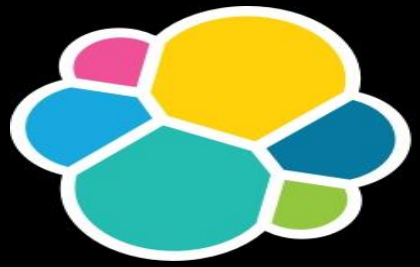

elastic
中文社区
vivo
腾讯云·云社区
ELASTIC 深圳
MEETUP
2019年4月20日
广东深圳市南山区高新科技园中区一路腾讯大厦2F多功能厅
http://elasticsearch.cn



Elastic
中文社区

Elasticsearch
实时高效聚合应用实践

跨越速运集团大数据中心　**李猛**

关于

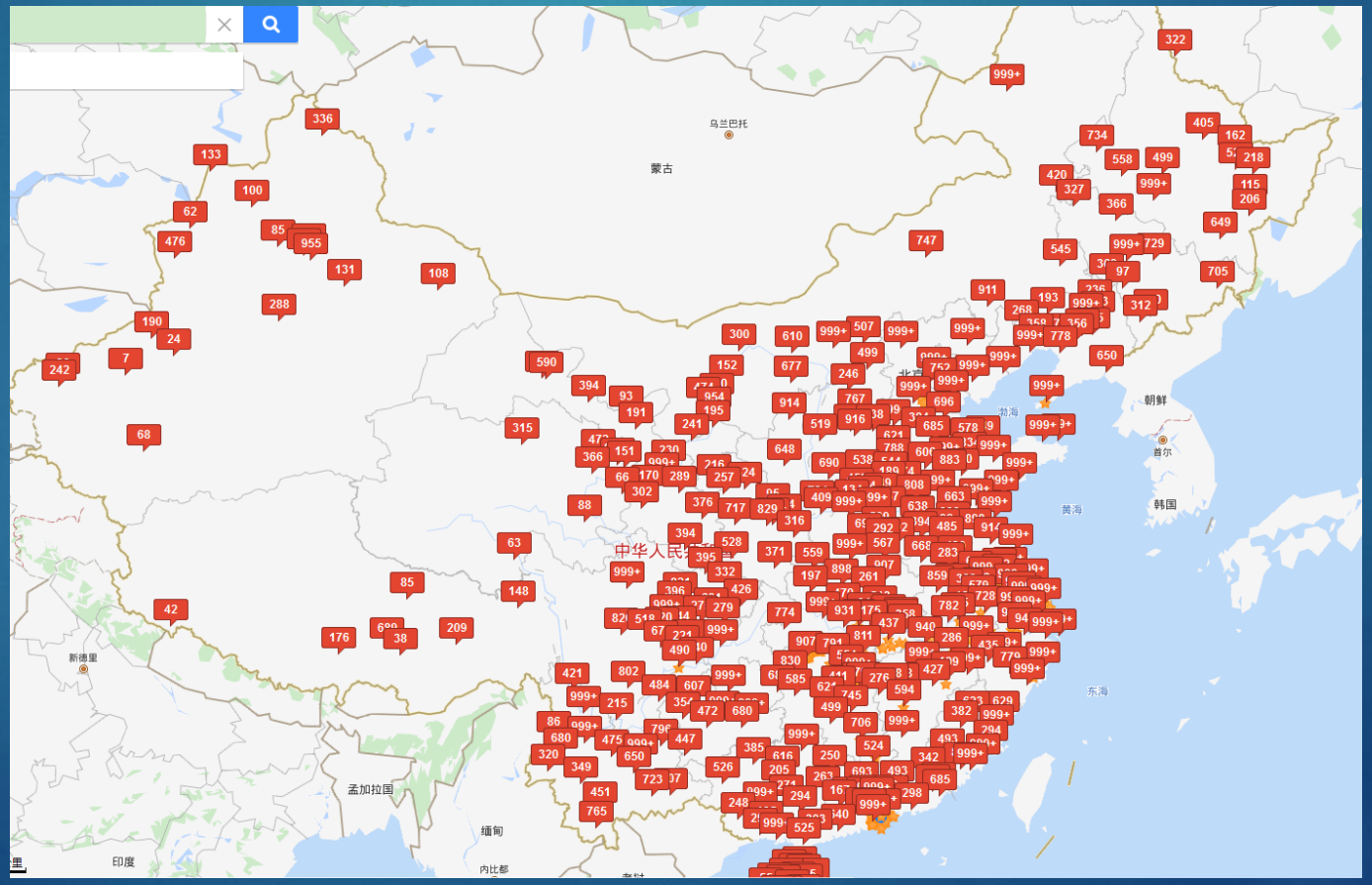
- 个人
  - 早期基于Lucene开发垂直搜索引擎
  - 2013年开始接触Elasticsearch 解决索引构建、部署、切换问题
  - 项目中大量使用解决各种业务需求：统计分析，业务搜索
- 案例应用
  - 最大规模集群：电子工厂生产线日志，数据量PB级；基于HDP平台做交叉分析；
  - 最有趣味项目：报表实时聚合查询
- 咨询培训
  - 业余给企业做Elastic-stack咨询培训

大纲

- 业务场景
- 技术选型
- 技术实践
- 聚合原理
- 经验总结

业务场景

- 公司属于物流速运行业，面向企业客户
- 下单客户的数量分布，
- 多种视角：全国、省、市、区，如下图



业务场景

- 在地图中展示下单客户数据量的分布情况
- 单日客户下单量XXX**万（此处省略一万字）**
- 支持多种维度组合查询

行政区域	四级行政区域：省、市、区、镇；数量5000+
组织架构	多层级组织架构：大区、小区；数量3000+
企业类型	企业类型划分：TO-B,TO-C,TO-X；数量10+
行业类型	企业行业类似：家居、服装；数量100+
业务类型	企业业务类型 ：A、B；数量2+
日期范围	日期滑动窗口：1～31天；

业务场景

- 业务数据模型：
  - 单个客户每日会下单多次，记录多次订单
  - 单个客户每日下单多次，也算做1个客户

原始业务数据模型

下单时间	客户编号	行政区域（4级）	企业类型	行业类型	业务类型	组织架构（多级）
2019-04-10 09:09:09	A001	广东省/深圳市/宝安区/沙井镇	B2B(编号 B001)	电子(编号C001)	寄件（0/1/2）	华东/浙江/杭州/上城/xxx市场部
2019-04-10 10:10:10	A001	广东省/深圳市/宝安区/沙井镇	B2B(编号 B001)	电子(编号C001)	收件（0/1/2）	华东/浙江/杭州/上城/xxx市场部
2019-04-10 11:11:11	A001	广东省/深圳市/宝安区/沙井镇	B2B(编号 B001)	电子(编号C001)	收件（0/1/2）	华东/浙江/杭州/上城/xxx市场部

业务需求数据模型

下单日期	客户编号	行政区域（4级）	企业类型	行业类型	业务类型	组织架构（多级）
2019-04-10	A001	广东省/深圳市/宝安区/沙井镇	B2B(编号 B001)	电子(编号C001)	寄件（0/1/2）	华东/浙江/杭州/上城/xxx市场部

业务场景

- 系统需求描述
  - 限定日期滑动窗口范围，最多1个月（X<=31天）
  - 历史数据可以查询过去1年（13个月）
  - 接口查询性能支持并发数50+，响应时间要求秒级内
  - 按照多个维度组合聚合返回客户数量
    - 行政区域
      - 全国视图，看各个省份的下单客户数量（X<=30）
      - XX省视图，XX市视图，XX区试图
    - 业务类型
      - 寄/收（X<=2）
      - 如果同一客户同时有2种类型，则按照1种类型合并处理，客户数依然记为1个

业务需求数据模型

行政区域（四级）	企业类型	行业类型	业务类型	客户数量
广东省/深圳市/宝安区/XX镇	B2B(编号 B001)	电子(编号C001)	寄件（0/1/2）	10000
广东省/深圳市/福田区/YY镇	B2c(编号 B002)	家居(编号C001)	收件（0/1/2）	20000

业务场景

- 业务问题需求抽象
  - 按照客户维度去重
  - 按照其它维度聚合
- 技术问题抽象本质
  - 两次聚合
    - distinct
    - group
  - 基数统计
    - 多种维度的基数

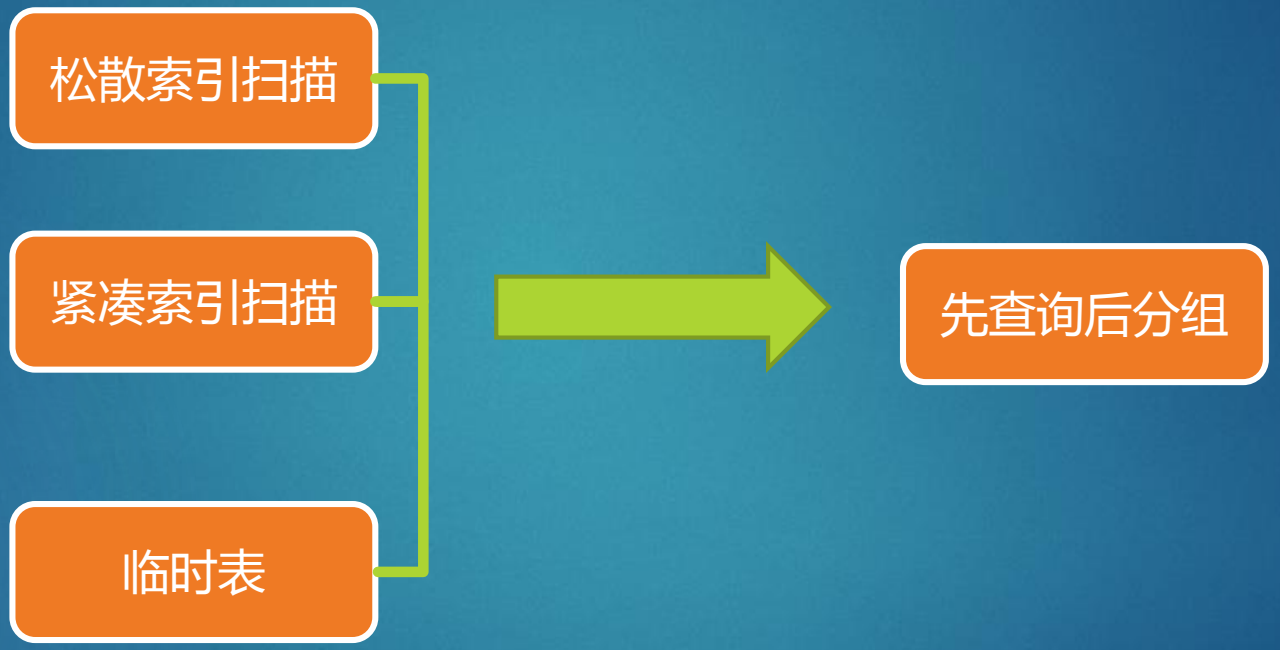
如何实现以上业务需求？

如何满足查询性能需求？

大纲

- 业务场景
- 技术选型
- 技术实践
- 聚合原理
- 经验总结

MySQL实现原理

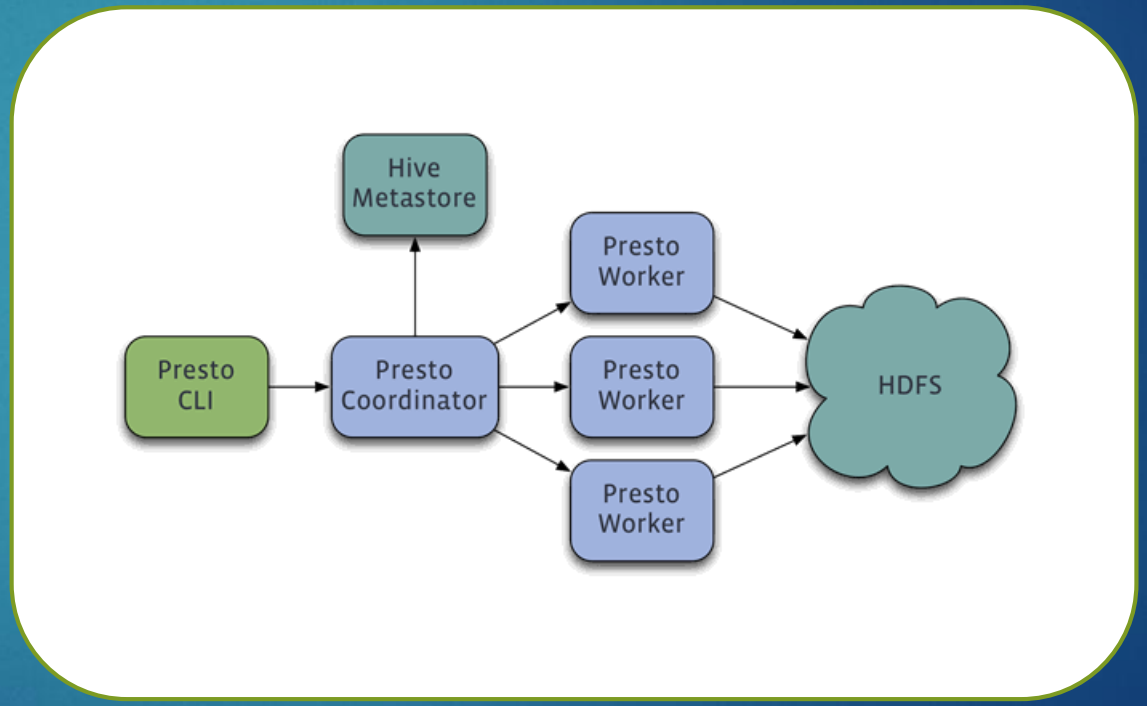


MySQL问题

- 单机资源
- 执行时间
- TPS/QPS
- 执行时间
- 基数问题
- 数据量问题

Prestodb实现原理

- Hive
  - 存储业务数据
- Presto worker
  - 执行计算

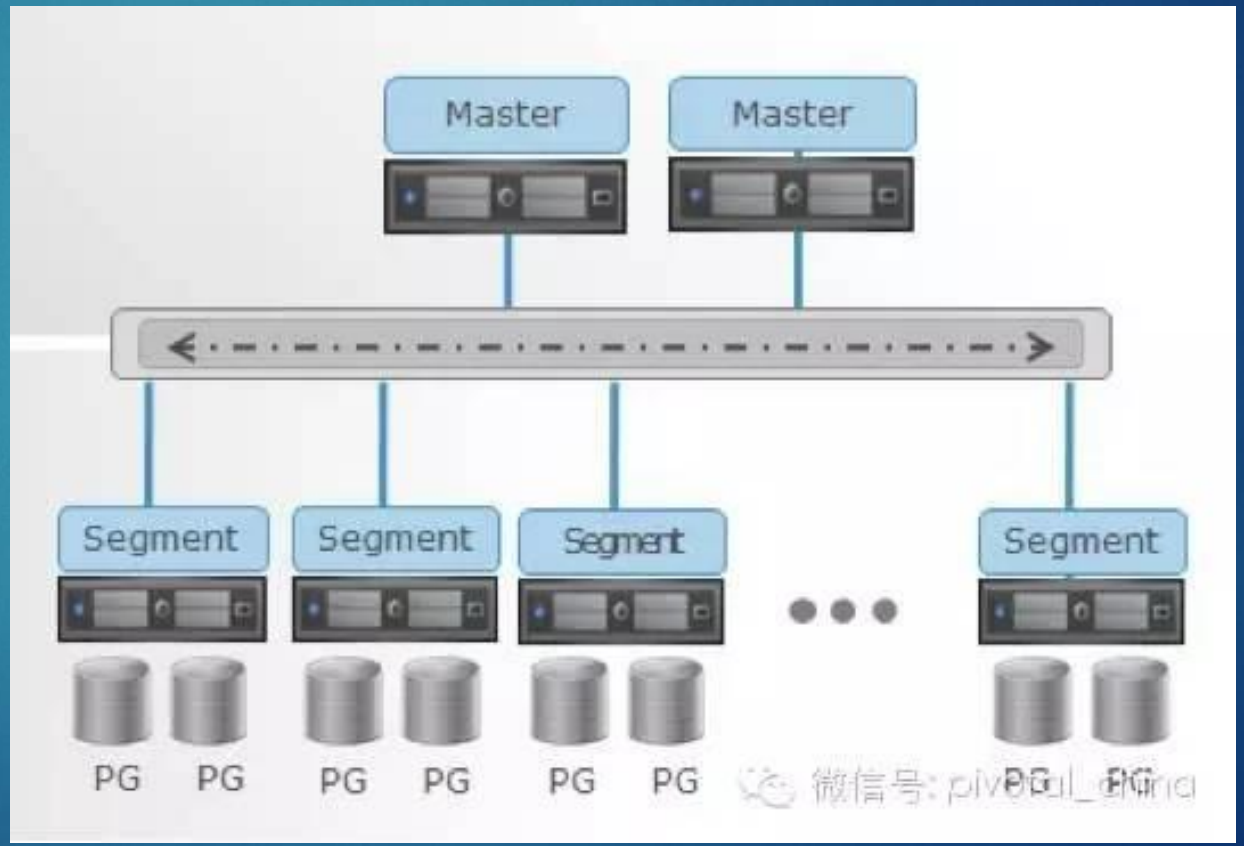


Prestodb问题

- 配置
  - 28个
  - 单实例5GB内存
- TPS/QPS
  - 至少2秒起，10s以上的也大量出现
- 并发数限制
  - 超过并发数限制，查询任务挂起
- 资源限制
  - Hadoop集群资源共享冲突限制
  - 集群中其它业务资源消耗

Greenplum实现原理

- MPP数据库
  - 数据吞吐量大
- 并行处理
  - 并行计算
- Pipeline计算模型
  - 内存驻留
  - 性能强于Hive十倍

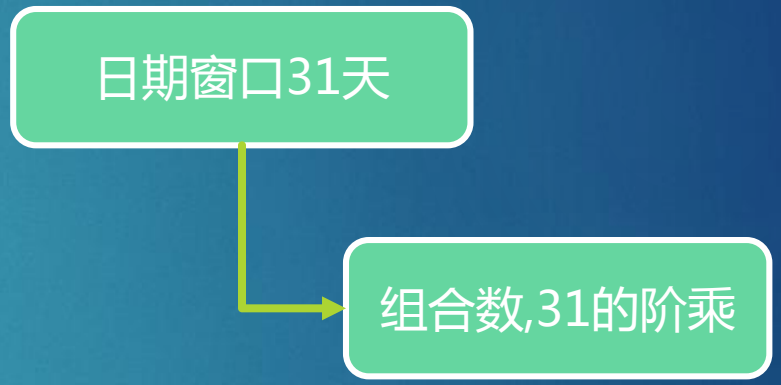


Greenplum问题

- OLAP场景
  - OLAP适合分析
  - 适用于数据仓库
- TPS/QPS
  - 至少1秒起
- 基数问题
  - 分组数量多
- 并发数限制
  - 超过并发数挂起
- MPP平台限制
  - 不适合高频率查询

XXXX穷举法

- 预计算所有查询条件组合
  - 所有维度
  - 几万种组合
  - 日期范围查询
    - 时间窗口31天
    - 组合数为31的 阶乘数 XXXX=31！
- 考虑过kylin
  - cube模型
    - 不适用此业务场景



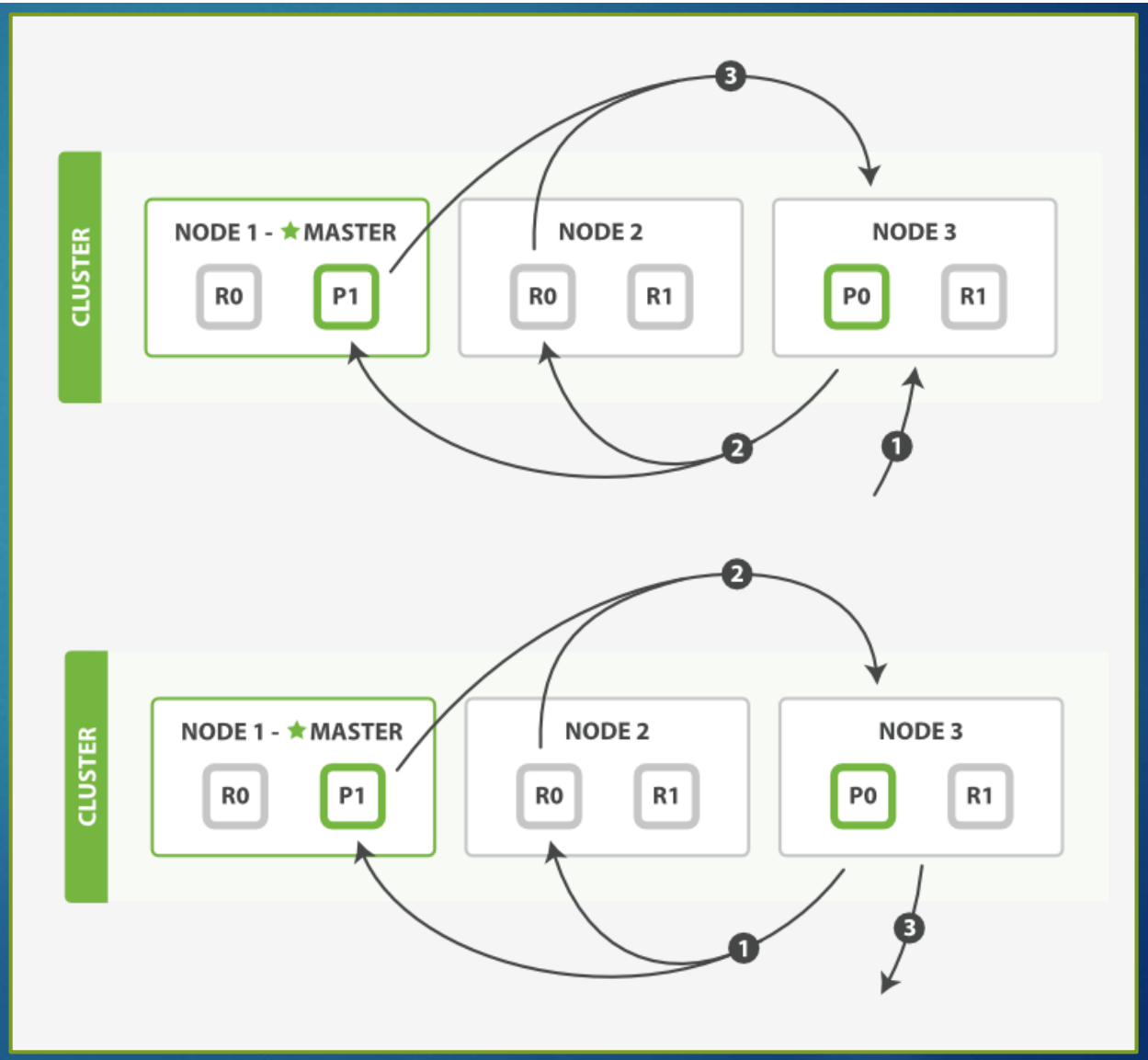
XXXX穷举法问题

- 数据量问题
  - 现有数据量
    - 每日下单客户数XXX万
- 维度数问题
  - 已知多个维度
  - 每个维度有多种枚举
  - 增加新维度需要重新计算
  - 每日计算量太大

	计算维度	计算维度数量
	行政区域	31（省/非港澳台）
	企业类型	10
	行业类型	100
	业务类型	2
	日期窗口	31阶乘
合计		31×10×100×2×31阶乘=XXXX

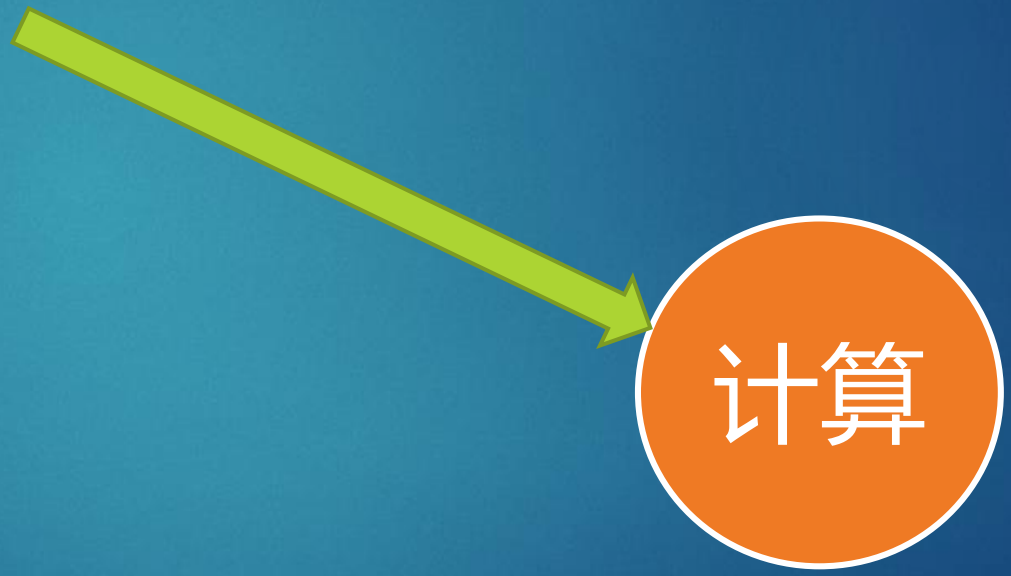
Elasticsearch实现原理

- 聚合计算
  - bucket 桶聚合
    - 分组
  - Term –bucket聚合
    - 多维度分组
- 并行处理
  - 分布式并行计算
  - 聚合-计算-合并



Elasticsearch问题

- 深度二次聚合问题
  - 业务需求
    - 首先要按照客户维度去重，获取筛选条件内符合的所有客户数量
      - 数量在XXX万
      - 基数统计
        - 基础统计下不接受子聚合
        - 概率问题
      - 数据模型简单，计算代价大
    - 按照多个维度分组统计客户数据量
      - 省、市、区、镇
      - 企业类型
      - 行业类型
      - 业务类型
- 单层聚合性能
  - 性能强悍、速度极快、响应速度、高频率查询

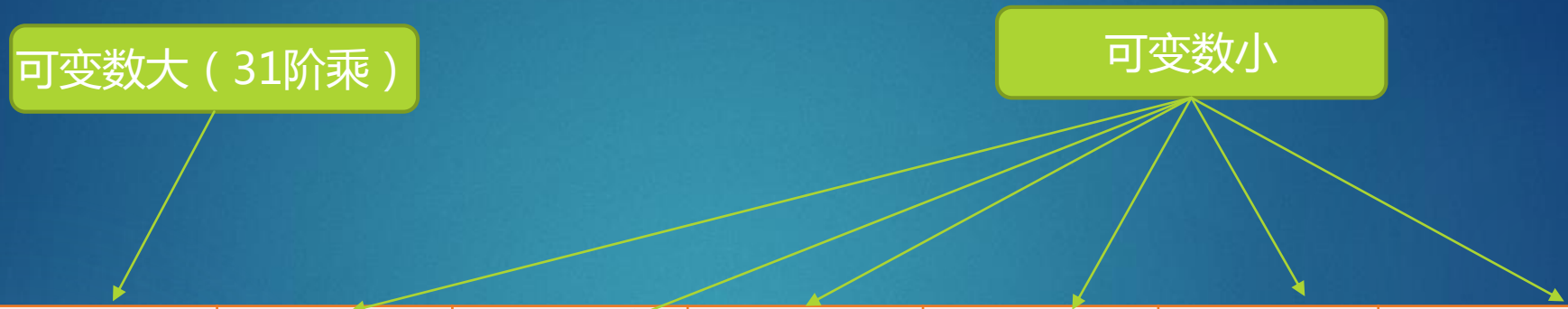


如何解决业务与技术问题？

大纲

- 业务场景
- 技术选型
- 技术实践
- 聚合特性
- 经验总结

数据模型



下单日期	客户编号	行政区域（4级）	企业类型	行业类型	业务类型	组织架构（多级）
2019-04-01	A001	广东省/深圳市/宝安区/沙井镇	B2B(编号 B001)	电子(编号C001)	寄件（0/1/2）	华东/浙江/杭州/上城/xxx市场部
2019-04-10	A001	广东省/深圳市/宝安区/沙井镇	B2B(编号 B001)	电子(编号C001)	收件（0/1/2）	华东/浙江/杭州/上城/xxx市场部

- 问题转换
  - 日期范围查询条件最大窗口31天，条件组合数量太多
  - 业务数据按行式存储，一个客户一个月最多有31条数据，按照31天范围聚合只能算做1个客户；
  - 能否基于客户维度 将1～31天的下单组合用列来存储，因为只算1个客户？
    - 用1条数据31列 来表达客户31天下单组合

数据模型

下单日期	客户	业务
2019-03-01	A001	1
2019-03-02	A001	2
2019-03-03	A001	1
2019-03-XX	A001	0/1/2
….	….	….
2019-03-31	A001	1

客户	下单日期	day 1	day2	day3	day4	day5	day6	day7	Day….X	day31
A001	2019-03-01	1	2	1	0	0	1	2	0 或 1或 2	2

数据模型

下单日期	客户	业务
2019-03-01	A001	1
2019-03-02	A001	2
2019-03-03	A001	1
2019-03-XX	A001	0/1/2
…	…	…
2019-03-31	A001	

行转列

客户	下单日期	day 1	day2	day3	day4	Day5	Day6	day7	Day….X	day31
A001	2019-03-01	1	2	1	0	0	1	2	0 或 1或 2	2

列合并

day4状态为2（day1到day4组合）

客户	下单日期	day 1	day2	day3	day4	day5	day6	day7	Day….X	day31
A001	2019-03-01	1	2	2	2	2	2	2	2	2

数据模型

- 查询过程
  - 先定位起始日期，按照下单日期
  - 依据结束日期获取指定的合并列，按照数据列
  - 多维聚合返回
- 举例：
  - 1：查询日期范围2019-03-01~2019-03-05的聚合数据

1：定位起始日期

2：指定数据列

客户	下单日期	day 1	day2	day3	day4	day5	day6	day7	Day….X	day31
A001	2019-03-01	1	2	2	2	2	2	2	2	2
A002	2019-03-01	1	2	2	2	2	2	2	2	2
A003	2019-03-02	1	2	2	2	2	2	2	2	2
A004	2019-03-02	1	2	2	2	2	2	2	2	2

数据模型

- 空间换时间
  - 每客户每天1条数据
    - 每天按照全量客户构建数据
  - 存储未来31天的组合数据
    - 数据刷新需要更新过去31天
- 矩阵
  - 行转列
    - 数据条数变少
  - 列合并
    - 固化变化

大数据平台：Hive

- Hive分工
  - 每日全量构建客户过去31天下单记录
    - 未产生业务交易的客户也需要补足，标记为0
  - 客户每日下单记录行转列
    - 31天数据（31条记录）转换为1条数据31列
    - 构建原始行转列矩阵
  - 客户每日下单记录列合并
    - 按照31天组合，以开始日期作为起点，往后推演31天，包括当前起始日期
    - 矩阵合并
  - 数据Hive To Elasticsearch
    - 基于Hive映射表，推送到Elasticsearch
    - 每日推送全量客户31天数据

大数据平台：Hive

- Hive行转列

stat_date	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15	d16	d17	d18	d19	d20	d21	d22	d23	d24	d25	d26	d27	d28	d29	d30	d31	l
2018-12-01	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
2018-12-02	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	
2018-12-03	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	
2018-12-04	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	
2018-12-05	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	
2018-12-06	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	
2018-12-07	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	
2018-12-08	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	
2018-12-09	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	
2018-12-10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	
2018-12-11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	
2018-12-12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	
2018-12-13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	
2018-12-14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	
2018-12-15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2018-12-16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2018-12-17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2018-12-18	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2018-12-19	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2018-12-20	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2018-12-21	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2018-12-22	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2018-12-23	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2018-12-24	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2018-12-25	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2018-12-26	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2018-12-27	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2018-12-28	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2018-12-29	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2018-12-30	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2018-12-31	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2019-01-01	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2019-01-02	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2019-01-03	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2019-01-04	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2019-01-05	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2019-01-06	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2019-01-07	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2019-01-08	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2019-01-09	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2019-01-10	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2019-01-11	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2019-01-12	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2019-01-13	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2019-01-14	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2019-01-15	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2019-01-16	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2019-01-17	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2019-01-18	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2019-01-19	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

大数据平台：Hive

- **Hive矩阵列合并**

```
greatest(d1, d2) as d2,
greatest(d1, d2, d3) as d3,
greatest(d1, d2, d3, d4) as d4,
greatest(d1, d2, d3, d4, d5) as d5,
greatest(d1, d2, d3, d4, d5, d6) as d6,
greatest(d1, d2, d3, d4, d5, d6, d7) as d7,
greatest(d1, d2, d3, d4, d5, d6, d7, d8) as d8,
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9) as d9,
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10) as d10,
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11) as d11,
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12) as d12,
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13) as d13,
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14) as d14,
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15) as d15,
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16) as d16,
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17) as d17,
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, d18) as d18,
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, d18, d19) as d19,
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, d18, d19, d20) as d20,
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, d18, d19, d20, d21) as d21,
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, d18, d19, d20, d21, d22) as d22,
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, d18, d19, d20, d21, d22, d23) as d23,
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, d18, d19, d20, d21, d22, d23, d24) as d24,
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, d18, d19, d20, d21, d22, d23, d24, d25) as d25
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, d18, d19, d20, d21, d22, d23, d24, d25, d26) a
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, d18, d19, d20, d21, d22, d23, d24, d25, d26,
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, d18, d19, d20, d21, d22, d23, d24, d25, d26,
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, d18, d19, d20, d21, d22, d23, d24, d25, d26,
greatest(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11, d12, d13, d14, d15, d16, d17, d18, d19, d20, d21, d22, d23, d24, d25, d26,
```

大数据平台：Elasticsearch

- ES分工
  - 按月度创建Index
    - 参照客户下单起始日期
  - 按月度索引查询聚合
    - 参照客户下单日期定为月度索引

性能指标

单次聚合<100ms

TPS>200

QPS>200

并发>100

Elasticsearch聚合内部原理？

大纲

- 业务场景
- 技术选型
- 技术实践
- 聚合原理
- 经验总结

bucket 聚合类型

- 名词解释
  - 桶聚合
  - 类似分组
- 执行过程
  - 优先创建bucket
  - 数据填充bucket
    - 创建子bucket
    - 填充计数
- 优点
  - 节约内存资源
    - 先创建分组后查询，节约内存资源
  - 对比mysql先查询后分组
    - 查询所有数据放在内存，消耗过多内存资源

doc_values数据结构

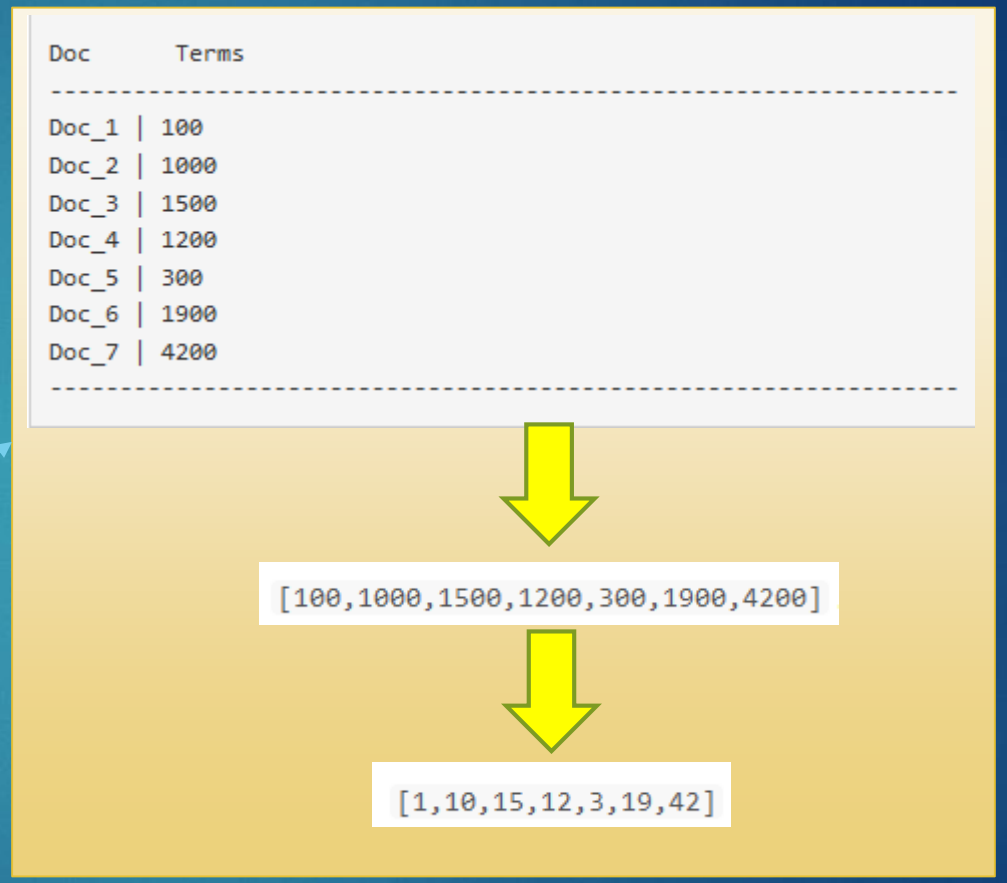
- 名词解释
  - 数据存储结构
  - 非倒排序索引数据结构
  - 正排索引
- 倒排序优势
  - 快速搜索文档
  - lucene底层是基于什么排序？
- 倒排序问题
  - 字段值排序
  - 字段值聚合
  - 字段值过滤
    - Gps范围搜索
  - 字段值脚本运算

Term	Doc_1	Doc_2	Doc_3
brown	X	X	
dog	X		X
dogs		X	X
fox	X		X
foxes		X	
in		X	
jumped	X		X
lazy	X	X	
leap		X	
over	X	X	X
quick	X	X	X
summer		X	
the	X		X

Doc	Terms
Doc_1	brown, dog, fox, jumped, lazy, over, quick, the
Doc_2	brown, dogs, foxes, in, lazy, leap, over, quick, summer
Doc_3	dog, dogs, fox, jumped, over, quick, the

doc values数据结构

- 数据结构
  - 列式存储
  - 存储数据原始值
  - 数据存储在磁盘上
- 使用限制
  - 默认所有字段开启
  - 非text/分词字段开启
- 列示特性
  - 列式压缩
    - 数字类型
      - 最大公约数机制
      - 偏移量编码
    - 字符串类型
      - 全局序号映射编码
  - 列式汇总



global ordinals数据结构

- 名词解释
  - 全局序号
  - 序列映射term
- 业务场景
  - 默认启用
  - 数据条目多
    - 百万级以上
  - term数量多
    - 万以上
- 优点
  - 海量数据聚合
  - 内存资源消耗少
    - 聚合时需要在内存创建所有的term-bucket；
    - 通过序号映射，减少内存中term的字符长度，降低内存消耗
  - 运行速度快

```
GET /_search
{
    "aggs" : {
        "tags" : {
            "terms" : {
                "field" : "tags",
                "execution_hint": global_ordinals 1
            }
        }
    }
}
```

global ordinals数据结构

- 生成机制
  - Keyword类型默认生效
  - 默认与倒排索引同步生成
  - 存储在doc_values数据结构中
- 生效机制
  - 默认延迟加载
  - 查询时第一次加载内存
    - 后续查询速度很快，生成doc_value
  - 可设置提前加载到内存
    - mapping中设置
    - 影响实时写入性能
  - 禁用doc_values
    - 减少资源消耗

```
PUT my_index/_mapping
{
  "properties": {
    "tags": {
      "type": "keyword",
      "eager_global_ordinals": true
    }
  }
}
```

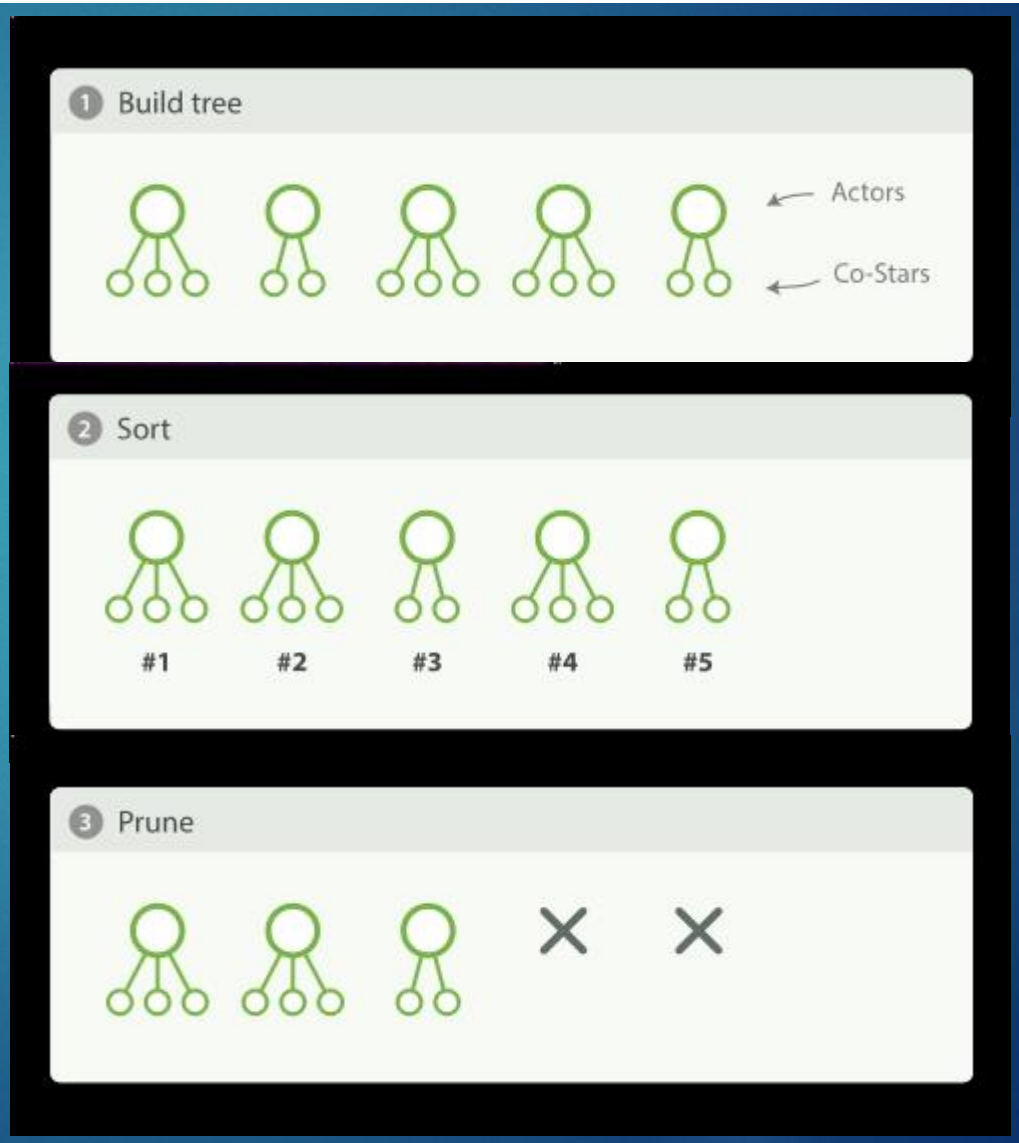
map数据结构

- 名词解释
  - 聚合采用map结构
  - 直接存储term
- 业务场景
  - 数据条目少
    - 百万以下
  - Term数量少
    - 万以下
- 优点
  - 直接创建map结构存储term聚合
  - 避免创建globe ordinals序号映射
  - 在数据量少的情况下，速度要更快

```
GET /_search
{
    "aggs" : {
        "tags" : {
            "terms" : {
                "field" : "tags",
                "execution_hint": "map" ❶
            }
        }
    }
}
```

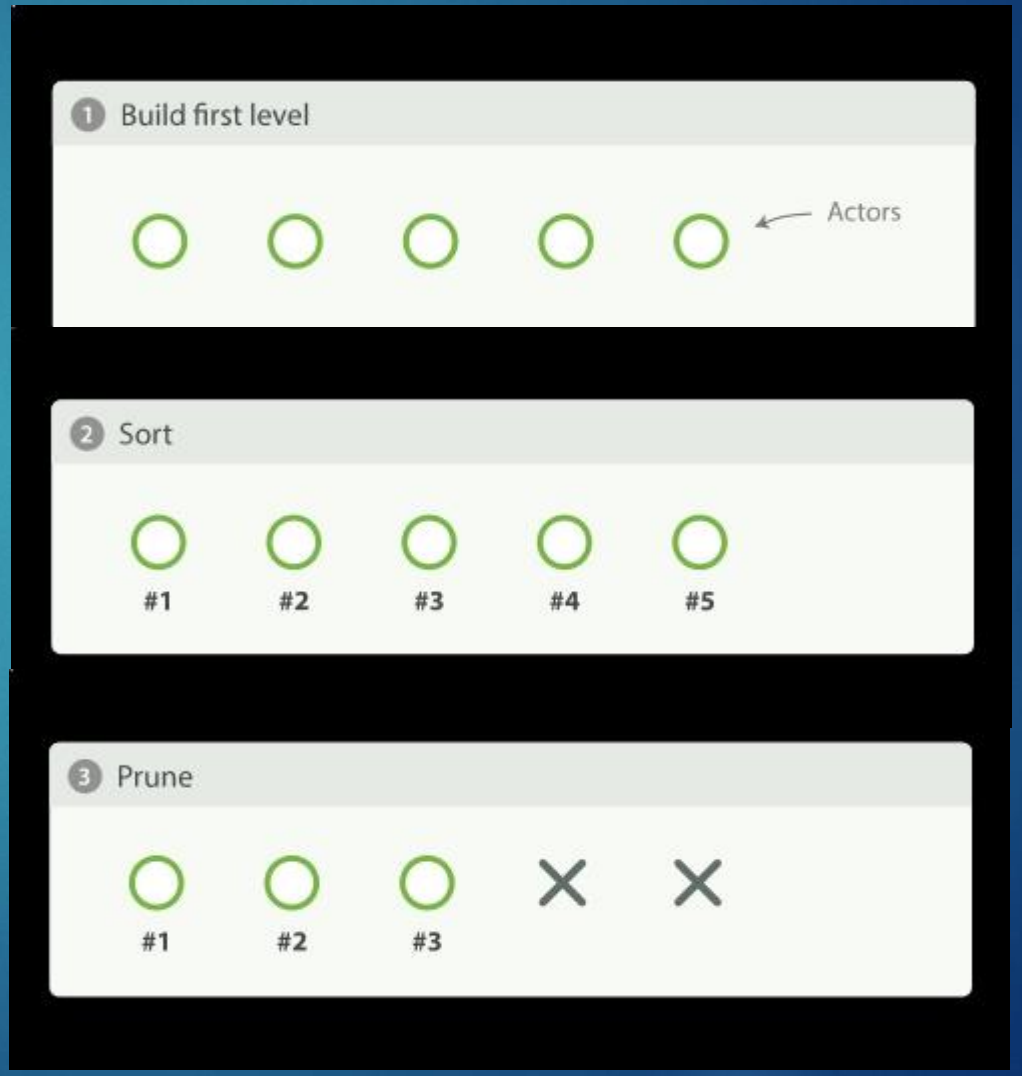
深度优先

- 名词解释
  - bucket聚合过程方式
  - term数据不大
  - 默认启用
    - 熊出没！！！！！！
- 业务场景
  - 桶数量固定
  - 文档数量多
  - 多维度聚合；桶嵌套
- 执行过程
  - 构建部分bucket桶
  - 按顺序构建bucket
  - 去掉Top-X之后的bucket



广度优先

- 名词解释
  - bucket聚合过程方式
  - term数据量大
  - 返回TopX的term-bucket
  - 内存线性增长关系
- 业务场景
  - 嵌套聚合
  - 取TopX桶聚合
    - TopX至少小于所有桶1/2
  - 聚合数量范围(term<=10000)
- 执行过程
  - 构建所有term-bucket桶
  - 排序
  - 去掉Top-X之后的bucket



深度优先/广度优先

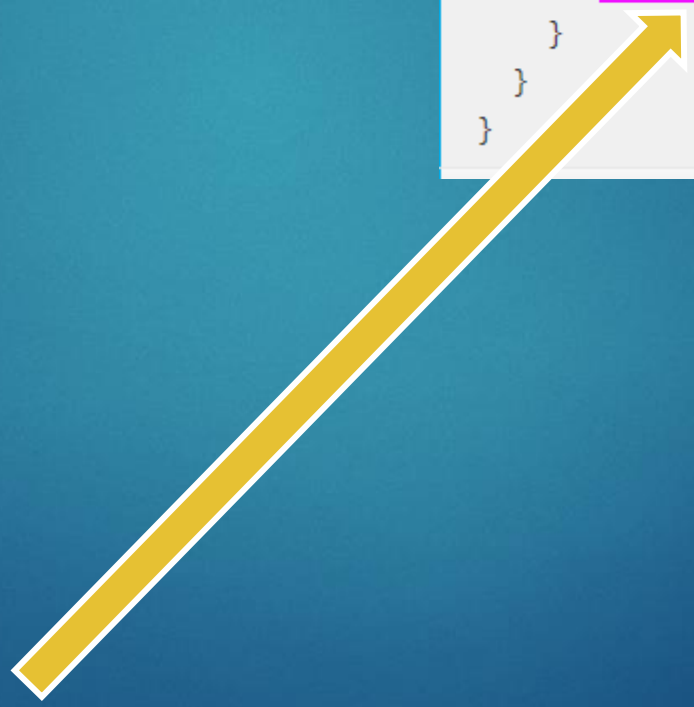
- 使用设置
  - collect_model
    - **breadth_first** 默认
    - depth_first

```
GET /_search
{
    "aggs" : {
        "actors" : {
            "terms" : {
                "field" : "actors",
                "size" : 10,
                "collect_mode" : "breadth_first" ❶
            },
            "aggs" : {
                "costars" : {
                    "terms" : {
                        "field" : "actors",
                        "size" : 5
                    }
                }
            }
        }
    }
}
```

fielddata特性

- 名词解释
  - 数据类型针对text类型
  - 基于分词且分多个词
  - 基于term-token做聚合
- 优点
  - 提前text类型分词且单独存储
  - 加载到heap堆中
  - 基于内存统计提高性能
- 限定条件
  - 基于某字段分词场景
  - term-bucket聚合方式
  - 需要配置生效

```
PUT my_index/_mapping
{
  "properties": {
    "my_field": { ❶
      "type":     "text",
      "fielddata": true
    }
  }
}
```



Elasticsearch其它聚合类型？

大纲

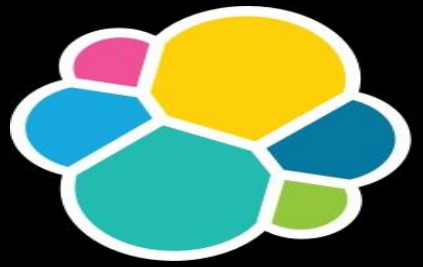
- 业务场景
- 技术选型
- 技术实践
- 聚合原理
- 经验总结

经验总结

- Elasticsearch能力边界
  - 深度二次聚合
  - 去重后聚合性能
  - 单一聚合性能
- 思维模型
  - 培养微观算法建模思维
  - 模型
  - 算法
  - 策略
- Elasticsearch应用结论
  - Elasticsearch+数据模型=无敌

ES在跨越其它应用场景

- 业务系统ERP
  - 业务查询
    - 分库分表合并查询问题
    - 宽表多条件查询
    - 复杂搜索条件/分词查询
  - 应用日志
    - 数据量日均TB级别以上
- 大数据BIGDATA
  - 报表需求
    - 大量聚合查询场景
  - 历史查询
    - 明细数据查询



Elastic

中文社区

谢谢 Thanks