Journal of System Simulation

Volume 27 | Issue 9 Article 25

8-7-2020

Order-independent Image Compositing in Heterogeneous Environments

Liu Ning
1. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China;;2. University of Chinese Academy of Sciences, Beijing 100049, China;;

Dengming Zhu
1. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China;;

Zhaoqi Wang
1. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China;;

Shi Min
1. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China;;3. North China Electric Power University, Beijing 102206, China;

Order-independent Image Compositing in Heterogeneous Environments

Abstract

Abstract: Conventional image compositing algorithms are designed for homogeneous environments. These algorithms do not take heterogeneity into account and do not make use of the properties of an order-independent compositing operator. As for order-independent image compositing problems in a heterogeneous environment, these algorithms usually can not make full use of the resources. *An order-independent image compositing algorithm in heterogeneous environments was proposed, which was able to distribute the workload according to computing and networking characteristics, and managed to optimize compositing scheme by utilizing the properties of order-independent operator.* A comprehensive evaluation on the ns-3 network simulator was conducted. Experiment results show when the environment has great heterogeneity, the algorithm will gain significant speedup over classical algorithm binary swap.

Keywords

image compositing, heterogeneous environment, parallel visualization, differential evolution, ns-3 simulator

第 27 卷第 9 期 系统仿真学报© Vol. 27 No. 9
2015 年 9 月 Journal of System Simulation Sep., 2015

异构环境下顺序无关的图像合成算法

刘宁[1,2]，朱登明[1]，王兆其[1]，石敏[1,3]
（1.中国科学院计算技术研究所，北京 100190；2.中国科学院大学，北京 100049；3.华北电力大学，北京 102206）

摘要：传统的图像合成算法是为同构环境设计的。对于异构环境下顺序无关的图像合成问题，这些算法不考虑环境的异构性，也没有利用顺序无关合成操作符的性质，容易造成资源的浪费。*针对这个问题，提出了一个异构环境下顺序无关的图像合成算法，它会根据环境的计算资源与网络资源更合理地进行任务分配，并利用顺序无关合成操作符的性质来优化合成方案。*在 ns-3 网络模拟器上进行了大量的实验。实验结果表明:当环境存在较大异构性时，该方法相对经典算法 binary swap 能够取得显著的加速比。

关键词：图像合成；异构环境；并行可视化；差分进化算法；ns-3 网络模拟器
中图分类号：TP391 文献标识码：A 文章编号：1004-731X (2015) 09-2117-09

Order-independent Image Compositing in Heterogeneous Environments

Liu Ning[1,2], *Zhu Dengming*[1], *Wang Zhaoqi*[1], *Shi Min*[1,3]

(1. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China;
2. University of Chinese Academy of Sciences, Beijing 100049, China; 3. North China Electric Power University, Beijing 102206, China)

Abstract: Conventional image compositing algorithms are designed for homogeneous environments. These algorithms do not take heterogeneity into account and do not make use of the properties of an order-independent compositing operator. As for order-independent image compositing problems in a heterogeneous environment, these algorithms usually can not make full use of the resources. *An order-independent image compositing algorithm in heterogeneous environments was proposed, which was able to distribute the workload according to computing and networking characteristics, and managed to optimize compositing scheme by utilizing the properties of order-independent operator.* A comprehensive evaluation on the ns-3 network simulator was conducted. Experiment results show when the environment has great heterogeneity, the algorithm will gain significant speedup over classical algorithm binary swap.

Keywords: image compositing; heterogeneous environment; parallel visualization; differential evolution; ns-3 simulator

引言

图像合成是并行可视化系统的一个重要阶段，它将各节点上的部分结果合成为最终结果。由于需要密集的节点间通信，它常常成为整个系统的瓶颈部分。

针对同构环境，研究人员已经提出了许多有效的图像合成算法，如 direct send[1]，binary swap[2]，2-3swap[3]，radix-k[4]和 pipelining image compositing [5]。这些算法均假定计算资源与网络资源是同构的，即各节点的计算能力与网络链接的带宽和延迟是相同的。因此任务分配也是以均匀方式进行的。同时，这些算法假定合成操作符是顺序相关的，即节点间合成的顺序会影响最终的合成结果。对于异构环境

收稿日期：2015-06-04 修回日期：2015-07-02；
基金项目：国家自然科学基金(61173067；61379085；61300131)；
作者简介：刘宁(1987-)，男，山东，博士生，研究方向为科学数据可视化；朱登明(1973-)，男，博士，副研究员，研究方向为自然现象模拟和科学数据可视化；王兆其(1966-)，男，博士，研究员，研究方向为虚拟现实和智能人机交互。

下顺序无关的图像合成问题，由于传统的合成算法既不考虑环境的异构性，也不会利用顺序无关操作符的性质，因此通常不能有效地利用异构环境的资源。

针对这个问题，本文提出了一个异构环境下顺序无关的图像合成算法。按照节点间是否存在确定的合成顺序，图像合成可以分为顺序相关与顺序无关两种方法。在科学数据可视化中，顺序相关的图像合成主要应用于直接体绘制中，而顺序无关的图像合成主要运用于面绘制中。我们提出的算法是针对顺序无关的图像合成的。顺序无关的图像合成比顺序相关的图像合成施加的限制少，因此有更大的优化空间。本文的方法将异构环境下顺序无关的图像合成问题建模为一系列数据交换方案的确定问题，这些数据交换方案决定着节点间的数据通信与任务量的分配，因此可以完整地描述整个图像合成过程。最优合成方案的确定可以归结为该模型的最优化问题。在确定最优合成方案时，本文的方法会将环境的异构性考虑进去，它能够根据节点计算能力与网络链接带宽和延迟更合理地进行任务分配，因此能够提高异构环境资源的利用效率。解空间的大小与节点数目的阶乘成正比，因此暴力穷举的方式无法使用。我们采用了一种启发式全局随机优化算法差分进化[6]来求解。差分进化作用于实数域，而图像合成方案解的形式则是一个整数序列，本文算法使用了一种称为 random keys[7]的技术对两个域进行相互转换。差分进化首先初始化一族解，然后这些解通过交叉和变异等操作进行进化，在这个过程中时间耗费小的解会被保留下来，而时间耗费大的解则被淘汰。最终从存活下来的解选择最优的解作为算法的结果。

我们在 ns-3[8]网络模拟器上进行了大量实验。ns-3 是一个离散事件模拟器，广泛运用在网络相关的科研领域。我们从节点数目、计算能力、链接带宽、链接延迟和图像大小等多方面与经典算法 binary swap 算法进行了比较，并对结果进行详细的分析。实验结果证明当环境的异构程度较大时，本文算法相比传统算法能够取得显著的加速比。

1 相关工作

Direct send 是一种简单的图像合成算法。在合成过程中，一个节点会直接将驻留在该节点但不属于该节点负责的数据发到相应的节点去。在节点数目较大时，它会造成网络阻塞。因此它只适合节点数目比较少的情况。

Binary swap 利用合成操作符的结合律大大减少了节点间通信的次数。该方法能够有效地避免网络阻塞，并且能够扩展到节点数目很大的情况。它的主要限制是参与节点的数目必须为 2 的幂次方。2-3 swap 是对 binary swap 的一个扩展，它能够作用于任意数目的节点上。

Radix-k 统一了 direct send 与 binary swap。也就是说，direct send 与 binary swap 可以视为 radix-k 的特例。Radix-k 将节点数目分解为 r 个因数之积，然后将合成过程分为 r 个阶段，在每个阶段中利用 direct send 进行组内的图像合成。Direct send 可以看为 $r=1$ 的情形，而 binary swap 则可以看为将 N 分解为若干个 2 乘积的情形。根据文献[9-10]提供的结果，radix-k 的性能要同时优于 direct send 和 binary swap。一方面，它将节点依次分为 r 个组，这样每个组内的节点数就要小于 direct send 的节点数目，因此它能够有效地避免网络阻塞的问题。另一方面，它的分组方案较 binary swap 的灵活，同一个组内包含的节点数一般会多于 binary swap 的两个，这样它就能利用网络通信和计算之间的重叠部分以提高图像合成的效率。

Pipelining image compositing 将节点组织为一个管线。这个算法的优点是它支持渐进式显示。在最终结果展示的时间方面，该算法的性能要优于 binary swap，因此它比较适合于进行远程可视化的场景。

上述这些算法均是为同构环境设计的，而且针对的是顺序相关的合成操作符。当环境的计算与网络资源变得异构的时候，这些算法可能会造成比较

严重的负载不均衡。顺序无关的合成操作符由于对节点间合成的顺序没有施加限制，因此它有更大的优化空间。而这一性质是传统算法所忽略的。本文提出的异构环境下顺序无关的图像合成算法既考虑了环境的异构性，又利用了顺序无关合成操作符的性质，因此能够更有效地利用异构资源。

2 问题定义

2.1 图像合成概述

假定一共有 N 个节点，编号分别为 1 到 N。因为我们考虑的是顺序无关的图像合成，所以节点间的相对顺序不重要。同时我们用 $S=\{i_1,i_2,\cdots,i_m\}$ 表示一个包含 m 个节点的集合。假设节点 i 持有局部绘制生成的部分图像 x_i。图像合成的目标是将这些部分图像进行融合并生成最终合成结果 x_F，即

$$x_F = x_1 \otimes x_2 \otimes \cdots \otimes x_N$$

这里 $\otimes$ 是一个图像合成操作符。顺序无关的图像合成操作符既符合结合律又符合交换律，顺序相关的合成操作符则只符合结合律而不符合交换律。顺序无关的操作符对合成过程施加的限制较少，因此相应地合成方案的优化空间也更大。

2.2 异构环境下顺序无关的图像合成

给定一个节点集合 S，一个节点对 $G=\{u,l\}$ 由 S 中两个不同的节点 u 和 l 组成。在合成过程的任一阶段，只有属于同一个节点对的节点才可以交换数据。

节点对是针对两个节点的，而节点集合 S 的一个数据交换方案 P 则描述了该节点集合所有节点的配对信息和任务分配信息。准确地讲，$P=(\{G_1,G_2,\cdots,G_{m/2}\},\lambda)$，其中 m 是节点集合 S 中包含节点的个数。这 m 个节点会形成 m/2 个节点对，分别由 $G_1,G_2,\cdots,G_{m/2}$ 表示。$\lambda\in[0,1]$ 是一个实数，它决定着各节点对两个节点间的任务分配。假定 S 需要处理的图像空间大小为 I，则节点对 G_k 中 u_k 的图像空间大小为 λI，而 l_k 的图像空间大小为 $(1-\lambda)I$。

对于一个节点集合 S 及 S 上面的一个数据交换方案 P，P 中节点对的第一个节点构成的节点集合 S_U 负责 S 图像空间的上半部分，各节点对的第二个节点构成的节点集合 S_L 负责 S 图像空间的下半部分，即

$$S_U=\{u_1,u_2,\cdots,u_{m/2}\}$$

$$S_L=\{l_1,l_2,\cdots,l_{m/2}\}$$

其中 u_k 和 l_k 是数据交换方案 P 中第 k 个节点对 G_k 的第一个节点和第二个节点。

整个图像合成过程可以用一系列的数据交换方案来描述。对于节点集合 S，首先确定其上的一个数据交换方案 P，该方案决定了节点间通信规则及每个节点分配的任务量，同时会形成两个小的节点集合 S_U 和 S_L。递归地对 S_U 和 S_L 进行划分，直到所形成新的节点集合无需再细分为止，此时合成过程完成。图 1 描述了这个过程。

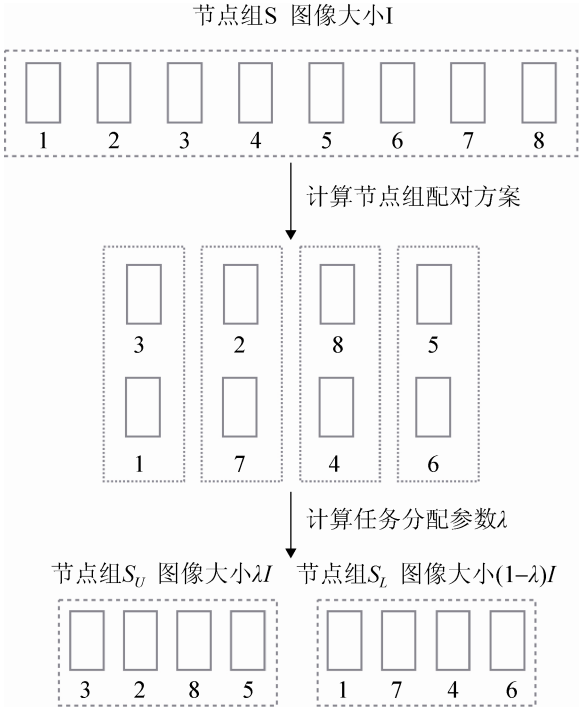

图 1 异构环境下顺序无关的图像合成示意图

3 我们的方法

3.1 记号

首先介绍一下文章中使用的记号，如表 1 所示。这些记号的具体意义将在后面具体介绍。

表 1 文章中使用的记号

符号	含义
$B(p,q)$	节点 p 与节点 q 之间的链接带宽
$L(p,q)$	节点 p 与节点 q 之间的链接延迟
$C(p)$	节点 p 的计算能力
S	一个节点集合
I	图像空间大小
P	一个节点集合的数据交换方案
$T_p(S,I,P)$	S 针对 I 和 P 的最优合成时间
$T_{opt}(S,I)$	S 针对 I 的最优合成时间
$t_p(S,I,P)$	S 针对 I 和 P 的划分时间

3.2 耗费函数

对于节点集合 S 和图像空间大小 I，整个图像合成的最优时间耗费记为：

$$T_{opt}(S,I)=\min_P T_P(S,I,P)$$

其中 $T_P(S,I,P)$ 是节点集合 S 对于图像空间大小 I 和一个特定的数据交换方案 P 的最优合成时间。它的定义如下：

$$T_P(S,I,P)=t_P(S,I,P)+\max(T_{opt}(S_U,\lambda I),T_{opt}(S_L,(1-\lambda)I))$$

其中 $t_P(S,I,P)$ 是节点集合 S 对于图像空间大小 I 和一个特定的数据交换方案 P 的划分时间。准确地说，

$$t_P(S,I,P)=\max_{G_k}(\max(L(l_k,u_k)+\frac{\lambda I}{B(l_k,u_k)}+\frac{\lambda I}{C(u_k)},L(u_k,l_k)+\frac{(1-\lambda)I}{B(u_k,l_k)}+\frac{(1-\lambda)I}{C(l_k)})) \quad (1)$$

即 $t_P(S,I,P)$ 表示 S 中最慢的节点对的合成时间，而某一个节点对的合成时间又是由它包含的两个节点中较慢的一个决定的。$t_P(S,I,P)$ 只考虑节点集合 S 本身划分的时间，而不考虑由 S 衍生出来的子节点集合的划分时间。

3.3 优化方法

我们采用自顶向下的方式对图像合成模型进行求解。为了求解 S 的最优数据交换方案 P，首先我们确定 S 的配对方案，即确定 S_U 与 S_L；然后我们确定 P 的任务分配参数 λ。在得到 S 的配对方案与 λ 之后，S_U 与 S_L 分别独立地进行最优数据交换方案的求解。这个过程一直持续到节点集合只包含两个节点而无需继续细分为止。

3.3.1 节点集合配对方案的确定

对于顺序相关的图像合成来说，节点间存在一个强的顺序约束，合成只能以一个预先定义好的顺序进行，因此节点配对方案是确定的，即

$$\binom{1}{2},\binom{3}{4},\ \cdots,\binom{m-1}{m} \quad (2)$$

这里 m 是节点集合中节点的数量。这个方案也是 binary swap 所采用的。对于顺序无关的图像合成来说，节点可以采用任意的合成顺序进行。因此我们有更大的自由度对配对方案进行优化。

对于一个有 m 个节点的节点集合，存在 $\Theta(m!)$ 种可能的配对方案。因此基于暴力穷举的方法是没有办法使用的。针对这个问题，我们采用差分进化算法来求得近似最优解。差分进化算法是一种随机全局优化算法，它可以用来对解空间巨大的情况求得近似最优解。差分进化算法随机地选择一族初始解，然后通过变异和交叉操作产生新的解，这些新产生的解与原有的解通过优胜劣汰的方式进行竞争，最后从幸存的解中选择最优解。由于差分进化在实数域上进行操作，而节点配对方案是一个整数序列，因为我们需要一个映射方案来实现两个域的转换。本文采用的是一个称为 random keys 的方法。在本小节的后续部分，我们首先介绍 random keys 编码方案，然后具体介绍差分进化算法的各个主要阶段。

(1) 解的编码。Random keys 提供了一种简单有效的实数域与整数序列相互转换的方法。首先它将一个实数数组进行排序，并记录各元素在排序好的数组中的 rank 值。这些 rank 值形成一个整数序列，而这个整数序列可以直接解释为节点集合的配对方案。例如，在一个 4 维 random keys 空间有一个点，$(0.45,0.76,0.21,0.55)$，进行映射之后我们得到 $(2,4,1,3)$ 这个整数序列对应于一个配对方案

$\begin{pmatrix}2\\4\end{pmatrix},\begin{pmatrix}1\\3\end{pmatrix}$

(2) 解的初始化。差分进化算法会随机地初始化一族解。假定参与节点的数量是 m，则差分进化算法会在 m-维 random keys 空间生成 NP 个点，并用它们作为初始解。这里 NP 代表初始解种群的大小。

(3) 解的进化。在初始解种群确定之后，它会进化若干代。这个迭代的次数是预先设定的。在每次迭代过程中，种群中的每个解会通过交叉和变异操作产生一个新的实验解。这个实验解会与原始解进行比较，两者中有较小时间耗费的会被保留下来进入到下一代，另外一个会被丢弃。进化完成后，差分进化算法从幸存的解中选择耗费最小的解作为最优解。

变异。对于第 G 代中的每个解 $x_{i,G}, i\in[1,NP]$，通过下面的方式产生一个变异解：

$$v_{i,G+1}=x_{r_1,G}+F\times(x_{r_2,G}-x_{r_3,G})$$

这里 $r_1,r_2,r_3\in[1,NP]$ 是 3 个互不相同且不等于 i 的随机下标；F 是一个缩放因子，一般介于 0 和 2 之间。

交叉。为了使得产生的解更具有多样性，差分进化算法引入了交叉的概念。经过交叉后，实验解具有下列形式：

$$u_{i,G+1}=(u_{1i,G+1},u_{2i,G+1},\cdots,u_{mi,G+1})$$

其中，$u_{ji,G+1}=\begin{cases}v_{ji,G+1} & \text{rand()}\leqslant CR\text{或}j=\text{rnbr(i)}\\ x_{ji,G} & \text{其它情况}\end{cases}$

在这里，$j\in[1,m]$，rand()是一个随机数生成器，它的输出在 0 和 1 之间；$CR\in[0,1]$ 是一个交叉常量；$\text{rnbr}(i)\in[1,m]$ 是一个随机选择的下标。

选择。为了决定某一个候选解是否能进入到第 G+1 代，实验解 $u_{i,G+1}$ 会与目标解 $x_{i,G}$ 进行比较。如果 $u_{i,G+1}$ 具有更小的时间耗费，则 $x_{i,G}$ 会被 $u_{i,G+1}$ 替代，否则旧的值 $x_{i,G}$ 则会被保留下来。

3.3.2 任务分配参数 λ 的确定

在 S 的配对方案即 S_U 和 S_L 确定之后，接下来需要确定数据交换方案 P 的任务分配参数 λ，这个参数决定 S_U 处理的图像空间大小占 S 图像空间大小的比例。根据公式(1)， S_U 中节点的最大时间耗费为：

$$t(S_U)=\max_{G_k}\left(L(l_k,u_k)+\frac{\lambda I}{B(l_k,u_k)}+\frac{\lambda I}{C(u_k)}\right)$$

相应地， S_L 中节点的最大时间耗费为：

$$t(S_L)=\max_{G_k}\left(L(u_k,l_k)+\frac{(1-\lambda)I}{\mathrm{B}(u_k,l_k)}+\frac{(1-\lambda)I}{C(l_k)}\right)$$

在这里，$t(S_U)$ 是关于 λ 的单调不减函数，而 $t(S_L)$ 是关于 λ 的单调不增函数。而节点集合 S 关于 P 的划分时间为：

$$t_P(S,I,P)=\max\left(t(S_U),t(S_L)\right)$$

它的示意图见图 2。可以看出当 $t(S_U)$ 和 $t(S_L)$ 相等时，划分时间 $t_P(S,I,P)$ 取最小值。可以使用基于二分查找的算法在预先给出的误差容忍范围内快速地计算出 λ。

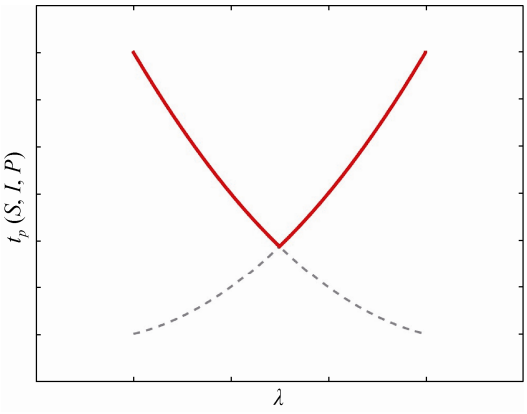

图 2 $t_P(S,I,P)$ 函数示意图

4 实验结果

4.1 实验环境搭建

我们在 ns-3 网络模拟器上进行了一系列的实验，参与节点的数目分别 8~64。ns-3 网络模拟器是一个离散事件模拟器，它被广泛地运用在与网络相关的科研领域。

4.1.1 网络拓扑生成

我们采用了全相联的网络拓扑结构。节点与节

点之间通过 Point-to-Point 的网络链接相连。各网络链接的带宽与延迟是根据指定的范围随机均匀选取的。

4.1.2 网络数据包生成

ns-3 网络模拟器提供了一些用于产生网络数据包的工具，OnOffApplication 就是其中之一。OnOffApplication 在 OnTime 时间段内持续生成 UDP 数据包，而在 OffTime 时间段里不产生数据包。在实际模拟中，我们根据图像空间大小来设定 OnTime 的值，同时将 OffTime 设为 0。OnOffApplication 的数据生成速率由相应网络链接的带宽决定。

4.1.3 图像合成模拟

Binary swap 的图像合成方案是确定的，如公式(2)所示。在收集到相应的计算能力信息和网络信息，我们的算法会给出一个针对该环境的经过优化的图像合成方案。这两个方案会分别被装载到 ns-3 模拟器中进行实际的模拟，相应的实验结果会被记录下来以供分析。

4.2 结果

在这一小节我们比较 binary swap 与我们算法之间的实验结果。控制图像合成模拟的参数如表 2 所示，主要包括节点数目、图像空间大小、节点计算能力、链接延迟和链接带宽。其中 BR 代表网络链接带宽的取值范围，在生成网络拓扑结构时，各链接的带宽会从这个范围中随机均匀地选取。LR 与 CR 也代表类似的概念。

表 2 实验中的控制参数

符号	含义
Nodes	参与节点的数目
I	图像空间大小(MB)
BR	链接带宽范围(Mbps)
LR	链接延迟范围(s)
CR	节点计算能力范围(MBps)

4.2.1 图像大小

我们分别将图像空间的大小设为 1MB、2MB 和 4MB，然后观察图像大小对算法效率的影响。其它的参数分别设定为：$Nodes=32$，$CR=[1,10^3]$，$BR=[1,10^2]$，$LR=[10^{-5},10^{-1}]$。实验结果如图 3 所示。

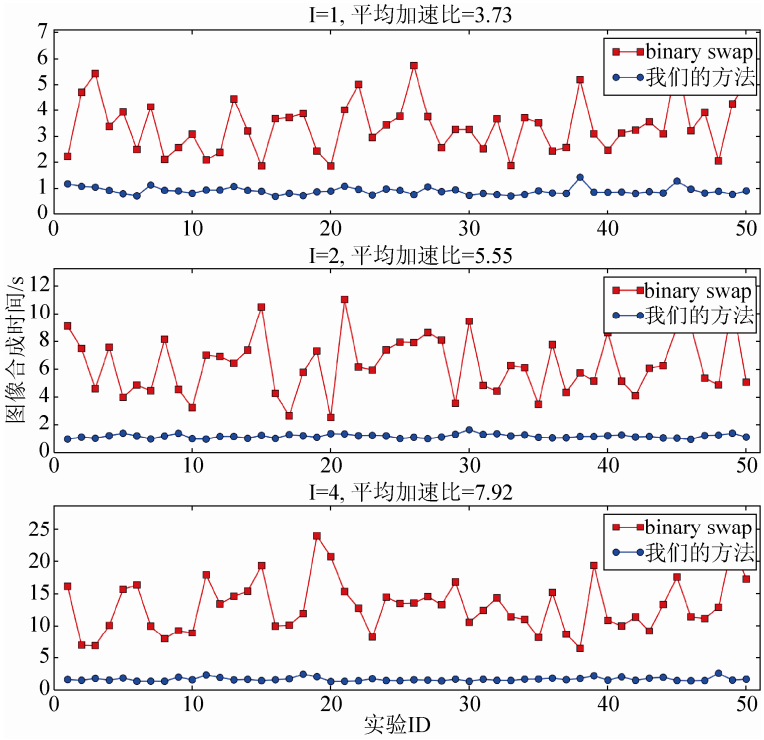

图 3 不同图像大小的结果

首先可以看到 binary swap 的结果震荡比较严重，这是因为它没有考虑环境的异构性，因此比较容易受到环境中慢的节点或者慢的链接的影响。同时，随着图像大小的增加，我们的算法相对于 binary swap 的平均加速比也在增加。这是因为图像空间越大，越多的数据需要被传输和处理，因此我们的算法拥有更多的空间去进行优化。

4.2.2 节点数目

我们分别将节点数目设为 8、16 和 64。其它的参数分别设定为：I=4，CR=[1,10^3]，BR=[1,10^2]，LR=[10^{-5},10^{-1}]。实验结果如图 4 所示。

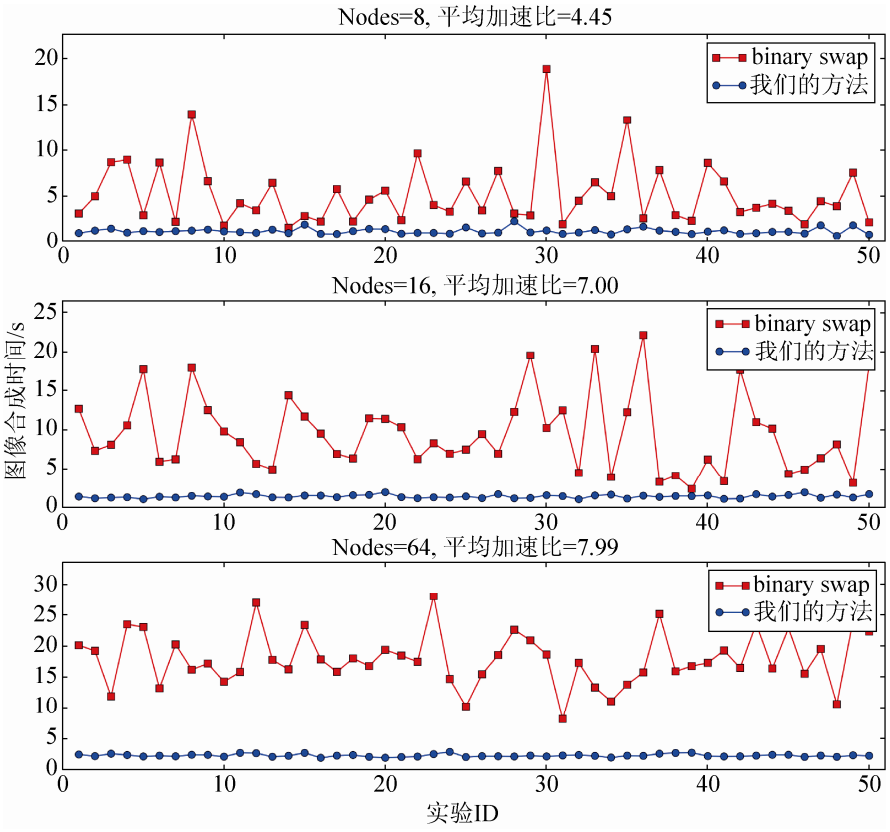

图 4 不同参与节点数目的结果

从实验结果可以看出随着参与节点数目的增加，我们算法相对于 binary swap 的加速比在增加。这是因为我们的方法从一个阶乘量级的解空间中选择最优解，参与节点数目越多，则优化空间越大。因此我们的方法能取得更大的加速比。

4.2.3 链接带宽

我们分别将链接带宽范围 BR 设定为[0.1, 1]、[0.1, 10]和[0.1, 100]。其它参数设定为：I=4，$Nodes$=32，CR=[1,10^3]，LR=[10^{-5},10^{-1}]。结果如图 5 所示。

随着网络链接带宽范围的不断变大，系统的异构性也在增加。Binary swap 比较容易受到系统异构性的影响，而我们的算法能够更合理地分配资源，在给定的条件下能够找到尽可能合理的合成方案。同时，环境异构性越大，我们算法表现出的性能越好。

4.2.4 链接延迟

我们分别将链接延迟范围 LR 设定为[10^{-5},10^{-3}]和[10^{-3},1]，其中前者模拟一个低延迟的环境，而后者模拟一个高延迟的环境。其它参数分别设定为：I=4，$Nodes$=32，CR=[1,10^3]，BR=[1,10^2]。结果如图 6 所示。

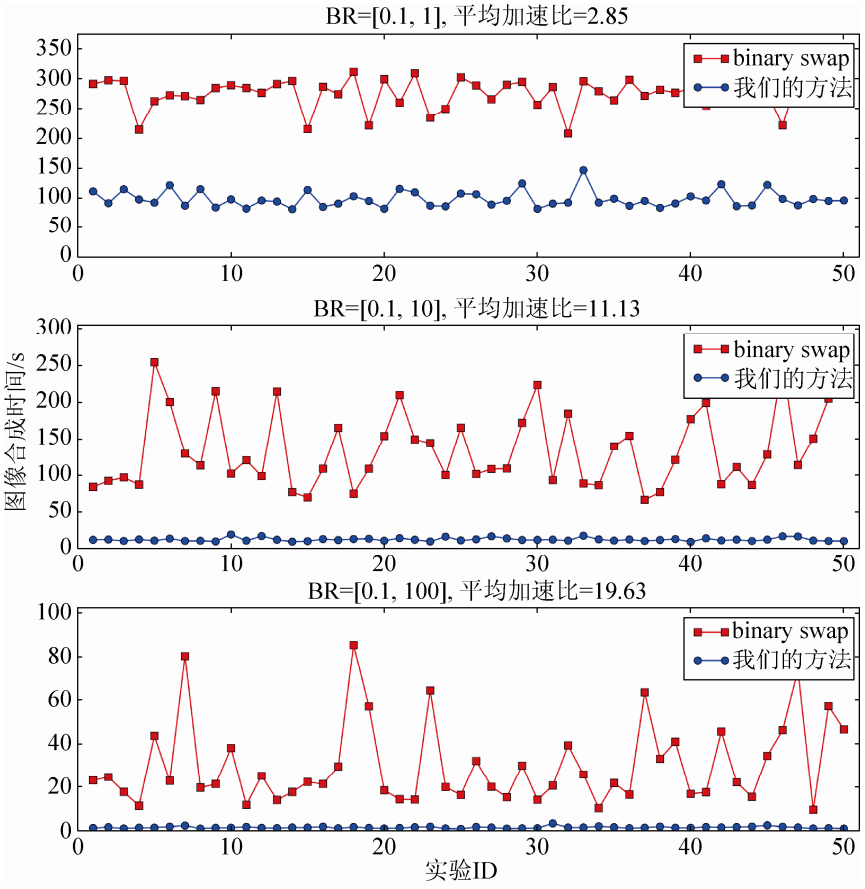

图 5 不同链接带宽的结果

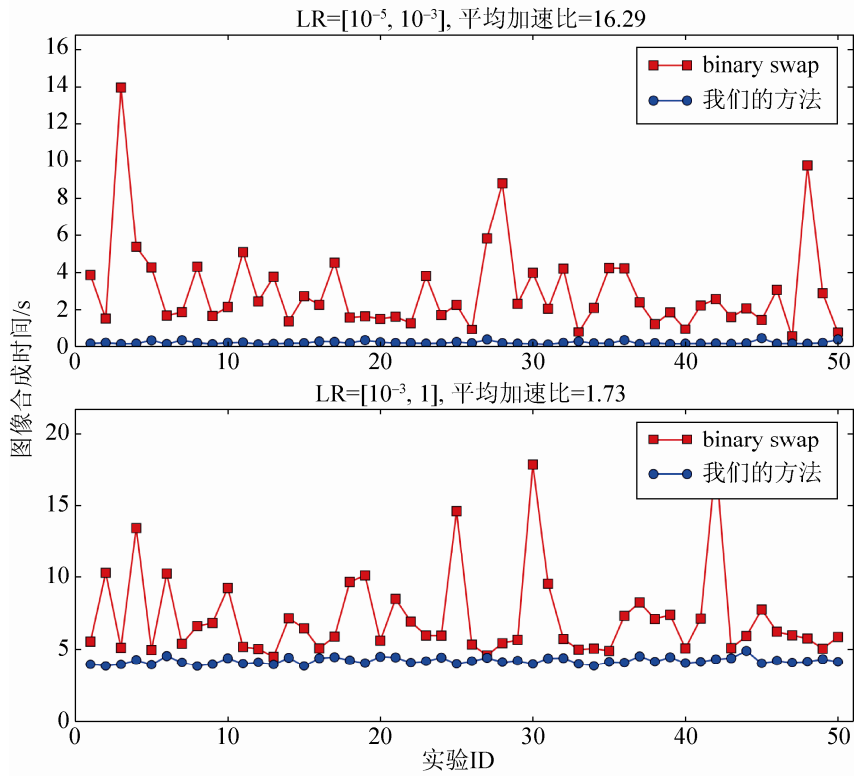

图 6 不同网络延迟的结果

在第一种延迟环境下，延迟对整个图像合成时间影响比较小，同时计算能力与网络带宽有比较大的异构性，因此我们算法相对于 binary swap 的加速比较大。在第二种延迟环境下，延迟是整个图像合成时间的决定性因素，而延迟本身的异构性不大，因此我们算法取得的加速比比较小。

5 结论

本文提出了一个异构环境下顺序无关的图像合成算法。它能够根据节点的计算能力与网络链接的延迟和带宽合理地分配任务，并且利用顺序无关合成操作符的性质来优化合成方案。本文对异构环境下顺序无关的图像合成问题给出了形式化的定义，并提出了相应的耗费函数。由于解空间巨大，我们使用了启发式全局随机优化算法差分进化对这个问题进行求解。我们在 ns-3 网络模拟器上进行了大量的实验，分别在图像空间大小、节点数目、网络带宽与网络延迟等方面与传统经典算法 binary swap 进行了对比，实验结果验证了我们方法的有效性。

参考文献:

[1] Neumann U. Parallel volume-rendering algorithm performance on mesh-connected multicomputers [C]// Proceedings of the Symposium on Parallel Rendering. New York, USA: ACM Press, 1993: 97-104.

[2] Ma K L, Painter J, Hansen C, *et al*. Parallel volume rendering using binary-swap compositing [J]. IEEE Computer Graphics and Applications(S0272-1716) , 1994, 14(4): 59-68.

[3] Yu H, Wang C, Ma K L. Massively parallel volume rendering using 2-3 swap image compositing [C]// Proceedings of the ACM/IEEE Conference on Supercomputing. New Jersey, USA: IEEE Press, 2008, 48:1-48:11

[4] Peterka T, Goodell D, Ross R, *et al*. A configurable algorithm for parallel image-compositing applications [C]// Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis. New York, USA: ACM Press, 2009: 4: 1-4:10

[5] Wu Q, Gao J, Chen Z, *et al*. Pipelining parallel image compositing and delivery for efficient remote visualization [J]. Journal of Parallel and Distributed Computing(S0743-7315), 2009, 69(3): 230-238.

[6] R Storn, K Price. Differential evolution a simple and efficient heuristic for global optimization over continuous spaces [J]. Journal of Global Optimization(S0925-5001), 1997, 11(4): 341-359.

[7] J C Bean. Genetic algorithms and random keys for sequencing and optimization [J]. ORSA Journal On Computing(S0899-1499), 1994, 6(2): 154-160.

[8] Ns-3 developers. ns-3 network simulator [EB/OL]. (2015-05-01). http://www.nsnam.org/

[9] Kendall W, Peterka T, Huang J, *et al*. Accelerating and benchmarking radix-k image compositing at large scale [C]// Proceedings of the Eurographics Conference on Parallel Graphics and Visualization. Aire-la-Ville, France: Eurographics Association, 2010: 101-110.

[10] Moreland K, Kendall W, Peterka T, *et al*. An image compositing solution at scale [C]// Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis. New York, USA: ACM Press, 2011, 25: 1-25: 10.