Proceedings of the 2011 Winter Simulation Conference
S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, eds.

INVESTIGATING THE MEMORY CHARACTERISTICS OF A MASSIVELY PARALLEL TIME WARP KERNEL

Akintayo Holder
Christopher D. Carothers

Computer Science Department
Rensselaer Polytechnic Institute
Troy, NY 12180, USA

ABSTRACT

Recently, Time Warp has shown that it achieves good strong scaling to hundreds of thousands of processors on modern supercomputer systems. These results were achieved on the Cray and IBM Blue Gene supercomputing platforms. In this paper, we investigate the ROSS Time Warp cache memory performance on (i) a commodity shared-memory desktop system based on the Intel E5504 processor and (ii) the IBM Blue Gene/L when configured to run over the standard Message Passing Interface (MPI) library.

1 INTRODUCTION

Time Warp (Jefferson 1985) is a speculative synchronization protocol for parallel discrete-event simulation. A Time Warp simulator processes events in local time stamp order and when it detects out of order event processing, it will rollback the simulator's state and return the simulation to an earlier, correct state with respect to virtual time. Rensselaer's Optimistic Simulation System (ROSS) (Holder and Carothers 2008, Bauer, Carothers, and Holder 2009) demonstrated that a Time Warp simulation kernel could scale on the IBM Blue Gene platform. They suggested that the robust performance was due to the problem set shrinking as the processor count increases. The smaller problem set fits into cache, the faster memory access mitigates the effects of processor starvation, this explanation for super linearity has been suggested by others (Gustafson 1990, Sutter 2008, Beane 2004). This cache study will examine how efficiently Time Warp systems like ROSS exploit the platform's memory hierarchy.

2 AN EFFICIENT APPROACH TO LARGE SCALE TIME WARP

The efficient and scalable execution of ROSS (Bauer, Carothers, and Holder 2009) is enabled by a number of software optimizations and algorithm choices. Of these optimizations, it appears that the asynchronous management of remote events yields robust performance on modern supercomputing hardware while running models that generated a large percentage of remote events. This includes the use of hash tables to speed remote event lookup for anti-message processing and flow control based on direct network event memory management to prevent senders overwhelming receivers. In ROSS, each event scheduler (one per processor) maintains a set of event hash tables. These hash tables provide the pointer that corresponds to a particular event identifier. During remote event cancellation, the processor element will receive an anti-message which contains the event identifier of the event that is about to be canceled. The event scheduler uses the hash tables to find the event, and then it begins the cancellation of the event. Each processor element maintains an array of hash tables, one for each other processor element in the system. To schedule or cancel a remote event, ROSS would create the event, call `MPI_Isend` to make an asynchronous send and return to event processing. Later, the scheduler will call `MPI_Testsome` to check if the send has completed. When the send completes, the kernel checks if the event has been canceled, and sends a cancel

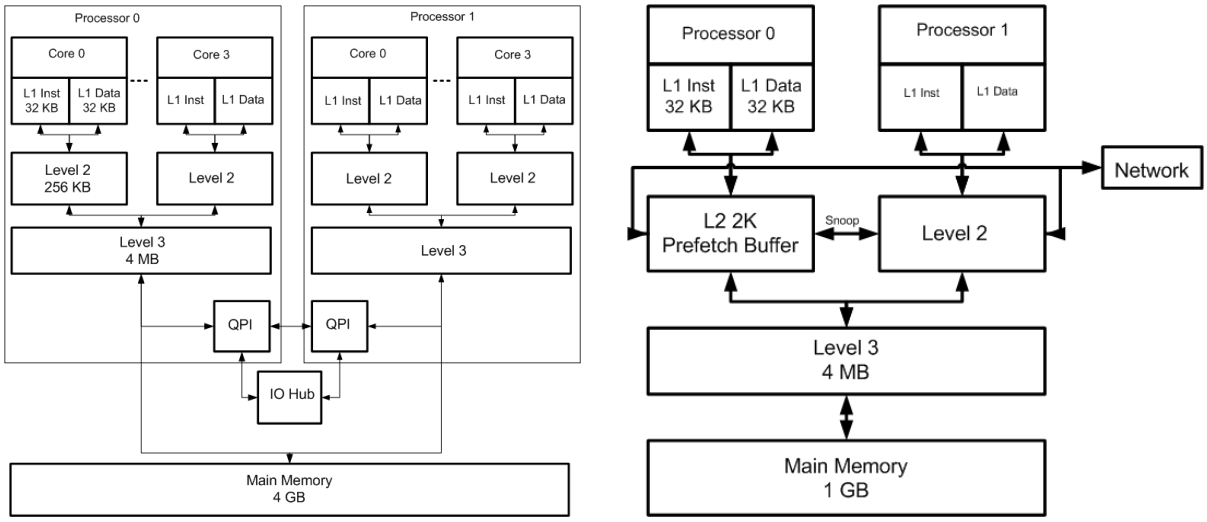

Figure 1: Memory Hierarchy of the Xeon E5504 Processor (left) and IBM Blue Gene/L Compute Node (right).

if needed. At the receiver, an incoming event or anti-message would be retrieved using an asynchronous receive (`MPI_Irecv`) which allows the processor to continue event processing. Like the sender, the receiver periodically checks for completed receives, then it copies the uncanceled receives into its event ready queue. It is of key importance that the scheduler "post" a significant number (tunable parameter) of `MPI_Irecv` operations prior to posting any `MPI_Isends` operations to insure error free MPI execution (e.g., deadlock, message overflow).

The event scheduler will iterate `batch` times through the primary event processing loop before polling the MPI network queues for completed asynchronous MPI receives and sends. When it encounters a completed send, it reclaims the event buffer and places it back on the event free-list. If there is a completed receive, the kernel will process any rollbacks caused by that event, enqueue it for LP processing and then post another asynchronous MPI receive in its place.

Every `batch * gvt_interval` times through the event processing loop a new Global Virtual Time (GVT) is computed. Here, the scheduler accounts for all remote messages. This is accomplished by using the MPI reduction operation on the number of sent and receive message counts, since the last GVT computation. Once the number of sent and received messages are equal, then each scheduler will compute its own local virtual time (LVT) and a MPI all-reduce operation is used to compute the minimum of LVTs which becomes the new GVT. Here, each scheduler receives the new GVT as part of the all-reduce operation. With the calculation of the new GVT, event memory with a timestamp less than GVT is reclaimed.

3 A METHOD FOR STUDYING CACHE BEHAVIOR

This study was conducted using the Performance Application Programming Interface (PAPI), which provides a source level interface to query hardware performance counters (Browne et al. 2000). This software interface allows the user to measure a specific section of an application for a particular or sampled set of hardware performance counters which include detailed performance counts for the cache hierarchy. With this interface, each processor's counters are queried and then metrics are calcuated. In this study, average of the queried/computed values is used.

The hardware platforms used in this study are an HP Z800 workstation and the IBM Blue Gene/L supercomputer. The HP Z800 is a shared memory workstation with 4GB of memory and two quad-core Intel Xeon E5504 processors shown in Figure 1 (left panel). Each core has a 32KB Level 1 instruction cache, a 32KB Level 1 data cache and a unified 256 KB Level 2 data cache. The unified 4MB Level 3

cache is shared by all four cores (Intel 2009a). The Intel Xeon E5504 has a two level Translation Lookaside Buffer (TLB) (Intel 2009b) which caches virtual address to physical address translations.

The IBM Blue Gene/L compute node has two 700 MHz PowerPC 440 processors where each processor has a 32 KB Level 1 instruction cache, a 32 KB Level 1 data cache and a 2KB unified Level 2 cache which operates as a prefetch buffer for the Level 3 cache and remain coherent by using snoopy cache protocol between the two cores. Finally, both cores share a 4 MB unified Level 3 cache and 1 GB of main memory(Figure 1 right panel). The Blue Gene/L nodes offer two modes of operation. The first is *virtual node mode* which allows two MPI tasks to be assigned, one to each core. The second is *co-processor mode* which assigns only one MPI tasks to core 0 within the node and core 1 is used to offload the MPI message processing. For all experiments here, we use the *co-processor* mode to gain more memory per core for ROSS experiments. This is different from previous studies (Holder and Carothers 2008, Bauer, Carothers, and Holder 2009) which used *virtual node* mode.

For PAPI, the Blue Gene/L's PowerPC 440 provides fifteen hardware performance counters, while the Intel E5504 provides thirty-four. Because of the non-uniformity in hardware performance counters across processor architecture, applications developers must select which of the available counters are most useful. For our cache memory analysis, we used the follow set of PAPI counters:

$$\texttt{L1\_MISS\_RATE} = \texttt{PAPI\_L2\_TCA}/(\texttt{PAPI\_L1\_DCA} + \texttt{PAPI\_L1\_ICA}) \tag{1}$$

$$\texttt{L2\_MISS\_RATE} = \texttt{PAPI\_L2\_TCM}/\texttt{PAPI\_L2\_TCA} \tag{2}$$

`PAPI_L1_DCA` and `PAPI_L1_ICA`, provided the number of accesses to the Level 1 data cache and the Level 1 instruction cache, respectively. `PAPI_L2_TCA`, provided the total number of accesses to the Level 2 cache and `PAPI_L2_TCM` provides the number of misses from the Level 2 cache. Equations 1 and 2 show how to calculate the miss rate for the Level 1 and Level 2 caches, respectively.

$$\texttt{L2\_DATA\_MISS\_RATE} = 1-(\texttt{PAPI\_L2\_DCH}/\texttt{PAPI\_L2\_DCA}) \tag{3}$$

$$\texttt{L3\_MISS\_RATE} = \texttt{PAPI\_L3\_TCM}/(\texttt{PAPI\_L3\_TCM}+\texttt{PAPI\_L3\_TCH}) \tag{4}$$

Because the IBM Blue Gene/L does not provide the performance counters for the Level 1 cache or Level 2 cache, Equations 1 and 2 are not computable. The IBM Blue Gene/L does have counters for the Level 2 data cache/prefetch buffer; `PAPI_L2_DCH` returns the number of hits from the Level 2 data cache, while `PAPI_L2_DCA` provides the total number of access to the Level 2 data cache. Equation 3 shows how the Level 2 data cache miss rate is calculated.

The IBM Blue Gene/L provides Level 3 cache counters, but these are not available on the HP Z800. `PAPI_L3_TCM` returns the number of Level 3 cache misses and `PAPI_L3_TCH` the hits, they are used to calculate the Level 3 miss rate (Equation 4). Other researchers (Mindlin et al. 2003) (Davis et al. 2004) (Brunheroto et al. 2005) addressed the absence of performance counters or other data collection issues by modeling the behavior of the IBM Blue Gene/L.

$$\texttt{TLB\_MISS\_RATIO} = \texttt{PAPI\_TLB\_TL}/\texttt{PAPI\_TOT\_CYC} \tag{5}$$

`PAPI_TLB_TL` returns the number of TLB misses, and `PAPI_TOT_CYC`, the number of instruction cycles used. The IBM Blue Gene does not have TLB misses because the mapping of virtual to physical addresses remain static during execution (Shmueli et al. 2008).

4 ANALYZING TIME WARP

The PHOLD benchmark is used in this performance study and is configured with 1,048,576 LPs, 10 events per LP and 100s of simulation time. This same configuration has been used in a number of previous

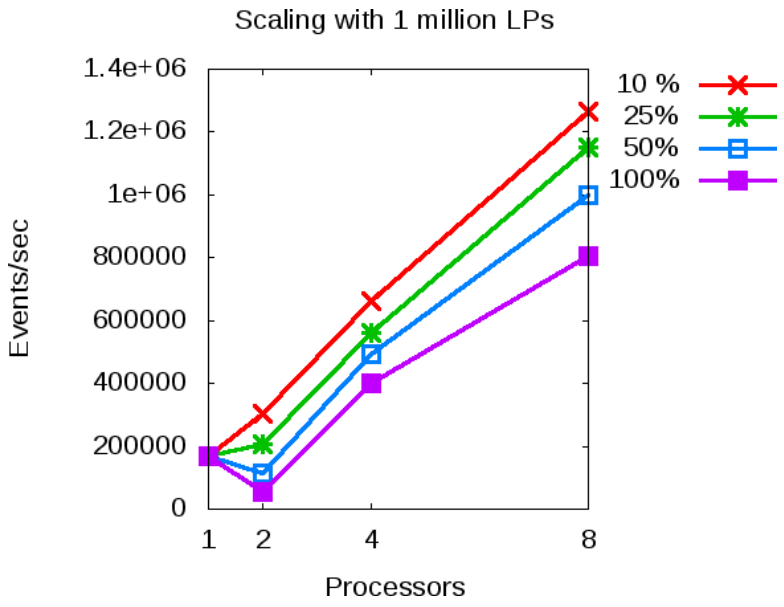

Figure 2: Performance of the PHOLD model, with varying amount of remote events, using ROSS on the Intel Xeon E5504: Committed events per second as function of the number of processors.

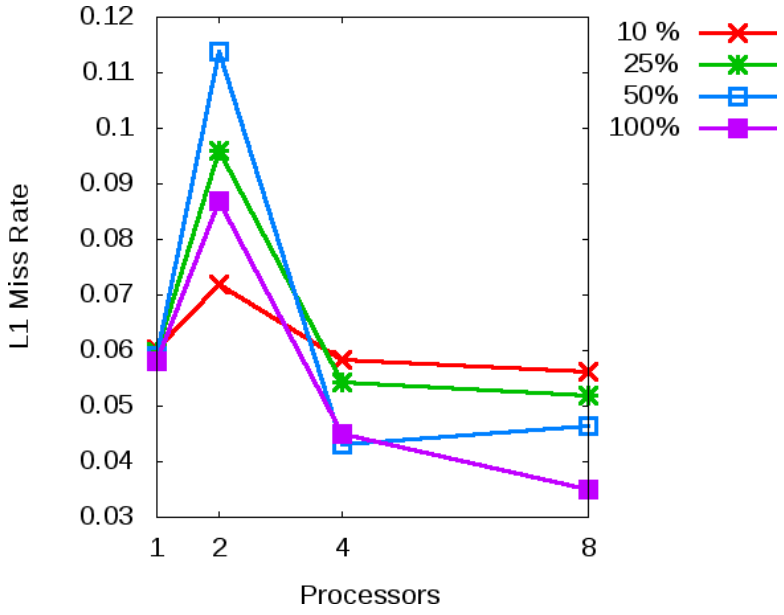

Figure 3: Performance of the PHOLD model, with varying amount of remote events, using ROSS on the Intel Xeon E5504: Level 1 Cache Miss Rate (Eq. 1) as a function of the number of processors.

performance studies (Perumalla 2007, Holder and Carothers 2008, Bauer, Carothers, and Holder 2009). As suggested, *batch* was set to 4, *gvt_interval* to 1024 and there were 16 KPs per PE (Bauer, Carothers, and Holder 2009).

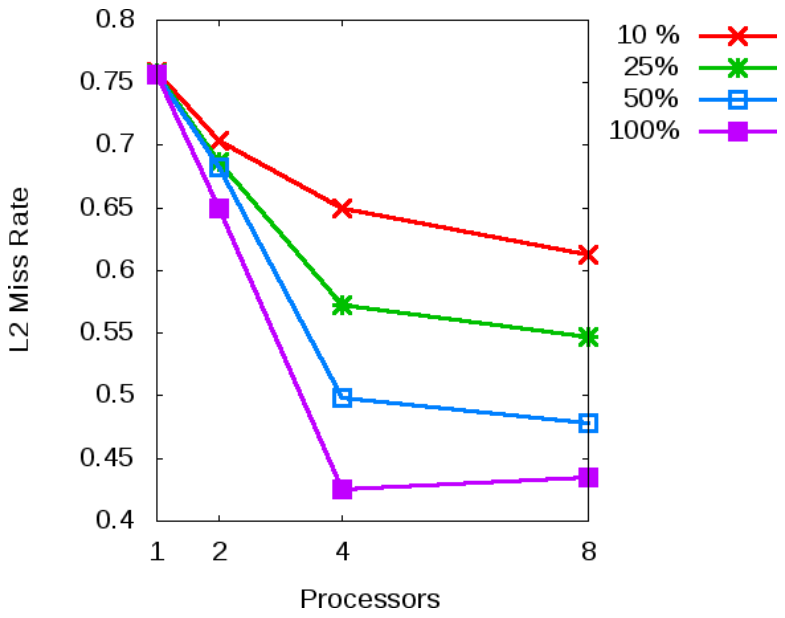

Figure 4: Performance of the PHOLD model, with varying amount of remote events, using ROSS on the Intel Xeon E5504: Level 2 Cache Miss Rate (Eq. 2) as a function of the number of processors.

4.1 Cache Usage Analysis: Time Warp on Shared Memory Workstation

Because of the high event density for the million LP PHOLD simulation (e.g., 10 million events will be processed on average per unit of simulation time), the simulation time was cut to 10 seconds on the HP Z800 workstation which means that each configuration will process about 100 million events. Figure 2 shows how ROSS's event-rate scales as the percentage of remote events in the simulation is varied for a given number of cores. In this study, the PHOLD model was configured with different remote event populations, ranging from 10 to 100 percent. The Linux operating system tries to balance the MPI/ROSS tasks across the processors, therefore adding a second core also adds a second processor which doubles the available L1 and L2 cache resources however L3 cache is shared within a block of 4 cores.

Figure 3 shows the Level 1 cache miss rate (Equation 1) of ROSS as the number of cores increases, while Figure 4 shows the Level 2 miss rate (Equation 2). Figure 5 shows how TLB misses are affected by varying the number of cores used and the percentage of remote events in the simulation, the TLB miss ratio is based on Equation 5.

When analyzing the performance of a parallel application, it is important to consider overall system performance and the performance of a single processor. Less than linear event-rate scaling indicates that per processor performance has decreased. The performance of ROSS depends on local event processing and remote event scheduling, but remote event scheduling is highly dependent on the MPI library.

The Level 1 cache miss rate increases with the second processor, then decreases with the additional cores. The TLB miss ratio also displayed similar behavior. This increase was due to adding a second processor, which indicates a conflict in the behavior of the local event processing and the remote event scheduling / message passing. With one core, ROSS ran in sequential mode and Level 1 cache and the TLB miss rates was moderate. Adding a second processor involved the MPI library and the optimistic parallel scheduler. As ROSS scaled from two to eight cores, the Level 1 cache miss rate and the TLB miss ratio fell in response to increasing cache and TLB capacity. The 10 percent remote event configuration fell the least, and eventually displayed the highest L1 cache miss rate and TLB miss ratio. This indicates that the decrease in miss rates was due to remote scheduling, as the configurations that communicated more, demonstrated lower overall miss rates. Level 2 cache behavior was similar. Here, the 10 percent

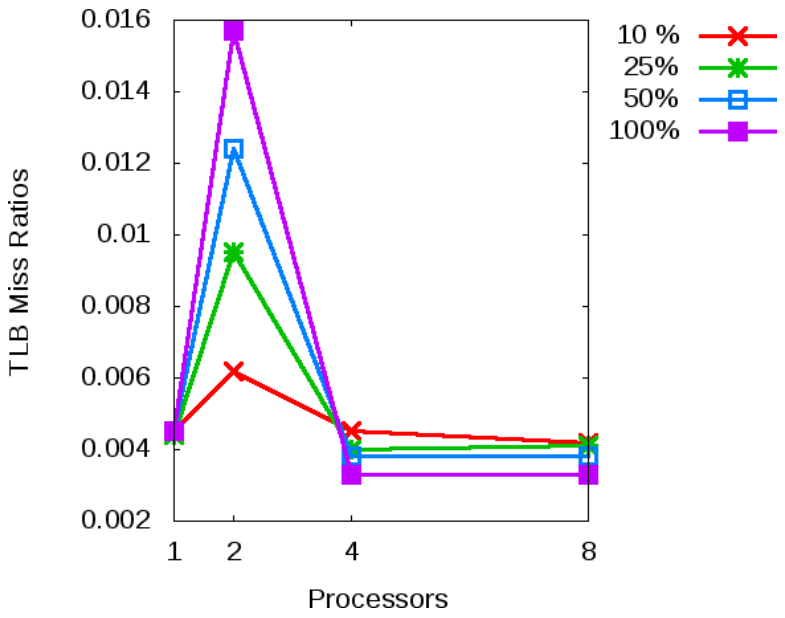

Figure 5: Performance of the PHOLD model, with varying amount of remote events, using ROSS on the Intel Xeon E5504: TLB Miss Rate (Eq. 5) as a function of the number of processors.

configuration showed the smallest improvements with increased processor count. The low Level 1 cache miss rates, and the high Level 2 cache miss rates may be due to (i) any infrequent Level 1 misses also being Level 2 misses and (ii) locality being restricted to small sections of the program. The latter is likely, because event memory usage is nearly random, but the network memory handling appears to be predictable (i.e., FIFO). This would also explain why the sequential code shows a higher Level 2 cache miss rate.

4.2 Cache Usage Analysis: Time Warp on the IBM Blue Gene/L

This series of experiments where conducted on an IBM Blue Gene/L supercomputer using PHOLD as configured in Section 4.1, except that these experiments used a 100s simulation time resulting in about 1 billion events being processed per run. The IBM Blue Gene was configured to run in co-processor mode, where one processor on the compute node was reserved for MPI message processing tasks. The performance counters are shared between the two running processors, therefore events triggered by the dedicated I/O processor are also included in this cache study. Figure 6 shows the committed event-rate of ROSS, while running PHOLD on the IBM Blue Gene/L. Figure 7 shows the Level 2 data cache miss rate (Equation 3), and Figure 8 shows the Level 3 cache miss rate (Equation 4).

For the Blue Gene/L, it appears that the availability of events will limit scaling when the problem size is held constant. It is observed that eventually there is insufficient event processing load to keep to all the processors busy. The IBM Blue Gene/L does not provide the counters needed to track processor idling, therefore an alternative approach was used. Here, a weak scaling analysis is used where the per processor workload is kept constant. This approach should eliminate the effect of starvation due to limited work. Weak scaling also prevents the problem set from shrinking as the number of cores increase. In the weak scaling experiment, the model is configured with 1024 LPs per processor and this ratio is kept fixed, so that the number of LPs doubles when the number of processors doubles. The other model parameters are the same as used in the prior PHOLD investigation.

Figure 9 shows the weak scaling performance of PHOLD on the IBM Blue Gene/L, Figure 10 shows the Level 2 data cache miss rate(Equation 3) and Figure 11 shows the Level 3 cache miss rate (Equation 4).

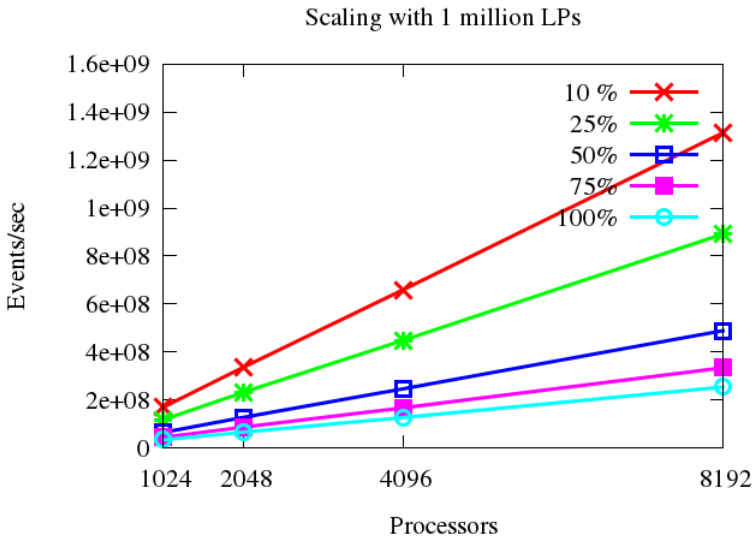

Figure 6: Performance of the PHOLD model, with varying amount of remote events, using ROSS on the Blue Gene/L: Committed events per second as function of the number of processors.

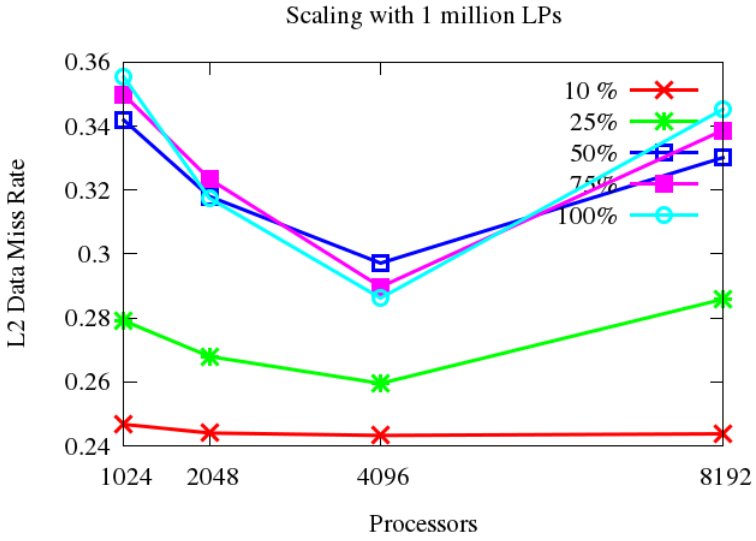

Figure 7: Performance of the PHOLD model, with varying amount of remote events, using ROSS on the Blue Gene/L: Level 2 Cache Miss Rate (Eq. 2) as function of the number of processors.

With the exception of the 10 percent remote event configuration, the per core event-rate decreased as the number of cores increased. This phenomenon was more pronounced with the weak scaling experiments. The behavior of the ROSS kernel, the MPI library and the dedicated I/O processor must be acknowledged, when considering the performance of ROSS on the IBM Blue Gene/L. It is probable that the MPI library and the I/O processor will exhibit better cache behavior than the ROSS kernel. This should result in better cache miss rates, but lower per processor event-rates, if remote communication dominates over event computation.

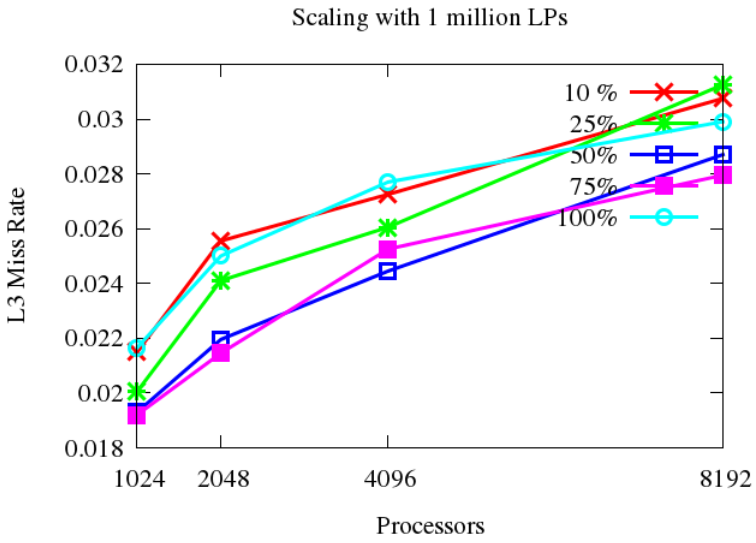

Figure 8: Performance of the PHOLD model, with varying amount of remote events, using ROSS on the Blue Gene/L: Level 3 Cache Miss Rate (Eq. 4) as function of the number of processors.

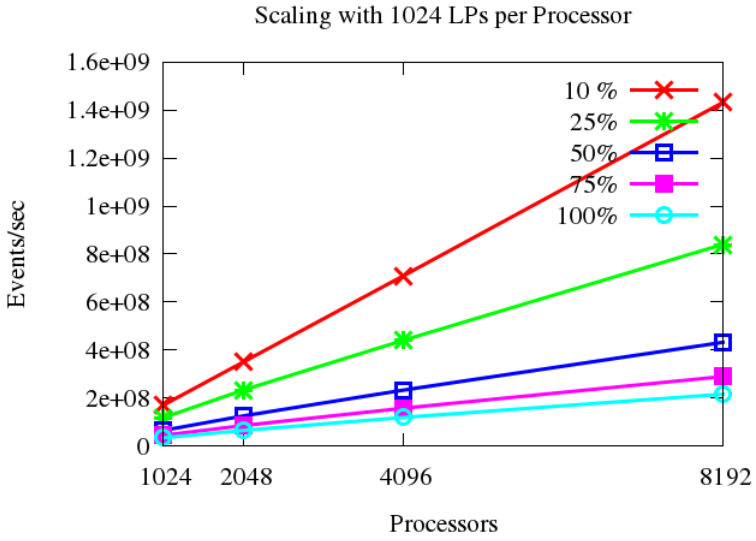

Figure 9: Weak Scalability of PHOLD using ROSS on the IBM Blue Gene/L. The event-rate of PHOLD when configured with 10 events per LP as a function of processor count.

The Level 2 data cache miss rate shows the higher remote event populations behave differently. With 50, 75 or 100 percent remote events, there was a drop in miss rate from 1024 to 4096 processors, followed by an increase from 4096 to 8192. This occurred with both strong and weak scaling, but the steeper climb with strong scaling indicates that it is related to the size of ROSS's communication infrastructure. The cost of PHOLD's communication grows with processor count, and strong scaling sends fewer messages to each processor, reducing data locality. It would appear that the Level 2 data cache/prefetch buffer miss rate improves until it is too big to fit, at which point the number of misses increase. The behavior of the 10

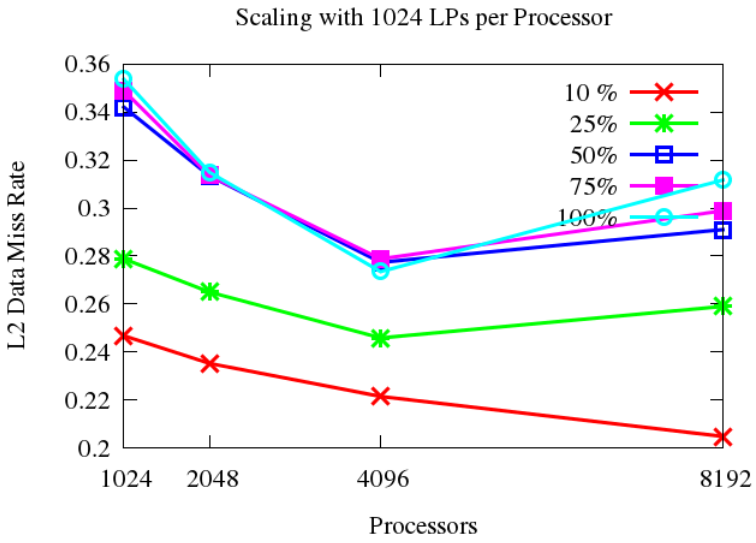

Figure 10: Impact of remote events on the behavior of PHOLD using ROSS on the IBM Blue Gene/L for weak scaling runs: Level 2 data cache miss rate (Eq. 3), running PHOLD configured with 10 events per LP, as a function of processor count.

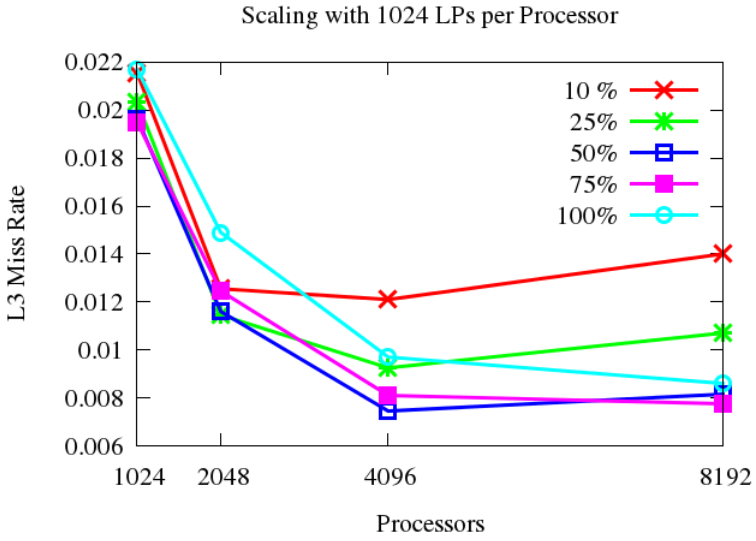

Figure 11: Impact of remote events on the behavior of PHOLD using ROSS on the IBM Blue Gene/L for weak scaling runs. Level 3 cache miss rate (Eq. 4), running PHOLD configured with 10 events per LP, as a function of processor count.

percent remote event configuration, shows that it gets little benefit from the locality in the communication code, and it does not communicate enough to suffer due to the ever changing destinations.

Level 3 cache miss behavior is more interesting, and contradicts the expected result as it appears that it increased (not decreased) with processor count under strong scaling, and decreased under weak scaling. Also, the remote event ratio was not responsible for any significant trends. The divergence in

the, relatively low, Level 3 cache miss rates indicates that local event behavior is responsible, probably due to the random nature event processing of the Time Warp kernel. That is the memory addresses for events become increasingly uncorrelated with event timestamps leading to a growth in cache misses. As the processor count increases, the problem almost fits into the Level 3 cache under strong scaling. However, weak scaling shows more remote event reuse and therefore more locality.

5 RELATED WORK

There has been other efforts to build large-scale optimistic discrete event simulation kernels such as μsik which examined the performance of Time Warp and other approaches to large-scale discrete event simulation (Perumalla 2007). While running PHOLD with a 10 percent remote event population, μsik achieved 214 million events per second on 8192 IBM Blue Gene/L processors with the optimistic approach, and 530 million events per second on 16,384 IBM Blue Gene/L processors with the conservative approach. The issues with the SCATTER model highlights the problem of building large-scale models. ROSS (Holder and Carothers 2008) achieved 853 million events per second on 16,384 IBM Blue Gene/L processors, while running the PHOLD benchmark with a 10 percent remote event population. ROSS (Bauer, Carothers, and Holder 2009) improved and achieved 12.26 billion events per second on 65,536 IBM Blue Gene/P processors with the same PHOLD simulation. ROSS also scaled linearly with a 100 percent remote event PHOLD simulation. DSIM on an Alpha server (Chen and Szymanski 2005) achieved 228 million events per second while using 1024 processors. It was forced to make allowances for the Quadrics network fabric and so the GVT algorithm reserved a number of processors and PHOLD scheduled events among the nearest neighbors. Additionally, In (Carothers and Perumalla 2010), ROSS was used a performance study to understand the performance of a global synchronization-based conservative approach compared to an purely optimistic approach when executed on massively parallel computing hardware (e.g., Blue Gene/L).

We note here that event-rate is a good measure parallel simulator performance when executing benchmarks like PHOLD but may not be for real applications. To date, the most scalable Time Warp execution for a real application (e.g., TLM) is now on nearly 128K cores (Seal and Perumalla 2012). Here, they report a weak scaling speedup in excess over 100,000 when using 127,500 Cray XT5 cores.

Additionally, There have been other efforts to understand how system architecture impacts the performance of Time Warp kernels. Carothers, Perumalla, and Fujimoto (1999) investigates the performance of Georgia Tech Time Warp (GTW) on an SGI Origin 2000. They observed that coherence protocols caused false sharing yield higher caches and TLB miss rates. Using reverse computation to implement rollback seemed to eliminate this problem.

6 CONCLUSIONS

This study shows that when ROSS runs the PHOLD model on the IBM Blue Gene/L with more than a thousand processors, the model state and events appear to fit within the per-processor 4MB Level 3 cache. This phenomenon occurs under both strong and weak scaling, implying that we can run bigger models without a significant change in cache behavior. Overall, a two to three percent Level 3 cache miss rate is observed, which seemed reasonable for a speculative application like Time Warp. The Level 2 prefetch buffer miss rate was higher, and it indicates that remote event scheduling dominates this cache level's miss rate statistics. The performance of ROSS on the HP Z800 was different from the IBM Blue Gene/L. Here, the 10% remote event configuration demonstrated the best event-rate but the highest miss rate. This indicates that the MPI library shows better cache behavior than the ROSS kernel. This is to be expected since the MPI implementation would take advantage of shared memory when it is available as well as use memory in a more predictable fashion.

For future work, we plan to conduct further studies to better isolate the impact of communication and the dedicated I/O processor. The IBM Blue Gene/L does have counters that monitor its communication,

so a next step would be to investigate how cache behavior correlates with network activity. Additionally, we would like to understand the cache memory behavior for conservative synchronization protocols.

ACKNOWLEDGMENTS

The authors would like to thank Vitali Morozov at ANL for helping them to better understand the role of the IBM Blue Gene/L's performance counters in relationship to the overall architecture. Blue Gene/L computational resources were provided by the Center for Computational Nanotechnology Innovations (CCNI) at Rensselaer Polytechnic Institute.

REFERENCES

Bauer, D. W., C. D. Carothers, and A. Holder. 2009. "Scalable Time Warp on Blue Gene Supercomputers". In *PADS*, edited by P. F. Reynolds and C. Tropper, 35–44. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Beane, G. L. 2004, August. "The effects of microprocessor architecture on speedup in distributed memory supercomputers". Master's thesis.

Browne, S., J. Dongarra, N. Garner, G. Ho, and P. Mucci. 2000. "A portable programming interface for performance evaluation on modern processors". 14 (4): 189–204.

Brunheroto, J. R., V. Salapura, F. F. Redigolo, D. Hoenicke, and A. Gara. 2005. "Data cache prefetching design space exploration for Blue Gene/L supercomputer". In *Proceedings of the 17th International Symposium on Computer Architecture on High Performance Computing*, edited by D. A. Jimnez and S. Mukherjee, 201–208. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Carothers, C. D., and K. S. Perumalla. 2010, December. "On Deciding between Conservative and Optimistic Approaches on Massively Parallel Platforms". In *Proceedings of the 2010 Winter Simulation Conference*, 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.

Carothers, C. D., K. S. Perumalla, and R. M. Fujimoto. 1999, December. "The effect of state-saving in optimistic simulation on a cache-coherent non-uniform memory access architecture". In *Proceedings of the 1999 Winter Simulation Conference*, edited by P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. Evans, Volume 2, 1624 –1633 vol.2. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Chen, G., and B. K. Szymanski. 2005, December. "DSIM: scaling time warp to 1,033 processors". In *Proceedings of the 2005 Winter Simulation Conference*, edited by M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 346–355. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Davis, K., A. Hoisie, G. Johnson, D. J. Kerbyson, M. Lang, S. Pakin, and F. Petrini. 2004. "A Performance and Scalability Analysis of the BlueGene/L Architecture". In *Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, SC '04, 41–52. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Gustafson, J. L. 1990, apr. "Fixed Time, Tiered Memory, and Superlinear Speedup". In *Proceedings of the 5th Distributed Memory Computing Conference*, Volume 2, 1255 –1260. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Holder, A., and C. D. Carothers. 2008. "Analysis of Time Warp on a 32,768 processor IBM Blue Gene/L Supercomputer". In *Proceedings of the 20th European Modeling and Simulation Symposium*, edited by M. A. Piera, F. Longo, R. M. Aguilar, and C. Frydman. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Intel 2009a, March. "Intel Xeon Processor 5500 Series, Datasheet, Volume 1". [Online; accessed 7-April-2011].

Intel 2009b, July. "The New Intel Xeon Processor 5500 Series Catapults Processor Performance, Scalability, and Efficiency Huge Win for the Storage Industry.". [Online; accessed 7-April-2011].

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

Mindlin, P., J. R. Brunheroto, L. D. Rose, and J. E. Moreira. 2003. "Obtaining Hardware Performance Metrics for the BlueGene/L Supercomputer". In *Proceedings of the International Conference on Parallel and Distributed Computing (Euro-Par)*, edited by H. Kosch, L. Böszörményi, and H. Hellwagner, Volume 2790 of *Lecture Notes in Computer Science*, 109–118. New York, NY: Springer.

Perumalla, K. S. 2007. "Scaling time warp-based discrete event execution to 10^4 processors on a Blue Gene supercomputer". In *CF '07: Proceedings of the 4th international conference on Computing frontiers*, 69–76. New York, NY, USA: ACM.

Seal, S. K., and K. S. Perumalla. To Appear In 2012. "Reversible Parallel Discrete Event Formulation of a TLM-based Radio Signal Propagation Model". *ACM Transactions on Modeling and Computer Simulation* 22 (1).

Shmueli, E., G. Almasi, J. Brunheroto, J. Castanos, G. Dozsa, S. Kumar, and D. Lieber. 2008. "Evaluating the effect of replacing CNK with linux on the compute-nodes of blue gene/l". In *Proceedings of the 22nd annual international conference on Supercomputing*, 165–174. New York, NY, USA: ACM.

Sutter, H. 2008, January. "Going Superlinear". [Online; accessed 7-April-2011].

AUTHOR BIOGRAPHIES

AKINTAYO HOLDER is a Ph.D. student in the Computer Science Department at Rensselaer Polytechnic Institute. His research interests are parallel and distributed computing. His email address is holdea@cs.rpi.edu.

CHRISTOPHER CAROTHERS is a Professor in the Computer Science Department at Rensselaer Polytechnic Institute. He received the Ph.D., M.S., and B.S. in Computer Science from Georgia Institute of Technology in 1997, 1996, and 1991, respectively. His research interests include parallel and distributed systems, simulation, networking, and computer architecture. His email address is chrisc@cs.rpi.edu.