计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200300069

限时点到多点跨数据中心传输的多源树调度算法

庄 奕 杨家海
清华大学网络科学与网络空间研究院 北京 100084
北京信息科学与技术国家研究中心 北京 100084
(stzhuangyi@gmail.com)

摘 要 随着各种云应用的数据规模的增大,越来越多的云服务提供商开始关注跨数据中心的大数据块传输(bulk transfer)。跨数据中心的大数据块传输面临的主要挑战是:如何找到最佳的资源调度算法,在用户指定的时限内,用最少的传输资源将用户的数据传输到指定的地点。文中设计了一种有效的带传输时限(transfer deadlines)的、点到多点(Point-to-MultiPoint,P2MP)的跨数据中心数据传输调度算法 MSTB(Multi-Source Tree-Based algorithm)。在多源机制和多播转发树的帮助下,MSTB 表现得比现有的最优方法更好。仿真实验结果表明,MSTB 可以在保证低传输完成时间和低计算复杂度的同时,增加最高达 91%的传输请求接受数,增加最高达 54%的有效吞吐量。

关键词:数据中心;调度算法;带传输时限;点到多点;多源点
中图法分类号 TP302

Multi-source Tree-based Scheduling Algorithm for Deadline-aware P2MP Inter-datacenter Transfers

ZHUANG Yi and YANG Jia-hai
Institute for Network Sciences and Cyberspace,Tsinghua University,Beijing 100084,China
Beijing National Research Center for Information Science and Technology,Beijing 100084,China

Abstract With the growth of the data volume for cloud applications,more and more cloud service providers pay attention to inter-datacenter bulk transfer. The main challenge of inter-datacenter bulk transfer is how to find the best resource scheduling algorithm,which uses the least resources to transfer the user's data to the specified destinations before the specified deadline. This paper proposes MSTB(Multi-Source Tree-Based) algorithm,an effective scheduling solution for deadline-aware P2MP inter-datacenter transfers. With the help of multi-source mechanism and multicast forwarding tree,MSTB outperforms the state-of-the-art method. Simulation experiments show that MSTB can increase the number of transfer requests accepted by up to 91% and increase effective throughput by up to 54% with short transfer completion time and low computation complexity.

Keywords Datacenter,Scheduling algorithm,Deadline-aware,Point-to-multipoint,Multi-source

1 引言

当今世界,越来越多的信息技术公司需要为来自全球的客户提供自己的服务,这些公司通常需要为此建立多个跨地区的数据中心。这些数据中心通常分布在不同的地理位置,并通过广域网(Wide Area Network,WAN)相连。而部署在这些数据中心上的云应用,通常需要通过 WAN 网在不同的数据中心之间进行大数据块传输(bulk data transfer)[1-3]。典型的例子包括分布式数据库的同步[4-5]、高分辨率的视频流[6]和搜索引擎的索引同步[2]等。随着数据规模的增大,针对这些大数据块传输构建一个可靠且高效的调度系统变得至关重要。

在大数据块传输中,尽管数据大小从几十太字节(TB)到几十拍字节(PB)不等,但这种类型的数据传输通常对网络延时不太敏感[7-8]。大部分大数据块传输只要求在一个不太短的时间区间内完成即可,也就是说,它们通常会给出一个期望的传输时限[8-11]。微软的最新调查[10]显示,对于跨数据中心的大数据块传输,100%的消费者认为它们有指定传输时限的需求,并且其中还有 64%的人愿意为此支付额外的费用。由此可以看出,一个能有效保证传输时限的网络必定会受到绝大部分消费者的青睐。

在对大数据块传输的研究中,点到多点传输是一个很重要的主题[11-13]。Noormohammadpour 等[13]表明,大部分跨数据中心的大数据块传输是从一个数据中心出发,同时发往许多其他数据中心。换句话说,跨数据中心的大数据块传输通常有一个源点和多个接收点。比如,在百度,有 90%的此类传输以全 WAN 网中至少 60%的其他数据中心为接收点,而

到稿日期:2020-03-10 返修日期:2020-05-01 本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(61432009,61462009)
This work was supported by the National Natural Science Foundation of China(61432009,61462009).
通信作者:杨家海(yang@cernet.edu.cn)

70%的此类传输以全 WAN 网中至少 80%的其他数据中心为接收点[14]。例如,在分布式数据库的跨地数据复制中,有大量的 P2MP 传输,每一个 P2MP 传输都把大数据块从一个数据中心发送到多个位于不同地区的其他数据中心[5]。

本文主要关注带传输时限的、跨数据中心的、大数据块传输的调度策略。借助软件定义网络(Software Defined Networking,SDN)的概念,我们建立了如下基本网络模型。当一个请求到来时,SDN 的控制器以在线的方式迅速决定是否接受这个请求。对于所有可接受的请求,模型需要提供对应的时空资源分配方式,以保证数据传输的可行性和高效性。对应的数据块会在指定的传输时限之前,从一个数据中心被发往若干个其他数据中心。本文假定所有的传输拥有相同的优先级。

针对带传输时限的、点到多点的、跨数据中心的大数据块传输调度任务,文中提出了一种高效的调度算法——MSTB 算法。通过本文提出的多源选择机制和分级多播机制,MSTB 算法会执行一系列操作,包括准入控制、多源选择、数据切分与选择以及路由与带宽分配等。相关仿真实验结果证明了 MSTB 算法的优越性,与当前最优的算法相比,MSTB 算法最高可多接受 91%的请求,多实现 54%的有效吞吐量。同时,与其他现有算法相比,MSTB 的计算复杂度更低。

2 相关工作

对跨数据中心传输的研究有很多,但对于带传输时限的、点到多点的、跨数据中心的大数据块传输,目前仍然没有足够高效的解决方案。本节将梳理相关的研究进展。

Amoeba 算法[8]是最早关注传输时限的工作之一。它主要关注单点到单点的带传输时限的跨数据中心传输,通过切分时间片,将网络拓扑在时间轴上展开并建图,并通过最小费用流算法求出最优解。与之相关的 PGA 算法[9]是在 Amoeba 的基础上扩展了传输时限的定义,允许用户在硬性传输时限(hard deadline)之外提出软性传输时限(soft deadline)的需求,通过线性规划建模并转化为对偶问题求得最优解。

AGE 算法[11]是目前该任务的最优算法,其本质上是将点到多点的传输拆成多个点到点传输。AGE 算法通过引入灵活的源点选择机制实现了高请求接受率。它的核心思想在于,当接收点收到完整的数据拷贝之后,该接收点也可以作为新的源点给未收到数据的其他接收点发送数据;并且,越快完成传输,请求接受率越高。AGE 算法仍然是一种传统的基于隧道的单播方法。其问题在于即使有若干加速计算的方式,其算法复杂度仍然很高。

如果简单地把点到多点的传输拆成多个单点到单点的传输,虽然一定程度上简化了问题,但会造成某些资源浪费,因为单点到单点的传输间有许多重叠的路径,相当于同一份数据在同一条网络链路上被重复传输了多次。因此,研究者提出通过建立多播转发树(multicast forwarding trees)的方式来克服这个问题,例如 DCCast[12],QuickCast[13]和 MTree[15]。不同于保证传输时限的目标,DCCast 算法以最小化传输时间为优化目标,通过斯坦纳树的概念建立多播树。多播树的性质决定了它每一条边的传输速率都是相等的,因此其局限是会遭遇最弱链路问题(Weakest Link problem)[13]。也就是说,多播树的传输速率决定于传输速率最慢的那条边。为了缓解这个问题,一些研究提出把接收点分成多个不同的集合,每个集合有自己的多播树,以此来减小多播树的规模。QuickCast 算法是通过层次聚类的方式,将接收点分成两个集合,通过额外支付少量的资源成本来大幅缩短传输完成时间。MTree 算法则以某种启发式的深度优先搜索建立多播树,但设定每棵多播树的传输速率在其整个生命周期中是固定的。现有的基于多播转发树的方法虽然一定程度上节省了资源,但在建树算法方面仍然不够灵活,还是受限于最弱链路问题。

3 调度算法

3.1 主要思想

本文提出了一种新的带传输时限的、点到多点的、跨数据中心传输调度算法——MSTB 算法。MSTB 算法克服了现有相关工作的若干关键弱点,并尝试以一种更有效的方式解决这个任务。MSTB 算法的核心思想基于以下两个关键观察。

(1)一个接收点可以同时从多个不同的源点接收数据。从直觉上考虑,多源点很可能比单源点拥有更高的总传输速率。目前绝大部分工作都假设一个接收点只能从一个源点获取数据,包括引入"源点选择"概念的 AGE 算法。然而,我们可以将原来的待传输数据切成若干数据块,然后由不同的源点发送不同的数据块到同一个接收点,这在概念上和实现上都是合理的。基于这个观察,本文提出了多源选择机制。

(2)对基于多播树的算法,可以借助"源点选择"的概念进行设计,即除了最初的源点,每一个接收到完整数据的接收点也可以作为多播树的根节点。因此,我们可以在一开始建立较小的多播树,然后等待接收点完成传输并成为新的源点,这些新的源点可以往外继续建立新的小多播树。毫无疑问,这种分级建立多播树的策略更加灵活,会显著降低最弱链路问题的影响。基于这个观察,本文提出了分级多播机制。

对于每一个请求,MSTB 算法会基于多源选择机制为每个接收点进行源点选择,并基于分级多播机制计算路由与带宽分配方案。如果一个满足传输时限需求的可行解被找到,MSTB 算法会接受这个请求并返回对应的路由、带宽分配方案、数据切割与选择方案。

3.2 相关概念

对于带传输时限的、点到多点的跨数据中心传输调度,本节给出若干关键概念的定义。

3.2.1 网络模型

本文对跨数据中心的网络进行建模,表示为 $G=(V,E)$。其中 V 表示点的集合,点对应数据中心;而 E 表示边的集合,边对应连接数据中心的真实链路。与现有的若干工作类似[8-9,11-13],本文将时间切成一个个时间片,每个时间片有相等的时长。默认系统可以获得每条边在每个时间片上的剩余可用资源。

3.2.2 请求完成时间

模型将传输过程所涉及的时间长度(从源点开始发数据的瞬间,到所有接收点恰好都收完所有数据的瞬间)作为传输完成时间。考虑到大数据块的数据规模和时间片的设定,我们可以忽略在此过程中的排队时延、建立多播树的额外时间

开销、数据切分与合并的额外时间开销[8,11-13,16]。因此，本文把请求完成时间等同于传输完成时间。

在一次具体的多播传输中，可以把传输完成时间分成两种更细致的指标：传输平均完成时间和传输最迟完成时间。传输平均完成时间表示该多播请求中每一个接收点的传输完成时间的平均值；传输最迟完成时间表示该多播请求中最后一个完成传输的接收点的传输完成时间。这两个指标所描述的角度略有差异，但总体上有较强的相关关系。在实验中同时观察这两个指标有助于更全面地理解模型性能。

3.2.3　P2MP 请求

对于 P2MP 传输请求，即点到多点传输请求，我们可以用一个五元组来表示：$R=\{S,D,f,t_1,t_2\}$。其中 S,D,f,t_1,t_2 分别表示源点集、接收点集、待传输数据的大小、请求的到达时间和请求指定的传输时限。特别地，最初 S 只包含一个源点。

3.3　MSTB 算法的关键模块分析

3.3.1　分级多播

分级多播机制的设计理念在于既很好地利用了基于多播树的方法的优点，又不过多地受到最弱链路问题的影响。其主要思路是，先以较小的代价建立一条从某一源点到某一接收点的路径作为多播树的“主干”，以主干的最大可用带宽作为多播树的实际使用带宽，然后尝试从主干往外伸展出“枝叶”，在不降低传输速率的前提下寻找更多的接收点。一条“主干”和若干“枝叶”共同构成一棵多播树。

分级多播的具体运行机制描述如下：首先，算法从所有合法的源点（包括原始源点和收满数据的接收点）同时出发，进行一次宽度优先搜索（Breadth First Search，BFS），以找到离任意源点最近的一个接收点。一旦找到，则建立一条连接这个接收点和对应源点的主干路径$\tilde{t}$。接着，算法会计算这条主干路径的最大可用带宽 $bandwidth(\tilde{t})$，以决定这棵多播树的传输速率。然后，算法会执行另一个不同的 BFS，该过程称为 stretch。stretch 会从主干路径上的每个点同时出发，寻找更多其他的合法接收点。在 stretch 过程中，为了避免传输速率下降，算法会忽略所有可用带宽不够支持 $bandwidth(\tilde{t})$ 的链路；对于找到的每一个新的接收点，算法会建立一条从已有多播树到该接收点的枝叶路径。当 stretch 过程完成后，即可得到一棵传输速率为 $bandwidth(\tilde{t})$ 的，包含一条主干路径和若干枝叶路径的多播树，显然，这棵多播树覆盖了原接收点集的一个子集。

在这个机制下，我们可以直观地理解为，算法主要是给主干路径接收点传输数据，而在保持原有传输速度的前提下，顺便给一些枝叶路径上的接收点一份同样的数据。

3.3.2　多源选择

在 AGE 算法中，允许收满数据的接收点作为源点的源点选择机制被验证是可行的[11]。类似地，多源选择机制也可以用几乎同样简单的方式去实现。简单来说，基于分级多播机制，在同一个时间槽内，算法会同时建立从不同源点出发的多棵不同的多播树，这意味着某些接收点可能会同时出现在多棵多播树里作非根节点，也就是说一个接收者可能接受来自多个源的数据。

3.3.3　数据切割与选择

在多源点的机制下，MSTB 算法需要把原始待传输的数据切分成若干数据块，并分别决定每一小块从哪个源点经由哪条路径达到指定接收点。为了避免接收点从不同的多播树收到重复的数据，我们为上文描述的 stretch 过程增加一个 stretch 约束。

首先给出 stretch 约束的相关定义，如表 1 所列。

表 1　stretch 约束相关定义

Table 1　Notations for stretch constraint

Name	Description
T	一棵多播转发树内的点的集合
$\tilde{t}$	主干路径上点的集合
$\tilde{b}$	枝叶路径上点的集合
$\tilde{d}$	主干路径上的唯一接收点
$Get(d,path)$	点 d 在路径 $path$ 中作为接收点
$Data(d)$	点 d 累计收到的所有数据

对每个接收点 d，定义概念 $Sponsor(d)$ 表示接收点 d 的“赞助商”：

$$\forall T\forall d(Get(d,\tilde{b}),\tilde{b}\subset T),Sponsor(d)=\tilde{d}(Get(\tilde{d},\tilde{t}),\tilde{t}\subset T) \tag{1}$$

对于每一个请求，最初每个接收点 d 的 sponsor 值 $Sponsor(d)$ 都初始化为 null。当接收点 d 通过某棵多播树的枝叶路径接收到数据时，我们称 $Sponsor(d)=\tilde{d}$。“赞助商”即指算法在传输数据给$\tilde{d}$时，顺便赞助了一份同样的数据给 d。

基于上面的定义，可以给出 stretch 约束的具体内容：对于任意出现在枝叶路径上的接收点 d，它可以从某一多播树中获得数据，仅当 d 从来没有出现在任意多播树的主干路径里作为接收点，且 $Sponsor(d)$ 的值为 null 或该多播树的主干路径接收点$\tilde{d}$。形式化描述为：

$$\forall T\forall d,\exists\tilde{b}(Get(d,\tilde{b}),\tilde{b}\subset T)\rightarrow(\neg\exists\tilde{t}(Get(d,\tilde{t}),\tilde{t}\subset T',T'\neq T)\text{ and }(Sponsor(d)=\text{null or }Sponsor(d)=\tilde{d}(Get(\tilde{d},\tilde{t}),\tilde{t}\subset T))) \tag{2}$$

多播树机制决定了在特定某次传输中，枝叶路径接收点 d 可以收到和$\tilde{d}$一样的数据。在 stretch 约束下，当 $Sponsor(d)$ 为 null 时，表示接收点 d 还未收到数据；当 $Sponsor(d)$ 为$\tilde{d}$时，表示 d 之前收到过和$\tilde{d}$一样的数据且没有其他的数据来源。在这两种情况下，d 收到的数据总是$\tilde{d}$收到的数据的一个子集。由此，可以得到一个推论：

$$\forall T\forall d(Get(d,\tilde{b}),\tilde{b}\subset T),Data(d)\subset Data(\tilde{d})(Get(\tilde{d},\tilde{t}),\tilde{t}\subset T) \tag{3}$$

可以证明，我们只需采取简单的贪心策略，在每次传输时只考虑主干路径接收者$\tilde{d}$，只传输$\tilde{d}$缺少的数据就不会产生数据重复的情况。因为主干路径接收点收到的数据是贪心选择的，所以它们不会收到重复数据。根据上面的推论，枝叶路径接收点收到的数据永远是主干路径接收点收到的数据的子集，所以它们也不会收到重复的数据。因此，不会发生数据重复的情况。

综上，对于数据的分割与选择，MSTB 算法可以采取一种只考虑主干路径的简单贪心策略。而且，这种数据分割与选

择在实际实现上也是完全可行的，毕竟在分割之后，我们可以等效地认为原 P2MP 请求被切分成若干个更小的 P2MP 请求。显然，这个过程没有造成额外的存储开销，具体的实现方式也与基本的 P2MP 传输一致。

3.3.4 数据合并

由于 MSTB 算法的多源选择机制，每个接收点都可能从多个源点收到数据块。因此，需要一个对应的后处理来进行数据合并。形如已有的一些基于多路径的方法[8,11,16]，我们用相同的方式实现数据合并，并且此过程产生的额外开销可以忽略。

3.3.5 多播转发树的实现

根据文献[13]，作为一种主流的 SDN 协议，OpenFlow[17] 在版本 v1.1 后已经支持转发到多个出端口，并且，许多新型的 SDN 交换机也已经支持该特性。

3.4 MSTB 算法实现

算法 1 描述了 MSTB 算法的基本流程。该算法接收两个输入：请求 R 和图 G。请求 R 指明了一个传输请求的源点集、接收点集、待传输数据量、到达时间和传输时限；图 G 描述了网络的拓扑结构和可用剩余资源。

算法 1 MSTB 算法

输入：请求 R={S,D,f,t_1,t_2}，图 G=(V,E)
输出：是否接受当前请求
1. 初始化 Sponsor(v)=null，t=t_1；
2. while t<t_2 and D 非空 do
3. S′=S；
4. while S′非空 do
5. 从 S′中的点开始，往外 BFS；
6. if 找到合法接收点 $\tilde{d}$ do
7. 回溯得到主干路径 $\tilde{t}$ 和对应源点 $\tilde{s}$
8. S′= S′−$\tilde{s}$；
9. stretch($\tilde{t}$)；
10. 依据所建多播树，更新 S,D；
11. else if 本轮未找到任何合法接收点 do
12. break；
13. t = t + 1；
14. if D 非空 do
15. return reject；
16. return accept；
17. procedure stretch($\tilde{t}$)
18. 计算主干路径最大可用带宽 bandwidth($\tilde{t}$)；
19. 从主干路径 $\tilde{t}$ 上的点往外 BFS，跳过可用带宽小于 bandwidth($\tilde{t}$) 的边；
20. for 每一个 BFS 搜到的点 v do
21. if v 没被 stretch 访问过 and (Sponsor(v)的值为 null 或 $\tilde{d}$) do
22. 回溯添加到达 v 的枝叶路径.

在初始化所有点的 *Sponsor* 值后，算法依据时间 t 进行迭代，若在时限内能让接收点集为空，则表示可以接受该请求。在每个时间槽内，先将源点集 S 复制一份得到 S'，通过迭代 S' 中的每个点，让尽可能多的源点作为多播树的根进行数据传输。每个源点最多只能建立一棵多播树，允许部分源点在某个时间槽内不建立多播树（可能有部分多播树在 BFS 时由于资源不足，没有找到合法接收点）。在对 S' 迭代时，若未找到任何合法接收点，说明虽然还有空闲的源点，但已经无法建立新的多播树，则结束在当前时间槽 t 的搜索。在对 S' 中的点作 BFS 时，若找到一个合法接收点，则意味着即将建立一棵新的多播树，算法通过回溯得到主干路径 $\tilde{t}$ 和主干路径接收点 $\tilde{d}$，更新 S'，然后进行 stretch 操作。

在 stretch 中，算法会首先计算主干路径的最大可用带宽 $bandwidth(\tilde{t})$，然后从主干路径上的点开始，跳过可用带宽不足 $bandwidth(\tilde{t})$ 的边进行 BFS，从而寻找其他合法的接收点，并建立枝叶路径。

4 实验结果与分析

4.1 实验设定

4.1.1 网络拓扑

如表 2 所列，本文依据两个真实的著名的跨数据中心网络的拓扑做了充分的仿真实验。与已有的工作类似[8,11]，本文假定每条网络链路都有 160Gbps 的带宽资源，其中有 5%～15%的带宽预留给交互流量(interactive traffic)。对于每条链路，每个时间槽的交互流量消耗的带宽资源数量是随机生成的。不同于大数据块传输，交互流量在现实中是高度可预测的，因此我们可以假定在实验开始时已知每条链路每个时间槽的交互流量。

表 2 仿真实验所用拓扑

Table 2 Network topologies used in simulation

Name	Description
GScale[2]	Google's inter-datacenter WAN which contains 12 nodes and 19 links
Equinix[18]	A worldwide inter-datacenter WAN from Equinix. It has 41 nodes and 81 links

4.1.2 P2MP 传输流

本文的模拟请求生成模型与近期的一些工作[7-9,11]类似。在把时间切成时间片的设定下，单个时间片的长度是 5min，即单位时间为 5min。本文的每一个实验结果都是 10 次持续 150 个单位时间(相当于 12 多个小时)的实验的平均值。请求到达时间以符合到达率为 λ 的泊松分布随机生成。最大传输时间(即从请求到达时间 t_1 到传输时限 t_2 的时间间隔长度)以符合均值为半小时(6 个单位时间)的指数分布随机生成。随机生成传输数据量的方法为：一个服从均值为 40Gbps 的指数分布的速率乘以其已经生成的最大传输时间。对于每个请求，我们在网络拓扑中随机选择一个点作为源点，随机选择 $\gamma(|V|-1)$ 个其他点作为接收点，其中 γ 是一个百分数，表示接收者数量占全网络其他数据中心总数的百分比。

4.1.3 对比算法

本文将 MSTB 算法与如下 3 种有代表性的算法进行性能对比。

(1)Amoeba 算法[8]：作为当前带传输时限的单播传输调度的最优算法，Amoeba 可以把多播请求简单地拆成多个单播请求，并逐个求解。为了适配本文场景，假定当且仅当所有对应的单播请求都满足传输时限需求，Amoeba 算法才会接受这个多播请求。对于每一个单播传输，Amoeba 算法会通

过求解最小费用流优化问题，使传输尽快完成。

(2)AGE 算法[11]：AGE 算法是目前带传输时限的、点到多点的跨数据中心传输调度的最优算法。该算法提出了“源点选择”这一新颖的概念，让已收满数据的接收点也可以作为新的源点。与 MSTB 算法类似，AGE 算法会动态维护源点集和接收点集。在每次迭代中，算法会遍历每一个“源点-接收点”点对，通过最大流算法找到最早可以完成传输的下一个接收点。

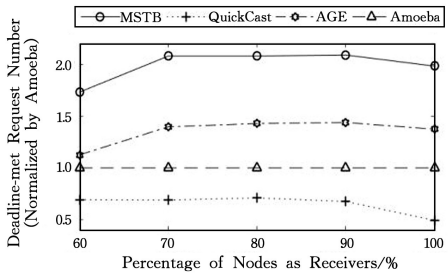
(3)QuickCast 算法[13]：作为目前基于多播树的点到多点传输调度的最优算法，QuickCast 算法的优化目标是最小化传输完成时间，而不考虑保证传输时限。它在处理请求时采用最大最小公平原则，而不是先来先服务原则。它尝试通过层次聚类的方式将接收点分成两个点集，然后使用 Greedy-FLAC 图论算法建立接近最优的斯坦纳树，作为实际的多播树。

4.2　实验结果

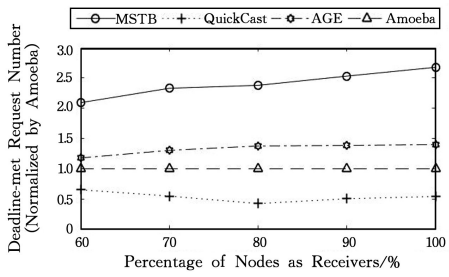
在本文的仿真实验中，通过在两个不同的网络拓扑上分别设定一系列不同的接收者数量，来观察表征算法性能的多个关键指标，包括请求接受数、有效吞吐、传输平均完成时间和传输最迟完成时间。目前大部分跨数据中心的大数据块传输都以多播的形式存在，且接收者的数目通常占 WAN 网所有节点的一半以上，例如在百度，有 90% 的请求发往全网至少 60% 的数据中心节点。因此，本文实验设定接收点数量系数 γ 在 60%～100% 的区间内变化。另外，请求到达率 λ 设为 2。

4.2.1　请求满足数

图 1 描述了请求接受数这一维度的性能对比，数据均用 Amoeba 算法的实验结果进行归一化，以更直接地描述算法间的性能对比。



(a)GScale



(b)Equinix

图 1　4 种调度算法在两个拓扑上的请求接受数对比

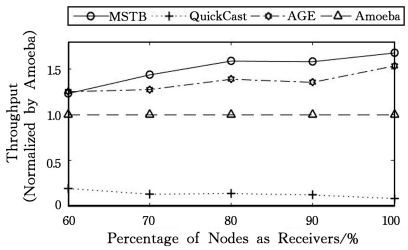
Fig. 1　Comparison of deadline-met request number for four approaches over two network topologies

从图 1 可以观察到，MSTB 算法在请求接受数方面稳定地保持最优。具体地，在 GScale 拓扑中，MSTB 算法最多比 AGE 算法多接受 48% 的请求；在 Equinix 拓扑中，MSTB 算法最多比 AGE 算法多接受 91% 的请求。

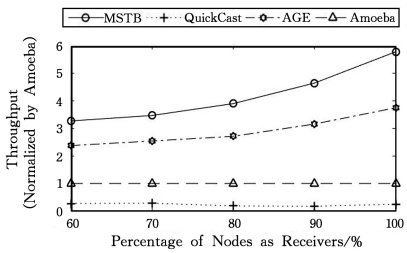
从图 1 可以观察到两个大致的趋势。一方面，在拓扑更为复杂的 Equinix 拓扑中，MSTB 算法的性能提升更加明显，说明在复杂的网络拓扑下，MSTB 算法有优秀的表现，甚至与普通算法有一定的差距，这验证了 MSTB 算法的鲁棒性。另一方面，在 Equinix 拓扑中，随着接收点数量的增加，MSTB 算法的性能提升越来越显著，这也说明了 MSTB 算法更能充分地挖掘网络拓扑的潜能。随着技术的发展，跨数据中心广域网的规模肯定会越来越大，因此 MSTB 算法所表现出的鲁棒性显得尤其重要。

4.2.2　有效吞吐量

有效吞吐量这一指标描述了各算法在时限内完成传输的数据量，图 2 给出了有效吞吐量的实验结果。



(a)GScale



(b)Equinix

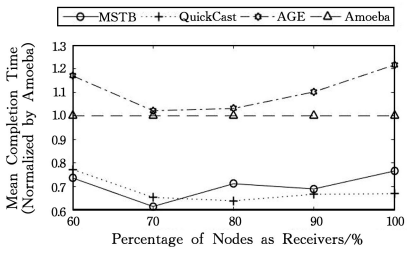
图 2　4 种调度算法在两个拓扑上的有效吞吐量对比

Fig. 2　Comparison of effective throughput for four approaches over two network topologies

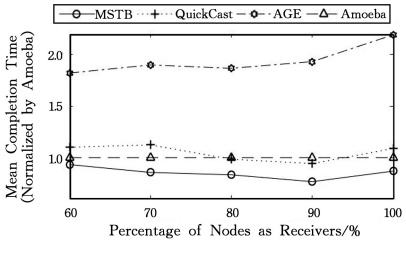
从图 2 可以看到有效吞吐量与请求接受数有几乎一致的数据趋势。具体地，在 GScale 拓扑中，MSTB 算法的有效吞吐量最多比 AGE 算法高 12%；在 Equinix 拓扑中，MSTB 算法的有效吞吐量最多比 AGE 算法高 54%。QuickCast 算法表现很差，可能是因为它仍遭受严重的最弱链路问题的困扰，并且其遵循最大最小公平原则的请求调度方式也会对效果产生一定影响。

4.2.3　完成时间

图 3 和图 4 给出了传输平均完成时间和传输最迟完成时间的实验结果，同样是用 Amoeba 算法的实验结果进行归一化。从图 3 和图 4 可以看出，在不同拓扑的不同设定下，MSTB 算法始终保持一个相对较低的、有竞争力的完成时间。值得指出的是，MSTB 算法在传输更多数据的情况下却实现了比 AGE 算法更少的完成时间，这充分证明了多源选择机制和分级多播机制的重要意义。



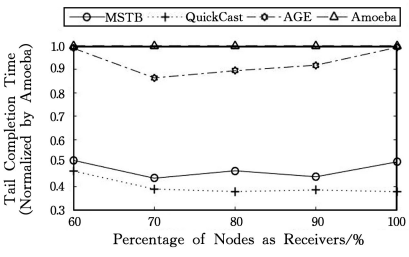
(a)GScale



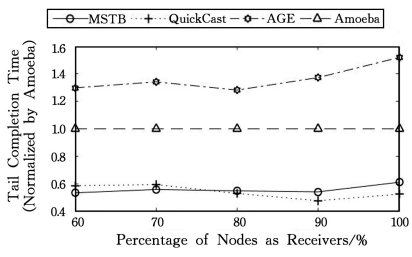
(b)Equinix

图 3 4 种调度算法在两个拓扑上的传输平均完成时间对比

Fig. 3 Comparison of mean completion time for four approaches over two network topologies



(a)GScale



(b)Equinix

图 4 4 种调度算法在两个拓扑上的传输最迟完成时间对比

Fig. 4 Comparison of tail completion time for four approaches over two network topologies

4.3 复杂度分析

本节分析并比较 MSTB 算法和 3 种对比算法的算法复杂度。在目前网络拓扑的节点规模之下，这 4 种算法都可以在较短的时间内得到计算结果。但随着技术的发展，跨数据中心广域网的节点规模一定会越来越大，计算复杂度的理论分析将有助于我们预判算法在未来更复杂的网络拓扑中的鲁棒性。

根据前文的定义可知一个广域网中的节点数目为 $O(|V|)$，由此可以推导出一个多播请求最多也只有 $O(|V|)$ 个接收点，在 MSTB 算法和 AGE 算法中，源点集和接收点集的规模也是 $O(|V|)$。

对于 MSTB 算法来说，在每个时间槽内，源点集的规模为 $O(|V|)$，意味着最多会建 $O(|V|)$ 次多播树。对于每棵多播树，建主干路径时维护了一个优先队列，复杂度为 $O(|V|\log|V|)$；建枝叶路径时，每条边最多访问 1 次，复杂度为 $O(|E|)$。因此，每棵多播树的建树复杂度为 $O(|V|\log|V|+|E|)$，总的算法复杂度为 $O(|V|(|V|\log|V|+|E|))$。

对于 Amoeba 算法来说，由于网络中所有数据中心的数量为 $O(|V|)$，因此一次多播请求最多可以拆分成 $O(|V|)$ 个单播请求。在求解每个单播请求时，Amoeba 算法使用了最小费用流算法，通常此类算法在这种数据规模下的复杂度为 $O((|V|^2)|E|)$（存在理论上最坏、复杂度更低的算法，但那些算法的常数部分非常大，在节点数目不多时总计算代价反而更大）[19]。因此，总的算法复杂度为 $O((|V|^3)|E|)$。

对于 AGE 算法来说，将接收点从接收点集移到源点集需要 $O(|V|)$ 次，其中每次都需要遍历所有可能的“源点-接收点”点对，遍历次数为 $O(|V|^2)$。在每次遍历中采用最大流算法，在这种数据规模下计算复杂度通常为 $O((|V|^2)|E|)$[19]。因此，总的计算复杂度为 $O((|V|^5)|E|)$。

QuickCast 算法包含了以下几个部分。首先，对所有点互相之间的距离进行预处理，复杂度为 $O(|V|^3)$；然后进行层次聚类，最多需要进行 $O(|V|)$ 次集合合并，每次集合合并需要计算集合两两之间的距离，而两个集合的距离是两个集合所有点对两两之间的距离之和，这一过程相当于遍历了所有点所有可能的两两组合，复杂度为 $O(|V|^2)$，则层次聚类过程的总复杂度为 $O(|V|^3)$；聚类完成后，采用 GreedyFLAC 算法建立斯坦纳树，复杂度为 $O((k^2)|V||E|)$，其中 k 是一个常数，表示近似系数；确定多播树后，根据最大最小公平原则进行带宽分配的代价为 $O((P^2)|V|)$，其中 P 是算法维护的仍在传输的多播树的数目。因此，总的计算复杂度为 $O((|V|^3)+(k^2)|V||E|+(P^2)|V|)$。

根据上面的分析可以看到，MSTB 在最坏情况下的算法复杂度在所有算法中属于比较低的。相比之下，作为目前的最优算法，AGE 算法的复杂度明显偏高，极大地受网络节点规模的影响。因此可以预见，当网络拓扑中节点数增加到上百个时，AGE 算法会处于几乎不可用的状态。Amoeba 算法和 QuickCast 算法在最坏情况下的算法复杂度与 MSTB 算法接近，但如 4.2 节的实验结果所示，其调度效果均远不如 MSTB 算法。

结束语 本文针对带传输时限的、点对多点的跨数据中心大数据块传输调度问题，设计了一种高效的调度算法 MSTB。其通过分段建立多播转发树的方式，克服了传统基于树的方法易受限于瓶颈链路。同时，通过灵活的多源点选择机制，配合相应的数据切割与选择策略，MSTB 算法能够更充分地挖掘网络资源的潜能。在保证传输完成时间和算法复杂度的前提下，MSTB 最高可多接受 91% 的请求，多实现 54% 的有效吞吐量。下一步工作可以研究是否能进一步提升 MSTB 算法的性能，并且围绕其多源选择机制和分级多播机制，探究对应的故障恢复策略。

参 考 文 献

[1] HONG C Y, KANDULA S, MAHAJAN R, et al. Achieving

high utilization with software-driven WAN[C]// Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM. 2013:15-26.

[2] JAIN S,KUMAR A,MANDAL S,et al. B4:Experience with a globally-deployed software defined WAN[J]. ACM SIGCOMM Computer Communication Review,2013,43(4):3-14.

[3] SUN G,XU Z,YU H,et al. Toward SLAs guaranteed scalable VDC provisioning in cloud data centers[J]. IEEE Access,2019,7:80219-80232.

[4] Multi-datacenter replication in cassandra[OL]. http://www.datastax.com/dev/ blog/multi-datacenter-replication.

[5] Azure sql database now supports powerful geo-replication features for all service tiers[OL]. https://goo.gl/aD5iRv.

[6] Mapping Netflix:Content Delivery Network Spans 233 Sites [OL]. http://datacenterfrontier.com/mapping-netflix-content-delivery-network.

[7] KANDULA S,MENACHE I,SCHWARTZ R,et al. Calendaring for wide area networks[C]// Proceedings of the 2014 ACM Conference on SIGCOMM. 2014:515-526.

[8] ZHANG H,CHEN K,BAI W,et al. Guaranteeing deadlines for inter-data center transfers[J]. IEEE/ACM Transactions on Networking,2016,25(1):579-595.

[9] LUO L,YU H,YE Z,et al. Online deadline-aware bulk transfer over inter-datacenter wans[C]// IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE, 2018: 630-638.

[10] JALAPARTI V,BLIZNETS I,KANDULA S,et al. Dynamic pricing and traffic engineering for timely inter-datacenter transfers[C]// Proceedings of the 2016 ACM SIGCOMM Conference. 2016:73-86.

[11] LUO L,KONG Y,NOORMOHAMMADPOUR M,et al. Deadline-Aware Fast One-to-Many Bulk Transfers over Inter-Datacenter Networks[J]. IEEE Transactions on Cloud Computing, 2019,doi:10.1109/TCC.2019.2935435.

[12] NOORMOHAMMADPOUR M,RAGHAVENDRA C S,RAO S,et al. Dccast:Efficient point to multipoint transfers across datacenters[C]// 9th Workshop on Hot Topics in Cloud Computing (HotCloud 17). 2017.

[13] NOORMOHAMMADPOUR M,RAGHAVENDRA C S,KANDULA S,et al. QuickCast:Fast and efficient inter-datacenter transfers using forwarding tree cohorts[C]// IEEE INFOCOM 2018-IEEE Conference on Computer Communications. IEEE, 2018:225-233.

[14] ZHANG Y,JIANG J,XU K,et al. BDS:a centralized near-optimal overlay network for inter-datacenter data replication[C]// Proceedings of the Thirteenth EuroSys Conference. 2018:1-14.

[15] JI S,LIU S,LI B. Deadline-aware scheduling and routing for inter-datacenter multicast transfers[C]// 2018 IEEE International Conference on Cloud Engineering (IC2E). IEEE,2018:124-133.

[16] WANG Y,SU S,LIU A X,et al. Multiple bulk data transfers scheduling among datacenters[J]. Computer Networks, 2014, 68:123-137.

[17] Tag:OpenFlow [OL]. https://www.opennetworking.org/tag/openflow.

[18] Equinix Cloud Exchange Fabric [OL]. https://www.equinix.com/interconnection-services/cloud-exchange-fabric.

[19] BANG-JENSEN J,GUTIN G Z. Digraphs:theory,algorithms and applications[M]. Springer Science & Business Media,2008.

ZHUANG Yi, born in 1995, master. His main research interests include cloud computing, and resource scheduling.

YANG Jia-hai, born in 1966, professor, Ph.D supervisor, is a member of China Computer Federation. His main research interests include network management, network measurement and security, cloud computing and network functions virtualization.