# **[Flink（一）Flink的入门简介](https://www.cnblogs.com/frankdeng/p/9400622.html)**

## **一. **Flink的引入****

        这几年大数据的飞速发展，出现了很多热门的开源社区，其中著名的有 Hadoop、Storm，以及后来的 Spark，他们都有着各自专注的应用场景。Spark 掀开了内存计算的先河，也以内存为赌注，赢得了内存计算的飞速发展。Spark 的火热或多或少的掩盖了其他分布式计算的系统身影。就像 Flink，也就在这个时候默默的发展着。

在国外一些社区，有很多人将大数据的计算引擎分成了 4 代，当然，也有很多人不会认同。我们先姑且这么认为和讨论。

首先第一代的计算引擎，无疑就是 Hadoop 承载的 MapReduce。这里大家应该都不会对 MapReduce 陌生，它将计算分为两个阶段，分别为 Map 和 Reduce。对于上层应用来说，就不得不想方设法去拆分算法，甚至于不得不在上层应用实现多个 Job 的串联，以完成一个完整的算法，例如迭代计算。

由于这样的弊端，催生了支持 DAG 框架的产生。因此，支持 DAG 的框架被划分为第二代计算引擎。如 Tez 以及更上层的 Oozie。这里我们不去细究各种 DAG 实现之间的区别，不过对于当时的 Tez 和 Oozie 来说，大多还是批处理的任务。

接下来就是以 Spark 为代表的第三代的计算引擎。第三代计算引擎的特点主要是 Job 内部的 DAG 支持（不跨越 Job），以及强调的实时计算。在这里，很多人也会认为第三代计算引擎也能够很好的运行批处理的 Job。

随着第三代计算引擎的出现，促进了上层应用快速发展，例如各种迭代计算的性能以及对流计算和 SQL 等的支持。Flink 的诞生就被归在了第四代。这应该主要表现在 Flink 对流计算的支持，以及更一步的实时性上面。当然 Flink 也可以支持 Batch 的任务，以及 DAG 的运算。

## **二. **Flink简介****

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设计为在所有常见的集群环境中运行，以内存速度和任何规模执行计算。

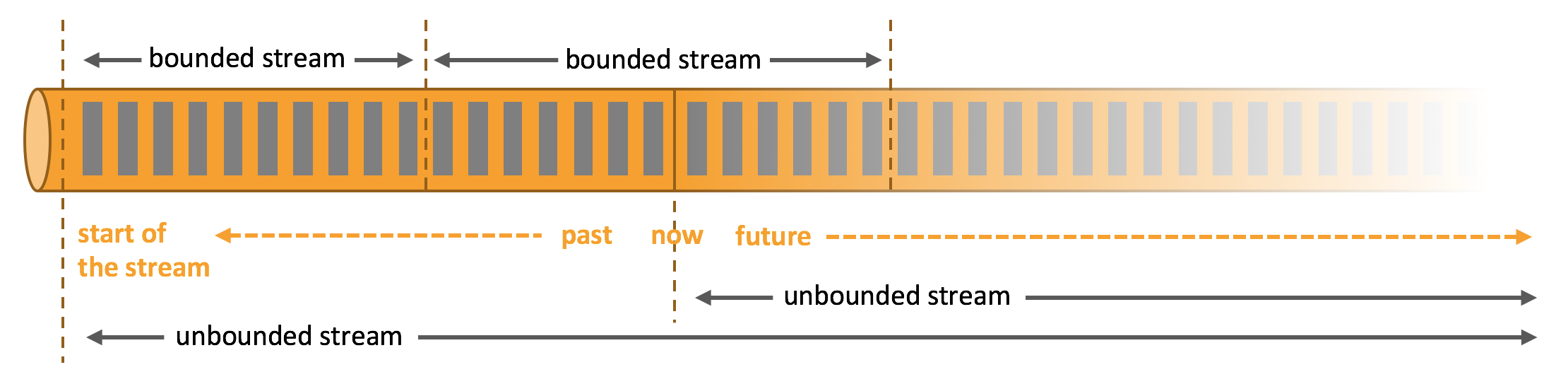
### **1.无界流和有界流**

任何类型的数据都是作为事件流产生的。信用卡交易，传感器测量，机器日志或网站或移动应用程序上的用户交互，所有这些数据都作为流生成。

数据可以作为无界或有界流处理。

无界流有一个开始但没有定义的结束。它们不会在生成时终止并提供数据。必须持续处理无界流，即必须在摄取事件后立即处理事件。无法等待所有输入数据到达，因为输入是无界的，并且在任何时间点都不会完成。处理无界数据通常要求以特定顺序（例如事件发生的顺序）摄取事件，以便能够推断结果完整性。

有界流具有定义的开始和结束。可以在执行任何计算之前通过摄取所有数据来处理有界流。处理有界流不需要有序摄取，因为可以始终对有界数据集进行排序。有界流的处理也称为批处理。



Apache Flink擅长处理无界和有界数据集。精确控制时间和状态使Flink的运行时能够在无界流上运行任何类型的应用程序。有界流由算法和数据结构内部处理，这些算法和数据结构专门针对固定大小的数据集而设计，从而产生出色的性能。

### **2.随处部署应用程序**

Apache Flink是一个分布式系统，需要计算资源才能执行应用程序。Flink与所有常见的集群资源管理器（如[Hadoop YARN](https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YARN.html)，[Apache Mesos](https://mesos.apache.org/)和[Kubernetes）集成，](https://kubernetes.io/)但也可以设置为作为独立集群运行。

Flink旨在很好地适用于之前列出的每个资源管理器。这是通过特定于资源管理器的部署模式实现的，这些模式允许Flink以其惯用的方式与每个资源管理器进行交互。

部署Flink应用程序时，Flink会根据应用程序配置的并行性自动识别所需资源，并从资源管理器请求它们。如果发生故障，Flink会通过请求新资源来替换发生故障的容器。提交或控制应用程序的所有通信都通过REST调用进行。这简化了Flink在许多环境中的集成。

### **3.以任何比例运行应用程序**

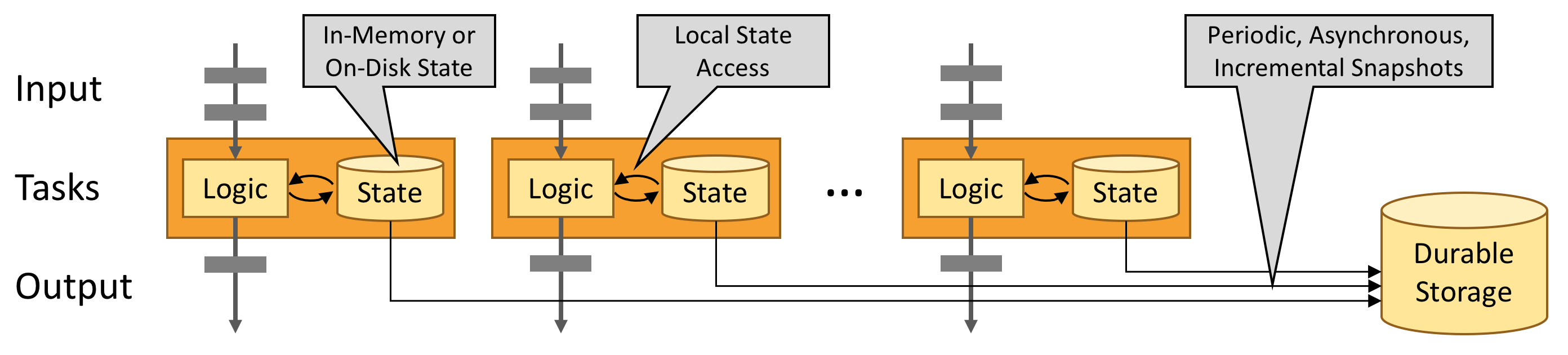
Flink旨在以任何规模运行有状态流应用程序。应用程序可以并行化为数千个在集群中分布和同时执行的任务。因此，应用程序可以利用几乎无限量的CPU，主内存，磁盘和网络IO。而且，Flink可以轻松维护非常大的应用程序状态。其异步和增量检查点算法确保对处理延迟的影响最小，同时保证一次性状态一致性。

[用户报告了](https://flink.apache.org/poweredby.html)在其生产环境中运行的Flink应用程序的[可扩展性数字令人印象深刻](https://flink.apache.org/poweredby.html)，例如

* 应用程序每天处理数万亿个事件，
* 应用程序维护多个TB的状态，以及
* 应用程序在数千个内核的运行。

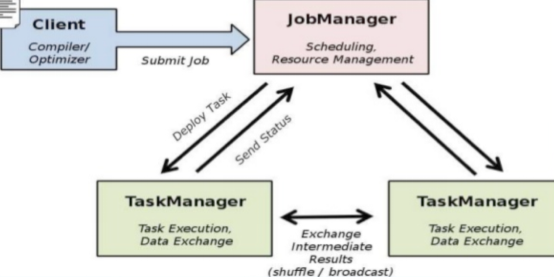
### **4.利用内存中的性能**

有状态Flink应用程序针对本地状态访问进行了优化。任务状态始终保留在内存中，或者，如果状态大小超过可用内存，则保存在访问高效的磁盘上数据结构中。因此，任务通过访问本地（通常是内存中）状态来执行所有计算，从而产生非常低的处理延迟。Flink通过定期和异步检查本地状态到持久存储来保证在出现故障时的一次状态一致性。



### **5.Flink的架构**

    Flink 可以支持本地的快速迭代，以及一些环形的迭代任务。并且 Flink 可以定制化内存管理。在这点，如果要对比 Flink 和 Spark 的话，Flink 并没有将内存完全交给应用层。这也是为什么 Spark 相对于 Flink，更容易出现 OOM 的原因（out of memory）。就框架本身与应用场景来说，Flink 更相似与 Storm。如果之前了解过 Storm 或者 Flume 的读者，可能会更容易理解 Flink 的架构和很多概念。下面让我们先来看下 Flink 的架构图。



我们可以了解到 Flink 几个最基础的概念，Client、JobManager 和 TaskManager。Client 用来提交任务给 JobManager，JobManager 分发任务给 TaskManager 去执行，然后 TaskManager 会心跳的汇报任务状态。看到这里，有的人应该已经有种回到 Hadoop 一代的错觉。确实，从架构图去看，JobManager 很像当年的 JobTracker，TaskManager 也很像当年的 TaskTracker。然而有一个最重要的区别就是 TaskManager 之间是是流（Stream）。其次，Hadoop 一代中，只有 Map 和 Reduce 之间的 Shuffle，而对 Flink 而言，可能是很多级，并且在 TaskManager 内部和 TaskManager 之间都会有数据传递，而不像 Hadoop，是固定的 Map 到 Reduce。

## **三. Flink**技术特点****

### **1. **流处理特性****

支持高吞吐、低延迟、高性能的流处理

支持带有事件时间的窗口（Window）操作

支持有状态计算的Exactly-once语义

支持高度灵活的窗口（Window）操作，支持基于time、count、session，以及data-driven的窗口操作

支持具有Backpressure功能的持续流模型

支持基于轻量级分布式快照（Snapshot）实现的容错

一个运行时同时支持Batch on Streaming处理和Streaming处理

Flink在JVM内部实现了自己的内存管理

支持迭代计算

支持程序自动优化：避免特定情况下Shuffle、排序等昂贵操作，中间结果有必要进行缓存

### **2. **API支持****

对Streaming数据类应用，提供DataStream API

对批处理类应用，提供DataSet API（支持Java/Scala）

### **3. **Libraries支持****

支持机器学习（FlinkML）

支持图分析（Gelly）

支持关系数据处理（Table）

支持复杂事件处理（CEP）

### **4. **整合支持****

支持Flink on YARN

支持HDFS

支持来自Kafka的输入数据

支持Apache HBase

支持Hadoop程序

支持Tachyon

支持ElasticSearch

支持RabbitMQ

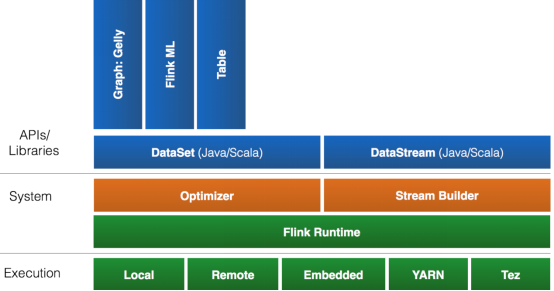
支持Apache Storm

支持S3

支持XtreemFS

### **5. **Flink生态圈****

Flink 首先支持了 Scala 和 Java 的 API，Python 也正在测试中。Flink 通过 Gelly 支持了图操作，还有机器学习的 FlinkML。Table 是一种接口化的 SQL 支持，也就是 API 支持，而不是文本化的 SQL 解析和执行。对于完整的 Stack 我们可以参考下图。



     Flink 为了更广泛的支持大数据的生态圈，其下也实现了很多 Connector 的子项目。最熟悉的，当然就是与 Hadoop HDFS 集成。其次，Flink 也宣布支持了 Tachyon、S3 以及 MapRFS。不过对于 Tachyon 以及 S3 的支持，都是通过 Hadoop HDFS 这层包装实现的，也就是说要使用 Tachyon 和 S3，就必须有 Hadoop，而且要更改 Hadoop 的配置（core-site.xml）。如果浏览 Flink 的代码目录，我们就会看到更多 Connector 项目，例如 Flume 和 Kafka。

## **四. Flink的编程模型**

Flink提供不同级别的抽象来开发流/批处理应用程序。

