Investigating the Effects of Trees and Butterfly Barriers on the Performance of Optimistic GVT Algorithm

Abdelrahman Elleithy, Syed S. Rizvi, and Khaled M. Elleithy
Computer Science and Engineering Department, University of Bridgeport, Bridgeport, CT USA
{aelleithy, srizvi, elleithy}@bridgeport.edu

Abstract- There is two approaches for handling timing constraints in a heterogeneous network; conservatives and optimistic algorithms. In optimistic algorithms, time constraints are allowed to be violated with the help of a time wrap algorithm. Global Virtue Time (GVT) is a necessary mechanism for implementing time wrap algorithm. Mattern [2] has introduced an algorithm for GVT based computation using a ring structure. which showed high latency. The performance of this optimistic algorithm is optimal since it gives accurate GVT approximation. However, this accurate GVT approximation comes at the expense of high GVT latency. Since this resultant GVT latency is not only high but may vary, the multiple processors involve in communication remain idle during that period of time. Consequently, the overall throughput of a parallel and distributed simulation system degrades significantly In this paper, we discuss the potential use of trees and (or) butterflies structures instead of the ring structure. We present our analysis to show the effect of these new mechanisms on the latency of the system.

I. Introduction

Many GVT algorithms were introduced in the literature. In [1] Chen *at. al.*, provided a comparison between 15 GVT algorithms. Table 1 [1] shows a detailed comparison between the different algorithms.

Mattern's GVT algorithm [2] proposed a 2-cut algorithm to avoid synchronizing all processors at the same wall clock. The two cuts define a past and a future point. In a consistent cut, no transient jobs can travel from the future to the past. Messages crossing the second cut from the future to the past do not need to be taken into account because these messages are guaranteed to have a timestamp larger than the GVT value.

Mattern's GVT algorithm uses a token passing to construct the two cuts. It uses two cuts C1 and C2. C1 is intended to inform each processor to begin recording the smallest time stamp where as C2 guarantees that no message generated prior to the first cut is in transient. A vector clock passed between processors monitors the number of transient messages sent to every processor. The token can leave the current processor only after all

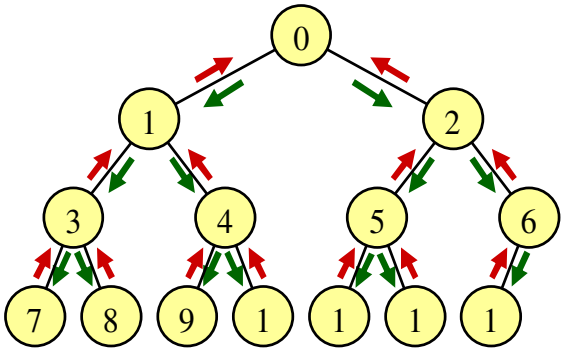

Fig.1. Tree barrier mechanism for synchronization among the logical processes, Green font arrow lines represent the LBTS computation and the new GVT announcement

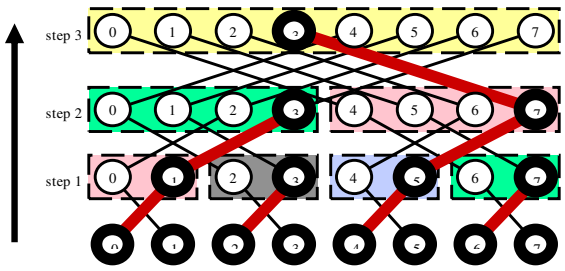

Fig. 2. Butterfly barrier mechanism between 8 LPS. Three steps are needed to complete the synchronization. The red font represents the synchronization for LP3

Table 1: Comparison between Different GVT Algorithms [1]

Authors	Idea	Ack	Vector	Channel	Scalability
Samadi [19]	Broadcast START and STOP messages to form overlapping intervals	Yes	No	Any	N/A
Bellenot [12]	Use message routing graph instead of broadcast	Yes	No	Any	104 [26]
Das and Sarkar [14]	Optimize the computation for hypercube topology	No	No	Maximum Delay	N/A
Baldwin, Chung and Chung [10]	Pass a token to form overlapping intervals	Yes	No	FCFS	N/A
Lin and Lazowska [16]	Send valley messages to reduce acknowledgement traffic	Implicit	No	Any	N/A
Mattern [17]	Construct two cuts such that no transient messages sent before the first cut exist	No	Yes	Any	12 [13]
Choe and Tropper [13]	Create multiple rounds of token passing to form the two cuts	No	No	Any	12 [13]
Tomlinson and Garg [22]	Build consistent cuts by using TGVT events	No	Yes	Any	N/A
Bauer and Sporrer [11]	Identify pairs of reports that form a consistent cut	No	No	FCFS	N/A
Srinivasan and Reynolds [20]	Use hardware-based global reduction	No	No	Any	N/A
Steinman, Lee, Wilson, and Nicol [21]	Use global reduction	No	No	Any	64 [21]
Perumalla and Fujimoto [18]	Use global reduction	No	No	Any	16 [18]
D'Souza, Fan, and Wilsey [15]	Report stragglers to a GVT manager	Yes	No	Any	2 [15]
Bauer, Yuan, Carothers, Yuksel, and Kalyanaraman [23]	Extend Fujimoto's shared-memory GVT algorithm with the notion of network atomic operations	No	No	Maximum Delay	16 [23]
Deelman and Szymanski [24]	Use vector and matrix clocks to keep track of messages in transit	No	Yes	Any	16 [24]

messages destined to it have been received. The second cut can be built with only one round of token passing. The creation of the second cut may incur a delay on each processor.

II. Related Work

In [3], a tree structure is used to implement a barrier mechanism blocking and releasing for Logical Process (LP) as shown in Figure 1. The tree barrier mechanism requires 2 $\log_2$ N steps and 2 (N-1) messages for *N* processors.

A butterfly mechanism is discussed in [3] to eliminate the need for broadcasting as shown in Figure 2. The butterfly mechanism requires log N steps to complete and N * Log N messages for N processors.

III. Analytical Model

In comparing centralized barriers, it is noticed that the butterfly mechanism has a better performance when comparing the required time as it needs half the number of steps (Figure 3). Butterfly barrier has the butterfly mechanism in terms of the required exchanged messages (Figure 4). It should be clearly noted in Fig. 3 that the performance of the tree barrier is much better than

the butterfly barrier for all values of N. This is due to the fact that the time complexity of the tree barrier is much lower than the butterfly barrier.

IV. Using Tree and Butterflies

By analyzing the ring structure used in Mattern's we notice that the ring works as follows:

1. C1 is constructed by sending a control message around the ring. Once the control message is received, the color of the processor changes from white to red then passes the message. This step of the algorithm will take (N-1) steps.
2. C2 is constructed by sending the control message around the ring. This step of the algorithm will take (N-1) steps.

We assume that initially all processors (nodes) and their neighbors that are organized in a minimal tree (i.e.., no cycles) based structure are colored white. In addition, we also assume that there should be one initiator of GVT computation that may also be considered as a root of the tree (i.e.., the node where message transmission starts). The moment initiator processor initiates GVT computation, it becomes red from white. At the same time, it starts a broadcast scheme to indirectly (i.e.., from node to edges) send control messages to all connected processors. Thus, this first transmission (the process of making red) of broadcast from root (i.e.., the initiator processor) to all its connected nodes is intended for the first cut C1.

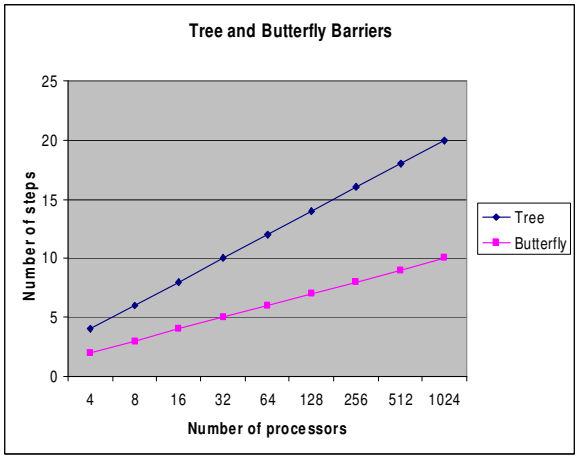

Fig. 3: Time Comparison between the Tree and the Butterfly Barriers with a random number of message transmissions with a large number of processors

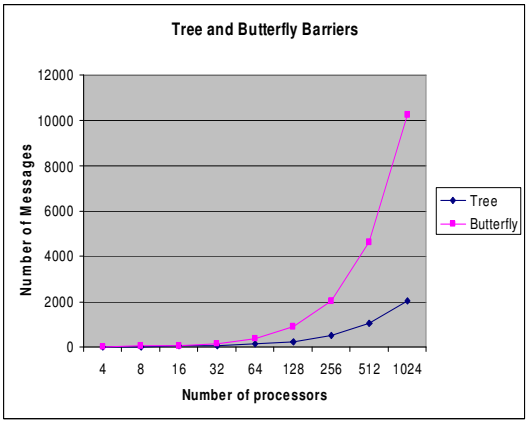

Figure 4: Communication Messages Comparison between the Tree and the Butterfly barriers for a large number of processors

According to our initial assumptions, Mattern's algorithm does not require acknowledgement messages but it does require the construction of the second cut C_2. We assume that, in order to construct the second cut C_2, we need the same number of messages that will propagate from processors (i.e.., the edges of the tree) to the initiator (i.e.., the root of the tree). Therefore, this implies that any processor in the given design which is the part of a balanced minimal tree must process two messages; one for constructing the first cut C_1 and the other for constructing the second cut C_2. The total number of steps in implementing the ring is 2 * (N-1).

Instead of using a ring structure, we can use a tree structure. The number of steps using the tree structure to implement Mattern algorithm is 2 * $\log_2$ (N) as per our discussion in section III. One can clearly observe in Fig. 4 that the number of messages transmitted with the tree barrier is much lower than the number of messages required for the butterfly barrier. This is especially true for a large number of processors. In other words, as we start increasing the number of processors in the system, the performance differences between the tree and the butterfly barrier is obvious. For instance, the number of messages transmitted for the tree barrier do not exceed to 2000 messages for even a large value of processors (typically 1000 processors) as shown in Fig. 4. Furthermore, if we use butterflies, the numbers of steps is $\log_2$ N. Figure 5 shows a comparison between using a ring and a butterfly in implementing the Mattern GVT algorithm. Fig. 6 represents the implementation of the butterfly barrier where four processors are organized and sending/receiving messages to each other. When compare the

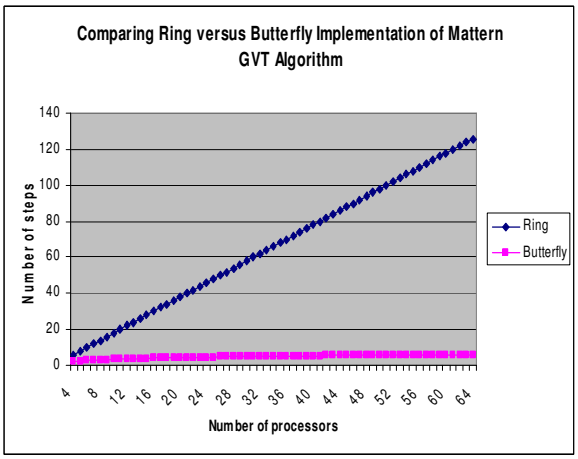

Figure 5: Comparison of using a ring and a butterfly in implementing Mattern GVT algorithm

performance of butterfly barrier with the tree barrier, one can clearly observe that the performance of butterfly is overlapping the tree barrier for a small number of processors (typically for 150 processors) as shown in Fig. 4. However, as we start increasing the number of processors (LPs > 150) in the system, the performance of butterfly degrades significantly than the tree barrier. This is due to the fact that the time complexity of the butterfly barrier is slightly higher than the tree barrier. On the other hand, when the performance of butterfly barrier is compared with the ring structure, the simulation results of Fig 5 suggest that the butterfly is clearly a better choice for using as a synchronization mechanism with the Mattern's GVT algorithm.

Although the actual number of steps has decreased significantly by using a butterfly compared to a ring in implementing Mattern's GVT algorithm, a large number of messages have been created. In the original ring implantation, there are only two control messages that are circulating in the ring. For a butterfly implementation, the number of messages is N * Log N.

To make it more clear, this barrier requires steps with the transmission of messages, since each processor must send and receive one message in each step of the algorithm. Thus, the asymptotic complexity of this barrier is clearly higher than the tree or ring structures which in turn give a higher value of latency.

It is worth mentioning that the asymptotic latency of butterfly is exactly the same as the merge algorithm where the total of N number of comparisons are analogous to the total number of N messages transmitted in one direction. From message complexity point of view, it is obvious that the latency of butterfly barrier for Mattern's GVT algorithm exists in a logarithmic region with a constant N.

In addition, our analysis demonstrates that one can achieve the same latency for Mattern's algorithm if we assume that two rounds of messages propagate from initiator to all processors (i.e.., intended for C1) and from all processors to the root (i.e.., intended for C2) in a tree barrier. However, the latency can be improved if parallel traversal of connected processors is allowed. The above discussion can be extended for a tree structure where the left and the right sub trees have different length.

V. Conclusion

In this paper, we have investigated on the possibility of using Trees and Butterfly barriers with the Mattern GVT algorithm. The simulation results have verified that the use of butterfly barriers is inappropriate with an asynchronous type of algorithm when the target is to improve the latency of the system. Since the latency is directly related to how many number of messages each

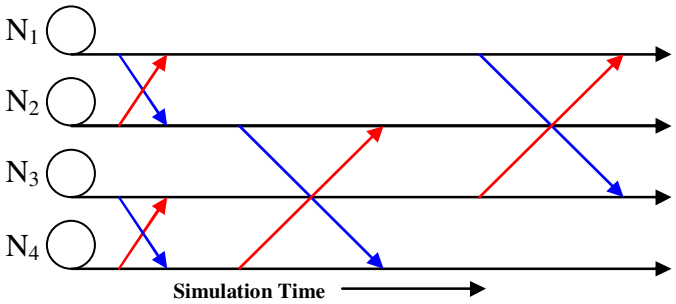

Fig. 6: Butterfly Barrier Organization: Arrows in the figure show that the node is arriving/reaching barrier to other processors. Once the LBTS computation is initiated by all processors, the execution of the messages will be halt unless all the processors achieve synchronization by knowing a Global minimum value

processor is sending, butterfly barrier may not be a good candidate to improve the latency of the GVT computation. However, we have shown that the latency of the GVT computation can be improved if the tree based structure is organized in a way that allows parallel traversing of each left and the right sub trees. The improvement in the latency has a higher cost of communications.

REFERENCES

1. Gilbert G. Chen and Boleslaw K. Szymanski, Time Quantum GVT: A Scalable Computation of the Global Virtual Time in Parallel Discrete Event Simulations, Scientific International Journal for Parallel and Distributed Computing, pages 423–435, Volume 8, no. 4, December 2007.
2. F. Mattern, Efficient algorithms for distributed snapshots and global virtual time approximation, J. Parallel and Distributed, Computing 18(4) (1993) pp. 423–434.
3. Fujimoto, R., Parallel and Distributed Simulation Systems, Willey Series on Parallel Distributed Computing, 2000.

Authors Biographies

Abdelrahman Elleithy has received his BS in Computer Science in 2007 from the Department of Computer Science and Engineering at the University of Bridgeport, Connecticut, USA . Abdelrahman is currently a MS student and expected to receive his MS in Computer Science in December 2008. Abdelrahman has research interests in wireless communications and parallel processing where he published his research results papers in national and international conferences.

SYED S. RIZVI is a Ph.D. student of Computer Engineering at University of Bridgeport. He received a B.S. in Computer Engineering from Sir Syed University of Engineering and Technology and an M.S. in Computer Engineering from Old Dominion University in 2001 and 2005 respectively. In the past, he has done research on bioinformatics projects where he investigated the use of Linux based cluster search engines for finding the desired proteins in input and outputs sequences from multiple databases. For last one year, his research focused primarily on the modeling and simulation of wide range parallel/distributed systems and the web based training applications. Syed Rizvi is the author of 45 scholarly publications in various areas. His current research focuses on the design, implementation and comparisons of algorithms in the areas of multiuser communications, multipath signals detection, multi-access interference estimation, computational complexity and combinatorial optimization of multiuser receivers, peer-to-peer networking, and reconfigurable coprocessor and FPGA based architectures.

DR. KHALED ELLEITHY received the B.Sc. degree in computer science and automatic control from Alexandria University in 1983, the MS Degree in computer networks from the same university in 1986, and the MS and Ph.D. degrees in computer science from The Center for Advanced Computer Studies at the University of Louisiana at Lafayette in 1988 and 1990, respectively. From 1983 to 1986, he was with the Computer Science Department, Alexandria University, Egypt, as a lecturer. From September 1990 to May 1995 he worked as an assistant professor at the Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia. From May 1995 to December 2000, he has worked as an Associate Professor in the same department. In January 2000, Dr. Elleithy has joined the Department of Computer Science and Engineering in University of Bridgeport as an associate professor. Dr. Elleithy published more than seventy research papers in international journals and conferences. He has research interests are in the areas of computer networks, network security, mobile communications, and formal approaches for design and verification.