面向大数据处理的数据流编程模型和工具综述

邹骁锋，阳王东，容学成，李肯立，李克勤
湖南大学信息科学与工程学院，湖南 长沙 410008

摘要
利用大数据计算平台对大量的静态数据进行数据挖掘和智能分析助推了大数据和人工智能应用的落地。在面临互联网、物联网产生的日益庞大的实时动态数据的处理需求时，数据流计算被逐步引入目前的一些大数据处理平台中。针对数据流的编程模型，比较了传统软件工程的面向数据流的分析和设计方法与目前针对大数据处理平台的数据流编程模型提供的结构定义和模型参考，分析了两者的差异和不足，总结了数据流编程模型的主要特征和关键要素。分析了目前数据流编程的主要方式以及与主流编程工具的结合，针对大数据处理的数据流计算业务需求，给出了可视化数据流编程工具的基本框架和编程模式。

关键词
数据流；编程模型；大数据处理；编程工具

中图分类号：TP391 文献标识码：A doi: 10.11959/j.issn.2096-0271.2020024

A survey of dataflow programming models and tools for big data processing

ZOU Xiaofeng, YANG Wangdong, RONG Xuecheng, LI Kenli, LI Keqin
College of Computer Science and Electronic Engineering, Hunan University, Changsha 410008, China

Abstract
The application of big data and artificial intelligence is promoted by data mining and intelligent analysis of a large number of static data using big data computing platform. In the face of the growing demand for real-time dynamic data processing generated by the Internet of things, dataflow computing has been gradually introduced into some big data processing platforms. Aiming at the programming model of data flow, the traditional software engineering design method for dataflow analysis and the structure definition and model reference provided by the current dataflow programming model for big data processing platform was compared, the differences and shortcomings were analyzed, and the main features and key elements of the dataflow programming model were summarized. The main methods of dataflow programming and the combination with the mainstream programming tools were analyzed, and the basic framework and programming mode of visual dataflow programming tools were presented according to the dataflow computing business requirements of big data processing.

Key words
data flow, programming model, big data processing, programming tool

1 引言

自21世纪以来，移动互联网、物联网和云计算等新的信息化技术被广泛应用，这些新兴的技术在应用过程中产生了海量的数据。随着海量数据的产生，处理海量数据的新的大数据处理技术被广泛研究，在大型计算机集群上使用分布式并行计算技术构建的分布式大数据处理平台也得到快速发展，从最初的Hadoop[1]及其生态系统发展到基于内存计算的Flink[2]、Storm[3]、Spark[4]等。基于传统的数据库技术，对大数据离线批处理分析的研究较多，相关应用也较为成熟。但是随着实时产生的数据增多，对流式数据（如视频）进行实时性分析的需求也越来越普遍，这些实时性强的应用领域的海量数据规模对目前的大数据处理平台提出了极大的挑战。经典的应用场景包括网站日志查询、城市实时监控、物联网传感器网络、自动化运维系统的异常检测等。在这些场景下，数据流持续不断地产生，并以大量、快速、时变的方式到达系统，需要系统快速可靠地进行处理。例如，一个城市交通检测系统中包含大量的摄像头，它们分布在不同的路口，每个摄像头收集当前路口的交通信息，持续地产生数据流，并将数据流发送到计算机系统进行处理。接收到数据流的计算机系统对数据流进行分析，进而监控城市道路的实时交通状态，以便最快地基于状态采取相应行动。因此，高通量、低时延是实时数据流计算系统的核心指标。

目前数据流计算采用的是数据流模型（dataflow model）。数据流模型将整个计算任务抽象为数据流图，以数据驱动的方式处理计算，以数据流为中心实现业务的处理过程。数据流计算模式将需要处理的数据分配到计算资源上，实现数据的计算与通信分离，并通过数据的到达来激发计算任务的调度和资源的分配，利用流水线的并行特性充分地挖掘数据流处理中潜在的并行性，进而充分发挥计算资源的性能，并提高资源间负载的均衡性。用户面向数据流描述数据计算的处理逻辑，在程序执行过程中，系统会通过数据触发机制，自发地处理在计算过程中由数据依赖引发的计算顺序问题，在编程过程中减少了因为数据依赖问题而产生的同步和阻塞操作，降低了并行编程的难度。在分布式并行编程领域，程序员使用数据流编程模型，不需要对底层的分布式系统有深入的了解和控制，只需把重点放在领域应用业务流程描述上，关注数据流程的处理过程即可，真正做到了面向领域编程。

本文首先比较了传统软件工程中面向数据流的分析和设计方法；随后详细地描述了几种目前大数据处理平台提供的数据流编程模型的结构定义和模型参考，并分析了两者的差异和不足，总结了数据流编程模型的主要特征和关键要素；最后分析了目前数据流编程的主要方式以及与主流编程工具的结合，针对大数据处理的数据流计算业务需求，给出了可视化数据流编程工具的基本框架和编程模式。

2 传统软件工程中面向数据流的分析和设计方法

软件工程中传统的结构化设计（structured design，SD）提供了一种面向数据流的设计方法[5]，该方法提供了针对业务需求的逻辑模型处理数据流的描述方式，并能够根据数据流处理的逻辑模型导出系统的软件模块结构。在软件开发过程中，面向数据流方法将需求分析阶段生

成的数据流图(data flow diagram, DFD)映射成表达软件系统结构的软件模块结构图。在面向数据流的分析方法中,数据流图用于描述系统中信息的处理加工和流动情况。在DFD中,系统的输入数据流经过一系列的变换最终成为系统的输出数据流,在这个过程中流动的就是信息流。DFD能够从业务的需求层面描述信息处理的逻辑模型,但是不能描述系统的执行模式。

2.1 传统软件工程中面向数据流的概念

面向数据流的分析和设计方法,数据流可划分为以下2种类型[6]。

(1)变换流

数据通过输入通路进入系统,进入系统的数据流在变换中心加工处理后变换成另一种数据流,再通过输出通路输出。具有这些特征的数据流被称为变换流。针对变换流的DFD通常由3个部分组成:输入、变换(加工处理)、输出。

(2)事务流

数据通过输入通路到达某一个处理步骤,在处理期间,系统会判定输入数据的类型,选择某个动作序列执行。这种类型的数据流被称为事务流,对事务流的处理单元叫作事务中心,事务流的处理路径从事务中心呈辐射状流出。事务流的DFD主要由以下3个部分组成:输入通路(输入的数据称为事务)、事务判定、根据事务类型选取一条执行路径。

2.2 数据流图

面向数据流的分析和设计方法中的DFD是描述系统中数据流的处理过程的一种图形化工具,它体现了一个系统把业务输入转换为业务输出所需的数据流加工处理过程,DFD的组成要素如**图1**所示,包括数据源点和数据汇点、数据流、数据加工或处理、数据存储[7]。利用DFD描述基于公式的即时家教系统的数据流处理过程如**图2**所示。

3 数据流模型

数据流模型(与冯·诺依曼模型的结构不同)于20世纪60年代末由麻省理工学院的Dennis团队提出[8]。数据流模型将整个计算任务抽象为一张数据流图,针对数据流的计算任务,根据数据流的处理过程和流向,其被划分为一系列细粒度的计算单元,数据流图可以采用有向无环图(directed acyclic graph, DAG)[9]描述。如**图3**所示,在数据流图中,节点表示计算单元,边表示节点之间的数据依赖关系,数据(即token)通过边从一个节点流向与之相连的下游节点。当某个计算单元的输入数据准备就绪,同时所需的计算资源也空闲时,该计算单元就会进入激活状态,就可以在运行时(runtime)被执行。数据流模型提供了天然的可并行和并发处理模式,计算单元之间可以采用异步执行的通信方式,并根据资源情况动态地调度计算任务,有效地解决了计算资源之间负载平衡的问题[10-11]。

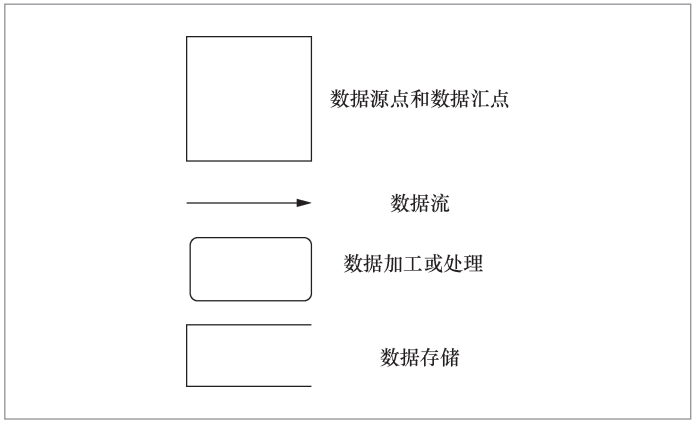

图1 DFD的组成要素

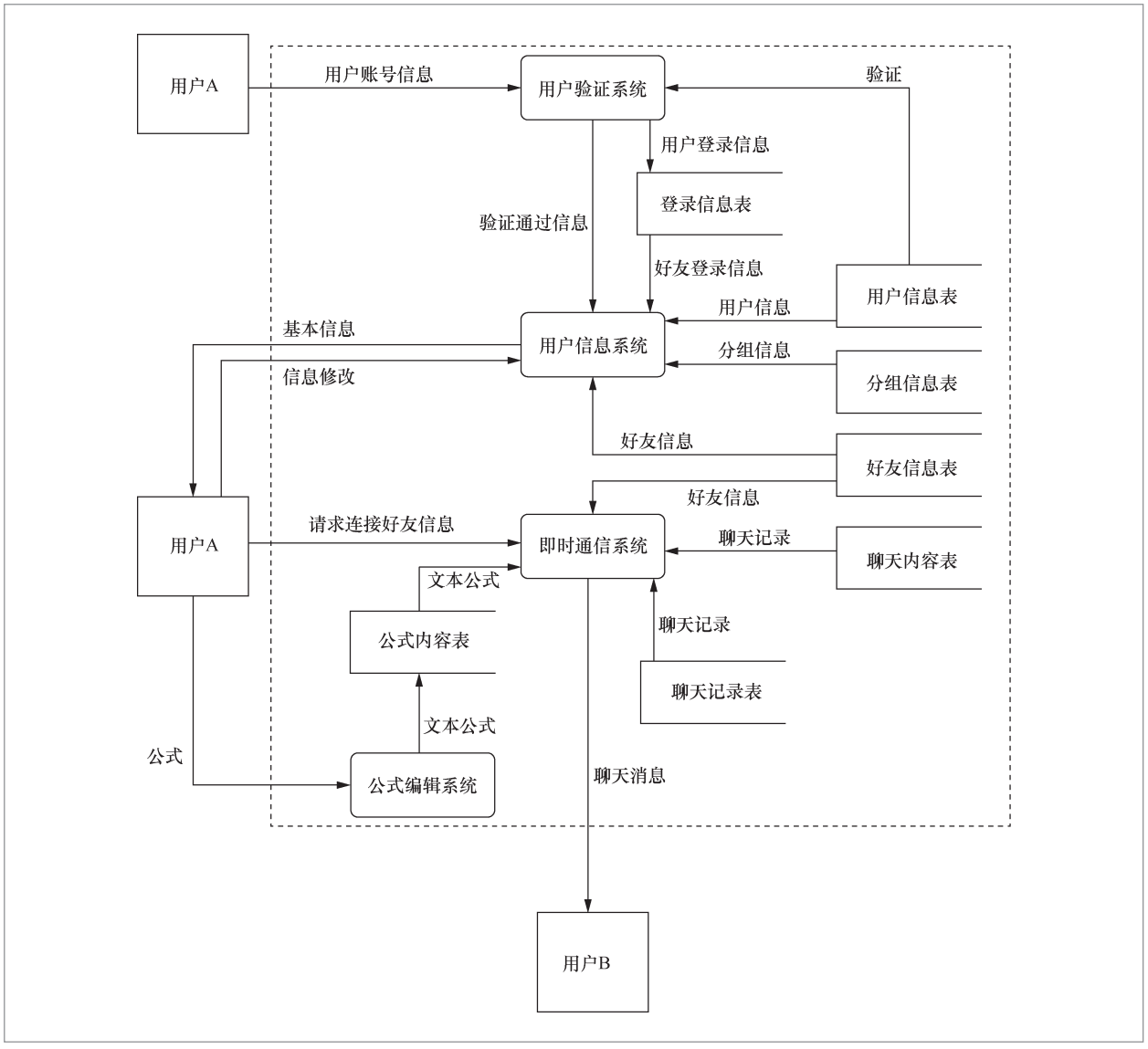

图 2 基于公式的即时家教系统的数据流处理过程

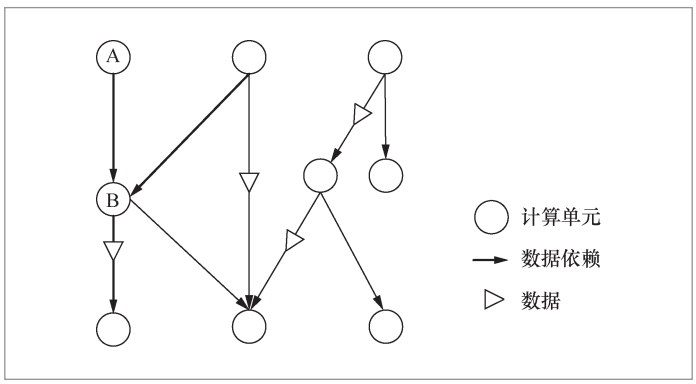

图 3 数据流图

根据对数据的处理方式，数据流模型可以分为2种：静态数据流模型和动态数据流模型。静态数据流模型的计算单元处理的数据集合是受到限制的，因此可以在编译期间对数据的处理进行优化；动态数据流模型的计算单元处理的数据集合是没有限制的。

数据流模型可以分为2个层面：一个层面是数据流的编程模型，其面向应用需求描述数据流的处理逻辑；另一个层

面是数据流的执行模型，其面向执行环境描述数据流的计算过程。数据流编程模型能够利用数据流处理的并行性以及数据流流向的单一性，大大简化编程人员的编程任务。

近几年，国内外研究人员对数据流编程模型、语言及工具进行了一系列研究。Li A等人[12]提出了一种在GPU上细粒度并行的数据流编程模型。Halbwachs N等人[13]提出了一种同步数据流编程语言LUSTER。苏志超等人[14]在神威•太湖之光超级计算机上设计了一种能高效地利用GPU片上计算资源的方法，有效地解决了数据流执行的并行计算问题，这种基于数据流的编程模型被称为SunwayFlow。针对异构并行计算机集群硬件平台存在多级并行结构的问题，杨瑞瑞等人[15]基于数据流应用程序和CPU/GPU异构计算特性，设计并实现了一个面向CPU-GPU异构协同的数据流编程模型。为了提高面向数据流的应用程序开发的可编程性，张维维等人[16]进一步提出了一种新的数据流编程模型——COStream，它提供了数据流编程语言和编译工具，大大降低了并行编程的难度。

4 基于数据流的编程模型

数据流编程语言以数据为核心，对施加在数据流上的面向业务领域的数据处理功能模块进行定义，把数据流的传递流动与数据流的处理进行分离，充分利用数据流的天然并行性，发挥数据流模型的并行性。数据流编程模型是专门针对流处理器设计的编程模型，它以数据流程序语言为基础，能清晰地描述数据流程序的业务逻辑，并针对分布式并行环境描述其业务程序的执行模式。目前主要有Apache Beam[17]、SWARM[18]、StreamIt[19]、COStream[16]、TensorFlow[20]等数据流编程模型。

4.1 Apache Beam

Beam是 Apache软件基金会（Apache Software Foundation，ASF）的项目。2017年5月17日ASF发布了其第一个稳定版2.0.0。目前的最新版本为2.16.0。Beam项目主要对数据流处理（包含有界数据集和无界数据集）的编程范式和接口进行了统一定义。基于Beam开发的数据流处理程序可以在多种分布式计算引擎上执行。Beam的架构如**图4**所示。

Beam编程模型主要由以下3个部分构成。

（1）Modes

Modes是Beam的模型，也是数据来源的I/O，由多种数据源或仓库的I/O组成，数据源支持批处理和流处理。

（2）Pipeline

Pipeline是Beam的管道，这个管道现在是唯一的。管道可以看成数据流的传递和存储通道，它的作用是连接数据和Runtimes平台。所有的批处理或流处理都

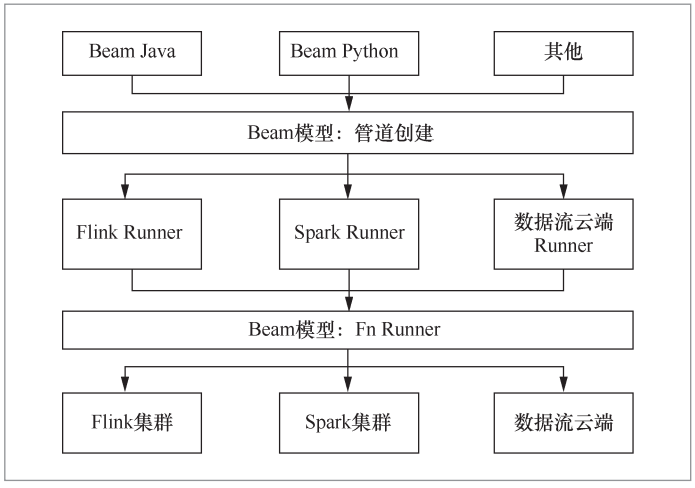

图 4 Beam 的架构

要通过这个管道把数据传输到后端的计算平台。管道可以连接多种数据源，也可以把数据流传递给不同的计算平台。

（3）Runtimes

Runtimes是大数据计算或处理平台，目前支持Direct Pipeline、Apache Spark、Apache Flink和Google Cloud Dataflow 4种大数据框架。其中Direct Pipeline仅支持本地，Apache Spark和Apache Flink同时支持本地和云端。Google Cloud Dataflow仅支持云端。

Beam提供了以下2个数据流编程的组件。

（1）Beam SDK

Beam SDK定义了提供一个统一的编程接口给上层应用的开发者，开发者可以利用提供的API开发分布式数据流处理的业务逻辑。开发者可以直接通过Beam SDK的接口开发数据流加工处理程序，不需要了解底层具体的大数据平台开发接口。Beam SDK对批处理的有界数据集和流处理的无界数据集都使用相同的类，并且使用相同的转换操作进行处理。

（2）Beam Pipeline Runner

Beam Pipeline Runner对用Beam SDK编写的数据流处理程序进行编译，并将其转换为具体大数据计算平台上的可执行的代码。从编程模式上来说，Beam分为3个部分：第一部分是利用Beam模型构建数据处理管道；第二部分是利用Beam SDK实现管道中数据处理的逻辑；第三部分是把包含数据处理逻辑的数据管道通过Beam Pipeline Runner编译成可在具体计算平台上执行的程序，在编译时，需要制定可执行的计算平台。最后把编译好的程序部署到分布式计算引擎上运行。

利用Apache Beam实现对输入的字符串数据流的单词计数，如**图5**所示。数据流计算流程如下。

- 格式化输入的文本数据。
- 将文本行转换成单个单词。
- 统计每个单词出现的次数。
- 格式化输出单词计数的结果。

4.2 SWARM

SWARM是一个运行时系统，其核心的执行模型是基于Codelet[21]的动态数据流模型[18]。它的目的是允许应用程序在单核、多核或众核计算机上良好运行，并且

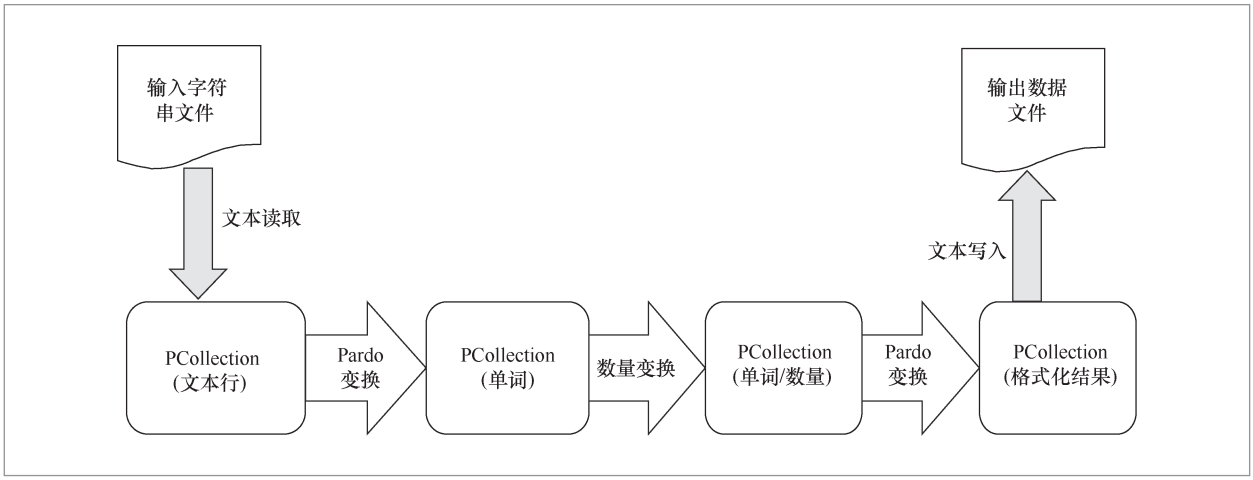

图 5 基于 Apache Beam 的单词计数数据流图

允许应用程序在计算集群或广域网之间以及不同类型的计算硬件之间透明地迁移。这将允许应用程序更轻松、更广泛地扩展，并使直接扩展软件路径成为可能。SWARM的模型适用于共享存储和分布式存储的并行计算机系统。

SWARM运行时系统采用数据流模型Codelet作为最基本的执行和调度单元。一个Codelet由以下4个部分组成。

- run fork，描述将被执行的任务，并推进程序执行的状态。
- cancelfork，描述错误发生时，回退程序的状态的方式。
- 描述上下文环境的类型，存储codelet的状态等信息。
- 输入数据的类型描述。

当创建一个Codelet实例时，会将该Codelet与上下文（context）框架关联。在给runfork提供输入数据或者在给cancalfork提供错误数据时，Codelet实例会进入就绪（enabeld）状态，能够被调度器调度。调度器选中准备就绪的Codelet，并分发给计算单元执行，此时Codelet的状态转换为激活（active）状态。Codelet在执行期间处于激活状态时，该执行进程不会被阻塞，直至执行结束，结束后Codelet就进入完成（completed）状态。

Codelet模型的第一个官方实现的运行时系统是SWARM，它为后续关于Codelet模型的实现等研究工作提供了很好的启示和参考意义。但SWARM还不成熟，运行时系统的适应性只是一个较为初步的、实验性的工作，其作为支撑数据流编程模型的实现还不完善，需要进一步研究。

4.3 StreamIt

StreamIt以Java语言程序为基础，根据同步数据流（synchronous data flow，SDF）模型进行数据流扩展，利用管道（pipeline）、拼接（splitjoin）和反馈循环（feedback loop）3种层次性的结构帮助编程人员对业务应用进行并行抽象。其最初是麻省理工学院针对RAW处理器开发的一种编程模型，后来延伸到数据流编程领域。

SDF模型由计算任务节点和边构成，其中数据流计算任务节点的最基本单元是actor，边表示计算任务节点之间的数据流动，在边上设置2个权值参数表示输入流和输出流的速率。在SDF模型中，actor分为2类：有状态和无状态。有状态的actor需要保存执行状态参数，以便为下次执行提供参数；无状态的actor则不需要保存执行状态参数。计算任务节点actor采用数据驱动的方式执行，当足够的数据到达输入边时，actor会被激活，并执行生成数据到输出边。StreamIt模型针对SDF模型提供了基本运算单元（filter）和核心模块（work），其中filter与SDF中的actor对应，work函数促使filter中计算任务的实现。StreamIt模型针对特定的处理器以及SDF模型中计算和通信隔离的机制，挖掘应用程序的深度并行性，在基于数据流的处理过程中构建面向数据流的编程模型，支持高级语言Java目标代码，并提供一定的并行化机制。

StreamIt模型是一种天然的并行编程模型，在程序设计过程中蕴含一些瓶颈问题。filter有时执行的代码量很小，但是数据传输量很大，导致程序性能降低。有状态的filter是串行执行的，即前面的filter没有执行完成时，后面的filter无法工作；在实际程序设计中，设计人员设计加入了过多无用的filter，造成过长的串行流水线结构，影响系统的整体效率。当某个filter中的具有高度并行性的语句经过普通处理器执行后仍然得不到很高的加速，经过编译后还存在大量的并行循环执行语句时，这

些语句中通常包含大量的浮点计算指令，往往会花费大量运行时间，而它们的运行仍然是串行方式。SteamIt模型为了提高并行度，会结合处理器数目将若干并行任务划分为相应区域，实际的问题是过多的划分将导致额外开销的加大，而过小的划分则会增加执行时间。

针对这些问题，StreamIt模型的研究围绕精简Pipeline、局部自动并行化、串并行自动拆分、多进多出filter等技术进行开展。目前，对于filter节点输入与输出端口来说，可以是一对一、一对多或多对一的。不过对于不同类型的数据流程序来说，节点为多进多出更符合程序固有模式。

4.4 COStream

COStream是一种层次型数据流编程模型，将SDF模型作为执行模型，利用DAG描述应用处理过程。COStream主要由数据流（stream）、操作（operator）和组合（composite）3个语法单元组成。

连接数据流图中的各个计算单元边的抽象stream是由一系列token组成的数据序列，stream为SDF中的actor提供可并行处理的数据流。数据流图中的计算节点用operator表示。COStream定义了composite结构，将不同节点连接构造成数据流图。composite结构属于高层次的复合结构，可以由一个或多个operator组成可重用的子数据流图，是对SDF中可复用子图的抽象。

目前COStream已成功应用在网络媒体等领域，但是存在一些局限性：COStream是以C语言为基础扩展而成的，需要对COStream语法进行进一步的完善与扩展，以提高语言的表达能力。杨秋吉等人[22]在COStream的基础上提出了面向Storm的编译优化框架。

4.5 TensorFlow

TensorFlow是一个针对深度学习的特定的数据流编程模型，它通过一些内置的函数将整个计算过程组成一张数据流图，用于数值计算。图6给出了一个简单的数据流计算过程图的例子，节点表示数学操作，边表示2个节点之间依赖的多维数组（即张量）。TensorFlow根据数据流图，自动地将计算任务调度到相应的计算资源上进行计算。用户使用TensorFlow提供的接口构造数据流图，描述业务的计算任务。

TensorFlow的框架十分灵活，具有良好的可移植性，TensorFlow目前支持多种计算平台，包括台式计算机、服务器、集群、移动端、云端服务器等。TensorFlow针对机器学习中的核心算法——梯度下降法中的求解微分运算进行优化，以实现机器学习算法在TensorFlow中的高效执行。TensorFlow简化了用户构建的深度学习的网络模型，只需要定义模型的结构和目标函数即可形成一个网络模型，在网络模型的执行过程中，TensorFlow会自动计算相关的微分导数，实现参数求解。同时，TensorFlow支持多种语言，它提供了Python、C++、Java接口构建用户程序，打破了编程语言的限制。

近几年TensorFlow在机器学习领域得到了广泛应用，特别是在深度学习领域取得了长足的发展。但是TensorFlow的本质并不是一个通用的分布式计算框架，它需要用户在客户端显示指定集群信息，另外需要手动拉起进程（worker）等任务（task），在资源管理和使用方面有很多不便。因此，TensorFlow由于其用途单一、分布式能力弱、对大规模数据

处理支持不足的特点，难以充分发挥大规模并行计算机系统的高性能计算算力。同时TensorFlow只是在任务调度的过程中使用了数据流的思想，其任务粒度比Codelet模型粗，任务的并发度比Codelet模型低。

4.6 小结

针对数据流的编程模型，传统的软件工程提供了一个利用数据流描述业务处理流程的需求分析方法。谷歌公司的开源项目Apache Beam为数据流编程结构提供了一个参考，提供了一个完整的数据流编程模型，还提供了基于Java和Python的接口开发包。也有些直接利用传统的编程语言（如Python和Java）描述数据流执行的模型（如DAG），这些模型缺乏统一的数据流编程模型定义。另外，TensorFlow提供了一个针对深度学习的特定的数据流编程模型，其提供的编程模型通过一些内置的函数构建数据流计算图，但是粒度很小，基本接近数据流执行模型。目前主要的数据流编程模型的特征对比见**表1**。

5 基于数据流的编程工具

传统的基于数据流的编程工具有2类：

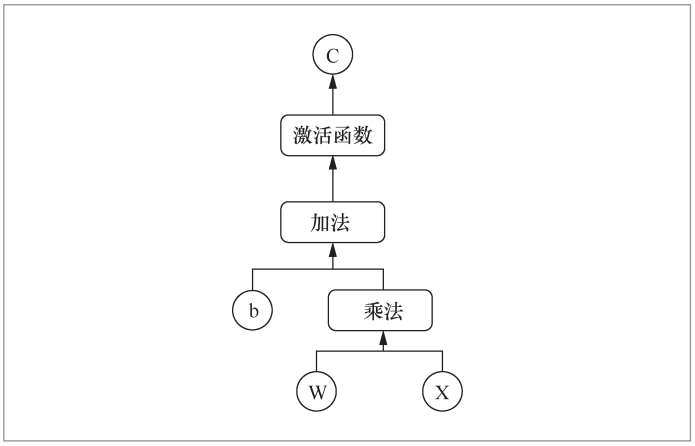

图 6 TensorFlow 数据流图

一类是软件工程中针对面向数据流的分析而设计的实例化（case）工具；另一类是面向运算级别的数据流编程工具。第一类是对业务模型的描述，相对于编程而言较为抽象，难以直接生产数据流程序；第二类偏向于对数据流执行模式的描述，其粒度较细，生产的代码接近于可执行的指令结构。目前针对大数据处理平台的数据流编程模型的编程工具主要分为3种形式。第一种形式是提供一个独立的图形化编辑器，可以通过可视化的模式构建数据流模型，从而提供数据流模型对应的代码。这种工具一般基于一个固定的框架构建数据流模型。第二种形式是在一种开发工具中提供一个插件，实现数据流程序的编写，并且利用开发工

表1 主要的数据流编程模型的特征对比

数据流编程模型	平台适应性	扩展性	适应场景
Apache Beam	能够适应不同的分布式计算平台	具有较好的可扩展性	能够适应复杂的数据流的处理
SWARM	自身提供计算引擎	具有硬件可扩展性	提供较底层的实现，目前提供的数据流处理功能较弱
StreamIt	适应特定的处理器平台	可扩展性较差	复杂的并行场景适应性较差
COStream	不同平台需要重新编译	可扩展性一般	高层的业务场景描述能力不够
TensorFlow	目前支持的平台越来越多	具有较好的可扩展性	主要针对深度学习

具提供的功能实现编译和运行集成。第三种形式是提供一套数据流编程模型的函数库，调用函数库中的函数构建数据流模型。

5.1 图形化数据流编程工具LabVIEW

LabVIEW[23]针对虚拟仪器程序提供面向数据流的模型构建和运行的一整套软件工具，包括采集、分析、显示和存储数据等一系列操作。LabVIEW中的程序框图上的节点表示计算任务，只有所有必要输入端的数据到达后才开始执行。节点执行后产生输出端数据，并将该数据传递给数据流路径中的下一个节点。数据流流动的过程描述了程序框图上虚拟仪器程序和函数的执行顺序。数据流计算图如**图7**所示。

5.2 COStream数据流程序图形编辑器

COStream图形编译器[24]是一个针对COStream编程语言的可视化编程工具，它将程序的编写、编译与运行集成在一起，方便用户进行数据流模型的构建，简化了开发过程。

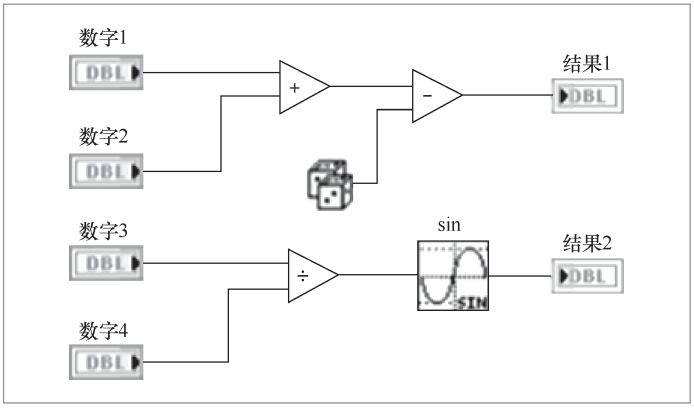

图 7 数据流计算图示例

COStream图形编辑器主要包含程序同步数据流图编辑和框架代码生成2个部分，以开源集成开发环境（Eclipse）插件的形式集成在Eclipse中。数据流图编辑器包含图形的绘制连接、图形编辑工具和图形管理功能。框架代码生成部分能够自动根据数据流图生成简洁的框架代码，并提供一定的框架优化策略。使用COStream图形编辑器可以通过简单的同步数据流编辑，生成对应的COStream代码，减少程序员的开发工作量，提高了代码编写的效率，并利用一些策略提高了生产代码的质量。COStream图形编辑器的图形绘制和代码生成如**图8**所示。

5.3 Oceanus-ML

Oceanus-ML旨在提供一套端到端（数据接入数据处理-特征工程-模型训练-模型评估）的在线学习解决方案。Oceanus-ML包含多样的数据处理函数，集成了丰富的在线学习及深度学习算法，用户通过简单的拖曳、填写参数，即可搭建完整的训练框架，并可轻松完成模型的训练、评估、流程部署。

对于用户来说，构建应用逻辑时，只需向画布中拖曳算子、填写参数、按逻辑连接算子，即可生成一个在线学习画布应用。

5.4 Sucuri数据流编程库

Sucuri[25]是一个简单的Python库，它用简单合理的语法提供了数据流编程。若要使用Sucuri库对应用程序进行并行化处理，程序员仅需识别其代码的并行化候选者，并实例化数据流图即可，其中每个节点均与此类函数之一关联，并且节点之间的边缘描述了函数之间的数据依赖

性。程序员可以使用Sucuri库进行数据流编程，实现代表重要的并行编程模式的2个基准，并在多核集群上执行。Sucuri数据流编程库构建的数据流图和代码如图9所示。

5.5 小结

目前的方法是将Apache Beam的函数库作为第三方函数库嵌入Java或者Python开发工具，在源代码级编写数据流程序，然后把编好的Java程序提交到Spark平台运行。但Apache Beam没有提供一种可视化开发工具。不同的数据流编程工具针对不同的领域，有些工具针对专用领域，有些工具面向通用领域，在易用性和可扩展性方面各有差异。上述数据流编程工具的特征对比见表2。

6 结束语

随着大数据 2.0 时代的到来，大数据的应用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。早期的处理模型（map/reduce[26]）早已力不从心，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，近年来涌现出很多大数据应用组件，如HBase[27]、Hive[28]、Kafka[29]、Spark、Flink等。开发者经常要用

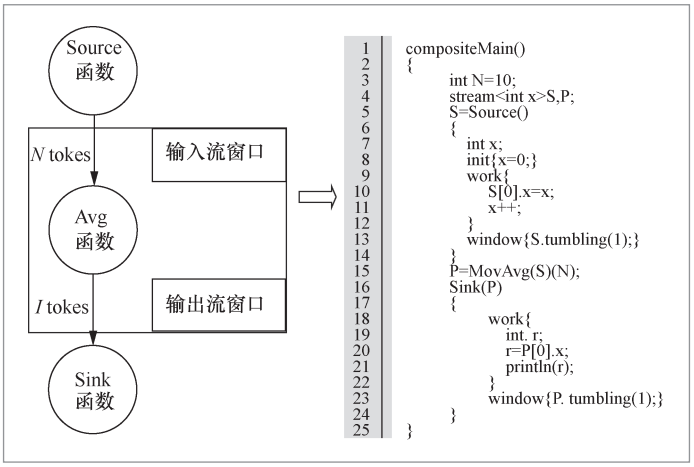

图 8 COStream 图形编辑器的图形绘制和代码生成

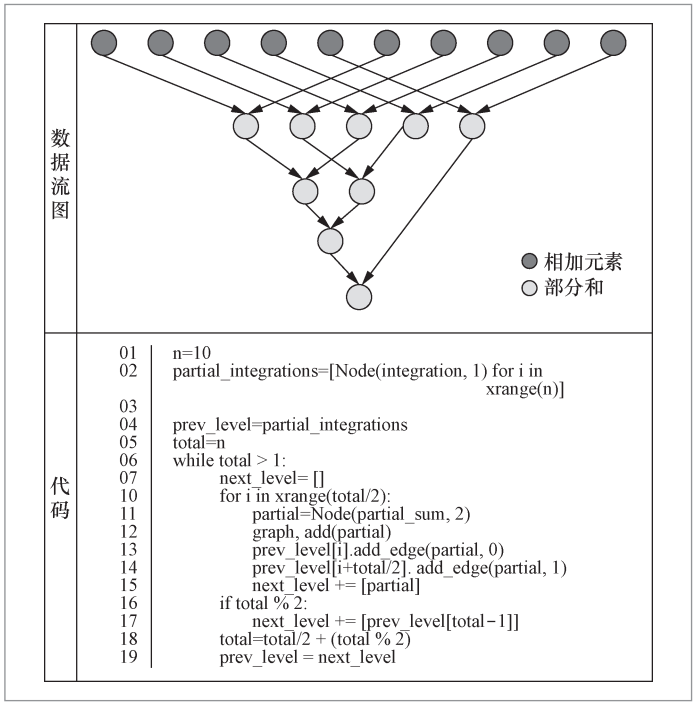

图 9 Sucuri 数据流编程库构建的数据流图和代码

表 2 数据流编程工具的特征对比

数据流编程工具	适应领域	可扩展性	编程模式	易用性	描述语言
LabVIEW	虚拟仪器仿真专用领域	较弱	图形化设计数据流图	较好	专用的G语言
COStream	通用的数据流计算编程	一般	提供编程库和图形化编程	一般	C语言
Oceanus-ML	大数据分析	一般	提供编程库和图形化编程	较好	扩展SQL
TensorFlow	深度学习	较好	提供编程函数库，利用现有的Python编程工具	较差	Python
Sucuri	并行数据流计算编程	较弱	提供编程库和图形化编程	较差	Python

到不同的技术、框架、API、开发语言和 SDK 应对复杂应用的开发，这大大增加了编程的难度。随着大数据应用的迅速发展，支持数据流计算的大数据处理平台日渐成熟，面向大规模分布式数据流应用的编程成为快速开发和部署数据流应用系统的关键，要满足面向大数据处理的数据流编程需求，需要符合以下3个特征。

- 能够对接业务需求，提供类似软件工程中DFD的丰富的数据流模型的业务描述能力，并且数据流编程模型能够适应不同的用户使用场景，提供与执行无关的抽象的统一编程模型。
- 能够对接不同的执行平台。数据流编程模型产生的代码能够部署到具体的大数据处理平台上，通过其数据流执行引擎进行自动解释和执行，不再需要开发人员的人工转换。
- 提供能够与执行环境适配、可扩展、可视化的数据流编程工具。编程工具能够导入执行环境的参数和算子，提供直观可拖曳的数据流模型图的构建，并能够自动实现数据流模型图和数据流程序代码之间的转换。

参考文献:

[1] VAVILAPALLI V K, MURTHY A C, DOUGLAS C, et al. Apache Hadoop YARN: yet another resource negotiator[C]// The 4th Annual Symposium on Cloud Computing. New York: ACM Press, 2013: 1-16.

[2] IQBAL M H, SOOMRO T R. Big data analysis: Apache Storm perspective[J]. International Journal of Computer Trends and Technology, 2015, 19(1): 9-14.

[3] CARBONE P, KATSIFODIMOS A, EWEN S, et al. Apache Flink: stream and batch processing in a single engine[J]. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2015, 36(4): 28-38.

[4] ZAHARIA M, XIN R S, WENDELL P, et al. Apache Spark: a unified engine for big data processing[J]. Communications of the ACM, 2016, 59(11): 56-65.

[5] MCDERMID J. Software engineering: apractitioner's approach[J]. Software Engineering Journal, 1995, 10(6): 266.

[6] JILANIC A A A, NADEEM A, KIM T H, et al. Formal representations of the data flow diagram: a survey[C]// The 2008 Advanced Software Engineering and Its Applications. Piscataway: IEEE Press, 2008: 153-158.

[7] REPA V. Object-oriented analysis with data flow diagram[M]. Heidelbery: Springer, 2013.

[8] DENNIS J B, FOSSEEN J B, LINDERMAN J P. Data flow schemas[C]// International Symposium on Theoretical Programming. Heidelbery: Springer, 1972: 187-216.

[9] KRAMER R, GUPTA R, SOFFA M L. The combining DAG: a technique for parallel data flow analysis[J]. IEEE Transactions on Parallel and Distributed Systems, 1994, 5(8): 805-813.

[10] DENNIS J B. Data flow supercomputers[J]. Computer, 1980(11): 48-56.

[11] MILUTINOVIC V, SALOM J, TRIFUNOVIC N, et al. Guide to dataflow supercomputing[M]. Heidelbery: Springer, 2015.

[12] LI A, BRAAK G J, CORPORAAL H, et al. Fine-grained synchronizations and dataflow programming on GPUs[C]// The 29th ACM on International Conference on Supercomputing. New York: ACM Press, 2015: 109-118.

[13] HALBWACHS N, CASPI P, RAYMOND P, et al. The synchronous data flow programming language LUSTRE[J]. Proceedings of the IEEE, 1991, 79(9): 1305-1320.

[14] 苏志超. 神威•太湖之光上数据流编程模型的设计与实现[D]. 合肥: 中国科学技术大学, 2018.
SU Z C. Design and implement of a dataflow programming model on sunway taihulight[D]. Hefei: University of Science and Technology of China, 2018.

[15] 杨瑞瑞. 面向多核 CPU/众核 GPU 异构集群的数据流编程模型研究[D]. 武汉: 华中科技大学, 2017.
YANG R R. A research of dataflow programming model oriented multi-core CPU/many-core GPU heterogeneous cluster[D]. Wuhan: Huazhong University of Science and Technology, 2017.

[16] 张维维, 魏海涛, 于俊清, 等. COStream: 一种面向数据流的编程语言和编译器实现[J]. 计算机学报, 2013, 36(10): 1993-2006.
ZHANG W W, WEI H T, YU J Q, et al. COStream: a language for dataflow application and compiler[J]. Chinese Journal of Computers, 2013, 36(10): 1993-2006.

[17] AKIDAU T, SCHMIDT E, WHITTLE S, et al. The dataflow model: a practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing[J]. Proceedings of the VLDB Endowment, 2015, 8(12): 1792-1803.

[18] LAUDERDALE C, KHAN R. Towards a codelet-based runtime for exascale computing: position paper[C]// The 2nd International Workshop on Adaptive Self-Tuning Computing Systems for the Exaflop Era. New York: ACM Press, 2012: 21-26.

[19] THIES W, KARCZMAREK M, AMARASINGHE S. StreamIt: a language for streaming applications[C]// The 11th International Conference on Compiler Construction. Heidelberg: Springer, 2002: 179-196.

[20] ABADI M, BARHAM P, CHEN J, et al. TensorFlow: a system for large-scale machine learning[C]// The 12th USENIX Symposium on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2016: 265-283.

[21] SUETTERLEIN J, ZUCKERMAN, STEPHANE, et al. An implementation of the Codelet model[M]. Heidelberg: Springer, 2013.

[22] 杨秋吉, 于俊清, 莫斌生, 等. 面向 Storm 的数据流编程模型与编译优化方法研究[J]. 计算机工程与科学, 2016, 38(12): 2409-2418.
YANG Q J, YU J Q, MO B S, et al. A data flow programming model and compiler optimization for Storm[J]. Computer Engineering and Science, 2016, 38(12): 2409-2418.

[23] BLUME P A. The LabVIEW style book[M]. Upper Saddle River: Prentice Hall, 2007.

[24] 杨燕. COStream 数据流程序图形编辑器的设计与实现[D]. 武汉: 华中科技大学, 2016.
YANG Y. The design and implementation of COStream graph editor[D]. Wuhan: Huazhong University of Science and Technology, 2016.

[25] ALVES T A O, GOLDSTEIN B F, FRANCA F M G, et al. A minimalistic dataflow programming library for Python[J]. Operations Research Letters, 2014, 5(1): 51-54.

[26] LIN Y A. Map-reduce for machine learning on multicore[C]// The 20th Annual Conference on Neural Information Processing Systems. Massachusetts: MIT Press, 2006: 281-288.

[27] TEAM A H B. Apache HBase reference guide[M]. California: [s.n.], 2016.

[28] HUAI Y, CHAUHAN A, GATES A, et al. Major technical advancements in Apache Hive[C]// The 2014 ACM SIGMOD International Conference on Management of Data. New York: ACM Press, 2014: 1235-1246.

[29] GARG N. Apache Kafka[M]. England: Packt Publishing Ltd, 2013.

作者简介

邹骁锋(1996-),男,湖南大学信息科学与工程学院博士生,主要研究方向为并行计算、数据挖掘和机器学习。

阳王东(1974-),男,湖南大学信息科学与工程学院教授,主要研究方向为分布式并行计算、机器学习。

容学成(1996-),男,湖南大学信息科学与工程学院硕士生,主要研究方向为大数据和机器学习。

李肯立(1970-),男,博士,湖南大学信息科学与工程学院教授,主要研究方向为高性能计算、人工智能和大数据。

李克勤(1963-),男,博士,湖南大学信息科学与工程学院教授,主要研究方向为并行计算、边缘计算和大数据。

收稿日期: 2020-01-19

通信作者: 阳王东, yangwangdong@163.com

基金项目: 国家重点研发计划基金资助项目(No.2018YFB1003401)

Foundation Item: The National Key Research and Development Program of China (No.2018YFB1003401)