软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(4):839−862 [doi: 10.13328/j.cnki.jos.004558]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

大数据流式计算:关键技术及系统实例*

孙大为[1], 张广艳[1,2], 郑纬民[1]

[1](清华大学 计算机科学与技术系,北京 100084)

[2](符号计算与知识工程教育部重点实验室(吉林大学),吉林 长春 130012)

通讯作者: 张广艳, E-mail: gyzh@tsinghua.edu.cn

摘 要: 大数据计算主要有批量计算和流式计算两种形态,目前,关于大数据批量计算系统的研究和讨论相对充分,而如何构建低延迟、高吞吐且持续可靠运行的大数据流式计算系统是当前亟待解决的问题且研究成果和实践经验相对较少.总结了典型应用领域中流式大数据所呈现出的实时性、易失性、突发性、无序性、无限性等特征,给出了理想的大数据流式计算系统在系统结构、数据传输、应用接口、高可用技术等方面应该具有的关键技术特征,论述并对比了已有的大数据流式计算系统的典型实例,最后阐述了大数据流式计算系统在可伸缩性、系统容错、状态一致性、负载均衡、数据吞吐量等方面所面临的技术挑战.

关键词: 大数据计算;流式计算;流式大数据;内存计算;系统实例

中图法分类号: TP311 **文献标识码**: A

中文引用格式: 孙大为,张广艳,郑纬民.大数据流式计算:关键技术及系统实例.软件学报,2014,25(4):839−862. http://www.jos.org.cn/1000-9825/4558.htm

英文引用格式: Sun DW, Zhang GY, Zheng WM. Big data stream computing: Technologies and instances. Ruan Jian Xue Bao/Journal of Software, 2014,25(4):839−862 (in Chinese). http://www.jos.org.cn/1000-9825/4558.htm

Big Data Stream Computing: Technologies and Instances

SUN Da-Wei[1], ZHANG Guang-Yan[1,2], ZHENG Wei-Min[1]

[1](Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

[2](Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education (Jilin University), Changchun 130012, China)

Corresponding author: ZHANG Guang-Yan, E-mail: gyzh@tsinghua.edu.cn

Abstract: Batch computing and stream computing are two important forms of big data computing. The research and discussions on batch computing in big data environment are comparatively sufficient. But how to efficiently deal with stream computing to meet many requirements, such as low latency, high throughput and continuously reliable running, and how to build efficient stream big data computing systems, are great challenges in the big data computing research. This paper provides a research of the data computing architecture and the key issues in stream computing in big data environments. Firstly, the research gives a brief summary of three application scenarios of stream computing in business intelligence, marketing and public service. It also shows distinctive features of the stream computing in big data environment, such as real time, volatility, burstiness, irregularity and infinity. A well-designed stream computing system always optimizes in system structure, data transmission, application interfaces, high-availability, and so on. Subsequently, the research offers detailed analyses and comparisons of five typical and open-source stream computing systems in big data environment. Finally, the research specifically addresses some new challenges of the stream big data systems, such as scalability, fault tolerance, consistency, load balancing and throughput.

* 基金项目: 国家自然科学基金(61170008, 61272055); 国家重点基础研究发展计划(973)(2014CB340402); 吉林大学符号计算与知识工程教育部重点实验室资助项目(93K172012K12)

收稿时间: 2013-09-07; 修改时间: 2013-11-21; 定稿时间: 2013-12-16; jos 在线出版时间: 2014-01-23

CNKI 网络优先出版: 2014-01-23 15:17, http://www.cnki.net/kcms/doi/10.13328/j.cnki.jos.000015.html

Key words: big data computing; stream computing; stream big data; memory computing; system instance

云计算、物联网、移动互连、社交媒体等新兴信息技术和应用模式的快速发展,促使全球数据量急剧增加,推动人类社会迈入大数据时代[1−4].一般意义上,大数据是指利用现有理论、方法、技术和工具难以在可接受的时间内完成分析计算、整体呈现高价值的海量复杂数据集合.大数据呈现出多种鲜明特征[3−7]:

- 在数据量方面,当前,全球所拥有的数据总量已经远远超过历史上的任何时期,更为重要的是,数据量的增加速度呈现出倍增趋势,并且每个应用所计算的数据量也大幅增加;
- 在数据速率方面,数据的产生、传播的速度更快,在不同时空中流转,呈现出鲜明的流式特征,更为重要的是,数据价值的有效时间急剧减少,也要求越来越高的数据计算和使用能力;
- 在数据复杂性方面,数据种类繁多,数据在编码方式、存储格式、应用特征等多个方面也存在多层次、多方面的差异性,结构化、半结构化、非结构化数据并存,并且半结构化、非结构化数据所占的比例不断增加;
- 在数据价值方面,数据规模增大到一定程度之后,隐含于数据中的知识的价值也随之增大,并将更多地推动社会的发展和科技的进步.此外,大数据往往还呈现出个性化、不完备化、价值稀疏、交叉复用等特征.

大数据蕴含大信息,大信息提炼大知识,大知识将在更高的层面、更广的视角、更大的范围帮助用户提高洞察力、提升决策力,将为人类社会创造前所未有的重大价值.但与此同时,这些总量极大的价值往往隐藏在大数据中,表现出了价值密度极低、分布极其不规律、信息隐藏程度极深、发现有用价值极其困难的鲜明特征.这些特征必然为大数据的计算环节带来前所未有的挑战和机遇,并要求大数据计算系统具备高性能、实时性、分布式、易用性、可扩展性等特征.

大数据价值的有效实现离不开 A,B,C 这三大要素,即,大分析(big Analytic)、大带宽(big Bandwidth)和大内容(big Content).其中,

(1) 大分析.通过创新性的数据分析方法实现对大量数据的快速、高效、及时的分析与计算,得出跨数据间的、隐含于数据中的规律、关系和内在逻辑,帮助用户理清事件背后的原因、预测发展趋势、获取新价值;

(2) 大带宽.通过大带宽提供良好的基础设施,以便在更大范围内进行数据的收集,以更快的速度进行数据的传输,为大数据的分析、计算等环节提供时间和数据量方面的基本保障;

(3) 大内容.只有在数据内容足够丰富、数据量足够大的前提下,隐含于大数据中的规律、特征才能被识别出来.

由此可见,大分析是实现途径,大带宽是基本保障,大内容是前提条件.

大数据的计算模式[7−10]可以分为批量计算(batch computing)和流式计算(stream computing)两种形态:

- 如图 1 所示,批量计算首先进行数据的存储,然后再对存储的静态数据进行集中计算.Hadoop 是典型的大数据批量计算架构,由 HDFS 分布式文件系统负责静态数据的存储,并通过 MapReduce 将计算逻辑分配到各数据节点进行数据计算和价值发现;
- 如图 2 所示,流式计算中,无法确定数据的到来时刻和到来顺序,也无法将全部数据存储起来.因此,不再进行流式数据的存储,而是当流动的数据到来后在内存中直接进行数据的实时计算.如 Twitter 的 Storm、Yahoo 的 S4 就是典型的流式数据计算架构,数据在任务拓扑中被计算,并输出有价值的信息.

流式计算和批量计算分别适用于不同的大数据应用场景:对于先存储后计算,实时性要求不高,同时,数据的准确性、全面性更为重要的应用场景,批量计算模式更合适;对于无需先存储,可以直接进行数据计算,实时性要求很严格,但数据的精确度要求稍微宽松的应用场景,流式计算具有明显优势.流式计算中,数据往往是最近一个时间窗口内的,因此数据延迟往往较短,实时性较强,但数据的精确程度往往较低.流式计算和批量计算具有明显的优劣互补特征,在多种应用场合下可以将两者结合起来使用.通过发挥流式计算的实时性优势和批量

计算的计算精度优势,满足多种应用场景在不同阶段的数据计算要求.

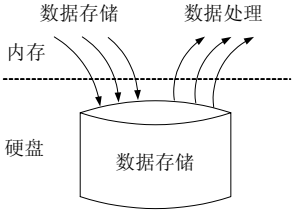


Fig.1 Big data batch computing
图 1 大数据批量计算

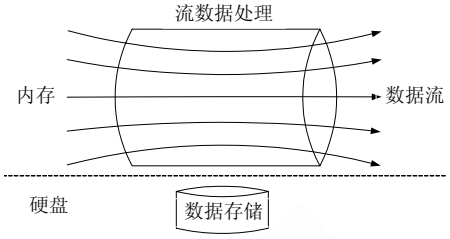


Fig.2 Big data stream computing
图 2 大数据流式计算

目前,关于大数据批量计算相关技术的研究相对成熟[3−10],形成了以 Google 的 MapReduce 编程模型、开源的 Hadoop 计算系统为代表的高效、稳定的批量计算系统,在理论上和实践中均取得了显著成果[7,11].关于流式计算的早期研究往往集中在数据库环境中开展数据计算的流式化,数据规模较小,数据对象比较单一.由于新时期的流式大数据呈现出实时性、易失性、突发性、无序性、无限性等特征,对系统提出了很多新的更高的要求.2010 年,Yahoo 推出 S4 流式计算系统,2011 年,Twitter 推出 Storm 流式计算系统,在一定程度上推动了大数据流式计算技术的发展和应用.但是,这些系统在可伸缩性、系统容错、状态一致性、负载均衡、数据吞吐量等诸多方面仍然存在着明显不足.如何构建低延迟、高吞吐且持续可靠运行的大数据流式计算系统,是当前亟待解决的问题.

本文以大数据流式计算系统的设计、优化和挑战为核心,系统地梳理和分析了当前大数据流式计算系统的研究和发展现状,总结了在金融银行业应用、互联网应用和物联网应用这三大典型领域中,流式大数据所呈现出的实时性、易失性、突发性、无序性、无限性等特征.给出了理想的大数据流式计算系统在系统结构、数据传输、应用接口、高可用技术等方面应该具有的关键技术特性,论述并对比了 5 款大数据流式计算系统,即,Twitter 的 Storm 系统、Yahoo 的 S4 系统、Facebook 的 Data Freeway and Puma 系统、Linkedin 的 Kafka 系统、Microsoft 的 TimeStream 系统.阐述了大数据流式计算系统在可伸缩性、系统容错、状态一致性、负载均衡、数据吞吐量等方面所面临的技术挑战.本文工作为构建低延迟、高吞吐且持续可靠运行的大数据流式计算系统提供了一些指导性原则,弥补了当前关于大数据流式计算的研究成果不足的局面.

本文第 1 节分析大数据流式计算的典型应用领域及其特征.第 2 节论述设计优良的大数据流式计算系统在系统结构、数据传输、应用接口、高可用技术等方面应该满足的关键技术要求.第 3 节分析对比 5 款比较典型的大数据流式计算系统.第 4 节具体阐述大数据流式计算在系统的可伸缩性、系统容错、状态一致性、负载均衡、数据吞吐量等方面所面临的新的挑战.最后,第 5 节对全文进行总结.

1 应用场景及数据特征

大数据流式计算主要用于对动态产生的数据进行实时计算并及时反馈结果,但往往不要求结果绝对精确的应用场景.在数据的有效时间内获取其价值,是大数据流式计算系统的首要设计目标,因此,当数据到来后将立即对其进行计算,而不再对其进行缓存等待后续全部数据到来再进行计算.

1.1 应用场景

大数据流式计算的应用场景较多[12−17],本文按照数据产生方式、数据规模大小以及技术成熟度高低这 3 个不同维度,选择金融银行业应用、互联网应用和物联网应用这 3 种典型应用场景,用于分析说明大数据流式计算的基本特征.从数据产生方式上看,它们分别是被动产生数据、主动产生数据和自动产生数据;从数据规模上看,它们处理的数据分别是小规模、中规模和大规模;从技术成熟度上看,它们分别是成熟度高、成熟度中和成熟度低的数据.

(1) 金融银行业的应用

在金融银行领域的日常运营过程中,往往会产生大量数据,这些数据的时效性往往较短.因此,金融银行领域是大数据流式计算最典型的应用场景之一,也是大数据流式计算最早的应用领域.在金融银行系统内部,每时每刻都有大量的往往是结构化的数据在各个系统间流动,并需要实时计算.同时,金融银行系统与其他系统也有着大量的数据流动,这些数据不仅有结构化数据,也会有半结构化和非结构化数据.通过对这些大数据的流式计算,发现隐含于其中的内在特征,可以帮助金融银行系统进行实时决策.

在金融银行的实时监控场景中,大数据流式计算往往体现出了自身的优势.如:

- 风险管理.包括信用卡诈骗、保险诈骗、证券交易诈骗、程序交易等,需要实时跟踪发现;
- 营销管理.如,根据客户信用卡消费记录,掌握客户的消费习惯和偏好,预测客户未来的消费需求,并为其推荐个性化的金融产品和服务;
- 商业智能.如,掌握金融银行系统内部各系统的实时数据,实现对全局状态的监控和优化,并提供决策支持.

(2) 互联网领域的应用

随着互联网技术的不断发展,特别是 Web 2.0 时代的到来,用户可以实时分享和提供各类数据.不仅使得数据量大为增加,也使得数据更多地以半结构化和非结构化的形态呈现.据统计,目前互联网中 75%的数据来源于个人,主要以图片、音频、视频数据形式存在,需要实时分析和计算这些大量、动态的数据.

在互联网领域中,大数据流式计算的典型应用场景包括:

- 搜索引擎.搜索引擎提供商们往往会在反馈给客户的搜索页面中加入点击付费的广告信息.插入什么广告、在什么位置插入这些广告才能得到最佳效果,往往需要根据客户的查询偏好、浏览历史、地理位置等综合语义进行决定.而这种计算对于搜索服务器而言往往是大量的:一方面,每时每刻都会有大量客户进行搜索请求;另一方面,数据计算的时效性极低,需要保证极短的响应时间;
- 社交网站.需要实时分析用户的状态信息,及时提供最新的用户分享信息到相关的朋友,准确地推荐朋友,推荐主题,提升用户体验,并能及时发现和屏蔽各种欺骗行为.

(3) 物联网领域的应用

在物联网环境中,各个传感器产生大量数据.这些数据通常包含时间、位置、环境和行为等内容,具有明显的颗粒性.由于传感器的多元化、差异化以及环境的多样化,这些数据呈现出鲜明的异构性、多样性、非结构化、有噪声、高增长率等特征.所产生的数据量之密集、实时性之强、价值密度之低是前所未有的,需要进行实时、高效的计算.

在物联网领域中,大数据流式计算的典型应用场景包括:

- 智能交通.通过传感器实时感知车辆、道路的状态,并分析和预测一定范围、一段时间内的道路流量情况,以便有效地进行分流、调度和指挥;
- 环境监控.通过传感器和移动终端,对一个地区的环境综合指标进行实时监控、远程查看、智能联动、远程控制,系统地解决综合环境问题.

这些对计算系统的实时性、吞吐量、可靠性等方面都提出很高要求.

大数据流式计算的 3 种典型应用场景的对比见表 1.

- 从数据的产生方式看,金融银行领域的数据往往是在系统中被动产生的,互联网领域的数据往往是人为主动产生的,物联网领域的数据往往是由传感器等设备自动产生的;
- 从数据的规模来看:金融银行领域的数据与互联网、物联网领域的数据相比较少;物联网领域的数据规模是最大的,但受制于物联网的发展阶段,当前实际拥有数据规模最大的是互联网领域;
- 从技术成熟度来看:金融银行领域的流式大数据应用最为成熟,从早期的复杂事件处理[18,19]开始就呈现了大数据流式计算的思想;互联网领域的发展,将大数据流式计算真正推向历史舞台;物联网领域的发展为大数据流式计算提供了重要的历史机遇.

Table 1 Scenarios contrast of stream computing of big data

表 1 大数据流式计算应用场景对比

应用场景	数据产生方式	数据规模	技术成熟度
金融银行	被 动	小	高
互 联 网	主 动	中	中
物 联 网	自 动	大	低

1.2 流式大数据特征

图 3 用有向无环图(directed acyclic graph,简称 DAG)描述了大数据流的计算过程,其中,圆形表示数据的计算节点,箭头表示数据的流动方向.

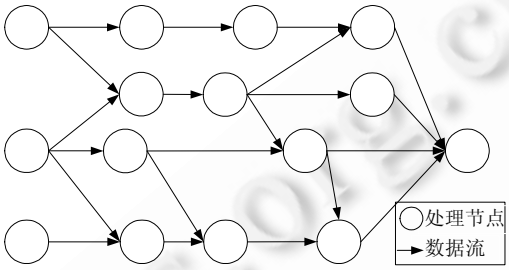


Fig.3 Directed acyclic graph

图 3 有向无环图

与大数据批量计算不同,大数据流式计算中的数据流主要体现了如下 5 个特征[11,20,21]:

(1) 实时性

流式大数据是实时产生、实时计算,结果反馈往往也需要保证及时性.流式大数据价值的有效时间往往较短,大部分数据到来后直接在内存中进行计算并丢弃,只有少量数据才被长久保存到硬盘中.这就需要系统有足够的低延迟计算能力,可以快速地进行数据计算,在数据价值有效的时间内,体现数据的有用性.对于时效性特别短、潜在价值又很大的数据可以优先计算.

(2) 易失性

在大数据流式计算环境中,数据流往往是到达后立即被计算并使用,只有极少数的数据才会被持久化地保存下来,大多数数据往往会被直接丢弃.数据的使用往往是一次性的、易失的,即使重放,得到的数据流和之前的数据流往往也是不同的.这就需要系统具有一定的容错能力,要充分地利用好仅有的一次数据计算机会,尽可能全面、准确、有效地从数据流中得出有价值的信息.

(3) 突发性

在大数据流式计算环境中,数据的产生完全由数据源确定,由于不同的数据源在不同时空范围内的状态不统一且发生动态变化,导致数据流的速率呈现出了突发性的特征.前一时刻数据速率和后一时刻数据速率可能会有巨大的差异,这就需要系统具有很好的可伸缩性,能够动态适应不确定流入的数据流,具有很强的系统计算能力和大数据流量动态匹配的能力.一方面,在突发高数据流速的情况下,保证不丢弃数据,或者识别并选择性地丢弃部分不重要的数据;另一方面,在低数据速率的情况下,保证不会太久或过多地占用系统资源.

(4) 无序性

在大数据流式计算环境中,各数据流之间、同一数据流内部各数据元素之间是无序的:一方面,由于各个数据源之间是相互独立的,所处的时空环境也不尽相同,因此无法保证数据流间的各个数据元素的相对顺序;另一方面,即使是同一个数据流,由于时间和环境的动态变化,也无法保证重放数据流和之前数据流中数据元素顺序的一致性.这就需要系统在数据计算过程中具有很好的数据分析和发现规律的能力,不能过多地依赖数据流间的内在逻辑或者数据流内部的内在逻辑.

(5) 无限性

在大数据流式计算中,数据是实时产生、动态增加的,只要数据源处于活动状态,数据就会一直产生和持续增加下去.可以说,潜在的数据量是无限的,无法用一个具体确定的数据实现对其进行量化.系统在数据计算过程中,无法保存全部数据:一方面,硬件中没有足够大的空间来存储这些无限增长的数据;另一方面,也没有合适的软件来有效地管理这么多数据;并且,需要系统具有很好的稳定性,保证系统长期而稳定地运行.

表 2 对比了大数据流式计算和大数据批量计算的需求.

Table 2 Scenario contrast between stream and batch big data

表 2 大数据流式、批量需求对比

性能指标	大数据流式计算	大数据批量计算
计算方式	实时	批量
常驻空间	内存	硬盘
时效性	短	长
有序性	无	有
数据量	无限	有限
数据速率	突发	稳定
是否可重现	难	易
移动对象	数据移动	程序移动
数据精确度	较低	较高

2 大数据流式计算关键技术

针对具有实时性、易失性、突发性、无序性、无限性等特征的流式大数据,理想的大数据流式计算系统应该表现出低延迟、高吞吐、持续稳定运行和弹性可伸缩等特性,这其中离不开系统架构、数据传输、编程接口、高可用技术等关键技术的合理规划和良好设计.

2.1 系统架构

系统架构是系统中各子系统间的组合方式,属于大数据计算所共有的关键技术,大数据流式计算需要选择特定的系统架构进行流式计算任务的部署.当前,大数据流式计算系统采用的系统架构[22−24]可以分为无中心节点的对称式系统架构(如 S4,Puma 等系统)以及有中心节点的主从式架构(如 Storm 系统):

(1) 对称式架构.如图 4 所示:系统中各个节点的功能是相同的,具有良好的可伸缩性;但由于不存在中心节点,在资源调度、系统容错、负载均衡等方面需要通过分布式协议实现.例如,S4 通过 Zookeeper 实现系统容错、负载均衡等功能;

(2) 主从式系统架构.如图 5 所示:系统存在一个主节点和多个从节点,主节点负责系统资源的管理和任务的协调,并完成系统容错、负载均衡等方面的工作;从节点负责接收来自于主节点的任务,并在计算完成后进行反馈.各个从节点间没有数据往来,整个系统的运行完全依赖于主节点控制.

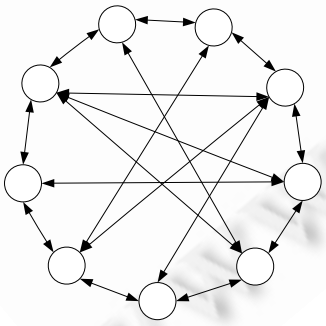


Fig.4 Symmetric architecture

图 4 对称式架构

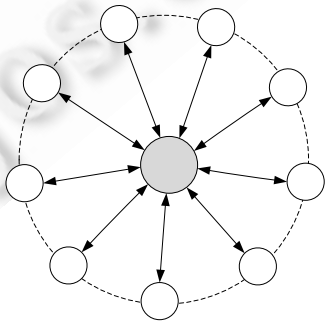


Fig.5 Master-Slave architecture

图 5 主从式架构

2.2 数据传输

数据传输是指完成有向任务图到物理计算节点的部署之后,各个计算节点之间的数据传输方式.在大数据流式计算环境中,为了实现高吞吐和低延迟,需要更加系统地优化有向任务图以及有向任务图到物理计算节点的映射方式.如图 6 所示,在大数据流式计算环境中,数据的传输方式分为主动推送方式(基于 push 方式)和被动拉取方式(基于 pull 方式)[24−26]:

(1) 主动推送方式.在上游节点产生或计算完数据后,主动将数据发送到相应的下游节点,其本质是让相关数据主动寻找下游的计算节点,当下游节点报告发生故障或负载过重时,将后续数据流推送到其他相应节点.主动推送方式的优势在于数据计算的主动性和及时性,但由于数据是主动推送到下游节点,往往不会过多地考虑到下游节点的负载状态、工作状态等因素,可能会导致下游部分节点负载不够均衡;

(2) 被动拉取方式.只有下游节点显式进行数据请求,上游节点才会将数据传输到下游节点,其本质是让相关数据被动地传输到下游计算节点.被动拉取方式的优势在于下游节点可以根据自身的负载状态、工作状态适时地进行数据请求,但上游节点的数据可能未必得到及时的计算.

大数据流式计算的实时性要求较高,数据需要得到及时处理,往往选择主动推送的数据传输方式.当然,主动推送方式和被动拉取方式不是完全对立的,也可以将两者进行融合,从而在一定程度上实现更好的效果.

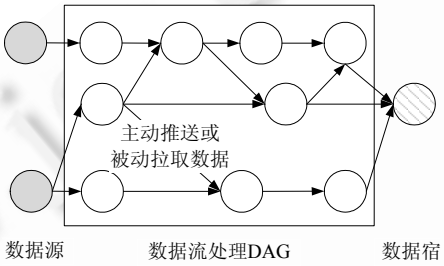


Fig.6 Transformation of data stream

图 6 数据流传输方式

2.3 编程接口

编程接口是方便用户根据流式计算的任务特征,通过有向任务图来描述任务内在逻辑和依赖关系,并编程实现任务图中各节点的处理功能.用户策略的定制、业务流程的描述和具体应用的实现,需要通过大数据流式计算系统提供的应用编程接口.良好的应用编程接口可以方便用户实现业务逻辑,可以减少用户的编程工作量,并降低用户系统功能的实现门槛[27−29].

当前,大多数开源大数据流式计算系统均提供了类似于 MapReduce 的类 MR 用户编程接口.例如:Storm 提供 Spout 和 Bolt 应用编程接口,用户只需要定制 Spout 和 Bolt 的功能,并规定数据流在各个 Bolt 间的内在流向,明确数据流的有向无环图,其他具体细节的实现方式用户不需要太多关心,即可满足对流式大数据的高效、实时计算;也有部分大数据流式计算系统为用户提供了类 SQL 的应用编程接口,并给出了相应的组件,便于应用功能的实现;StreamBase 系统不仅为用户提供了类 SQL 的应用编程接口来描述计算过程,也借助图形化用户视窗为用户提供了丰富的组件.

2.4 高可用技术

大数据批量计算将数据事先存储到持久设备上,节点失效后容易实现数据重放;而大数据流式计算对数据不进行持久化存储.因此,批量计算中的高可用技术不完全适用于流式计算环境,需要根据流式计算新特征及其新的高可用要求,有针对性地研究更加轻量、高效的高可用技术和方法.

大数据流式计算系统高可用是通过状态备份和故障恢复策略实现的.当故障发生后,系统根据预先定义的

策略进行数据的重放和恢复.按照实现策略,可以细分为被动等待(passive standby)、主动等待(active standby)和上游备份(upstream backup)这 3 种策略[30−34]:

(1) 被动等待策略

如图 7 所示:主节点 B 进行数据计算,副本节点 B'处于待命状态,系统会定期地将主节点 B 上的最新的状态备份到副本节点 B'上.出现故障时,系统从备份数据中进行状态恢复.被动等待策略支持数据负载较高、吞吐量较大的场景,但故障恢复时间较长,可以通过对备份数据的分布式存储缩短恢复时间.该方式更适合于精确式数据恢复,可以很好地支持不确定性计算应用,在当前流式数据计算中应用最为广泛.

(2) 主动等待策略

如图 8 所示:系统在为主节点 B 传输数据的同时,也为副本节点 B'传输一份数据副本.以主节点 B 为主进行数据计算,当主节点 B 出现故障时,副本节点 B'完全接管主节点 B 的工作,主副节点需要分配同样的系统资源.该种方式故障恢复时间最短,但数据吞吐量较小,也浪费了较多的系统资源.在广域网环境中,系统负载往往不是过大时,主动等待策略是一个比较好的选择,可以在较短的时间内实现系统恢复.

(3) 上游备份策略

如图 9 所示:每个主节点均记录其自身的状态和输出数据到日志文件,当某个主节点 B 出现故障后,上游主节点会重放日志文件中的数据到相应副本节点 B'中,进行数据的重新计算.上游备份策略所占用的系统资源最小,在无故障期间,由于副本节点 B'保持空闲状态,数据的执行效率很高.但由于其需要较长的时间进行恢复状态的重构,故障的恢复时间往往较长.如当需要恢复时间窗口为 30 分钟的聚类计算,就需要重放该 30 分钟内的所有元组.可见,对于系统资源比较稀缺、算子状态较少的情况,上游备份策略是一个比较好的选择方案.

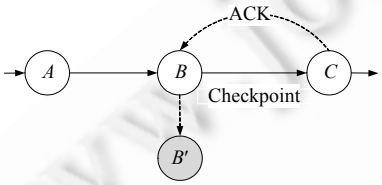


Fig.7 Passive standby

图 7 被动等待策略

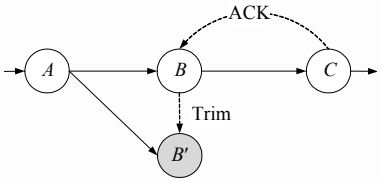
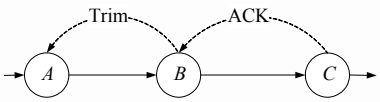
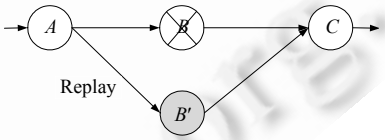


Fig.8 Active standby

图 8 主动等待策略



(a)



(b)

Fig.9 Upstream backup

图 9 上游备份策略

表 3 从 5 个方面详细对比了上述 3 种高可用策略,实际应用中可以根据具体环境进行选择.

Table 3 Contrast of three high availability strategies

表 3 3 种高可用策略对比

性能指标	被动等待策略	主动等待策略	上游备份策略
系统性能	低	高	高
恢复速度	中	高	低
资源使用	中	高	低
精准恢复	是	否	否
适用范围	局域网	广域网	局域网或广域网

2.5 其他关键技术

此外,大数据流式计算系统也离不开其他相关关键技术的支持,包括:

- 系统故障恢复.快速地实现从故障状态到一种正确状态的恢复,满足系统的高效运行需求;
- 系统资源调度.实现对系统中资源的最佳利用,提高资源的利用率,保证任务的完成和能耗的节省;
- 负载均衡策略.实现对系统中的任务的动态、合理的分配,动态适应系统负载情况,保证系统中的任务均衡和稳定地运行;
- 数据在任务拓扑中的路由策略.促进系统中负载均衡策略的高效实现、数据的合理流动及快速处理.

3 系统实例分析

现有的大数据流式计算系统实例有 Twitter 的 Storm 系统[35]、Yahoo 的 S4(simple scalable streaming system)系统[36]、Facebook 的 Data Freeway and Puma 系统[37]、Linkedin 的 Kafka 系统[38]、Microsoft 的 TimeStream 系统[39]、Hadoop 之上的数据分析系统 HStreaming[40]、IBM 的商业流式计算系统 StreamBase[41]、Berkeley 的交互式实时计算系统 Spark[42]、专门进行复杂事件处理(complex event processing,简称 CEP)的 Esper[43]系统等.本文选择当前比较典型的、应用较为广泛的、具有代表性的前 5 款大数据流式计算系统进行实例分析.

3.1 Storm系统

Storm[35,44−46]是 Twitter 支持开发的一款分布式的、开源的、实时的、主从式大数据流式计算系统,最新版本是 Storm 0.8.2,使用的协议为 Eclipse Public License 1.0,其核心部分使用了高效流式计算的函数式语言 Clojure 编写,极大地提高了系统性能.但为了方便用户使用,支持用户使用任意编程语言进行项目的开发.

(1) 任务拓扑

任务拓扑(topology)是 Storm 的逻辑单元,一个实时应用的计算任务将被打包为任务拓扑后发布,任务拓扑一旦提交后将会一直运行着,除非显式地去中止.一个任务拓扑是由一系列 Spout 和 Bolt 构成的有向无环图,通过数据流(stream)实现 Spout 和 Bolt 之间的关联,如图 10 所示.其中,Spout 负责从外部数据源不间断地读取数据,并以 Tuple 元组的形式发送给相应的 Bolt;Bolt 负责对接收到的数据流进行计算,实现过滤、聚合、查询等具体功能,可以级联,也可以向外发送数据流.

数据流是 Storm 对数据进行的抽象,它是时间上无穷的 Tuple 元组序列,如图 11 所示,数据流是通过流分组(stream grouping)所提供的不同策略实现在任务拓扑中流动.此外,为了满足确保消息能且仅能被计算 1 次的需求,Storm 还提供了事务任务拓扑.

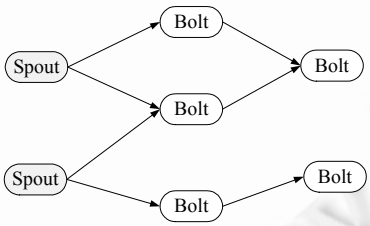


Fig.10 Task topology of storm

图 10 Storm 任务拓扑

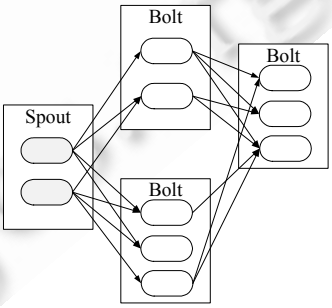


Fig.11 Data stream group of storm

图 11 Storm 数据流组

(2) 作业级容错机制

用户可以为一个或多个数据流作业(以下简称数据流)进行编号,分配一个唯一的 ID,Storm 可以保障每个编号的数据流在任务拓扑中被完全执行.所谓的完全执行,是指由该 ID 绑定的源数据流以及由该源数据流后续生

成的新数据流经过任务拓扑中每一个应该到达的 Bolt,并被完全执行.如图 12 所示,两个数据流被分配一个 *ID*=1,当且仅当两个数据流分别经过 Bolt 1 和 Bolt 2,最终都到达 Bolt 3 并均被完全处理后,才表明数据流被完全执行.

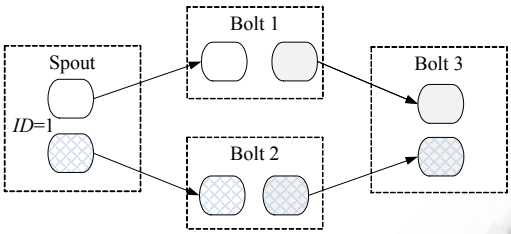


Fig.12 Fully implement of data stream task

图 12 数据流作业完全执行

Storm 通过系统级组件 Acker 实现对数据流的全局计算路径的跟踪,并保证该数据流被完全执行.其基本原理是为数据流中的每个分组进行编号,并通过异或运算来实现对其计算路径的跟踪.

作业级容错的基本原理是:

A xor A=0.

A xor B … xor B xor A=0,当且仅当每个编号仅出现 2 次.

作业级容错的基本流程是:在 Spout 中,系统会为数据流的每个分组生成一个唯一的 64 位整数,作为该分组的根 ID.根 ID 会被传递给 Acker 及后续的 Bolt 作为该分组单元的唯一标识符.同时,无论是 Spout 还是 Bolt,每次新生成一个分组的时候,都会重新赋予该分组一个新的 64 位的整数的 ID.Spout 发送完某个数据流对应的源分组后,并告知 Acker 自己所发射分组的根 ID 及生成的那些分组的新 ID,而 Bolt 每次接受到一个输入分组并计算完之后,也将告知 Acker 自己计算的输入分组的 ID 及新生成的那些分组的 ID,Acker 只需要对这些 ID 做一个简单的异或运算,就能判断出该根 ID 对应的消息单元是否计算完成.

(3) 总体架构

Storm 采用主从系统架构,如图 13 所示,在一个 Storm 系统中有两类节点(即,一个主节点 Nimbus、多个从节点 Supervisor)及 3 种运行环境(即,master,cluster 和 slaves)构成,其中,

- 主节点 Nimbus 运行在 master 环境中,是无状态的,负责全局的资源分配、任务调度、状态监控和故障检测:一方面,主节点 Nimbus 接收客户端提交来的任务,验证后分配任务到从节点 Supervisor 上,同时把该任务的元信息写入 Zookeeper 目录中;另一方面,主节点 Nimbus 需要通过 Zookeeper 实时监控任务的执行情况,当出现故障时进行故障检测,并重启失败的从节点 Supervisor 和工作进程 Worker;
- 从节点 Supervisor 运行在 slaves 环境中,也是无状态的,负责监听并接受来自于主节点 Nimbus 所分配的任务,并启动或停止自己所管理的工作进程 Worker,其中,工作进程 Worker 负责具体任务的执行.一个完整的任务拓扑往往由分布在多个从节点 Supervisor 上的 Worker 进程来协调执行,每个 Worker 都执行且仅执行任务拓扑中的一个子集.在每个 Worker 内部,会有多个 Executor,每个 Executor 对应一个线程.Task 负责具体数据的计算,即,用户所实现的 Spout/Blot 实例.每个 Executor 会对应一个或多个 Task,因此,系统中 Executor 的数量总是小于等于 Task 的数量.

Zookeeper 是一个针对大型分布式系统的可靠协调服务和元数据存储系统,通过配置 Zookeeper 集群,可以使用 Zookeeper 系统所提供的高可靠性服务.Storm 系统引入 Zookeeper,极大地简化了 Nimbus,Supervisor,Worker 之间的设计,保障了系统的稳定性.Zookeeper 在 Storm 系统中具体实现了以下功能:(a) 存储客户端提交的任务拓扑信息、任务分配信息、任务的执行状态信息等,便于主节点 Nimbus 监控任务的执行情况;(b) 存储从节点 Supervisor、工作进程 Worker 的状态和心跳信息,便于主节点 Nimbus 监控系统各节点运行状态;(c) 存储整个集群的所有状态信息和配置信息,便于主节点 Nimbus 监控 Zookeeper 集群的状态,在出现主 Zookeeper

节点挂掉后可以重新选取一个节点作为主 Zookeeper 节点,并进行恢复.

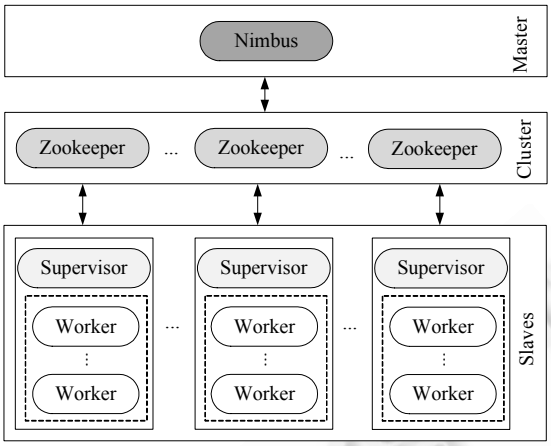


Fig.13 Storm architecture

图 13 Storm 系统架构

(4) 系统特征

Storm 系统的主要特征为:(a) 简单编程模型.用户只需编写 Spout 和 Bolt 部分的实现,因此极大地降低了实时大数据流式计算的复杂性;(b) 支持多种编程语言.默认支持 Clojure,Java,Ruby 和 Python,也可以通过添加相关协议实现对新增语言的支持;(c) 作业级容错性.可以保证每个数据流作业被完全执行;(d) 水平可扩展.计算可以在多个线程、进程和服务器之间并发执行;(e) 快速消息计算.通过 ZeroMQ 作为其底层消息队列,保证了消息能够得到快速的计算.

(5) 存在不足

Storm 系统存在的不足主要包括:资源分配没有考虑任务拓扑的结构特征,无法适应数据负载的动态变化;采用集中式的作业级容错机制,在一定程度上限制了系统的可扩展性.

3.2 S4系统

S4[36,46−49]是 Yahoo 支持开发的一款分布式的、可扩展的、可插拔的、对称的大数据流式计算系统,最新版本是 S4 0.6.0,使用的协议为 Apache License 2.0,编程语言为 Java.

(1) 处理单元 PE

处理单元 PE(processing element)如图 14 所示,是 S4 中的基本计算单元,由 4 个组件构成,即:(a) 函数.实现了与该处理单元 PE 相对应的功能和配置;(b) 事件类型.规定了该处理单元 PE 所接收的事件类型;(c) 主键.规定了该处理单元 PE 所关心的事件主键;(d) 键值.规定了该处理单元 PE 所匹配的键值.

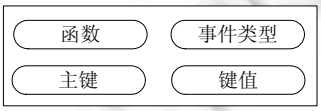


Fig.14 Processing element PE

图 14 处理单元 PE

处理单元 PE 只关心与其事件类型相匹配的事件,并仅仅处理与其主键、键值相一致的事件,即,只有事件类型、主键、键值全部匹配后,处理单元 PE 才会处理该类事件.当一个新事件没有可以匹配的处理单元 PE 时,系统将会为该事件新创建一个处理单元 PE.因此,需要高效、动态地创建、管理和删除处理单元 PE;同时,处理单元 PE 的类型设计及其拓扑结构也需要更合理地规划.

有一类处理单元 PE 位于 S4 的输入层,它们没有主键、键值,只需事件类型相匹配,即对该类事件进行处理.通常情况下,该类处理单元 PE 所计算的事件为原始输入事件,其输出事件会被新增主键、键值,以便后续处理单元 PE 进行计算.

(2) 任务拓扑结构

在 S4 系统中,数据流是由事件的有序序列(K,A)构成的,其中,K,A 分别表示该类型事件的若干个 *key* 和若干个 *attribute*,*key* 和 *attribute* 都是 *tuple-valued*,即,*key*=*value* 的元组值.事件在各个处理单元 PE 中被计算,在处理单元 PE 之间流动,处理单元 PE 之间的逻辑构成了一个有向无环图.

图 15 描述了一个统计 Top K 热点单词的实例.

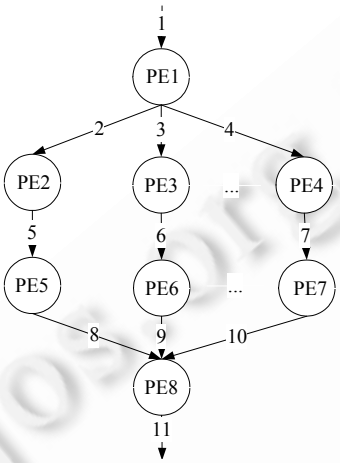


Fig.15 Task topology instance

图 15 任务拓扑实例

在图 15 所示的有向无环图中,节点表示处理单元 PE,实现对数据流的计算和新数据流的输出,有向边表示事件的有序序列(K,A)及其流向.在该实例中,实现了对于流式数据中的 Top K 热点单词的统计,其数据流的具体内容见表 4,其中,数据流 1 是初始化数据流,因此其主键值为空,键值为实时流入的文本数据,在处理单元 PE1 中被分割为各个单词,形成了新的数据流,其事件类型为单词统计,主键为 word=x,键值为 count=y,并分别分流到处理单元 PE2、处理单元 PE3、处理单元 PE4 等节点中进行计算,并再次形成了新的数据流,其事件类型为单词数更新,主键为 SortID=x,键值为 word=y,count=z,并分别分流到处理单元 PE5、处理单元 PE6、处理单元 PE7 等节点中进行计算,最后在处理单元 PE8 中进行汇总和排序,得出当前的 Top K 个热点单词.

Table 4 Data stream content

表 4 数据流内容

数据流	事件类型	主键	键值
1	查询	无	I mean what I said and I said what I mean …
2	单词统计	word=said	count=2
3	单词统计	word=what	count=2
4	单词统计	word=I	count=4
5	单词数更新	SortID=2	word=said, count=8
6	单词数更新	SortID=5	word= what, count=17
7	单词数更新	SortID=10	word=said, count=57
8,9,10	汇总降序	TopK=100	word=[w:cnt]
11	输出	无	total-count

(3) 处理节点 Pnode

在 S4 的处理节点 Pnode 中,如图 16 所示,由处理空间和传输空间组成,其中,

- 在处理空间中,事件监听系统主要用于监听并分发接收到的事件计算请求,并由调度分配系统将事件分配到处理单元集 PEC(processing element container)上进行计算,处理单元集 PEC 以适当的顺序调用适当的处理单元 PE,并保证每个主键 *key* 的处理单元 PE 都会被映射到一个确定的处理节点 Pnode 上.之后,处理节点 Pnode 或者发出输出事件,或者向传输层请求协助,向指定逻辑节点发送消息.其中,处理单元集 PEC 由一个处理节点 Pnode 中内部的多个处理单元 PE 组成.处理单元 PE 是事件计算的最小单元,接受一个或多个来自于事件源或其他处理单元PE的事件进行计算,之后,分发一个或多个计算后的事件到其他处理单元 PE 或输出结果.各个处理单元 PE 间相互独立,它们之间通过事件构成关联,事件在各处理单元 PE 间以数据流的形式进行传输;
- 在传输空间中,主要通过路由管理、负载均衡、集群管理、容错管理等实现对事件流的路由选择、负载均衡、逻辑影射、故障恢复到备用节点等方面的管理和功能,并通过 Zookeeper 系统在 S4 集群节点间实现一致性协作.S4 通过插件式的架构来动态选择信息传输协议,对于控制信息,通常采用可靠传输协议,如 TCP,保障控制信息传输的可靠性.对于数据信息,通常采用不可靠传输协议,如 UDP,保障数据信息的高吞吐量.

(4) 系统架构

S4 采用了对等式系统架构,如图 17 所示.

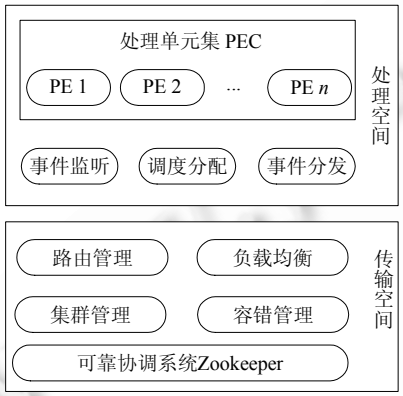


Fig.16 Processing node PNode

图 16 处理节点 PNode

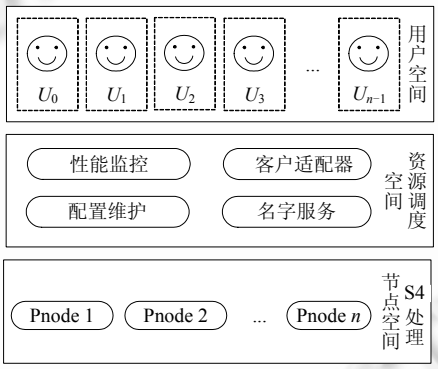


Fig.17 S4 architecture

图 17 S4 系统结构

在一个 S4 系统中,由用户空间、资源调度空间和 S4 处理节点空间组成,其中,在用户空间中,多个用户可以通过本地的客户端驱动实现服务的请求访问;在资源调度空间中,为用户提供了客户适配器,通过 TCP/IP 协议实现用户的客户端驱动与客户适配器间的连接和通信,多个用户可以并发地与多个客户适配器进行服务请求;在 S4 处理节点空间中,提供了多个处理节点 Pnode,进行用户服务请求的计算.各个处理节点间保持相对的独立性、对等性和高并发性,极大地提高了系统的性能,并通过 Hash 方式将事件路由到一个或多个目标处理节点 Pnode 上.

(5) 存在不足

S4 系统存在的不足主要包括:当数据流到达速度超过一定界限时,到达速度越高,系统数据处理的错误率越大;不支持系统节点的热插拔,所有对节点的调整都必须离线进行;仅支持部分容错,即,节点失效转移时会丢失原节点内存中的状态信息.

3.3 Data Freeway and Puma系统

Data Freeway and Puma[29,37,50−53]是 Facebook 支持开发的一款基于 Hive/Hadoop 的、分布式的、高效率的、数据传输通道和大数据流式计算系统.

(1) Data Freeway 系统

Data Freeway 是 Facebook 支持开发的一款可扩展数据流架构(scalable data stream framework),可以有效地支持 4 种数据间的传输,即,文件到文件、文件到消息、消息到消息和消息到文件.其系统结构如图 18 所示,Data Freeway 数据流架构由 4 个组件构成,即,Scribe,Calligraphus,Continuous Copier 和 PTail.Scribe 组件位于用户端,其功能是将用户的数据通过 RPC 发送到服务器端;Calligraphus 组件实现了对日志类型的维护与管理,其功能是通过 Zookeeper 系统,将位于缓冲区中的数据并发写到 HDFS 中;Continuous Copier 组件的功能是实现在各个 HDFS 系统间进行文件的迁移;PTail 组件实现了并行地将文件输出.

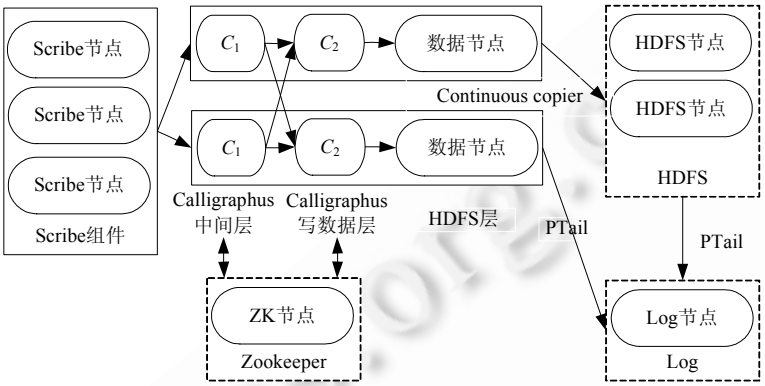


Fig.18 Data Freeway architecture

图 18 Data Freeway 系统架构

(2) Puma 系统

Puma 是 Facebook 的可靠数据流聚合引擎(reliable stream aggregation engine)系统,如图 19 所示,当前最新版本为 Puma3 系统.

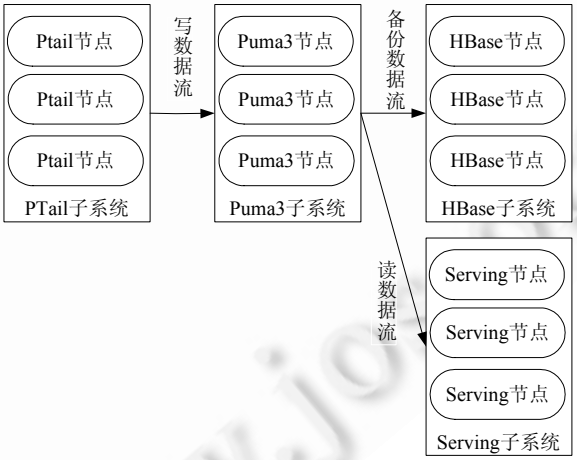


Fig.19 Puma3 architecture

图 19 Puma3 系统架构

Puma3 在本地内存中实现了数据聚合功能,极大地提高了数据的计算能力,有效地降低了系统延迟.Puma3 系统实现时,在 Calligraphus 阶段通过聚合主键完成对数据的分片,其中,每个分片都是内存中的哈希表,每个表项对应一个 Key 及用户定义的聚合方法,如统计、求和、平均值等操作.HBase 子系统会定期地从 Puma3 中将内存中的数据备份到 HBase 中,进行数据的持久化存储.只有当 Puma3 发生故障时,才从 HBase 中读取副本,进

行数据的重放,实现对因故障丢失数据的恢复;在无故障的情况下,HBase 子系统不参与数据的计算,因此提高了数据的计算能力.

(3) 存在不足

Data Freeway and Puma 系统存在的不足主要包括:数据延迟在秒级,无法满足大数据流式计算所需要的毫秒级应用需求;将哈希表完全放入内存的加速机制,导致内存需求量大;资源调度策略不够简单、高效,不能灵活适应连续的工作负载.

3.4 Kafka系统

Kafka[38,54−56]是 Linkedin 所支持的一款开源的、分布式的、高吞吐量的发布订阅消息系统,可以有效地处理互联网中活跃的流式数据,如网站的页面浏览量、用户访问频率、访问统计、好友动态等,最新版本是 Kafka 0.8,开发语言是 Scala,可以使用 Java 进行编写.

Kafka 系统在设计过程中主要考虑到了以下需求特征:消息持久化是一种常态需求;吞吐量是系统需要满足的首要目标;消息的状态作为订阅者(consumer)存储信息的一部分,在订阅者服务器中进行存储;将发布者(producer)、代理(broker)和订阅者(consumer)显式地分布在多台机器上,构成显式的分布式系统.形成了以下关键特性:在磁盘中实现消息持久化的时间复杂度为 $O(1)$,数据规模可以达到 TB 级别;实现了数据的高吞吐量,可以满足每秒数十万条消息的处理需求;实现了在服务器集群中进行消息的分片和序列管理;实现了对 Hadoop 系统的兼容,可以将数据并行地加载到 Hadoop 集群中.

(1) 系统架构

Kafka 消息系统的架构是由发布者(producer)、代理(broker)和订阅者(consumer)共同构成的显式分布式架构,即,分别位于不同的节点上,如图 20 所示.各部分构成一个完整的逻辑组,并对外界提供服务,各部分间通过消息(message)进行数据传输.其中,发布者可以向一个主题(topic)推送相关消息,订阅者以组为单位,可以关注并拉取自己感兴趣的消息,通过 Zookeeper 实现对订阅者和代理的全局状态信息的管理,及其负载均衡的实现.

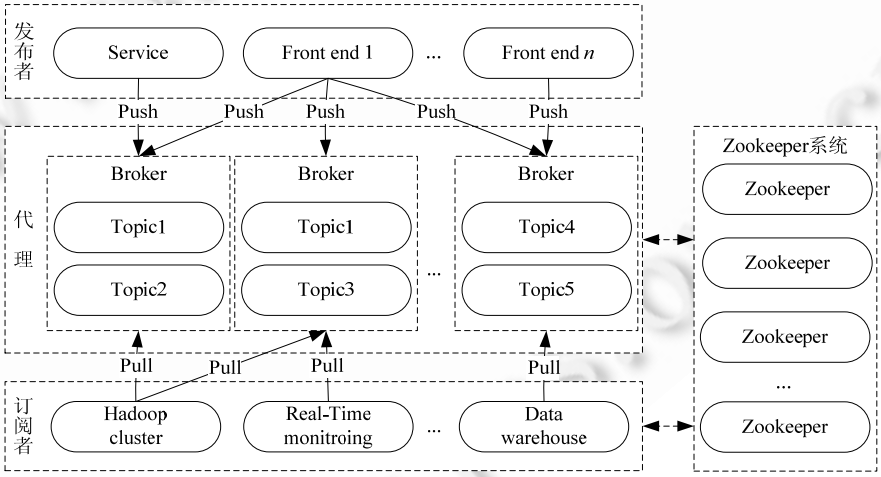


Fig.20 Kafka architecture

图 20 Kafka 系统架构

(2) 数据存储

Kafka 消息系统通过仅仅进行数据追加的方式实现对磁盘数据的持久化保存,实现了对大数据的稳定存储,并有效地提高了系统的计算能力.通过采用 Sendfile[38,54]系统调用方式优化了网络传输,减少了 1 次内存拷贝,提高了系统的吞吐量,即使对于普通的硬件,Kafka 消息系统也可以支持每秒数十万的消息处理能力.此外,在 Kafka 消息系统中,通过仅保存订阅者已经计算数据的偏量信息,一方面可以有效地节省数据的存储空间,另一

方面,也简化了系统的计算方式,方便了系统的故障恢复.

(3) 消息传输

Kafka 消息系统采用了推送、拉取相结合的方式进行消息的传输,其中,当发布者需要传输消息时,会主动地推送该消息到相关的代理节点;当订阅者需要访问数据时,其会从代理节点中进行拉取.通常情况下,订阅者可以从代理节点中拉取自己感兴趣的主题消息.

(4) 负载均衡

在 Kafka 消息系统中,发布者和代理节点之间没有负载均衡机制,但可以通过专用的第 4 层负载均衡器在 Kafka 代理之上实现基于 TCP 连接的负载均衡的调整.订阅者和代理节点之间通过 Zookeeper 实现了负载均衡机制,在 Zookeeper 中管理全部活动的订阅者和代理节点信息,当有订阅者和代理节点的状态发生变化时,才实时进行系统的负载均衡的调整,保障整个系统处于一个良好的均衡状态.

(5) 存在不足

Kafka 系统存在的不足主要包括:只支持部分容错,即,节点失效转移时会丢失原节点内存中的状态信息;代理节点没有副本机制保护,一旦代理节点出现故障,该代理节点中的数据将不再可用;代理节点不保存订阅者的状态,删除消息时无法判断该消息是否已被阅读.

3.5 TimeStream系统

TimeStream[39,57−60]是 Microsoft 在 StreamInsight 的基础上开发的一款分布式的、低延迟的、实时连续的大数据流式计算系统,通过弹性替代机制,可以自适应因故障恢复和动态配置所导致的系统负载均衡的变化,使用 C#.NET 来编写.

TimeStream 的开发是基于大数据流式计算以下两点来考虑的:(a) 连续到达的流式大数据已经远远超出了单台物理机器的计算能力,分布式的计算架构成为必然的选择;(b) 新产生的流式大数据必须在极短的时间延迟内,经过相关任务拓扑进行计算后,产生出能够反映该输入数据特征的计算结果.

(1) 任务拓扑结构

TimeStream 中的数据计算逻辑是基于数据流 DAG 实现的,如图 21 所示,在数据流 DAG 中的每个顶点 v,在获取输入数据流 i 后,触发相关操作 f_v,产生新数据流 o,并更新顶点 v 的状态从 τ 到 τ',即,$(\tau',o)=f_v(\tau,i)$.

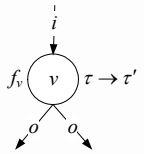


Fig.21 Vertex of task topology in data stream

图 21 数据流任务拓扑顶点

在 TimeStream 中,一个数据流子图 sub-DAG 是指在数据流 DAG 中,两顶点及该两顶点间的全部顶点和有向边的集合,即,满足:对于数据流子图 sub-DAG 中任意两顶点 v_1 和 v_2,以及数据流 DAG 中任意一顶点 v,若顶点 v 位于顶点 v_1 和 v_2 的有向边上,那么顶点 v 一定是数据流子图 sub-DAG 的一个顶点.数据流子图 sub-DAG 在逻辑上可以简化为一个与其功能相同的顶点,如图 22 所示,在一个由 7 个顶点所组成的数据流 DAG 中,由顶点 v_2,v_3,v_4 和 v_5 及其有向边所构成的数据流子图 sub-DAG,可以简化为一个输入数据流为 i、输出数据流为 o 的逻辑顶点.

(2) 弹性等价替代

在 TimeStream 中,当出现服务器故障或系统负载剧烈持续变化的情况时,可以通过数据流子图 sub-DAG 间、数据流子图 sub-DAG 与顶点间以及各顶点间的弹性等价替代,动态、实时地适应系统的负载变化需求.具体而言,弹性等价替代可以进一步细分为 3 种情况:

(a) 顶点间的弹性等价替代.当数据流 DAG 中的任意一顶点 v 出现故障不能正常工作时,系统会启动一

个具有相同功能的顶点 v',并接管顶点 v 的工作;

(b) 数据流子图 sub-DAG 与顶点间的弹性等价替代.如图 22 所示,当整个系统的负载过轻时,为了节省系统的资源,可以通过一个新的顶点 v 代替由顶点 v_2,v_3,v_4 和 v_5 所组成的数据流子图 sub-DAG,该新顶点 v 将实现数据流子图 sub-DAG 的全部功能;反之,当系统的负载过重时,也可以用一个数据流子图 sub-DAG 代替任意一个顶点 v,实现功能的分解和任务的分担;

(c) 数据流子图 sub-DAG 间的弹性等价替代.如图 23 所示,右侧由顶点 v_2,v_3,v_4 和 v_5 所组成的数据流子图 sub-DAG 实现了 HashPartition,Computation 和 Union 等功能,但当系统的 Computation 功能的计算量突然持续增大后,用左侧由顶点 $v_8,v_9,v_{10},v_{11},v_{12}$ 和 v_{13} 所组成的数据流子图 sub-DAG 弹性等价替代右侧的子图,实现了将 Computation 计算节点由 2 个增加到 4 个,提高了 Computation 的计算能力.

通过弹性等价替代机制可以有效地适应系统因故障和负载的变化对系统性能产生的影响,保证系统性能的稳定性;但在弹性等价替代的过程中,一定要实现替代子图或顶点间的等价,并尽可能地进行状态的恢复.所谓的等价,即对于相同的输入,子图或顶点可以在功能上产生相同的输出,唯一存在的区别在于其性能的不同.状态的恢复是通过对数据流 DAG 中的依赖关系跟踪机制[39]来实现,并尽可能全面地进行系统状态的恢复.

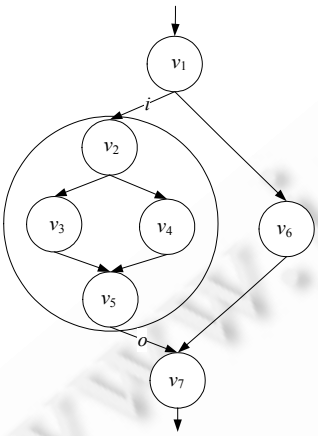


Fig.22 Data stream sub-DAG

图 22 数据流子图 sub-DAG

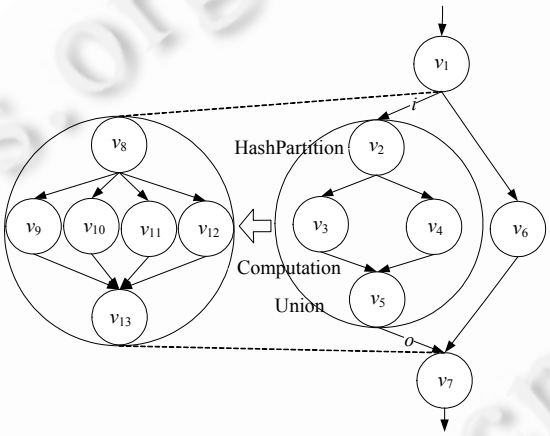


Fig.23 Resilient substitution of data stream sub-DAG

图 23 数据流子图 sub-DAG 弹性等价替代

(3) 系统架构

在 TimeStream 的系统结构中,实现了资源分配、节点调度、故障检测等功能.

如图 24 所示,位于头节点(head node)中的集群管理器(cluster manager,简称 CM)实现了对系统资源的管理和任务的分配,位于计算节点(compute node)的节点服务器(node service,简称 NS)实现了对计算节点的管理和维护.当一个新的数据流任务进入系统被计算时:首先,系统为该任务分配一个全局唯一的查询协调器(query coordinator,简称 QC),查询协调器 QC 向集群管理器 CM 请求资源运行任务的数据流 DAG;其次,向节点服务器 NS 请求调度顶点处理器(vertex processes,简称 VP),并实现数据流 DAG 的构建;再次,实施数据计算;最后,查询协调器 QC 和顶点处理器 VP 均会实时地跟踪系统的运行情况,并定期地将相关元数据信息保持到数据库中,在出现系统故障或负载剧烈持续变化的情况时,可以通过这些被永久保存的元数据进行系统状态的恢复和实时动态的调整.

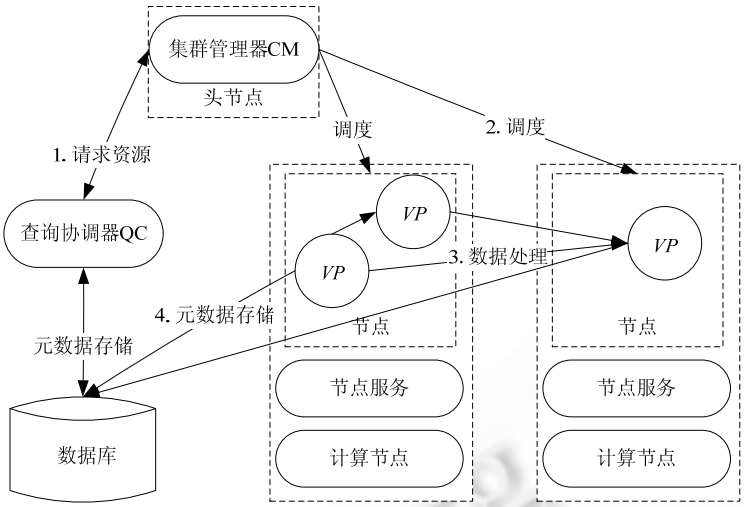


Fig.24 TimeStream architecture

图 24 TimeStream 系统架构

(4) 存在不足

TimeStream 系统存在的不足主要包括:数据延迟在秒级,无法满足毫秒级的应用需求;基于依赖关系跟踪的容错机制降低了系统性能,当系统规模为 16 个节点时,系统吞吐量下降了 10%左右.

3.6 对比分析

表 5 从 13 个主要方面对 Storm 系统、S4 系统、Data Freeway and Puma 系统、Kafka 系统和 TimeStream 系统进行了对比分析.

Table 5 Contrast of data stream systems

表 5 数据流系统对比

性能指标	Storm 系统	S4 系统	Data Freeway and Puma 系统	Kafka 系统	TimeStream 系统
系统架构	主从	对称	对称	主从	主从
数据传输	拉取	推送	推送	推送拉取	拉取
应用接口	MR 接口	MR 接口	MR 接口	MR 接口	SQL 接口
高可用性	上游备份策略	被动等待策略	主动等待策略	被动等待策略	上游备份策略
开发语言	Clojure, Java	Java	Java	Scala	C#.NET
容错机制	作业级容错	部分容错	部分容错	部分容错	依赖关系跟踪
精确恢复	否	否	否	否	是
资源利用率	高	低	低	低	高
状态持久化	否	是	是	否	是
数据去重	否	否	是	否	否
编程模型	纯编程	编程+XML	纯编程	纯编程	纯编程
负载均衡	不支持	不支持	不支持	部分支持	支持
典型应用	社交网络	广告投放	站点统计	好友动态	微博情感分析

可以看到:

- 在体系结构方面:Storm,Kafka,TimeStream 选择了主从式体系结构,S4 和 Data Freeway and Puma 均选择了对称式体系结构;
- 在应用接口方面:Storm,S4,Puma,Kafka 均选择了类 MapReduce 接口,简化了用户的编程;TimeStream 选择了用户更为熟悉的类 SQL 接口.此外,HStreaming 已为用户提供了更为方便的基于拖拽的可视化接口;
- 在开发语言方面:S4 和 Puma 均选择了 Java 语言;Storm 的核心代码虽然选择了 Clojure 语言,但也支持

Java 语言;

- 在高可用策略方面:S4 和 Kafka 均选择了被动等待策略,因此其资源利用率比较低;Data Freeway and Puma 选择了主动等待策略;Storm,TimeStream 选择了上游备份策略,相应的资源利用率比较高;
- Storm,S4,Data Freeway and Puma 和 Kafka 目前均不支持数据的精确恢复、负载均衡等功能,但面向金融领域的 StreamBase 支持数据的精确恢复.

如图 25 所示,批量计算相关的大数据系统,如批量处理系统(如 MapReduce)、大规模并行数据库等,在数据吞吐量方面具有明显优势,但在系统响应时间方面往往在秒级以上.而当前的流式计算相关的大数据系统,如流式处理系统、内存数据库、CEP(复杂事件处理)等,在系统响应时间方面虽然维持在毫秒级的水平,但数据吞吐量往往在 GB 级别,远远满足不了大数据流式计算系统对数据吞吐量的要求.通常情况下,一个理想的大数据流式计算系统在响应时间方面应维持在毫秒级的水平,并且数据吞吐量应该提高到 PB 级及其以上水平.

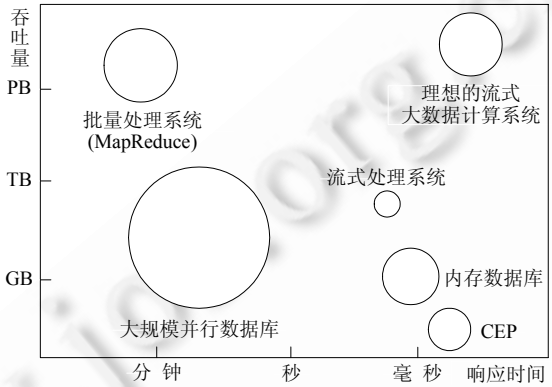


Fig.25 Compare of streaming systems and batch systems in throughput and response time

图 25 流式系统和批量计算系统在吞吐量、响应时间方面的对比

4 面临的技术挑战

流式大数据在实时性、无序性、无限性、易失性、突发性等方面均呈现出了诸多新的鲜明特征,因此,传统的先存储后计算的批量数据计算理念不适用于大数据流式计算的环境中,使得大数据流式环境中的数据计算在系统的可伸缩性、系统容错、状态一致性、负载均衡、数据吞吐量等方面[61−65]均面临着前所未有的新的挑战.

4.1 可伸缩性

在大数据流式计算环境中,系统的可伸缩性是制约大数据流式计算系统广泛应用的一个重要因素.Storm,Kafka,TimeStream 等系统没有实现对系统可伸缩性的良好支持:一方面,流式数据的产生速率在高峰时期会不断增加且数据量巨大,持续时间往往很长,因此需要大数据流式系统具有很好的“可伸”的特征,可以实时适应数据增长的需求,实现对系统资源进行动态调整和快速部署,并保证整个系统的稳定性;另一方面,当流式数据的产生速率持续减少时,需要及时回收在高峰时期所分配的但目前已处于闲置或低效利用的资源,实现整个系统“可缩”的友好特征,并保障对用户是透明的.因此,系统中资源动态的配置、高效的组织、合理的布局、科学的架构和有效的分配,是保障整个系统可伸缩性的基础,同时,又尽可能地减少不必要的资源和能源的浪费.

大数据流式计算环境中的可伸缩性问题的解决,需要实现对系统架构的合理布局、系统资源的有序组织、高效管理和灵活调度,在保证系统完成计算的前提下,尽量少地太久、太多地占用系统资源,通过虚拟化机制实现软、硬件之间的低耦合,实现资源的在线迁移,并最终解决大数据流式计算环境中的可伸缩性问题.

4.2 系统容错

在大数据流式计算环境中,系统容错机制是进一步改善整个系统性能、提高计算结果的满意度、保证系统

可靠持续运行的一个重要措施,也是当前大多数大数据流式计算系统所缺失的.如 S4,Puma,Kafka 等系统实现了对部分容错的支持,Storm 系统实现了对作业级容错的支持,TimeStream 系统通过依赖关系跟踪实现了对容错的部分支持.大数据流式计算环境对容错机制提出了新的挑战:

- 一方面,数据流是实时、持续地到来,呈现出时间上不可逆的特征,一旦数据流流过,再次重放数据流的成本是很大的,甚至是不现实的.由于数据流所呈现出的持续性和无限性,也无法预测未来流量的变化趋势;
- 另一方面,在流式大数据的计算过程中,大部分“无用”的数据将被直接丢弃,能被永久保存下来的数据量是极少的,当需要进行系统容错时,其中不可避免地会出现一个时间段内数据不完整的情况;
- 再则,需要针对不同类型的应用,从系统层面上设计符合其应用特征的数据容错级别和容错策略,避免不必要的资源浪费及应用需求的不吻合.

大数据流式计算环境中的容错策略的确定,需要根据具体的应用场景进行系统的设计和权衡,并且需要充分考虑到流式大数据的持续性、无限性、不可恢复性等关键特征.但是,没有任何数据丢失的容错策略也未必是最佳的,需要综合统筹容错级别和资源利用、维护代价等要素间的关系.但在对系统资源占用合理、对系统性能影响可接受的情况下,容错的精度越高必将越好.

4.3 状态一致性

在大数据流式计算环境中,维持系统中各节点间状态的一致性对于系统的稳定、高效运行、故障恢复都至关重要.然而,当前多数系统不能有效地支持系统状态的一致性,如 Storm,Kafka 等系统尚不支持维护系统状态的一致性,S4,TimeStream 等系统也仅实现了在一定程度上对状态一致性的支持.大数据流式计算环境对状态一致性提出了新的挑战:一方面,在系统实时性要求极高、数据速率动态变化的环境中,维护哪些数据的状态一致性,如何从高速、海量的数据流中识别这些数据是一个巨大的挑战;另一方面,在大规模分布式环境中,如何组织和管理实现系统状态一致性的相关数据,满足系统对数据的高效组织和精准管理的要求,也是一个巨大的挑战.

大数据流式计算环境中的状态一致性问题的解决,需要从系统架构的设计层面上着手.存在全局唯一的中心节点的主从式架构方案无疑是实现系统状态一致性的最佳解决方案,但需要有效避免单点故障问题.通常情况下,在大数据流式计算环境中,程序和数据一旦启动后,将会常驻内容,对系统的资源占用也往往相对稳定.因此,单点故障问题在大数据流式计算环境中并没有批量计算环境中那么复杂.批量计算环境中的很多策略将具有很好的参考和借鉴价值.

4.4 负载均衡

在大数据流式计算环境中,系统的负载均衡机制是制约系统稳定运行、高吞吐量计算、快速响应的一个关键因素.然而,当前多数系统不能有效地支持系统的负载均衡,如 Storm,S4 等系统均不支持负载均衡机制, Kafka 系统实现了对负载均衡机制的部分支持:一方面,在大数据流式计算环境中,系统的数据速率具有明显的突变性,并且持续时间往往无法有效预测,这就导致在传统环境中具有很好的理论和实践效果的负载均衡策略在大数据流式计算环境中将不再适用;另一方面,当前大多数开源的大数据流式计算系统在架构的设计上尚未充分地、全面地考虑整个系统的负载均衡问题,在实践应用中,相关经验的积累又相对缺乏,因此,给大数据流式计算环境中负载均衡问题的研究带来了诸多实践中的困难和挑战.

大数据流式计算环境中的负载均衡问题的解决,需要结合具体的应用场景,系统地分析和总结隐藏在大数据流式计算中的数据流变化的基本特征和内在规律,结合传统系统负载均衡的经验,根据实践检验情况,不断进行相关机制的持续优化和逐步完善.

4.5 数据吞吐量

在大数据流式计算环境中,数据吞吐量呈现出了根本性的增加.在传统的流式数据环境中,如 CEP,所处理的数据吞吐量往往在 GB 级别,满足不了大数据流式计算环境对数据的吞吐量的要求.在大数据流式计算环境中,数据的吞吐量往往在 TB 级别以上,且其增长的趋势是显著的.然而,当前流式数据处理系统,如 Storm,S4 等,均无

法满足 TB 级别的应用需求.

大数据流式计算环境中的数据吞吐量问题的解决,一方面需要从硬件的角度进行系统的优化,设计出更符合大数据流式计算环境的硬件产品,在数据的计算能力上实现大幅提升;另一方面,更为重要的是,从系统架构的设计中进行优化和提升,设计出更加符合大数据流式计算特征的数据计算逻辑.

5 结 论

流式大数据作为大数据的一种重要形态,在商业智能、市场营销和公共服务等诸多领域有着广泛的应用前景,并已在金融银行业、互联网、物联网等场景的应用中取得了显著的成效.但流式大数据以其实时性、无序性、无限性、易失性、突发性等显著特征,使得其与传统批量大数据在数据计算的要求、方式等方面有着明显的不同,也使得当前诸多数据计算系统无法进一步更好地适应流式大数据在系统可伸缩性、容错、状态一致性、负载均衡、数据吞吐量等方面所带来的诸多新的技术挑战.

本文从大数据环境中流式数据的特征切入,以大数据流式计算架构的设计、优化和挑战为核心,系统地梳理和分析了当前大数据环境中的关于大数据流式计算系统的研究和发展现状,从系统架构的角度分析了一个设计优良的大数据流式计算系统应该在系统结构、数据传输、应用接口、高可用技术等诸多关键技术上进行优化.同时,本文详细地分析和对比了当前在实践中具有很好的应用基础、较为典型的 5 款大数据流式计算系统,并具体阐述了大数据流式计算在系统的可伸缩性、系统容错、状态一致性、负载均衡、数据吞吐量等方面所面临的新的挑战,实现了对流式大数据环境中数据计算架构、关键问题及其技术挑战的深入研究.

可以看出,大数据流式计算的研究和应用仍处于很不成熟的阶段,这与其广泛的市场需求和应用前景很不吻合.为了促进大数据流式计算的成熟、稳健发展,亟待全面、系统、深入地开展相关理论和实践的研究工作.在未来的研究工作中,将进一步深化对大数据流式计算架构及其关键技术的研究,并结合详细的应用需求,开发、部署、测试并优化面向特定应用领域的大数据流式计算系统,进一步推动大数据流式计算理论、方法、技术与系统的研究与发展.

致谢 在此,我们向对本文的工作给予支持和建议的老师和同学表示感谢.

References:

[1] Lynch C. Big data: How do your data grow? Nature, 2008,455(7209):28−29. [doi: 10.1038/455028a]

[2] Kobielus A. The role of stream computing in big data architectures. 2013. http://ibmdatamag.com/2013/01/the-role-of-stream-computing-in-big-data-architectures/

[3] Li GJ, Cheng XQ. Research status and scientific thinking of big data. Bulletin of Chinese Academy of Sciences, 2012,27(6): 647−657 (in Chinese with English abstract).

[4] Wang YZ, Jin XL, Cheng XQ. Network big data: Present and future. Chinese Journal of Computers, 2013,36(6):1125−1138 (in Chinese with English abstract).

[5] Feng ZY, Guo XH, Zeng DJ, Chen YB, Chen GQ. On the research frontiers of business management in the context of big data. Journal of Management Sciences in China, 2013,16(1):1−9 (in Chinese with English abstract).

[6] Morales GDF. SAMOA: A platform for mining big data streams. In: Proc. of the 22th Int'l World Wide Web Conf. (WWW 2013). Rio de Janeiro: ACM Press, 2013. 777−778. http://www.engineeringvillage.com/search/doc/detailed.url?SEARCHID=M3862b207144cdd0c07bM34761017816565&pageType=quickSearch&CID=quickSearchDetailedFormat&DOCINDEX=1&database=7&format=quickSearchDetailedFormat&tagscope=&displayPagination=yes

[7] Meng XF, Ci X. Big data management: Concepts, techniques and challenges. Journal of Computer Research and Development, 2013,50(1):146−169 (in Chinese with English abstract).

[8] Lim L, Misra A, Mo TL. Adaptive data acquisition strategies for energy-efficient, smartphone-based, continuous processing of sensor streams. Distributed and Parallel Databases, 2013,31(2):321−351. [doi: 10.1007/s10619-012-7093-3]

[9] Li BD, Mazur E, Diao YL. SCALLA: A platform for scalable one-pass analytics using MapReduce. ACM Trans. on Database Systems, 2012,37(4):1−43. [doi: 10.1145/2389241.2389246]

[10] Yang D, Rundensteiner EA, Ward M. Mining neighbor-based patterns in data streams. Information Systems, 2013,38(3):331−350. [doi: 10.1016/j.is.2012.08.001]

[11] Qin XP, Wang HJ, Du XY, Wang S. Big data analysis—Competition and symbiosis of RDBMS and MapReduce. Ruan Jian Xue Bao/ Journal of Software, 2012,23(1):32−45 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4091.htm [doi: 10.3724/SP.J.1001.2012.04091]

[12] Tallon PP. Corporate governance of big data: Perspectives on value, risk, and cost. Computer, 2013,46(6):32−38. [doi: 10.1109/MC.2013.155]

[13] Talia D. Clouds for scalable big data analytics. Computer, 2013,46(5):98−101. [doi: 10.1109/MC.2013.162]

[14] Chen HC, Chiang RHL, Storey VC. Business intelligence and analytics: From big data to big impact. MIS Quarterly, 2012,36(4): 1165−1188.

[15] Li JZ, Liu XM. An important aspect of big data. Journal of Computer Research and Development, 2013,50(6):1147−1162 (in Chinese with English abstract).

[16] Demirkan H, Delen D. Leveraging the capabilities of service-oriented decision support systems: Putting analytics and big data in cloud. Decision Support Systems, 2013,55(1):412−421. [doi: 10.1016/j.dss.2012.05.048]

[17] Agrawal D, Das S, El AA. Big data and cloud computing: Current state and future opportunities. In: Proc. of the 14th Int'l Conf. on Extending Database Technology (EDBT 2011). Uppsala: ACM Press, 2011. 530−533. [doi: 10.1145/1951365.1951432]

[18] Cugola G, Margara A. Deployment strategies for distributed complex event processing. Computing, 2013,95(2):129−156. [doi: 10.1007/s00607-012-0217-9]

[19] Zappia I, Paganelli F, Parlanti D. A lightweight and extensible complex event processing system for sense and respond applications. Expert Systems with Applications, 2012,39(12):10408−10419. [doi: 10.1016/j.eswa.2012.01.197]

[20] Hoi SCH, Wang JL, Zhao PL, Jin R. Online feature selection for mining big data. In: Proc. of the ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (SIGKDD 2012). Beijing: ACM Press, 2012. 93−100. [doi: 10.1145/2351316.2351329]

[21] Michael K, Miller KW. Big data: New opportunities and new challenges. Computer, 2013,46(6):22−24. [doi: 10.1109/MC.2013.196]

[22] Scalosub G, Marbach P, Liebeherr J. Buffer management for aggregated streaming data with packet dependencies. IEEE Trans. on Parallel and Distributed Systems, 2013,24(3):439−449. [doi: 10.1109/TPDS.2012.65]

[23] Malensek M, Pallickara SL, Pallickara S. Exploiting geospatial and chronological characteristics in data streams to enable efficient storage and retrievals. Future Generation Computer Systems, 2013,29(4):1049−1061. [doi: 10.1016/j.future.2012.05.024]

[24] Cugola G, Margara A. Processing flows of information: From data stream to complex event processing. ACM Computing Surveys, 2012,44(3):15:1−62. [doi: 10.1145/2187671.2187677]

[25] Lim L, Misra A, Mo TL. Adaptive data acquisition strategies for energy-efficient, smartphone-based, continuous processing of sensor streams. Distributed and Parallel Databases, 2013,31(2):321−351. [doi: 10.1007/s10619-012-7093-3]

[26] He JY, Chaintreau A, Diot C. A performance evaluation of scalable live video streaming with nano data centers. Computer Networks, 2009,53(2):153−167. [doi: 10.1016/j.comnet.2008.10.014]

[27] Vianna E, Comarela G, Pontes T, Almeida J, Almeida V, Wilkinson K, Kuno H, Dayal U. Analytical performance models for MapReduce workloads. Int'l Journal of Parallel Programming, 2013,41(4):495−525. [doi: 10.1007/s10766-012-0227-4]

[28] Chatziantoniou D, Pramatari K, Sotiropoulos Y. Supporting real-time supply chain decisions based on RFID data streams. Journal of Systems and Software, 2011,84(4):700−710. [doi: 10.1016/j.jss.2010.12.011]

[29] 杨栋.Beyond MapReduce:谈 2011 年风靡的数据流计算系统.2013.http://www.programmer.com.cn/9642/

[30] Tatbul N, Ahmad Y, Çetintemel U, Hwang JH, Xing Y, Zdonik S. Load management and high availability in the borealis distributed stream processing engine. In: Proc. of the 2nd Int'l Conf. on GeoSensor Networks (GSN 2006). Boston: IEEE Press, 2006. 66−85. [doi: 10.1007/978-3-540-79996-2_5]

[31] Balazinska M, Hwang J, Shah MA. Fault-Tolerance and high availability in data stream management systems. In: Proc. of the Encyclopedia of Database Systems. 2009. 1109−1115. [doi: 10.1007/978-0-387-39940-9_160]

[32] Zhang Z, Gu Y, Ye F, Yang H, Kim M, Lei H, Liu Z. A hybrid approach to high availability in stream processing systems. In: Proc. of the 30th IEEE Int'l Conf. on Distributed Computing Systems (ICDCS 2010). Genova: IEEE Press, 2010. 138−148. [doi: 10.1109/ICDCS.2010.81]

[33] Nagano K, Itokawa T, Kitasuka T, Aritsugi M. Exploitation of backup nodes for reducing recovery cost in high availability stream processing systems. In: Proc. of the 14th Int'l Database Engineering and Applications Symp. (IDEAS 2010). Montreal: ACM Press, 2010. 61−63. [doi: 10.1145/1866480.1866490]

[34] Aritsugi M, Nagano K. Recovery processing for high availability stream processing systems in local area networks. In: Proc. of the IEEE Region 10 Conf. (TENCON 2010). Fukuoka: IEEE Press, 2010. 1036−1041. [doi: 10.1109/TENCON.2010.5686443]

[35] Storm. 2013. http://storm-project.net/

[36] Neumeyer L, Robbins B, Nair A, Kesari A. S4: Distributed stream computing platform. In: Proc. of the 10th IEEE Int'l Conf. on Data Mining Workshops (ICDMW 2010). Sydney: IEEE Press, 2010. 170−177. [doi: 10.1109/ICDMW.2010.172]

[37] Borthakur D, Sarma JS, Gray J, Muthukkaruppan K, Spigeglberg N, Kuang HR, Ranganathan K, Molkov D, Mennon A, Rash S, Schmidt R, Aiyer A. Apache hadoop goes realtime at Facebook. In: Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2011 and PODS 2011). Athens: ACM Press, 2011. 1071−1080. [doi: 10.1145/1989323.1989438]

[38] Apache Kafka, A high-throughput distributed messaging system. 2013. http://kafka.apache.org/design.html

[39] Qian ZP, He Y, Su CZ, Wu ZJ, Zhu HY, Zhang TZ, Zhou LD, Yu Y, Zhang Z. TimeStream: Reliable stream computation in the cloud. In: Proc. of the 8th ACM European Conf. on Computer Systems (EuroSys 2013). Prague: ACM Press, 2013. 1−14. [doi: 10.1145/2465351.2465353]

[40] Stoess J, Steinberg U, Uhlig V, Kehne J, Appavoo J, Waterlang A. A lightweight virtual machine monitor for Blue Gene/P. Int'l Journal of High Performance Computing Applications, 2012,26(2):95−109. [doi: 10.1177/1094342011434815]

[41] Hoppe A, Gryz J. Stream processing in a relational database: A case study. In: Proc. of the 11th Int'l Database Engineering and Applications Symp. (IDEAS 2007). Banff: IEEE Press, 2007. 216−224. http://www.engineeringvillage.com/search/doc/detailed.url?SEARCHID=M3862b207144cdd0c07bM2bd41017816565&pageType=quickSearch&CID=quickSearchDetailedFormat&DOCINDEX=1&database=1&format=quickSearchDetailedFormat&tagscope=&displayPagination=yes

[42] Spark, lightning-fast cluster computing. 2013. http://spark-project.org/

[43] Esper, event stream intelligence: Esper & NEsper. 2013. http://esper.codehaus.org/

[44] Storm wiki. 2013. http://en.wikipedia.org/wiki/Storm

[45] Storm tutorial. 2013. https://storm.canonical.com/Tutorial

[46] Simoncelli D, Dusi M, Gringoli F, Niccolini S. Scaling out the performance of service monitoring applications with BlockMon. In: Proc. of the 14th Int'l Conf. on Passive and Active Measurement (PAM 2013). Hong Kong: IEEE Press, 2013. 253−255. [doi: 10.1007/978-3-642-36516-4_26]

[47] Chauhan J, Chowdhury SA, Makaroff D. Performance evaluation of Yahoo! S4: A first look. In: Proc. of the 7th Int'l Conf. on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC 2012). Victoria: IEEE Press, 2012. 58−65. [doi: 10.1109/3PGCIC.2012.55]

[48] S4, distributed stream computing platform. 2013. http://incubator.apache.org/s4/

[49] Zhong BY. Stream computing StreamBase Yahoo S4 borealis comparis. 2013. http://oracle-abc.wikidot.com/zh:stream-computing-streambase-yahoo-s4-borealis-comparison

[50] Squicciarini AC, Shehab M, Wede J. Privacy policies for shared content in social network sites. VLDB Journal, 2010,19(6):777−796. [doi: 10.1007/s00778-010-0193-7]

[51] Deng DP, Chuang TR, Shao KT, Mai GS, Lin TE, Lennens R, Hsu CH, Lin HH, Kraak MJ. Using social media for collaborative species identification and occurrence: Issues, methods, and tools. In: Proc. of the 1st ACM SIGSPATIAL Int'l Workshop on Crowdsourced and Volunteered Geographic Information (GEOCROWD 2012). Redondo Beach: ACM Press, 2012. 22−29. [doi: 10.1145/2442952.2442957]

[52] Segulja C, Abdelrahman TS. Architectural support for synchronization-free deterministic parallel programming. In: Proc. of the 18th IEEE Int'l Symp. on High Performance Computer Architecture (HPCA 2012). New Orleans: IEEE Press, 2012. 1−12. [doi: 10.1109/HPCA.2012.6169038]

[53] HiC2011—Realtime data streams and Analytics-Data Freeway and Puma-Facebook. 2013. http://ishare.iask.sina.com.cn/f/22023896.html

[54] Auradkar A, Botev C, Das S, *et al*. Data infrastructure at LinkedIn. In: Proc. of the IEEE 28th Int'l Conf. on Data Engineering (ICDE 2012). Arlington: IEEE Press, 2012. 1370−1381. [doi: 10.1109/ICDE.2012.147]

[55] Efficient data transfer through zero copy, zero copy, zero overhead. 2013. https://www.ibm.com/developerworks/linux/library/j-zerocopy/

[56] Kafka, distributed publish-subscribe messaging system. 2013. http://data.linkedin.com/opensource/kafka/

[57] Guo ZY, McDirmid S, Yang M, Zhuang L, Zhang P, Luo YW, Bergan T, Bodik P, Musuvathi M, Zhang Z, Zhou LD. Failure recovery: When the cure is worse than the disease. In: Proc. of the 14th USENIX Conf. on Hot Topics in Operating Systems (USENIX 2013). Santa Ana Pueblo: ACM Press, 2013. 1−6. http://research.microsoft.com/apps/pubs/default.aspx?id=191008

[58] Ali M, Badrish C, Goldstein J, Schindlauer R. The extensibility framework in Microsoft StreamInsight. In: Proc. of the IEEE 27th Int'l Conf. on Data Engineering (ICDE 2011). Hannover: IEEE Press, 2011. 1242−1253. [doi: 10.1109/ICDE.2011.5767878]

[59] Chandramouli B, Goldstein J, Barga R, Riedewald M, Santos I. Accurate latency estimation in a distributed event processing system. In: Proc. of the IEEE 27th Int'l Conf. on Data Engineering (ICDE 2011). Hannover: IEEE Press, 2011. 255−266. http://www.engineeringvillage.com/search/doc/detailed.url?SEARCHID=M3862b207144cdd0c07bM2d0a1017816565&pageType=quickSearch&CID=quickSearchDetailedFormat&DOCINDEX=1&database=1&format=quickSearchDetailedFormat&tagscope=&displayPagination=yes

[60] Ali M, Chandramouli B, Fay J, Wong C, Drucker S, Raman BS. Online visualization of geospatial stream data using the WorldWide telescope. VLDB Endowment, 2011,4(12):1379−1382.

[61] Qin XP, Wang HJ, Li FR, Li CP, Chen H, Zhou H, Du XY, Wang S. New landscape of data management technologies. Ruan Jian Xue Bao/Journal of Software, 2013,24(2):175−197 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4345.htm [doi: 10.3724/SP.J.1001.2013.04345]

[62] Qi KY, Zhao ZF, Fang J, Ma Q. Real-Time processing for high speed data stream over larger scale data. Chinese Journal of Computers, 2012,35(3):477−490 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2012.00477]

[63] Toyoda M, Sakurai Y, Ishikawa Y. Pattern discovery in data streams under the time warping distance. VLDB Journal, 2013,22(3): 295−318. [doi: 10.1007/s00778-012-0289-3]

[64] Malensek M, Pallickara SL, Pallickara S. Exploiting geospatial and chronological characteristics in data streams to enable efficient storage and retrievals. Future Generation Computer Systems, 2013,29(4):1049−1061. [doi: 10.1016/j.future.2012.05.024]

[65] Farid DW, Zhang L, Hossain A, Rahman CM, Strachan R, Sexton G, Dahal K. An adaptive ensemble classifier for mining concept drifting data streams. Expert Systems with Applications, 2013,40(15):5895−5906. [doi: 10.1016/j.eswa.2013.05.001]

附中文参考文献:

[3] 李国杰,程学旗.大数据研究:未来科技及经济社会发展的重大战略领域——大数据的研究现状与科学思考.中国科学院院刊,2012,27(6):647−657.

[4] 王元卓,靳小龙,程学旗.网络大数据:现状与展望.计算机学报,2013,36(6):1125−1138.

[5] 冯芷艳,郭迅华,曾大军,陈煜波,陈国青.大数据背景下商务管理研究若干前沿课题.管理科学学报,2013,16(1):1−9.

[7] 孟小峰,慈祥.大数据管理:概念、技术与挑战.计算机研究与发展,2013,50(1):146−169.

[11] 覃雄派,王会举,杜小勇,王珊.大数据分析——RDBMS 与 MapReduce 的竞争与共生.软件学报,2012,23(1):32−45. http://www.jos.org.cn/1000-9825/4091.htm [doi: 10.3724/SP.J.1001.2012.04091]

[15] 李建中,刘显敏.大数据的一个重要方面:数据可用性.计算机研究与发展,2013,50(6):1147−1162.

[61] 覃雄派,王会举,李芙蓉,李翠平,陈红,周烜,杜小勇,王珊.数据管理技术的新格局.软件学报,2013,24(2):175−197. http://www.jos.org.cn/1000-9825/4345.htm [doi: 10.3724/SP.J.1001.2013.04345]

[62] 亓开元,赵卓峰,房俊,马强.针对高速数据流的大规模数据实时分析方法.计算机学报,2012,35(3):477−490. [doi: 10.3724/SP.J.1016.2012.00477]

孙大为(1985－),男,安徽六安人,博士,助理研究员,CCF 会员,主要研究领域为大数据,云计算.

E-mail: sundaweicn@163.com

郑纬民(1946－),男,教授,博士生导师,CCF 高级会员,主要研究领域为并行与分布式计算,存储系统.

E-mail: zwm-dcs@tsinghua.edu.cn

张广艳(1976－),男,博士,副教授,CCF 会员,主要研究领域为大数据计算,网络存储,分布式计算.

E-mail: gyzh@tsinghua.edu.cn