ORCHESTRA: An Asyncrhonous Non-Blocking Distributed GVT Algorithm

Tommaso Tocci
Barcelona Supercomputing Center (BSC), Sapienza University of Rome
tommaso.tocci@bsc.es

Abstract—Taking advantage of high computing capabilities of modern distributed architectures is fundamental to run large-scale simulation models based on the Parallel Discrete Event Simulation (PDES) paradigm. In particular, by exploiting clusters of modern multi-core architectures it is possible to efficiently overcome both the power and the memory wall. This is more the case when relying on the speculative Time Warp simulation protocol. Nevertheless, to ensure the correctness of the simulation, a form of coordination such as the GVT is fundamental. To increase the scalability of this mandatory synchronization, we present in this paper a coordination algorithm for clusters of share-everything multi-core simulation platoforms which is both wait-free and asynchronous. The nature of this protocol allows any computing node to carry on simulation activities while the global agreement is reached.

1 Introduction

A classical technique to achieve high-performance simulation runs is to rely on Parallel-DES (PDES) [1]. PDES grounds on partitioning the simulation model into several distinct objects, known as Logical Processes (LPs), which concurrently execute simulation events, possibly on a distributed environment.

To deliver high-performance simulation runs, a core aspect of PDES systems is synchronization, which ensures causally-consistent (i.e. timestamp-ordered) execution of simulation events at each LP. Several synchronization protocols have been proposed, among which the optimism-oriented ones, such as the Time Warp protocol [2], are a viable solution to tackle simulation performance aspects. In Time Warp, events are processed speculatively, thus significantly exploiting parallelism, while causal consistency is guaranteed through rollback/recovery techniques, which restore the simulation model to a correct state upon the a-posteriori detection of consistency violations. These are originated when LP_a schedules a new event destined to LP_b having a timestamp lower than the one of some event already speculatively processed by LP_b. In case this occurs, the rollback of LP_b might also require undoing the send operation of events that were produced by LP_b during the rolled back portion of the computation. This is done via anti-messages (carrying anti-events), which annihilate the originally-sent events, thus possibly causing cascading rollbacks across chains of LPs.

This high level of independence among different LPs is the key to high-performance simulation. In fact, this execution model tries to capture time independence, without any manual intervention from the simulation model developer. Nevertheless, Time Warp has the need for a global notion of time. In fact, a core abstraction is the Global Virtual Time (GVT), which is defined as the smallest timestamp among events (or anti-events) that are still unprocessed, or that are currently being processed. The GVT allows to identify the commitment horizon of the speculative simulation run—no LP can ever rollback to simulation time preceding the GVT value [2]. Its value is used both to execute actions that cannot be subject to rollback, such as displaying/inspecting intermediate simulation results [3], [4], and to reclaim memory [5] (the *fossil collection* operation).

To determine the GVT value, some sort of coordination among the computing nodes is required. This coordination can significantly affect the simulation performance, in a way that is directly affected by the organization of the computing environment. This is an aspect that must be explicitly taken into account, since computing architectures have recently hit some physical limits: the *power wall* [6] and the *memory wall* [7] have posed strict limitations on what can be done with out-the-shelf computers.

2 Related Works

Several GVT algorithm have been proposed in the last 30 years, and usually they are designed for a specific underlying architecture: either focusing on shared-memory (e.g. [8], [9]) or distributed computers. In 2008 Chen and Szymanski [10] presented an in-dept comparison and tried to classify them. As outlined in their study, some of these algorithms rely on assumptions or make use of some properties that limit their scalability. For instance, the usage of message acknowledgments (e.g. [11], [12], [13]) in order to cope with non observability of transient messages, is an obstacle to scalability, because it introduces big overhead in the communication. Mattern proposed a *two-cuts* [14] approach in which by using colors, it is possible to keep track of the transient messages without requiring explicit acknowledgments. Only recently, with the goal of reaching very large-scale simulations, some hybrid algorithms appeared. Lin and Yao [15] proposed the first GVT algorithm targeting a multi-thread PDES platform, that exploits the shared-memory capability of each machine while reaching a global virtual time agreement among them.

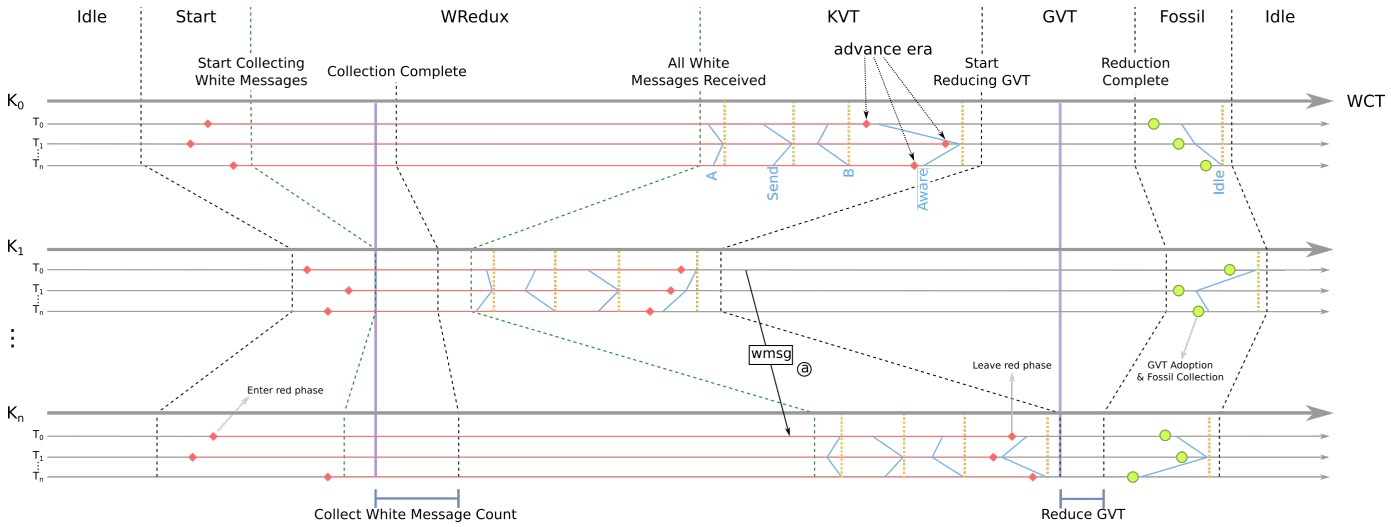

Fig. 1. Representation of phases of the distributed GVT algorithm

3 Reference System Model

Communication latency and cluster nodes topology can really impact on the simulation performance. To enforce highly-scalable simulations and give them the flexibility to adapt to the underlying network interconnection, it is important to design a truly distributed software architecture and to avoid any use of centralized algorithms.

Each computing node (i.e., a core in the distributed system) is associated with a thread. The LPs are distributed among all the simulation threads in such a way that each of the thread will carry on simulation activities of a subgroup of LPs bound to it. The threads running on the same machine are grouped together into the same simulation kernel and they share a portion of the main memory. Communication among threads running on different kernels need to pass through the interconnection network, while threads living within the same kernel will take advantage of shared-memory for local communication activities.

Unlike the organization used in [15], all the threads in our architecture have exactly the same role, meaning that no special thread is used to carry on communication tasks. This homogeneity between all the worker threads simplify the scheduling process and allows to better distribute the simulation workload among the nodes of the cluster.

4 Asynchronous Non-Blocking GVT

The algorithm to compute the GVT that we present here works at two different levels. It uses the wait-free algorithm proposed in [9] to reach consensus between threads of a single kernel while implementing a variant of the two cuts Mattern's idea [14] to make all the kernels agree on a common global virtual time. The 2-color scheme [14] is used to solve the well known *transient message problem*. A thread becames red colored while it is participating on a GVT round, while normally it is white. Every outgoing message takes the color of the sender thread. This coloring scheme allows to include all the messages that are in transit in the current GVT agreement round. In fact, while red messages are always accounted by the sender, the white ones will be accounted by the receiver. As shown in Figure 1 each thread participating in a GVT round, pass through the phases of the algorithm that are actually determined by a combination of the state of the thread itself and the shared state of the kernel to which it belongs. At the beginning of each GVT round, during the *Start* phase, every thread switch its color to red. After that, every kernel collects the number of white messages that have been sent to it. Once the kernel has verified that no white message destined to it is still in transit, it triggers the start of the local *Kernel Virtual Time* (KVT) algorithm. This local algorithm, executed at each kernel, is the one presented in [9] where, by exploiting shared memory, all the threads reach an agreement upon the KVT. In addition to the orginal version, during this phase, every thread needs to leave from the red state and enter again into the white one. This allow to collect the minimum timestamps among all the red messages sent and to account for it in the current KVT calculation. Once all the KVTs are locally computed, all the kernels participate to a global reduction in order to elect the minimum among these values as the new GVT.

Acknowledgments

This study has been carried on in close cooperation with Alessandro Pellegrini, Postdoctoral Researcher at the Department of Computer Science and System Engineering at "Sapienza" University of Rome. All the work has been supervised by Josep Casanovas from the Barcelona Supercomputing Center and supported by Toyotaro Suzumura from T.J. Watson Research Center and visiting professor at BSC. This research has been funded by JST, CREST Strategic Basic Research Programs in collaboration with Barcelona Supercomputing Center (BSC).

References

[1] R. M. Fujimoto, "Performance of Time Warp under synthetic workloads," 1990.

[2] D. R. Jefferson, "Virtual Time," *ACM Transactions on Programming Languages and System*, vol. 7, no. 3, pp. 404–425, 1985.

[3] F. Antonacci, A. Pellegrini, and F. Quaglia, "Consistent and efficient output-stream management in optimistic simulation platforms," in *Proceedings of the 2013 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, ser. PADS. ACM, 2013, pp. 315–326. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2486092.2486133

[4] D. Cucuzzo, S. D'Alessio, F. Quaglia, and P. Romano, "A Lightweight Heuristic-based Mechanism for Collecting Committed Consistent Global States in Optimistic Simulation," in *DS-RT*, 2007, pp. 227–234.

[5] S. R. Das and R. M. Fujimoto, "Adaptive Memory Management and Optimism Control in Time Warp," *ACM Transactions on Modeling and Computer Simulation*, vol. 7, no. 2, pp. 239–271, 1997.

[6] H. Sutter, "The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software," *Dr. Dobb's Journal*, vol. 30, no. 3, pp. 202–210, 2005. [Online]. Available: http://www.gotw.ca/publications/concurrency-ddj.htm

[7] S. a. McKee, "Reflections on the memory wall," *Proceedings of the first conference on computing frontiers on Computing frontiers - CF'04*, p. 162, 2004. [Online]. Available: http://portal.acm.org/citation.cfm?doid=977091.977115

[8] R. M. Fujimoto and M. Hybinette, "Computing Global Virtual Time in Shared-Memory Multiprocessors," *ACM Transactions on Modeling and Computer Simulation*, vol. 7, no. 4, pp. 425–446, 1997.

[9] A. Pellegrini and F. Quaglia, "Wait-Free Global Virtual Time Computation in Shared Memory TimeWarp Systems," in *2014 IEEE 26th International Symposium on Computer Architecture and High Performance Computing*. IEEE, oct 2014, pp. 9–16. [Online]. Available: http://ieeexplore.ieee.org/document/6970641/

[10] G. G. Chen and B. K. Szymanski, "Time Quantum GVT: A Scalable Computation of the Global Virtual Time in Parallel Discrete Event Simulations," vol. 8, no. 4, pp. 423–435, 2008.

[11] B. Samadi, "Distributed Simulation Algorithms and Performance Analysis," Ph.D. dissertation, Computer Science Department, University of California, Los Angeles, 1985.

[12] S. Bellenot, "Global Virtual Time algorithms," in *Proceedings of the SCS Multiconference on Distributed Simulation*, 1990, pp. 122–127.

[13] R. Baldwin, M. J. Chung, and Y. Chung, "Overlapping window algorithm for computing GVT in Time Warp.pdf," pp. 534–541, 1991.

[14] F. Mattern, "Efficient Algorithms for Distributed Snapshots and Global Virtual Time Approximation." *Journal of Parallel Distributed Computing*, vol. 18, no. 4, pp. 423–434, 1993.

[15] Z. Lin and Y. Yiping, "An asynchronous GVT computing algorithm in neuron time warp-multi thread," no. Gillespie 1977, pp. 1115–1126, 2015.

Tommaso Tocci chasing his passion for the informatics, he graduated in computer science at the Sapienza University of Rome. He is concluding the master of Science in Engineering in Computer Science at the same university, focusing on distributed systems. At the moment he is employed at Barcelona Supercomputing Center (BSC), working on a parallel discrete event simulation platform and developing distributed synchronization algorithms.