# 归约算法统一描述

熊玉庆
(上海应用技术学院计算机系 上海 201418)

**摘 要** 归约算法在并行计算中应用广泛,目前有很多归约算法应用于不同的情形。这些归约算法各不相同,逻辑拓扑是造成区别的关键。为了统一描述归约算法,揭示它们的共性,给出了一个逻辑拓扑的定义及其性质。在此基础上,给出了归约算法的统一描述,以利于对归约算法的理解,从而设计适应不同应用和环境的归约算法。该描述也可视为可集成不同语义的归约算法框架,从而有助于设计具有新语义的归约算法。本质上,该统一描述是一个归约算法形式定义,有助于验证归约算法的正确性。

**关键词** 归约算法,逻辑拓扑,算法描述,并行计算
**中图法分类号** TP301 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.11.021

## Towards Unified Description for Reduction Algorithms

XIONG Yu-qing
(Department of Computer Science, Shanghai Institute of Technology, Shanghai 201418, China)

**Abstract** Reduction algorithms are used widely in parallel computing. There are many reduction algorithms applied to different situations. These reduction algorithms are different from each other. Logical topology is the key cause to make the differences. In order to unify the description of reduction algorithms, and uncover their common feature, a definition of logical topology and its properties were presented in this paper. Based on it, a unified description for reduction algorithms was proposed. The description contributes to understand reduction algorithms and thus to design efficient reduction algorithms for different computational applications and environments. It can also be regarded as a reduction algorithmic framework for integrating different semantics and thus maybe guide to design new reduction algorithms with the semantics. Essentially, the unified description is a definition of reduction algorithms formally, and so it helps to verify correctness of reduction algorithms.

**Keywords** Reduction algorithm, Logic topology, Algorithm description, Parallel computing

## 1 引言

归约操作被广泛应用于大量的并行科学应用中,尤其常常在影响性能的关键路径上,这使得归约算法设计成为影响科学应用程序性能的重要因素[1]。

目前存在很多归约算法,它们适用于不同的情形。例如,对短消息,二项树(binomial tree)归约算法最佳[2];对长消息,则 Rabenseifner 在文献[3]提出的归约算法更好。这些归约算法似乎彼此相差很大,但是它们之间差异的根源在于逻辑拓扑。逻辑拓扑是在一个分布操作中控制消息如何传递的机制[4]。逻辑拓扑思想的使用向人们揭示这样一个事实:即便执行一个分布操作的时间不能减少,但该操作在各处理器上的执行时间分布却可以改变[4]。

尽管分布操作逻辑拓扑很重要,但是目前并不存在归约操作逻辑拓扑的确切定义。本文给出了一个逻辑拓扑形式定义,并得到该定义的几个基本性质。在此基础上,得到一个归约算法的统一描述。事实上,该描述也可视为归约算法的一个形式定义。

到稿日期:2014-10-15 返修日期:2014-12-25
熊玉庆(1964—),男,博士,副教授,主要研究方向为并行计算。

该描述将所有树型一到多归约算法纳入到一个统一的框架中,这有助于对归约算法的理解,从而有助于设计适应不同应用和环境的归约算法。该描述也可看作可集成不同语义的归约算法框架,从而有助于设计具有新语义的归约算法。该描述也可视为归约算法的一个形式定义,从而有助于验证归约算法的正确性。

本文第2节给出归约算法逻辑拓扑定义及其性质;第3节给出归约算法统一描述;第4节给出约算法统一描述的正确性证明;最后进行总结。

## 2 归约算法逻辑拓扑形式定义

设进程集合 $P=\{p_0,p_1,\cdots,p_{n-1}\}$,其中,$p_0$ 称为集聚进程(accumulative process),$N=\{\alpha_0,\alpha_1,\cdots,\alpha_l\}$ 是 $P$ 的一个划分,$l>0$。$T$ 是一棵有向加权树,其节点集合为 $N$,根为 $\alpha_0$,方向为从叶节点到根节点,权重集合为 $W$,$W$ 也称为时间步集合,它包含最小权重 0。从叶节点到根节点的每条路径上权重都是严格递增的。树 $T$ 中任意入度为 $d$ 的非叶节点 $\alpha$,存在一个从由 $d$ 条进入边构成的集合到 $\alpha$ 的映射 $f_\alpha$,该映射称

为 $\alpha$ 上的关联映射。

定义后继函数 $S:2^P\times W\to P\cup\{*\}$ 如下：

$S(\alpha,t)=p,p\neq *$，当且仅当存在 $p\in\beta,\beta\in N,\alpha\in N,e$ 是一条从 $\alpha$ 到 $\beta$ 的边，$\beta$ 上的关联映射 $f_\beta(e)=p$，并且 $e$ 的权重等于 $t$；否则，$S(\alpha,t)=*$。

设进程 $p\in\alpha,f_\alpha^{-1}(p)=\{e_1,e_2,\cdots,e_l\},l>1$。假设树 $T$ 中边 $e_1,e_2,\cdots,e_l$ 的权重分别是 $t_1,t_2,\cdots,t_l$，并且 $t_1<t_2<\cdots<t_l$，则 $t_{i+1}$ 称为进程 $p$ 在时间步 $t_i$ 的下一步，这里 $1\leqslant i\leqslant l-1$。

**定义 1**（归约操作逻辑拓扑） 归约操作逻辑拓扑 $TOP$ 定义为：

$$TOP=\bigcup_{S(\alpha,i)\neq *,\alpha\in N,i\in W}\{\alpha\times\{i\}\times\{S(\alpha,i)\}\}$$

其中，$N$ 是进程集合 $P$ 的一个划分，$W$ 是时间步集合，$S$ 是后继函数。

例 1 设进程集合 $P=\{p_0,p_1,\cdots,p_7\},N=\{\alpha_0,\alpha_1,\alpha_2,\alpha_3,\alpha_4\}$ 是 $P$ 的一个划分，其中，$\alpha_0=\{p_0\},\alpha_1=\{p_3,p_6\},\alpha_2=\{p_1,p_2\},\alpha_3=\{p_4,p_5\},\alpha_4=\{p_7\}$。树 $T$ 如图 1 所示。时间步集合 $W=\{0,1\}$，$\alpha_0$ 上的关联映射 $f_{\alpha_0}$ 是 $f_{\alpha_0}(\alpha_1\alpha_0)=p_0$，$f_{\alpha_0}(\alpha_2\alpha_0)=p_0$，$\alpha_1$ 上的关联映射 $f_{\alpha_1}$ 是 $f_{\alpha_1}(\alpha_3\alpha_1)=p_3$，$f_{\alpha_1}(\alpha_4\alpha_1)=p_6$。由后继函数定义，可得到后继函数 $S$ 为：$S(\alpha_2,0)=p_0,S(\alpha_1,1)=p_0,S(\alpha_3,0)=p_3,S(\alpha_4,0)=p_6$，对其它 $\langle\alpha,i\rangle\in 2^P\times W,S(\alpha,i)=*$。这里 $p_0$ 是集聚进程。

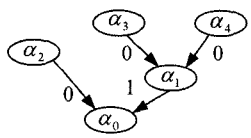



图 1 树 T

由归约操作逻辑拓扑定义，可得到下列归约操作逻辑拓扑：

$TOP=\{\langle p_1,0,p_0\rangle,\langle p_2,0,p_0\rangle,\langle p_4,0,p_3\rangle,\langle p_5,0,p_3\rangle,\langle p_7,0,p_6\rangle,\langle p_3,1,p_0\rangle,\langle p_6,1,p_0\rangle\}$

由上面逻辑拓扑 $TOP$，可得 $f_{\alpha_0}^{-1}(p_0)=\{\alpha_1\alpha_0,\alpha_2\alpha_0\}$，$\alpha_1\alpha_0$ 和 $\alpha_2\alpha_0$ 的权重分别是 1 和 0，故 $p_0$ 在时间步 0 的下一步是 1。

归约操作逻辑拓扑 $TOP$ 是 2-tree，如图 2 所示。

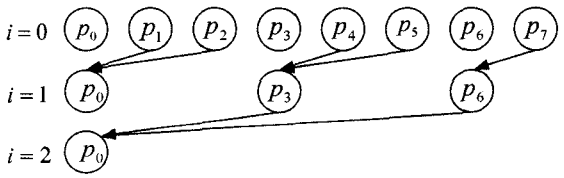



图 2 归约操作 2-tree 型逻辑拓扑

**定理 1** 对任意归约逻辑拓扑 $TOP$ 以及每个非集聚进程 $p\in P$，存在唯一进程 $q$ 使得在某个时间步 $i\in W$，有 $\langle p,i,q\rangle\in TOP$。$p$ 称为 $q$ 的前驱，$q$ 称为 $p$ 的后继。

证明：设 $p\in\alpha,\alpha\in N$。由于 $p$ 是非集聚进程，因此 $\alpha$ 不是树 $T$ 的根。这样，在树 $T$ 中必定存在唯一节点 $\beta$ 使得存在一条从 $\alpha$ 到 $\beta$ 的边 $e$。因此，在 $\beta$ 中存在唯一进程 $q$ 使得 $f_\beta(e)=q$。由此可得 $S(\alpha,i)=q$，这里 $i$ 是树 $T$ 中边 $e$ 的权重。由归约操作逻辑拓扑的定义，可得到 $\langle p,i,q\rangle\in TOP$。

由上所述，可得到以下推论。

**推论 1** 集聚进程没有后继进程。

**定理 2** 在归约逻辑拓扑 $TOP$ 中，对集聚进程 $p_0\in P$ 和任意非集聚进程 $q$，存在唯一一条从 $q$ 到 $p_0$ 的路径，即唯一地存在进程 $q_1,q_2,\cdots,q_k$ 和时间步 $i_1,i_2,\cdots,i_{k+1},k\geqslant 0$ 使得 $\langle q,i_1,q_1\rangle\in TOP,\langle q_1,i_2,q_2\rangle\in TOP,\cdots,\langle q_k,i_{k+1},p_0\rangle\in TOP$。

证明：设 $q\in\alpha,\alpha\in N$。由于 $q$ 是非集聚进程，$\alpha$ 不是树 $T$ 的根。由图论可知树 $T$ 中存在唯一从 $\alpha$ 到 $\alpha_0$ 的路径，设此路径为 $\alpha e_1\beta_1e_2\beta_2\cdots e_k\beta_ke_{k+1}\alpha_0$。由关联函数的定义可知，在 $\beta_1$ 中存在进程 $q_1$，$\beta_2$ 中存在进程 $q_2,\cdots,\beta_k$ 中存在进程 $q_k$，$\alpha_0$ 中存在进程 $q_0$，使得，$f_{\beta_1}(e_1)=q_1,\cdots,f_{\beta_k}(e_k)=q_k,f_{\alpha_0}(e_{k+1})=q_0$，这里 $f_{\beta_1},\cdots,f_{\beta_k},f_{\alpha_0}$ 分别是 $\beta_1,\cdots,\beta_k,\alpha_0$ 上的关联函数。再由后继函数定义，可得 $S(\alpha,i_1)=q_1,S(\beta_1,i_2)=q_2,\cdots,S(\beta_k,i_{k+1})=q_0$，其中，$i_1,i_2,\cdots,i_{k+1}$ 分别是边 $e_1,e_2,\cdots,e_{k+1}$ 的权重。因此，由归约逻辑拓扑定义，可得到

$\langle q,i_1,q_1\rangle\in TOP,\langle q_1,i_2,q_2\rangle\in TOP,\cdots,\langle q_k,i_{k+1},p_0\rangle\in TOP$

## 3 归约算法统一描述

设归约操作中的运算符为$\oplus$，$\oplus$满足交换律和结合律，即

$x\oplus y=y\oplus x,(x\oplus y)\oplus z=x\oplus(y\oplus z)$

设 *SEND* 和 *RECV* 是一对点到点的通信原操作。在 *SEND*(*msg*,*q*)中，*msg* 是发送消息缓冲首址，*q* 是接收消息的目的进程。在 *RECV*(*msg*,*g*)中，*msg* 接收消息缓冲首址，*p* 是发送消息的目的进程。当在 *RECV*(*msg*,*p*)中 *p* 是 *any_source* 时，表明调用进程将接收由任意进程发送的消息。

假设 $p$ 是一个进程，它调用归约算法，$p_0$ 是集聚进程。一个归约算法的统一描述如图 3 所示。

```
input:msg held by each process and TOP
output:A global-sum of the reduction operation
if there exists q such that ⟨q,i,p⟩∈TOP then
  i←min{i:⟨q,i,p⟩∈TOP};
  repeat
    while there exists q such that ⟨q,i,p⟩∈TOP do
      RECV(recv_msg,any_source);
      /* any_source may not cause any error by Theorem 1 */
      msg←msg ⊕ recv_msg;
      TOP←TOP−{⟨q,i,p⟩};
    end
    i←the next time step of p at i;
  until there does not exist such time step i of p;
end
if p≠p0 then
  SEND(msg,r);
  /* There exists such process r and time step j that ⟨p,j,r⟩∈TOP
     by Theorem 1 */
end
```

图 3 归约算法统一描述

## 4 归约算法统一描述的正确性

**定理 3** 统一描述的归约算法不会导致死锁。

证明：假设算法导致死锁，则在进程集合 $P$ 中存在进程 $q_1,q_2,\cdots,q_m,m\geqslant 2$，使得 $q_2$ 等待 $q_1$ 发送的消息，$q_3$ 等待 $q_2$ 发送的消息，$\cdots$，$q_1$ 等待 $q_m$。根据算法，这意味着存在时间步 $t_1,t_2,\cdots,t_m$ 使得 $\langle q_1,t_1,q_2\rangle\in TOP,\langle q_2,t_2,q_3\rangle\in TOP,\cdots,\langle q_m,t_m,q_1\rangle\in TOP$，这样 $\{q_1,q_2,\cdots,q_m\}$ 中每个成员都有一个后继。由推论 1，它们都不是集聚进程。设 $p_0$ 是集聚进程，根据定理 2，存在进程 $q_i\in\{q_1,q_2,\cdots,q_m\}$ 使得在 $P-\{q_1,q_2,\cdots,q_m\}$ 中存在进程 $r_1,r_2,\cdots,r_l,l\geqslant 0$，以及时间步 $j_1,j_2,\cdots,$

$j_{l+1}$，有$\langle q_i, j_1, r_1\rangle \in TOP, \langle r_1, j_2, r_2\rangle \in TOP, \cdots, \langle r_l, j_{i+1}, p_0\rangle \in TOP$。这样，$q_i$ 有两个不同的后继 $r_1$ 和 $q_{i-1}$（如果 $i<m$）或者 $r_1$ 和 $q_1$（如果 $i=m$），这与定理 1 矛盾。

**定理 4** 归约算法执行后，集聚进程持有所有进程中的数据经过⊕运算后的结果。

证明：运用定理 2 以及⊕运算的交换律和结合律即可得到结果。

**结束语** 由于造成归约算法相互不同的根本原因在于逻辑拓扑的不同，为了揭示归约算法的根本共性，首先形式地给出了归约操作逻辑拓扑的定义并得到该定义的基本性质。在此基础上，给出了一个归约算法的统一描述。描述将所有基于树型的多到一归约算法纳入到同一个框架中，这有助于人们对归约算法的理解，因为它提供了另一种分析归约算法的视角。实际上，该描述也是一个归约算法的形式定义。

## 参 考 文 献

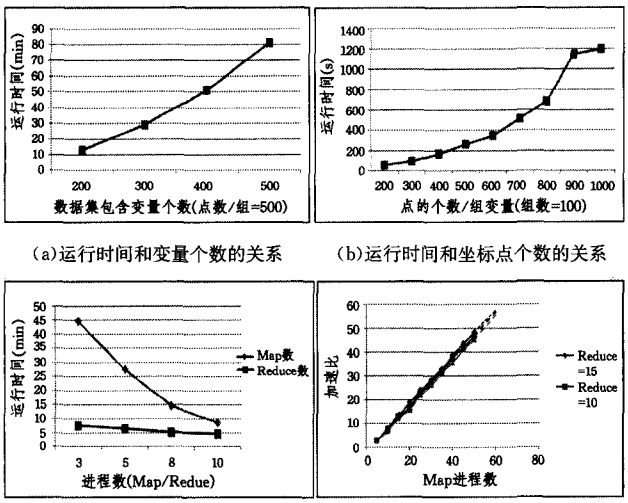

[1] Balaji P, Kimpe D. On the Reproducibility of MPI Reduction Operations: ANL/MCS-P4093-0713[R]. Argonne, IL, USA: Argonne National Laboratory, 2013

[2] Thakur R, Rabenseifner R, Gropp W. Optimization of Collective Communication Operations in MPICH: ANL/MCS-P1140-0304 [R]. Argonne, IL, USA: Argonne National Laboratory, 2004

[3] Rabenseifner R. New optimized MPI reduce algorithm [OL]. http://www.hlrs.de/organization/par/services/models/mpi/myreduce.html

[4] Dongarra J J, Whaley R C. A User's Guide to the BLACS v1.0.1: UT CS-95-281[R]LAPACK Working Note #94, University of Tennessee, 1995


（上接第 83 页）




(a)运行时间和变量个数的关系　(b)运行时间和坐标点个数的关系

(c)运行时间与系统资源的关系　(d)进程数与加速比的关系

图 7　算法性能分析

从实验(a)、(b)可知，在相同系统资源下，算法的运行时间与数据集中的变量个数正相关，与每组变量中包含的点个数正相关。从实验(c)可知，算法在相同数据规模时，系统资源越多，算法运行的时间越短，且 Map 进程数的改变对算法运行时间的影响更明显。实验(d)显示了改进后算法的加速比与系统资源的关系，由图可知，在相同数据规模时并行化后算法的加速比与 Map 进程数(并行度)呈线性增长的关系，而 Reduce 进程数则影响较弱，从而并行算法中 Map 任务的并行度对算法性能的提升起到了主要作用。实验(d)中虚线部分表示对线性关系趋势的预测，说明了该算法具有良好的可扩展性，能够随着集群中并行计算单元数的增加而得到有效的扩展。

**结束语** 鉴于 MIC 算法在处理包含大量变量数据集时算法性能方面的不足，对其进行了基于 MapReduce 模型的并行化，并用实验证明改进后算法具有较好的可扩展性和优良的加速比，同时说明了改进后算法与系统资源、数据规模的关系。

## 参 考 文 献


[1] Zhou A Y. Data intensive computing-challenges of data management techniques[J]. Communications of CCF, 2009, 5(7): 50-53

[2] Białynicki-Birula I, Mycielski J. Uncertainty relations for information entropy in wave mechanics [J]. Communications in Mathematical Physics, 1975, 44(2): 129-132

[3] Wells III W M, Viola P, Atsumi H, et al. Multi-modal volume registration by maximization of mutual information[J]. Medical Image Analysis, 1996, 1(1): 35-51

[4] Batina L, Gierlichs B, Prouff E, et al. Mutual information analysis: a comprehensive study[J]. Journal of Cryptology, 2011, 24(2): 269-291

[5] Reshef D N, Reshef Y A, Finucane H K, et al. Detecting novel associations in large data sets[J]. Science, 2011, 334(6062): 1518-1524

[6] Labrinidis A, Jagadish H V. Challenges and opportunities with big data[J]. Proceedings of the VLDB Endowment, 2012, 5(12): 2032-2033

[7] McAfee A, Brynjolfsson E, Davenport T H, et al. Big Data[J]. The management revolution. Harvard Bus Rev, 2012, 90(10): 61-67

[8] Lohr S. The age of big data[Z]. New York Times, 2012, 16(4): 10-15

[9] Dean J, Ghemawat S. MapReduce: a flexible data processing tool [J]. Communications of the ACM, 2010, 53(1): 72-77

[10] Wang H, Huang W, Zhang Q, et al. An improved algorithm for the packing of unequal circles within a larger containing circle [J]. European Journal of Operational Research, 2002, 141(2): 440-453

[11] White T. Hadoop: The definitive guide [M]. O'Reilly Media, Inc., 2012

[12] Groot S, Kitsuregawa M. Jumbo: Beyond MapReduce for workload balancing [C] // 36th International Conference on Very Large Data Bases. Singapore, 2010

[13] Zheng Q. Improving MapReduce fault tolerance in the cloud [C]// 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW). IEEE, 2010: 1-6

[14] Lee K H, Lee Y J, Choi H, et al. Parallel data processing with MapReduce: a survey[J]. ACM SIGMOD Record, 2012, 40(4): 11-20

[15] McKenna A, Hanna M, Banks E, et al. The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data[J]. Genome research, 2010, 20(9): 1297-1303