百度智能运维实践：异常检测

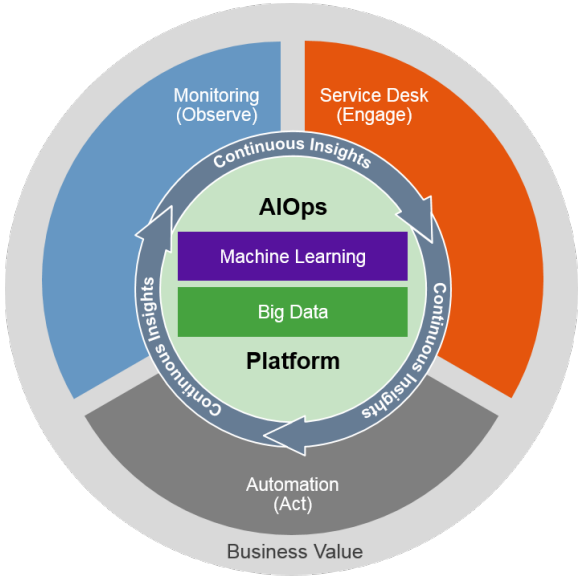
王博、姚睿尧、潘成龙
百度运维部

目录

- 背景
- 异常检测
- 时序数据存储
- 时序数据采集汇聚计算

AIOps智能运维解读

- Gartner 的解读
  - Big Data + Machine Learning 驱动
  - 三大场景：
    - Automation
    - Monitoring
    - Service Desk



- 百度的实践
  - 智能化：运维策略库
  - 数据化：运维知识库
  - 工程化：运维开发框架

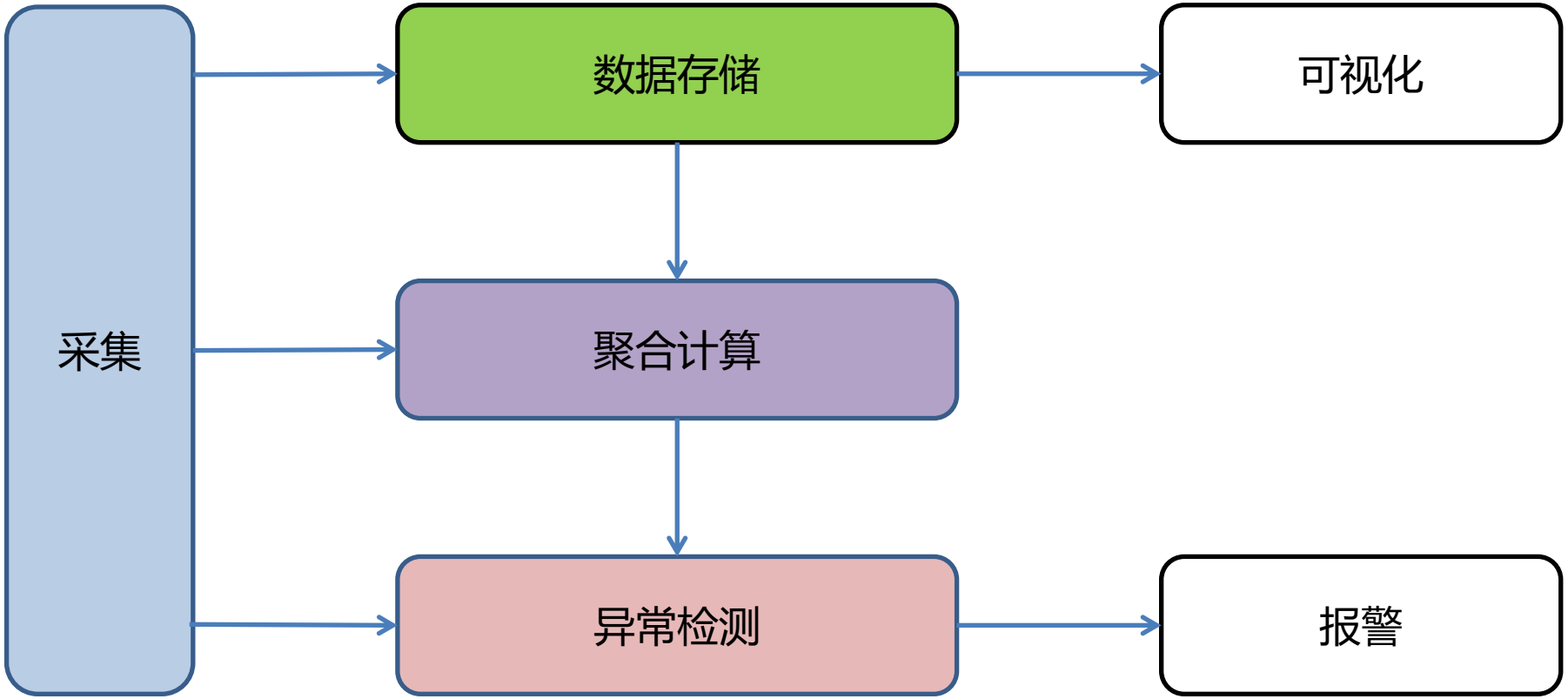
  - 典型的几个运维场景：
    - 故障处理
    - 客服咨询
    - 部署变更
    - 容量管理

Source：Gartner Report
IT Operations Analytics Must Be Placed Within an AIOps Context.
Will Cappelli (Research VP) | 26 August 2016

Baidu 百度

监控是服务高可用的基础

百度智能监控系统全景



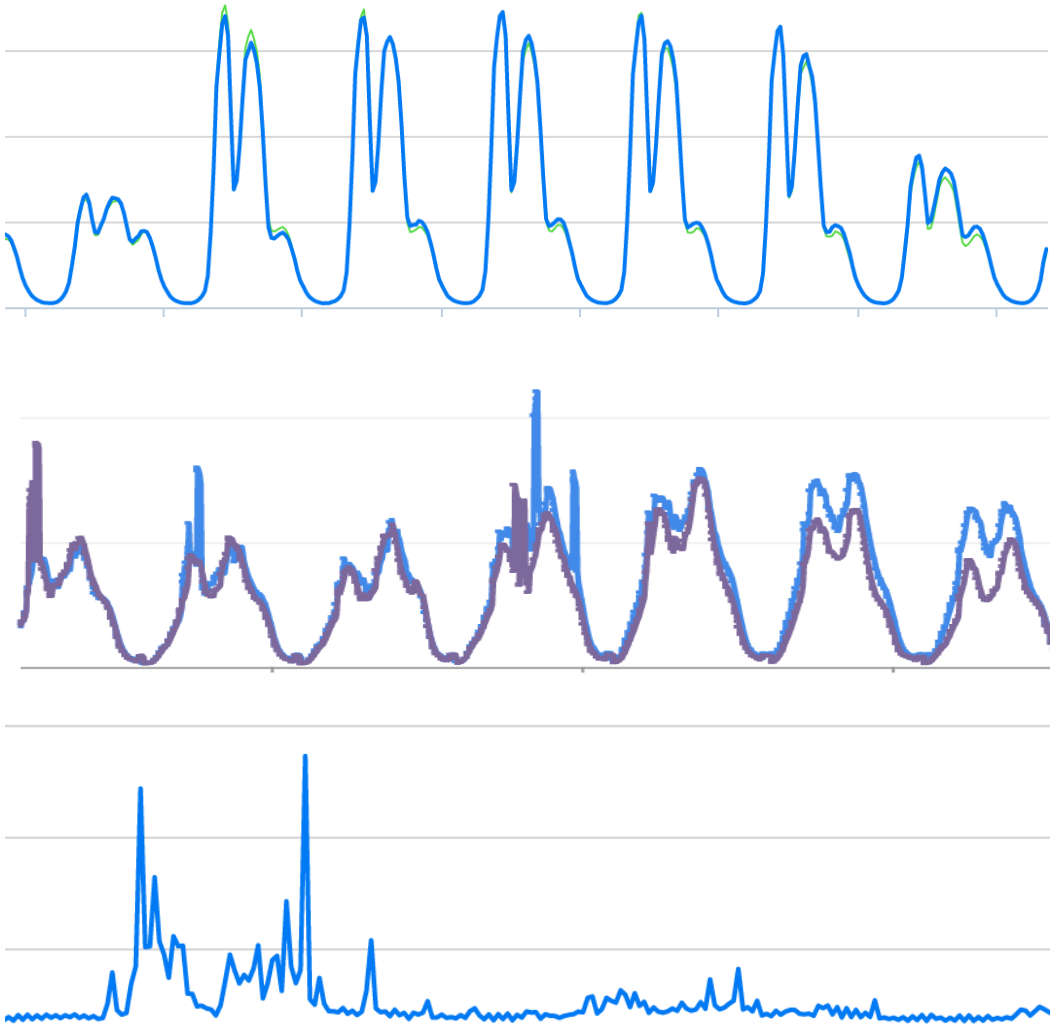
异常检测

目录

- “异常检测”的挑战
- 通用异常检测介绍
  - 累计恒定阈值
  - 突升突降
  - 同比算法
- 算法选择决策树和参数智能配置算法

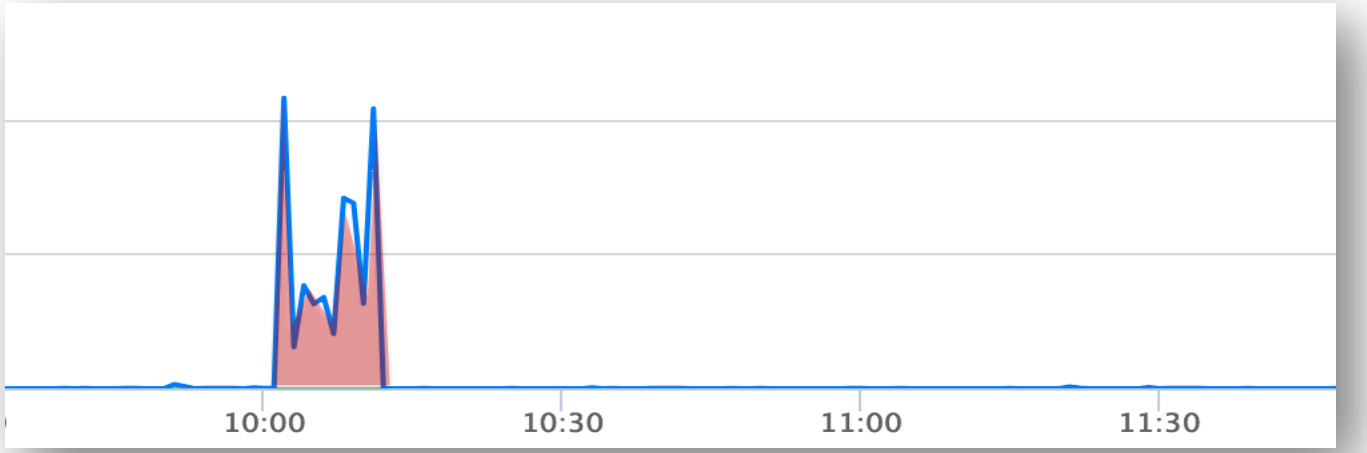
“异常检测”的挑战

- 数据规模大：百万级指标
- 异常检测需求差异明显
  - 业务类型：搜索、广告、地图、糯米…
  - 数据种类：请求数、拒绝数、响应时间、流水、订单等



恒定阈值类算法

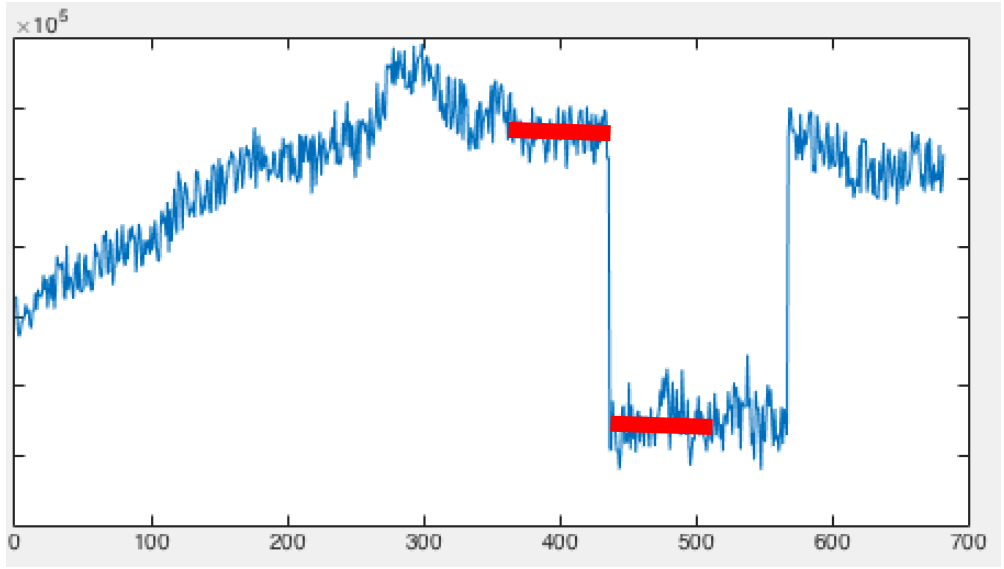
- 单点恒定阈值
  - 单点抖动->filter?
  - 单点抖动=101 or =1.2w?
- 累计恒定阈值
  - 考虑累计量 $s(t) = \frac{x_p + x_{p-1} + \cdots + x_{p-w+1}}{w}$

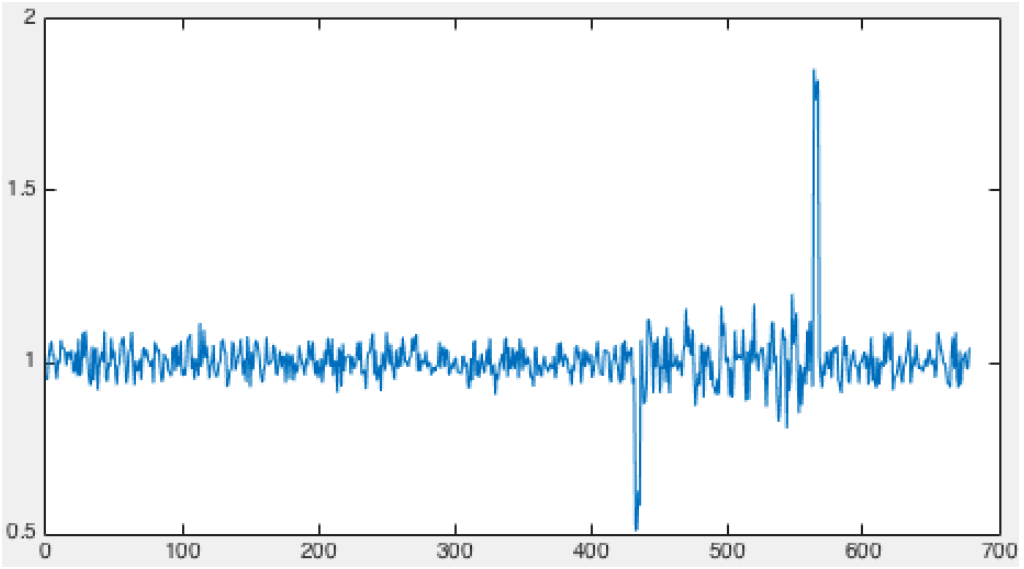


突升突降类算法

- 突变的含义是发生了均值漂移
- 空间转换：

$$r\,(t) = \frac{x_p + x_{p-1} + \cdots + x_{p-w+1}}{x_{p-w} + x_{p-w-1} + \cdots + x_{p-2w+1}}$$

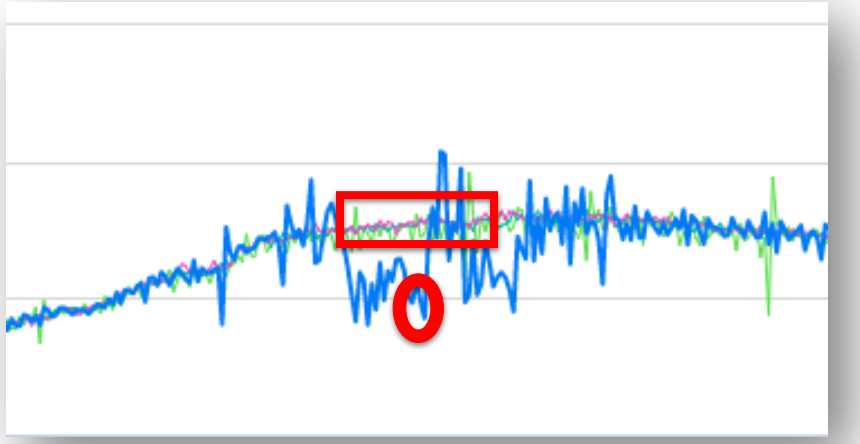


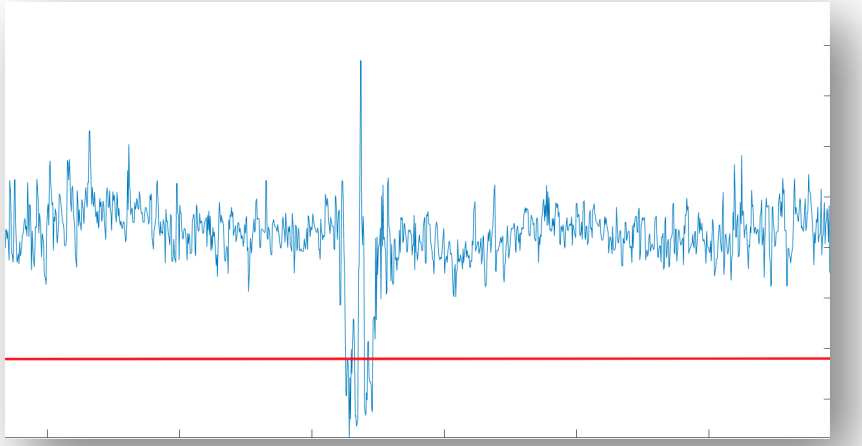


同比类算法

- 假设：每天同时刻的数据分布相似
- 参数估计：求正态分布的均值、方差
- 空间转换：z-score

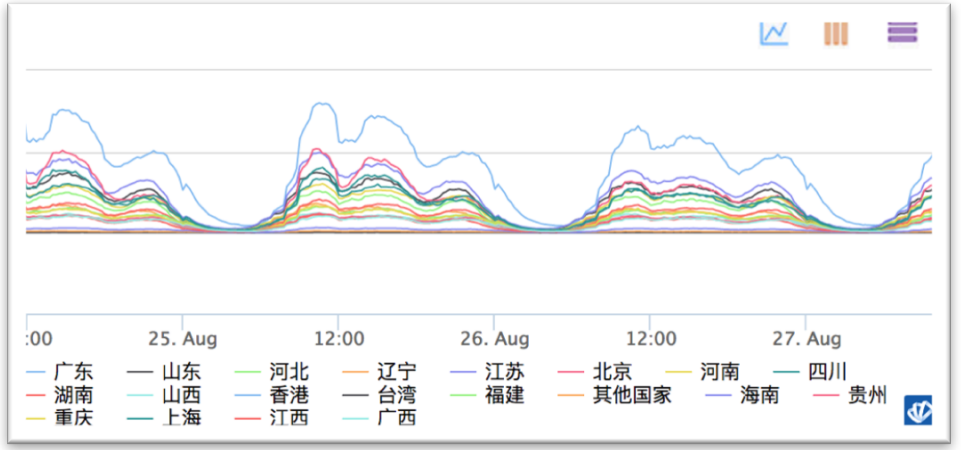
$$z(t) = \frac{x_t - mean(x_{t-kT-w}: x_{t-kT+w})}{std(x_{t-kT-w}: x_{t-kT+w})}$$





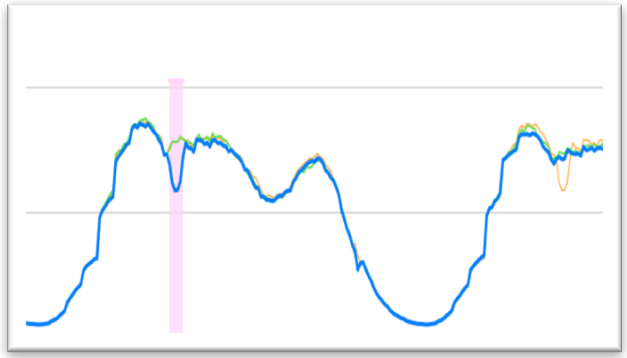
算法选择决策树&参数自动配置算法

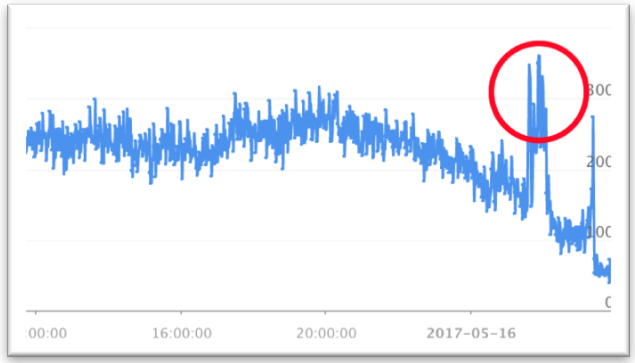
- 曲线数量多
  - 不同的曲线需要用不同的算法
- 参数配置成本高
  - 工作日和休假日不同
  - 白天和晚上不同
- 参数需要定期维护

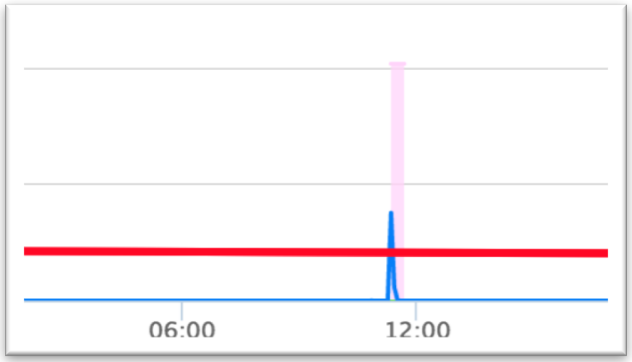


算法选择决策树

- 需求：建立数据与算法的映射
- 常见的场景：
  - 周期数据->同比算法
  - 全局远大于局部波动->突升突降
  - 全局小于局部波动->恒定阈值
- 问题
  - 如何判断数据周期性
  - 如何界定数据的全局与局部波动范围



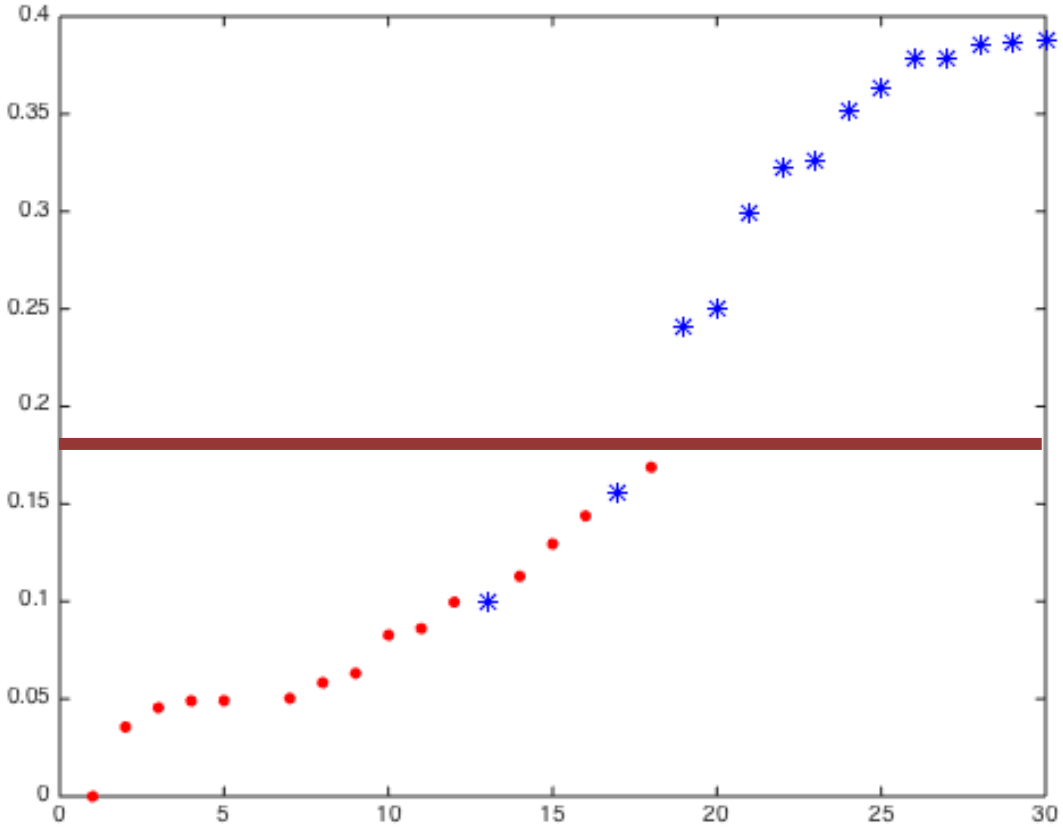




周期数据判断方法

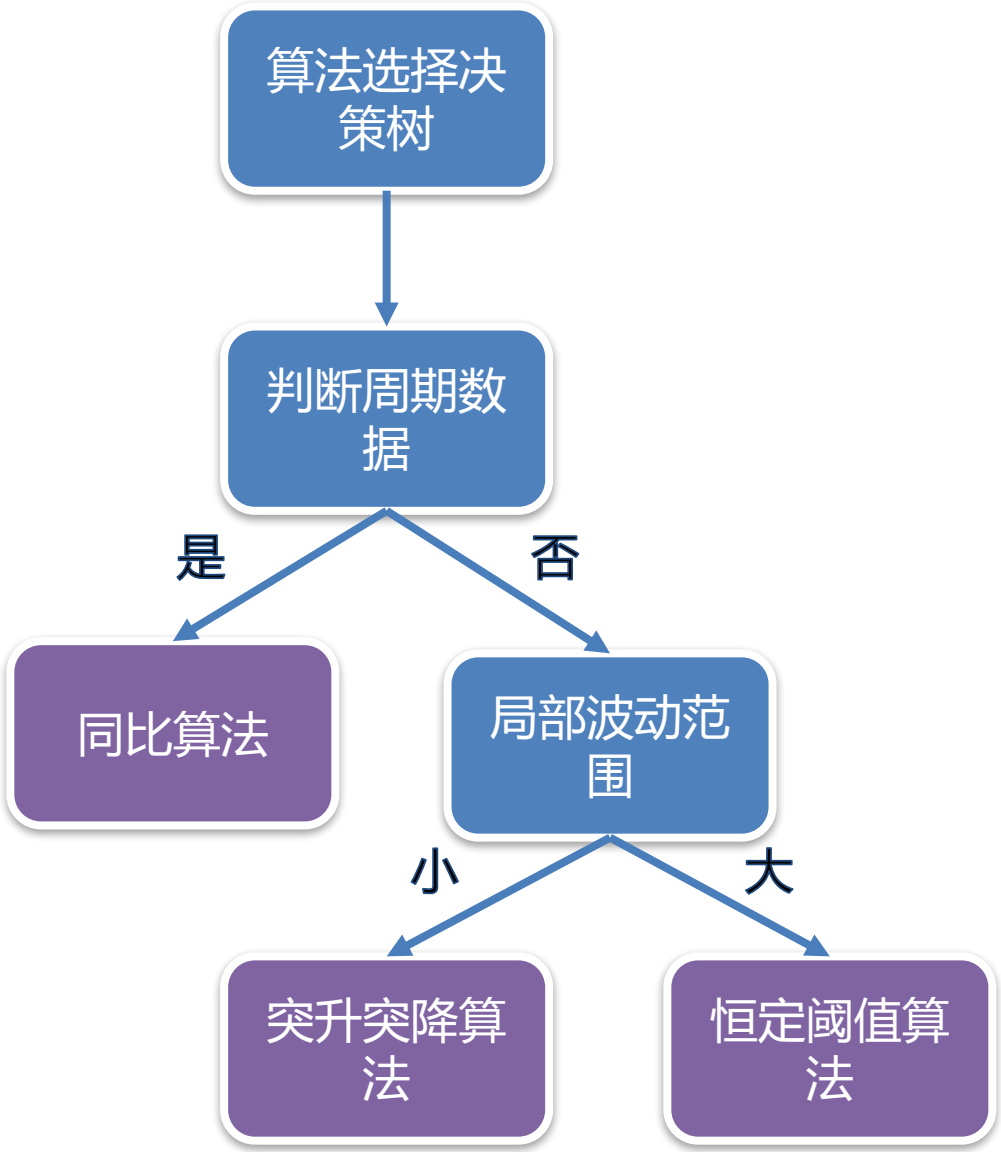
- Power cepstrum
- 基于差分的数据周期特性判断方法
  - 假设：大部分周期数据的周期都是1天
  - 判断步骤
    - 周期内cusum-归一化
    - 周期间数据差分
    - 使用方差进行分类判断

$$x'_t = \frac{x_t}{\sum_{t=0}^{T} x_t}$$



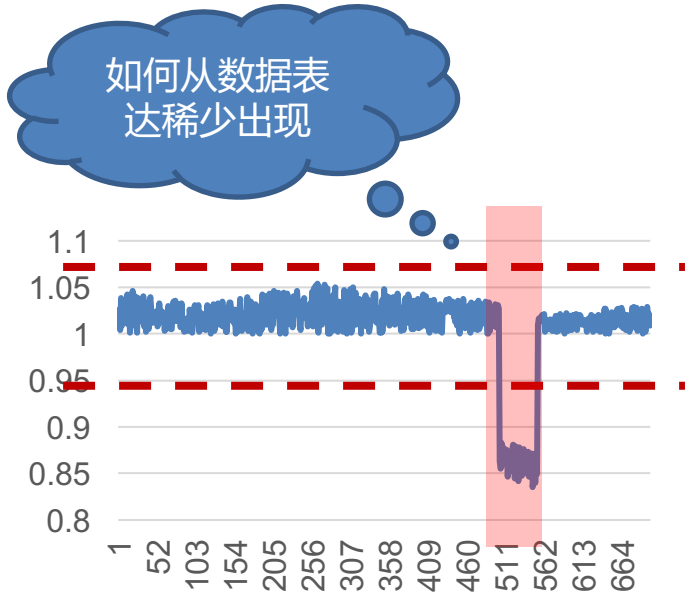
数据的全局与局部波动范围

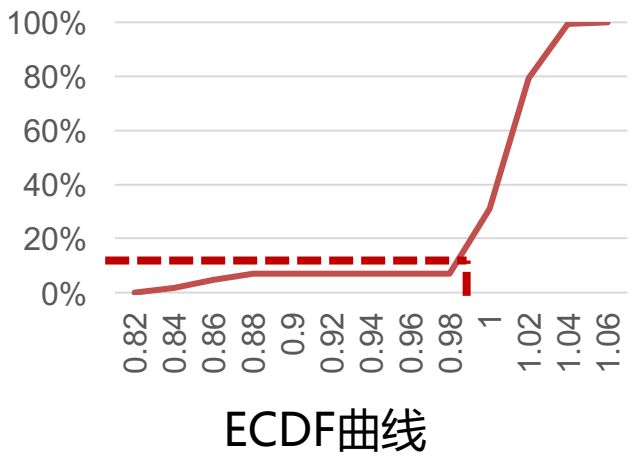
- 全局数据波动：一段时间的方差
- 局部波动范围：小波变换的高频振幅



参数自动配置算法

- 故障相对正常稀少出现
  - 用ECDF表达出现概率
  - 使用经验故障概率确定阈值
- 经验故障概率和实际故障概率不一致
  - 阈值偏严格
  - 补偿系数

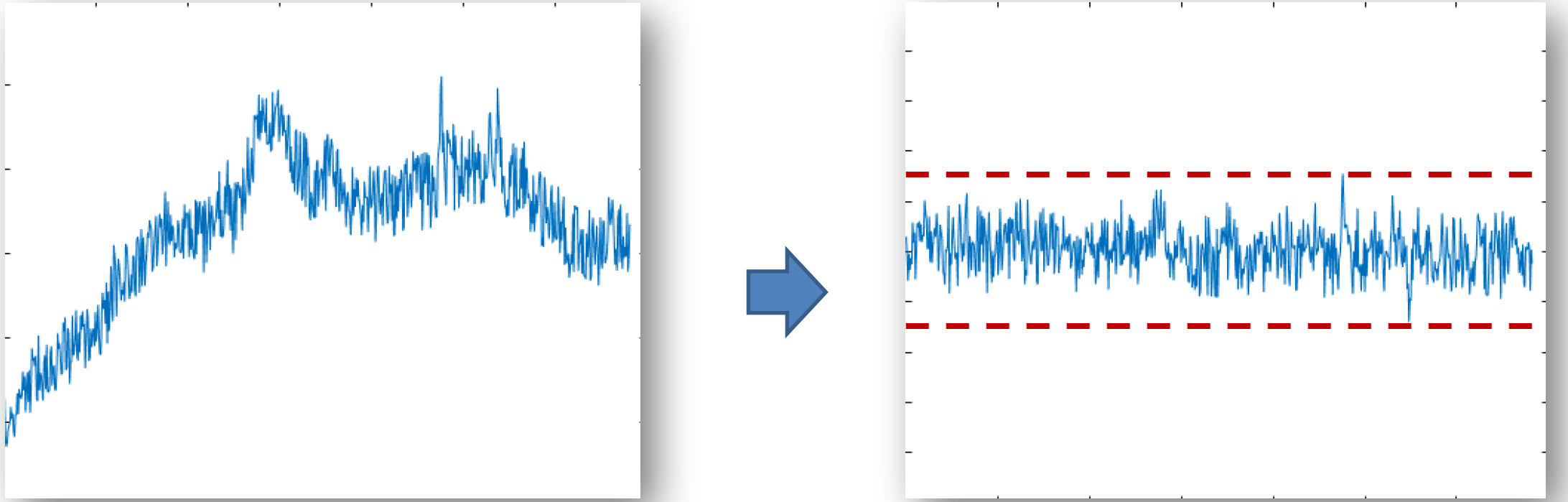




ECDF曲线

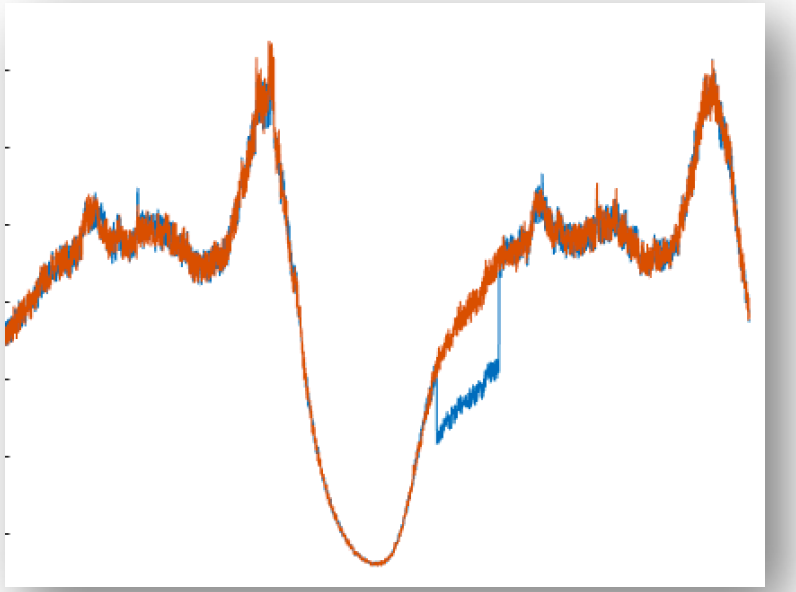
参数自动配置算法

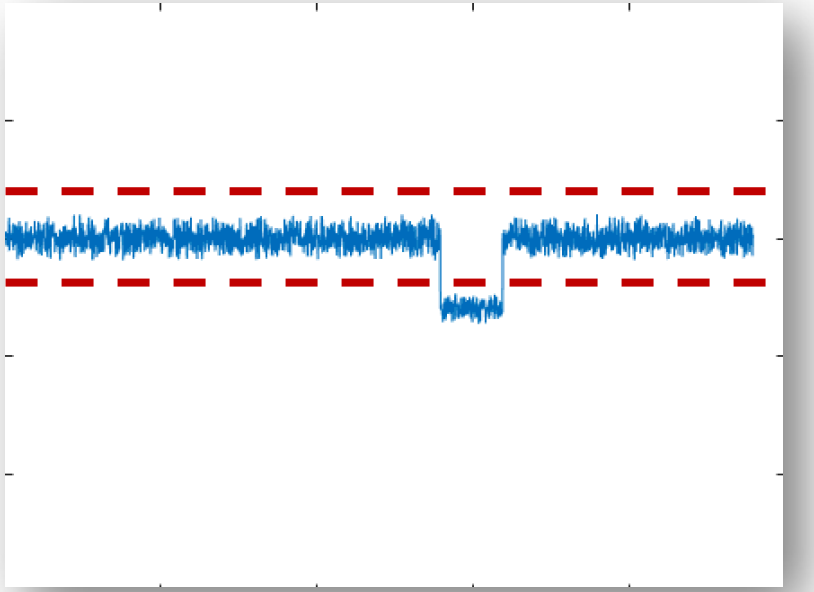
- 突升突降转换到r空间 $r(t) = \dfrac{x_p + x_{p-1} + \cdots + x_{p-w+1}}{x_{p-w} + x_{p-w-1} + \cdots + x_{p-2w+1}}$



参数自动配置算法

- 同比算法转换到z空间 $z(t) = \frac{x_t - mean(x_{t-kT-w}: x_{t-kT+w})}{std(x_{t-kT-w}: x_{t-kT+w})}$





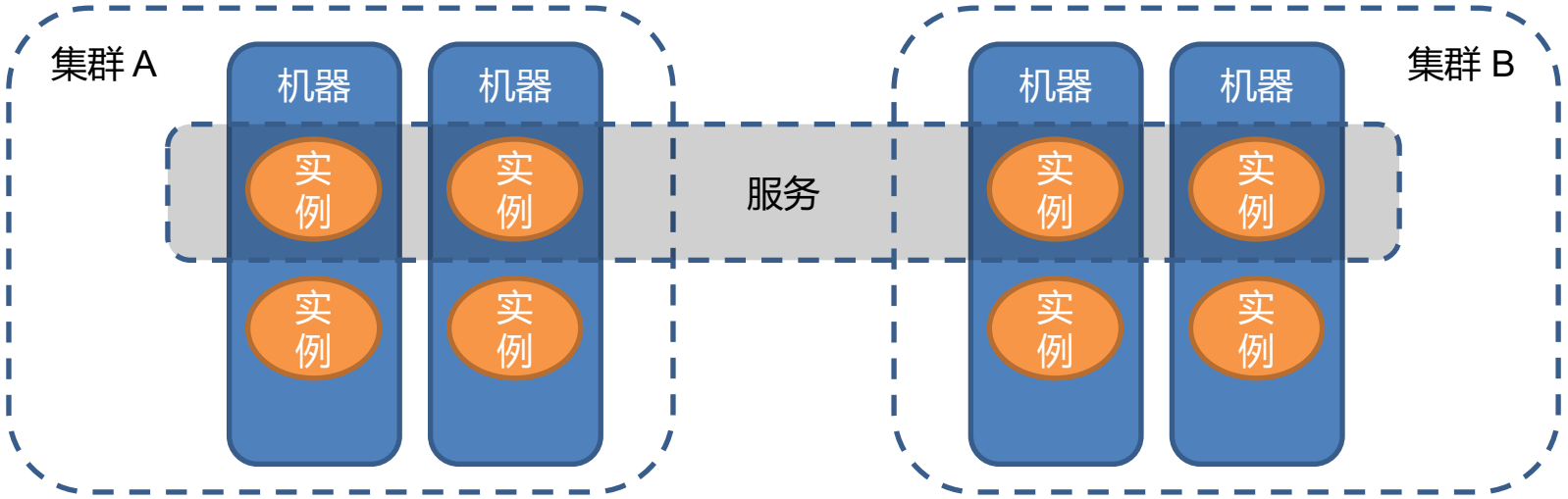
时序数据存储

目录

- 时序数据的特征
- 时序数据存储的技术挑战（百度）
- 时序数据库设计

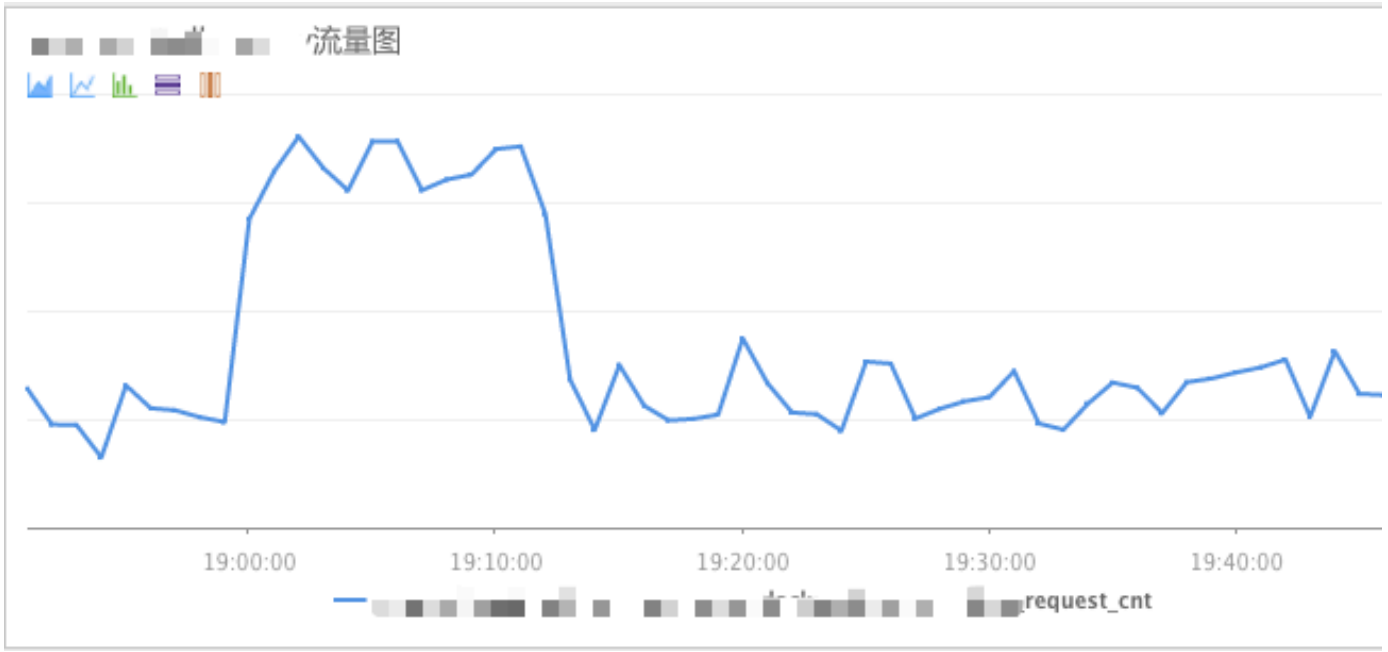
监控中的时序数据

- 监控对象
  - 数十万服务器，全网部署的采集 Agent
  - 机器/操作系统级：物理机/虚拟机，内核/文件系统等
  - 实例级：容器、进程等
  - 逻辑对象：服务、集群、自定义服务组、域名、网络拓扑等



监控中的时序数据

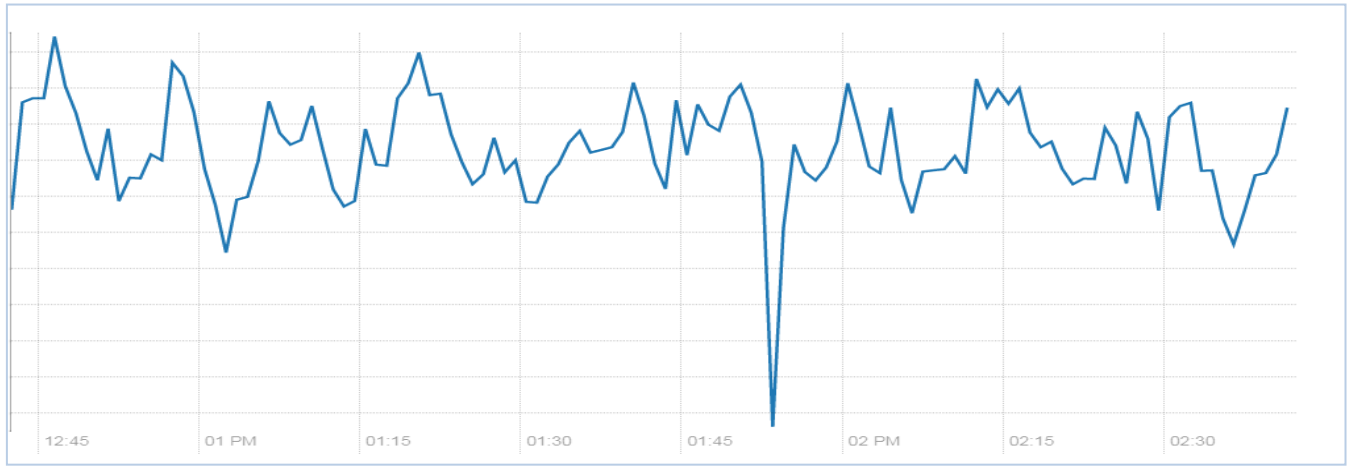
- 时间序列（Time Series）
  - 监控对象 ⊗ 监控项
  - CPU 使用率曲线、PV 曲线



PV 曲线

监控数据的存储模型

- 时间序列（Time Series）：object + tags + metric + datapoints
  - 监控对象(object)：1.nginx.www.tc　　【机器/实例/服务/机房/网络等】
  - 监控项(metric)： pv　　【资源消耗/业务指标/服务状态等】
  - 标签(tag，或称维度) ： province=shanghai，isp=unicom
  - 数据点(datapoint)：(timestamp, value)=(1504594100, 5000)



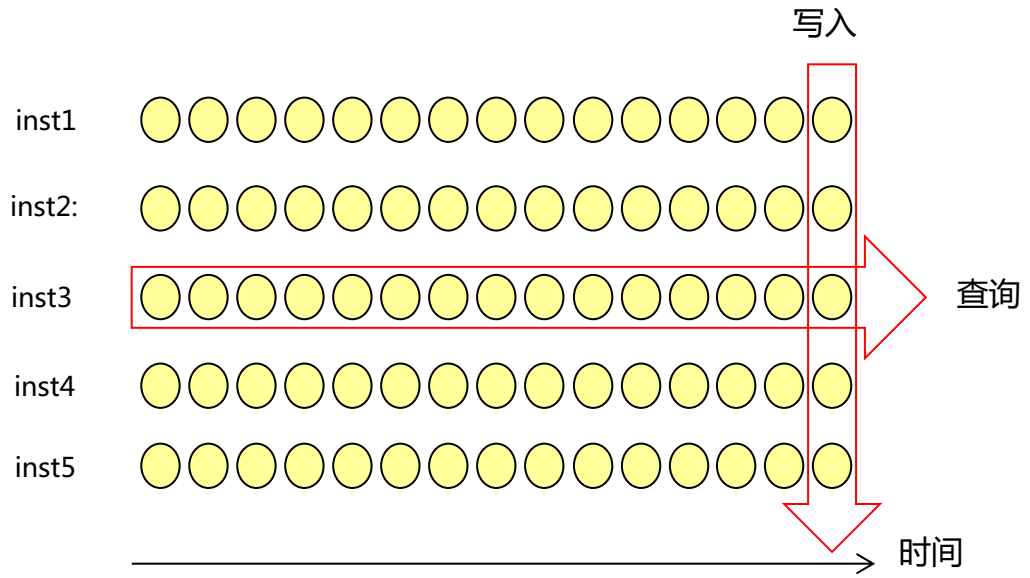
监控时序数据使用场景

- 多元查询需求
  - Web：趋势图、报表（年/月/天同环比）、热力图（网络连通性矩阵）
  - 数据分析：最新数据查询（异常检测）、近线计算（二次计算）、批量查询（离线分析）...
  - 查询途径：API、CLI-tools

网络连通性矩阵

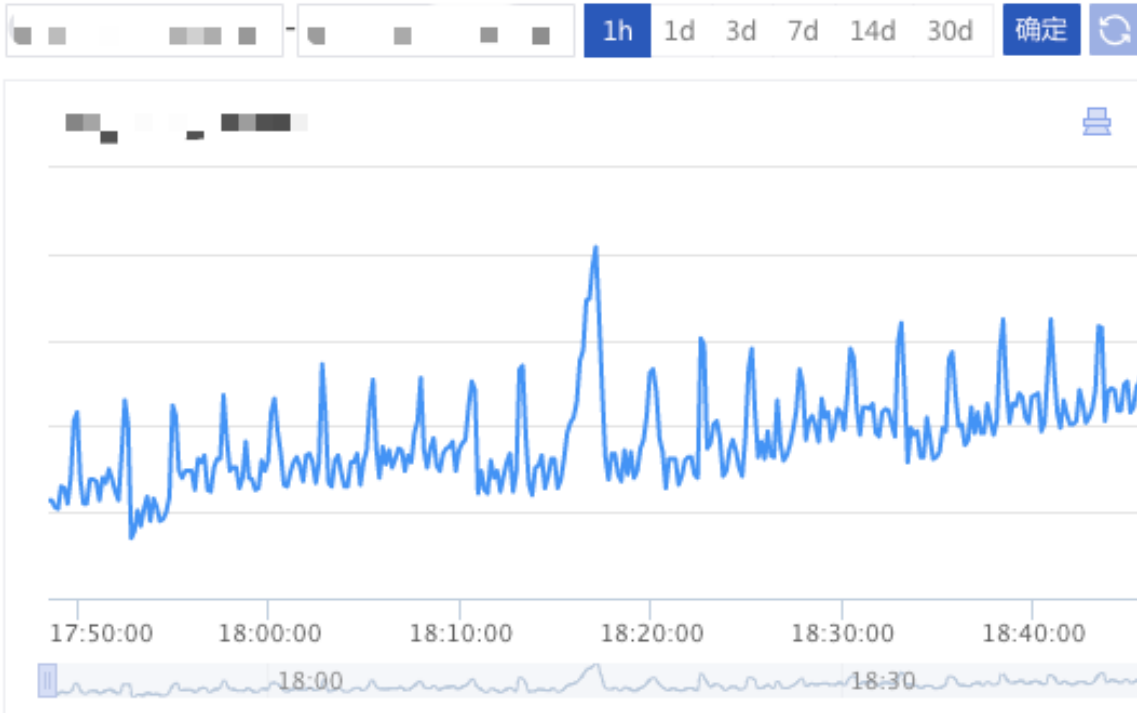
时序数据读写特征

- 读写正交：随机写、顺序读
  - 随机写：多种时间序列各自写入
  - 顺序读：查询一段时间的数据
- 写多读少：写入占比 95% ~ 99%
  - 数据全部写入
  - 关注少量核心指标



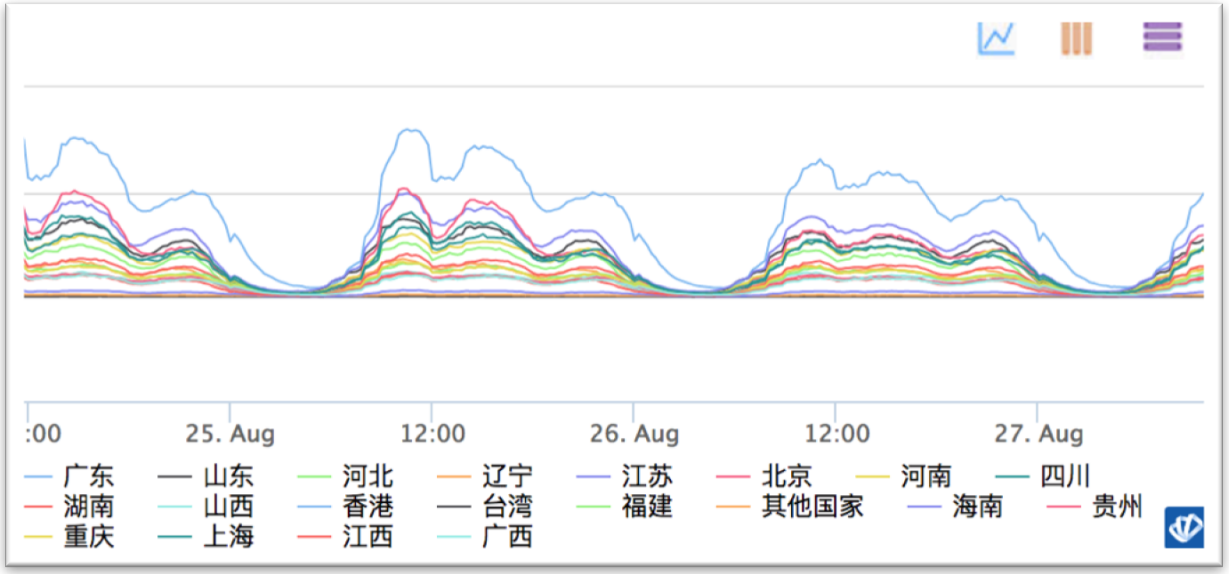
时序数据读写特征

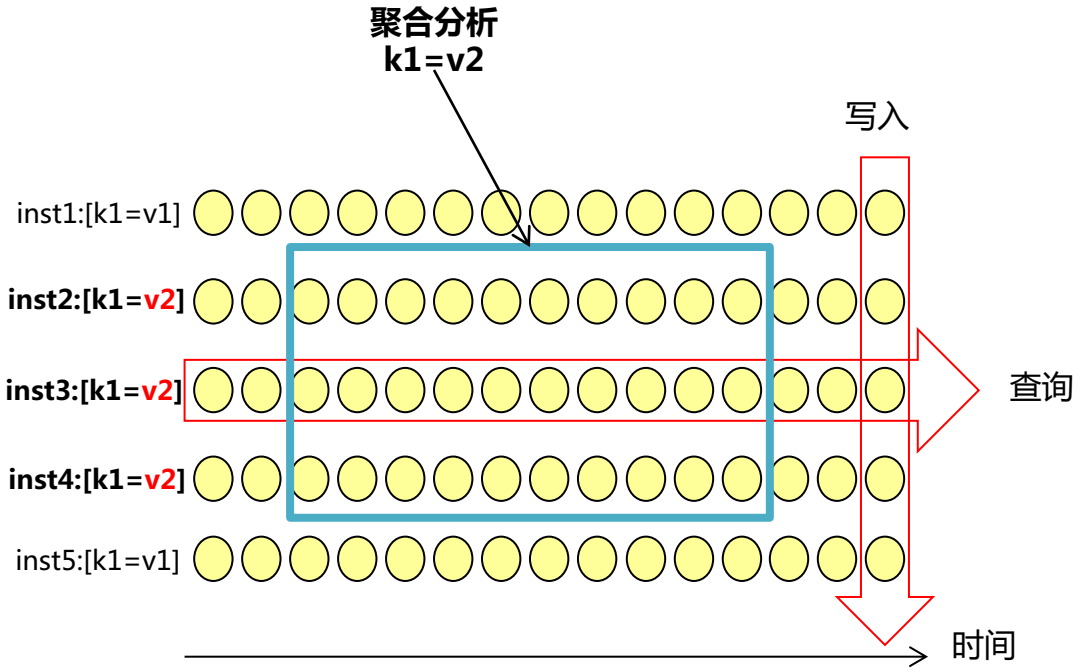
- 最新数据高频访问
  - 关注最近的数据
  - 趋势图默认展示最近一小时
  - 异常检测高频查询最近数据
- 历史数据长期保存
  - 年/月/天的同、环比
  - 数据定制存档



时序数据读写特征

- 按标签（tag）的聚合分析
  - 查找包含指定 tag 的时间序列
  - 服务在华北集群联通机房的总 PV





目录

- 时序数据的特征
- 时序数据存储的技术挑战（百度）
- 时序数据库设计

时序数据存储的技术挑战（百度）

- 监控规模
  - 监控对象：千万级
  - 监控项 和 标签组合： 平均百级别
  - 指标规模（曲线数）：10亿级
- 系统吞吐量
  - 采集周期：10s，甚至5s
  - 监控数据点写入：几千万/每秒，达几万亿/每天
  - 查询量：几万次/每秒，达几十亿/每天
- 可用性/性能
  - 持续高负载： 7 * 24 小时
  - 可用性： 99.99%
  - 查询性能：500ms p99th

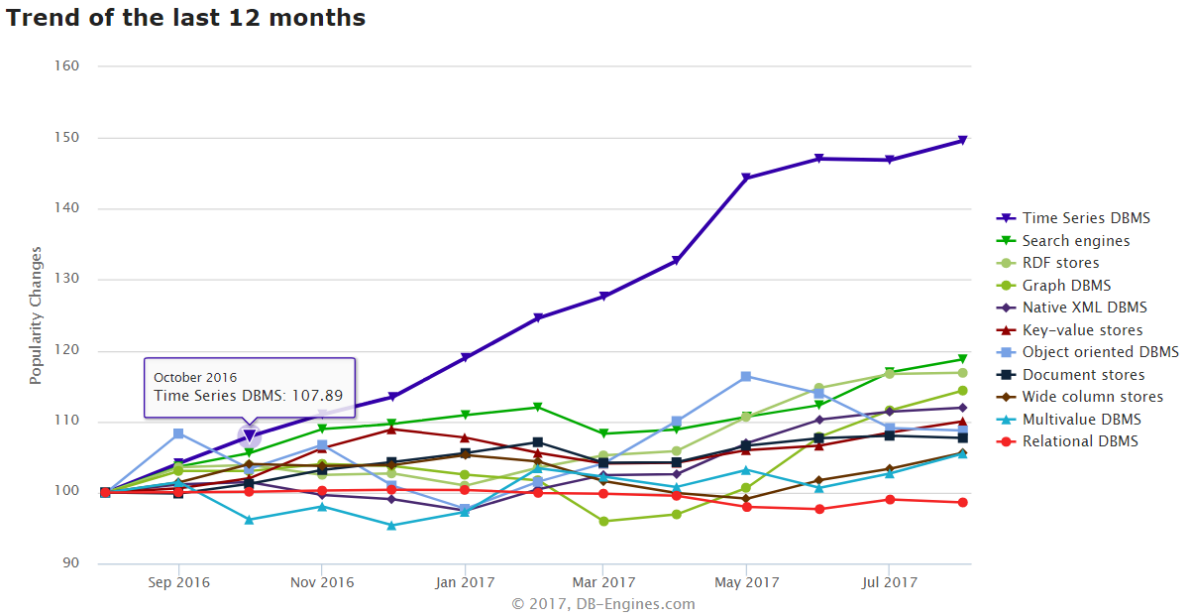
Baidu 百度

目录

- 时序数据的特征
- 时序数据存储的技术挑战（百度）
- 时序数据库设计

业界与开源方案

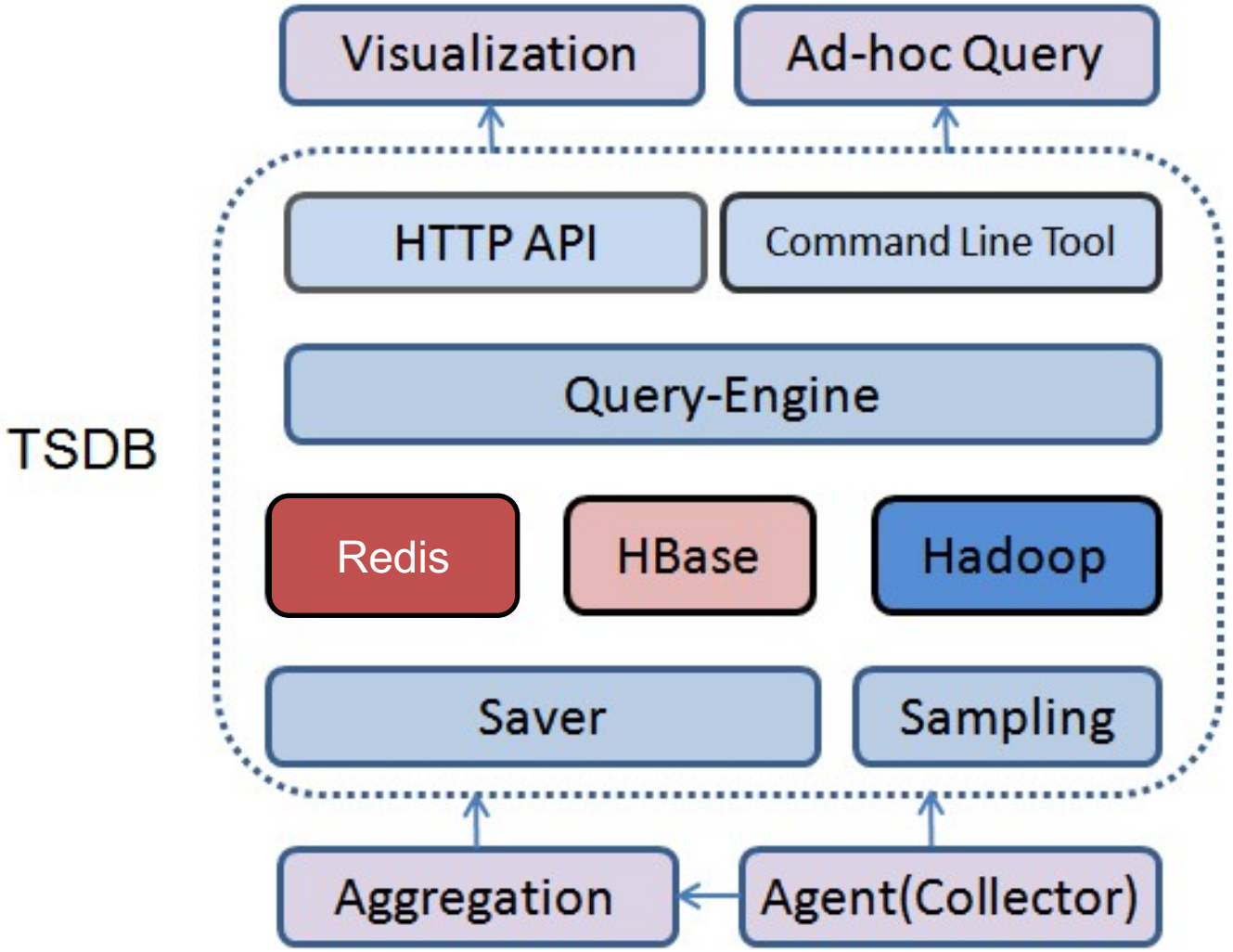
- 开源
  - InfluxDB：底层是自研TSM存储引擎
  - OpenTSDB：底层使用HBase
  - KairosDB：底层使用Cassandra
  - Prometheus：基于LevelDB存储引擎
- 业界
  - Facebook：Gorilla(自研)+ HBase
  - Twitter：Cache+Manhattan(自研)
  - 共同点
  - 底层存储基于LSM(Log-structured Merge-Tree)



Source： DB-Engines

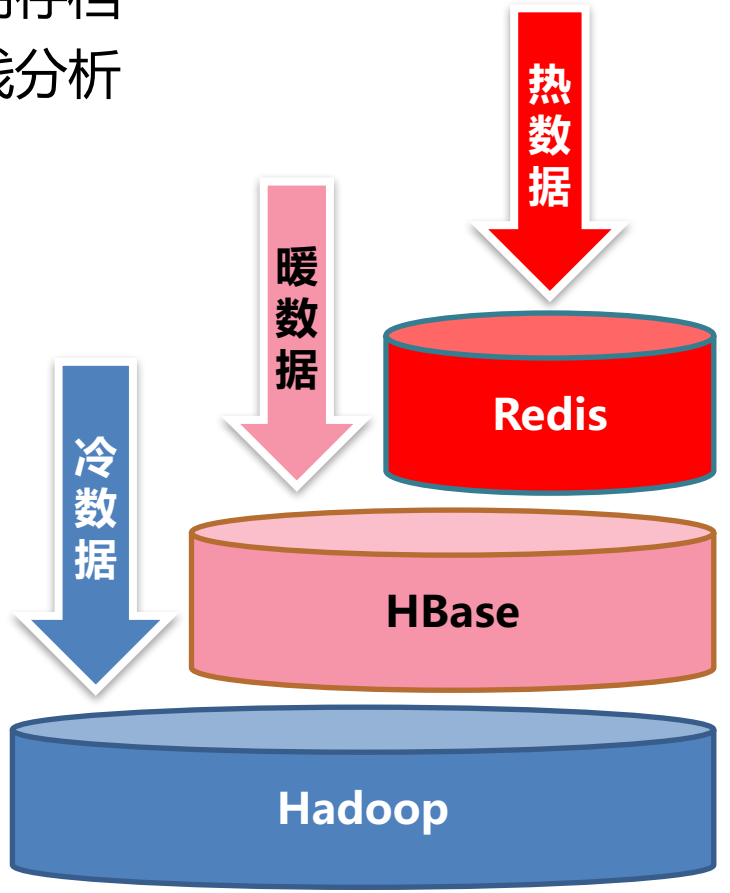
流行趋势：最近1年

时序数据库（TSDB）的架构



应对多元化需求——层次化存储

- 存储层次
  - 关键/热数据缓存到 Redis
  - 暖数据存储在 HBase
  - 冷数据存储在 Hadoop
- HBase
  - 分布式表格存储，可扩展性
  - 写优化（基于 LSM 将随机写转化为批量顺序写）
  - 业界有成功案例
- Redis
  - 成熟的缓存存储
  - 百度自研的分布式 Redis
- Hadoop
  - 长期存档
  - 离线分析



时序数据的存储——HBase

- 行健： <entity_id><metric_id><timestamp_hour>
  - 监控对象+标签： object:[tagk1=tagv1,...,tagkN=tagvN] → entity_id
  - 监控项：metric → metric_id
  - 时间戳：timestamp → timestamp_hour + timestamp_offset
- 数据格式：借鉴 OpenTSDB，按小时的存成一行，如下表格
  - 列为 timstamp_offset，单元格存储 value
- 按维度的二级索引
  - 建立维度到 entity_id 的二级索引

Row Key	+0	+10	+20	...	+3590
0x**01037130**06829384**1292148000**	82	78	84	...	89

性能和成本——缓存与压缩

- 压缩算法
  - Facebook Gorilla 时序数据压缩
  - Delta-of-delta 压缩时间戳
  - XOR 压缩数值
- 压缩效果
  - 在原有基础上 80% 的存储节省（内存）
  - <8% 的 CPU 成本
- 应用方案
  - 基于 Redis
  - 节点无状态
  - 支持多数据类型

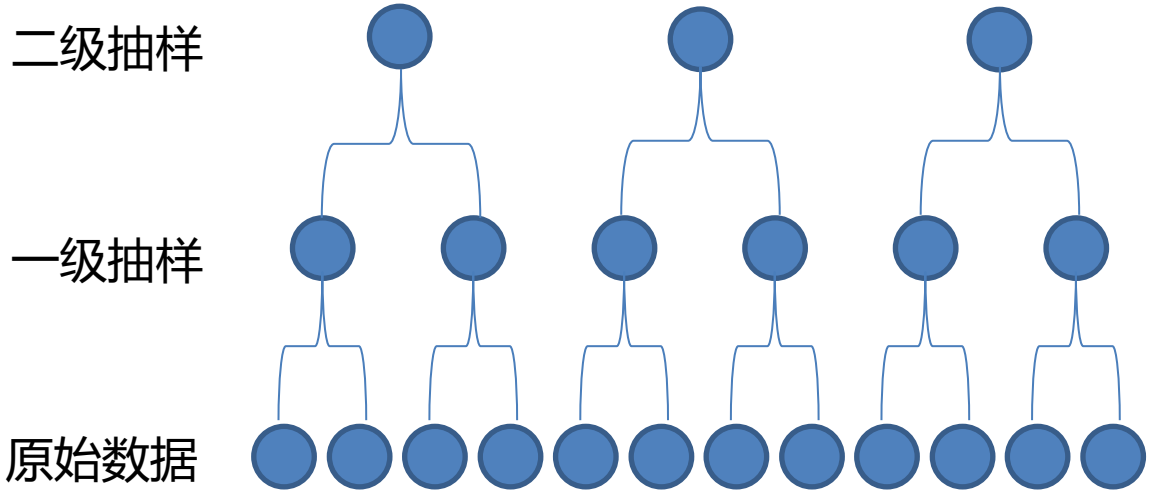
N-2 timestamp	02:00:00	-
N-1 timestamp	02:01:02	Delta: 62
timestamp	02:02:02	Delta: 60
		Delta of deltas: -2

Previous Value	12.0	0x4028000000000000
Value	24.0	0x4038000000000000
XOR	-	0x0010000000000000

11 leading zeros, # of meaningful bits is 1

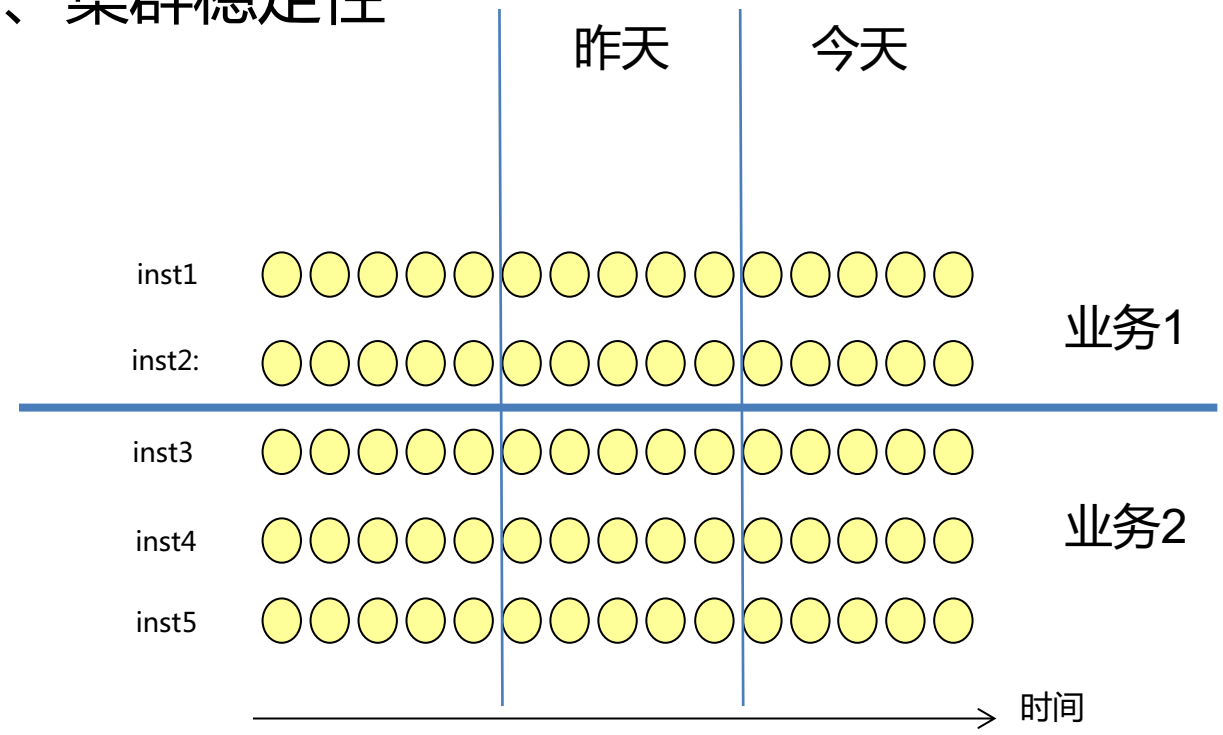
数据保存策略——分级统计抽样

- 统计抽样
  - 统计产生 {max, min, sum, count}
  - 查询上对用户透明
- 分级数据保存策略
  - 原始数据在线保存数十天
  - 一级抽样保存数月
  - 二级抽样存储数年



稳定性和性能的优化——多级拆分

- 按业务类型分集群
- 按租户分库
- 按时间分表：性能、数据保存策略、集群稳定性
  - 不同时间序列随机均匀分布
  - 同一时间序列按时间连续分布
  - 数据的批量过期清除
  - 减小 HBase 的 compaction 压力



时序数据采集汇聚计算

场景

对于一个服务，我们希望

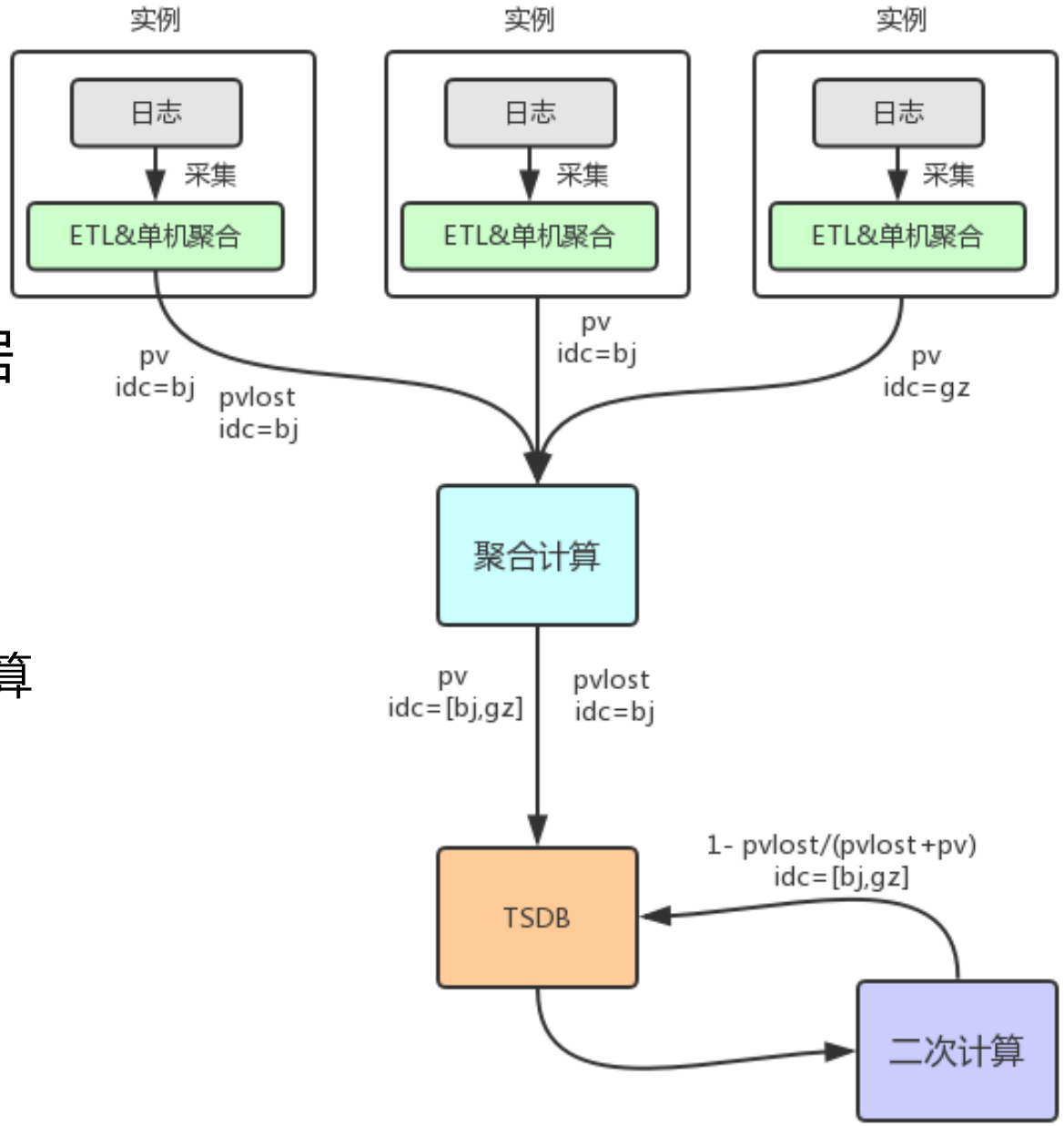
- 指标采集
  - 总体pv，分省份、运营商、接口的pv
  - 总体平均响应时间，分省份，运营商，接口的平均响应时间
  - 不同HTTP状态码的流量
  - 长尾流量数目，占整体的比例
  - 分集群分机房CPU、内存等资源占用情况
  - ...
- 系统异常时能够尽快感知
  - 根据业务需求在10s级到分钟级

典型架构

- “重存储“架构
  - 采集端只负责数据采集
  - 存储端存储所有原始数据，查询时进行多维度聚合
- 问题
  - 原始数据量级大，存储成为系统热点、大规模查询和计算时负载高
  - 按需计算，在指标数和纬度组合量大的情况下指标时效性无法保证

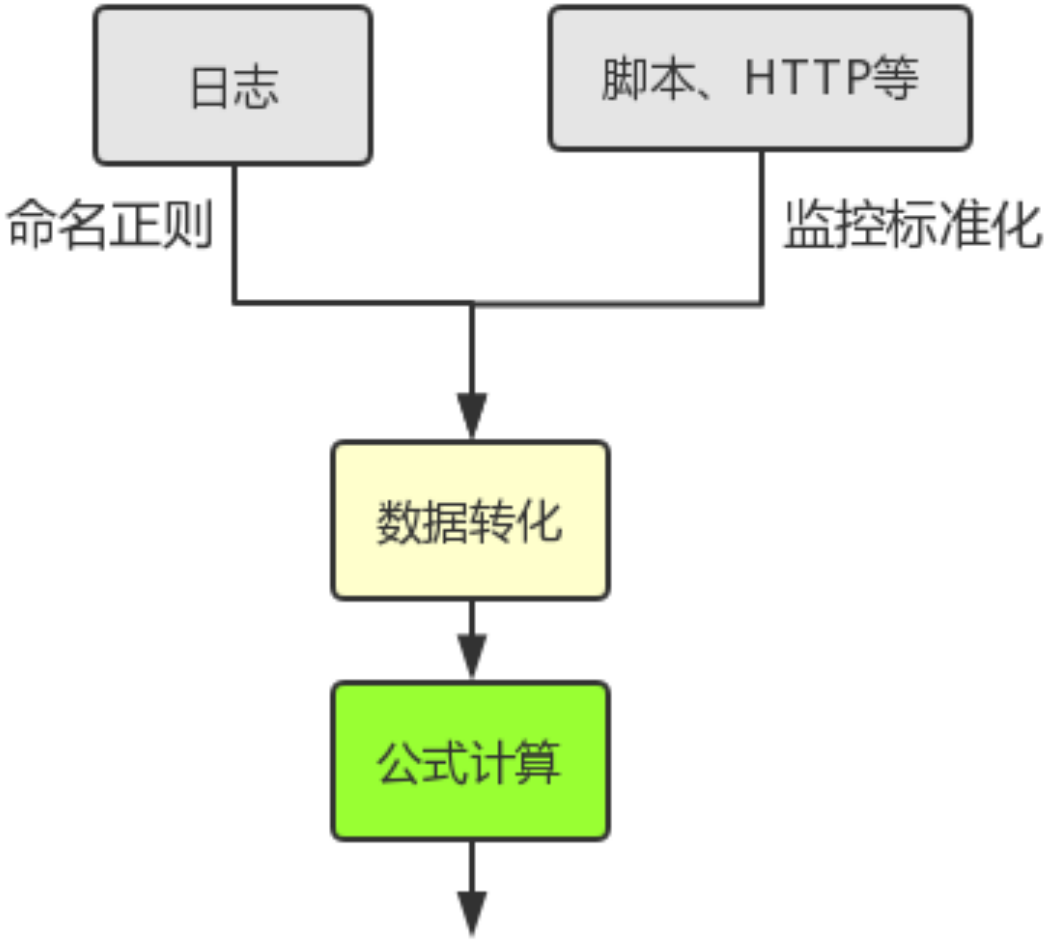
架构

- 多纬度数据预聚合，不保存原始数据
- 分散计算能力，多层次聚合
- 区分场景，混合流式和批量计算
  - 大规模，秒级，简单规则，流式计算
  - 中等规模，分钟级，复杂规则，批量计算



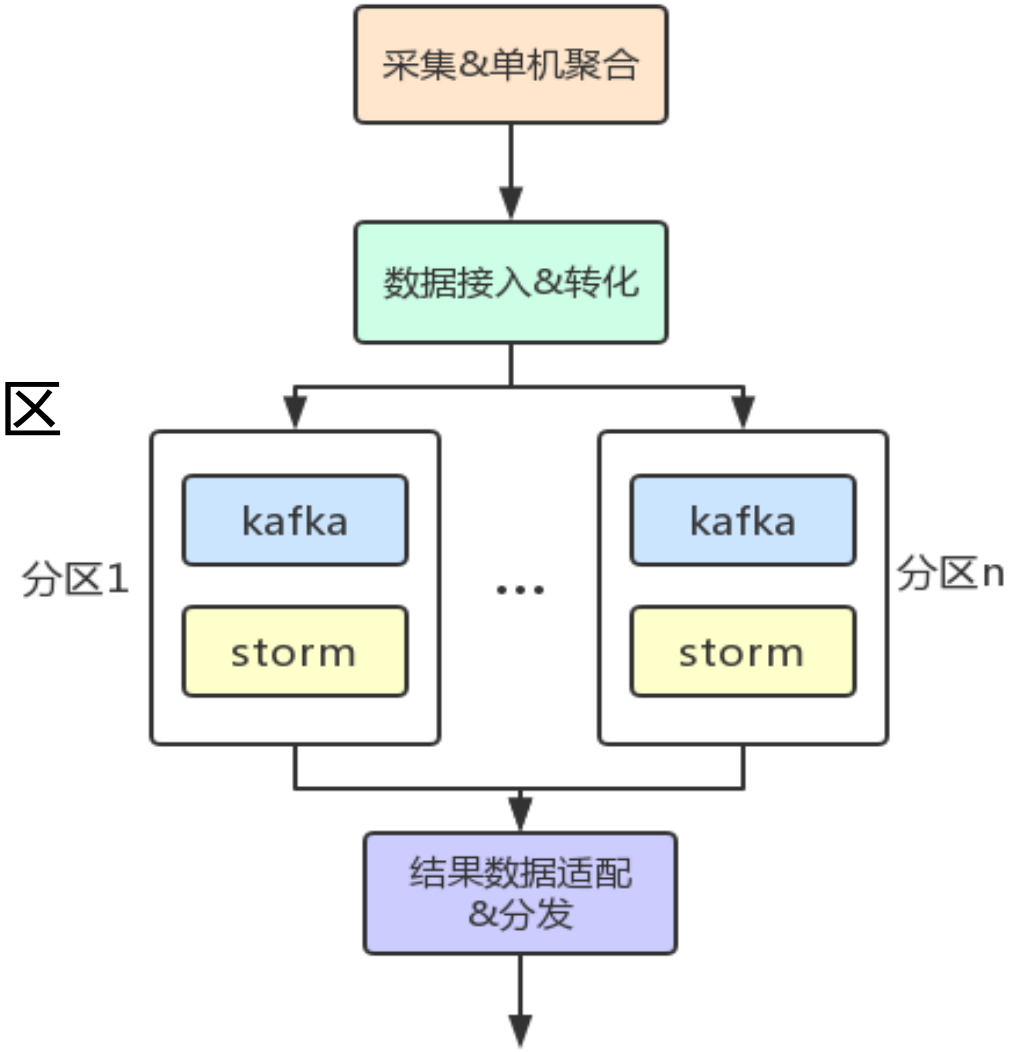
采集

- 异构采集源，数据规范化
  - 命名正则进行日志信息提取
  - 监控标准化
- 采集端ETL
  - 数据转化，运维信息提取
    - 从ip中提取省份，运营商，城市信息
    - 纬度值映射
    - url路径提取
    - ...
  - 公式计算，数据派生
    - 根据现有指标计算出新指标
    - 条件过滤



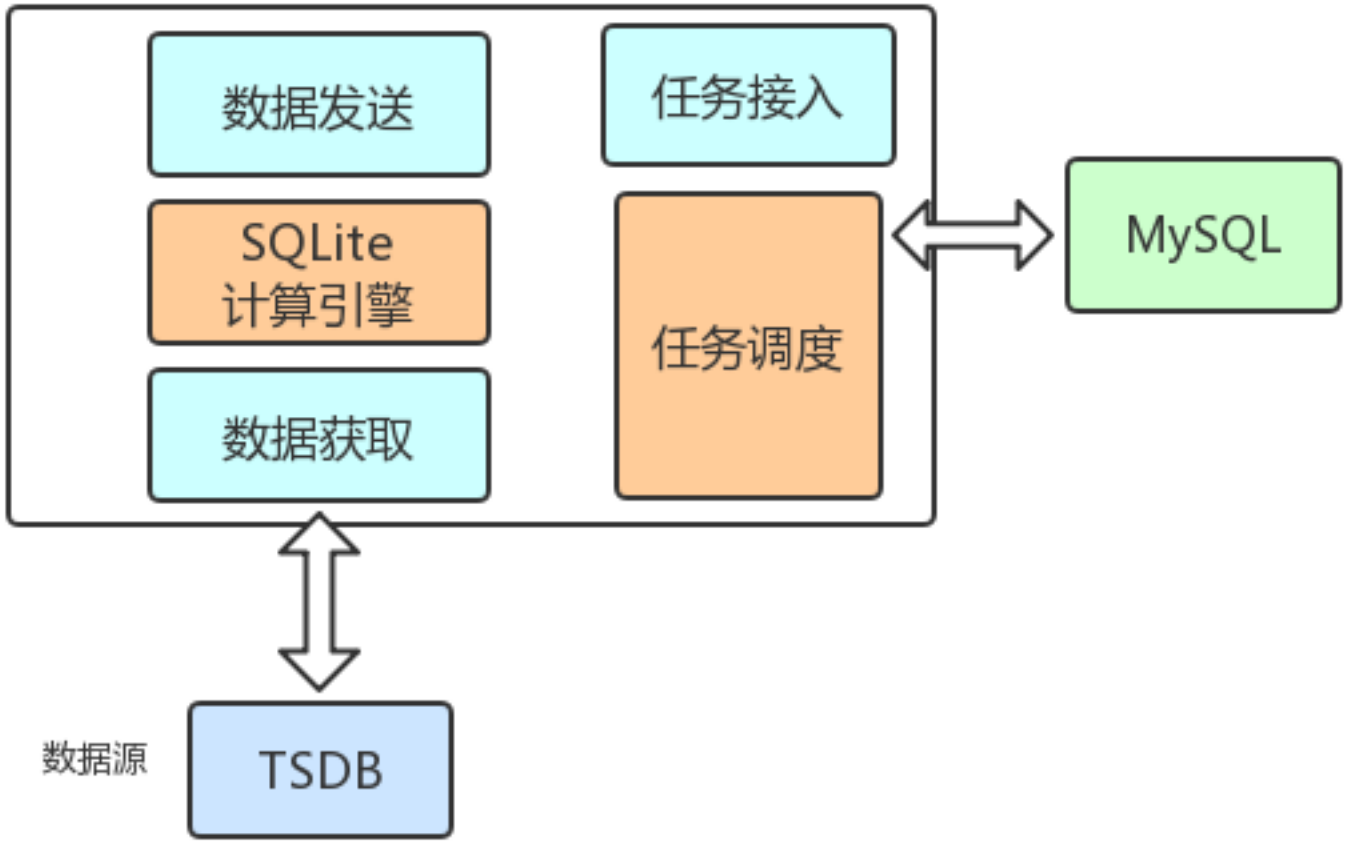
聚合计算

- 流式计算
- 通用计算需求：max/min/sum/avg/cnt
- 统一流量接入和输出
- 按照产品、数据量等纬度进行逻辑以及物理分区



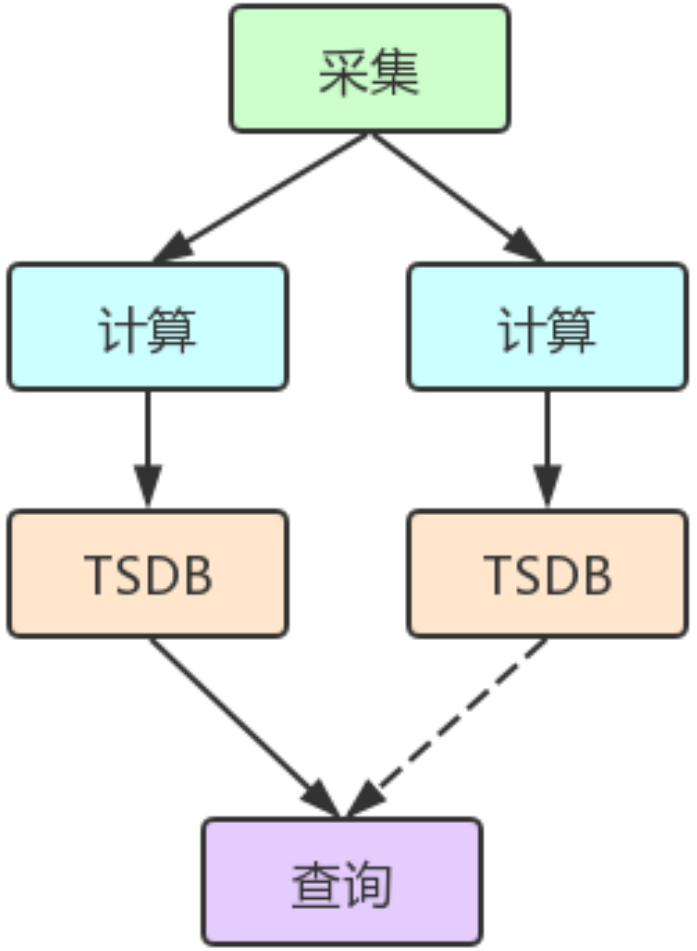
二次计算

- 近线计算
- 计算规则动态变化
- 复杂规则，支持SQL标准算子



高可用性

- 双集群热备
- 自动流量切换
- 弹性扩容和集群恢复能力



THANKS

AIOps智能运维

AIOps智能运维

Baidu 百度