

# Performance of communication patterns for extreme-scale analysis and visualization

T Peterka,<sup>1</sup> W Kendall,<sup>2</sup> D Goodell,<sup>1</sup> B Nouanesengsey,<sup>3</sup> H-W Shen,<sup>3</sup> J Huang,<sup>2</sup> K Moreland,<sup>4</sup> R Thakur,<sup>1</sup> and R B Ross<sup>1</sup>

<sup>1</sup>Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA

<sup>2</sup>Department of Electrical Engineering and Computer Science, University of Tennessee at Knoxville, Knoxville, TN 37996, USA

<sup>3</sup>Department of Computer Science and Engineering, The Ohio State University, Columbus, OH 43210, USA

<sup>4</sup>Sandia National Laboratories, Albuquerque, NM 87185, USA

Email: tpeterka@mcs.anl.gov

**Abstract.** Efficient data movement is essential for extreme-scale parallel visualization and analysis algorithms. In this research, we benchmark and optimize the performance of collective and point-to-point communication patterns for data-parallel visualization of scalar and vector data. Two such communication patterns are global reduction and local nearest-neighbor communication. We implement scalable algorithms at tens of thousands of processes, in some cases to the full scale of leadership computing facilities, and benchmark performance using large-scale scientific data.

## 1. Introduction

Through funding initiatives such as SciDAC, the U.S. Department of Energy's Office of Science has embarked on a historic path in computational and computer science. At tera- and petascale today, and certainly at exascale in the future, data analysis and visualization will continue to be pivotal components in the success of this endeavor [1]. Efficient data movement, whether storage access or network communication, is essential for the success of parallel analysis algorithms at extreme scale.

This paper summarizes the performance of two network communication patterns for parallel analysis and visualization: global reduction and local nearest-neighbor communication. Global reduction is collective communication whereby all processes participate in merging their results into one solution. Local nearest-neighbor is an assortment of sparse collective neighborhoods, and each process communicates with only those processes in its immediate vicinity. Case studies of the scalability of the Radix-k algorithm for parallel image compositing and of parallel particle tracing are presented as examples of each communication model. The two case studies represent high impact long-standing scalability challenges with the potential to impede progress in extreme-scale visualization; one is a critical bottleneck in rendering while the other is a first step in many flow visualization and feature extraction algorithms.

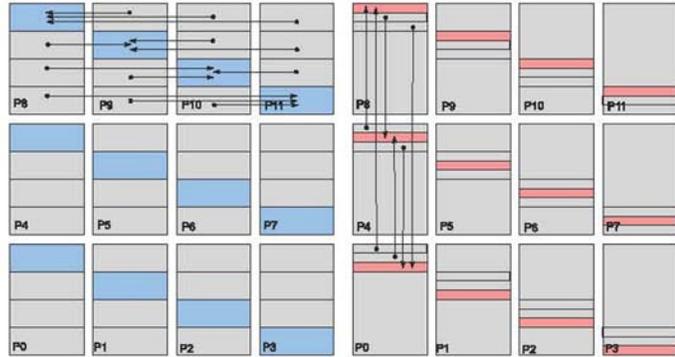
## 2. Algorithms and Data Structures for Communication in Visualization

Sort-last parallel visualization algorithms are data-parallel approaches that partition the data space among processes and execute the same visualization task concurrently on each process. Upon completion of this step, each process owns an output image that must be merged with all the other processes' images to form a single result. Image composition is the name given to this stage, and it relies on an efficient global reduction communication pattern. Previously, direct-send [2] and binary swap [3] were the accepted best practices for performing image composition, although recently Radix-k [4,5] has demonstrated significant performance gains on a variety of machine architectures.

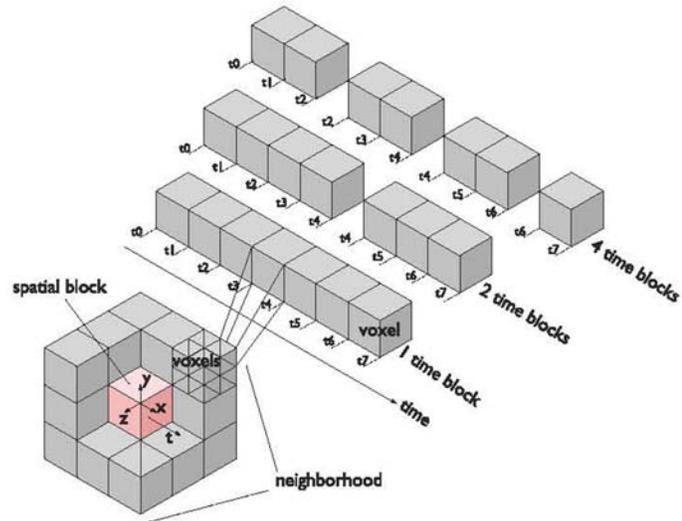
Radix-k enables the amount of communication concurrency to be tuned to the architecture by factoring the number of processes into a number of rounds and a number of communicating partners in a group in each round (see Figure 1). By configuring the  $k$ -values (group size in each round) appropriately, the available network bisection bandwidth can be approached without exceeding it and generating contention for messages. With Radix-k, direct-send and binary swap simply become two of the numerous valid configurations possible for a given number of processes. Radix-k also overlaps the communication of messages with the reduction computation as much as possible, further improving performance.

Some analysis and visualization applications require a local message exchange among neighbors instead of global reduction. This is the case when particles are advected by a flow field for computing streamlines or pathlines. As each particle crosses block boundaries, it is relayed to the process that owns the neighboring block. Or, as Pugmire et al. [6] demonstrated, a new data block can be loaded by the same process as an alternative to handing the particle to another process. In our algorithm, the message sizes involved in communicating particles are smaller than the data movement required to load another data block from storage. Thus, we use a static block distribution and focus our efforts instead on efficient communication.

Figure 2 shows the basic block structure used in our nearest-neighbor communication. The dataset consists of



**Figure 1.** Illustration of the Radix-k parallel image compositing algorithm on 12 processes. Any factorization of the total number of processes is a valid configuration; in this example, two rounds of [4,3] are used. Groups of 4 are formed in the first round (left), and groups of 3 are formed in the second round (right). Within each group, a local direct-send pattern is executed.



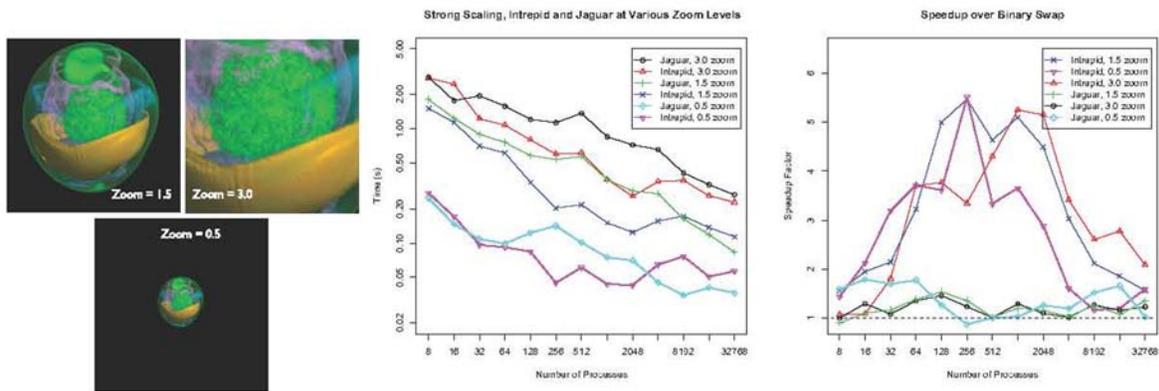
**Figure 2.** The data structures for nearest-neighbor communication of 4D particles are overlapping neighborhoods of 81 4D blocks. Each block consists of voxels and has extents in the  $(x, y, z, t)$  dimensions. In the example above, eight time-steps are grouped into either one, two, or four blocks in the time dimension.

multiple time-steps, one per file, containing 3D velocity vectors; and because we accommodate time-varying datasets and unsteady flow fields, all particles and blocks are 4D entities. A neighborhood consists of a central block surrounded by 80 other neighbors, for a total neighborhood size of 81 blocks. That is, the neighborhood is a  $3 \times 3 \times 3 \times 3$  region comprising the central block and any other block adjacent in space and time. These neighborhoods are partially overlapping; hence, the neighbor relation is reflexive and symmetric but not transitive. A particle is transferred from one block to another within a neighborhood whenever one or more particle coordinates  $(x, y, z, t)$  exceed the block boundary in any of the four dimensions.

### 3. Performance Results

We tested Radix-k image compositing and parallel particle tracing on some of the largest supercomputers in the world and present selected results below. The Argonne Leadership Computing Facility has the IBM Blue Gene/P (BG/P) *Intrepid* system while the Oak Ridge National Center for Computational Sciences maintains the Cray XT5 *Jaguar* system. Intrepid has 160 K cores while Jaguar has 219 K cores; Jaguar is currently the fastest supercomputer in the world according to the June 2010 Top 500 listing, while Intrepid is currently ranked ninth.

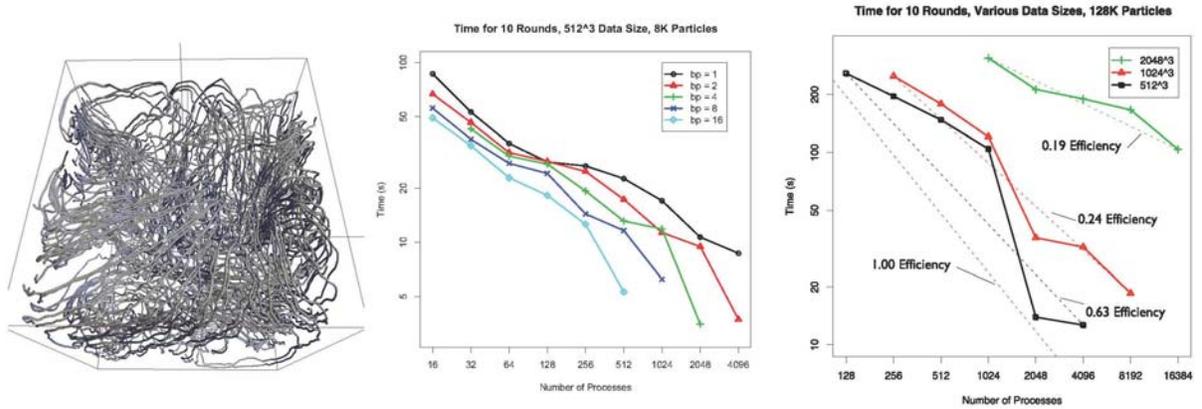
Figure 3 shows the results of testing Radix-k in a volume rendering application applied to a core-collapse supernova dataset on both Intrepid and Jaguar. In this research, bounding box and run-length encoding optimizations were implemented in Radix-k, and the compositing algorithm was inserted in our parallel volume rendering code [7].



**Figure 3.** When optimizations such as bounding boxes and run-length encoding are implemented in Radix-k for the volume rendering of core-collapse supernovae (left), the compositing time (center) scales on both Intrepid and Jaguar out to 32 K processes, plotted in log-log scale. Speedup of Radix-k over binary swap is shown at the right.

Bounding box and run-length encoding optimizations permitted the use of higher k-values than would otherwise be possible. The combination of accelerations with higher k-values allowed us to scale to 32,768 processes, as shown in the center panel of Figure 3. Tests were conducted at three zoom levels with the camera facing down the z-axis. The image size for these tests was 64 megapixels, a wall-size image with the same resolution as 32 HD TVs. At 32,768 processes, such an image can be composited in 0.08 seconds, or at 12.5 frames per second. The right panel of Figure 3 shows the speedup of Radix-k over binary swap in the same application, with identical optimizations applied. Up to five times faster performance resulted in some instances, and at least three times shorter compositing time was reported in most cases on Intrepid. On Jaguar, Radix-k is approximately 1.5 times faster than binary swap.

Figure 4 shows initial results of our parallel particle tracing algorithm on a dataset of the mixing behavior of warm and cold water in a confined region. The data have been resampled from their original topology onto a regular grid. Particle tracing consists of a number of iterations; one iteration involves numerically integrating the particle along the flow field until it reaches the boundary of its current block and then passing the particle to the neighboring block. Our tests ran for 10 iterations; the left panel of Figure 4 shows the result for a small number of particles, 400 in total. A single time-step of data is used here to model static flow.



**Figure 4.** The scalability of parallel nearest-neighbor communication for particle tracing of thermal hydraulics data is plotted in log-log scale. The left panel shows 400 particles tracing streamlines in this flow field. The center panel shows time for 8 K particles and a data size of  $512^3$ . Five curves are shown for different partitioning strategies: one block per process and round-robin partitioning with 2, 4, 8, and 16 blocks per process. In the right panel, 128 K particles are traced in three data sizes:  $512^3$  (134 million cells),  $1024^3$  (1 billion cells), and  $2048^3$  (8 billion cells). End-to-end scaling efficiency includes I/O (reading the vector dataset from storage and writing the output particle traces).

The center panel shows strong scaling up to 4,096 processes. The data size is  $512^3$ , and this time 8,192 total particles are used. Currently a simple round-robin distribution scheme is used to balance the computational load among processes; a process contains one or more data blocks distributed in the volume. The center panel shows distributions ranging from one block per process to 16 blocks per process. In almost all cases, the performance improves as the number of blocks per process increases; load is more likely to be distributed evenly and the overhead of managing multiple blocks remains small. Obviously, there is a limit to the effectiveness of a process having many small blocks: the cost of aggregating their particles into one message increases; moreover, the ratio of their surface area to volume grows, resulting in more communicating and less computing.

We recognize that round-robin distribution does not always produce acceptable load balancing. In our case, randomly seeding a dense set of particles throughout the domain works in our favor, but this need not be the case, as demonstrated by Pugmire et al. [6]. We are investigating load-balanced partitioning under less ideal conditions.

Continuing with this example, the right panel shows the scalability of larger data size and number of particles. Here, 131,072 seeds are randomly placed in the domain. Three sizes of the same thermal hydraulics data are tested:  $512^3$ ,  $1024^3$ , and  $2048^3$ ; the larger sizes were generated by upsampling the original size. As in the center panel, all of the data points represent end-to-end time including I/O. Tracing 131,072 particles in the smallest data size can be accomplished in 13 seconds; the medium data size in 19 seconds, and the largest data size in about 1-1/2 minutes (approximately 50% of this time is I/O). End-to-end scaling efficiency is also plotted for the three curves; in order from smallest to largest data size, these efficiencies are 63%, 24%, and 19%, respectively.

#### 4. Summary

Data movement is a critical part of analysis operations at scale. Any nontrivial parallel decomposition of analysis tasks requires communication among processes that can amount to a significant portion of the total analysis run time. We investigated and optimized communication motifs for two commonly used operations: global reduction and nearest-neighbor communication.

Image composition in data-parallel rendering is the motivation for exploring new global reduction algorithms. With Radix-k, we reduced compositing time by up to a factor of five compared to binary swap and composed wall-size images at nearly interactive rates. We are currently implementing the Radix-k algorithm in the IceT compositing library [8], which will enable its use in production visualization tools.

Parallel particle tracing motivated our work in nearest-neighbor communication. We successfully scaled our parallel particle tracing code to 16,384 processes on data sizes up to 8 billion grid points, or 96 GB. We are actively researching load-balancing and data-partitioning algorithms, benchmarking time-varying results, and testing our code on adaptive mesh grids as well.

#### Acknowledgments

We thank John Blondin, Tony Mezzacappa, Paul Fischer, and Aleks Obabko, the Argonne Leadership Computing Facility, and the National Center for Computational Sciences at Oak Ridge National Laboratory. This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357. Work is also supported by DOE with agreement No. DE-FC02-06ER25777.

#### References

- [1] Dongarra J. 2009. International exascale software project roadmap, draft report, Tech. rep. [www.exascale.org](http://www.exascale.org).
- [2] Hsu W M. 1993. *Proc. 1993 Parallel Rendering Symposium*, Segmented Ray Casting for Data Parallel Volume Rendering (San Jose, CA) pp. 7–14.
- [3] Ma K L, Painter J S, Hansen C D, and Krogh M F. 1994. *IEEE Computer Graphics and Applications*, Parallel Volume Rendering Using Binary-Swap Compositing **14** pp. 59–68.
- [4] Peterka T, Goodell D, Ross R, Shen H W, and Thakur R. 2009. *SC'09: Proceedings of the Conference on High Performance Computing Networking, Storage, and Analysis*, A Configurable Algorithm for Parallel Image-Compositing Applications (New York, NY, USA: ACM) pp. 1–10.
- [5] Kendall W, Peterka T, Huang J, Shen H W, and Ross R. 2010. *Proceedings of Eurographics Symposium on Parallel Graphics and Visualization EG PGM'10*, Accelerating and Benchmarking Radix-k Image Compositing at Large Scale (Norrköping, Sweden).
- [6] Pugmire D, Childs H, Garth C, Ahern S, and Weber G H. 2009. *SC'09: Proceedings of the Conference on High Performance Computing Networking, Storage, and Analysis*, Scalable Computation of Streamlines on Very Large Datasets (New York, NY, USA: ACM) pp. 1–12.
- [7] Peterka T, Yu H, Ross R, Ma K L, and Latham R. 2009. *ICPP 09: Proceedings of the 2009 International Conference on Parallel Processing*, End-to-End Study of Parallel Volume Rendering on the IBM Blue Gene/P (Washington, DC, USA: IEEE) pp. 566–573.
- [8] Moreland K, Wylie B, and Pavlakos C. 2001. *PVG'01: Proceedings of the IEEE 2001 Symposium on Parallel and Large-Data Visualization and Graphics*, Sort-last Parallel Rendering for Viewing Extremely Large Data Sets on Tile Displays (Piscataway, NJ, USA: IEEE Press) pp. 85–9.2