REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 2001	~~03/99 - 03/01~~, Final Report 01 Mar 99 - 28 Feb 01

4. TITLE AND SUBTITLE

Enabling Technologies for Collaborative Simulators on Heterogeneous Networks.

5. FUNDING NUMBERS

~~ARO - MA 39415 - CI - RIP~~

DAAD19-99-1-0017

6. AUTHOR(S)

V. Rego

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Purdue University, W. Lafayette, IN 47906

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

U. S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

39415.5-CI-RIP

11. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12 a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12 b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This project deals with an experimental approach to developing enabling technology for the efficient support of collaborative simulation engines on heterogeneous networked platforms. The goal is to achieve collaborative operations of two kinds: cooperative operations between distinct but neighboring numerical solvers (simulation engines), and also collaboration between users who interact with neighboring solvers during model development, experimentation and production mode operation. To deliver efficient runtime support, we propose an architecture that reuses legacy components from existing tool suites, and also offers a methodology for developing new multithreaded simulation solvers. Our focus on efficient and portable runtime support departs from the traditional view of layering application software on top of existing communication libraries and/or protocols such as TCP/IP. We believe that most of the functionality and performance problems faced by such systems (e.g., scalability, multiprotocols, communicating threads, asynchronous low-latency, high throughput communication) are directly tied to architectural issues of runtime support and protocol layering. Our idea is to eliminate the layering that separates applications from protocols, and view an application as consisting of protocol threads and communication threads. We minimize --- and eventually, given direct access to secure network devices, propose to eliminate --- OS kernel involvement in communication. We emphasize software portability by exploiting portable user-space threads and communication systems, and adhering to strictly portable options in threads management and protocol design.

14. SUBJECT TERMS	15. NUMBER OF PAGES
multiprotocols, threads, solvers, kernel, simulation.	2
	16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT	18. SECURITY CLASSIFICATION ON THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-2[illegible]

Standard Form 298 (Rev.2-89)
Prescribed by ANSI Std. 239-18
298-102

20011113 145

MASTER COPY: PLEASE KEEP THIS "MEMORANDUM OF TRANSMITTAL" BLANK FOR REPRODUCTION PURPOSES. WHEN REPORTS ARE GENERATED UNDER THE ARO SPONSORSHIP, FORWARD A COMPLETED COPY OF THIS FORM WITH EACH REPORT SHIPMENT TO THE ARO. THIS WILL ASSURE PROPER IDENTIFICATION. NOT TO BE USED FOR INTERIM PROGRESS REPORTS; SEE PAGE 2 FOR INTERIM PROGRESS REPORT INSTRUCTIONS.

MEMORANDUM OF TRANSMITTAL

U.S. Army Research Office
ATTN: AMSRL-RO-BI (TR)
P.O. Box 12211
Research Triangle Park, NC 27709-2211

☐ Reprint (Orig + 2 copies)
☐ Technical Report (Orig + 2 copies)
☐ Manuscript (1 copy)
☒ Final Progress Report (Orig + 2 copies)
☐ Related Materials, Abstracts, Theses (1 copy)

CONTRACT/GRANT NUMBER: ARO-MA-39415-CI-RIP DAAD19-99-1-0017

REPORT TITLE: Enabling Technologies for Collaborative Simulators on Heterogeneous Networks

is forwarded for your information.

SUBMITTED FOR PUBLICATION TO (applicable only if report is manuscript):

Sincerely,

Vernon Rego

Based on our experience with software system development and our expertise with collaborative numerical solvers (simulation-engines), our objective in this projects is to develop a computational methodology and software environment that supports collaborative solvers. We utilized the funding in this project to purchase a high-end Sun Enterprise multiprocessor and three Dell workstations with wireless hubs. These machines are used, along with another Sun multiprocessor, to perform multiprocessor-cluster experiments. Our focus is on runtime support for collaboration between solvers, and runtime support for user-level collaboration between analysts (simulation engine developers and/or users) who interact with solvers on neighboring sub-domains. At the simulation-engine level, collaborative support implies efficient mechanisms for managing distributed threads and synchronization (API, functionality and efficiency issues) and scalability (threads, and ommunication). At the user-level, collaborative support implies efficient communication protocols (multiprotocol support, realtime, multiway, low-latency and high-throughput communication) that provide all the requisite efficiency and functionality for user-level collaborative interactions.

Borrowing from our experiences with CLAM (multithreaded protocols) and ParaSol (speculative, migrant-threads based distributed simulation), we propose an architecture for which the major portion of our research deals with issues at the lowest layer. Here, the Ariadne and Arachne threads systems offer threads functionality, and the Clam communication substem offers highly efficient communication functionality. The kernel layer exports Ariadne or Arachne threads to the solver layer, to new or legacy solvers like Ellpack, provides support for threads scheduling and process synchronization via speculative executions. We are currently experimenting with a methodology in which application layer code can be written without use of host ids, explicit send/receive or processor synchronization primitives, with appropriate help from the solver layer, object layer and kernel layer.

We have had considerable success with migrant-threads and object proxies, and in implementing optimistic and speculative executions in the ParaSol simulation system. We are currently using this methodology to synchronize processors, buidling functionality into the kernel layer. The object layer is an application-independent layer, but requires object definitions for every interface (boundary) in the problem space. Because these definitions depend on the data (problem) and not the solver, we are attempting to define them by class libraries which represent different boundary geometries. With this, the object layer is made completely independent of the data structures used by legacy systems, and also provides for object definitions in new multithreaded solvers. We initially plan to offer users a "bind_subdomain(A, hostid)" primitive that, at the start of execution, statically binds subdomain A to a solver running on proc (hostid). Since the object layer will be equipped with an object location mechanism, users are relieved of the responsibility of programming in terms of processor "ids". Once "bind_subdomain(A, hostid)" runs to establish the object-processor map, any invocation of object A by a thread automatically and transparently migrates the thread to processor hostid which hosts subdomain A. We have used this system very effectively in both Ariadne and also in ParaSol.

Because a collaborative solver must iterate and communicate with neighboring solvers in a chaotic way, communication is highly irregular. To enable such unpredictable communication patterns and yet enhance communication performance (i.e., by eliminating high polling costs which involve several layers of a communication protocol like TCP/IP), and to make speculative executions possible, all (interface) objects in the object layer must provide SAVE and RESTORE methods, so that they can be transparently checkpointed at appropriate times that are indexed by iteration indices. These checkpointed states will be (transparently) used to restore objects, whenever necessary, during speculative computations.

The SOLVER LAYER provides for either legacy or new solvers. Using envelopes, modular legacy solvers like EllPack can be implemented as three fat threads: (1).~an ''input" thread that a user can interact with for dynamic parameter modification, (2).~an ''output" thread which offers dynamic display of simulation results, and (3).~a ''solver" thread that iterates over its subdomain, given boundary values. New solvers can be fully designed in terms of an arbitrary number of threads, to maximize the potential benefits of design simplicity, concurrency on multiprocessors, and performance. For example, even with as few as three threads for legacy solvers like EllPack, Ariadne's time-slicing feature enables separate actions to be performed concurrently: exeution, output visualization and runtime modification of parameters. Depending on the legacy code, further concurrency and object-definition support for checkpointing and restoration (to offer more flexibility at the application-level) may be also be possible.

Publications

1. Multithreading Techniques and Applications, (in preparation) *J. Sang and V. Rego.*
2. Lazy Algorithms in Parallel Discrete Event Simulation, (in preparation), R. Pasquini and V. Rego.