工程应用

doi: 10.3969/j.issn.1673-5692.2018.02.007

基于 Hadoop、Spark 及 Flink 大规模数据分析的性能评价

代明竹，高嵩峰*

（北京建筑大学，北京 100044）

摘 要：针对目前缺少大型数据分析框架间的横向比较问题，使用有代表性的大数据工作负载，对 Hadoop、Spark 和 Flink 进行性能和可扩展性等因素的比较评价。此外，通过调整工作负载的一些主要参数，例如 HDFS 块大小、输入数据大小、互连网络或线程配置等，描述了这些框架的行为模式特征。实验结果分析表明，对于非排序的基准测试程序，使用 Spark 或 Flink 替代 Hadoop，分别带来平均 77% 和 70% 执行时间的降低。整体上，Spark 的性能结果最好；而 Flink 通过使用的显式迭代程序，极大提高了迭代算法的性能。

关键词：大数据；分析框架；基准测试程序；模型

中图分类号：TP311.13；TP311.52　　**文献标识码**：A　　**文章编号**：1673-5692(2018)02-149-07

Framework Performance Evaluation Based on Hadoop, Spark and Flink Large-scale Data Analysis

DAI Ming-zhu, GAO Song-feng*

(Beijing University of Civil Engineering and Architecture, Beijing 100044, China)

Abstract: In view of the lack of lateral comparison between large data analysis frameworks, the representative big data workload and the factors such as performance and scalability is considered for comparing and evaluating Hadoop, Spark and Flink, which fills gaps in research. In addition, describing the characteristics of these frameworks' behavior patterns by adjusting some main parameters of workload, such as HDFS block size, input data size, interconnection network or thread configuration. The experimental results show that for non sorting benchmark programs, the use of Spark or Flink instead of Hadoop brings average execution time reduction of 77% and 70%, respectively. On the whole, Spark has the best performance results. And the performance of the iterative algorithm is greatly improved by the explicit iterative program used by Flink.

Key words: Big data; Analytical Framework; Benchmarking Program; Model

0 引 言

过去十年中，大数据[1]知识已经得到了社会的普遍认同和采用，这主要得益于提供给终端用户强大功能的新技术，使得终端用户[2]的关注点放在数据执行的转换上，而非算法的并行化方面。

目前，很多研究学者在大数据的处理方面进行了很多研究。如 Apache Hadoop[3]就是处理大数据的技术，即一个开源的 MapReduce 模型。Hadoop 的

收稿日期：2018-02-28　　修订日期：2018-04-01

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

成功主要得益于其并行化抽象性、容错性和可扩展性的体系结构,该结构支持分布式存储和大型数据集[4]的处理。然而,Hadoop 对数据进行处理时所执行的冗余内存复制和磁盘操作[5],对其性能造成了严重的限制。文献[6]综述了 12 个典型的基于 MapReduce 的大数据处理平台,分析对比它们的实现原理和适用场景,抽象其共性;介绍基于了 MapReduce 的大数据分析算法,包括搜索算法[7]、数据清洗/变换算法等,将这些算法按照 MapReduce 实现方式分类,分析影响算法性能的因素;将大数据处理算法抽象为外存算法,并对外存算法的特征加以梳理。

Apache Spark[8]和 Apache Flink[9]具有易于使用的 API,高级数据处理操作符和增强的性能,因此得到了更多的关注。文献[10]对 Hadoop 与 Spark 的应用场景进行了分析,阐述了 Hadoop 与 Spark 各自所适应的应用场景。虽然 Spark 和 Flink 的开发人员都给出了性能试验的结果,但目前缺乏在这些框架之间公平的横向比较。此类分析对于识别当前技术的优缺点来说非常重要,且有助于开发者确定未来的大数据系统的最优特性。

因此,本文旨在对 Hadoop,Spark 和 Flink 进行同等条件下的性能评估。本文主要工作总结如下:1)通过各种批处理和迭代的工作负载,对 Hadoop,Spark 和 Flink 的性能做出比较评价;2)描述了一些实验参数在整体性能上的影响特征。结果表明,Spark 性能最优,是更大的 Apache 项目,其框架更加成熟,且在市场份额和社会关注方面均占据优势。然而,Flink 包含了一些有趣和新颖的设计理念,且部分理念得到了 Spark 的采用。Flink 通过一个自定义的对象序列化程序,实现了持久性内存管理的透明化使用,减少了垃圾收集的开销,而且 Flink 使用的显式迭代程序,极大提高了迭代算法的性能。

1 三种大型数据处理方法介绍

1.1 Hadoop

作为 MapReduce 模型的实际标准实现,Hadoop 已经被很多机构广泛采用,以存储和计算大型数据集。包括两个组件:1) Hadoop 分布式文件系统(HDFS);2) Hadoop MapReduce 引擎。MapReduce 模型基于两个用户定义函数,即映射(map)和约减(reduce),这两个函数计算由键值对表示的数据记录。映射函数提取每个键值对的相关特征,而约减函数则使用这些特征得到期望的结果。

1.2 Spark

Spark 提供了用户能够执行的数据转换的多样性,同时依然包括了一些用于基于键的计算的操作符,这使得 Spark 特别适用于实施经典的、基于键的 MapReduce 算法。Spark 的编程模型以被称为“弹性分布式数据集”(RDD)的抽象概念为基础,RDD 将数据对象保留在内存中,以降低磁盘和网络操作带来的开销[8]。此类处理方式特别适用于在同一数据集上进行多次转换的算法,例如迭代算法等。通过在内存中对中间结果进行存储,Spark 避免了迭代之间的 HDFS 使用,由此实现了对此类工作负载的性能优化。

1.3 Flnik

Flink 通过使用内存处理技术,改善了 Hadoop 的性能。其使用的高效内存数据结构中,包含序列化的数据而非 Java 对象,避免了过多的垃圾收集。Flink 进行批处理的编程模型以 DataSet 的概念为基础,DataSet 通过高阶操作进行转换。与 Spark 不同,Flink 代表着一个真正的流式处理引擎,因为其能够从一个操作到另一个操作、按元组逐个发送数据,而不需要在批处理中执行计算。对于批处理,批量数据被视为流数据的有限集合。Flink 中包括显式迭代操作符,应用了批量迭代器,以得到完整的 DataSets,而增量迭代器则仅被应用于在上一次迭代地过程中发生了变化的数据项上。

2 实验设置及结果分析

2.1 实验设置

2.1.1 测试平台配置

本文在 DAS-4[11]上执行了评估实验,DAS-4 是一个多核集群,通过 InfiniBand(IB)和千兆以太网(GbE)互连。表 1 给出了这个系统主要的硬件和软件特性。每个节点包括 8 个核心,14 G 内存和 2 个 1TB 的磁盘。

本文使用大数据评价工具(BDEv)进行实验,该工具是 MapReduce 评价程序的改进版本。BDEv 可以自动化实施框架的配置、输入数据集的生成、实验的执行、以及结果的采集。

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

表 1　DAS－4 节点配置

硬件配置	
CPU	2 个 Intel Xeon E5620 Westmere
CPU 速度/Turbo	2.4 GHz/2.66 GHz
核心数量	8
内存	24 GB DDR3
磁盘	2 个 1 TB 的 HDD
网络	IB(40 Gbps) 和 GbE
软件配置	
OS 版本	CentOS 发行版 6.6
内核	2.6.32－358.18.1.E16.x86_64
Java	Oracle JDK 1.8.0_25

2.1.2　框架

在软件设置方面,本文的实验评价中使用了稳定版本的 Hadoop(2.7.2)、Spark(1.6.1) 和 Flink(1.0.2)。Spark 和 Flink 均以独立模式部署在 HDFS 2.7.2 中。本文根据相对应的用户指南和系统特性对所有框架进行了精心配置。

表 2 给出了本文实验使用的框架配置。除了 GbE 实验之外,所有框架的网络接口被配置于使用 IP over InfiniBand(IPoIB)。

表 2　Hadoop,Spark 和 Flink 框架的配置

Hadoop		Spark		Flink	
HDFS 数据块大小	128 MB	HDFS 数据块大小	128 MB	HDFS 数据块大小	128 MB
副本因子	3	副本因子	3	副本因子	3
映射器/约减器的堆大小	3.3 GB	执行器的堆大小	18.8 GB	TaskManager 堆大小	18.8 GB
每节点映射器	4	每节点 Worker 数	1	每节点 TaskManager 数量	8
每节点约减器	4	Worker 核心数	8	TaskManager 核心数	8
shuffle(置乱) 并行复制	20			每节点网络缓冲区	512
IO 排序 MB	600 MB			TaskManager 内存预分配	失败
IO 排序溢出百分比	80%			IO 排序溢出百分比	80%

2.1.3　基准

表 3 给出了实验使用的基准及其特征,例如 CPU 限制、I/O 限制(磁盘和网络) 或迭代次数。其中第三列和第四列给出了输入数据集的大小和使用的数据集生成器。其中还包括了基准代码来源。因此,每个框架以相同的算法为基础,使用一个基准实施,从 HDFS 读取相同的输入,并向 HDFS 写入相同的输出。虽然算法保持不变,每个框架根据其可用的功能,采用一个优化后的版本。因此,每个基准使用相同算法的不同实施,以得到相同的结果。

表 3　基准来源

基准	特征	输入数据大小	输入生成器	Hadoop	Spark	Flink
WordCount	CPU 限制	100 GB	RandomTextWriter	Haddop ex.	改编自 ex.	改编自 ex.
Grep	CPU 限制	10 GB	RandomTextWriter	Haddop ex.	改编自 ex.	改编自 ex.
TeraSort	I/O 限制	100 GB	TeraGen	Haddop ex.	改编自文献 [12]	改编自文献 [12]
连接组件	迭代(8 次)	9 GB	DataGen	Pegasus	Graphx	Gelly
PageRank	迭代(8 次)	9 GB	DataGen	Pegasus	改编自 ex.	改编自 ex.
K 均值	迭代(8 次)	26 GB	GenKMenasDataset	Mahout	MLlib	改编自 ex.

(1) WordCount: 对输入数据集中每个词语出现进行计数。Hadoop 发行版中,WordCount 及其输入数据生成器,RandomTextWriter 均作为一个样例(表中的“ex.”) 给出。

(2) Grep: 对输入数据集中正则表达式的匹配进行计数。Hadoop 发行版中包括了 Grep,Spark 和 Flink 的源代码则改编自样例。Grep 的数据生成器也是 RandomTextWriter。

(3) TeraSort: 对大小为 100 字节的键值元组进行排序。Hadoop 中已经包括了 TeraSort 和 TeraGen 数据生成器的实现。然而,TeraSort 没有作为一个样例提供给 Spark 和 Flink,其源代码源于文献[12]。

(4) 连接组件: 寻找一个图形的连接组件[13] 的迭代式图算法。其被囊括在 Pegasus[14] 中,Pegasus 是建立在 Hadoop 之上的一个图形挖掘系统。而对

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

于 Spark 和 Flink，连接组件则分别由 Graphx[14] 和 Gelly[15] 提供支持，Graphx 和 Gelly 均为面向图形的 API。使用 DataGen 工具对输入数据集进行设置，该工具被包括在 HiBench 基准套件中。

(5) PageRank：通过统计链接到每个元素的其他元素的数量和质量，对元素进行评级的迭代图算法。Pegasus 中包含了用于 Hadoop 的 PageRank，用于 Spark 和 Flink 的源代码则来自于该样例。

(6) K 均值：将一组样本划分到 K 个聚类[16] 中的迭代聚类算法。Apache Mahout[17] 在 Hadoop 上实现了这一算法，并给出了数据集生成器，GenKmeansDataSet，而 Spark 则使用了其机器学习库 MLlib 提供的高效实现。

2.1.4 实验评价

为了进行全面的实验分析，本文的评价包括了两个方面：框架的性能和一些配置参数的影响。使用 13、25、37、49 个节点执行了测试基准。每个集群大小 n 表示 1 个主节点和 $n-1$ 个从属节点。表 3 中给出了每个基准的输入数据大小。

实验中使用了不同的 HDFS 数据块大小、输入数据大小，并考虑到了最大集群规模下的网络互连和线程配置。本文在实验中选取了三个基准：WordCount、TeraSort 和 PageRank，分别代表着三种类型的工作负载：CPU 限制、IO 限制和迭代。

使用上述相同大小的输入数据，分别对数据块大小为 64、128、256 和 512 MB 的 HDFS 进行了评价(见表 3)。在数据规模实验中，WordCount 和 TeraSort 处理了 100、150、200 和 250 GB 的数据，而 PageRank 则处理了 9、12.5、16 和 19.5 GB 的数据。本文实验评价的网络互连为 GbE 和 IPoIB，使用每个接口对网络进行配置，以进行 shuffle 操作和 HDFS 复制。这些实验都使用了最大化的数据规模，以最大限度提高测试基准的计算要求，其中 WordCount 和 TeraSort 的最大数据为 250 GB，PageRank 的最大数据为 19.5 GB。

网络的线程配置决定了每个节点的计算资源被分配到 Java 进程和线程的方式。另一方面，Hadoop 在映射器和约减器之间分配 CPU 核心，其中映射器和约减器均为单线程进程。另一方面，Spark 和 Flink 则分别使用 Worker 和 Taskmanager 作为并行处理多个任务的多线程管理器。本文的评价试验中，每个管理器的管理器数量/核心数量的配置为 1/8，2/4，4/2 和 8/1。

2.2 实验结果

2.2.1 性能和可扩展性

所有基准的执行时间如图 1 所示。这些图证明了与 Hadoop 相比，Spark 和 Flink 实现了重要的性能改进。当使用最大的集群规模时，Spark 在 WordCount 和 K 均值中取得了最优的结果，Flink 则在 PageRank 中表现较佳。Spark 和 Flink 在 Grep、TeraSort 和连接组件上则得到了相似的结果。

Spark 在 WordCount 中得到了最好的结果，这是因为其 API 提供了 *reduceByKey*() 函数(按键约减函数)，对每个词语的出现次数进行加和。Flink 则使用 *groupBy*().*sum*() 方法(分组后加和)，该方法针对此类工作负载的优化较差。此外，与其他基准相比，WordCount 的 CPU 约束性让 Flink 的内存优化变得不太明显，甚至在计算结果时带来了一定量的额外开销。在 Grep 中，Spark 和 Flink 的性能大幅超越了 Hadoop。最重要的原因是 MapReduce 的 API 不适用于这一基准。在 Hadoop 中，该基准使用两个 MapReduce 作业：一个用于搜索模式，另一个对结果进行排序。这使得生成并写入 HDFS 的内存副本的数量非常大。Spark 和 Flink 采用的方法则与之不同，其依靠 *filer*() 函数(过滤函数)对输入行进行匹配，而不用对其进行复制。接下来，对选出的行进行计数，并在内存中排序。此外，Hadoop 在 *map*() 函数内执行模式匹配，该函数仅能使用一半的节点 CPU 核心。在 Spark 和 Flink 中，所有操作的并行度被设为集群中的 CPU 核心的总数量。

在 TeraSort 基准中，Hadoop 与 Spark 和 Flink 表现出了最小的性能差异。这主要是因为 Hadoop 最初就是针对排序而设计，这是 MapReduce 引擎的核心组件之一。尽管 Spark 和 Flink 的性能优于 Hadoop，但 Hadoop 的高扩展性使其可以得到较好的结果，特别是当使用的节点数量为 49 个的时候。Spark 和 Flink 的在此基准上的实验统计结果则不相上下。

对于迭代算法(图 1)，Spark 和 Flink 的性能明显优于 Hadoop。如上所述，Flink 和 Spark 都提供了图算法 Graphx 和 Gelly 的优化库，这两个框架的连接组件的实验结果非常相似。与之不同，PageRank 的实现则是从样例中推导出的。在该基准中，Flink 得到了最优的性能，这主要是由于其使用了增量迭代，即仅处理那些没有达到最终值的元素。但是，Spark 在 K 均值上得到了最优的结果，这得益于其优化的 MLlib 库。

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

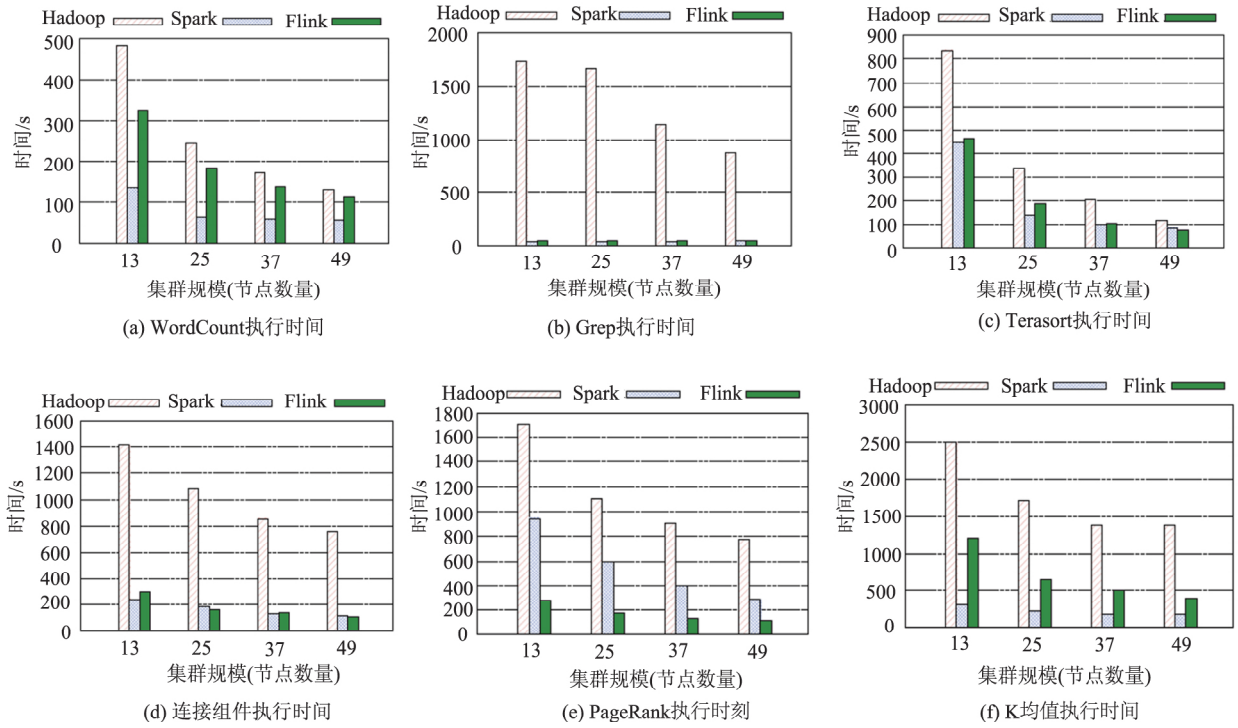

图 1　实验性能结果

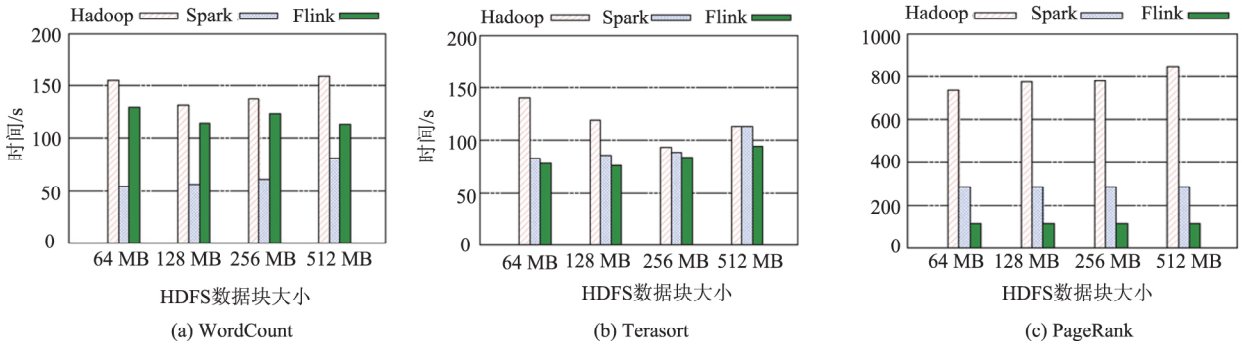

图 2　不同 HDFS 数据块大小的影响

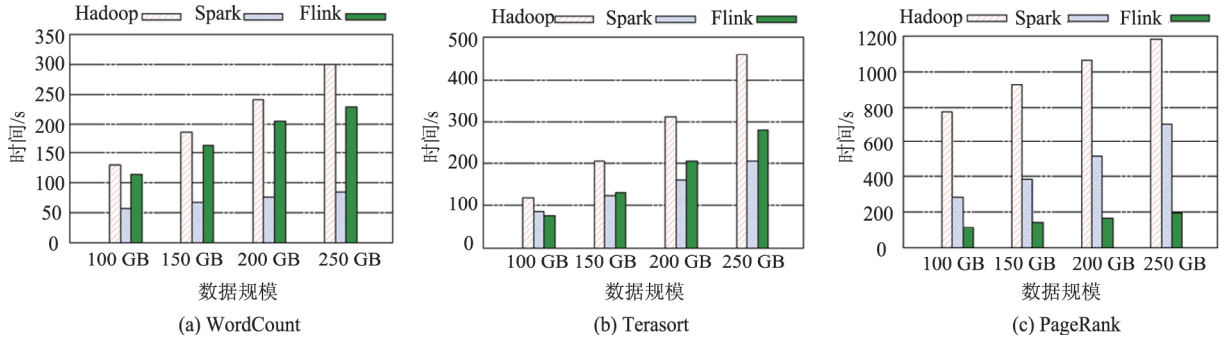

图 3　执行不同规模问题时的时间比较

2.2.2　配置参数的影响

(1) HDFS 数据块大小: 图 2 给出了使用不同的 HDFS 数据块大小所得到的性能数值。HDFS 在一定大小的数据块中对数据进行管理，数据块的大小决定了每个任务中读取的数据量。在 WordCount 基准的测试中，数据块的大小对性能的影响不大，唯一

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

表 4 不同网络配置的运行时间比较(秒)

	Hadoop		Spark		Flink	
	Gbe	iPolB	Gbe	iPolB	Gbe	iPolB
WordCount	300	300	120	80	230	230
Terasort	520	450	360	200	320	280
PageRank	1400	1190	900	750	200	200

表 5 不同线程配置的运行时间比较(s)

	WordCount				Terasort				PageRank			
	4/4	5/3	6/2	7/1	4/4	5/3	6/2	7/1	4/4	5/3	6/2	7/1
Hadoop	300	300	300	250	490	520	700	1100	1200	1350	1510	2400
Spark	100	100	150	300	200	250	300	510	700	600	580	750
Flink	280	270	400	0	300	300	550	0	200	150	200	0

的例外是 Hadoop,其在数据块被配置为 128MB 时得到了最优的结果。Spark 在数据块设为 64MB 时得到了最优的结果,而 Flink 则在数据块被设为 128 MB 和 512 MB 时取得了几乎相同的结果。数据块大小对 TeraSort 基准的影响较大,而最优的数值则取决于框架: Hadoop 为 256 MB,Spark 和 Flink 为 64 MB。在 PageRank 基准中,Hadoop 在数据块设为 64 MB 时得到了最优的结果,其性能与数据块的大小成反比。在该基准中,Spark 和 Flink 没有受到 HDFS 数据块大小的影响。Hadoop 在 HDFS 中存储中间结果,因此其受到数据块大小配置的影响较大。Spark 和 Flink 将中间结果存储在内存中,因此仅在读取初始数据和写入最终输出时,会受到 HDFS 配置的影响。

(2) 输入数据的规模: 图 3 给出了执行不同规模问题时的性能。在 WorkCount 基准中,Hadoop 和 Flink 的曲线变化幅度要大于 Spark,因此对于 WorkCount 基准来说,Spark 的扩展性更好。TeraSort 基准的测试中也是如此,且随着输入数据的增大,Spark 与 Flink 之间的差距也随之变大。因此,对于 TeraSort 基准,Spark 是最具扩展性的框架。在 PageRank 基准测试中,Flink 表现出了高于 Hadoop 和 Spark 的可扩展性,后两者的变化幅度更大。由于 Flink 使用的增量迭代避免了对整个数据集的重新处理。而且 Flink 通过高效内存管理避免了主要的垃圾收集,使得 Flink 成为 PageRank 测试中扩展性最好的框架。

(3) 互连网络: 表 4 给出了当使用 GbE 和 IPoIB 时三个框架的性能。网络不仅会影响到 shuffle 阶段节点间的通信,而且在向 HDFS 进行写入操作的过程中也会造成影响。在 WordCount 中,Spark 是唯一从 IPoIB 的使用中获得略微的性能提升的框架。TeraSort 是本文实验评价中,网络密集程度最高的基准,其中所有框架的性能都得到了提升。在 TeraSort 中使用 IPoIB 分别给 Hadoop,Spark 和 Flink 带来了 12%,41% 和 11% 的性能改善。在 PageRank 中,当使用 IPoIB 时,Hadoop 和 Spark 都出现了性能提升,而 Flink 的数值则保持不变。因此,IPoIB 提供的高带宽对基于块的流水线数据处理的 Spark 的适用性要高于基于元组的 Flink。

(4) 线程配置: 表 5 给出了在不同的线程配置下,三个框架的性能。在 Hadoop 中,最佳配置为 4 个映射器和 4 个约减器,WordCount 是一个例外,其中的最佳配置为 7 个映射器和 1 个约减器。这是因为 WordCount 的 CPU 约束的行为,其中大部分计算通过映射器执行,因此增加映射器的数量能够减少执行时间。Spark 中,8 核心/1 个 Worker 是最优配置,PageRank 则是一个例外,其中最优配置为 2 核心/4 个 Worker。Spark 采用相同的 JVM 对 PageRank 的不同迭代进行计算,涉及到了大量的目标创建/破坏。因此,使用更小的 JVM 能够降低垃圾收集停止的开销。然而,这样也会降低每个 Worker 中的并行度,并对在其中计算的一些服务进行复制。尽管该配置带来的性能很差,但 Spark 成功执行了所有的基准。不同于 Spark,当在 Flink 中配置 8 个任务管理器/1 个核心时,实验没有顺利完成完成(内存不足)。从中可以看到,Flink 中不适合采用较小的 JVM。在 Flink 中,迭代算法受到垃圾收集的影响较小,因为其为了避免 Java 目标的创建和破坏而进行了内存优化。

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

3 结　语

本文对 Hadoop,Spark 和 Flink 的可扩展性方面的性能进行了评价,并且在实验中考虑到了不同的框架配置。实验结果表明: 当使用最大的集群规模时,Spark 在 WordCount 和 K 均值中取得了最优的结果,Flink 则在 PageRank 中表现较佳。Spark 和 Flink 在 Grep、TeraSort 和连接组件上则得到了相似的结果,Spark 和 Flink 各有优势。此外,Spark 纳入了更丰富的操作集合和更多的工具。而 Flink 包含了一些有趣和新颖的设计理念,部分设计理念也得到 Spark 的采用。

未来,本文计划进一步研究更多的配置参数对这些框架的性能所造成的影响(例如溢出阈值、网络缓冲等)。还打算开展一个类似的评价,但主要着眼于利用 Spark、Flink 和其他流处理框架,对工作负载进行流式处理。

参考文献:

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

[2] 徐计,王国胤,于洪. 基于粒计算的大数据处理 [J]. 计算机学报,2015,38(08):1497-1517.

[3] White T, Cutting D. Hadoop: the definitive guide [J]. O'reilly Media Inc Gravenstein Highway North, 2009, 215, 41(11):1-4.

[4] 孙竞, 余宏亮, 郑纬民. 支持分布式存储删冗的相似文件元数据集合索引 [J]. 计算机研究与发展, 2013, 50(1): 197-205.

[5] Veiga J, Expósito R R, Taboada G L, et al. Analysis and evaluation of MapReduce solutions on an HPC cluster [J]. Computers & Electrical Engineering, 2016, 50(C): 200-216.

[6] 宋杰,孙宗哲,毛克明,鲍玉斌,于戈. MapReduce 大数据处理平台与算法研究进展 [J]. 软件学报,2017,28(03):514-543.

[7] 王李进,尹义龙,钟一文. 逐维改进的布谷鸟搜索算法 [J]. 软件学报,2013,24(11): 2687-2698.

[8] Zaharia M, Chowdhury M, Franklin M J, et al. Spark: cluster computing with working sets [C] // Usenix Conference on Hot Topics in Cloud Computing. USENIX Association, 2010: 10-17.

[9] Alexandrov A, Bergmann R, Ewen S, et al. The Stratosphere platform for big data analytics [J]. Vldb Journal, 2014, 23(6): 939-964.

[10] 冯兴杰,王文超. Hadoop 与 Spark 应用场景研究 [J]. 计算机应用研究,2018,(09):1-8.

[11] Karasti H, Baker K S, Millerand F. Infrastructure Time: Long-term Matters in Collaborative Development [J]. Computer Supported Cooperative Work, 2010, 19(3-4): 377-415.

[12] Garcíagil D, Ramírezgallego S, García S, et al. A comparison on scalability for batch big data processing on Apache Spark and Apache Flink [J]. Big Data Analytics, 2017, 2(1): 1-12.

[13] 林香,黄致建,郝艳华. 弧形燕尾型榫连接组件三维接触分析 [J]. 武汉理工大学学报(信息与管理工程版),2010,32(3): 427-429.

[14] Kang U, Tsourakakis C E, Faloutsos C. PEGASUS: A Peta-Scale Graph Mining System Implementation and Observations [C] // Ninth IEEE International Conference on Data Mining. IEEE Computer Society, 2009: 229-238.

[15] Junghanns M, Teichmann N, Rahm E. Analyzing extended property graphs with Apache Flink [C] // ACM SIGMOD Workshop on Network Data Analytics. ACM, 2016: 301-310.

[16] 王慧贤,靳惠佳,王娇龙,江万寿. k 均值聚类引导的遥感影像多尺度分割优化方法 [J]. 测绘学报,2015,44(5): 526-532.

[17] Faghri F, Hashemi S H, Babaeizadeh M, et al. Toward Scalable Machine Learning and Data Mining: the Bioinformatics Case [J]. 2017, 35(7): 1098-1106.

作者简介

代明竹(1991—),女,北京人,硕士研究生,主要研究方向为管理信息系统;
E-mail: wxcgxy@163.com

高嵩峰(1972—),男,辽宁人,博士研究生,副教授,主要研究方向为管理信息系统。

(C)1994-2019 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net