Flink详细介绍(一)

唉. 2019-10-18 17:38:46 26071 收藏 61

分类专栏： Flink 文章标签： Flink

版权

Flink

专栏收录该内容

11 篇文章6 订阅

订阅专栏

文章目录

一、概述

1.1、流处理技术的演变

1.2、初识Flink

1.3、Flink核心计算框架

二、Flink基本架构

2.1、JobManager和TaskManager

2.2、无界数据流和有界数据流

2.3、数据流编程模型

三、Flink运行架构

3.1、任务提交流程

3.2、TaskManager与Slots

3.3、DataFlow

3.3、并行数据流

3.5、task和operator chains

3.6、任务调度流程

一、概述

1.1、流处理技术的演变

Storm提供了低延迟的流处理，但是为了实时性付出了一些代价：很难实现高吞吐，并且正确性没有达到通常所需的水平换句话说它并不能保证exactly-one，要想保证正确性级别开销也是很大的。

低延迟和高吞吐流处理系统中维持良好的容错性很困难，但是人们想到了一种替代方法：将连续时间中的流数据分割成一系列微小的批量作业。这就是Spark批处理引擎上运行的Spark Streaming所使用的方法。

Storm Trident是对Storm的延伸，底层流处理引擎就是基于微批处理方法来进行计算的，从而实现了exactly-one语义，但是在延迟性方面付出了很大的代价。

Flink这一技术框架上可以避免上述弊端并且拥有所需的诸多功能，还能按照连续事件高效的处理数据，Flink部分特性如下图所示：

1.2、初识Flink

Flink主页在其顶部展示了该项目的理念：Flink是分布式、高性能、随时可以用以及准确的流处理应用程序打造的开源流处理框架。

Apache Flink是一个框架和分布式处理引擎，用于对无界和有界数据流进行有状态计算。Flink被设计在所有常见的集群环境中运行，以内存执行速度和任意规模来执行计算。

1.3、Flink核心计算框架

Flink核心计算框架是Flink Runtime执行引擎，是一个分布式系统，能够接受数据流程序并在一台或多台机器上以容错方式执行。

Flink Runtime执行引擎可以作为Yarn的应用程序在集群上运行也可在Mesos集群上运行，还可以在单机上运行(对于调试Flink应用程序来说非常有用)。

Flink分别提供了面向流式处理的接口和面向批处理的接口。Flink提供了用于流处理的DataStream API和用于批处理的DataSet API。Flink的分布式特点体现在它能够在成百上千机器上运行，它将大型计算任务分成许多小的部分每个机器执行一部分。

二、Flink基本架构

2.1、JobManager和TaskManager

JobManager：也称之为Master用于协调分布式执行，它们用来调度task协调检查点，协调失败时恢复等。Flink运行时至少存在一个master处理器，如果配置高可用模式则会存在多个master处理器，其中有一个是leader其他都是standby。

TaskManager：也称之为Worker，用于执行一个dataflow的task(或者特殊的subtask)、数据缓冲和datastream的交换，Flink运行时至少会存在一个worker处理器。

Master和Worker处理器可以直接在物理机上启动或者通过相Yarn这样的资源调度框架。

Worker连接到Master，告知自身的可用性进而获得任务分配。

2.2、无界数据流和有界数据流

无界数据流：无界数据流有一个开始但是没有结束，它们不会在生成时终止 并提供数据，必须连续处理无界流，就是说必须在获取后立即处理event。处理无界数据通常要求以特定顺序获取event，以便能够推断结果完整性，无界流的处理称为流处理。

有界数据流：有界数据流有明确定义的开始和结束。可以在执行任何计算之前通过获取所有数据来处理有界流，处理有界流不需要有序获取，因为可以始终对有界数据尽心排序，有界流的处理也称为批处理。

批处理的特点是有界、持久、大量，批处理非常适合需要访问全套记录才能完成的计算工作，一般要用于离线统计。流处理的特点是无界、实时、流处理方式无需针对整个暑假集执行操作，而是对通过系统传输的每个数据项执行操作，一般用于实时统计。

Spark中批处理由SparkSql实现，流处理由SparkStreaming实现。

那么Flink如何同时实现批处理和流处理的呢？？Flink将批处理(即处理有限的静态数据)视作一种特殊的流处理。

Flink是一个面向分布式数据流处理和批量数据处理的开源计算平台，能够基于同一个Flink运行时，提供支持流处理和批处理两种类型应用的功能。流处理一般需要支持低延迟、Exactly-once保证，而批处理需要支持高吞吐、高效处理。

Flink完全支持流处理就是说作为流处理看待时输入数据流是无界的，批处理被作为一种特殊的流处理只是它的输入数据流被定义为有界的。

2.3、数据流编程模型

Flink提供了不同级别的抽象，以开发流或批处理作业如下图所示：

大多数应用并不需要上述的底层抽象，而是针对核心API进行编程，比如DataStream API(有界或无界数据流)以及DataSet API(有界数据集)。可以在表与DataStream/DataSet之间无缝切换，以允许程序将Table API与DataStream以及DataSet混合使用。

三、Flink运行架构

3.1、任务提交流程

Flink提交任务后Client向HDFS上传Flink的jar包和配置，之后向Yarn ResourceManager提交任务，ResourceManager分配容器资源并通知对用的NodeManager启动ApplicationMaster，ApplicationMaster启动后加载Flink的jar包和配置构建环境，然后启动JobManager，之后ApplicationMaster向ResourceManager申请资源启动TaskManager，ResourceManager分配容器资源后由ApplicationMaster加载Flink的jar包和配置构建环境并启动TaskManager，TaskManager启动后向JobManager发送心跳包，并等待JobManager向其分配任务。

3.2、TaskManager与Slots

每个TaskManager是一个JVM进程，它可能会在独立的线程上执行一个或多个subtask。为了控制一个worker能接收多少个task，worker通过task slot来进行控制(一个worker至少有一个task slot)。

每个task slot表示TaskManager拥有资源的一个固定大小的子集。一个TaskManager有三个slot，那么会把他管理的内存分配给三个slot。资源slot化意味着一个subtask将不需要跟来自其他job的subtask竞争被管理的内存，取而代之的都会是它将有一定数量的内存储备。slot仅用来隔离task受管理的内存。

通过调整task slot的数量，允许用户定义subtask之间如何互相隔离。一个TaskManager一个slot意味着每个task group运行在独立的JVM中，一个TaskManager多个slot意味着更多的subtask共享一个JVM。在同一个JVM进程中的task将共享TCP连接(基于多路复用)和心跳信息。它们也可能共享数据集合数据结构，因此这减少了每个task的负载。

Task Slot是静态的概念是指TaskManager具有的并发执行能力，并行度是动态概念即TaskManager运行程序时实际使用的并发能力。

3.3、DataFlow

Flink程序由Source、Transformation、Sink三个核心组件组成，Source主要负责数据读取，Transformation主要负责对属于的转换操作，Sink主要负责最终数据的输出，在各个组件之间流转的数据成为流(streams)。

Flink程序基础构建模块是流与转换，Flink的DataSet API所使用的DataSets内部也是steam。一个stream可以看成一个中间结果而一个Transformations是以一个或多个stream作为输入的某种operation，利用这些stream进行计算从而产生一个或多个result stream。

在运行时候Flink上运行的程序会被映射成streaming dataflows，包含了streams和transformations operators。每一个dataflow以一个或多个sources开始一个或多个sinks结束，dataflow类似于任意的DAG。

3.3、并行数据流

Flink程序的执行具有并行性、分布式的特性。执行过程中一个stream包含一个或多个stream partition，而每一个operator包含一个或多个operator subtask，这些operator subtasks在不同的线程、不同的物理机或不同的容器中彼此互不依赖得执行。

一个特定operator的subtask的个数被称之为其并行度。一个stream并行度总是等同于其producing operator的并行度。一个程序中不同的operator可能具有不同的并行度。

Stream在operator之间传输数据的形式可以是one-to-one模式也可以是redistributing模式，具体取决于operator种类。

one-to-one：stream(比如在source和map operator之间)维护者分区以及元素的顺序。意味着map operator的subtask看到的元素个数以及顺序跟source operator的subtask生产的元素个数、顺序相同，map、fliter、flatmap等算子都是one-to-one的对应关系。

redistributing：这种操作会改变数据的分区个数。每一个operator subtask依据所选择的transformation发送数据到不同的目标subtask。例如，keyBy()基于hashCode重分区、broadcast和rebalance会随机重新分区，这些算子都会引起redistribute过程，而redistribute过程就类似于Spark中的shuffle过程。

3.5、task和operator chains

出于分布式执行的目的Flink将operator的subtask连接在一起形成task，每个task在一个线程中国执行。将operators连接成task是有效的优化；这样做可以减少线程之间的切换和基于缓存区的数据交换，在减少时延同时提升吞吐量。连接行为可以在编程API进行指定。下图是5个subtask以五个并行的线程来执行：

3.6、任务调度流程

客户端不是运行时和程序执行一部分，但它用于准备并发送dataflow给master，然后客户端断开连接或者维持连接以等待接收计算结果，客户端可以以两种方式运行：要么作为作为Java/Scala程序的一部分被程序触发执行，要么以命令行./bin/flink run方式运行

————————————————

版权声明：本文为CSDN博主「唉.」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/weixin\_44240370/article/details/102594925