Storm相关知识

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 版本 | 修订者 | 修订描述 |
| 2017.12.28 | v1 | 林 清 | 初稿 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[1. 介绍 2](#_Toc502302359)

[2. 定位和应用场景 2](#_Toc502302360)

[3. Storm和Spark、以及Hadoop MR的比较 2](#_Toc502302361)

[4. 数据源 2](#_Toc502302362)

[5. 数据输出存储 3](#_Toc502302363)

[6. Strom开发 3](#_Toc502302364)

[6.1 开发简单 3](#_Toc502302365)

[6.2 多语言支持 3](#_Toc502302366)

[7. Storm的单点故障 3](#_Toc502302367)

[8. 消息容错（Exactly-Once） 4](#_Toc502302368)

[9. JStorm 4](#_Toc502302369)

# 介绍

Storm是一个分布式的，可靠的，容错的数据流处理系统。

# 定位和应用场景

Storm的定位是流式计算引擎，官方的项目定位 ------“百万数据量级别实时数据处理的标杆”。

官方提供唯一性能指标：单节点处理100 byte \* 1000000条数据/秒，硬件需求是2x Intel [E5645@2.4Ghz，24GB](mailto:E5645@2.4Ghz，24GB)内存。

Storm最大特点是极佳的实时性，并且有可靠的事务机制，数据的处理完全精准（表现就是，不会丢数据！！！）。应用场景包括，实时金融系统（高频交易、股票交易等等）、分布式RPC管道、网站推荐系统、预警系统、日志分析系统等等，凡是不能忍受1秒延时的数据处理，可以考虑一下能不能使用Storm。

Storm官网列出了Storm用户（http://storm.apache.org/releases/current/Powered-By.html），包括：Twitter、雅虎、阿里、百度、爱奇艺等等。比较值得关注的是阿里，他们大规模用了Storm，并且用Java重构了Apache Storm，搞了一个JStorm，还开源了。

Storm的主要竞争对手实际上是Twitter开源的另一个大数据实时分析系统Heron（没有深入调研这个东西，但是看到一篇阿里的文章说这个东西架构设计有重大缺陷，并且性能也不咋地！） 。

# Storm和Spark、以及Hadoop MR的比较

首先，Spark、Hadoop MR的定位都是批处理系统，虽然Spark Streaming 也能进行流式计算，但是本质上还是准实时的。他们在意的是处理数据时，数据的吞吐量。

Storm定位是实时系统，数据来一条处理一条，在实时性、事务机制、健壮性 / 容错性、动态调整并行度等能力都胜过上面两个。

但是如果业务对实时性的要求不是太高，Spark Streaming有更优秀的吞吐量，并且有丰富的生态，配合和Spark Core、Spark SQL可以对中间数据进行交互式查询等操作。

# 数据源

Storm的最佳数据源是消息队列，官方给出的推荐包括：Kestrel（twitter开源的消息中间件）、RabbitMQ / AMQP、Kafka、JMS（指所有支持JMS协议的中间件）、Amazon Kinesis。当然Storm处理的数据也可来自数据库，或者API调用。

# 数据输出存储

Storm本身是一个实时计算的框架，没有提供存储服务（Hadoop定位是批处理，包含：HDFS存储，数据起止都要落盘）。

Storm可以接入大部分DB，必要时可以和Hive对接。

# Strom开发

## 开发简单

Storm本身概念抽象很少，最核心的只有：spout、bolts、topologies。Storm提供了一套Trident API，在更高的层次上封装了一些概念，开发难度比较低（Trident和spout、bolts、topologies的关系类似JDBC和ORM）。

spout：数据源，从消息队列或者其他的一些API调用中获取数据；

bolts：处理节点，处理上游节点（spout或者bolts）传过来的数据，之后将数据传递给下游；

topologies：spout和bolts构成的有向无环图（DAG），部署一个topologies之后，他会无限地运行下去；

## 多语言支持

Storm本身是基于Clojure编写的，且运行在JVM上。但是，Storm上运行的topologies可以用任意语言编写。在非JVM场景下使用Storm时，主要用标准输出/输入作为与topologies任务通信的信道，消息格式是JSON或者普通文本。（关于多语言支持更本质的原因是：Nimbus本身是一个thrift服务，topologies被定义为thrift结构体，支持通过RPC去调用另一种语言的接口。thrift本身是facebook开源的跨语言的服务框架。）。Spark API只支持Scala、Java以及Python（目前完善的只有Scala），多语言这一点Storm有优势。

# Storm的单点故障

Storm是经典的master/slave模式，在Storm中有两种节点：

Nimbus 节点（Master节点）：负责监听topology的提交，向工作节点分发代码，并且在集群上分配topology任务。

Supervisors节点（Slaver节点）：负责从Nimbus 节点下载代码，并且根据Nimbus 的指示启动worker进程。

上面提到的Supervisors进程、Nimbus 进程、worker进程都是无状态的，任意重启不会影响整个Storm集群，并且有Watchdog程序监控Supervisors进程（Supervisors进程本身可以认为是他管理的Work进程的Watchdog）、Nimbus 进程，一旦这些进程挂了，立即重启。

虽然，Nimbus 进程是状态的，但是Storm 0.8.2之前没有Nimbus 节点没有HA支持。网上的一篇博客提到，Storm的作者认为Nimbus单点故障的不是Storm最紧急和严重的问题，所以一直没有把HA特性放到开发日程（Nimbus单点故障后，Storm集群已经运行的任务没有影响，但是Storm无法发布新的topology，WebUI也无法访问）。当前Storm的版本到1.1.1，nimbus-HA依然有一些问题（主要问题是：节点选举机制、topology代码本地存储，虽然Git上有解决方案，但是都还没有和入Storm的官方github分支）。

# 消息容错（Exactly-Once）

Storm可以保障每条数据必定处理，并且不会丢失数据（官方称为数据Exactly-Once，每条消息有且只有一次，即消息传输既不会丢失也不会重复。如何保障，官网有[详细说明](http://storm.apache.org/releases/current/Guaranteeing-message-processing.html)）。

在Storm中，一条数据称为Tuple。一个Tuple在Storm处理的过程中会生成一个Tuple Tree（原因是bolt会生成子数据）。Storm监控Tuple Tree上的所有节点，只有当这些节点都处理完成之后，Storm才认为该Tuple已经处理OK。监控的方式主要是等待每个Tuple节点ACK，一旦超时就认为这个节点的发生了失败，并重发Tuple。由于监控Tuple Tree需要占用内存，如果Tuple Tree（规模可能会上百万）的规模太大可能会导致节点内存耗尽。所以Storm的Tuple追踪算法是比较核心的。

# JStorm

JStorm是阿里基于Storm用Java重写的流式计算引擎。JStorm也是开源的，官网：<http://www.jstorm.io>。

阿里号称，JStorm的性能是Strom的4倍，并且功能比Storm更全面，是他的超集，并且JStorm的master节点实现了HA，这一点是Storm没有的。

JStorm官网有针对JStorm/Apache Storm/Apache Flink/Heron的性能测试报告（JStorm/Apache Storm/Apache Flink/Heron）。