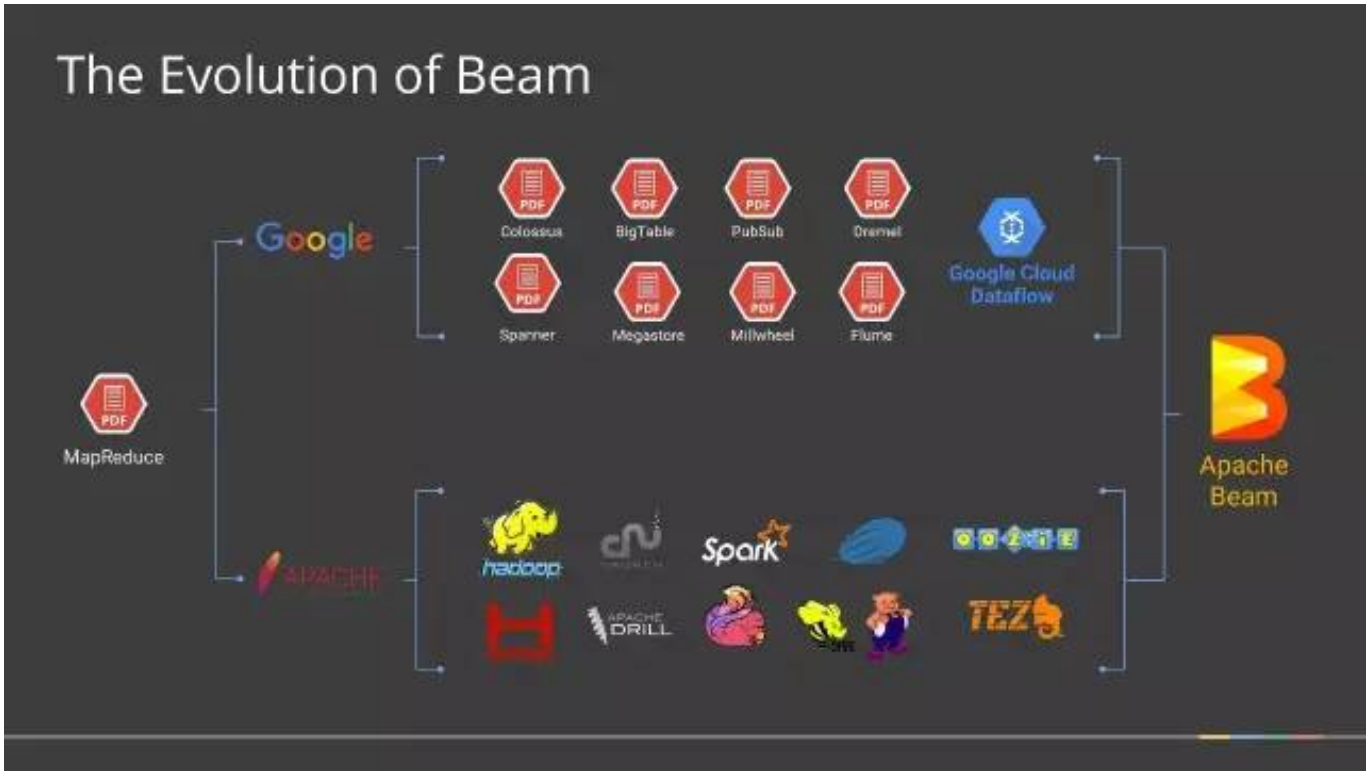
为什么Google用Apache Beam彻底替换掉MapReduce

1月10日，Apache软件基金会宣布，Apache Beam成功孵化，成为该基金会的一个新的顶级项目，基于Apache V2许可证开源。

2003年，谷歌发布了著名的大数据三篇论文，史称三驾马车：Google FS、MapReduce、Big Table。虽然谷歌没有公布这三个产品的源码，但是她这三个产品的详细设计论文开启了全球的大数据时代！从Doug Cutting大神根据谷歌的论文实现出Hadoop+MapReduce的雏形，到Hadoop生态圈各种衍生产品的蓬勃发展，再到后来的Spark、流式计算等等，所有的一切都要归功于、源自这三篇论文。可惜谷歌虽然开启了这个伟大的时代，却始终仅仅满足于偶尔发表一两篇论文以强调自己在理论和工程上的领导地位，从来没有亲身参与进来，尤其是没有为开源生态做出什么贡献，因而一直没有从大数据市场获得什么实在的好处。

痛定思痛，谷歌开始走开源之路，将自己的标准推广给社区。从众所周知的Kubernetes，到2016年2月谷歌高调宣布将Apache Beam（原名Google DataFlow）贡献给Apache基金会孵化，再到最近大热的Tensorflow等等，动作不断。Apache Beam被认为是继MapReduce，GFS和BigQuery等之后，谷歌在大数据处理领域对开源社区的又一个非常大的贡献。

也就是说，在大数据处理的世界里，谷歌一直在内部闭源，开发并使用着BigTable、Spanner、Millwheel等让大家久闻大名而又无缘一见的产品，开源世界演进出了Hadoop、Spark、Apache Flink等产品，现在他们终于殊途同归，走到一起来了。



如果想及时了

解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

为什么要推出开源的Apache Beam？

Apache Beam的主要负责人Tyler Akidau在他的博客中提到他们做这件事的理念是：

> 要为这个世界贡献一个容易使用而又强大的模型，用于大数据的并行处理，同时适用于流式处理和批量处理，而且在各种不同平台上还可以移植。

那这一次为什么不是又酷酷的发表一篇论文，然后退居一旁静静的观察呢？为什么要联合一众伙伴为大家直接提供可以运行的代码了呢？原因主要有两点：

1、尽管在过去谷歌一直是闭源的，但在为云客户服务的过程中，谷歌已经认识到了开源软件的的巨大价值，比如基于谷歌三篇论文产生的Hadoop社区就是一个非常好的例子。思想上的转变使Apache Beam的诞生成为可能；
2、就Beam这个项目而言，要成功的必要条件之一是，必须有已经开源的Runner为Beam模型提供充分的支持，这样它才会在自建云和非谷歌云的场景下成为一个非常有竞争力的备选方案。去年Apache Flink在他们的系统内采用了Beam模型，这一条件也得到了满足；

无利不起早，谷歌这样做也是有着直接商业动机的，就是希望能有尽可能多的Apache Beam数据处理流水线可以运行在谷歌的Cloud Dataflow上，别忘了这是Apache Beam的原型。进一步说，采用开源的方式来引导这件事，也是有许多直接好处的：

1、支持Apache Beam的Runner越多，它作为一个平台的吸引力就越大；
2、使用Apache Beam的用户越多，想在谷歌云平台上运行Apache Beam的用户也就越多；
3、开发Apache Beam过程中吸引到的伙伴越多，那对这样的数据处理模型的推广就越有利；

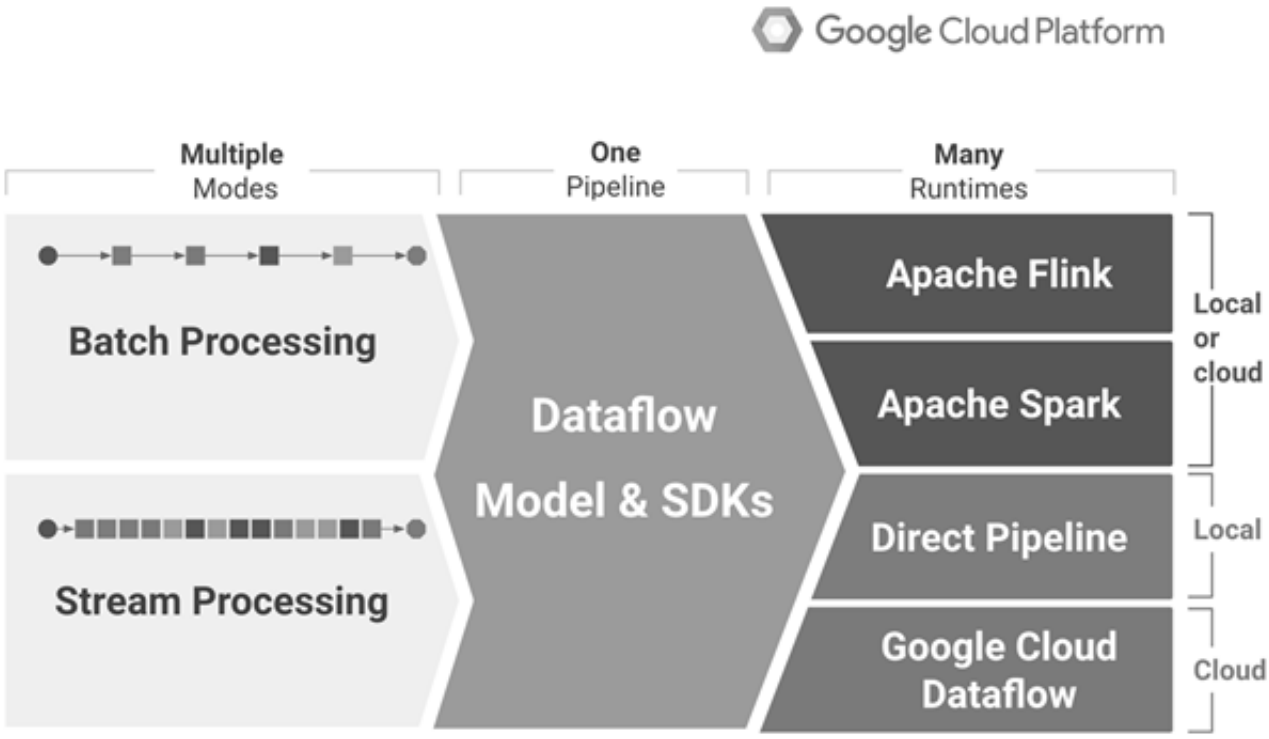
而且，好处也不会全都归于谷歌，Apache Beam项目中的所有参与方都会受益。如果在构建数据处理流水线时存在着这样一个可移植的抽象层，那就会更容易出现新的Runner，它们可以专注于技术创新，提供更高的性能、更好的可靠性、更方便的运维管理等。换句话说，消除了对API的锁定，就解放了处理引擎，会导致更多产品之间的竞争，从而最终对整个行业起到良性的促进作用。

谷歌坚信Apache Beam就是数据批量处理和流式处理的未来。这么做会为各种不同的Runner营造一个健康的生态系统，让它们之间相互竞争，而最后可以让用户得到实在的好处。

Apache Beam是什么

要说Apache Beam，先要说说谷歌Cloud Dataflow。Dataflow是一种原生的谷歌云数据处理服务，是一种构建、管理和优化复杂数据流水线的方法，用于构建移动应用、调试、追踪和监控产品级云应用。它采用了谷歌内部的技术Flume和MillWhell，其中Flume用于数据的高效并行化处理，而MillWhell则用于互联网级别的带有很好容错机制的流处理。该技术提供了简单的编程模

型，可用于批处理和流式数据的处理任务。她提供的数据流管理服务可控制数据处理作业的执行，数据处理作业可使用DataFlow SDK创建。



如果想及时了解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

Apache Beam本身不是一个流式处理平台，而是一个统一的编程框架，它提供了开源的、统一的编程模型，帮助你创建自己的数据处理流水线，实现可以运行在任意执行引擎之上批处理和流式处理任务。Beam对流式计算场景中的所有问题重新做了一次归纳，然后针对这些问题提出了几种不同的解决模型，然后再把这些模型通过一种统一的语言给实现出来，最终这些Beam程序可以运行在任何一个计算平台上（只要相应平台——即Runner实现了对Beam的支持）。它的特点有：

1、统一的：对于批处理和流式处理，使用单一的编程模型；
2、可移植的：可以支持多种执行环境，包括Apache Apex、Apache Flink、Apache Spark和谷歌Cloud Dataflow等；
3、可扩展的：可以实现和分享更多的新SDK、IO连接器、转换操作库等；

Beam特别适合应用于并行数据处理任务，只要可以将要处理的数据集分解成许多相互独立而又可以并行处理的小集合就可以了。Beam也可以用于ETL任务，或者单纯的数据整合。这些任务主要就是把数据在不同的存储介质或者数据仓库之间移动，将数据转换成希望的格式，或者将数据导入一个新系统。

Beam主要包含两个关键的部分：

Beam SDK

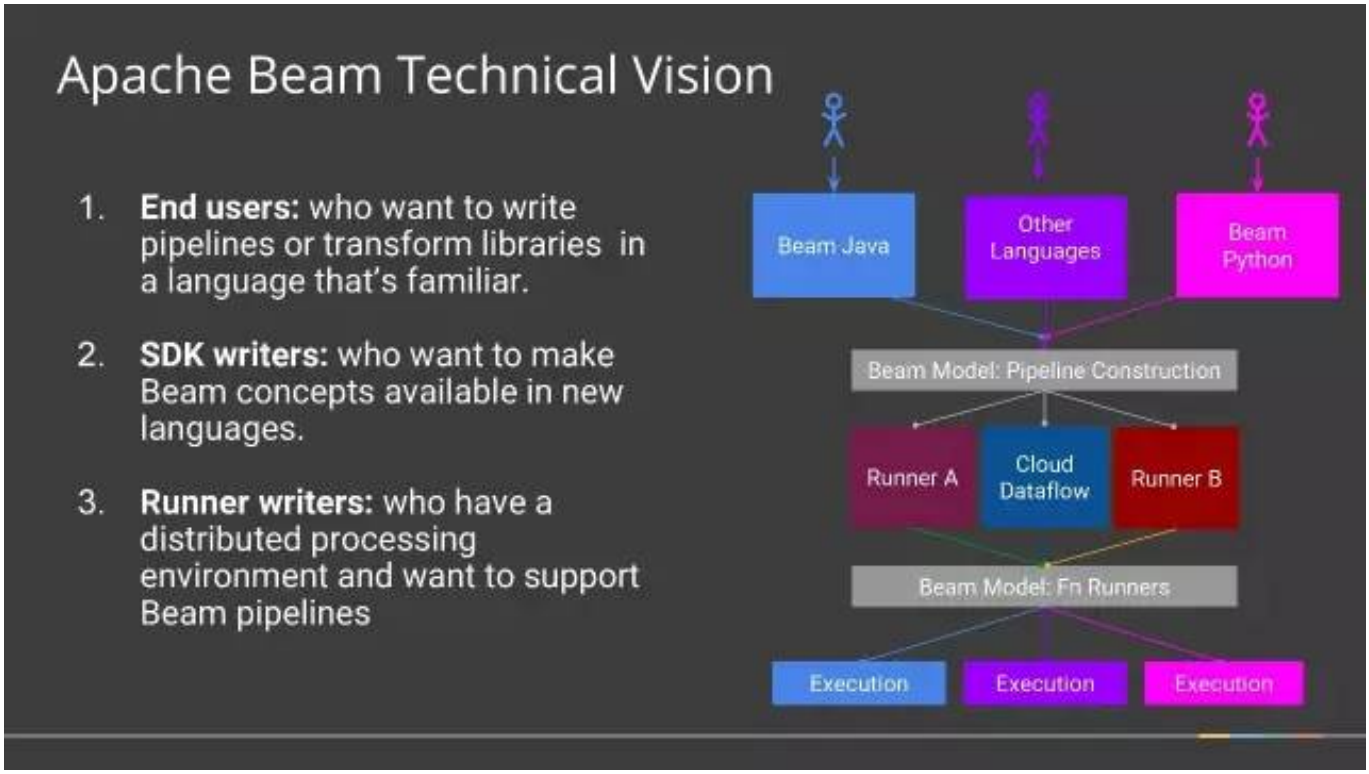
Beam SDK提供一个统一的编程接口给到上层应用的开发者，开发者不需要了解底层的具体的大数据平台的开发接口是什么，直接通过Beam SDK的接口，就可以开发数据处理的加工流程，不管输入是用于批处理的有限数据集，还是流式的无限数据集。对于有限或无限的输入数据，Beam SDK都使用相同的类来表现，并且使用相同的转换操作进行处理。Beam SDK可以有不同编程语言的实现，目前已经完整地提供了Java，python的SDK还在开发过程中，相信未来会有更多不同的语言的SDK会发布出来。

Beam Pipeline Runner

Beam Pipeline Runner将用户用Beam模型定义开发的处理流程翻译成底层的分布式数据处理平台支持的运行时环境。在运行Beam程序时，需要指明底层的正确Runner类型。针对不同的大数据平台，会有不同的Runner。目前Flink、Spark、Apex以及谷歌的Cloud DataFlow都有支持Beam的Runner。

需要注意的是，虽然Apache Beam社区非常希望所有的Beam执行引擎都能够支持Beam SDK定义的功能全集，但是在实际实现中可能并不一定。例如，基于MapReduce的Runner显然很难实现和流处理相关的功能特性。就目前状态而言，对Beam模型支持最好的就是运行于谷歌云平台之上的Cloud Dataflow，以及可以用于自建或部署在非谷歌云之上的Apache Flink。当然，其它的Runner也正在迎头赶上，整个行业也在朝着支持Beam模型的方向发展。

那大家可以怎样与Beam做亲密接触呢？



如果想及时了解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

如上图所示，主要有三个方面：

1、数据处理：直接使用已有的自己熟悉语言的SDK，根据Beam模型去定义并实现自己的数据处理流程；
2、SDK实现：用新的编程语言去根据Beam概念实现SDK，这样大家以后在编程语言方面就可以有更多选择了；
3、Runner实现：将已有的分布式数据处理平台作为一种新的Runner，接入Beam模型；

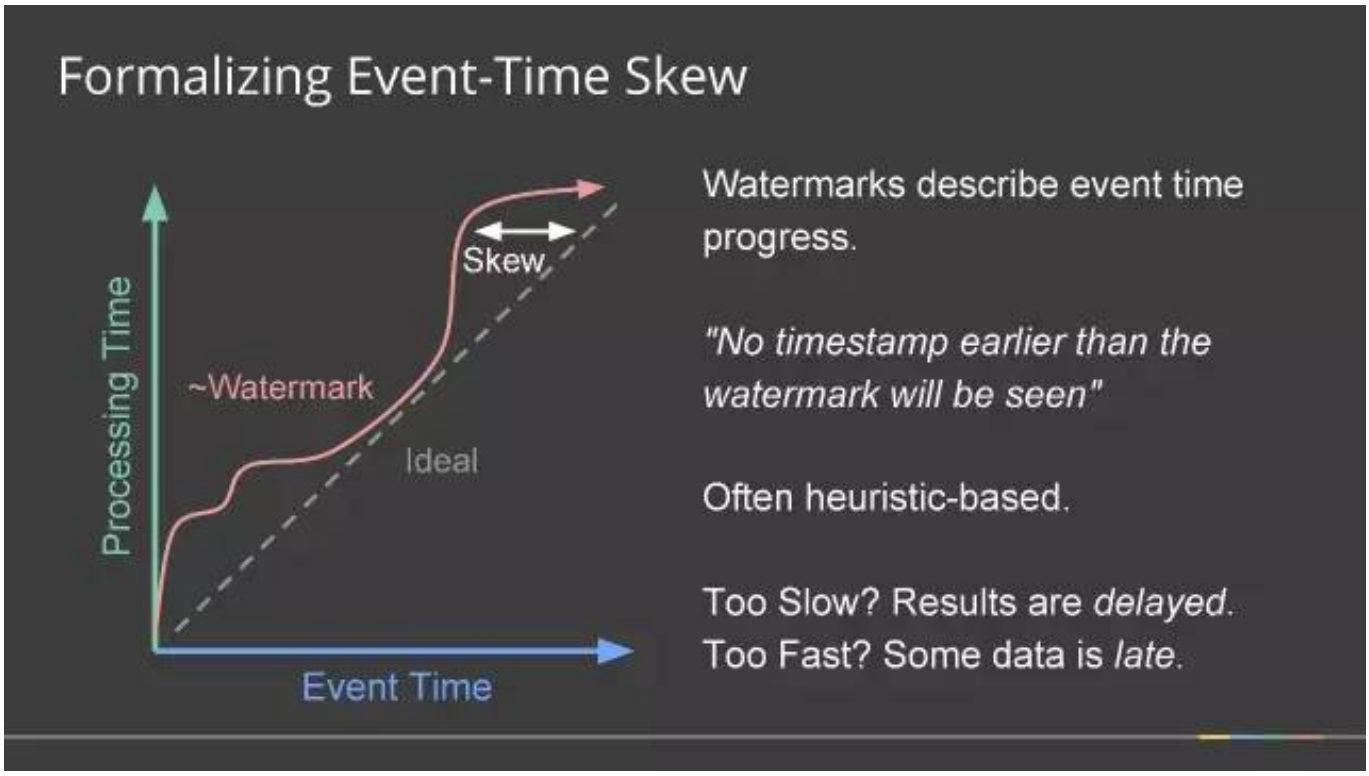
Beam是怎么做的？

在任何一个设计开始之前，都先要确定问题，Beam也不例外。

1、数据。分布式数据处理要处理的数据类型一般可以分为两类，有限的数据集和无限的数据流。有限的数据集，比如一个HDFS中的文件，一个HBase表等，特点是数据提前已经存在，一般也已经持久化，不会突然消失，不会再改变。而无限的数据流，比如kafka中流过来的系统日志流，或是从Twitter API拿到的Twitter流等等，这类数据的特点是，数据动态流入，无穷无尽，无法全部持久化。一般来说，批处理框架的设计目标是用来处理有限的数据集，流处理框架的设计目标是用来处理无限的数据流。有限的数据集可以看做是无限的数据流的一种特例，但是从数据处理逻辑的角度，这两者并无不同之处。

2、时间。Process Time是指数据进入分布式处理框架的时间，而Event-Time则是指数据产生的时间。这两个时间通常是不同的，例如，对于一个处理微博数据的流计算任务，一条2016-06-01-12:00:00发表的微博经过网络传输等延迟可能在2016-06-01-12:01:30才进入到流处理系统中。批处理任务通常进行全量的数据计算，较少关注数据的时间属性，但是对于流处理任务来说，由于数据流是无情无尽的，无法进行全量的计算，通常是对某个窗口中得数据进行计算，对于大部分的流处理任务来说，按照时间进行窗口划分，可能是最常见的需求。

3、乱序。对于流处理框架处理的数据流来说，其数据的到达顺序可能并不严格按照Event-Time的时间顺序。如果基于Process Time定义时间窗口，数据到达的顺序就是数据的顺序，因此不存在乱序问题。但是对于基于Event Time定义的时间窗口来说，可能存在时间靠前的消息在时间靠后的消息之后到达的情况，这在分布式的数据源中可能非常常见。对于这种情况，如何确定迟到数据，以及对于迟到数据如何处理通常是很棘手的问题。
Beam模型处理的目标数据是无限的时间乱序数据流，不考虑时间顺序或是有限的数据集可看做是无限乱序数据流的一个特例。

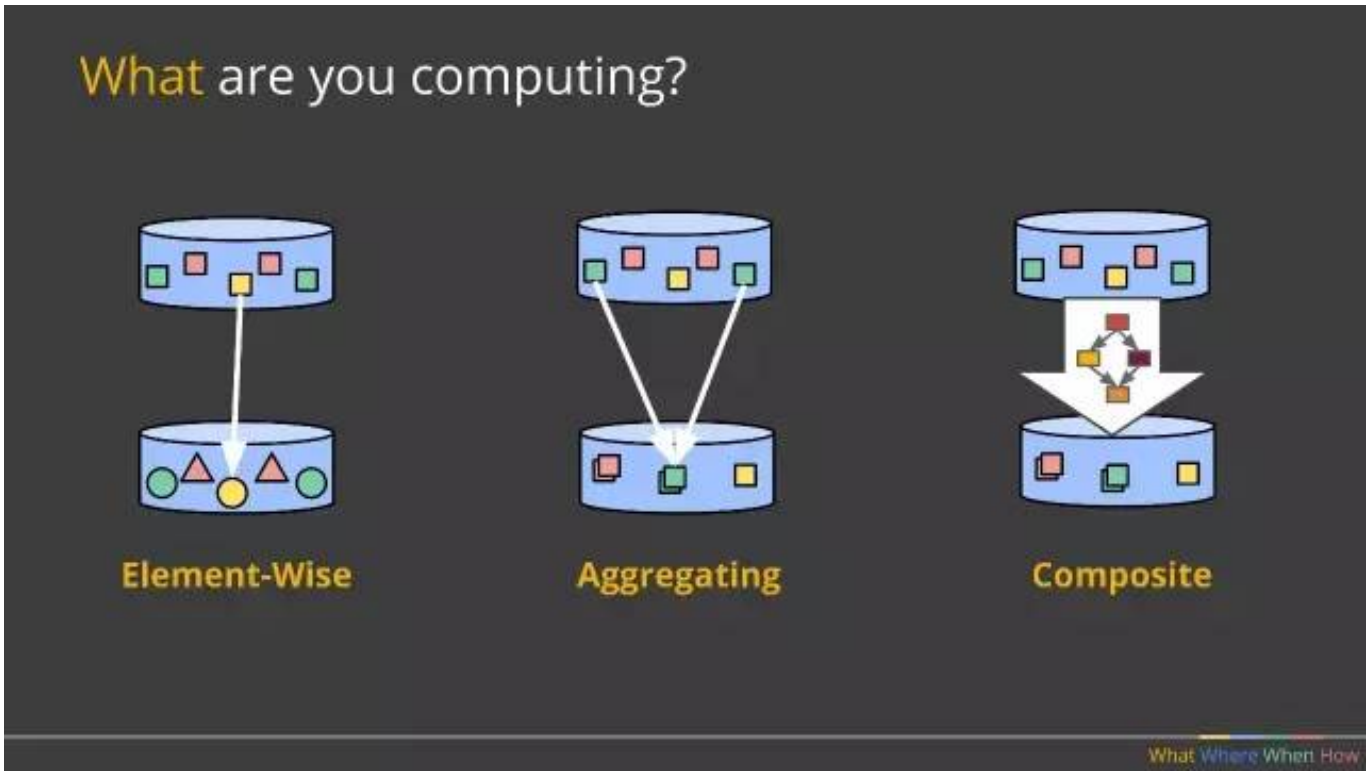


如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

如上图，其中虚线是最理想的，表示处理时间和事件时间是相同的，红线是实际上的线，也叫水位线（Watermark），它一般是通过启发式算法算出来的。

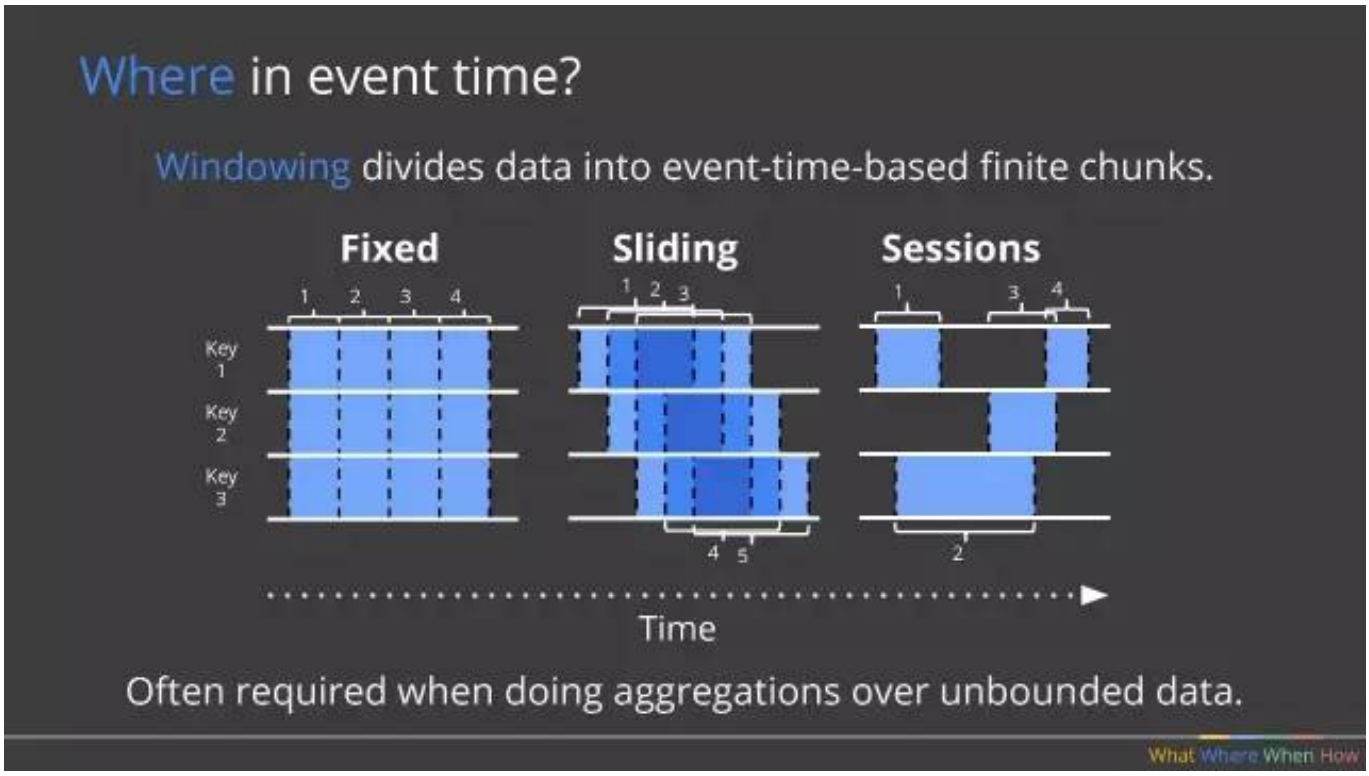
接下来从问题中抽象出四个具体的问题：

A：What are you computing，对数据的处理是哪种类型，数据转换、聚合或者是两者都有。例如，Sum、Join或是机器学习中训练学习模型等。在Beam
SDK中由Pipeline中的操作符指定。如图：



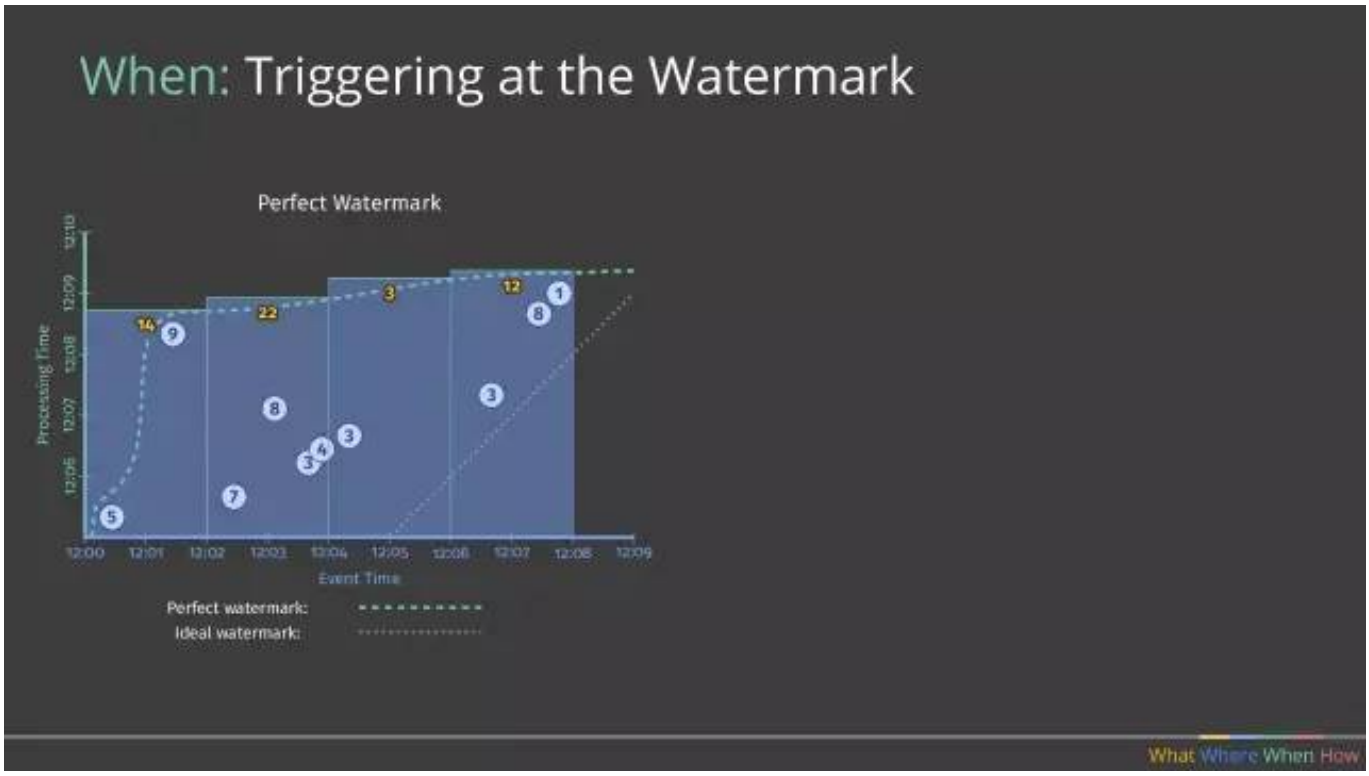
如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

B：Where in event time，数据在什么范围中计算？例如，基于Process-Time的时间窗口？基于Event-Time的时间窗口？滑动窗口等等。在Beam SDK中由Pipeline中的窗口指定：

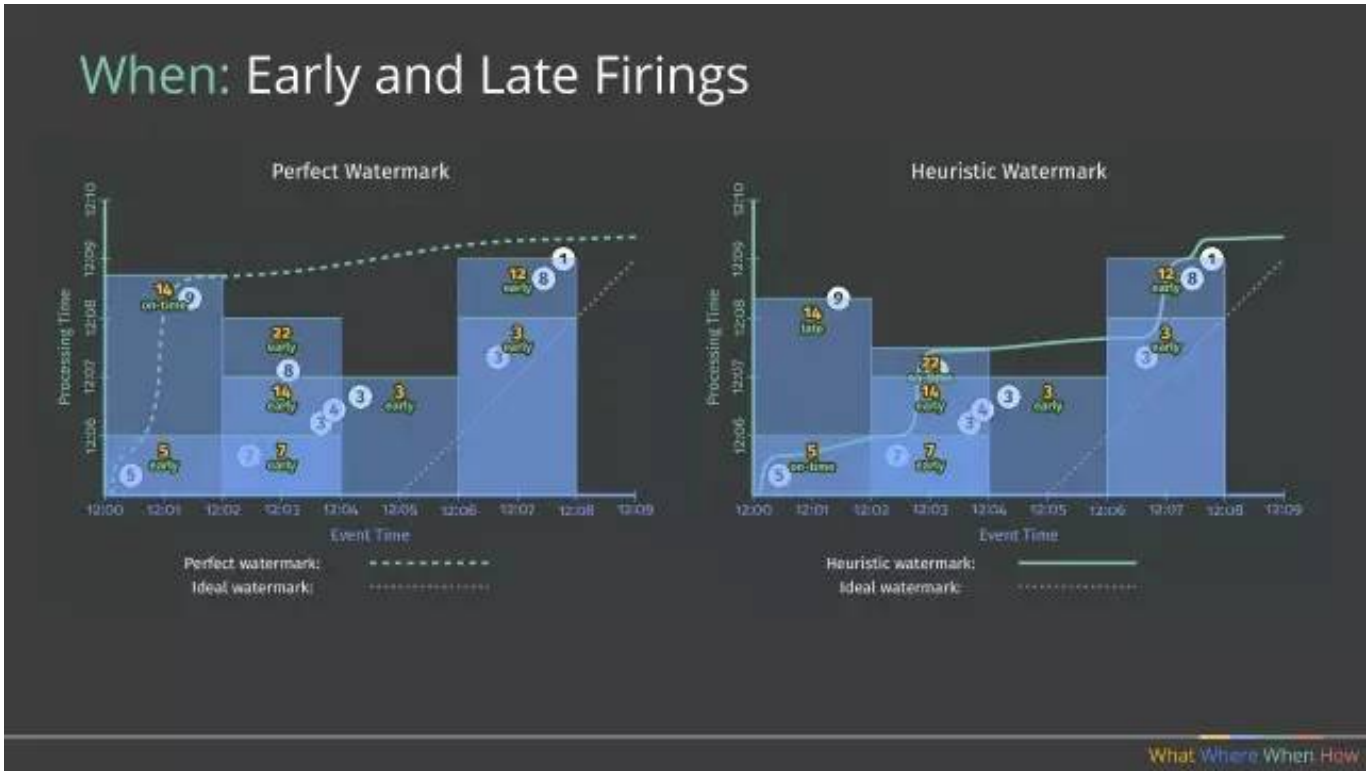


如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

C：When in processing time，何时将计算结果输出？在这里引入了一个Trigger机制，Trigger决定何时将计算结果发射出去，发射太早会丢失一部分数据，丧失精确性，发射太晚会导致延迟变长，而且会囤积大量数据，何时Trigger是由水位线来决定的，在Beam
SDK中由Pipeline中的水位线和触发器指定。

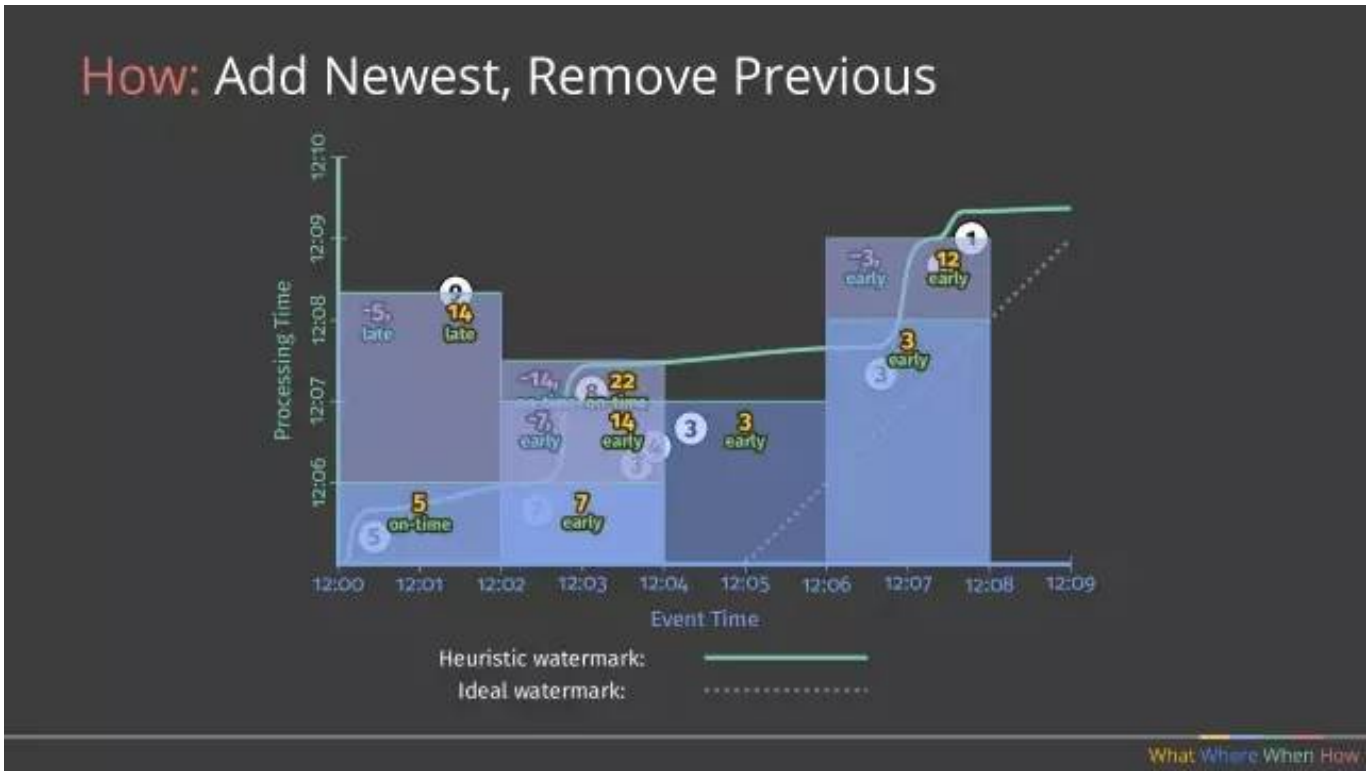


如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop



如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

D：How do refinements relate，迟到数据如何处理？例如，将迟到数据计算增量结果输出，或是将迟到数据计算结果和窗口内数据计算结果合并成全量结果输出。在Beam SDK中由Accumulation指定。



如果想及时了解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

Beam模型将”WWWH“四个维度抽象出来组成了Beam SDK，用户在基于Beam SDK构建数据处理业务逻辑时，每一步只需要根据业务需求按照这四个维度调用具体的API即可生成分布式数据处理Pipeline，并提交到具体执行引擎上执行。“WWWH”四个维度的抽象仅仅关注业务逻辑本身，和分布式任务如何执行没有任何关系。

在QCon旧金山2016上，Apache Beam的创始人Tyler Akidau分享了该项目的设计理念和架构。

友商的看法

随着分布式数据处理不断发展，新的分布式数据处理技术也不断被提出，业界涌现出了越来越多的分布式数据处理框架，从最早的Hadoop MapReduce，到Apache Spark、Apache Storm，以及更近的Apache Flink、Apache Apex等。新的分布式处理框架可能带来的更高的性能、更强大的功能和更低的延迟，但用户切换到新的分布式处理框架的代价也非常大：需要学习一个新的数据处理框架，并重写所有的业务逻辑。解决这个问题的思路包括两个部分，首先，需要一个编程范式，能够统一、规范分布式数据处理的需求，例如，统一批处理和流处理的需求。其次，生成的分布式数据处理任务应该能够在各个分布式执行引擎上执行，用户可以自由切换分布式数据处理任务的执行引擎与执行环境。Apache Beam正是为了解决以上问题而提出的。

如Apache Beam项目的主要推动者Tyler Akidau所说：

> 为了让Apache Beam能成功地完成移植，我们需要至少有一个在部署自建云或非谷歌云时，可以与谷歌Cloud Dataflow相比具备足够竞争力的Runner。如Beam能力矩阵所示，Flink满足我们的要求。有了Flink，Beam已经在业界内成了一个真正有竞争力的平台。

对此，Data Artisan的Kostas Tzoumas在他的博客中说：

> 在谷歌将他们的Dataflow SDK和Runner捐献给Apache孵化器成为Apache Beam项目时，谷歌希望我们能帮忙完成Flink Runner，并且成为新项目的代码提交者和PMC成员。我们决定全力支持，因为我们认为：1、对于流处理和批处理来说Beam模型都是未来的参考架构；2、Flink正是一个执行这样数据处理的平台。在Beam成形之后，现在Flink已经成了谷歌云之外运行Beam程序的最佳平台。

我们坚信Beam模型是进行数据流处理和批处理的最佳编程模型。我们鼓励用户们在实现新程序时采用这个模型，用Beam API或者Flink DataStream API都行。

目前主流流数据处理框架Flink、Spark、Apex以及谷歌的Cloud DataFlow等都有了支持Beam的Runner。

结论

“在谷歌公司里已经没人再使用MapReduce了”！谷歌云的主要负责人Mete Atamel如是说。谷歌坚信Apache Beam就是数据批处理和流处理的未来。Apache Beam的模型对无限乱序数据流的数据处理进行了非常优雅的抽象，“WWWH”四个维度对数据处理的描述非常清晰与合理，Beam模型在统一了对无限数据流和有限数据集的处理模式的同时，也明确了对无限数据流的数据处理方式的编程范式，扩大了流处理系统可应用的业务范围。随着Apache Beam的成功孵化，随着越来越多的编程语言可用、越来越多的分布式数据处理平台支持Beam模型，我们的确可以尽情畅想美好的未来。在今年4月的QCon北京2017上，QCon将邀请PayPal架构师、Apache Beam贡献者及PPMC成员Amit Sela来进一步分享Apache Beam的相关技术。

本文转自：为什么Google用Apache Beam彻底替换掉MapReduce