

# PROTEUS: A High-Performance Parallel-Architecture Simulator

Eric A. Brewer  
Adrian Colbrook

Chrysanthos N. Dellarocas  
William E. Weihl

Parallel Software Group\*, MIT Laboratory for Computer Science

## 1 Introduction

PROTEUS is an execution-driven simulator for MIMD machines. Like Tango [3] and RPPT [2], it directly executes most instructions to achieve very high performance. Despite exceptional speed PROTEUS produces accurate simulations and has reproduced several published results. Its modular structure allows a wide range of simulated architectures and simplifies customization. PROTEUS is a complete parallel-systems tool: it provides integrated support for data collection and display and for nonintrusive monitoring and debugging. Finally, PROTEUS allows users to tradeoff accuracy and performance to achieve the maximum performance for their accuracy requirements.

PROTEUS extends the use of code augmentation to provide stack-overflow detection, profiling, and explicit control of user code costs. Augmentation also prevents threads from getting too far ahead in simulated time, which increases the accuracy of the simulation.

A technical report on PROTEUS is available [1].

## 2 Versatility and Modularity

PROTEUS exploits its modular structure to simplify replacement and customization of specific aspects of the simulator. The modular structure provides two very important abilities. First, it allows users to tailor PROTEUS to a particular architecture. We have simulated both the nCUBE, a message-passing machine, and Alewife, a shared-memory multiprocessor. The modules control the network (or bus), cache protocols, runtime system, and memory attributes such as full/empty bits and atomic memory operations. It is simple to experiment with part of the architecture while keeping the rest unchanged.

Second, the modular structure promotes multiple implementations of a given module, which allows users to

\*Supported by the National Science Foundation, grant CCR-8716884; by DARPA, Contract N00014-89-J-1988; and by an equipment grant from Digital Equipment Corporation. Eric Brewer is supported by an Office of Naval Research Fellowship, Chris Dellarocas by a Starr Foundation Fellowship, and Adrian Colbrook by Science and Engineering Research Council Postdoctoral Fellowship. E-mail: brewer@lcs.mit.edu

	Network Model	
	Analytical	Hop-by-hop
Uniform Cost	1,500,000	700,000
No Caching	1,000,000	400,000
Coherent Cache	500,000	120,000

Numbers are in simulated cycles per second.

Table 1: The relative system performance of the six combinations of network and cache modules for the 8-queens application running on an 8x8 mesh. The simulations were run on a DECstation 5000.

switch between very accurate versions and very fast versions. The required accuracy of a module depends on the end goals of the simulation. For example, users studying scheduling require very accurate costs in the operating system module but may not need detailed network simulation.

The ability to switch among versions of a module provides a simple way to trade accuracy for performance. Table 1 shows the relative system performance for six combinations of cache and network modules. There is more than a ten-fold difference in performance between the least and most accurate. Most simulations achieve well over one million simulated cycles per second, since high accuracy is not usually required during development. Later, users can switch to more accurate modules *without modifying their code*.

## 3 Performance

PROTEUS has an overhead of about a factor of two for simulating sequential code, compared with a factor of several hundred for simulators that interpret each instruction. Overall, PROTEUS is 10 to 100 times faster than such simulators. With PROTEUS, only about 4% of the simulation time is spent in user code. The network and cache simulation require most of the time, which explains the effectiveness of the high-performance modules.

Unlike Tango [3], PROTEUS uses a custom lightweight-threads package that prevents context switches from dominating the simulation overhead. Context switching is reduced to about 4% of the overhead, which improves system performance by an order of magnitude.<sup>1</sup> The threads package performs *partial*

<sup>1</sup>A new version of Tango uses lightweight threads.

context switches if the switch occurs at a procedure call boundary. Invariants hold at procedure boundaries that limit the amount of state that must be saved. Typically, 98% of all context switches are partial.

## 4 Validation

Evidence for the accuracy of PROTEUS comes from several sources. PROTEUS reproduced published results [5] comparing sorting algorithms on an nCUBE/7. We used the original code, modified to use our runtime-system interface, and adjusted the simulated hardware costs to match those of the nCUBE/7. The timing results were within 3% for *every* data point. PROTEUS also reproduced published results [4] that compared locking algorithms on shared-memory machines (Sequent Symmetry, BBN Butterfly).

In general, any effect that we expected to see has actually appeared. Equally important is that all unexpected results have (so far) proven to be real effects rather than inaccuracies introduced by PROTEUS.

## 5 Monitoring and Debugging

A vital asset of PROTEUS is its support for monitoring and debugging. Real multiprocessors suffer from the *probe effect*: the addition of monitoring code may cause the monitored effect to disappear. This prevents programmers from collecting additional data for debugging. PROTEUS provides *nonintrusive* monitoring and debugging: users can add arbitrary debugging code without affecting the behavior or timing of the simulation. PROTEUS' deterministic nature and its support for nonintrusive debugging allow users to reproduce bugs despite the presence of additional debugging code.

Although deterministic, PROTEUS can reproduce multiple executions of a nondeterministic application, an ability unique to PROTEUS among multiprocessor simulators. PROTEUS chooses pseudo-randomly between two events with the same timestamp; such events are viewed as a race condition. Since the decision is pseudo-random it is deterministic, which ensures repeatability. At the same time, changing the seed affects the outcome of the race conditions and thus leads to a different execution of the nondeterministic application.

## 6 Data Collection and Display

PROTEUS also provides an integrated subsystem for data collection and display. PROTEUS' graphics capabilities make it simple to evaluate algorithms and architectures: users can create graphs quickly that answer their questions and provide new insight. The key is a simple but powerful graph language that specifies how to interpret the trace file.

In addition to several predefined trace-file data types, users can define their own, which allows users to generate high-quality application-specific graphs in

very little time. Typically, it takes only a few minutes to define a new data type and specify its interpretation using the graph language.

The graph generator produces line graphs, bar graphs, and tables, and can combine multiple graphs onto the same axes. It also can merge data from multiple simulations; this simplifies comparison of an algorithm across a range of architectures, machine sizes, or other architectural parameter. The generator uses the X Window system and produces POSTSCRIPT hard-copy.

## 7 Conclusion

PROTEUS provides a unique combination of flexibility, performance, and accuracy. Its modular structure simplifies customization and independent replacement of individual parts of the simulator; this promotes modules for specific architectures and multiple implementations to provide a variety of performance and accuracy combinations.

The accurate versions of modules allow PROTEUS to reproduce published results. The validation experiments provide a significantly increased level of confidence in PROTEUS' results.

PROTEUS ensures repeatability even in the presence of additional monitoring or debugging code, which allows users to track down bugs incrementally. In addition, nonintrusive monitoring supports both performance tuning and accurate data collection.

The data collection and display tools give PROTEUS an unusual level of effectiveness – users can collect and display exactly the data they need. The support for user-defined data types and user-specified graphs provides full access to the insight available through simulation.

The primary use of PROTEUS so far has been the design and implementation of a portable parallel language and runtime system. It also has supported research on concurrent algorithms, synchronization, operating systems, and fault tolerance. This work shows that it is effective for real applications: one study involved over 3000 simulations of a 10,000-line application.

- 
- [1] E. A. Brewer, C. N. Dellarocas, A. Colbrook, and W. E. Weihl. "PROTEUS: A High-Performance Parallel-Architecture Simulator." MIT Technical Report MIT/LCS/TR-516, Sept. 1991.
  - [2] R. G. Covington et al. "The Rice Parallel Processing Testbed." *Proceedings of the 1988 ACM SIGMETRICS Conference*, May 1988.
  - [3] H. Davis, S. R. Goldschmidt, J. Hennessy. "Multiprocessor Simulation and Tracing Using Tango." *Proceedings of ICPP '91*, Aug. 1991.
  - [4] J. M. Mellor-Crummey and M. L. Scott. "Synchronization without Contention." *Proceedings of ASPLOS-IV*, April 1991.
  - [5] M. J. Quinn. "Analysis and Benchmarking of Two Parallel Sorting Algorithms: Hyperquicksort and QuickMerge." *BIT*, 29(2):239-250, 1989.