Proceedings of the 1990 Winter Simulation Conference
Osman Balci, Randall P. Sadowski, Richard E. Nance (eds.)

A TIME WARP IMPLEMENTATION OF SHARKS WORLD

Matthew T. Presley
Peter L. Reiher

Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, California 91109

Steven Bellenot

Department of Mathematics
Florida State University
Tallahassee, Florida 32306

ABSTRACT

This paper discusses the TWOS implementation of the Sharks World simulation, a simple simulation of sharks and fish swimming through a toroidal world. It covers the design of the simulation and presents performance results for it. Sharks World performs very well under TWOS in a variety of configurations. Also, the paper discusses the process of writing the application and changes and improvements that could be made to it.

1. INTRODUCTION

Sharks World is a simulation of sharks and fish swimming through a two-dimensional toroidal world. When a shark and a fish are within a certain distance, the shark will eat the fish, causing the fish to disappear from the simulation. Sharks and fish do not interact in any other way. Sharks may swim through each other, as may fish. Each shark or fish starts with an initial speed and direction of movement. Neither changes at any time for any fish or shark. Sharks and fish do not reproduce.

Sharks World is not meant to be a realistic biological simulation. Rather, it is an extremely simple benchmark demonstrating a basic form of objects interacting at a distance [Conklin et al. 1990]. Sharks World has been implemented using several different types of synchronization mechanisms for parallel simulation. This paper describes an implementation using an optimistic synchronization method. It was written for the Time Warp Operating System, an optimistic parallel discrete event simulation operating system developed at the Jet Propulsion Laboratory.

The Time Warp Operating System (TWOS) is a special-purpose distributed operating system whose goal is to run discrete event simulations on parallel hardware as quickly as possible. TWOS embodies the theory of virtual time, described in [Jefferson 1985].

TWOS extracts parallelism (and hence, speedup) from a distributed simulation by running the component objects of the simulation simultaneously on different nodes of the parallel processor. Simulation objects communicate via timestamped messages, and the arrival of one or more messages at an object at a given simulation time causes an event at that object. TWOS handles all synchronization between the nodes of the parallel processor, allowing the simulation designer to view the system as executing all events in strict timestamp order. In actuality, TWOS permits faster nodes to run ahead of slower nodes in simulation time. If an event proves to have been done before an earlier event that has an effect on it, TWOS maintains sufficient information to completely roll back that erroneous event and undo any of its consequences. A more complete description of the system can be found in [Jefferson et al. 1987].

TWOS has extracted high speedups from many different simulations, including theater-level military simulations [Wieland et al. 1989], computer network simulations [Presley et al. 1989], and biological simulations [Ebling et al. 1989]. One of the fundamental benchmarks used by TWOS is a simulation called Pucks [Hontalas and Beckman 1989]. Pucks simulates two-dimensional pucks moving and colliding on a frictionless table with cushioned sides. This problem is similar to Sharks World, as both simulations deal with objects moving in space and interacting at a distance. Pucks and Sharks World have some significant differences, however, in the form of the playing surface and in the rules of interaction.

This paper describes the TWOS implementation of Sharks World. It discusses the architecture of the simulation, presents performance figures for several different Sharks World scenarios, and examines why TWOS provides the performance it does. It also discusses methods of improving TWOS' performance on this application, and the possible effects of adding more complexity to the simulation.

2. SHARKS WORLD IN TWOS

The TWOS implementation of Sharks World divides the toroidal world up into a varying number of equally sized sectors. The world can be split into an arbitrary number of sectors, and the horizontal and vertical dimensions of a sector need not be equal. Each sector has responsibility for actions happening in its area of space. These sectors are the only objects in the simulation. Sharks and fish are not explicitly modelled as objects.

Each sector has responsibility for keeping track of all sharks and fish currently within its boundaries. A sector also has responsibility for notifying other sectors when one of its sharks or fish enters that other sector, or when a shark's attack radius first crosses the boundary of that other sector. Unlike the implementation in [Conklin et al. 1990], there are no special border regions within sectors.

Every sector maintains two lists, one of fish within the sector and one of sharks that may kill fish within the sector. The shark list contains not only those sharks that are currently within the sector, but also all sharks whose attack radius currently intersects the sector. These latter are called *visible sharks*. The sector determines when a fish or shark will leave, and which sector it will enter. The sector then sends a message to the next sector the creature will enter, informing it when the creature will arrive, at what point, and with what velocity and direction of movement. That next sector will in turn determine where the creature is to go next, and when, and send another message to that effect to the next sector to be entered.

The other activity in the model is sharks preying on fish. Sharks will eat any fish within their attack radius. Since the simulation is not being run in time steps, and since sharks are always hungry and take no time to eat a fish, the issue of which of two fishes in the attack radius will be eaten does not arise. If they enter the shark's attack radius at different times, the shark will eat both of them at their differing moments of entry. In the very rare cases where two fish enter the radius at exactly the same instant in simulation time, the shark will eat both of them at once. Eating a fish is not an event in the simulation. Only crossing sector boundaries and having a shark's attack radius cross sector boundaries are modelled as events.

When a fish enters a sector, the sector compares the fish's data to that of all sharks the sector knows about, both those within the sector and those visible in other sectors. The sector determines which shark, if any, will kill the fish. If no shark will kill the fish, the sector determines which sector the fish will enter next, and sends a message notifying that sector of the fishes arrival in the simulation future. If the fish does fall within the attack radius of a known shark, the fish is marked as being dead. No message is sent to the next sector. However, the sector does not yet discard its information about the fish, as some other shark may enter the sector later, yet get to the fish before the first shark.

When a shark becomes visible to a sector, the sector currently responsible for that shark sends a message to the sector about to see the shark. This message is similar to the fish arrival message. It contains the shark's position at the simulation time when it became visible, the shark's speed, and its direction of motion. The sector receiving the message checks its list of fish to determine which fish the shark will attack. Both live and dead fish are checked, since this shark may get to a supposedly dead fish before the shark that appears to have eaten it. Any fish previously listed as alive becomes dead. For all such fish, the sector performing this computation sent a message to the next sector the fish was to enter. That message must be cancelled, as dead fish don't move any more.

TWOS does not currently contain a message cancellation primitive, so cancellation of messages relating to deceased fish is done within the application. Sectors maintain information about the fish entry messages they have sent to other sectors. Should a shark enter a sector and eat a fish previously expected to escape the sector, this information will indicate that some other sector was erroneously informed of the fish's arrival. That erroneous message is cancelled by sending another message directing the receiver to ignore the first message. Note that this behavior is not related in any way to Time Warp message cancellation, and has nothing to do with Time Warp rollback. Such cancellations are equally necessary in the sequential version of the simulation, as they represent explicitly modelled guesses about future behavior and correction of those guesses as better information arrives.

Once a sector has marked a fish as dead, it knows that the fish will definitely die. The only uncertainty is which shark will eat it, at which point, at what time. Therefore, the sector maintains information about dead fish until it is certain that the fish will not be eaten by some other shark at some earlier simulation time.

A sector's lists of sharks and fish is garbage collected during each event. If the time of departure of the fish is before the time of the current event, that entry can be garbage collected. Also, if the time at which the shark's attack radius no longer intersects the sector has passed, the shark's entry can be garbage collected. Finally, if the time at which a fish died is earlier than the time of the current event, the fish's entry can be garbage collected.

3. THE PERFORMANCE OF TWOS SHARKS WORLD

The TWOS implementation of Sharks World achieves very good performance. It produces speedups of up to 29.5, often has efficiencies of greater than 50%, and sometimes achieves up to 69% of the theoretically possible speedup, as determined by critical path analysis. This section presents some performance results for TWOS Sharks World.

In the curves presented, two internal parameters of the simulation are varied. First, the total number of creatures present in the world at the start of the simulation is varied from 32 to 2048. (In each case, half of the creatures are sharks and half are fish.) Second, the decomposition of the world into sectors is varied. The world is split into either 64 or 256 sectors. Since sectors are the only objects in the simulation, the number of sectors puts a firm upper limit on the possible parallelism of the simulation. For instance, a simulation of a world split into 16 sectors cannot possible improve its speedup on 16 nodes by adding more nodes. (Runs were made for 4-by-4 sector decompositions, but the parallelism available in a simulation with only 16 objects is limited, so none of these curves are plotted here.)

The performance results shown here are for runs of 2000 time units. Creatures move at speeds approximately between 50 and 200 space units per time unit. The world is 64K space units on each side, so a typical creature would swim around the toroidal world 3 times during the run. Other Sharks World studies have used simulations that ran 100,000 time units, but the limited testing time available made it impossible to make sufficient TWOS runs at that length.

The configuration files used to assign sectors to nodes of the parallel processor were not carefully balanced. Assignment was round robin, ensuring that no node received more than 1 more sector than any other. No attempt was made to ensure that each node performed about the same amount of work as its fellows. Dynamic load management was not used for these runs.

The runs were made with TWOS version 2.4 running on the BBN GP1000 under the Mach operating system. A few performance figures are also included for the same machine running the Chrysalis Operating System.

The performance figures presented are speedups against the run time of TWSIM 2.4. TWSIM is a fast sequential simulator that runs on one node of the same hardware as the TWOS runs. It uses a splay-tree to implement a single event list, and has been extensively optimized for speed. It uses strictly sequential methods, with no rollbacks or antimessages, and no parallelism. TWSIM has the same user interface as TWOS, so exactly the same user application is run under both systems. TWOS speedups are found by dividing the TWSIM time for an application by the TWOS time. In all cases, each point plotted is the average of three separate runs using the identical configuration.

No runs with more than 2048 creatures are shown here. TWOS has handled Sharks World runs with 32,000 creatures, but TWSIM cannot handle many more than 2048 creatures with the current implementation of Sharks World, so good speedup numbers are not possible for larger numbers of creatures.

Figure 1 shows the speedup TWOS achieved for runs with the world divided into an 8-by-8 grid of sectors. The different curves show the speedups achieved for numbers of creatures between 32 and 2048. As the number of creatures increases, the average amount of work necessary to detect fish within the attack radii of sharks increases, increasing granularity. As expected, TWOS provides increasingly better speedups as the granularity of the computation increases. The best speedups are for 2048 creatures, the case with the highest granularity.

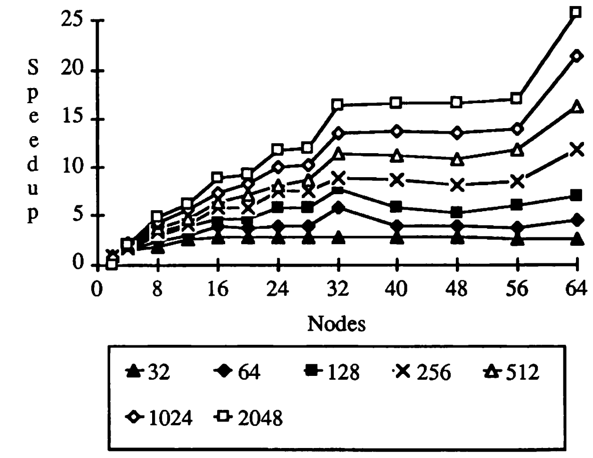

Figure 1. 8X8 Sector Sharks World Speedup

Figure 1 also demonstrates the importance of configuration and load balancing. For large numbers of creatures, the speedup curves show plateaus between 32 and 64 nodes, with little or no performance improvement at 40, 48, or 56 nodes. The reason for this phenomenon is that the entire simulation consists of 64 objects (one per sector), each of which does approximately the same amount of work. Thus, configurations that have exactly the same number of objects on each nodes are better balanced. In particular, when the number of nodes exceeds 32, some nodes have only one object, while other nodes have 2. The speedup for these runs is limited by the speed of the nodes hosting two objects, just as it was for 32 nodes. At 64 nodes, each node has exactly one object, allowing a substantial increase in speedup.

8-by-8 sector runs were not made for more than 64 nodes. Since the simulation only contains 64 objects, using more than 64 nodes would never improve speedups.

The speedups obtained for the 8-by-8 sector runs ranged between .63 (a slowdown) for 1024 creatures on 2 nodes to 25.5 for 2048 creatures on 64 nodes. The best speedup obtained for a 32 creature run was 2.75 on 20 nodes, indicating how little speedup is available for that simulation. Efficiencies (defined as the speedup divided by the number of nodes) ranged from 4% (32 creatures on 64 nodes) to 60% (2048 creatures on 8 nodes). The efficiency for the best speedup was 40%.

Figure 2 shows the same curves for a division of the world into 16-by-16 sectors, or 256 total objects. Points are shown only for 512, 1024, and 2048 creatures, as those simulations provided the best speedups. The maximum speedups obtained by this version of

Sharks World was 24.45, lower than the maximum for the 8-by-8 version. More interesting, though, is that the 16-by-16 sector runs do not show the plateaus that the 8-by-8 sector runs showed. Since there are more objects in the 16-by-16 sector runs, TWOS is able to better balance the runs are 48 or 56 nodes. If we were able to run on 192 nodes, the plateau would probably reappear, running no faster than 128 nodes, with a big gain at 256 nodes, as each object got its own node.

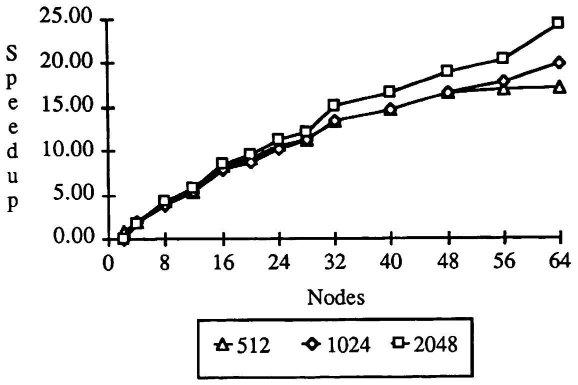

Figure 2. 16-by-16 Sector Sharks World Speedup

The 16-by-16 sector runs probably do not achieve as high a speedup as the corresponding 8-by-8 sector runs for 1024 and 2048 creatures because their granularity of computation is lower. The same number of creatures are present for both cases, and they swim to the same places, so each sector has fewer creatures to keep track of. Much of the granularity of Sharks World events comes from comparing lists of sharks and fish to check for intersections, so fewer entries in each sector's list means less work per sector, leading to a higher ratio of overhead to user work, and hence less speedup. The granularity of the 8-by-8 sector simulation with 2048 creatures is around 7 milliseconds per event. For the 16-by-16 sector 2048 creature runs, the granularity is 3 milliseconds per event. (TWOS actually gives better speedups for most simulations with granularities around 10 or 15 milliseconds per event.)

The 16-by-16 sector decomposition of 512 creature runs actually provides better speedups than the 8-by-8 sector decomposition for all numbers of nodes except 2 nodes. Since the same granularity argument applies here, the difference is probably due to superior load balancing by the 16-by-16 sector decomposition for 512 creatures.

Figure 3 shows some results for runs made on the Chrysalis Operating System. Both Mach and Chrysalis run on the same hardware, so the differences are wholly due to software, mostly below the level of TWOS. Chrysalis is a less capable system, but is also smaller and less intrusive. Figure 3 shows the two speedup curves for 8-by-8 sector 2048 creature runs under both Chrysalis and Mach. There is little difference, except that the Chrysalis speedup for 64 nodes is a bit higher. This is due to certain Mach initialization overheads that occur for high numbers of nodes. The phenomenon is less apparent at lower numbers of nodes because the overhead depends on the number of nodes, and because the longer run times of the low number of nodes tend to conceal the initialization overhead.

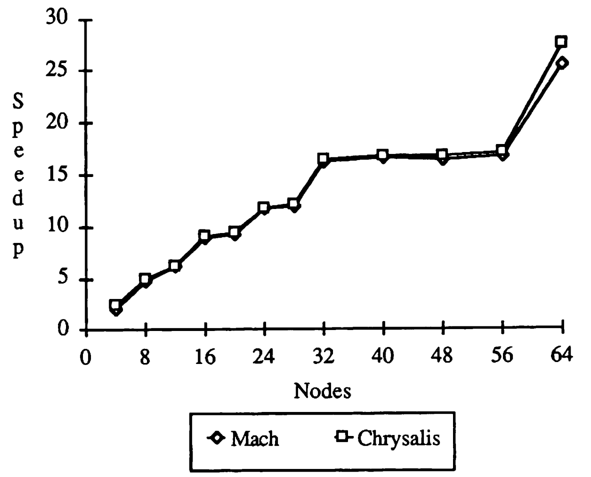

Figure 3. 8-by-8 Sector 2048 Creature Speedups

Figure 4 shows the 16-by-16 sector 2048 creature speedups for both Mach and Chrysalis. Here, the Chrysalis improvement becomes clear much sooner, as the load balancing problem caused by the small number of objects in the 8-by-8 sector runs does not obscure it. Starting around 40 nodes, the Chrysalis version of TWOS gets much better speedup than the Mach version. At 64 nodes, the Chrysalis version of this run got the best speedup obtained from any Sharks World run under TWOS, 29.5.

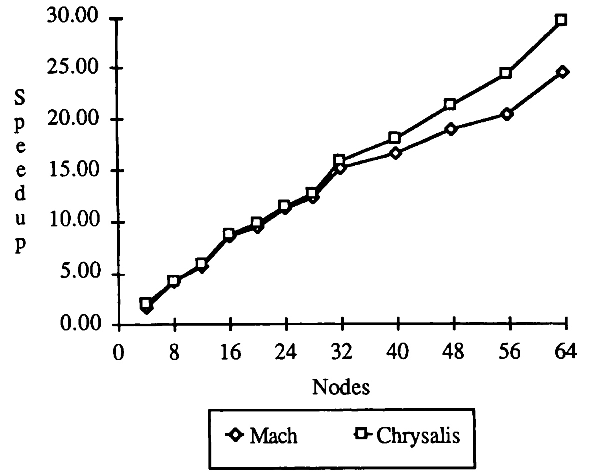

Figure 4. 16-by-16 Sector 2048 Creature Speedups

TWOS normally runs with lazy cancellation of messages [Berry 1986]. Sharks World is an application that should get great benefit from lazy cancellation of messages, since the only messages ever cancelled in Sharks Worlds runs are those sent when fish expected to cross a sector boundary later prove to have been eaten. As [Reiher et al 1990] showed, different applications can perform well or poorly with either lazy cancellation or aggressive cancellation. Figure 5 shows Sharks World's performance with lazy cancellation vs. aggressive cancellation. (TWOS contains a switch permitting it to run in either mode.)

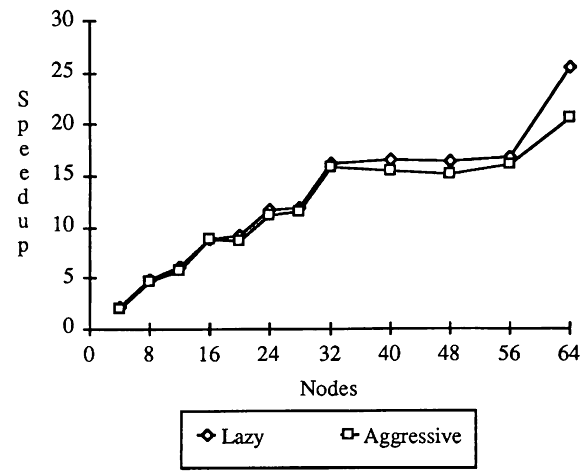

Figure 5. Lazy Versus Aggressive Cancellation Speedups

Figure 5 demonstrates that lazy cancellation has a clear advantage over aggressive cancellation for Sharks World, particularly at large numbers of nodes. (The lazy cancellation speedup is better than the aggressive cancellation speedup for all points but one. At 16 nodes, aggressive cancellation's speedup is insignificantly better.) Somewhat surprisingly, though, lazy cancellation is not all that much better. At most, it is around 25% faster than aggressive cancellation. But the aggressive cancellation run performs far more message cancellations than the lazy cancellation run does, by as high a factor as 838. Sharks World thus demonstrates an actual application for which lazy cancellation is really better. Also, it demonstrates that negative messages and rollbacks may not have disastrous effects on a simulation's performance. Despite sending as many as 76,000 negative messages, TWOS still provided reasonable performance. Clearly, most of the negative message activity took place off the critical path of computation, thereby impeding the important computation very little. This result serves as further evidence of the practicality of optimistic synchronization methods.

TWSIM can determine the critical path length of a simulation, where critical path length is defined as the longest set of event run times that must be processed sequentially. Critical path length sets a theoretical limit on the speedup obtainable by a simulation. (TWOS is able to exceed critical path speedup under certain conditions [Berry 1986], but does not do so for typical applications.) For a 4-by-4 sector simulation with 2048 creatures, TWOS on 16 nodes was able to achieve better than 69% of the critical path speedup.

TWSIM is unable to handle runs of Sharks World with many more than 2048 creatures, making it impossible to get a proper speedup figure for such runs. However, a TWOS run with 32,000 creatures was made successfully. While normal speedup figures are unavailable, some speedup comparisons can be made. This run was made 16-by-16 sectors, out to 2000 time units. One of the statistics produced was total committed event time. Total committed event time is constant between TWOS runs and TWSIM runs, by definition, as it is the sum of all time spent running committed events in TWOS. Since TWSIM runs only committed events, its event time should be the same.

One measure of speedup is zero overhead speedup. In essence, this is the speedup against a zero-overhead sequential version of the run, and is thus a much more conservative estimate of speedup than speedup versus TWSIM's run time. For the 32,000 creature run, the zero overhead speedup was 23.7 on 71 nodes. For comparison, the best zero overhead speedup for the 16-by-16 2048 creature run was 15.9.

This improvement in performance is due to an increase in the granularity of events. The more creatures in the simulation, the more comparison between sharks and fish are likely to be made per event. The granularity for the 2048 creature run was 3 milliseconds per event, while the granularity for the 32,000 creature run is 13 milliseconds, more than four times as great. At the extreme, the granularity per event will totally dominate the overhead per event, and the actual speedup for the simulation would be the same as the zero overhead speedup. At this point, parallelism and load balancing would be the only limitations on speedup.

4. DEVELOPING SHARKS WORLD FOR TWOS

The process of developing Sharks World for TWOS provided results almost as interesting as the performance figures. The code for Sharks World was written from scratch in two man weeks. No code from other TWOS simulations was used, and almost all of the algorithms were developed independently of other simulations. The resulting code included the ability to fully configure the Sharks World tests at run time, in terms of the number of sectors in the world, the number of creatures, and even the individual attack radii of the sharks. Initialization could be done by sending a single message to each of the sectors.

Debugging was also very easy. Most of the problems encountered related to performance, not correctness. (In certain cases, fixing these problems gave better run times, but poorer speedups.) All of the tests reported on in section 3 were performed without a single failure, except for runs that consumed too much memory to complete.

The code for Sharks World comprises 1200 lines of C code, including comments. It does not contain any code specific to Time Warp style simulation, other than the data structures necessary to link to TWOS. No code explicitly attempts to take advantage of TWOS's synchronization method, or even takes notice of it.

5. POSSIBLE IMPROVEMENTS

Sharks World is a rather unrealistic simulation that has several obvious possibilities for improvement. First, sharks never change their directions to attack fish, and fish never take evasive actions. The TWOS version of the code could easily have the creatures change the direction and speed of their movements for any of a variety of reasons. There is no evidence to suggest that any major changes would be necessary to the simulation to add this capability, nor that doing so would harm speedup. In fact, since deciding when to change directions and speeds would require extra calculations, leading to higher computational granularity per event, this change might well improve speedup.

Sharks and fish have very simple behavior in the basic model. If their behavior were more complex, they might have to be modelled as independent objects, rather than simply being table entries at sectors. Making this change would be more difficult than handling speed changes, but TWOS already has run several simulations that feature moving objects explicitly modelled, so it could easily be done for Sharks World, as well.

Sharks and fish never reproduce in the basic model. Assuming that they were not independently modelled, they could do so very simply. New table entries for sharks and fish could be made periodically, introducing new creatures to the world. Having reproduction be driven by proximity of two creatures of the same type would also be simple. If the creatures were independently modelled as objects, TWOS' dynamic object creation facility would still allow reproduction.

The simulation could easily be generalized to include an entire food chain of different types of creatures preying on each other. The TWOS implementation could be easily altered to handle this change.

The granularity per event could be artificially increased by adding delay loops to the simulation. Doing so would permit experiments to discover the inherent parallelism of the Sharks World model, by increasing granularity to the point that the per-event overhead was swamped by the granularity.

Sharks World has not been run with TWOS' dynamic load management facility turned on. Doing so would provide potentially interesting results, particularly in the cases where there were relatively few objects per node. For instance, figure 3 shows a plateau in the speedup curve between 32 and 64 nodes where some nodes host one object and some nodes host two. In this case, load management would tend to move objects back and forth throughout the simulation. The effect of this behavior on load management might prove very interesting. In addition, load management might provide slightly improved speedups for other scenarios.

6. CONCLUSIONS

Sharks World proved to be a very simple application to code for TWOS. It was coded and debugged in two man weeks, around four weeks after it was first discussed. After the initial debugging and tuning, it ran all tests without modifications.

Sharks World provides good performance under TWOS. It has demonstrated speedups as high as 29.5, with efficiencies as high as 63.75%. With sufficient numbers of sectors, TWOS provides a smooth speedup curve. If the number of sectors is close to the number of nodes, the speedup curve is likely to have a plateau between the points of two objects per node and one object per node. Sharks World does not have especially favorable event granularity for TWOS, so it would probably provide even better speedups with higher granularities.

Sharks World provides a test example for lazy cancellation versus aggressive cancellation. Due to its nature, lazy cancellation should do very well with Sharks World, and does indeed do better than aggressive cancellation. However, despite massive numbers of negative messages, aggressive cancellation provides acceptable performance comparable to that of lazy cancellation.

Sharks World was not expected to be a particularly good application for TWOS, but its performance has been more than acceptable. The development effort demonstrates how quickly and easily small simulations can be ported to TWOS. The resulting implementation is very flexible and could be easily altered if the model were changed, probably with little impact on performance.

ACKNOWLEDGEMENTS

This work was funded by the U.S. Army Model Improvement Program (AMIP) Management Office (AMMO), NASA contract NAS7-918, Task Order RE-182, Amendment No. 239, ATZL-CAN-DO.

The authors thank David Jefferson, Mike Di Loreto, Phil Hontalas, Brian Beckman, Fred Wieland, Leo Blume, Joe Ruffles, John Gieselman, Lawrence Hawley, and John Wedel, for their work on TWOS, TWSIM, and TWOS applications. We also thank Jack Tupman and Herb Younger for managerial support, and Harry Jones of AMMO, and John Shepard and Phil Lauer of CAA for sponsorship.

REFERENCES

Berry, O. (1986), "Performance Evaluation of the Time Warp Distributed Simulation Mechanism," Ph.D. dissertation, Department of Computer Science, University of Southern California, Los Angeles, CA.

Conklin, D., Cleary, J., and Unger, B. (1990), "The Sharks World (A Study in Distributed Simulation Design)," In Proceedings of the SCS Multiconference on Distributed Simulation, D. Nichol, Ed. Society For Computer Simulation, San Diego, CA, 157-160.

Ebling, M., Di Loreto, M., Presley, M., Wieland, F., and Jefferson, D. (1989), "An Ant Foraging Model Implemented On the Time Warp Operating System," In *Proceedings of the SCS Multiconference on Distributed Simulation*, Unger, B. and Fujimoto, R., Eds., Society For Computer Simulation, San Diego, CA, 21-28.

Hontalas, P. and Beckman, B. (1989), "Performance of the Colliding Pucks Simulation On the Time Warp Operating System (Part 2: A Detailed Analysis)," In *Proceedings of the 1989 Summer Computer Simulation Conference*, Clema, J. Ed., Society For Computer Simulation, San Diego, CA, 91-95.

Jefferson, D. (1985), "Virtual Time," *ACM Transactions on Programming Languages and Systems 7*, 3.

Jefferson, D., Beckman, B., Wieland, F., Blume, L., Di Loreto, M., Hontalas, P., Laroche, P., Sturdevant, K., Tupman, J., Warren, V., Wedel, J., Younger, H., and Bellenot, S. (1987), "Distributed Simulation and the Time Warp Operating System," *ACM Operating Systems Review 21*, 4, 77-93.

Presley, M., Ebling, M., Wieland, F., Jefferson, D. (1989), "Benchmarking the Time Warp Operating System With a Computer Network Simulation," In Proceedings of the SCS Multiconference on Distributed Simulation, Unger, B. and Fujimoto, R., Eds., Society For Computer Simulation, San Diego, CA, 8-13.

Wieland, F., Hawley, L., Feinberg, A., Di Loreto, M., Blume, L., Ruffles, J., Reiher, P., Beckman, B., Hontalas, P., Bellenot, S. (1989), "The Performance of a Distributed Combat Simulation With the Time Warp Operating System," *Concurrency: Practice and Experience 1*, 1, 35-50.