架构部
王刚

聚美优品

大数据
技术栈选择与落地

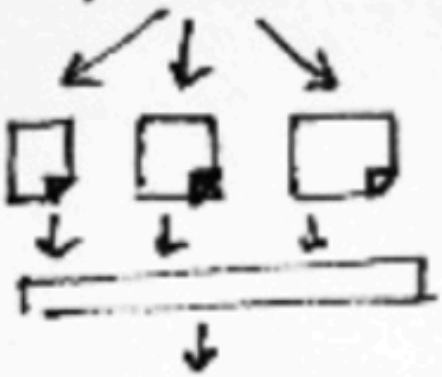
声明

本文所有内容均为个人主观理解

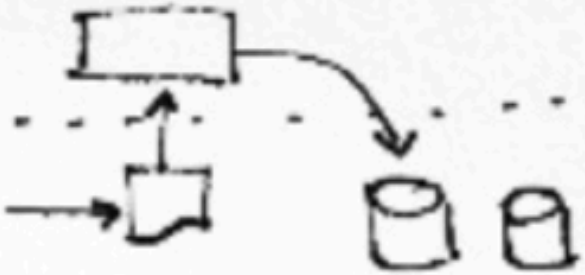
大部分资料来源于网络

GFS

MapReduce

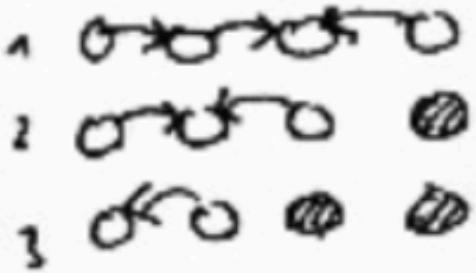


BigTable

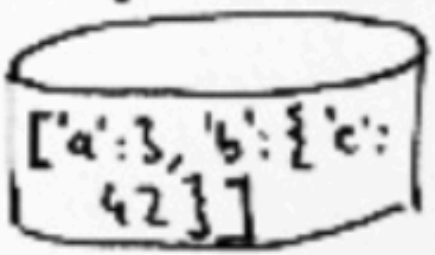


Percolator

Pregel



Dremel



Big Data can solve our problems?

NO

Big Data is just a buzzword.

You need (3R):

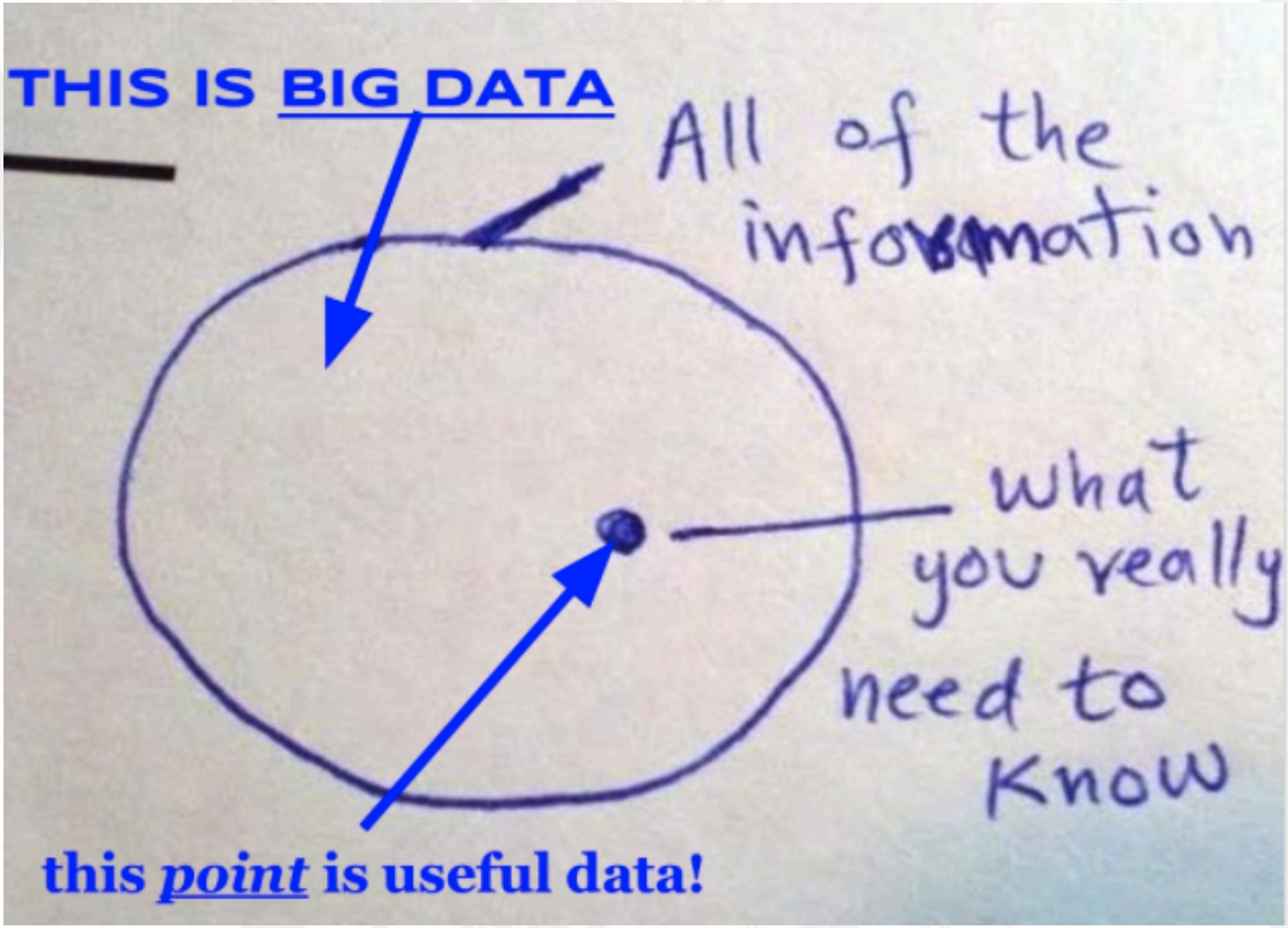
1. Solve right problems
2. Build the right team
3. Use right tools

常见应用场景

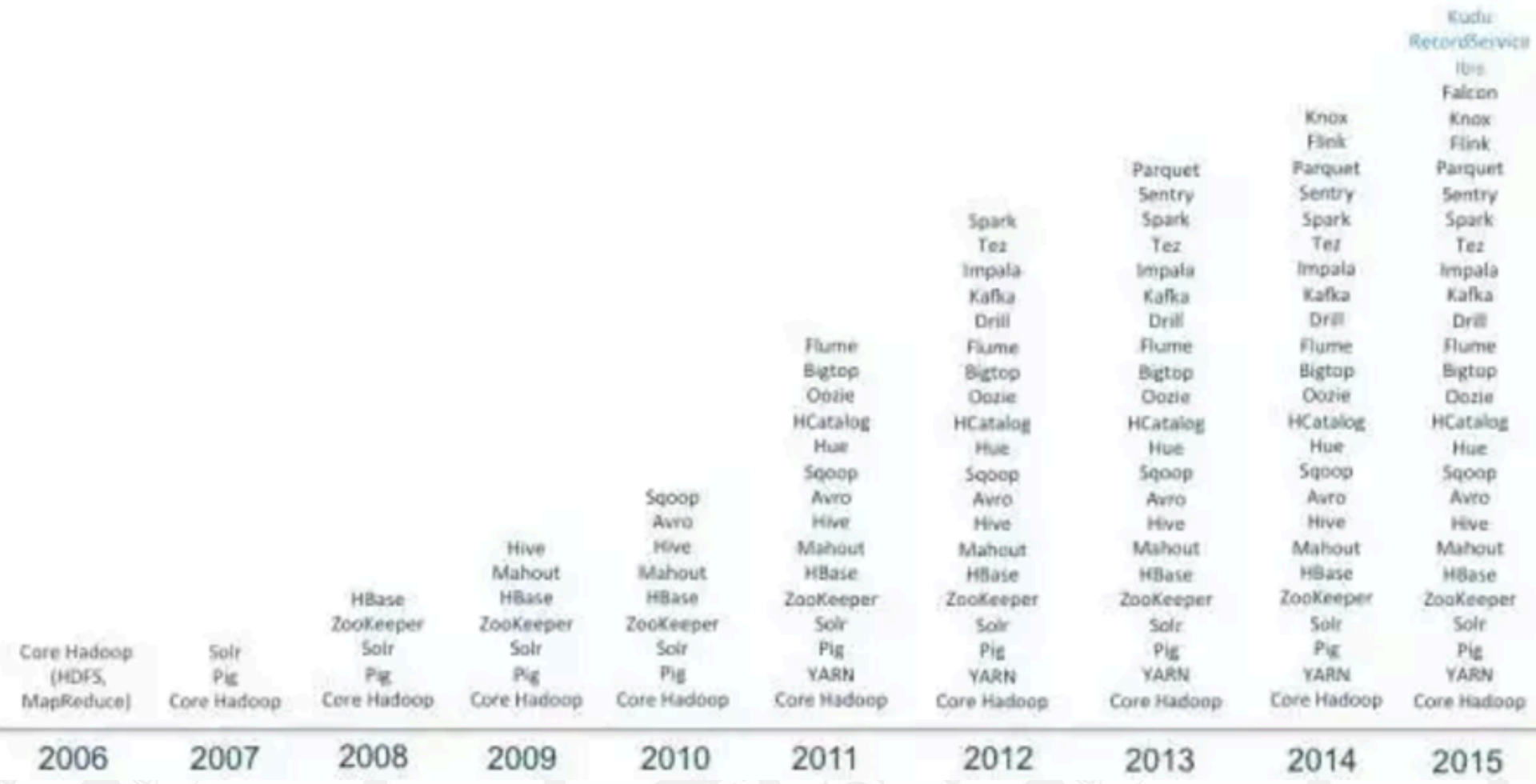
- 批处理
- 流处理
- 即席查询
- 数据更新
- 全文检索
- 图计算
- 机器学习
- ……

常见工作方向

- ETL
- OLAP
- 数仓
- 建模与挖掘
- 数据安全
- 数据可视化
- 系统研发
- 系统运维
- ……


THIS IS BIG DATA
All of the information
what you really need to know
this point is useful data!

部分主要技术


Core Hadoop (HDFS, MapReduce)
Solr Pig Core Hadoop
HBase ZooKeeper Solr Pig Core Hadoop
Hive Mahout HBase ZooKeeper Solr Pig Core Hadoop
Sqoop Avro Hive Mahout HBase ZooKeeper Solr Pig Core Hadoop
Flume Bigtop Oozie HCatalog Hue Sqoop Avro Hive Mahout HBase ZooKeeper Solr Pig YARN Core Hadoop
Spark Tez Impala Kafka Drill Flume Bigtop Oozie HCatalog Hue Sqoop Avro Hive Mahout HBase ZooKeeper Solr Pig YARN Core Hadoop
Parquet Sentry Spark Tez Impala Kafka Drill Flume Bigtop Oozie HCatalog Hue Sqoop Avro Hive Mahout HBase ZooKeeper Solr Pig YARN Core Hadoop
Knox Flink Parquet Sentry Spark Tez Impala Kafka Drill Flume Bigtop Oozie HCatalog Hue Sqoop Avro Hive Mahout HBase ZooKeeper Solr Pig YARN Core Hadoop
Kudu RecordService Ibis Falcon Knox Flink Parquet Sentry Spark Tez Impala Kafka Drill Flume Bigtop Oozie HCatalog Hue Sqoop Avro Hive Mahout HBase ZooKeeper Solr Pig YARN Core Hadoop
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015

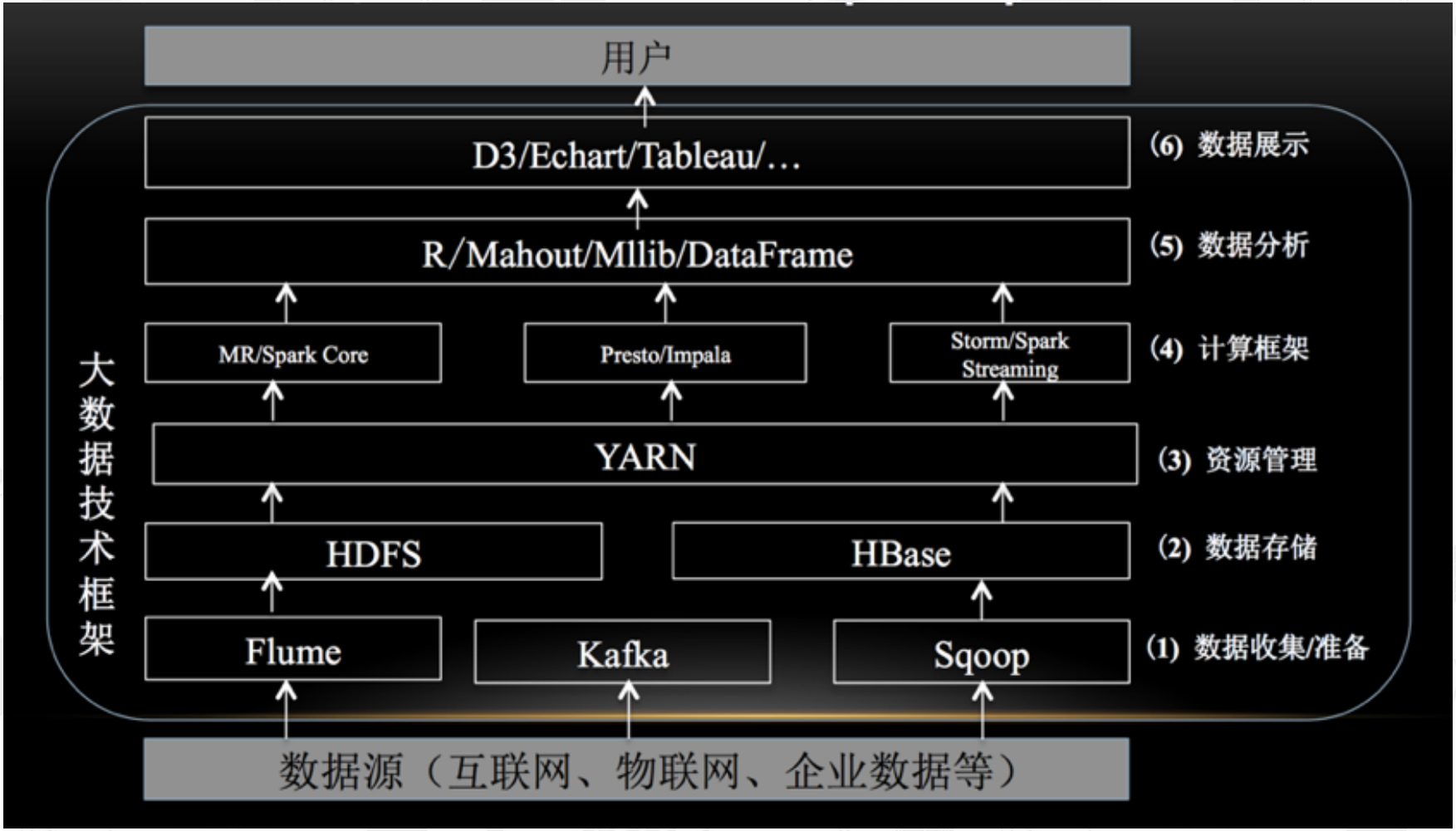
大数据主流技术栈

Data exchange	  
Script	  
Distribute engine	Phoenix Kylin       kudu
Resource Manager	YARN 
distribute FileSystem	HDFS  TACHYON lustre
Storge format	 ORCFile

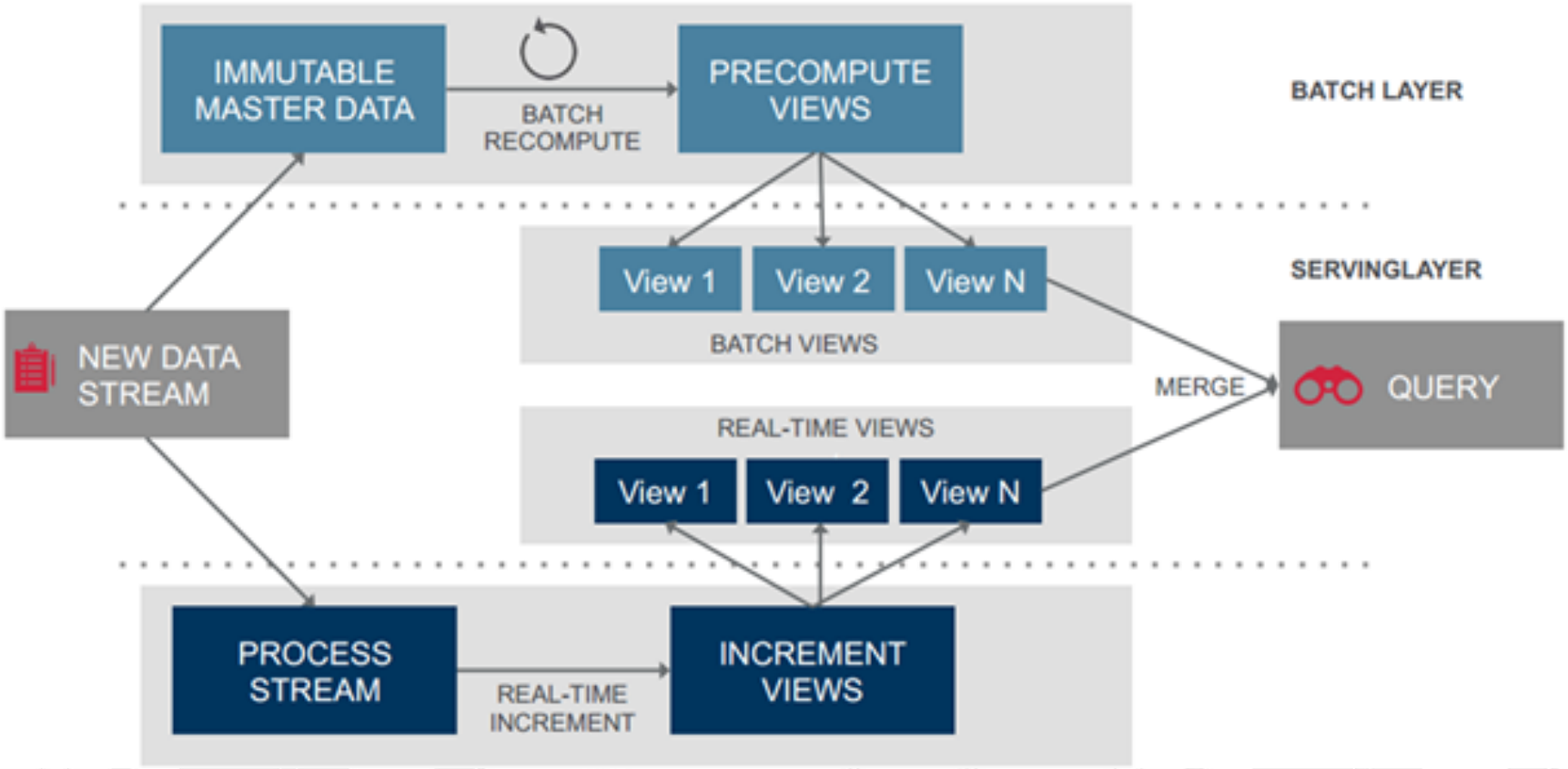
主流技术栈应用场景

DB	NO-SQL	MPP	search	streaming	graph	Machine learning	ETL
MySQL PostgreSQL	APACHE HBASE Cassandra mongoDB FOR GIANT IDEAS Couchbase	greenplum Impala HAWQ	lucene Apache Solr	APACHE STORM Flink Spark	GraphLab Spark Flink APACHE GIRAPH	MPICH Spark mahout Flink	HIVE hadoop Spark Flink

“标准通用”架构


用户
D3/Echart/Tableau/…
(6) 数据展示
R/Mahout/Mllib/DataFrame
(5) 数据分析
MR/Spark Core
Presto/Impala
Storm/Spark Streaming
(4) 计算框架
YARN
(3) 资源管理
HDFS
HBase
(2) 数据存储
Flume
Kafka
Sqoop
(1) 数据收集/准备
数据源（互联网、物联网、企业数据等）
大数据技术框架

“Lambda”架构


IMMUTABLE MASTER DATA
BATCH RECOMPUTE
PRECOMPUTE VIEWS
BATCH LAYER
View 1
View 2
View N
BATCH VIEWS
SERVINGLAYER
NEW DATA STREAM
MERGE
QUERY
REAL-TIME VIEWS
View 1
View 2
View N
PROCESS STREAM
REAL-TIME INCREMENT
INCREMENT VIEWS

“Lambda”架构

- BigData
- Data Analytics
- Reactive Programing
- Functional Programing
- Streaming Computation

=>　　Lambda

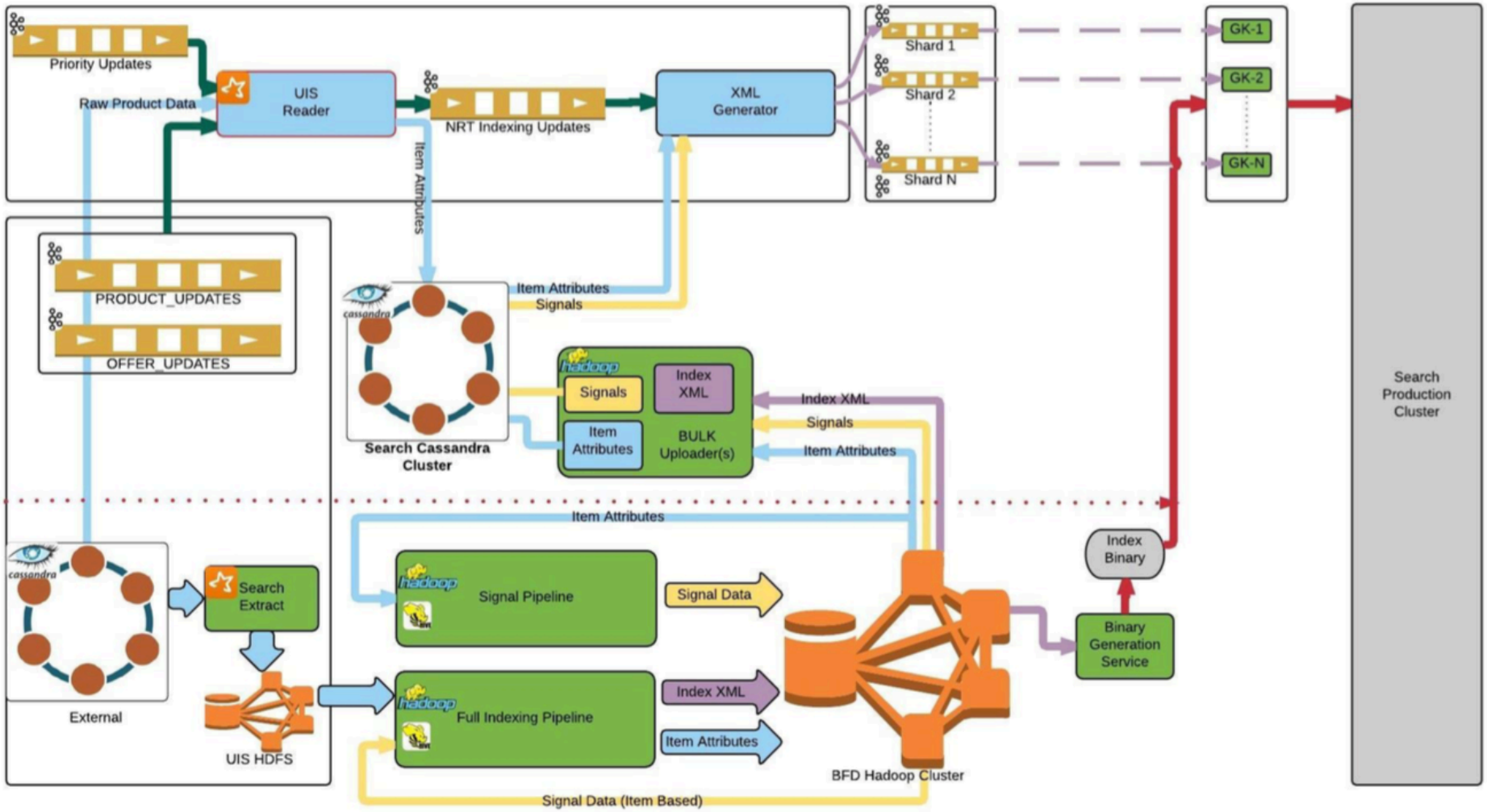
“Lambda”架构原则

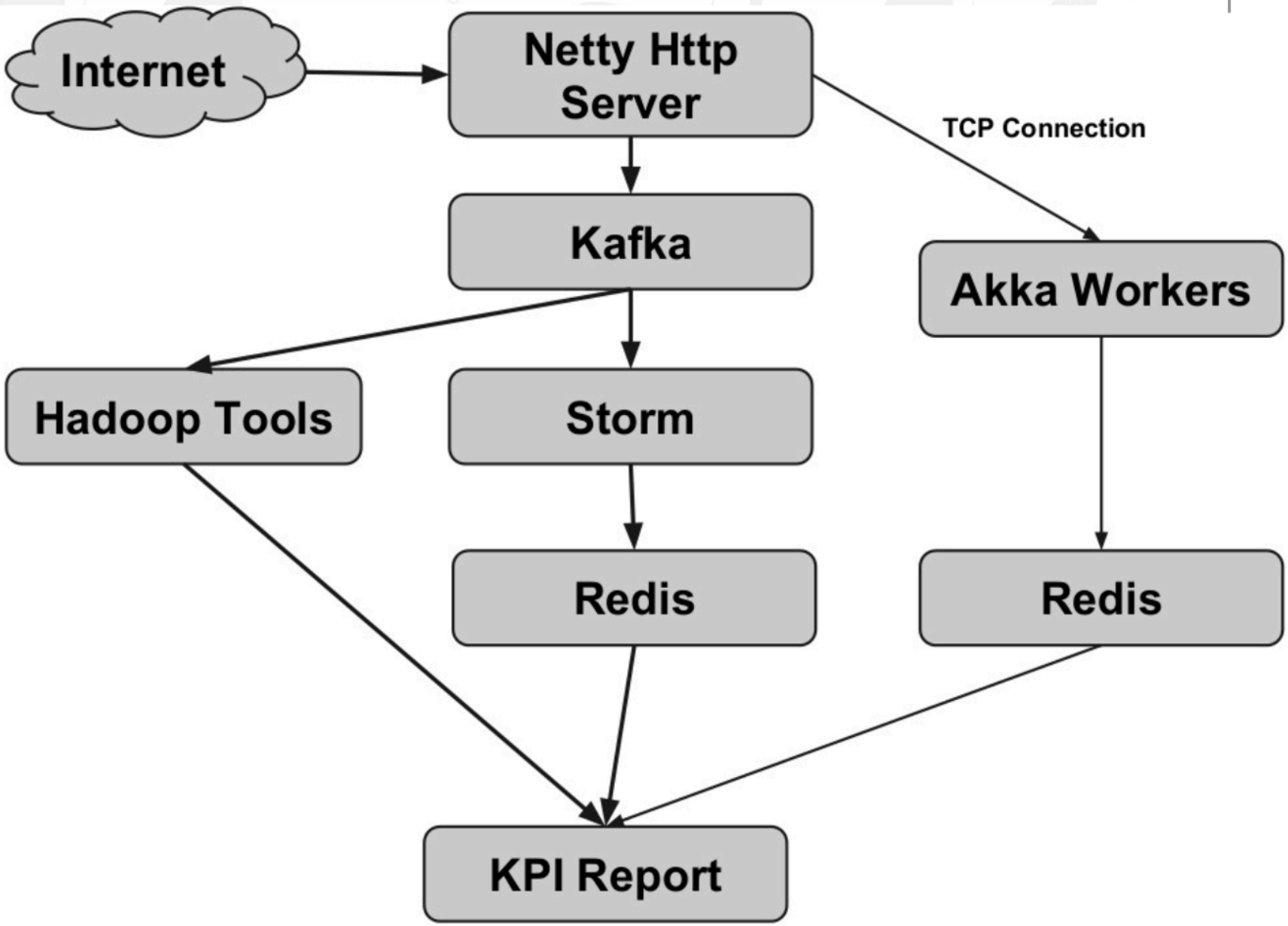
- 人为容错性(human fault-tolerance)
  - 系统易数据丢失或数据损坏，大规模时可能是不可挽回的
- 数据不可变性(data immutability)
  - 数据存储在它的最原始的形式不变的，永久的。
- 重新计算(recomputation)
  - 因为上面两个原则，运行函数重新计算结果是可能的。

技术选型与落地

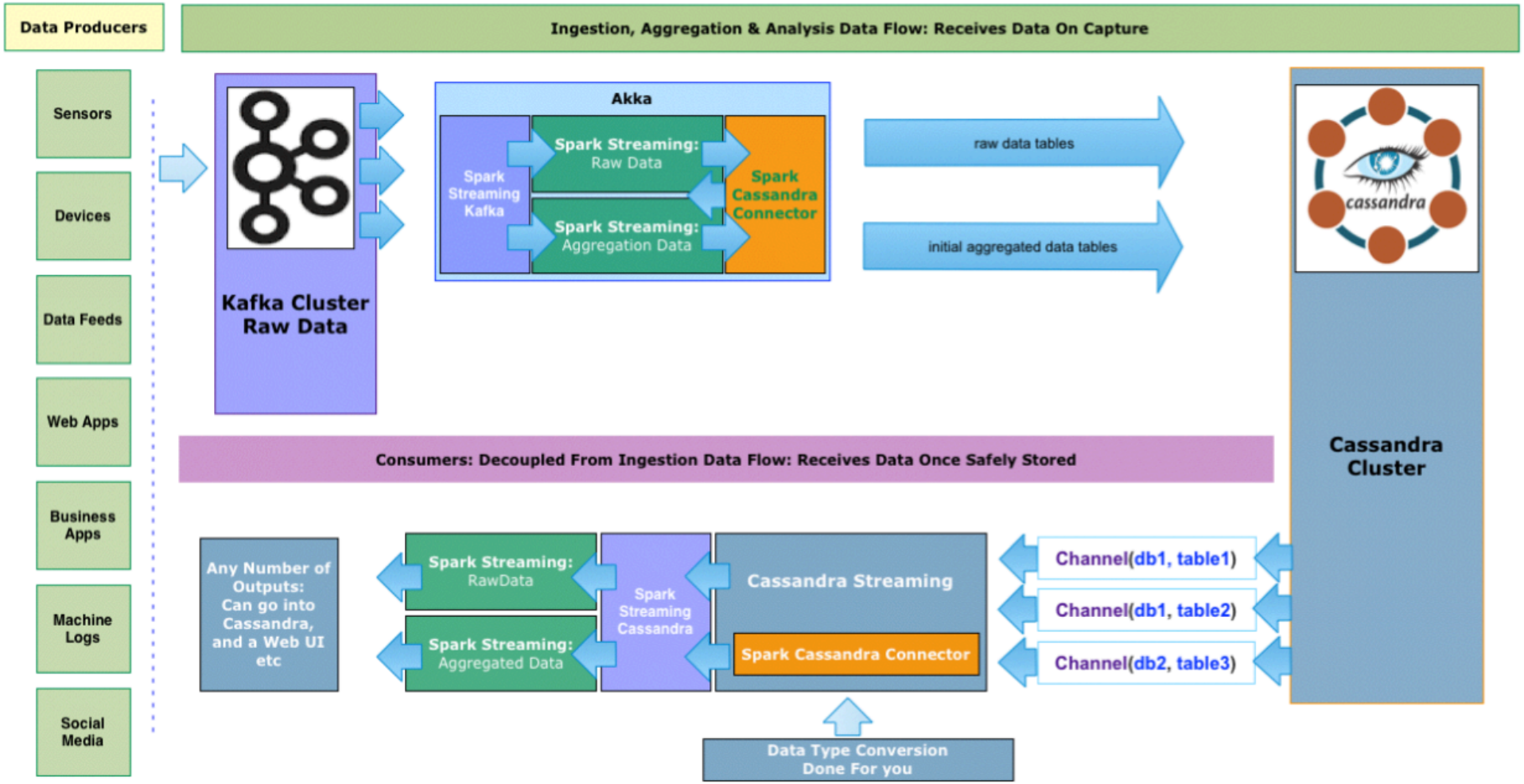
- 消息队列
  - Kafka
- 批处理
  - Hive/Spark
- 流处理
  - Strom/Flink/Spark Streaming
- 压缩
  - Snappy/Lzo
- 存储
  - HDFS/Cassandra/Redis
- 查询服务
  - Hive/Impala/Spark SQL/other…

沃尔玛“lambda”架构





the *open-source* lambda architecture at eClick



回顾一下我们
在工作中遇到了哪些问题?

常见问题“集锦”

- 数据增长迅速，超出预期
- 短时间数据量波动，影响流处理稳定性
- 新增数据时效性问题
- 数据质量问题
- 业务需求增长速度大于资源增长
- 需求响应速度问题
- Count distinct(uv)
- ……

常见解决办法

- 加强计算能力，资源换时间
  - Spark / Flink
- 增加即席查询能力
  - Impala / Presto
- 选择列式存储优化性能
  - Parquet / ORC
- 合理选择压缩,减少存储并提高资源利用率
  - Snappy / Lzo
- 使用Backpressure机制适当削峰
- 终极大法: 加资源

平稳落地

- Spark / Flink
  - Spark 相对 Flink 更成熟，社区也更活跃
  - 详细理解官方文档，善用stackoverflow+jira+cdh doc，spark 95%以上使用/优化都在这4个里面
- Impala / Presto
  - Impala 是 Cloudera 亲儿子，性能表现优异，和 hadoop 体系兼容度更高
  - 主要参考资料为 cdh doc
- Parquet / ORC
  - 没什么太多好说的，有 spark 和 impala，基本就决定用 parquet，一般没太多优化，直接使用即可
- Snappy / Lzo
  - 一般来说，snappy 性价比更高

新增问题1

需要快速随机访问
怎么办?

解决办法

- 增加 HBase 作为辅助存储
  - 备选: Cassandra / MongoDB
- 延伸问题
  - HBase API 比较复杂，成本略高
  - HBase 除 rowkey 访问外性能很差，想提升性能需要增加二级索引，会额外开销存储成本
- 延伸问题解决方案
  - SQL on HBase
  - Phoenix(老牌)，支持二级索引
  - Astro(华为开源)

新增问题2

随机访问条件过多或不明确怎么办?

解决办法

- Plan A
  - Elasticsearch / Solr 替代 HBase
- Plan B
  - Elasticsearch / Solr 作为 HBase 二级索引

新增问题3

历史数据要做更新
怎么办?

解决办法

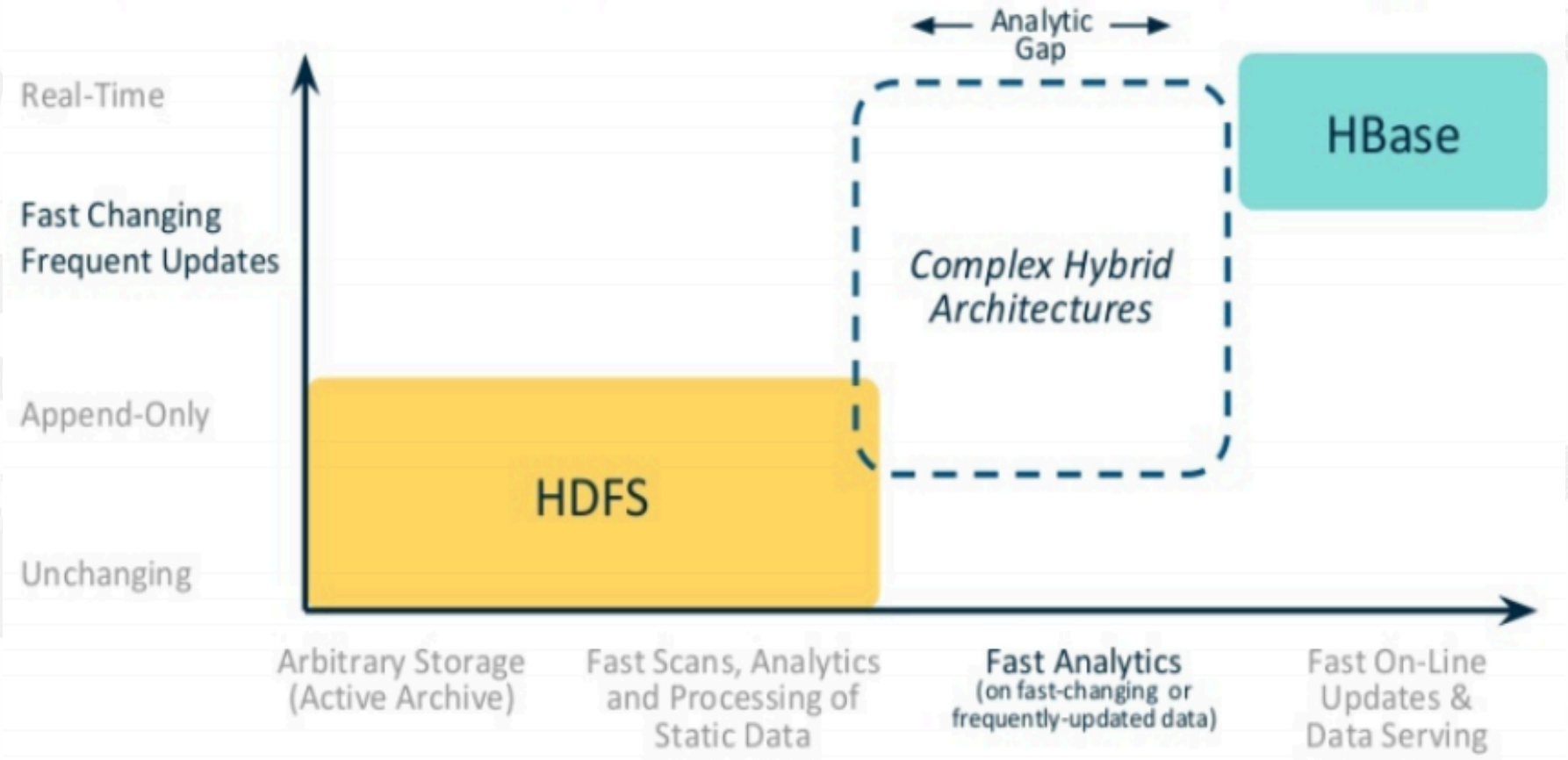
无

Lambda 架构原则：数据不可变性
(data immutability)

阶段性总结

- 采集: flume/sqoop
- 消息队列: kafka
- 批处理: hive/flink/spark core
- 流处理: storm/flink/spark streaming
- 压缩: snappy/lzo
- 列存储: parquet/orc
- 存储: hdfs/hbase/cassandra/redis
- 索引: hbase/cassandra/elasticsearch/solr
- 查询服务: hive/spark/phoenix/es thrift
- 即席查询: impala/es thrift

为什么会这样


Analytic Gap
Real-Time
Fast Changing
Frequent Updates
Append-Only
Unchanging
HBase
Complex Hybrid Architectures
HDFS
Arbitrary Storage (Active Archive)
Fast Scans, Analytics and Processing of Static Data
Fast Analytics
(on fast-changing or frequently-updated data)
Fast On-Line Updates & Data Serving

有什么问题?

方案涉及技术过多

对研发/运维能力要求较高

不能快速的实施部署

问题排查/定位较繁琐

个人推荐

KUDU

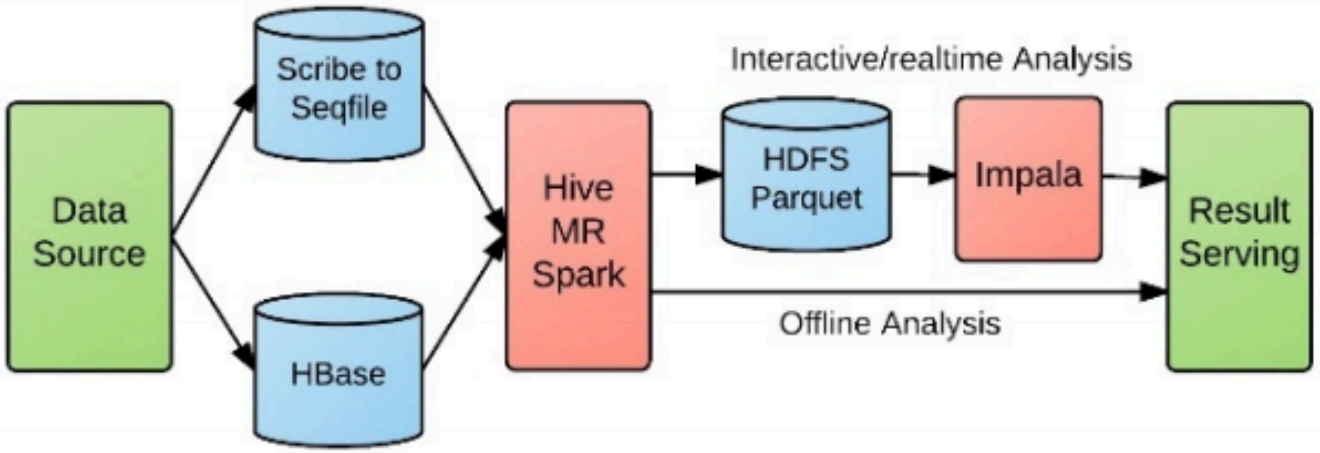
http://kudu.apache.org/

Kudu

- 参考 Google spanner 实现
- scan 和 random access 同时具有高性能表现
- cpu + 磁盘利用率均很高
- 和 parquet 媲美的性能，但支持update
- insert / update 性能优异
- 通过 impala 提供教完善的 SQL 支持
- KV 存储，类似于 HBase
- Hadoop 生态高度集成，但不依赖hadoop生态
- C++语言编写，无 gc 问题，提供 C++和 Java API

Kudu


Fast
hadoop
HDFS
with Parquet
KUDU
APACHE
HBASE
Analytics (Fast Scans)
Slow
Slow
Online (Fast Random Access)
Fast



Long pipeline

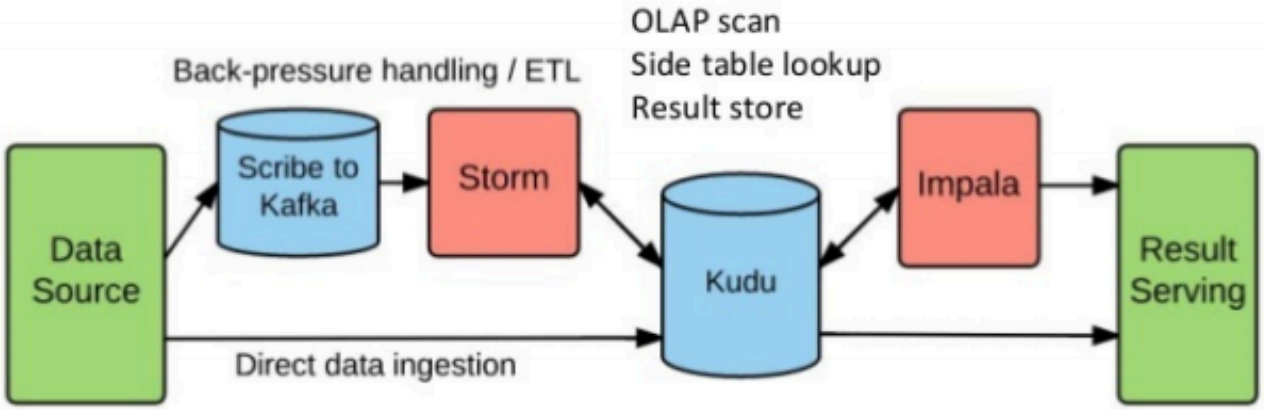
- High data latency (approx 1 hour – 1 day)
- Data conversion pains

No ordering

- Log arrival (storage) order is not exactly logical order
- Must read 2 – 3 days of data to get all of the data points for a single day

⬆ Before kudu

⬇ After kudu



ETL pipeline

- 0 – 10s data latency
- Apps that need to avoid backpressure or need ETL

Direct pipeline (no latency)

- Apps that don't require ETL or backpressure handling

七牛云 QINIU | IT大咖说 知识分享平台

谢 谢

聚美优品
JUMEI.COM