SDCC
中国软件开发者大会
SOFTWARE DEVELOPER CONFERENCE CHINA

CSDN
全球最大中文IT社区

分布式数据库挑战与分析

- 卢亿雷 From AdMaster
- johnlya@163.com
- @johnlya

目录

- 分布式数据库介绍
- OLTP和OLAP对比分析
- 分布式数据库遇到的问题分析
- 分布式数据库实际案例分析
- AdMaster案例分析

- **原子性 (Atomic)**

  整个事务中的所有操作，要么全部完成，要么全部不完成，不可能停滞在中间某个环节。事务在执行过程中发生错误，会被回滚（Rollback）到事务开始前的状态，就像这个事务从来没有执行过一样。

- **一致性 (Consistent)**

  在事务开始之前和事务结束以后，数据库的完整性约束没有被破坏。

- **隔离性 (Isolated)**

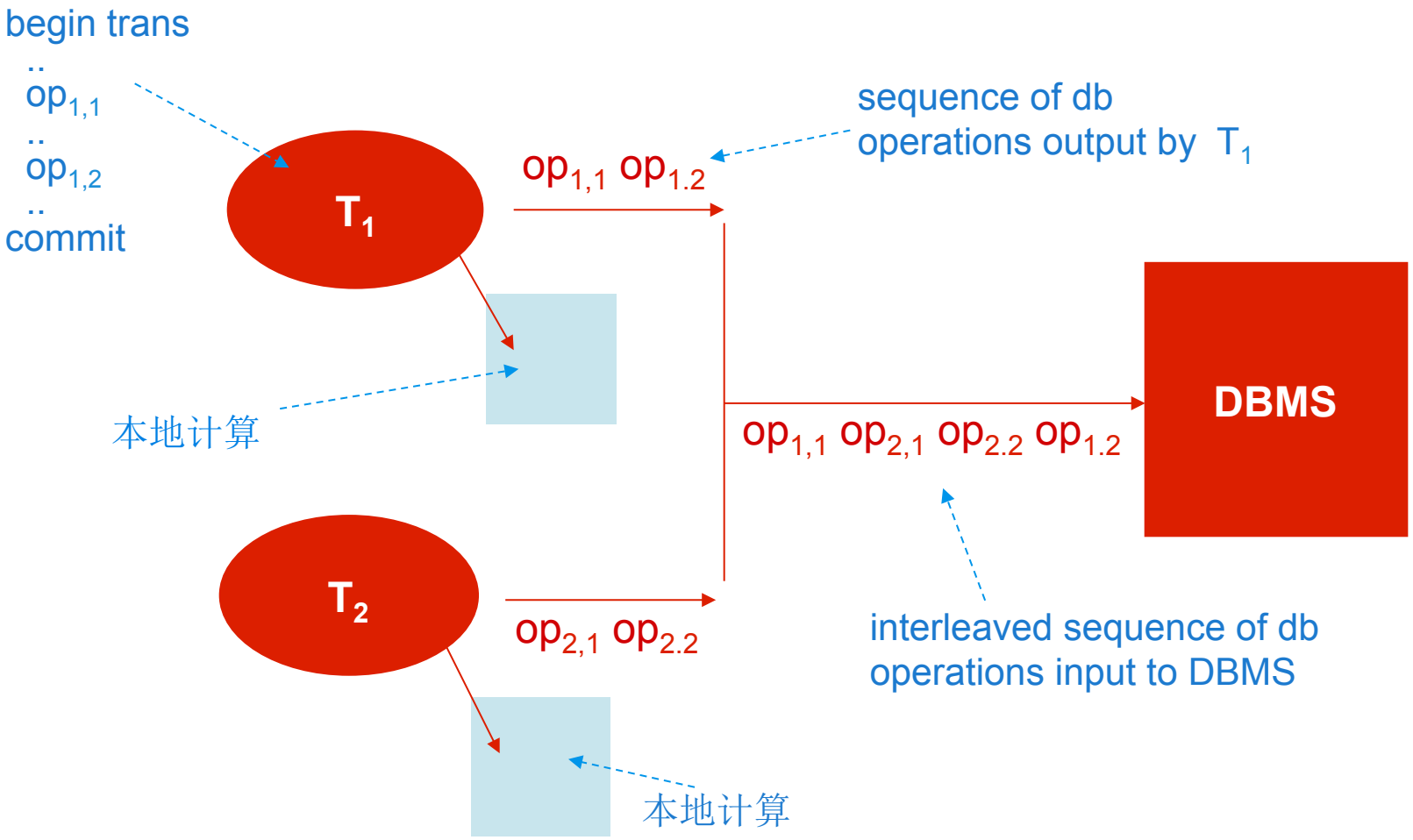
  隔离状态执行事务，使它们好像是系统在给定时间内执行的唯一操作。如果有两个事务，运行在相同的时间内，执行相同的功能，事务的隔离性将确保每一事务在系统中认为只有该事务在使用系统。这种属性有时称为串行化，为了防止事务操作间的混淆，必须串行化或序列化请求，使得在同一时间仅有一个请求用于同一数据。

- **持久性 (Durable)**

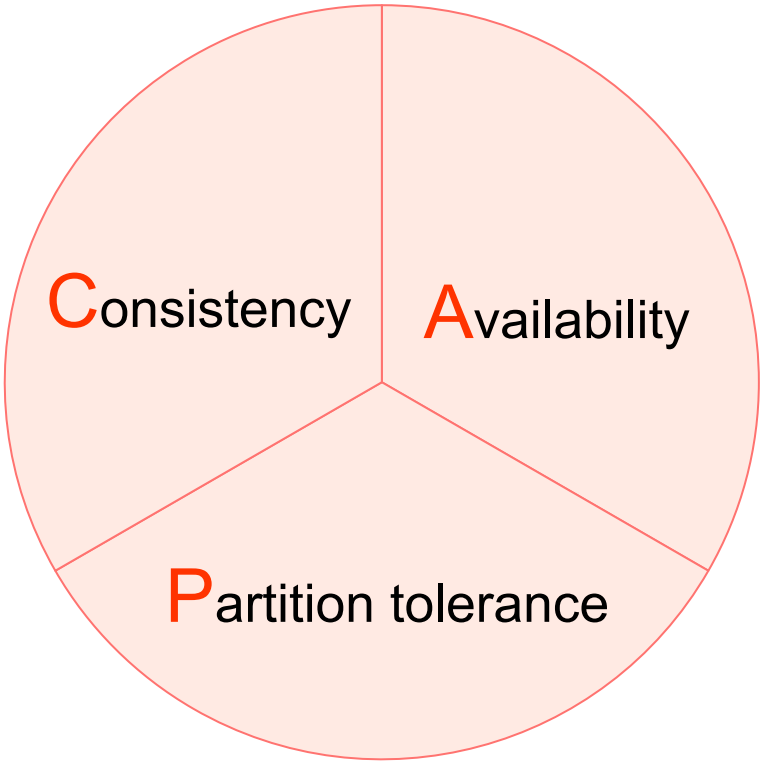
  在事务完成以后，该事务所对数据库所作的更改便持久的保存在数据库之中，并不会被回滚。

ACID实现方式： 第一种是Write ahead logging，也就是日志式的方式。
第二种是Shadow paging

分布式数据库介绍-隔离性



分布式数据库介绍-CAP



System is available during software and hardware upgrades and node failures.

一致性（C）：在分布式系统中的所有数据备份，在同一时刻是否同样的值。

可用性（A）：在集群中一部分节点故障后，集群整体是否还能响应客户端的读写请求。（可用性不仅包括读，还有写）

分区容忍性（P）：集群中的某些节点在无法联系后，集群整体是否还能继续进行服务。

- **Key/Value 或 ‘the big hash table’.**
  - Amazon S3 (Dynamo)
  - Voldemort
  - Scalaris
  - Memcached (in-memory key/value store)
  - Redis
- **Schema-less**
  - Cassandra (column-based)
  - CouchDB (document-based)
  - MongoDB(document-based)
  - Neo4J (graph-based)
  - HBase (column-based)
  - ElasticSearch(document-based)

OLTP和OLAP对比分析

- **联机事务处理OLTP（On-line transaction processing）**
  - OLTP是传统的关系型数据库的主要应用，主要是基本的、日常的事务处理，例如银行交易。
- **联机分析处理OLAP（On-Line Analytical Processing）**
  - OLAP是数据仓库系统的主要应用，支持复杂的分析操作，侧重决策支持，并且提供直观易懂的查询结果。

OLTP和OLAP对比分析

- **用户和系统的面向性**
  - 面向顾客（事务） VS. 面向市场（分析）
- **数据内容**
  - 当前的、详细的数据 VS. 历史的、汇总的数据
- **数据库设计**
  - 实体－联系模型(ER)和面向应用的数据库设计 VS. 星型/雪花模型和面向主题的数据库设计

OLTP和OLAP对比分析

- **数据视图**
  - 当前的、企业内部的数据 VS. 经过演化的、集成的数据
- **访问模式**
  - 事务操作 VS. 只读查询（但很多是复杂的查询）
- **任务单位**
  - 简短的事务 VS. 复杂的查询
- **访问数据量**
  - 数十个 VS. 数百万个

OLTP和OLAP对比分析

- **用户数**
  - 数千个 VS. 数百个
- **数据库规模**
  - 100M-数GB VS. 100GB-数TB
- **设计优先性**
  - 高性能、高可用性 VS. 高灵活性、端点用户自治
- **度量**
  - 事务吞吐量 VS. 查询吞吐量、响应时间

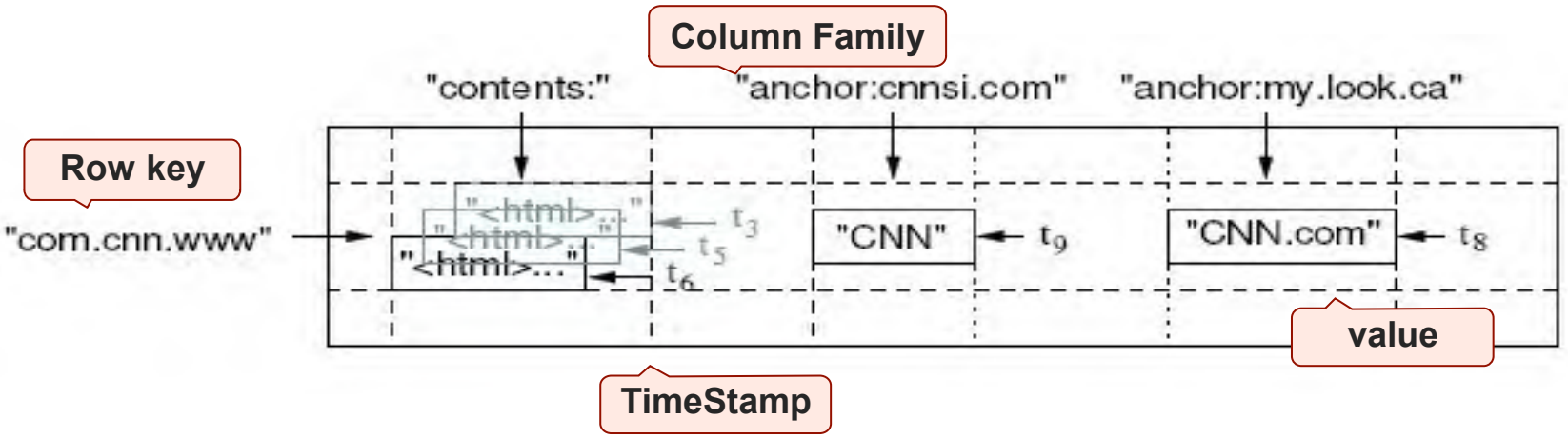
OLTP和OLAP对比分析

	OLTP	OLAP
用户	操作人员,低层管理人员	决策人员,高级管理人员
功能	日常操作处理	分析决策
DB 设计	面向应用	面向主题
数据	当前的, 最新的细节的, 二维的分立的	历史的, 聚集的, 多维的集成的, 统一的
存取	读/写数十条记录	读上百万条记录
工作单位	简单的事务	复杂的查询
用户数	上千个	上百个
DB 大小	100MB-GB	100GB-TB

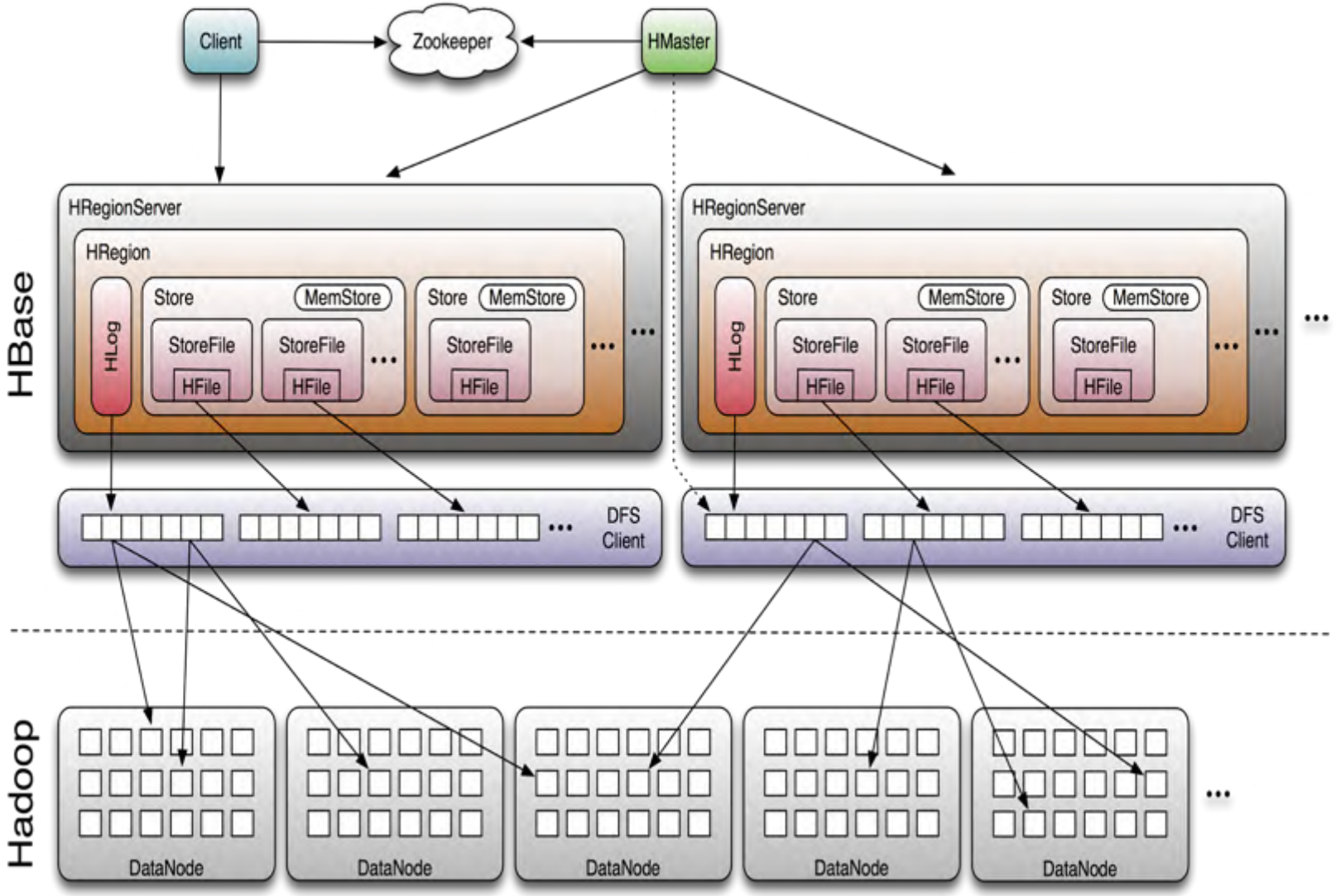
分布式数据库遇到的问题分析

- 如在SSD存储下怎么样提高数据库性能？
- 如怎么样在高可靠性的前提下提高数据库插入和查询性能？
- 如在支持分布式事务前提下如何提高数据调用性能？
- 如对几千至几万列的复杂查询提供秒级返回结果？

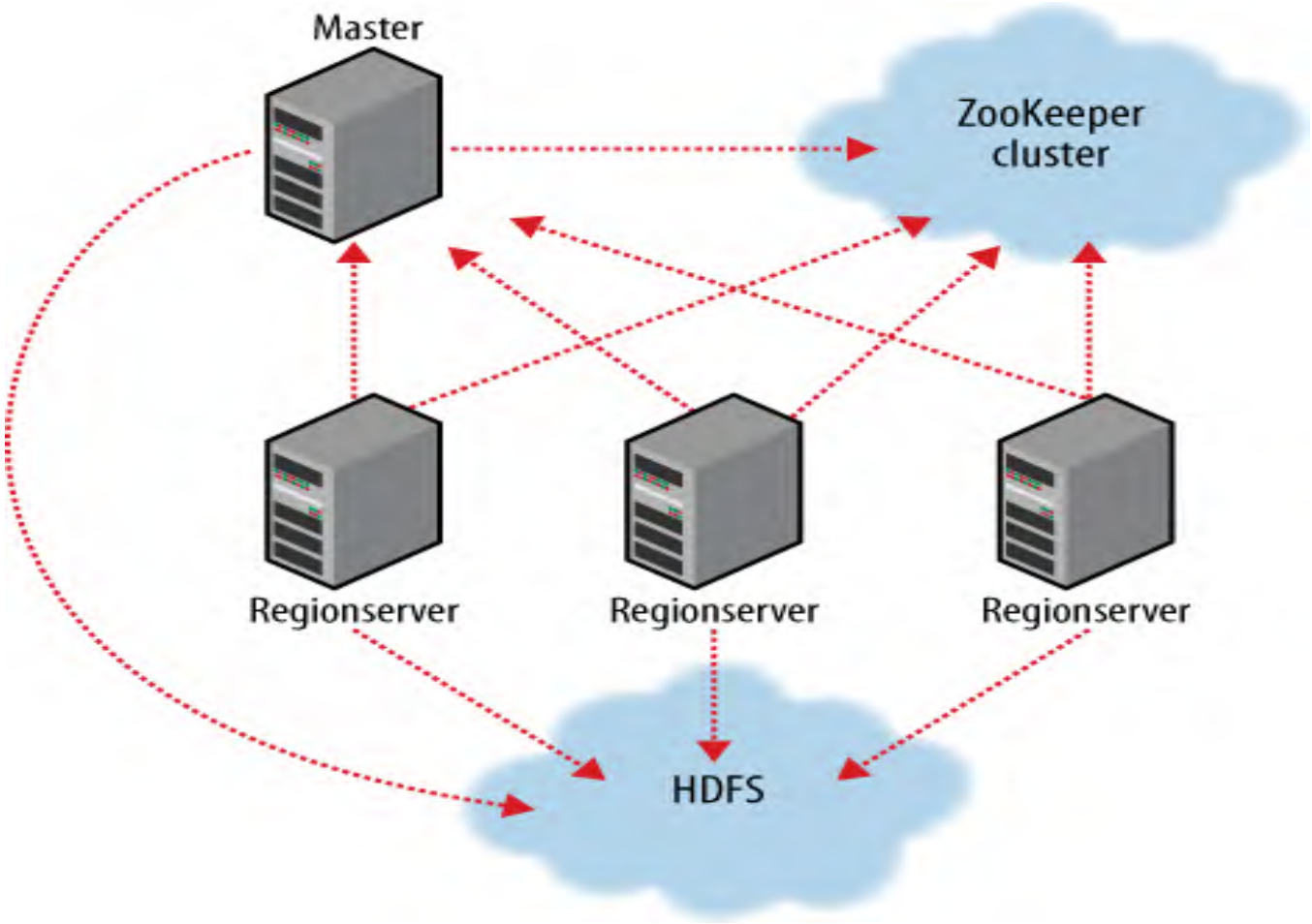
- **按列排序**
- **列族定义**
  - 每个列族可以包含任意多的列
  - 每个列可以有任意多的版本
  - 列只有在有值时才存在
  - 列本身是排序的
- **(Row, Family: Column, Timestamp) → Value**



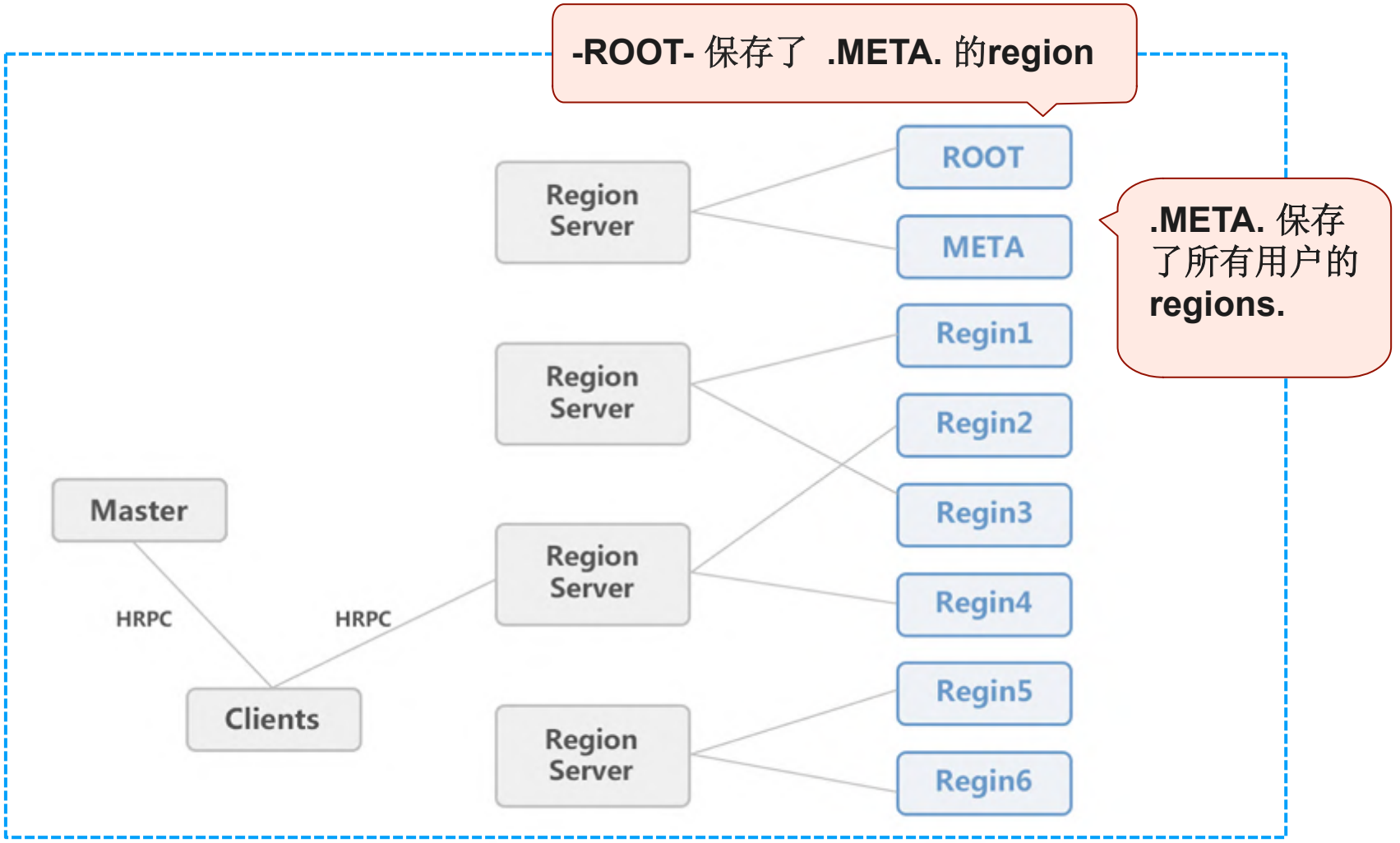
分布式数据库实际案例-HBase



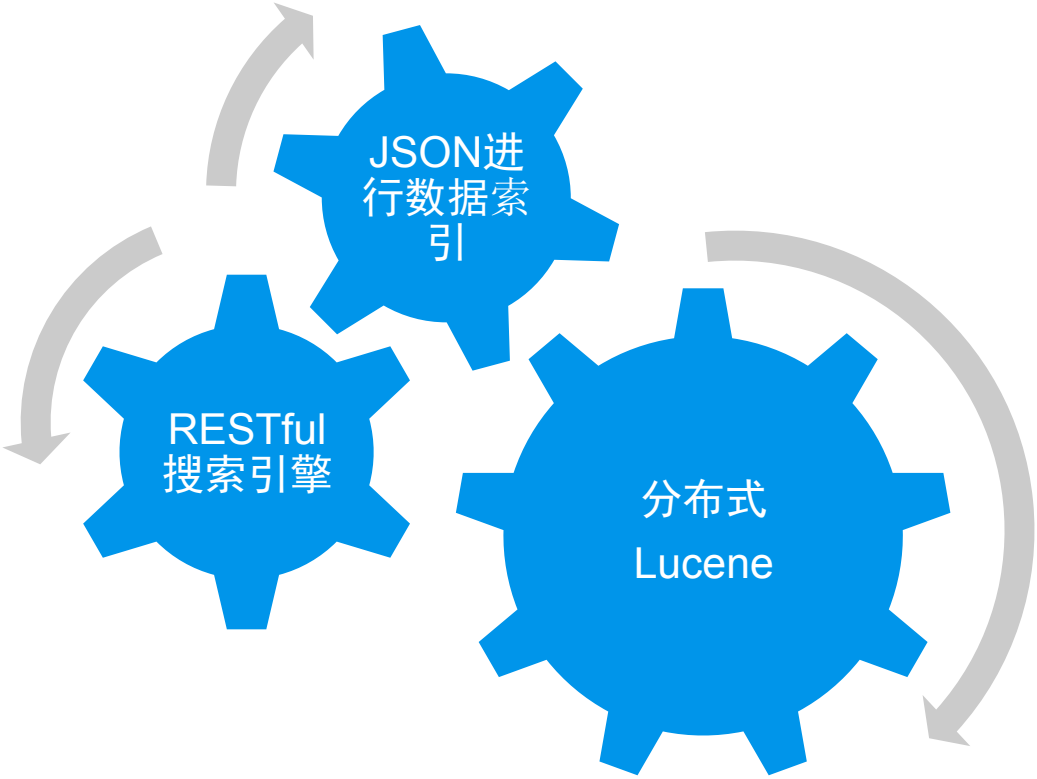
分布式数据库实际案例-HBase



分布式数据库实际案例-HBase

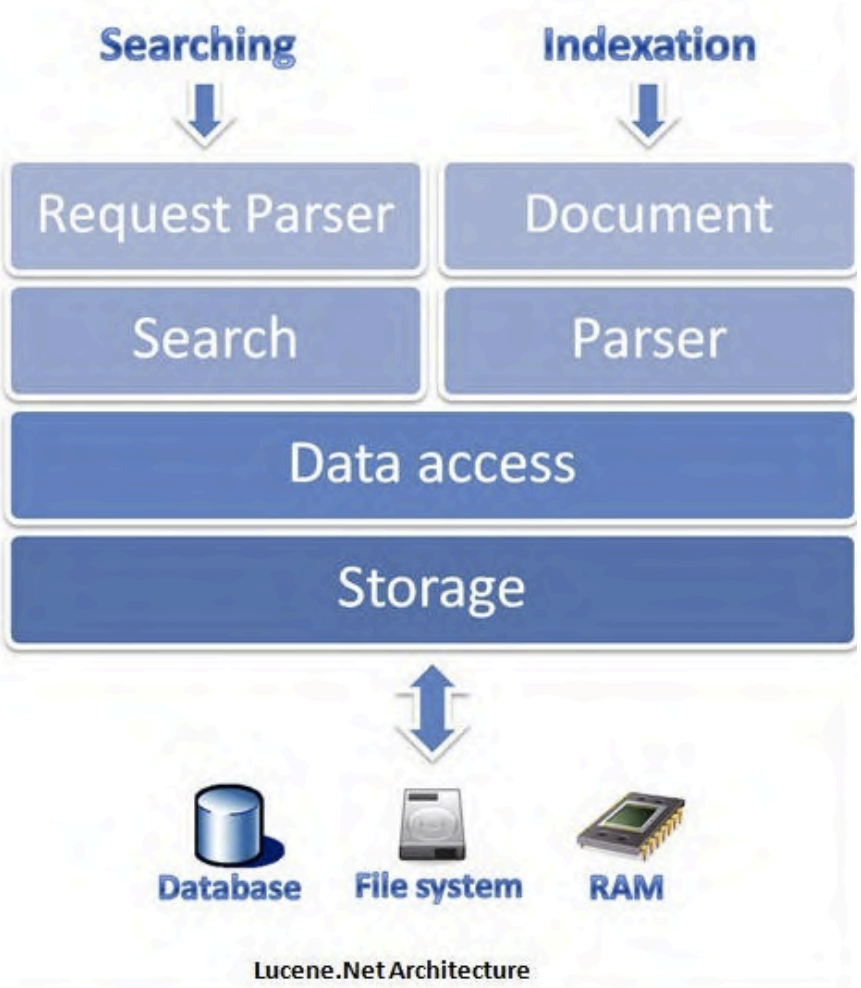


分布式数据库实际案例-ElasticSearch

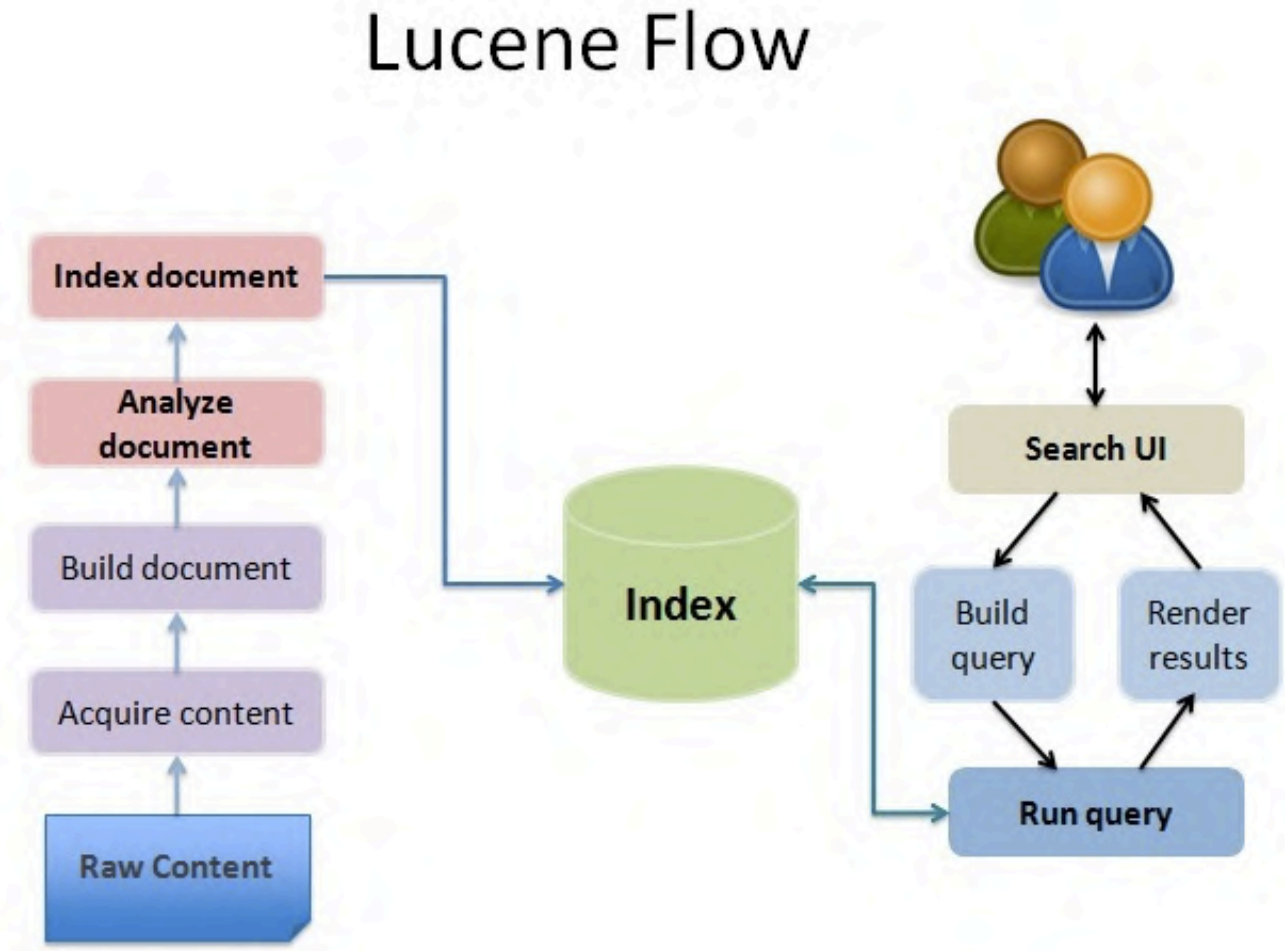


Github 使用 Elasticsearch 搜索 20TB 的数据，包括 13 亿的文件和 1300 亿行的代码

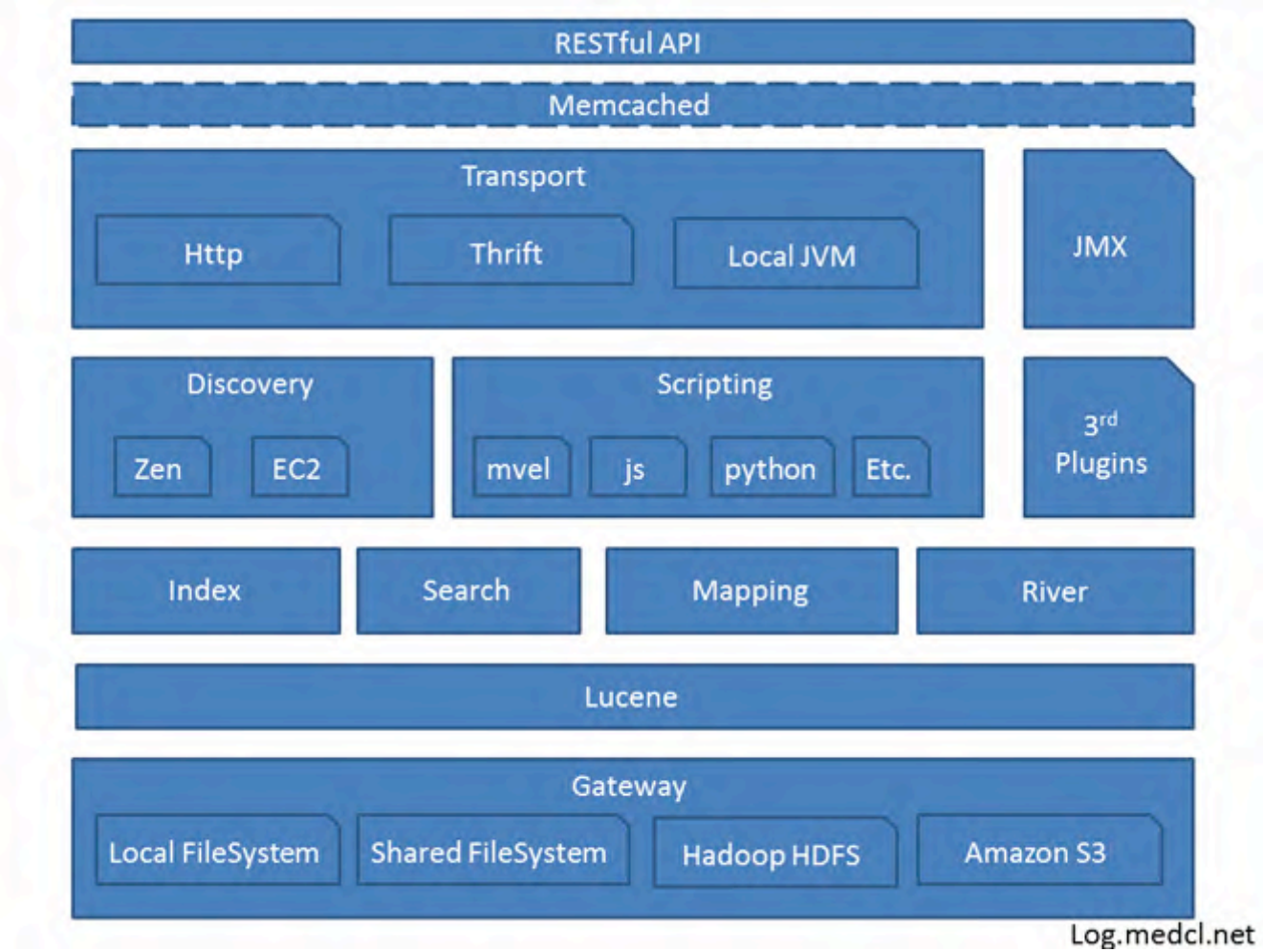
分布式数据库实际案例-ElasticSearch



Lucene.Net Architecture



分布式数据库实际案例-ElasticSearch



Log.medcl.net

Framework	API	Management	Storage Format
Druid	JSON-based HTTP calls	ZK	Bitmap indexing
Pinot	PQL	Apache-Helix/ZK	Multiple types

- 测试条件
  - 记录条数分为100亿以内和1000亿条
  - 服务器数量为70台，配置为:CPU 12核，内存96G，硬盘48T
  - 测试语句：select count(*) from test where age > 25 and gender > 0 and os > "500" and sc in ("0001009","0002036","0016030",”…") or bs>585 and group by age,gender,os,bs
  - 总共11列：动态列为3列（多值列），普通列为11列

- 100亿

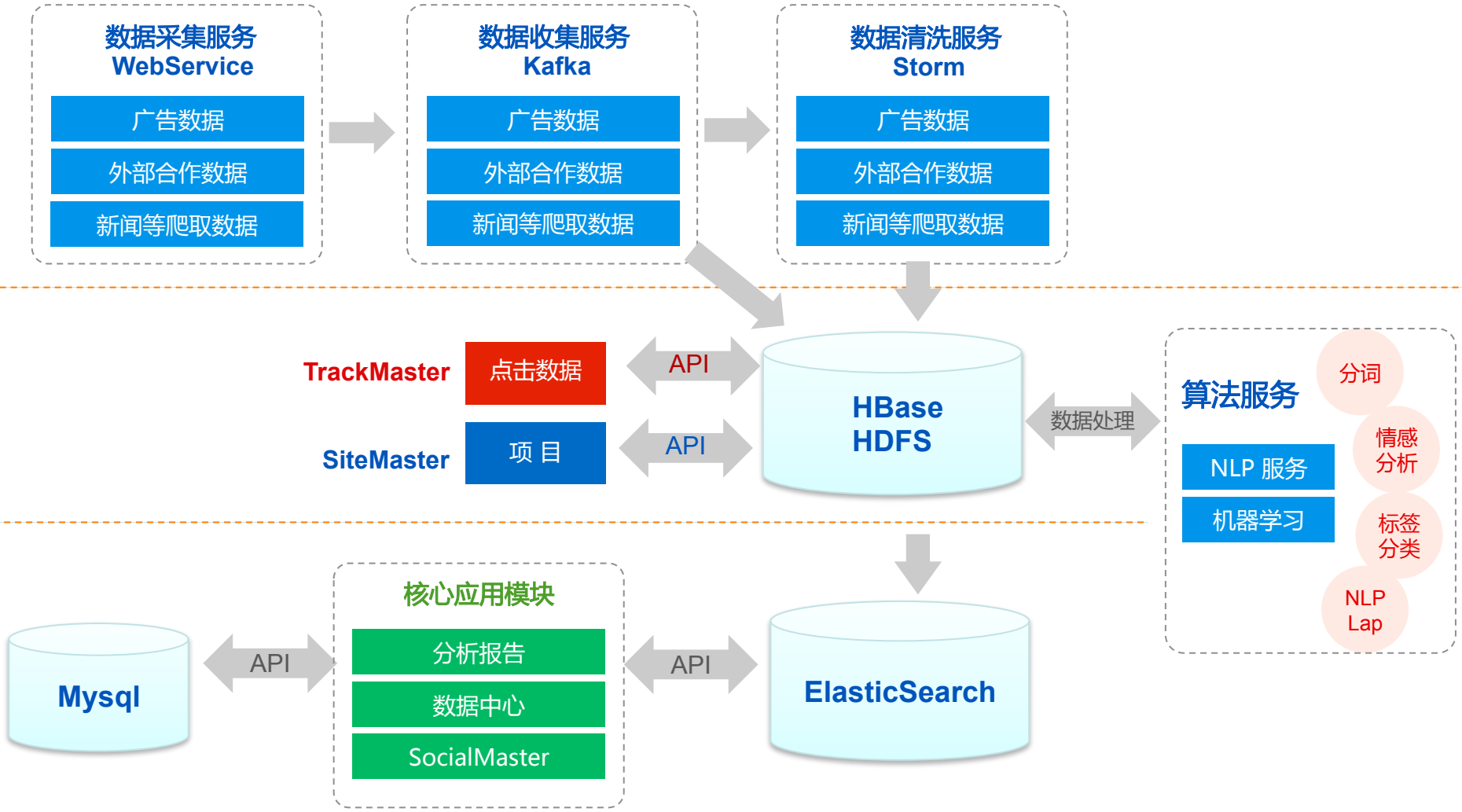
	单次	并发5个	并发10个
ElasticSearch	4002ms	4888ms	10194ms
Pinot	3911ms	failed	failed

- 1000亿

	单次	并发5个	并发10个
ElasticSearch	19005ms	21005ms	27736ms
Pinot	19019ms	failed	failed

- 每天请求数超过 100 亿
- 每天增长超过 5TB 级数据
- 每天对几千亿条记录进行上 1000 种维度的计算
- 客户有流式、实时、离线需求

AdMaster案例分析



- 用 ES 的 Alias 特性实现数据的水平扩展
- 用 Kibana 分析和展现数据（ELK）
- 多条件聚合查询，布尔查询
- 定制分词插件（IK），实现对特殊字符的精确匹配
- 用 Scroll 方式快速获取批量数据

Q & A

邮箱: johnlya@163.com

微信: johnlya