基于深度增强学习的数据中心网络coflow调度机制

马 腾,胡宇翔,张校辉
(国家数字交换系统工程技术研究中心,河南郑州 450002)

摘 要: 最小化语义相关流的平均完成时间是数据中心网络流量管理面临的难题之一.受人工智能领域深度增强学习方向的最新研究进展启发,本文提出一种的新的语义相关流调度机制.将带宽约束的语义相关流调度问题转化为连续的学习过程,通过学习以往策略实现最佳调度.引入反向填充和有限复用机制,保证系统的工作保持性和无饥饿性.仿真结果表明,在不同的网络负载下,本文提出的调度机制均使得语义相关流的平均完成时间小于其他调度机制,尤其是网络负载较大时,相比最先进的调度机制,性能提升约50%.

关键词: 数据中心网络;语义相关流;流调度
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2018)07-1617-08
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2018.07.011

Deep Reinforcement Learning Based Coflow Scheduling in Data Center Networks

MA Teng, HU Yu-xiang, ZHANG Xiao-hui
(*National Digital Switching System Engineering & Technology Research Center, Zhengzhou, Henan* 450002, *China*)

Abstract: Coflow completion time minimization is one of the challenges of traffic management in data center networks. Inspired by the newest research progress in deep reinforcement learning, which is one direction of artificial intelligence, this paper proposes a novel coflow scheduling mechanism. It translates the coflow scheduling problem with bandwidth constraint into a continuous learning process. By learning the previous decisions, the best scheduling is obtained. By introducing back filling and limited multiplexing mechanisms, the system is work-conserving and starvation-free. Simulation results show that, under different network load, compared with other scheduling mechanisms, the average coflow completion time is reduced. Especially when the network load is heavy, the proposed mechanism achieves about 50% performance improvement than the state-of-the-art scheduling mechanism.

Key words: data center network; coflow; flow scheduling

1 引言

当前的数据中心网络(Data Center network, DCN)中,经常采用Map Reduce[1],Spark[2],Dryad[3]等分布式数据处理平台完成相关计算任务.这些分布式数据处理平台的流量特征具有一个共同点:系统收到的任何计算任务将被分解成多个子任务,每个计算子任务分别由不同的主机完成,产生的中间数据在主机之间相互交换,作为下一个子任务各个执行主机的数据输入.在这种情况下,只要其中一条流对应的中间数据没有到达,下一个阶段计算任务就无法执行.有关统计结果指出,中间数据的传输占用的时间超过了总计算任务完成时间的33%,某些情况下达到50%[4].有鉴于此,优化分布式计算任务的数据传输,对提高DCN中应用程序的性能非常重要.

加州大学伯克利分校的Mosharaf Chowdhury博士将这种具有语义相关性的一组并发流称为一个coflow[5].为优化DCN中应用程序的性能,需要降低coflow的完成时间(Coflow Completion Time, CCT),这与传统网络降低单个流的完成时间具有天壤之别.从资源

收稿日期:2017-03-17;修回日期:2017-06-13;责任编辑:孙瑶
基金项目:国家973重点基础研究发展计划(No. 2013CB329104);国家863高技术研究发展计划(No. 2013AA013505)

利用率的角度讲,当执行某个计算阶段的主机等待上一阶段的处理结果时,相应虚拟机的 CPU 处于闲置状态,造成资源浪费. 所以,优化 coflow 的完成时间能够提高 DCN 的计算资源利用率,从而获得应用性能和 DCN 吞吐率的双重提升.

注意到在人工智能领域,结合深度学习和增强学习的深度增强学习方法取得了突破性成果,以此方法为核心的 AlphaGo[6~8] 接连战胜一大批人类围棋高手,成为举世瞩目的焦点. 深度增强学习算法近期被应用到了通信网络领域[9],解决网络资源管理问题. 受此启发,本文提出一种新的 coflow 调度机制——DeepCS (Deep Reinforcement Learning Based Coflow Scheduling). 它把 coflow 的资源视图视为待处理的图像,从而将带宽约束的 coflow 调度问题转化为时间连续的学习过程,通过学习以往策略实现 coflow 最佳调度.

2 相关工作

面向平均完成时间最小化的 DCN 流量调度方法归纳起来,可分为两类:一是针对单个流,最小化其平均完成时间;二是针对一组具有语意相关性的流,即 coflow,实现平均完成时间的最小化. 前一类的代表性工作有 D³[10]、PDQ[11]、pFabric[12] 等,后一类的代表性工作有 Varys[13], Baraat[14], Repair[15] 等. 鉴于后一类算法的 coflow平均完成时间更小,本节仅介绍后一类算法的代表工作.

Varys 采用集中式实现方案,作为抢占式系统,它对完成时间最小的流优先进行调度,降低了 DCN 中 coflow 的平均完成时间. 调度时,Varys 控制不同子流发送的速率,使一个 coflow 下所有流同一时刻完成传送,这样做节省了带宽,对其它的 coflow 传输有利.

Baraat 是分布式调度系统. 它在应用层为每个流附上一个 ID,先进入系统的 coflow 优先级较高,优先得到调度,于是网络中的流量实现了 coflow 级的 FIFO 调度. 为防止网络中的大流阻塞队头,Baraat 引入有限复用机制,允许后进入系统的小流和先进入系统的大流共享带宽. 另外,Baraat 允许交换机之间通告阻塞情况,从而减小发送速率、节省带宽.

Rapier 将调度机制和路由选择联合起来,优化 coflow的传输. 通过部署支持 openflow[16] 的交换机和控制器,集中式计算 coflow 的最优传输路径成为可能,从而不再依赖之前 DCN 中默认的 ECMP 路由. 但大量部署 openflow 交换机一方面要求 Rapier 只能在支持软件定义的 DCN 中使用,另一方面,集中式的路径计算增加了控制器的计算负担,控制开销很大,影响了 Rapier 的性能.

关于 coflow 调度的研究已经取得了不少成果,但这些工作均采用固定的调度策略,DCN 业务流量特征发生变化的情况下,调度算法的性能有待提高,有必要设计基于学习的自适应调度机制,这也是本文工作的出发点和直接动力.

3 coflow 调度机制

3.1 相关定义

定义 1 coflow(语意相关流) 两个主机集群之间传输的具有语义相关性的一组并发流量称为一个coflow.

定义 2 coflow 完成时间 从 coflow 第一个流开始传输至最后一个流传输完毕所用时间称为 coflow 完成时间.

对任意的流 $f_i \in c, i \in \{1,2,\cdots,|c|\}$,其起始时间、终止时间分别为 $\mathrm{start}(f_i)$、$\mathrm{end}(f_i)$,则 coflow c 的起始时间为 $\mathrm{start}(c) = \min\limits_{f_i} \mathrm{start}(f_i)$,终止时间为 $\mathrm{end}(c) = \max\limits_{f_i} \mathrm{end}(f_i)$,流完成时间为 $\bar{c} = \mathrm{end}(c) - \mathrm{start}(c)$.

定义 3 网络情景 在观测时间窗口内,网络边缘所有输入-输出链路带宽资源被流量占用形成的一组描述网络状态的快照,称为该时刻的网络情景.

当前研究 DCN 流量调度时,通常将数据中心网络内部视为理想化的无阻塞交换结构,网络的拥塞主要集中在边缘链路. 这样处理原因在于来自产业界 DCN 的统计数据表明,DCN 的核心链路很少出现严重、长期拥塞,但边缘链路与之相反[17]. 有鉴于此,本文将网络情景的概念限制到边缘链路.

定义 4 任务情景 在观测时间窗口内,网络中待处理的任务根据占用带宽和流传输时间在网络边缘所有输入-输出链路上形成的一组投影,称为该时刻的任务情景. 本文中的“任务”概念等同于一个 coflow.

需要说明的是,如果一个任务在某些输入输出链路没有流量传输,则任务在该链路的投影分量为 0. 网络情景、任务情景本质上均是一组图像.

为方便后文建立模型,这里将用到的变量统一列表,如表 1 所示.

3.2 深度增强学习系统

增强学习(Reinforcement Learning,RL)[18] 是受到生物能够有效适应环境的启发,以试错机制与环境进行交互,通过最大化累积奖赏的方式来学习最优策略. 增强学习系统由 4 个基本部分组成:状态 s,动作 a,状态转移概率 P,和奖赏信号 r,如图 1 所示. 策略 $\pi: S \to A$ 被定义为从状态空间到动作空间的映射. t 时刻智能体在当前状态 s_t 下根据策略 π 选择动作 a_t,执行该动作并以概率 $P^{a_t}_{s_t,s_{t+1}}$ 转移到下一状态 s_{t+1},同时接收到环境反馈回来的奖赏 r_t. 增强学习的目标是通过调整策略来最大化累积折扣奖赏. 策略 π 的累积折扣奖赏表达式为

$V^{\pi}(s)=E\left(\sum_{t=0}^{\infty}\gamma^{t}r_{t}\right)$,其中 $\gamma\in[0,1]$ 为折扣因子,使得临近奖赏比未来奖赏更重要.

表 1　变量符号列表

变量符号	变量定义
P_i^I、P_i^E	输入、输出端口
B_i	端口 P_i 总带宽
$\mathrm{Rem}(P_i)$	端口 P_i 的剩余带宽
$C_k=\{f_{ij}^k\}$	包含若干子流的 coflow C_k
Ω	网络中的 coflow 组成的集合
$M_k=[s_{ij}^k]$	coflow C_k 的流量矩阵
$W_k=[w_{ij}^k]$	coflow C_k 的已完成传输的流量矩阵
T_k	coflow C_k 的完成时间
f_{ij}^k	从端口 P_i 到 P_j 并隶属于 coflow C_k 的子流
s_{ij}	从端口 P_i 到 P_j 的总流量大小
s_{ij}^k	从端口 P_i 到 P_j 并隶属于 coflow C_k 的流量大小
r_{ij}	从端口 P_i 到 P_j 的总传输速率
r_{ij}^k	流 f_{ij}^k 的传输速率
t_{ij}^k	流 f_{ij}^k 的完成时间

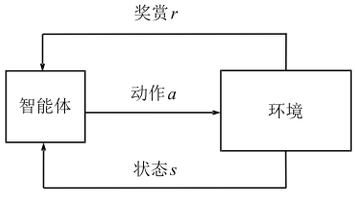


图1　增强学习的基本模型

智能体根据策略选择动作,一般定义系统的策略 $\pi(s,a):S\times A\rightarrow[0,1]$ 是一个状态-动作对到概率的映射,表示在状态 s 下选择动作 a 的概率,满足 $\sum_{a\in A}\pi(s,a)=1$. 解决实际问题时,通常状态-动作对非常之多,要求增强学习拥有泛化能力,能够利用有限学习经验、记忆完成大范围空间有效知识的获取与表示. 通常采用函数逼近方法(称为策略函数),其本质是采用参数化函数 $\pi_\theta(s,a)$ 逼近状态-动作对到概率的映射关系,这里的 θ 称为策略参数. 用于表示策略的逼近函数形式多样,在近期的研究工作中,深度神经网络(Deep Neural Network,DNN[19])被成功的用于求解大规模增强学习系统[6,8]. DNN 的优势在于特征提取的时候不需要人为手动设计,而采用纯学习的方式完成. 本文将 DNN 引入增强学习框架,构成深度增强学习系统,如图 2 所示.

策略梯度算法是直接逼近、优化策略,求取最优策略的有效手段. 增强学习的最终目标是最大化累积折扣奖赏(称为估值函数),策略梯度的表达式为

$$\nabla_\theta E_{\pi_\theta}\left(\sum_{t=0}^{\infty}\gamma^{t}r_{t}\right)=E_{\pi_\theta}(\nabla_\theta\log\pi_\theta(s,a)Q^{\pi_\theta}(s,a)) \quad (1)$$

式中,$Q^{\pi_\theta}(s,a)$ 表示根据策略 π_θ 在状态 s 时选择动作 a 所获得的累积折扣奖赏. 策略梯度算法的核心要义是在当前策略下通过观测运行轨迹获得梯度估计. 在简

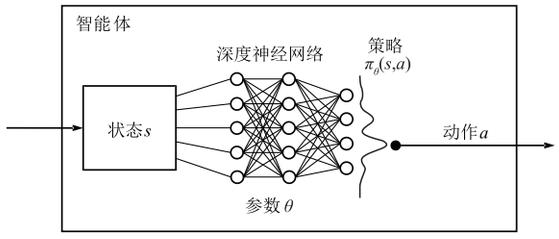


图2　智能体的构成

单蒙特卡洛方法中,智能体对轨迹进行多点采样,通过计算得出累积折扣奖赏 v_t,作为 $Q^{\pi_\theta}(s_t,a_t)$ 的无偏估计,然后按照下式对策略参数 θ 进行更新[20]:

$$\theta\leftarrow\theta+\alpha\sum\nolimits_{t}\nabla_\theta\log\pi_\theta(s_t,a_t)v_t \quad (2)$$

式中,α 表示步长. 这是 REIFORCE 算法给出的更新公式,其中 $\nabla_\theta\log\pi_\theta(s,a)$ 给出了每次迭代的步进方向,而 v_t 则决定了步进的大小. 虽然 REINFORCE 算法得到的梯度估计是无偏估计,但是其方差过大,影响算法的收敛速度,导致算法效率低下. 本文后面的内容将采用回报基线法,通过引入一个合理的足够小的变量(即 α)截取 v_t 的返回值,以有效的减少梯度估计的方差.

3.3　算法总体设计

由定义可知,网络情景以一组图像的形式刻画了网络边缘链路的带宽因 coflow 调度而被各条流量占用的情况,任务情景则以图像的形式刻画了边缘链路带宽占用情况可能出现的变化. 图 3 联合给出了网络情景、任务情景的一个示例. 最左端的一列图像中,不同颜色的区域代表不同已经被调度的任务的流量,时间范围从当前时刻到未来 5 个单位时间. 以红色任务为例,该任务对应的 coflow 的一条流需要在输入端 1 的链路上传输,占用 2 个单位带宽,持续 3 个单位时间. 其他列的图像均代表某个任务的任务情景. 以第二列为例,第一幅图像表示任务 1 得到调度时,对应的 coflow 的某条流将占用输入端 1 链路的 2 个单位带宽,传输时间为 2 个时间单位. 后文将给出任务情境中带宽和传输时间的计算方法.

任务进入系统,形成等待队列. 由于深度神经网络每次只能处理固定数量的图像,因此需要将等待队列的长度限制为一个常数 M,超出等待队列的任务不列为当前任务调度的考察对象. 当等待队列中一个或者多个任务已得到调度,则后续的任务进入等待队列进行调度. 等待队列中的任务按到达系统的时间排序. 在每一个离散时刻,调度器选择等待队列中 M 个任务中的一个进行调度,调度算法的动作空间记为 $\{\varnothing,1,2,3,\cdots,M\}$,这里动作 $a=i$ 代表等待队列中的第 i 个任务将被调度,$a=\varnothing$ 代表当前时刻不选择任何任务进行调度. 另外,当选择了无效的调度任务(当前系统不能满足任务所需带宽资源)时,系统不执行调度操作. 随着

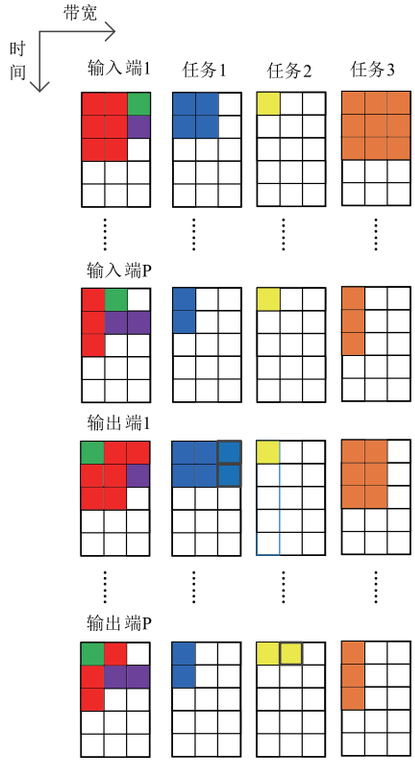


图3 网络情景的示例

时间点的推移,任务不断在合适的时刻得到调度. 每当一个任务得到调度,流量开始传输,传输过程不间断直至该流传输完毕. 同时,网络情景和任务情景均发生改变,新的任务进入等待队列.

奖赏信号 r 是增强学习系统的一个重要组成部分. 它通过评价动作的执行效果引导系统完成特定目标. 本文的目标是最小化 coflow 的平均完成时间,设定奖赏信号 $r=-|\mathcal{J}|$,其中 $\mathcal{J}$ 为当前时刻网络中尚未传输完毕的 coflow 集合. 注意到当折扣因子 $\gamma=1$ 时累积奖赏恰好等于网络中未完成传输的 coflow 数量,因此最大化累积奖赏相当于最小化 coflow 平均完成时间.

3.4 情景生成

DeepCS 的着眼点是为每个 coflow 达到完成时间分配最小带宽,而且使得属于一个 coflow 的流同时完成传输. 事实上,一个 coflow 的所有流中,完成时间最大的流决定了 coflow 的最终完成时间,该流所在的输入和输出链路可认为是影响该 coflow 传输的瓶颈链路. 为尽可能降低流完成时间,在瓶颈链路上流量 f^k_{i*j*} 应该满载传输,故 coflow C_k 的完成时间表达式.

$$\begin{aligned} T_k &= t^k_{ij} = t^k_{i*j*} \\ &= \max\left(\max_i \frac{\sum_j s^k_{ij} - w^k_{ij}}{\mathrm{Rem}(P^I_i)}, \max_j \frac{\sum_i s^k_{ij} - w^k_{ij}}{\mathrm{Rem}(P^E_j)}\right), \\ &\quad \forall (i,j) \neq (i^*,j^*) \end{aligned} \tag{3}$$

式(3)中,$\mathrm{Rem}(P^I_i)$ 表示 P^I_i 的输入链路剩余带宽,$\mathrm{Rem}(P^E_j)$ 表示 P^E_j 的输出链路剩余带宽. 于是对 $\forall f^k_{ij} \in C_k$,相应输入或输出链路分配给流 f^k_{ij} 的带宽为 $r^k_{ij} = s^k_{ij}/t^k_{ij}$.

coflow 中各流分配的带宽和完成时间的确定,意味着情景生成过程已经结束. 情景生成算法的具体流程如算法 1 所示.

算法 1 情景生成算法的伪代码

输入:DCN 中所有 coflow 的流量矩阵 $\{M_k\}$,边缘链路剩余带宽 $\mathrm{Rem}(P)$
输出:分配给所有 coflow 的流的带宽、传输时间 (r^k_{ij}, t^k_{ij})
1: Procedure Time_Bandwidth_Allocation(coflows Ω, BandwidthRem(P))
2: for Ω 中的每一个 coflow C_k
3: 根据式(3)计算 t^k_{i*j*}:瓶颈链路上的流 f^k_{i*j*} 的完成时间
4: for coflow C_k 中的每一条流 f^k_{ij}
5: $t^k_{ij} \leftarrow t^k_{i*j*}$
6: 根据式(3)计算分配给 f_{ij} 的带宽 r^k_{ij}
7: 更新 Rem(P)
8: end for
9: end for
10: return each (r^k_{ij}, t^k_{ij})
11: end procedure

3.5 训练过程

本文训练方法的输入为 3.1 节描述的一系列图像,包括网络情景、任务情景,输出为可选动作的概率分布. 训练过程以幕(episode,增强学习中的常用概念)为单位进行. 在每一幕中,固定数量的任务到达网络并根据策略函数进行调度,当所有任务执行完成后,该幕结束.

为使训练结果具有一般性,训练集中的任务到达时序应该遍历各种随机过程. 实际训练时,每一次迭代过程包含 N 个幕,这 N 个幕均采用任务到达时序符合特定随机过程的一个训练集,以探测在当前策略函数下动作空间的概率分布,并利用探测到的结果提升策略函数在各类训练集下的训练效果. 不同的迭代过程采用任务到达时序符合不同随机过程的训练集. 记录每次迭代过程每一幕的系统状态、动作、奖赏信息,并利用记录到的数据计算幕中 t 时刻的累积(折扣)奖赏 v_t. 根据上述数据,结合前文提到的梯度更新表达式,即可通过迭代完成训练. 训练过程伪代码如算法 2 所示.

算法 2 coflow 调度训练算法的伪代码

输入:刻画网络情景、任务情景的一系列图像
输出:特定网络情景、任务情景下可选动作的概率分布
1: procedure training_algorithm(images …)
2: for 每一次迭代过程
3: $\Delta\theta \leftarrow 0$

4:　for 每个训练集
5:　　for $i=1:N$
6:　　执行第 i 幕训练
7:　　记每个离散时刻的执行结果录$\{s_k^i,a_k^i,r_k^i\}\sim\pi_\theta,k=1,\cdots,L_i$
8:　　计算累积奖赏 $v_t^i=\sum_{s=t}^{L_i}\gamma^{s-t}r_s^i$
9:　　for $t=1:L$
10:　　计算、截取 v_t 的返回值 $b_t=\frac{1}{N}\sum_{i=1}^{N}v_t^i$
11:　　for $i=1:N$
12:　　$\Delta\theta\leftarrow\Delta\theta+\alpha\nabla_\theta\log\pi_\theta(s_t^i,a_t^i)(v_t^i-b_t^i)$
13:　　end for
14:　　end for
15:　　end for
16:　end for
17:　$\theta\leftarrow\theta+\Delta\theta$
18:　end for
19:end procedure

上述算法是 REINFORCE 算法的一种变体. 采用 REINFORCE 算法估计梯度时,存在方差较大、算法收敛慢的弊端. 为克服这种弊端,一般采用回报基线法,截取 v_t 的返回值. 回报基线法形式多种多样,本文采取的办法是对相同训练集下各幕相同时刻的返回值求取平均数[21],如算法 2 第 10 行所示.

3.6　两个补充机制

DeepCS 是一个实时的 coflow 调度系统,它具有如下的重要特性:(1)工作保持性,即当且仅当网络中没有待传输的流量时,才允许网络资源处于空闲状态;(2)无饥饿性,不允许某个 coflow 处于无限长的等待状态.

针对工作保持性,DeepCS 在深度增强学习算法的基础上,再提出反向填充机制(Back Filling Mechanism),目的是充分利用边缘链路上的剩余带宽传输流量,进一步降低 coflow 平均完成时间. 反向填充机制主要传输尚未得到调度的 coflow 集合中的流,以减少该 coflow 的数据量,降低 coflow 的平均完成时间,其伪码如算法 3 所示.

算法 3　反向填充机制的伪代码(情景生成算法)

输入:DCN 中所有边缘链路剩余带宽 Rem(P),尚未传输的 coflow 集合 Ω_R
输出:分配给若干 coflow 的带宽 r_{ij}^k
1:Procedure Back _ Filling(coflows Ω_R,BandwidthRem(P))
2:　for DCN 交换网络的每一个端口 P_i
3:　　while Rem(P_i^I) >0
4:　　for 所有 $C_k\in\Omega_R,P_j^E$
5:　　$\delta=\min(\text{Rem}(P_i^I),\text{Rem}(P_j^E))$
6:　　$r_{ij}^k=\delta$
7:　　更新 Rem(P)
8:　　end for
9:　　end while
10:　end for
11:　return r_{ij}^k
12:end procedure

算法 3 中第 5、6 行根据“木桶原理”将一条边缘链路的全部剩余带宽分配给了尚未启动的某条 coflow,保证了系统的工作保持性.

针对无饥饿性,借鉴文献[22,23],DeepCS 引入有限复用机制(limited multiplexing mechanism),起始时刻,在每条边缘链路上预留少量带宽,用以传输长时间得不到调度的 coflow,避免这些 coflow 一直处于等待状态. 引入复用因子 $\varepsilon\in(0,1]$,记端口 P_i 处的预留带宽为 $\varepsilon*B_i$,则正常参与调度的边缘链路带宽为$(1-\varepsilon)*B_i$,后文将通过实验讨论参数 ε 对 DeepCS 性能的影响.

结合深度增强学习算法、反向填充机制、有限复用机制,这里给出 DeepCS 的整体工作流程,如算法 4 所示. 其中 9~13 行为有限复用机制的伪代码,判断当潜集合中有无等待时间超出预设值 Γ 的 coflow,若存在,则执行有限复用机制,利用预留带宽处理该 coflow 的传输.

算法 4　DeepCS 系统工作流程

1: Procedure Main( )
2:　初始化 Rem(P_i) = $\{B_i\}$,$W_k=[w_{ij}^k]=\{0\}$
3:　执行训练过程 training _ algorithm(images …)
4:　while $\Omega\neq\varnothing$
5:　　生成情景 Time _ Bandwidth _ Allocation(Ω,Rem(P))
6:　　在线执行深度增强学习系统,选取 coflow C_{k*} 进行调度
7:　　$\Omega\leftarrow\Omega\backslash\{C_{k*}\}$
8:　　执行反向填充机制 Back _ Filling(Ω,Rem(P))
9:　　for Ω 中的每一个 coflow C_k
10:　　if C_k. waitingtime( ) >Γ
11:　　执行有限复用机制
12:　　end if
13:　　end for
14:　　更新 Ω、Rem(P_i)
15:　end while
16: end procedure

4　性能评估

4.1　系统结构

DeepCS 系统包含两个部分:一是集中式的调度器,二是分布式的端系统执行组件. 系统总体架构如图 4 所示. 鉴于拥塞仅发生在边缘链路,DeepCS 不在网络内部的交换机中部署任何功能模块. 端系统执行组件是运行于各物理服务器内的一个守护程序,主要负责发送 coflow 信息、链路资源占用信息到中心调度器. 发送端

的 coflow 数据准备好后,守护程序向 coflow 信息收集模块报告 coflow 相关信息;待接收端准备好后,中心调度器根据当前网络情景和任务情景,采用深度增强学习算法做出最佳调度策略,确定 coflow 传输起始时间、发送速率,将结果发放到端系统执行组件. 端系统执行组件负责在中心调度器确定的起始时间、发送速率下启动 coflow 的流传输过程.

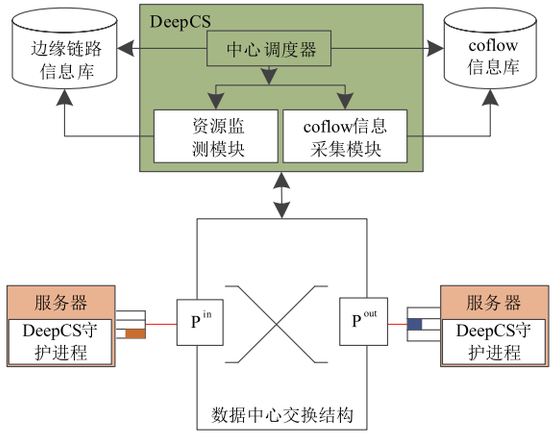


图4 DeepCS系统架构

4.2 仿真环境搭建

通过在网络仿真器 ns3 上进行包级仿真,测试 DeepCS 性能. 实验中用到的流量数据均是通过观测产业界数据中心网络的 coflow 流量模式而生成的. 从宽度(一个 coflow 包含的流数量)、容量(一个 coflow 包含的数据量)两个维度刻画 coflow. 实验时,网络中 coflow 数量变化范围为 10 到 100,宽度变化范围为 1 到 50,且 60% 的 coflow 宽度小于 16. 进入交换网络的 coflow 数量服从到达速率 $\lambda \in [0.2, 0.8]$ 的泊松分布,借此模拟流量的动态变化特性. 32 台服务器通过非阻塞交换网络连接,接入链路带宽为 1Gbps. 流量发送端和接收端以轮询方式均匀分布在边缘链路,其数量依据不同的 coflow 而变化,但需保证在大约 80% 的 coflow 中,发送端多于接收端.

系统仿真时采用的对比算法分别为:

(1)基线算法——TCP. 网络中所有的流利用 TCP 协议公平竞争带宽,以实现 max-min 公平.

(2)pFabric. 这是一种最优的面向单个流的调度算法,在无阻塞交换网络下,本实验采用 pFabric 的理想变体作为对比算法之一.

(3)Varys. 这是一种 coflow 级基于优先级的调度策略,采用的核心方法是最小瓶颈优先策略,是目前性能最好的 coflow 级调度算法.

在网络仿真器 ns3 上对比了 DeepCS 和 TCP、pFabric、Varys 等调度算法的平均流完成时间随网络负载和复用因子的变化情况及分析. 这里,将方法 1 相对方法 2 性能的优化定义为 $(CCT_2 - CCT_1)/CCT_2$,其中 CCT_1、CCT_2 分别表示方法 1 和方法 2 下 coflow 的平均完成时间. 后文中的性能均是较基线算法获得的性能的提升量.

4.3 算法性能对比

(1)不同网络负载下的 coflow 平均完成时间

将复用因子固定为 0.2,coflow 到达速率 λ 依次取 0.2,0.4,0.6 和 0.8,以模拟不同网络负载. 在网络仿真器 ns3 上对比了 DeepCS 和 TCP、pFabric、Varys 等调度算法的 coflow 平均完成时间随到达速率 λ 的变化情况. 各算法的 coflow 平均完成时间如图 5 所示.

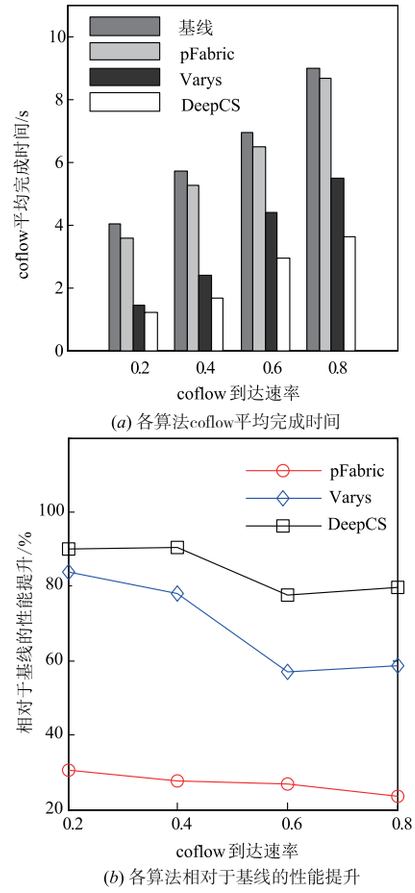


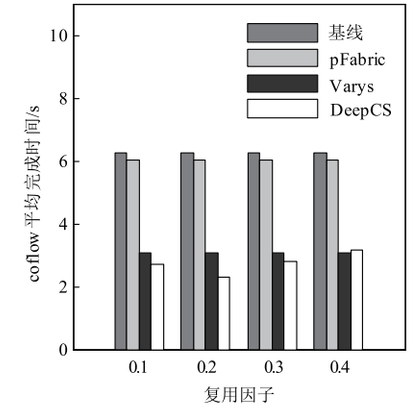
图5 不同网络负载下的平均流完成时间及性能提升对比

从图 5 可以可以得到如下结论:①随 coflow 到达速率的增加,各算法的 coflow 平均完成时间均增加,原因在于到达速率增加使得网络中的 coflow 数量持续、加速增加,调度算法的工作压力加大、流等待时间增大,引来 coflow 平均完成时间的增加;②在不同的 coflow 到达速率下,DeepCS 的 coflow 平均完成时间均优于其他调度算法,这是因为 DeepCS 在上线应用前,经历了在各种流量模式下的不断的学习过程,通过学习掌握了针对不同流量模式下减少 coflow 平均完成时间的最优调度

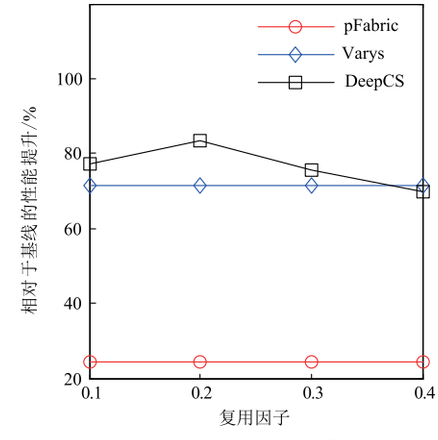
策略;③当 coflow 到达速率较低时,DeepCS 与 Varys 性能相当,优于 pFabric,随 coflow 到达速率的不断增加,DeepCS 性能的提升较 Varys 和 pFabric 更为明显. coflow 到达速率在 0.4 ~ 0.6 时,DeepCS 的性能高出 Varys 约 50%. 这说明网络负载较大时,pFabric 和 Varys 的性能有更大的提升空间.

(2)不同复用因子下的 coflow 平均完成时间

在 DeepCS 中,将 coflow 的到达速率固定为 0.5. 在网络仿真器 ns3 上对比了 DeepCS 和 TCP、pFabric、Varys 等调度算法的 coflow 平均完成时间随复用因子的变化情况. 各算法的 coflow 平均完成时间见图 6 所示.



(*a*) 各算法coflow平均完成时间



(*b*) 各算法相对于基线的性能提升

图6　不同复用因子下的coflow平均完成时间及性能提升对比

从图6 可以看出:①由于 pFabric 和 Varys 都没有使用有限复用机制,因此它们的 coflow 平均完成时间不受复用因子 ε 的影响,其图像为直线;②ε 的取值直接影响到 DeepCS 的系统性能,即 coflow 平均完成时间,在特定区间取值时,DeepCS 的性能优于 Varys,超出该区间,DeepCS 性能下降,至少劣于 Varys,图中 $\varepsilon = 0.4$ 时就出现了 DeepCS 不及 Varys 的情况;③ε 的取值对 DeepCS 性能的影响既不是线性的,也不是单调的,而是呈现先正向提高、再反向降低的特征. 此处可以得到的启发是,可能存在最优的 ε 值,使得系统性能最好,这需要反复调试才能得到. 实际使用时,可通过有限次调试,逐步缩小最优 ε 的取值范围,加快搜索进程.

5　结论

为降低数据中心网络中语义相关流的平均完成时间,本文从人工智能领域深度增强学习方向的最新研究进展中受到启发,提出了一种的新的语义相关流调度机制 DeepCS. DeepCS 的突出特性在于它将带宽约束的语义相关流调度问题转化为连续的学习过程,通过学习以往策略实现最佳调度. 进一步给出了两个补充机制:反向填充和有限复用机制,保证了系统的工作保持性和无饥饿性. 仿真结果表明 DeepCS 较其他调度机制具有性能上的显著优势.

参考文献

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

[2] ZAHARIA M, CHOWDHURY M, FRANKLIN M J, et al. Spark: cluster computing with working sets[J]. HotCloud, 2010, 10(10 – 10): 95.

[3] ISARD M, BUDIU M, YU Y, et al. Dryad: distributed data-parallel programs from sequential building blocks[A]. Proceedings of ACM EuroSys[C]. Lisboa: ACM, 2007. 59 – 72.

[4] CHOWDHURY M, ZAHARIA M, MA J, et al. Managing data transfers in computer clusters with orchestra[J]. ACM Special Interest Group on Data Communication, 2011, 41(4): 98 – 109.

[5] CHOWDHURY M, STOICA I. Coflow: A networking abstraction for cluster applications[A]. Proceedings of ACM HotNets[C]. Redmond: ACM, 2012. 31 – 36.

[6] SILVER D, HUANG A, MADDISON C, et al. Mastering the game of Go with deep neural networks and tree search [J]. Nature, 2016, 529(7587): 484 – 489.

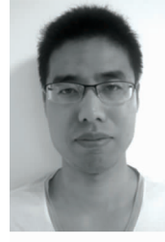
[7] 赵冬斌,邵坤,朱圆恒,等. 深度强化学习综述:兼论计算机围棋的发展[J]. 控制理论与应用, 2016, 33(6): 701 – 717.
ZHAO Dong-bin, SHAO Kun, ZHU Yuan-heng, et al. Review of deep reinforcement learning and discussions on the development of computer Go[J]. Control Theory & Applications, 2016, 33(6): 701 – 717. (in Chinese)

[8] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529 – 533.

[9] MAO H, ALIZADEH M, MENACHE I, et al. Resource

management with deep reinforcement learning[A]. Proceedings of the ACM HotNets[C]. Atlanta: ACM, 2016. 50 – 56.
[10] WILSON C, BALLANI H, KARAGIANNIS T, et al. Better never than late: meeting deadlines in datacenter networks[J]. ACM Special Interest Group on Data Communication, 2011, 41(4): 50 – 61.
[11] HONG C, CAESAR M, GODFREY P B, et al. Finishing flows quickly with preemptive scheduling[J]. ACM Special Interest Group on Data Communication, 2012, 42(4): 127 – 138.
[12] ALIZADEH M, YANG S, SHARIF M, et al. Pfabric: Minimal near-optimal datacenter transport[A]. ACM SIGCOMM Computer Communication Review[C]. Hong Kong: ACM, 2013. 435 – 446.
[13] CHOWDHURY M, ZHONG Y, STOICA I. Efficient coflow scheduling with varys[A]. Proceedings of ACM SIGCOMM[C]. Chicago: ACM, 2014. 443 – 454.
[14] DOGAR F R, KARAGIANNIS T, BALLANI H, et al. Decentralized task-aware scheduling for data center networks[A]. Proceedings of the ACM SIGCOMM Computer Communication Review[C]. Chicago: ACM, 2014. 431 – 442.
[15] ZHAO Y, CHEN K, BAI W, et al. Rapier: Integrating routing and scheduling for coflow-aware data center networks[A]. Proceedings of IEEE INFOCOM[C]. Hong Kong: IEEE, 2015. 424 – 432.
[16] MC KEOWN N, ANDERSON T, BALAKRISHNAN H, et al. OpenFlow: enabling innovation in campus networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69 – 74.
[17] JEYAKUMAR V, ALIZADEH M, MAZIèRES D, et al. EyeQ: Practical network performance isolation at the edge[A]. Proceedings of NSDI[C]. Lombard: USENIX, 2013. 297 – 312.
[18] SUTTON R S, BARTO A G. Reinforcement Learning: An Introduction[M]. Cambridge: MIT Press, 1998.
[19] HAGAN M T, DEMUTH H B, BEALE M H, et al. Neural Network Design[M]. USA: PWS Publishing Company Boston, 1996.
[20] SUTTON R S, MCALLESTER D A, SINGH S P, et al. Policy gradient methods for reinforcement learning with function approximation[A]. Proceedings of NIPS[C]. Denver: NIPS, 2000. 1057 – 1063.
[21] SCHULMAN J, LEVINE S, ABBEEL P, et al. Trust region policy optimization[A]. Proceedings of ICML[C]. Lille: IMLS, 2015. 1889 – 1897.
[22] JIANG J, MA S, LI B, et al. Adia: achieving high link utilization with coflow-aware scheduling in data center networks[J]. IEEE Transactions on Cloud Computing, 2016, 99(11): 1 – 10.
[23] MA S, JIANG J, LI B, et al. Chronos: meeting coflow deadlines in data center networks[A]. Proceedings of the IEEE International Conference on Communications (ICC)[C]. Kuala Lumpur: IEEE, 2016. 1 – 6.

作者简介



马 腾(通讯作者) 男,1987 年生,陕西西安人,国家数字交换系统工程技术研究中心博士研究生,主要研究方向为数据中心网络,网络体系结构.
E-mail: ndscmt@163.com

胡宇翔 男,1982 年生,河南周口人,国家数字交换系统工程技术研究中心副研究员,主要研究方向为网络体系结构.

张校辉 男,1979 年生,河南洛阳人,国家数字交换系统工程技术研究中心讲师,主要研究方向为网络体系结构、网络安全.