数据仓库

数据仓库（Data Warehouse）,可简写为DW或DWH,数据仓库，是为了企业所有级别的决策制定计划过程，提供所有类型数据类型的战略集合。它出于分析性报告和决策支持的目的而创建。为需要业务智能的企业 ,为需要指导业务流程改进、监视时间，成本，质量以及控制等；

数据仓库能做啥?

年度销售目标的制定，需要根据以往的历史报表进行决策，不能随便制定，优化业务流程。

例如：某电商平台某品牌的手机，在过去5年主要的的购买人群的年龄在什么年龄段，在那个季节购买量人多，这样就可以根据这个特点为目标人群设定他们主要的需求和动态分配产生的生产量，和仓库的库存。

数据仓库的特点

数据仓库是面向主题的

与传统的数据库不一样，数据仓库是面向主题的，那什么是主题呢？首页主题是一个较高乘次的概念，是较高层次上企业信息系统中的数据综合，归类并进行分析的对象。在逻辑意义上，他是对企业中某一个宏观分析领域所涉及的分析对象。（说人话：就是用户用数据仓库进行决策所关心的重点方面，一个主题通常与多个操作信息型系统有关，而操作型数据库的数据组织面向事务处理任务，各个任务之间是相互隔离的）；

数据仓库是集成的

数据仓库的数据是从原来的分散的数据库数据（mysql等关系型数据库）抽取出来的。操作型数据库与DSS（决策支持系统）分析型数据库差别甚大。第一，数据仓库的每一个主题所对应的源数据在所有的各个分散的数据库中，有许多重复和不一样的地方，且来源于不同的联机系统的数据都和不同的应用逻辑捆绑在一起；第二，数据仓库中的综合数据不能从原来有的数据库系统直接得到。因此子在数据进入数据仓库之前，必然要经过统一与综合，这一步是数据仓库建设中最关键，最复杂的一步，所要挖成的工作有： 要统计源数据中所有矛盾之处，如字段的同名异议、异名同义、单位不统一，字长不统一等。 进行数据的综合和计算。数据仓库中的数据综合工作可以在原有数据库抽取数据时生成，但许多是在数据仓库内部生成的，即进入数据仓库以后进行综合生成的

数据仓库的数据是随着时间的变化而变化的

数据仓库中的数据不可更新是针对应用来说的，也就是说，数据仓库的用户进行分析处理是不进行数据更新操作的。但并不是说，在从数据集成输入数据仓库开始到最后被删除的整个生存周期中，所有的数据仓库数据都是永远不变的。

数据仓库的数据是随着时间变化而变化的，这是数据仓库的特征之一。这一特征主要有以下三个表现： 数据仓库随着时间变化不断增加新的数据内容。数据仓库系统必须不断捕捉OLTP数据库中变化的数据，追加到数据仓库当中去，也就是要不断的生成OLTP数据库的快照，经统一集成增加到数据仓库中去；但对于确实不在变化的数据库快照，如果捕捉到新的变化数据，则只生成一个新的数据库快照增加进去，而不会对原有的数据库快照进行修改。

数据库随着时间变化不断删去旧的数据内容 。数据仓库内的数据也有存储期限，一旦过了这一期限，过期数据就要被删除。只是数据库内的数据时限要远远的长于操作型环境中的数据时限。在操作型环境中一般只保存有60~90天的数据，而在数据仓库中则要需要保存较长时限的数据（例如：5~10年），以适应DSS进行趋势分析的要求。

数据仓库中包含有大量的综合数据，这些综合数据中很多跟时间有关，如数据经常按照时间段进行综合，或隔一定的时间片进行抽样等等。这些数据要随着时间的变化不断地进行从新综合。因此数据仓库的数据特征都包含时间项，以标明数据的历史时期。

数据仓库的数据是不可修改的

数据仓库的数据主要提供企业决策分析之用，所涉及的数据操作主要是数据查询，一般情况下并不进行修改操作。数据仓库的数据反映的是一段相当长的时间内历史数据的内容，是不同时点的数据库快照的集合， 以及基于这些快照进行统计、综合和重组的导出数据，而不是联机处理的数据。数据库中进行联机处理的书库进过集成输入到数据仓库中，一旦数据仓库存放的数据已经超过数据仓库的数据存储期限，这些数据将从当前的数据仓库中删去。因为数据仓库只进行数据查询操作，所以数据仓库当中的系统要比数据库中的系统要简单的多。数据库管理系统中许多技术难点，如完整性保护、并发控制等等，在数据仓库的管理中几乎可以省去。但是由于数据仓库的查询数据量往往很大，所以就对数据查询提出了更高的要求，他要求采用各种复杂的索引技术；同时数据仓库面向的是商业企业的高层管理层，他们会对数据查询的界面友好性和数据表示提出更高的要求；

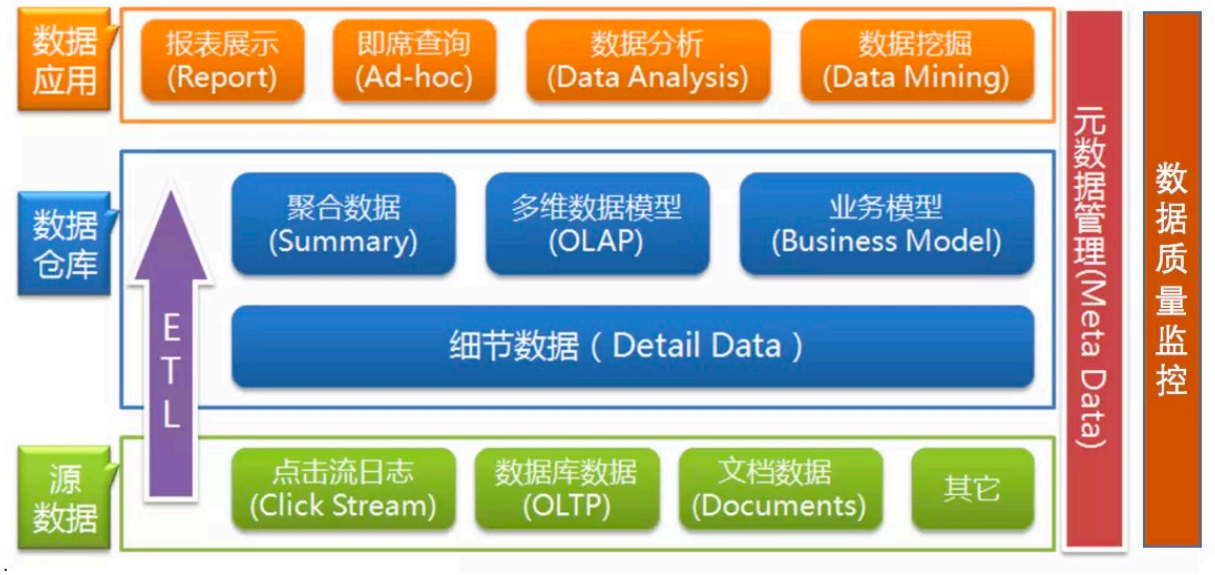
数据库

https://my.oschina.net/hblt147/blog/3017817

实时数仓建设

随着近些年大数据相关技术的飞速发展，数据的在助力业务的发展方面越发重要；数据仓库为企业的决策提供所有数据类型支持的战略集合。它是单个数据存储，出于分析性报告和决策支持目的而创建。 为需要业务智能的企业，提供指导业务流程改进、监视时间、成本、质量以及控制。基于hive的离线数据仓库模型已经发展了很多年，市面的优秀参考实践已比较多，但是数据本身的价值随着时间的流逝逐步减弱，因此在数据发生后必须尽快的达到用户的手中。实时数仓的需求构建的需求也应用而生。

数据仓库模型架构



实时数仓

- 数据仓库的建设主要包括数据的采集、数据的处理、数据归档、数据应用四个方面。
- 当前主要的应用场景包括报表展示、即席查询、BI展示、数据分析、数据挖掘、模型训练等方面。
- 数据仓库的建设是面向主题的、集成性的、不可更新的、时许变化的。

实时VS离线

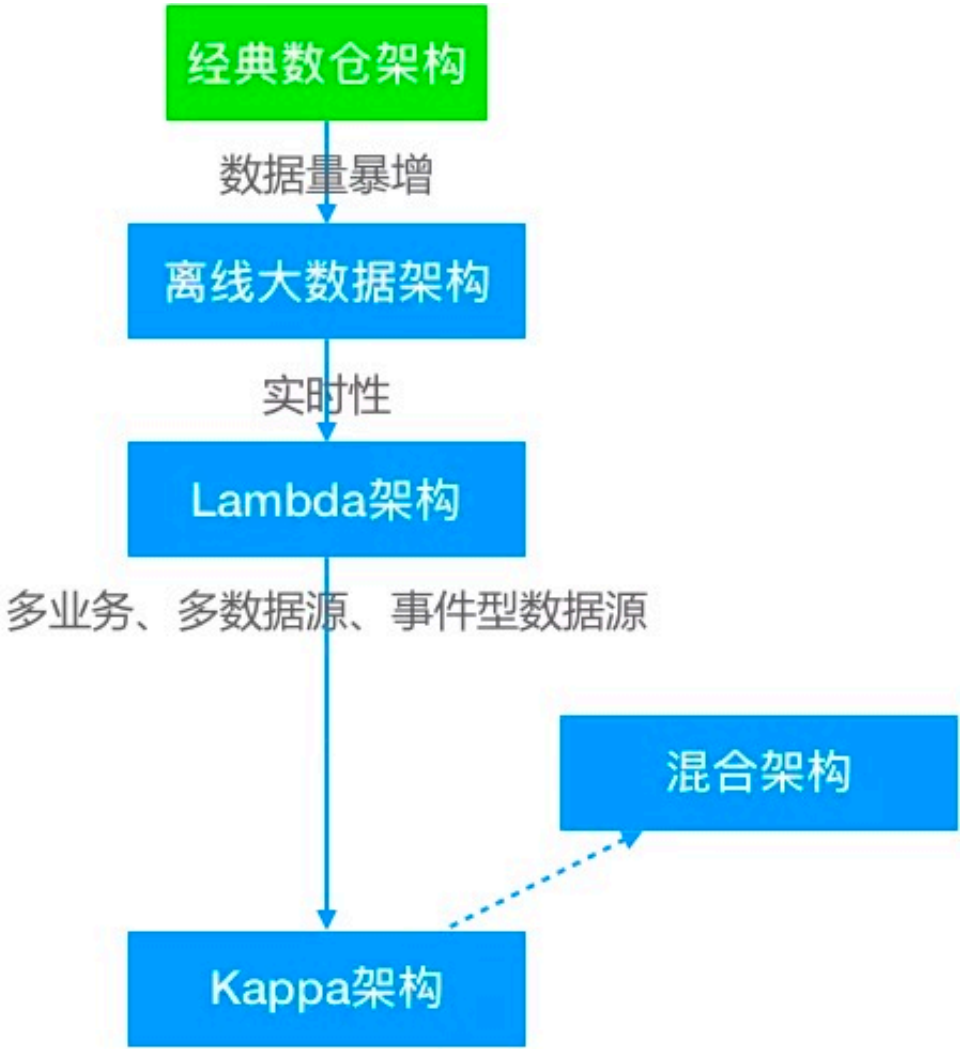
离线数据仓库主要基于sqoop、hive等技术来构建T+1的离线数据，通过定时任务每天拉取增量数据导入到hive表中，然后创建各个业务相关的主题维度数据，对外提供T+1的数据查询接口。 实时数仓当前主要是基于数据采集工具，如canal等将原始数据写入到Kafka这样的数据通道中，最后一般都是写入到类似于HBase这样存储系统中，对外提供分钟级别、甚至秒级别的查询方案。

实时数仓的的实施关键点

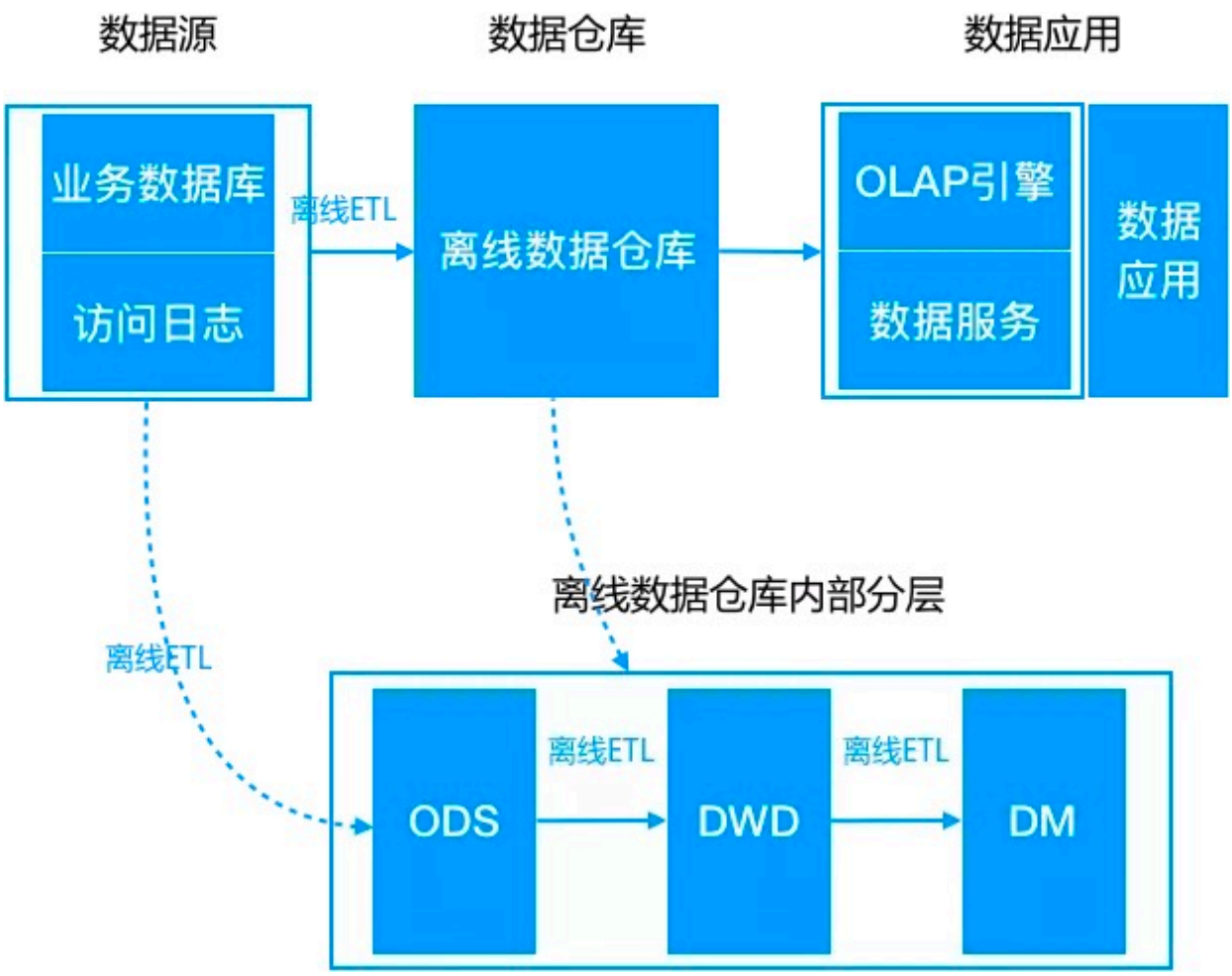
- 端到端数据延迟、数据流量的监控
- 故障的快速恢复能力
- 数据的回溯处理，系统支持消费指定时间端内的数据
- 实时数据从实时数仓中查询，T+1数据借助离线通道修正
- 数据地图、数据血缘关系的梳理
- 业务数据质量的实时监控，初期可以根据规则的方式来识别质量状况


数据源
数据存储
数据应用
原始状态
ERP、CRM、SCM
离线ETL
数据库：
SQLServer、Oracle、Greenplum等
报表
OLAP
计算&存储大量异构数据
构建：大数据离线计算
存储：NoSQL存储
ERP、CRM、SCM
Log、IoT设备、埋点
离线ETL
数据库：SQLServer、Oracle、Greenplum等
NoSQL：Hadoop、Hive、Hbase、Redis等；
OLAP：Presto、Impala、Kylin等；
搜索引擎：ES
报表
OLAP
实时产生结果
构建：数据事件化、实时ETL
存储：OLAP引擎
应用：OLAP、自动决策
ERP、CRM、SCM
Log、IoT设备、埋点
数据事件化
实时ETL
数据库：SQLServer、Oracle、Greenplum等
NoSQL：Hadoop、Hive、Hbase、Redis等
OLAP：Presto、Impala、Kylin等；
搜索引擎：ES
报表
OLAP
自动决策

数据仓库架构的演变


经典数仓架构
数据量暴增
离线大数据架构
实时性
Lambda架构
多业务、多数据源、事件型数据源
混合架构
Kappa架构

离线大数据架构

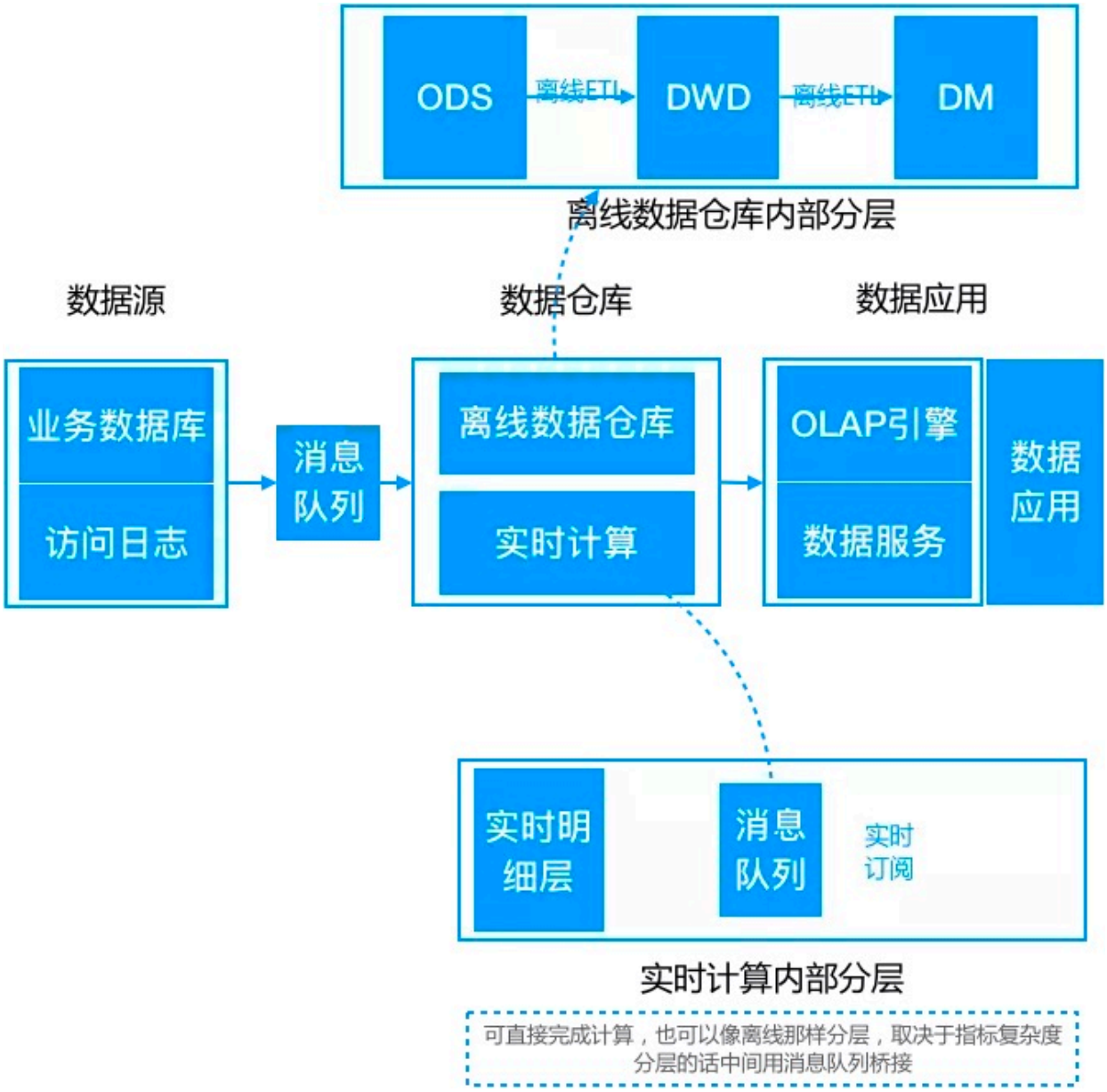


Lambda架构

Lambda架构问题：

1.同样的需求需要开发两套一样的代码 这是Lambda架构最大的问题，两套代码不仅仅意味着开发困难（同样的需求，一个在批处理引擎上实现，一个在流处理引擎上实现，还要分别构造数据测试保证两者结果一致），后期维护更加困难，比如需求变更后需要分别更改两套代码，独立测试结果，且两个作业需要同步上线。

2.资源占用增多：同样的逻辑计算两次，整体资源占用会增多（多出实时计算这部分）



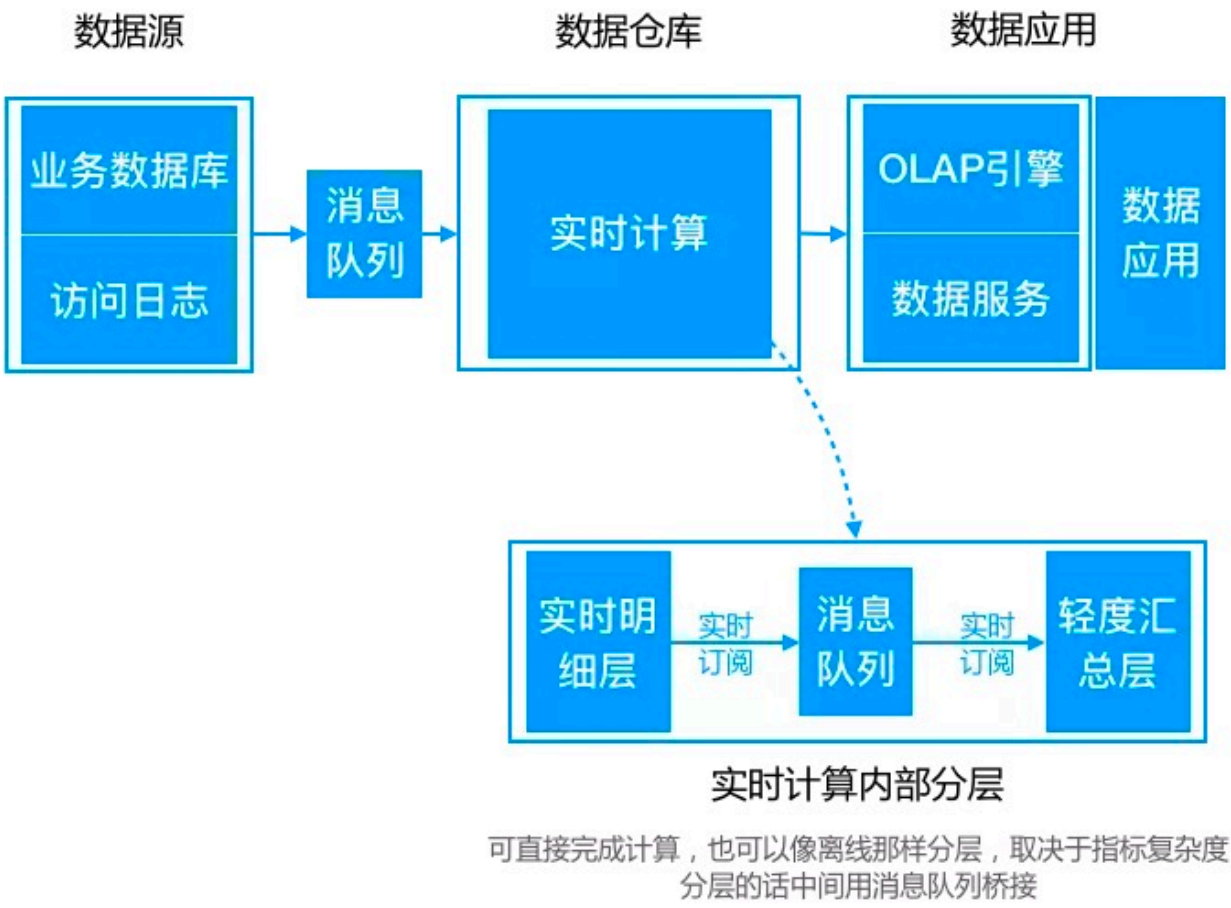
Kappa架构

Lambda架构虽然满足了实时的需求，但带来了更多的开发与运维工作，其架构背景是流处理引擎还不完善，流处理的结果只作为临时的、近似的值提供参考。后来随着Flink等流处理引擎的出现，流处理技术很成熟了，这时为了解决两套代码的问题，LickedIn 的Jay Kreps提出了Kappa架构

Kappa架构可以认为是Lambda架构的简化版（只要移除lambda架构中的批处理部分即可）。

在Kappa架构中，需求修改或历史数据重新处理都通过上游重放完成。

Kappa架构最大的问题是流式重新处理历史的吞吐能力会低于批处理，但这个可以通过增加计算资源来弥补。



Kappa架构的重新处理过程

重新处理是人们对Kappa架构最担心的点，但实际上并不复杂：

1.选择一个具有重放功能的、能够保存历史数据并支持多消费者的消息队列，根据需求设置历史数据保存的时长，比如Kafka，可以保存全部历史数据。

2.当某个或某些指标有重新处理的需求时，按照新逻辑写一个新作业，然后从上游消息队列的最开始重新消费，把结果写到一个新的下游表中。

3.当新作业赶上进度后，应用切换数据源，读取2中产生的新结果表。

4.停止老的作业，删除老的结果表。

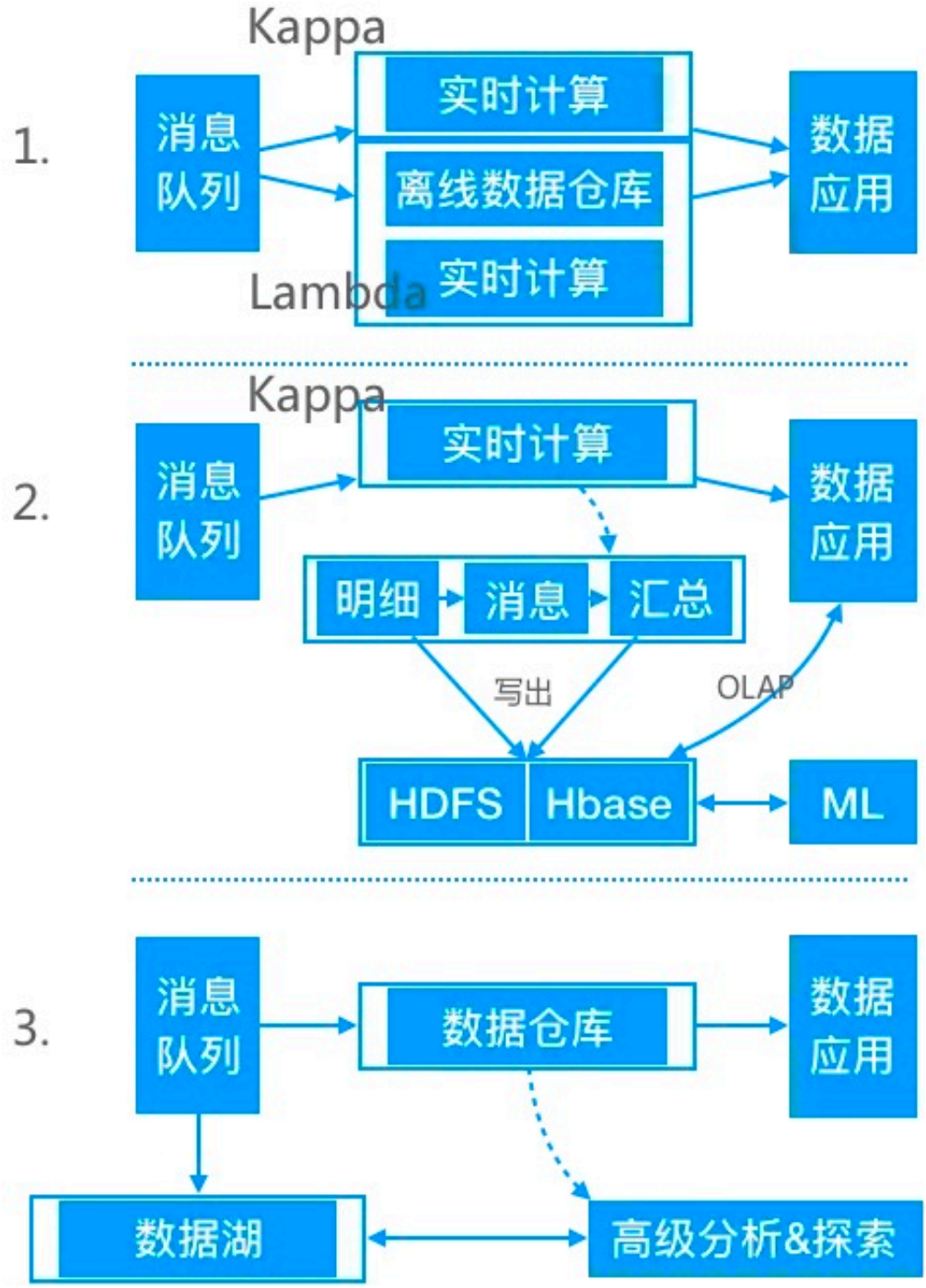
4.4 Lambda架构与Kappa架构的对比

对比项	Lambda架构	Kappa架构
实时性	实时	实时
计算资源	批和流同时运行，资源开销大	只有流处理，仅针对新需求开发阶段运行两个作业，资源开销小
重新计算时吞吐	批式全量处理，吞吐较高	流式全量处理，吞吐较批处理低
开发、测试	每个需求都需要两套不同代码，开发、测试、上线难度较大	只需实现一套代码，开发、测试、上线难度相对较小
运维成本	维护两套系统（引擎），运维成本大	只需维护一套系统（引擎），运维成本小

在真实的场景中，很多时候并不是完全规范的Lambda架构或Kappa架构，可以是两者的混合，比如大部分实时指标使用Kappa架构完成计算，少量关键指标（比如金额相关）使用Lambda架构用批处理重新计算，增加一次校对过程。（1）

Kappa架构并不是中间结果完全不落地，现在很多大数据系统都需要支持机器学习（离线训练），所以实时中间结果需要落地对应的存储引擎供机器学习使用，另外有时候还需要对明细数据查询，这种场景也需要把实时明细层写出到对应的引擎中。（2）参考后面的案例

另外，随着数据多样性的发展，数据仓库这种提前规定schema的模式显得越来难以支持灵活的探索&分析需求，这时候便出现了一种数据湖技术，即把原始数据全部缓存到某个大数据存储上，后续分析时再根据需求去解析原始数据。简单的说，数据仓库模式是schema on write，数据湖模式是schema on read。（3）



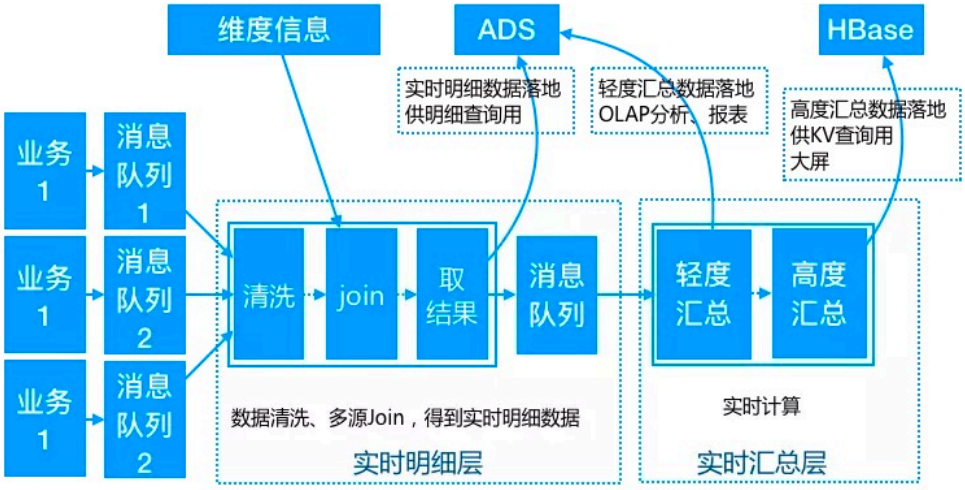
实时数仓案例

整体设计

整体设计如下图，基于业务系统的数据，数据模型采用中间层的设计理念，建设仓配实时数仓；计算引擎，选择更易用、性能表现更佳的实时计算作为主要的计算引擎；数据服务，选择天工数据服务中间件，避免直连数据库，且基于天工可以做到主备链路灵活配置秒级切换；数据应用，围绕大促全链路，从活动计划、活动备货、活动直播、活动售后、活动复盘五个维度，建设仓配大促数据体系。

数据模型

不管是从计算成本，还是从易用性，还是从复用性，还是从一致性……，我们都必须避免烟囱式的开发模式，而是以中间层的方式建设仓配实时数仓。与离线中间层基本一致，我们将实时中间层分为两层。



第一层DWD公共实时明细层

实时计算订阅业务数据消息队列，然后通过数据清洗、多数据源join、流式数据与离线维度信息等的组合，将一些相同粒度的业务系统、维表中的维度属性全部关联到一起，增加数据易用性和复用性，得到最终的实时明细数据。这部分数据有两个分支，一部分直接落地到ADS，供实时明细查询使用，一部分再发送到消息队列中，供下层计算使用；

第二层DWS公共实时汇总层

以数据域+业务域的理念建设公共汇总层，与离线数仓不同的是，这里汇总层分为轻度汇总层和高度汇总层，并同时产出，轻度汇总层写入ADS，用于前端产品复杂的olap查询场景，满足自助分析和产出报表的需求；高度汇总层写入Hbase，用于前端比较简单的kv查询场景，提升查询性能，比如实时大屏等；

注：

1.ADS是一款提供OLAP分析服务的引擎。开源提供类似功能的有，Elastic Search、Kylin、Druid等；

2.案例中选择把数据写入到Hbase供KV查询，也可根据情况选择其他引擎，比如数据量不多，查询压力也不大的话，可以用mysql

3.因主题建模与业务关系较大，这里不做描述

数据保障

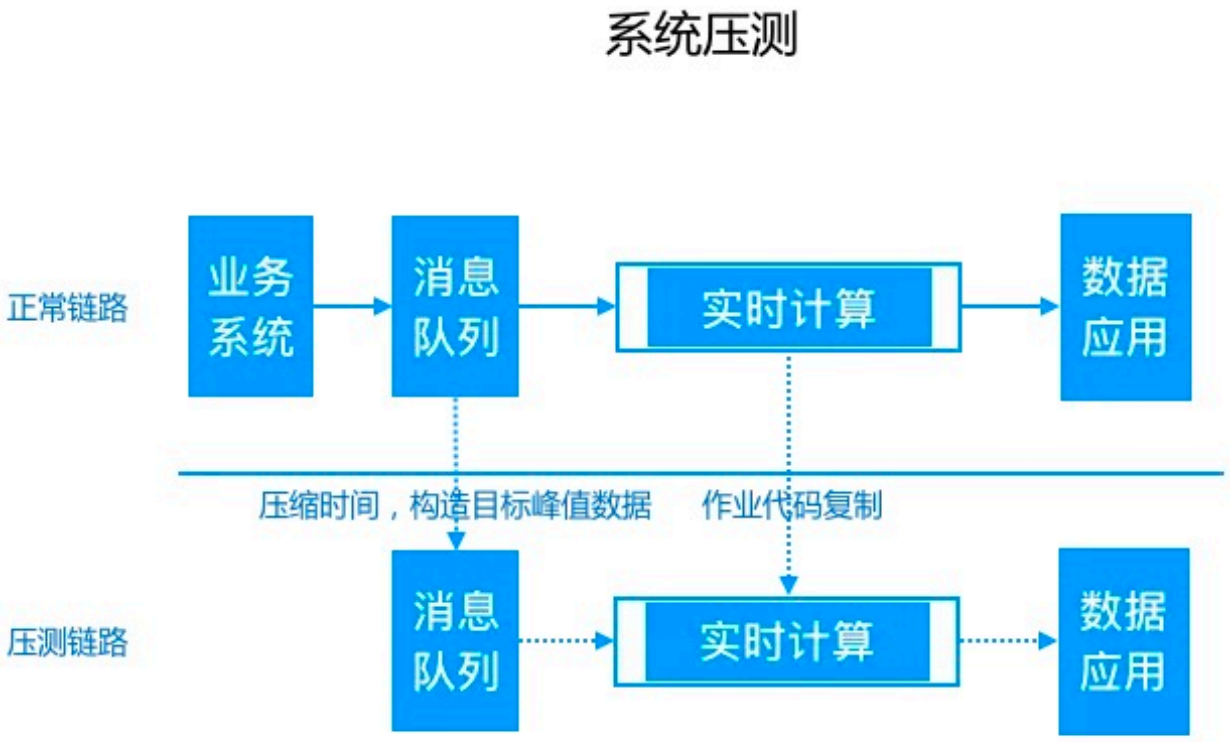
集团每年都有双十一等大促，大促期间流量与数据量都会暴增。

实时系统要保证实时性，相对离线系统对数据量要更敏感，对稳定性要求更高。

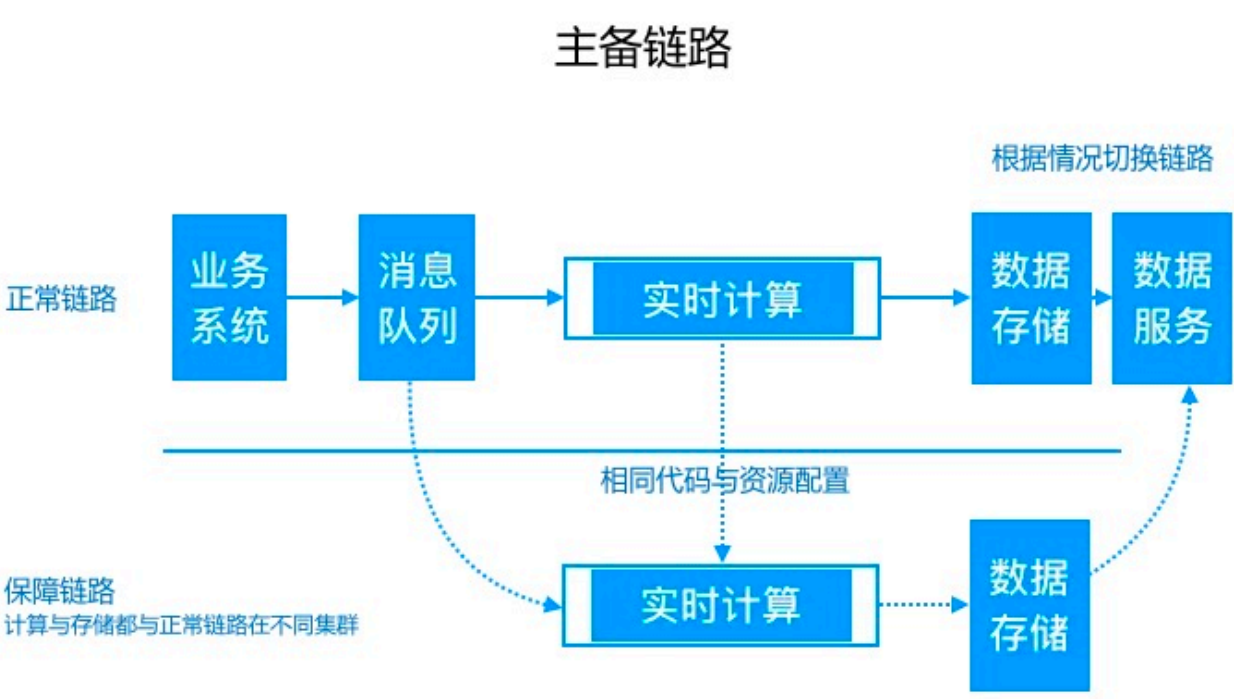
所以为了应对这种场景，还需要在这种场景下做两种准备：

- 大促前的系统压测；

- 大促中的主备链路保障；



压测的主要目的是产出实时计算在大促过程所需资源及其配置



主备链路保障的目的是在主链出现问题能通过备链提供服务，可以只针对高优先级的作业做主备链路，并且不限于一条备链。