Warp Speed: Executing Time Warp on 1,966,080 Cores

Peter D. Barnes, Jr.
Lawrence Livermore National Laboratory
7000 East Avenue
Livermore, CA 94550
pdbarnes@llnl.gov

Christopher D. Carothers
Dept. of Computer Science
Rensselaer Polytechnic Institute
110 8th Street
Troy, NY 12180
chrisc@cs.rpi.edu

David R. Jefferson
Lawrence Livermore National Laboratory
7000 East Avenue
Livermore, CA 94550
jefferson6@llnl.gov

Justin M. LaPre
Dept. of Computer Science
Rensselaer Polytechnic Institute
110 8th Street
Troy, NY 12180
laprej@cs.rpi.edu

ABSTRACT

Time Warp is an optimistic synchronization protocol for parallel discrete event simulation that coordinates the available parallelism through its rollback and antimessage mechanisms. In this paper we present the results of a strong scaling study of the ROSS simulator running Time Warp with reverse computation and executing the well-known PHOLD benchmark on Lawrence Livermore National Laboratory's *Sequoia* Blue Gene/Q supercomputer. The benchmark has 251 million PHOLD logical processes and was executed in several configurations up to a peak of 7.86 million MPI tasks running on 1,966,080 cores. At the largest scale it processed 33 trillion events in 65 seconds, yielding a sustained speed of 504 billion events/second using 120 racks of *Sequoia*. This is by far the highest event rate reported by any parallel discrete event simulation to date, whether running PHOLD or any other benchmark. Additionally, we believe it is likely to be the largest number of MPI tasks ever used in any computation of any kind to date.

ROSS exhibited a super-linear speedup throughout the strong scaling study, with more than a 97x speed improvement from scaling the number of cores by only 60x (from 32,768 to 1,966,080). We attribute this to significant cache-related performance acceleration as we moved to higher scales with fewer LPs per core.

Prompted by historical performance results we propose a new, long term performance metric called *Warp Speed* that grows logarithmically with the PHOLD event rate. As we define it our maximum speed of 504 billion PHOLD events/sec corresponds to *Warp 2.7*.

We suggest that the results described here are significant because they demonstrate that direct simulation of planetary-scale discrete event models are now, in principle at least, within reach.

Categories and Subject Descriptors

C.1.4 [**Computer Systems Organization**]: PROCESSOR ARCHITECTURES—*Parallel Architectures*; I.6.8 [**Computing Methodologies**]: SIMULATION AND MODELING—*Discrete Event, Parallel*

General Terms

Experimentation, Performance

Keywords

Blue Gene/Q, Parallel Discrete-Event Simulation, Time Warp

1. INTRODUCTION

In 2009, the Time Warp synchronization protocol demonstrated highly efficient strong scaling (*i.e.*, model size held constant while processor or core count is increased) using Rensselaer's Optimistic Simulation System (ROSS) [2]. Here, we extend those results by demonstrating continued good scaling behavior at much larger scales on the new Blue Gene/Q supercomputer, *Sequoia*, located at Lawrence Livermore National Laboratory (LLNL). *Sequoia* is dedicated to the National Nuclear Safety Administrations' (NNSA's) Advanced Simulation and Computing (ASC) program for stewardship of the nation's nuclear weapons stockpile, which is a joint effort by LLNL, Los Alamos National Laboratory and Sandia National Laboratories. Jefferson's Time Warp [20] is an *optimistic synchronization protocol.* The key idea behind it is to allow the parallel *logical processes* of the simulation to execute freely and speculatively as much as possible, and to synchronize using *rollback* and *anti-messages* when necessary to correct for events executed out of order.

Since 2009, ROSS has been used in a number of large-scale modeling and simulation efforts. Most recently it has been used to model next generation exascale storage systems [23, 25]. Additionally, Liu et al. [24] created one of the first massively parallel discrete event models of a multi-dimensional torus network that is capable of model sizes in excess of 1 billion nodes. More recently, Mubarak et al. [27] demonstrated strong scaling of a highly accurate Dragonfly network model using both Blue Gene/P and Blue Gene/Q supercomputers. The dragonfly network is particularly interesting because of the use of its global channel links. One might think that these links would have the potential to induce excessive cascading rollbacks, but that was not the case. Using 65,536 MPI tasks mapped to 16,384 Blue Gene/Q cores, the efficiency was over 99% for a dragonfly network configured with 50 million nodes, and yielded an event rate in excess of 1 billion events/second. Similarly, Gonsiorowski et al. [16] demonstrate over 130x performance improvement for a 1 billion gate circuit model based off the OpenSparc T2 processor design when using 1024 Blue Gene/L processors compared with a baseline performance of a single Xeon processor. The rollback mechanism has often been viewed as the limiting factor in the Time Warp mechanism because the number and depth of rollbacks is usually not explicitly bounded [26], so that the rollback behavior is sometimes characterized as exhibiting "risk" or even considered (somewhat colorfully) as having a "dark side" [28]. However, our results to date suggest that the potential for complex rollback dynamics were not realized and ROSS continues to demonstrate highly efficient and scalable results in tackling some of today's most challenging discrete event simulations.

The key question asked in this this paper is: How much more scalability is possible? To address this we conducted a detailed experimental study of ROSS executing the PHOLD benchmark [13] model on two systems: the two-rack 32,768-core Blue Gene/Q located at Rensselaer's Computational Center for Nanotechnology Innovations (CCNI) supercomputer and *Sequoia*, a 120-rack, 1,966,080-core Blue Gene/Q supercomputer, currently the second-ranked supercomputer in the world according to the Top 500 list. In the next Section (Section 2), we describe the Blue Gene/Q architecture, followed by the design of ROSS in Section 3 and our proposed *Warp Speed* performance metric in Section 4. Section 5 presents the detailed performance results from both computing platforms. Section 6 places these results in the context of prior research and provides some thoughts on what new extreme-scale models this level of performance will enable. Section 7 summarizes these results, draws some conclusions and lays out some paths for future research.

2. THE BLUE GENE/Q ARCHITECTURE

The Blue Gene/Q is the third generation system in the IBM Blue Gene supercomputer family with the Blue Gene/L and /P coming previously in 2004 and 2007, respectively. The Blue Gene/Q has made a substantial leap in computational capabilities over these previous generations. The largest systems are currently the 48 rack *Mira* system located at Argonne National Laboratory (ANL) and the 120 rack *Sequoia* system located at LLNL. (Sequoia was recently split into a 96-rack system that retains the name Sequoia and a 24-rack system named Vulcan).

The Blue Gene/Q's A2 processor [11] (one per Blue Gene/Q node), shown in Figure 1, is a 45 nm chip comprised of 18 cores (PU's in the Figure) running at 1.6 GHz. The first 16 cores are exclusively used to execute user-level compute tasks while a 17th core is reserved to perform OS functionality. To improve overall processor manufacturing yields, the 18th core is only enabled if during final testing one of the other cores fails to operate correctly. The dedicated OS core design enables the use of a much more capable compute-node

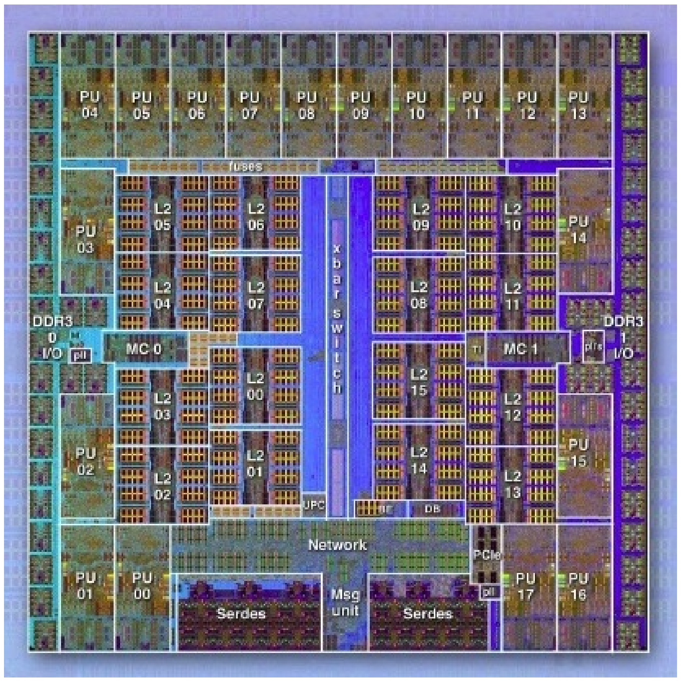

Figure 1: IBM Blue Gene/Q A2 18-way processor layout diagram from HPCWire via Google Images.

OS than prior Blue Gene systems, and supports `mmap`, shared libraries, etc) without introducing OS jitter [3]. Each core supports up to 4 hardware thread or MPI task contexts, with each context having a full complement of thirty-two 64-bit registers. Each core supports up to 4-wide double precision SIMD calculations. In terms of execution models, the Blue Gene/Q supports up to 64 MPI tasks per node (up to 4 tasks per core) or a mix of MPI tasks and threads. The peak performance of the A2 processor is 204.8 gigaflops, and it consumes up to 55 watts of power. Blue Gene/Q nodes do not have GPUs as some other large scale supercomputer nodes do. In general, since the ROSS/PHOLD benchmark does not make much use of floating point operations, floating point performance is not a performance or scaling barrier.

A fully-connected crossbar switch connects all 16 cores to the 32 MB, L2 cache (realized as 16, 2 MB L2 cache units), thus all cores have a uniform cache-memory access. The amount of memory is fixed on each compute node at 16 gigbytes, with a total bandwidth of 42.6 GB/sec. In contrast to the Blue Gene/P, which only has four PowerPC 450 cores operating a 850 MHz, resulting in 13.6 gigaflops, the Blue Gene/Q provides a 15x peak floating point performance boost. However, a potential barrier to reaching that performance increase is the available node memory bandwidth. The Blue Gene/P provided 13.6 GB/sec which yields a memory bandwidth to FLOP ratio of 1. However, the Blue Gene/Q has a memory bandwidth to FLOP ratio of only 0.21.

Lowering this ratio even further relative to that of the Blue Gene/P is the doubling of the address space size from 32 bits on the Blue Gene/P to 64 bits on the Blue Gene/Q. Thus, for pointer intensive applications like ROSS, the effective pointer access rate has only improved by a factor of 1.5x (((42.6 GB/s/64 bits)/(13.6 GB/s/32 bits)) in going from the Blue Gene/P to the Blue Gene/Q. To help mitigate the lower overall memory bandwidth to FLOP ratio, the Blue Gene/Q provides an L1 cache prefetch engine for each core along with scalable atomic operations.

Connected to each node is a five dimensional (5-D) torus network with 10 serial links per node, each capable of sending and receiving 2 GB/sec [8]. An 11th link is dedicated to I/O and connects to the I/O node set. Unlike previous Blue Gene systems which have a dedicated I/O node for a set of compute nodes (*e.g.*, `pset`), the Blue Gene/Q has a more flexible approach which pulls the I/O nodes outside of the primary compute fabric. In this configuration, up to eight I/O nodes, identical to the compute nodes from a hardware perspective, are grouped into a drawer. A single rack of Blue Gene/Q can drive up to 8 drawers (64 nodes) of I/O nodes. Each I/O node can service about 2 GB/sec of parallel file system data.

3. DESIGN OF ROSS

ROSS is an open source, massively parallel discrete event simulation engine supporting both YAWNS-like conservative and Time Warp optimistic event scheduling algorithms [5]. (See: `http://ross.cs.rpi.edu` for download details.) A central feature of ROSS' overall design has been its efficient use of memory and cache-aware implementation. In particular, taken from its previous shared-memory implementation [4], ROSS uses pointers to data structures as opposed to indexing into arrays. For a detailed discussion of ROSS' data structures and core algorithms used to process events, including GVT, we refer the reader to [2]. In this paper, we focus more on memory usage in ROSS and the overall event processing workflow.

In the current version of ROSS, MPI is used to spawn an instance of the model (MPI task), including all the functions for event scheduling and processing, rollback, and GVT computations. During initialization, we allocate space for the logical process (LP) state, all events (network and application), random number generator (RNG) states, and rollback support data structures. In this version of ROSS the MPI tasks use no internal threading. ROSS yields the best performance on the Blue Gene/Q system when there are four MPI tasks per core, corresponding to the four hardware contexts in each core. Thus, a node on the Blue Gene/Q will support 64 MPI tasks, each running one instance of ROSS and hosting many logical processes (LPs) of the simulation model (in this case, PHOLD).

Once memory allocations are complete, ROSS initializes each LP by first computing the correct random number generator (RNG) seed states based on the LP's id. The RNG is linear congruential with a period of 2^{121}. Each seed takes 128 bits, represented as four 32 bit integers [22]. To ensure that RNG seeds do not correlate, the start state of each seed is 2^{70} calls apart from the previous seed. Additionally, care is taken such that for any fixed LP count, the seeds are deterministicly computed irrespective of the processor count. This ensures that models runs will be deterministic and identical at different scales in a strong scaling study. Finally, we note that ROSS can support multiple RNGs per LP to avoid correlations across activities within an LP that would otherwise be independent.

Next, ROSS invokes the `init` function pointer for each LP to initialize the model's state and to schedule the start set of events for that LP. In order to schedule a new event, the application allocates a new event from the free event list. Each MPI task has its own free event list. During initialization all start events must be sent to "self" because not all LPs are initialized and installation of the mapping function between LPs and MPI tasks may not be complete.

Once all the LPs have have completed their initialization, the optimistic or conservative event scheduler is invoked. Focusing on the optimistic scheduler, there are two key parameters that control its behavior. The first is `GVT_interval`, which is the number of times through the main scheduler loop before a GVT computation will be started. As described in [2], GVT calculation in ROSS is a synchronous activity that uses the `MPI_Allreduce` collective operation. The second key scheduler parameter is `batch`. Inside the main scheduler loop, `batch` events are processed before polling the network for remote events and checking to see if a GVT computation is needed.

Polling for remote events uses a combination of asynchronous `MPI_Iprobe` and `MPI_Irecv` operations. Between successive GVT computations, on average `GVT_interval` × `batch` events will be processed by each MPI task. On the receipt of a remote message (from another task), a container is allocated pointing to the new event buffer. This container is placed in an AVL Tree [12] to facilitate fast event cancellation when an anti-message arrives. AVL Trees provide a very fast search operation, using the anti-message's timestamp, sender MPI rank, and event age as a key to find the event that must be canceled. If that event has already been processed, then the LP will be rolled back to the event just prior to the canceled event in virtual time order. The ROSS rollback mechanism uses a *reverse computation* [6] mechanism in which the model supplies a reverse event handler that reverts the LP state to the point before the event being rolled back, including the states of any RNGs that were used during event processing.

Each LP in the course of processing an event can schedule new events to be processed at some future point in virtual time. If the newly scheduled event is local (*i.e*, the destination LP is mapped to the current MPI task), then that event is immediately placed in the task's priority queue, which is implemented using a Splay Tree [35]. Otherwise, the event and its model data are sent in a contiguous memory block over MPI using the asynchronous `MPI_Isend`. ROSS manages its own `MPI_request` memory buffers which are used for the `MPI_Isend` and `MPI_Irecv` operations to ensure a sufficient number of `MPI_Isend` and `MPI_Irecv` are posted/in progress at all times.

Whenever a new GVT value is computed, all events with timestamps less than GVT are *fossil collected*. To speed up this process, ROSS creates a special container data structure called a *kernel process* (KP) which holds all the processed events in single linked list for a group of LPs. This approach reduces the amount of search require to find old event data to reclaim. Typically, each MPI task will have between 8 and 64 KPs.

4. WARP SPEED

There is a growing history of work on massively parallel simulation on state of the art supercomputer systems. The first PDES performance study to focus on the Blue Gene supercomputer platform is [30], in which PHOLD performance results for conservative, optimistic and mixed-mode PDES protocols on the Blue Gene/L were presented using the *μsik* parallel discrete event simulator. The next two Blue Gene/PDES performance studies [19, 2] demonstrated the performance of ROSS as it was in 2008. These studies were the first to demonstrate event rates from the 100's of millions to billions of events per second, and scalabil-

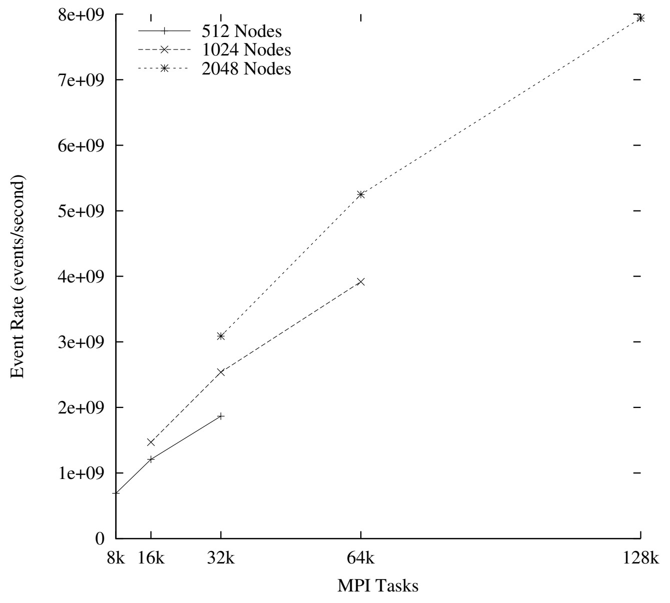

Figure 2: CCNI: PHOLD event rate performance as a function of MPI tasks for configurations of 1 rank per core, 2 ranks per core and 4 ranks per core. All runs are configured with 5,242,880 LPs, 8 `batch` events, 512 `GVT_interval` and 0.10 `lookahead`.

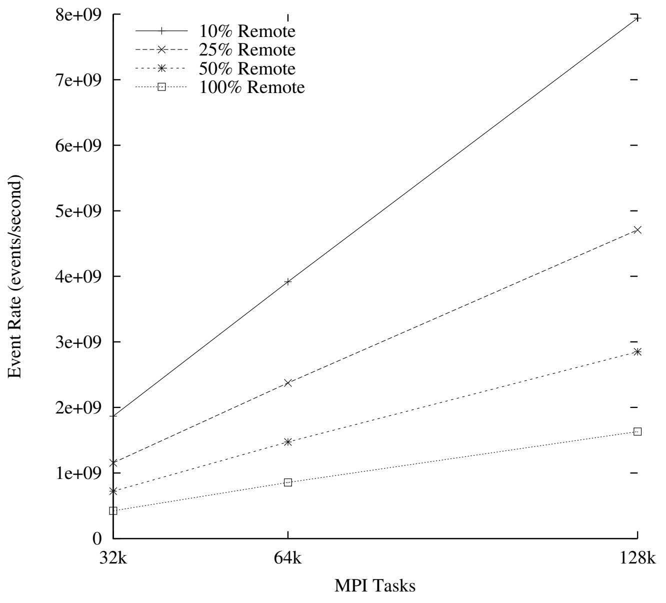

Figure 3: CCNI: PHOLD event rate performance as a function of MPI tasks for configurations with 10%, 25%, 50% and 100% remote communication fraction. All runs are configured with 5,242,880 LPs, 8 `batch` events, 512 `GVT_interval` and 0.10 `lookahead`.

ity to 65,536 cores. Additionally, Perumalla demonstrated epidemic models [32] using 65,536 Cray XT5 processors, and in [31] he demonstrated simulation of massively parallel MPI programs (with millions of virtual MPI tasks) using over 216,000 CrayXT 5 cores. Carothers and Perumalla [5] provided experimental results and guidance for determining when conservative or optimistic synchronization should be used for massively parallel discrete event simulation.

Thus, since 2007, our community has observed an event rate growth in PHOLD from 100 million to over 500 billion as we described in Section 5. With a nearly 5000x performance improvement in only 6 years, plotting historic speeds no longer makes sense on a linear scale. We suggest that it is time to move to a logarithmic simulator speed metric, similar to the Richter scale for earthquake amplitude, or to decibels used in many engineering domains. We propose that any discrete event simulator, whether conservative or optimistic, that achieves an overall sustained (net) event rate on the PHOLD benchmark of p events per second be described as achieving *Warp* w speed where

$$w = log_{10}(p) - 9$$

Observe that a *Warp* speed is less than zero for PHOLD event rates less than 1 billion per second. To obtain a *Warp 10.0* would require the ability to process 10 exa-events, or 10^{19} events per second.

5. EXPERIMENTAL RESULTS

Our experimental study used two separate parallel systems. The first is a two-rack, 32,768-core, 418 teraflop Blue Gene/Q system located at Rensselaer's Computational Center for Nanotechnology Innovations (CCNI). This system has 32,768 cores, 32 terabytes of RAM and is configured with 64 I/O nodes (8 drawers) making it one of the most I/O rich Blue Gene configurations for its size fielded today. The CCNI's Blue Gene/Q system is configured with driver level V1R2M0 Efix 13. This driver provides low level functionality to the Blue Gene/Q system especially related to the Parallel Active Message Interface(PAMI) [7, 21]. The MPI implementation uses PAMI to efficiently transmit messages within the 5-D torus network. The ROSS compiler settings used for all Blue Gene/Q runs were: `-qflag=i:i -qattr=full -O3`.

The second system was the *Sequoia* Blue Gene/Q system located at LLNL. Most of the time this system has been, and will be, comprised of 96 racks, 1,572,864 cores and over 1.5 petabytes of RAM, with a peak performance of 20 petaflops. In that configuration it is currently the number #2-ranked supercomputer in the world on the Top500 list (see: `top500.org`).

However, for a brief period during which this study was conducted, *Sequoia* was joined with the *Vulcan* 24-rack system to form a single 120-rack system with 1,966,080 cores and nearly 2 petabytes of RAM. While the compiler on *Sequoia* is the same as the CCNI system, the *Sequoia* experiments used two different driver levels in the OS: V1R2M0 Efix 13 and V1R2M0 Efix 15. The Efix 15 driver enabled MPI to scale to 96 and 120 racks, which was not possible with Efix 13. Additionally, for the Efix 15 runs, we had to add the following set of environment variable options in order to force PAMI and MPI to allocate the right amounts of internal memory:

- `BG_MAPCOMMONHEAP=1`

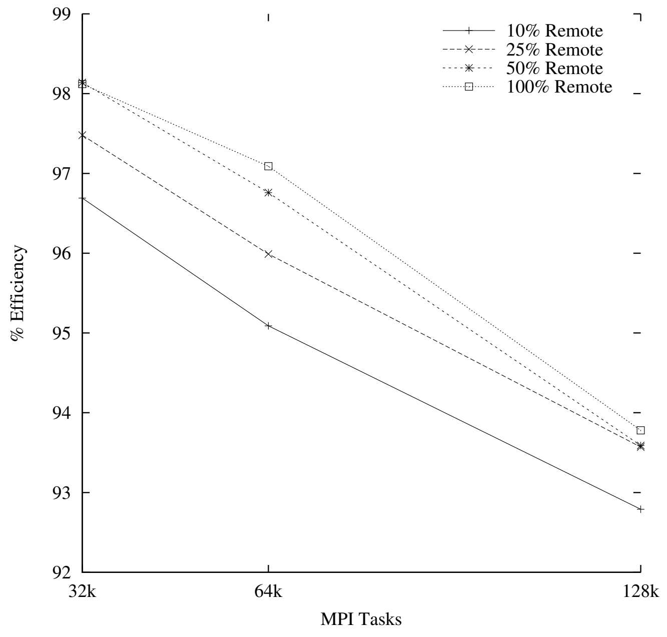

Figure 4: CCNI: PHOLD Event Efficiency (see text for definition) as a function of MPI tasks for configurations with 10%, 25%, 50%, and 100% remote communication fraction. For all data points, each node executes 64 MPI tasks. All runs are configured with 5,242,880 LPs, 8 `batch` events, 512 `GVT_interval` and 0.10 `lookahead`.

- `BG_SHAREDMEMSIZE=128`
- `PAMI_CLIENT_SHMEMSIZE=5M`
- `PAMI_GLOBAL_SHMEMSIZE=70M`
- `PAMI_SHMEM_NNODE_THRESHOLD=131072`

The PHOLD benchmark model is a derivative of the classic HOLD model [37] that was extended for parallel discrete event simulation in [13] and for reverse computation in [6]. We varied a number of ROSS and runtime configuration parameters, including: `batch`, `GVT_interval`, MPI tasks per core, and the percentage of remote communications. Each PHOLD event either schedules an event for "self" or for some other randomly selected LP, in either case with an exponentially-distributed time-stamp increment having mean of 1.0. An optional static nonzero parameter, `lookahead`, can be added to the time-stamp increment to enable direct comparison between optimistic schedulers (which can handle events scheduled with arbitrarily small positive delays) and conservative schedulers (which require a predetermined positive (nonzero) lower bound on event delays).

All *Sequoia* runs were configured with exactly 251,658,240 LPs, 16 events per LP, 8 KPs per MPI rank, 10% remote communication fraction, 0.1 `lookahead`, 8 `batch`, 512 `GVT_interval`, and 4 MPI tasks per core. The 10% remote fraction means that 90% of the events that each LP scheduled were "local" (*i.e.*, scheduled for itself) and 10% were "remote" (*i.e.*, scheduled for a different LP uniformly randomly selected from among all of the others). In case anyone wants to do a similar scaling experiment with a conservative synchronization algorithm we introduced a small amount of `lookahead`, 0.1, although the Time Warp algorithm makes no use of this. We set the mean of the exponential part of the future event time distribution to 0.90. The event itself does no substantial computational work other than random number generation for selecting the destination LP and future time-stamp. At the largest scale of 120 racks and 7,864,320 MPI ranks this yields a relatively small workload of exactly 32 LPs and an average of 512 live events per MPI rank.

We first present the results from the CCNI Blue Gene/Q and followed by our performance results from *Sequoia*.

5.1 CCNI Blue Gene/Q Results

These early CCNI tests were intended both to provide comparison to the *Sequoia* results at small scales and to mimic the same LP workloads that *Sequoia* would have at 96 racks so that we might anticipate what to expect at the largest scale. For the CCNI tests all runs were configured with 5,242,880 LPs, 8 `batch` events, 512 `GVT_interval` and 0.10 `lookahead`.

In the first series of experiments, we investigated the impact of over-committing MPI tasks to each Blue Gene/Q core. Recall that the A2 processor supports up to four hardware thread contexts per core. Thus, it is possible to support 1, 2 or 4 MPI tasks or threads per core. Figure 2 reports the PHOLD event rate performance as a function of the total number of MPI tasks for run sizes of 512, 1024 and 2048 nodes, which correspond to using a 1/2 rack, 1 full rack and 2 full racks. Unlike previous hyper-threaded architectures, such as those used by Intel [38], we observe nearly linear event rate performance improvement as each node is configured to use 2 or 4 threads. We do observe less than linear scaling when going from 2 to 4 threads but it is significantly better than we have observed on our hyper-threaded platforms. The peak event rates are nearly 2 billion/sec, 4 billion/sec and 8 billion/sec for 32,768, 65,536 and 131,072 MPI tasks respectively. In comparison to our previous peak PHOLD performance of just over 12 billion [2] on 65,536 Blue Gene/P cores, we observe here that this particular model has 5x the LPs and events, leading to greater overheads in the priority queue and event processing because of the larger memory footprint (*i.e.*, more cache misses).

The next two figures report the PHOLD event rate performance (Figure 3) and efficiency (Figure 4) as a function of the MPI Task count, varying the remote event percentages. While each remote event percentage curve has its own scaling slope, we do not observe any decrease in performance as the node count increases. This suggests that the 5-D torus network of the Blue Gene/Q enables the PHOLD model to continue to scale up to at least 131,072 MPI tasks. This is in spite of the large number of remote events; in one case 100% of the events are scheduled to remote LPs. The peak event rates at 131,072 MPI tasks for 10%, 25%, 50% and 100% remote event communications are 8 billion, 5 billion, 3 billion, and nearly 2 billion, respectively. In terms of overall speedup, we observe that the slope of the 10% remote communication line is super-linear. We attribute this scalability to better cache performance as more cores are used for the same size problem.

The PHOLD model efficiency mirrors the event rate performance as shown in Figure 4. Here, we observe that the overall efficiency ranges between 98% down to 93%. This suggests the Blue Gene/Q network is quickly moving messages between LPs so that rollback rates are low even at high remote event communication rates.

Cores	MPI Tasks	$Events_{Total}$	$Events_{RB}$	$Events_{Net}$	Efficiency	Time (sec)	Speed	Warp Speed
16,384	65,536	3.316E+13	1.750E+11	3.298E+13	0.9947	14964.2	2.20E+09	0.34
16,384	65,536	3.316E+13	1.750E+11	3.298E+13	0.9947	15027.2	2.20E+09	0.34
16,384	65,536	3.316E+13	1.750E+11	3.298E+13	0.9947	15027.7	2.19E+09	0.34
32,768	131,072	3.325E+13	2.678E+11	3.298E+13	0.9919	6799.6	4.85E+09	0.69
32,768	131,072	3.325E+13	2.678E+11	3.298E+13	0.9919	6822.7	4.83E+09	0.68
32,768	131,072	3.325E+13	2.678E+11	3.298E+13	0.9919	6821.5	4.84E+09	0.68
65,536	262,144	3.339E+13	4.054E+11	3.298E+13	0.9879	3128.5	1.05E+10	1.02
65,536	262,144	3.339E+13	4.054E+11	3.298E+13	0.9879	3128.2	1.05E+10	1.02
65,536	262,144	3.339E+13	4.054E+11	3.298E+13	0.9879	3127.6	1.05E+10	1.02
131,072	524,288	3.358E+13	5.988E+11	3.298E+13	0.9822	1445.1	2.28E+10	1.36
131,072	524,288	3.358E+13	5.988E+11	3.298E+13	0.9822	1447.1	2.28E+10	1.36
131,072	524,288	3.358E+13	5.988E+11	3.298E+13	0.9822	1447.1	2.28E+10	1.36
786,432	3,145,728	3.455E+13	1.562E+12	3.298E+13	0.9548	202.6	1.63E+11	2.21
786,432	3,145,728	3.455E+13	1.563E+12	3.298E+13	0.9548	200.7	1.64E+11	2.22
786,432	3,145,728	3.455E+13	1.562E+12	3.298E+13	0.9548	200.8	1.64E+11	2.22

Table 1: SEQUOIA: Raw PHOLD performance data for 1, 2, 4, 8, and 48 rack runs, each rack containing 16,384 cores. All runs were configured with 251,658,240 LPs, 16 circulating events per LP, 8 KPs per MPI task, 10% remote communication fraction (fraction of events sent to random non-self LP), 0.1 `lookahead`, 8 `batch`, 512 `GVT_interval` and 4 MPI tasks per core. The net number of events processed for all runs was exactly 32,984,968,283,642 (*i.e.*, 32 trillion), since the runs were perfectly deterministic. $Cores$ is the total number of Sequoia cores (16 per node) used in each execution; $MPITasks$ is the total number of MPI Tasks used, which in all cases is 4x the number of Cores; $Events_{Total}$ is the total number events executed, including those rolled back; $Events_{RB}$ is the number of events rolled back; $Events_{Net}$ is the number of net events executed, i.e. events committed, not rolled back; $Efficiency$ is $Events_{net}/Events_{Total}$; $Time$ is total wallclock execution time in seconds; $Speed$ is $Events_{net}/Time$ and *Warp Speed* is $log_{10}(Speed) - 9$ All runs were executed between 01/24/2013 and 02/05/2013 under driver Efix #13.

Cores	MPI Tasks	$Events_{Total}$	$Events_{RB}$	$Events_{Net}$	Efficiency	Time (sec)	Speed	Warp Speed
32,768	131,072	3.325E+13	2.679E+11	3.298E+13	0.9919	6377.8	5.17E+09	0.71
32,768	131,072	3.325E+13	2.679E+11	3.298E+13	0.9919	6378.3	5.17E+09	0.71
65,536	262,144	3.339E+13	4.053E+11	3.298E+13	0.9879	2912.4	1.13E+10	1.05
65,536	262,144	3.339E+13	4.053E+11	3.298E+13	0.9879	2912.4	1.13E+10	1.05
131,072	524,288	3.358E+13	5.986E+11	3.298E+13	0.9822	1323.8	2.49E+10	1.40
131,072	524,288	3.358E+13	5.986E+11	3.298E+13	0.9822	1323.0	2.49E+10	1.40
393,216	1,572,864	3.407E+13	1.084E+12	3.298E+13	0.9682	379.7	8.69E+10	1.94
393,216	1,572,864	3.407E+13	1.084E+12	3.298E+13	0.9682	379.7	8.69E+10	1.94
786,432	3,145,728	3.454E+13	1.554E+12	3.298E+13	0.9550	174.7	1.89E+11	2.28
786,432	3,145,728	3.454E+13	1.554E+12	3.298E+13	0.9550	174.7	1.89E+11	2.28
1,572,864	6,291,456	3.521E+13	2.222E+12	3.298E+13	0.9369	82.6	3.99E+11	2.60
1,572,864	6,291,456	3.521E+13	2.222E+12	3.298E+13	0.9369	82.6	3.99E+11	2.60
1,966,080	7,864,320	3.547E+13	2.490E+12	3.298E+13	0.9298	65.4	5.04E+11	2.70
1,966,080	7,864,320	3.548E+13	2.490E+12	3.298E+13	0.9298	65.5	5.04E+11	2.70

Table 2: SEQUOIA: Raw PHOLD performance data for 2, 4, 8, 24, 48, 96 and 120 rack runs. All runs were configured with 251,658,240 LPs, 16 events per LP, 8 KPs per MPI task, 10% remote communications, 0.1 `lookahead`, 8 `batch`, 512 `GVT_interval`, and 4 MPI tasks per core. The net events processed for all runs was 32,984,968,283,642 (*i.e.*, 33 trillion), and was fully deterministic across all job runs. Column headings have the same meaning as in Table 1. Runs were executed between 03/08/2013 and 03/11/2013 under driver Efix #15.

5.2 Sequoia Blue Gene/Q Results

As shown in Tables 1 and 2, we report the raw performance data collected from two distinct series of strong scaling runs on *Sequoia*. In Table 1, we show data performance for runs on 1, 2, 4, 8, and 48 racks for 3 runs each. The reason for the unusual rack scaling (not strictly powers of 2) is that these are sweet spots for the Blue Gene/Q 5-D torus network. In some other configurations the torus becomes only a mesh (*e.g.*, without wraparound) which can increase message latency. However, the configurations above were always using a full torus.

The performance we observed showed very high efficiencies and record breaking event rates. For example, at 1 rack (65,536 MPI tasks), the event rate is just above 2.2 billion and requires just over 4 hours (4hr, 10mins) to execute the 32 trillion events. However, as we increase the core count,

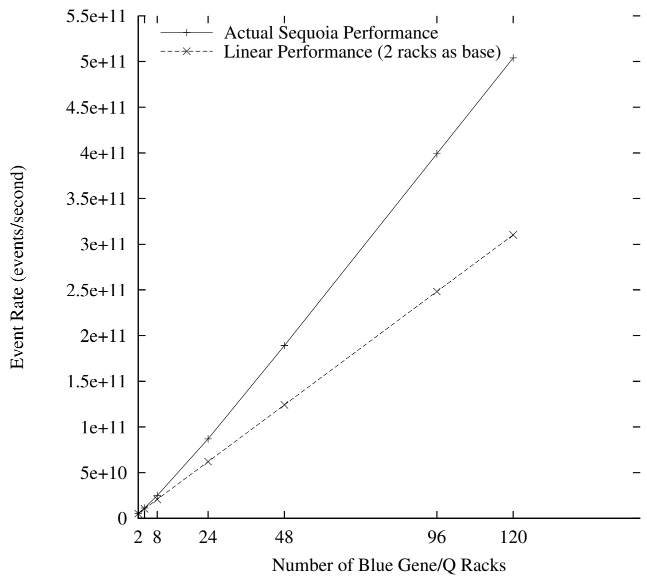

Figure 5: SEQUOIA: PHOLD event rate performance as a function of Blue Gene/Q racks for configurations of 2, 8, 24, 48, 96 and 120 racks. The linear performance curve uses the 2 rack event rate performance as a baseline. Event rate data taken from Table 2.

we observe PHOLD's event rate not only increasing steadily, but doing so at a super-linear pace. At two racks, the event rate has more than doubled to 4.85 billion, and at eight racks the event rate is more than quadrupled again to 22.8 billion. At 48 racks the event rate is 164 billion, and the speedup from 1 to 48 racks is 74x. The overall simulator event efficiencies reflect the super-linear performance with a range of 99% down to 93%. These experiments ran during the time window of `01/24/2013` to `02/05/2013` when *Sequoia's* driver level (as designated by IBM) was Efix #13.

The second series of experiments shown in Table 2 were conducted between `03/08/2013` and `03/11/2013`. At this time, *Sequoia* was running a much updated driver, Efix #15, that enabled MPI to execute on 96 and 120 rack configurations at four tasks per core. Here we observe an unprecedented level of performance. First, Efix #15 version of the driver has improved the lower rack count performance when compared with that shown in Table 1. For example, the 2-rack performance improved from 4.85 to 5.17 billion events/second which is nearly a 7% increase. Even at the higher end, our 48 rack performance has improved by over 6% (from 164.5 to 174.7 billion events/sec). However, the *piéce de résistance* comes at 96 and 120 racks. Here, we observe event rates of 399 billion and 504 billion respectively, with an overall range speedup (2 to 120 racks) of 97x, and both with an event efficiency of nearly 93%. Figure 5, plots the *Sequoia's* event rate performance compared to strictly linear performance scaling using the 2-rack performance as the base.

Now, a critical question is what accounts for the high amount of super-linear speedup across both sets of experiments? The answer lies with the A2 processor's unique

Data struct	Size	Used Count	Subtotal
PE struct	520 bytes	1	0.5 KB
KP struct	104 bytes	8	0.8 KB
LP struct	232 bytes	40	9.3 KB
Network data	8 bytes	7250	58 KB
AVL Tree	40 bytes	7250	290 KB
Event + model	128 bytes	7250	928 KB
Total			**1.287 MB**

Table 3: ROSS Memory Usage at 96 racks for a single MPI rank. Total memory working set per node is just 82 MB.

memory hierarchy. Recall that the memory bandwidth is only 42.6 GB/sec. However, as more Blue Gene/Q racks are used to execute the PHOLD model, more and more of the model code and data fits within the 32 MB L2 cache on each node. This cache operates at 563 GB/sec, which is over 13x faster than main memory [11]. The memory usage of ROSS for the PHOLD model at 96 racks is shown in Table 3. Each MPI rank consumes about 1.3 MB, for a total node memory requirement of just over 82 MB. A similar analysis shows a requirement of only 65 MB at the 120 rack scale. With 32 MB of L2 cache available and a small working set of less than 82 MB, coupled with an advanced data prefetching engine, ROSS is largely executing out of L2 cache at these extremely high core counts.

These super-linear results also underscore the significant capabilities of the Blue Gene/Q network. For the 10% remote communications, each LP is sending to other randomly chosen LPs located far apart within the 5-D torus network. On a 1 million-node, 5-D torus network, remote messages will travel on average 1/4 of the torus circumference in each of the 5 dimensions. Since the circumference is 16 hops, that is a mean of 5x4 = 20 hops. Thus, we are observing substantial super-linear performance in spite of a workload exhibiting a nearly worst-case, globally random communication pattern.

In comparison to previous Blue Gene/P results [2], while some degree of super-linear performance was observed, the high degree of super-linear performance shown here was not observed. Additionally, we note that the previous Blue Gene/P results yielded very high efficiency at 16 LPs per MPI ranks however at 8 LPs per MPI rank with 128K Blue Gene/P cores the amount of parallelism was insufficient and a cascade of rollbacks ensued, lowering the overall event rate performance. Consequently, we believe with our current PHOLD configuration there is additional parallelism to exploit for even greater performance. We observe that if a 240 rack Blue Gene/Q system where available, the current PHOLD model would still have 16 LPs per MPI rank and thus have a good opportunity to scale to an event rate of just over 1 trillion events per second, assuming only a linear increase in performance.

6. HISTORY & DISCUSSION

The PHOLD benchmark has a long history going back to Fujimoto's Time Warp synthetic workload paper [13]. In that original paper, no event rate data was presented, but a sizable 54x speedup on a 64-way BBN Butterfly system was reported. Most papers from that era reported only speedup relative to a sequential execution, and did not single out

Year	Author/Publication	Event Rate	Warp
1990	Fujimoto [13]	N/A	N/A
1995	Fujimoto, et al. [14]	101,000	-3.99
1996	Hao, et al. [18]	95,000	-4.02
2000	Carothers, et al. [4]	375,000	-3.43
2005	Chen, et al. [10]	228,000,000	-0.64
2006	Bauer, et al. [1]	10,000,000	-2.00
2007	Perumalla [30]	214,000,000	-0.67
2008	Holder, et al. [19]	853,000,000	-0.07
2009	Bauer, et al. [2]	12,260,000,000	1.09
2010	Carothers, et al. [5]	3,000,000,000	0.48
2011	Perumalla, et al. [29]	10,000,000,000	1.00
2013	This Paper	504,000,000,000	2.70

Table 4: PHOLD History: Raw PHOLD event rate data from 1995 through present day. Note, the PHOLD configurations across these prior results vary.

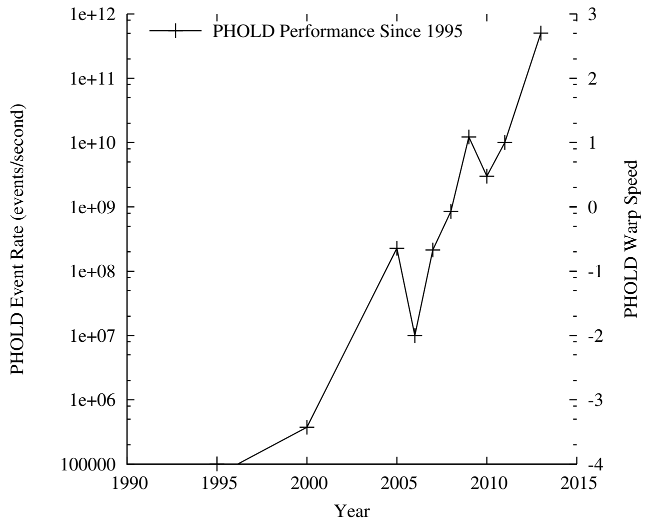

Figure 6: PHOLD History: PHOLD event rate (left side) and *Warp* number (right side) as a function of the year the performance data was published.

total event rate as a key metric. On closer examination of reported performance results, we were able to find a number of event rate results going all the way back to 1995, as shown in Table 4. We hasten to note that we are only looking at PHOLD performance and do not mean to diminish the significance of high performance obtained using other models. For example, in 1994 Greenberg, et al. [17] developed a conservative synchronization algorithm that could leverage the fast SIMD features of the MasPar-1, 16K core multiprocessor system for modeling dynamic channel assignment schemes in wireless telephone networks. They obtained a speedup of 120x over a optimized serial implementation. Similarly in 2003, Fujimoto et al. [15], used 1536 processors from the Lemieux cluster at the Pittsburgh Supercomputing Center on a synthetic 10 million node campus network topology and created a traffic model that produced over 106 million network packet events per second.

Previously published results using PHOLD were usually measured on small clusters consisting typically of less than 32 nodes. It was not until Chen et al. [9] that anyone used a supercomputer of significant power for its day and reported the PHOLD event rate performance. In that case it was also using the Lemieux cluster, consisting of 750, 4-way AlphaServer processors. Of course the historic PHOLD configurations are not directly comparable given the array of different hardware systems and PHOLD configurations. Still, overall, from 1995 to the present, the event rate performance of PHOLD has increased by a factor of around 5,000,000. The exponential performance improvement is shown in Figure 6. Here, both the event rate and *Warp* number are shown on separate y-axises. We observe in both this Figure and Table 4 that the peak *Warp Speed* has increase from -4.0 in 1995 to 2.70 based on the data presented here.

With a massively parallel simulator that is capable of *Warp 2.7* (*e.g.*, 504 billion events-per-second), what *big, hairy, audacious, problems (BHAPs)* might we now contemplate addressing? We submit that the Internet in all its vertical and horizontal architecture has become an integral stratum of what one might call the *human sustainability network (HSN)*. The HSN is really the network of networks on which all of our collective daily lives depend. This includes power networks (*e.g.*, smart grids), fresh water networks, fuel/pipeline networks, transportation networks (air, land and sea vehicles), retail networks (Walmart, Target), financial networks (markets, banks, credit and insurance companies), manufacturing networks, and social networks. Because of the coupling of all of these networks by the Internet (via wired, wireless, cellular and satellite communication systems), we are now seeing how a drought in India can lead to a power outage impacting over 600,000,000 people [33]; how errant code in a trading bot results in a stock market flash crash [36]; and how the housing bubble results in a freeze of credit markets leading to the inability of industrial giants like General Electric to finance their daily operations [34]. All these networks are interconnected at various vertical and horizontal levels and it appears that their level of interdependence will only grow in the future.

One way of looking at the significance of the results we report here is that direct simulations of planetary-scale networks are now, in principle at least, within reach. Our *Sequoia* benchmarks had *only* 251 million LPs, because that was all we could fit in RAM at the scale of 1 *Sequoia* rack. However, at the largest scale of 120 racks, we had enough RAM available for 100 times as many PHOLD LPs, i.e. 25 billion! By comparison, there are *only* 7 billion people on Earth, and there are *only* 4 billion unique IPv4 addresses, and our infrastructure and commerce networks are generally even smaller. We are reaching the point where our simulation capability is limited more by our ability to develop, maintain, and validate models than by our ability to execute them.

7. CONCLUSIONS AND FUTURE WORK

In this paper we have demonstrated a new standard in Time Warp performance by efficiently executing the PHOLD benchmark model using the ROSS simulator running Time Warp with reverse computation on 1,966,080 Blue Gene/Q cores, yielding an unprecedented event rate of 504 billion. We emphasize that these performance results were obtained using a computational workload that generates a nearly worst-case, globally random communication patterns. This continues an exponential growth in peak PDES benchmark speed, which prompts us to propose a new performance metric called *Warp Speed* which grows logarithmically with PHOLD event rate. We are now at *Warp 2.7*.

With these results however, a number of questions still remain. First, our results used an *MPI everywhere* approach and avoided the complexities associated with using a hybrid execution model combining MPI tasks with threads. At first pass, one might think there would be an immediate and significant performance gain. However, our concern with using a hybrid execution model is that it could break the balanced workload on each MPI tasks. As previously noted, each MPI rank does a bit of all parallel simulation activity, from helping to compute GVT to managing MPI messages and polling the network. With threads it is unclear how decompose the simulator work to keep that balance. Adding to the complexity is the fact on Blue Gene/Q systems PAMI (the low level messaging layer) already creates lockless, shared-memory pools across MPI ranks that reside on the same compute node [21]. This functionality appears similar to the sender and receiver pools used by Fujimoto and Panesar [14] to greatly improve cache performance on the KSR shared memory machine. Thus, it is unclear how moving to threads will actually improve performance.

A second open question is the issue of load balancing at this scale. While a great deal of research has been done in the area of dynamic and static load distribution for parallel discrete event simulation, no research has been done for irregular systems with greater than 100,000 cores, much less millions of cores. This is a difficult open question.

Third, there is an open issue regarding efficient checkpointing as part of an overall failure recovery process for Time Warp simulation executing at this scale. While it might seem that because a Time Warp system supports a rollback and recovery mechanism, it is not immediately obvious how to adapt the reverse computation functionality for use in situations where failures can occur on any processor cycle or packet transfer instead of at well-defined event boundaries.

Finally, these results motivate us and hopefully others in the parallel simulation community to consider a wide range of new application domains that could benefit from this level of performance such as multi-network systems like the *Human Sustainability Network*, medical applications like discrete human brain models, and new computational methods like hybrid discrete-continuum models, to name but just a few.

8. ACKNOWLEDGMENTS

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344, and Argonne National Laboratory ("Argonne") under Contract No. DE-AC02-06CH11357. This research used the *Sequoia* IBM Blue Gene/Q computing resources at Lawrence Livermore National Laboratory, and additional Blue Gene/Q computing resources were provided by the Computational Center for Nanotechnology Innovations (CCNI) located at Rensselaer Polytechnic Institute. Part of the CCNI's Blue Gene/Q was acquired using funds from the NSF MRI Program, Award #1126125. Prof. Carothers' time was supported by AFRL Award FA8750-11-2-0065 and DOE Award DE-SC0004875.

9. REFERENCES

[1] D. W. Bauer and C. D. Carothers. Eliminating remote message passing in optimistic simulation. In *WSC '06: Proceedings of the 38th conference on Winter simulation*. Winter Simulation Conference, December 2006.

[2] D. W. Bauer Jr., C. D. Carothers, and A. Holder. Scalable time warp on blue gene supercomputers. In *Proceedings of the 2009 ACM/IEEE/SCS 23rd Workshop on Principles of Advanced and Distributed Simulation*, pages 35–44, Washington, DC, USA, 2009. IEEE Computer Society.

[3] P. Beckman, K. Iskra, K. Yoshii, S. Coghlan, and A. Nataraj. Benchmarking the Effects of Operating System Interference on Extreme-Scale Parallel Machines. *Cluster Comput.*, 11:3–16, 2008.

[4] C. D. Carothers, D. Bauer, and S. Pearce. Ross: A high-performance, low-memory, modular time warp system. *Journal of Parallel and Distributed Computing*, 62(11):1648 – 1669, 2002.

[5] C. D. Carothers and K. S. Perumalla. On deciding between conservative and optimistic approaches on massively parallel platforms. In *Winter Simulation Conference'10*, pages 678–687, 2010.

[6] C. D. Carothers, K. S. Perumalla, and R. M. Fujimoto. Efficient optimistic parallel simulations using reverse computation. *ACM Transactions on Modeling and Computer Simulation*, 9(3):224–253, 1999.

[7] D. Chen, N. Eisley, P. Heidelberger, S. Kumar, A. Mamidala, F. Petrini, R. Senger, Y. Sugawara, R. Walkup, B. Steinmacher-Burow, A. Choudhury, Y. Sabharwal, S. Singhal, and J. J. Parker. Looking under the hood of the ibm blue gene/q network. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, pages 69:1–69:12, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[8] D. Chen, N. A. Eisley, P. Heidelberger, R. M. Senger, Y. Sugawara, S. Kumar, V. Salapura, D. L. Satterfield, B. Steinmacher-Burow, and J. J. Parker. The ibm blue gene/q interconnection network and message unit. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 26:1–26:10, New York, NY, USA, 2011. ACM.

[9] G. Chen and B. K. Szymanski. Dsim: scaling time warp to 1,033 processors. In *WSC '05: Proceedings of the 37th conference on Winter simulation*, pages 346–355. Winter Simulation Conference, 2005.

[10] G. Chen and B. K. Szymanski. Time quantum GVT: A scalable computation of the global virtual time in parallel discrete event simulations. *Scalable Computing: Practice and Experience: Scientific International Journal for Parallel and Distributed Computing*, pages 425–446, 2007.

[11] G. Chiu, P. Coteus, and R. Wisniewski. Blue gene/q overview and update. `http://www.alcf.anl.gov/sites/www.alcf.anl.gov/files/IBM\_BGQ\_Architecture\_0.pdf`, 2011.

[12] C. C. Foster. Information retrieval: information storage and retrieval using avl trees. In *Proceedings of the 1965 20th national conference*, ACM '65, pages 192–205, New York, NY, USA, 1965. ACM.

[13] R. M. Fujimoto. Performance of time warp under synthetic workloads, January 1990.

[14] R. M. Fujimoto and K. S. Panesar. Buffer management in shared-memory time warp systems. In *Proceedings of the ninth workshop on Parallel and distributed simulation*, PADS '95, pages 149–156, Washington, DC, USA, 1995. IEEE Computer Society.

[15] R. M. Fujimoto, K. Perumalla, A. Park, H. Wu, M. H. Ammar, and G. F. Riley. Large-scale network simulation âĂŞ how big? how fast. In *In Symposium on Modeling, Analysis and Simulation of Computer Telecommunication Systems (MASCOTS*, 2003.

[16] E. Gonsiorowski, C. Carothers, and C. Tropper. Modeling large scale circuits using massively parallel discrete-event simulation. In *Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2012 IEEE 20th International Symposium on*, pages 127–133, Aug.

[17] A. G. Greenberg, B. D. Lubachevsky, P. E. Wright, and D. M. Nicol. Efficient massively parallel simulation of dynamic channel assignment schemes for wireless cellular communications. In *Workshop on Parallel and Distributed Simulation*, pages 187–194, 1994.

[18] F. Hao, K. Wilson, R. Fujimoto, and E. Zegura. Logical process size in parallel simulations. In *Proceedings of the 28th conference on Winter simulation*, WSC '96, pages 645–652, Washington, DC, USA, 1996. IEEE Computer Society.

[19] A. Holder and C. D. Carothers. Analysis of time warp on a 32,768 processor ibm blue gene/l supercomputer. In *2008 Proceedings European Modeling and Simulation Symposium (EMSS)*, 2008.

[20] D. R. Jefferson. Virtual time. *ACM Trans. Program. Lang. Syst.*, 7(3):404–425, 1985.

[21] S. Kumar, A. R. Mamidala, D. A. Faraj, B. Smith, M. Blocksome, B. Cernohous, D. Miller, J. Parker, J. Ratterman, P. Heidelberger, D. Chen, and B. Steinmacher-Burrow. Pami: A parallel active message interface for the blue gene/q supercomputer. *Parallel and Distributed Processing Symposium, International*, 0:763–773, 2012.

[22] P. L'Ecuyer and T. H. Andres. A random number generator based on the combination of four lcgs. *Math. Comput. Simul.*, 44(1):99–107, 1997.

[23] N. Liu, C. Carothers, J. Cope, P. Carns, R. Ross, A. Crume, and C. Maltzahn. Modeling a leadership-scale storage system. In *Proceedings of the 9th international conference on Parallel Processing and Applied Mathematics - Volume Part I*, PPAM'11, pages 10–19, Berlin, Heidelberg, 2012. Springer-Verlag.

[24] N. Liu and C. D. Carothers. Modeling billion-node torus networks using massively parallel discrete-event simulation. In *Proceedings of the 2011 IEEE Workshop on Principles of Advanced and Distributed Simulation*, PADS '11, pages 1–8, Washington, DC, USA, 2011. IEEE Computer Society.

[25] N. Liu, J. Cope, P. Carns, C. D. Carothers, R. Ross, G. Grider, A. Crume, and C. Maltzahn. On the role of burst buffers in leadership-class storage systems. In *In Proceedings of the 28th IEEE Conference on Mass Storage Systems and Technologies (MSST 2012)*. IEEE, 2012.

[26] B. D. Lubachevsky, A. Shwartz, and A. Weiss. An analysis of rollback-based simulation. *ACM Transactions on Modeling and Computer Simulation*, 1(2):154–193, 1991.

[27] M. Mubarak, C. D. Carothers, R. Ross, and P. Carns. Modeling a million-node dragonfly network using massively parallel discrete event simulation. In *3rd International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS12) held as part of SC12*, 2012.

[28] D. M. Nicol and X. Liu. The dark side of risk (what your mother never told you about time warp). In *PADS '97: Proceedings of the eleventh workshop on Parallel and distributed simulation*, pages 188–195, Washington, DC, USA, 1997. IEEE Computer Society.

[29] K. Perumalla, A. Park, and V. Tipparaju. Gvt algorithms and discrete event dynamics on 129k+ processor cores. In *High Performance Computing (HiPC), 2011 18th International Conference on*, pages 1–11, 2011.

[30] K. S. Perumalla. Scaling time warp-based discrete event execution to 104 processors on a blue gene supercomputer. In *CF '07: Proceedings of the 4th international conference on Computing Frontiers*, pages 69–76, New York, NY, USA, 2007. ACM.

[31] K. S. Perumalla. $\mu\pi$: A scalable and transparent system for simulation mpi programs. In *In Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, 2010.

[32] K. S. Perumalla and S. K. Seal. Reversible parallel discrete-event execution of large-scale epidemic outbreak models. In *In Proceedings of the 24th Workshop on Principles of Advanced and Distributed Simulation*, 2010.

[33] J. Romero. Energy-wise blog: Lack of rain a leading cause of indian grid collapse. *IEEE Spectrum*, July 2012.

[34] P. Schweizer. *Throw Them All Out.* Houghton Mifflin Harcount Publishing Company, New York, 2011.

[35] D. D. Sleator and R. E. Tarjan. Self-adjusting binary search trees. *J. ACM*, 32(3):652–686, July 1985.

[36] E. Ullman. “errant code? it's not just a bug”, new york times, the opinion pages. http://www.nytimes.com/2012/08/09/opinion/after-knight-capital-new-code-for-trades.html, August 8th, 2012.

[37] J. Vaucher and P. Duval. A comparison of simulation event list algorithms. *Communications of the ACM*, 18(4):223–230, 1975.

[38] G. Yaun, C. D. Carothers, and S. Kalyanaraman. Large-scale tcp models using optimistic parallel simulation. In *Proceedings of the seventeenth workshop on Parallel and distributed simulation*, PADS '03, pages 153–, Washington, DC, USA, 2003. IEEE Computer Society.