An Experimental Analysis Environment for Logical Process Simulation Algorithms

Bing Wang
School of Computer Science
National University of Defense Technology
410073, Changsha, P.R. China
bw107@informatik.uni-rostock.de

Jan Himmelspach
Institute of Computer Science
Joachim Jungius Str. 10
18059 Rostock, Germany
jh194@informatik.uni-rostock.de

Roland Ewald
Institute of Computer Science
Joachim Jungius Str. 10
18059 Rostock, Germany
re027@informatik.uni-rostock.de

Yiping Yao
School of Computer Science
National University of Defense Technology
410073, Changsha, P.R. China
ypyao1@tom.com

Adelinde M. Uhrmacher
Institute of Computer Science
Joachim Jungius Str. 10
18059 Rostock, Germany
lin@informatik.uni-rostock.de

ABSTRACT

The notion of *logical processes* (LPs) is a widely used modeling paradigm in parallel and distributed discrete-event simulation (PDES). Nevertheless the comparison among different simulation algorithms for LP models still remains difficult: there are too many combinations of algorithms to be explored, often simulation systems only provide a small subset of available algorithms, and many m&s frameworks blur the boundary between model logic and simulation algorithm, which hampers extensibility and comparability. We present an environment for the experimental analysis of simulation algorithms for logical processes. It separates between model and simulator, is extensible, and facilitates a fair comparison of algorithms. We illustrate the functioning of the environment by presenting experimental results for well-known simulation algorithms and a benchmark model.

Categories and Subject Descriptors

I.6.7 [**Simulation Support Systems**]: Environments; I.6.8 [**Types of Simulation**]: Discrete-event,parallel,distributed; D.2.8 [**Metrics**]: Performance measures

Keywords

Parallel and Distributed Simulation, Discrete-Event Simulation, Logical Process, Performance Analysis

1. INTRODUCTION

Many algorithms have been proposed in modeling and simulation (m&s) during the last decades, but most of them have merely been realized in the m&s systems created around them [8]. In addition, many simulation systems do not separate clearly between model and simulator. This leads to a questionable mixture, which cannot guarantee that the model is a valid representation of the system under study, or that the simulation algorithm works correctly for different models. From the view of the "Experimental Algorithmics" community [7], this makes a fair comparison of them difficult or even impossible. A thorough and reliable evaluation of the existing solutions is therefore highly demanded, at best in an open environment which allows others to reproduce the obtained results. This should also help to overcome the various pitfalls one encounters when analyzing algorithms empirically (e.g., cf. [6, 4]). One of the paradigms to model discrete event-based systems is the *logical process* (LP) paradigm.

SIMUTools 2009 Rome, Italy

2. AN EXTENSIBLE FRAMEWORK FOR LP SIMULATION ALGORITHMS

The term of *logical process* (LP) was introduced as a metaphor for the physical processes a system consists of [2]. LPs are the core units of execution in most parallel discrete-event simulations. How these events are transferred and how processes are triggered to execute them depends on the synchronization protocol (the simulation algorithm). As we separate model and simulator, the simulation algorithms, realizing the different synchronization schemes, are "exchangeable entities". In addition, there is a large variety of auxiliary algorithms that can be combined with the simulation algorithms, e.g., event queues, partitioning methods, or algorithms for load balancing.

Figure 1 outlines some central entities in the experimental analysis environment. The set of LPs that defines a model is represented by `IScenario`. As JAMES II [5] requires a processor to allow stopping, pausing, and resuming a simulation, the PDES simulators consist of two components: a *protocol simulator*, which implements all operations demanded by JAMES II, and the actual simulators for LPs, which get notified by the protocol simulator when they shall

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5678
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5678

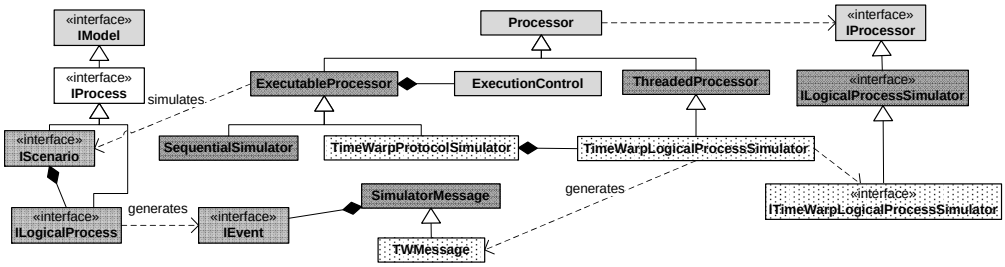

Figure 1: The framework of the experimentation environment: Based on the James II core (grey), we developed separate classes and interfaces for models (horizontal lines) and backbone classes for PDES algorithms (diagonal lines). As an example, the central classes for Time Warp are outlined (dots).

stop or resume. Many central entities are expressed by Java interfaces (see Figure 1) and thus do not provide any logic. This allows to replace almost every aspect of this environment with alternatives.

3. EXPERIMENTAL ANALYSIS

Each algorithm that shall be analyzed experimentally demands careful realization and thorough validation with "good" problem instances (models) [6]. Fortunately, there are several well-known benchmark models that we can use for a first analysis of PDES algorithm performance. One of those, the PHOLD model [3], consists of an arbitrary number of LPs, each of which sends an event to a neighbor when an event is received. The simulation runs have been executed on a Windows XP 64 workstation with two 2.5 GHz QuadCore Xeon Processors and 8 GB of RAM (Java Scimark 2.0 [1] result of 765.3 points, JAMES II v 0.6 and Sun's JRE 1.6.07 for 64-bit Win). The PDES algorithms were only tested on this multi-core machine, using a single virtual machine - instead of executing them on a set of hosts that communicate over a network connection.

Hence, communication costs are low in comparison to classical PDES experiments and multiple hosts. This moves the focus of the experiments towards the question of how large the protocol-induced overhead in relation to the speed-up by using multiple cores really is – a use case that might become more and more important, as the number of cores per CPU is likely to increase over the next years. Several PHOLD topologies were tested: *full* denotes a complete LP graph, i.e., each LP has all other LPs as neighbors, *grid* is a two-dimensional toroidal grid, and in *ring* all LPs form a ring, so that each LP has only two neighbors (tightly, medium, and sparsely connected models). We chose PHOLD model size m from $\{4, 16, 36, 64, 100, 144, 196\}$. The number of concurrent PHOLD events was set to $\frac{1}{2}m$. While the execution times of the barrier synchronization protocol are almost identical for all three topologies (Fig. 2), the performance of null message synchronization clearly depends on the interconnectedness of the LPs. In case of a fully connected graph, its performance suffers from the overhead of sending null messages to all neighbors.

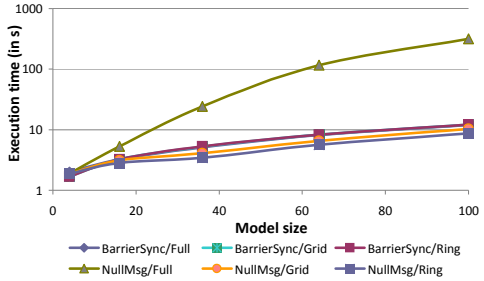

Figure 2: Performance of synchronization schemes on different PHOLD topologies.

4. CONCLUSIONS

We introduced an environment for the development and experimental analysis of algorithms for the LP paradigm. Future research will be to develop and evaluate further simulation algorithms, and the interplay with partitioning and load balancing schemes shall be explored carefully. For the framework to grow into a research resource for distributed simulation algorithms, potential evaluation schemes, and performance data alike, we invite other people to join us: to repeat our findings, to add new algorithms, and to evaluate these algorithms (`http://www.jamesii.org/`).

5. REFERENCES

[1] Java scimark: http://math.nist.gov/scimark2/.

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

[3] R. M. Fujimoto. Performance of Time Warp under synthetic workloads. In *Proc. of the SCS Multiconf. on Distributed Simulation*, pages 23–28, 1990.

[4] I. P. Gent, S. A. Grant, E. MacIntyre, P. Prosser, P. Shaw, B. M. Smith, and T. Walsh. How not to do it. Technical report, University of Leeds, May 1997.

[5] J. Himmelspach and A. M. Uhrmacher. Plug'n simulate. In *Proceedings of the Spring Simulation Multiconf.*, pages 137–143. IEEE Computer Society, Mar. 2007.

[6] D. Johnson. A theoretician's guide to the experimental analysis of algorithms. In *Fifth and Sixth DIMACS Implentation Challenges*, 2002.

[7] C. McGeoch. Experimental algorithmics. *Comm. of the ACM*, 50(11):27–31, November 2007.

[8] K. Perumalla. μsik: A micro-kernel for parallel/distributed simulation systems. In *Workshop on Parallel and Distributed Simulation*, pages 59–68, Monterey, CA, June 2005. IEEE.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5678
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5678