ISSN 1673-9418 CODEN JKYTA8
Journal of Frontiers of Computer Science and Technology
1673-9418/2000/00(00)-0000-00
doi: 10.3778/j.issn.1673-9418.

E-mail: fcst@vip.163.com
http://www.ceaj.org
Tel: +86-10-89056056

支持 SDN 的 Hadoop 中的时间最小化任务调度*

孙怀英[1,2]，虞慧群[1+]，范贵生[1]，陈丽琼[3]

1. 华东理工大学 计算机科学与工程,上海 200237
2. 上海市计算机软件评测重点实验室,上海 201112
3. 上海应用技术大学 计算机科学与信息工程系,上海 200235

Time minimized task scheduling in Hadoop with SDN*

SUN Huai-Ying[1,2],YU Hui-Qun[1+], FAN Gui-Sheng[1], CHEN Li-Qiong[3]

1. Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China
2. Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai 201112, China
3. Department of Computer Science and Information Engineering, Shanghai Institute of Technology, Shanghai 200235, China

+ Corresponding author: E-mail: yhq@ecust.edu.cn

SUN Huai-Ying, YU Hui-Qun, FAN Gui-Sheng, CHEN Li-Qiong. Time minimized task scheduling in Hadoop with SDN. Journal of Frontiers of Computer Science and Technology, 2000, 0(0): 1-000.

Abstract: Software defined network (SDN) is a network architecture which can separate out the network control functions in the infrastructures and centrally deploy them into a controller. In real-word Hadoop system, there is a NP-complete problem of minimizing the job completion time. This paper combines Hadoop with SDN, with the network control ability of SDN, the available residual bandwidth of the network can be gained as an significant parameter of task scheduling. According to this, a task scheduling algorithm (RBA)based on residual bandwidth is proposed, which can get the approximate optimal allocation schemes for tasks in a job, achieving the goal of

*The NSF of China under grants No. 61702334 and No. 61772200(国家自然科学基金), Shanghai Pujiang Talent Program under grants No. 17PJ1401900(上海市浦江人才计划). Shanghai Municipal Natural Science Foundation under Grants No.17ZR1406900 and 17ZR1429700(上海市自然科学基金). Educational Research Fund of ECUST under Grant No.ZH1726108(华东理工大学教育研究基金). The Collaborative Innovation Foundation of Shanghai Institute of Technology under Grant No. XTCX2016-20(上海应用技术大学协同创新基金项目)

Received 2000-00, Accepted 2000-00.

minimizing the job completion time. Finally, several simulation experiments are conducted to verify the performance of RBA in terms of job completion time, task data locality, and computation time. Experimental results show that RBA is generally better than the other 3 algorithms.

Key words: SDN; Hadoop; scheduling; bandwidth; time

摘　要: 软件定义网络(SDN)是一种能将基础设备的网络控制功能分离并集中地部署到控制器中的网络架构。实际的 Hadoop 系统中，存在一个最小化作业完成时间的 NP 完全问题。本文在 Hadoop 中引入 SDN，利用 SDN 的网络控制能力，将网络中的可用剩余带宽作为任务调度的重要参数，并因此提出任务调度算法 RBA。使用 RBA 可获得任务的近似最优分配方案，从而实现作业完成时间的最小化。最后，通过仿真实验验证 RBA 在作业完成时间、任务数据本地性及计算时间方面的性能。实验结果表明，总体上 RBA 较其他 3 个算法是更优的。

关键词: SDN; Hadoop;带宽;任务调度;时间

文献标志码：A　　**中图分类号**：****

1 引言

软件定义网络(SDN)是一种可以将基础设备的网络控制功能分离并集中地部署到控制器中的网络架构。它可以让企业有着前所未有的可编程性、自动化能力、网络控制能力。因此，SDN 为企业建立更具有创新性的、易管理的和高扩展性的数据中心创造了丰富的机会。SDN 中的领导技术是基于 OpenFlow 协议[1]，这已经是用于 SDN 的一个标准，且已经在各种网络和网络产品得到广泛使用[2][3]。OpenFlow 的网络监控和流量监控能力可以对大数据处理系统如 Hadoop 的性能提高提供重要的信息。因此，基于 OpenFlow 的 SDN 数据中心是当前一个重要的发展趋势。

大数据处理是此类数据中心的重要应用之一。大规模数据处理是如今常用的因特网应用如搜索引擎、在线地图服务、社交网络等的重要部分。现有的云计算系统如 MapReduce[4]、Hadoop[5]、Dryad[6]、Spark[7]等都是基于简单的并行处理模式，主要用于数据密集型的应用。其中，Hadoop 是 MapReduce 的一个开源实现，已经成为了雅虎、亚马逊、Facebook 等大型企业的通用计算系统[5]。但在实际的 Hadoop 系统中，存在一个影响整体系统性能的 NP 完全问题，解决的关键是要能找到任务的近似最优分配方案，从而最小化作业的完成时间[8]。

现有的最小化作业完成时间算法，多是从提高任务数据本地性(如果一个任务在执行时是从其对应的计算节点的本地磁盘读取输入数据，则该任务称为数据本地型任务，否则为数据远程型任务)方面着手，如 Hadoop 默认调度器(HDS)[9]和 Jin 等[10]提出的 BAR。它们存在的问题是：高度保证任务数据本地性，可能存在将任务持续分配给高负载节点的情况，或者没有充分利用网络中的可用带宽，造成任务的长时间等待，不能实现任务的近似最优分配，从而不能最小化作业的完成时间。

由于网络带宽是稀有资源，在充分保证数据的本地性情况下，充分利用可用的网络带宽资源，对于作业完成时间的降低是至关重要的。本文的贡献如下：

(1)将 SDN 引入 Hadoop 中，使用 SDN 的带宽控制能力，将网络中的可用剩余带宽作为任务调度的重要参数；

(2)定义作业完成时间问题的描述，根据网络中的剩余带宽提出任务调度算法 RBA，因此可获得任务的近似最优分配，实现作业完成时间的最小化；

(3)通过仿真实验，验证 RBA 在作业完成时间、任务数据本地性及计算时间方面的性能。

本文结构如下，第 2 部分介绍的是相关工作，第 3 部分介绍的是问题描述及算法 RBA，第 4 部分是仿真实验，第 5 部分是结论和未来工作。

2 相关工作

文献[11,12]指出任务调度对 Hadoop 的作业完

成效果有着重要影响。文献[8-10,13-17]是关于改善 Hadoop 中作业调度的性能的。文献[9]中的 Hadoop 默认调度器(HDS)主要是保证任务的数据本地性，不断的将任务分配给其输入数据所在的节点，因此在作业完成时间方面没有什么优化。Zaharia 等[14]提出了延时调度，以解决数据本地性和公平性的冲突问题。但是引入的延时可能会导致资源空闲。Tan 等[15]认为当前的调度器并非对 map 任务和 reduce 任务进行联合优化，这可能会产生作业饥饿及不利的数据本地性。因此，提出一个耦合调度器，将 map 和 reduce 任务结合起来。由于 Hadoop 假设所有的集群节点是为一个单独的用户服务的，所以耦合调度器无法在共享环境下保证高性能[16]。因此，Seo 等[16]提出预先取数据和预先 shuffle 的方案，但传输数据时的带宽占用时间是较难进行有效的减少的。

Fischer 等[17]研究了 Hadoop 中的任务分配，提出理想化的 Hadoop 模型，来评估任务分配的代价。而事实上，任务分配是一个 NP 完全问题，只能是找到近似最优的方案[8,11]。所以，Jin 等[10]提出 BAR 调度器，BAR 首先产生一个任务初始分配结果，接着，通过不断的调整初始分配方案将作业的完成时间降低到最小。在本文的第 4 部分的例子中看出，BAR 和 RBA 相比于 HDS 的作业完成时间是提早了 2.12s 和 4s，RBA 比 BAR 提早了 1.88s。所以，相对而言，RBA 是达到了近似最优的效果的。Wang 等[18]关注的是使用 SDN 为大数据应用配置物理拓扑和路由，但是没有关注作业调度的问题。文献[8]提出的 BASS 算法，同样将网络带宽作为任务分配的参数，但 BASS 在进行任务分配前，需要对网络中的最大带宽按时间进行分片，若所需的带宽分片不足以传输对应的输入数据，则不会将任务分配给远程节点。

3 问题描述

Hadoop 用于大规模数据的处理，其中的输入处理文件通常是分割成多个数据块存放于各个服务器节点上。为了有效的进行数据处理，Hadoop 的每个作业分成多个子任务，每个子任务会分配到一个服务器节点上进行数据块的处理。所以，当作业所有的子任务都完成了，该作业的状态才为完成。

Hadoop 集群网络中各个链路的带宽利用率是不尽相同的，对应的最大可用剩余带宽也不尽相同。因此，本文将 SDN 引入 Hadoop 中，通过 SDN 的网络控制能力可获得实时可用的网络带宽，从任务的数据副本所在的节点和目的节点(任务可能分配到的节点)之间的所有路径中，选择具有最大可用剩余带宽的路径作为传输数据的工作路径(路径的最大可用剩余带宽值等于路径上实时可用带宽值最小的链路对应的带宽值)。该路径的最大可用剩余带宽即为任务分配时的重要参数。RBA 在充分考虑任务数据的本地性的情况下，充分利用网络中的最大剩余可用带宽，实现任务的近似最优分配，从而达到最小化作业完成时间的目的。本文常用符号如表 1 所示。

表 1. 本文常用符号

Variables	***Meanings***
Jbr_i	作业请求 i
$jt_{i,j}$	Jbr_i 的第 j 个任务
jds_{ij}	任务 $jt_{i,j}$ 的输入数据分片大小
jwd_{ij}	任务 $jt_{i,j}$ 的指令数
snd_s	节点 s
Ewd_i	snd_s 中已有的工作量
Psd_i	snd_i 的处理速度
$pth_{src,de}$	snd_{src} 和 snd_{de} 之间的工作路径
$Bwh_{src,de}$	路径 $pth_{src,de}$ 的可用剩余带宽
$Tras_{ij}$	任务 $jt_{i,j}$ 的数据传输时间
$Cpt_{ij,s}$	任务 $jt_{i,j}$ 在 snd_s 上的计算时间
$Ext_{ij,s}$	任务 $jt_{i,j}$ 在 snd_s 上的执行时间
$AT_{ij,s}$	snd_s 能开始执行任务 $jt_{i,j}$ 的时间
$CT_{ij,s}$	任务 $jt_{i,j}$ 在 snd_s 的完成时间
CT_{Jbri}	Jbr_i 的完成时间

定义 1 引入 SDN 的 Hadoop 集群中的任务调度需求是一个三元组：$\Xi=(Jbr, snd, Gph)$，其中：

(1) Jbr 是要在时间[0，T]内完成的用户作业请求。作业请求 $Jbr_i=(jbt_i, T_i)$，$jbt_i=\{jt_{i,1}, jt_{i,2}, ...jt_{i,n}\}$为该作业的任务集合，任务 $jt_{i,j}=\{jds_{ij}, jwd_{ij}\}$，$jds_{ij}$ 为任务的输入数据分片大小(MB)；jwd_{ij} 为任务对应的指令数；

(2) snd 是执行任务的计算节点，节点 $snd_i=(Ewd_i, Psd_i)$，Ewd_i 为节点已有的工作量(指令数)，Psd_i 为节点的处理速度(MIPS)；

(3) $Gph=(V, Lnk)$为 Hadoop 集群网络拓扑，V 为拓扑中的顶点集合，表示集群中的所有计算节点和交换机节点，$Lnk=(lik_{ij}, mbd_{ij}, ar_{ij})$为顶点间的链

路集合，lnk_{ij} 为链路编号，下标 i，j 表示链路两端的顶点号，mbd_{ij} 为该链路的最大带宽，ar_{ij} 为该链路当前可用带宽百分比。

定义 2 数据分布是使用二分图 $G=((T\cup S), E)$ 表示，令 $m=|T|$ 且 $n=|S|$，其中 T 是所有待分配的任务集合，S 为计算节点的集合，$E\subset(T\times S)$ 是 T 和 S 之间的边的集合，边 $e(t,s)$ 表示任务 $t\in T$ 的输入数据是放置在节点 $s\in S$ 上，且 $S_{G.pre}(t)$ 是任务 t 所对应的输入数据所在的计算节点集合。假设 $|S_{G.pre}(t)|\geqslant 1$ 表示图 G 中任务顶点 t 的度至少为 1。

定义 3 分配策略是一个函数 $Aloc:T\rightarrow S$，将任务 $t\in T$ 分配给计算节点 $Aloc(t)\in S$。设 ϑ 为一个分配策略，则任务 t 分配到的计算节点记为 $\vartheta(t)$。在 ϑ 分配下，任务 t 是本地的，当且仅当 $\vartheta(t)$ 是在配置图 G 中有边 $e(t,\vartheta(t))$，否则 t 为远程的。设 $lonb_{\vartheta}$ 和 $renb_{\vartheta}$ 分别为作业中的本地任务数和远程任务数。如，任务 $jt_{i,j}$ 在 ϑ 分配下，分配到的计算节点为 $\vartheta(jt_{i,j})$，且该分配下节点的任务数据本地率 DR 为：

$$DR=lonb_{\vartheta}/(lonb_{\vartheta}+renb_{\vartheta}) \tag{1}$$

在分配方案 ϑ 下，计算节点 snd_s 可能会分配到多个任务，假设任务在一个计算节点上是按顺序执行的，计算节点 s 的负载为当前需要完成的工作量(指令数)，在分配方案 ϑ 下，计算节点的负载为：

$$Sjwd_s=Sjwd_s^{init}+Sjwd_s^{task}(\vartheta) \tag{2}$$

其中 $Sjwd_s^{init}=Ewd_s$ 为节点 snd_s 的初始负载。

节点 snd_s 处理当前分配方案 ϑ 所分配的任务集合 $task$ 对应的总的工作量为：

$$Sjwd_s^{task}(\vartheta)=\sum_{t\in task:\ \vartheta(t)=s} jwd_t \tag{3}$$

节点 snd_{src} 到节点 snd_{dek} 的工作路径为：

$$pth_{scr,\,de}=\{lnk_{scr,\,j},\ lnk_{f,\,e},...,\ lnk_{c,\,dek}\} \tag{4}$$

假设 $Tras_{ij}$ 记录的是 $jt_{i,j}$ 所需的输入数据从源节点 snd_{dasrc} 传输到目的节点 snd_{dek} 的时间，$Bwh_{dasrc,dek}$ 记录的是 snd_{dasrc} 到 snd_{dek} 的工作路径的可用剩余带宽，则 $Tras_{ij}$ 为：

$$Tras_{ij}=jds_{ij}/Bwh_{dasrc,\,dek} \tag{5}$$

任务 $jt_{i,j}$ 在 snd_{dek} 上的计算时间 $Cpt_{ij,dek}$ 为：

$$Cpt_{ij,\,dek}=jwd_{ij}/Psd_{dek} \tag{6}$$

任务 $jt_{i,j}$ 在 snd_{dek} 上总的执行时间 $Ext_{ij,dek}$ 为：

$$Ext_{ij,\,dek}=Cpt_{ij,\,dek}+Tras_{ij} \tag{7}$$

节点 snd_{dek} 能开始执行 $jt_{i,j}$ 的时间 $AT_{ij,dek}$ 为：

$$AT_{ij,\,dek}=Sjwd_{dek}/Psd_{dek} \tag{8}$$

$CT_{ij,dek}$ 记录的是 $jt_{i,j}$ 在 snd_{dek} 上的完成时间，对于一个任务 $jt_{i,j}$，目标函数(9)是用于在集群中找到可用节点 snd_q，$jt_{i,j}$ 在该节点上的完成时间最早。

$$snd_q=snd(argmin_qCT_{ij,\,q})\qquad 1\leqslant q\leqslant n \tag{9}$$

而从全局的角度看一个作业的完成情况，公式(10)是为了优化最慢完成的任务 $jt_{i,j'}$ 的完成时间，对应的最晚任务完成时间即为作业 Jbr_i 的完成时间：

$$\begin{aligned}&\min\{CT_{ij',k'}=\max CT_{ij,k}\}(1\leqslant j,j'\leqslant m,1\leqslant k,k'\leqslant n)\\&CT_{Jbn}=\max CT_{ij}(1\leqslant j\leqslant m)\end{aligned} \tag{10}$$

其中，CT_{Jbri} 为作业 Jbr_i 的完成时间，m 是作业 Jbr_i 的任务数，n 是 Hadoop 集群中的节点个数。

3.1 基于剩余带宽的 RBA 算法

基于 SDN 的 Hadoop 集群，可以根据 SDN 的网络控制能力，获得各个链路实时可用的带宽 Bwh_{rl}。当任务 $jt_{i,j}$ 对应的本地节点 $snd_{ij,loc}$ 最早的可用时间 $AT_{ij,loc}$ 比当前集群中除本地节点之外的可用节点 $snd_{ij,el}$ 最早的可用时间 $AT_{ij,el}$ 更晚时，则选择任务输入数据的所有本地节点到目的节点之间的路径中，可用剩余带宽($Bwh_{rl,el}$)最大的路径作为最终传输任务输入数据的工作路径。带宽 $Bwh_{rl,el}$ 是任务调度时的重要参数。若该带宽可以保证任务在 $snd_{ij,el}$ 上的完成时间 $CT_{ij,el}$ 小于在 $snd_{ij,loc}$ 上的完成时间 $CT_{ij,loc}$，则将任务分配给远程节点 $snd_{ij,el}$，否则分配给 $snd_{ij,loc}$。这样不仅充分考虑任务数据的本地性，同时充分利用网络中的最大可用剩余带宽，从全局的角度进行任务的分配，得到任务近似最优分配，并实现作业完成时间的最小化。

因为实际场景中，网络中每个链路的剩余可用带宽是不一定相同的，所以要计算使用 $snd_{ij,loc}$ 与 $snd_{ij,el}$ 间工作路径上的实时可用带宽传输对应输入数据所需的时间 $Tras_{ij,rl}$，调度器则将工作路径上的各个链路的带宽及 $Tras_{ij,rl}$ 通过控制器分配给对应的任务 $jt_{i,j}$，确保这条路径上的链路的带宽在任务 $jt_{i,j}$ 需要传输输入数据时是可用的。通过提供最大可用剩余带宽给任务进行数据移动，等移动结束后就将带宽收回，可以达到充分利用可用带宽的目的。RBA 算法如 Algorithm 1 所示。

3.2 案例

图 1 描述的是一个引入了 SDN 的 Hadoop 集

群，由 4 个节点组成，分别为 $node_1$、$node_2$、$node_3$、$node_4$（假设 4 个节点可用空闲时间分别为：2s、8s、19s、6s），4 个 *SDN-switch*，分别为 $SDN\text{-}switch_1$、$SDN\text{-}switch_2$、$SDN\text{-}switch_3$、$SDN\text{-}switch_4$，1 个 *SDN Controller*，1 个 *Scheduler*，其中有 7 个链路分别为 $link_1$、$link_2$、…、$link_7$，当前待执行的作业 Jbr_i，有 9 个任务 $jt_{i,j}$ (j=1,..,9)需要分配给节点执行，每个任务的输入数据块在两个不同的节点上有备份(方框里的圆圈表示对应任务的输入数据副本)。

Algorithm 1. 基于剩余带宽的 RBA 算法

Input:待执行的作业队列 JbQ，n 个节点，集群拓扑 Gph

Output:每个作业的任务分配方案

1:从队列中取出一个作业 Jbr_i，对应的任务数目为 m_i，计算出集群中所有节点的可用时间 AT

2:for (j = 1, 2, . . . , m_i) do

3:从 SDN controller 获得每个链路的剩余可用带宽 Bwh_{rl}

4:找出任务 $jt_{i,j}$ 所有的本地节点放入集合 snd_{loc}，并找出本地节点中的最早可用节点 $snd_{ij,loc}$

5:为任务 $jt_{i,j}$ 找到一个除本地节点外的最早时间 $AT_{ij,minw}$ 可用的节点 $snd_{ij,el}$

6:if ($AT_{ij,loc} \le AT_{ij,el}$) then 将任务 $jt_{i,j}$ 分配给节点 $snd_{ij,loc}$

7: else if($AT_{ij,loc}>AT_{ij,el}$)then

8:根据 Gph，选择可用剩余带宽($Bwh_{rl,el}$)最大的路径作为最终传输任务输入数据的工作路径，数据源节点为任务对应的一个本地节点 snd_{src}

9: 使用公式(2)-(8)计算 $CT_{ij,el}$ 及其可以确保 $CT_{ij,el}<CT_{ij,loc}$ 所需的带宽 $Bwh_{ij,el}$

10: if ($Bwh_{ij,el}<=Bwh_{rl,el}$) then

11: 将任务分配给远程节点 $snd_{ij,el}$

12:同时计算使用 snd_{src} 与 $snd_{ij,el}$ 间工作路径的实时可用带宽传输对应数据所需的时间 $Tras_{ij,rl}$，并将对应的带宽及 $Tras_{ij,rl}$ 分配给对应的任务 $jt_{i,j}$

13: else 将任务 $jt_{i,j}$ 分配给数据所在的本地节点 $snd_{ij,loc}$

14: end if

15: else if (没找到可用的本地节点) //snd_{loc} 不为空，但其中所有本地节点是该任务不可用的

16: then 将任务分配给远程节点 $snd_{ij,el}$

17：执行和第 8 行一样的工作

18: 计算该工作路径（$Bwh_{rl,el}$）传输对应数据所需的时间 $Tras_{ij,rl}$，并将工作路径上的各个链路的带宽及 $Tras_{ij,rl}$ 分配给对应的任务 $jt_{i,j}$

19:end if

20:end for

假设每个任务的输入数据块是 64MB，节点与交换机之间链路（$link_1$、$link_2$、$link_5$、$link_7$）的最大带宽为 100Mb/s，剩余交换机之间的链路的最大带宽为 500Mb/s ($link_3$、$link_4$、$link_6$)，图中链路上的百分数表示的是某时刻 T 对应的链路上的可用剩余带宽比例。假设 4 个节点的处理能力是一样的，每个任务的工作量大小是一样的，则可将各个任务在对应节点上的计算时间视为相同值 Cpt_{com}，为了便于比较计算结果，在此设为 Cpt_{com}=10.

根据 Algorithm 1 的描述，RBA 对作业 Jbr_i 的 9 个任务的分配结果如表 2 所示。以任务 $jt_{i,1}$ 为例，4 个节点对于 $jt_{i,1}$ 的初始可用空闲是时间分别为 $AT_{i1,1}=2s$，$AT_{i1,2}=8s$，$AT_{i1,3}=19s$，$AT_{i1,4}=6s$，而 $jt_{i,1}$ 的本地节点 $snd_{i1,loc}$ 为 $node_2$ 或 $node_3$，因为 $AT_{il,2}<AT_{il,3}$，所以，选择 $snd_{il,loc}=node_2$，而当前除 $node_2$ 和 $node_3$ 之外的最早可用节点为 $snd_{i1,el}= node_1$，且 $AT_{i1,1}<AT_{i1,loc}=AT_{i1,2}$，所以可以考虑在带宽允许的条件下将 $jt_{i,1}$ 分配给 $node_1$ 而非 $snd_{i1,loc}= node_2$。

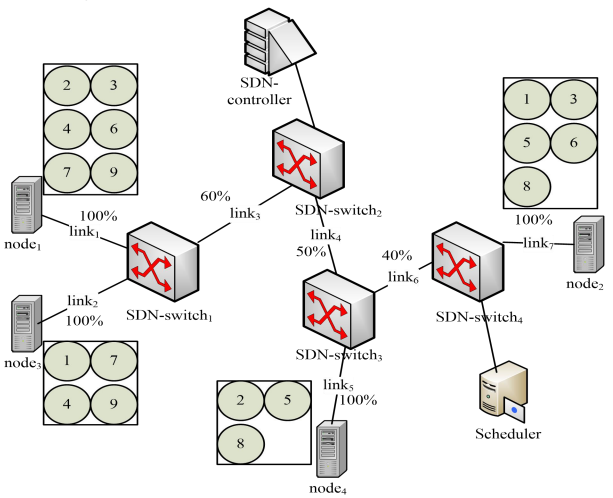

图 1.引入 SDN 的 Hadoop 集群拓扑

通过计算可得 $jt_{i,1}$ 在 $snd_{il,loc}$ 的完成时间为 $CT_{i1,loc}=CT_{i1,2}=8+10=18s$，而在 $snd_{i1,el}$ 上的完成时间为 $CT_{i1,el}=CT_{i1,1}=2+10+64B/Tras_{i1,rl}$，只要 $CT_{i1,loc} > CT_{i1,el}$，即可计算出对所需的最小传输带宽 $Tras_{i1,rl}=85.3Mb/s$，而此时，无论是从 $node_2$ 还是 $node_3$ 将 $jt_{i,1}$ 的输入数据传输给 $node_1$，对应工作路径上的实时最小可用剩余带宽为 100%*100Mb/s =100Mb/s>$Tras_{i1,rl}$=85.3Mb/s，所以，可以将 $jr_{i,1}$ 分配给 $node_1$，且选择从 $node_3$ 复制 $jt_{i,1}$ 所需的输入数据到 $node_1$，以减少链路的占用。

同时，根据链路的实时可用带宽计算传输输入数据块所需的真实时间 $Tras_{i1,31}$，并将 $Tras_{i1,31}$ 分配

给对应的任务 $jt_{i,1}$，通过 SDN-controller 将 $Tras_{i1,31}$ 分配给对应路径上的所有链路，以保证 $jt_{i,1}$ 在传输数据时，对应链路上的网络带宽是可用的。对于剩下的任务，RBA 以同样的方式进行分配，由表 2 的结果可知，$jt_{i,9}$ 为最慢的任务，所以其对应的完成时间即为作业 Jbr_i 的完成时间，为 37.12s。

表 2. RBA 的执行结果(*Asi-to* 意为任务分配给节点)

task	*Asi-to*	*AT (s)*	*Tras(s)*	*Cpt(s)*	*Ct(s)*
$jt_{i,1}$	$node_1$	2	5.12	10	17.12
$jt_{i,2}$	$node_4$	6	0	10	16
$jt_{i,3}$	$node_2$	8	0	10	18
$jt_{i,4}$	$node_1$	17.12	0	10	27.12
$jt_{i,5}$	$node_4$	16	0	10	26
$jt_{i,6}$	$node_2$	18	0	10	28
$jt_{i,7}$	$node_3$	19	0	10	29
$jt_{i,8}$	$node_4$	26	0	10	36
$jt_{i,9}$	$node_1$	27.12	0	10	**37.12**

HDS 是先找到任务的本地节点，并把任务分配给最早可用的本地节点。若找不到任务对应的数据本地节点，则将任务随机分配给一个节点。以 $jt_{i,1}$ 为例，对应的输入数据副本存放在 $node_2$ 和 $node_3$ 上，$AT_{i1,2}<AT_{i1,3}$，所以，将 $jt_{i,1}$ 分配给 $node_2$，剩下的任务同样是以此种方式分配。当分配 $jt_{i,9}$ 时，$node_4$ 在 26s 就可用，所以，HDS 将任务 $jt_{i,9}$ 作为非本地任务分配给 $node_4$，所以传输时间为 5.12s，任务 $jt_{i,9}$ 的完成时间为 41.12s，即作业 Jbr_i 的完成时间为 41.12s，相比于 RBA ，慢了 4s，具体的分配结果如表 3 所示。

表 3. HDS 的执行结果

task	*Asi-to*	*AT (s)*	*Tras(s)*	*Cpt(s)*	*Ct(s)*
$jt_{i,1}$	$node_2$	8	0	10	18
$jt_{i,2}$	$node_1$	2	0	10	12
$jt_{i,3}$	$node_1$	12	0	10	22
$jt_{i,4}$	$node_3$	19	0	10	29
$jt_{i,5}$	$node_4$	6	0	10	16
$jt_{i,6}$	$node_2$	18	0	10	28
$jt_{i,7}$	$node_1$	22	0	10	32
$jt_{i,8}$	$node_4$	16	0	10	26
$jt_{i,9}$	$node_4$	26	5.12	10	**41.12**

BAR 分成了两个阶段，第一个阶段，使用和 HDS 类似的方式，对任务进行分配，所以得到的是和 HDS 一样的初步分配结果[8]；第二个阶段，会找到第一阶段中完成时间最晚的那个任务 $jt_{i,k}$，查看是否可能将该任务 $jt_{i,k}$ 放在哪个远程节点 $node_{rem}$ 上执行，使得 $jt_{i,k}$ 完成的时间更早。如果找到这样的远程节点 $node_{rem}$，则将 $jt_{i,k}$ 分配给该 $node_{rem}$，重复此步骤，直到找不到这样的远程节点。如本例子中，由第一阶段的分配结果可知，在 $node_4$ 上执行的任务 $jt_{i,9}$ 为最晚完成的任务，$CT_{i9,4}$ 为 41.12s，经过查找，可知，$jt_{i,9}$ 可在 $node_3$ 完成的时间为 39s，比在 $node_4$ 完成的更早，进一步分析，只有 $node_3$ 能让 $jt_{i,9}$ 完成的更早，所以，最后将 $jt_{i,9}$ 分配给 $node_3$，即作业的完成时间为 39s，具体分配结果如表 4 所示。

表 4 BAR 的执行结果

task	*Asi-to*	*AT (s)*	*Tras(s)*	*Cpt(s)*	*Ct(s)*
$jt_{i,1}$	$node_2$	8	0	10	18
$jt_{i,2}$	$node_1$	2	0	10	12
$jt_{i,3}$	$node_1$	12	0	10	22
$jt_{i,4}$	$node_3$	19	0	10	29
$jt_{i,5}$	$node_4$	6	0	10	16
$jt_{i,6}$	$node_2$	18	0	10	28
$jt_{i,7}$	$node_1$	22	0	10	32
$jt_{i,8}$	$node_4$	16	0	10	26
$jt_{i,9}$	$node_3$	29	0	10	**39**

由以上例子可知，相比于 HDS、BAR，RBA 在缩小作业完成时间方面是更占优势的。接下来，通过仿真实验进一步验证所提方法的有效性。

4 仿真实验

在此部分，通过仿真实验，验证 RBA 和 HDS、BAR、BASS 在作业完成时间、数据本地性、计算时间方面的性能差异。

4.1 实验环境

在仿真实验中，数据分布是由[9]中的方法生成的。假设所有的计算节点是在同一个机架上的。数据块的副本设置为 3，这是 Hadoop 中默认设置[9]。实际中，计算节点的初始负载可以通过 Zaharia's 的方法[13]获得。而在本文的实验中，设置初始负载为[0,*W*]的随机值[10]，*W* 为集群的负载，*W* 较小代表集群中大部分的计算节点将处于空闲状态，*W* 较大表示集群中有部分计算节点将暂时不可用。假设集群中计算节点的处理速度都是 *Psd*，任务的工作量为(0,*jwd*]中的一个随机值。

在服务器节点上部署 Open vSwitch(OVS)[18]即可得到 SDN 交换机。在服务器节点上部署 NOX 即可获得 SDN 控制器，SDN 控制器内部有整个集群的拓扑结构，调用 SDN 控制器的 API 即可获取

网络各个链路的实时可用带宽信息。本文仿真实验中，设置交换机个数为 30，随机生成一个满足要求的集群拓扑结构。每个链路的实时可用的最大剩余带宽为(0,*E*]中的一个随机值，不同的 *E* 值表示不同的网络状态。*E* 为 200Mb/s 表示网络状态较差，*E* 为 2000Mb/s 则表示网络状态较好。

4.2 实验和结果分析

4.2.1 作业完成时间的评估

实验 1：为了验证网络状态的影响，通过调整网络链路的最大带宽为 200Mb/s、400Mb/s、600Mb/s、800Mb/s、1000Mb/s、1400Mb/s、1800Mb/s、2000Mb/s，将以上 4 个算法进行比较。同时，设置 *W* 为 40，说明大部分的计算节点将快速进入可用状态。设置任务输入数据为 300MB，节点处理速度 *Psd* 为 20，*jwd* 为 100。实验分别设置了 30 个计算节点，20 个作业(400 个任务)；60 个计算节点，20 个作业(400 个任务)两种计算环境。最后记录的是作业平均的完成时间。(RBA_30_400 表示 30 个节点，400 个任务时，RBA 的结果；RBA_60_400 表示 60 个节点，400 个任务，RBA 的结果。其他表述同理)

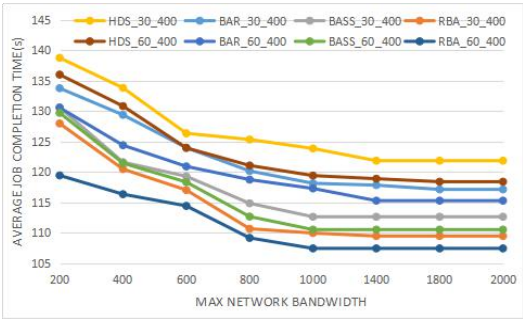

图 2. 网络带宽对作业完成时间的影响

实验结果如图 2 所示。在两种计算环境中，随着网络中带宽的增大，4 种算法的作业完成时间都会有减少的趋势。因为 4 种算法都会将部分任务分配给远程节点，带宽的增大，一定程度上是可以降低任务输入数据在传输过程中的时间。同时，在网络中的最大带宽为 200Mb/s 至 800Mb/s 之间时，4 种算法对应的作业的完成时间减少的趋势都较大，当最大带宽大于 800Mb/s 时，4 种算法的作业完成时间减少的趋势会减缓。这是因为，作业的完成时间是由任务的处理时间和任务数据的传输时间两部分组成的，当带宽增大到一定程度时，带宽对于传输时间的改善会趋于稳定。

而 HDS 并没有将带宽作为任务分配时的参数。BAR 是在 HDS 的分配基础上，尝试对作业中最晚完成的任务进行调整，若最晚完成的任务在其它节点能提前完成，则对该任务进行重新分配，以降低作业的完成时间。所以，实质上，BAR 也是没有将带宽作为任务分配时的参数。对于 RBA 而言，是以网络中的最大可用剩余带宽作为任务分配时的重要参数，每次进行任务分配时，若任务的最早本地节点的可用时间大于远程最早可用节点，网络中的最大可用剩余带宽若能使得任务在该远程最早可用节点上的完成时间更早，则将该任务分配给该远程节点。所以，RBA 充分考虑了任务数据本地性，同时也充分利用网络中的最大可用剩余带宽，将任务分配给远程节点的方式，减少任务的等待时间，从而降低作业的完成时间。BASS 也是将网络带宽作为任务分配的参数，但是在进行任务分配前，需要对网络中的最大带宽按时间进行分片，相比于 RBA 这是附加的工作，且 RBA 充分利用网络中的最大可用剩余带宽较为适用于实际场景。且从图 2 看出，总体上 RBA 的作业完成时间是较优于 BASS、BAR 和 HDS 的。

4.2.2 数据本地性的评估

实验 2：根据实验 1 的不同计算环境，4 种算法执行后，分配给本地节点的任务数占总任务数的比例有所差异，即检验 4 个算法在维持任务本地性方面的性能差异。各个节点对应的本地任务的数据本地率计算如公式(1)所示。结果如图 3 所示。

由图 3 可得出，在两种计算环境中，随着网络带宽的增加，4 种算法对应的 *DR* 值都有所降低，这是因为，带宽的增加，任务输入数据的传输时间会相对有所降低，任务的完成时间会降低，节点对应的可用时间会有所降低，则任务整体的分配情况在不同的带宽下会有所变化，对应的 DR 结果值也会有所不同。如带宽为 1000Mb/s(1800Mb/s)时，BAR 与 HDS(60 个节点，400 个任务时)的 DR 值分别为 91.5%、91.4%(91.3%、91.2%)，因为 BAR 通过不断的调整，可能会将作业最晚完成的任务分配给其对应的本地节点，降低了作业的完成时间，同时提高了 BAR 算法执行结果的 DR 值。图 3 中，整体上 HDS 和 BAR 比 BASS 和 RBA 的 DR 值高。且 BASS 和 RBA 的 DR 值绝对值相差为 0.4%左右。这是因为，BASS 和 RBA 都会在分配任务时，将带宽作为重要参数，不同的是 RBA 使用的是网络中的最大剩余可用带宽，而 BASS 是将带宽按时间分片进

行分配，所以存在链路的时间分片不可用的情况，因此两者最后的任务分配情况会有差异，对应 DR 值也有所不同,但两者的作业完成时间由图 2 可知，RBA 的值是较优于 BASS 的。

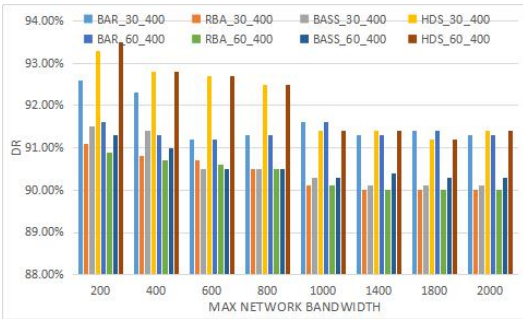

图 3. 网络带宽对 *DR* 的影响

综上可得，虽然在任务数据本地性比例上来说，RBA 不是最优的，但是 RBA 可以在充分保证任务数据本地性的情况下，充分利用网络中的最大可用剩余带宽，实现任务的近似最优分配，达到最小化作业完成时间的目的。

4.2.3 计算时间的评估

实验 3，主要是验证 RBA 和 HDS、BASS、BAR 在计算时间方面的性能差异。设置网络中链路最大带宽为 1000 Mb/s，计算节点为 90，*W* 为 800，任务输入数据为 300MB，节点处理速度 *Psd* 为 20，*jwd* 为 100。实验结果记录的是执行 4 个算法产生任务的分配方案所消耗的时间，取值为 10 次运行时间的平均值，保证结果的客观性。

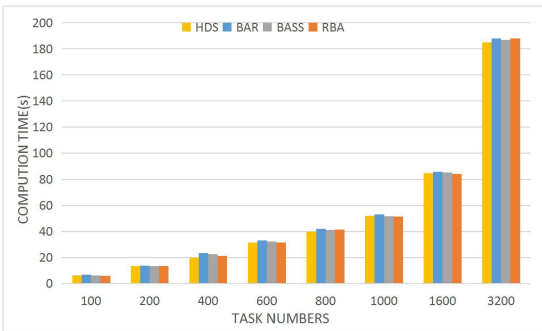

图 4 计算时间

实验 3 的结果如图 4 所示。在计算时间方面，4 个算法之间的差异总体不是很大，如，任务为 3200 个时，RBA 的计算时间为 187.913，与计算时间最少的 HDS(为 184.957)相差 2.956。所以，综合以上 3 个实验可得，在类似的计算时间内，使用 RBA 可以获得比其它 3 个算法更好的效果。即，RBA 总体上是较优于其他 3 个算法。

5 结论和未来工作

本文将 SDN 引入 Hadoop，利用 SDN 的带宽控制能力，将网络中的可用剩余带宽作为任务调度的重要参数；定义作业完成时间问题的描述，在充分考虑任务本地性的情况下，根据网络中的可用剩余带宽提出任务调度算法 RBA，因此可获得任务的近似最优分配方案，实现作业完成时间的最小化。最后通过仿真对比实验，验证了 RBA 在作业完成时间、任务数据本地性及计算时间方面的性。总体上 RBA 是优于其它 3 个算法的。

未来的一个工作是考虑 Hadoop 配置(如 reduce 个数)对算法的影响，并将 RBA 运用到真实的 Hadoop 环境中，与 Hadoop 的资源管理器 yarn 中的 capacity 调度器和 fair 调度器进行比较并改进。同时，实际中，计算系统会有出现故障的情况，本文在 Hadoop 中引入了 SDN，同样可能会存在链路或节点故障等，因此，在任务调度时进行故障容错也是未来的一个研究方向。

References

[1]Dave T. OpenFlow: Enabling Innovation in Campus Networks[J]. Acm Sigcomm Computer Communication Review, 2008, 38(2): 69-74.

[2]Olson M. HADOOP: Scalable, Flexible Data Storage and Analysis[J]. Iqt Quarterly, 2010.

[3]Narayan S, Bailey S, Daga A, et al. OpenFlow Enabled Hadoop over Local and Wide Area Clusters[C]// SC Companion: High PERFORMANCE Computing, NETWORKING Storage and Analysis. IEEE Computer Society, 2012:1625-1628.

[4]Costa P A R S, Ramos F M V, Correia M. Chrysaor: Fine-Grained, Fault-Tolerant Cloud-of-Clouds MapReduce [C]// Ieee/acm International Symposium on Cluster, Cloud and Grid Computing. IEEE Press, 2017:421-430.

[5]Deng J, Tyson G, Cuadrado F, et al. Keddah: Capturing Hadoop Network Behaviour[C]// IEEE, International Conference on Distributed Computing Systems. IEEE, 2017:2143-2150.

[6]Li H, Fox G, Qiu J. Performance Model for Parallel Matrix Multiplication with Dryad: Dataflow Graph Runtime[C]// Second International Conference on Cloud and Green Computing. IEEE, 2013:675-683.

[7]Hong S, Choi W, Jeong W K. GPU in-Memory Processing Using Spark for Iterative Computation[C]// Ieee/acm International Symposium on Cluster, Cloud and Grid Computing. IEEE, 2017:31-41.

[8]Qin P, Dai B, Huang B, et al. Bandwidth-Aware Scheduling With SDN in Hadoop: A New Trend for Big Data[J]. IEEE Systems Journal, 2014, PP(99):1-8.

[9]White T. Hadoop - The Definitive Guide: Storage and

Analysis at Internet Scale. 1st ed.OReilly Media, Inc., 2015.
[10]Jin J, Luo J, Song A, et al. BAR: An Efficient Data Locality Driven Task Scheduling Algorithm for Cloud Computing[C]// Ieee/acm International Symposium on Cluster, Cloud and Grid Computing. IEEE, 2011: 295-304.
[11]Zhang X, Wu Y, Zhao C. MrHeter: improving MapReduce performance in heterogeneous environments[J]. Cluster Computing, 2016, 19(4):1-11.
[12]Zhang J T, Huang H J,Wang X. Resource provision algorithms in cloud computing:A survey[J]. Network and Computer Applications, 2016;64:23-42.
[13]Zaharia M, Borthakur D, Sarma J S, et al. Job scheduling for multi-user MapReduce clusters[J]. 2009.
[14]Zaharia M, Borthakur D, Sen J, et al. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling[C]// European Conference on Computer Systems. ACM, 2010:265-278.
[15]Tan J, Meng X, Zhang L. Coupling task progress for MapReduce resource-aware scheduling[C]// INFOCOM, 2013 Proceedings IEEE. 2013:1618-1626.
[16]Seo S, Jang I, Woo K, et al. HPMR: Prefetching and pre-shuffling in shared MapReduce computation environment[C]// IEEE International Conference on CLUSTER Computing and Workshops. IEEE, 2010:1-8.
[17]Fischer M, Su X, Yin Y. Assigning tasks for efficiency in Hadoop: extended abstract[C]// ACM Symposium on Parallelism in Algorithms and Architectures. ACM, 2010:30-39.
[18]Wang G, Ng T S E, Shaikh A, Programming your network at run-time for big data applications[C]// The Workshop on Hot Topics in Software Defined Networks. ACM, 2012:103-108.