DATA DISTRIBUTION MANAGEMENT FOR HIGH PERFORMANCE DISTRIBUTED SIMULATION IN RESOURCE-CONSTRAINT ENVIRONMENT

Pankaj Gupta and Ratan K. Guha
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816, U.S.A.
E-mail: {pgupta, guha}@cs.ucf.edu

KEYWORDS
DDM, HLA, Distributed Simulation, Resource Constraint Computing, Scalable Computing.

ABSTRACT

Data Distribution Management (DDM) is responsible in parallel and distribution simulation, especially in large-scale distributed modeling and simulation applications, for limiting and controlling the data exchanged and reducing the processing requirements of federates. In high-performance distributed simulation, system scalability can be seriously inhibited by limits on resources such communication bandwidth, memory, and CPU availability. In this paper, we present the design and implementation of a resource-efficient enhancement to the *P-Pruning algorithm* for data distribution management problem in High Level Architecture. We also present a performance evaluation study in a memory-constraint environment. The *Memory-Constraint* P-Pruning algorithm deploys I/O efficient data-structures for optimized memory access at run-time. The simulation results show that the *Memory-Constraint P-Pruning* DDM algorithm is faster than the P-Pruning algorithm and utilizes memory at run-time more efficiently. It is suitable for high performance distributed simulation applications as it improves the scalability of the P-Pruning algorithm by several order in terms of number of federates.

INTRODUCTION

Distributed simulation is a cost-effective technique for system studies in research, modeling, and training. The High Level Architecture (HLA) presents a framework for modeling and simulation within the Department of Defense (DoD). The goal of this architecture is to interoperate multiple simulations and facilitate the reuse of simulation components. HLA allows interconnection of simulations, devices, and human operators in a common federation. It builds on composability, letting designer construct simulations from pre-built components. Each computer-based simulation system is called a *federate* and the group of interoperating systems is called a *federation*. HLA specifications—incorporated as IEEE 1516 standard—were developed to provide reusability and interoperability.

The HLA Run-Time Infrastructure (RTI) provides a set of services used to interconnect simulation during a federation execution. These RTI services are grouped into six categories: federation management, declaration management, object management, ownership management, time management, and data distribution management. RTI provides a degree of portability and simulation interoperability. It allows federates to join and resign, declare their intent to publish information, send information about objects, attributes and interaction, and synchronize time. In distributed simulation environment, every action taking place on a simulator that may affect or may be of interest to another simulator, requires a message. In a large-scale distributed simulation, such as those encountered in high performance applications, simulating many objects that are of interest to other objects can result in increased communication across the network, on the scale of $O(n^2)$. DDM is responsible for limiting and controlling the data exchanged in a simulation. It also aims at reducing the processing requirements of simulation hosts, or federates, by communicating updates regarding interactions and state information only to federates that require them.

This paper presents the design and implementation of a resource-constraint enhancement to the *P-Pruning algorithm* for the DDM region matching problem. It also presents a performance evaluation study in a memory-constraint simulated environment. The simulation results show that *Memory-Constraint P-Pruning* algorithm improves the scalability of P-Pruning algorithm by several order in terms of number of federates and overlapping interest regions in the distributed simulation. We use IEEE 1516 specifications for representing the federates, and their publisher and subscriber regions. The rest of the paper is organized as follows. The next section provides a background on the importance of data distribution techniques, concepts of routing space and memory as a resource, and a review of related work in DDM research. The P-Pruning algorithm for DDM matching problem with its three sub-procedures is presented in the following section. Later, the resource constraint issues in data distribution management are highlighted and a memory-efficient enhancement in P-Pruning algorithm is proposed. After this, the performance evaluation implementation details and simulation results are discussed. The last section presents the concluding remarks with directions on future research work.

BACKGROUND

In this section, we highlight the importance of DDM, explain the concept of routing space, and introduce the notion of memory as a resource in high-performance distributed simulation. We also provide a review of related work in DDM and memory-constraint algorithms.

Concept of Routing Space

DDM is based on a multi-dimensional coordinate system called a routing space. For example, a two-dimensional routing space might represent the play box in a virtual environment. A rectangular publisher region within the routing space is associated with each update message generated by a publishing federate. Receiving federates declare their interests via rectangular subscriber regions within the routing space. If the publisher region associated with a message overlaps with the subscriber region of a federate, the message is routed to that subscribing federate. By calculating the intersection of publisher and subscriber regions, the Run-Time Infrastructure in HLA establishes connectivity between sender and receiver federates for routing updates and interactions. Each overlapping subscriber and publisher federate joins a multicast group to facilitate the message transfer. For example, in Figure 1, updates using publisher region *P* are routed to federates subscribing to region S_1, but not to federates subscribing to region S_2.

Memory as a Resource

A typical access time to internal main memory (RAM) is in the order of nano-seconds, while access time to external memory (such as hard disk) is in the order of milliseconds. Thus, the access times of internal and external memory differ by a factor of million. In many large-scale distributed simulation applications, the communication between internal and external memory, and not the internal computation time, is actually bottleneck in the computation. Also, as the application size is scaled, the Input/Output (I/O) requirements can lead to serious memory crunch. Modern operating systems use sophisticated paging and data pre-fetching strategies to minimize the effect of I/O bottleneck and ensure that the accessed data is present in the internal memory. However, these strategies are general in nature and cannot exploit the properties of a specific problem. Hence, we need to design solutions which consider a memory of limited size.

In memory-constraint approach, we view the system memory as a resource that has to be optimally allocated among the processes. The problem is how to deploy efficient data-structures and reorganize the data at run-time so that the DDM computation is not as memory intensive as encountered in practical simulations. I/O efficient data-structures are the key tools in developing a resource-efficient approach. Also, dynamic memory-management strategy that provides efficient garbage

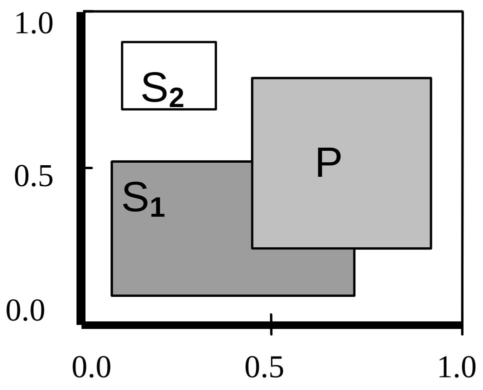

Figure 1: Two-dimensional routing space with subscriber regions: S_1 and S_2, and a publisher region *P*

collection to reduce unnecessary memory leak at run-time is crucial. The primary motivation in the resource-constraint approach is to devise a scalable, memory-efficient solution for high-performance distributed simulation applications.

Related Work

The earliest work on DDM research appears in Van Hook et al. 1996. The HLA specification, key elements in its architecture, and implementation are described in Dahmann et al. 1998. An overview about DDM and related research work appears in (Guha and Bassiouni 2002; Gupta and Guha 2005; Gupta and Guha 2006). Since 1995, different DDM algorithms have been proposed such as the fixed-grid (Tan et al. 2000a), dynamic-grid (Boukerche and Dzermajko 2004), region-matching (Van Hook et al. 1996), agent-based (Tan et al. 2001), and hybrid-method (Tan et al. 2000b). Performance-evaluation study of different DDM strategies appear in (Boukerche and Dzermajko 2004; Gupta and Guha 2005).

Federated Simulations Development Kit (FDK) is an implementation of HLA architecture developed at Georgia Institute of Technology. FDK has been used as the platform for HLA-based distributed simulation research in (Fujimoto et al. 2000; Gupta and Guha 2005). Advanced memory management schemes such as hierarchical data-caching and pre-fetching that can be applicable to resource constraint conditions related to DDM appear in Wang and Guha 2001. A review of I/O efficient external memory data-structures appears in Arge et al. 2001. Memory-efficient routines and implementation details appear in Blunden 2003, while Meyer et al. 2003 provide a good source of algorithms for memory hierarchies. Directions on I/O efficient algorithms and dynamic memory allocation in simulation appear in (Nielsen 1977; Vengroff and Vitter 1996).

THE P-PRUNING ALGORITHM FOR DDM

Given a set of federates, *F*, each federate F_i has publisher and subscriber regions within the routing space. The DDM problem is to find a set of multicast groups *MCG*, whose each element is a subset of *F* at any time *t*. Each multicast group member, MCG_i, is composed of federates whose publisher region overlaps with the subscriber region of federates in MCG_i at time *t*. In this section, we describe the P-Pruning DDM algorithm (Gupta and Guha 2005) which computes the multicast groups in three steps: List Computation, MCG Population, and MCG Pruning. The central idea in this algorithm is to compute the matching of publisher and subscriber regions first on the basis of the overlap on X-axis only and then correct the overlap information by checking the Y-axis. Each publisher and subscriber region is described by four-coordinate system in the routing space $[((P_i)_{X1},(P_i)_{X2}),((P_i)_{Y_1},(P_i)_{Y_2})]$ and $[((S_i)_{X1},(S_i)_{X2}),((S_i)_{Y_1},(S_i)_{Y_2})]$, respectively. The entire algorithm is based on an array, ListX, whose size is equal to *R*, *i.e.*, the length of the routing space X-axis. The elements in ListX array correspond to the coordinates in X-axis of the routing space.

List Computation: The List Computation sub-procedure scans the publisher and subscriber regions of all the federates once, and stores the information about their coordinates at each point of the ListX array. At the end of the List Computation procedure, each point of ListX array has three entries: publisher region, X1 subscriber region, and X2 subscriber region. For any element (or point) *x* of ListX array, the publisher region entry corresponds to all the publisher regions P_i, whose $(P_i)_{X_1}$ coordinate coincides with point *x*. The X1 subscriber region entry corresponds to all subscriber regions S_j, whose $(S_j)_{X_1}$ coordinate coincides with point *x* and X2 subscriber region entry corresponds to the all the subscriber regions S_j, whose $(S_j)_{X_2}$ coordinate coincides with point *x*.

MCG Population: The MCG Population sub-procedure scans each element of the ListX array and checks the coordinates of publisher region, X1 subscriber region, and X2 subscriber region for overlap condition. This procedure creates the DDM multicast groups based on information stored in ListX array, but it considers only the overlap information on X-dimension of the routing space. A multicast group is assigned to an element of the ListX array, if there is a publisher region P_i whose $(P_i)_{X_1}$ coordinate coincides with this element of ListX. Also, a multicast group is created at a point on ListX only if there exists at least one publisher region at this point and there is at least one subscriber region overlapping with this publisher region on X-axis. When a region is included in a multicast group, it implies that the federate owning this particular region is also actually a member of this multicast group. Thus, this procedure creates a set of multicast groups that may include some multicast groups that are having publisher and subscriber region as members, but these member regions may not actually overlap on the Y-axis of the routing space. Overall, this procedure computes the entire information faster by avoiding the simultaneous checking of X-axis and Y-axis overlap.

MCG Pruning: The errors in creation of multicast group *MCG* are now corrected by the final MCG Pruning sub-procedure. The pruning sub-procedure verifies that the regions in multicast group *MCG* actually overlap on Y-axis, and it eliminates any non-overlapping subscriber from the specific multicast group. It also verifies that every multicast group has at least one subscriber region after this step. At the end of this process, it deletes any multicast group having no subscriber region.

Since the two main sub-procedures in the algorithm perform the function of multicast group *Population* and *Pruning*, this algorithm is called *P-Pruning* algorithm. A detailed description of the P-Pruning algorithm with pseudo-codes, performance evaluation results, and average-case computational complexity appears in Gupta and Guha 2005. We have shown that the P-Pruning algorithm is efficient as compared to three other DDM algorithms in term of computation time, memory usage at run-time and number of multicast groups using average-case analysis and extensive performance evaluation simulations.

Computation Results and Analysis

The List Computation sub-procedure has complexity of O(n), where n is the number of federates in the distributed simulation. The MCG Computation sub-procedure runs for between O(n) and O(n^2) depending on the density of the regions within the routing space. For the MCG Pruning sub-procedure, complexity is O(n) times. The total number of multicast groups in this algorithm is limited by O(n), which is significantly lesser than other DDM algorithms. In the simulation experiments, we generated federates with publisher and subscriber regions, whose coordinates were randomly distributed within the routing space. The simulation results shown in the graphs of Figures 2 and 3 use the data averaged for 25 instances of each condition, *i.e.*, size of routing space and number of federates. The grid cell dimensions are applicable only for the fixed-grid and dynamic-grid algorithms. Each set of input conditions had 10, 20, 30, 40, and 50 federates in the simulation environment with 50 x 50 routing space. The graph in Figure 2 show the comparison of computation time required by P-Pruning, region-matching, fixed-grid, and dynamic-grid DDM algorithms (Gupta and Guha 2005) for finding multicast groups. The graph in Figure 3 show the comparison of memory utilized at run-time for the four DDM algorithms in similar setting. The results show that the P-Pruning DDM algorithm computes the multicast groups faster than any of the three algorithms and also uses system memory more efficiently. The P-Pruning algorithm is faster than region-matching, fixed-grid, and dynamic-grid DDM algorithms as it avoid the quadratic computation step involved in these algorithms. By populating the

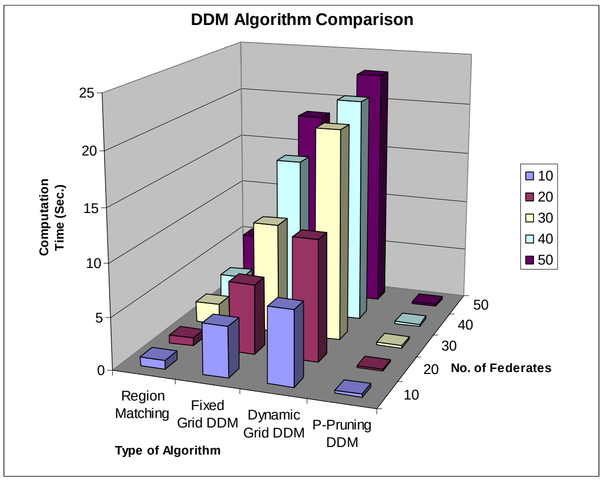

Figure 2: Performance Evaluation of P-Pruning Algorithm for Routing Space 50 x50 and Grid Size 2 x 2

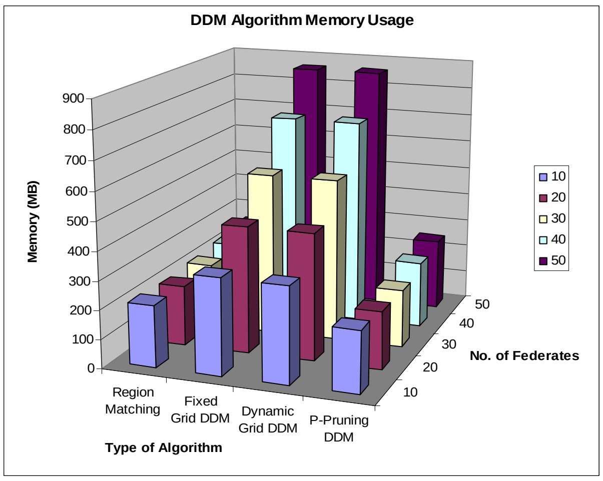

Figure 3: Comparison of Memory Usage by DDM Algorithms for Routing Space 50 x 50 and Grid Size 2 x 2

multicast group first only on the basis of X-axis information and pruning the multicast group of unwanted subscriber regions in another step, it avoids the computational overheads of other algorithms.

I/O EFFICIENT RESOURCE-CONSTRAINT STRATEGY FOR DDM

In this section, we explore the resource-constraint issues in the DDM algorithms and present a memory-efficient enhancement to the P-Pruning algorithm.

Resource-Constraint Issues in DDM Implementation

In Gupta and Guha 2005, we compared the performance of P-Pruning algorithm with region-matching, fixed-grid and dynamic-grid DDM algorithm through simulation studies. During the simulation experiments, it was observed that the performance of DDM algorithms is adversely affected as the number of federates is increased in the simulation environment. In practice, system scalability can be seriously inhibited by limits on bandwidth and computation. While this is not totally unexpected; for a DDM algorithm to be effective and deployable in high performance modeling and simulation applications, it must be scalable. In general, the performance of all DDM algorithms is severely affected by limitations in system resources such as communication bandwidth, memory, and CPU availability. Hence, we have considered the system memory as a resource in this research. In practical distributed simulation applications, the designers should deploy efficient data structures to achieve the dual goal of reducing computation time and memory utilization.

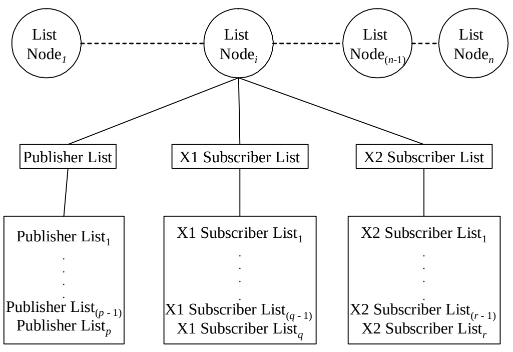

Figure 4: Representation of Memory-Efficient Data Structure

A Memory-Efficient Strategy for Data Distribution Management

The P-Pruning algorithm is not resource-efficient because it does not conserve memory. In a resource-constraint environment, the system memory is limited and special routines are needed for developing scalable solutions. We now present a memory-efficient enhancement to the P-Pruning algorithm. We consider the system memory as a resource and modify the P-Pruning algorithm for optimal utilization of this resource. In memory-efficient P-Pruning algorithm, the List Computation sub-procedure is modified by incorporating a resource-efficient data structure. We define a node which maintains three different types of lists: Publisher region list, X1 subscriber region list, and X2 subscriber region list. The set of node is represented as list which replaces the ListX array in the List Computation sub-procedure. The set of nodes can be viewed as disjoint set of forests, where each node stores three different trees. This representation reduces the memory allocated at run-time significantly for the DDM computation and also improves the computation time as evident from the performance evaluation results.

Data Structure Design: The structure of each node in the disjoint set is shown in Figure 4. There are n nodes in the list, and each node maintains three different lists of size p, q, and r. Here, p = number of publisher regions; q = number of X1 subscriber regions; and r = number of X2 subscriber regions. The list node is represented using the class structure shown in Figure 5. The three lists in disjoint set are populated in List

```
class List_Node
{ public:
            vector<Region> Pub_Region;
            vector<Region> X1_Sub_Region;
            vector<Region> X2_Sub_Region;

            List_Node();
            ~List_Node();
};

vector<List_Node>  X_List(X_Axis_Length);
```

Figure 5: Class Structure to Represent the Disjoint Set of Forest

Computation sub-procedure and the multicast groups are built using the disjoint set in the MCG population sub-procedure. Using the new structure, we can reduce the memory allocated at run-time and reduce the access time during computations.

PERFORMANCE EVALUATION OF RESOURCE-CONSTRAINT P-PRUNING ALGORITHM

In this section, we describe the performance evaluation of the P-Pruning DDM and *Memory-Constraint* P-Pruning algorithm. The study was aimed at modeling high-performance distributed simulation scenario and implemented in C++ under Windows XP running on a Pentium IV 3 GHz PC with 512 MB RAM and 2500 MB virtual memory. We used object-oriented class structures to represent federates, their publisher and subscriber regions, the grid cells, and the multicast groups.

Simulation Implementation and Analysis

In the simulation experiments, we generated federates with publisher and subscriber regions, whose coordinates were randomly distributed within the routing space. Each federate F_i has one publisher region P_i and one subscriber region S_i. The simulation results shown in Figures 6, 7, and 8 use the data averaged for 100 instances of each condition, *i.e.*, size of routing space and number of federates. The graph in Figure 6 shows the comparison of memory utilized at run-time by the *Memory-Constraint* P-Pruning and P-Pruning algorithms for distributed simulation having routing space of 4,000 x 4,000 and number of federates ranging from 100 to 4,000. Figure 7 shows the comparison of computation time required for the *Memory-Constraint* P-Pruning and conventional P-Pruning implementation for the similar range of routing space and number of federates in simulation environment. It is evident from these graphs that the *Memory-Constraint* version uses constant memory as compared to the P-Pruning algorithm. It also requires less computation time. The graph in Figure 8 shows the memory utilization for the routing space upto 20000 x 20000 and the number of federates ranging from 100 to 20,000. This result demonstrates the scalable nature of *Memory-Constraint* P-Pruning algorithm. The P-Pruning algorithm could simulate only upto 4,000 federates due to inefficient memory utilization at run-time.

From the performance evaluation study of two versions of the P-Pruning DDM algorithm, it is clear that the *Memory-Constraint* P-Pruning DDM algorithm provides the region overlapping information efficiently with respect to important metrics: computation time and memory usage at run-time. The list of disjoint forests minimizes I/O requirements and optimizes memory access at run-time.

CONCLUSIONS AND FUTURE WORK

In this paper, we presented the design and performance evaluation of a resource-efficient enhancement to the P-Pruning algorithm for DDM. By deploying efficient data structures, the *Memory-Constraint* P-Pruning DDM algorithm scales well in high performance distributed-simulation environment. The *Memory-Constraint* P-Pruning algorithm requires less

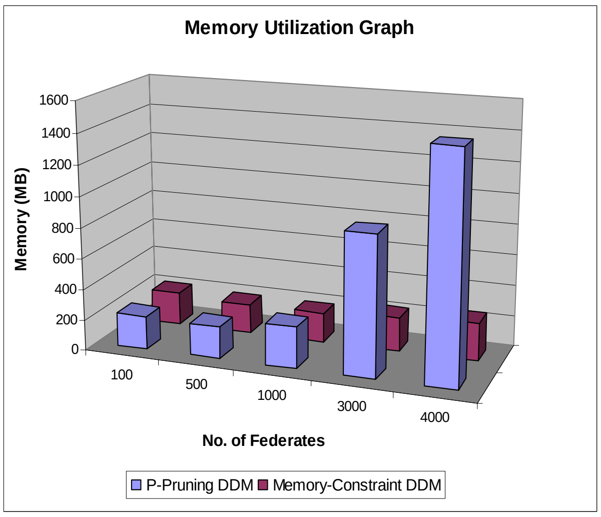

Figure 6: Comparison of Memory Utilization by the Memory-Constraint and P-Pruning DDM Algorithms

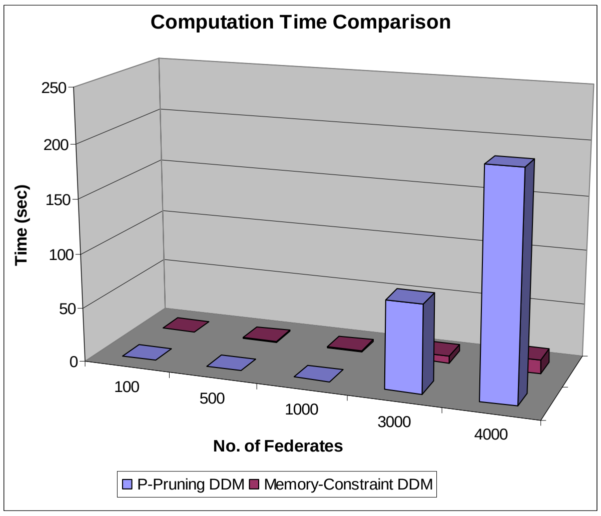

Figure 7: Comparison of Computation Time for Routing Space from 100 x 100 to 4000 x 4000

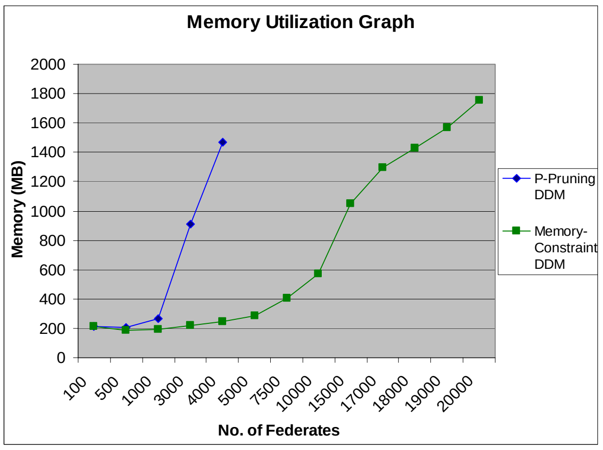

Figure 8: Memory Utilization for Distributed Simulation with 20,000 Federates

computation time and utilizes memory at run-time more efficiently as compared to the P-Pruning algorithm. In the near future, we plan to extend our work to a distributed DDM algorithm implementation on cluster computers, incorporate resources such as communication bandwidth in the resource-constraint analysis, and investigate the scalability issues. The distributed implementation of P-Pruning algorithm will provide a scalable and resource-efficient DDM approach.

ACKNOWLEDGEMENTS

The authors would like to thank the reviewers for their valuable comments. This work was supported by ARO under grant DAAD19-01-1-0502. The views and conclusions herein are those of the authors and do not represent the official policies of the funding agencies or the University of Central Florida, Orlando.

REFERENCES

Arge, L.; L. Toma; and J. Vitter. 2001. "I/O-Efficient Algorithms for Problems on Grid-based Terrains". ALENEX' 00. *ACM Journal of Experimental Algorithmics*, 6, No. 1.

Blunden, B. 2003. "Memory Management: Algorithms and Implementation in C/C++". Plano, TX. Wordware Publishing, 2003.

Boukerche, A. and C. Dzermajko. 2004. "Performance evaluation of Data Distribution Management strategies", *Concurrency and Computation: Practice and Experience*, 16, No. 15, 1545-1573.

Dahmann, J. S.; R.M. Fujimoto; and R.M. Weatherly. 1998. "The DoD High Level Architecture: an Update". *In Proceedings of the 1998 Winter Simulation Conference*, Washington DC.

FDK—Federated simulations development kit. http://www.cc.gatech.edu/computing/pads/fdk.html

Fujimoto, R.; T. McLean, K. Perumalla; and I. Tacic. 2000. "Design of high-performance RTI software". In Proceedings of Distributed Simulations and Real- time Applications (DS-RT), San Francisco, CA.

Guha, R. K. and Bassiouni M. 2002. "Simulation Methods and Applications". *Simulation Practice and Theory*, 9, No. 3-5, 91-93.

Gupta, P. and R. K. Guha. 2005. "Design, Analysis, and Performance Evaluation of an Efficient Algorithm for Data Distribution Management in High Level Architecture." Computer Science Technical Report CS-TR-05-12, School of Computer Science, University of Central Florida, Orlando, FL, December 2005.

Gupta, P. and R. K. Guha. 2006. "Design and Implementation of an Efficient Algorithm for Data Distribution Management in High Level Architecture". In *Proceedings of the 4th Symposium on Design, Analysis, and Simulation of Distributed Systems*, 2006 Spring Simulation Multiconference, Huntsville, Alabama, April 2-6, 2006.

Meyer, U.; P. Sanders; and J. Sibeyn. 2003. *Algorithms for Memory Hierarchies*. Springer-Verlag, Berlin.

Nielsen, N.R. 1977. "Dynamic memory allocation in computer simulation". *Communications of the ACM*, 20, No. 11, 864-873.

Vengroff, D. E. and Vitter, J. S., 1996. "I/O-Efficient Algorithms and Environments. Strategic Directions in Computing Research". *ACM Computing Surveys*, 28A, No. 4.

Tan, G.; R. Ayani; Y.S. Zhang; and F. Moradi. 2000. "Grid-based data management in distributed simulation". In *Proceedings of the 33rd Annual Simulation Symposium*, Washington DC, U.S.A. 16-20 April, 7-13.

Tan, G.; Y. Zhang, Y.; and R. Ayani. 2000. "A hybrid approach to data distribution management", In *Proceedings of the Fourth IEEE International Workshop on Distributed Simulation and Real-Time Applications*, San Francisco, CA, 25-17 August, 55-61.

Tan, G.; X. Liang; F. Moradi; and S. Taylor. 2001. "An agent-based DDM for High Level Architecture", In *Proceedings of the 15th Workshop on Parallel and Distributed Simulation*, 15-18 May, 75-82.

Van Hook, D. J.; S.J. Rak; and J.O. Calvin. 1996. "Approaches to RTI Implementation of HLA Data Distribution Management Services". *Fifteenth Workshop on Standards for the Interoperability of Distributed Simulations*, 96-14-084, September 16-20, 1996.

Wang, J. Z. and R.K. Guha. 2001. "A novel data caching scheme for multimedia servers". *Simulation Practice and Theory*, 9, No. 3-5, 193-213.

AUTHOR BIOGRAPHIES

Pankaj Gupta is a Ph.D. candidate and graduate research assistant in the School of Computer Science at the University of Central Florida (UCF). His research interests include distributed simulations, parallel computing, graph theory and discrete optimization. He was recipient of the 2001 Graduate Merit Scholarship from the Office of Graduate Studies, UCF.

Ratan K. Guha received the Ph.D. degree in computer science from the University of Texas at Austin in 1970. He is currently a professor of computer science at the University of Central Florida, Orlando. His research interests include distributed systems, parallel and distributed simulation, computer networks, security protocols, modeling and simulation, and computer graphics. He is a member of the ACM, IEEE, IEEE Computer Society, and SCS and is currently serving as a member of the Board of Directors of SCS.