Proceedings of the 2002 Winter Simulation Conference
E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

FAST CELL LEVEL ATM NETWORK SIMULATION

Xiao Zhong-e
Rob Simmonds
Brian Unger

Dept. Computer Science,
University of Calgary
CANADA

John Cleary

Dept. Computer Science,
University of Waikato
NEW ZEALAND

ABSTRACT

This paper presents performance results for cell level ATM network simulations using both sequential and parallel discrete event simulation kernels. Five benchmarks are used to demonstrate the performance of the simulation kernels for different types of model. The results demonstrate that for the type of network models used in the benchmarks, the TasKit simulation kernel is able to outperform all of the other kernels tested both sequentially and in parallel. For one benchmark TasKit is shown to outperform a conventional sequential simulation kernel by a factor of 3. For the same benchmark TasKit is shown to outperform the best of the other parallel kernels tested by a factor of 6. The paper explains how this performance advantage is achieved and cautions that additional research into automatic model partitioning will be essential to make this technology accessible to the general simulation community.

1 INTRODUCTION

This paper presents performance results for cell level simulations of ATM networks. Results are presented for 5 benchmark models using 2 simulated networks. The benchmarks employ different types and amounts of synthetic traffic. Each benchmark is run on 1, 2, 4, 8 and 16 processors of a shared memory multiprocessor using 3 parallel discrete event simulation (PDES) kernels. The results are compared to the performance achieved running the same benchmarks using an efficient sequential simulation kernel.

The benchmarks are the same as those previously presented in Unger *et. al.* (2000). The results presented here include those for the sequential kernel CelKit (Gomes *et. al.* 1995), the conservative PDES kernel WaiKit (Cleary and Tsai 1997) and the optimistic kernel WarpKit (Xiao and Unger 1995) all of which were compared in (Unger *et. al.* 2000). This paper adds results for the TasKit kernel (Xiao *et. al.* 1999) that like WaiKit implements a conservative PDES algorithm, but employs a multi-level scheduling scheme that has been demonstrated to perform effectively in network simulators.

The results presented show that TasKit outperforms the other kernels in all of the experiments. For one large benchmark model TasKit executes 3 times faster than CelKit on a single processor and up to 26 times faster on 16 processors. TasKit is shown to run 6 times faster than the best of the other parallel algorithms on this benchmark.

It should be noted that cell level ATM network simulations have low event granularities and a topological structure that particularly suits constervative PDES kernels. Other simulation models with larger event granularites and less well defined communication paths can perform poorly using these algorithms. The purpose of this paper is only to show the reliative performance of simulation kernels using different algorithms for one class of models.

The rest of the paper is laid out as follows. Section 2 gives an overview of the 4 kernels that will be compared. Section 3 describes the benchmark models used for the comparison. Section 4 presents the performance results collected. Finally, Section 5 provided concluding remarks and summary.

2 SIMULATION KERNELS

This section gives an overview of the 4 kernels for which benchmark results are presented in this paper. The descriptions are brief due to space restrictions. The reader is directed to other papers that more fully describe each of the kernels.

The simulation kernels described in this section all use a modeling API implementing the *logical process modeling methodology* (Chandy and Misra 1979). With this the physical system being modeled is viewed as a set of interacting physical processes that only interact by exchanging messages. The physical processes are mapped to logical processes (LPs) and the messages are mapped to events in the simulator. The two conservative PDES kernels, WaiKit and TasKit, also require a channel-based communication view to be employed. With this channels are statically defined between pairs of LPs that may communicate during the simulation. These channels are required to calculate which events are currently safe to execute at each LP.

2.1 CelKit

The CelKit kernel, formally known as SimKit (Gomes *et. al.* 1995)), is a sequential simulation kernel using the "central event list" (CEL) approach. During each execution session the smallest timestamped event is removed from the CEL and executed by its destination LP. During the execution session new events with timestamps at least as great as the current event could be generated, each of which is placed into the CEL. Causal ordering is maintained by the CEL which is implemented as a priority queue. The performance of a CEL simulator owes much to the efficiency of its priority queue implementation. CelKit employs a splay tree (Sleator and Tarjan 1985) based CEL.

2.2 WarpKit

The WarpKit kernel (Xiao and Unger 1995) is an optimistic parallel simulation kernel implementing the Time Warp algorithm (Jefferson 1985). WarpKit is based on an early version of the GTW kernel (Fujimoto 1990), but has been extensively modified.

LPs are statically partitioned and assigned to processors. After each execution session, the next LP to be executed is the one with the smallest timestamped event at the local processor. Events are executed without checking if other events that have not yet been received by an LP will later invalidate the work done in executing this event. In Time Warp, causality errors are detected when straggler events (i.e., events with timestamps smaller than the timestamp of the last event executed by this LP) arrive. At this point, events executed prematurely are rolled back. This is achieved by reinserting future events into the LP's event queue and restoring the state of the LP with values stored during its forward execution. Since executing events early can lead to new events being generated and dispatched in error, special cancellation events called *anti-messages* are dispatched to eliminate errant events. These anti-messages can also trigger rollback if their destination LP has executed the target event before the anti-message is received.

WarpKit employs fast shared-memory global virtual time calculation (Xiao *et. al.* 1995) and incremental state saving (Cleary *et. al.* 1994). Like other Time Warp kernels, WarpKit's performance benefits from loosely coupled operation, but suffers from the overhead of state saving and rollback.

2.3 WaiKit

WaiKit (Cleary and Tsai 1997) is a conservative PDES simulation kernel using a modified version of the Chandy Misra Bryant (CMB) algorithm (Chandy and Misra 1979, Bryant 1977). Conservative algorithms operate by calculating a safe-time value for an LP representing the minimum timestamp of any event that could arrive at the LP in the future. Using the safe-time, LPs can execute local events with timestamps less than the safe-time with certainty that no causality error could occur. In WaiKit, statically defined channels are used to exchange both events and safe-time information between communicating LPs. Each channel has a clock representing the minimum timestamp of any event that could arrive on the channel in the future. Each LP calculates a safe-time at the beginning of its execution session as the minimum input channel clock. At the end of an execution the LP updates the clocks of all its output channels to be the minimum of timestamp of any event passed along the channel and the minimum timestamp that could be given to any event that could be inserted into the channel by this LP in the future.

The major innovation in WaiKit was to use a fixed LP scheduling order at each processor and to order the LPs such that work would be followed though the system without the need for priority based scheduling. The aim is to have each LP execution session either produce events that will be used by the next LP to execute, or to increase the safe-time of the following LP enabling it to advance futher in its ensuing execution session. This technique was shown to be very effective for some simulations, but could suffer badly in other cases where LPs would often be scheduled that were not currently able to make progress.

2.4 TasKit

TasKit (Xiao *et. al.* 1999) builds on ideas introduced in WaiKit and introduced the concept of scheduling based on critical channels. In TasKit, LPs are partitioned into groups known as tasks. These tasks could be allocated statically to processors, but in TasKit the tasks are placed in a central queue accessed by all processors. This provides a simple form of dynamic load balancing, though it could introduce problems with contention and cache pollution.

The LP to task partitioning decisions are based on topology. LPs that communicate often tend to be grouped together. TasKit has two types of task employing different internal scheduling mechanisms. The cluster task type has a single event queue and acts locally like a CEL sequential simulator bounded by the current minimum input channel clock. The fixed schedule task, described as pipe task in Xiao *et. al.* (1999), always executes the LPs it holds in the same order each time the task is scheduled for execution. This results in very efficient scheduling for these LPs, though

LPs are only assigned to fixed schedule tasks if the graph defined with the LPs as vertices and channels as edges, is directed acyclic (i.e., is a DAG).

The scheduling of tasks is performed using the Critical Channel Traversing (CCT) (Xiao *et. al.* 1999). At the end of an execution session, a task assigns one of its input channels as its critical channel. This channel must be one of the input channels sharing the minimum input channel clock. When the task at the sending end to this channel finishes its execution session, it observes that the channel is critical and schedules the destination task for execution. This greatly reduces the chance that tasks will begin an execution session before their minimum input channel clock has advanced, which often proved to be a problem with WaiKit.

One of the implementation goals for TasKit was to reduce the number of mutual exclusion operations required for parallel operation on a shared memory computer. The channels used to pass events between LPs use single writer / single reader semantics the do not require locks. They do require the use of memory barriers on computers that don't support a sequentially consistent view of memory, but these barriers do not require communication between processors and are relatively inexpensive to perform. A lock is used on the central task scheduling queue.

3 ATM SIMULATION BENCHMARKS

Two ATM network topologies are used as the basis for five benchmarks in the performance study. The first of these, called Wnet, models the physical topology of a regional ATM test-bed network in western Canada. The second model, called the NTN, models the physical topology of a Canada-wide experimental ATM network called the National Test Network. Each benchmark model uses a mix of traffic types to produce light, medium and heavy loads.

Table 1 summarizes the six benchmark scenarios. The first three columns give the number of traffic source/sink pairs for the three major traffic types, i.e., Ethernet (models aggregate LAN traffic), MPEG (models MPEG video traffic), and TCP (models TCP/IP based Internet traffic). Message flow on TCP connections is intermittent and bursty. The two columns in “number of nodes” show the number of ATM switches (SW) in a scenario and the number of gateways to the ATM network (i.e., multiplexer/demultiplexer units). The next column shows the number of LPs in the network's logical representation. The final column shows the number of tasks used to group LPs for the simulations using TasKit.

3.0.1 Wnet

The Wnet benchmark topology represents a regional ATM test-bed in Western Canada. This network connects five universities in three provinces. The network consists of 11 ATM switches, spanning a geographic distance of approximately 800 kilometers. Two benchmark scenarios, Wnet-1 and Wnet-2, represent different traffic mixes. Wnet-1 contains very light overall traffic load (10 Ethernet, 2 MPEG), but on one site the traffic load is very high and the line connected to the site becomes saturated. Wnet-2 represents the typical traffic in the real Wnet test-bed - JPEG video for distance education offerings between universities, TCP for file transfers between researchers and supercomputer centers, and Ethernet traffic as a generator of background traffic load between sites. Among the 25 traffic sources in Wnet-2 (2 Deterministic, 2 Bernoulli, 10 Ethernet, 8 MPEG, 3 TCP), the 3 TCP traffic streams account for 70% of the total traffic load.

3.0.2 NTN

The NTN network topology represents the Canada-wide ATM National Test Network (as of March 1996). The network has 54 ATM switches, and spans a geographic distance of approximately 3000 kilometers. The backbone of the network runs at 45 Mbps, and provides connectivity between five regional ATM test-beds (one of which is Wnet).

The NTN is a good example of a network with a high delay-bandwidth product. That is, the combination of high transmission speeds and large end-to-end propagation delays means that hundreds or thousands of ATM cells can be in transit at any time in the network links. These types of network scenarios are of particular interest to ATM networking researchers evaluating, for example, the effectiveness of feedback-based congestion control policies, the effectiveness of network traffic management on a national scale and end-to-end quality of service guarantees in large inter-networks.

The traffic in the NTN benchmarks consists of a mix of MPEG/JPEG video streams, TCP file transfers, and highly bursty LAN data traffic. Approximately 75% of the traffic sources have their traffic sinks on the same regional network. The remaining traffic flows traverse the national backbone.

The 3 benchmarks for the the NTN network model are NTN-1, NTN-2, and NTN-3. All 3 scenarios have the same 355 traffic sources deployed at the same locations. However, different traffic loads are produced by changing the parameters of the sources. The average link utilization in each model is 30% for NTN-1, 50% for NTN-2 and 80% for NTN-3.

4 PERFORMANCE

This section presents performance results comparing the performance of simulation executions using 5 benchmark models with 4 simulation kernels. First the relative performance of simulation runs using the CelKit sequential kernel and the TasKit kernel on a single processor workstation are presented. Then the performance of the 3 parallel capable

Table 1: Summary of Wnet and NTN Benchmark Scenarios

benchmark models	number of traffic source/sink pairs			number of nodes		num. of LPs	num. of tasks
	ether	mpeg	tcp	SW	gate		
Wnet-1	10	2	0	11	9	181	26
Wnet-2	10	8	3	11	6	173	12
NTN-1	62	269	24	54	39	1381	112
NTN-2	62	269	24	54	39	1381	112
NTN-3	62	269	24	54	39	1381	112

kernels is compared to the performance of CelKit on a 16 processor SGI Power Challenge parallel computer.

4.1 Single Processor Execution

This section provides results demonstrating the performance of a version of the TasKit kernel compiled specifically for operation on a single processor computer. This version of TasKit is identical to the parallel version, except that all of the mutual exclusion operations (e.g., locks and memory barriers) are removed using preprocessor directives.

Table 2 shows the relative performance of sequential TasKit and CelKit for the 5 ATM-TN benchmarks running on an SGI O2 R5000 workstation. As can be seen, TasKit sequential has a performance advantage over CelKit in every case.

The performance advantage achieved depends greatly on the number of events that can be executed in each task execution session. The more events executed, the lower the overall scheduling overhead. Also, cache locality is improved with the same set of event buffers being reused within each execution session and the same set of LPs executing many events.

Wnet-2 introduces TCP traffic that introduces feedback and thus reduces the concurrency in the model. Therefore, the time windows available for an execution session is smaller for Wnet-1 than for Wnet-2 resulting in slightly less advantage being gained using TasKit for Wnet-2. This difference is small since the greater overall traffic load in Wnet-2 increases the number of events that can be executed in any fixed period.

The results for the NTN benchmarks show that the advantage gained by using TasKit increases as the traffic load increases. This is due to the greater efficiency achieved when large numbers of events can be executed in each task execution session.

4.2 Parallel Execution

This section compares the performance of the 3 parallel enabled simulation kernels to the performance achieved using CelKit for the same set of benchmarks. All the benchmarks were run on a 16 processor SGI Power Challenge shared memory parallel computer.

In each case a performance measure was obtained running CelKit on a single processor. Then performance results were obtained for TasKit, WaiKit and WarpKit, running on 1, 2, 4, 8 and 16 processors. The graphs presented plot the performance of the 3 parallel kernels to the performance result achieved with CelKit.

4.2.1 WNet Results

Figure 1 compares the performance of the 4 kernels for the Wnet-1 benchmark. The performance of TasKit is superior to that of all the other kernels and increases as more processors are utilized. The performance of TasKit on 1 processor is just under twice that achieved using CelKit. This performance is similar to the performance achieved using the sequential version of TasKit on a single processor machine. This demonstrates the low overhead of mutual exclusion locks in the SGI IRIX operating system.

The performance of WaiKit is also better than that of CelKit in all cases. Here WaiKit benefits from its very low scheduling overhead. It is clear however that the scheduling scheme employed in TasKit massively outperforms the simpler WaiKit approach.

The performance of WarpKit demonstrates the problems of using Time Warp for simulations with very low event execution granularity. The overhead of state saving means that even running on a single processor, where no rollbacks can occur, the performance of WarpKit is less than half that of CelKit.

The results for the Wnet-2 benchmark (Figure 2) again show TasKit's superior performance. In this case none of the algorithms scale as well as they did for the Wnet-1 benchmark. This is explained by the feedback in the model introduced by the TCP flows in the Wnet-2 configuration. In this case TasKit only manages to execute 5 times as fast as CelKit even when 16 processors are employed. Note however that TasKit runs 2 times as fast as CelKit on a single processor and 3.5 times faster than CelKit on 2 processors.

The WaiKit performance suffers in this case due to TCP flows. These increase the dependences between adjacent

Table 2: TasKit Sequential Performance Relative to CelKit

	Wnet-1	Wnet-2	NTN-1	NTN-2	NTN-3
Speedup	1.96	1.93	2.32	2.59	3.18

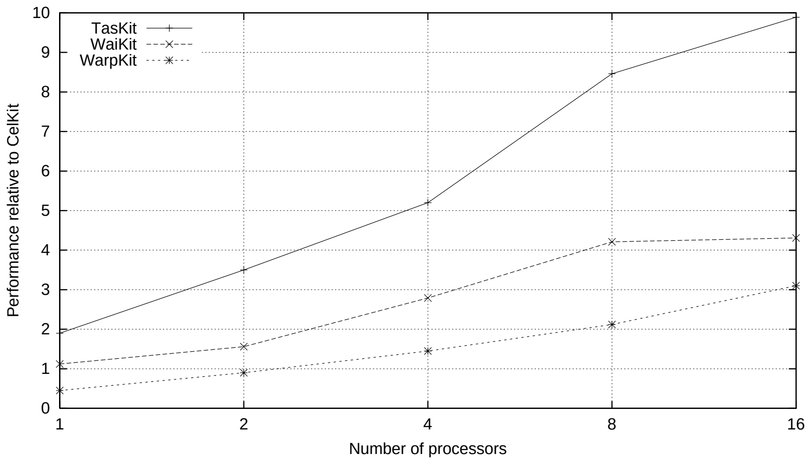

Figure 1: Performance Results for Wnet-1 Benchmark Scenario

LPs and increase the chance that any particular LP scheduled will have no work to do. Therefore, even though the same network is modeled as for Wnet-1, WarpKit outperforms WaiKit given enough processors.

4.2.2 NTN Results

The best results obtained using TasKit are achieved with the NTN benchmark. The performance advantage in this case is due in part to the structure of the network model being ideally suited to being mapped to TasKit's fixed schedule tasks. The graphs in this section show two sets of results for TasKit. One labeled "TasKit", use the same traffic models as the other systems. The second set of results labeled "TasKit-SD", use a modified version of the TCP traffic model that implements a lookahead optimization described here as *silence detection*. This advances the clock on an output port carrying a TCP stream when it is determined that the current TCP connection is complete. The clock is advanced to the time at which the next connection will be started. Results for this case are shown separately since with additional work it could have been possible to perform similar modifications to the traffic models used in the other experiments. These SD results are only provided to show that additional work by the modeler can lead to greater performance.

Results of the NTN-1 benchmarks (Figure 3) show that TasKit achieves a maximum performance of 14 times faster than CelKit on 16 processors using the standard traffic model. The modified version provides just over 16 times the performance of CelKit on the same number of processors. Both WaiKit and WarpKit perform poorly, with both achieving less than half the performance of CelKit on a single processor. The poor performance of WarpKit can again be explained by the overhead inherent in state saving during forward execution. On 16 processors the performance of WarpKit is over 3 times that of CelKit. Note that speedup achieved by WarpKit relative to itself is 6, which is slightly better than the speedup that TasKit achieves relative to itself. The performance of WaiKit with this model again suffers from feedback in TCP flows. While it does achieve some increase in performance with increased numbers of processors, it still performs the worst for this benchmark.

Performance results for the NTN-2 and NTN-3 benchmarks (Figures 4 and 5) show similar properties. In each case, for benchmarks not using silence detection the performance of TasKit is better than all the other algorithms in all cases. In both these sets of results there is little difference between the performance of WaiKit and WarpKit on 1, 2 and 4 processors. On 8 and 16 processors the better scaling properties of WarpKit cause it to perform better than WaiKit. Again, despite the large performance advantage gained by using TasKit over using WarpKit, the relative performance using the same kernel on 1 and 16 processors is very close. This shows that the major a achievement of TasKit is its ability to provide excellent performance on small numbers of processors. The performance is best when silence detection is employed with a maximum of a 17% performance advantage over not using silence detection on 16 processors of NTN-2.

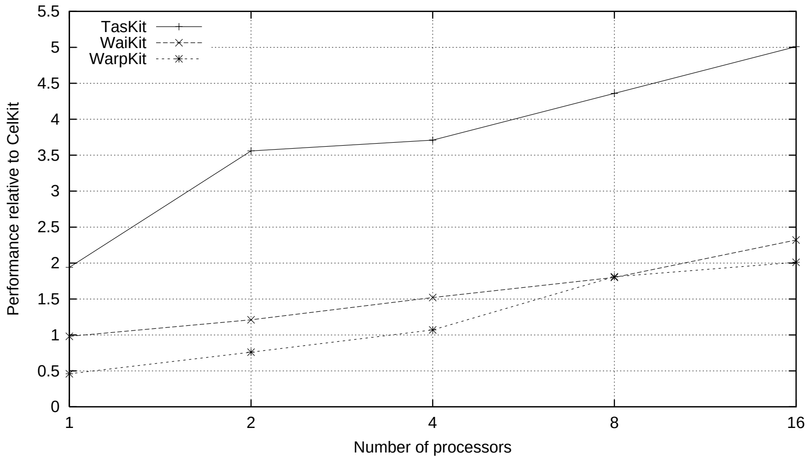

Figure 2: Performance Results for Wnet-2 Benchmark Scenario

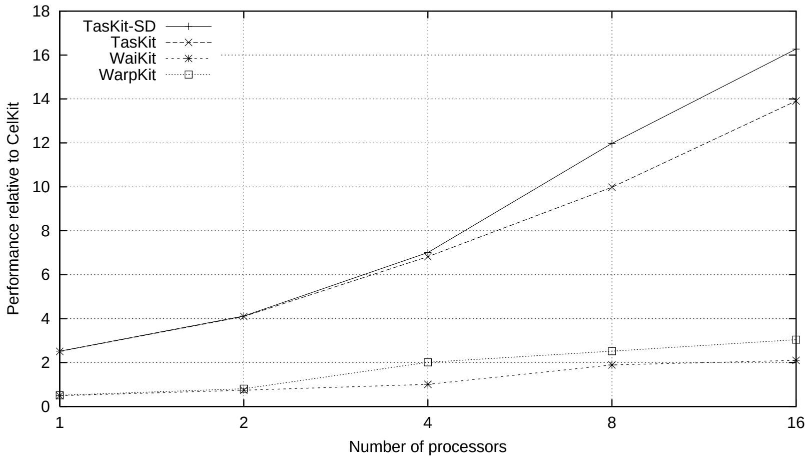

Figure 3: Performance Results for NTN-1 Benchmark Scenario

5 SUMMARY AND CONCLUSIONS

This paper has presented performance results of running cell level ATM network simulations on a shared memory parallel computer. Each benchmark was run using 1 purely sequential simulation kernel and 3 parallel capable kernels. Experiments were performed executing the benchmarks on 1, 2, 4, 8 and 16 processors.

The results show that the TasKit kernel was able to execute up to 26 times faster than a conventional central event list kernel for one of the benchmarks on 16 processors. As pointed out, a possibly more interesting result is that using TasKit on a single processor resulted in over 3 times faster execution then achieved using the more conventional sequential kernel for this benchmark. Therefore, even at 26 times faster than CelKit, the speedup achieved by TasKit on 16 processors relative to its performance on 1 processor was only 8.6. This indicates that even greater performance may be achievable if the core scheduling mechanism employed in TasKit was made more scalable.

Each of the benchmarks presented in this paper was carefully partitioned to achieve the highest performance for each of the kernels. Since the parallel kernels are far more sensitive to correct partitioning than the sequential kernel, their performance has been improved in comparison by this "expert" tuning. An automatic partitioning scheme tuned to the requirements of the task based scheduling will be essential to really take advantage of TasKit's technologies in a general simulation framework. Development of such a scheme is currently underway.

ACKNOWLEDGMENTS

The TeleSim group is funded by grants from the Canadian Natural Sciences and Engineering Research Coun-

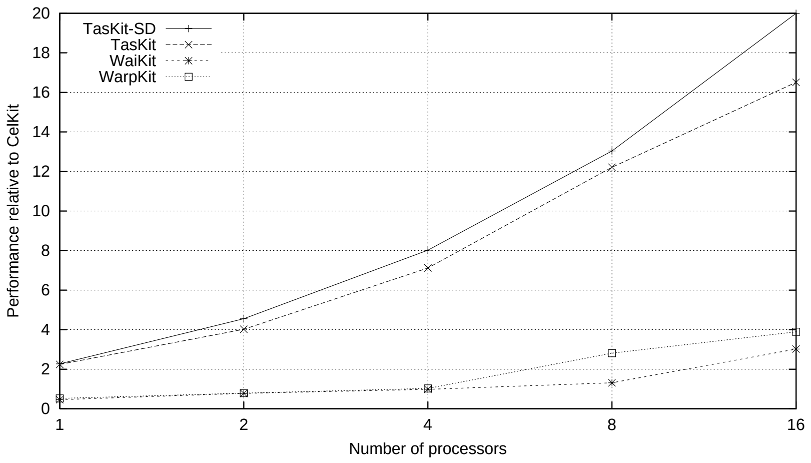

Figure 4: Performance Results for NTN-2 Benchmark Scenario

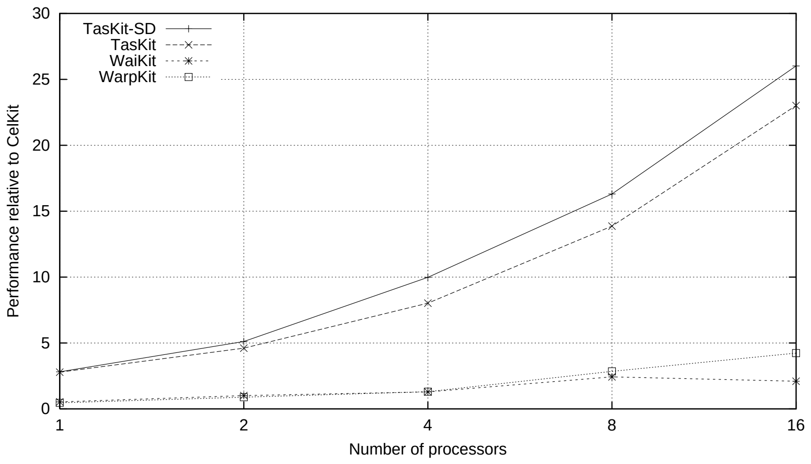

Figure 5: Performance Results for NTN-3 Benchmark Scenario

cil (NSERC) and Alberta Science and Research Authority (ASRA).

REFERENCES

Bryant. R. 1977. Simulation of Packet Communication Architecture Computer Systems. Technical Report MIT/LCS/TR-188, MIT.

Chandy K. M. and J. Misra. 1979. Distributed simulation: A case study in design and verification of distributed simulation. *IEEE Transactions on Software Engineering*, 5(5):440–452.

Cleary J., F. Gomes, B. Unger, X. Zhonge, and R. Thudt. 1994. Cost of state saving and rollback. In *Proceedings of the 1994 Workshop on Parallel and Distributed Simulation* pages 94–101. The Society for Computer Simulation.

Cleary J.G., and J. Tsai. 1997. Performance of a conservative simulator of ATM networks. In *Proceedings of the 11th Workshop on Parallel and Distributed Simulation*, pages 142–145. The Society for Computer Simulation.

Fujimoto R.M. 1989. Time Warp on a shared memory multiprocessor. *Transactions of the Society for Computer Simulation*, 6(3):211–239.

Gomes F., S. Franks, B. Unger, Z. Xiao, J. Cleary, and A. Covington. 1995. SIMKIT: A high performance logical process simulation class library in C++. In *Proceedings of the 1995 Winter Simulation Conference*, pages 706–713. The Society for Computer Simulation.

Jefferson D.R. 1985, Virtual time. *ACM Transactions on Programming Languages and Systems*, pages 404–425.

Sleator D., and R. Tarjan. 1985. Self-adjusting binary search trees. *Journal of the ACM*, 32(3):652–686.

Unger B.W., X. Zhonge, J.G. Cleary, J. Tsai, and C. Williamson. 2000. Parallel shared-memory simulator performance for large ATM networks. *ACM Transactions on Modeling and Computer Simulation*, 10(4):358–391.

Xiao Z., and B. Unger. 1995. Report on warpkit - performance study and improvement. Technical Report Technical Report 98-628-19, Computer Science Department, University of Calgary.

Xiao Z., B. Unger, R. Simmonds, and J. Cleary. 1999. Scheduling Critical Channels in Conservative Parallel Discrete Event Simulation. In *Proceeding of the 13th Workshop on Parallel and Distributed Simulation*.

AUTHOR BIOGRAPHIES

XIAO ZHONG-E is a Research Associate in the Department of Computer Science at the University of Calgary. His research interests include parallel discrete event simulation, network protocol modeling and simulation, logic programming, and parallel and distributed software development. He earned the B.Sc. in Electrical Engineering from Jiao-Tong University, Shanghai, was selected for advanced studies in a nation wide competition, and was a Visiting Scholar in Computer Science at the University of Calgary.

ROB SIMMONDS is an Adjunct Assistant Professor in the Department of Computer Science at the University of Calgary and currently manages the TeleSim research group. His main research interests are in parallel simulation, parallel network emulation and distributed systems. Dr. Simmonds holds B.Sc. and Ph.D. degrees from the School of Mathematical Sciences at the University of Bath in England.

BRIAN UNGER is the President and CEO of iCORE, the "informatics Circle of Research Excellence", and is a Professor of Computer Science at the University of Calgary where he leads the TeleSim research group with Dr. Simmonds. iCORE is a not-for-profit corporation aimed at attracting and funding research leaders in the information and communications sciences at Alberta universities. Dr. Unger received a Ph.D. in Information and Computer Science from the University of California, San Diego, M.Sc. from the University of California, and B.Sc. from Loyola University, Los Angeles.

JOHN CLEARY is a full Professor of Computer Science at the University of Waikato and Chief Technology Officer and co-founder of ReelTwo Ltd which is using machine learning techniques for text categorization. His current research interests include optimistic algorithms, logic programming, network measurement, and text categorization. Dr. Cleary holds an M.Sc. in Mathematics and a Ph.D. in Electrical Engineering from the University of Canterbury in New Zealand.