数据仓库与大数据工程

Data Warehouse and Big Data Engineering

第二部分 数据生成与采集

北京交通大学计算机与信息技术学院

内容提纲

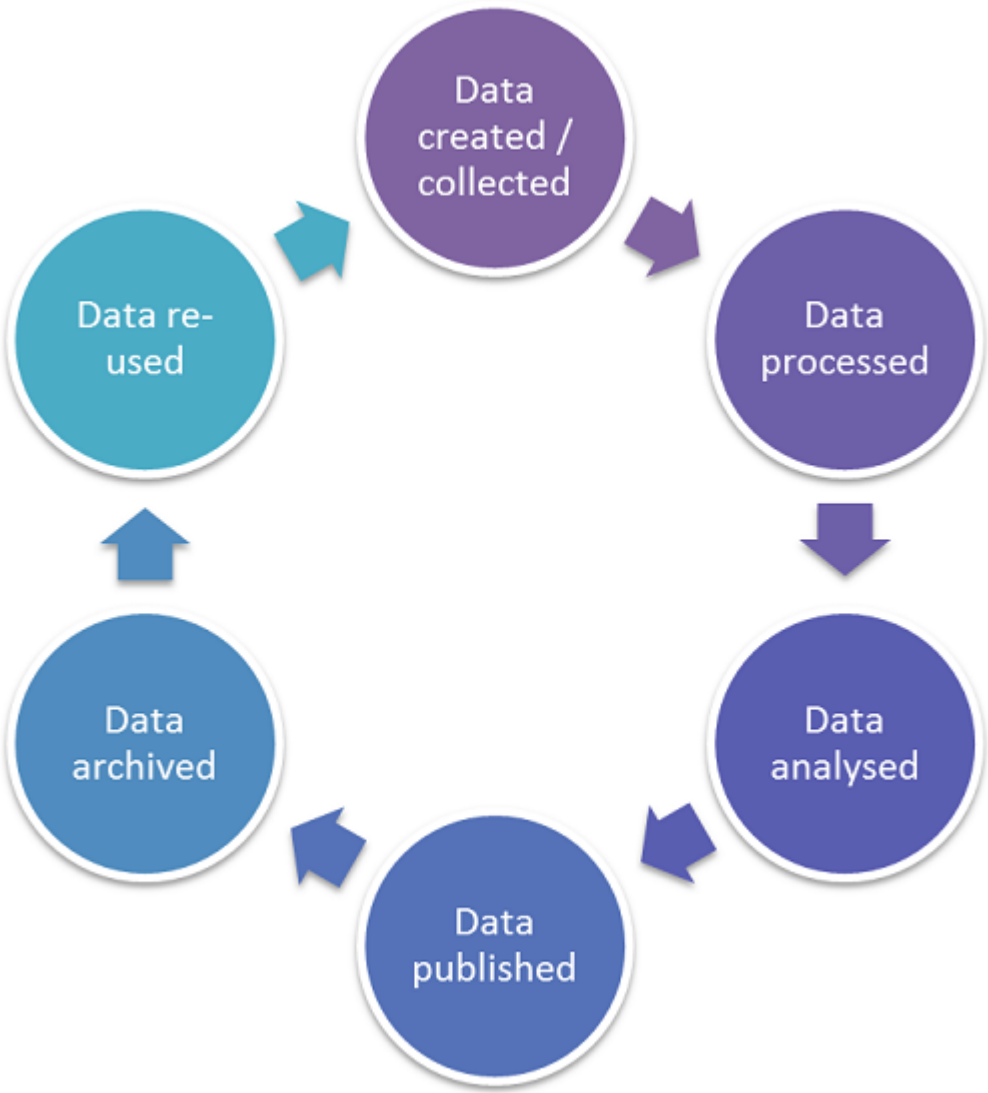
- **数据生命周期**
- 数据产生、类型与规模
- 数据生态圈及OLTP系统生态圈
- OLTP系统与DW/BD平台间数据流
- 面向OLTP系统的数据采集与传输方法
- 常见数据采集工具介绍
- 本部分实验要求

1. 数据生命周期的定义

- **数据生命周期**
  - **从数据的创建和保存开始，到数据最终过时并被删除的整个周期**
- Data life cycle refers to
  - the time from creation and the initial storage of data to the time that the data becomes obsolete and it is deleted.
  - the sequence of stages that a particular unit of data goes through from its initial generation or capture to its eventual archival and/or deletion at the end of its useful life.

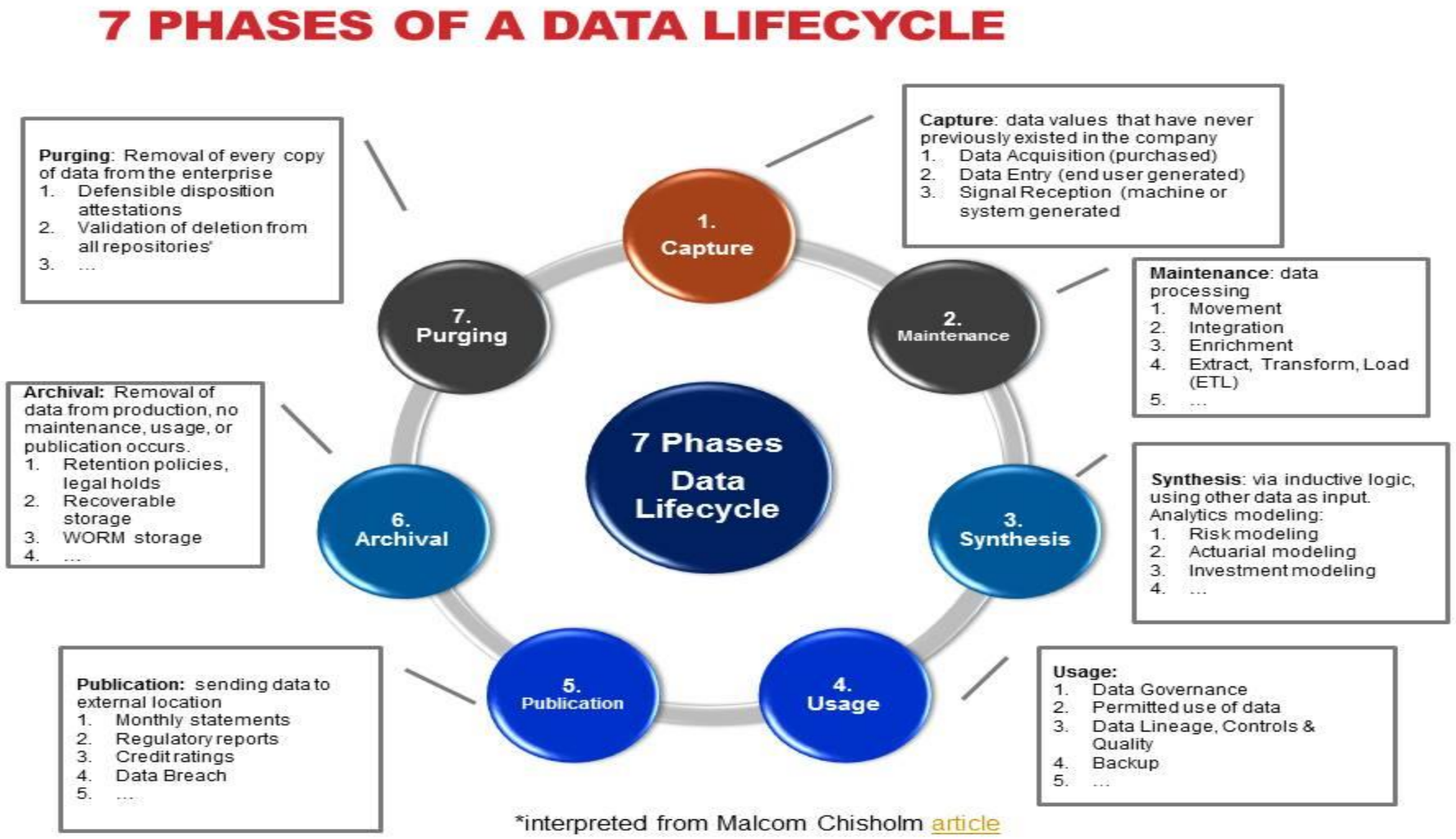
2. Phases of Data Life Cycle

其他观点1

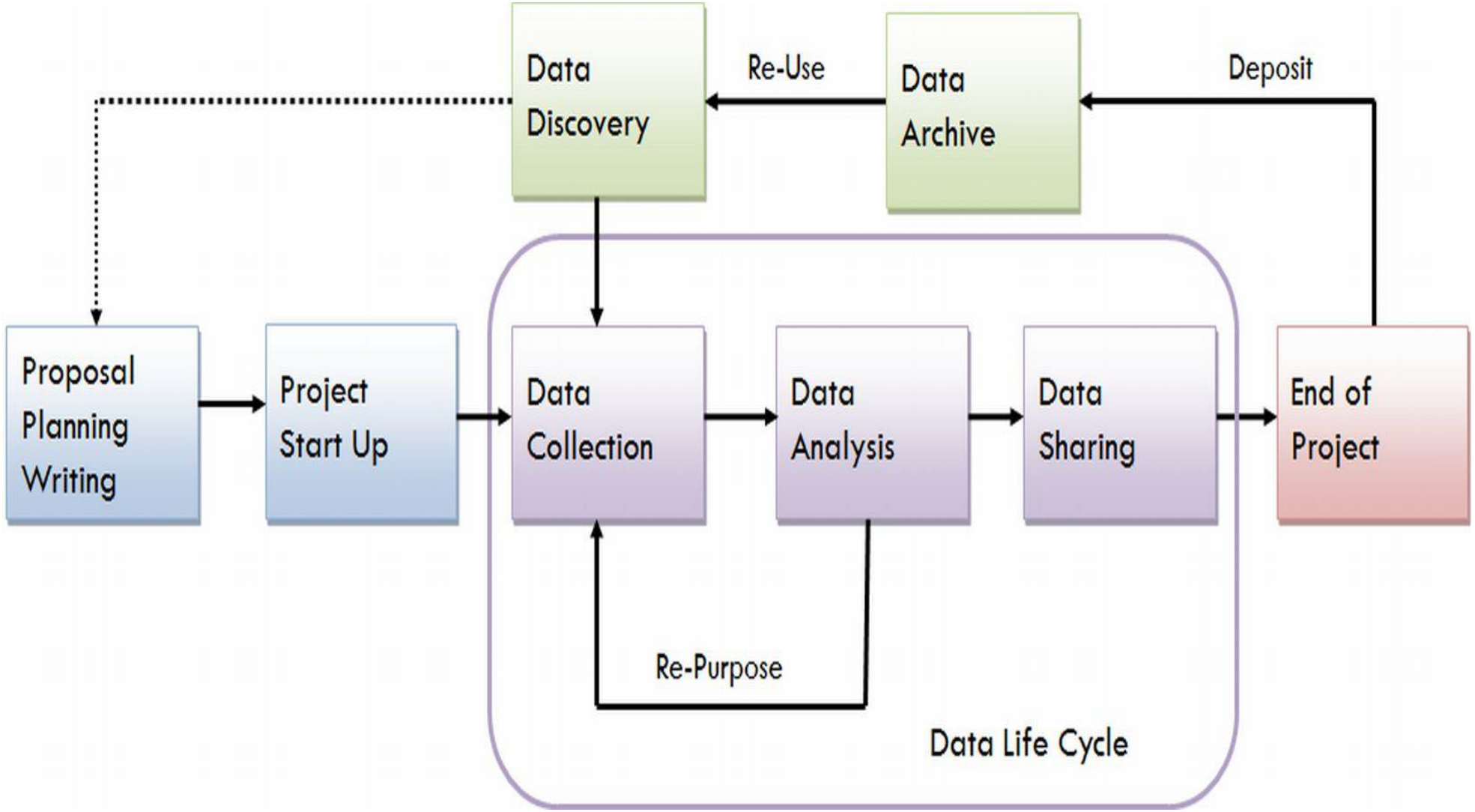

Data created / collected
Data processed
Data analysed
Data published
Data archived
Data re-used

缺少清除

其他观点2


7 PHASES OF A DATA LIFECYCLE
7 Phases Data Lifecycle
1. Capture
2. Maintenance
3. Synthesis
4. Usage
5. Publication
6. Archival
7. Purging
Capture: data values that have never previously existed in the company
1. Data Acquisition (purchased)
2. Data Entry (end user generated)
3. Signal Reception (machine or system generated
Maintenance: data processing
1. Movement
2. Integration
3. Enrichment
4. Extract, Transform, Load (ETL)
5. …
Synthesis: via inductive logic, using other data as input. Analytics modeling:
1. Risk modeling
2. Actuarial modeling
3. Investment modeling
4. …
Usage:
1. Data Governance
2. Permitted use of data
3. Data Lineage, Controls & Quality
4. Backup
5. …
Publication: sending data to external location
1. Monthly statements
2. Regulatory reports
3. Credit ratings
4. Data Breach
5. …
Archival: Removal of data from production, no maintenance, usage, or publication occurs.
1. Retention policies, legal holds
2. Recoverable storage
3. WORM storage
4. …
Purging: Removal of every copy of data from the enterprise
1. Defensible disposition attestations
2. Validation of deletion from all repositories'
3. …
*interpreted from Malcom Chisholm article

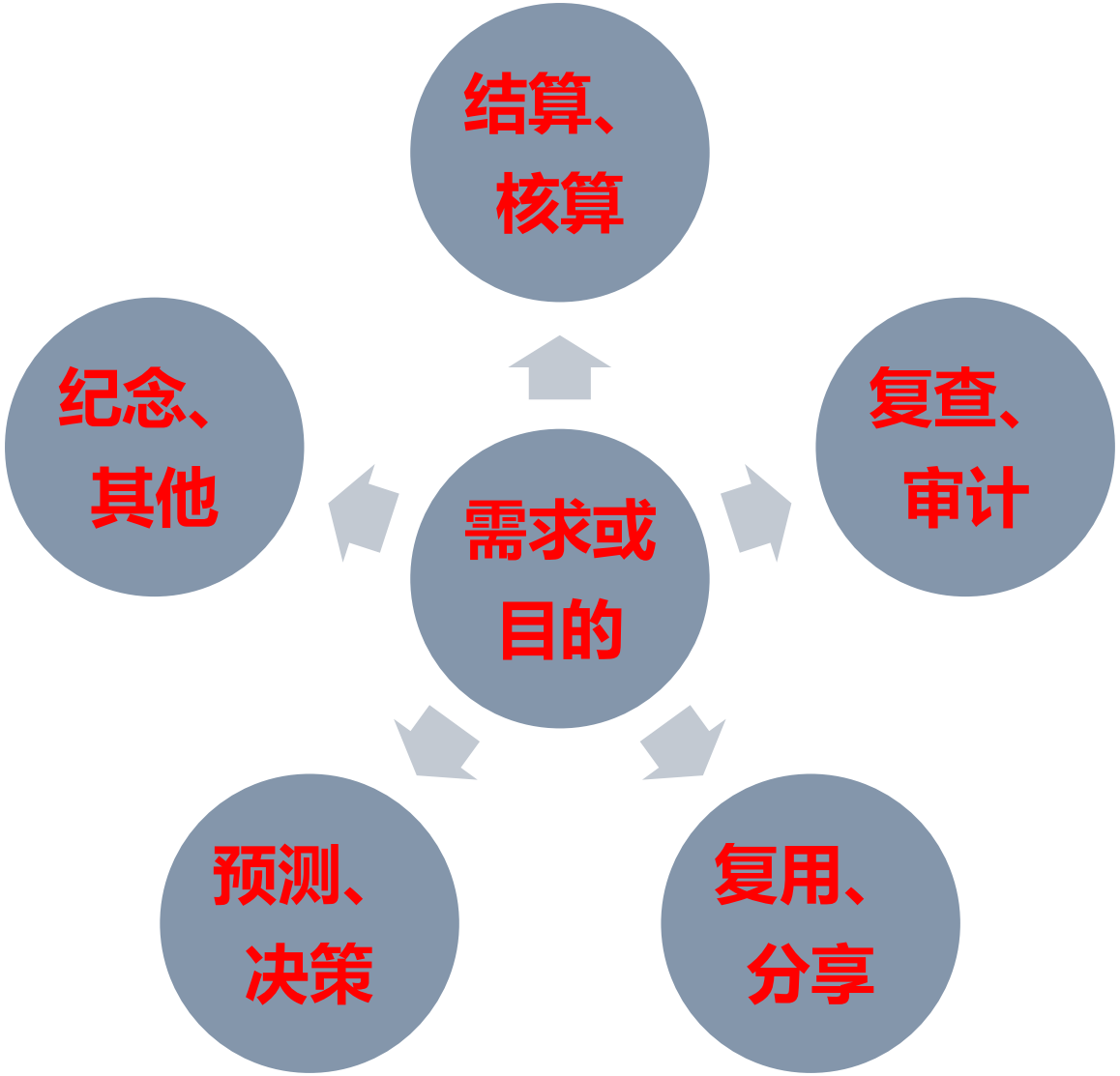
其他观点3：从工程过程的角度看周期



内容提纲

- 数据生命周期
- **数据产生、类型与规模**
- 数据生态圈及OLTP系统生态圈
- OLTP系统与DW/BD平台间数据流
- 面向OLTP系统的数据采集与传输方法
- 常见数据采集工具介绍
- 本部分实验要求

1. 人类记录数据的需求



2. 记录内容范围

案例1—银行各种记录系统

- 银行记录系统记录内容
  - 窗口业务
    - 存、取、贷款、认购、认证、销户业务
  - ATM操作
    - 查询、转账、存款等业务
  - 网银APP操作
    - 查询、转账、贷款、购买、付款、…
  - 业务场景视频
  - 客服
    - 用户操作、语音、反馈
  - …

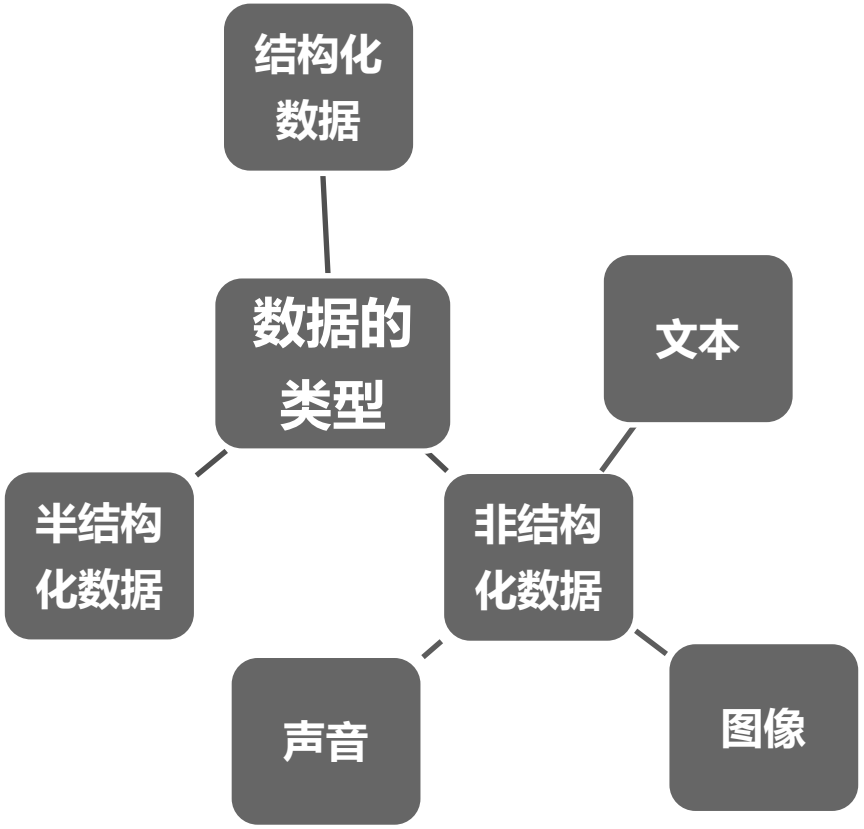
案例2—民航各种记录系统

- 航班执行及飞机飞行记录内容
  - 驾驶员行为
  - 气象
  - 飞行状态、飞行轨迹
  - 起飞、关舱门、撤轮挡、降落
  - …
- 与旅客有关的记录内容
  - 查票、购票、退票、改签
  - 客服
  - 安检、值机、登机
  - …

案例3—SNS

- 社交媒体
  - 视频
  - 短文本
  - 消息
  - 行为记录
  - 操作记录
  - 用户账户及用户间关系
  - …

3. 数据的类型与规模



类型复杂，规模巨大

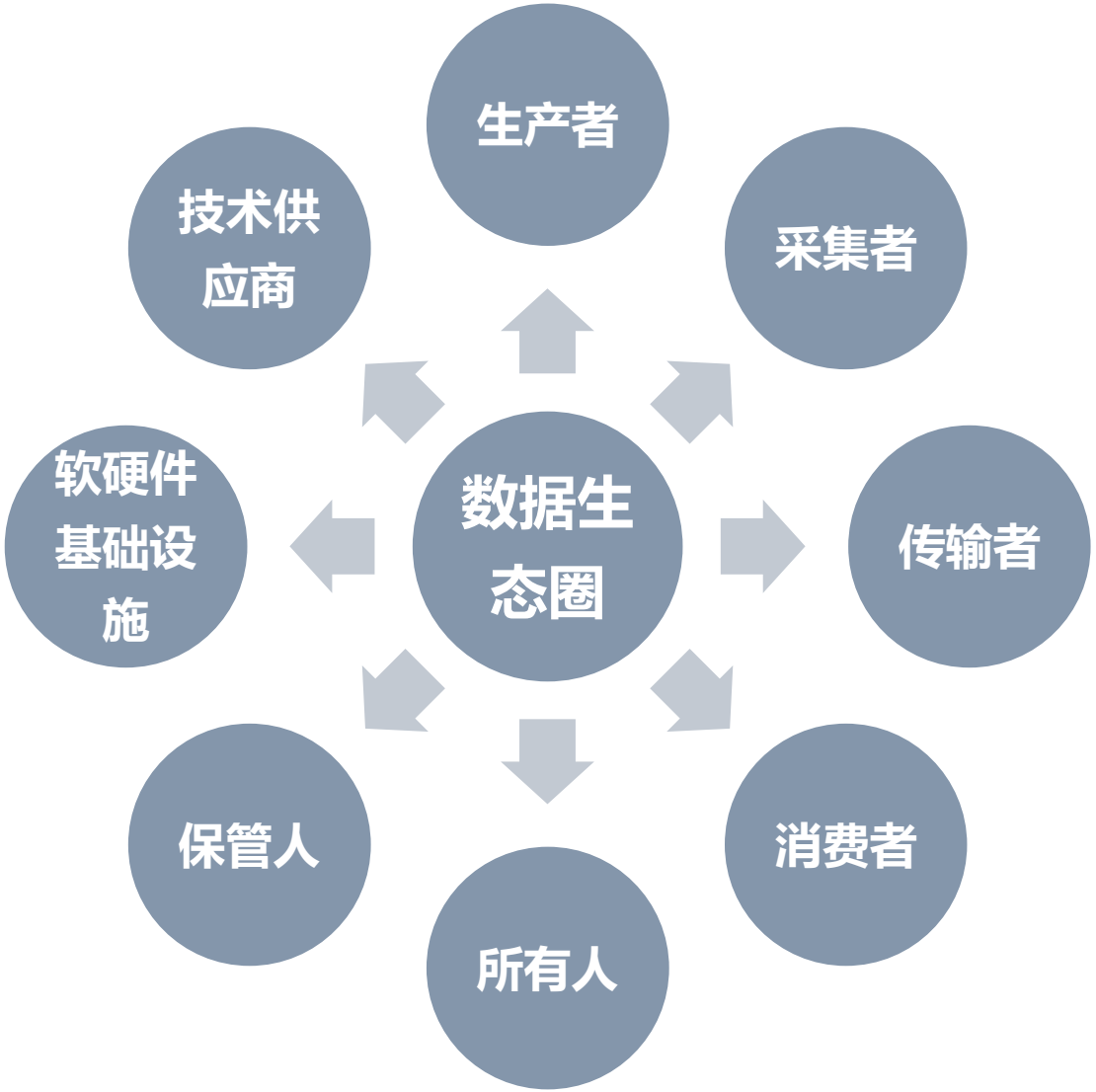
数据规模巨大的主要原因:

1. **单条数据量大**
2. **来源多—用户多**
3. **业务繁忙—单个终端单位时间生成的数据条数**

内容提纲

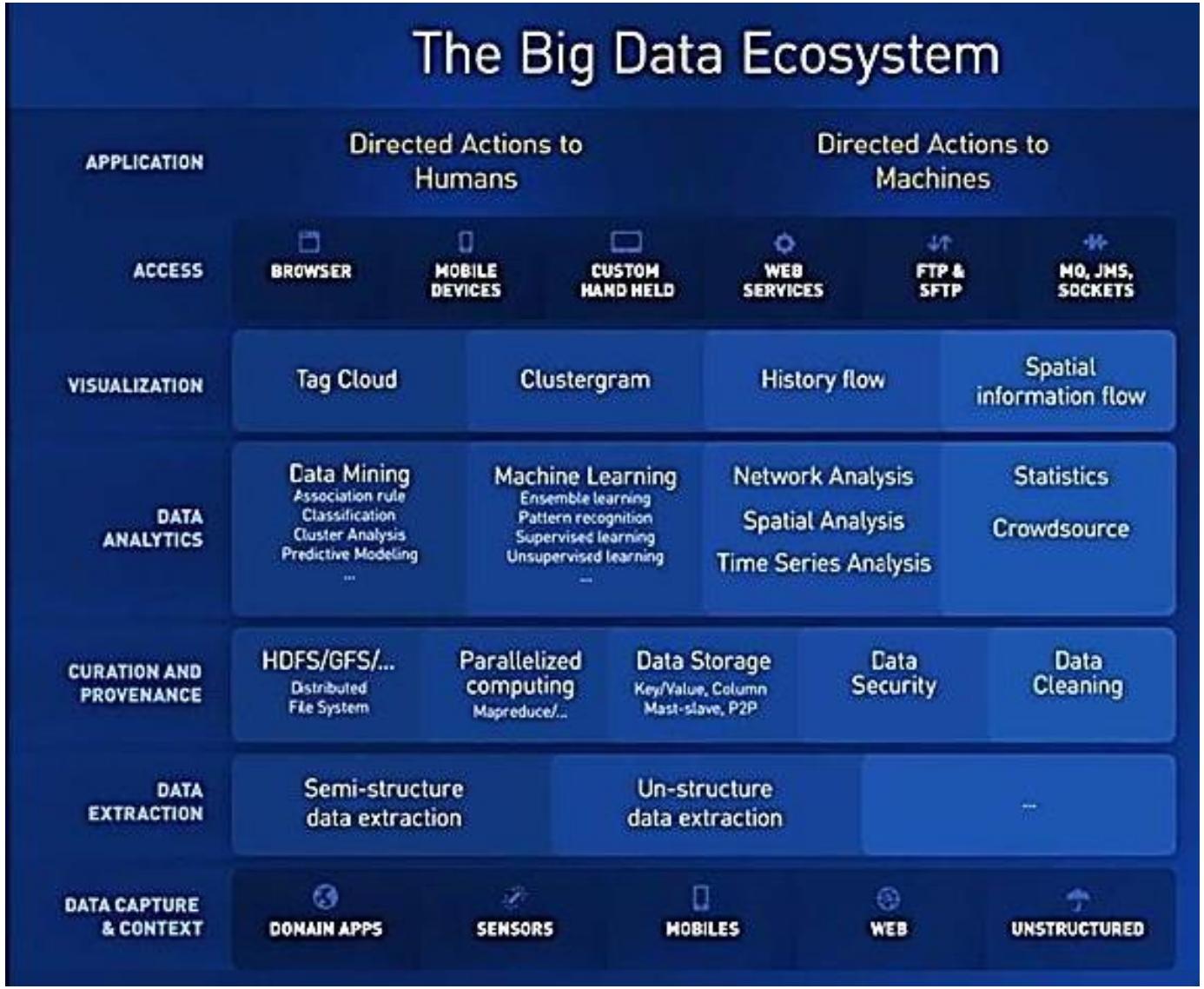
- 数据生命周期
- 数据产生、类型与规模
- **数据生态圈及OLTP系统生态圈**
- OLTP系统与DW/BD平台间数据流
- 面向OLTP系统的数据采集与传输方法
- 常见数据采集工具介绍
- 本部分实验要求

1. 数据生态圈—The Data Ecosystem

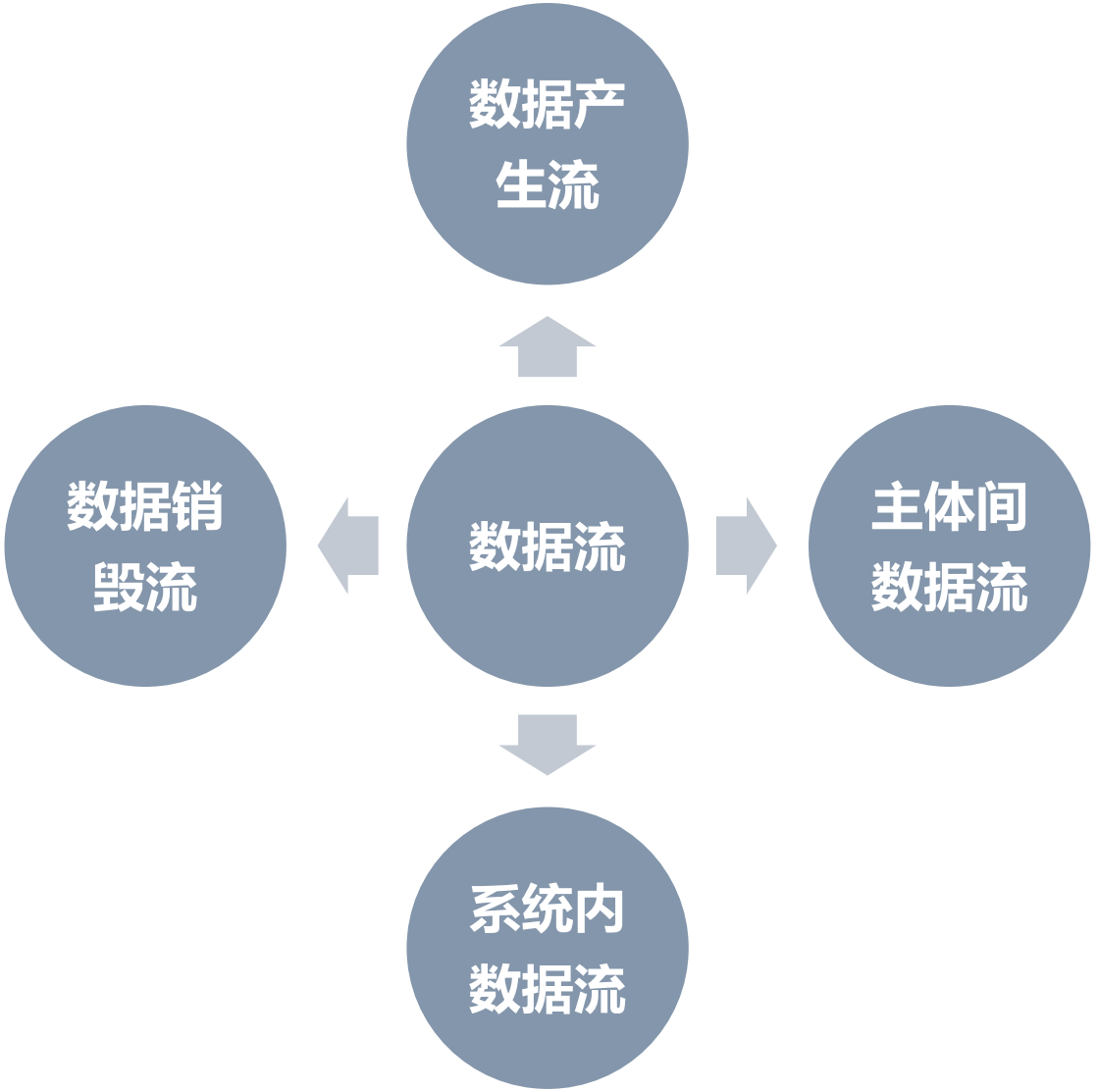


数据生态圈中的各类主体

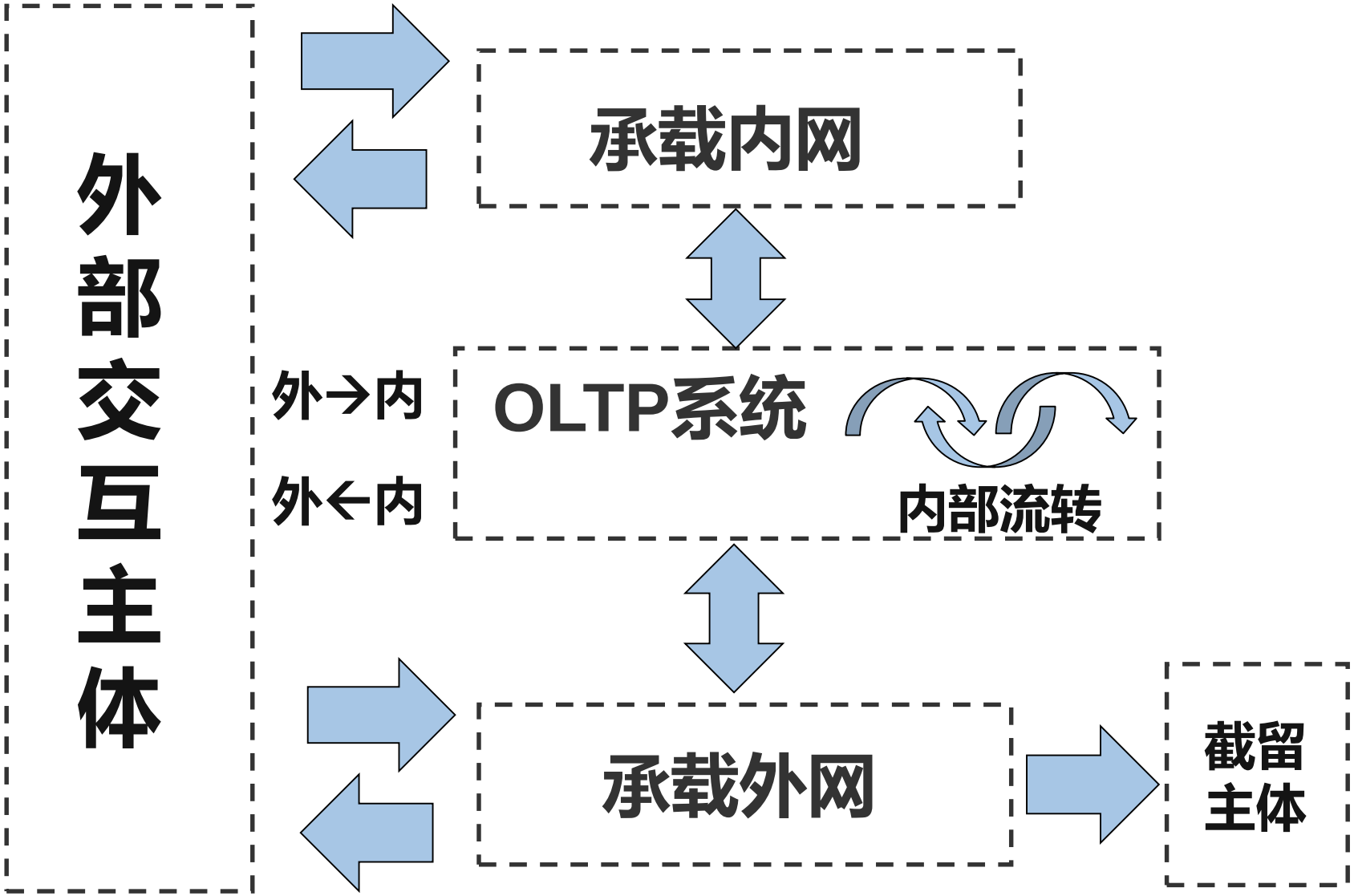
大数据生态圈一种提法


The Big Data Ecosystem
APPLICATION
Directed Actions to Humans
Directed Actions to Machines
ACCESS
BROWSER
MOBILE DEVICES
CUSTOM HAND HELD
WEB SERVICES
FTP & SFTP
MQ, JMS, SOCKETS
VISUALIZATION
Tag Cloud
Clustergram
History flow
Spatial information flow
DATA ANALYTICS
Data Mining
Association rule
Classification
Cluster Analysis
Predictive Modeling
...
Machine Learning
Ensemble learning
Pattern recognition
Supervised learning
Unsupervised learning
...
Network Analysis
Spatial Analysis
Time Series Analysis
Statistics
Crowdsource
CURATION AND PROVENANCE
HDFS/GFS/...
Distributed File System
Parallelized computing
Mapreduce/...
Data Storage
Key/Value, Column
Mast-slave, P2P
Data Security
Data Cleaning
DATA EXTRACTION
Semi-structure data extraction
Un-structure data extraction
...
DATA CAPTURE & CONTEXT
DOMAIN APPS
SENSORS
MOBILES
WEB
UNSTRUCTURED

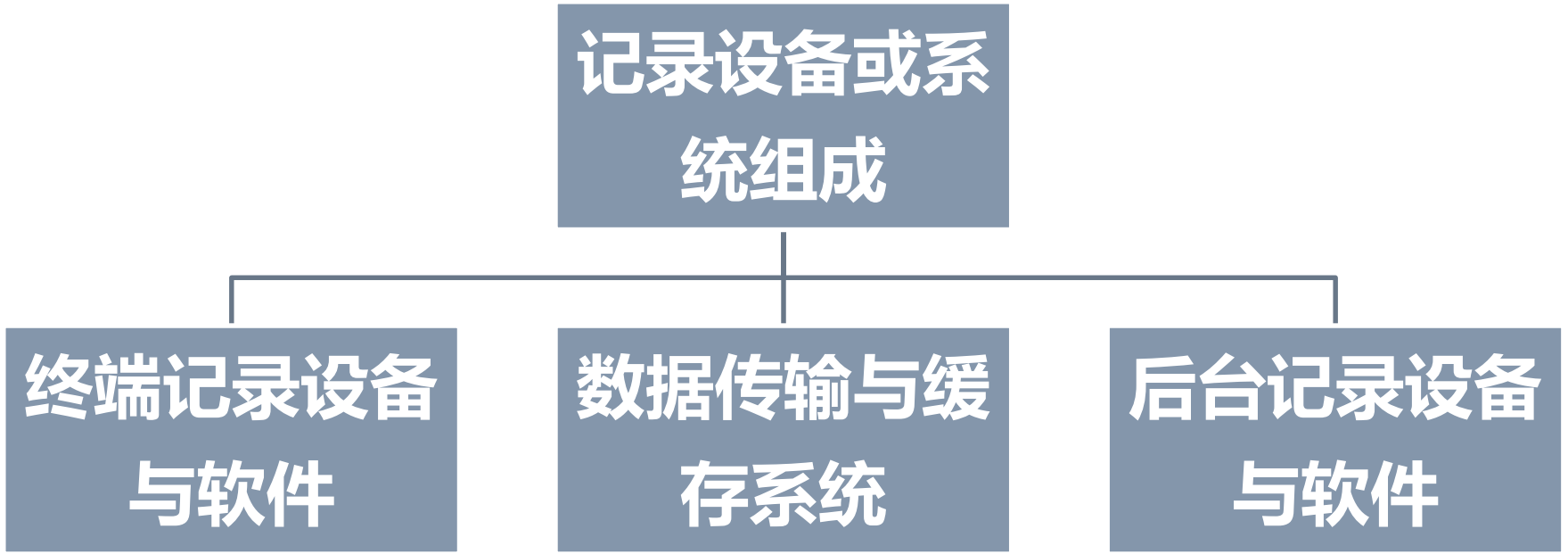
2. 数据生态圈中的数据流



3. OLTP系统数据生态圈与数据流



4. 数据产生流：记录工具—记录设备或系统



终端 **网络** **服务集群**

5. 从外部到OLTP系统内部

- **外部到内部**的数据流
  - 数据由**用户**通过**终端**进入系统内部
  - 环境状态数据通过**感知采集设备**进入系统内部
  - 其他系统的数据经主动**采集进入系统**
    - 企业外系统
    - 企业内系统

6. 从系统内部到系统外部

- **返回结果**给用户端
  - 用户操作结果、推送结果
- 提供数据**给下游系统**
- 数据**公开**
  - 数据依法披露、社会公益需求
- 提供数据**给外部系统**
  - 企业内数据仓库与大数据平台
  - 企业内部其他系统
  - 上级单位系统、下级单位系统、协作单位系统

7. 内部跨层流动

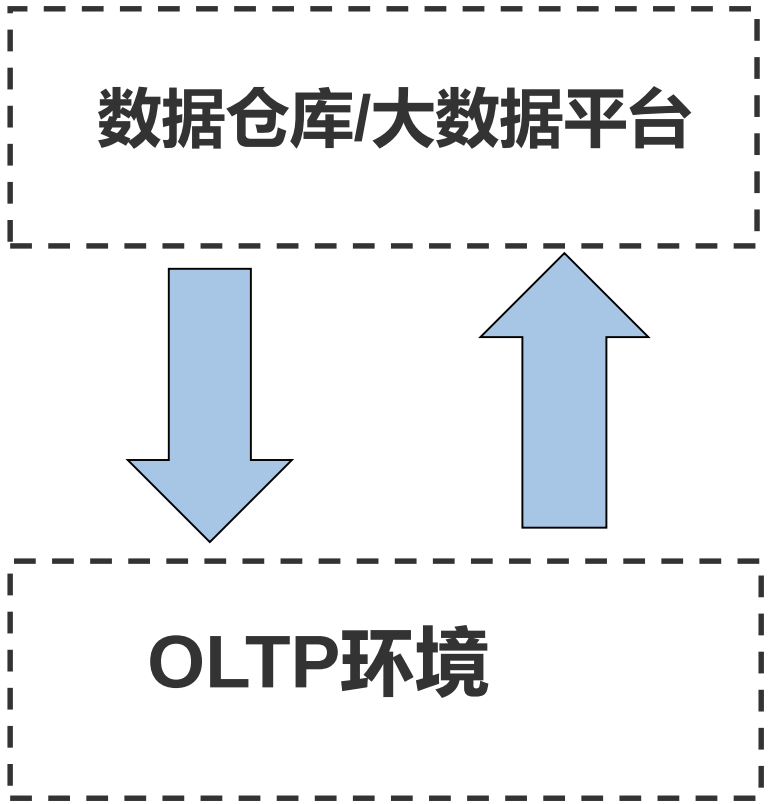
- **服务端与终端**之间的流动
  - 终端到服务端、服务端到终端
  - 终端到缓冲端、服务端到缓冲端
- **服务端内部**不同层级流程
  - 冷备、热备
  - 同步
  - 汇总统计
  - 归档
  - 对外出口缓冲

内容提纲

- 数据生命周期
- 数据产生、类型与规模
- 数据生态圈及OLTP系统生态圈
- **OLTP系统与DW/BD平台间数据流**
- 面向OLTP系统的数据采集与传输方法
- 常见数据采集工具介绍
- 本部分实验要求

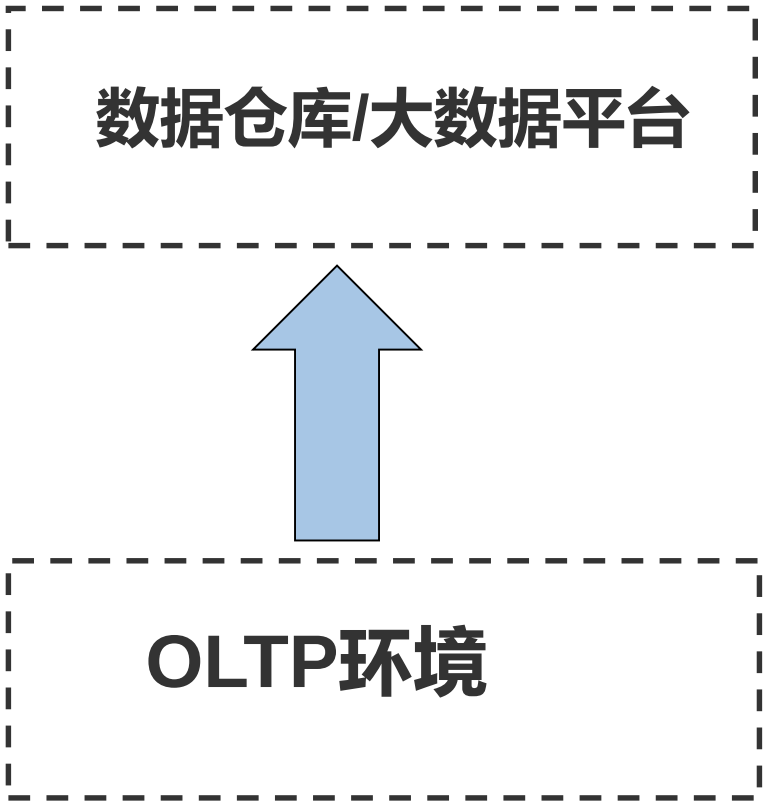
1. 企业内大平台间数据流

流动**内容**、功能**目标**
流动**时机**、软硬件**设施**



2. OLTP环境→数据仓库/大数据平台

流动内容、功能目标、流动时机、软硬件设施



流动内容：

基础事实

功能目标：

抽取、转换、载入，服务于后续数据利用

流动时机：

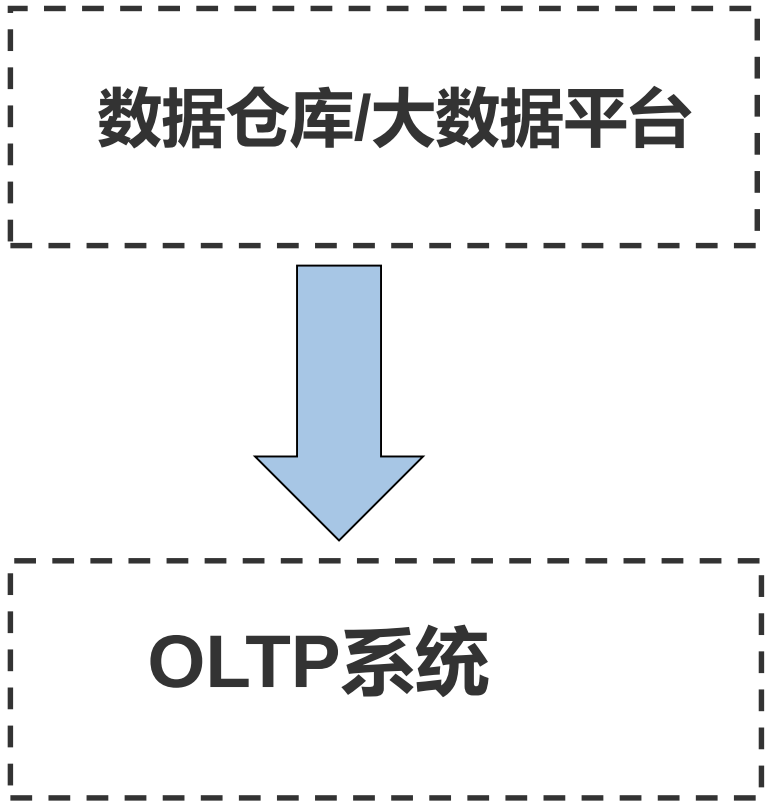
实时、近实时、小时级、天级、离线

软硬件设施：

软件程序、抽取服务器

3. 数据仓库/大数据平台→OLTP环境

流动内容、功能目标、流动时机、软硬件设施



流动内容：

分析挖掘结果：信息流、知识流

功能目标：

反馈、回写信息，服务于OLTP业务提升

流动时机：

实时、近实时、小时级、天级、事件触发

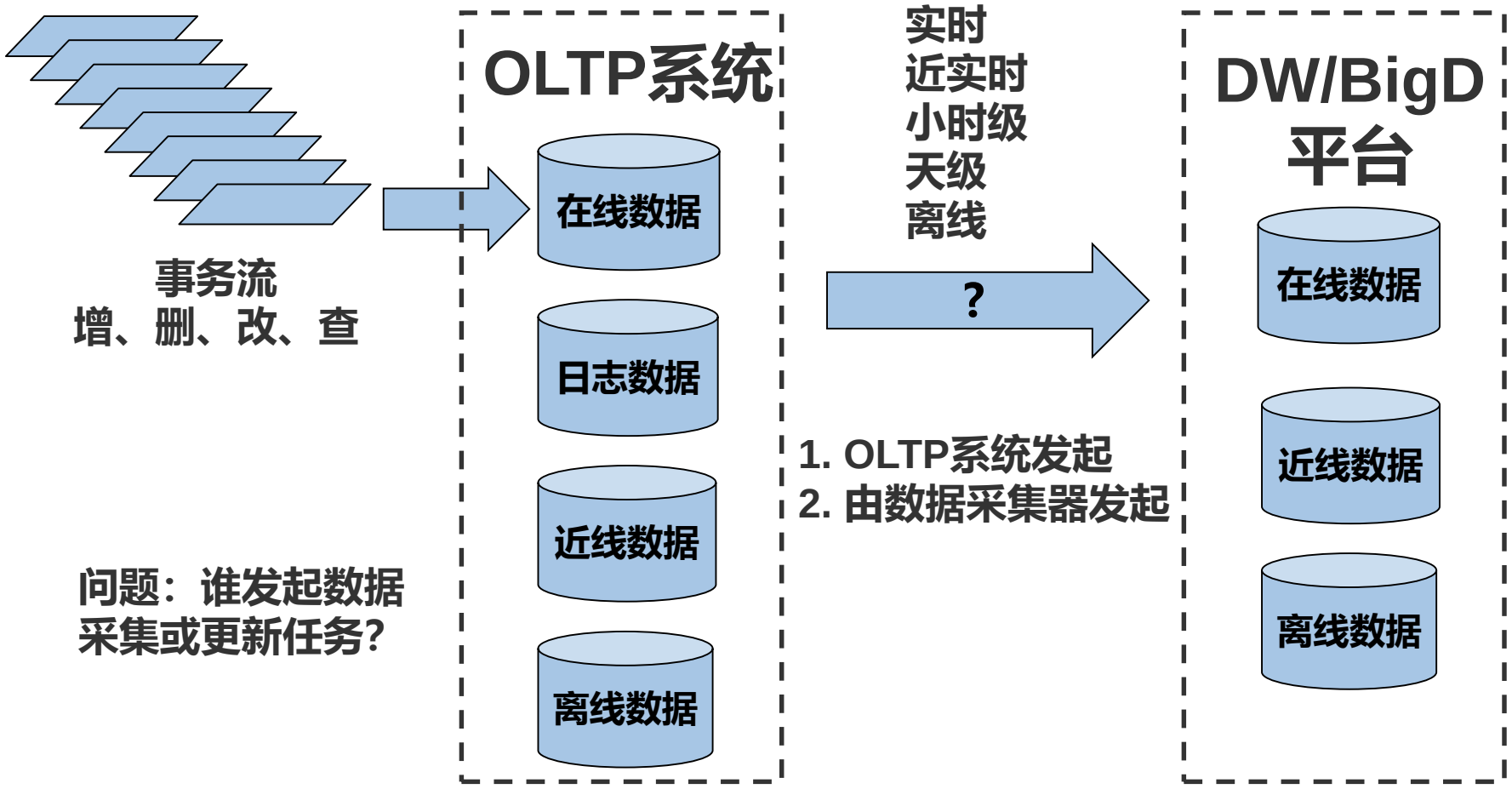
软硬件设施：

回写服务器及软件

内容提纲

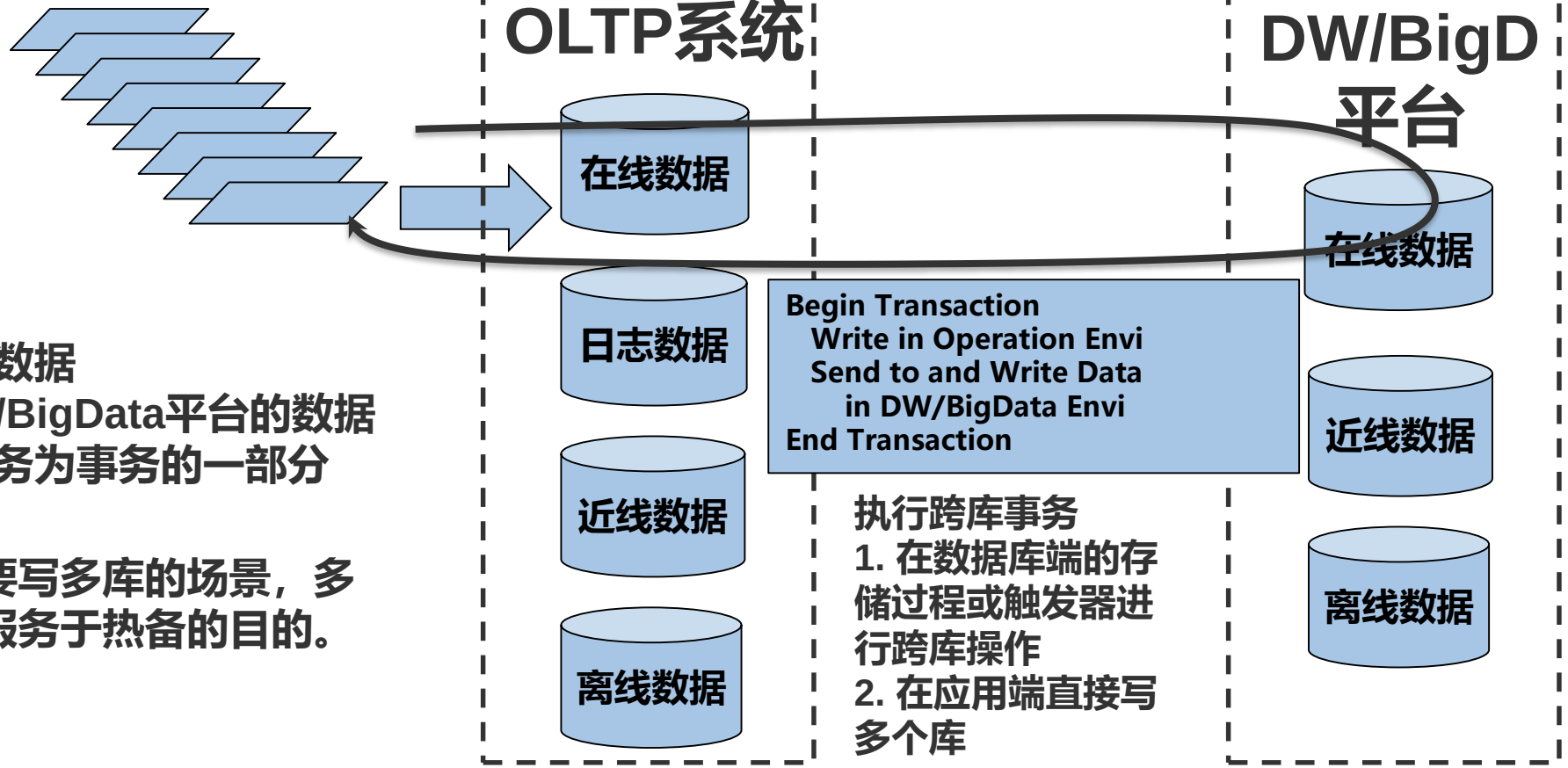
- 数据生命周期
- 数据产生、类型与规模
- 数据生态圈及OLTP系统生态圈
- OLTP系统与DW/BD平台间数据流
- **面向OLTP系统的数据采集与传输方法**
- 常见数据采集工具介绍
- 本部分实验要求

1. ETL时效性及任务发起主体



2. OLTP系统作为发起主体以达到实时性

事务流：增、删、改、查

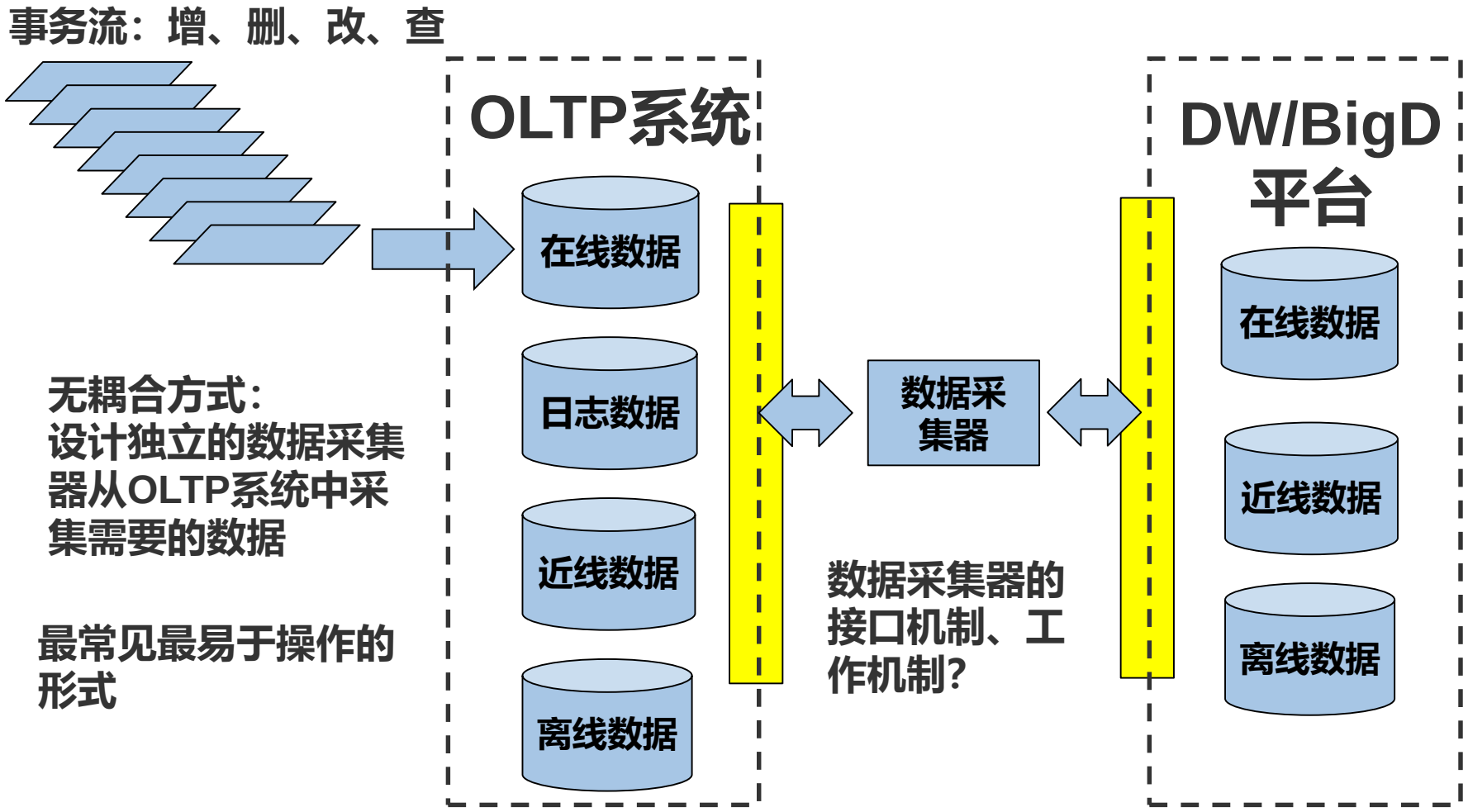


实时更新数据
在DW/BigData平台的数据更新的任务为事务的一部分

适合于需要写多库的场景，多数情况下服务于热备的目的。

存在的问题：1. 两类性质完全不同的平台紧密耦合，拖累OLTP系统，需要具有相当强大的硬件系统和软件系统作为保障和支撑，成本高。2. 不适用于大批量数据。

3. 由数据采集器发起

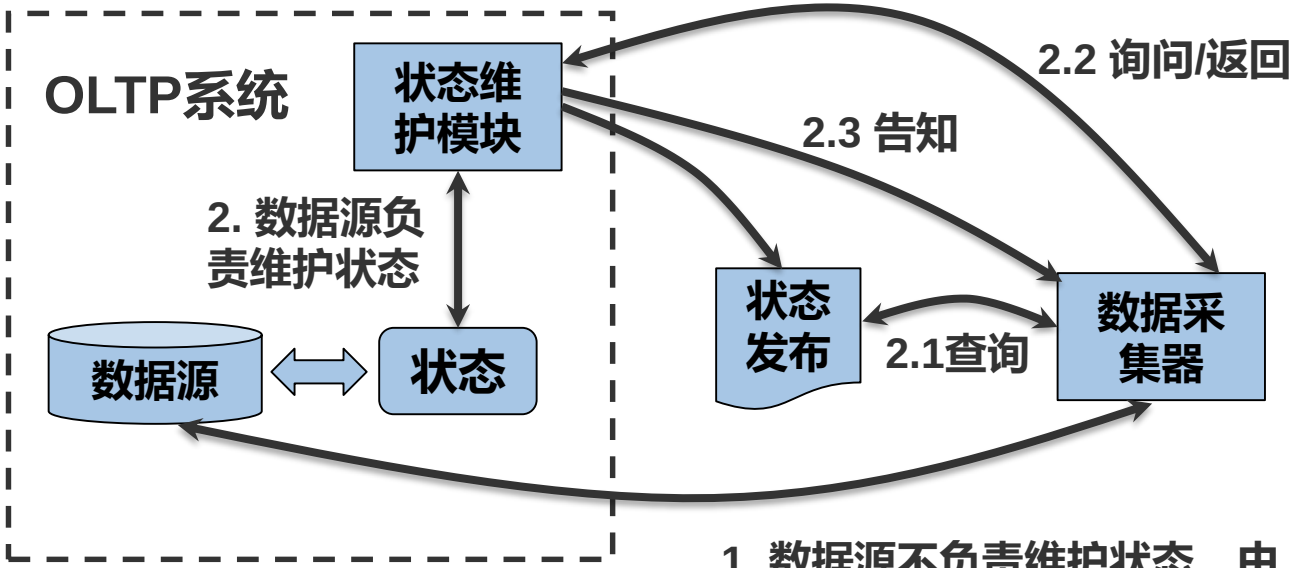


存在的问题：无法实现真正的实时。

4. 实时性需求与解耦合需求间的矛盾

- 数据采集的实时性需求
  - 实时性是必然的需求，实时性要求流程中各个环节衔接紧密，**延时尽可能的短**，前后环节间**合作紧密**
- 系统间的解耦合需求
  - 确保OLTP系统、数据采集模块、DW/BigData平台之间的相互独立性，提高系统的**性能稳定性与安全性，降低系统建设成本**
- 重要任务
  - 如何能在**解耦合的前提下**，尽可能地**提高实时性**

5. 数据源变化状态维护与感知策略



一般由数据源负责维护数据是否发生变化。

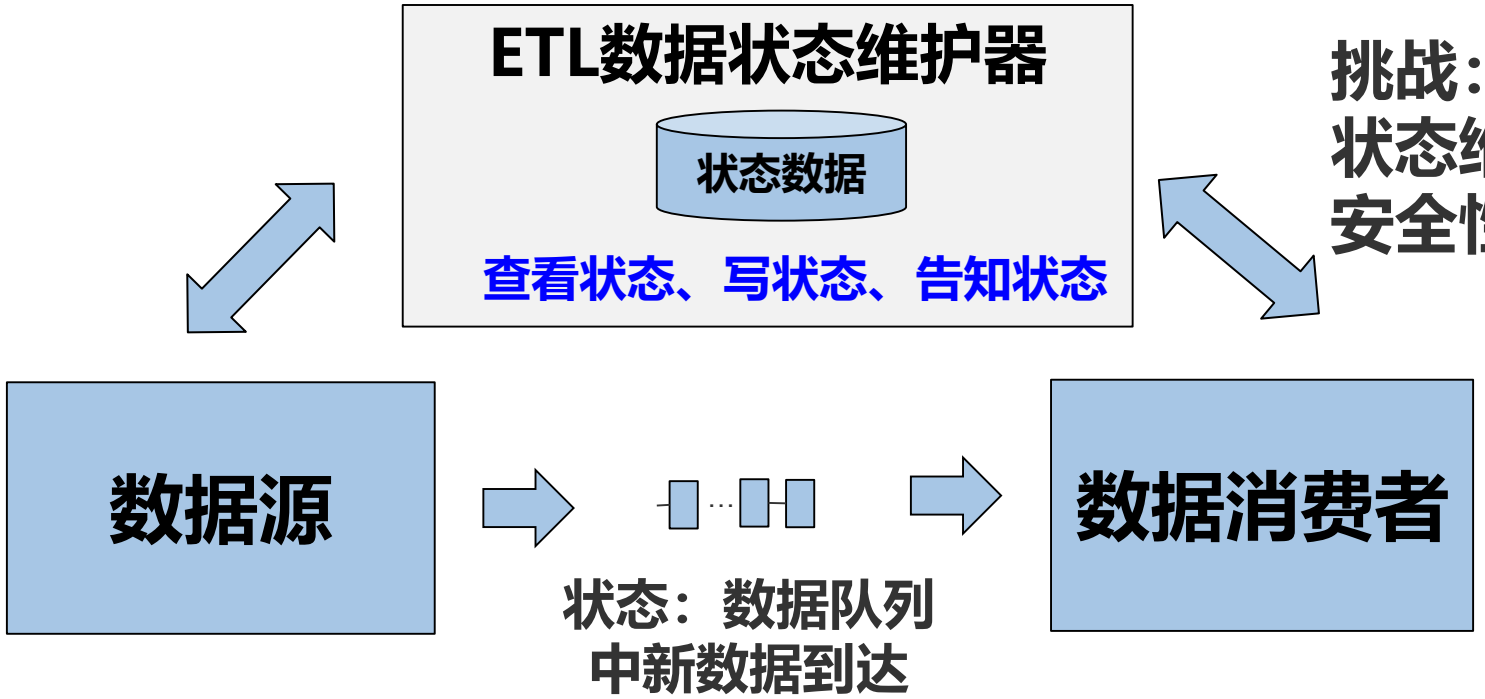
2.1 发布状态，有授权者可查
2.2 不发布状态，有授权者可查，可能告诉你，也可能不告诉你
2.3 一旦有数据变化主动告诉你

1. 数据源不负责维护状态，由采集者直接感知变化状态。
问题：权限问题、成本问题、安全性问题

问题：2.1、2.2、2.3三种模式，数据采集器的地位有什么差异？哪种模式下地位最高？状态发布有什么平台？

状态变化感知机制

挑战：
数据源非常多，
状态变化频繁，
状态规模非常大，
状态需求方很多



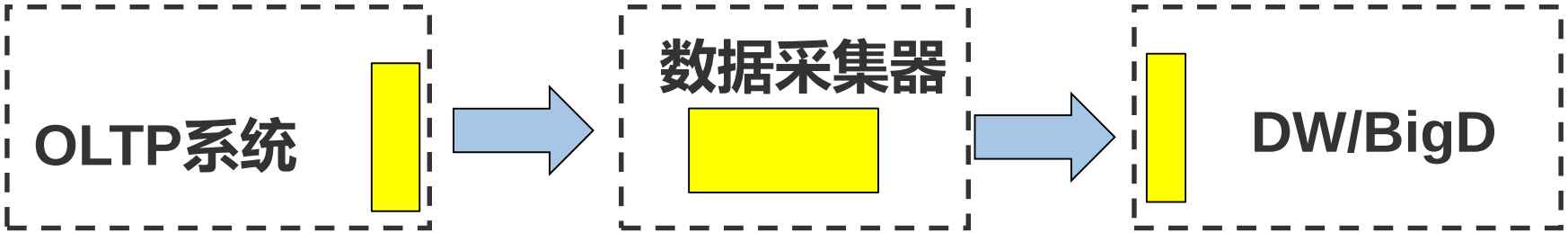
挑战：
状态维护器本身效率
安全性保证机制？

消费者如何知道状态发生变化？

自己现写程序干这事，行不行，存在什么问题？
有没有现成成熟的解决方案？

6. 时效性与数据缓冲问题

- 数据缓冲的必然性
  - 数据生成速度与数据采集速度之间不匹配
  - 数据完整性要求或集成需求



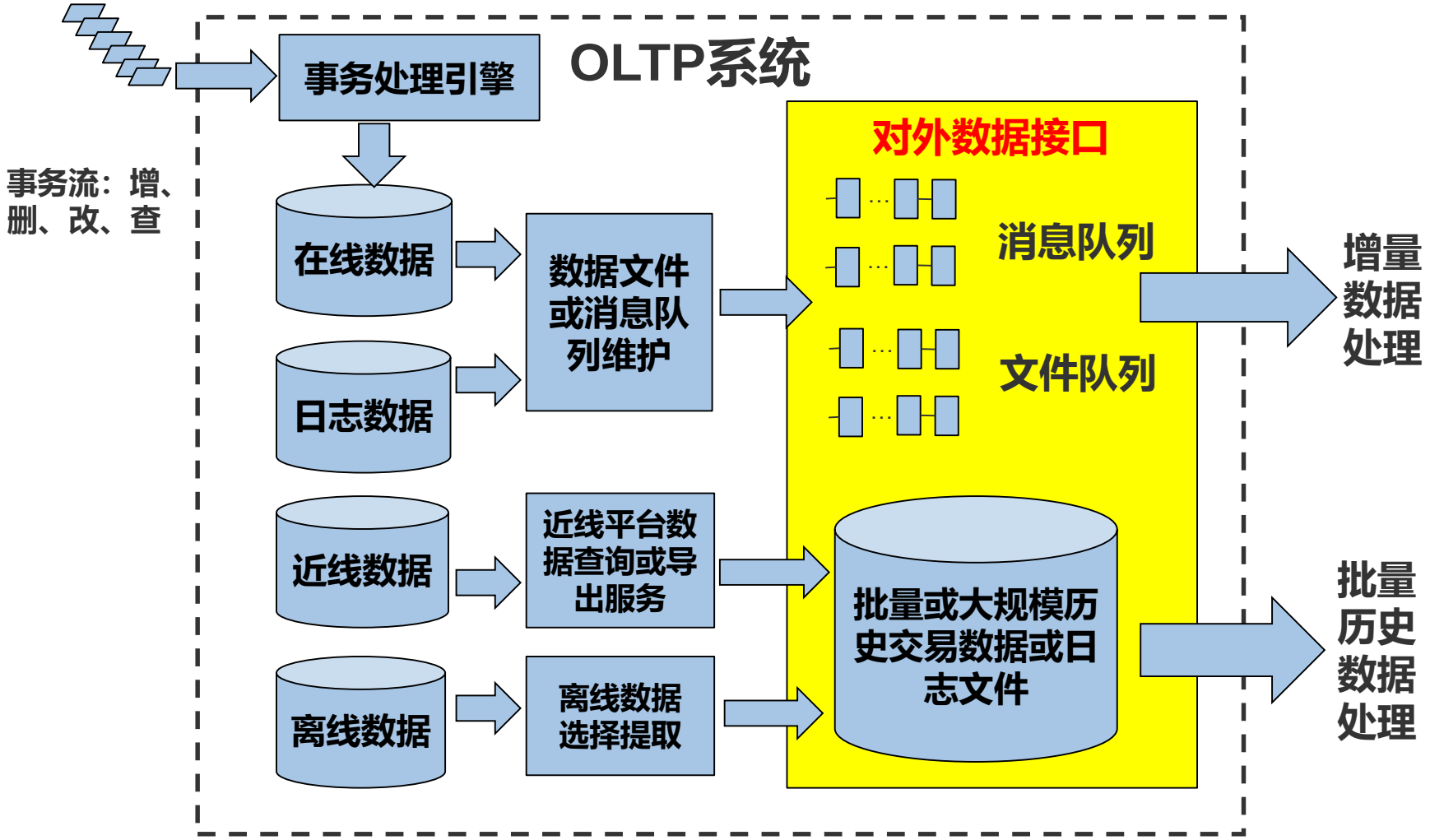
一般在三层都存在不同类型的数据缓冲，常见形式包括：
文件队列、消息队列、变化表、历史数据、离线数据文件、日志文件

时效性要求越高，需要的缓冲空间就越小，但是，要求的缓冲机构比较复杂，中间及后续处理能力越高。时效性越低，要求的缓冲空间就越大，数据传送机制相对比较简单，达到天级别时，甚至需要OLTP系统的暂停业务服务。

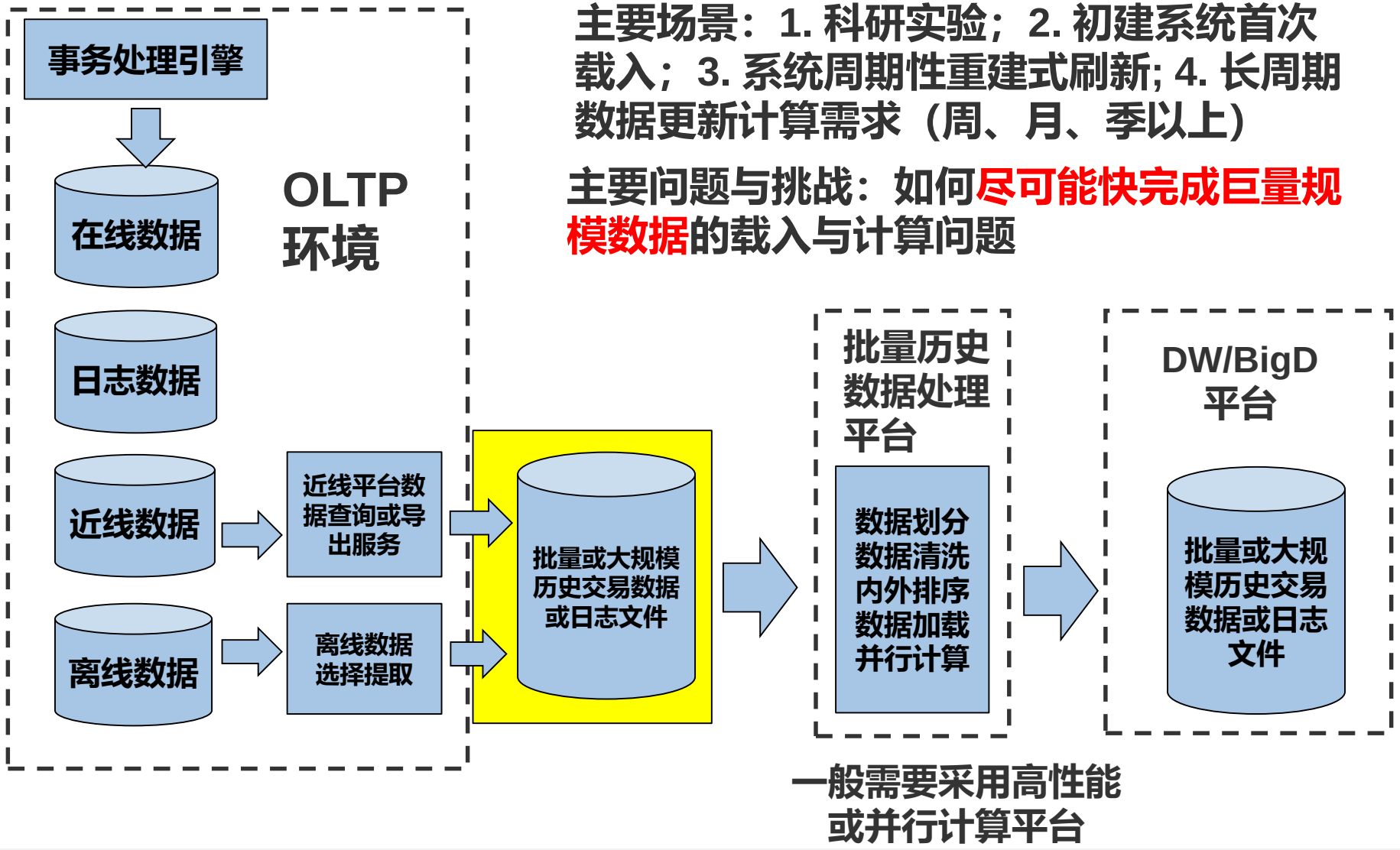
注意：

各层**缓冲还可能溢出**，溢出了如何处理，也是一个重要的问题

7. OLTP系统对外提供数据的接口问题



8. 批量历史数据处理

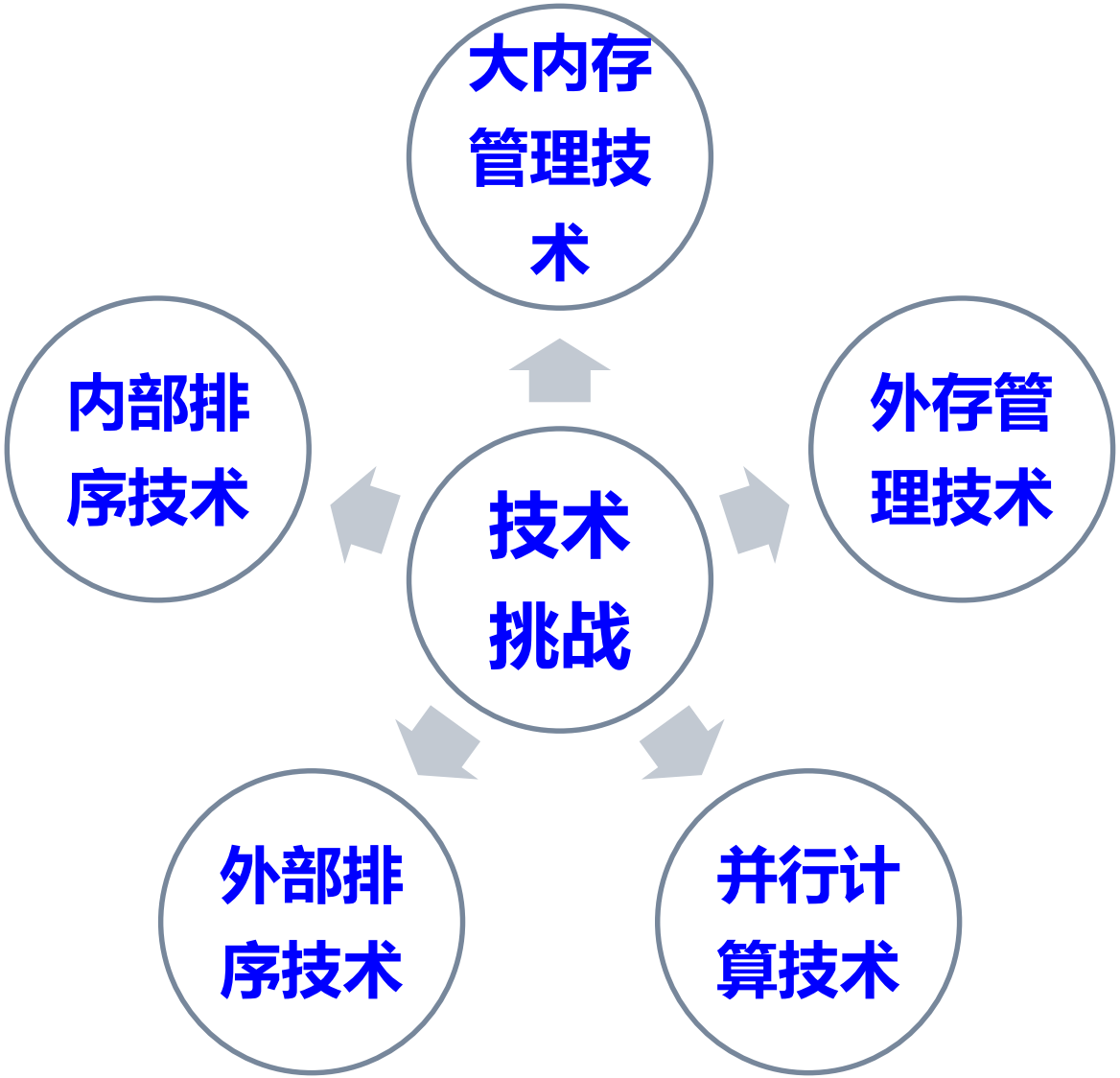


主要场景：1. 科研实验；2. 初建系统首次载入；3. 系统周期性重建式刷新; 4. 长周期数据更新计算需求（周、月、季以上）

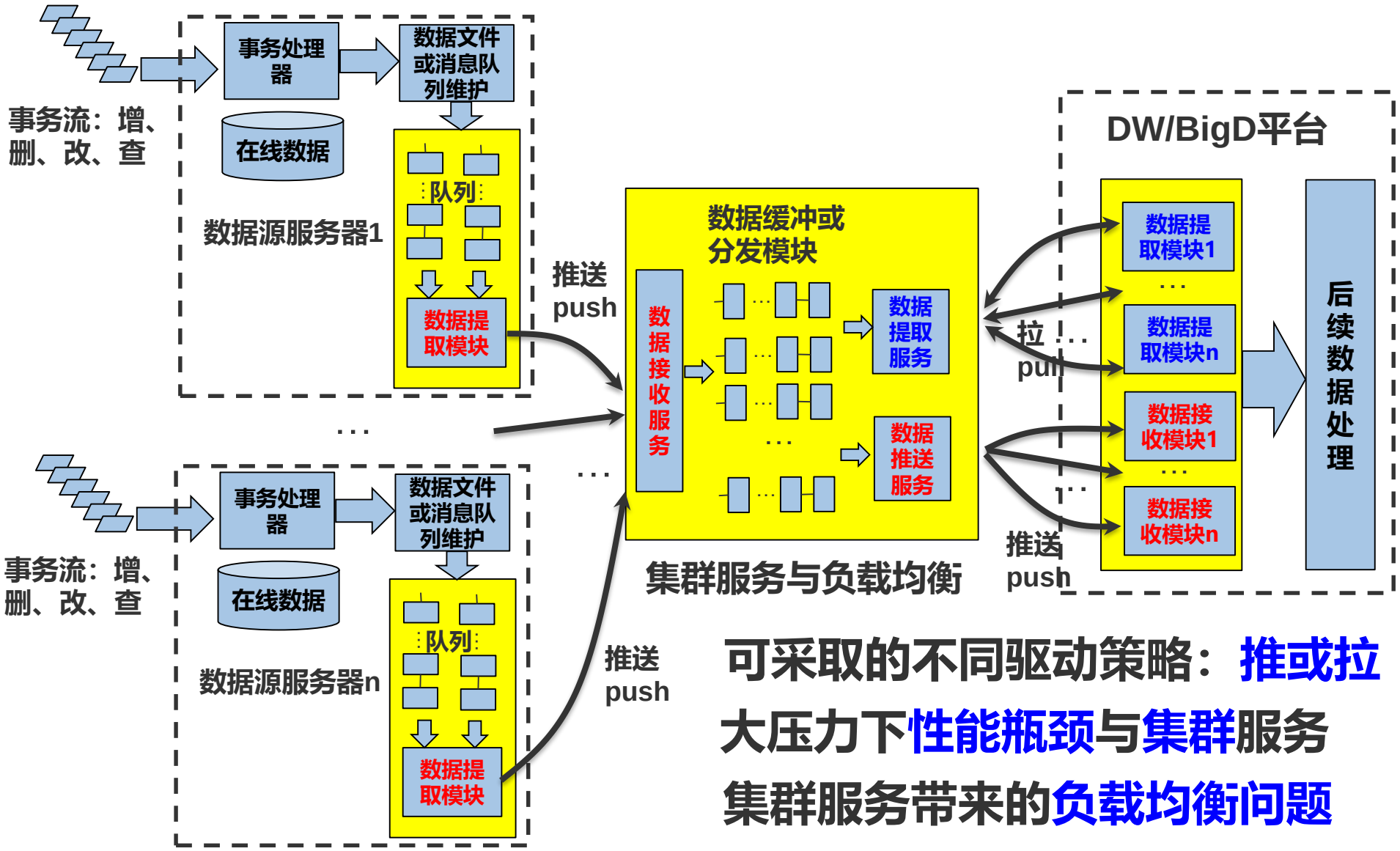
主要问题与挑战：如何**尽可能快完成巨量规模数据**的载入与计算问题

一般需要采用高性能或并行计算平台

批量数据处理涉及到的主要技术挑战



9. 新增数据采集机制



可采取的不同驱动策略：**推或拉**

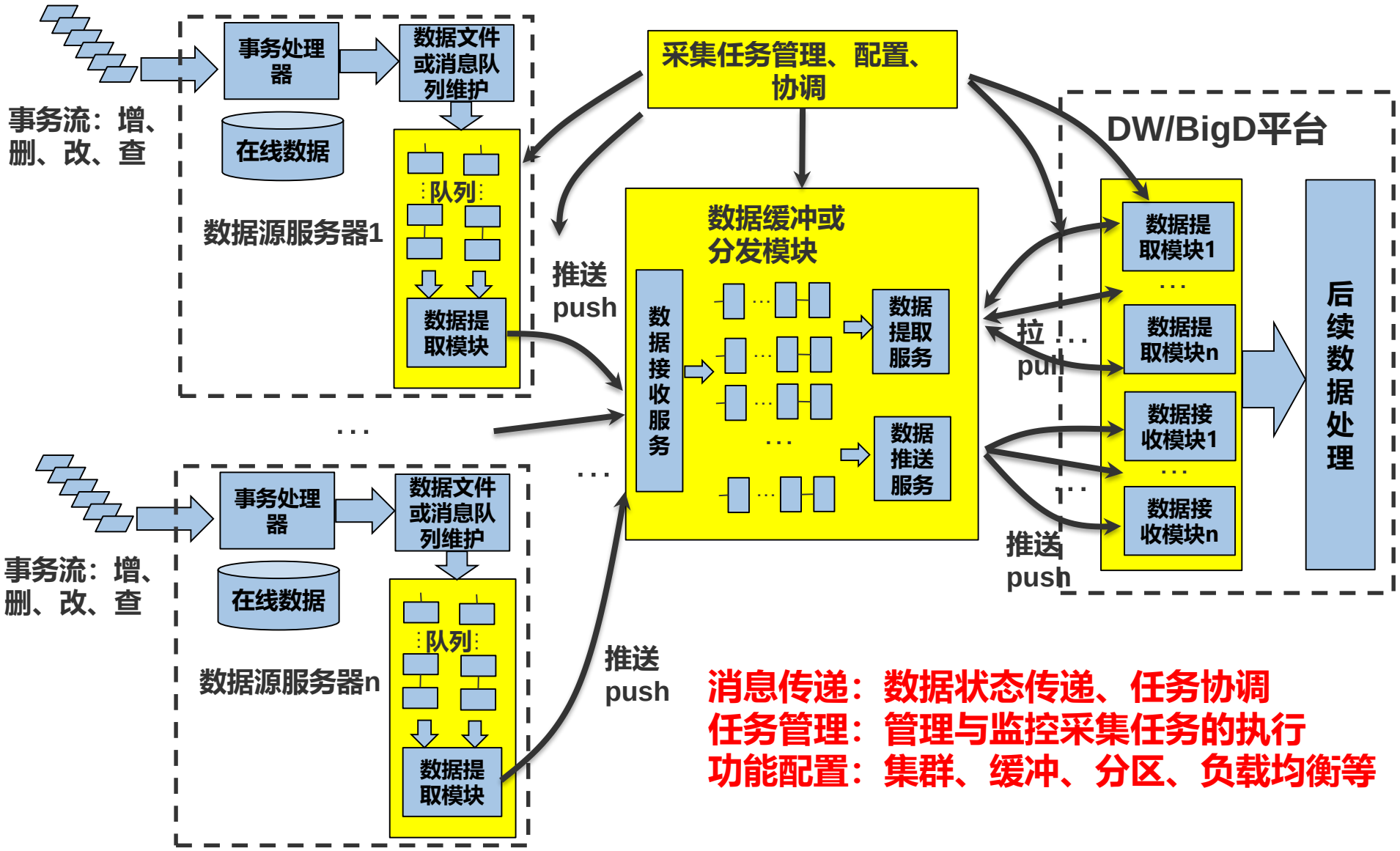
大压力下**性能瓶颈**与**集群**服务

集群服务带来的**负载均衡问题**

10. 采集任务实现与执行

- 各层采集任务实现
  - 一般需要**人工定制编写**采集各环节的程序
- 采集任务的执行模式
  - 人工或事件驱动**偶发执行**
  - **周期性执行**—常规模式
- 需要编写实现采集计划任务
- 计划任务**定时执行**或由**工具驱动**
- 一般需要对采集计划任务进行管理

11. 采集任务管理与采集器功能配置



消息传递：数据状态传递、任务协调
任务管理：管理与监控采集任务的执行
功能配置：集群、缓冲、分区、负载均衡等

缓冲队列的故障、持久化问题

- 队列存储表示机制
  - 内存队列，内存队列加日志
  - 文件队列：持久化
- 可能的故障
  - 系统重启、介质故障
- 故障容忍度
  - 是否允许数据丢失
  - 是否采用内存队列
- ACID支持问题

内容提纲

- 数据生命周期
- 数据产生、类型与规模
- 数据生态圈及OLTP系统生态圈
- OLTP系统与DW/BD平台间数据流
- 面向OLTP系统的数据采集与传输方法
- **常见数据采集工具介绍**
- 本部分实验要求

数据采集相关常见的工具

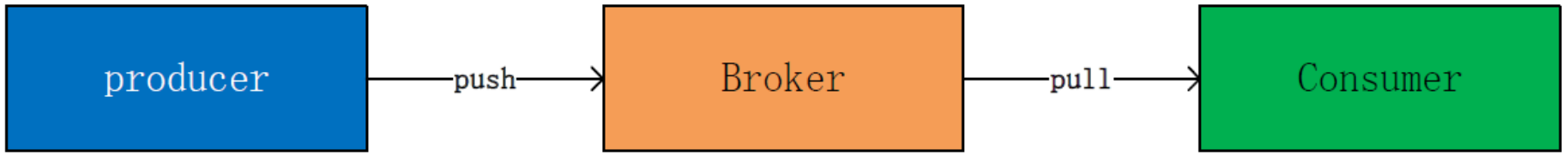
- Kafka
- MQ
- Flume
- Datastage
- Powercenter
- Teradata ETL Automation
- Informatica
- …

1. Kafka简介

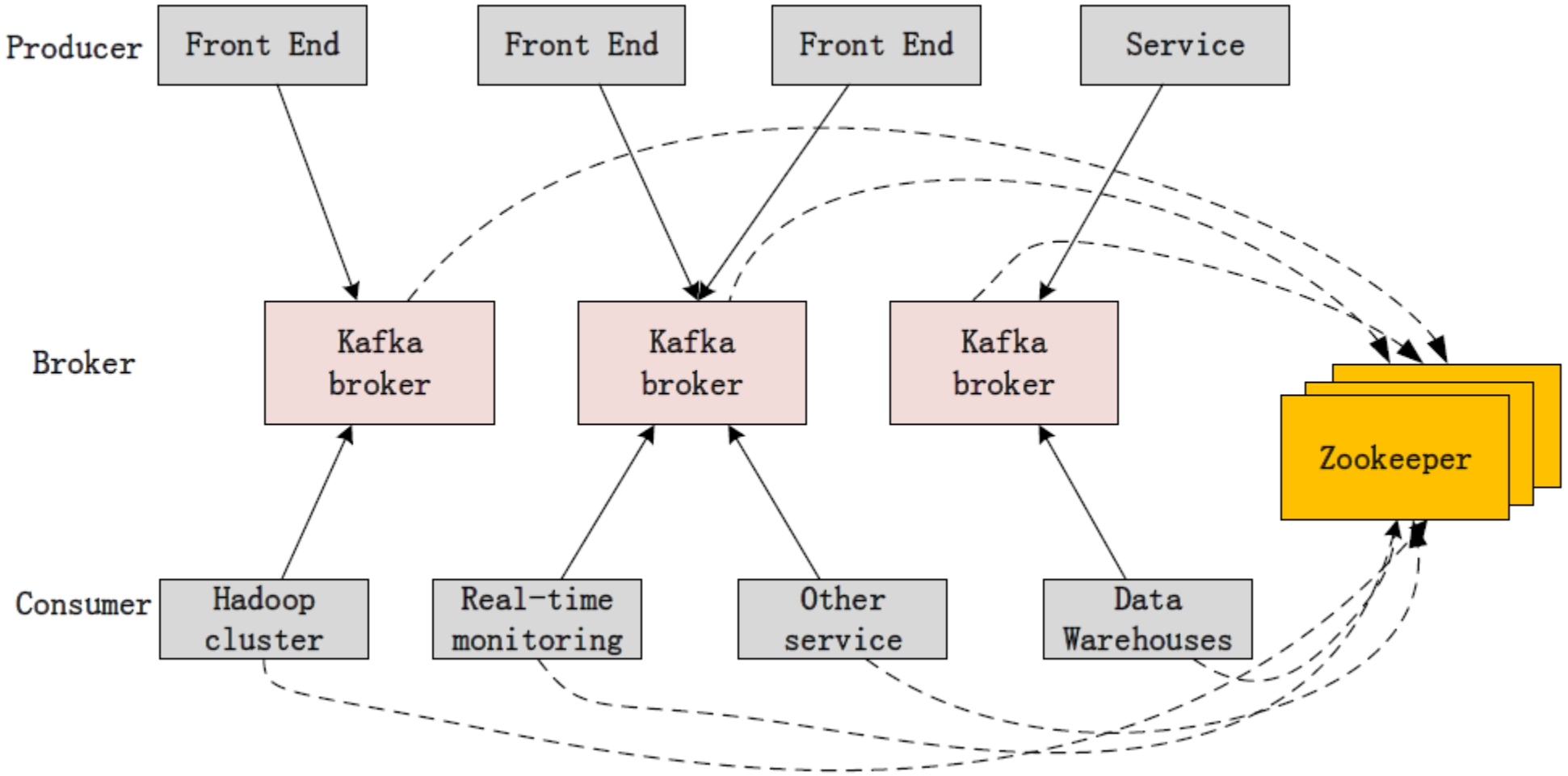
- Kafka是最初由Linkedin公司开发，是一个分布式、分区的、多副本的、多订阅者，基于zookeeper协调的分布式日志系统，常见可以用于web/nginx日志、访问日志，消息服务等等，Linkedin于2010年贡献给了Apache基金会并成为顶级开源项目。
  - 以时间复杂度为O(1)的方式提供消息持久化能力，即使对TB级以上数据也能保证常数时间复杂度的访问性能。
  - 高吞吐率。即使在非常廉价的商用机器上也能做到单机支持每秒100K条以上消息的传输。
  - 支持Kafka Server间的消息分区，及分布式消费，同时保证每个Partition内的消息顺序传输。
  - 同时支持离线数据处理和实时数据处理。
  - Scale out：支持在线水平扩展

基本术语

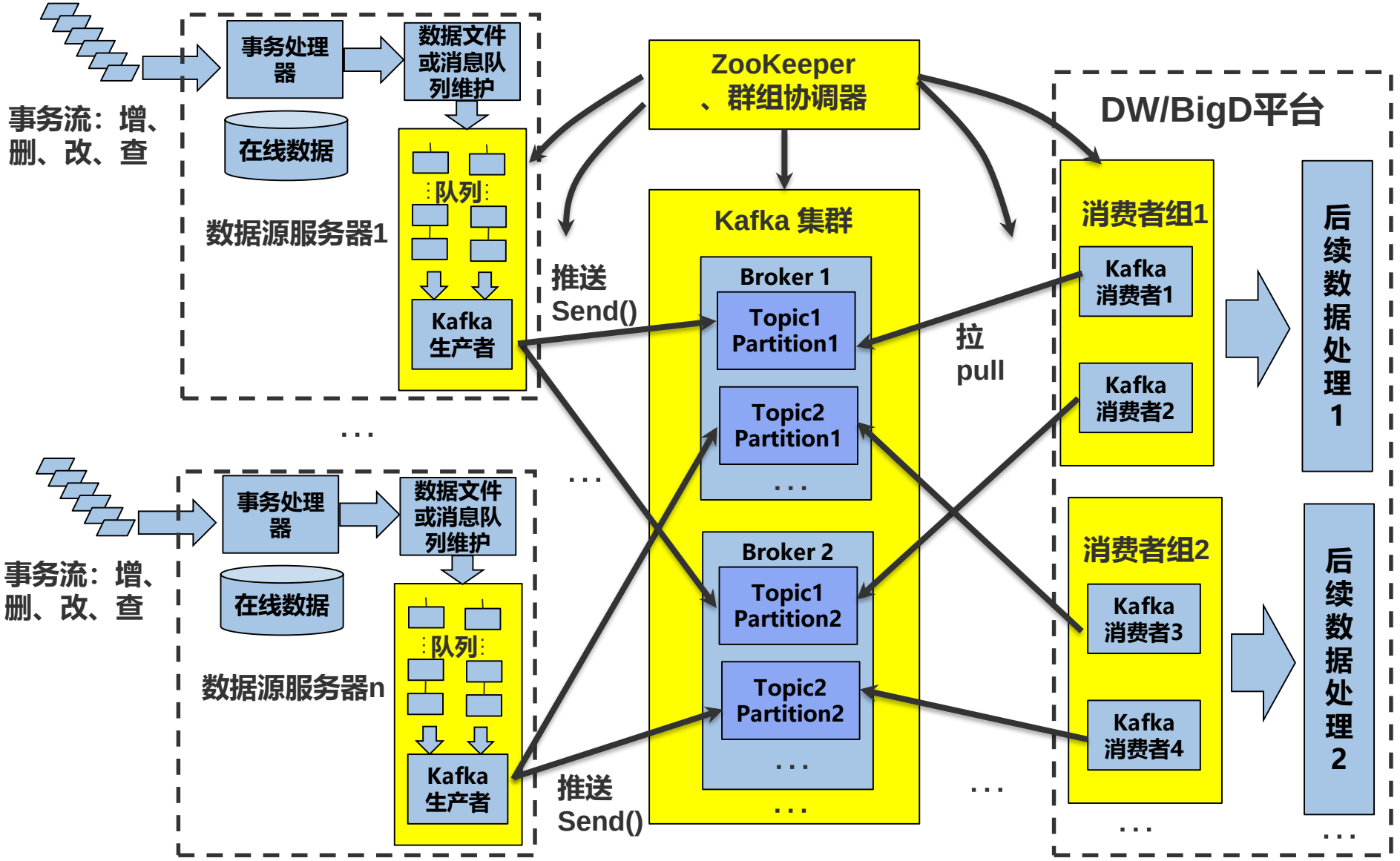
- 代理（broker）：组成Kafka 集群的单元。Kafka以一个拥有一台或多台服务器的分布式集群形式运行着，每一台服务器称为broker。
- 生产者（producer）：根据对于主题的选择向Kafka 的发布消息，即向broker push 消息的一系列进程。生产者负责决定某一条消息该被被发往选定主题（topic）的哪一个分区（partition）。
- 消费者（consumer）：向主题注册，并且接收发布到这些主题的消息，即消费一类消息的进程或集群。



分布式消息订阅和发布系统架构

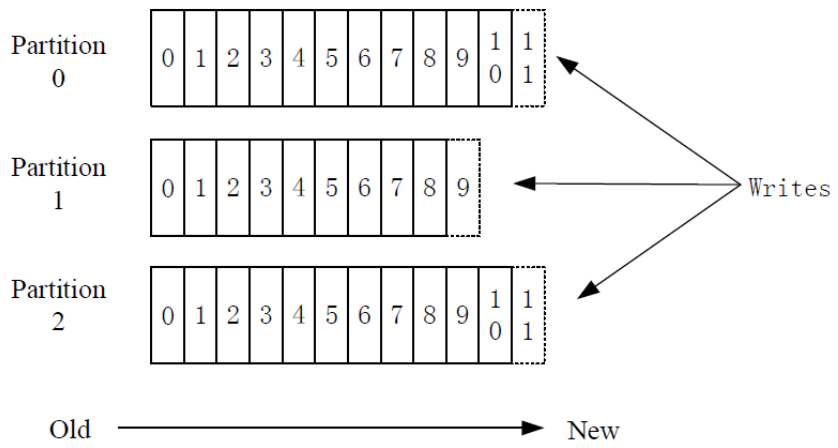


应用实例：Kafka—行业最流程集成架构



基本术语

- 主题（topic）：Kafka 用于区分所发布消息的类别，即一个主题包含一类消息。
- 分区（partitions）：Kafka 为于每一个主题维护了若干个队列，称为分区。
  - 假设有一个拥有3 个分区的主题，其中主题（topic）和分区关系如下图



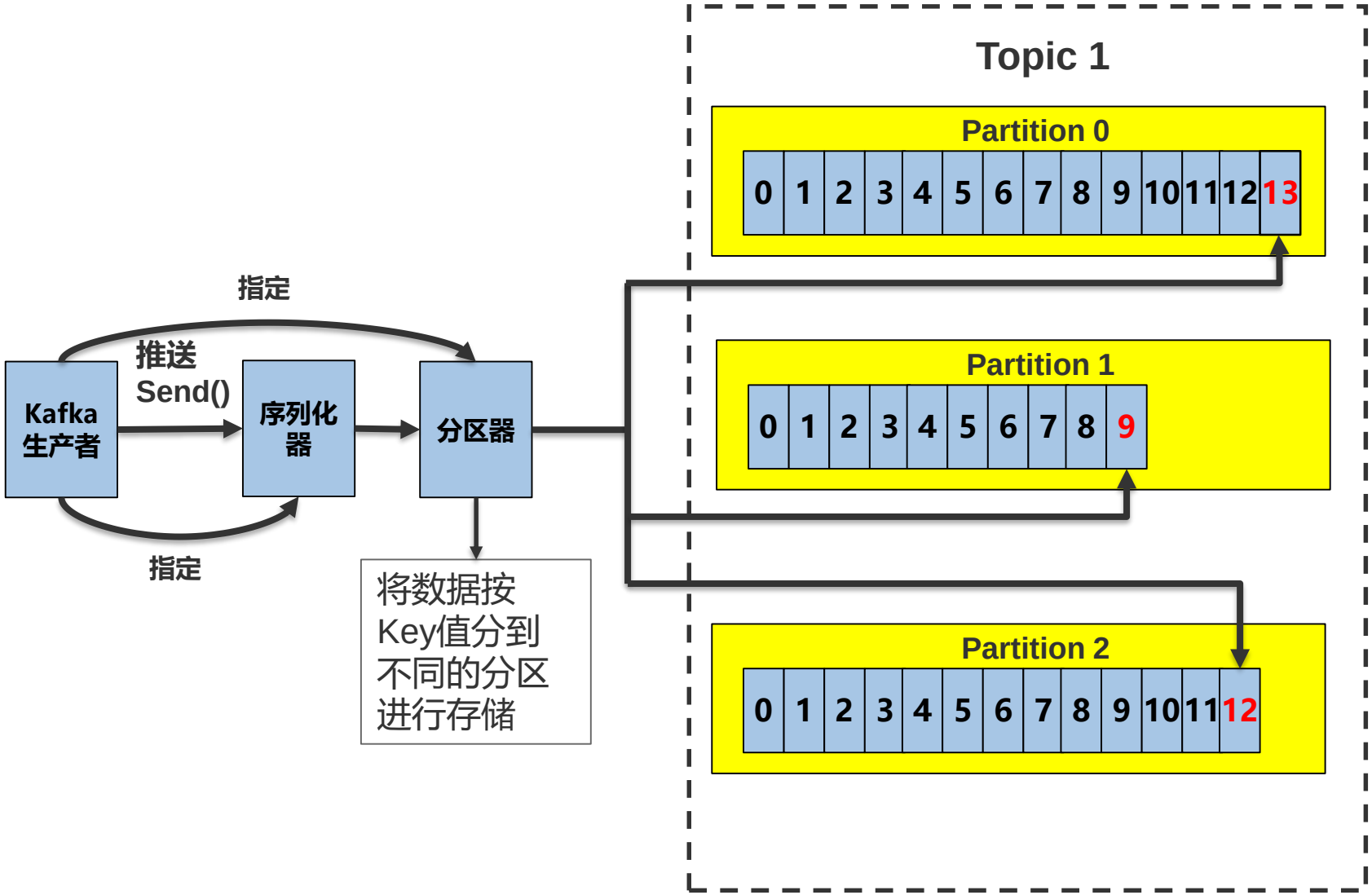
Kafka 中每个主题的每一个分区是一个有序写入、不可变的消息序列，一个topic 下可以拥有多个分区。

消息偏移量（offset）是Kafka 赋予每个分区（partition）内的每条消息一个唯一的递增的序列号，称为消息偏移量（offset）。

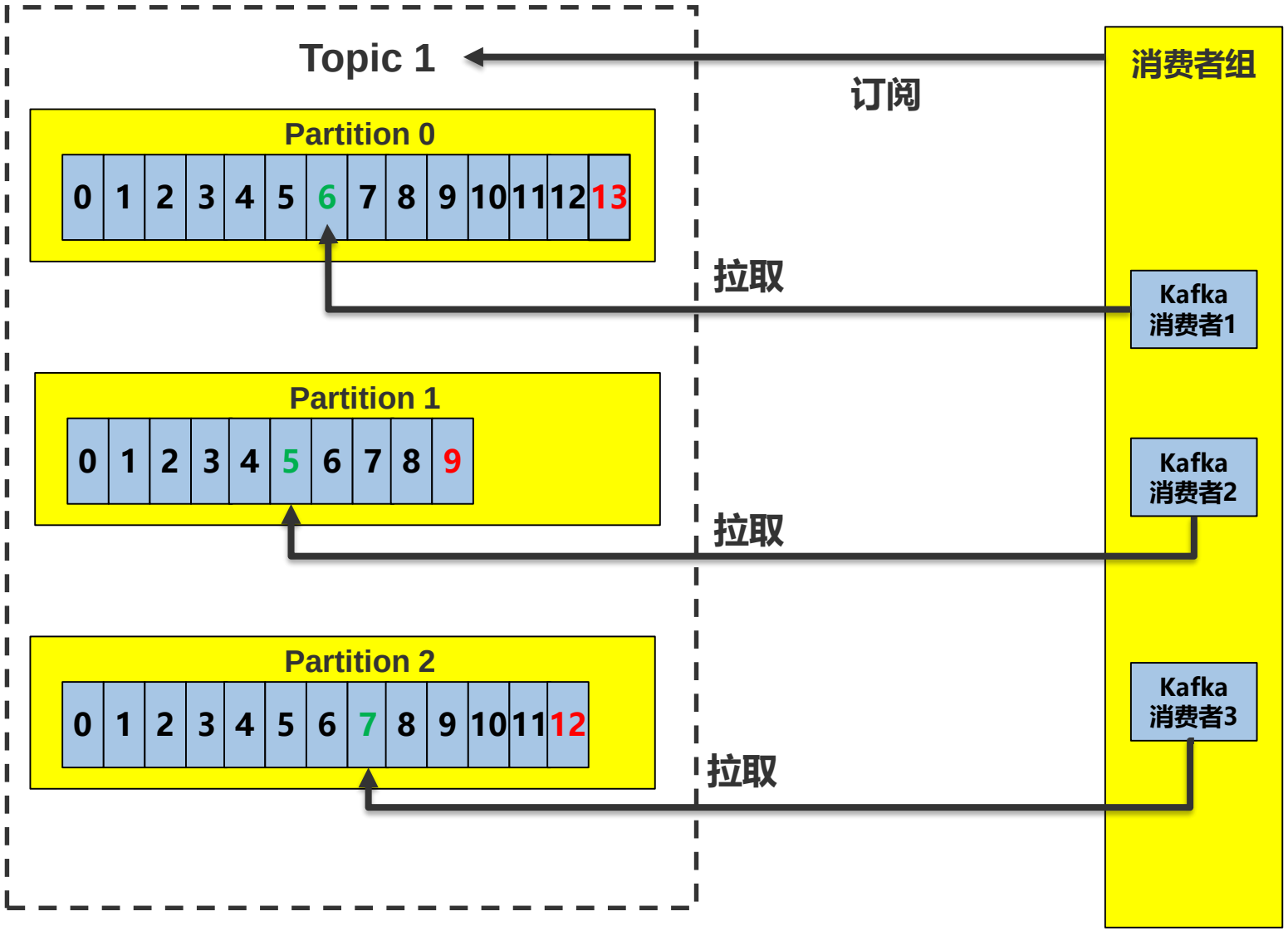
基本术语

- 消费者群组（Consumer Group）是有若干个消费者组成的集体。每个Consumer 属于一个特定的Consumer Group。
- 建组的**主要目的，方便通知**
  - Kafka 采用将Consumer 分组的方式实现一个主题（Topic）的消息的广播（发给所有的Consumer）和单播（发给某一个Consumer）。
- 副本（replications）：即分区的备份，以便容错，分布在其他broker 上，每个broker 上只能有这个分区的0 到1 个副本，即最多只能有一个。

Kafka 生产者消息发送

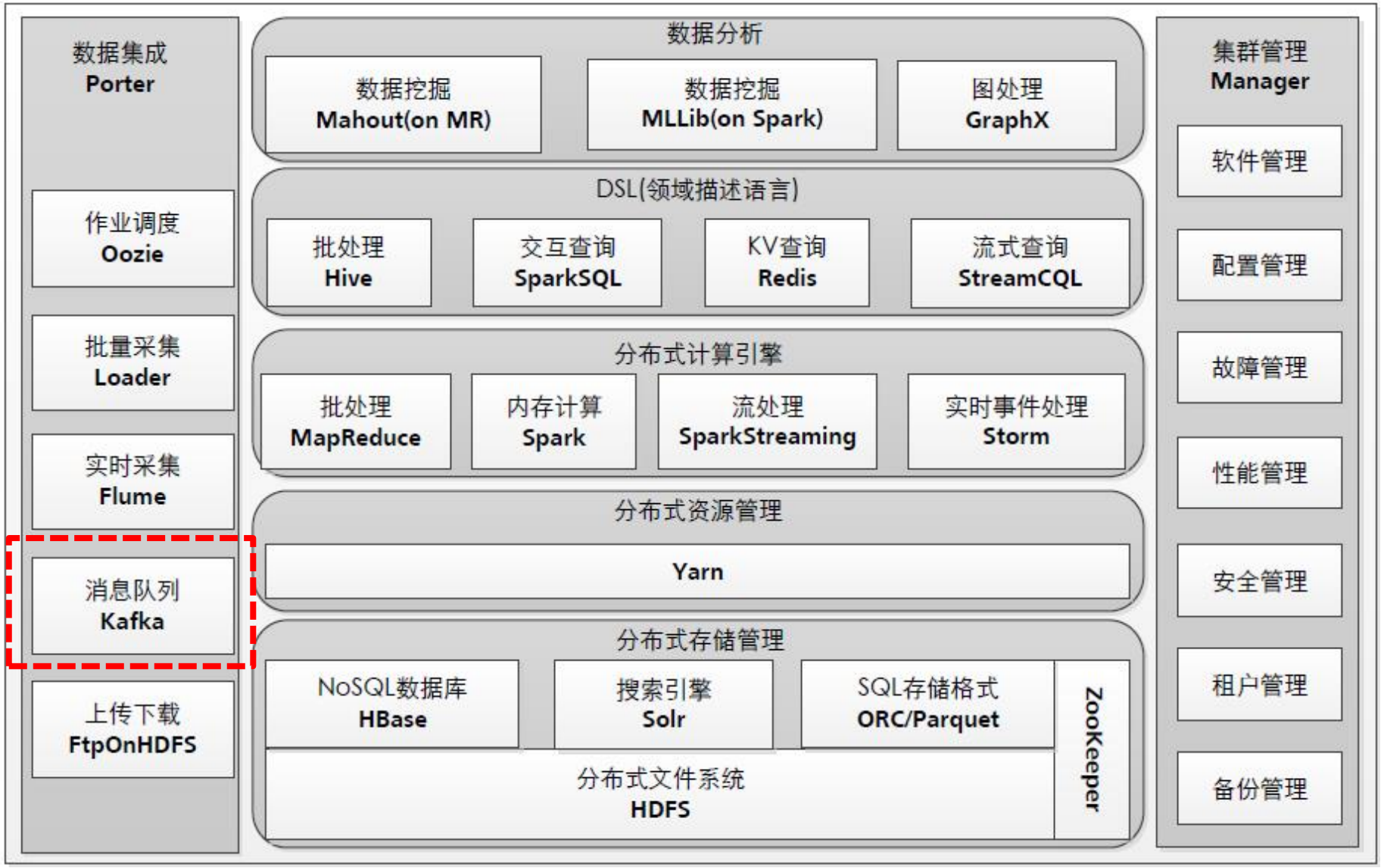


Kafka 消费者消息拉取



分区的分配由群组协调器进行维护

Kafka的位置


数据集成
Porter
作业调度
Oozie
批量采集
Loader
实时采集
Flume
消息队列
Kafka
上传下载
FtpOnHDFS
数据分析
数据挖掘
Mahout(on MR)
数据挖掘
MLLib(on Spark)
图处理
GraphX
DSL(领域描述语言)
批处理
Hive
交互查询
SparkSQL
KV查询
Redis
流式查询
StreamCQL
分布式计算引擎
批处理
MapReduce
内存计算
Spark
流处理
SparkStreaming
实时事件处理
Storm
分布式资源管理
Yarn
分布式存储管理
NoSQL数据库
HBase
搜索引擎
Solr
SQL存储格式
ORC/Parquet
分布式文件系统
HDFS
ZooKeeper
集群管理
Manager
软件管理
配置管理
故障管理
性能管理
安全管理
租户管理
备份管理

分布式消息订阅和发布系统架构

- 多个代理者（broker）协同合作，组成了Kafka 集群。
- Kafka 的集群架构采用P2P（peer to peer）模式。集群中没有**主节点**，所有节点都平等作为消息的处理节点。
  - 优点是没有单点问题，一部分节点宕机，服务仍能够正常，缺点是很难达成数据的一致性和多机备份，如果一部分节点宕机会导致数据的丢失。
- Kafka 为避免上述的问题采用主节点选举机制
  - 利用ZooKeeper，对于每一个主题（topic）的分区（partitions），选出一个leader-broker（主节点），其余broker 为followers（从节点），leader 处理消息的写入和备份；当leader 宕机，采用选举算法，从followers中选出新的leader，以保障服务的可用，同时保障了消息的备份和一致性。

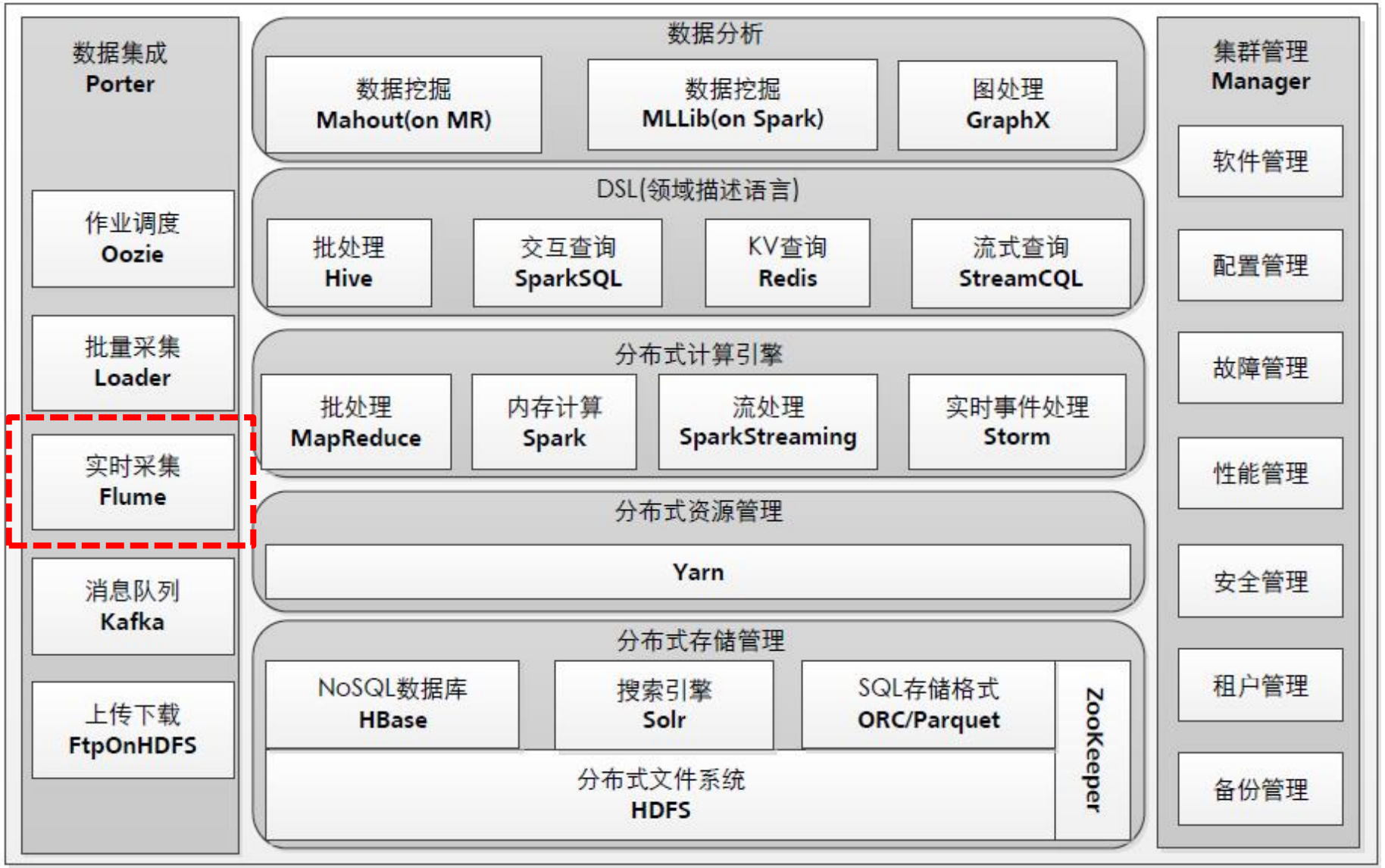
分布式消息订阅和发布系统架构

- 生产者（producer）和消费者（consumer）部署在各个业务逻辑中被频繁地调用，三者通过ZooKeeper管理协调请求和转发。
- Producer 到broker 的过程是push，也就是有数据就推送到broker，而consumer到broker 的过程是pull，是通过consumer 主动去拉取数据的，而不是broker 把数据主动发送到consumer端的。

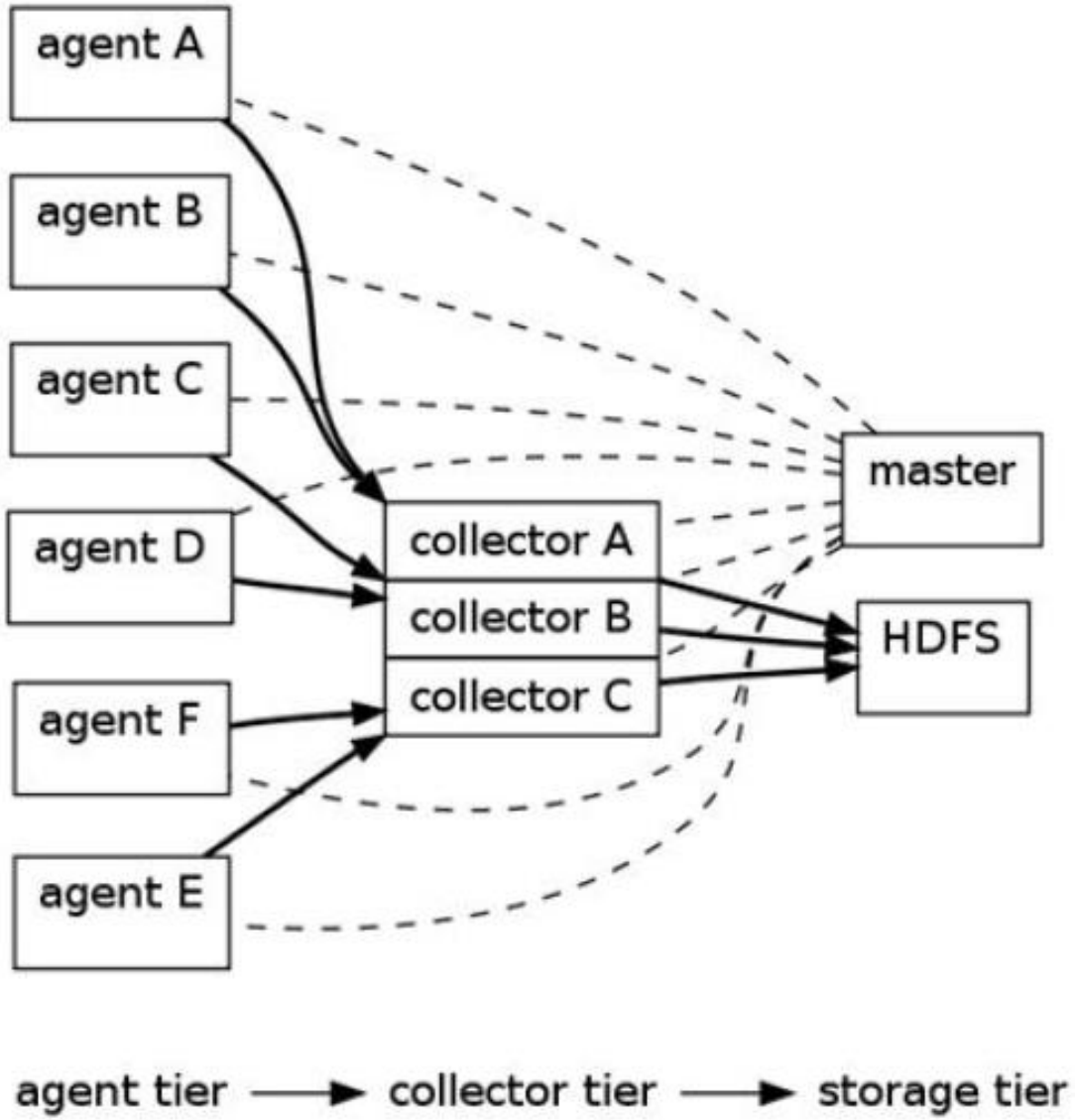
2. Flume简介

- Flume最早是Cloudera提供的日志收集系统，目前是Apache下的一个孵化项目，Flume支持在日志系统中定制各类数据发送方，用于收集数据；
- 同时，Flume提供对数据进行简单处理，并写到各种数据接受方（可定制）的能力。 Flume是一个分布式、可靠和高可用的海量日志采集、聚合和传输的系统。

Flume的位置


数据集成
Porter
作业调度
Oozie
批量采集
Loader
实时采集
Flume
消息队列
Kafka
上传下载
FtpOnHDFS
数据分析
数据挖掘
Mahout(on MR)
数据挖掘
MLLib(on Spark)
图处理
GraphX
DSL(领域描述语言)
批处理
Hive
交互查询
SparkSQL
KV查询
Redis
流式查询
StreamCQL
分布式计算引擎
批处理
MapReduce
内存计算
Spark
流处理
SparkStreaming
实时事件处理
Storm
分布式资源管理
Yarn
分布式存储管理
NoSQL数据库
HBase
搜索引擎
Solr
SQL存储格式
ORC/Parquet
分布式文件系统
HDFS
ZooKeeper
集群管理
Manager
软件管理
配置管理
故障管理
性能管理
安全管理
租户管理
备份管理

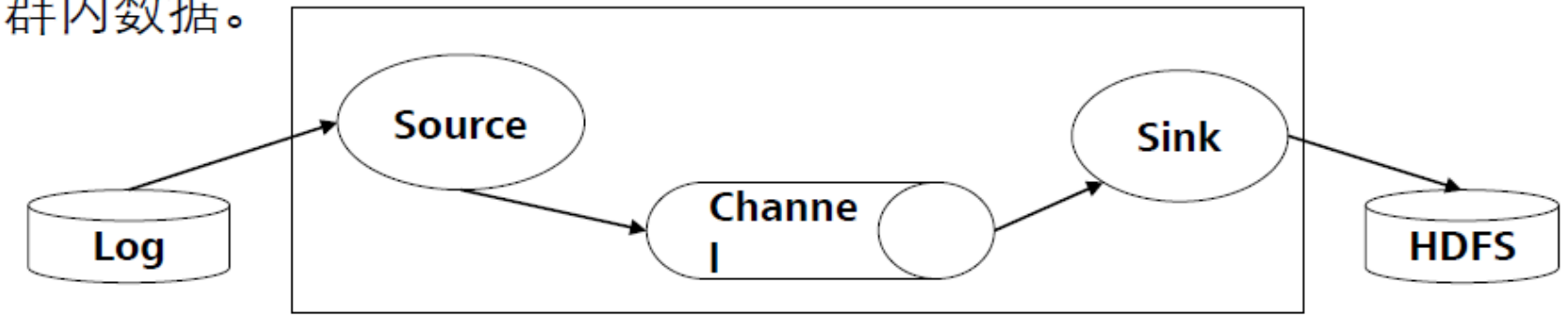
Flume架构



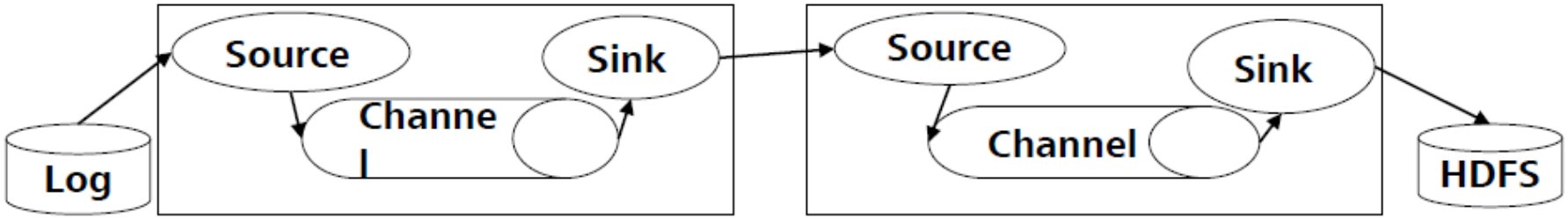
- Flume的架构中最重要的抽象是data flow（数据流），data flow描述了数据从产生，传输、处理并最终写入目标的一条路径（实线描述了data flow）。 Agent用于采集数据，agent是flume中产生数据流的地方，同时，agent会将产生的数据流传输到collector。对应的，collector用于对数据进行聚合，往往会产生一个更大的流。

Flume架构

- **Flume**基础架构：**Flume** 可以单节点直接采集数据，主要应用于集群内数据。



- **Flume**多**agent**架构：**Flume**可以将多个节点连接起来，将最初的数据源经过收集，存储到最终的存储系统中。主要应用于集群外的数据导入到集群内。



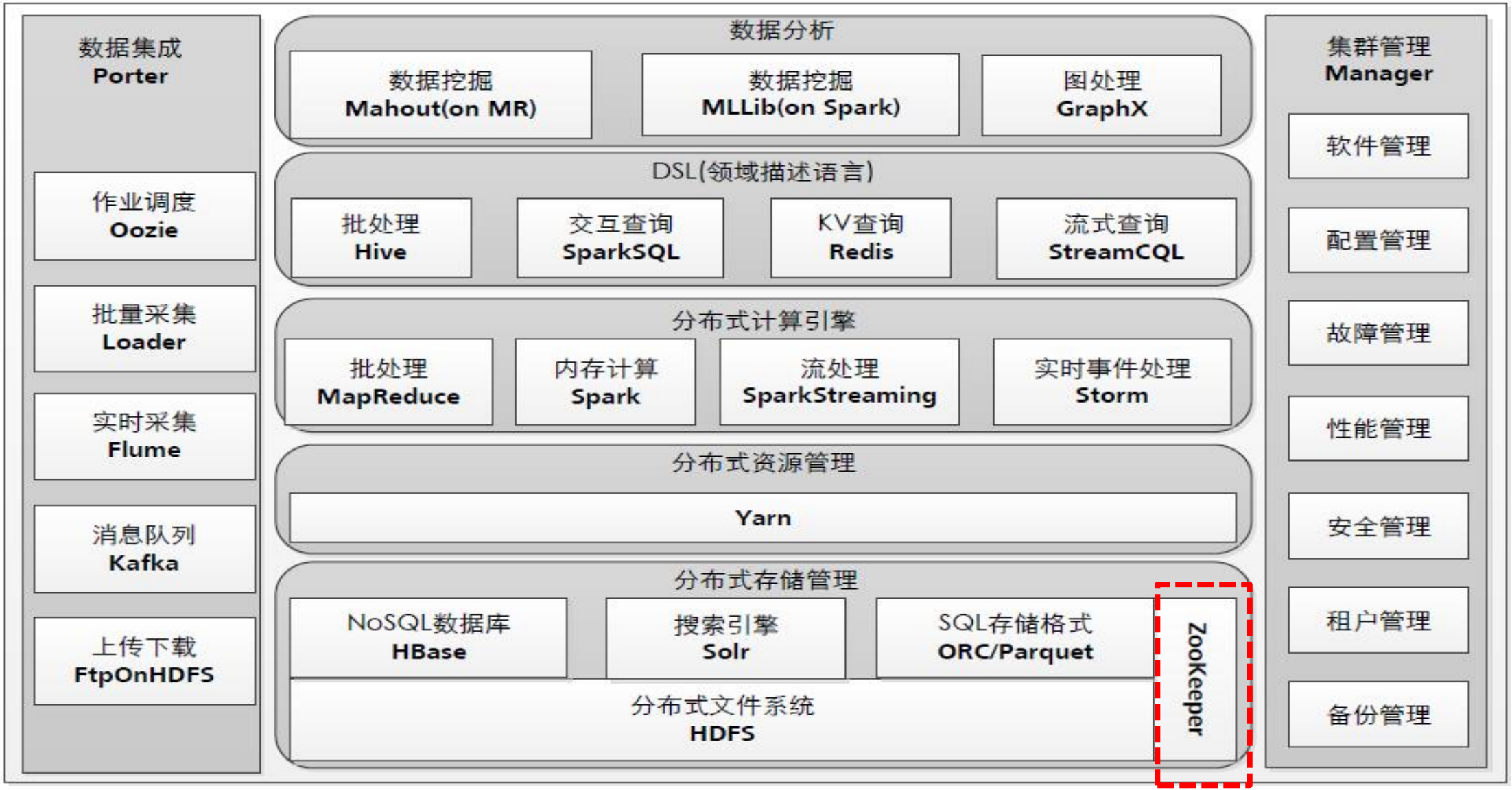
Flume架构

- 收集数据有2种主要工作模式，如下：
  - 1. Push Sources：外部系统会主动地将数据推送到Flume中，如RPC、syslog。
  - 2. Polling Sources：Flume到外部系统中获取数据，一般使用轮询的方式，如text和exec。
- 注意，在Flume中，agent和collector对应，而source和sink对应。Source和sink强调发送、接受方的特性（如数据格式、编码等），而agent和collector关注功能。

3. ZooKeeper简介

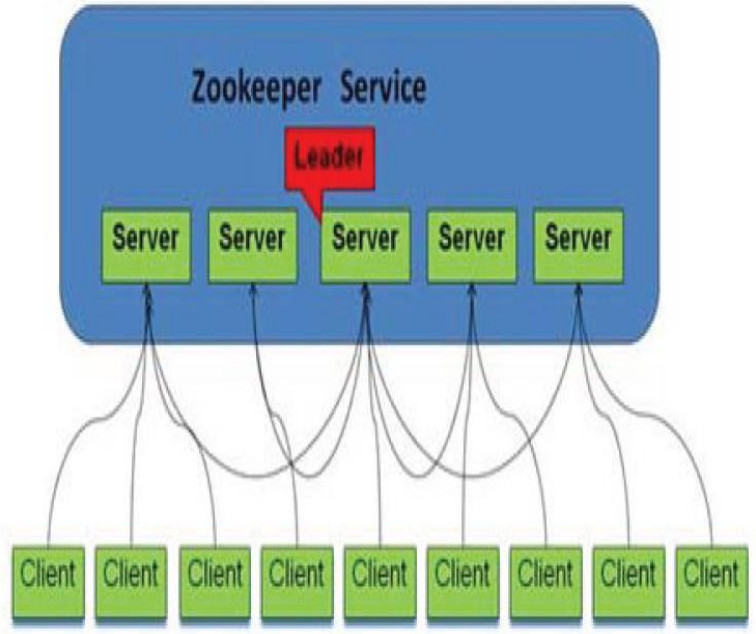
- ZooKeeper，是一个高可用的分布式数据管理与系统协调框架，是Hadoop的一个子项目，是一种分布式的、开源的、应用于分布式应用的**协作服务**，主要是用来解决分布式应用中经常遇到的一些数据管理问题，如：**统一命名服务、状态同步服务、集群管理、分布式应用配置项的**管理等
- ZooKeeper 作为一个分布式的服务框架，主要用来解决分布式集群中应用系统的一致性问题，它能提供基于类似于文件系统的目录节点树方式的数据存储，但是 ZooKeeper 并不是用来专门存储数据的，它的作用主要是用来维护和监控你存储的数据的状态变化。**通过监控这些数据状态的变化**，从而可以达到基于数据的集群管理。

ZooKeeper的位置



ZooKeeper的系统架构

- ZooKeeper不仅可以单机提供服务，同时也支持多机组成集群来提供服务（如图所示）。
- ZooKeeper中的角色主要包括领导者、学习者和客户端



角色		描述
领导者（Leader）		领导者负责进行投票的发起和决议，更新系统状态
学习者（Learner）	跟随者（Follower）	Follower 用于接收客户请求并向客户端返回结果，在选主过程中参与投票
	观察者（ObServer）	ObServer 可以接收客户端连接，将写请求转发给 leader 节点。但 ObServer 不参加投票过程，只同步 leader 的状态。ObServer 的目的是为了扩展系统，提高读取速度
客户端（Client）		请求发起方

ZooKeeper的设计目的

- 最终一致性：client不论连接到哪个Server，展示给它都是同一个视图，这是ZooKeeper最重要的性能；
- 可靠性：具有简单、健壮、良好的性能，如果消息被其中一台服务器接受，那么它将被所有的服务器接受；
- 实时性：ZooKeeper保证客户端将在一个时间间隔范围内获得服务器的更新信息，或者服务器失效的信息；但由于网络延时等原因，ZooKeeper不能保证两个客户端能同时得到刚更新的数据，如果需要最新数据，应该在读数据之前调用sync()接口；
- 等待无关（wait-free）：慢的或者失效的client不得干预快速的client的请求，使得每个client都能有效的等待；
- 原子性：更新只能成功或者失败，没有中间状态；

Zookeeper的特点

- 顺序性：包括全局有序和偏序两种：全局有序是指如果在一台服务器上消息a在消息b前发布，则在所有Server上消息a都将在消息b前被发布；偏序是指如果一个消息b在消息a后被同一个发送者发布，a必将排在b前面。
- 简易性：通过一种和文件系统很像的层级命名空间来让分布式进程互相协同工作，实现了高性能、高可靠性和有序的访问。
- 可用性：组成ZooKeeper的各个服务器之间能够相互通信，在内存中保存服务器状态，也保存了操作日志，并且持久化快照，只要大多数服务器是可用的，那么，ZooKeeper就是可用的。
- 有序性：使用数字来对每个更新进行标记，保证ZooKeeper交互的有序。
- 高效性：表现在以读为主的系统上。

内容提纲

- 数据生命周期
- 数据产生、类型与规模
- 数据生态圈及OLTP系统生态圈
- OLTP系统与DW/BD平台间数据流
- 面向OLTP系统的数据采集与传输方法
- 常见数据采集工具介绍
- **本部分实验要求**

实验二：大数据采集技术实验

- 实验环境
  - Kafka集群；
  - MySQL数据库；
  - 编程语言：Java（推荐使用）、Scala、C++等；

实验二：大数据采集技术实验

实验内容

- 首先，设计用于存储结构化的用户行为日志数据的MySQL数据模型
- 其次，设计并实现三种不同类型的模拟数据抓取的Kafka生产者，将文本文件形式的日志数据发送到Kafka集群中的相应话题中。三种不同类型的Kafka生产者为：
  - 模拟单数据源实时抓取的Kafka生产者；
  - 模拟多数据源实时抓取的Kafka生产者；
  - 模拟多粒度数据抓取的Kakfa生产者；
- 最后，设计并实现一个模拟数据解析的Kafka消费者，将Kafka集群中相应话题中的文本形式的日志数据记录解析为结构化数据，并存储到MySQL中自己设计的相应关系表中。

实验二：大数据采集技术实验

- 实验步骤
  - 设计存储结构化日志数据的MySQL关系表；
  - 根据原始数据情况，创建Kafka话题并尝试设置不同的分区数目进行后续实验；
  - 根据原始数据和创建的Kafka话题，设计并实现三种不同类型的模拟日志抓取的Kafka生产者；
  - 根据Kafka中存储日志数据的话题中的数据量情况，设计并实现日志解析与结构化数据存储的Kafka消费者；
- 实验输出
  - 按照给定实验报告模板制作实验报告
  - 包含独自实验过程中的关键分析、图、表、代码和截图

本部分结束！