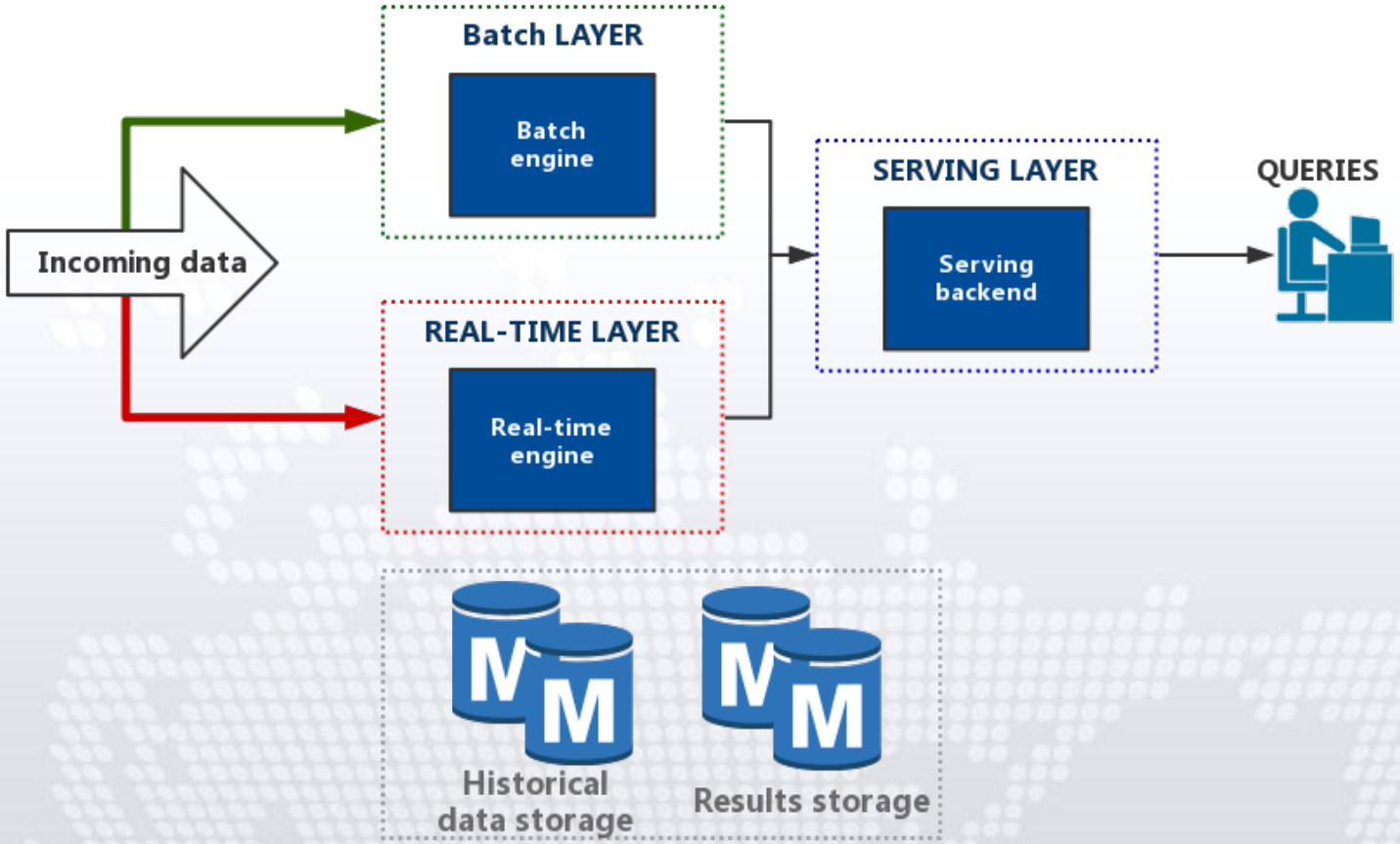
2019

企业级数据仓库实战

Lambda架构

Lambda架构图



主要特点：

主要分为批处理层、实时流层、服务层，其中批处理层主要进行离线处理，实时流层主要进行实时数据处理，服务层合并两者结果数据统一对外提供服务

优点

1. 稳定

目前有相当一部分的数据仓库的架构采取这样的方式，原因在于批处理和实时处理是分开的，分别处理不同频率的数据，双链路也保证了整个数据的稳定性

2. 技术实现难度低、开发成本小

这类型的架构比较适合进行需求的版本迭代，先离线满足需求，再通过实时链路完善数据的实时性，两者可以分版本上线。在实际的生产环境中，一般只会在部分场景使用实时流技术进行开发，开发成本相对比较低

3. 数据回溯相对比较容易

当上游数据出现异常时，可以通过批处理进行数据的快速恢复，并覆盖错误的数据

缺点

1. 统计口径不一致问题

由于两条链路本身是独立的，而且多半是由不同的人分阶段进行开发，难免会造成统计口径不一致。我们在前面的PPT中提到，在构建事实表时需要遵从一个规则，多种数据在进行合并时，必须遵从一致性原则。如果两者统计口径不一致，就会导致整个数据被污染，变得不可用，甚至影响到业务决策

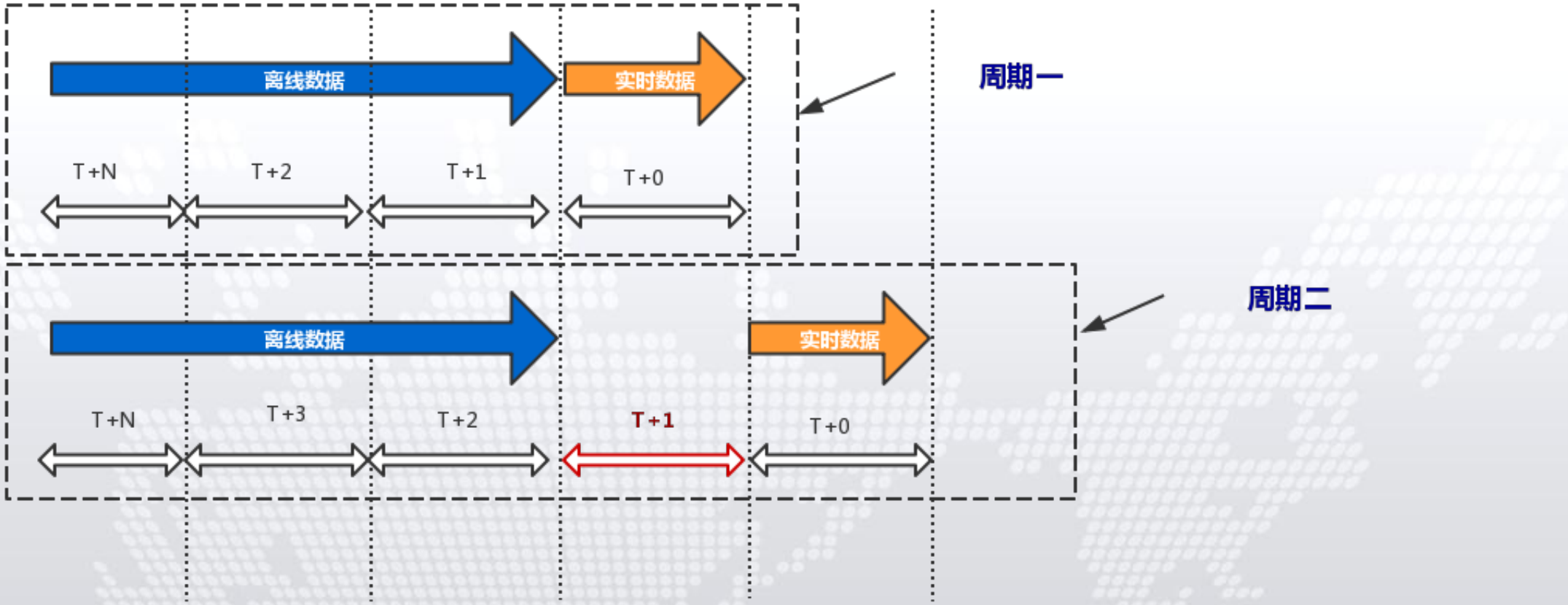
2. 代码冗余

同样的代码逻辑，在批处理链路实现一次，在流处理上再次处理一次，从代码管理上，完全是多余的

3. 零点切换问题

一般情况下，离线负责历史数据统计，实时负责当日数据的统计。在零点时，问题出现了，离线链路一般是T+1，在零点附近的几个小时内，离线链路还没有处理完数据，这个时候会出现在应用层无法获取到昨日的数据，只有前天和今天的数据。

零点切换问题



一般处理这种情况，可以取最近今日和昨日的实时数据+昨日之前的离线数据构成全量数据，即

方案一：

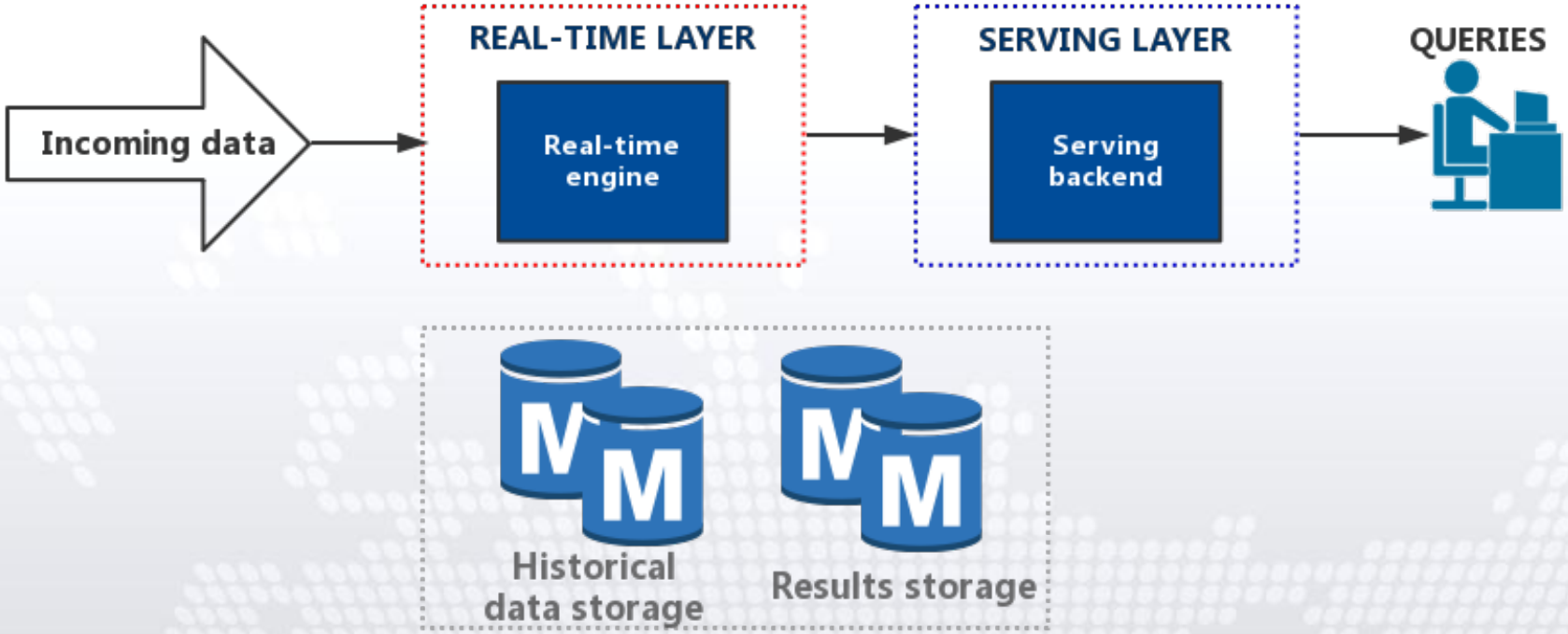
全量数据=今日实时数据+昨日和昨日之前离线数据

方案二：

全量数据=今日实时数据+昨日实时数据+昨日之前离线数据

Kappa架构

Kappa架构



主要特点：

只使用实时数据作为数据处理层，统一离线和实时链路

优点

1. 整体数据链路清晰
   由于去掉了批处理层，基本只保留实时层，架构简单清晰

2. 规避统计口径不一致问题
   因为只有一条单一的链路，也就不存在统计口径不一致问题

3. 无零点问题
   应用层不需要进行数据merge，零点可以平滑切换

缺点

1. 回溯数据成本比较大

回溯数据必须要重新进行消息的消费，且需要清理历史数据，想比较批处理而言，成本会比较高

2. 开发周期比较长

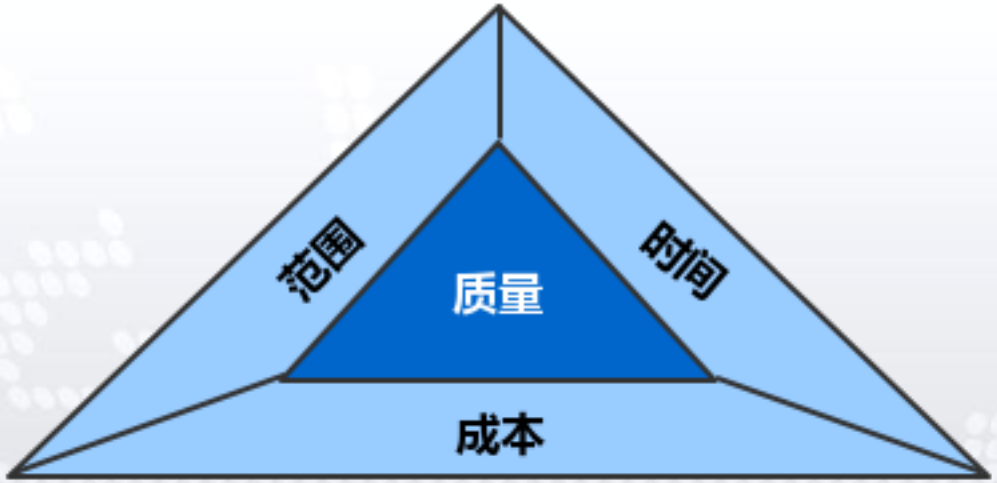
相比较批处理而言，流处理目前的实现语言主要以Scala和Java为主，一般开发周期要长于使用如Hive SQL 方式进行开发，不过目前很多的流处理框架也支持SQL，但是部分函数还支持的不是很好

3. 机器成本问题

一般流处理数据所耗费的机器成本会更高一些，会大量用到一些非结构化存储，且任务常驻内存

混合架构

混合架构即是讲这两种架构结合起来使用，针对不同的场景使用不同的架构
目前实际的工作中，两种架构都会用到，那么两种架构到底孰优孰劣，还是要根据实际情况进行选择。



项目三角形：范围、时间、成本、质量，彼此之间寻求一个平衡

如果你的数据本身不太要求实时性，那么单独的批处理链路即可，如果要求一定的实时性，就可以采用Lambda架构，但是团队成员需要有一定的实时流处理开发能力。如果业务对实时性要求非常高，那么这个时候应该讲流处理的能力摆在第一位，批处理作为一个补充。所有这些，都是为了保证最终的数据产出的质量。

THANK YOU FOR YOUR GUIDANCE.

谢谢