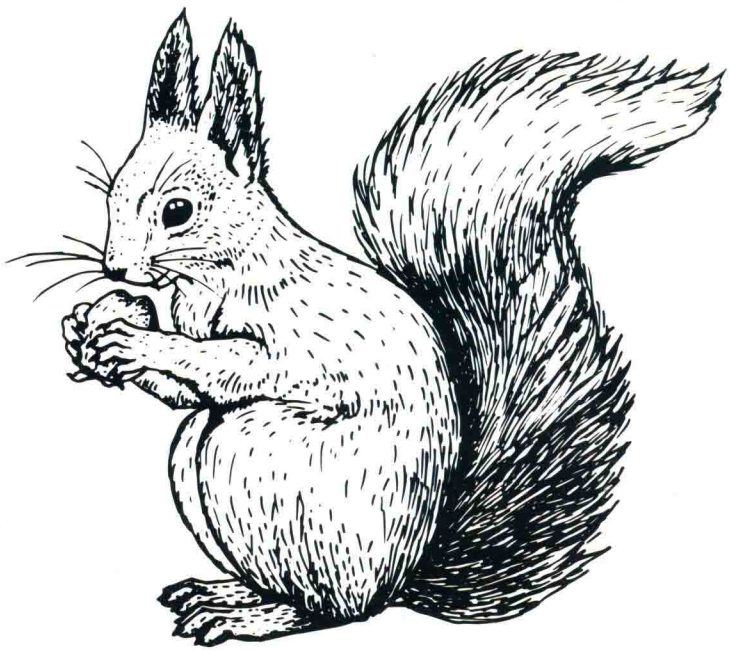
HZ BOOKS
华章IT

作者是资深架构师和流式计算专家，第四范式AI项目架构师，曾就职于明略数据

从功能、原理、实战和调优4个维度循序渐进讲解利用Flink进行分布式流式应用开发，指导读者从零基础入门到进阶

大数据
技术丛书



Principle, Practice and Performance Optimization about Flink

Flink原理、实战与性能优化

张利兵◎著

机械工业出版社
China Machine Press

大数据技术丛书

Flink原理、实战与性能优化

张利兵◎著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Flink 原理、实战与性能优化 / 张利兵著 . —北京：机械工业出版社，2019.4（2019.6 重印）
（大数据技术丛书）

ISBN 978-7-111-62353-3

I. F… II. 张… III. 数据处理软件 IV. TP274

中国版本图书馆 CIP 数据核字（2019）第 056258 号

Flink 原理、实战与性能优化

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：张锡鹏
责任校对：殷 虹
印 刷：北京诚信伟业印刷有限公司
版 次：2019 年 6 月第 1 版第 3 次印刷
开 本：186mm × 240mm 1/16
印 张：18
书 号：ISBN 978-7-111-62353-3
定 价：79.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426 88361066
投稿热线：（010）88379604
购书热线：（010）68326294
读者信箱：hzit@hzbook.com

Preface 前　言

为什么要写这本书

记得在几年前刚开始做流式计算相关的项目时，发觉项目对实时性和数据量的要求很高，无奈求助于 Flink 开源社区（后文简称“社区”），在社区中发现可以使用的流式框架有很多，例如比较主流的框架 Apache Storm、Spark Streaming 等，Apache Flink（简称“Flink”）也在其中。于是笔者开始对各种流式框架进行详细研究，最后发现能同时支持低延迟、高吞吐、Excactly-once 的框架只有 Apache Flink，从那时起笔者就对 Flink 这套框架充满兴趣，不管是其架构还是接口，都可以发现其中包含了非常优秀的设计思想。虽然当时 Flink 在社区的成熟度并不是很高，但笔者还是决定将 Flink 应用在自己的项目中，自此开启了 Flink 分布式计算技术应用之旅。

刚开始学习 Flink，对于没有分布式处理技术和流式计算经验的人来说会相对比较困难，因为其很难理解有状态计算、数据一致性保障等概念。尤其在相关中文资源比较匮乏的情况下，需要用户在官网以及国外的技术网站中翻阅大量的外文资料，这在一定程度上对学习和应用 Flink 造成了阻碍。笔者在 2018 年参加了一场由 Flink 中文社区组织的线下交流活动，当时听了很多领域内专家将 Flink 应用在不同业务场景中的分享，发现 Flink 这项技术虽然优秀，但是国内尚未有一本能够全面介绍 Flink 的中文书籍，于是笔者决定结合自己的实际项目经验来完成一本 Flink 中文书籍，以帮助他人学习和使用 Flink 这项优秀的分布式处理技术。

阿里巴巴在 2019 年 1 月开源了其内部 Flink 的分支项目 Blink，并推动社区将 Blink 中优秀的特性合并到 Flink 主干版本中，一时间 Flink 在国内的发展被推向了高潮，成为很多公司想去尝试使用的新技术。因此笔者相信未来会有更多的开发者参与到 Flink 社区中来，Flink 也将在未来的大数据生态中占据举足轻重的位置。

读者对象

本书从多个方面对 Flink 进行了深入介绍，包括原理、多种抽象接口的使用，以及 Flink 的性能监控与调优等方面，因此本书比较适合以下类型的读者。

- ❑ 流计算开发工程师
- ❑ 大数据架构工程师
- ❑ 大数据开发工程师
- ❑ 数据挖掘工程师
- ❑ 高校研究生以及高年级本科生

如何阅读本书

本书共分为 10 章，各章节间具有一定的先后关系，对于刚入门的读者，建议从第 1 章开始循序渐进地学习。

对于有一定经验的读者可以自行选择章节开始学习。如果想使用 Flink 开发流式应用，则可以直接阅读第 4 章、第 5 章，以及第 7 章之后的内容；如果想使用 Flink 开发批计算应用，则可以选择阅读第 5 章以及第 7 章之后的内容。

勘误和支持

除封面署名外，参加本书编写工作的还有：张再胜、尚越、程龙、姚远等。由于笔者水平有限，编写时间仓促，书中难免会出现一些错误或者不准确的地方，恳请读者批评指正。由于 Flink 技术的参考资料相对较少，因此书中有些地方参考了 Flink 官方文档，读者也可以结合 Flink 官网来学习。书中的全部源文件可以从 GitHub 网站下载，地址为 https://github.com/zhanglibing1990/learning-flink。同时笔者也会将相应的功能及时更新。如果你有更多宝贵的意见可以通过 QQ 群 686656574 或电子邮箱 zhanglibing1990@126.com 联系笔者，期待能够得到你们的真挚反馈。

致谢

在本书的写作过程中，得到了很多朋友及同事的帮助和支持，在此表示衷心感谢！

感谢我的女朋友，因为有你的支持，我才能坚持将本书顺利完成，谢谢你一直陪伴在我的身边，不断鼓励我前行。

感谢机械工业出版社华章公司的编辑杨福川和张锡鹏，在这半年多的时间中始终支持我的写作，你们的鼓励和帮助引导我顺利完成全部书稿。

谨以此书献给我最亲爱的家人，以及众多热爱 Flink 的朋友！

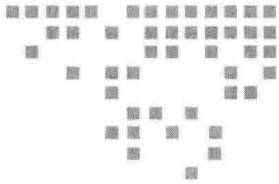
总结

本书最开始介绍 Flink 的发展历史，然后对 Flink 批数据和流数据的不同处理接口进行介绍，再对 Flink 的部署与实施、性能优化等方面进行全面讲解。经过系统完整地了解和学习 Flink 分布式处理技术之后，可以发现 Flink 有很多非常先进的概念，以及非常完善的接口设计，这些都能让用户更加有效地处理大数据，特别是流式数据处理。随着大数据技术的不断发展，Flink 也在大数据的浪潮中奋勇前行。越来越多的用户也参与到 Flink 社区的开发中，尤其是近年来随着阿里巴巴的推进，Blink 的开源在一定程度上推动了 Flink 在国内大规模的落地。相信在不久的将来，Flink 会逐渐成为国内乃至全球不可或缺的分布式处理引擎，笔者也相信 Flink 在流式数据处理领域会有新的突破，能够改变目前大部分基于批处理的模式，让分布式数据处理变得更加高效，使得数据处理成本不断降低。

张利兵

2019 年

目　录 *Contents*



第 1 章 Chapter 1

Apache Flink 介绍

本章对 Apache Flink 从多个方面进行介绍，让读者对 Flink 这项分布式处理技术能够有初步的了解。1.1 节主要介绍了 Flink 的由来及其发展历史，帮助读者从历史的角度了解 Flink 这项技术发展的过程。1.2 节重点介绍了 Flink 能够支持的各种实际业务场景、Flink 所具备的主要特性、Flink 组成部分及其基本概念等内容，最后在 1.4 节中介绍了 Flink 的基本架构以及主要组成部分。

1.1 Apache Flink 是什么

在当前数据量激增的时代，各种业务场景都有大量的业务数据产生，对于这些不断产生的数据应该如何进行有效的处理，成为当下大多数公司所面临的问题。随着雅虎对 Hadoop 的开源，越来越多的大数据处理技术开始涌入人们的视线，例如目前比较流行的大数据处理引擎 Apache Spark，基本上已经取代了 MapReduce 成为当前大数据处理的标准。但随着数据的不断增长，新技术的不断发展，人们逐渐意识到对实时数据处理的重要性。相对于传统的数据处理模式，流式数据处理有着更高的处理效率和成本控制能力。Apache Flink 就是近年来在开源社区不断发展的技术中的能够同时支持高吞吐、低延迟、高性能的分布式处理框架。

在 2010 年至 2014 年间，由柏林工业大学、柏林洪堡大学和哈索普拉特纳研究所联合发起名为“Stratosphere: Information Management on the Cloud”研究项目，该项目在当时的社区逐渐具有了一定的社区知名度。2014 年 4 月，Stratosphere 代码被贡献给 Apache 软件基金会，成为 Apache 基金会孵化器项目。初期参与该项目的核心成员均是 Stratosphere 曾经的核心成员，之后团队的大部分创始成员离开学校，共同创办了一家名叫 Data Artisans 的公司，其主要业务便是将 Stratosphere，也就是之后的 Flink 实现商业化。在项目孵化期间，项目 Stratosphere 改名为 Flink。Flink 在德语中是快速和灵敏的意思，用来体现流式数据处理器速度快和灵活性强等特点，同时使用棕红色松鼠图案作为 Flink 项目的 Logo，也是为了突出松鼠灵活快速的特点，由此，Flink 正式进入社区开发者的视线。

2014 年 12 月，该项目成为 Apache 软件基金会顶级项目，从 2015 年 9 月发布第一个稳定版本 0.9，到目前撰写本书期间已经发布到 1.7 的版本，更多的社区开发成员逐步加入，现在 Flink 在全球范围内拥有 350 多位开发人员，不断有新的特性发布。同时在全球范围内，越来越多的公司开始使用 Flink，在国内比较出名的互联网公司如阿里巴巴、美团、滴滴等，都在大规模使用 Flink 作为企业的分布式大数据处理引擎。

Flink 近年来逐步被人们所熟知，不仅是因为 Flink 提供同时支持高吞吐、低延迟和 exactly-once 语义的实时计算能力，同时 Flink 还提供了基于流式计算引擎处理批量数据的计算能力，真正意义上实现了批流统一，同时随着阿里对 Blink 的开源，极大地增强了 Flink 对批计算领域的支持。众多优秀的特性，使得 Flink 成为开源大数据数据处理框架中的一颗新星，随着国内社区不断推动，越来越多的国内公司开始选择使用 Flink 作为实时数据处理技术。在不久的将来，Flink 也将会成为企业内部主流的数据处理框架，最终成为下一代大数据处理的标准。

1.2 数据架构的演变

近年来随着开源社区的发展，越来越多新的技术被开源，例如雅虎的 Hadoop 分布式计算框架、UC 伯克利分校的 Apache Spark 等，而伴随着这些技术的发展，促使着企业数据架构的演进，从传统的关系型数据存储架构，逐步演化为分布式处理和存储的架构。

1.2.1　传统数据基础架构

如图 1-1 所示，传统单体数据架构（Monolithic Architecture）最大的特点便是集中式数据存储，企业内部可能有诸多的系统，例如 Web 业务系统、订单系统、CRM 系统、ERP 系统、监控系统等，这些系统的事务性数据主要基于集中式的关系性数据库（DBMS）实现存储，大多数将架构分为计算层和存储层。存储层负责企业内系统的数据访问，且具有最终数据一致性保障。这些数据反映了当前的业务状态，例如系统的订单交易量、网站的活跃用户数、每个用户的交易额变化等，所有的更新操作均需要借助于同一套数据库实现。

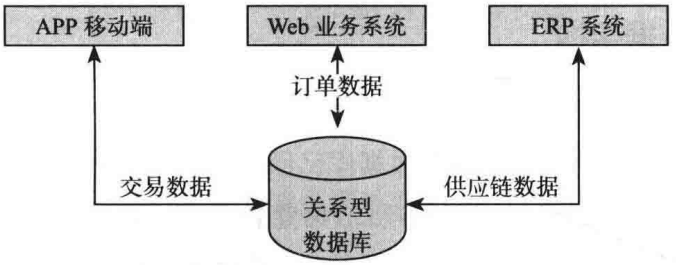


图 1-1　传统数据结构

单体架构的初期效率很高，但是随着时间的推移，业务越来越多，系统逐渐变得很大，越来越难以维护和升级，数据库是唯一的准确数据源，每个应用都需要访问数据库来获取对应的数据，如果数据库发生改变或者出现问题，则将对整个业务系统产生影响。

后来随着微服务架构（Microservices Architecture）的出现，企业开始逐渐采用微服务作为企业业务系统的架构体系。微服务架构的核心思想是，一个应用是由多个小的、相互独立的微服务组成，这些服务运行在自己的进程中，开发和发布都没有依赖。不同的服务能依据不同的业务需求，构建的不同的技术架构之上，能够聚焦在有限的业务功能。

如图 1-2 所示，微服务架构将系统拆解成不同的独立服务模块，每个模块分别使用各自独立的数据库，这种模式解决了业务系统拓展的问题，但是也带来了新的问题，那就是业务交易数据过于分散在不同的系统中，很难将数据进行集中化管理，对于企业内部进行数据分析或者数据挖掘之类的应用，则需要通过从不同的数据库中进行数据抽取，将数据从数据库中周期性地同步到数据仓库中，然后在数据仓库中进行数据的抽取、转换、加载（ETL），从而构建成不同的数据集市和应用，提供给业务系统使用。

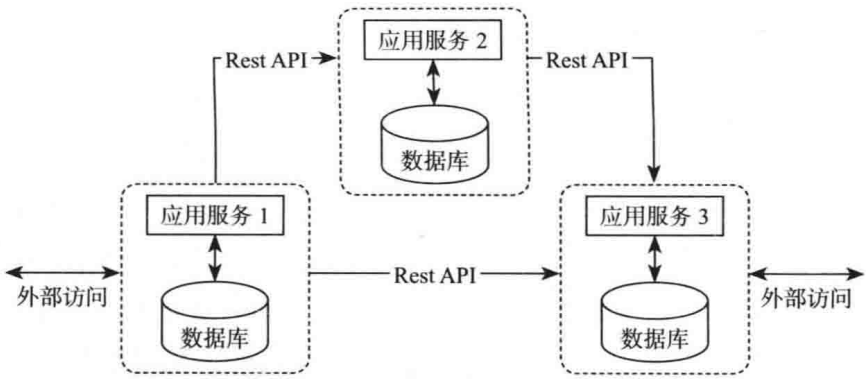


图 1-2 微服务架构

1.2.2 大数据数据架构

起初数据仓库主要还是构建在关系型数据库之上，例如 Oracle、Mysql 等数据库，但是随着企业数据量的增长，关系型数据库已经无法支撑大规模数据集的存储和分析，因此越来越多的企业开始选择基于 Hadoop 构建企业级大数据平台。同时众多 Sql-On-Hadoop 技术方案的提出，也让企业在 Hadoop 上构建不同类型的数据应用变得简单而高效，例如通过使用 Apache Hive 进行数据 ETL 处理，通过使用 Apache Impala 进行实时交互性查询等。

大数据技术的兴起，让企业能够更加灵活高效地使用自己的业务数据，从数据中提取出更多重要的价值，并将数据分析和挖掘出来的结果应用在企业的决策、营销、管理等应用领域。但不可避免的是，随着越来越多新技术的引入与使用，企业内部一套大数据管理平台可能会借助众多开源技术组件实现。例如在构建企业数据仓库的过程中，数据往往都是周期性的从业务系统中同步到大数据平台，完成一系列 ETL 转换动作之后，最终形成数据集市等应用。但是对于一些时间要求比较高的应用，例如实时报表统计，则必须有非常低的延时展示统计结果，为此业界提出一套 Lambda 架构方案来处理不同类型的数据。如图 1-3 所示，大数据平台中包含批量计算的 Batch Layer 和实时计算的 Speed Layer，通过在一套平台中将批计算和流计算整合在一起，例如使用 Hadoop MapReduce 进行批量数据的处理，使用 Apache Storm 进行实时数据的处理。这种架构在一定程度上解决了不同计算类型的问题，但是带来的问题是框架太多会导致平台复杂度过高、运维成本高等。在一套资源管理平台中管理不同类型的计算框架使用也是非常困难的事情。总而言之，Lambda 架构是构建大数据应用程序的一种很有效的解决方案，但

是还不是最完美的方案。

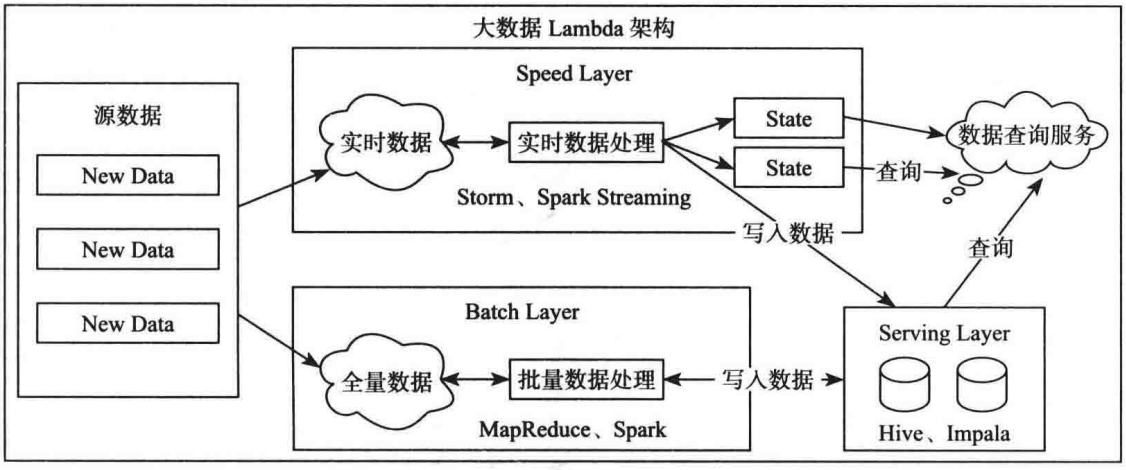


图 1-3　大数据 Lambda 架构

后来随着 Apache Spark 的分布式内存处理框架的出现，提出了将数据切分成微批的处理模式进行流式数据处理，从而能够在一套计算框架内完成批量计算和流式计算。但因为 Spark 本身是基于批处理模式的原因，并不能完美且高效地处理原生的数据流，因此对流式计算支持的相对较弱，可以说 Spark 的出现本质上是在一定程度上对 Hadoop 架构进行了一定的升级和优化。

1.2.3　有状态流计算架构

数据产生的本质，其实是一条条真实存在的事件，前面提到的不同的架构其实都是在一定程度违背了这种本质，需要通过在一定时延的情况下对业务数据进行处理，然后得到基于业务数据统计的准确结果。实际上，基于流式计算技术局限性，我们很难在数据产生的过程中进行计算并直接产生统计结果，因为这不仅对系统有非常高的要求，还必须要满足高性能、高吞吐、低延时等众多目标。而有状态流计算架构（如图 1-4 所示）的提出，从一定程度上满足了企业的这种需求，企业基于实时的流式数据，维护所有计算过程的状态，所谓状态就是计算过程中产生的中间计算结果，每次计算新的数据进入到流式系统中都是基于中间状态结果的基础上进行运算，最终产生正确的统计结果。基于有状态计算的方式最大的优势是不需要将原始数据重新从外部存储中拿出来，从而进行全量计算，因为这种计算方式的代价可能是非常高的。从另一个角度讲，用户无须通

过调度和协调各种批量计算工具，从数据仓库中获取数据统计结果，然后再落地存储，这些操作全部都可以基于流式计算完成，可以极大地减轻系统对其他框架的依赖，减少数据计算过程中的时间损耗以及硬件存储。

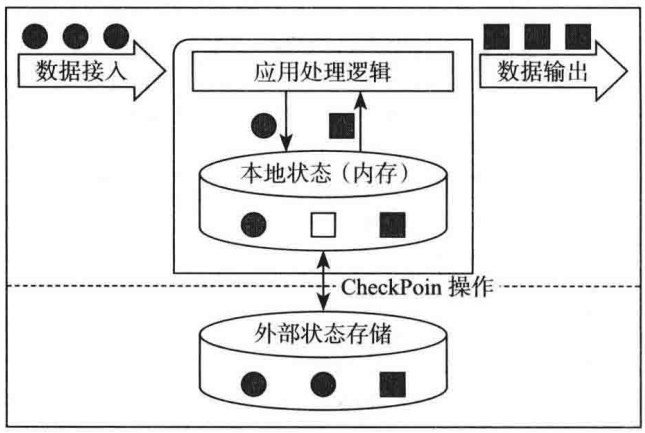


图 1-4 有状态计算架构

如果计算的结果能保持一致，实时计算在很短的时间内统计出结果，批量计算则需要等待一定时间才能得出，相信大多数用户会更加倾向于选择使用有状态流进行大数据处理。

1.2.4 为什么会是 Flink

可以看出有状态流计算将会逐步成为企业作为构建数据平台的架构模式，而目前从社区来看，能够满足的只有 Apache Flink。Flink 通过实现 Google Dataflow 流式计算模型实现了高吞吐、低延迟、高性能兼具实时流式计算框架。同时 Flink 支持高度容错的状态管理，防止状态在计算过程中因为系统异常而出现丢失，Flink 周期性地通过分布式快照技术 Checkpoints 实现状态的持久化维护，使得即使在系统停机或者异常的情况下都能计算出正确的结果。

Flink 具有先进的架构理念、诸多的优秀特性，以及完善的编程接口，而 Flink 也在每一次的 Release 版本中，不断推出新的特性，例如 Queryable State 功能的提出，容许用户通过远程的方式直接获取流式计算任务的状态信息，数据不需要落地数据库就能直接从 Flink 流式应用中查询。对于实时交互式的查询业务可以直接从 Flink 的状态中查询最新的结果。在未来，Flink 将不仅作为实时流式处理的框架，更多的可能会成为一套实

时的状态存储引擎，让更多的用户从有状态计算的技术中获益。

Flink 的具体优势有以下几点。

（1）同时支持高吞吐、低延迟、高性能

Flink 是目前开源社区中唯一一套集高吞吐、低延迟、高性能三者于一身的分布式流式数据处理框架。像 Apache Spark 也只能兼顾高吞吐和高性能特性，主要因为在 Spark Streaming 流式计算中无法做到低延迟保障；而流式计算框架 Apache Storm 只能支持低延迟和高性能特性，但是无法满足高吞吐的要求。而满足高吞吐、低延迟、高性能这三个目标对分布式流式计算框架来说是非常重要的。

（2）支持事件时间（Event Time）概念

在流式计算领域中，窗口计算的地位举足轻重，但目前大多数框架窗口计算采用的都是系统时间（Process Time），也是事件传输到计算框架处理时，系统主机的当前时间。Flink 能够支持基于事件时间（Event Time）语义进行窗口计算，也就是使用事件产生的时间，这种基于事件驱动的机制使得事件即使乱序到达，流系统也能够计算出精确的结果，保持了事件原本产生时的时序性，尽可能避免网络传输或硬件系统的影响。

（3）支持有状态计算

Flink 在 1.4 版本中实现了状态管理，所谓状态就是在流式计算过程中将算子的中间结果数据保存在内存或者文件系统中，等下一个事件进入算子后可以从之前的状态中获取中间结果中计算当前的结果，从而无须每次都基于全部的原始数据来统计结果，这种方式极大地提升了系统的性能，并降低了数据计算过程的资源消耗。对于数据量大且运算逻辑非常复杂的流式计算场景，有状态计算发挥了非常重要的作用。

（4）支持高度灵活的窗口（Window）操作

在流处理应用中，数据是连续不断的，需要通过窗口的方式对流数据进行一定范围的聚合计算，例如统计在过去的 1 分钟内有多少用户点击某一网页，在这种情况下，我们必须定义一个窗口，用来收集最近一分钟内的数据，并对这个窗口内的数据进行再计算。Flink 将窗口划分为基于 Time、Count、Session，以及 Data-driven 等类型的窗口操作，窗口可以用灵活的触发条件定制化来达到对复杂的流传输模式的支持，用户可以定义不同的窗口触发机制来满足不同的需求。

（5）基于轻量级分布式快照（Snapshot）实现的容错

Flink 能够分布式运行在上千个节点上，将一个大型计算任务的流程拆解成小的计算过程，然后将 task 分布到并行节点上进行处理。在任务执行过程中，能够自动发现事件