JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 20, 643-664 (2004)

A Rotate-Tiling Image Compositing Method for Sort-Last Parallel Volume Rendering Systems on Distributed Memory Multicomputers*

CHIN-FENG LIN, SHIH-KUAN LIAO+, YEH-CHING CHUNG++ AND DON-LIN YANG+
Department of Information Management
Chang Jung Christian University
Tainan, 711 Taiwan
+*Department of Information Engineering*
Feng Chia University
Taichung, 407 Taiwan
++*Department of Computer Science*
National Tsing Hua University
Hsinchu, 300 Taiwan

The binary-swap (BS) and the parallel-pipelined (PP) methods are two well-known image compositing methods for sort-last parallel volume rendering systems. However, these two methods either restrict the number of processors to a power-of-two or require many communication steps to transform image data that results in high data communication overheads. In this paper, we present an efficient image compositing method, the rotate-tiling (RT) method, for sort-last parallel volume rendering systems on distributed memory multicomputers. According to the number of initial blocks of a partial image, the number of processors, the image sizes, and the characteristics of parallel machines, the RT method can fully utilize all available processors and minimize the data communication overheads. To evaluate the performance of the RT method, both theoretical analysis and experimental test of the BS, the PP, and the RT methods are conducted. In the theoretical analysis, we derive the best performance bound of the RT method in terms of the number of initial blocks of a partial image, the number of processors, the image sizes, and the characteristics of parallel machines. In the experimental test, we implemented these three image compositing methods on an IBM SP2 parallel machine and a PC cluster. The experimental results show that the RT method outperforms the BS and the PP methods for all test samples and match the results analyzed in the theoretical analysis.

Keywords: image compositing, sort-last parallel volume rendering system, binary-swap, parallel-pipelined, rotate-tiling, distributed memory multicomputers

1. INTRODUCTION

Volume rendering [3, 4, 8, 17, 19] can be used to analyze the shape and volumetric property of three-dimensional objects in research areas such as medical imaging and scientific visualizing. However, most volume rendering methods that produce effective

Received April 25, 2003; revised August 5, 2003; accepted September 12, 2003.
Communicated by H. Y. Mark Liao.
* This work was partially supported by the NSC of ROC under contract NSC89-2213-E-035-032.
A preliminary version of this work was appeared in *Proceedings of IEEE International Parallel and Distributed Processing Symposium 2001*.

visualizations are computation intensive [14]. It is difficult for them to achieve interactive rendering rates for large volume data. In addition, volume data is too large to be stored in the memory of a single processor. One way to solve the above problems is to parallelize the volume rendering techniques on distributed memory multicomputers [20-23].

A sort-last parallel volume rendering system on distributed memory multicomputers [17] consists of three stages, the data partitioning stage, the data rendering stage, and the image compositing stage. In the data partitioning stage, a volume data is partitioned into sub-volumes by an efficient data partitioning method and the sub-volumes are distributed to processors [6]. In the data rendering stage, each processor uses a volume rendering algorithm on the assigned sub-volume to generate a partial image. In the image compositing stage, the partial images generated by processors are composited to form the final image. When the number of processors is large, the image compositing stage becomes a bottleneck of a sort-last parallel volume rendering system. The reason is that the partial images in processors are required a considerable amount of time to composite when the number of processors is large. Hence, a good image compositing method is very important to the performance for sort-last parallel volume rendering systems.

Many techniques to improve the performance for image compositing of the sort-last parallel volume rendering system have been proposed in the literatures [1, 2, 13, 18, 24]. In general, they can be classified into the following three categories:

- Efficient Data Communication Scheme: Methods in this category try to minimize the data communication overheads by using efficient data communication schemes to send and receive the partial images of processors [13, 18].
- Efficient Data Compression Scheme: Methods in this category try to reduce the communication data sizes by using some efficient data compression schemes [1, 24].
- Hybrid: Methods in this category try to minimize the data communication overheads and reduce the communication data sizes simultaneously [2].

In this paper, we focus on finding an efficient data communication scheme for image compositing to minimize data communication overheads. The binary-swap (BS) [18] and the parallel-pipelined (PP) [13] methods are two well-known data communication schemes for image compositing on a sort-last parallel volume rendering system. The BS method is a divide-and-conquer algorithm. In each communication step, the partial image in each processor is first divided into two equal halves. Two processors are then paired to exchange and composite half of their partial images. After $\log P$ communication steps, each processor contains a portion of the final image, where P is the number of processors. The final image then can be obtained by gathering each potion of the final image from processors. An example of the BS method is shown in Fig. 1.

The PP method was designed for the mesh network. Given $P = P_{row} \times P_{col}$ processors, there are two stages for image compositing in the PP method. In the first stage, the data communication topology of the row processors is treated as a ring. Initially, the partial image in each processor is divided into P_{row} blocks. There are $P_{row} - 1$ row communication steps. In each row communication step, for processors in the same row, a processor sends one block to its next processor and receives a block from its previous processor. The block received by each processor is then composited using the "over" opera-

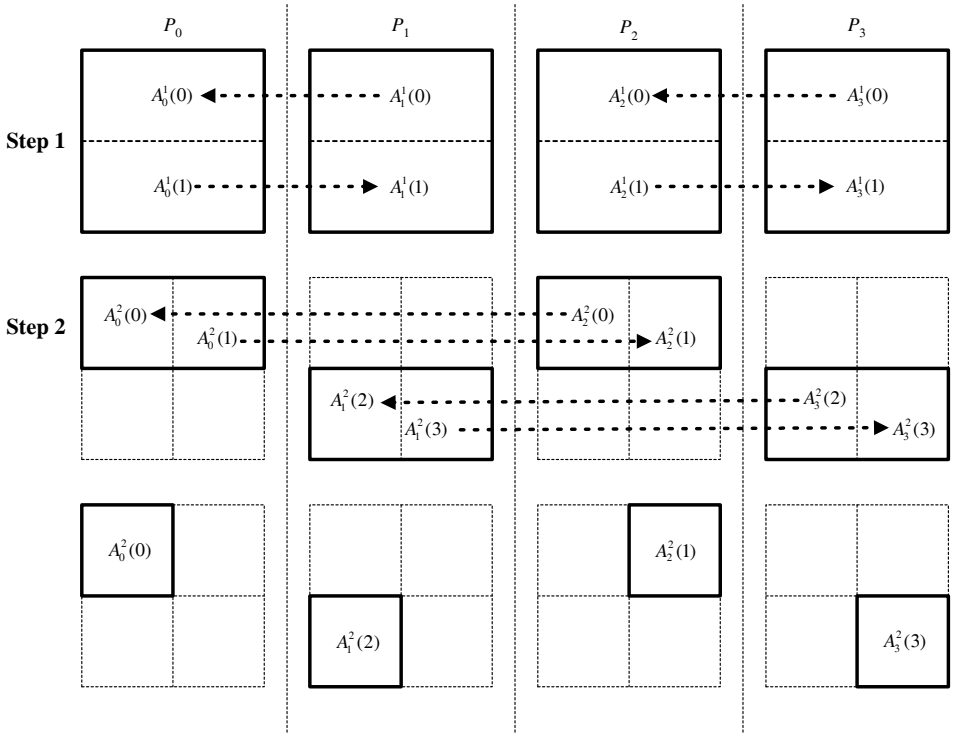


Fig. 1. An example of the BS method by using four processors.

tion. After the first stage, each processor holds a block that contains the accumulated results along the entire column. In the second stage, the data communication topology of the column processors is treated as a ring. Initially, the block, which produced in the first stage, in each processor is divided into P_{col} sub-blocks. There are $P_{col} - 1$ column communication steps. The data communication process of the second stage is similar as that of the first stage. After $(P_{row} + P_{col} - 2)$ communication steps, each processor hold one sub-block of the final image. The final image can be then obtained by gathering each sub-block from processors. An example of the PP method is given in Fig. 2.

The advantage of the BS method is that it enables more parallelism in the image compositing stage and keeps all processors busy in all communication steps. However, this method can only be applied to the case where the number of processors is a power-of-two. The advantage of the PP method is that it can be implemented with arbitrary number of processors. The disadvantage is that each processor needs $(P_{row} + P_{col} - 2)$ communication steps to perform image compositing. When $(P_{row} + P_{col} - 2)$ is large, the data communication overhead is high. To overcome the drawbacks described above, we present an efficient data communication scheme, the rotate-tiling (RT) method, which can be used with arbitrary number of processors and can minimize the data communication overheads.

Given P processors, in the RT method, the partial image in each processor can be divided into N blocks initially, where N is an arbitrary positive integer. In each communi-

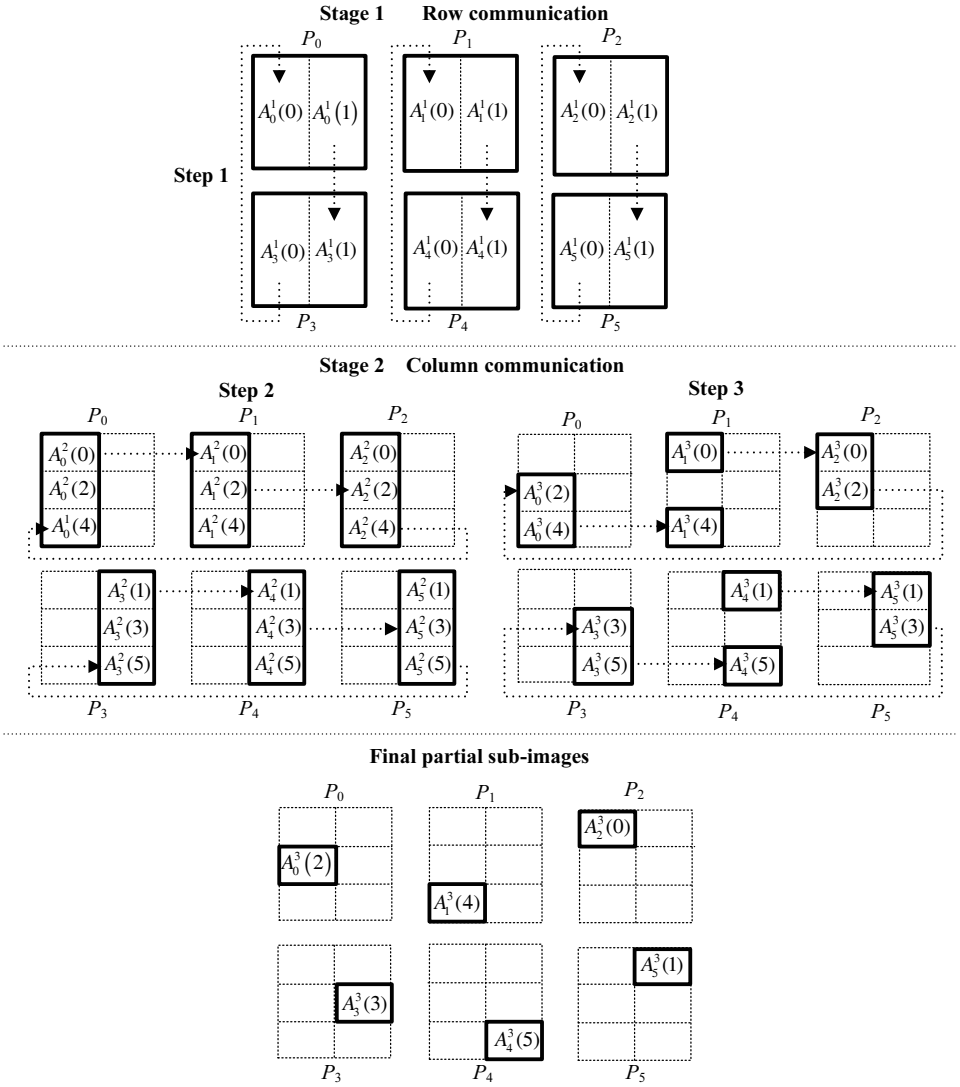


Fig. 2. An example of the PP method by using 2×3 processors.

cation step, a processor sends/receives some blocks to/from other processors according to the send/receive equations. After performing the send/receive operations, each processor uses the "over" operation to composite blocks it received. Each block in a processor is then divided into two equal halves. Continuing the above process, after $(\lceil \log P \rceil - 1)$ communication steps, each processor contains a portion of the final image. The final image then can be obtained by gathering each portion of the final image from processors.

In a data communication scheme, the number of communication steps and block sizes that are sent/received in each communication step affect the total image compositing time. The smaller the number of communication steps, the less the communication

startup time. The smaller the block sizes, the less the data transmission time. However, the smaller the block sizes, the higher the communication steps. Therefore, it is a trade-off for the block sizes and the number of communication steps. The BS and the PP methods use a fixed way to set these two parameters. In the RT method, different values of N lead to different performance. To analyze the effect of N in the RT method, the image compositing time is parameterized by the number of initial blocks of a partial image, the number of processors, the image sizes, and the characteristics of parallel machines. By fixing the number of processors, the image sizes, and the characteristics of parallel machines, we can determine the value of N that leads to the best performance.

To evaluate the performance of the RT method, we compare the proposed method with the BS and the PP methods. In the theoretical analysis, we analyze the theoretical performance of these three methods, and find the best performance bound of the RT method. In the experimental test, we implemented these three methods on an SP2 parallel machine and a PC cluster. The experimental results show that the RT method outperforms the BS and the PP methods for all test samples.

A preliminary version of this work was appeared in [16]. There are some differences between [16] and this paper.

- In [16], we only discuss the 1D version of the PP method. In this paper, we discuss the 2D version of the PP method.
- In [16], the RT method is divided into the $2N$_RT and the N_RT methods. In this paper, P and N can be arbitrary positive integers, that is, the RT method can handle the case where both P and N are odd positive integers as well.
- In [16], we did not compare the theoretical performance of the BS, the PP, and the RT methods. In this paper, we have compared the theoretical performance of the BS, the PP, and the RT method. We also have derived the best theoretical performance bound of N for the RT method.
- In [16], for the experimental tests, we only show the results of the BS, the PP (the 1D version), and the RT methods for different numbers of N with 32 processors on an IBM SP2 machine. In this paper, we have conducted the experimental tests on an IBM SP2 machine and a PC cluster. We show that the theoretical analysis matches the experimental results for the BS, the PP (the 2D version), and the RT methods. We also added the experimental results for the BS, the PP, and the RT methods with various numbers of processors and image sizes.

The rest of this paper is organized as follows. In section 2, we present the RT method for image compositing of a sort-last parallel volume rendering system in detail. We then analyze the theoretical performance of the RT method along with the BS and the PP methods in section 3. In this section, we also describe how to find the best performance bound of the RT method. In section 4, the experimental results of the BS, the PP, and the RT methods on an IBM SP2 parallel machine and a PC cluster will be given.

2. THE ROTATE-TILING METHOD

To combine the advantages of the BS and the PP methods, in the RT method, we

derive the send/receive equations based on the indexing operations of the BS method and the ring rotation topology of the PP method. In the RT method, the number of processors (P) and the number of initial blocks of a partial image (N) can be arbitrary positive integers ($P > 0$ and $N > 0$). To composite partial images of processors, the RT method consists of the following steps:

Step 1: The partial image in each processor initially is divided into N equal blocks and the blocks are numbered from 0 to $N - 1$. We use $A_r^k(b)$ to represent the block of processor P_r with block number b in the kth communication step, where $0 \le b \le N - 1$, $0 \le r \le P - 1$, and $1 \le k \le \lceil \log P \rceil$.

Step 2: For each block number b, those processors that own block b form a ring, where $0 \le b \le N - 1$. Let R_b^k denote the ring of block b in the kth communication step, $Q = |R_b^k|$ denote the number of processors in R_b^k, and $R_b^k(x)$ denote the xth processor in R_b^k, where $0 \le x \le Q - 1$. Note that the rank of $R_b^k(x)$ is less than that of $R_b^k(x+1)$.

Step 3: For each R_b^k, starting from processor $R_b^k(x \bmod Q)$, $R_b^k(x \bmod Q)$ receives a block from $R_b^k((x+1) \bmod Q)$, $R_b^k((x+2) \bmod Q)$ receives a block from $R_b^k((x + 3) \bmod Q)$, …, and $R_b^k((x+2\times(\lfloor Q/2 \rfloor - 1)) \bmod Q)$ receives a block from $R_b^k((x+2\times(\lfloor Q/2 \rfloor - 1)+1) \bmod Q)$.

Step 4: Each processor uses the "over" operation to composite blocks it received.

Step 5: For each R_b^k, remove the processors that sent blocks to others from the ring.

Step 6: For each block $A_r^k(b)$ in R_b^k, partition $A_r^k(b)$ into two equal halves and the two equal halves are numbered as $A_r^{k+1}(2b)$ and $A_r^{k+1}(2b+1)$, respectively.

Step 7: Continue Steps 2-6 ($\lceil \log P \rceil - 1$) times, the final image can be obtained.

An example is given in Fig. 3 to explain the above steps. In Fig. 3, $P = 3$ and $N = 4$. Fig. 3 (a) shows the initial status (Step 1). Fig. 3 (b) shows the ring for each block number and the send/receive processor pairs (Steps 2-4). Fig. 3 (c) shows the removal of sent blocks from rings and the partitioning of each remained blocks in rings into two equal halves (Steps 5-6). Fig. 3 (d) shows the second communication step (Steps 2-6). Fig. 3 (e) shows the final image obtained by the RT method.

In real implementation, the send/receive processor pairs in rings can be determined by the following send/receive equations. We assume that, in the kth communication step, P_r sends block $A_r^k(m)$ to P_i and receives block $A_j^k(n)$ from P_j, where r, i, and j are processor ranks; k is a positive integer; and m and n are block numbers ($0 \le m, n \le 2^{k-1}N - 1$). The send equation for the RT method is given below,

$$P_r(A_r^k(m)) \rightarrow P_i, \text{ where } \begin{cases} l = 0,1,\ldots,k-1 \\ w = 0,1,\ldots,\lceil P/N \rceil - 1 \\ v = 0,1,\ldots,\lceil P/2^k \rceil \\ m = ((r - 2^{k-1} + 2\times l) \bmod 2^k) \times 2^{k-1} + v \times 2^{2k-1} + l + P \times w \\ i = (r - 2^{k-1} + l) \bmod P. \end{cases} \tag{1}$$

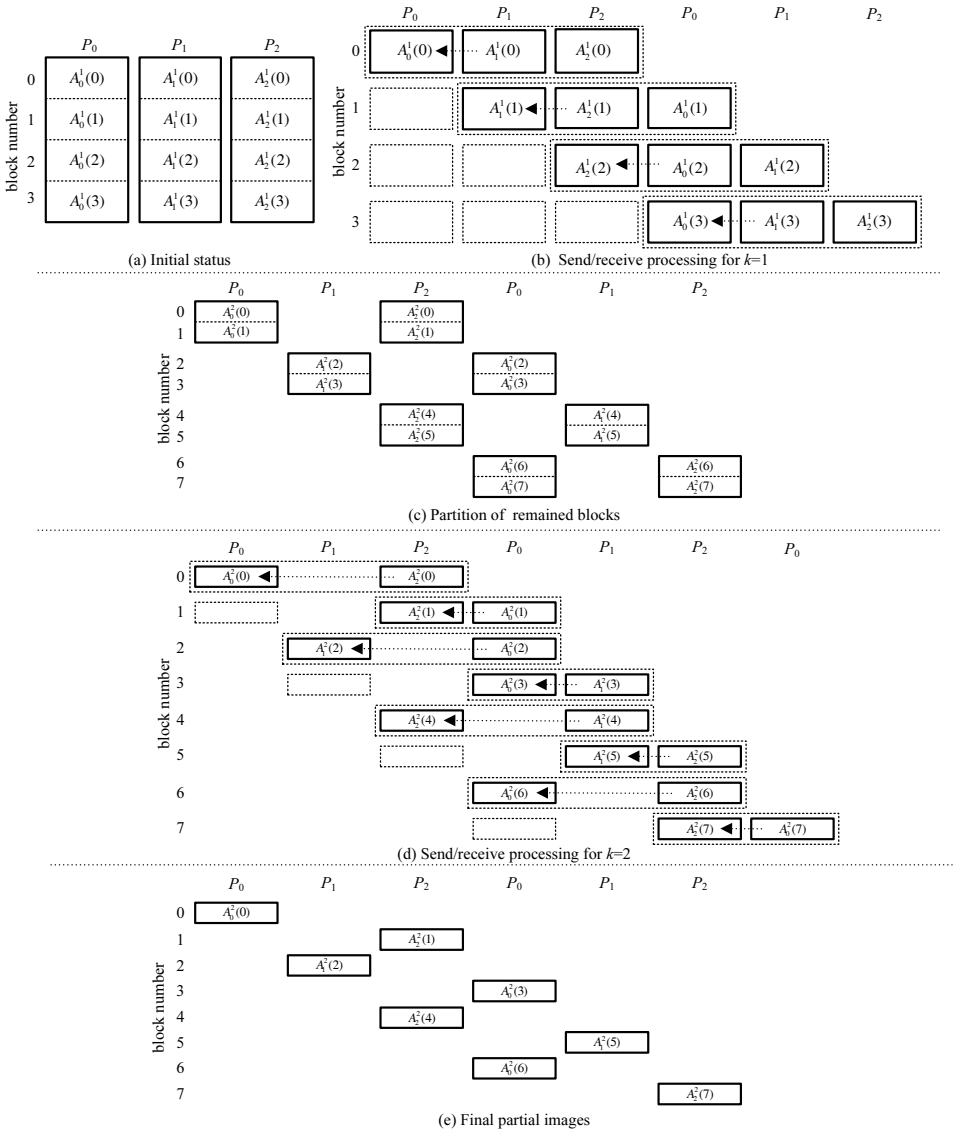


Fig. 3. The behavior of the RT method by using $P = 3$ and $N = 4$.

The receive equation for the RT method is given below,

$$P_r \leftarrow P_j(A_j^k(n)), \text{ where } \begin{cases} l = 0,1,\ldots,k-1 \\ w = 0,1,\ldots,\lceil P/N \rceil - 1 \\ v = 0,1,\ldots,\left\lceil P/2^k \right\rceil \\ m = ((r+l) \bmod 2^k) \times 2^{k-1} + v \times 2^{2k-1} + l + P \times w \\ j = (r + 2^{k-1} - l) \bmod P. \end{cases} \quad (2)$$

According to Eqs. (1) and (2), a processor can determine the send/receive blocks without the construction of the rings of blocks. The algorithm of the RT method is given as follows.

Algorithm RT_Method(P, A, N)
/* P is the number of processors. */
/* A is the partial image size of each processor. */
/* N is the number of initial block of a partial image. */
1. Each processor partitions its partial image A into N blocks, $A_r^1(0), \ldots, A_r^1(N-1)$;
2. **for** $k = 1$ to $\lceil \log P \rceil$ **do** {
3. **for** each processor P_r **do parallel** {
4. P_r sends block $A_r^k(m)$ to P_i according to Eq. (1);
5. P_r receives block $A_j^k(n)$ from P_j according to Eq. (2);
6. P_r composites the received $A_j^k(n)$ with its local block $A_r^k(n)$;
7. For each block $A_r^k(b)$ in P_r, divide $A_r^k(b)$ into two equal sub-blocks $A_r^{k+1}(2b)$ and $A_r^{k+1}(2b+1)$;
8. }
9. }
end_of_RT_Method

---

3. THEORETICAL ANALYSIS OF THE BS, THE PP, AND THE RT METHODS

In this section, we first derive the theoretical performance of the BS, the PP, and the RT methods. We then analyze the effect of the number of initial blocks of a partial image for the RT method. A summary of the notations used in the theoretical analysis is given below.

- P – The number of processors ($P > 0$).
- P_{row} – The number of row processors on a mesh network for the PP method.
- P_{col} – The number of column processors on a mesh network for the PP method.
- P_i – The processor with rank i, where $i = 0, 1, 2, \ldots, P-1$.
- A – The image size in pixels.
- N – The number of initial blocks of a partial image in the RT method ($N > 0$).
- $S(M)$ – The number of communication steps of method M.
- T_s – The startup time of a communication channel.
- T_p – The data transmission time per byte.
- T_o – The computation time of the "over" operation per pixel.
- $T_{total}(M)$ – The total image compositing time of method M.
- $T_{comp}^k(M, P_i)$ – The data communication time of P_i in the kth communication step of method M.
- $T_{comp}^k(M, P_i)$ – The data computation time of P_i in the kth communication step of method M.

- $A_i^k(M, P_i)$ – The block size will be sent/received by P_i in the kth communication step of method M.

To analyze the theoretical performance of the BS, the PP, and the RT methods, in the cost model, a synchronous communication mode is used. In this model, all processors start their computation after each processor completes its communication. In real situation, an asynchronous communication mode can be applied as well. However, it is difficult to analyze the theoretical performance if an asynchronous communication mode is used. According to above notations, the cost model of method M is defined as

$$T_{total}(M) = \sum_{k=1}^{S(M)} \max\{T_{comm}^k(M, P_i) + T_{comp}^k(M, P_i)\}. \quad (3)$$

In our communication model, we assume that each processor can communicate with all other processors in one communication step. $T_{comp}^k(M, P_i)$ is defined as

$$T_{comm}^k(M, P_i) = \delta_i^k \times T_s + A_i^k(M, P_i) \times T_p, \quad (4)$$

where δ_i^k is the number of processors that P_i sends data to in the kth communication step. In our computation model, we assume that each pixel of a block received from another processor is composited using the "over" operation. $T_{comp}^k(M)$ is therefore defined as

$$T_{comp}^k(M, P_i) = A_i^k(M, P_i) \times T_o. \quad (5)$$

3.1 The Total Image Compositing Time of the BS, the PP, and RT Method

In the BS method, there are log P communication steps. In the kth communication step, the block size sent or received by a processor is $\frac{A}{2^k}$, where $k = 1, \ldots, \log P$. The data communication and computation time for each processor in the kth communication step are $T_s + \frac{A}{2^k} \times T_p$ and $\frac{A}{2^k} \times T_o$, respectively. We have $T_{total}(\text{BS}) = \sum_{k=1}^{\log P} (T_s + \frac{A}{2^k} T_p) + \sum_{k=1}^{\log P} \frac{A}{2^k} T_o$ $= (\log P)T_s + (1 - \frac{1}{2^{\log P}})(T_p + T_o)A$.

In the PP method, given $P = P_{row} \times P_{col}$ processors, there are $(P_{row} + P_{col} - 2)$ communication steps. For the first stage, there are $(P_{row} - 1)$ communication steps. The block size sent or received by a processor is $\frac{A}{P_{row}}$. The data communication and computation time for each processor are $T_s + \frac{A}{P_{row}} \times T_p$ and $\frac{A}{P_{row}} \times T_o$, respectively. For the second stage, there are $(P_{col} - 1)$ communication steps. The block size sent or received by a processor is $\frac{A}{P}$. The data communication and computation time for each processor in are $T_s + \frac{A}{P} \times T_p$ and $\frac{A}{P} \times T_o$, respectively. We have $T_{total}(\text{PP}) = \{ \sum_{k=1}^{P_{row}-1} (T_s + \frac{A}{P_{row}} T_p) + \sum_{k=1}^{P_{row}-1} \frac{A}{P_{row}} T_o \} + \{ \sum_{k=1}^{P_{col}-1} (T_s + \frac{A}{P} T_p) + \sum_{k=1}^{P_{col}-1} \frac{A}{P} T_o \} = (P_{row} + P_{col} - 2)T_s + (1 + \frac{P_{col} - 1}{P} - \frac{1}{P_{row}})(T_p + T_o)A$

$= (P_{row} + P_{col} - 2)T_s + (1 - \frac{1}{P})(T_p + T_o)A.$

In the RT method, there are $\lceil \log P \rceil$ communication steps. In the first communication step (k = 1), the maximum number of send/receive operations performed by processors is $\left\lceil \frac{N}{P} \right\rceil$. The block size in each sent or received by a processor is $\frac{A}{N}$, the maximum data communication and computation time among processors are $\left\lceil \frac{N}{P} \right\rceil \times T_s + \left\lceil \frac{N}{P} \right\rceil \frac{A}{N} \times T_p$ and $\left\lceil \frac{N}{P} \right\rceil \frac{A}{N} \times T_o$, respectively. In the kth communication step, where $k > 1$, the maximum number of send/receive operations performed by processors is $\left\lceil \frac{2B_k}{P} \right\rceil$, where

$$B_k = \begin{cases} N & \text{for } k = 1 \\ B_{k-1} - \lfloor B_{k-1} / P \rfloor & \text{for } k > 1 \end{cases} \quad (6)$$

The block size in each sent or received by a processor is $\frac{A}{2^{k-1}N}$, where k = 2, …, $\lceil \log P \rceil$. The maximum data communication and computation time among processors are $\left\lceil \frac{2B_k}{P} \right\rceil T_s + \left\lceil \frac{2B_k}{P} \right\rceil \frac{A}{N2^{k-1}} T_p$ and $\left\lceil \frac{2B_k}{P} \right\rceil \frac{A}{N2^{k-1}} T_o$, respectively. We have $T_{total}(\mathrm{RT}) = (\left\lceil \frac{N}{P} \right\rceil + \sum_{k=2}^{\lceil \log P \rceil} \left\lceil \frac{2B_k}{P} \right\rceil) T_s + (\left\lceil \frac{N}{P} \right\rceil \frac{A}{N} + \sum_{k=2}^{\lceil \log P \rceil} \left\lceil \frac{2B_k}{P} \right\rceil \frac{A}{N2^{k-1}}) T_p + (\left\lceil \frac{N}{P} \right\rceil \frac{A}{N} + \sum_{k=2}^{\lceil \log P \rceil} \left\lceil \frac{2B_k}{P} \right\rceil \frac{A}{N2^{k-1}}) T_o = (\sum_{k=1}^{\lceil \log P \rceil} \left\lceil \frac{2B_k}{P} \right\rceil + \left\lceil \frac{N}{P} \right\rceil - \left\lceil \frac{2N}{P} \right\rceil) T_s + \frac{A}{N} (\sum_{k=1}^{\lceil \log P \rceil} \left\lceil \frac{2B_k}{P} \right\rceil \frac{1}{2^{k-1}} + \left\lceil \frac{N}{P} \right\rceil - \left\lceil \frac{2N}{P} \right\rceil) (T_p + T_o)$. In here, we do not derive the upper bound of $T_{total}(\mathrm{RT})$. If we use the upper bound of $T_{total}(\mathrm{RT})$ instead of the equation, the best performance bound of $T_{total}(\mathrm{RT})$ derived in section 3.3 will not be correct.

3.2 Theoretical Performance Comparisons of These Three Image Compositing Methods

According to the above analysis, we summarize the theoretical performance of the BS, the PP, and the RT methods in Table 1.

Table 1. The theoretical performance for the BS, the PP, and the RT methods.

M	S(M)	A_i^k (M)	T_{total}(M)
BS	$\log P$	$\frac{A}{2^k}$	$(\log P)T_s + (1 - \frac{1}{2^{\log P}})(T_p + T_o)A$
PP	$P_{row} + P_{col} - 2$	$\frac{A}{P_{row}}$ or $\frac{A}{P}$	$(P_{row} + P_{col} - 2)T_s + (1 - \frac{1}{P})(T_p + T_o)A$
RT	$\lceil \log P \rceil$	$\left\lceil \frac{2B_k}{P} \right\rceil \frac{A}{N2^{k-1}}$	$(\sum_{k=1}^{\lceil \log P \rceil} \left\lceil \frac{2B_k}{P} \right\rceil + \left\lceil \frac{N}{P} \right\rceil - \left\lceil \frac{2N}{P} \right\rceil) T_s + \frac{A}{N} (\sum_{k=1}^{\lceil \log P \rceil} \left\lceil \frac{2B_k}{P} \right\rceil \frac{1}{2^{k-1}} + \left\lceil \frac{N}{P} \right\rceil - \left\lceil \frac{2N}{P} \right\rceil) (T_p + T_o)$

For the PP method, the best performance of T_{total}(PP) is $(2\sqrt{P}-2)T_s+(1-\frac{1}{P})(T_p + T_o)A$, that is, when $P_{row} = P_{col}$. The reason is that the value of $(P_{row}+P_{col}-2)T_s+(1-\frac{1}{P})(T_p + T_o)A$ is the smallest when $P_{row} = P_{col} = \sqrt{P}$. In the following analysis, for the PP method, the best performance of T_{total}(PP) is used. From Table 1, for the BS and the PP methods, we have $T_{total}(\text{BS}) \le T_{total}(\text{PP})$ when the number of processors is a power-of-two. For the BS and the RT methods, if $T_{total}(\text{RT}) < T_{total}(\text{BS})$, we have

$$(\sum_{k=1}^{\lceil \log P \rceil}\left\lceil \frac{2B_k}{P} \right\rceil+\left\lceil \frac{N}{P} \right\rceil-\left\lceil \frac{2N}{P} \right\rceil)T_s+\frac{A}{N}(\sum_{k=1}^{\lceil \log P \rceil}\left\lceil \frac{2B_k}{P} \right\rceil\frac{1}{2^{k-1}}+\left\lceil \frac{N}{P} \right\rceil-\left\lceil \frac{2N}{P} \right\rceil)(T_p+T_o) < (\log P)T_s +$$

$$(1-\frac{1}{2^{\log P}})(T_p+T_o)A \Rightarrow \frac{\sum_{k=1}^{\lceil \log P \rceil}\left\lceil \frac{2B_k}{P} \right\rceil+\left\lceil \frac{N}{P} \right\rceil-\left\lceil \frac{2N}{P} \right\rceil-\log P}{(1-\frac{1}{2^{\log P}})-\frac{1}{N}(\sum_{k=1}^{\lceil \log P \rceil}\left\lceil \frac{2B_k}{P} \right\rceil\frac{1}{2^{k-1}}+\left\lceil \frac{N}{P} \right\rceil-\left\lceil \frac{2N}{P} \right\rceil)} < \frac{(T_p+T_o)A}{T_s}. \quad (7)$$

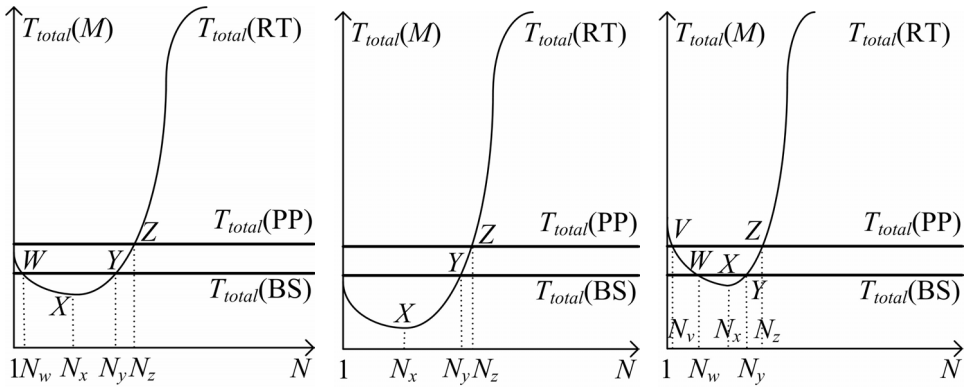
For the PP and the RT methods, if $T_{total}(\text{RT}) < T_{total}(\text{PP})$, we have

$$(\sum_{k=1}^{\lceil \log P \rceil}\left\lceil \frac{2B_k}{P} \right\rceil+\left\lceil \frac{N}{P} \right\rceil-\left\lceil \frac{2N}{P} \right\rceil)T_s+\frac{A}{N}(\sum_{k=1}^{\lceil \log P \rceil}\left\lceil \frac{2B_k}{P} \right\rceil\frac{1}{2^{k-1}}+\left\lceil \frac{N}{P} \right\rceil-\left\lceil \frac{2N}{P} \right\rceil)(T_p+T_o) < (2\sqrt{P}-2)T_s$$

$$+(1-\frac{1}{P})(T_p+T_o)A \Rightarrow \frac{\sum_{k=1}^{\lceil \log P \rceil}\left\lceil \frac{2B_k}{P} \right\rceil+\left\lceil \frac{N}{P} \right\rceil-\left\lceil \frac{2N}{P} \right\rceil-2\sqrt{P}+2}{(1-\frac{1}{P})-\frac{1}{N}(\sum_{k=1}^{\lceil \log P \rceil}\left\lceil \frac{2B_k}{P} \right\rceil\frac{1}{2^{k-1}}+\left\lceil \frac{N}{P} \right\rceil-\left\lceil \frac{2N}{P} \right\rceil)} < \frac{(T_p+T_o)A}{T_s}. \quad (8)$$

Given a parallel machine, an image size, and the number of processors, the values of P, A, T_s, T_p, and T_o are fixed. The values of the right side of Eqs. (7) and (8), that is, $\frac{(T_p+T_o)A}{T_s}$, are constants. Let $U = \frac{(T_p+T_o)A}{T_s}$, $K = \frac{\sum_{k=1}^{\lceil \log P \rceil}\left\lceil \frac{2B_k}{P} \right\rceil+\left\lceil \frac{N}{P} \right\rceil-\left\lceil \frac{2N}{P} \right\rceil-\log P}{(1-\frac{1}{2^{\log P}})-\frac{1}{N}(\sum_{k=1}^{\lceil \log P \rceil}\left\lceil \frac{2B_k}{P} \right\rceil\frac{1}{2^{k-1}}+\left\lceil \frac{N}{P} \right\rceil-\left\lceil \frac{2N}{P} \right\rceil)}$, and $L = \frac{\sum_{k=1}^{\lceil \log P \rceil}\left\lceil \frac{2B_k}{P} \right\rceil+\left\lceil \frac{N}{P} \right\rceil-\left\lceil \frac{2N}{P} \right\rceil-2\sqrt{P}+2}{(1-\frac{1}{P})-\frac{1}{N}(\sum_{k=1}^{\lceil \log P \rceil}\left\lceil \frac{2B_k}{P} \right\rceil\frac{1}{2^{k-1}}+\left\lceil \frac{N}{P} \right\rceil-\left\lceil \frac{2N}{P} \right\rceil)}$. According to Eqs. (7) and (8), we can see that N determines the results of these two inequalities. Since the values of P, A, T_s, T_p, and T_o are fixed, T_{total}(BS) and T_{total}(PP) are constants; and T_{total}(RT) depends on N. The theoretical performance of the BS, the PP, and the RT methods can be drawn as Fig. 4 in terms of the values of P, A, T_s, T_p, and T_o. In Fig. 4, we show the case where P is a power-of-two. From Fig. 4 (a), we can see that T_{total}(BS) and T_{total}(PP) are constants. T_{total}(RT) depends on N and forms a curve. If $K = U$, the intersections of the curve of T_{total}(RT) and the line of T_{total}(BS) are W and Y. It means that T_{total}(RT) = T_{total}(BS) when $K =$

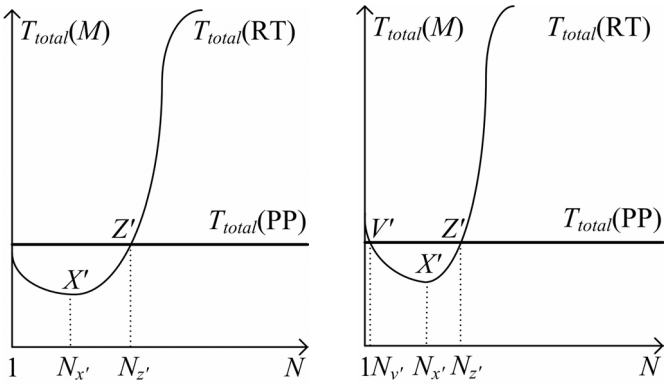
U, that is, $N = N_w$ or $N = N_y$. If $N_w < N < N_y$, we have T_{total}(RT) < T_{total}(BS). If $L = U$, the intersection of the curve of T_{total}(RT) and the line of T_{total}(PP) is Z. It means that T_{total}(RT) = T_{total}(PP) when $L = U$, that is, $N = N_z$. If $N < N_z$, we have T_{total}(RT) < T_{total}(PP). From Fig. 4 (b) and Fig. 4 (c), we have similar observations as those of Fig. 4 (a).



(a) Case 1. (b) Case 2. (c) Case 3.

Fig. 4. Theoretical performance for the three image compositing methods when P is a power-of-two.

For the case where P is not a power-of-two is shown in Fig. 5. Since the BS method can not be applied in this case, only the line of T_{total}(PP) and the curve of T_{total}(RT) are shown. From Fig. 5 (a), we can see that if $L = U$, the curve of T_{total}(RT) intersects the line of T_{total}(PP) at point Z'. It means that T_{total}(RT) = T_{total}(PP) when $L = U$, that is, $N = N_{z'}$. If $N < N_{z'}$, we have T_{total}(RT) < T_{total}(PP). From Fig. 5 (b), we have similar observations as those of Fig. 5 (a). From Fig. 4 and Fig. 5, we can see that the curve of T_{total}(RT) has a lowest point X (or X') in Fig. 4 (or Fig. 5). It means that $N = N_x$ (or $N = N_{x'}$) is the best performance bound of the RT method for Fig. 4 (or Fig. 5). In the following, we will derive the best performance bound of the RT method in terms of the values of P, A, T_s, T_p, and T_o.



(a) Case 1. (b) Case 2.

Fig. 5. Theoretical performance for the three image compositing methods when P is not a power-of-two.

3.3 The Best Performance Bound of *N* for the RT Method

The total image compositing time of the RT method T_{total}(RT) is $(\sum_{k=1}^{\lceil \log P \rceil} \left\lceil \frac{2B_k}{P} \right\rceil + \left\lceil \frac{N}{P} \right\rceil - \left\lceil \frac{2N}{P} \right\rceil) T_s + \frac{A}{N}(\sum_{k=1}^{\lceil \log P \rceil} \left\lceil \frac{2B_k}{P} \right\rceil \frac{1}{2^{k-1}} + \left\lceil \frac{N}{P} \right\rceil - \left\lceil \frac{2N}{P} \right\rceil)(T_p + T_o)$ that is parameterized by P, N, A, T_s, T_p, and T_o. To find the best performance bound of the RT method, we compare the image compositing time when the number of initial blocks of a partial image is N and $N + 1$, respectively. Let $T_{RT}(N)$ and $T_{RT}(N + 1)$ represent the image compositing time when the number of initial blocks of a partial image is N and $N + 1$, respectively. We have $T_{RT}(N) = (\sum_{k=1}^{\lceil \log P \rceil} \left\lceil \frac{2B_k}{P} \right\rceil + \left\lceil \frac{N}{P} \right\rceil - \left\lceil \frac{2N}{P} \right\rceil) T_s + \frac{A}{N}(\sum_{k=1}^{\lceil \log P \rceil} \left\lceil \frac{2B_k}{P} \right\rceil \frac{1}{2^{k-1}} + \left\lceil \frac{N}{P} \right\rceil - \left\lceil \frac{2N}{P} \right\rceil)(T_p + T_o)$ and $T_{RT}(N + 1) = (\sum_{k'=1}^{\lceil \log P \rceil} \left\lceil \frac{2B_{k'}}{P} \right\rceil + \left\lceil \frac{N+1}{P} \right\rceil - \left\lceil \frac{2N+2}{P} \right\rceil) T_s + \frac{A}{N+1}(\sum_{k'=1}^{\lceil \log P \rceil} \left\lceil \frac{2B_{k'}}{P} \right\rceil \frac{1}{2^{k'-1}} + \left\lceil \frac{N+1}{P} \right\rceil - \left\lceil \frac{2N+2}{P} \right\rceil)(T_p + T_o)$.

The difference of $T_{RT}(N)$ and $T_{RT}(N + 1)$is $\Delta T = T_{RT}(N + 1) - T_{RT}(N) = (\sum_{k'=1}^{\lceil \log P \rceil} \left\lceil \frac{2B_{k'}}{P} \right\rceil + \left\lceil \frac{N+1}{P} \right\rceil - \left\lceil \frac{2N+2}{P} \right\rceil) T_s + \frac{A}{N+1}(\sum_{k'=1}^{\lceil \log P \rceil} \left\lceil \frac{2B_{k'}}{P} \right\rceil \frac{1}{2^{k'-1}} + \left\lceil \frac{N+1}{P} \right\rceil - \left\lceil \frac{2N+2}{P} \right\rceil)(T_p + T_o) - (\sum_{k=1}^{\lceil \log P \rceil} \left\lceil \frac{2B_k}{P} \right\rceil + \left\lceil \frac{N}{P} \right\rceil - \left\lceil \frac{2N}{P} \right\rceil) T_s - \frac{A}{N}(\sum_{k=1}^{\lceil \log P \rceil} \left\lceil \frac{2B_k}{P} \right\rceil \frac{1}{2^{k-1}} + \left\lceil \frac{N}{P} \right\rceil - \left\lceil \frac{2N}{P} \right\rceil)(T_p + T_o)$. By setting $\Delta T = 0$, we have

$$\frac{(\sum_{k'=1}^{\lceil \log P \rceil} \left\lceil \frac{2B_{k'}}{P} \right\rceil + \left\lceil \frac{N+1}{P} \right\rceil - \left\lceil \frac{2N+2}{P} \right\rceil) - (\sum_{k=1}^{\lceil \log P \rceil} \left\lceil \frac{2B_k}{P} \right\rceil + \left\lceil \frac{N}{P} \right\rceil - \left\lceil \frac{2N}{P} \right\rceil)}{\frac{1}{N}(\sum_{k=1}^{\lceil \log P \rceil} \left\lceil \frac{2B_k}{P} \right\rceil \frac{1}{2^{k-1}} + \left\lceil \frac{N}{P} \right\rceil - \left\lceil \frac{2N}{P} \right\rceil) - \frac{1}{N+1}(\sum_{k'=1}^{\lceil \log P \rceil} \left\lceil \frac{2B_{k'}}{P} \right\rceil \frac{1}{2^{k'-1}} + \left\lceil \frac{N+1}{P} \right\rceil - \left\lceil \frac{2N+2}{P} \right\rceil)} = \frac{(T_p + T_o)A}{T_s}. \quad (9)$$

According to Eq. (9), by setting the values of P, A, T_s, T_p, and T_o, we can determine the value of N that leads to the best performance bound of the RT method. For example, assume that the values of P, A, T_s, T_p, and T_o are 32, 512^2, 5×10^{-6}, 4×10^{-8}, and 2×10^{-7}, respectively. According to Eq. (9), the value of N is 3.5. It means that when N is equal to 3.5, the RT method can produce the best performance than other values of N. In the RT method, N must be an arbitrary positive integer. For the given example, N is equal to 3 or 4.

4. EXPERIMENTAL RESULTS

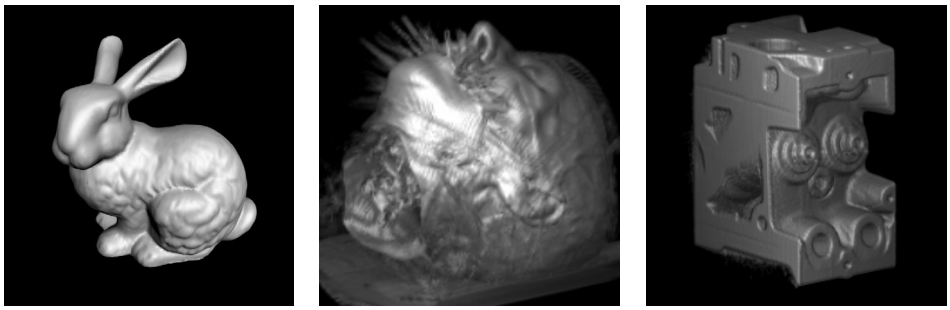
To evaluate the performance of the RT method, we implemented the RT method along with the BS and the PP methods on an IBM SP2 parallel machine [7] and a PC cluster. The IBM SP2 parallel machine is located at National Center of High Performance Computing (NCHC) in Taiwan. This super-scalar architecture uses an IBM RISC System/6000 POWER2 CPU with a clock rate of 66.7 MHz. There are 40 IBM POWER2 nodes in this system, and each node has a 128KB first-level data cache, a 32KB first-level instruction cache, and 128MB of memory space. Each node is connected to a

low-latency, high-bandwidth interconnection network called High Performance Switch (HPS). The startup time of a communication channel, data transmission time per byte, and computation time of the "over" operation per pixel are 5×10^{-6}, 4×10^{-8}, and 2×10^{-7} second, respectively. The PC cluster is located at Parallel and Distributed System Laboratory at Feng Chia University in Taiwan. Each node in the PC cluster uses an INTEL Pentium III CPU with a clock rate of 800 MHz. There are 32 CPUs in this system, and each node has 512KB first-level data cache and 256MB of memory space. Each node is fully connected by Myrinet. The startup time of a communication channel, data transmission time per byte, and computation time of the "over" operation per pixel are 3.5×10^{-7}, 8.4×10^{-9}, 3.3×10^{-8} second, respectively.

A sort-last parallel volume rendering system consists of three stages, the data partitioning, the data rendering, and the image compositing stages. To implement these data communication schemes, in the data partitioning stage, we use the efficient 2-D partitioning schemes [15] to distribute volume data to processors. In the data rendering stage, each processor uses the shear-warp factorization [9-12] volume rendering method to generate a partial image. In the image compositing stage, the partial images are composited using the BS, the PP, and the RT methods to form a final image. On the IBM SP2 machine, we implement the BS, the PP, and the RT methods using C and MPICH 0 message passing libraries. In the PC cluster, we implement the BS, the PP, and the RT methods using C and MPICH_GM message passing libraries.

Three volume datasets are used to evaluate the performance of these composition methods. The first test sample is a "*Bunny*" dataset generated from the CT scan of a bunny, and the dimensions of the dataset is $512 \times 512 \times 362$. The second test sample is a "*Head*" dataset generated from the CT scan of a human head, and the dimensions of the dataset is $256 \times 256 \times 225$. The third test sample is an "*Engine*" dataset, which is the CT scan of an engine block and the dimensions of the dataset is $256 \times 256 \times 110$. Fig. 6 shows the final images of the three test samples. Each image is grayscale color and contains 512×512 pixels.

There are four major factors affecting the performance of the RT method, the number of initial blocks of a partial image, the number of processors, the final image sizes, and the characteristics of parallel machines. In the following, we compare the performance of the BS, the PP, and the RT methods in terms of these four factors.



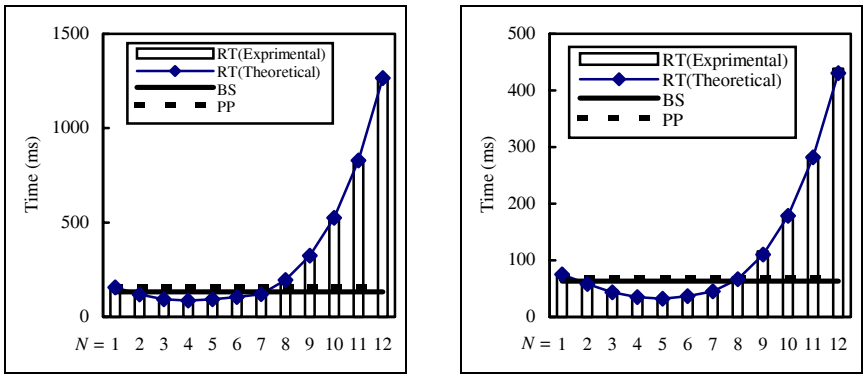
(a) *Bunny*. (b) *Head*. (c) *Engine*.

Fig. 6. The final images for the three test samples.

4.1 Performance for Various Numbers of Initial Blocks of a Partial Image

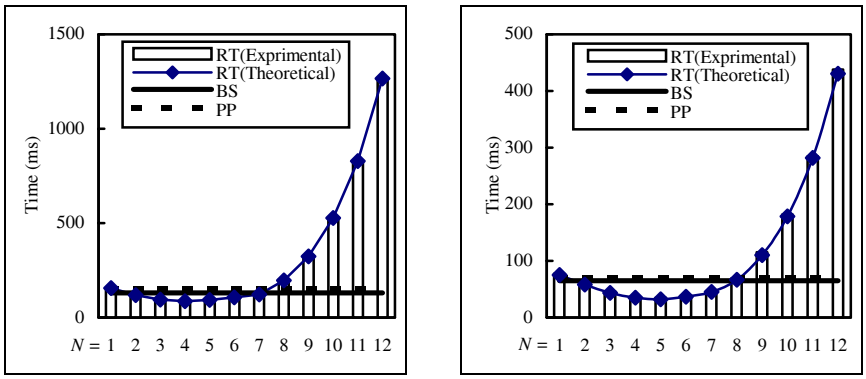
Fig. 7 (a) shows the theoretical and experimental image compositing time of the BS, the PP, and the RT methods for test sample "*Bunny*" with various numbers of initial blocks of a partial image on a 32-node IBM SP2 machine. In this case, the values of P, A, T_s, T_p, and T_o are 32, 512^2, 5×10^{-6}, 4×10^{-8}, and 2×10^{-7}, respectively. According to Eq. (9), when N is equal to 3.5, the RT method can produce the best theoretical performance for the test sample. In Fig. 7 (a), when the number of initial blocks is equal to 4, the RT method can produce the best experiment performance for the test sample. The experimental result therefore matches the theoretical analysis for the test sample.

Fig. 7 (b) shows the theoretical and experimental image compositing time of the three image compositing methods for test sample "*Bunny*" with various numbers of initial blocks of a partial image on a 32-node PC cluster. In this case, the values of P, A, T_s, T_p,



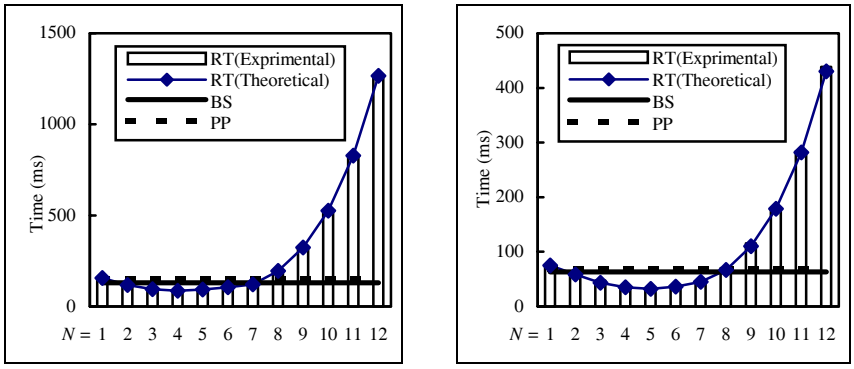
(a) On IBM SP2. (b) On PC cluster.

Fig. 7. The theoretical and experimental image compositing time of the BS, the PP, and the RT methods for test sample "*Bunny*" with various numbers of initial blocks of a partial image.



(a) On IBM SP2. (b) On PC cluster.

Fig. 8. The theoretical and experimental image compositing time of the BS, the PP, and the RT methods for test sample "*Head*" with various numbers of initial blocks of a partial image.



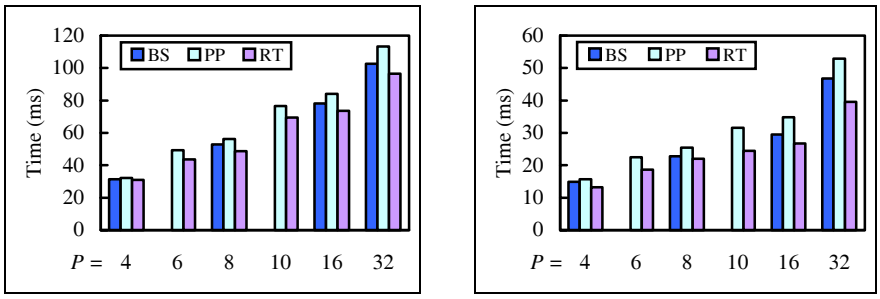
(a) On IBM SP2. (b) On PC cluster.

Fig. 9. The theoretical and experimental image compositing time of the BS, the PP, and the RT methods for test sample "*Engine*" with various numbers of initial blocks of a partial image.

and T_o are 32, 512^2, 3.5×10^{-7}, 8.4×10^{-9}, and 3.3×10^{-8}, respectively. According to Eq. (9), when N is equal to 4.6, the RT method can produce the best theoretical performance for the test sample. In Fig. 7 (b), when the number of initial blocks is equal to 5, the RT method can produce the best experiment performance for the test sample. The experimental result therefore matches the theoretical analysis for the test sample.

4.2 Performance Comparisons with Various Numbers of Processors

Fig. 10 (a) shows the image compositing time of the three image compositing methods for test sample "*Bunny*" with various numbers of processors on an IBM SP2 machine. In this case, the image size is 512^2 pixels. The number of processors P is set from 4 to 32. In Fig. 10 (a), for each different number of processors, we only show the best performance of the PP and the RT methods. Since the BS method can only be applied to the case where the number of processors is a power-of-two, for the case where P is not a power-of-two, only the best performance of the PP and the RT methods are shown. From Fig. 10 (a), we can see that the image compositing time of the RT method is better than that of the BS and the PP methods.



(a) On IBM SP2. (b) On PC cluster.

Fig. 10. The image compositing time of the BS, the PP, and the RT methods for test sample "*Bunny*" with various numbers of processors.

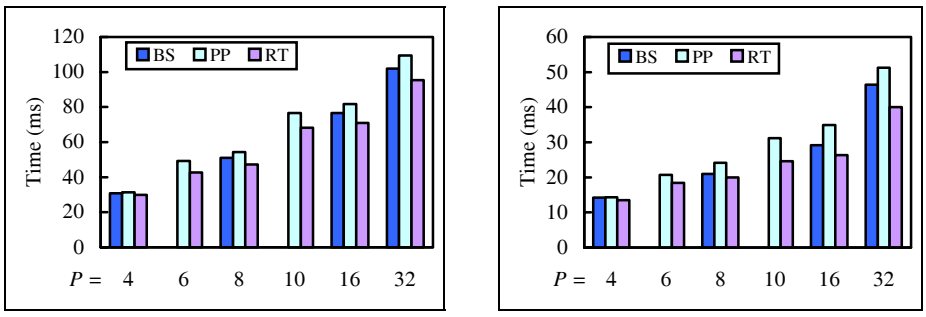
In Fig. 10 (a), for the BS method, one may think that its performance is the same as that of the RT method when the number of processors is a power-of-two. When $N = 2$, the image compositing time of the RT method is the same as that of the BS method. However, in Fig. 8, we show the best performance of the RT method. The value of N may not be equal to 2 for each different number of processors. For $P = 4$, 8, 16, and 32, the values of N that produce the best performance for the RT method 2, 3, 3, and 4, respectively. According to Eq. (9), for $P = 4$, 8, 16, and 32, the values of N that produce the best theoretical performance for the RT method are 2.3, 2.5, 3.1, and 3.5, respectively. The experimental results match the theoretical analysis for the test sample. Since the RT method can determine the value of N to get the best performance in terms of the values of P, A, T_s, T_p, and T_o while the BS method always sets the value of N to 2, in general, the RT method has better performance than the BS method.

Fig. 10 (b) shows the image compositing time of the BS, the PP, and the RT methods for test sample "*Bunny*" with various numbers of processors on a PC cluster. In Fig. 10 (b), we show the best performance of the RT method. For $P = 4$, 8, 16, and 32, the values of N that produce the best performance for the RT method are 3, 3, 4, and 5, respectively. According to Eq. (9), for $P = 4$, 8, 16, and 32, the values of N that produce the best performance for the RT method are 2.5, 3.2, 3.8, and 4.6, respectively. The experimental results match the theoretical analysis for the test sample.

Figs. 11 and 12 show the image compositing time of the three image compositing methods for test samples "*Head*" and "*Engine*", respectively, with various numbers of processors. From Figs. 11 and 12, we have similar results as those of Fig. 10.

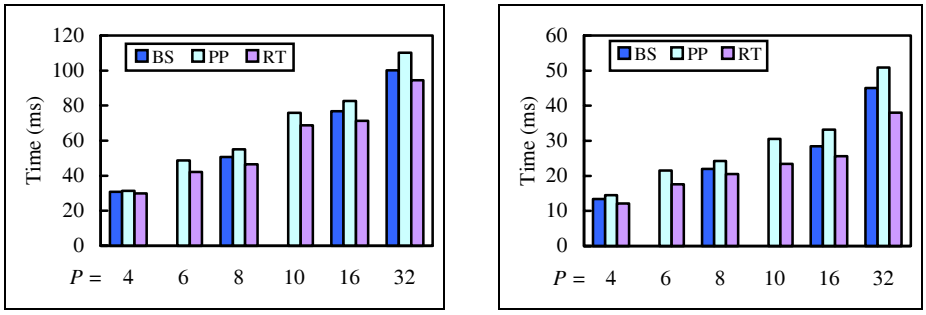
4.3 Performance Comparisons with Various Image Sizes

Fig. 13 shows the image compositing time of the three methods for test sample "*Bunny*" with various image sizes on a 32-node IBM SP2 machine and a 32-node PC cluster. The test image sizes are 128^2, 256^2, 512^2, and 1024^2 pixels. In Fig. 13, for each test image size, we only show the best performance of the PP and the RT methods. From Fig. 13, we can see that the image compositing time of the RT method is better than that of the BS and the PP methods. The reason is the same as that described for Fig. 10.



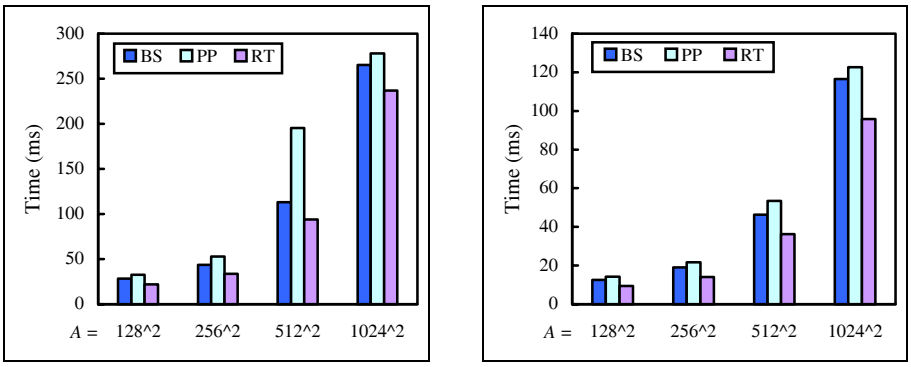
(a) On IBM SP2. (b) On PC cluster.

Fig. 11. The image compositing time of the BS, the PP, and the RT methods for test sample "*Head*" with various numbers of processors.



(a) On IBM SP2. (b) On PC cluster.

Fig. 12. The image compositing time of the BS, the PP, and the RT methods for test sample "*Engine*" with various numbers of processors.



(a) On IBM SP2. (b) On PC cluster.

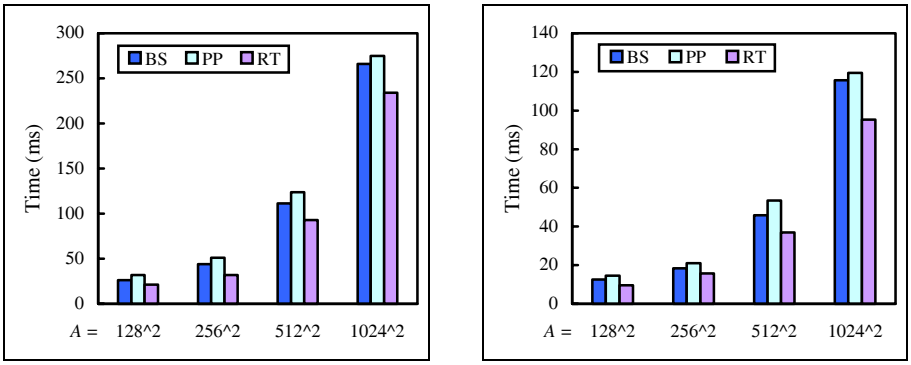
Fig. 13. The image compositing time of the BS, the PP, and the RT methods for test sample "*Bunny*" with various image sizes.

Figs. 14 and 15 show the image compositing time of the three image compositing methods for test samples "*Head*" and "*Engine*", respectively, with various image sizes. From Figs. 14 and 15, we have similar results as those of Fig. 13.

The volume data test samples, the number of processors, the final image sizes, and the characteristics of parallel machines are the major effect factors for the rendering rate. In our experimental, in general, the rendering rate achieves about 4-18 frames per second.

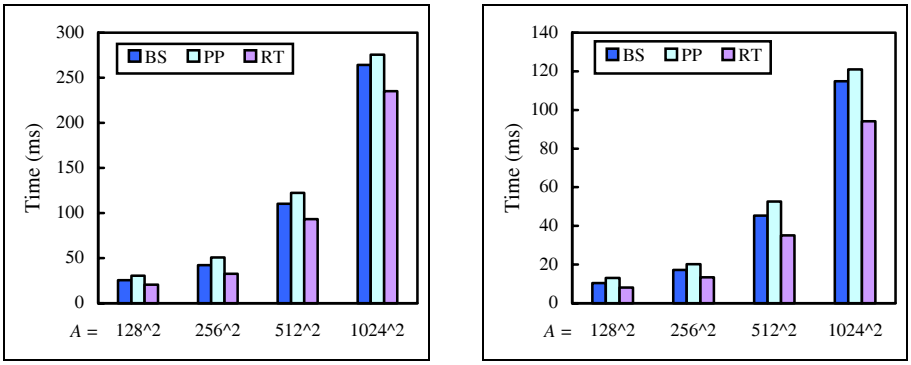
5. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented the RT image compositing method for the sort-last parallel volume rendering systems on distributed memory multicomputers. In the RT method, we derived the send/receive equations based on the indexing operations of the BS method and the ring rotation topology of the PP method. In the theoretical analysis,



(a) On IBM SP2. (b) On PC cluster.

Fig. 14. The image compositing time of the BS, the PP, and the RT methods for test sample "*Head*" with various image sizes.



(a) On IBM SP2. (b) On PC cluster.

Fig. 15. The image compositing time of the BS, the PP, and the RT methods for test sample "*Engine*" with various image sizes.

we have analyzed the performance of the BS, the PP and the RT methods in terms of the number of initial blocks of a partial image, the number of processors, the image sizes, and the characteristics of parallel machines; and derived the best performance bound of the RT method. In the experimental test, we have shown that the RT method has better performance than that of the BS and the PP methods for all test samples.

As mentioned in introduction, an efficient data compression scheme is another way to reduce the image compositing time. In the future, we plan to exploit an efficient data compression method to reduce the data transmission sizes in image compositing. We then can compare the efficient data compressions method with [1] and [24]. By combining the RT method with the efficient data compressions method, we can compare their performance with that of [2].

REFERENCES

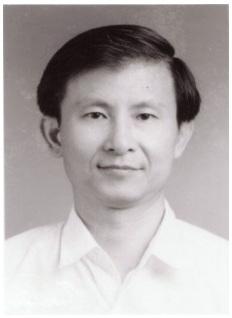
1. J. Ahrens and J. Painter, "Efficient sort-last rendering using compression-based image compositing," in *Proceedings of the 2nd Eurographics Workshop on Parallel Graphics and Visualization*, 1998, pp. 145-151.
2. M. Cox and P. Hanrahan, "Pixel merging for object-parallel rendering: a distributed snooping algorithm," in *Proceedings of 1993 Parallel Rendering Symposium* (*PRS '93*), 1993, pp. 49-56.
3. R. A. Drebin, L. Carpenter, and P. Hanrahan, "Volume rendering," in *Proceedings of SIGGRAPH '88*, Vol. 22, 1988, pp. 65-74.
4. E. Groeller and W. Purgathofer, "Coherence in computer graphics," Technical Reports TR-186-2-95-04, Institute of Computer Graphics 186-2 Technical University of Vienna, 1995.
5. W. D. Gropp and E. Lusk, "User's guide for MPICH, a portable implementation of MPI," ANL-96/6, Mathematics and Computer Science Division, Argonne National Laboratory, 1996.
6. W. M. Hsu, "Segmented ray casting for data parallel volume rendering," in *Proceedings of 1993 Parallel Rendering Symposium* (*PRS '93*), 1993, pp. 7-14.
7. IBM, IBM AIX Parallel Environment, *Parallel Programming Subroutune Reference*.
8. A. Kaufman, *Volume Visualization*, IEEE Computer Society Press, 1991.
9. P. Lacroute, "Fast volume rendering using a shear-warp factorization of the viewing transformation," Ph.D. dissertation, Stanford University, 1995.
10. P. Lacroute, "Real-time volume rendering on shared memory multiprocessors using the shear-warp factorization," in *Proceedings of 1995 Parallel Rendering Symposium* (*PRS '95*), 1995, pp. 15-22.
11. P. Lacroute, "Analysis of a parallel volume rendering system based on the shear-warp factorization," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 2, 1996, pp. 218-231.
12. P. Lacroute and M. Levoy, "Fast volume rendering using a shear-warp factorization of the viewing transformation," in *Proceedings of SIGGRAPH '94*, 1994, pp. 451-458.
13. T. Y. Lee, C. S. Raghavendra, and J. B. Nicholas, "Image compositing schemes for sort-last polygon rendering on 2D mesh multicomputers," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 2, 1996, pp. 202-217.
14. M. Levoy, "Efficient ray tracing of volume data," *ACM Transactions on Graphics*, Vol. 9, 1990, pp. 245-261.
15. C. F. Lin, Y. C. Chung, and D. L. Yang, "Parallel shear-warp factorization volume rendering using efficient 1-D and 2-D partitioning schemes on distributed memory multicomputers," *The Journal of Supercomputing*, Vol. 22, 2002, pp. 277-302.
16. C. F. Lin, Y. C. Chung, and D. L. Yang, "A rotate-tiling image composition method for parallel volume rendering on distributed memory multicomputers," in *Proceeding of IEEE International Parallel and Distributed Processing Symposiums* (*IPDPS 2001*), 2001, (CD-ROM).
17. K. L. Ma, J. S. Painter, C. D. Hansen, and M. F. Krogh, "A data distributed, parallel algorithm for ray-traced volume rendering," in *Proceedings of 1993 Parallel Rendering Symposium* (*PRS '93*), 1993, pp. 15-22.

18. K. L. Ma, J. S. Painter, C. D. Hansen, and M. F. Krogh, "Parallel volume rendering using binary-swap composition," *IEEE Computer Graphics and Applications*, Vol. 14, 1994, pp. 59-68.
19. U. Neumann, "Parallel volume rendering algorithm performance on mesh connected multicomputer," in *Proceeding of IEEE Visualization '93 Parallel Rendering*, 1993, pp. 97-104.
20. J. P. Singh, A. Gupta, and M. Levoy, "Parallel visualization algorithms: performance and architectural implications," *Computer*, Vol. 27, 1994, pp. 45-55.
21. J. Wilhelms and A. V. Gelder, "A coherent projection approach for direct volume rendering," in *Proceedings of SIGGRAPH '91*, Vol. 25, 1991, pp. 275-283.
22. C. M. Wittenbrink and A. K. Somani, "Permutation warping for data parallel volume rendering," in *Proceedings of 1993 Parallel Rendering Symposium* (*PRS '93*), 1993, pp. 57-60.
23. T. S. Yoo, U. Neumann, H. Fuchs, S. M. Pizer, T. Cullip, J. Rhoades, and R. Whitaker, "Direct visualization of volume data," *IEEE Computer Graphics and Applications*, Vol. 12, 1992, pp. 63-71.
24. D. L. Yang, J. C. Yu, and Y. C. Chung, "Efficient compositing methods for the sort-last-sparse parallel volume rendering system on distributed memory multicomputers," *The Journal of Supercomputing*, Vol. 18, 2001, pp. 201-220.

Chin-Feng Lin (林金鋒) received his B.S. and Ph.D. degrees in Computer Science from Feng Chia University in 1996 and 2004, respectively. He joined the Department of Information Management at Chang Jung Christian University as an assistant professor in 2004. His research interests include data visualization, parallel and distributed processing, high performance computing, parallel rendering, grid computing, virtual reality, and high level architecture. He is a member of IEEE computer society and ACM.

Shih-Kuan Liao (廖士寬) received his B.S. and M.S. degrees in Electrical Engineering from National Cheng Kung University in 1986 and 1988, respectively. He joined the Department of Information Management at National Taichung Institute of Technology as a lecture in 1991. He is currently a Ph.D. candidate in the Department of Information Engineering at Feng Chia University. His research interests include parallel and distributed computing, scientific visualization, and computer vision.

Yeh-Ching Chung (鍾葉青) received a B.S. degree in Information Engineering from Chung Yuan Christian University in 1983, and the M.S. and Ph.D. degrees in Computer and Information Science from Syracuse University in 1988 and 1992, respectively. He joined the Department of Information Engineering at Feng Chia University as an associate professor in 1992 and became a full professor in 1999. From 1998 to 2001, he was the chairman of the department. In 2002, he joined the Department of Computer Science at National Tsing Hua University as a full professor. His research interests include parallel and distributed processing, pervasive computing, cluster computing, grid computing, embedded software, and system software for SOC design. He is a member of the IEEE computer society and ACM.



Don-Lin Yang (楊東麟) received the B.E. degree in Computer Science from Feng Chia University in 1973, the M.S. degree in Applied Science from the College of William and Mary in 1979, and the Ph.D. degree in Computer Science from the University of Virginia in 1985. He was a staff programmer at IBM Santa Teresa Laboratory from 1985 to 1987 and a member of technical staff at AT&T Bell Laboratories from 1987 to 1991. Since 1991, he has been with Feng Chia University, where he was in charge of the University Computer Center from 1993 to 1997 and the head of the Department of Information Engineering and Computer Science from 2001 to 2003. Dr. Yang is currently a professor and Dean of Information Technology Office, Feng Chia University. His research interests include distributed and parallel computing, image processing, and data mining. He is a member of the IEEE computer society and the ACM.