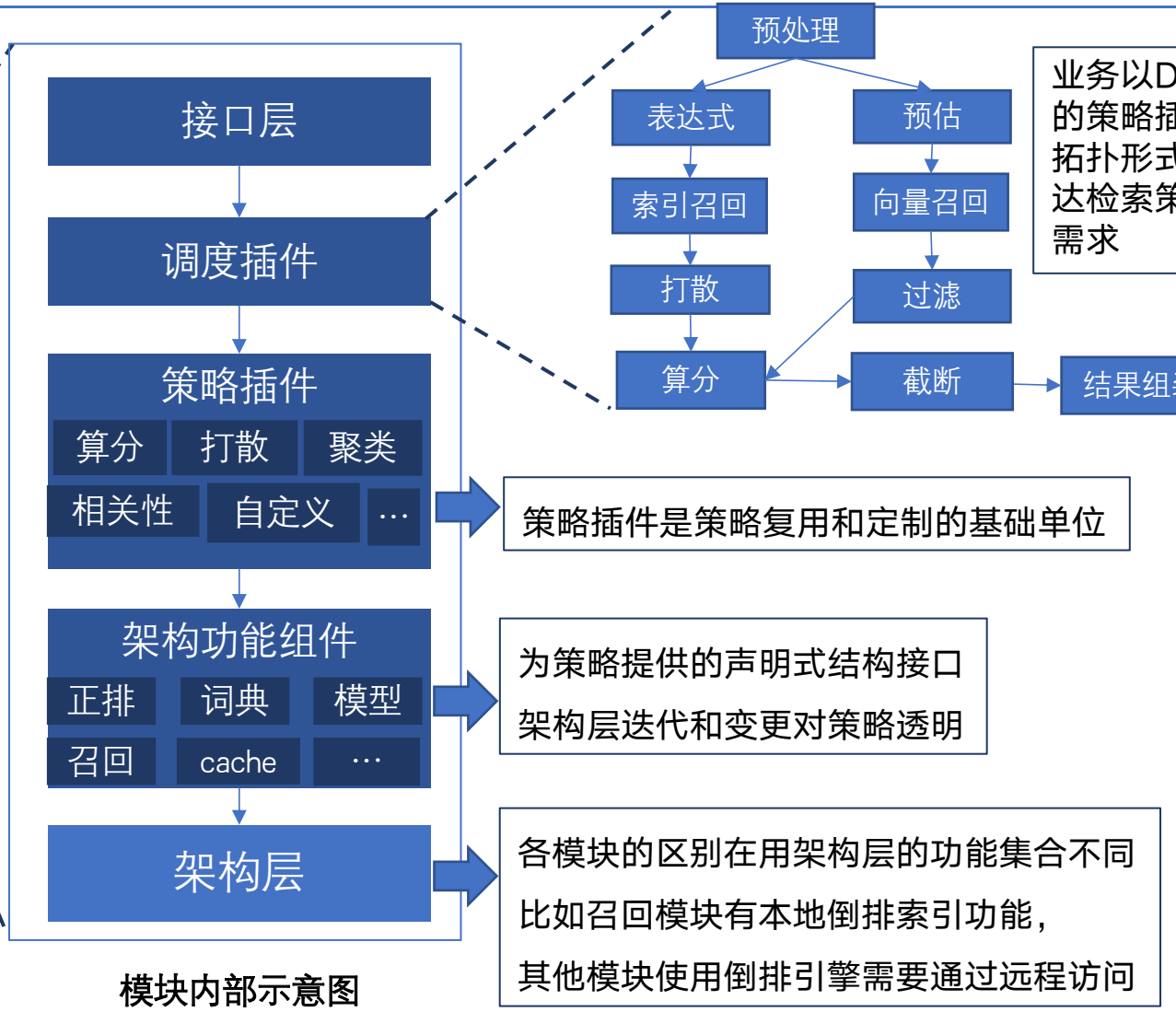
系统特点	技术挑战
多元异构业务，架构复杂度高 • 数十个产品线业务，从孵化到深耕各类业务 • 搜索本身足够复杂，叠加业务异构，复杂度非常高	**复杂系统导致维护成本飙升** • 几十套部署，异构复杂链路，需要专业人员维护；架构升级包袱重
流量和规模大，服务质量要求高 • 数百亿的天级入口流量，数十亿的天级建库量 • 运营卡片、热点事件会导致流量的剧烈波动 • 民生、商业类很多场景不能接受流量和时效性损失	**服务质量保障压力大** • 系统需要适配流量和数据量的快速变化 • 一旦出现问题，损失会很大
变更频繁，需求密集 • 周级别数百次业务变更和数十次的业务需求 • 更多变更是自动的，比如数据更新，流量成分变化	**交付周期不可控** • 大量的业务需求依赖架构排期支持 • 业务变更导致的计算和存储的变更无法快速满足

用一套架构同时满足多元异构大规模复杂业务是检索系统面临的核心挑战，通过不断增强系统的弹性化能力去自适应的满足不同业务的场景和需求是应对架构挑战的主要思路。

百度搜索中台检索系统背景——系统架构简介



检索系统示意图



模块内部示意图

业务迭代方式：1．DAG插件配置（大部分业务）； 2．DAG ＋ 自定义策略插件（少量业务）

弹性化架构演进

弹性化架构演进

阶段一：自动化运维保障服务质量

- 在云原生思路下，单体服务向云化发展，实现了自动化运维，保障系统服务质量

•阶段二：自动化交付提升交付效率

- 针对搜索场景的存储和计算特点，建设弹性化伸缩机制
- 在中台建设的思路下，实现自动化交付，大幅提升交付效率

•阶段三：自适应优化降低成本和保持架构合理性

- 在服务质量和交付效率的约束下，加入成本因子指导系统的弹性化调整
- 打破自动交付导致的局部最优陷阱问题，避免业务处于不合理的架构状态

弹性化架构演进

阶段一：自动化运维保障服务质量

自动化运维保障服务质量——故障和高频变更引发服务质量风险

场景1：检索流量波动

频繁的**业务运营**导致流量波动

突发热点事件会导致流量的剧烈波动

业务逻辑或策略的变更引起流量变化

场景2：建库流量和复杂度波动

B端业务，UGC场景数据**建库规模不定**

建库策略高频更新，**计算复杂度变化**

场景3：机器故障或长尾

机器规模快速增加，**故障次数**快速增加

长尾引发**数据不一致**，**检索拒绝**等问题

问题与分析

问题：

- 依赖频繁的人工干预和运维
- 每个业务场景不同，需要有经验的同学运维
- 一旦处理不及时持续的有损

典型的单体服务或『非弹性』架构的问题

解决思路

- 改造成**云化服务**，自动处理故障迁移等场景
- 建设根据**负载弹性化伸缩**的能力，兜住常态下的流量和数据量的波动

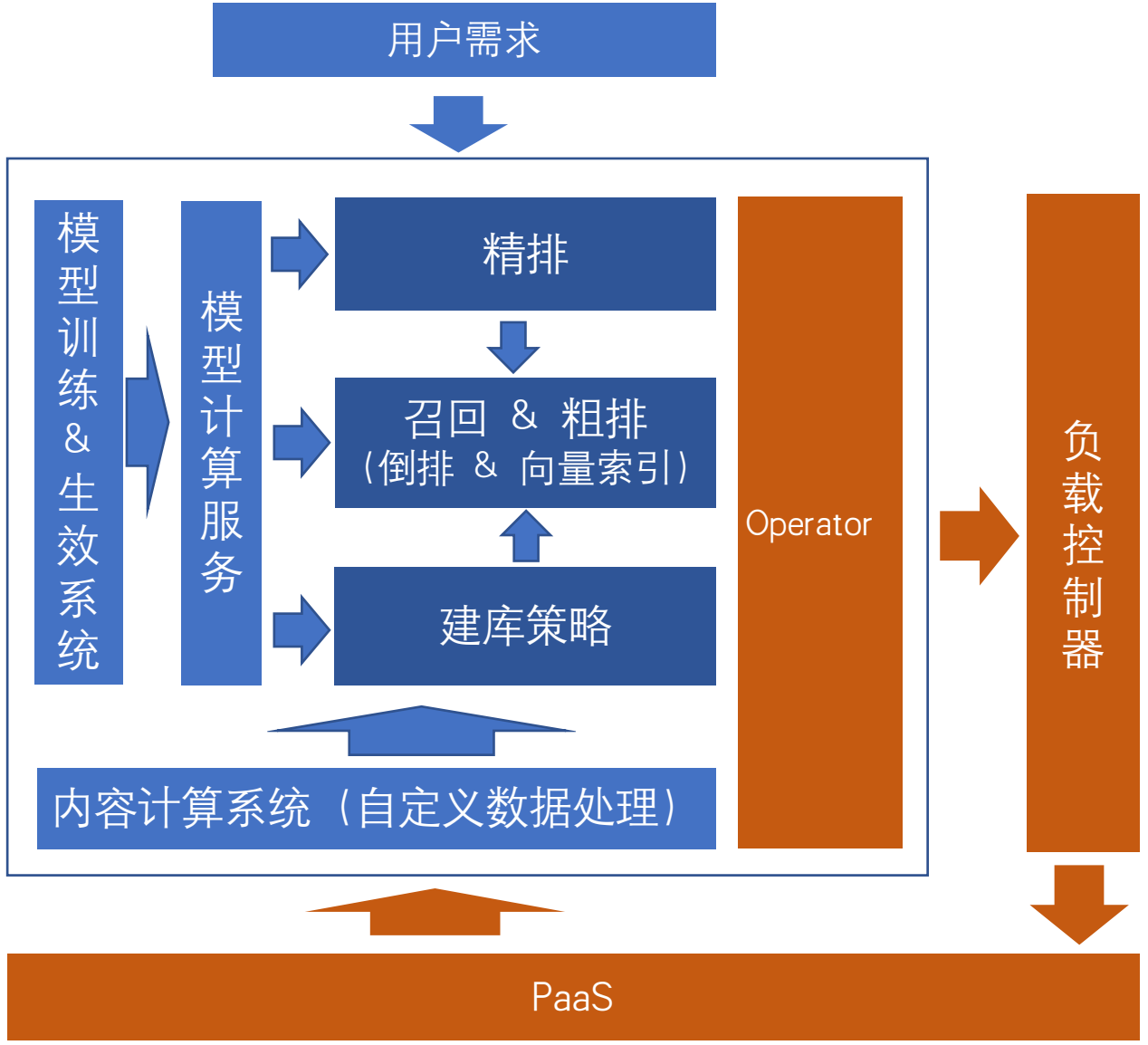
自动化运维保障服务质量——服务弹性化实现自动化运维

- **微服务架构建设**
  - 云原生改造，支持迁移和快速启动退场
  - 实现检索系统Operator，支持存储服务迁移
  - PaaS负责故障实例的自动迁移

- **负载控制器**
  - 根据资源占用情况进行资源quota调整
  - quota调整到云原生上限后进行扩缩容伸缩
  - 一些资源异常的实例可以主动进行驱逐

效果：

- **人工运维介入次数减少90%+**
- **时效性/稳定性case减少80%+**



弹性化架构演进

阶段二：自动化交付提升交付效率

自动化交付提升交付效率——需求和变更带来交付效率挑战

架构挑战：

- 很多高频系统变更场景负载控制器无法覆盖，交付效率低
- 业务高频变更，变更操作本身效率低，影响整体交付效率

解决思路：

- 扩展系统弹性化能力，实现高频业务变更场景的**自动化交付**

场景1：存储容量变更

背景：孵化到深耕的发展数据指数式增长

问题：数据调整需要精细化的控制，从人工评估到变更完成需要周级别

场景2：负载严重偏斜

背景：检索用户需求比较集中，大部分业务都会在负载上出现冷热的区分

问题：负载调度器难以处理负载偏斜的场景，往往存储或计算会被大量浪费

场景3：内容计算策略高频变更

背景：深耕业务高频迭代内容计算策略，需要去变更流式计算任务

问题：系统缺乏编排能力，依赖业务自己去配置流式拓扑等信息，需要小时级别，新同学需要天级别

场景4：时效性和负载不相关

背景：**部分**内容计算不是CPU密集型；部分业务进入系统之前有较长的通路

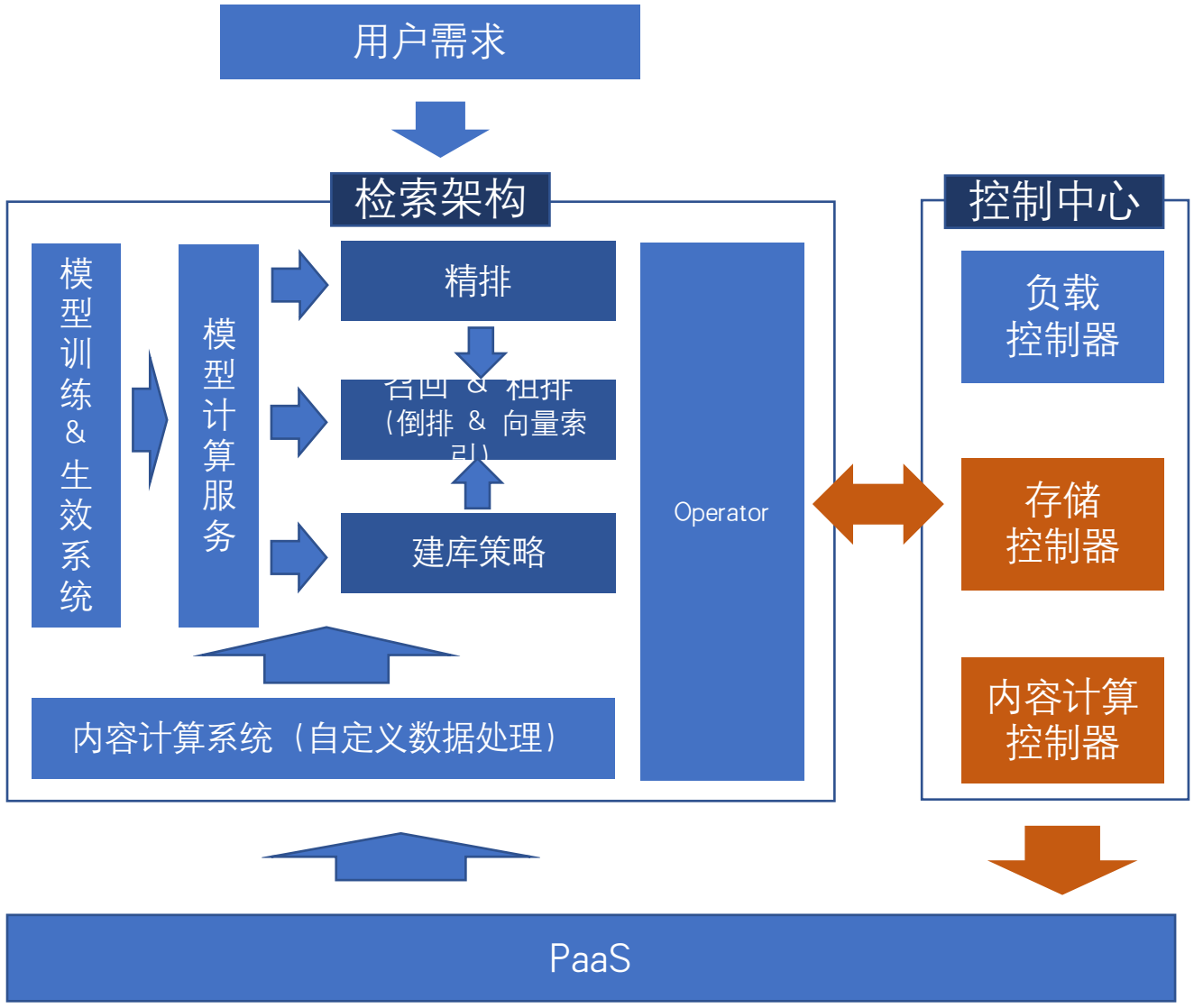
问题：时效性和负载不相关，负载控制器无法保证时效性，需要人工干预

自动化交付提升交付效率——存储和计算的弹性化实现自动化交付

自动化交付即在服务质量指标约束下，系统自动化满足业务需求和变更的能力

检索系统分为三类服务，分别建设弹性化机制：

- **无状态计算服务**
  - 以精排服务为代表
  - 受负载影响，由**负载控制器**负责弹性化满足
- **包含存储的服务**
  - 以倒排索引服务为代表
  - 受存储容量和计算负载同时影响，由**存储控制器**负责弹性化满足
- **近线和离线的计算**
  - 关注吞吐和时效性，由**内容计算控制器**负责弹性化满足

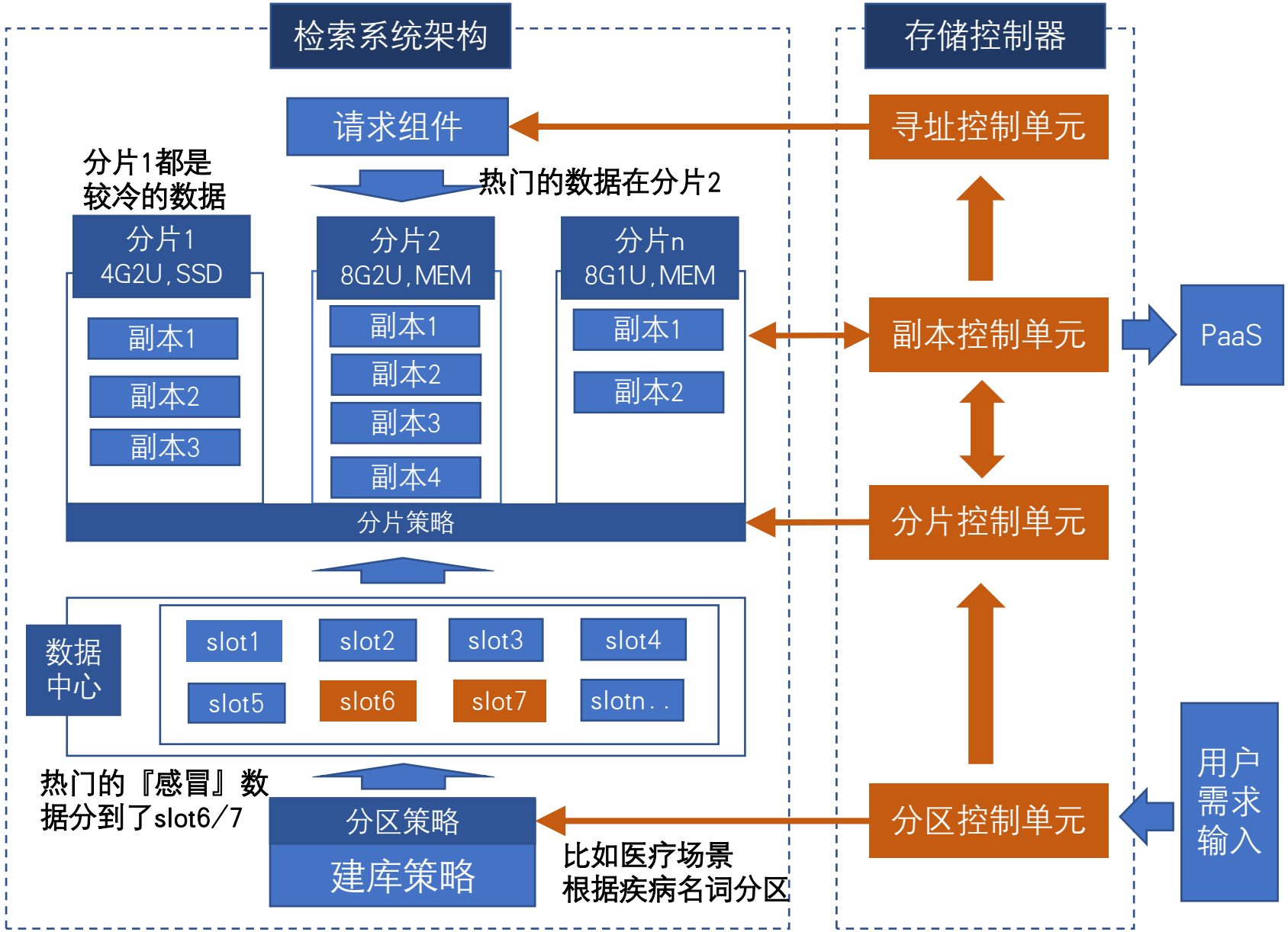


存储控制器——存储服务的自动化交付

目标：解决带有存储的服务的自动化交付问题

架构组成：

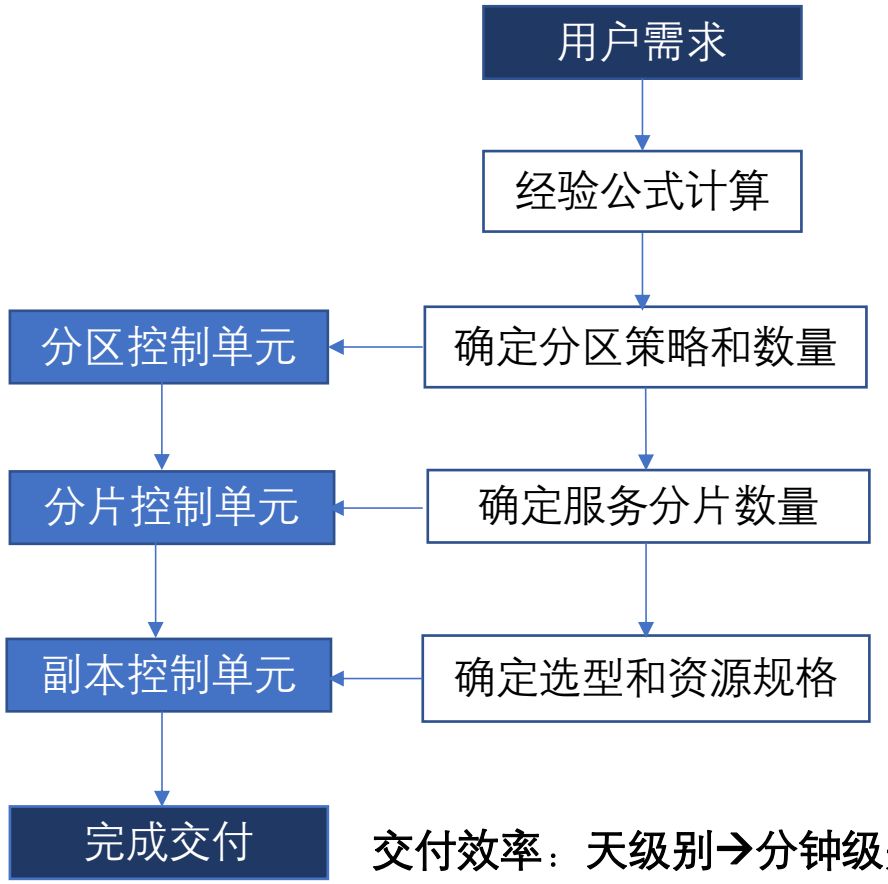
- 分区控制单元
- 分片控制单元
- 副本控制单元
- 寻址控制单元



分区(slot)：一组数据，数据管理的最小单位
分片(shard)：承载一组slot数据的服务
副本(replica)：分片中的一个实例
存储介质：SSD，内存，DISK等介质区分
服务规格：预定义的一系列容器大小

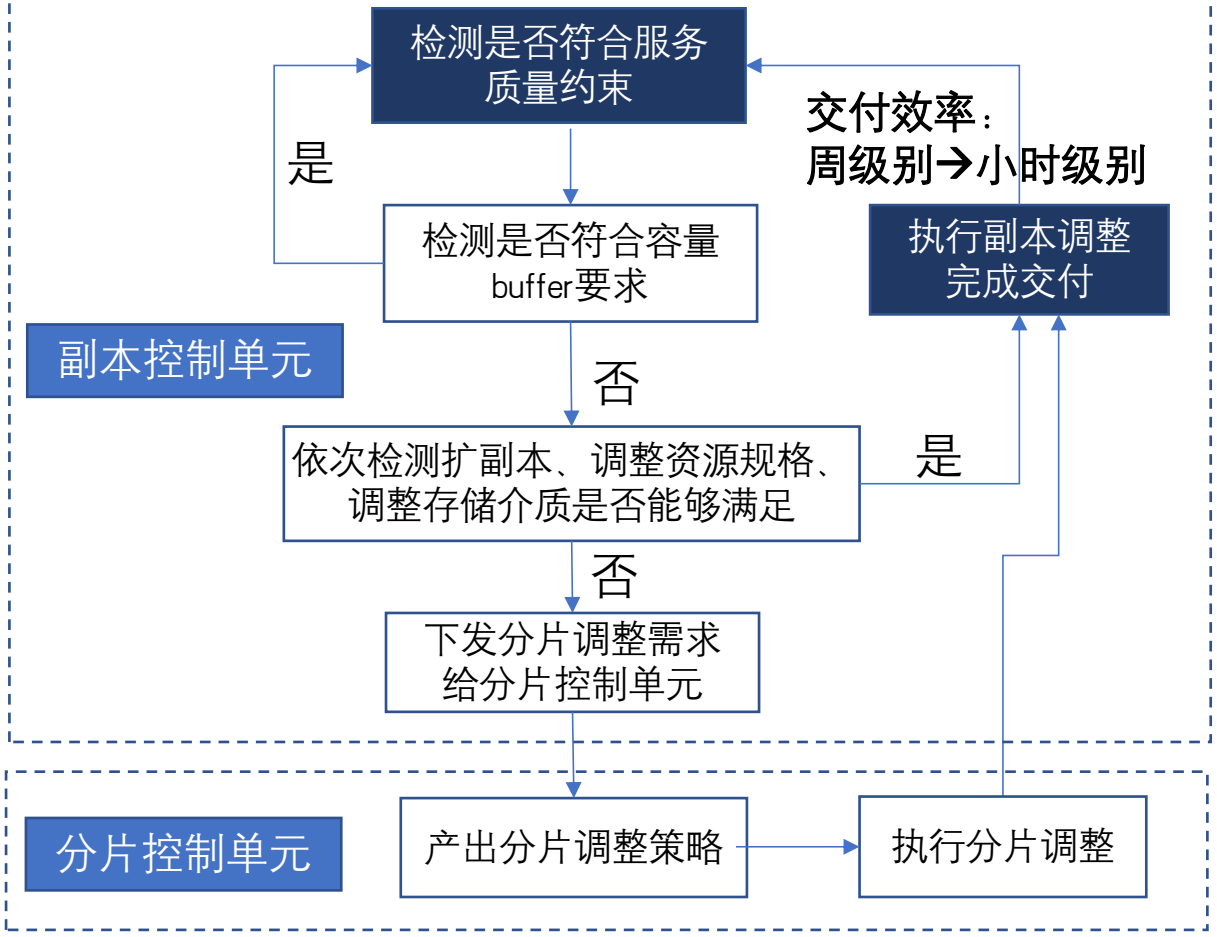
存储控制器——工作流程

接入（先验）流程：用户需求发起存储控制流程



交付效率：天级别→分钟级别

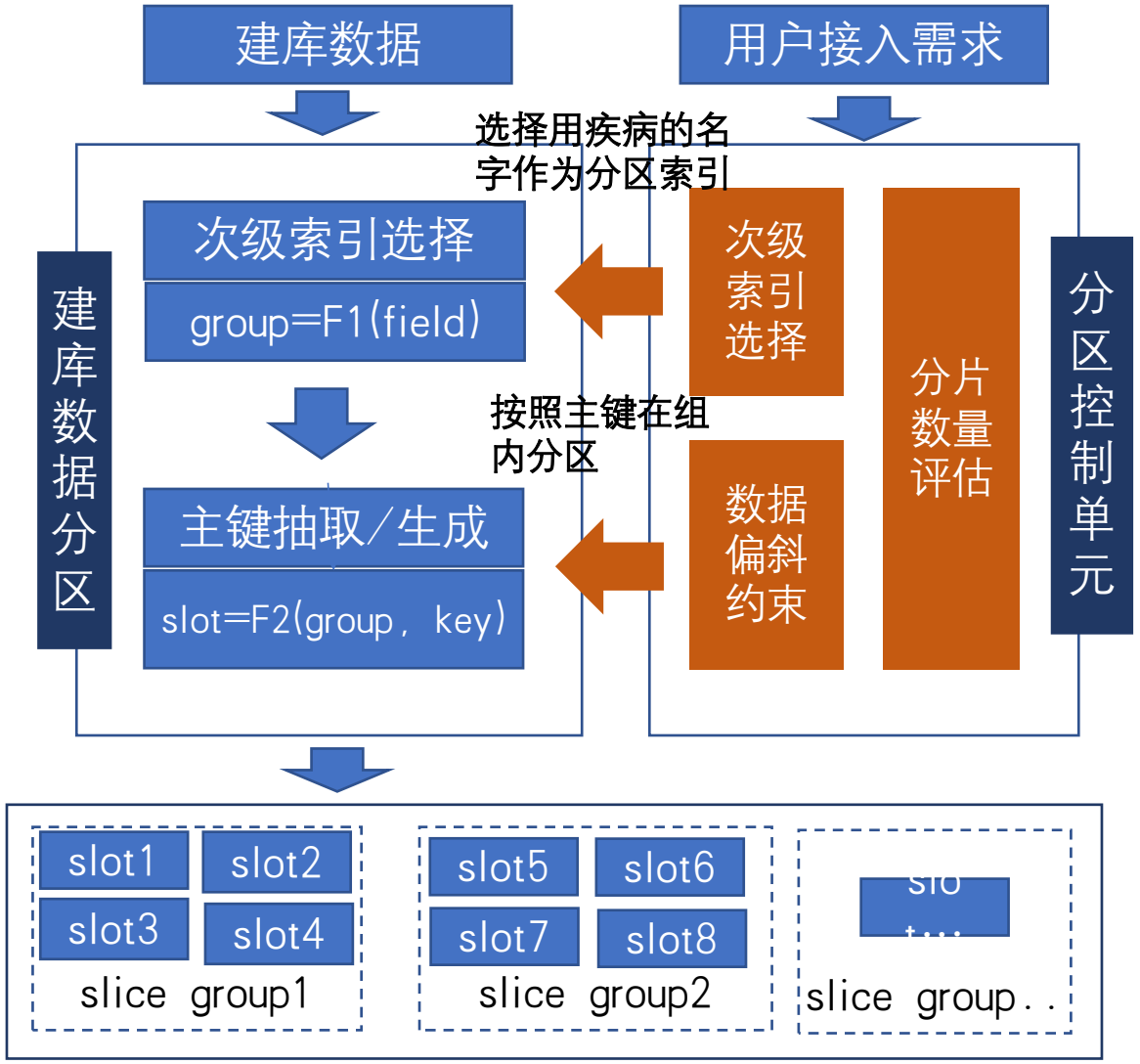
迭代（后验）流程：根据线上服务状态发起存储控制流程



存储控制器——数据分区策略

搜索场景主要是根据term表达式查询（召回）和根据主键去查询（筛选、排序、统计等场景），主要使用基于散列的一些分区策略

- **基于主键和次级索引的散列分区策略：**
  1. **根据容量预估确定分区数量**
     - 静态分区，不可调整
  2. **根据请求模式选择次级索引域（slice field）**
     - 冷热数据分离
     - 减少请求分区扇出
  3. **约束数据的偏斜（skew ratio）**
     - 为分片策略提供更好的灵活性
     - 简化再平衡策略的复杂度和成本



存储控制器——数据分片和副本控制策略

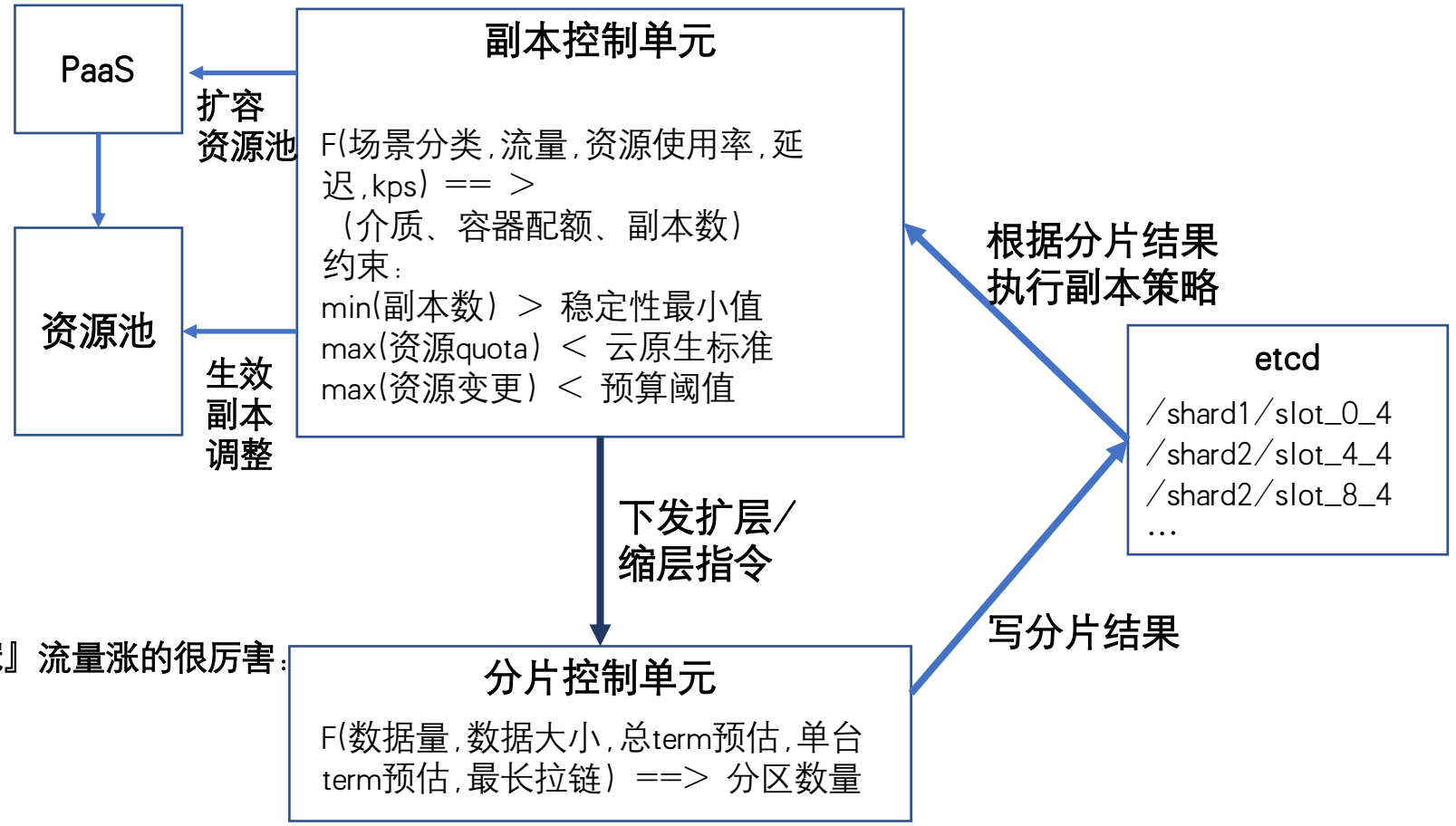
分片控制和副本控制在一起工作，他们组成了数据库控制系统中比较常见的balancer(再平衡控制器)

分片控制策略

- 假设每个分区数据都是均匀的
- 假设都通过标准容器和介质情况下

副本控制策略

- 管理资源池，保障资源供给
- 自动检测和补偿偏斜的工作负载
- 在稳定性约束下自动交付



数据分片和副本控制流程

比如之前『感冒』所在的分片流量更大，现在『新冠』流量涨的很厉害：

1、检测资源quota不足了

2、这个分片依次进行了quota调整/副本调整

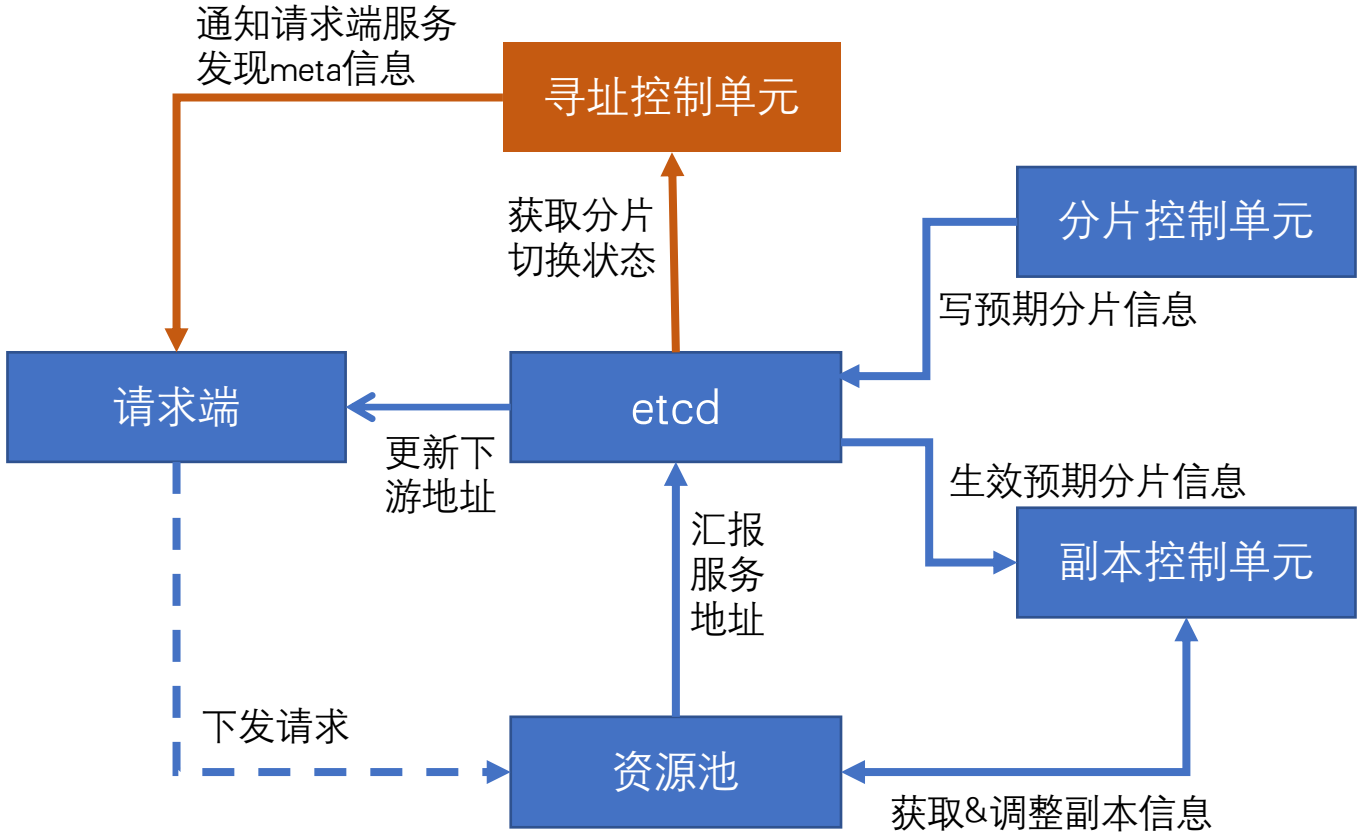
3、相关的数据量也增加了很多，做了横向的扩容

4、重走这个逻辑进行再平衡

存储控制器——寻址控制单元

寻址控制单元功能：

- **生效基于etcd的服务发现策略**
  - 通知请求端用于服务发现的meta信息
- **管理分片变更切换流程**
  - 每次分片变更会修改meta信息版本
  - 通过控制生效的meta版本管理分片切换流程



服务发现和寻址流程

存储控制器——效果示例

场景1：存储容量变更

背景：孵化到深耕的发展数据指数式增长

问题：数据调整需要精细化的控制，从人工评估到变更完成需要周级别

解决方案：

- 副本控制器发现存储容量不足的情况
- 分片控制器通过分区和服务的绑定关系来调整容量
  - 每个分片分区多了，整体分片减少，即缩容
  - 每个分片分区少了，整体分片增加，即扩容
- 扩缩分片后走副本控制器再均衡逻辑调整副本数和quota等信息

效果：小时级别完成典型的容量变更

场景2：负载严重偏斜

背景：检索用户需求比较集中，大部分业务都会在负载上出现冷热的区分

问题：负载调度器难以处理负载偏斜的场景，往往存储或计算会被大量浪费

解决方案：

- 支持按照数据属性做聚合，兼顾数据和计算负载偏斜（数据偏斜尽量小，计算偏斜尽量大）
- 支持不同层使用不同的资源配额，有不同的副本数，最小化成本支持计算负载偏斜

效果：可以支撑常见的负载偏斜场景

内容计算控制器——内容计算系统的自动化交付

目标：解决内容计算系统自动化交付问题

场景3：内容计算策略高频变更

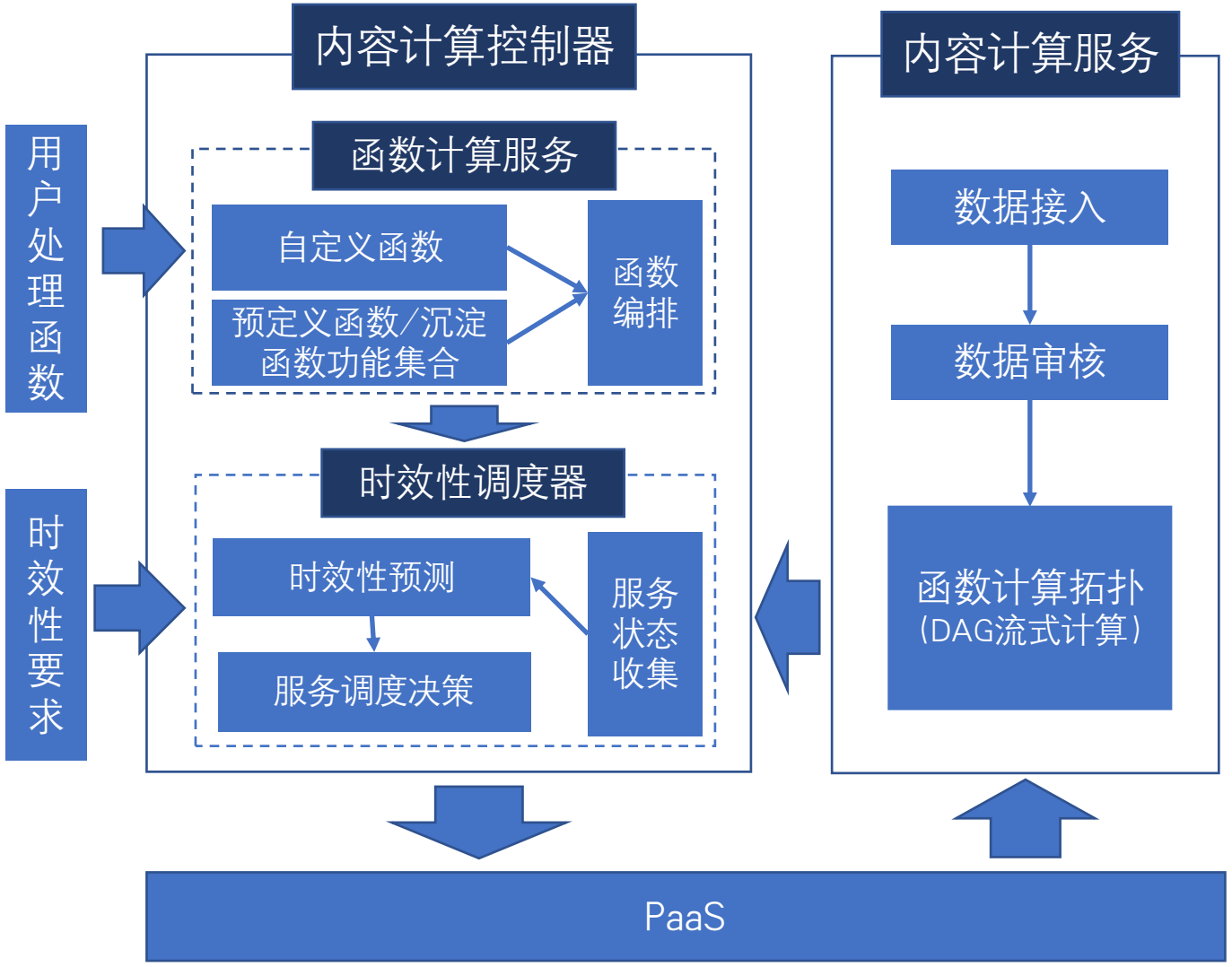
问题： 系统缺乏编排能力，依赖业务自己去配置流式拓扑等信息，需要小时级别，新同学天级别

- **用户变更/接入效率低下的问题**
  - 建设**函数计算服务**，用户聚焦业务逻辑
  - 编排用户函数，流式计算对业务透明

场景4：时效性和负载不相关

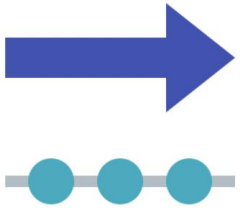
问题：时效性和负载不相关，负载控制器无法保证时效性，需要人工干预

- **时效性无法自动满足的问题**
  - 打造**时效性调度器**，以时效性指标作为弹性化调度的重要目标



内容计算控制器——函数计算服务

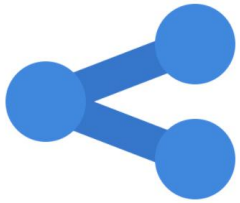
vs-lambda Air



通过模板快速创建函数数据流，在线写代码进行调试，简单方便高效接入，屏蔽底层细节，Serverless由此开始。五分钟快速入门

通过模板创建 →

vs-lambda Pro



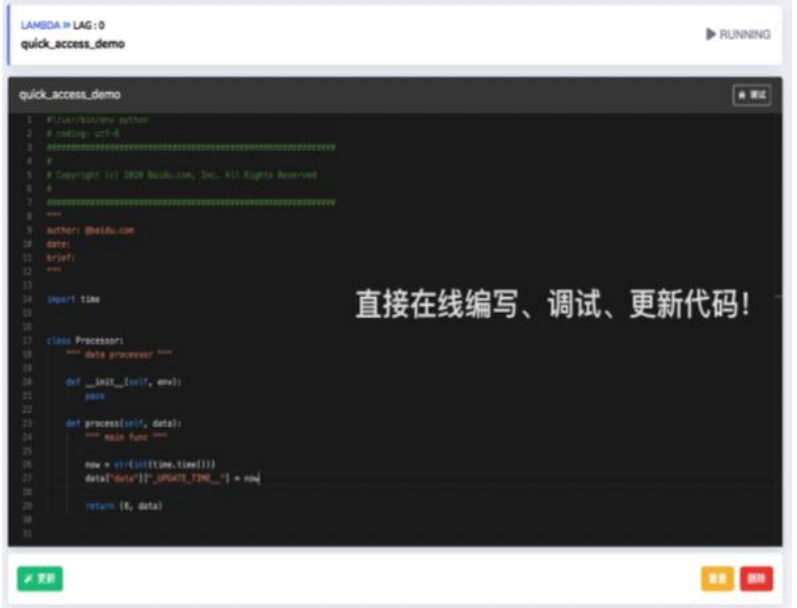
通过设计拓扑，完成更为复杂的数据处理任务，多种通用算子，灵活强力。通过例子了解更多

通过拓扑创建 →

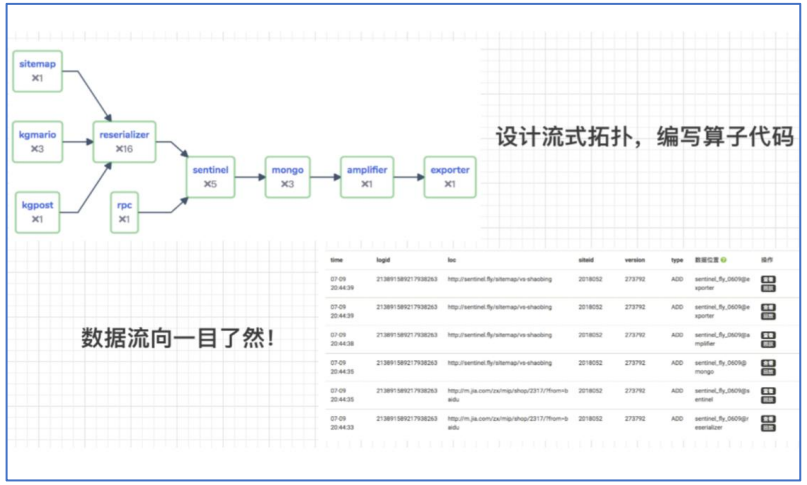
vs-Timer

创建定时函数，定时从数据源，比如Mongo，FTP，HDFS，MySQL等导入数据到数据流或者导出数据进行建库

创建定时函数 →



- vs—lambda：用户仅需关注业务逻辑，小时级接入，分钟级变更生效
- vs—timer：内容触发，极速数据接入
- 大量预定义函数/沉淀函数功能
  - 以API的方式在自定义函数中使用
  - 以计算函数的方式在拓扑中引用

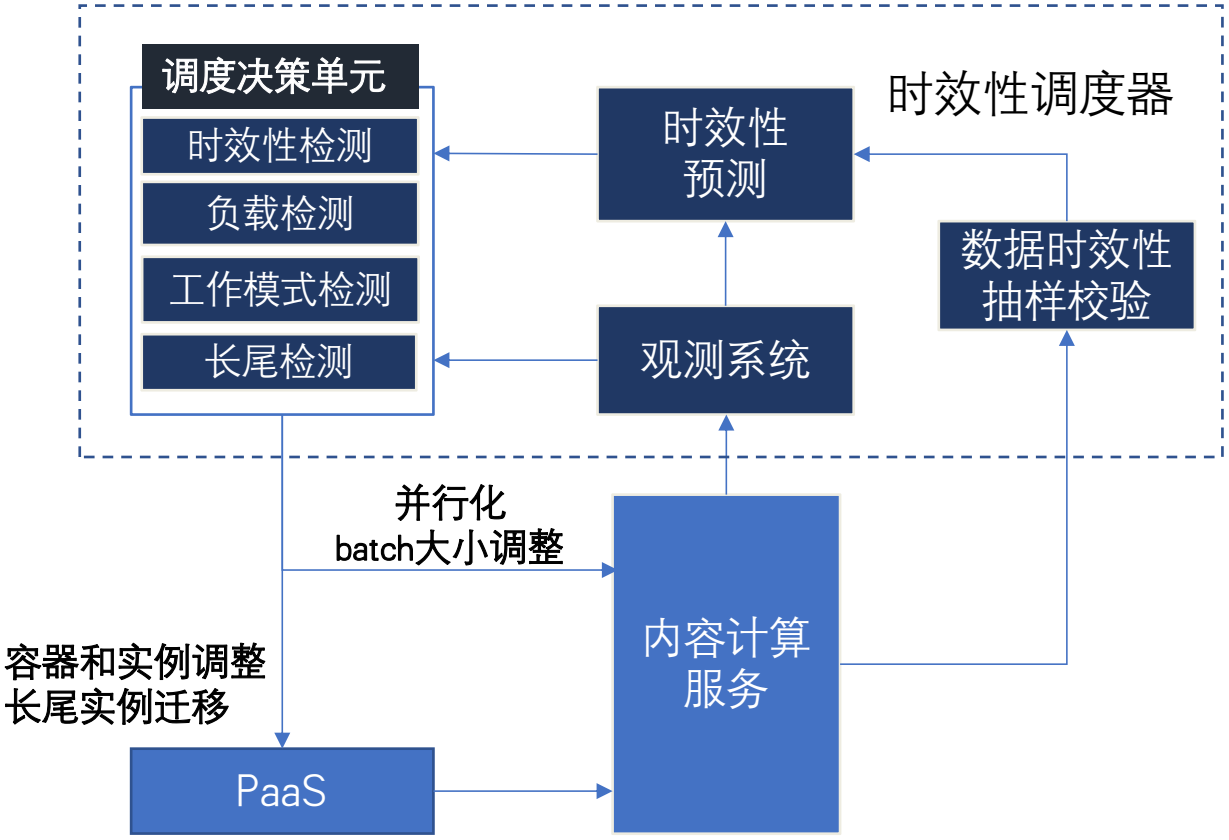


内容计算控制器——时效性调度器

挑战：如何自适应保障时效性，特别是秒级时效性业务

思路：

1. **在事件触发调度中引入时效性预测机制**
   - 利用堆积和处理耗时指标以及指标的变化进行预测
   - 随机抽样数据解包调整预测值（端到端的下发时间写到包里面）
2. **使用技术手段自适应时效性保障**
   - 并行化与批量化调整
   - 容器规格调整
   - 实例伸缩
   - 长尾实例迁移



时效性调度器工作机制

弹性化架构演进

阶段三：自适应优化降低成本和保持架构合理性

自适应优化降低成本和保持架构合理性——局部最优化陷阱

存储和计算的弹性能力建设让系统的交付效率有了大幅的提高，但是我们发现经过一段时间的迭代，有些业务的成本是超出预期的，而且架构存在不合理的现象

case1：数据存储的冷热属性发生了变化

比如医疗场景『症状』比『疾病』更能体现冷热区别了

↓

分类1：有些接入时确定的属性不合适了

case2：存储容量调整过程中的选型问题

比如磁盘介质的存储引擎随流量扩容的过程中某个时刻成本会超过内存介质

↓

分类2：自动交付过程中无法确定哪种最优方案

case3：内容计算根据时效性调度过程中的成本问题

比如对于数据吞吐抖动的业务按照try best去保障时效性会导致成本高出很多

↓

分类3：一些控制策略对不同的业务最优值不同

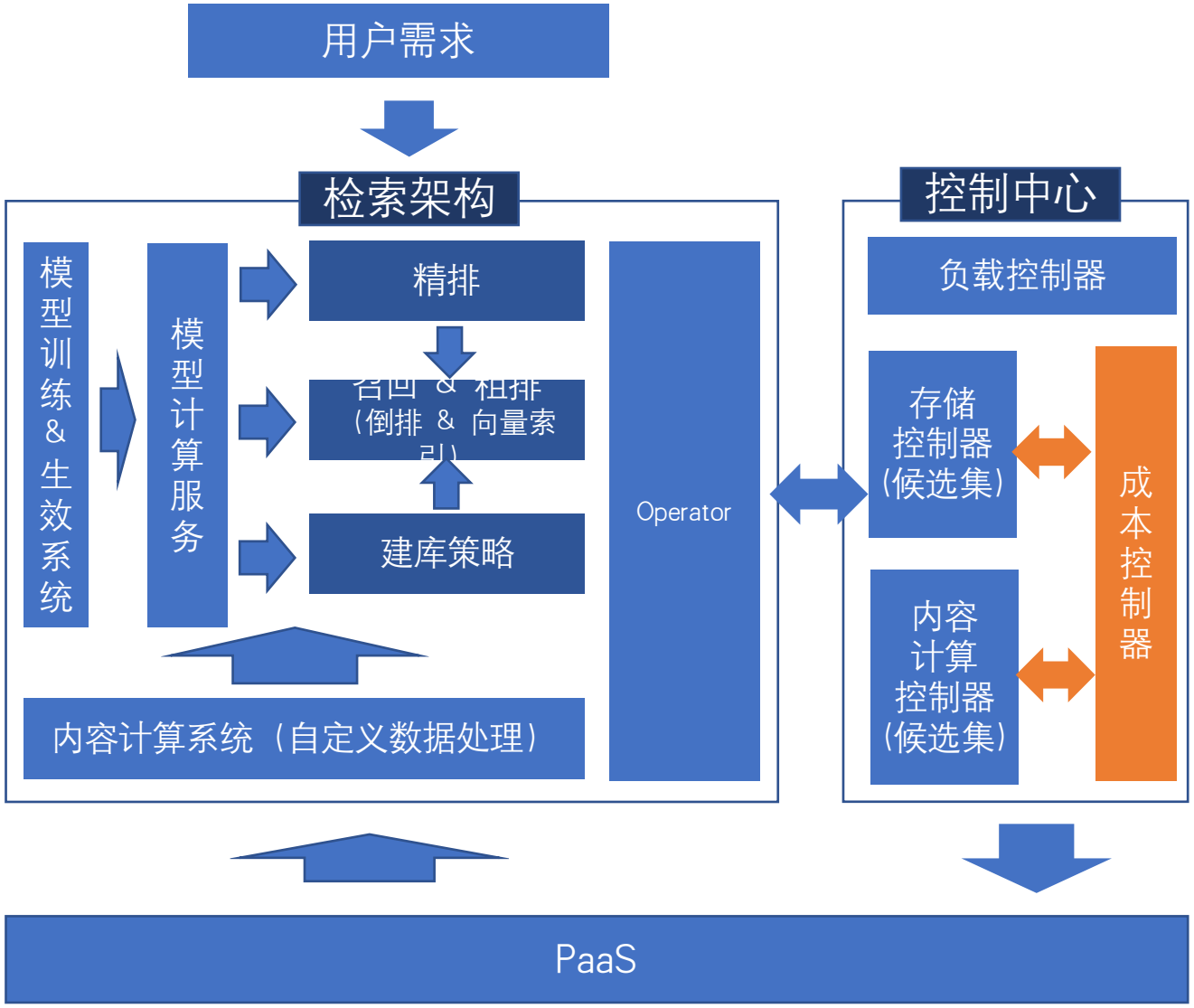
问题本质都是在自动交付过程中，系统进入了局部最优，需要建设更上层的弹性机制，自适应的优化，打破局部最优

自适应优化降低成本和保持架构合理性——引入成本约束实现全局化弹性调整

整体的工作过程分为两个部分：

- **全局化思路产出架构方案候选集**
  - 存储方案候选集——**模拟接入**
    - 利用系统运行过程中的观测数据，模拟数据接入的流程（**更准的先验数据**）
  - 内容计算候选集——**策略分类**
    - 设置不同的调度策略组，模拟任务在不同策略上进行调度（**随机化实验**）
- **引入成本控制器来决策各种候选的方案**
  - 计算候选集和线上状态的成本对比
  - 利用已建设的弹性机制进行调整

这部分的弹性化空间很大，更多的策略我们还在进行中



展望：超自动化交付

超自动化交付——需求理解阶段引发交付效率风险

用户通过配置DAG的策略配置将策略插件组合起来来满足检索需求，这个部分对于业务来说是有一定门槛的：

配置成本高

- **比如搜索场景频繁使用的语义检索**
  - *建库侧配置预估生成向量*
  - *配置向量生效倒排和向量索引*
  - *检索 query 配置预估生成向量*
  - *倒排计算使用 nnscore 过滤*
  - *使用 nnscore 作为排序因子参与排序（粗排和精排）*
  - *配置向量索引召回参与混排*

推荐的使用方案至少涉及7个插件的配置组合使用

配置不对导致结果错误

- **比如精确召回场景，按价格排序返回**
  - *如果只配精排层的价格排序，没有配粗排，导致最终的 topk 不是全局 topk*
  - *如果检索配了召回策略，建库侧没有声明要建 sorted 拉链，会退化成全局排序，在长拉链截断场景下结果不准*

这类问题新同学使用时会经常出现

结果正确，但不是最佳方案

- **比如数据过滤功能**
  - *有的过滤项能作为表达式的一部分，可以大幅提升过滤的效率*
  - *有的过滤策略复杂，不能作为表达式，在一些轻量级筛选截断后执行过滤也能大幅提升效率*

相当大比例的容量增长都是由于这类非最佳实践导致的

这些都是典型的场景，通过模板的方式将这些子功能实现，作为基础功能让业务直接使用

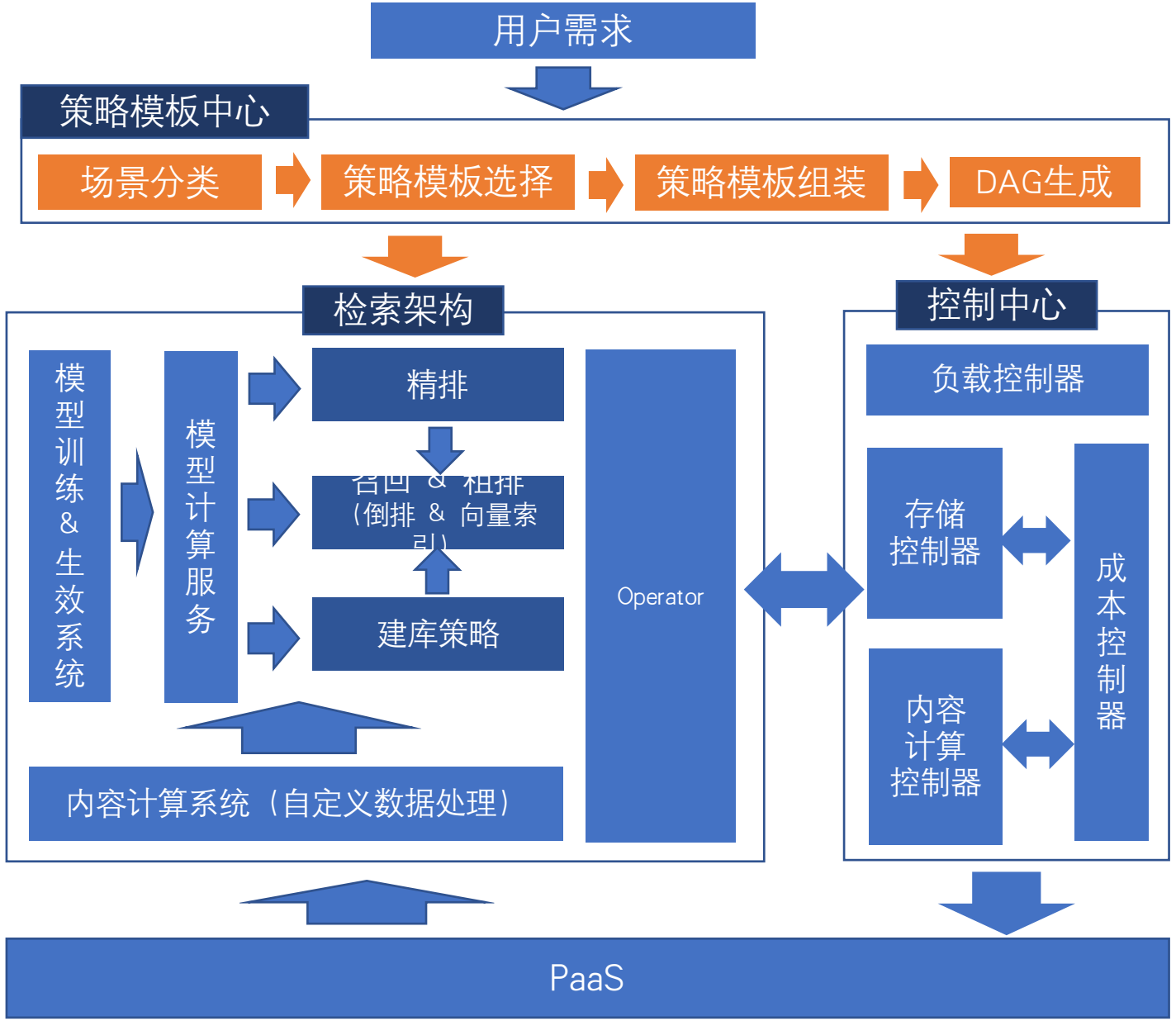
超自动化交付——策略插件模板机制打通需求到交付的全流程

工作流程

- 业务声明式的表达需求，将检索系统当黑盒
- 系统负责理解业务，对原始需求进行转换
- 弹性化架构负责确定实现方案以及落地

目标效果：

- **业务接入效率3人日**
- **业务迭代天级交付比例90%+**



QCon+ 案例研习社

扫码学习大厂案例

学习前沿案例，向行业领先迈进

40 个
热门专题
—
行业专家把关内容筹备，
助你快速掌握最新技术发展趋势

200 个
实战案例
—
了解大厂前沿实战案例，
为 200 个真问题找到最优解

40 场
直播答疑
—
40 位技术大咖，每周分享最新
技术认知，互动答疑

365 天
持续学习
—
视频结合配套 PPT
畅学 365 天

THANKS

软件正在改变世界

SOFTWARE IS CHANGING THE WORLD