## Apache Beam实战指南之基础入门

原创 2017-10-08 张海涛 [AI前线](https://mp.weixin.qq.com/s/rrGR7sWRkrOZ82TrrqFT9Q" \l "#)



策划 & 审校｜Natalie

本文是 **Apache Beam 实战指南系列文章** 的第一篇内容，将简要介绍 Apache Beam 的发展历史、应用场景、模型和运行流程、SDKs ，并结合 Beam 的应用示例和代码剖析带你进一步了解 Beam 的运用原理。  
  
**更多精彩文章请添加微信“AI 前线”（ID：ai-front）**前言：大数据 2.0 时代不期而至

随着大数据 2.0 时代悄然到来，大数据从简单的批处理扩展到了实时处理、流处理、交互式查询和机器学习应用。早期的处理模型 (Map/Reduce) 早已经力不从心，而且也很难应用到处理流程长且复杂的数据流水线上。另外，近年来涌现出诸多大数据应用组件，如 HBase、Hive、Kafka、Spark、Flink 等。开发者经常要用到不同的技术、框架、API、开发语言和 SDK 来应对复杂应用的开发。这大大增加了选择合适工具和框架的难度，开发者想要将所有的大数据组件熟练运用几乎是一项不可能完成的任务。

面对这种情况，Google 在 2016 年 2 月宣布将大数据流水线产品（Google DataFlow）贡献给 Apache 基金会孵化，2017 年 1 月 Apache 对外宣布开源 Apache Beam，2017 年 5 月迎来了它的第一个稳定版本 2.0.0。在国内，大部分开发者对于 Beam 还缺乏了解，社区中文资料也比较少。InfoQ 期望通过 **Apache Beam 实战指南系列文章** 推动 Apache Beam 在国内的普及。

本文将简要介绍 Apache Beam 的发展历史、应用场景、模型和运行流程、SDKs 和 Beam 的应用示例。欢迎加入 Beam 中文社区深入讨论和交流。

概述

大数据处理领域的一大问题是：开发者经常要用到很多不同的技术、框架、API、开发语言和 SDK。取决于需要完成的是什么任务，以及在什么情况下进行，开发者很可能会用 MapReduce 进行批处理，用 Apache Spark SQL 进行交互请求（interactive queries），用 Apache Flink 进行实时流处理，还有可能用到基于云端的机器学习框架。

近两年涌现的开源大潮，为大数据开发者提供了十分富余的工具。但这同时也增加了开发者选择合适工具的难度，尤其对于新入行的开发者来说。这很可能拖慢、甚至阻碍开源工具的发展：把各种开源框架、工具、库、平台人工整合到一起所需工作之复杂，是大数据开发者常有的抱怨之一，也是他们支持专有大数据平台的首要原因。

Apache Beam 发展历史

Beam 在 2016 年 2 月成为 Apache 孵化器项目，并在 2016 年 12 月升级成为 Apache 基金会的顶级项目。通过十五个月的努力，一个稍显混乱的代码库，从多个组织合并，已发展成为数据处理的通用引擎，集成多个处理数据框架，可以做到跨环境。

Beam 经过三个孵化器版本和三个后孵化器版本的演化和改进，最终在 2017 年 5 月 17 日迎来了它的第一个稳定版 2.0.0。发布稳定版本 3 个月以来，Apache Beam 已经出现明显的增长，无论是通过官方还是社区的贡献数量。Apache Beam 在谷歌云方面也已经展示出了“才干”。

Beam 2.0.0 改进了用户体验，重点在于框架跨环境的无缝移植能力，这些执行环境包括执行引擎、操作系统、本地集群、云端以及数据存储系统。Beam 的其他特性还包括如下几点：

API 稳定性和对未来版本的兼容性。

有状态的数据处理模式，高效的支持依赖于数据的计算。

支持用户扩展的文件系统，支持 Hadoop 分布式发文件系统及其他。

提供了一个度量指标系统，可用于跟踪管道的执行状况。

网上已经有很多人写过 Beam 2.0.0 版本之前的资料，但是 2.0.0 版本后 API 很多写法变动较大，本文将带着大家从零基础到 Apache Beam 入门。

Apache Beam 应用场景

Google Cloud、PayPal、Talend 等公司都在使用 Beam，国内包括阿里巴巴、百度、金山、苏宁、九次方大数据、360、慧聚数通信息技术有限公司等也在使用 Beam，同时还有一些大数据公司的架构师或研发人员正在一起进行研究。Apache Beam 中文社区正在集成一些工作中的 runners 和 sdk IO，包括人工智能、机器学习和时序数据库等一些功能。

以下为应用场景的几个例子：

Beam 可以用于 ETL Job 任务

Beam 的数据可以通过 SDKs 的 IO 接入，通过管道可以用后面的 Runners 做清洗。

Beam 数据仓库快速切换、跨仓库

由于 Beam 的数据源是多样 IO，所以用 Beam 可以快速切换任何数据仓库。

Beam 计算处理平台切换、跨平台

Runners 目前提供了 3-4 种可以切换的平台，随着 Beam 的强大应该会有更多的平台提供给大家使用。

Apache Beam 运行流程IMG_257

4-1 数据处理流程

如图 4-1 所示，Apache Beam 大体运行流程分成三大部分：

Modes

Modes 是 Beam 的模型或叫数据来源的 IO，它是由多种数据源或仓库的 IO 组成，数据源支持批处理和流处理。

Pipeline

Pipeline 是 Beam 的管道，所有的批处理或流处理都要通过这个管道把数据传输到后端的计算平台。这个管道现在是唯一的。数据源可以切换多种，计算平台或处理平台也支持多种。需要注意的是，管道只有一条，它的作用是连接数据和 Runtimes 平台。

Runtimes

Runtimes 是大数据计算或处理平台，目前支持 Apache Flink、Apache Spark、Direct Pipeline 和 Google Clound Dataflow 四种。其中 Apache Flink 和 Apache Spark 同时支持本地和云端。Direct Pipeline 仅支持本地，Google Clound Dataflow 仅支持云端。除此之外，后期 Beam 国外研发团队还会集成其他大数据计算平台。由于谷歌未进入中国，目前国内开发人员在工作中对谷歌云的使用应该不是很多，主要以前两种为主。为了使读者读完文章后能快速学习且更贴近实际工作环境，后续文章中我会以前两种作为大数据计算或处理平台进行演示。

Beam Model 及其工作流程

Beam Model 指的是 Beam 的编程范式，即 Beam SDK 背后的设计思想。在介绍 Beam Model 之前，先简要介绍一下 Beam Model 要处理的问题域与一些基本概念。

数据源类型。分布式数据来源类型一般可以分为两类，有界的数据集和无界的数据流。有界的数据集，比如一个 Ceph 中的文件，一个 Mongodb 表等，特点是数据已经存在，数据集有已知的、固定的大小，一般存在磁盘上，不会突然消失。而无界的数据流，比如 Kafka 中流过来的数据流，这种数据的特点是数据动态流入、没有边界、无法全部持久化到磁盘上。Beam 框架设计时需要针对这两种数据的处理进行考虑，即批处理和流处理。

时间。分布式框架的时间处理有两种，一种是全量计算，另一种是部分增量计算。我给大家举个例子：例如我们玩“王者农药”游戏，游戏的数据需要实时地流向服务器，掉血情况会随着时间实时变化，但是排行榜的数据则是全部玩家在一定时间内的排名，例如一周或一个月。Beam 针对这两种情况都设计了对应的处理方式。

乱序。对于流处理框架处理的数据流来说，数据到达大体分两种，一种是按照 Process Time 定义时间窗口，这种不用考虑乱序问题，因为都是关闭当前窗口后才进行下一个窗口操作，需要等待，所以执行都是有序的。而另一种，Event Time 定义的时间窗口则不需要等待，可能当前操作还没有处理完，就直接执行下一个操作，造成消息顺序处理但结果不是按顺序排序了。例如我们的订单消息，采用了分布式处理，如果下单操作所属服务器处理速度比较慢，而用户支付的服务器速度非常快，这时最后的订单操作时间轴就会出现一种情况，下单在支付的后面。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麻烦的事情。

Beam Model 处理的目标数据是无界的时间乱序数据流，不考虑时间顺序或有界的数据集可看做是无界乱序数据流的一个特例。Beam Model 从下面四个维度归纳了用户在进行数据处理的时候需要考虑的问题：

**What**。如何对数据进行计算？例如，机器学习中训练学习模型可以用 Sum 或者 Join 等。在 Beam SDK 中由 Pipeline 中的操作符指定。

**Where**。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 的窗口指定。

**When**。何时输出计算结果？例如，在 1 小时的 Event-Time 时间窗口中，每隔 1 分钟将当前窗口计算结果输出。在 Beam SDK 中由 Pipeline 的 Watermark 和触发器指定。

**How**。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。

Beam Model 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”四个维度只是从业务的角度看待问题，并不是全部适用于自己的业务。做技术架构一定要结合自己的业务使用相应的技术特性或框架。Beam 做为“一统”的框架，为开发者带来了方便。

Beam SDKs

Beam SDK 给上层应用的开发者提供了一个统一的编程接口，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界数据集，还是流式的无界数据集。对于这两类输入数据，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 拥有不同编程语言的实现，目前已经完整地提供了 Java 的 SDK，Python 的 SDK 还在开发中，相信未来会发布更多不同编程语言的 SDK。

Beam 2.0 的 SDKs 目前有：

**Amqp**：高级消息队列协议。

**Cassandra**：Cassandra 是一个 NoSQL 列族（column family）实现，使用由 Amazon Dynamo 引入的架构方面的特性来支持 Big Table 数据模型。Cassandra 的一些优势如下所示：

高度可扩展性和高度可用性，没有单点故障

NoSQL 列族实现

非常高的写入吞吐量和良好的读取吞吐量

类似 SQL 的查询语言（从 0.8 版本起），并通过二级索引支持搜索

可调节的一致性和对复制的支持灵活的模式

**Elasticesarch**：一个实时的分布式搜索引擎。

**Google-cloud-platform**：谷歌云 IO。

**Hadoop-file-system**：操作 Hadoop 文件系统的 IO。

**Hadoop-hbase**：操作 Hadoop 上的 Hbase 的接口 IO。

**Hcatalog**：Hcatalog 是 Apache 开源的对于表和底层数据管理统一服务平台。

**Jdbc**：连接各种数据库的数据库连接器。

**Jms**：Java 消息服务（Java Message Service，简称 JMS）是用于访问企业消息系统的开发商中立的 API。企业消息系统可以协助应用软件通过网络进行消息交互。JMS 在其中扮演的角色与 JDBC 很相似，正如 JDBC 提供了一套用于访问各种不同关系数据库的公共 API，JMS 也提供了独立于特定厂商的企业消息系统访问方式。

**Kafka**：处理流数据的轻量级大数据消息系统，或叫消息总线。

**Kinesis**：对接亚马逊的服务，可以构建用于处理或分析流数据的自定义应用程序，以满足特定需求。

**Mongodb**：MongoDB 是一个基于分布式文件存储的数据库。

**Mqtt**：IBM 开发的一个即时通讯协议。

**Solr**：亚实时的分布式搜索引擎技术。

**xml**：一种数据格式。

Beam Pipeline Runners

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无法达到这一期望。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。

Beam 2.0 的 Runners 框架如下：

**Apex**

诞生于 2015 年 6 月的 Apache Apex，其同样源自 DataTorrent 及其令人印象深刻的 RTS 平台，其中包含一套核心处理引擎、仪表板、诊断与监控工具套件外加专门面向数据科学家用户的图形流编程系统 dtAssemble。主要用于流处理，常用于物联网等场景。

**Direct-java**

本地处理和运行 runner。

**Flink\_2.10**

Flink 是一个针对流数据和批数据的分布式处理引擎。

**Gearpump**

Gearpump 是一个基于 Akka Actor 的轻量级的实时流计算引擎。如今流平台需要处理来自各种移动端和物联网设备的海量数据，系统要能不间断地提供服务，对数据的处理要能做到不丢失不重复，对各种软硬件错误能平滑处理，对用户的输入要能实时响应。除了这些系统层面的需求外，用户层面的接口还要能做到丰富而灵活，一方面，平台要提供足够丰富的基础设施，能最简化应用程序的编写；另一方面，这个平台应提供具有表现力的编程 API，让用户能灵活表达各种计算，并且整个系统可以定制，允许用户选择调度策略和部署环境，允许用户在不同的指标间做折中取舍，以满足特定的需求。Akka Actor 提供了通信、并发、隔离、容错的基础设施，Gearpump 通过把抽象层次提升到 Actor 这一层，屏蔽了底层的细节，专注于流处理需求本身，能更简单而又高效地解决上述问题。

**Dataflow**

2016 年 2 月份，谷歌及其合作伙伴向 Apache 捐赠了一大批代码，创立了孵化中的 Beam 项目（最初叫 Apache Dataflow）。这些代码中的大部分来自于谷歌 Cloud Dataflow SDK——开发者用来写流处理和批处理管道（pipelines）的库，可在任何支持的执行引擎上运行。当时，支持的主要引擎是谷歌 Cloud Dataflow。

**Spark**

Apache Spark 是一个正在快速成长的开源集群计算系统。Apache Spark 生态系统中的包和框架日益丰富，使得 Spark 能够执行高级数据分析。Apache Spark 的快速成功得益于它的强大功能和易用性。相比于传统的 MapReduce 大数据分析，Spark 效率更高、运行时速度更快。Apache Spark 提供了内存中的分布式计算能力，具有 Java、Scala、Python、R 四种编程语言的 API 编程接口。

实战：开发第一个 Beam 程序8.1 开发环境

下载安装 JDK 7 或更新的版本，检测 JAVA\_HOME 环境变量。本文示例使用的是 JDK 1.8。

下载 maven 并配置，本文示例使用的是 maven-3.3.3。

开发环境 myeclipse、Spring Tool Suite 、IntelliJ IDEA，这个可以按照个人喜好，本文示例用的是 STS。

8.2 开发第一个 wordCount 程序并且运行

1 新建一个 maven 项目

IMG_258

2 在 pom.xml 文件中添加两个 jar 包

IMG_259

3 新建一个 txtIOTest.java

IMG_260

写入以下代码：

IMG_261

4 因为 Windows 上的 Beam2.0.0 不支持本地路径，需要部署到 Linux 上，需要打包如图，此处注意要把依赖 jar 都打包进去。

IMG_262

IMG_263

5 部署 beam.jar 到 Linux 环境中

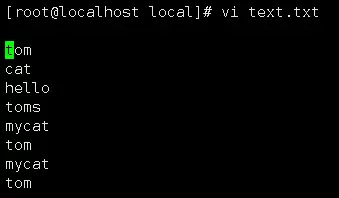
使用 Xshell 5 登录虚拟机或者 Linux 系统。用 rz 命令把刚才打包的文件上传上去。其中虚拟机要安装上 jdk 并配置好环境变量。

我们可以用输入 javac 命令测试一下。

IMG_264

我们把 beam.jar 上传到 /usr/local/ 目录下面，然后新建一个文件，也就是源文件。命令：touch  text.txt　命令：chmod o+rwx　text.txt

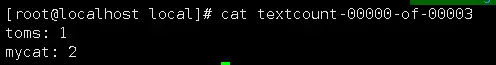
修改 text.txt 并添加数据。　命令：vi text.txt



运行命令：java -jar beam.jar，生成文件。

IMG_266

用 cat 命令查看文件内容，里面就是统计的结果。



8.3 实战剖析

我们可以通过以上实战代码进一步了解 Beam 的运用原理。

第一件事情是搭建一个管道（Pipeline），例如我们小时候家里浇地用的“水管”。它就是连接水源和处理的桥梁。

PipelineOptions pipelineOptions = PipelineOptionsFactory.create();// 创建管道

第二件事情是让我们的管道有一个处理框架，也就是我们的 Runtimes  。例如我们接到水要怎么处理，是输送给我们城市的污水处理厂，还是其他。这个污水处理厂就相当于我们的处理框架，例如现在流行的 Apache Spark 或 Apache Flink。这个要根据自己的业务指定，如下代码中我指定了本地的处理框架。

pipelineOptions.setRunner(DirectRunner.class);

第三件事情也是 Beam 最后一个重要的地方，就是模型 (Model)，通俗点讲就是我们的数据来源。如果结合以上第一件和第二件的事情说就是水从哪里来，水的来源可能是河里、可能是污水通道等等。本实例用的是有界固定大小的文本文件。当然 Model 还包含无界数据，例如 kafka 等等，可以根据的需求灵活运用。

pipeline.apply(TextIO.read().from("/usr/local/text.txt")).apply  
("ExtractWords", ParDo.of(new DoFn<String, String>() //后省略

最后一步是处理结果，这个比较简单，可以根据自己的需求处理。希望通过代码的实战结合原理剖析可以帮助大家更快地熟悉 Beam 并能够简单地运用 Beam。

总结

Apache Beam 是集成了很多数据模型的一个统一化平台，它为大数据开发工程师频繁换数据源或多数据源、多计算框架提供了集成统一框架平台。Apache Beam 社区现在已经集成了数据库的切换 IO，未来 Beam 中文社区还将为 Beam 集成更多的 Model 和计算框架，为大家提供方便。

##### 作者介绍

张海涛，目前就职于海康威视云基础平台，负责云计算大数据的基础架构设计和中间件的开发，专注云计算大数据方向。Apache Beam 中文社区发起人之一，如果想进一步了解最新 Apache Beam 动态和技术研究成果，请加微信 cyrjkj 入群共同研究和运用。