

## 基于瓶颈感知的多级反馈队列 Coflow 调度机制

都繁杰<sup>1</sup>, 李 静<sup>1</sup>, 郭志勇<sup>2</sup>, 任颖文<sup>2</sup>, 尹晓宇<sup>3</sup>, 董小菱<sup>3</sup>

(1.南京航空航天大学 计算机科学与技术学院, 南京 211106; 2.国家电网有限公司信息通信分公司, 北京 100761;

3.国网安徽省电力有限公司信息通信分公司, 合肥 231299)

**摘要:** Coflow 作为并行计算框架的典型流量模型, 降低 Coflow 的完成时间(CCT)成为云计算领域的研究热点。现有 Coflow 调度机制未考虑云数据中心内网络瓶颈问题, 容易造成网络拥塞, 导致 CCT 增加。针对该问题, 构建基于瓶颈感知的 Coflow 调度机制 Bamq。利用 Lagrange 对偶优化 Coflow 调度模型, 以加快 Coflow 流速并增大吞吐量, 从而降低 CCT。通过设计多级反馈队列机制, 降低吞吐量对网络拥塞产生的影响, 根据已发流的大小、宽度和流速信息, 构建瓶颈因子以动态调整多级队列的优先级, 实现拥塞感知, 提高 Coflow 调度性能。在 Facebook 真实数据集上进行实验, 结果表明, 相比 Baraat、Varys、Aalo 机制, 该机制的 CCT 平均缩短 21.3%, 吞吐量平均提高 17.9%, 能够有效提高链路的利用率。

**关键词:** Coflow 调度; 多级反馈队列; 队列稳定性; 流量调度; 云数据中心

开放科学(资源服务)标志码(OSID):



中文引用格式: 都繁杰, 李静, 郭志勇, 等. 基于瓶颈感知的多级反馈队列 Coflow 调度机制[J]. 计算机工程, 2022, 48(10): 193-201, 211.

英文引用格式: DU F J, LI J, GUO Z Y, et al. Coflow scheduling mechanism for multi-level feedback queue based on bottleneck perception[J]. Computer Engineering, 2022, 48(10): 193-201, 211.

## Coflow Scheduling Mechanism for Multi-Level Feedback Queue Based on Bottleneck Perception

DU Fanjie<sup>1</sup>, LI Jing<sup>1</sup>, GUO Zhiyong<sup>2</sup>, REN Yingwen<sup>2</sup>, YIN Xiaoyu<sup>3</sup>, DONG Xiaoling<sup>3</sup>

(1.College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China;

2.Information and Communication Branch of State Grid Corporation, Beijing 100761, China;

3.Information and Communication Branch of State Grid Anhui Electric Power Co., Ltd., Hefei 231299, China)

**[Abstract]** Coflow is a typical traffic model based on a parallel computing framework. Reducing the Coflow Completion Time (CCT) has become a popular research topic in cloud computing. The existing Coflow scheduling mechanism does not consider the network bottleneck in the cloud data center, which easily causes network congestion and increases the CCT. Hence, Coflow scheduling mechanism, Bamq, based on bottleneck perception is constructed. The Lagrange duality is used to optimize the Coflow scheduling model such that the Coflow flow rate and throughput are increased, whereas the CCT is reduced. The multi-level feedback queue mechanism reduces the effect of throughput on network congestion. Based on the size, width, and flow rate of the sent flow, the bottleneck factor is constructed to dynamically adjust the priority of multi-level queue, realize congestion perception, and enhance the performance of Coflow scheduling. Experiments on the Facebook dataset show that compared with the Baraat, Varys and Aalo mechanisms, the CCT of this mechanism is reduced by 21.3%, whereas the throughput is increased by 17.9% on average. The proposed mechanism can significantly reduce the CCT and effectively improve link utilization.

**[Key words]** Coflow scheduling; multi-level feedback queue; queue stability; flow scheduling; cloud data center

DOI: 10.19678/j.issn.1000-3428.0062472

基金项目: 国家电网有限公司科技项目“业务应用改造上云与全链路运行分析技术研究”(SGAHXTOOXYQT2100008)。

作者简介: 都繁杰(1997—), 男, 硕士研究生, 主研方向为云计算、流量调度; 李 静(通信作者), 副教授、博士; 郭志勇, 工程师; 任颖文、尹晓宇、董小菱, 助理工程师。

收稿日期: 2021-08-25

修回日期: 2021-10-18

E-mail: duwuxi@nuaa.edu.cn

## 0 概述

随着云计算、大数据的高速发展,云计算数据中心内部的通信量呈爆炸性增长<sup>[1]</sup>,逐步代替传统的数据中心。海量通信数据的处理需要大量计算节点协同完成,数据并行计算框架(如 Apache Spark<sup>[2]</sup>、Dryad<sup>[3]</sup>、MapReduce<sup>[4]</sup>、CIEL<sup>[5]</sup>等)广泛应用于云计算数据中心。在数据并行计算框架中,用户上传计算任务,云数据中心通过一系列计算和通信阶段得到计算结果,在收到所有计算结果后才能展示给用户。由于通信阶段可能占任务完成时间的50%以上,因此将显著影响应用程序的性能。然而,传统的流级优化难以满足应用程序低延迟、高吞吐量的要求,因此在应用程序优化方面不够有效。在这种情况下,Coflow应运而生,弥补了应用程序与框架之间的差距。

Coflow被称为具有语义相关的一组通信数据流,可以捕获并行应用程序中的通信信息<sup>[6]</sup>。Coflow能够高效地对网络资源使用的应用程序级语义进行建模。在建模过程中将Coflow作为网络资源分配或调度的基本元素,以实现优化目标的目的。Coflow完成时间(Coflow Completion Time, CCT)缩短的问题通常被称为Coflow调度问题。因此,Coflow调度已被广泛应用于数据中心传输设计中。为降低Coflow的平均CCT,Coflow调度方法应运而生,如 Varys<sup>[7]</sup>、Aalo<sup>[8]</sup>、Baraat<sup>[9]</sup>等,它们在一定程度上缩短平均CCT。然而大多数Coflow调度机制未考虑网络瓶颈。云计算数据中心内部存在网络链接故障、路由不均衡问题,容易造成链路拥塞,产生流量传输瓶颈。因此,忽略网络中的瓶颈会导致Coflow优先级判断错误及核心链路上不必要的带宽争用,极大影响CCT的性能。

本文根据实际场景中Coflow的特点,提出基于瓶颈感知的多级反馈队列Coflow调度机制。将瓶颈感知机制引入到数学建模过程中,分析Coflow调度的典型场景,描述调度过程的实体信息,根据链路状态与Coflow大小、宽度等信息,确定流的最大速率。同时基于链路状态与Coflow已发数据、流速、宽度等信息确定Coflow的优先级,缩短Coflow的平均完成时间。

## 1 相关工作

为提高云数据中心应用的性能,云数据中心需要优化流量调度。传统的优化对象是面向单个数据流,其优化指标为单个数据通信流的完成时间(Flow Completion Time, FCT),主要的研究工作包括 pHost<sup>[10]</sup>、MJS<sup>[11]</sup>等。该类方法通常是对单一数据流进行调度,因此无法实现整个网络资源最优调度的目的。Coflow是具有相关语义和并发性能目标的并发流集合,其优化CCT能够真正优化云计算数据中心的通信。

研究人员对Coflow调度进行研究。文献[7]提出启发式调度算法,以最小化平均CCT达到Coflow截止时间。文献[8]使用Coflow-Aware最少服务(D-CLAS)算法,根据已经发送的数据量,将Coflow划分为各种优先级。与上述集中式工作不同,文献[9]提出一种用于任务感知调度的分布式算法,将任务意识引入到网络优化中。CODA<sup>[12]</sup>是借助机器学习技术检测单个流中的Coflow。MPLBF<sup>[13]</sup>是通过贪婪策略设计一种高效的在线调度方法,提高在线调度的性能。DRGC<sup>[14]</sup>调度算法分析了多资源环境中Coflow调度问题,并对调度序列优先级进行排序。IAOA调度算法<sup>[15]</sup>根据作业的权重和瞬时网络状况动态调度Coflow。MCS调度算法<sup>[16]</sup>联合利用多个Coflow级别属性进一步缩短Coflow完成时间。NC-DRF调度算法<sup>[17]</sup>为不同租户提供对资源的隔离访问并保证调度性能,证明最小化CCT的单一Coflow路由和调度问题为NP-hard<sup>[7]</sup>。CoRBA算法<sup>[18]</sup>综合考虑路由和带宽分配,以确定路由的最优带宽分配。MCRS算法<sup>[19]</sup>基于叶脊拓扑结构设计多跳Coflow路由和调度策略,以最小化CCT。OMCoflow算法<sup>[20]</sup>同时考虑路由和流量调度,具有理论性能的在线算法。以上调度算法将网络抽象为无阻塞大型交换机,而忽略了网络资源受限的问题。Fai调度算法<sup>[21]</sup>根据流速和字节发送检测瓶颈,提高带宽分配的效率;DBA调度算法<sup>[22]</sup>是通过改进最小剩余时间优先的启发式方法,有效识别网络内瓶颈。在多个Coflow的情况下,Coflow之间和Coflow内部的路径发生重叠现象,并且流将争夺相同的链接资源,产生链路间瓶颈。

云数据中心存在网络内链路故障、路由不平衡等问题,核心链路上可能发生拥塞,流量的瓶颈将转移到网络内部。网络内瓶颈决定实际可使用多少带宽流,直接影响CCT。针对网络中的瓶颈问题,本文从瓶颈感知的角度入手,分析云数据中心架构下Coflow调度典型场景,提出基于瓶颈感知的多级反馈队列Coflow调度机制,并通过模拟实验从Coflow的平均完成时间和吞吐量两个角度对调度模型进行有效评估。

## 2 多级反馈队列Coflow调度机制

本文首先对Coflow调度进行分析,构建Coflow调度体系;然后对调度机制进行数学建模,以减小CCT;最后参考已有优化调度的方法,设计适用于当前场景的调度方法。本文结合瓶颈感知、增加链路利用率等需求,设计基于瓶颈感知的多级反馈队列Coflow调度机制Bamq。

### 2.1 Coflow调度分析

传统的流级优化难以满足应用程序低延迟、高吞吐量的要求,因此在应用程序优化方面不够有效。

与传统的流定义不同,Coflow由具有独立输入和输出的多个并行流组成,且具有多个属性。假设第 $i$ 个Coflow表示为四元组 $C_i = \langle S, W_k, L_e, f \rangle$ 。其中: $S$ 表示Coflow流的总大小; $W_k$ 表示Coflow流的宽度,即并行流的个数; $L_e$ 表示Coflow流的长度,即并行流中最大流的大小; $f$ 表示Coflow的并行流; $C_i = \{f_1, f_2, \dots, f_q\}$ 表示第 $i$ 个Coflow包含 $q$ 个并行子流 $\{f_1, f_2, \dots, f_q\}$ 。Coflow调度场景如图1所示。

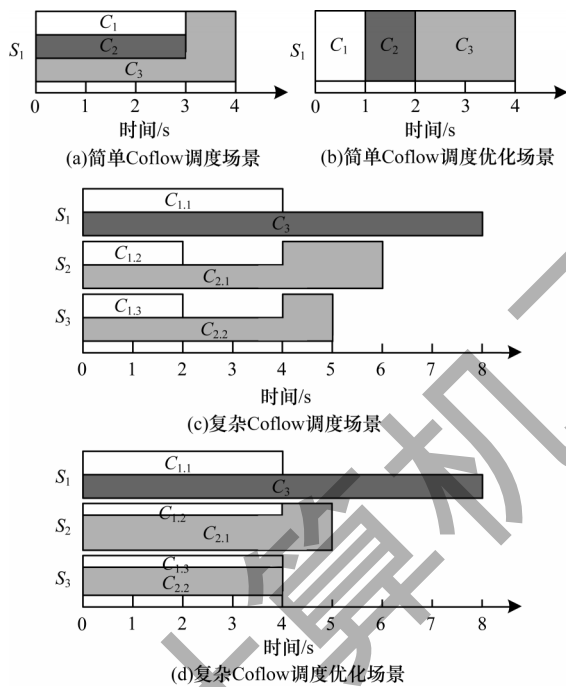


图1 Coflow调度场景

Fig.1 Coflow scheduling scenarios

云计算数据中心存在网络瓶颈问题。图1(a)有3个Coflow同时到达数据输入端口( $S_1$ ),若平均分配带宽,则会造成带宽争用,形成瓶颈。图1(a)的CCT为 $(10/3)=(3+3+4)/3$ ,图1(b)的CCT为 $(7/3)=(1+2+4)/3$ ,CCT明显降低,可以看出若及时发现瓶颈,并调整带宽分配可降低CCT。图1(c)存在3个Coflow( $C_1, C_2, C_3$ ),其中 $C_1$ 包括3个流( $C_{1.1}, C_{1.2}, C_{1.3}$ ), $C_2$ 包括2个流( $C_{2.1}, C_{2.2}$ ), $C_3$ 则为较大的流,其完成时间较长,暂不讨论。当 $C_1, C_2$ 同时到达数据输入端口( $S_1, S_2, S_3$ ),会形成链路间瓶颈,若平均分配带宽,能够有效降低CCT,图1(c)的CCT为 $(5)=(4+6)/2$ ,图1(d)的CCT $(4.5)=(4+5)/2$ 。

网络瓶颈包括网络内瓶颈与链路间瓶颈,忽略网络瓶颈会导致Coflow性能下降。为提高Coflow性能,需要增大Coflow流的速率,提高吞吐量。然而单方面增大吞吐量会造成网络拥塞,导致链路争用,产生大量链路间瓶颈,造成Coflow性能下降。因此,在增大吞吐量与避免产生瓶颈之间实现平衡成为研究热点。

针对以上问题,本文设计基于瓶颈感知的多级反馈队列Coflow调度机制Bamq,通过对云数据中

心进行建模,实现流速最大化。由于单方面增大吞吐量会形成链路争用,造成Coflow流排队,产生链路间瓶颈,因此设计基于Lyapunov优化的流量调度进行队列建模,使得数据在有限时间内离开发送队列,确保队列稳定,避免产生网络拥塞,从而提高Coflow调度性能。

## 2.2 Coflow调度过程建模

假设一个云计算数据中心有 $N$ 个主机、 $M$ 台交换机,并具有任意的拓扑结构以形成 $L$ 条链路,将流量传输过程抽象成无向图 $G(V, E)$ ,其中 $|V|=N+M, |E|=L$ 。对于无向图 $G$ ,节点 $V$ 对应云计算数据中心的一个节点,每条边 $E$ 代表一个全双工链路。

假设Coflow所有内部流同时到达端口,即使内部流异步到达,可将它们视为一起到达,并将最后一个流的到达时间视为Coflow的到达时间。

### 2.2.1 数学模型

Map/Reduce的计算阶段和CPU的计算速度都与网络的传播速度直接相关。当CPU计算速度更快时,主机接收数据后被立即处理而不用排队,因此可将网络传输时间作为reducer的完成时间。当网络传输更快时,主机接收到的数据不能被及时处理,导致数据排队、缓存,因此在网络完成传输后需要额外的计算时间。因为Coflow调度过程要关注网络调度问题,在数学分析中将忽略任务在每个reducer上的计算时间,而只考虑网络传输时间。以上问题抽象化可简化数学建模过程,在实验评估中,由于实验使用真实的网络环境,因此不需要强制遵从这些抽象问题。数学模型的参数设置如表1所示。

表1 数学模型的参数设置

Table 1 Parameter settings of mathematical model

| 参数                            | 定义                          |
|-------------------------------|-----------------------------|
| $f_k^w, k \in K, w \in W_k$   | Coflow $k$ 的第 $w$ 个流的完成时间   |
| $S_k(t), k \in K$             | 第 $k$ 个Coflow当前已发送数据量       |
| $B_l$                         | 链路 $l$ 的带宽                  |
| $\tilde{b}_l, l \in L$        | 链路 $l$ 的占用带宽                |
| $p_{kl}, p_{kl} \in \{0, 1\}$ | $p_{kl}$ 表示流 $k$ 是否占用链路 $l$ |
| $d_f(t)$                      | 时间 $t$ 内流 $f$ 的剩余数据量        |
| $r_f^l(t)$                    | 链路 $l$ 中流 $f$ 分配的带宽速率       |
| $b_f^l(t)$                    | 单位时间链路 $l$ 中流 $f$ 分配的带宽     |
| $\bar{r}_f^l(t)$              | 链路 $l$ 中流 $f$ 分配的期望带宽速率     |
| $\beta r_f^l$                 | Coflow 等待队列的瓶颈因子            |
| $C_k, k \in K$                | 第 $k$ 个Coflow的完成时间          |

在不损失通用性的情况下,假设Coflow  $C_i$  包含关于流  $C_i$  的所有信息,在时间  $T_i$  到达网络时立即开始传输,当时间  $t > T_i$  时,流  $C_i$  被分配到速率  $r_f^l(t)$  的路径上。当  $r_f^l(t)$  为零时,说明此流正等待传输。在此基础上,将调度问题表示为时间  $t$  内的优化问题,如式(1)所示:



$$\text{Minimize: } \sum_{k=1}^K C_k(t) \quad (1)$$

其中,式(1)应该满足的约束条件如下:

$$\text{Subject to: } \sum_{k=1}^K b_k(t) p_{kl} = B_l \quad (2)$$

$$b_k(t) = \sum_{i=1}^{W_k} \tilde{r}_f^i(t) > 0, \forall W_k \quad (3)$$

$$C_k(t) = \sum_{i=1}^{W_k} f_{i,j}^k, k \in K, i, j \in M, \forall i, j, k \quad (4)$$

$$f_{i,j}^{W_k} = \frac{S_k(t) + d_f(t)}{\tilde{r}_f^i(t)}, \forall f \quad (5)$$

对式(1)进行变换得:

$$\text{Minimize: } - \sum_{k=1}^K \sum_{w=1}^{W_k} \frac{\tilde{r}_f^i(t)}{S_k(t) + d_f(t)} \quad (6)$$

式(2)表示链路状态,在任何时刻聚合流带宽不能超过链路容量,即网络内瓶颈。为简化模型,式(3)在单位时间内将 Coflow 分配的带宽看作 Coflow 分配的流速和。一方面,为最小化 CCT,该问

$$L(\tilde{r}_f^i, \lambda, v) = f_0(\tilde{r}_f^i) + \sum_{i=1}^{W_k} \lambda_i f_i(\tilde{r}_f^i) + \sum_{j=1}^L v_j h_j(\tilde{r}_f^i) = \left( - \sum_{k=1}^K \sum_{w=1}^{W_k} \frac{1}{S_k(t) + d_f(t)} - \sum_{i=1}^{W_k} \sum_{j=1}^L \lambda_i + \sum_{j=1}^L \sum_{k=1}^K \sum_{w=1}^{W_k} v_j p_{kl} \right) \tilde{r}_f^i(t) - \sum_{j=1}^L v_j B_j \quad (7)$$

$$g(\lambda, v) = \inf_{r \in D} L(\tilde{r}_f^i, \lambda, v) = \inf_{r \in D} \left( f_0(\tilde{r}_f^i) + \sum_{i=1}^{W_k} \lambda_i f_i(\tilde{r}_f^i) + \sum_{j=1}^L v_j h_j(\tilde{r}_f^i) \right) = \inf_{r \in D} \left( \left( - \sum_{k=1}^K \sum_{w=1}^{W_k} \frac{1}{S_k(t) + d_f(t)} - \sum_{i=1}^{W_k} \sum_{j=1}^L \lambda_i + \sum_{j=1}^L \sum_{k=1}^K \sum_{w=1}^{W_k} v_j p_{kl} \right) \tilde{r}_f^i(t) - \sum_{j=1}^L v_j B_j \right) \quad (8)$$

对偶函数是一族关于 $(\lambda, v)$ 仿射函数的逐点下确界。此外,对偶函数是原问题最优解 $p^*$ 的下界,即对于任意 $\lambda \geq 0$ 和 $v$ 成立,如式(9)所示:

$$g(\lambda, v) \leq p^* \quad (9)$$

设 $(\tilde{r}, \lambda, v)$ 为原问题的一个可行解,如式(10)

$$g(\lambda, v) = \begin{cases} - \sum_{j=1}^L v_j B_j, - \sum_{k=1}^K \sum_{w=1}^{W_k} \frac{1}{S_k(t) + d_f(t)} - \sum_{i=1}^{W_k} \sum_{j=1}^L \lambda_i + \sum_{j=1}^L \sum_{k=1}^K \sum_{w=1}^{W_k} v_j p_{kl} = 0 \\ -\infty, - \sum_{k=1}^K \sum_{w=1}^{W_k} \frac{1}{S_k(t) + d_f(t)} - \sum_{i=1}^{W_k} \sum_{j=1}^L \lambda_i + \sum_{j=1}^L \sum_{k=1}^K \sum_{w=1}^{W_k} v_j p_{kl} \neq 0 \end{cases} \quad (11)$$

虽然式(11)成立,但是当 $g(\lambda, v) = -\infty$ 时,对偶函数的意义不大。只有当 $\lambda \geq 0$ 且 $(\lambda, v) \in \text{dom}g$ 时,即 $g(\lambda, v) \neq -\infty$ 时,对偶函数才能给出 $p^*$ 的一个非平凡下

$$\max g(\lambda, v) = \begin{cases} - \sum_{j=1}^L v_j B_j, - \sum_{k=1}^K \sum_{w=1}^{W_k} \frac{1}{S_k(t) + d_f(t)} - \sum_{i=1}^{W_k} \sum_{j=1}^L \lambda_i + \sum_{j=1}^L \sum_{k=1}^K \sum_{w=1}^{W_k} v_j p_{kl} = 0 \\ -\infty, - \sum_{k=1}^K \sum_{w=1}^{W_k} \frac{1}{S_k(t) + d_f(t)} - \sum_{i=1}^{W_k} \sum_{j=1}^L \lambda_i + \sum_{j=1}^L \sum_{k=1}^K \sum_{w=1}^{W_k} v_j p_{kl} \neq 0 \end{cases} \quad (12)$$

$$\text{s.t. } \lambda \geq 0$$

该问题可以等价:

$$\begin{aligned} &\text{Maximize} - \sum_{j=1}^L v_j B_j \\ &\text{s.t. } \sum_{j=1}^L \sum_{k=1}^K \sum_{w=1}^{W_k} v_j p_{kl} - \sum_{k=1}^K \sum_{w=1}^{W_k} \frac{1}{S_k(t) + d_f(t)} \geq 0 \end{aligned} \quad (13)$$

题可看作是一个线性规划问题, $1/(S_k(t) + d_f(t))$ 可以看作流 $f$ 的流速 $\tilde{r}_f^i(t)$ 的权重,然后加权和求最小,为了使加权和最小,即需要增大流速 $\tilde{r}_f^i(t)$ ;另一方面,增大流速会造成链路争用,产生链路间瓶颈,导致拥塞,反而会增加总体 CCT。因此, Coflow 调度过程需要在增大吞吐量与减少瓶颈之间实现平衡。

此外,最小化 CCT 是 NP-hard 问题。在实际生产中,网络中多个 Coflow 可能同时竞争链路资源,导致 Coflow 调度更加复杂。凸优化广泛应用于机器学习等领域。凸优化是通过寻找这些非凸优化问题中“凸”的结构,从而解决众多非凸优化的问题。最小化 CCT 的 NP-hard 问题是通过将原问题转换为凸问题,提高求解速率。

## 2.2.2 Lagrange 优化

Lagrange 对偶<sup>[23]</sup>思想是在目标函数中考虑问题的约束条件,原问题的 Lagrange 函数如式(7)所示。原函数的 Lagrange 对偶函数如式(8)所示。

$$L(\tilde{r}, \lambda, v) = f_0(\tilde{r}) + \sum_{i=1}^{W_k} \lambda_i f_i(\tilde{r}) + \sum_{j=1}^L v_j h_j(\tilde{r}) \leq f_0(\tilde{r}) \quad (10)$$

因此,  $g(\lambda, v) = \inf_{r \in D} L(\tilde{r}_f^i, \lambda, v) \leq L(\tilde{r}, \lambda, v) \leq f_0(\tilde{r})$ 。每个可行解 $\tilde{r}$ 都满足 $g(\lambda, v) \leq f_0(\tilde{r})$ ,如式(11)所示:

$$L(\tilde{r}, \lambda, v) = f_0(\tilde{r}) + \sum_{i=1}^{W_k} \lambda_i f_i(\tilde{r}) + \sum_{j=1}^L v_j h_j(\tilde{r}) \leq f_0(\tilde{r}) \quad (10)$$

因此,  $g(\lambda, v) = \inf_{r \in D} L(\tilde{r}_f^i, \lambda, v) \leq L(\tilde{r}, \lambda, v) \leq f_0(\tilde{r})$ 。每个可行解 $\tilde{r}$ 都满足 $g(\lambda, v) \leq f_0(\tilde{r})$ ,如式(11)所示:

$$L(\tilde{r}, \lambda, v) = f_0(\tilde{r}) + \sum_{i=1}^{W_k} \lambda_i f_i(\tilde{r}) + \sum_{j=1}^L v_j h_j(\tilde{r}) \leq f_0(\tilde{r}) \quad (10)$$

因此,  $g(\lambda, v) = \inf_{r \in D} L(\tilde{r}_f^i, \lambda, v) \leq L(\tilde{r}, \lambda, v) \leq f_0(\tilde{r})$ 。每个可行解 $\tilde{r}$ 都满足 $g(\lambda, v) \leq f_0(\tilde{r})$ ,如式(11)所示:

因此,不等式的对偶问题是在满足约束 $\lambda \geq 0$ 的条件下极大化对偶函数 $g(\lambda, v)$ ,如式(12)所示:

$$L(\tilde{r}, \lambda, v) = f_0(\tilde{r}) + \sum_{i=1}^{W_k} \lambda_i f_i(\tilde{r}) + \sum_{j=1}^L v_j h_j(\tilde{r}) \leq f_0(\tilde{r}) \quad (10)$$

因此,不等式的对偶问题是在满足约束 $\lambda \geq 0$ 的条件下极大化对偶函数 $g(\lambda, v)$ ,如式(12)所示:

因此,不等式的对偶问题是在满足约束 $\lambda \geq 0$ 的条件下极大化对偶函数 $g(\lambda, v)$ ,如式(12)所示:

因此,不等式的对偶问题是在满足约束 $\lambda \geq 0$ 的条件下极大化对偶函数 $g(\lambda, v)$ ,如式(12)所示:

因此,不等式的对偶问题是在满足约束 $\lambda \geq 0$ 的条件下极大化对偶函数 $g(\lambda, v)$ ,如式(12)所示:

因此,不等式的对偶问题是在满足约束 $\lambda \geq 0$ 的条件下极大化对偶函数 $g(\lambda, v)$ ,如式(12)所示:

$$\nabla f_0(\tilde{r}^l) + \sum_{i=1}^{W_i} \lambda_i^* \nabla f_i(\tilde{r}^l) + \sum_{j=1}^L v_j^* \nabla h_j(\tilde{r}^l) = 0 \quad (14)$$

显然,必存在 $(\lambda^*, v^*)$ 使得下式成立:

$$\begin{aligned} & -\sum_{k=1}^K \sum_{w=1}^{W_k} \frac{1}{S_k(t) + d_f(t)} - \sum_{i=1}^{W_i} \sum_{k=1}^K \lambda_i^* + \sum_{j=1}^L \sum_{k=1}^K \sum_{l=1}^L v_j^* p_{kl} = 0 \\ & -\sum_{i=1}^{W_i} \lambda_i^* \leq 0, \sum_{j=1}^L v_j^* \left( \sum_{k=1}^K \sum_{l=1}^L p_{kl} \tilde{r}_l(t) - B_j \right) = 0 \end{aligned} \quad (15)$$

因此,在上述凸优化过程中满足KKT最优性条件。本文调度机制通过Lagrange优化将原问题转换为凸优化问题进行求解,加快求解速度,求解每条链路的最优解 $\tilde{r}^l$ ,使得CCT最小。Lagrange优化是通过增加Coflow流的流速来减小CCT,客观上会造成链路争用,形成网络拥塞。

### 2.2.3 Lyapunov 优化

Lagrange 优化<sup>[24]</sup>在增大流速过程中会占用链路带宽,造成Coflow流在等待队列中积压,导致网络拥塞。本文设计全新的瓶颈因子,确保队列的稳定性,以优化多级反馈队列的Coflow调度。队列稳定性指网络总是在追求一个理想的状态,以确保所有到达的数据在有限时间内离开缓冲区,实现高性能与公平性之间的平衡。在现实场景中,网络大多数处于不平衡状态,但队列稳定性可以使其具有理想的均衡特性。

发送队列模型如式(16)~式(19)所示:

$$\text{Maximize } \sum_{i=1}^L \left( \frac{1}{\beta} \right) \ln(1 + \beta r_f^l(t)) \quad (16)$$

$$\text{s.t. } \sum_{i \in N(l)} r_f^l(t) \leq B_l, \forall l \in \{1, 2, \dots, L\} \quad (17)$$

$$\beta = \frac{S_k}{W_k}, k \in K, r_f^l \in [0, \tilde{r}^l], \forall l \in \{1, 2, \dots, L\} \quad (18)$$

$$\phi_i(r_f^l) = \left( \frac{1}{\beta} \right) \ln(1 + \beta r_f^l) \quad (19)$$

其中: $\tilde{r}^l$ 为流最大流速; $\ln(1 + \beta r_f^l(t))$ 为严格的凸函数; $\beta r_f^l$ 为优先级系数,即瓶颈因子。虚拟队列中Coflow流的优先级与 $\beta$ 相关,即Coflow流发送的数据越大,其数据本身也就越大,并且其宽度越小时,优先级越高。 $r_f^l$ 是Coflow流的流速,流速状态直接表示链路的拥塞情况,因此,流速 $r_f^l$ 对虚拟队列中Coflow流的优先级的影响极为关键。虚拟队列如式(20)所示:

$$Q_l(t+1) = \max \left[ Q_l(t) + \sum_{i \in L(l)} r_f^l(t) - B_l, 0 \right] \quad (20)$$

其中: $Q_l(t)$ 为虚拟队列的长度,其与分配Coflow流的流速 $r_f^l$ 及链路带宽 $B_l$ 直接相关。

到达云计算平台的独立任务具有随机性,离散时间排队过程 $Q_l(t)$ 达到队列稳定状态,如式(21)所示:

$$\lim_{t \rightarrow \infty} \frac{E\{Q(t)\}}{t} = 0 \quad (21)$$

Lyapunov 函数如式(22)所示:

$$L(Q(t)) = \frac{1}{2} \sum_i Q_i^2(t) \quad (22)$$

则单位时间内Lyapunov漂移可以表示为:

$$\Delta(Q(t)) = E(L(Q(t+1)) - L(Q(t)) | Q(t)) \quad (23)$$

本文通过Lyapunov漂移 $\Delta(Q(t))$ 求解队列稳定性。虚拟队列的DPP如式(24)所示:

$$-V\phi_i(r_f^l(t)) + \Delta(Q(t)) \quad (24)$$

其中: $V$ 为惩罚因子,通过最小化式(24),得到网络稳定性。然后直接求解随机和非线性Lyapunov漂移式(24)。本文通过在式(25)中推导出式(24)的上界,然后以极小化式(24)的上界为目标,得到极小化式(24)。基于Lyapunov优化的启发式搜索算法中, $\forall V \geq 0$ 和 $\forall Q(t)$ ,使得式(24)转化为:

$$-V\phi_i(r_f^l(t)) + \Delta(Q(t)) \leq \alpha - V\phi_i(r_f^l(t)) + \sum_i Q_i(t)(r_f^l(t) - B_l) \quad (25)$$

其中: $\alpha$ 为最差情况值的上界。 $\alpha$ 值是有限的,由于集合 $\chi$ 被假定为连续的,因此对于每个时间间隙有 $\alpha \geq \sum_i (r_f^l - B_l)^2$ 。在式(20)两边同时平方,可以得到如下不等式:

$$\begin{aligned} [Q_l(t+1)]^2 & \leq [Q_l(t)]^2 + [r_f^l(t) - B_l(t)]^2 - \\ & 2Q_l(t)(B_l(t) - r_f^l(t)) \end{aligned} \quad (26)$$

对不等式(26)求和,得到:

$$\begin{aligned} & \frac{\sum_l ([Q_l(t+1)]^2 - [Q_l(t)]^2)}{2} \leq \\ & \frac{\sum_l ([r_f^l(t) - B_l(t)]^2)}{2} - \sum_l Q_l(t)(B_l(t) - r_f^l(t)) \end{aligned} \quad (27)$$

因此,Drift-plus-penalty算法的输入是监控队列 $Q_1(t), Q_2(t), \dots, Q_L(t)$ ,输出是每个流期望流速 $r_f^l(t) = (r_1^l(t), r_2^l(t), \dots, r_f^l(t))$ ,算法步骤如下:

1) 对于时隙 $t$ ,网络控制器监控队列 $Q_1(t), Q_2(t), \dots, Q_L(t)$ ,选择 $r(t) = (r_1^l(t), r_2^l(t), \dots, r_f^l(t)) \in \chi$ ,使式(28)最小化:

$$\text{minimize } -V \sum_{i=1}^N \phi_i(r_f^l(t)) + \sum_{l=1}^L Q_l(t) \left[ \sum_{f \in N(l)} r_f^l(t) \right] \quad (28)$$

2) 通过式(20)更新虚拟队列;

3) 更新平均变量:

$$\bar{r}_f^l(t) = \frac{1}{t+1} \sum_{\tau=0}^t r_f^l(\tau) = \bar{r}_f^l(t) \left( \frac{t}{t+1} \right) + r_f^l(t) \left( \frac{1}{t+1} \right) \quad (29)$$

给定时间间隙 $t$ ,通过搜寻 $r_f^l(t) \in \chi$ 使得 $r_f^l(t)$ 最小,使用最小化 $r_f^l(t)$ 来更新下一个时隙的队列值 $Q_1(t+1), Q_2(t+1), \dots, Q_L(t+1)$ ,得特解:

$$r_i^*(t) = \left[ V \frac{1}{\beta(U_i(t)-1)} \right]_0^{r_{\max}}, U_i(t) = \sum_{l=1}^L Q_l(t) \quad (30)$$

Coflow 调度假定所有到达者都被立即放置在路径的所有链接上,是近似于实际网络动态排队的方法。

## 2.3 调度机制设计

基于瓶颈感知的 Coflow 调度机制 Bamq 分为调节机制优化和调度模型 2 个部分。

### 2.3.1 调度机制优化

整个调度机制分为信息收集、数据预处理、性能优化 3 个步骤。

1) 信息收集。Coflow 是具有语义相关的一组通信数据流,若一组 Coflow 流没有发送,便无法提前获取每个流发送的字节数、Coflow 关系和 Coflow 宽度信息,采用文献[8]和文献[12]提出的两种方法来收集信息。信息收集一方面采集 Coflow 相关信息,另一方面监控整体网络状态。当 Coflow 较多时,会

产生链路竞争,导致网络拥塞,信息收集模块采集 Coflow 调度过程中的网络信息,将信息传给调度器进行网络优化。

2) 数据预处理。在初始化阶段, Coflow 调度采用简单的调度机制,即平均分配每个流的带宽,等待队列采用 FIFO 的策略。

3) 性能优化。传统的 Coflow 调度采用平均分配带宽的策略,造成大量的剩余空间,影响 Coflow 的调度性能。针对上述问题, Lagrange 优化增大部分 Coflow 的流速,从而降低 CCT。因增大吞吐量会导致 Coflow 流产生链路争用,造成网络拥塞。因此, Bamq 通过 Lyapunov 优化,确定调度队列中不同 Coflow 流的优先级,达到增大吞吐量与减少拥塞平衡的目的。

### 2.3.2 调度模型

根据 Coflow 调度机制,基于瓶颈感知的多级反馈队列 Coflow 调度模型如图 2 所示。

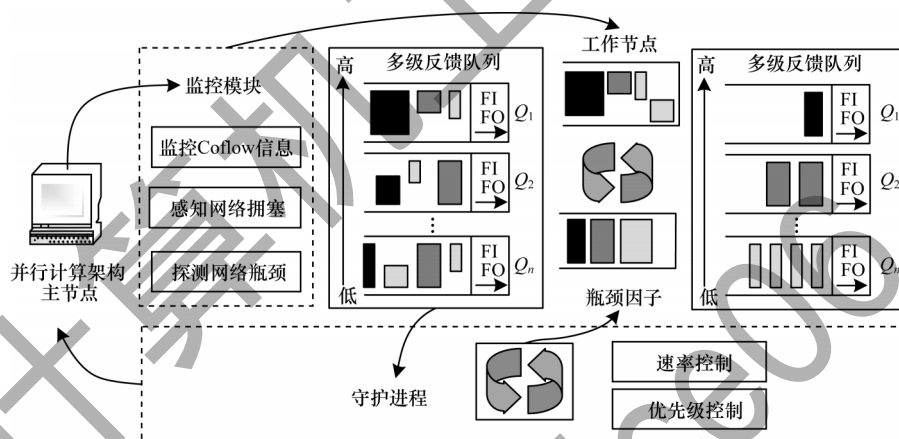


图2 本文调度模型结构

Fig.2 Structure of the proposed scheduling model

Coflow 调度模型包括主节点、工作节点、守护进程和监视器。主节点是系统的大脑,起着控制器的作用;工作节点具有优化信息的作用;守护进程则作为全局调度器,协同处理主节点,从工作节点收集 Coflow 信息,执行 Coflow 优先级调度;监控模块则收集 Coflow 信息,监控链路状态以判断拥塞情况。

当发送一组 Coflow 后,监视器收集 Coflow 信息,包括每个流的发送字节数、Coflow 关系和 Coflow 宽度信息,为调度提供初始信息。守护进程根据监控信息,初始化调度过程,采用简单的调度策略处理数据,即平均分配每个流的带宽,等待队列采用 FIFO 的策略等。同时,监控节点实时监控链路状况,检测到拥塞时,立即通知主节点的调度器进行优化。

随着 Coflow 流的增加,简单的调度机制已不能满足 CCT 需求,通过 Lagrange 优化,增大部分流流速,从而缩短 CCT。然而增大吞吐量会产生链路竞争,部分流会进入等待队列,影响 CCT 性能。Lyapunov 优化根据等待队列中的 Coflow 流的瓶颈因子,为其分

配不同的优先级,并进行优先调度,达到增大吞吐量与减少拥塞的稳定状态,最终降低 CCT。

### 算法2 Bamq 算法

输入  $B, Q, A, f_k^w, l$

输出  $\tilde{r}_i^l(t)$

1./\*Coflow 调度机制\*/

2.初始化  $\sigma$ /\*初始化链路拥塞阈值  $\sigma$ \*/

3.for  $i \leftarrow 1$  to  $I$  do

4 update

5./\*利用 Lagrange 优化调度\*/

6.while( $Q_i$ )

7.初始化  $f_i \leftarrow f_k^w, W_k, S_k(t), d_i(t) >$

8.计算 Coflow 完成时间  $T$

9.通过梯度下降求解式(15),得到每个流的期望速率  $\tilde{r}_i^l(t)$

10.if( $\tilde{r}_i^l(t) > \sigma \tilde{b}_i, l \in L$ )/\*判断链路是否拥塞\*/

11. $F_i = F_i - f_i$ , break

12.Let  $(A \leftarrow F_i) \wedge (Q_i \leftarrow Q_i - F_i)$

13.end while

14./\*利用 Lyapunov 优化调度\*/

15.Let  $F_i = F_i + f_i$



- 16.通过 DPP 解式(30)
  - 17.得到每个流的最终速率  $\tilde{r}_i(t)$
  - 18.update  $(A \leftarrow F_i) \wedge (Q_i \leftarrow Q_i - F_i)$
  - 19.end for
- 注  $\sigma$  表示链路拥塞因子,  $A$  表示已发送队列。

3 模拟实验

本文基于小规模流级别的模拟器和开源网络模拟器,使用 Facebook 的真实数据集对 Bamq 调度机制进行验证,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 Bamq 调度机制的有效性。

3.1 实验设置

3.1.1 调度模型实验环境

CoflowSim 是基于 Java 语言的开源调度器,已经实现了多种 Coflow 调度机制,如 Varys、Aalo,被广泛应用于 Coflow 领域。本文在单机模拟下进行实验, CoflowSim 模拟器部署环境具体为: Intel®Core™i7-9700 CPU 3.00 GHz、64 GB 内存,操作系统为 Linux version 4.15.0-142-generic, JDK 为 java 1.8.0\_231。实验使用 maven 工具将 CoflowSim 导入到 IDEA,通过 CoflowSim 内部数据中心的抽象模型,验证 Coflow 调度机制。

本文采用 Facebook 改进的数据集。原始的 Facebook 数据集包含 3 000 台 150 机架的 MapReduce 集群,其合成后数据包含 526 个 Coflow 信息。每个数据包含以下信息: Coflow ID, mapper 数量, reduce 数量, 总 Shuffle 字节数, 最大 Shuffle 字节数, 持续时间, Coflow 已发送字节数, Coflow 宽度等。由于 Facebook 记录的 Coflow 仅包含发送主机、接收主机和传送的字节数,没有包含流级别信息,且不包含任务信息。因此,本文实验随机划分为多个 Coflow 任务,使其构成依赖关系。Facebook 数据集如图 3 所示。

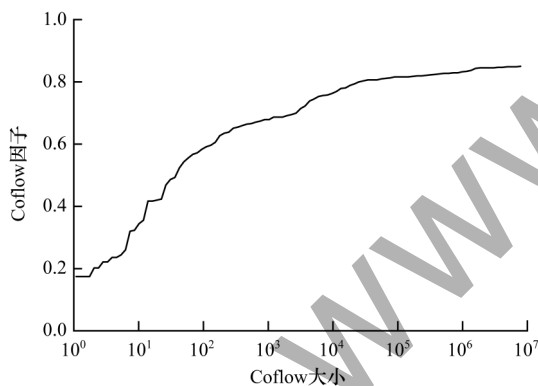


图3 Facebook 数据集相关信息

Fig.3 Related information of Facebook dataset

Facebook 数据类型如表 2 所示。根据长度 (Long 和 Short) 和宽度 (Narrow 和 Wide) 分为 4 类, SN 表示短窄流 (Short-Narrow); LN 表示长窄流 (Long-Narrow); SW 表示短宽流 (Short-Wide); LW 表示长宽流 (Long-Wide)。若 Coflow 流的最长流小

于 5 MB, 则为短流 (Short); 若 Coflow 流的宽度小于 50, 则为窄流 (Narrow)。此外, 基于 Facebook 记录的 Coflow 信息不包含流速、吞吐量等信息, 因此需要使用开源网络模拟器, 模拟链路状况。

表 2 Facebook 数据集的 Coflow 类型

Table 2 Coflow types of Facebook data set %

| Coflow 类型 | 大小占比 | 内存占比  |
|-----------|------|-------|
| SN        | 52   | 0.01  |
| LN        | 16   | 0.67  |
| SW        | 15   | 0.22  |
| LW        | 17   | 99.10 |

NS-3 (Network Simulator) 是一个用于 Internet 系统的离散事件网络模拟器, 用于面向对象编程 C++ 开发的开源项目。NS-3 模拟器抽象出几个核心概念, 即节点是连接到网络的最基本实体, 包括用于应用程序、协议和网络设备的容器类。网络系统是分配 MAC 地址, 以及配置节点的协议栈。基于这些特点, NS-3 模拟器简化了 api 的繁琐工作, 用于构建各种类型的仿真场景。NS-3 模拟器部署环境为: Ubuntu16.04.7 LTS、ns-allinone-3.32、gcc5.4.0。实验通过编写 C++ 代码模拟链路情况, 验证调度机制的有效性。

3.1.2 实验对比

Bamq 调度机制是一种改进的多级队列调度机制, 通过与以下 3 种经典机制对比, 以验证 Bamq 调度的有效性。

1) Baraat 机制: 用于数据中心的分布式任务感知调度系统, 将任务感知调度问题转化成流优先级问题。通过一种简单的启发式方法设置流的优先级, 结合优先级和显式速率协议的优点, 解决繁重任务的实时识别并相应地改变多路复用的级别等问题。

2) Varys 机制: 是一种已知信息下最短作业优先的多级队列反馈机制, 基于网络瓶颈处的完成时间调度 Coflow, 然后使用最小化分配期望持续时间算法将速率分配给各个流, 进而降低平均 CCT。

3) Aalo 机制: 提出 Coflow-Aware Least-Attained Service 的 Inter-Coflow 调度器, 是最少获得优先服务的多级队列反馈机制, 为每个 Coflow 分配优先级且该优先级随着 Coflow 已经发送总字节数的增加而减小, 以此降低平均 CCT。

Coflow 调度过程的性能指标如下:

1) CCT: 是 Coflow 调度机制中最主要的指标, 分别计算平均 CCT 和 95% CCT。为消除极值影响, 95% CCT 是将每种机制前 2.5% 和后 2.5% 结果去掉, 重新计算结果。通过对 CCT 进行归一化处理, 得到 Coflow 完成时间。

2) 吞吐量: 是数据中心应用关注的基本指标。在数据中心的, 除了数据量小的 Coflow 外, 还有数据量大的后台 Coflow (>1GB) 用于数据更新。如此大的 Coflows 占传输总字节的 99.1%, 因此吞吐量也能评价调度性能。

### 3.2 实验结果与分析

本文选择 CCT 与吞吐量 2 个指标作为判断标准。CCT 指标作为判断 Coflow 调度性能的直接指标被广泛使用。为有效对比链路的利用率,本文对不同调度机制的吞吐量进行对比。

#### 3.2.1 CCT 指标

本文对 Facebook 数据集中 526 个 Coflow 进行随机分配任务,其任务到达时间遵循参数为  $\theta$  的 Poisson 分

布,并将 4 种不同类型的 Coflow 流作为工作负载。图 4 表示 Baraat、Varys、Aalo 与 Bamq 的 Coflow 完成时间对比。从图 4(a)可以看出,Bamq 的 CCT 平均比 Baraat、Varys、Aalo 分别提高 9.2%、13.5%、39.1%。这是因为 Bamq 考虑网络内瓶颈,从而比 Varys 具有更精确的 Coflow 优先级和带宽分配,Bamq 提高了链路利用率。从图 4(b)可以看出,95% CCT 中 Bamq 的 CCT 平均比 Baraat、Varys、Aalo 降低 9.6%、14.6%、39.7%。Bamq 和 Baraat 提高了链路利用率。

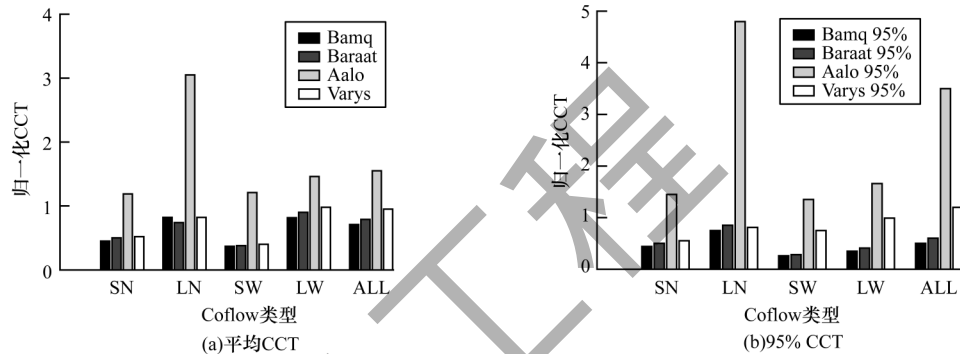


图 4 不同调度机制的 CCT 对比

Fig 4 CCT comparison among different scheduling mechanisms

图 5 表示 4 种调度机制 Coflow 完成时间的累积分布函数(Cumulative Distribution Function, CDF)。从图 5 可以看出,在这两种负载下,Bamq 几乎在所

有百分比上都优于 Varys、Baraat 和 Aalo。在负载较小时,Bamq 的 CCT 与 Baraat 相似,主要与收敛速度相关。这个结果与图 4 的结果一致。

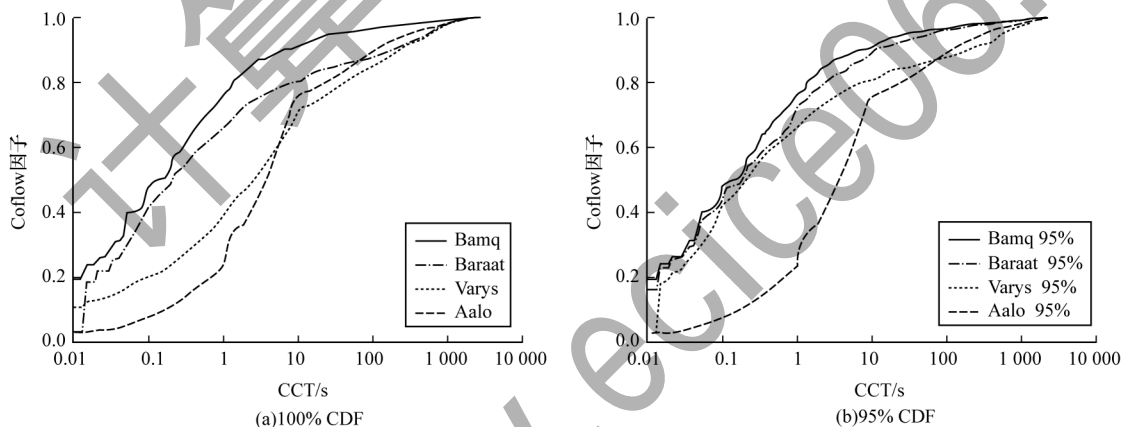


图 5 不同调度机制的 Coflow 累积分布函数对比

Fig 5 Cumulative distribution function of Coflow comparison among different scheduling mechanisms

#### 3.2.2 吞吐量

Baraat、Varys、Aalo、Bamq 这 4 种调度机制在不同工作负载下的吞吐量对比如图 6 所示。吞吐量是通过整个网络传输的所有数据量除以所有 Coflow 的最大完成时间。在较低负荷下,4 种算法具有相似的吞吐量,在更大的负载下,Bamq 的吞吐量明显高于 Varys,略低于 Baraat,相对于 Baraat, Bamq 有一个收敛过程吞吐量损失很小。因此,Bamq 在优化 CCT 的同时保持了良好的链接利用率。

为验证 Bamq 调度机制的收敛性,本文随机选取 3 个流,3 个流的收敛速度如图 7 所示。除了给初始流包让路外,在短时间内 3 个流大多收敛到最优速率,收敛前的平均速率也不小于最优速率,因为流量

是从较大值开始收敛的。

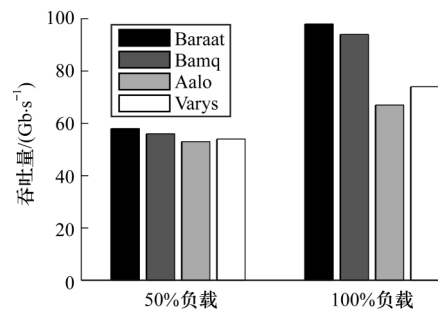


图 6 不同调度机制的吞吐量对比

Fig 6 Throughput comparison among different scheduling mechanisms



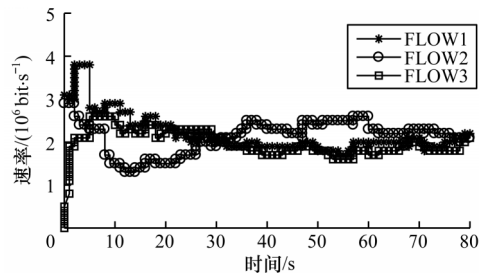


图7 流的收敛速度

Fig.7 Convergence rate of flow

3.2.3 结果分析

Bamq 调度机制是一种改进的多级队列调度机制。初始化 Coflow 优先级,然后在队列下,最大化 Coflow 流的流速,以此降低 CCT。通过设置多级队列来分配其他 Coflow 流,以此减低拥塞。

Bamq 根据 Coflow 的已发大小、宽度、流速设计了全新的瓶颈因子,根据瓶颈因子大小决定 Coflow 的优先级。流速是识别链路拥塞情况的关键信息,根据流速大小可以快速判断网络状况。为评估 CCT 与 Coflow 宽度等属性的相关性,本文通过选择 Person 相关系数 (PCC) 作为评估指标。PCC 定义如下:

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} \quad (31)$$

PCC 可以很好衡量两个变量的线性关系,值越大,表明两个变量的线性相关性越强。Person 相关性系数在计算 2 个总体之间的相关性时要求两个总体必须符合正态分布,对 Person 相关性系数取对数,使其符合正态分布。Coflow 属性相关度如表 3 所示。

表3 Coflow 属性相关度

Table 3 Correlation degree of Coflow attributes

| 属性  | 大小   | 宽度   | 长度   | CCT  |
|-----|------|------|------|------|
| 大小  | 1.00 | 0.92 | 0.70 | 0.97 |
| 宽度  | 0.92 | 1.00 | 0.35 | 0.82 |
| 长度  | 0.70 | 0.35 | 1.00 | 0.81 |
| CCT | 0.97 | 0.82 | 0.81 | 1.00 |

从表 3 可以看出,CCT 与 Coflow 大小强相关,由于 CCT 反映 Coflow 的完成时间,必然与 Coflow 大小线性相关。CCT 与 Coflow 宽度强相关,因此,Bamq 的瓶颈因子可以很好地反映 CCT 性能。Coflow 大小与宽度强相关。在 Hadoop 分布式框架中,Map 任务的数量与输入数据的大小相关,即输入数据越大,map 任务越多,其相应的 reduce 任务越多,Coflow 宽度越大。

4 结束语

针对在云数据中心 Coflow 调度过程中存在网络瓶颈的问题,本文提出基于瓶颈感知的多级反馈队列 Coflow 调度机制 Bamq。利用 Lagrange 对偶优化 Coflow 流速,以充分利用链路带宽,从而减少 CCT,为减少网络拥塞,利用 Lyapunov 优化多级反馈队列模型,以确保队列稳定性。在 Facebook 的真实数据集和开源网络模拟器 NS-3 上的实验结果表明,

Bamq 调度机制在降低平均 CCT 的前提下,能够增大吞吐量,并且提高链路利用率。下一步将利用光电路交换机制优化流量调度,并结合边缘技术提高核心互联网的带宽利用率。

参考文献

[1] 李文信,齐恒,徐仁海,等. 数据中心网络流量调度的研究进展与趋势[J]. 计算机学报,2020,43(4):600-617. LI W X, QI H, XU R H, et al. Data center network flow scheduling progress and trend [J]. Chinese Journal of Computers, 2020, 43 (4): 600-617. (in Chinese)

[2] ZAHARIA M, CHOWDHURY M, FRANKLIN M J, et al. Spark: cluster computing with working sets [EB/OL]. [2021-07-22]. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.374.2156&rep=rep1&type=pdf>.

[3] ISARD M, BUDIU M, YUAN Y, et al. Dryad: distributed data-parallel programs from sequential building blocks [EB/OL]. [2021-07-22]. <http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=060CD3BCF423260745F996B762464E19?doi=10.1.1.538.4272&rep=rep1&type=pdf>.

[4] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters [J]. Communications of the ACM, 2008, 51 (1): 107-113.

[5] MURRAY D G, SCHWARZKOPF M, SMOWTON C, et al. CIEL: a universal execution engine for distributed data-flow computing [C]//Proceedings of the 8th Symposium on Networked Systems Design and Implementation. New York, USA: ACM Press, 2011: 113-126.

[6] CHOWDHURY M, STOICA I. Coflow: a networking abstraction for cluster applications [C]//Proceedings of the 11th ACM Workshop on Hot Topics in Networks. New York, USA: ACM Press, 2012: 31-36.

[7] CHOWDHURY M, ZHONG Y, STOICA I. Efficient Coflow scheduling with Varys [C]//Proceedings of ACM Conference on SIGCOMM. New York, USA: ACM Press, 2014: 443-454.

[8] CHOWDHURY M, STOICA I. Efficient Coflow scheduling without prior knowledge [J]. ACM SIGCOMM Computer Communication Review, 2015, 45 (5): 393-406.

[9] DOGAR F R, KARAGIANNIS T, BALLANI H, et al. Decentralized task-aware scheduling for data center networks [J]. ACM SIGCOMM Computer Communication Review, 2014, 44 (4): 431-442.

[10] GAO P X, NARAYAN A, KUMAR G, et al. pHost: distributed near-optimal datacenter transport over commodity network fabric [C]//Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies. New York, USA: ACM Press, 2015: 1-12.

[11] ZENG Y, YE B, TANG B, et al. Scheduling Coflows of multi-stage jobs under network resource constraints [J]. Computer Networks, 2021, 184: 1-10.

[12] HONG Z, LI C, YI B, et al. CODA: toward automatically identifying and scheduling Coflows in the dark [C]//Proceedings of ACM SIGCOMM Conference. New York, USA: ACM Press, 2016: 160-173.

(上接第 201 页)

- [13] ZHANG S, ZHANG S, QIAN Z, et al. Efficient scheduling for multi-stage coflows[J]. CCF Transactions on Networking, 2019, 2(2): 83-97.
- [14] ZHANG J, GUO D, LI K, et al. Coflow scheduling in the multi-resource environment [J]. IEEE Transactions on Network and Service Management, 2019, 16(2): 783-796.
- [15] WANG Z, ZHANG H, SHI X, et al. Efficient scheduling of weighted Coflows in data centers[J]. IEEE Transactions on Parallel and Distributed Systems, 2019, 30(9): 2003-2017.
- [16] WANG S, ZHANG J, HUANG T, et al. Multi-attributes-based Coflow scheduling without prior knowledge [J]. IEEE-ACM Transactions on Networking, 2018, 26(4): 1962-1975.
- [17] WANG L, WANG W. Fair Coflow scheduling without prior knowledge [C]//Proceedings of the 38th International Conference on Distributed Computing Systems. Washington D. C. , USA: IEEE Press, 2018: 22-32.
- [18] SHI L, LIU Y, ZHANG J, et al. Coflow scheduling in data centers: routing and bandwidth allocation [J]. IEEE Transactions on Parallel & Distributed Systems, 2021, 32(11): 2661-2675.
- [19] CHEN Y, WU J. Joint Coflow routing and scheduling in leaf-spine data centers [J]. Journal of Parallel and Distributed Computing, 2021, 148: 83-95.
- [20] LI Y, JIANG H C, TAN H, et al. Efficient online coflow routing and scheduling [C]//Proceedings of the 17th ACM International Symposium. New York, USA: ACM Press, 2016: 161-170.
- [21] LIU L, XU H, GAO C, et al. Bottleneck-aware Coflow scheduling without prior knowledge [C]//Proceedings of IEEE Conference on Computer Communications Workshops. Washington D. C. , USA: IEEE Press, 2020: 50-55.
- [22] ZHANG T, SHU R, SHAN Z, et al. Distributed bottleneck-aware coflow scheduling in data centers [J]. IEEE Transactions on Parallel and Distributed Systems, 2019, 30(7): 1565-1579.
- [23] PALOMAR D P, MUNG C. A tutorial on decomposition methods for network utility maximization [J]. IEEE Journal on Selected Areas in Communications, 2006, 24(8): 1439-1451.
- [24] PENG M, YU Y, XIANG H, et al. Energy-efficient resource allocation optimization for multimedia heterogeneous cloud radio access networks [J]. IEEE Transactions on Multimedia, 2016, 18(5): 879-892.

编辑 薛晋栋