Eurographics Symposium on Parallel Graphics and Visualization (2009)
J. Comba, K. Debattista, and D. Weiskopf (Editors)

A Decomposition Approach for Optimizing Large-Scale Parallel Image Composition on Multi-Core MPP Systems

J. Nonaka and K. Ono[†]

Computational Science Research Program, RIKEN, Japan

Abstract

In recent years, multi-core processor architecture has emerged as the predominant hardware architecture for high performance computing (HPC) systems. In addition, computational nodes based on SMP (symmetric multiprocessing) and NUMA (non-uniform memory architecture) have become increasingly common. Traditional parallel image composition algorithms were not primarily designed to take advantage of the combined message passing and shared address space parallelism provided by modern massively parallel processing (MPP) systems. This therefore might result in undesirable performance loss. In this study, we have investigated the use of a simple decomposition approach to take advantage of these different hardware characteristics for optimizing the parallel image composition process. Performance evaluation was carried out on a multi-core, multi-processor architecture based T2K Open Supercomputer, and we obtained encouraging results showing the effectiveness of the proposed approach. This approach also seems promising to tackle the large-scale image composition problem on next-generation HPC systems where an ever increasing number of processing cores are expected.

Categories and Subject Descriptors (according to ACM CCS): I.3.1 [Computer Graphics]: Hardware Architecture—Parallel Processing I.3.2 [Computer Graphics]: Graphics Systems—Distributed/network graphics

1. Introduction

Scientific computing and visualization have played an important role in computer-aided scientific discovery supported by HPC resources. The size and complexity of data sets generated from numerical simulations have increased following the continuous increase in computational power and network bandwidth of HPC systems. Currently, several large-scale high-performance scientific computing projects are being executed around the globe. To meet the computational resource hungry requirements of these scientific applications, larger and more powerful HPC systems are under development. For instance, IBM, NASA (SGI and Intel), and RIKEN (NEC, Fujitsu, and Hitachi) have announced the building of supercomputers with tens of petaflops performance.

Recent trends in modern high performance computing (HPC) system architecture shows an increasing adoption of multi-core, multi-processing computational nodes [Top]. As the CPU hardware moves toward multi-core configuration, the computational nodes are moving toward SMP (Symmetric Multi Processing) and NUMA (Non-Uniform Memory Access) architectures. In addition, as the node size increases, hierarchical multi-level network topology has become widespread. This therefore results in a heterogeneous bandwidth performance across the entire system. Currently, three of the most powerful supercomputers in Japan are based on T2K Open Supercomputer architecture [T2K], that is, a NUMA-based multi-core MPP architecture.

The use of HPC systems for visualization has received increasing attention as a feasible, and sometimes the most adequate, approach for post-processing. This is because it can avoid costly and sometimes prohibitive data transfer of numerical simulation results to graphics capable systems. [CFN03, TYRG*06, KLMaYT07, PYRM08, RCM07]. Visualization on the HPC side can also be useful to remove unnecessary simulation results during a parameter survey, or to extract only necessary portions of the data in order to be transferred and visualized on a visualization cluster. In such case, low and medium resolution images might be sufficient for interactive visualization on the HPC side, since

[†] {jorji | keno}@riken.jp

DOI: 10.2312/EGPGV/EGPGV09/071-078

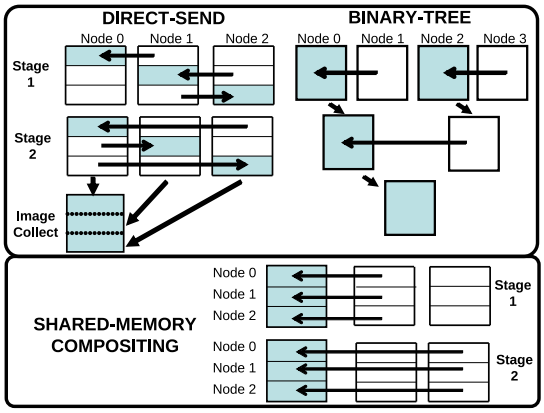

Figure 1: *Examples of image composition methods for distributed and shared-memory environments.*

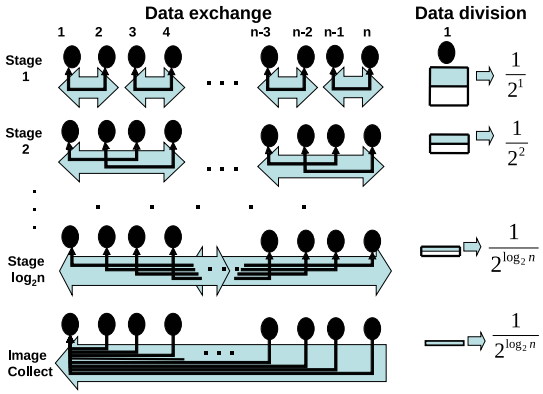

Figure 2: *Data exchange and decomposition pattern of Binary-Swap image composition.*

high-resolution, high-quality real-time visualization can be left for the graphics hardware accelerated visualization systems. Sort-last parallel rendering [MCEF94] emerges as a natural candidate for visualization on the HPC side, however the required final image composition process can become a serious problem since it requires intense data communication.

Traditional parallel image composition algorithms were not designed with hybrid programming model in mind. Most of them are designed for pure distributed or pure shared memory parallel systems. Although distributed memory applications can work on systems with full or partial shared memory address space, a loss in performance might occur when the hybrid distributed and shared memory programming model is not taken into consideration. In this study, we have investigated a simple decomposition approach in order to take advantage of the hybrid programming model.

In order to verify the effectiveness of the proposed decomposition approach, we executed a performance evaluation on a multi-core MPP system. For this purpose, we used the *Todai Combined Cluster* (hereafter called *Todai T2K*), a T2K Open Supercomputer installed at University of Tokyo, Japan. We obtained encouraging results showing that this approach can effectively optimize large-scale image composition process on systems with heterogeneous memory architecture. The decomposition approach also shows promising to tackle the large-scale image composition on ever increasing number of processing cores verified on recent leading-edge HPC systems.

The remainder of this paper is organized as follows. In Section 2, we describe the sort-last image composition with special attention to the Binary-Swap method. In Section 3, we present the proposed decomposition approach for optimizing parallel image composition. Experimental results and discussions are presented in Section 4, and we conclude by presenting some future works in Section 5.

2. Sort-Last Image Composition

Sort-last image composition method is responsible for the final stage of the parallel rendering pipeline. That is, full size images generated by the rendering process are composited, or merged, by using alpha blending or z-buffer techniques, in order to produce the final image. In this study, we focused on alpha blending which is more complex than z-buffer method since it requires correct ordering of the entire set of images during the composition process. Although z-buffer method does not depend on the composition order, it is worth noting that this approach can directly be applied to this method without further modifications. Several image composition algorithms for distributed as well as shared-memory environments have been proposed so far, and some examples are depicted in Figure 1. Most of methods were originally designed for pure distributed memory environments, and these are generally grouped into three main categories: *Direct Send* [Hsu93, Neu93], *Parallel Pipeline* [LRN96], and *Binary-Tree* which includes *Binary-Swap* [MPHK94].

Figure 1 shows the original Direct Send method based on image decomposition approach. Parallel Pipeline method improved the message exchange pattern of Direct Send method in order to avoid link contention. It also included a support for image composition on a 2D array of composition nodes. Current Direct Send derived methods also include support for 3D distribution of composition nodes. Parallel Pipeline method has proven suitable for small size parallel systems. In fact, this method has been used on commercial parallel visualization applications such as *AVS/Express PST* (Parallel Support Toolkit) [AVS] and *CEI Ensight DR* (Distributed Rendering) [CEI]. However, for larger image composition process, Direct Send and Binary-Swap methods seem more appropriate. Hardware-based solutions for image composition have also been proposed so far. It includes devices which were commercially available such as HP Sepia-2 [LMS*01] and Mitsubishi Precision MPC [MOM*01]. In

addition, hardware-assisted solutions using MPC [NKS*04] and NPU (Network Processing Unit) [PMD*07] has also been proposed. Although these solutions have proven effective on small and medium size clusters, in a large-scale HPC environment the pure software-based image composition becomes the most appropriate approach.

Several optimizations both for Direct Send [SML*03, SMW*05, EP07] and Binary-Swap [AP98, YYC01, TIH03, SKN04, LCY07, YWM08] have been proposed so far. Most of them have focused on reducing the data size to be transmitted or on minimizing the number of data transmission. In the specific case of large-scale image composition, *Scheduled Linear Image Composition* method, or SLIC [SML*03] for short, emerged as a potential candidate due to its communication cost in the order of $O(n.n^{1/3})$ thanks to the use of on-the-fly optimized scheduling. This therefore makes the number of required message transmission comparable to Binary-Swap, $O(n.log_2 n)$, when up to 1024 composition nodes are involved. However, in the order of tens of thousands of composition nodes, SLIC might require two times more message transmission. Binary-Swap has been limited to the use of power of two composition nodes. However, *2- 3 Swap Image Composition* [YWM08] has recently been proposed as a generalization of Binary-Swap to an arbitrary number of composition nodes. In this method, Binary-Swap algorithm is still applied when power of two compositing nodes are available. In this study, we have primarily focused on Binary-Swap method due to its theoretical high scalability potential.

2.1. Binary-Swap Image Composition

Binary-Swap is considered a highly optimized binary-tree method where the rendering nodes are kept busy as much as possible during the entire image composition stages. Binary-Swap is perhaps the most used, and has been widely researched generating several optimization techniques. As shown in Figure 2, during the Binary-Swap image composition process, the image is recursively divided into two parts. Half of them is exchanged between pairs of composition nodes. The other half is then composited with the received image taking into consideration the correct ordering. Although data size required for sending, receiving, and blending at each stage diminishes as the image composition stage advances, the communication distance doubles at each stage. This linear increase in distance has great potential to compromise the network traffic when the number of composition nodes (n) increases.

At the end, each node will possess $1/n$ of the original image size as the final composited image. These final composited image fragments distributed across the composition nodes are required to be gathered and reconstructed at the main composition node (*root node*). Since each image fragment size is $1/n$, thus the amount of data to be gathered will be equivalent to the total image size. A feasible approach for this step is to use the available MPI collective functions such as *MPI_Gather*. This stage has been ignored for small size parallel systems since seldom affected the composition performance. However, when the number of nodes increases it has a great potential to become a serious problem.

There is a vast and rich literature on sort-last image composition method, and a detailed theoretical performance analysis for both shared memory [RH00], and distributed memory [CMF05, Tay02], parallel computing systems can be found. The total time required for the parallel image composition is usually the summation of times required to read the images (t_{read}), to actually perform the image composition ($t_{compose}$), to collect the composited subimages ($t_{collect}$), and to write the final image (t_{write}). In pure software rendering context such as those executed at HPC system side, the time for reading the image (t_{read}) usually can be ignored since the rendered image is already stored at the main memory. The time for writing (t_{write}) the final image represents the time for effectively flushing to a file or the time required for displaying onto a display device. The time for composition ($t_{compose}$) and collecting ($t_{collect}$) are usually the most costly and defines the upper bound of achievable performance. Thus in this study we focused on these two parameters.

The Binary-Swap composition time (t_{BS}) can be expressed as shown in Equation 1. In this equation, the term n corresponds to the number of composition nodes, and P represents the total number of pixels in the image. The $t_{compose}$ term includes the time for sending (t_{send}), receiving (t_{recv}) and alpha blending (t_{blend}) at each image composition stage. Since modern network interconnect supports full duplex communication, the time for sending and receiving data between pairs of composition nodes can be substituted by t_{comm} ($=max(t_{send}, t_{recv})$). All these components are directly influenced by the image size as well as the pixel size of the rendered image. In addition, t_{comm} is directly influenced by the network bandwidth (B) and latency (L). On the other hand, t_{blend} is directly influenced by the processor performance.

$$
\begin{aligned}
t_{BS}(n) &= \left(\sum_{i=1}^{log_2 n} t_{compose_i}\right) + t_{collect_n} \\
&= \left[\sum_{i=1}^{log_2 n} \left(t_{comm_i} + t_{blend_i}\right)\right] + t_{gather_n} \\
&\textit{where} \\
t_{comm_i} &= \max\left(L_{send_i} + \frac{\frac{P}{2^i}}{B_{send_i}}, L_{recv_i} + \frac{\frac{P}{2^i}}{B_{recv_i}}\right) \\
t_{blend_i} &= \frac{\frac{P}{2^i}}{B_{blend_i}} \\
t_{gather(n)} &= L_{gather(n)} + \frac{P}{B_{gather(n)}}
\end{aligned}
\tag{1}
$$

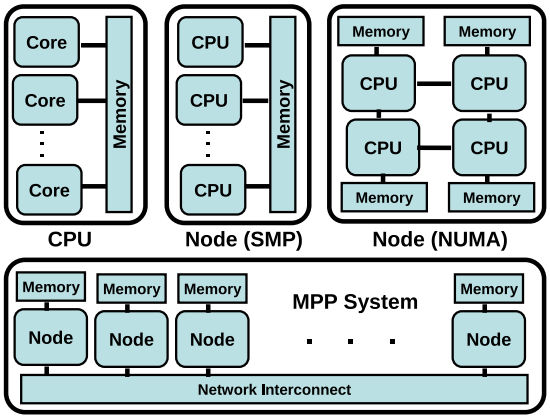

Figure 3: *Multi-core MPP system architecture.*

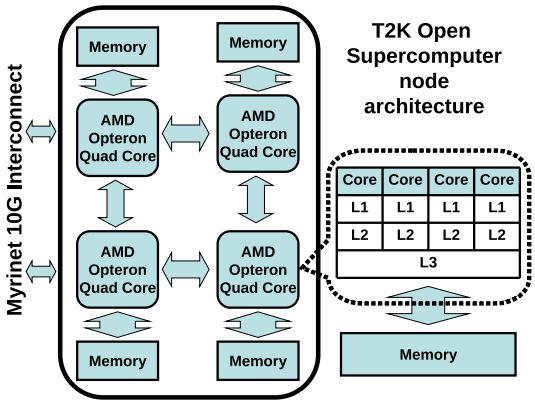

Figure 4: *NUMA node architecture of Todai T2K.*

3. Decomposition Approach for Image Composition

Multi-core architecture has emerged as the "standard" building blocks of modern CPUs. With the introduction of high-performance multi-core processors, the multi CPU nodes have become the predominant computational nodes of modern HPC systems. Figure 3 shows an example of multi-core, multi-processing hardware architecture used in modern HPC systems. Usually, computational nodes have SMP or NUMA based shared-memory, and these are distributed across the entire system generating different memory access costs. Figure 4 depicts the NUMA-based node architecture of T2K Open Supercomputer utilized in this evaluation. Although memory access time is dependent on the access path, all these 16 processing cores have access to the entire memory present in the computational node.

Modern message passing library takes advantage of these kind of shared memory address space by executing message exchanging via memory copy in order to avoid data transmission through the network interconnect. Although this can be beneficial to all distributed memory oriented image composition methods, it does not take full advantage of the shared-memory address space parallelism. For instance, during the intra-node image composition, the communication time (t_{comm}) and the collecting time (t_{gather}) could be minimized, or even eliminated, since all the required data are already present in the shared-memory. The inter-node communication through the available network interconnect is generally several orders of magnitude slower than intra-node data communication. In addition, it only provides smaller data communication bandwidth thus a simple decomposition to intra-node and inter-node image composition might be extremely beneficial to avoid intense inter-node data communication. An extension to large-scale inter-node image composition is straightforward, and the next session will discuss some examples of decomposition in detail.

3.1. Shared-Memory Compositing (SMC) + Binary-Swap (BS)

Shared-Memory Compositing (*SMC*) [RH00] appears as a prime candidate for taking advantage of the shared-memory address space provided by multi-core, multi-processor computational nodes. By taking advantage of the shared-memory environment, *SMC* can eliminate the t_{comm} and t_{gather} required by some distributed memory oriented methods such as Binary-Swap. In this case, *SMC* image composition time will be equivalent to the time for alpha blending (t_{blend}) a full image size using k compositing nodes (computational cores), since each k node will be responsible for c ompositing $1/k$ size of the image. However, it should be taken into consideration that in NUMA systems, memory access performance is affected when accessing data stored at non-local memory.

$$
\begin{aligned}
t_{SMC+BS}(n) &= \max_{Blocks1:m}\left(t_{SMC(k)}\right) + t_{BS(m)} \\
&\textit{where} \\
t_{SMC(k)} &\approx t_{blend}(P) \\
t_{BS}(m) &= \left\lceil \sum_{j=1}^{log_2 m} \left(t_{comm_j} + t_{blend_j}\right) \right\rceil + t_{gather(m)}
\end{aligned}
\tag{2}
$$

Considering that the total number of composition nodes n can be decomposed into m groups of k shared memory composition nodes, the required image composition time (t_{SMC+BS}) for the *SMC* intra-node and *BS* inter-node image composition will be as shown in Equation 2. After the optimized intra-node image composition via *SMC* method, only a light-weight *BS* inter-node image composition will be required. In this case, image composition of only m images will be necessary. The reduced number of composition nodes will alleviate potential network contention and facilitate collective communication. It is worth noting that the optimum performance will be obtained when the images to be composited are distributed continuously among neighboring

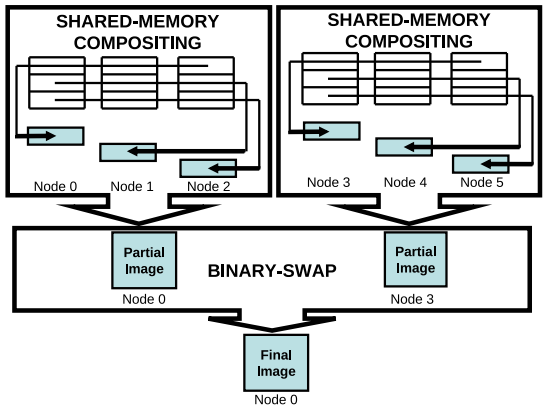

Figure 5: *Combined SMC + BS image composition.*

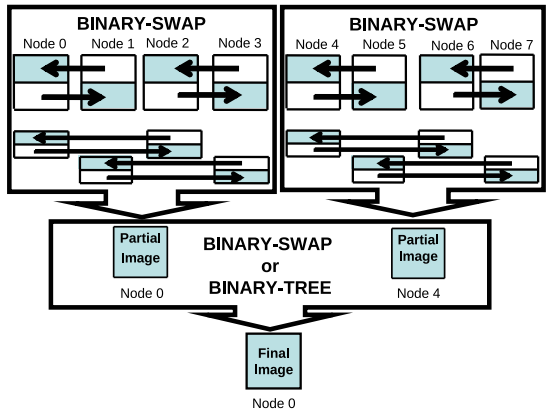

Figure 6: *Combined BS + BS (or BT) image composition.*

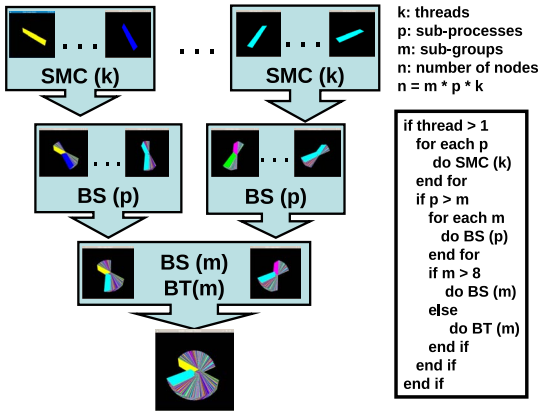

Figure 7: *Image composition using decomposition approach.*

compositing nodes (computational cores). This is probably the ordinary situation for the traditional data decomposition and distribution approach. A performance loss will be inevitable when this sequence is broken, for instance, by parallel rendering schemes using dynamic load balancing. However, the performance loss might be balanced by the performance gain in the rendering process.

3.2. Binary-Swap (BS) + Binary-Swap (BS) (or Binary-Tree (BT))

The aforementioned approach can be extended to alleviate large-scale inter-node image composition problem. Image composition nodes can be grouped in subgroups to execute concurrent light-weight inter-node image composition. This will result in another light-weight image composition on the following stage. In the specific case of *BS*, the decomposition process is facilitated because of its hierarchical binary-tree configuration. As shown in Figure 6, *BS* composition nodes can be grouped into subgroups in order to concurrently perform *BS* composition. Depending on the number of remaining *BS* nodes, further decomposition might be applied. On the other hand, if the number of remaining nodes is small, *BS* can be substituted by Binary-Tree in order to avoid the final image collecting process. Equation 3 shows the required image composition times when combining *BS* with *BS* (t_{BS+BS}), and *BS* with *BT* (t_{BS+BT}). Figure 7 shows a possible decomposition scheme for large-scale image compositon combining *SMC*, *BS*, and *BT* methods.

$$
\begin{aligned}
t_{BS+BS}(n) &= \max_{Blocks1:m}\left(t_{BS(p)}\right) + t_{BS(m)} \\
t_{BS+BT}(n) &= \max_{Blocks1:m}\left(t_{BS(p)}\right) + t_{BT(m)} \\
&\textit{where} \\
t_{BS}(p) &= \left\lceil \sum_{a=1}^{log_2 m} \left(t_{comm_a} + t_{blend_a}\right) \right\rceil + t_{collect(p)} \\
t_{BS}(m) &= \left\lceil \sum_{b=1}^{log_2 m} \left(t_{comm_b} + t_{blend_b}\right) \right\rceil + t_{collect(m)} \\
t_{BT}(m) &= \sum_{c=1}^{log_2 m} \left(t_{comm_c} + t_{blend_c}\right)
\end{aligned}
\tag{3}
$$

4. Experimental Results

4.1. Experimental Setup

We implemented a simple parallel image composition application using C programming language together with MPI communication library and OpenMP directives. This application generates 32-bit RGBA images, on-the-fly. We used image sizes of 512x512 and 1024x1024 for performance evaluation. These image resolutions can be considered sufficient for interactive visualization on the HPC side. For instance, for feature extraction, data reduction, or data selection during a parameter survey. Therefore, complex vi-

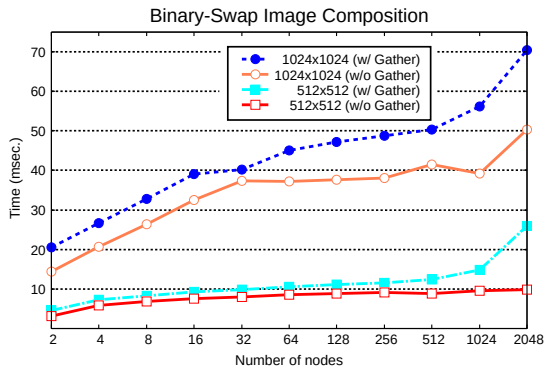

Figure 8: *BS execution time.*

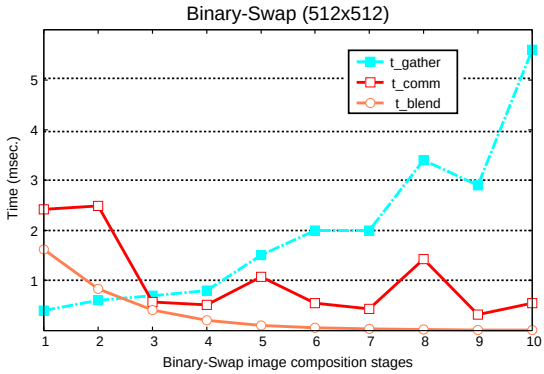

Figure 9: *BS component execution time.*

sual data exploration through high-quality, high-resolution real-time visualization can be left to a more graphics capable visualization clusters. We opted for randomly generating full colored image without any background pixel in order to force the execution of alpha blending throughout the entire image. This therefore eliminates the performance variation due to the different ratios of foreground and background pixels in different images. We did not apply any acceleration technique, such as bounding box or image compression, in order to verify the lower bound of the image composition performance.

The *Todai T2K* used for performance evaluation is composed of four clusters with a total of 952 (512 + 128 + 256 + 56) computational nodes. Each node possesses four AMD Quad Core Opteron 2.3GHz, 32 GB of RAM, and 250 GB of local HDD. It uses Linux as the Operating System. Hitachi compiler as well as Intel compiler can be used with message passing library, based on MPICH-MX, and OpenMP directives, based on OpenMP 2.0. Although some measurements were carried out using 256 nodes, or 4096 computational cores, most of the measurements were carried out using up to 128 nodes, or 2048 computational cores. Hitachi compiler was used to generate the binary code and *numactl* was used to force processor-memory affinity. In most of the cases, the processes were mapped continuously on the neighboring computational cores. We measured the image composition time using traditional *MPI_Wtime* function. Ten successive image composition using different input images (randomly generated) have been executed for each measurement and the best measured time was selected.

4.2. BS Composition Performance

Figure 8 shows the measured *BS* image composition time using up to 2048 *BS* composition nodes. We measured the image composition time with and without applying the final image collecting process in order to observe the contribution of the final collecting stage. We could observe that a special care on this stage is required when optimizing the entire image composition process. Apart of that, we could observe a considerable performance degradation on both image sizes when using large number of compositing nodes, that is, in the order of thousands.

Figure 9 shows the average execution time of the three main components of *BS* when compositing 512x512 images using up to 1024 nodes. As *BS* stage advances, the image data size required for sending, receiving, and blending is reduced by half. Therefore a proportional reduction on the execution time such as shown by t_{blend} might be expected. However, we could verify that the execution time involving network communication does not show such kind of behavior as the *BS* composition stage advances. In this graph, t_{blend} and t_{comm} should be accumulated as the composition stage advances, however t_{gather} only shows the required time for executing final image collecting process at each of the stages.

4.3. SMC+BS Composition Performance

Figures 10 and 11 show the measured time of combined intra-node *SMC* and inter-node *BS* image composition using image sizes of 512x512 and 1024x1024. In these figures, "*SMC(16)-BS*" represents the image composition time when using 16 *SMC* nodes, that is, 16 *threads*. On the other hand, "*SMC(4)-BS*" represents the time when using 4 *SMC* nodes. As expected, similar performance on both image sizes were obtained when using up to 4 nodes. Although intra-node *BS* composition takes advantage of the optimized message passing using shared-memory, the *SMC* has always outperformed *BS* during the intra-node image composition. From 8 nodes, *SMC(4)-BS* starts the *BS* composition and we can verify a considerable performance drop compared to *SMC(16)-BS*. From 32 nodes, all these three approaches will be executing inter-node *BS* image composition. However, each one will be in different *BS* composition stage thus they will be sending, receiving, and blending different amounts of data. For instance, when using 32 nodes, *BS* will have 32 *BS* nodes and will be in the fifth *BS* stage; *SMC(4)-BS* will have 8 *BS* nodes and will be in the third *BS* stage; and *SMC(16)-BS*

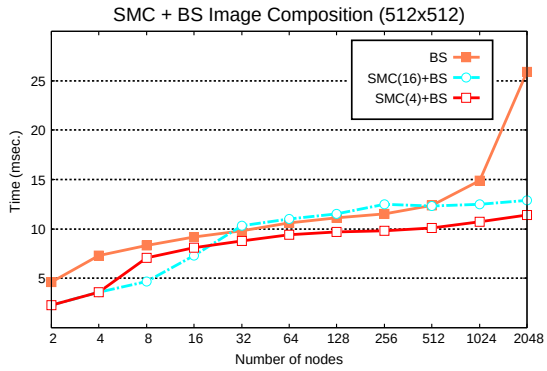

Figure 10: *SMC+BS image composition time for image size of 512x512.*

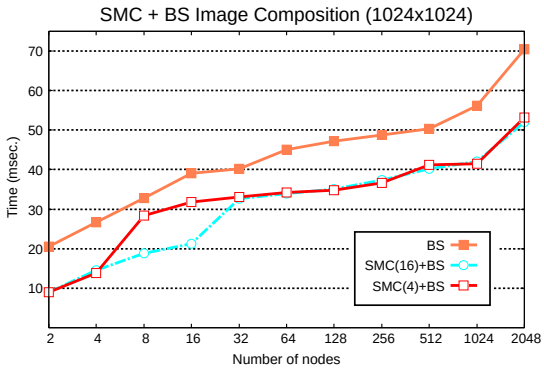

Figure 11: *SMC+BS image composition time for image size of 1024x1024.*

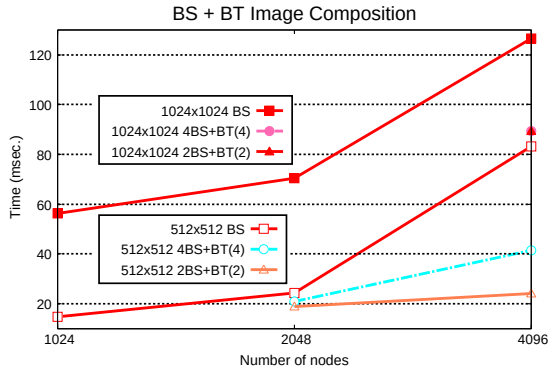

Figure 12: *BS+BT image composition time, using two and four subgroups of BS nodes.*

will have only 2 *BS* nodes and will be in the first *BS* stage. The other fact is that the number of nodes involved in the final image gathering will also be different in these three approaches.

4.4. BS + BS (or BT) Composition Performance

Figure 12 shows the measured time of combined inter-node *BS* with *BT* image composition using image sizes of 512x512 and 1024x1024. In both cases, we divided the entire *BS* nodes into two and four subgroups. In both cases, Binary-Tree method was used in the final stage of image composition. "*4BS+BT(4)*" represents the measured time when using four subgroups, and "*2BS+BT(2)*" represents the measured time when using two subgroups. We could observe that both subgroup sizes can effectively minimize the performance degradation. However, best results were obtained when dividing it into two subgroups. For larger number of composition nodes, different combinations of *SMC*, *BS*, and *BT* can be considered. However, further investigations are required in order to obtain the optimum decomposition size as well as the best combination of image composition methods.

5. Conclusions

In this paper, we presented a simple decomposition approach for optimizing large-scale image composition on multi-core MPP architecture. This architecture, which is becoming increasingly common these days, provides a hybrid memory environment combining distributed and shared-memory address space. We investigated a simple decomposition approach striving to optimize the entire image composition process. We applied intra-node *SMC* image composition to the shared-memory portion and we obtained considerable performance increase. This was because *SMC* does not require the communication and final collecting processes compared to other distributed memory oriented methods. In addition, the inter-node image composition, on the distributed memory portion, takes advantage of the reduction in the number of images to carry out light-weight image composition using reduced number of composition nodes. To tackle the large-scale inter-node image composition problem, we have investigated the decomposition of *BS* process. This decomposition was greatly facilitated because of its hierarchical tree structure. We could observe that this inter-node decomposition approach is useful to reduce the performance degradation when a large number of *BS* nodes are involved. The flexibility of this approach enables the use of different decomposition schemes as well as the use of different combinations of image composition methods. However, further investigation for calculating the optimum decomposition sizes as well as for selecting the best combination of composition methods are required and these are left as our future works. Future works also include the investigation of other large-scale image composition methods such as *Direct-Send* and *2-3 Swap* image composition methods.

Acknowledgments

This research was conducted using the T2K Open Supercomputer (*Todai Combined Cluster*) at the University of Tokyo under the "HPC Special Project". This research was supported by the National Project on "Next-generation Integrated Living Matter Simulation" of Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan.

References

[AP98] AHRENS J., PAINTER J.: 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.

[AVS] AVS: Advanced Visual Systems. http://www.avs.com/.

[CEI] CEI: Computational Engineering International. http://www.ensight.com/.

[CFN03] CHEN L., FUJISHIRO I., NAKAJIMA K.: Optimizing parallel performance of unstructured volume rendering for the earth simulator. *Parallel Comput. 29*, 3 (2003), 355–371.

[CMF05] CAVIN X., MION C., FILBOIS A.: COTS cluster-based sort-last rendering: Performance evaluation and pipelined implementation. In *Proceedings of the IEEE Visualization Conference* (2005), pp. 111,118.

[EP07] EILEMANN S., PAJAROLA R.: Direct send compositing for parallel sort-last rendering. In *Proceedings of the Eurographics Symposium on Parallel Graphics and Visualization* (2007), pp. 29–36.

[Hsu93] HSU W. M.: Segmented ray casting for data parallel volume rendering. In *PRS '93: Proceedings of the 1993 Symposium on Parallel Rendering* (1993), pp. 7–14.

[KLMaYT07] KWAN-LIU MA AND C. W., YU H., TIKHONOVA A.: In-situ processing and visualization for ultrascale simulations. *Journal of Physics: Conference Series (Proceedings of DOE SciDAC 2007 Conference) 78* (2007), 012043.

[LCY07] LIN C.-F., CHUNG Y.-C., YANG D.-L.: TRLE–an efficient data compression scheme for image composition of volume rendering on distributed memory multicomputers. *J. Supercomput. 39*, 3 (2007), 321–345.

[LMS*01] LOMBEYDA S., MOLL L., SHAND M., BREEN D., HEIRICH A.: Scalable interactive volume rendering using off-the-shelf components. In *Proceedings of the IEEE 2001 Symposium on Parallel and Large-Data Visualization and Graphics* (2001), pp. 115–121.

[LRN96] LEE T.-Y., RAGHAVENDRA C. S., NICHOLAS J. B.: Image composition schemes for sort-last polygon rendering on 2D mesh multicomputers. *IEEE Transactions on Visualization and Computer Graphics 2*, 3 (1996), 202–217.

[MCEF94] MOLNAR S., COX M., ELLSWORTH D., FUCHS H.: A sorting classification of parallel rendering. *IEEE Computer Graphics and Applications 14*, 4 (1994), 23–32.

[MOM*01] MURAKI S., OGATA M., MA K.-L., KOSHIZUKA K., KAJIHARA K., LIU X., NAGANO Y., SHIMOKAWA K.: Next-generation visual supercomputing using PC Clusters with volume graphics hardware devices. In *Proceedings of the 2001 ACM/IEEE Conference on Supercomputing (CDROM)* (2001), pp. 51–51.

[MPHK94] MA K.-L., PAINTER J. S., HANSEN C. D., KROGH M. F.: Parallel volume rendering using binary-swap image composition. *Computer Graphics and Application 14*, 4 (1994), 59–68.

[Neu93] NEUMANN U.: Parallel volume-rendering algorithm performance on mesh-connected multicomputers. In *PRS '93: Proceedings of the 1993 Symposium on Parallel Rendering* (1993), pp. 97–104.

[NKS*04] NONAKA J., KUKIMOTO N., SAKAMOTO N., HAZAMA H., WATASHIBA Y., LIU X., OGATA M., KANAZAWA M., KOYAMADA K.: Hybrid hardware-accelerated image composition for sort-last parallel rendering on graphics clusters with commodity image compositor. In *VolViS 2004: Proceedings of the IEEE/SIGGRAPH Symposium on Volume Visualization and Graphics 2004* (2004), pp. 17–24.

[PMD*07] PUGMIRE D., MONROE L., DAVENPORT C. C., DUBOIS A., DUBOIS D., POOLE S.: NPU-based image compositing in a distributed visualization system. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (2007), 798–809.

[PYRM08] PETERKA T., YU H., ROSS R., MA K.-L.: Parallel volume rendering on the IBM Blue Gene/P. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Parallel Graphics and Visualization* (2008).

[RCM07] RAO A. R., CECCHI G., MAGNASCO M.: High performance computing environment for multidimensional image analysis. *BMC Cell Biology 8*, Suppl 1 (2007), S9.

[RH00] REINHARD E., HANSEN C.: A comparison of parallel compositing techniques on shared memory architectures. In *Proceedings of the Third Eurographics Workshop on Parallel Graphics and Visualisation* (2000), pp. 115–123.

[SKN04] SANO K., KOBAYASHI Y., NAKAMURA T.: Differential coding scheme for efficient parallel image composition on a PC cluster system. *Parallel Comput. 30*, 2 (2004), 285–299.

[SML*03] STOMPEL A., MA K.-L., LUM E. B., AHRENS J., PATCHETT J.: SLIC: Scheduled linear image compositing for parallel volume rendering. In *PVG '03: Proceedings of the 2003 IEEE Symposium on Parallel and Large-Data Visualization and Graphics* (2003), p. 6.

[SMW*05] STRENGERT M., MAGALLÓN M., WEISKOPF D., GUTHE S., ERTL T.: Large volume visualization of compressed time-dependent datasets on GPU Clusters. *Parallel Comput. 31*, 2 (2005), 205–219.

[T2K] T2K: T2K Open Supercomputer Alliance. http://www.open-supercomputer.org/.

[Tay02] TAY Y. C.: A comparison of pixel complexity in composition techniques for sort-last rendering. *Journal of Parallel and Distributed Computing 62*, 1 (2002), 152 – 171.

[TIH03] TAKEUCHI A., INO F., HAGIHARA K.: An improved binary-swap compositing for sort-last parallel rendering on distributed memory multiprocessors. *Parallel Comput. 29*, 11-12 (2003), 1745–1762.

[Top] TOP500: Top500 supercomputer sites. http://www.top500.org/.

[TYRG*06] TU T., YU H., RAMIREZ-GUZMAN L., BIELAK J., GHATTAS O., MA K.-L., O'HALLARON D. R.: From mesh generation to scientific visualization: An end-to-end approach to parallel supercomputing. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing* (2006), p. 91.

[YWM08] YU H., WANG C., MA K.-L.: Massively parallel volume rendering using 2-3 swap image compositing. In *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing* (2008), pp. 1–11.

[YYC01] YANG D.-L., YU J.-C., CHUNG Y.-C.: Efficient compositing methods for the sort-last-sparse parallel volume rendering system on distributed memory multicomputers. *J. Supercomput. 18*, 2 (2001), 201–220.