Visualization Tool for Development of Communication Algorithms and a Case Study Using the K Computer

Syunji Yazaki
The University of Electro-Communications
Tokyo, Japan
Email: yazaki.syunji@uec.ac.jp

Ryohei Suzuki
Cresco Ltd.
Tokyo, Japan
Email: r-suzuki@cresco.co.jp

Fumiyoshi Shoji
RIKEN Advanced Institute for Computational Science
Hyogo, Japan
Email: shoji@riken.jp

Kenichi Miura
Fujitsu Limited
Kanagawa, Japan
Email: k.miura@jp.fujitsu.com

Hiroaki Ishihata
Tokyo University of Technology
Tokyo, Japan
Email: ishihata@stf.teu.ac.jp

***Abstract*—In this paper, we introduce our visualization tool, the Communication Log Viewer (CLV), that assists the development of collective communication algorithms. We also present visualization results as a case study. CLV visualizes information regarding node events and network statistics in linked multiple views. CLV also has a function for analyzing the results obtained from network simulators and actual machines in the same framework, which is useful when developers repeatedly test their algorithms on a simulator and an actual system. For a case study, we visually evaluated two all-to-all algorithms on the full system of the K computer that has 82,944 nodes. As a result, we confirmed that an optimized all-to-all algorithm implemented for the K computer performed better than an all-to-all implemented in Open MPI. We also confirmed that the barrier operation used in the K computer's Message Passing Interface (MPI) functions keep link utilization high. However, there is also a trade-off between the number of barriers and link utilization.**

***Keywords*–Visualization; Mesh/Torus network; All-to-all**

I. INTRODUCTION

Parallel application programmers frequently utilize collective communications implemented in Message Passing Interface (MPI) libraries to design applications. Collective communications usually produce a large number of communications, especially on the peta-scale parallel systems that consist of tens of thousands of nodes. In the applications running on such large systems, communication takes longer than computation.

Optimizing communication algorithms is an important means of maximizing the performance of parallel applications [1]. Many parallel systems listed in the Top500 [2], such as the Cray XK7 [3], Blue Gene/Q [4], and K computer [5], employ mesh/torus topology. Mesh/torus topology generally provides better scalability with respect to hardware cost. However, the bisection bandwidth is relatively narrow compared to that of other topologies, such as Fattree [6] and Dragonfly [7].

Visualization tools that abstract and visualize communication behavior are necessary tools for optimizing communication algorithms [8]. Developers repeatedly test communication algorithms under development on network simulators and actual systems to find potential areas for optimization. The test results are usually obtained as huge logfiles and extensive numerical data. Looking at the logfiles and numerical data alone, it is difficult to determine potential areas for optimization.

We briefly presented our visualization tool, the Communication Log Viewer (CLV), that supports the design of collective communication algorithms in [9]. Our tool has a function that visualizes both the results obtained from a network simulator and an actual system in the same framework. Our tool also visualizes both events that occur in the node and statistics regarding traffic in the network simultaneously with linked multiple views. This enables the user to distinguish quickly which events in the nodes correspond to which congested network links.

In this paper, we describe the details of CLV that can visualize both the node events and network statistics. We also show a visual evaluation of all-to-all on a full system of the K computer that has 82,944 nodes. In the rest of this paper, Section II explains the workflow for developing communication algorithms and Section III presents related work. Section IV then describes the features of CLV. Section V provides a case study, and Section VI concludes the paper.

II. WORKFLOW AND REQUIREMENTS FOR VISUALIZATION

A workflow to develop collective communication algorithms involves the following steps.

1) Designing an algorithm that takes into account the network architecture of the target system
2) Testing the algorithm on a network simulator and generating simulation logfiles
3) Analyzing and evaluating the behavior and efficiency of the algorithm based on the information in the logfiles
4) Implementing the algorithm on the target system, if the algorithm has achieved the expected performance in the simulation
5) Evaluating the algorithm based on logfiles and numerical data obtained from the performance counters of the target system

Considering this workflow, the following functions are needed in visualization tools:

R1 Visualizing the simultaneous network statistics and node events with concise association
R2 Mapping the information to the actual network structure of the target system
R3 Showing the information in multiple linked views
R4 Displaying concise information by filtering
R5 Supporting the outputs obtained both from simulators and actual systems in the same framework

 ISBN: 978-1-61208-389-6

Visualizing both the network statistics and node events in the same tool allows the developer to easily find bottleneck links and the events that cause them. Additionally, the developer also can intuitively understand which part of a network is heavily used by observing the topologically mapped information. The tool should provide multiple views with multiple levels of abstraction because needed information is sometimes lost at different levels of abstraction. In addition, information that is not the current focus of the investigation must be filtered out by the developer. As discussed above, developers evaluate their algorithms on a simulator and target system. Considering this, the tools should analyze both outputs obtained from simulators and actual systems within a single framework.

III. Related work

Bhatele et al. visualized communications that occurred in an Adaptive Mesh Refinement (AMR) application [10]. Existing visualization tools, such as Jumpshot [11], ParaProf [12], and Vampir [13], visualize a profiler's outputs. Other well-known tools such as Open|SpeedShop [14] and TAU [15] provide an integrated environment for performance analysis. These tools summarize outputs obtained by the profiles in graphs, tables, and figures. They are useful for analyzing events that occur in the nodes. However, these tools basically do not support network statistics because profilers cannot be designed to acquire the network statistics.

Minkenberg and Rodriguez proposed a simulation environment to support the development of high performance computing systems [16]. An MPI task simulator works with a network simulator in this simulation environment to emulate parallel applications running with specific topology and hardware construction. This environment uses Paraver [17] to visualize the communication.

SimCon [18] was developed to find appropriate overlay networks for running parallel applications. This simulator displays networks based on physical distances and links between pairs of communicating nodes. Users can understand the communication situation from the simulation results. Gamblin et al. [19] also developed a tool to evaluate parallel applications to optimize node mapping. This tool presents a topological view of the network and places the information on this view. By observing this view, users can intuitively check the communication situation. However, these tools also require external tools to obtain network statistics.

Landge et al. [20] developed a visualization tool to analyze packet traffic on the torus network. This tool focuses especially on recent Blue Gene systems. It provides two linked views to show an overview of the packet traffic. One view is a 2D projection, mainly used to show brief trends and patterns of the traffic. The other view is a 3D topological view that maps traffic patterns to the physical structure of a target network. This tool allows application developers to understand link utilization and find bottleneck links intuitively. However, application developers still need to combine it with another tool to inspect the cause of the bottleneck.

IV. CLV

A. System summary

CLV is designed to implement two requirements, R1 and R5, into a single tool. Existing visualization tools introduced in Section III have some features that implement R2, R3, and R4. R1 and R5 can also be realized by combining multiple tools. However, there is no integrated tool that implements R1 and R5. CLV also has basic functions that are implemented in existing visualization tools. In addition, CLV can read and analyze logfiles obtained both from simulators and actual systems in the same framework. This feature is useful when developers need to test their algorithms on a simulator and an actual system repeatedly. As of now, Booksim [21] and Message Flow Simulator (MFS) [22] can be used as the simulators, and the K computer using the Tofu Performance Analysis (Tofu PA) [23], is targeted for the actual system. The developer can use existing simulators or performance measurement tools for traffic data acquisition. CLV provides an utility to convert the data format to feed the data into CLV.

CLV mainly provides two views: time series and topological. The time series view summarizes network statistics in the time series. This view is also used to select a particular time on which to focus. All views in CLV are linked from the time series view. When a user selects a time in the time series view, all other views show the data at that time. The topological view maps the events that occurred in the nodes and network statistics to the physical structure of the network in 3D space. As of now, CLV can visualize the messages sent as the events. This view shows link utilization as well as the duration of the communication delay. Users can filter the information by selecting nodes or links in this view. The users also can understand where the messages sent events occurred in the network from this view.

CLV was implemented in C++ with OpenGL library. Thus, the CLV can be compiled on any OS that has OpenGL implementation. We also used the OpenGL Utility Toolkit (GLUT) [24] to construct the user interface.

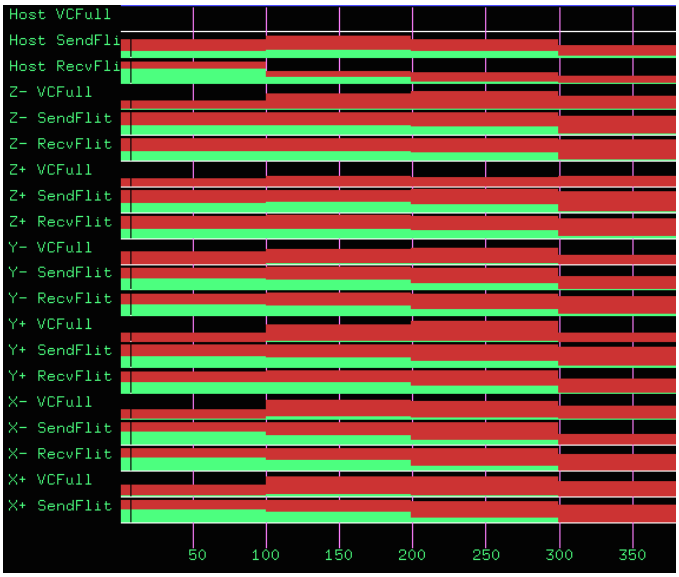


Figure 1. Example of the time series view. Maximum and average values are represented by red and green bars, respectively.

B. Time series view

Figure 1 shows an example of the time series view. This screenshot shows the link utilization for sending (SendFlit) and receiving (RecvFlit) for all links in each dimension. It also shows the communication delay (VCFull). The horizontal

 ISBN: 978-1-61208-389-6

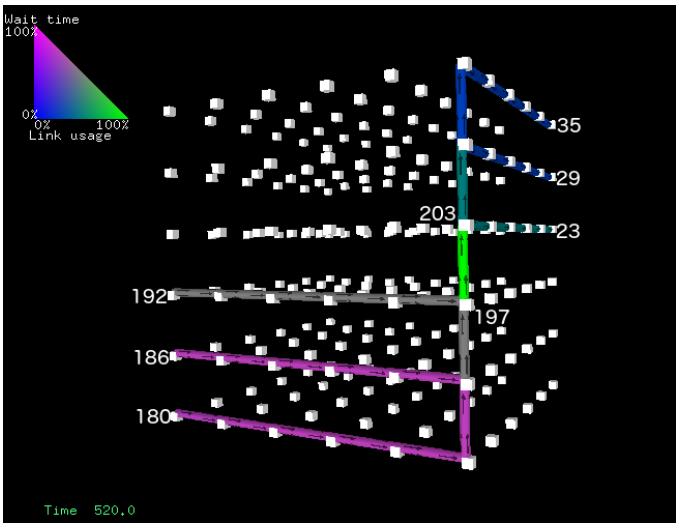


Figure 2. Examples of the topological view showing link utilization and communication delay

line represents the elapsed times. The time unit is an interval time of data specified in the logfiles. The link utilization and communication delay are defined as the fraction of bandwidth and number of wait cycles in each time unit, respectively. Dimensions "X," "Y," and "Z" correspond to the first, second, and third dimensions of the network. "Host" refers to a link connected from a node to a router (switch) in the network. The "+" and "-" symbols indicate the directions of the links.

The graphs in the screenshot indicate the maximum and average values by the red and green bars, respectively. If the average value (green) is close to the maximum value (red), most messages on all the links in those directions and dimensions have transferred efficiently. In contrast, if these values are far apart, a small number of links are heavily used, while most of the remaining links are not utilized.

C. Topological view

In the topological view, information is mapped to the physical structure of a network topology in 3D space. Figure 2 shows an example of the topological view. The triangle in the top left corner of the figure is a legend that specifies link utilization and communication delay by color.

The situation in Figure 2 is such that the three source nodes at the lower front left nodes (180, 186, and 192) are communicating with three other destination nodes at the upper back right (23, 29, and 35). We added the node numbers to the screenshot for this explanation.

Small arrows are printed on the links. These are showing a direction of the messages sent. From the direction of the arrows, the users can understand that the events of the messages sent occurred in node 180, 186, and 192.

The colors represent link utilization and communication delay. The green link between nodes 197 and 203 indicates that this link is fully used (100%). Furthermore, we can see that the links around this green link are colored dark green and gray. The user can determine that these links have lower link utilization from the legend. There are purple and gray links in the lower half of the view. This indicates that the packets on the purple links wait for a long time, while packets on the gray links only wait for a short time. From this view, the user can guess that the packets on the purple links are blocked while the packets on the gray links are transmitted through the green links. In the same way, users can guess the cause of low link utilization from this view.

V. Case Study

A. Comparing all-to-all algorithms on a simulator

We first compared two all-to-all algorithms using CLV on simulation results. We simulated all-to-all communications on a $10 \times 10 \times 10$ 3D mesh network using Booksim. In this simulation, each node sends one message to 999 other nodes. One message consists of 200 flits, and one flit is sent per cycle. Dimension order routing is used as the routing algorithm. Node numbers (ranks) are assigned in the order of x, y, and z dimensions.

We chose to compare two all-to-all algorithms using a simple spread algorithm and an algorithm optimized for torus. We refer to the simple spread and torus optimized algorithms as A2A and A2AT, respectively, in this paper. A2A is used in many MPI libraries such as MPICH [25], MVAPICH [26], and Open MPI [27]. A destination node number in A2A is calculated by $(src + i) \mod N$ $(i = 1, 2, \ldots, N-1)$. Here, src and N represent a source node number and the total number of nodes in the network, respectively. A2AT is an optimum topology-aware algorithm for mesh/torus networks that we previously proposed [28]. We have shown that A2AT performs better than a modified version of A2A. Here, we investigate the communication efficiency with respect to performance using the CLV visualization result.

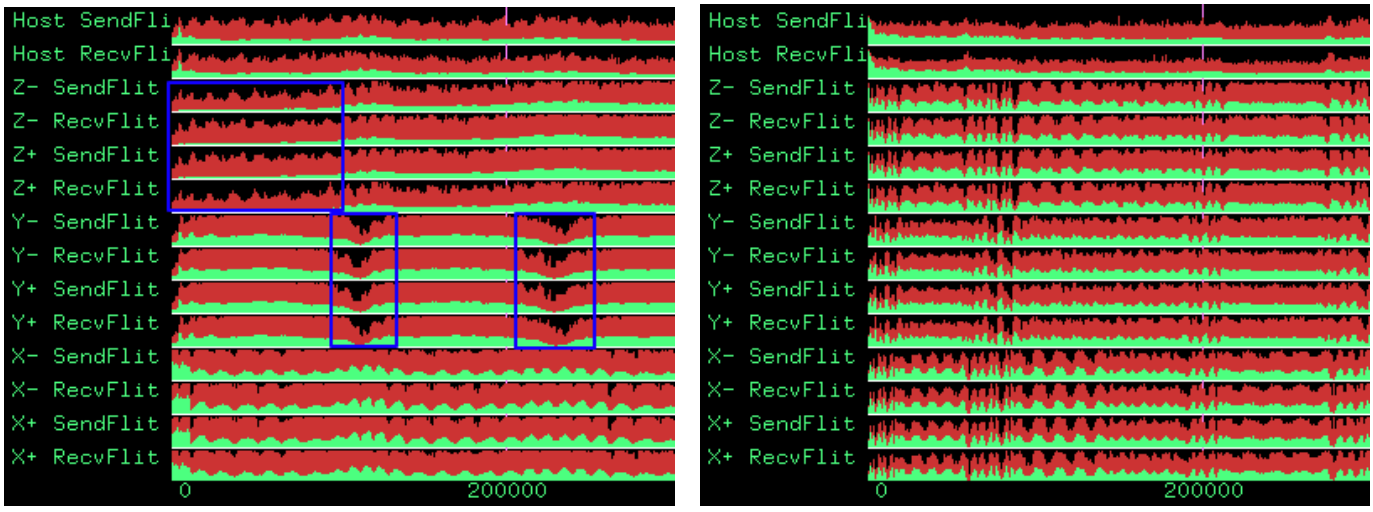
Figure 3 presents the visualization results of both algorithms in the time series view. The figures show the first part of each simulation result. Link utilization in the y and z dimensions in Figure 3(a) decreases several times, as indicated by the rectangles. This indicates that A2A does not utilize links in the y and z dimensions at this time. In contrast, there are no large black spaces in Figure 3(b). This indicates that A2AT utilizes all links at any time. We confirmed from these results that A2AT provides better performance by utilizing the links in y and z dimensions to avoid using the bottleneck links.

B. Comparing all-to-all algorithms on the K computer

1) Preparation: The K computer implements a tuned all-to-all algorithm in its MPI library [29]. We refer to this MPI library as the Tofu MPI in this paper because the K computer employs a 6-dimensional mesh/torus network called the Tofu interconnect. In this algorithm, the order in which the messages are sent is modified while considering the routing algorithm of the K computer. The barrier operation is also used to make the messages sent in this algorithm uniform. We evaluated this algorithm on a full system of the K computer at the RIKEN Advanced Institute of Computer Science in Japan [30].

The fourth, fifth, and sixth dimensions are labeled a, b, and c, respectively, in the Tofu interconnect. The K computer has 82,944 $(= 24 \times 18 \times 16 \times 2 \times 3 \times 2)$ computation nodes, excluding I/O nodes. The physical construction of the K computer was $24 \times 18 \times 17 \times 2 \times 3 \times 2$. However, nodes located on $z = 0$ are reserved for I/O nodes. Therefore, the full size of the system that could be used was $24 \times 18 \times 16 \times 2 \times 3 \times 2$ because no I/O nodes joined into the MPI communicator. Note that the y, a, and c dimensions are mesh networks while the others are

 ISBN: 978-1-61208-389-6



(a) A2A (b) A2AT

Figure 3. Comparison of simulation results for A2A and A2AT on a $10 \times 10 \times 10$ 3D mesh network in the time series view.

TABLE I. SUMMARY OF NETWORK STATISTICS OBTAINED WITH THE TOFU PA DURING ALL-TO-ALL COMMUNICATION WITH 32,768 BYTES of DATA

	Run time	Samples	Period	Raw logfile size
Tofu MPI	2.77 s	= 277	× 10 ms	7.9 GB
Open MPI	22.94 s	= 2,294	× 10 ms	59 GB

torus networks.

We ran the tuned all-to-all algorithms implemented in Tofu MPI and an existing all-to-all algorithm implemented in Open MPI with 32,768 bytes of data. We also obtained logfiles using the Tofu PA. TABLE I summarizes the logfiles that we obtained. The total amount of the messages to be sent was 205.0 TB.

Here, we estimate the ideal communication time of an all-to-all communication on the K computer. The lower bound of all-to-all communication time (L) for a mesh/torus network can be generalized as

$$L = (\alpha \lfloor \frac{k_b}{2} \rfloor \lceil \frac{k_b}{2} \rceil (\prod_{i=0}^{n-1} k_i) m)/B. \qquad (k_i \neq k_b) \qquad (1)$$

Here, k_b represents the sizes of the dimensions that have bottleneck links. The links in the longest dimension are the bottlenecks. Variables n and k_i indicate the number of dimensions and size of the i-th dimension, respectively. Note that the dimension corresponding to k_b will be skipped. Parameters m and B represent message size and link bandwidth, respectively. Parameter α is set to $1/2$ or 1 depending on whether the bottleneck dimension has wrap-around links (torus) or not (mesh).

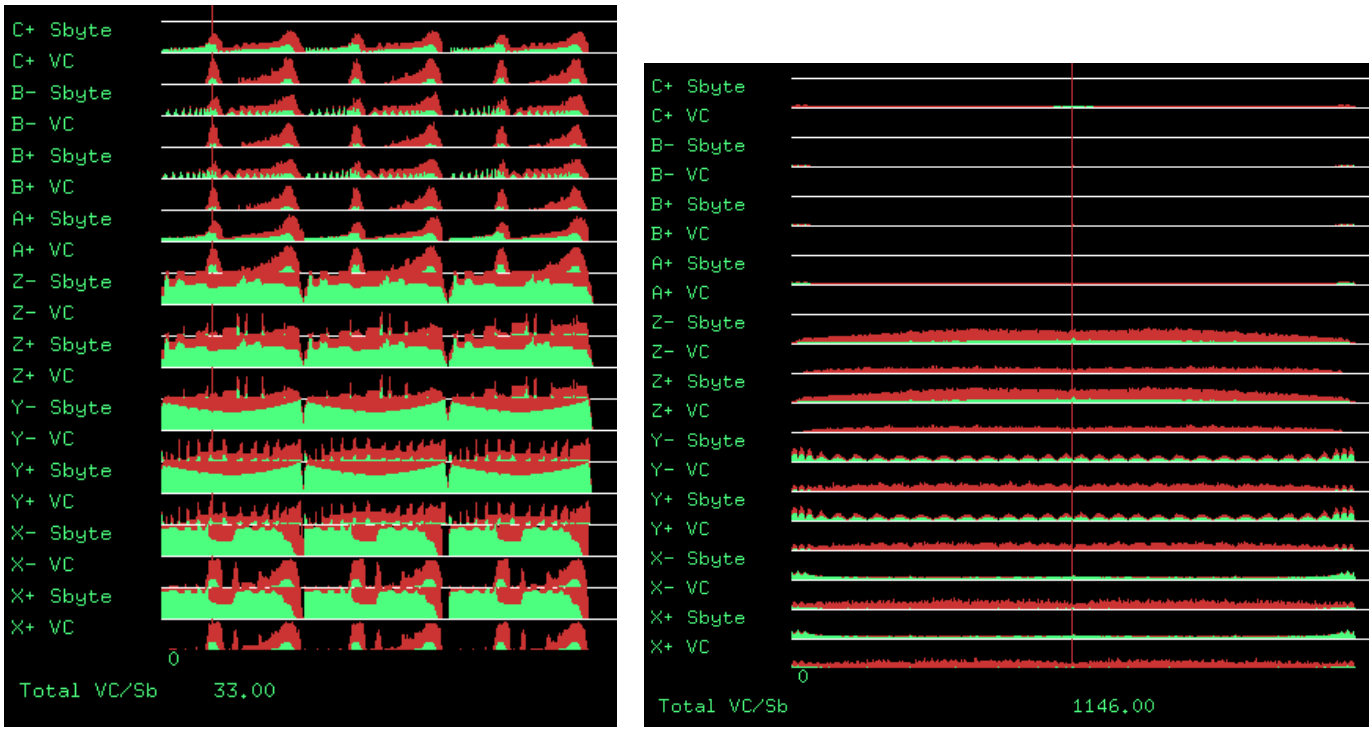
We need to determine a dimension size k_b that includes bottleneck links to calculate an ideal communication time from (1). Links in the longest dimension are basically bottleneck links. Yet, the x dimension is longer than the y dimension in the K computer network. However, the bandwidth of the x dimension is twice that of the y dimension because the x dimension is a torus network. Therefore, the bottleneck links of the K computer are in the y dimension. The effective bandwidth of a single link in the K computer was 4.76 GB/s. Thus, the optimum communication time for an all-to-all communication in the K computer can be calculated to be 2.57 s from (1). From the table, we can find that the Tofu MPI spent 2.77 s on an all-to-all communication. This means that the tuned all-to-all algorithm achieved 1.08 times the optimum communication time.

2) Visualization of all-to-all algorithms: We then visualized an all-to-all communication using the logfiles obtained by the Tofu PA. The logfiles included various network statistics. However, the logfiles do not include information corresponding to the node events such as amount of traffic from the nodes to the nearest routers. Thus, we only visualized the network statistics at this time. We visualized the fraction of bandwidth and communication delay based on the number of sent packets and the number of wait cycles needed to inject packets.

Figure 4 shows screenshots of the visualization results in the time series view. The link utilization and message delay are represented by "Sbyte" and "VC" in Figure 4. Both average and maximum link utilization (Sbyte) of the Tofu MPI that is shown in Figure 4(a), were higher than those of Open MPI that is shown in Figure 4(b). By simply looking at these visualization results, we can intuitively understand that the Tofu MPI utilized more links than Open MPI. Link utilizations in the a, b, and c dimensions are relatively low compared to other dimensions in both figures. Dimensions a, b, and c construct a small $2 \times 3 \times 2$ sub-network, hence links in this small sub-network are not heavily used.

We also found that link utilization in the x, y, and z dimensions drops twice in Figure 4(a). The all-to-all algorithm implemented in the Tofu MPI uses the barrier operations at these two points to send messages uniformly. Right before the barrier points, it can be seen that the average link utilization indicated in green in the x dimension is declining. It then recovers after the barrier points. However, there are also black gaps around the barrier points. This indicates that the communications are being blocked during these cycles. This could be the barrier penalty. From this observation, we concluded that the barrier operations work well to keep the average link

 ISBN: 978-1-61208-389-6



(a) Tofu MPI (b) Open MPI

Figure 4. Comparison of all-to-all algorithms on full system of the K computer (82,944 nodes) in the time series view.

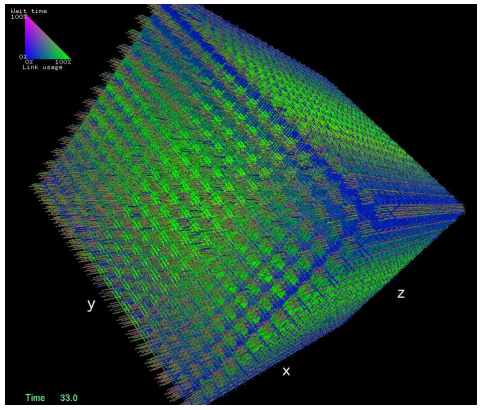
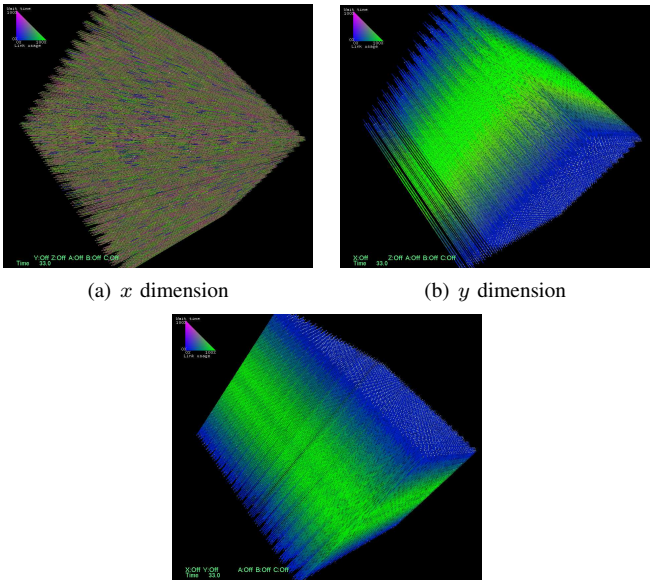


Figure 5. All-to-all communication using the Tofu MPI on full system of the K computer (82,944 nodes) in the topological view at 0.33 s



(a) x dimension (b) y dimension

(c) z dimension

Figure 6. Views of Figure 5 filtered by dimension.

utilization high. However, there is a trade-off between the number of barriers and average link utilization.

Figure 5 shows a screenshot of the topological view of the Tofu MPI at 0.33 s. The red vertical line in Figure 4(a) indicates this time. We can easily focus on this time by selecting it in the time series view. Links near the middle of the y and z dimensions are green, as can be seen in Figure 5. This indicates that these links were fully utilized. However, links near the edge of the network that were not used as much are colored blue. This is because the y and z dimensions were a mesh network, which has no wrap-around links. Thus, links near the corner of the network were not used much more than the links in the central area of the network.

We next filtered this topological view by each dimension for further analysis. The visualization results are shown in Figure 6. In Figure 6(a), we can see many brown links in the x dimension. In contrast, there are no brown links, in the y and z dimensions in Figures 6(b) and (c). This indicates that many packets were blocked only in the x dimension. The K computer

employs routing that sends messages to the x dimension first. Thus, in all-to-all communication, many packets are injected into links in x dimension within a short time slot. This leads to congestion in the x dimension. However, the delay is not critical, as many links are brown. We suppose that messages are sent at a constant pace by utilizing the barrier operations explained above.

VI. Conclusions

We introduced CLV, a visualization tool to assist the development of collective communication algorithms. CLV provides time series and topological views that visualized information about node events and network statistics.

We also presented a case study to demonstrate the effectiveness of CLV. We compared the performances of simple (A2A) and topology-aware (A2AT) designs of all-to-all algorithms both in a simulation and on an actual system. We ran two all-to-all communication algorithms implemented in the Tofu MPI and Open MPI to compare their performances. The algorithm implemented in the Tofu MPI was optimized for the K computer. We found that the Tofu MPI achieved 1.08 times the optimum all-to-all communication time on the full system of the K computer that includes 82,944 nodes. We confirmed that the barrier operation used in the Tofu MPI effectively keeps link utilization high. We also pointed out that there is a trade-off between the number of barriers and link utilization for further optimization.

As of now, CLV can only visualize the messages sent as the node events. For the future work, other node events that affect network utilization should be visualized to help optimization of communication algorithms.

VII. Acknowledgments

We would like to thank Mr. Yuji Oinaga and Mr. Naoki Shinjo of Fujitsu Ltd. for supporting this work. A part of this work was supported by JSPS KAKENHI Grant Number 2533016.

References

[1] Y. Gong, B. He, and J. Zhong, "Network performance aware MPI collective communication operations in the cloud," IEEE Transactions on Parallel and Distributed Systems, no. 1, p. 1, 2013.

[2] Top500 lists. [Online]. Available from: http://www.top500.org/, Feb., 2015.

[3] B. Bland, "Titan - early experience with the titan system at oak ridge national laboratory," in 2012 SC Companion: High Performance Computing, Networking, Storage and Analysis (SCC), pp. 2189–2211, Nov. 2012.

[4] S. Kumar, A. Mamidala, P. Heidelberger, D. Chen, and D. Faraj, "Optimization of mpi collective operations on the ibm Blue Gene/Q supercomputer," Intl. Journal of High Performance Computing Applications, vol. 28, no. 4, pp. 450–464, 2014.

[5] Y. Ajima, T. Inoue, S. Hiramoto, T. Shimizu, and Y. Takagi, "The tofu interconnect," IEEE Micro, vol. 32, no. 1, pp. 21–31, 2012.

[6] C. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," Computers, IEEE Transactions on, vol. C-34, no. 10, pp. 892–901, Oct. 1985.

[7] J. Kim, W. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in 35th Intl. Symposium on Computer Architecture, 2008. ISCA '08., pp. 77–88, Jun. 2008.

[8] K. E. Isaacs, A. Gimnez, I. Jusufi, T. Gamblin, A. Bhatele, M. Schulz et al., "State of the art of performance visualization," in Proc. of Eurographics Conf. on Visualization 2014, Jun. 2014, lLNL-CONF-652873.

[9] R. Suzuki and H. Ishihata, "Visualization tool for network topology aware communication algorithm development," in Supercomputing 2012 Research Poster, Nov. 2012.

[10] A. Bhatele, T. Gamblin, K. E. Isaacs, B. T. N. Gunney, M. Schulz, P.-T. Bremer et al., "Novel views of performance data to analyze large-scale adaptive applications," in Supercomputing 2012, pp. 31:1–31:11. Los Alamitos, CA, USA: IEEE Computer Society Press, 2012.

[11] A. Chan, W. Gropp, and E. Lusk, "An efficient format for nearly constant-time access to arbitrary time intervals in large trace files," Scientific Programming, vol. 16, pp. 155–165, Apr. 2008.

[12] R. Bell, A. Malony, and S. Shende, "ParaProf: A portable, extensible, and scalable tool for parallel performance profile analysis," in Proc. of Intl. Euro-Par Conf., pp. 17–26, Aug. 2003.

[13] A. Knupfer, H. Brunst, J. Doleschal, M. Jurenz, M. Lieber, H. Mickler et al., The Vampir Performance Analysis Tool-Set, M. Resch, R. Keller, V. Himmler, B. Krammer, and A. Schulz, Eds. Springer Berlin Heidelberg, 2008.

[14] M. Schulz, J. Galarowicz, D. Maghrak, W. Hachfeld, D. Montoya, and S. ord, "Open|speedshop: An open source infrastructure for parallel performance analysis," Scientific Programming, vol. 16, no. 2-3, pp. 105–121, 2008.

[15] S. Shende and A. D. Malony, "The TAU parallel performance system," in Intl. Journal of High Performance Computing Applications, vol. 20, no. 2, pp. 305–312, 2006.

[16] C. Minkenberg and G. Rodriguez, "Trace-driven co-simulation of high-performance computing systems using omnet++," in Proc. of the 2nd Intl. Conf. on Simulation Tools and Techniques, pp. 65:1–65:8, 2009.

[17] G. Jost, H. Jin, J. Labarta, J. Gimenez, and J. Caubet, "Performance analysis of multilevel parallel applications on shared memory architectures," in Proc. of Intl. Parallel and Distributed Processing Symposium, pp. 22–26, Apr. 2003.

[18] M. Esch, J. Botev, H. Schloss, A.Hohfeld, I. Scholtes, and B. Zech, "SimCon - a simulation and visualization environment for overlay networks and large-scale applications," in Proc. of the 1st Intl. Conf. on Simulation Tools and Techniques for Communications, Networks and Systems, pp. 1–9, 2008.

[19] T. Gamblin, M. Schulz, T. Bremer, J. Levine, and V. Pascucci, "Intuitive performance visualization techniques for topological analysis on capability machines," in IPSJ SIG Technical Reports, vol. 2011, no. 51, pp. 1–8, Jul. 2011.

[20] A. Landge, J. Levine, A. Bhatele, K. E. Isaacs, T. Gamblin, M. Schulz et al., "Visualizing network traffic to understand the performance of massively parallel simulations," IEEE Transactions on Visualization and Computer Graphics, vol. 18, no. 12, pp. 2467–2476, Dec. 2012.

[21] W. Dally and B. Towles, Principles and Practices of Interconnection Networks. M. Kaufmann Publishers Inc.: San Francisco, 2003.

[22] S. Yazaki and H. Ishihata, "Message flow simulator for evaluating communication algorithms," in Proc. of The Ninth IASTED Intl. Conf. on Parallel and Distributed Computing and Networks 2010, pp. 291–298, Feb. 2010.

[23] K. Ida, Y. Ohno, and S. Inoue, "Performance profiling and debugging on the k computer," FUJITSU, vol. 63, no. 3, pp. 287–331, May 2012 (in Japanese).

[24] GLUT. [Online]. Available from: https://www.opengl.org/resources/libraries/glut/, Feb., 2015.

[25] MPICH. [Online]. Available from: http://www.mpich.org/, Feb., 2015.

[26] MVAPICH. [Online]. Available from: http://mvapich.cse.ohio-state.edu/, Feb., 2015.

[27] Open MPI. [Online]. Available from: http://www.open-mpi.org/, Feb., 2015.

[28] S. Yazaki, H. Takaue, Y. Ajima, T. Shimizu, and H. Ishihata, "An efficient all-to-all communication algorithm for mesh/torus networks," in Proc. of The 10th IEEE Intl. Symposium on Parallel and Distributed Processing with Application, pp. 277–284, 2012.

[29] Fujitsu, "Information processing system and method of controlling the same across-reference to related application," US Patent no. 14/087043, 2012.

[30] RIKEN Advanced Institute for Computational Science. [Online]. Available from: http://www.aics.riken.jp/en/, Feb., 2015.

 ISBN: 978-1-61208-389-6