USING MASSIVELY PARALLEL SIMULATION FOR MPI COLLECTIVE COMMUNICATION MODELING IN EXTREME-SCALE NETWORKS

Misbah Mubarak

Computer Science Department
Rensselaer Polytechnic Institute
110 8th Street
Troy, NY 12180, USA

Christopher D. Carothers

Computer Science Department
Rensselaer Polytechnic Institute
110 8th Street
Troy, NY 12180, USA

Robert B. Ross

MCS Division
Argonne National Laboratory
9700 S. Cass Avenue
Argonne, IL 60439, USA

Philip Carns

MCS Division
Argonne National Laboratory
9700 S. Cass Avenue
Argonne, IL 60439, USA

ABSTRACT

MPI collective operations are a critical and frequently used part of most MPI-based large-scale scientific applications. In previous work, we have enabled the Rensselaer Optimistic Simulation System (ROSS) to predict the performance of MPI point-to-point messaging on high-fidelity million-node network simulations of torus and dragonfly interconnects. The main contribution of this work is an extension of these torus and dragonfly network models to support MPI collective communication operations using the optimistic event scheduling capability of ROSS. We demonstrate that both small- and large-scale ROSS collective communication models can execute efficiency on massively parallel architectures. We validate the results of our collective communication model against the measurements from IBM Blue Gene/Q and Cray XC30 platforms using a data-driven approach on our network simulations. We also perform experiments to explore the impact of tree degree on the performance of collective communication operations in large-scale network models.

1 INTRODUCTION

The increasing data and networking demands of large-scale scientific applications place a challenge on the network designers of extreme-scale systems. Exascale system architectures are predicted to have O(100K) compute nodes and a node interconnect bandwidth of up to 1 TiB/s (Dongarra 2012, Department of Energy 2013). Codesign of such extreme-scale networks is critical in order to identify the best network design options for extreme-scale systems before building physical hardware. While other techniques such as analytical modeling can also be used to guide initial design decisions, high-fidelity simulation offers an unmatched ability to evaluate real-world application workloads in detail on proposed architectures.

Considerable research is in progress to determine a network topology that minimizes packet latency and utilizes maximum network bandwidth for a variety of communication patterns. Among the network topologies for extreme-scale systems, torus networks are popular because of their efficient local communication patterns. They have been widely used in the Blue Gene series of supercomputers (Chen et al. 2011), Cray XT5

and Cray XE6 systems (Vaughan et al. 2011). Another emerging class of network topologies is the low-latency, low-diameter networks, such as dragonfly, which has been used in recent Cray XC30 system (Alverson et al. 2012). In our previous work, we have presented simulations of MPI point-to-point messaging on top of million-node dragonfly and torus networks using Rensselaer Optimistic Simulation System (ROSS) (Carothers et al. 2002). This paper presents an extension to our large-scale discrete-event models of the torus and dragonfly network topologies to accurately predict the performance of MPI collective communication operations. The contributions of this paper are as follows:

- We extend the capability of our existing large-scale, discrete-event torus and dragonfly network simulations to model the MPI collective communication ability.
- Collective communication modeling in large-scale torus and dragonfly simulations is demonstrated to execute efficiently on both small- and large-scale problem sizes, as shown by the experiments executed in Section 4. Experiments are executed on network sizes of up to 512K nodes to determine the effect of tree radix in collective communication on the overall network performance.
- We validate the accuracy of our collective communication model by comparing the collective latency of the model with measurements from IBM Blue Gene/Q and Cray XC30 platforms.
- We evaluate the performance of our simulation methodology with a strong scaling study of a 65K node network model. We also compare the performance of ROSS conservative and optimistic synchronization protocols for the collective communication model.

The remainder of the paper is organized as follows. We describe the background work on the network models and collective operations in Section 2. Section 3 presents an algorithmic description for simulating the collective communication algorithms in ROSS. Section 4 validates the model and illustrates an example using the model to study the impact of tree radix on collective performance. Section 5 analyzes the performance of the simulator itself. Related work is discussed in Section 6. Closing remarks and future work are discussed in Section 7.

2 BACKGROUND

In this section we briefly provide background on the simulation of torus networks, dragonfly networks, and collective operations.

2.1 Torus network simulation

A torus is a k-ary n-cube network with $N = k^n$ nodes arranged in an N-dimensional grid with k nodes in each dimension (Dally and Towles 2004). A torus network has no dedicated routers. Each node is directly attached to $2 * n$ other nodes, and all nodes are capable of routing messages in multiple dimensions. Torus networks are popular in HPC deployments because they yield high throughput for nearest-neighbor communication as well as good path diversity under concurrent workloads.

In previous work we developed parallel discrete event models of large-scale torus networks and validated them on the IBM Blue Gene computing platforms (Liu et al. 2012, Mubarak et al. 2014). We then demonstrated the ability to simulate networks with over 1 million nodes with high fidelity. These simulations have been used to explore how different application traffic patterns (both nearest neighbor and bisection bandwidth) respond to different network topology parameters. We also observed that large-scale discrete event simulation can play a valuable role in network design: network behaviors that emerge at large scale are not necessarily captured by small-scale models. These simulations focused exclusively on point-to-point network communication patterns, however.
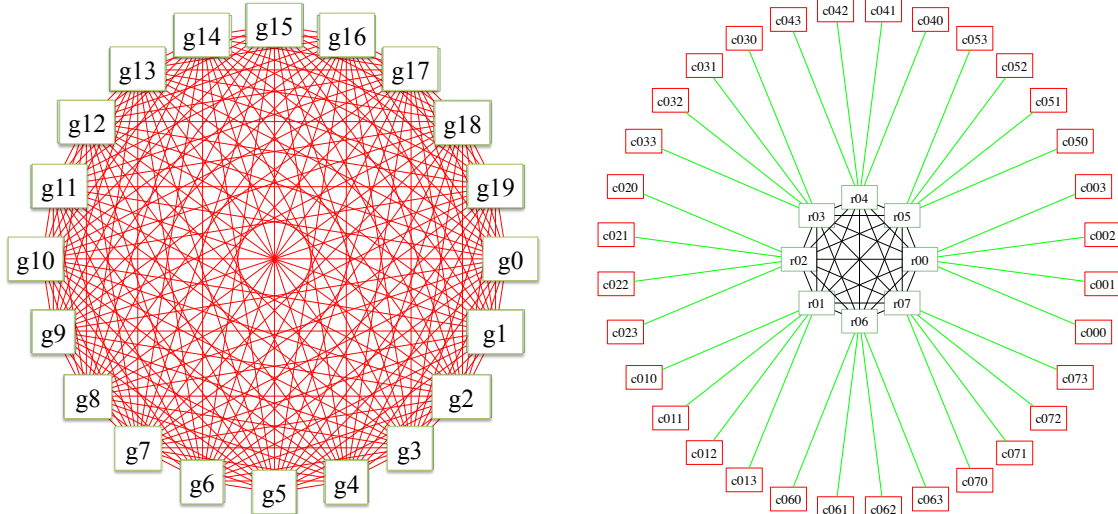
Figure 1: Dragonfly intergroup and intragroup settings with a=8, p=4, h=4 (some groups are not shown because of graphic rendering purposes).

2.2 Dragonfly network simulation

The dragonfly topology is an alternative to the torus topology in which nodes are organized into a hierarchy with multiple groups and dedicated routers. Dragonfly networks have recently gained popularity in systems such as the Cray XC (Alverson et al. 2012). An example of a dragonfly network is shown in Figure 1. The dragonfly topology offers advantages over the torus for some HPC algorithms by reducing the number of hops needed for nonlocal communication. Each router has p nodes connected to it, and each group has a routers. Routers within a group are connected via local channels. Each router also also uses h global channels for connectivity across groups. The recommended dragonfly configuration sets the topology parameters to $a = 2p = 2h$ in order to balance local and global link bandwidth. The total number of nodes N in the network is determined by $N = p * a * g$ (Kim et al. 2009).

In previous work we developed a dragonfly network model that was validated against the `booksim` cycle-accurate simulator and then demonstrated to scale up to 50 million simulated nodes while processing 1.33 billion events per second. The simulation was used to evaluate the impact of routing strategies in large-scale dragonfly networks (Mubarak et al. 2012). As in the torus case, this simulator focused exclusively on point-to-point messaging.

Both the torus and dragonfly models have now been unified under a common framework, known as *modelnet*, that allows the two network models (and others like them) to be used interchangeably in broader HPC system models with a consistent API. The modelnet framework also unifies common functionality across network models, such as mapping model entities to MPI processes for parallel simulation and decomposing messages into packets for transmission.

2.3 Collective communication workloads

The performance of large-scale scientific applications often heavily depends on the speed of MPI collective operations (Almási et al. 2005). Vetter et al. (2013) surveyed 13 prominent HPC applications and found that while communication strategies varied significantly in each case, collective communication calls may constitute up to 60% of the communication operations. `MPI_allreduce()` was the most common operation observed. Although synthetic point-to-point network patterns reveal a variety of insights into broader system design, they are not sufficient for performance prediction or application codesign. Collective

communication models are essential for studying the behavior of specific scientific applications on future network topologies.

Many MPI collective operations are implemented by forming a tree of the MPI processes involved. For example, MPICH2 uses a recursive doubling algorithm in a tree for Allgather operations, a binary tree algorithm for short messages in broadcast operations, and a recursive-halfing algorithm in a tree for reduce-scatter operations (Thakur and Rabenseifner 2005). The Blue Gene implementation of collective operations incorporates a machine-optimized versions of MPI collectives that are an extension of MPICH and Van de Geign algorithms (Almási et al. 2005, Thakur and Rabenseifner 2005). In most of these operations, there is a fan-in phase where the data is collected by the root of a tree from other nodes, followed by an optional computation phase where the root node performs some computation on the data. There also is a fan-out phase where the data computed at the root is distributed to all other nodes.

Publicly available HPC network traces, such as the DOE mini-app traces provided by the Design Forward project (Department of Energy 2014), already provide access to detailed instrumentation of large scale applications including both point-to-point and collective communication operations. These traces can be used in conjunction with high-fidelity HPC network simulation to study the impact of network parameters on relevant large-scale applications.

3 SIMULATING COLLECTIVE COMMUNICATION

In this section, we describe how we extend the capability of our point-to-point dragonfly and torus network models to support a collective communication model.

3.1 Modeling collective communication using ROSS

Discrete-event simulation frameworks, such as ROSS, are composed of logical processes (LPs) where each LP models the state of a distinct component in the system. LPs in ROSS interact with each other by exchanging time-stamped messages. Synchronization of these time-stamped messages is a key issue in parallel discrete-event simulation. The most popular approaches to parallel synchronization can be classified as conservative or optimistic protocols. In the conservative approach, events are processed in the correct time-stamped order by waiting until it is safe to process events; there is no risk of affecting the logical past state of an LP. Synchronization points are re-established frequently by introducing communication among processors, as argued by Nicol et al. (1989). The second approach, optimistic synchronization, allows each LP to independently progress as far in the simulated time as possible, therefore speeding the simulation. Events must be rolled back and re-executed in the correct order if an LP receives an out-of-order event from another LP.

Our collective communication modeling algorithm in ROSS models the fan-in, fan-out, and computation phases of the collective communication operations where all nodes in a network form a tree of radix *n*. ROSS dragonfly and torus network models comprise two LP types: MPI process LPs and network compute node LPs. The dragonfly model also includes LPs that represent dedicated routers within the network fabric. We assume a mapping of one MPI process per compute node since we are interested in the interconnection network among the network nodes. In order to model collective communication for dragonfly and torus networks, a tree of all MPI process LPs participating in the collective operation is formed. Once the collective operation starts, the MPI process LPs, positioned as leaves of the collective tree, send an event at their parents' LPs, simulating the fan-in phase of collective operations. At each level within the tree, a delay is added into each event that corresponds to measured time that would have occurred in the real network hardware being modeled. After the parent LP receives an event from all its children LPs, it sends a message to its parent LP. The process continues up the tree until an event is sent to the root MPI process LP. Figure 2-(a) shows this process among the MPI process LPs. Followed by this fan-in phase, an event corresponding to the computation delay is sent to model the computation phase of the collective operation

(a) Collective modeling Fan-in phase   (b) Computation phase at root node   (c) Collective modeling Fan-out phase
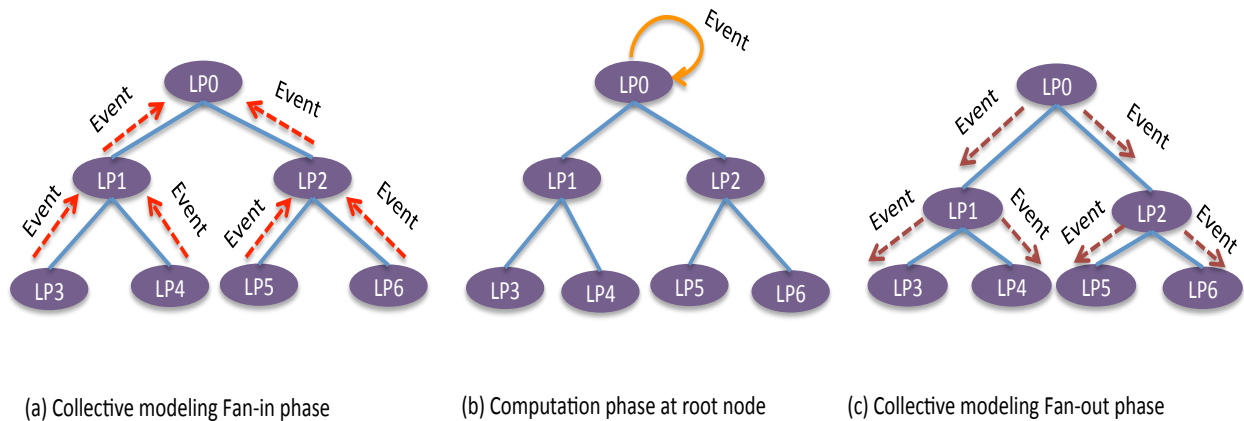
Figure 2: Example of ROSS collective communication model.

at the root node (see Figure 2-(b)). In order to model the collective fan-out phase, the root MPI process LP sends an event to its child LPs, and the chain of events travels down to the leaf MPI process LPs.

3.2 Workload description

To carry out the performance study of the collective modeling algorithm in ROSS, we modeled a workload that is close to the behavior of a large-scale, unstructured finite-element flow solver, PHASTA (Sahni et al. 2009). The communication phase of PHASTA mostly consists of two stages. The first stage uses point-to-point operations (either MPI sends or receives) to communicate with neighboring MPI processes. In the second stage of parallel implicit solvers, a series of `MPI_allreduce()` operations is used to perform a global summation. The parallel implicit solver involves several iterations of these two phases.

In our simulated workload for collective modeling, we have modeled several iterations of these two phases of communication where the first phase involves a certain number of point-to-point MPI calls among neighboring MPI processes and the second phase involves a series of collective communication calls. We model MPI point-to-point communication operations (MPI sends) on MPI process LPs that translate into network calls on the compute nodes of dragonfly/torus networks. Currently, our network models assume that `MPI_Irecv` operations are posted prior to the arrival of an incoming message. Thus, we are not modeling the case where the receipt of an `MPI_Isend` operation is delayed because there was not matching `MPI_Irecv` operation. This delay scenario is rare in practice with extreme-scale-ready HPC codes because the developers take special care to make sure all `MPI_Irecv` operations are posted prior to issuing any `MPI_Isend` operations. To model collective communication calls in the workload, we use the ROSS implementation of the algorithm described in Section 3.1.

4   EVALUATION

In this section, we first describe how we validate the accuracy of our collective model against the Blue Gene/Q and Cray XC30 collective performance. We then present performance data that demonstrates the impact of tree degree in collective communication on overall simulated network latency using a workload that extensively makes collective communication calls.

The performance experiments in this section are executed on two architectures: the Computational Center for Innovation's RSA cluster and the Argonne Leadership facility's Tukey cluster. The RSA cluster is comprised of 32 compute nodes with each node having eight 3.3 GHz Intel Xeon processors. Each node has 256 GB of available RAM, with the entire system having 8 TB of RAM. The Tukey system has a total of 96 compute nodes with each node having two 2 GHz AMD Opteron processors. Each node has 64
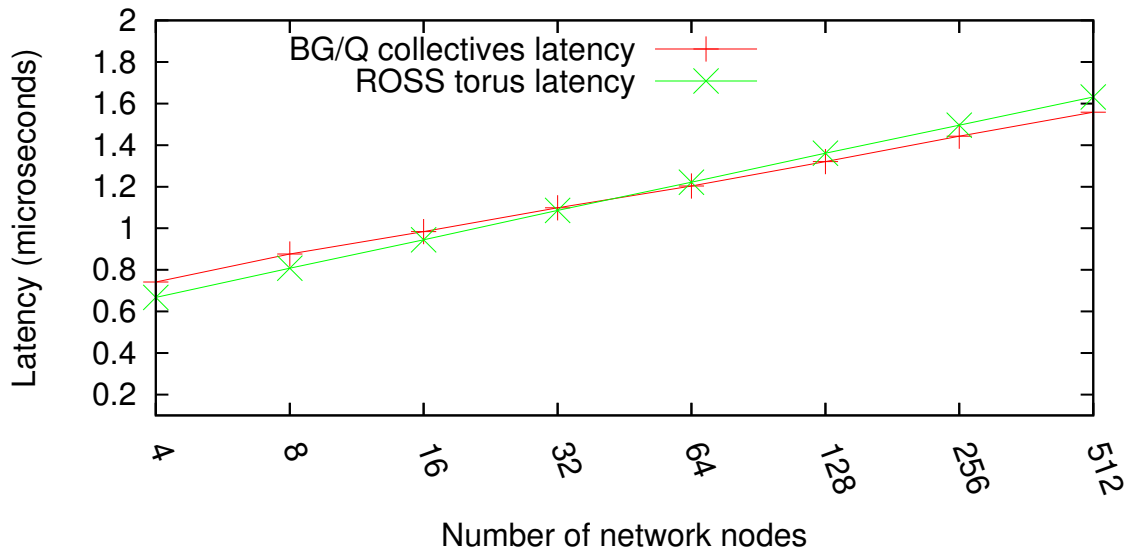
(a) ROSS torus latency vs. Real (IBM BG/Q) torus latency.
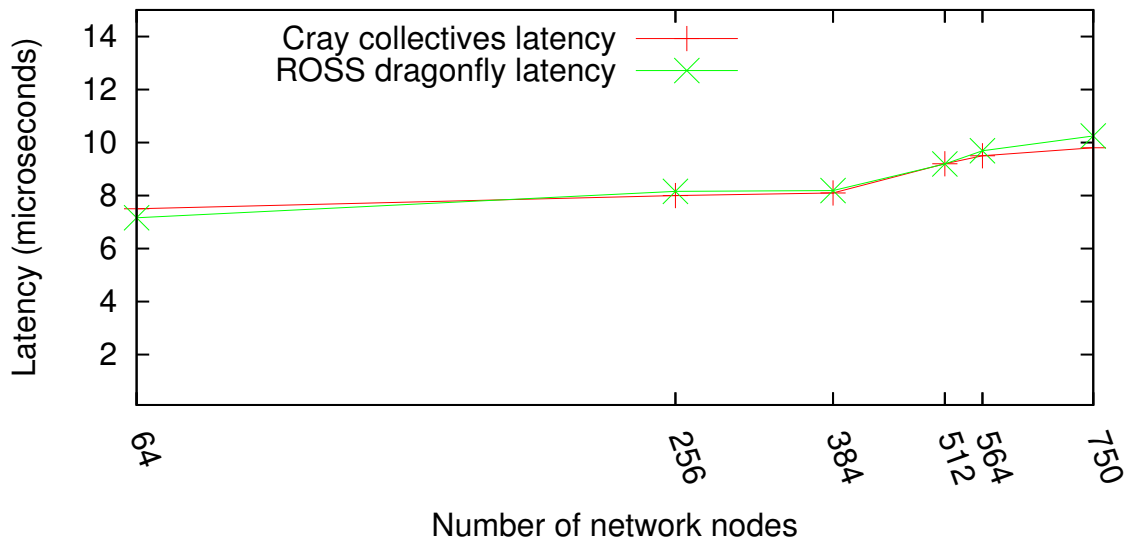
(b) ROSS dragonfly latency vs. Real (Cray XC30) latency.

Figure 3: Simulated and real latency of collective operations as a function of the number of nodes (software/MPI level overheads are not taken into account).

GiB of RAM and 8 cores per CPU (16 cores per node). Its aggregate performance is over 98 TF double precision, and the entire system has 6 TB of RAM.

4.1 Validation Study

To validate our ROSS torus collective modeling, we compared it with the latency of collective operations on the Blue Gene/Q system reported by (Chen et al. 2011). We used a data-driven approach to validate our simulation by configuring our ROSS collective communication tree with the measured latency per hop for 8-byte collective floating-point operations reported by (Chen et al. 2011). Using these measured latency
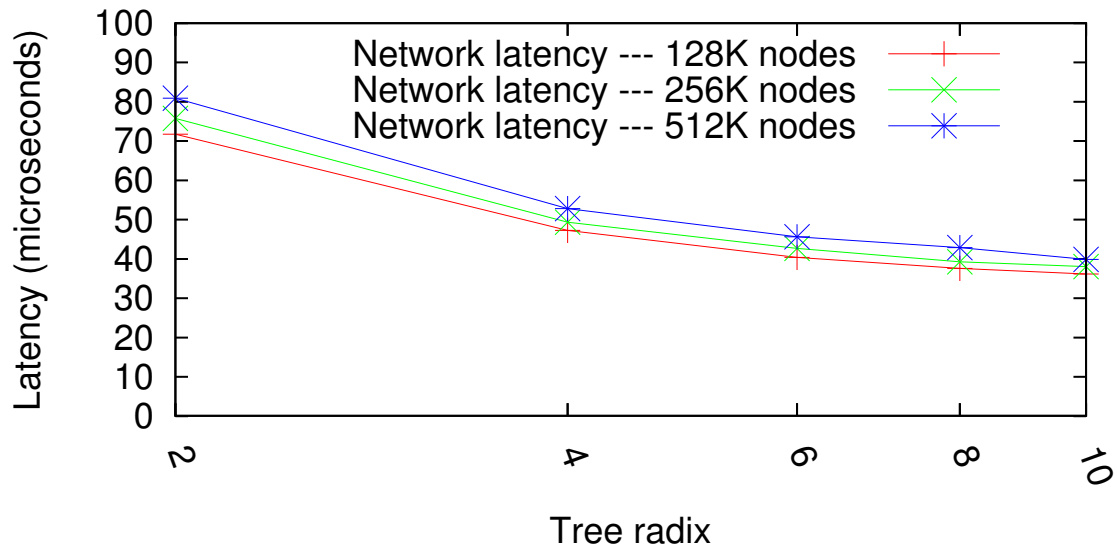
Figure 4: Simulated latency of torus network in microseconds with varying tree radix (workload: 25 consective collective operations with a small computation delay in between the calls).

trends and varying the number of nodes in the simulation, we obtain close latency agreement with the collective performance data reported for Blue Gene/Q as shown in Figure 3(a).

To validate the ROSS dragonfly collective modeling, we compared our collective modeling algorithm with the performance of `MPI_Allreduce()` latency reported by (Alverson et al. 2012) see Figure 15, which reports latency in terms of number of processes with 16 processes scheduled per node. Since we are interested in finding latency among network nodes, we measured latency in terms of network nodes instead of processes. We used the same data-driven approach to validate our dragonfly simulation by configuring the modeled tree with the latency per level and adjusting the branching ratio or degree of the tree since the Cray XC30 can support a tree with a branching ratio up to 32. A comparison of the Cray's reported latency trends and the ROSS dragonfly simulation is shown in Figure 3(b). Because of the high branching ratio of the tree, the latency changes slowly for the Cray XC30. We also kept a high branching ratio of the collective tree in our model to achieve a close latency agreement.

4.2 Network performance with varying tree degree

The branching ratio, or tree degree, is a determining factor in the performance of collective operations. To evaluate the effect of increasing tree degree on overall collective performance, we simulated a workload that has extensive calls for collective operations with a small computation delay (20 ns) in between the collective communication calls.

Figure 4 shows the latency trends of a 128K, 256K and 512K torus network with varying tree radix. Only hardware level collective overheads are taken into account on a per-tree level basis using data from the Blue Gene/Q performance study (Chen et al. 2011). One can see that the simulated network latency tends to decrease as the overall tree degree of the collective communication tree increases, leading to an improved overall collective performance. From a degree 2 to degree 4, there is a sharp drop in latency, whereas the latency difference becomes less evident as the tree degree grows larger. Beyond degree 10, the drop in latency due to tree radix is negligible (not shown in the figure).

The above experiment is an example of how large-scale discrete-event simulation can be used to find the best configuration parameters for collective communication in a network of any size. This capability

can be combined with workload information incorporating both point-to-point and collective operations in order to evaluate performance of entire large-scale scientific applications.

5 SIMULATOR PERFORMANCE

In this section, we demonstrate strong scaling of our collective communication model on 128K nodes. We also compare the simulation run-time of the ROSS conservative synchronization protocol with the optimistic synchronization protocol using a workload that has a combination of point-to-point and collective operations as described in Section 3.2.

5.1 Strong scaling of simulation performance with optimistic synchronization protocol

ROSS uses the time warp synchronization protocol to process events, which leads to an efficient simulation by allowing each LP to execute events optimistically until it detects an out-of-order event. When an out-of-order event is detected, the simulator rolls back previous events and then re-executes them in the correct order. Optimistic scheduling in ROSS has been shown to dramatically reduce the simulation run-time and state-saving overhead (Barnes Jr et al. 2013). Using the reverse computation capability in ROSS, our network simulations scale to a large processor count, even in cases where the models have a low look-ahead. In this section, we carry out a strong-scaling study of a 65K node torus network model on the Argonne Leadership Computing Facility (ALCF) system Tukey, with a simulation workload that has a phase of several nearest-neighbor point-to-point messages followed by another phase of collective communication operations, as described in Section 3.2.

Table 1: Strong scaling of 65K nodes torus network simulation with optimistic synchronization protocol. Number of collective calls per iteration = 10, number of point-to-point messages per iteration = 320 (message size=32 bytes).

Number of Processes	4	16	64	256	1024
Event efficiency(%)	98.91	97.65	97.12	96.10	94.46
Event rate (events/sec)	8,612.51	30,143	108,665	373,593	1,219,590
Running time (seconds)	9,202.9	2,629.43	729.39	212.15	64.98
Net events processed (million)	79.25	79.25	79.25	79.25	79.25

Table 1 demonstrates the strong-scaling results of the simulation performance for a 65K nodes torus network model. We have evaluated the performance results using three metrics: committed event rate per second, time to complete the simulation, and event efficiency. These metrics provide a picture of how the simulation performs with varying processor counts. Event efficiency in ROSS determines the amount of useful work that the simulation has performed. Equation 1 shows how ROSS event efficiency is calculated.

$$event\_efficiency = 1 - \frac{rolled\_back\_events}{total\_committed\_events} \tag{1}$$

Since ROSS uses the reverse computation mechanism by rolling back events, the simulator efficiency is proportional to the number of events that are rolled back. With zero rollbacks, the simulator yields 100% efficiency. Table 1 shows that the event efficiency with the 65K node network model is at minimum 94.46%, which indicates that the simulation is not spending excessive time in rolling back events. As the number of processes increases, the simulation event rate also scales with a maximum event rate of 1.2 million events per seconds on 1,024 MPI processes. Because of the increasing event rate, the simulation run-time also keeps decreasing as the number of MPI processes increase.

5.2 Comparison of simulation runtime with conservative synchronization protocol

The (optional) ROSS conservative mode protocol employs a global synchronization mechanism in which LPs are allowed to process events between a globally computed time stamp. The computed lowest time stamp is the smallest unprocessed event in the simulation plus a global look-ahead value. This approach requires more frequent synchronization among processes than does the Time Warp synchronization protocol and can lead to a slower simulation whenever the look-ahead value is small, as argued by Carothers and Perumalla (2010). In this section, we compare the simulation run-time of the network models with the ROSS conservative mode against the optimistic event-scheduling mode. The workload for this performance study comprises 40 point-to-point messages each having a message size of 512 bytes followed by 5 collective communication operations. There are 10 such iterations of point-to-point and collective communication phases.
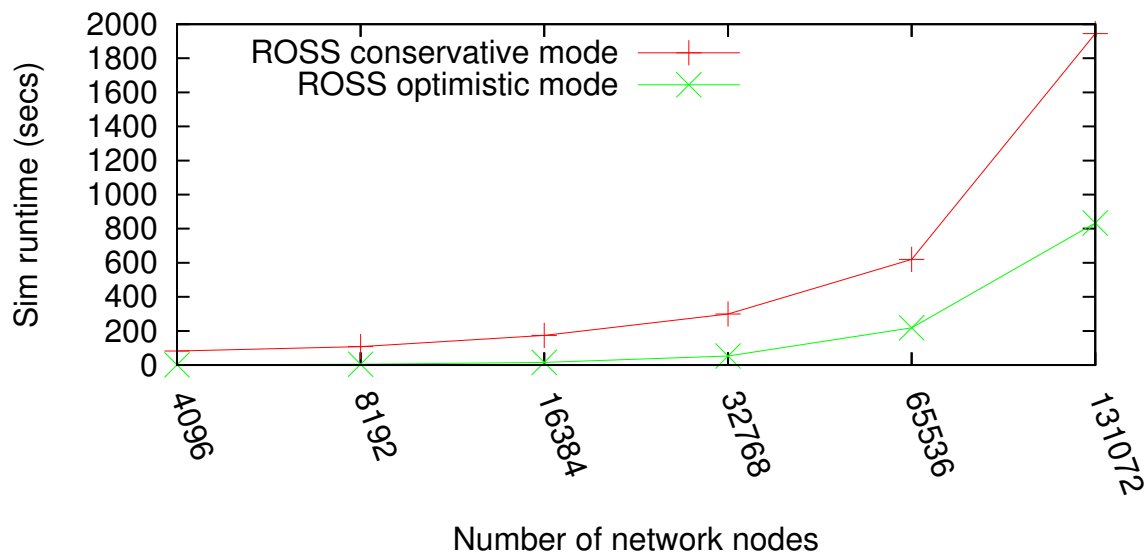

Figure 5: ROSS simulation performance: conservative vs. optimistic mode with a message payload size = 512 bytes, number of point-to-point messages per iteration = 40, number of collective operations per iterations = 5, number of communication iterations = 10.

Figure 5 presents a comparison of the simulation run-time for the ROSS conservative vs. optimistic mode with torus network sizes ranging from 4K nodes to 128K nodes. For both small and large network sizes, the optimistic mode performs more efficiently than the conservative mode. Excessive global synchronization required in the conservative mode is one factor that influences the conservative simulation performance. Another factor is the size of the look-ahead values. The event time stamps are architecture-specific, and the look-ahead values must be set to a correspondingly low value to prevent out-of-order event execution.

6   RELATED WORK

Previous discrete event simulators have implemented MPI collective operations using other techniques. Girona et al. (2000) modeled collective communication in the Dimemas simulator by using analytical models, while Rodriguez et al. (2004) fit linear models to Dimemas trace replays to generalize them. Hoefler et al. (2010) model collective communications in the LogGOPSim simulator by decomposing them into individual point to point operations based on common MPI library algorithms. The LogGOPSim simulator is a sequential discrete event simulator that models communication costs for point-to-point messages using a variation of the LogGP (Alexandrov et al. 1995) analytical model. SimGrid offers SMPI, a tool through

which large-scale MPI applications can be simulated online on a single node by using memory footprint reduction and sampling execution iterations (Clauss et al. 2011).

Algorithms for collective communications have been studied extensively in the literature, with particular focus on tuning algorithms for use on different networks, scales, and message sizes. Mitra et al. (1995) implemented a set of collective communication algorithms optimized for Intel supercomputers in the iCC library. Thakur and Rabenseifner (2005) developed collective algorithms for switched networks with the goal of minimizing latency for small messages and minimizing bandwidth use for large messages. Chan et al. (2007) developed a general, parameterized family of collective algorithms to be used for automatic optimization. Each of these works describes specific algorithms for expressing collective operations in terms of individual point-to-point messages.

7 CONCLUSION AND FUTURE WORK

Discrete-event simulation is becoming an increasingly important tool for exploring design space of future supercomputers, and many complex scientific applications now rely heavily on MPI collective operations. In this work we have therefore extended the capability of our large-scale torus and dragonfly network models to support collective communication operations. We have carried out a study to determine the impact of collective tree configuration on network performance. A simulation performance study for small- and large- scale network sizes has also been carried out using ROSS optimistic event scheduling. We have validated our ROSS collective model against the performance measurements reported for the Blue Gene/Q and Cray XC platforms. Overall, our work is geared at providing network designers with a capability to simulate certain configurations of a network with collective and point-to-point MPI communication.

As part of the future work, we plan to use real application communication traces, such as those collected as part of the Design Forward mini-app characterizations (Department of Energy 2014), to drive our network simulations. This will be a step towards enabling the network design space exploration using communication traces of complex scientific applications. We also plan to execute our collective communication model on Blue Gene systems and evaluate the simulation performance.

8 ACKNOWLEDGEMENTS

REFERENCES

Alexandrov, A., M. F. Ionescu, K. E. Schauser, and C. Scheiman. 1995. "LogGP: Incorporating long messages into the LogP Model – one step closer towards a realistic model for parallel computation". In *Proceedings of the Seventh Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '95, 95–105. New York, NY, USA: ACM.

Almási, G., P. Heidelberger, C. J. Archer, X. Martorell, C. C. Erway, J. E. Moreira, B. Steinmacher-Burow, and Y. Zheng. 2005. "Optimization of MPI collective communication on BlueGene/L systems". In *Proceedings of the 19th annual international conference on Supercomputing*, 253–262. ACM.

Alverson, B., E. Froese, L. Kaplan, and D. Roweth. 2012. "Cray XC series network". Technical Report White Paper WP-Aries01-1112, Cray Inc.

Barnes Jr, P. D., C. D. Carothers, D. R. Jefferson, and J. M. LaPre. 2013. "Warp speed: executing time warp on 1,966,080 cores". In *Proceedings of the 2013 ACM SIGSIM conference on Principles of Advanced Discrete Simulation*, 327–336. ACM.

Carothers, C. D., D. Bauer, and S. Pearce. 2002. "ROSS: A high-performance, low-memory, modular Time Warp system". *Journal of Parallel and Distributed Computing* 62 (11): 1648–1669.

Carothers, C. D., and K. S. Perumalla. 2010. "On deciding between conservative and optimistic approaches on massively parallel platforms". In *Proceedings of the 2010 Winter Simulation Conference (WSC)*, edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, 678–687. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Chan, E., M. Heimlich, A. Purkayastha, and R. van de Geijn. 2007, September. "Collective communication: Theory, practice, and experience: Research articles". *Concurr. Comput. : Pract. Exper.* 19 (13): 1749–1783.

Chen, D., N. A. Eisley, P. Heidelberger, R. M. Senger, Y. Sugawara, S. Kumar, V. Salapura, D. L. Satterfield, B. Steinmacher-Burow, and J. J. Parker. 2011. "The IBM Blue Gene/Q interconnection network and message unit". In *2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2011.*, 1–10. IEEE.

Clauss, P.-N., M. Stillwell, S. Genaud, F. Suter, H. Casanova, and M. Quinson. 2011. "Single node on-line simulation of MPI applications with SMPI". In *Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International*, 664–675. IEEE.

Dally, W. J., and B. P. Towles. 2004. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann.

Department of Energy 2013. "Exascale initiative: Design forward program". Accessed July 31, 2013, "http://www.exascaleinitiative.org/design-forward".

Department of Energy 2014. "Design Forward characterization of DOE mini-apps". Accessed June 05 2014, http://portal.nersc.gov/project/CAL/designforward.htm.

Dongarra, J. 2012. *On the future of high-performance computing: How to think for peta and exascale computing*. Hong Kong University of Science and Technology.

Girona, S., J. Labarta, and R. M. Badia. 2000. "Validation of Dimemas Communication Model for MPI Collective Operations". In *Proceedings of the 7th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, 39–46. London, UK, UK: Springer-Verlag.

Hoefler, T., T. Schneider, and A. Lumsdaine. 2010, June. "LogGOPSim - Simulating large-scale applications in the LogGOPS model". In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, 597–604: ACM.

Kim, J., W. Dally, S. Scott, and D. Abts. 2009. "Cost-efficient dragonfly topology for large-scale systems". *Micro, IEEE* 29 (1): 33–40.

Liu, N., C. Carothers, J. Cope, P. Carns, and R. Ross. 2012. "Model and simulation of exascale communication networks". *Journal of Simulation* 6 (4): 227–236.

Mitra, P., D. G. Payne, L. Shuler, R. Geijn, and J. Watts. 1995. "Fast collective communication libraries, please". In *Proceedings of the Intel Supercomputing Users' Group Meeting*.

Mubarak, M., C. D. Carothers, R. Ross, and P. Carns. 2012. "Modeling a million-node dragonfly network using massively parallel discrete-event simulation". In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion,*, 366–376. IEEE.

Mubarak, M., C. D. Carothers, R. B. Ross, and P. Carns. 2014. "A case study in using massively parallel simulation for extreme-scale torus network codesign". In *Proceedings of the 2nd ACM SIGSIM/PADS conference on Principles of Advanced Discrete Simulation*, 27–38. ACM.

Nicol, D. M., C. C. Michael, and P. Inouye. 1989. "Efficient aggregation of multiple PLs in distributed memory parallel simulations". In *Proceedings of the 21st Conference on Winter Simulation*, edited by K. J. Musselman, P. Heidelberger, and E. A. MacNair, 680–685. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Rodriguez, G., R. M. Badia, and J. Labarta. 2004. "Generation of simple analytical models for message passing applications". In *Proceedings of the Euro-Par Conference*, 183–188.

Sahni, O., C. D. Carothers, M. S. Shephard, and K. E. Jansen. 2009. "Strong scaling analysis of a parallel, unstructured, implicit solver and the influence of the operating system interference". *Scientific Programming* 17 (3): 261–274.

Thakur, R., and R. Rabenseifner. 2005. "Optimization of collective communication operations in MPICH". *International Journal of High Performance Computing Applications* 19:49–66.

Vaughan, C., M. Rajan, R. Barrett, D. Doerfler, and K. Pedretti. 2011. "Investigating the impact of the Cielo Cray XE6 architecture on scientific application codes". In *IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011*, 1831–1837. IEEE.

Vetter, J. S., S. Lee, D. Li, G. Marin, C. McCurdy, J. Meredith, P. C. Roth, and K. Spafford. 2013. "Quantifying architectural requirements of contemporary extreme-scale scientific applications". In *International Workshop on Performance Modeling, Benchmarking and Simulation of HPC Systems (PMBS13)*. Denver, CO.

AUTHOR BIOGRAPHIES

MISBAH MUBARAK is a Ph.D. student in the Computer Science Department at Rensselaer Polytechnic Institute (RPI) . Her research interests are modeling and simulation of massively parallel systems. She received her master's degree in computer science from RPI as a U.S. Fulbright scholar. She also has experience in developing multitier enterprise applications at CERN, Switzerland, and Teradata corporation. Her email address is mubarm@cs.rpi.edu.

CHRISTOPHER D. CAROTHERS is a professor in the Computer Science Department at Rensselaer Polytechnic Institute. He received the Ph.D., M.S., and B.S. from Georgia Institute of Technology in 1997, 1996, and 1991, respectively. Professor Carothers is an NSF CAREER Award winner as well as Best Paper award winner at the PADS workshop for 1999, 2003, and 2009. Since joining Rensselaer, he has secured research funding from a wide variety of agencies including the NSF, the U.S. Department of Energy, Army Research Laboratory and Air Force Research Laboratory, as well as several companies, including IBM, General Electric, and AT&T. His simulation research involves the creation of high-fidelity models of extreme-scale wireless and wired networks and computer systems. Additionally, Professor Carothers serves as the director for the Center for Computational Innovations (CCI) at Rensselaer. CCI is a partnership among Rensselaer and IBM. The center currently supports a network of more than 850 researchers, faculty, and students from 50 universities, government laboratories, and companies across a diverse spectrum of disciplines. Currently, the CCI operates HPC resources exceeding one petaflop in compute power. The flagship supercomputer system is a 1 PF IBM Blue Gene/Q with 80 terabytes of memory. His email address is chrisc@cs.rpi.edu.

ROBERT B. ROSS is a computer scientist in the Mathematics and Computer Science Division of Argonne National Laboratory and a senior fellow in the Northwestern-Argonne Institute for Science and Engineering. He is also an adjunct assistant professor in the Department of Electrical and Computer Engineering at Clemson University. He received his Ph.D. in computer engineering from Clemson University in 2000. He currently holds several leadership positions at Argonne and in the U.S. DOE computing community, including serving as deputy director of the Scientific Data Management, Analysis, and Visualization Institute and as co-lead of the Data Management component of the DOE Office of Science Exascale Computing activity. His email address is rross@mcs.anl.gov.

PHILIP CARNS is a software development specialist in the Mathematics and Computer Science Division at Argonne National Laboratory. He received his Ph.D. in computer engineering from Clemson University in 2005. He also has industry experience in applying grid technology to business data management. His email address is carns@mcs.anl.gov.