Technical Note

Evaluation of Three Quads Using Matrix Transpose

Jiachao Zhang[1,a)] Shin-ichiro Mori[1,b)] Shinji Tomita[2,†1,c)]

Received: October 13, 2017, Accepted: December 8, 2017

Abstract: For medium scale commodity cluster, a high cost performance interconnection network named Three Quads was proposed. Topological features of Three Quads include 1) diameter of two, 2) embed-ability of 3D torus and many well-known topologies, and 3) fault tolerance and path diversity. All the hardware components of Three Quads are essentially small-scale and off-the-shelf commodity hardware to achieve superior cost performance. In this paper, the performance of the Three Quads is evaluated using a matrix transpose application and results show that the Three Quads gives appealing performance comparable to a large scale and expensive enterprise switch fabric.

Keywords: parallel processing, cluster computing, interconnection network, interactive simulation

1. Introduction

Low-latency data analysis of large streaming data is gradually becoming of greater importance as an HPC application. In this context, in analysis of temporal results of on-going simulation, in situ visualization for simulations plays an important role because it could avoid costly data transfer for post-visualization, and thus could save valuable supercomputer time in the exascale era [1]. Furthermore, in the interactive simulation environments, users may wish to steer the simulation process as a reaction to the visualization results, or data analysis, thus the simultaneous and interactive execution of simulation and visualization is required [2]. In such environments, two different kinds of communication patterns for simulation and visualization would co-exist at the same time, so that the system for the interactive simulation should have a network architecture which can efficiently deal with these types of communication. In order to satisfy these requirements, we had proposed a high cost performance interconnection network named Three Quads [3].

Three Quads was designed for the medium scale commodity cluster expecting that a system on this scale would serve as a computing resource for various interactive use cases. All the hardware components of Three Quads are essentially small-scale and off-the-shelf commodity switches, rather than an expensive large-scale enterprise switch, to achieve superior cost performance. In order to investigate this performance of Three Quads in real system [4], we have focused on a matrix transpose application as an demonstrative application and developed a parallel matrix transpose algorithm for Three Quads. We then measured performance on an actual system. As the results, we could confirm that Three Quads shows appealing performance comparable to a large-scale expensive enterprise switch fabric.

In the following sections, the architecture of Three Quads and the parallel matrix transpose algorithm are respectively described in Sections 2 and 3. Our experiment is then described in detail in Section 4.

2. Background

2.1 The Scube

The Scube is a 64 node PC-based cluster system equipped with multi-core CPU and GPU as accelerators on each node. The Scube is capable of simultaneously computing simulation and visualization tasks in parallel using both CPU and GPU. Through a host PC as the client terminal of simulation servers, users can steer simulations and observe the final result of the visualization task.

Since both simulation and visualization tasks themselves are expected to be executed in parallel using 64 nodes of Scube, the interconnection network is mandatory to have the capability to satisfy the bandwidth requirements for both tasks.

2.2 Three Quads

Three Quads is an interconnection network for medium scale PC cluster for interactive scientific simulation. For such a system, there exist several communication patterns, like 3D near-neighbor communication and 3D reduction type communication, which must be implemented efficiently. Needless to say, 3D near-neighbor communication pattern is mandatory to directly map 3D simulation space onto processor space. And 3D reduction type communication pattern is required to deliver and gather information to or from processors in both simulation and visualization applications, like real-time 3D image composition [5].

In addition to such architectural requirements, we also severely considered the hardware cost to implement the network. It was because we thought this kind of interactive simulation system would be owned by a relatively small organization such as a laboratory in a university or a research institute, as their dedicated resources, rather than a huge supercomputing center where the

[1] University of Fukui, Fukui 910–8507, Japan
[2] Kyoto University, Kyoto 606–8501, Japan
[†1] Presently with The Kyoto College of Graduate Studies for Informatics, Kyoto 606–8225, Japan
[a)] zhang@sylph.fuis.u-fukui.ac.jp
[b)] moris@u-fukui.ac.jp
[c)] tomita.shinji.v30@kyoto-u.jp

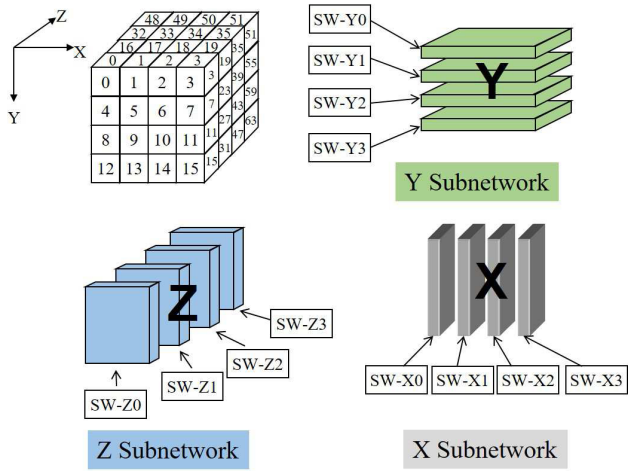

Fig. 1 Node assignment to switches in Three Quads (In case of 64 nodes).

batch and queue-based resource sharing is enforced. As a result, in the practical development of Scube, we focused on the medium scale commodity cluster system and limit the use of fairly expensive giant switch. Thus, Three Quads could be implemented merely with inexpensive small-scale Gigabit Ethernet switches. Nevertheless, Three Quads makes it possible to use of different kinds of simulation and visualization by utilizing its redundant subnetworks and appropriate scheduling of communications.

Figure 1 illustrates the topology of Three Quads. The up-left subfigure shows one possible nodes arrangement in the case of 64 nodes where each node of whole cluster is logically arranged along three axes, Z, Y, and X. According to its position along each axis, each node could be labeled with a 3D coordinates (N_z, N_y, N_x) $(0 \leq N_z, N_y, N_x \leq 3$, in 64 nodes case). The structure of Three Quads is built by three groups of switches (SW-Z, SW-Y, SW-X) that each consists of several independent small-scale off-the-shelf Gigabit Ethernet switches (GbE SW, for short). Nodes labeled with $(i, *, *)$ are connected to SW-Z_i, nodes labeled with $(*, j, *)$ are connected to SW-Y_j and nodes labeled with $(*, *, k)$ are connected to SW-X_k.

In "Three Quads", "Three" refers to there are three independent subnetworks, and "Quads" refers to the fact that each coverage of any GbE switch is the nodes locating inside a rectangular area of X-Y, Y-Z, or Z-X planes. Three Quads can be thought as a 2D extension of base-m 3-cube [6] [*1]. In this sense, Three Quads can be represented as a base-m^2 3-cube. Here we briefy summarize the main features as follows: diameter of two, embed-ability of many well-known topologies like 3D torus, fault tolerance and path diversity [3].

3. Parallel Matrix Transpose Algorithm for Three Quads

Matrix transposition is a fundamental and frequently used operation in numerical computations whereas along with all-to-all communications that incur severe network congestion and lower the overall network throughputs, in many network topologies. So we choose it as the representive application to evaluate the practical performance of Three Quads. Here we propose a parallel matrix transpose algorithm as the demonstrative application to show how matrix transposition could be solved efficiently on Three Quads without any congestion.

[*1] Through this extension, not only the 3D 7 neighbour communication but also 3D 19 neighbour communication [2], [7] can be achieved in one hop.

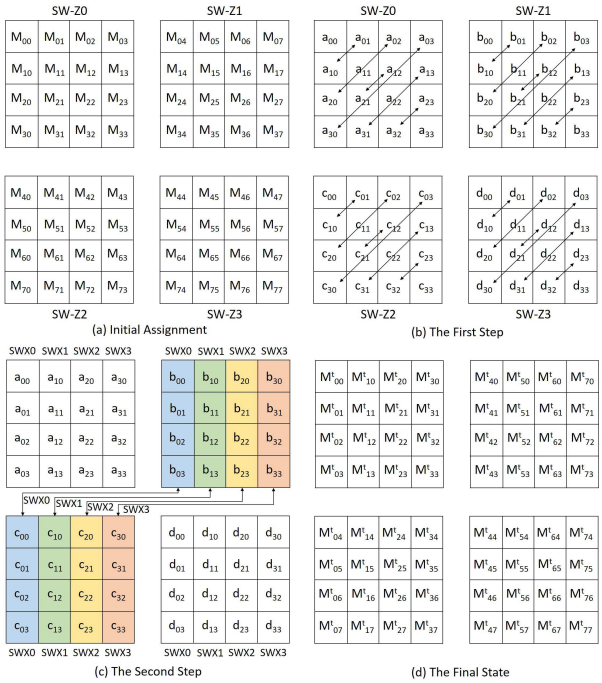

Fig. 2 Matrix transpose on Three Quads.

Assume that the size of the whole matrix to be transposed on 64 nodes Scube is M by M, and each node will be assigned a N by N submatrix ($M = 8N$). **Figure 2** describes the main process considering the two dimensional expansion of switches of Z-subnetwork. And we also assume the submatrix M_{ij} of whole matrix M is assigned to the node at the i-th row and j-th column in Fig. 2. For simplicity of explanation, the matrix M can be represented using four submatrices A, B, C, and D which could be further divided into sub-matrices $a_{ij}, b_{ij}, c_{ij}, d_{ij}$ $(0 \leq i, j \leq 3)$ as follows. Now the transpose of the whole matrix can be done in two steps.

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}. \tag{1}$$

First Step computes A^t, B^t, C^t, D^t. It's very obvious that any communication to get transpose of each submatrices in this step could be done within a single SW-Z switch. Therefore we can get a new matrix M' without any network congestion.

$$M' = \begin{pmatrix} A^t & B^t \\ C^t & D^t \end{pmatrix}. \tag{2}$$

Second Step swaps B^t and C^t. In this step, there is nothing to do for A^t and D^t. Just consider the 3D coordinates of two nodes in swapping b^t_{ij} and c^t_{ij}, we can find that the two digits of the coordinates of each communication pair's nodes are totally same. It means any communication in this step can be done in a single switch belonged to SW-X or SW-Y. So there is still no congestion in this step.

Further optimizations, like blocking and communication pipelining, of this algorithm make it possible to fuse these two steps into one step because the sub-networks used in these two

steps are distinct and all the communications required in this algorithm are congestion free.

4. Perfomance Evaluation Using Matrix Transpose

4.1 Experimental Environment

4.1.1 Detailed Implementation of Parallel Matrix Transpose Algorithm

In order to demonstrate the benefits of Three Quads topology, we have implemented the two-steps matrix transpose algorithm illustrated in the previous section. Whole 64-nodes of Scube cluster are used in this experiment [*2].

Each node will be in charge of partial submatrix and exchange its data with its corresponding partner in each step of the algorithms. Submatrix M_{ij} of size 8,192 by 8,192 is initially assigned to Node_k where $k = (i/4) \times 32 + (i\%4) \times 4 + (j/4) \times 16 + (j\%4)$, by default [*3].

The workloads of each node may differ according to their submatrix's location in whole, for example, some may require both step 1 and step 2, some may merely require one step, and nodes on the diagonal line have no need to perform any inter-node communication.

In order to demonstrate the possibility of network traffic scheduling, a synchronization of all nodes is enforced after step 1. Therefore, exactly one group of switches (SWX, SWY or SWZ) is used at each of step 1 and step 2. It means that the two-hop connectivity between any two nodes can be preserved without interfering with network traffic in step 1 and step 2.

Furthermore, to extract the performance of the full duplex communication between two nodes, we have improved the above program. The improved program creates two threads in each node: one for the socket server and the other for the socket client and establishes two socket connections in parallel between two nodes. In this program, for each socket, one directional communication is made from server to client, thus full duplex communication becomes possible by using two sockets simultaneously. In contrast, the original program performs bidirectional communication over a single socket connection, thus it can extract only half duplex performance of the network. In the following discussion, if we need to identify these two programs, the original and the improved programs are referred to as HD program and FD program, respectively. Here, we have to mention that the improved program may apply a high message volume handling pressure to achieve full duplex communication speed.

4.1.2 Hardware Environment

The Three Quads interconnection is built by three groups of small-scale Gigabit Ethernet switches. In case of 64 nodes configuration, 16 ports switch is sufficient as each of these switches. However, in order to investigate the performance of Three Quads, we prepare an enterprise switch [8] which can accommodate whole 64 nodes in one switch. Logically, this enterprise switch behaves like a full crossbar, so that matrix transpose can be written in one step based on direct swapping of submatrices between corresponding two nodes pairs.

But physically, almost all enterprise switches are implemented as two-layered switch where several switch modules are interconnected through an upper layer backbone switch or high-bandwith buses [*4]. Due to this internal structure of the switch, a performance bottleneck may appear if high throughput all-to-all communication is required.

In this experiment, for simplicity, we replace the 4 SW-X switches with one enterprise switch with 4 switch modules. When we investigate the performance of Three Quads, we let each of 4 switch modules works as one of 4 SW-X switches of Three Quads and logically prohibit communication across two different switch modules. When we execute the programs designated for Three Quads, we use Y subnetwork during step 1 and X subnetwork during step 2.

4.2 Results and Discussions

Elapsed time of matrix transpose programs is measured on Scube cluster using two network configurations: Three Quads and Enterprise Switch. The results are summarized in **Table 1**.

In this table, we could confirm that Enterprise Switch always outperforms Three Quads. This is because the estimated theoretical speed of the one step matrix transpose program for Enterprise Switch is roughly twice as fast as the two steps program for Three Quads, theoretically. However, we have to emphasize that the observed differences are not 100% but rather they are 46% and 14% for half duplex and full duplex environments, respectively.

In both network configurations, the FD program is faster than the HD program and the observed speedup of FD program versus the HD program is respectively 1.63 times and 1.27 times for Three Quads and Enterprise Switch. So, we could confirm that the advantage of full duplex communication is better utilized in Three Quads than in Enterprise Switch.

In total, cost effective Three Quads shows the reasonable and appealing performance against the expensive Enterprise Switch.

For further analysis of these results, we made a detailed analysis of the exection time on each node in the case of Three Quads configurations. **Figures 3** and **4** show the detailed breakdown of the elapsed time on each node, for HD program and FD program, respectively. In these figures, each bar represents the breakdown of the execution time on each node and the bars are aligned horizontally in increasing order of execution time.

The execution time on each node is composed of several components: initialization, the local submatrix transposition, step 1 communication, waiting time, step 2 communication. Initializa-

Table 1 Elapsed time for matrix transpose [s].

Network Configuration	Three Quads	Enterprise Switch
Half Duplex	19.2	13.2
Full Duplex	11.8	10.4

[*2] Due to maintenance needs, some nodes and switches have slightly different configurations. But the differences are negligible in analyzing performance behaviors.

[*3] In this experiment, matirix size was set relatively large to demonstrate the topological advantage.

[*4] Network latency of these switches is negligible in this experiment and is up to 6 us in Enterprised Switch [8]. Though the latency information of the switches [9] used in Three Quads is not available, its successor [10] has a latency of 3.4 us.

Fig. 3 Execution time of each node on Three Quads with HD program.

Fig. 4 Execution time of each node on Three Quads with FD program.

tion includes local submatrix initialization and socket initialization for inter node communication required in both step 1 and step 2.

In Fig. 3, distribution of the execution time of HD program is shown. According to the workloads of each node, they can be categorized in four groups: 1) Nodes without communication; 2) Nodes with communication only in step1; 3) Nodes with communication only in step2; 4) Nodes with communication in both step 1 and 2. The bars shown in this figure exactly correspond to these groups. The execution time of the nodes in the same group is almost equivalent in this figure and the time for group 3 and group 4 is also the same since we enforce synchronization at the end of step 1. We could also confirm that the time for step 1 and step 2 is quite similar.

In Fig. 4, distribution of the execution time of HD program is shown. Similar results are observed between two figures, except that the communication time reduces to roughly 60% in FD program. However, unlike the figure for the HD program, we were able to find the evidence that there is some difference in communication time even in between the nodes belonging to the same group. This phenomenon is taking place in both steps 1 and 2. And we also find some differences in the communication time for steps 1 and 2. At the slowest nodes, they are 5.7 s and 6.3 s for step 1 and step 2. These differences seem to be incurred because of the heavy traffic needed to achieve all-to-all full duplex communication inside a switch.

On the other hand, in order to analyze the optimal performance of Enterprise Switch, we also made some additional experiments. At first, we reconsidered the initial node assignment of submatrices to maximize the communication which can complete inside a switch module of Enterprise Switch, though this assignment is not natural but rather ad hoc. As a result of this reassignment, the execution time slightly reduced to 11.2 s and 10.1 s for half duplex and full duplex communication, respectively. Even in this situation, Three Quads were 1.71 times and 1.17 times slower than Enterprise Switch in half duplex and full duplex respectively, which are still faster than the estimated theoretical speed.

However we could not obtain the benefit of full duplex communication even in this assignment, due to the saturation phenomenon inside a switch module and the existence of inter-module communication [*5]. As long as the internal structure of Enterprise Switch is not the pure crossbar, these disadvantages cannot be alleviated, thus the relative performance of Three Quads versus the Enterprise Switch becomes even better.

[*5] We have made some other experiments to confirm the saturation phenomenon under full duplex communication.

5. Conclusions

Results from the performance study of congestion free matrix transpose algorithm allowed confirming that Three Quads shows reasonable and appealing performance versus the expensive Enterprise Switch.

Due to the high one-hop coverage of Three Quads, many useful simulation programs can be implemented without any specific routing functionality. So we did not address the issue of a routing mechanism for Three Quads, so far. However, in order to utilize many other interesting features of Three Quads, as a future work, we would like to consider the FPGA-based hardware implementation of the routing functionality to achieve low-latency routing without suffering loss in computation resources in any node.

Acknowledgments This research is supported in part by JSPS Grants-in-Aid for Scientific Research #16100001, #25280042 and #25540043.

References

[1] Yu, H. et al.: In situ visualization for large-scale combustion simulations, *IEEE Computer Graphics and Applications*, Vol.30, No.3, pp.45–57 (2010).
[2] Zhang, J. et al.: A real-time GPU-based coupled fluid-structure simulation with haptic interaction, *Proc. 15th Int. Conf. Computer and Information Science*, *ICIS '16*, pp.709–714, IEEE (2016).
[3] Yoshimura, T. et al.: Three Quads: An Interconnection Network for Interactive Simulations, *Systems Modeling and Simulation*, pp.362–366, Springer Japan (2007).
[4] Yoshimura, T. et al.: Implementation of a large scale visualization cluster with commodity GPU, *Proc. 33rd Symp. Visualization* (*J. Visualization Society of Japan*), Vol.25, Suppl.1, pp.277–280 (2005).
[5] Ogata, M., Kajihara, K., Kikukawa, T. and Terada, T.: An evaluation of the communication cost of parallel processing in real-time simulations using an image-composition device, *Information and Media Technologies*, Vol.3, No.3, pp.519–528 (2008).
[6] Tanabe, N., Nakamura, S., Suzuoka, T. and Oyanagi, S.: Implementation of the prodigy parallel AI machine and performance evaluation of communication, *Syst. Comp. Jpn.*, Vol.23, Issue 8, pp.1–12 (1992).
[7] Shimokawabe, T. et al.: Peta-scale phase-field simulation for dendritic solidification on the TSUBAME 2.0 supercomputer, *Proc. 2011 Int'l Conf. for High Performance Computing, Networking, Storage and Analysis*, Article No. 3 (2011).
[8] HP, HP ProCurve Switch 4200vl Series QuickSpec, available from ⟨https://h20195.www2.hpe.com/v2/getpdf.aspx/c04111740.pdf?ver=7⟩ (accessed 2018-02-13).
[9] Buffalo, LSW-GT-16NSR Hardware Manual, available from ⟨http://buffalo.jp/download/manual/l/lswgt16nsr.html⟩ (accessed 2017-06-11).
[10] Buffalo, Specification of Gigabit Switching Hub, available from ⟨http://buffalo.jp/product/wired-lan/lan-hub/lsw3-gt-nsr⟩ (accessed 2017-06-11).

Jiachao Zhang was born in 1987. He received his B.E. in Computer Science and Technology from Sichuan University at China in 2010 and M.E. in computer application technology from China Electronics Technology Group Corporation's Academy of Electronics and Information Technology at China in 2013. Now he is a Ph.D. student in information science at University of Fukui at Japan. His research interests include software project, parallel processing, simulation and visualization. He is a member of ACM and IPSJ.

Shin-ichiro Mori was born in 1963. He received his B.E. in Electronics from Kumamoto University in 1987 and M.E. and Ph.D. in Computer Science from Kyushu University in 1989 and 1995, respectively. From 1992 to 1995, he was a research associate in the Faculty of Engineering, Kyoto University. From 1995 to 2006, he was an associate professor in the Department of Computer Science, Kyoto University. Since 2006, he has been a professor in the Department of Information Science, University of Fukui. His research interests include computer architecture, parallel processing and visualization. He is a member of IEEE, ACM, EUROGRAPHICS and IPSJ.

Shinji Tomita was born in 1945. He received his B.E., M.E. and Ph.D. degrees in Electronics from Kyoto University in 1968, 1970, and 1973, respectively. He was a research associate from 1973 to 1978 and an associate professor from 1978 to 1986 both in the Department of Computer Science, Kyoto University. From 1986 to 1991, he was a professor in the Department of Information Systems, Kyushu University. From 1991 to 1998, he was a professor in the Faculty of Engineering, Kyoto University. From 1998 to 2009, he was a professor in the Graduate School of Informatics, Kyoto University. In 2009, he received Prof. Emeritus title from Kyoto University in 2009. From 2009 to 2017, he was a professor and administrative director of Institute for Integrated Cell-Material Sciences, Kyoto University. Since 2017, he has been a processor of the Kyoto College of Graduate Studies for Informatics. His current interests include computer architecture and parallel processing. He is a member of IEEE, ACM, IEICE and IPSJ. He was a board member of IPSJ directors in 1995, 1996, 1998 and 1999. He is a fellow of IEICE and IPSJ.