

一种近似最小有效瓶颈优先的 Coflow 调度机制

李文信¹, 周晓波², 徐仁海², 齐 恒¹, 李克秋²

(1. 大连理工大学 计算机科学与技术学院, 辽宁 大连 116024; 2. 天津大学 智能与计算学部, 天津 300350)

摘 要: 针对先验知识未知场景下的 Coflow 调度问题, 提出一种近似最小有效瓶颈优先的 Coflow 调度方法。通过结合 Coflow 当前大小和宽度决定 Coflow 的调度顺序, 并区分出流大小以及短与长等特征的 Coflow, 从而加大调度优化的空间。实验结果表明, 与先验知识未知场景下的 Aalo 方法相比, 该方法可使 Coflow 的平均完成时间降低 33.2%, 相较于先验知识已知场景下的 SEBF 方法, Coflow 平均完成时间与其仅有 7.3% 的性能差距。

关键词: 数据中心; 并行计算; Coflow 调度; 流量调度; 近似最小有效瓶颈优先

开放科学(资源服务)标志码(OSID):



中文引用格式: 李文信, 周晓波, 徐仁海, 等. 一种近似最小有效瓶颈优先的 Coflow 调度机制[J]. 计算机工程, 2019, 45(10): 19-25, 32.

英文引用格式: LI Wenxin, ZHOU Xiaobo, XU Renhai, et al. An approximate smallest-effective-bottleneck-first Coflow scheduling mechanism[J]. Computer Engineering, 2019, 45(10): 19-25, 32.

An Approximate Smallest-Effective-Bottleneck-First Coflow Scheduling Mechanism

LI Wenxin¹, ZHOU Xiaobo², XU Renhai², QI Heng¹, LI Keqiu²

(1. School of Computer Science and Technology, Dalian University of Technology, Dalian, Liaoning 116024, China;

2. College of Intelligence and Computing, Tianjin University, Tianjin 300350, China)

[Abstract] Aiming at the Coflow scheduling problem in the prior knowledge unknown scene, an Approximate Smallest-Effective-Bottleneck-First (A-SEBF) Coflow scheduling method is proposed. Coflow's scheduling order is determined by combining the current size and width of Coflow, and the Coflow is characterized by large and small flows, as well as features such as fat, short and thin, so as to increase the space for scheduling optimization. Experimental results show that compared with the Aalo method in the prior knowledge unknown scene, the method can reduce the average completion time of Coflow by 33.2%. Compared with the SEBF method in the prior knowledge known scene, the average completion time of Coflow lags only 7.3% in performance.

[Key words] data center; parallel computing; Coflow scheduling; flow scheduling; Approximate Smallest-Effective-Bottleneck-First (A-SEBF)

DOI: 10.19678/j.issn.1000-3428.0054041

0 概述

随着大数据技术的发展, 在云数据中心需要处理的数据规模呈指数型增长, 使得并行计算框架(如 MapReduce^[1] 和 Spark^[2]) 成为云服务中数据处理分析的主流平台。在上述计算框架中, 一个作业包含多个计算阶段, 如 MapReduce 作业中的 map 映射阶段和 reduce 归约阶段。在一个计算阶段(如 reduce)

开始之前, 需要启动多条流以传输来自前一计算阶段(如 map)所产生的中间数据。由于这些中间数据体量大, 因此在网络中的传输对相应的作业性能有较大的影响。文献[3]研究结果表明, 中间数据流的传输时间占整个作业完成时间的比例高达 50%。因此, 优化中间数据流的传输对分布式并行计算应用的性能至关重要。

近年来, Coflow^[4] 作为一种流量模型, 被广泛地用于并行计算作业通信阶段。Coflow 是一个并行计

基金项目: 国家重点研发计划(2016YFB1000205); 国家自然科学基金重点项目(61432002)。

作者简介: 李文信(1991—), 男, 博士, 主研方向为云计算、大数据技术; 周晓波, 副教授; 徐仁海, 博士研究生; 齐 恒, 副教授; 李克秋, 教授、博士生导师。

收稿日期: 2019-03-01

修回日期: 2019-05-03

E-mail: xiaobo.zhou@tju.edu.cn

算作业中 2 个不同计算阶段间的数据流的集合。为提升分布式并行计算应用的性能,需要优化 Coflow 的完成时间 (Coflow Completion Time, CCT),而不是单个流的完成时间 (Flow Completion Time, FCT)。这主要是因为一个并行计算作业的完成时间依赖于其所有中间网络数据流的完成,而不是单个流的完成。例如,在 MapReduce^[1]中,一个计算阶段不可能在它收到前一个计算阶段的所有中间数据流之前完成。文献[4-6]研究表明,优化 Coflow 的平均完成时间能够明显提升分布式并行计算应用的性能。

现有针对 Coflow 平均完成时间的优化工作,主要通过设计合理的 Coflow 调度机制来实现,平均完成时间的优化主要分为两类。一类是先验知识已知的 Coflow 调度机制^[5-7],其中先验知识主要指 Coflow 中流的数据量。在这类工作中,Varys^[5]在平均 Coflow 完成时间方面的性能最好,它利用最小有效瓶颈优先的 Coflow 调度策略,来近似地模拟最短作业优先启发式算法,从而优先选择小 Coflow。但是 Varys 依赖于 Coflow 中流量大小信息,这种信息在许多实际场景中都是无法事先预知的。另一类工作是先验知识未知的 Coflow 调度^[8-9]。Aalo^[8]是这类工作的代表,其根据当前已经发送的数据量来决定 Coflow 的优先级。由于不需要事先知晓 Coflow 的流量信息,因此 Aalo 的实用性很高。但是其根据 Coflow 总体发送数据量决定优先级过于粗糙,且容易错失一些潜在的优化空间。

本文通过分析现有研究的不足并结合实际场景中 Coflow 的特点,提出一种近似最小有效瓶颈优先调度机制(A-SEBF)。通过联合 Coflow 当前已发送数据和 Coflow 宽度这 2 个信息决定 Coflow 的优先级,并基于 Facebook 公开的 MapReduce/Hive 数据集,与先验知识已知调度器 Varys 和先验知识未知调度器 Aalo 进行仿真实验对比。

1 相关工作

数据中心网络流量调度按照调度粒度可分为基于单个流的调度和基于 Coflow 的调度。基于单个流的工作虽然能够优化流的平均完成时间^[10-11],但它们不考虑流与流之间的协同依赖关系,无法适用于 Coflow 的调度场景中^[12-13]。在基于 Coflow 的调度研究工作中,文献[3]提出 Orchestra 方法以对并行计算应用产生的流进行调度,并于 2012 年提出 Coflow 模型^[4]来描述并行计算应用 2 个计算阶段间并发流的集合。至此,基于 Coflow 的研究开始得到广泛的关注。现有研究工作按照调度目标可分为以性能为导向的 Coflow 调度^[5-8]和以公平为导向的 Coflow 调度^[9]。这两类方法又可进一步分为先验知识已知的 Coflow 调度和先验知识未知的 Coflow 调度。

1)以性能为导向的 Coflow 调度。为提升分布式并行计算作业的性能,网络操作者应以最快的方式将尽可能多的 Coflow 传输完成。以性能为导向的 Coflow 调度研究工作主要以最小化 Coflow 的平均完成时间为目标。其中,一部分研究工作适用于先验知识已知的场景^[5-7,14-16],另外一部分则适用于先验知识不可知的场景^[8]。在先验知识已知的场景中^[17-18],Varys^[5]采用最小有效瓶颈优先的启发式策略来调度 Coflow。此外,Baraat^[6]方法利用先进先出和公平共享的调度策略来减少 Coflow 的平均完成时间。在先验知识未知的场景中,为减少平均 Coflow 完成时间,Aalo^[8]提出 D-CLAS (Discretized Coflow Aware Least Attained Service) 方法。D-CLAS 按照已发送数据大小将 Coflow 划分到不同的优先级队列中,并在每个队列中以先进先出 (FIFO) 的方式对 Coflow 进行调度。然而,由于总体数据量小的 Coflow 不代表其瓶颈流数据量小,使得它容易忽略一些潜在的优化空间。

2)以公平为导向的 Coflow 调度。较多研究人员认为同等地对待所有 Coflow 对部分有着特殊需求的 Coflow 会不公平。在先验知识已知的场景中^[19-21],文献[19]针对 Coflow 延迟敏感度进行效益函数建模,并建立一个字典序最大化问题以优化 Coflow 的效益总和,但是他们的方法不能提供性能隔离。为此,文献[20]提出 Coflex 以兼顾 Coflow 平均完成时间和 Coflow 间的性能隔离。此外,HUG^[22]也能实现这种性能隔离保障。但是当 Coflow 流量信息未知时,这些方法将不再适用。为此,wenian^[9]提出 NC-DRF 调度器,以在先验知识不可知场景下实现 Coflow 间的隔离保障。以上方法虽然能提供公平或性能隔离保障,但它们在减少 Coflow 平均完成时间方面的性能欠佳。

总体来看,现有研究在最小化 Coflow 平均完成时间的性能方面欠佳,尤其是在 Coflow 流量信息无法预知的场景下,还有较大的提升空间。

2 研究背景

本节介绍 Coflow 及数据中心网络抽象模型,并结合实例来详细论述本文的研究动机。

2.1 Coflow 抽象模型

一个 Coflow 是指一个分布式并行计算作业的 2 个不同计算阶段间数据流的集合^[4]。Coflow 中的流仅包含中间传输的数据流,而不涉及通信过程中产生的控制流。Coflow 抽象模型允许并行计算应用明确地表述其网络需求,例如最小化 Coflow 完成时间或确保 Coflow 内所有流在截止时间前完成。在数据中心中,许多并行计算作业会同时运行。因此,大量 Coflow 会共存于数据中心,并相互争用网络资源。Coflow 调度机制的目标是最小化 Coflow 的平

均完成时间,以提高应用层作业的性能。

由于 Coflow 包含多个流,因此不能仅用数据大小来刻画它。Coflow 包含很多特征,如长度、宽度、大小等^[5]。长度指 Coflow 中数据量最大的流的大小;宽度指 Coflow 中并发流的数目;大小指 Coflow 中所有流大小的总和。在很多实际场景中,Coflow 的大小不能预先知晓,这主要是由流水线作业^[23]、多波任务执行^[24]以及任务失败^[1,25]等原因所导致。例如,在流水线多阶段作业中(如 MapReduce Online^[23]),数据一经产生就直接被传送到目的地,且仅当所有数据传输完毕,才能知道流的大小。为此,本文重点关注 Coflow 大小不可知的场景。

2.2 数据中心网络抽象模型

本文考虑将整个数据中心网络抽象为一个无阻塞的连接所有服务器的大交换机^[26],并且只关注其出入端口。这种网络抽象模型在全平分带宽拓扑^[27]下是合理的,并且大量地被现有研究工作所采纳^[5-9,19-21]。在这种大交换机模型中,每个入端口对应所连接服务器的外向链路,而每个出端口对应所连接服务器的入链路。

如图 1 所示,一个包含 3 台服务器的数据中心网络被抽象为一个无阻塞的大交换机。在图 1 中,有 3 个 Coflow,分别用灰色(C1)、黑色(C2)和白色(C3)表示。C1 有 2 条流,数据大小均为 2。C2 和 C3 均只有 1 条流,数据大小分别为 4 和 3。需要注意的是,图 1 中入端口的虚拟队列是用来表述流的源端点和目的端点的。例如,在服务器 1 中,C1 的其中一条流需要传送 2 个单位的数据给服务器 2,C2 的流需要传输 4 个单位的数据到服务器 2;在服务器 2 中,C1 和 C3 分别需要传输 2 个单位和 3 个单位的数据给服务器 3。

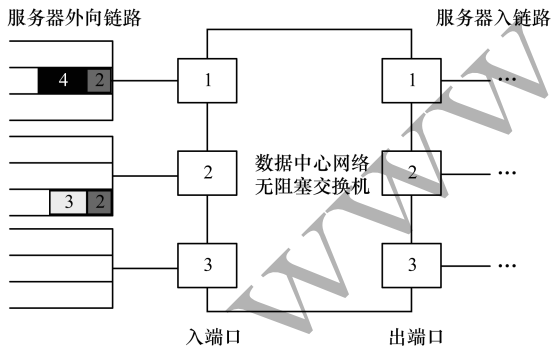


图 1 数据中心网络抽象模型示意图

2.3 研究动机

在先验知识未知的场景下,当前最先进的调度方法是 Aalo^[8]。Aalo 采用一个集中控制器来统计每个 Coflow 的进度。进度是指截止到目前,一个 Coflow 在所有端口已经发送的数据总和。具体到每个端口中,Aalo 实现了一个本地调度器,并结合多级

队列机制以近似模拟最小 Coflow 优先的调度策略。Aalo 会根据 Coflow 的进度将其放置在不同的优先级队列中。在队列间采用加权公平队列调度方法,而在队列内部则使用先进先出的机制来决定流的发送顺序。从本质上来讲,Aalo 对最小 Coflow 优先做了粗略的模仿,但与此同时,忽略了 Coflow 宽度信息,使得它容易错失一些优化机会。

为了更好地说明 Aalo 的不足,本文针对图 1 中的 Coflow 来分析不同调度机制得到的不同调度结果。假设图 1 中 Coflow 的数据大小信息能够预知,那么最优的调度就是采用 Varys 的最小有效瓶颈优先方法。如图 2 所示,在最优调度机制下,C1 首先被调度,其次是 C3 和 C2。假设每个端口单位时间内只能传输 1 个单位的数据,那么在最优调度下,Coflow 的平均完成时间为 $(2 + 5 + 6) / 3 = 4.33$ 。

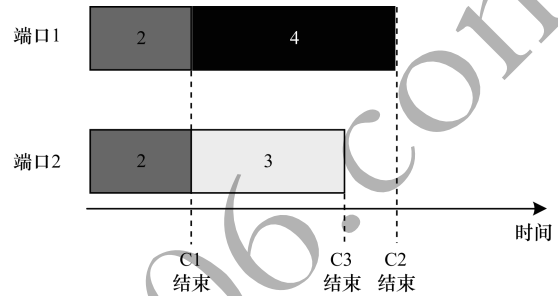


图 2 最优调度结果

图 3 给出了在 Coflow 大小信息不能事先知晓的情况下采用 Aalo 调度器所得到的调度结果。假设在 Aalo 调度器中,队列的阈值为 1,2,4,8,...,那么 Aalo 所能实现的 Coflow 平均完成时间就是 $(4.5 + 5 + 6) / 3 = 5.17$ 。在这个例子中,Aalo 具体工作流程如下:一旦 Coflow 的流到达,它们进入最高优先级队列,然后随着 Coflow 发送数据总量的增加,逐步下移到低优先级队列。优先级移动的基本原则是一旦 Coflow 发送数据总量超过当前队列的预先设定的阈值,那么该 Coflow 就会被移动到下一个低优先级队列中。

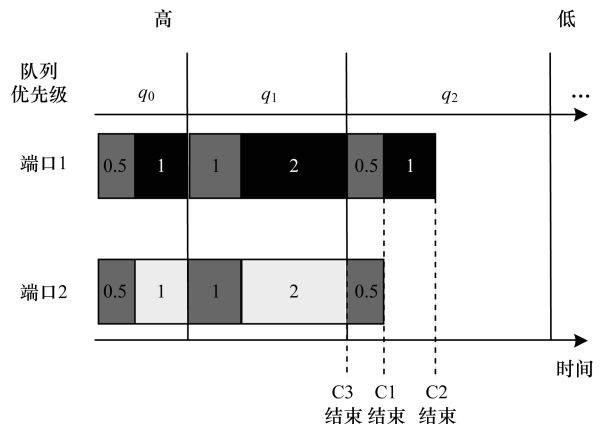


图 3 Aalo 调度结果

在 Aalo 的设计中,队列的阈值是基于已经发送的总体数据量来制定,并且单个队列的阈值对所有的 Coflow 都一样。当一个宽度较大的 Coflow 得到调度时,它能迅速地到达队列阈值,因为相比于瘦的 Coflow,胖的 Coflow 能够在更多的端口上传输数据。然而,一旦它到达阈值并移动到低优先级队列后,则需要等待高优先级的所有其他 Coflow。在这种情况下,原本应该优先被调度的胖而短的 Coflow 却需要等待瘦长的 Coflow。这也是为何 Aalo 所实现的平均 Coflow 完成时间与 Varys 所实现的最优值相差甚远。

如果在调度时能够将宽度考虑进来,那么 Coflow 的平均完成时间将能够得到进一步的优化。如图 4 所示,每个队列的阈值依然为 1,2,4,8,...。但是 Coflow 优先级队列移动的原则不再是基于已经发送的总体数据量,而是基于平均每条流已经发送的数据量。例如,当 C1 在优先级队列 q_0 中平均每条流发送的数据达到 1 时,C1 就被移动到低优先级队列 q_2 中。可以看出,在这种调度机制下,Coflow 的平均完成时间是 $(3+5+6)/3=4.67$ 。

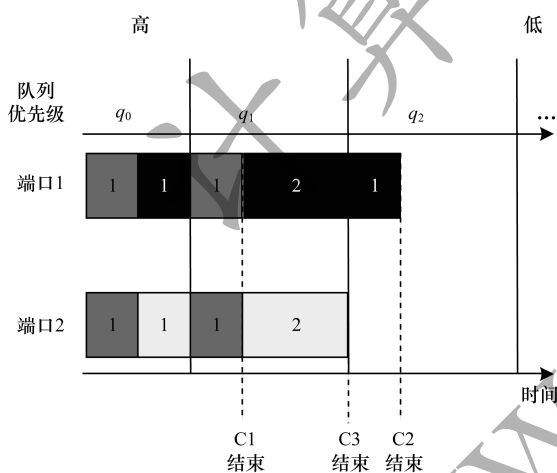


图 4 本文调度机制实现结果

实际上,不同 Coflow 的宽度存在着很大的差异。图 5 和图 6 分别展示了 Coflow 宽度与 Coflow 数据量大小的关系,其中的数据是来自于 Facebook 数据中心的 MapReduce/Hive 数据集^[5,7-8]。可以看出,无论是发送端口还是接收端口,都存在端口数目少但数据量大的 Coflow。若在调度中,这些 Coflow 很有可能被认为是小 Coflow 并因此具有高优先级,这显然与最优的 Coflow 调度思想相违背。为此,本文后续部分将通过设计近似最小有效瓶颈优先调度机制来解决这类问题。

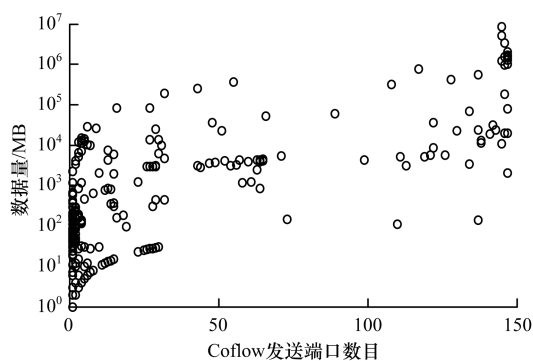


图 5 Coflow 发送端口数目与 Coflow 数据量的关系

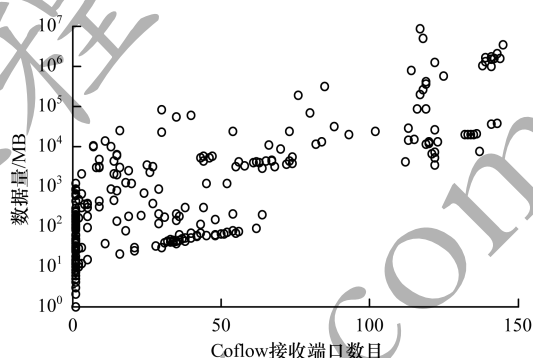


图 6 Coflow 接收端口数目与 Coflow 数据量的关系

3 A-SEBF 设计

3.1 整体工作流程

本节首先介绍 A-SEBF 的整体工作流程。A-SEBF 调度机制如图 7 所示。

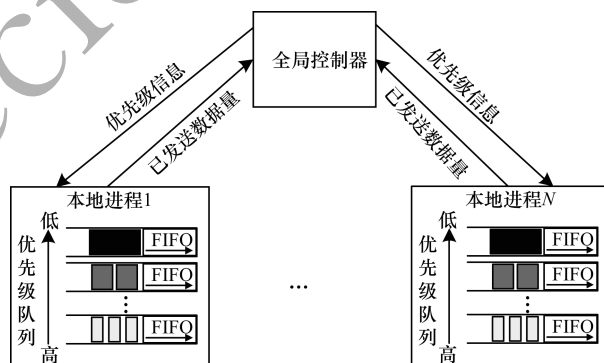


图 7 A-SEBF 调度机制

A-SEBF 流程与 Aalo^[8] 类似,且涉及 2 个部分:全局控制器和本地进程。其中每台服务器主机运行一个本地进程。全局控制器定期地收集每台服务器主机中所有流截止到当前已经发送的数据量。全局控制器根据收集的结果,得到所有 Coflow 当前已经成功发送的数据量大小,并结合 Coflow 的宽度信息计算瓶颈因子。然后通过比较瓶颈因子与不同优先级队列的阈值来决定每个 Coflow 应该放在哪个优

先队列。每台服务器主机根据接收到的优先级信息,将 Coflow 移动到对应的优先级队列。同一优先级队列采用先进先出调度,不同优先级队列间采用按权公平队列调度策略。与 Aalo 类似,本文使用阈值以 10 为底的指数型增长的优先级队列,也即第 $h+1$ 个队列的阈值是第 h 个的 10 倍。

表 1 给出本文内容所涉及的变量和定义说明。

表 1 算法参数定义

参数	定义
i, j	主机索引
k	Coflow 索引
b_i^{out}	主机 i 的出口带宽
b_j^{in}	主机 j 的入口带宽
d_{ij}^k	Coflow k 需要从主机 i 传输给主机 j 的数据量
T_k	独占网络带宽时, Coflow k 的完成时间
S_k	Coflow k 当前已经发送的数据量
W_k	Coflow k 的宽度, 即其内部并发流的数目
λ_k	Coflow k 的瓶颈因子

3.2 A-SEBF 调度机制

本节首先介绍最小有效瓶颈优先 (Smallest-Effective-Bottleneck-First, SEBF) 调度机制。SEBF 是 Varys 针对 Coflow 设计的调度机制, 并且是公认的在先验知识已知场景中的最优调度机制。SEBF 的核心思想是找出每个 Coflow 的瓶颈所在, 并据此决定 Coflow 的优先级。具体来讲, 假设有一个 Coflow k , 其需要从主机 i 传输 d_{ij}^k 的数据量到主机 j 。假设主机 i 的出口带宽为 b_i^{out} , 主机 j 的入口带宽为 b_j^{in} , 那么该 Coflow 的完成时间至少为 T_k :

$$T_k = \max \left(\max_i \frac{\sum_j d_{ij}^k}{b_i^{\text{out}}}, \max_j \frac{\sum_i d_{ij}^k}{b_j^{\text{in}}} \right)$$

在 Varys 的 SEBF 方法中, T_k 被认为是 Coflow k 的瓶颈因子。因为它是假设在得到所有带宽前提下, 一个 Coflow 中瓶颈流所需的完成时间。基于此, SEBF 就是让 T_k 越小的 Coflow 优先级越高。Varys 的实验结果表明, 相比于 Coflow 长度最短优先、Coflow 宽度最窄优先以及 Coflow 数据总量最小优先等调度方法, SEBF 能够明显减少 Coflow 的平均完成时间。SEBF 虽然能表现出较好的性能, 但是其依赖于每个 Coflow 的数据大小信息, 例如 d_{ij}^k 。而在很多实际场景中, 如 2.1 节所述, 这种信息是无法事先预知的。为此, 本文提出近似最小有效瓶颈优先调度机制, 称为 A-SEBF。

A-SEBF 主要基于 2 个信息对 Coflow 进行优先级排序。第 1 个信息是 Coflow 的当前大小, 也即截止到目前, 一个 Coflow 通过整个数据中心网络已经发送的数据量。由于 Coflow 大小通常呈现重尾分

布趋势^[5], 因此 Coflow 当前大小能够很好地代表 Coflow 的实际大小, 并且在实际场景中也能很容易地被获取到^[28]。第 2 个信息是 Coflow 的宽度, 宽度是指 Coflow 中并发流的个数。如 2.1 节所述, 准确的流的数目很难知晓, 但是能够知道一个 Coflow 通过多少个端口来发送或接收数据。由于在许多实际的场景中, 相比于发送端, 接收端更容易造成拥堵^[29], 因此, 本文采用接收端口的数目来表示一个 Coflow 的宽度信息。

A-SEBF 的核心就是基于 Coflow 当前大小以及 Coflow 宽度信息来评估该 Coflow 的瓶颈因子, 并根据瓶颈因子大小来决定不同 Coflow 的优先级。假设 Coflow k 的当前大小和接收端口数分别为 S_k 和 W_k , 那么该 Coflow 的瓶颈因子 λ_k 被定义为:

$$\lambda_k = \frac{S_k}{W_k}$$

在 A-SEBF 中, 瓶颈因子 λ_k 越小的 Coflow, 优先级越高。值得注意的是, A-SEBF 与 Aalo 中的 D-CLAS 调度机制不同。D-CLAS 本质上模拟的是 Coflow 大小最小优先调度机制, 即当前发送数据量 S_k 来判断 Coflow 优先级, 容易错失一些潜在的优化机会。假如, 有 2 个 Coflow k_1 和 k_2 , k_1 的宽度大于 k_2 , 但其长度小于 k_1 。在某个特定时刻 t , k_1 当前大小 S_{k_1} 大于 k_2 的当前大小 S_{k_2} , 那么根据 Aalo 的 D-CLAS 调度机制, k_2 会优于 k_1 被调度。然而, 根据最优的最小瓶颈优先 SEBF 策略, k_1 应该优先于 k_2 被调度。如果将 Coflow 的宽度信息兼顾起来, 若利用瓶颈因子 λ_k 决定 Coflow 调度顺序, 则能够在一定程度上消除这种现象。尤其是当 Coflow 当前大小差距不大, 而宽度差异明显时, 基于瓶颈因子 λ_k 的调度就越能体现出其在优化 Coflow 平均完成时间的优势。

4 实验评估

本节基于 Facebook 真实数据集对所提出的最小有效瓶颈优先调度机制进行性能测试, 并通过对比现有先验知识已知和先验知识未知场景下的调度机制来验证本文提出的调度机制的有效性。

4.1 实验设置

实验是在单机模拟环境下进行的, 具体包括一个 Dual-Core Intel(R) Core(TM) i7-5500U 2.40 GHz CPU 和 8 GB 内存。此外, 采用 Java 语言在 CoflowSim^[30] 开源框架的基础上实现了本文所提出的 A-SEBF 调度算法。CoflowSim 是一个专用于 Coflow 调度的模拟器, 目前在 Coflow 的相关研究^[5,8,17] 中得到了广泛应用。

本文采用由 Facebook 提供的 MapReduce/Hive 数据集^[5,7-8], 该数据集是从一个包含 3 000 台服务

器和 150 个机架的集群中收集起来的,该数据集一共包含 526 个在不同时间到达的 Coflow。数据集中每个记录行包括以下信息:Coflow ID,到达时间,mapper 数量,mapper-m 的位置,Reducer 的数量和 reducer-r 的位置,以及该 reducer 需要读取的数据量。需要注意的是,原始数据被压缩到了机架级别,因此,mapper 和 reducer 位置的取值范围是 0~149。

为不对数据集做任何修改,本文模拟一个包含 150 台服务器主机的数据中心网络,其中每台服务器的出口和入口带宽容量均设置为 1 Gb/s。

本文主要对比以下 2 个方法:

1) Aalo。该方法在 Coflow 信息不知晓的前提下,利用 D-CLAS 策略来调度 Coflow。

2) Varys。该方法在 Coflow 信息知晓的场景中,利用 SEBF 策略对 Coflow 做最优的调度。

本文通过定义 $\frac{AugCCT_2 - AugCCT_1}{AugCCT_2}$ 性能指标,来

表示方法 1 在方法 2 的基础上 Coflow 平均完成时间降低的百分比,其中, $AugCCT_1$ 和 $AugCCT_2$ 分别是方法 1 和方法 2 所能实现的 Coflow 的平均完成时间。

4.2 结果分析

图 8 给出了在不同调度方法下所有 Coflow 的平均完成时间,其中,横轴为不同的调度方法,纵轴为完成时间度量。从图 8 中可以看到,A-SEBF 实现的平均完成时间明显少于 Aalo,且与 Varys 十分相近。更具体地讲,Aalo、A-SEBF 和 Varys 这 3 种方法实现的平均 Coflow 完成时间分别是 46 098 ms、30 800 ms 以及 28 528 ms。相比于 Aalo 方法,A-SEBF 能够将平均 Coflow 完成时间降低 $(46\ 098 - 30\ 800)/46\ 098 = 33.2\%$ 。造成性能提升的主要原因是 Aalo 在调度中仅考虑当前已发送数据量,没有考虑 Coflow 宽度的影响,使得它容易错过一些潜在的优化机会。相比于 Varys 方法,A-SEBF 仅有 $(30\ 800 - 28\ 528)/28\ 528 = 7.3\%$ 的差距。但是 Varys 假设先验知识已知,在实际场景中得不到很好的应用,而本文重点关注如何在先验知识未知的场景。上述结果表明,本文提出的 A-SEBF 是一种先验知识未知场景下的高效调度器,且它能够实现接近于先验知识已知场景下的最优平均 Coflow 完成时间。

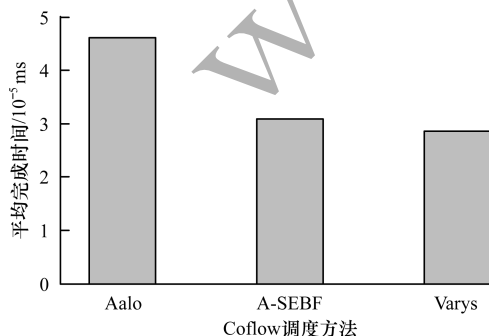


图 8 Coflow 平均完成时间

为进一步理解不同方法的 Coflow 完成时间,图 9 给出了不同调度方法实现的 Coflow 完成时间的累积分布函数曲线。其中,横轴表示完成时间并以对数刻度呈现,纵轴表示累积分布函数(Cumulative Distribution Function, CDF)值。图 9 能够表示出完成时间小于任何一个数值的 Coflow 数目占 Coflow 总数的比例,可以看出,在 3×10^5 ms 之前,Aalo 与 A-SEBF 这 2 条曲线走势差别不大或基本吻合;但在 3×10^5 ms 后,A-SEBF 曲线明显高于 Aalo,部分时候甚至高于 Varys 曲线。具体来看,在 Aalo 方法下,97.91% 的 Coflow 的完成时间小于 10^6 ms;而在 A-SEBF 方法下,有 99.24% 的 Coflow 的完成时间小于 10^6 ms。结合 Coflow 大小的重尾分布特性,不难得出 A-SEBF 在平均 Coflow 完成时间方面优于 Aalo。

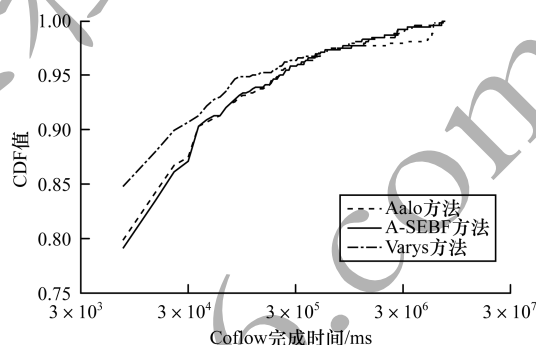


图 9 Coflow 完成时间的累积分布函数曲线

由于 Coflow 的完成时间实际上取决于其瓶颈流的完成时间,不同的方法只是在不同场景下利用不同的策略来找出这种瓶颈流。为说明瓶颈流的完成快慢情况,图 10 给出了不同方法下 Coflow 中瓶颈流量大小与 Coflow 完成时间的散点图。其中,横轴是瓶颈流的大小,纵轴是 Coflow 的完成时间,横纵轴都以对数刻度形式呈现。需要说明的是,因为本文在对 A-SEBF 和 Aalo 方法进行测验时是假设 Coflow 大小信息不可知的,图 10 的结果不违背本文的先验知识不可知的场景。在测验结束后,会记录每个 Coflow 的完成时间、总体数据量大小以及最长流的大小等信息。从以上结果可以看出,A-SEBF 能够将 Coflow 中瓶颈流的完成时间置于 Aalo 和 Varys 之间。这证明了本文所提的 A-SEBF 调度机制具有较好的性能。

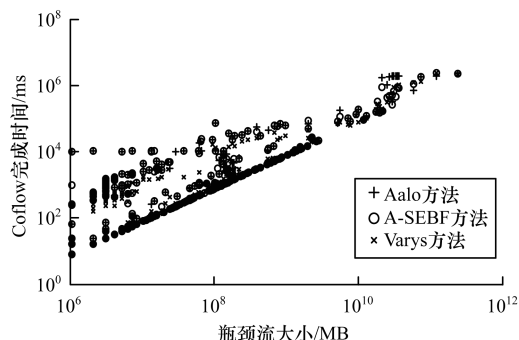


图 10 Coflow 中瓶颈流大小与 Coflow 完成时间的关系

5 结束语

本文在先验知识未知的场景下,提出一种近似最小有效瓶颈优先的 Coflow 调度机制 A-SEBF。A-SEBF 在决定各个 Coflow 优先级时兼顾 Coflow 大小和宽度 2 个因素,从而加大 Coflow 调度优化的空间。通过 Facebook 公开的 Coflow 数据集,对 A-SEBF 调度机制进行仿真分析。实验结果表明,相较于当前先进的先验知识未知的 Aalo 调度器, A-SEBF 调度机制在“最小化平均 Coflow 完成时间”方面有 33.2% 的性能提升;相较于先验知识已知的 Varys 调度器,仅有 7.3% 的性能损失。下一步将在真实集群环境下对本文提出的 A-SEBF 调度机制进行分布式优化与实现。

参考文献

- [1] 高军,黄献策.基于 Hadoop 平台的相关性权重算法设计与实现[J].计算机工程,2019,45(3):26-31.
- [2] 戚荣志,王志坚,黄宜华,等.基于 Spark 的并行化组合测试用例集生成方法[J].计算机学报,2018,41(6):1064-1079.
- [3] CHOWDHURY M, ZAHARIA M, MA J, et al. Managing data transfers in computer clusters with orchestra [C]//Proceedings of ACM Special Interest Group on Data Communication. New York, USA: ACM Press, 2011:98-109.
- [4] CHOWDHURY M, STOICA I. Coflow: a networking abstraction for cluster applications [C]//Proceedings of ACM Workshop on Hot Topics in Networks. New York, USA: ACM Press, 2012:31-36.
- [5] CHOWDHURY M, ZHONG Yuan, STOICA I. Efficient Coflow scheduling with varys [C]//Proceedings of ACM Special Interest Group on Data Communication. New York, USA: ACM Press, 2014:443-454.
- [6] DOGAR F R, KARAGIANNIS T, BALLANI H, et al. Decentralized task-aware scheduling for data center networks [C]//Proceedings of ACM Special Interest Group on Data Communication. New York, USA: ACM Press, 2014:431-442.
- [7] AGARWAL S, RAJAKRISHNAN S, NARAYAN A, et al. Sincronia: near-optimal network design for Cflows [C]//Proceedings of ACM Special Interest Group on Data Communication. New York, USA: ACM Press, 2018:16-29.
- [8] CHOWDHURY M, ZHONG Yuan, STOICA I. Efficient Coflow scheduling without prior knowledge [C]//Proceedings of ACM Special Interest Group on Data Communication. New York, USA: ACM Press, 2015:393-406.
- [9] WANG Luping, WANG Wei. Fair Coflow scheduling without prior knowledge [C]//Proceedings of IEEE International Conference on Distributed Computing Systems. Piscataway, USA: IEEE Press, 2018:22-32.
- [10] 董谦,李俊,马宇翔,等.软件定义网络中基于分段路由的流量调度方法[J].通信学报,2018,39(11):23-35.
- [11] 谭群芳,鲁晓霞.基于 SDN 的云数据中心网络解决方案[J].电信工程技术与标准化,2017(1):25-28.
- [12] ALIZADEH M, YANG Shuang, SHARIF M, et al. Pfabric: minimal near-optimal data center transport [C]//Proceedings of ACM Special Interest Group on Data Communication. New York, USA: ACM Press, 2013:435-446.
- [13] BAI Wei, CHEN Li, CHEN Kai, et al. Information-agnostic flow scheduling for commodity data centers [C]//Proceedings of USENIX Conference on Networked Systems Design and Implementation. Berkeley, USA: USENIX, 2015:455-468.
- [14] QIU Zhen, STEIN C, ZHONG Yuan. Minimizing the total weighted completion time of Coflows in datacenter networks [C]//Proceedings of ACM Symposium on Parallelism in Algorithms and Architectures. New York, USA: ACM Press, 2015:294-303.
- [15] ZHAO Yangming, CHEN Kai, BAI Wei, et al. RAPIER: integrating routing and scheduling for coflow-aware data center networks [C]//Proceedings of IEEE Conference on Computer Communications. Piscataway, USA: IEEE Press, 2015:424-432.
- [16] LI Yupeng, JIANG Shaofeng, TAN Haisheng, et al. Efficient online Coflow routing and scheduling [C]//Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing. New York, USA: ACM Press, 2016:161-170.
- [17] LI Weixin, YUAN Xu, LI Keqiu, et al. Leveraging endpoint flexibility when scheduling Coflows across geo-distributed datacenters [C]//Proceedings of IEEE Conference on Computer Communications. Piscataway, USA: IEEE Press, 2018:873-881.
- [18] SUSANTO H, JIN Hao, CHEN Kai. Stream: decentralized opportunistic inter-coflow scheduling for datacenter networks [C]//Proceedings of IEEE International Conference on Network Protocols. Piscataway, USA: IEEE Press, 2016:1-10.
- [19] CHEN Li, CUI Wei, LI Baochun, et al. Optimizing Coflow completion times with utility max-min fairness [C]//Proceedings of IEEE Conference on Computer Communications. Piscataway, USA: IEEE Press, 2016:1-9.
- [20] WANG Wei, MA Shiyao, LI Bo, et al. Coflex: navigating the fairness-efficiency tradeoff for Coflow scheduling [C]//Proceedings of IEEE Conference on Computer Communications. Piscataway, USA: IEEE Press, 2017:1-9.
- [21] WANG Luping, WANG Wei, LI Bo. Utopia: near-optimal Coflow scheduling with isolation guarantee [C]//Proceedings of IEEE Conference on Computer Communications. Piscataway, USA: IEEE Press, 2018:891-899.

(上接第 25 页)

- [22] CHOWDHURY M, LIU Zhenhua, GHODSI A, et al. HUG: multi-resource fairness for correlated and elastic demands [C] // Proceedings of USENIX Conference on Networked Systems Design and Implementation. Berkeley, USA; USENIX, 2016: 407-424.
- [23] CONDIE T, CONWAY N, ALVARO P, et al. MapReduce online [C] // Proceedings of USENIX Conference on Networked Systems Design and Implementation. Berkeley, USA; USENIX, 2010: 313-327.
- [24] ANANTHANARAYANAN G, GHODSI A, WANG A, et al. Pacman: coordinated memory caching for parallel jobs [C] // Proceedings of USENIX Conference on Networked Systems Design and Implementation. Berkeley, USA; USENIX, 2012: 267-280.
- [25] ZAHARIA M, CHOWDHURY M, DAS T, et al. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing [C] // Proceedings of USENIX Conference on Networked Systems Design and Implementation. Berkeley, USA; USENIX, 2012: 15-28.
- [26] POPA L, KUMAR G, CHOEDHURY M, et al. FairCloud: sharing the network in cloud computing [C] // Proceedings of ACM Special Interest Group on Data Communication. New York, USA; ACM Press, 2012: 187-198.
- [27] GREENBERG A, HAMILTON J R, JAIN N, et al. VL2: a scalable and flexible data center network [C] // Proceedings of ACM Special Interest Group on Data Communication. New York, USA; ACM Press, 2009: 51-62.
- [28] NAIR J, WIERMAN A, ZWART B. The fundamentals of heavy-tails: properties, emergence, and identification [C] // Proceedings of ACM International Conference on Measurement and Modeling of Computer Systems. New York, USA; ACM Press, 2013: 387-388.
- [29] MONTAZERI B, LI Yilong, ALIZADEH M, et al. Homa: a receiver-driven low-latency transport protocol using network priorities [C] // Proceedings of ACM Special Interest Group on Data Communication. New York, USA; ACM Press, 2018: 221-235.
- [30] Flow-level simulator for Coflow scheduling used in Varys and Aalo [EB/OL]. [2019-02-11]. <https://github.com/coflow/coflowsim>.