УДК 681.3.06

А. А. Александров

Институт динамики систем и теории управления СО РАН
ул. Лермонтова, 134, Иркутск, 664033, Россия

E-mail: elate@mail.ru

РАСПРЕДЕЛЕННОЕ ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕ: ТЕХНОЛОГИИ, МЕТОДЫ, СРЕДСТВА

В статье приводится обзор технологий, методов и средств организации распределенного имитационного моделирования. Предлагается авторский подход к моделированию вычислительных систем, основанный на использовании высокоуровневых систем GPSS/World, GPSS/H и разработанной коммуникационной библиотеки, обеспечивающей взаимодействие между распределенными сегментами GPSS-модели.

Ключевые слова: имитационное моделирование, распределенная GPSS-модель.

Введение

Развитие информационных технологий и постоянный рост требований, исходящих от научно-исследовательских приложений, обусловили создание новых вычислительных инфраструктур, которые предназначены для решения целого класса задач, не предполагающих тесного взаимодействия между параллельными процессами. Использование предоставляемых возможностей высокопроизводительной вычислительной техники требует подготовки высококвалифицированных специалистов, способных применять перечисленные выше средства для решения прикладных задач. Одной из таких задач является исследование и анализ сложных систем (экономических, производственных, вычислительных и др.). В каждый момент времени состояние подобной системы может зависеть от большого числа управляющих переменных, случайных событий, связей между объектами системы и ограничений разных типов. Исследование подобных систем аналитическим путем зачастую становится невозможным. Поэтому возникает необходимость в применении средств и методов имитационного моделирования [4]. Применение статистических методов, неизбежное при имитационном моделировании, требует больших затрат машинного времени и вычислительных ресурсов [2]. Одним из путей решения данной проблемы является использование суперкомпьютерной вычислительной техники, в том числе кластерных архитектур (рис. 1).

В этом случае встает вопрос об использовании параллельного или распределенного имитационного моделирования. В литературе (см., например, [12]) эти два направления различают следующим образом: использование симметричных мультипроцессоров (SMP) или массово параллельных систем (MPP) определяется как параллельное моделирование, а проведение исследований в локальных или глобальных вычислительных сетях (которые могут включать в качестве узлов SMP или MPP машины) обычно принимается за распределенное моделирование. В параллельном моделировании главным вопросом является ускорение процесса решения задачи. В распределенном моделировании больше внимания уделяется проблемам взаимодействия между различными программно-аппаратными средствами, повторного использования библиотек моделей и выполнения многовариантных расчетов.

В работе рассматриваются существующие технологии организации распределенного имитационного моделирования, анализируются их преимущества и недостатки, предлагаются специализированные методы и средства моделирования вычислительных систем.

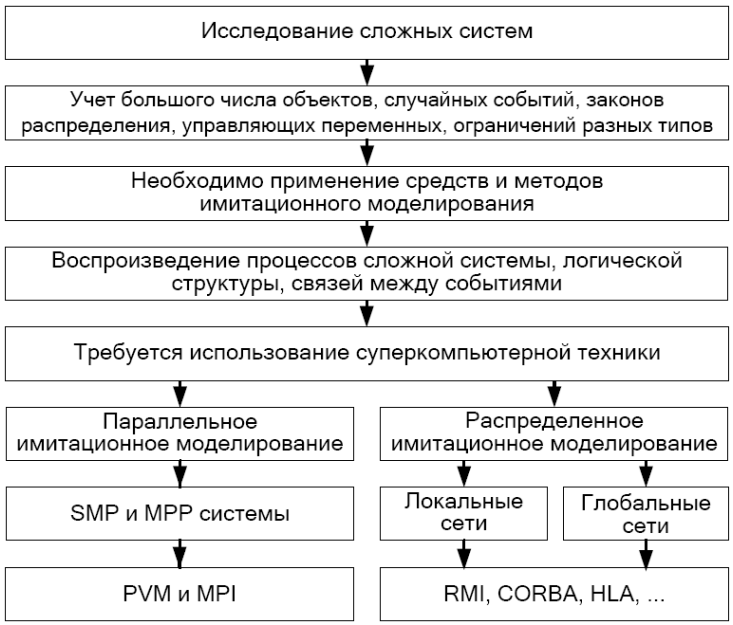

Рис. 1. Схема использования суперкомпьютерной вычислительной техники при имитационном моделировании сложных систем

1. Системы имитационного моделирования

Применение современной высокопроизводительной техники играет существенную роль в исследовании свойств и изучении поведения моделей. В связи с этим успешность таких исследований напрямую связана с выбором языка для реализации алгоритмов модели. Поэтому большое значение играет правильность выбора такого языка моделирования или даже среды, включающей в себя соответствующий язык [7]. Основные преимущества языков имитационного моделирования по сравнению с языками общего назначения заключаются в удобстве программирования при реализации модели системы и концептуальной направленности языка моделирования на определенный класс систем (например, системы массового обслуживания). В литературе [2] выделяют обширное количество различных систем имитационного моделирования, нацеленных на различные классы задач и имеющих как специализированные языки [3], так и вспомогательные средства построения моделей.

Системы имитационного моделирования в сравнении с универсальными языками программирования дают ряд преимуществ: обеспечивают исследователя естественной средой для создания имитационных моделей и предоставляют такие возможности, как генерирование случайных чисел с заданным распределением вероятности, продвижение модельного времени, обработка списков текущих и будущих событий и др.; имеют встроенные механизмы представления параллельных процессов; имитационные модели, созданные с помощью систем моделирования, как правило, проще модифицировать и использовать.

Основными требованиями при выборе системы моделирования, с учетом ее функционирования в распределенной вычислительной среде, являются ее отказоустойчивость, синхронизация времени и возможность организации взаимодействия сегментов модели между собой в процессе моделирования [12].

Системы, которые предлагают подобные функциональные возможности, трудно найти [16]. Такие возможности, в том или ином виде, заложены в таких системах семейства GPSS, как GPSS/World и GPSS/H [9].

2. Распределенное имитационное моделирование

На сегодняшний день разработаны разные модели, методы и средства распределенного имитационного моделирования. В том числе такие известные системы, как Недис-Р, Мера [5], Диана [6], SLX и др. Однако применение этих разработок для моделирования распределенных вычислительных систем осложнено необходимостью представления и анализа

дополнительного уровня описания моделируемой системы – концептуальной схемы ее предметной области. Поэтому возникает необходимость в такой коммуникационной среде, которая бы поддерживала высокоуровневые системы имитационного моделирования и позволяла бы организовать обмен данными и синхронизацию времени между сегментами имитационной модели. Можно выделить множество известных механизмов передачи данных, применяемых при организации распределенных вычислений.

1. PVM (англ. Parallel Virtual Machine) [13] – это свободно распространяемая библиотека передачи сообщений. Позволяет использовать вычислительные узлы, связанные в сеть как параллельный компьютер с распределенной памятью, объединяя их в так называемую виртуальную машину.

Преимущества PVM: организация виртуальной машины использующей горизонтальный кластер рабочих станций; объединение гетерогенных вычислительных узлов и динамическое подключение узлов во время выполнения.

2. MPI (англ. Message Passing Interface) [15] – это низкоуровневая библиотека передачи сообщений, разработанная для высокопроизводительных параллельных вычислений. В отличие от PVM, MPI предоставляет функции как библиотека и не представляет собой виртуальную машину.

Преимущества MPI: гибкость, наличие большого числа функций обмена сообщениями; высокая производительность и поддержка собственных типов данных.

3. CORBA (англ. Common Object Request Broker Architecture) [1] – это технологический стандарт, проектируемый и разрабатываемый консорциумом OMG. Задача CORBA – объединить вызовы программ, написанных на разных языках и работающих как на локальных, так и на удаленных узлах сети.

В преимуществах CORBA отмечается: обеспечение межвзаимодействия процессов; приведение типов данных; отделение декларативных интерфейсов от их реализации; поддержка ряда языков, таких как C, C++, ADA, SmallTalk и Java; масштабируемость.

4. RMI (англ. Remote Method Invocation) [2] – система первоначально разработанная Sunsoft которая основывалась на Java Development Kit (JDK). Основное предназначение RMI – это упразднение различий между вызовами локальных и удаленных процедур.

Преимущества RMI: предоставление разработчику абстракции на уровне простого вызова методов объектов и инкапсуляция многих деталей передачи сообщений по сети; оперирование понятиями передачи объектов в отличие от передачи байтов, как, например, в socket-сетях; отсутствие необходимости в реализации специального протокола при взаимодействии сервера и клиента.

К сожалению, описанные выше технологии обладают общим ключевым недостатком – отсутствием функций для синхронизации модельного времени между сегментами модели. Необходимость наличия таких функций является необходимым условием для организации распределенного моделирования.

3. Технология HLA

Другим подходом к построению распределенных систем имитационного моделирования является использование архитектуры распределенного моделирования HLA (англ. High Level Architecture – Высокоуровневая архитектура) [3], которая разработана службой имитационного моделирования DMSO (англ. Defense Modeling and Simulation Office – Оборонный отдел имитации и моделирования) Министерства обороны США.

Основными компонентами HLA являются реализация инфраструктуры RTI (англ. Run-Time Infrastructure – Инфраструктура времени выполнения), включающая серверную часть и клиентские части, и библиотека классов JavaBinding, которая предназначена для реализации взаимодействия между распределенными моделями, написанными на Java.

HLA имеет несколько преимуществ перед указанными выше технологиями, а именно: она обладает поддержкой различных механизмов синхронизации времени, открытой специфика-

[1] См.: The Object Management Group – http://www.omg.org/.
[2] См.: Java Remote Method Invocation – http://java.sun.com/javase/technologies/.
[3] См.: High Level Architecture – https://www.dmso.mil/public/transition/hla/.

цией и может применяться как в распределенном, так и в параллельном моделировании. В качестве примеров, где заявлена поддержка стандартов HLA в России, можно привести системы Мера [5] и AnyLogic [4].

Можно выделить ряд существенных недостатков архитектуры HLA: это весьма сложный стандарт; распределенные системы имитационного моделирования, построенные на базе HLA, имеют определенный предел масштабируемости; в рамках HLA нужно разрабатывать дополнительные средства для обеспечения отказоустойчивости процесса распределенного моделирования; использование моделями различных версий RTI приводит к тому, что эти модели не могут взаимодействовать между собой; разработчики систем моделирования не хотят использовать HLA из-за того, чтобы их пользователи не имели доступа к программам других производителей.

Учитывая проведенный анализ, следует подчеркнуть, что для взаимодействия распределенных сегментов можно рассматривать различные технологии, такие как CORBA, RMI, коммуникационные библиотеки MPI, PVM, а также архитектуру HLA. Но эти технологии не решают всех проблем организации распределенных вычислений [2], порождают весьма сложную реализацию, настройку, эксплуатацию и не могут в полной мере (по рассмотренным причинам) обеспечить требуемый набор возможностей для организации распределенной среды имитационного моделирования [16].

4. Синхронизация времени

Первые алгоритмы синхронизации времени для проведения распределенного имитационного моделирования были описаны в работах [10; 11] и сосредоточены в области консервативной парадигмы синхронизации. Этот подход основан на блокировке параллельных процессов, выполняющихся на вычислительных узлах сети. Преимущества консервативного подхода заключаются в обеспечении безошибочной синхронизации, так как производится составление цепи будущих событий каждого узла и обеспечивается выполнение этих событий в четкой возрастающей последовательности. Главная проблема использования консервативного протокола состоит в определении «безопасного» момента времени [1; 12], при котором нужно передвигать время всей распределенной модели к новому моменту цепи будущих событий.

При этом большинство консервативных алгоритмов реализовано на вычислении так называемой *нижней границы временных меток* цепи будущих событий распределенной модели [1]. Вычисление этого показателя позволяет обнаружить, является ли следующее событие списка будущих событий безопасным или нет. Консервативные алгоритмы обладают большим недостатком – необходимостью сдерживать параллельные процессы для обеспечения безопасной синхронизации времени. Зачастую это сказывается на производительности всего имитационного процесса, особенно если длительность выполнения параллельных сегментов различается на несколько порядков.

Другим подходом к синхронизации времени является использование оптимистического алгоритма (иначе называемого алгоритмом деформации времени), который впервые опубликован в работе [14]. Если сегмент получает временную метку меньшую, чем уже обработанные события, то выполняется откат и сегмент переходит в состояние, которое было до обработки этого события благодаря аппарату контрольных точек (КТ). Такой алгоритм налагает ряд требований на моделируемую систему, таких как: отсутствие операций, которые нельзя откатить, (например, операции ввода-вывода); наличие большого внешнего накопителя для сохранения состояний логических процессов в КТ; необходимостью поддержки механизма откатов и восстановления с КТ системой имитационного моделирования.

Использование оптимистического алгоритма синхронизации требует разработки специализированных систем имитационного моделирования. В условиях большой популярности традиционных систем имитационного моделирования, таких как GPSS/World или GPSS/H, реализация консервативного алгоритма позволит задействовать перечисленные системы имитационного моделирования без внесения в них изменений, что позволит легко использовать существующие системы имитационного моделирования.

[4] См.: High Level Architecture – https://www.dmso.mil/public/transition/hla/.

5. Организация распределенного моделирования на основе специализированной коммуникационной библиотеки

В данном разделе предлагается авторский подход к организации распределенного имитационного моделирования вычислительных систем.

Общая схема процесса выполнения имитационной модели, состоящей из M сегментов, на вычислительном кластере, включающем N узлов, представлена на рис. 2. Исполнительная система среды представлена в виде двухуровневой архитектуры управления вычислительным кластером. На верхнем уровне функционирует управляющая программная надстройка, транслирующая запросы от генератора распределенной модели в формат системы пакетной обработки (СПО). На нижнем уровне управление запуском заданий осуществляется модулями СПО (в качестве системы СПО использовалась версия системы Condor для операционной системы Windows).

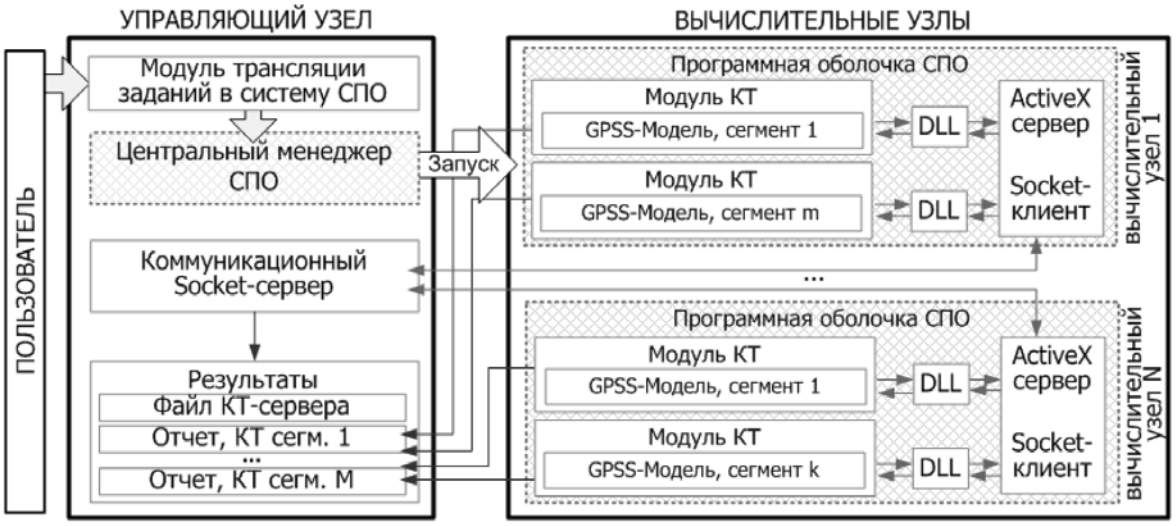

Рис. 2. Общая схема процесса выполнения распределенной имитационной модели

Взаимодействие пользователя с вычислительным кластером осуществляется посредством Web-интерфейса [8] и модуля трансляции заданий в систему СПО, который передает паспорт заданий и имитационную модель от пользователя к центральному менеджеру СПО для запуска заданий на узлах вычислительного кластера.

Для обеспечения отказоустойчивости процесса распределенного моделирования, в дополнение к коммуникационной библиотеке, разработан модуль создания КТ. Этот модуль предназначен для инициализации сегментов GPSS распределенной модели и запуска системы имитационного моделирования GPSS/World. Кроме того, модуль создания КТ следит за ходом исполнения сегментов модели и создает необходимые КТ в соответствии с пользовательскими установками. Созданные КТ позволяют просматривать промежуточные отчеты о состоянии процесса моделирования в интервале между началом и окончанием всего процесса. Подсистема СПО, установленная на каждом из узлов кластерного пула, занимается запуском соответствующих модулей создания КТ на каждом из узлов вычислительного кластера.

Коммуникационная библиотека предназначена для обеспечения взаимодействия сегментов распределенной GPSS-модели. Библиотека разработана на основе клиент-серверной архитектуры и состоит из коммуникационного сервера, ActiveX-сервера и dll-библиотеки. Коммуникационный сервер представляет собой многопоточное асинхронное .NET-приложение, созданное на языке C#. Dll-библиотека предназначена для связи системы имитационного моделирования с ActiveX-сервером, вызывается внутренними средствами языка GPSS. ActiveX-сервер, в свою очередь, создает сетевое Socket-соединение с коммуникационным сервером для передачи сообщений.

6. Алгоритм взаимодействия сегментов распределенной GPSS-модели

Использование существующих высокоуровневых сред имитационного моделирования семейства GPSS в распределенной вычислительной среде требует разработки принципиально новой распределенной имитационной модели и специализированного алгоритма, обеспечивающего синхронизированное выполнение сегментов этой модели.

Распределенную имитационную модель вычислительной системы можно представить в виде структуры $M =< S, E, T, R >$, где $S = \{s_1, s_2, ..., s_n\}$ – множество сегментов имитационной модели; n – количество сегментов; $E = \{e_1, e_2, ..., e_w\}$ – множество событий, происходящих в процессе выполнения имитационной модели; w – количество событий; $T = \{t_1, t_2, ..., t_k\}$ – множество временных меток событий из E; k – количество меток; $R = \{r_1, r_2, ..., r_l\}$ – множество ресурсов вычислительной среды; l – количество событий.

Множество временных меток событий сегмента частично упорядочено, т. е. $t_1^i \le t_2^i \le ... \le t_j^i$, $t_j^i \in T$, $i \in \{1, 2, ..., n\}$ – номер сегмента; $j \in \{1, 2, ..., w\}$ – номер события.

Множества S, E, T, R связаны между собой отношениями $L \subseteq S \times R$, $SE \subseteq S \times E$, $ET \subseteq E \times T$ в общем случае типа многие-ко-многим.

Сегменты модели могут обмениваться сообщениями. Например, $em_j^i(x, t)$ – это сообщение от *i*-го сегмента *j*-му сегменту, содержащее некоторый объем данных x и временную метку t (датировку сообщения).

Сегмент s_i будем называть генерирующим сегментом, если в момент времени t он отправляет сообщение $em_j^i(x, t)$ сегменту s_j.

Сегмент s_j будем называть поглощающим сегментом, если он получает сообщение $em_j^i(x, t)$ от сегмента s_i, отправленное в момент времени t.

Во множестве событий *i*-го сегмента E_i событие e_{end}^i, означающее, что процесс моделирования в этом сегменте завершен.

Алгоритм синхронизации представлен в виде приведенной ниже последовательности шагов:

1. Сервер запускает r клиентских приложений:

$$for(i = 1, 2, ..., r)\ \{server\_client\_start()\},\ 1 \le r \le n.$$

2. Клиенты запускают систему моделирования GPSS/World для каждого сегмента s_i:

$$for(i = 1, 2, ..., n)\ \{client\_gpss\_start()\}.$$

Следует заметить, множество клиентских приложений связано с множеством сегментов отношением типа одинкомногим, т. е. один клиент может управлять несколькими сегментами.

3. Экземпляры системы моделирования GPSS/World запускают сегменты s_i на выполнение:

$$\forall i = \{1, 2, ..., n\},\ gpss\_segment\_start(s_i).$$

Но процесс моделирования в сегментах еще не активизирован.

4. Для сегментов s_i соответствующие экземпляры GPSS/World формируют цепи будущих событий этих сегментов E_i:

$$\forall i = \{1, 2, ..., n\} : \neg\exists e_j^i = e_{end},\ gpss\_create(s_i \to E_i).$$

5. Клиенты, управляющие выполнением сегментов s_i, формируют подмножества $EM_i \subseteq E_i$ событий отправки сообщений i-м сегментом и отправляют их на сервер:

$$\forall i = \{1, 2, ..., n\} : \neg\exists e_j^i = e_{end},\ client\_create(s_i \to EM_i),$$

$$\forall i = \{1, 2, ..., n\} : \neg\exists e_j^i = e_{end},\ client\_send(EM_i).$$

6. Сервер принимает множества EM_i и формирует из них множество Q:

$$\forall i = \{1, 2, ..., n\} : \neg\exists e_j^i = e_{end},\ server\_receiv(EM_i),$$

$$\forall i = \{1, 2, ..., n\} : \neg\exists e_j^i = e_{end},\ Q = \bigcup_{i=\overline{1,n}} EM_i = \{q_1, q_2, ..., q_v\}.$$

7. Сервер определяет минимальную (ближайшую) временную метку будущих событий передачи сообщений $t_{\min}$:

$$t_{\min} = \min_{j=\overline{1,v}}(t_j).$$

8. Сервер рассылает временную метку $t_{\min}$ всем сегментам s_i:

$$for(i = 1, 2, ..., r)\ \{server\_send(t_{\min})\}.$$

9. Клиенты принимают временную метку $t_{\min}$ и устанавливают в сегментах s_i таймеры прерывания процесса моделирования (синхронизации времени) на это время:

$$for(i = 1, 2, ..., r)\ \{client\_receive(t_{\min})\},$$

$$\forall i = \{1, 2, ..., n\} : \neg\exists e_j^i = e_{end},\ client\_set(t_{\min} \to s_i).$$

10. Клиенты осуществляют запуск сегментов s_i:

$$\forall i = \{1, 2, ..., n\} : \neg\exists e_j^i = e_{end},\ client\_process\_start(s_i).$$

11. Экземплярами GPSS/World осуществляется выполнение процесса моделирования в сегментах s_i:

$$\forall i = \{1, 2, ..., n\} : \neg\exists e_j^i = e_{end},\ gpss\_process\_run(s_i).$$

12. Сервер дожидается в каждом сегменте s_i либо завершения процесса моделирования (возникновения события e_{end}), либо срабатывания таймера прерывания этого процесса:

$$\forall i = \{1, 2, ..., n\},\ server\_monitor(s_i).$$

13. Проверка завершения процесса моделирования:

$$if\ (\forall i = \{1, 2, ..., n\} : \exists e_j^i = e_{end})\ then\ \{goto\ END\}\ else\ \{goto\ 4\}.$$

END. Завершение прогона модели.

Заключение

В статье рассмотрены существующие подходы, методы и средства для организации распределенного имитационного моделирования сложных систем. Проведен обзор параллельных и распределенных технологий, таких как PVM, MPI, CORBA, RMI. Отмечается, что они обладают общим характерным недостатком – отсутствием методов синхронизации времени, что делает их применение для организации распределенного моделирования достаточно сложным. Описаны преимущества и недостатки архитектуры HLA, предназначенной для распределенного моделирования. Анализ недостатков показывает необходимость разработки собственного решения для обеспечения взаимодействия удаленных сегментов модели.

Предложен авторский подход к организации распределенного имитационного моделирования, базирующийся на использовании разработанной коммуникационной библиотеки, обеспечивающей синхронизацию времени и обмен данными между сегментами распределенной модели. Представленный подход позволил упростить организацию распределенного имитационного моделирования, использовать среды имитационного моделирования GPSS/World, GPSS/H и решить ряд перечисленных выше проблем.

С помощью разработанной коммуникационной библиотеки были решены следующие задачи:

- задействованы традиционные высокоуровневые средства имитационного моделирования, такие как системы GPSS/World и GPSS/H, широко популярные в России;
- обеспечен обмен данными между сегментами имитационной модели;
- реализована синхронизация времени между сегментами распределенной модели на основе консервативного алгоритма синхронизации с блокировкой логических процессов.

Список литературы

1. *Замятина Е. Б.* Современные теории имитационного моделирования: Специальный курс. Пермь: ПГУ, 2007. 119 с.

2. *Кельтон В., Лоу. А.* Имитационное моделирование. СПб.: Питер; Киев: Изд-я группа BHV, 2004. 847 с.

3. *Киндлер Е.* Языки моделирования: Пер. с чеш. М.: Энергоатомиздат, 1985. 288 с.

4. *Нейлор Т.* Машинные имитационные эксперименты с моделями экономических систем. М.: Мир, 1975. 500 с.

5. *Окольнишников В. В.* Разработка системы распределенного имитационного моделирования // Информационные технологии. 2006. № 12. С. 28–31.

6. *Смелянский Р. Л. Бахмуров А. Г., Костенко В. А.* Среда моделирования DYANA: синтез, анализ и оптимизация вычислительных систем реального времени // Сб. докл. I Междунар. конф. «Цифровая обработка сигналов и ее применения». МЦНТИ. 1998. Т. 4. С. 152.

7. *Советов Б. Я., Яковлев С. А.* Моделирование систем. 5-е изд. М.: Высш. шк., 2007. 344 с.

8. *Феоктистов А. Г., Корсуков А. С.* Архитектура системы управления вычислительным кластером невыделенных рабочих станций // Параллельные вычисления и задачи управления: Тр. III Междунар. конф. (PACO'2006). М.: ИПУ РАН, 2006. С. 492–497.

9. *Шрайбер Т. Дж.* Моделирование на GPSS. М.: Машиностроение, 1980. 592 с.

10. *Chandy K. M., Misra J.* Distributed Deadlock Detection // Assoc. Comput. Mach. Trans. Computer Systems. 1983. № 1. P. 144–156.

11. *Chandy K. M., Misra J.* Distributed Simulation: A Case study in Design and Verification of Distributed Programs // IEEE Trans. Software Eng. 1979. P. 440–452.

12. *Fujimoto R. M.* Parallel and Distributed Simulation Systems. USA: Wiley, 2000. 300 p.

13. *Geist A., Beguelin A., Dongarra J., Jiang W., Man-chek R.* PVM: Parallel Virtual Machine // A Users' Guide and Tutorial for Networked Parallel Computing. USA: The MIT Press, Cambridge, 1994. 279 p.

14. *Jefferson D. R.* Virtual Time // Assoc. Comput. Mach. Trans. Programming Languages and Systems. 1985. № 7. P. 404–425.

15. *MacDonald N., Minty E., Harding T., Brown S.* Writing Message Passing Parallel Programs with MPI. Edinburgh: University of Edinburgh, 2008. 80 p.

16. *Strassburger S.* Distributed Simulation Based on the High Level Architecture in Civilian Application Domains // Dissertation (Dr.-Ing.). Magdeburg. Otto-von-Guericke University, 2000. 242 p.

Материал поступил в редколлегию 22.05.2009

A. A. Aleksandrov

DISTRIBUTED SIMULATION MODELLING: TECHNOLOGIES, METHODS AND TOOLS

A review of technologies, methods and tools for distributed simulation modeling is proposed in this article. The new approach for simulation of computational systems is shown. It is based on utilization of high-level simulation systems such as GPSS/World, GPSS/H and communicational library which allows distributed segments of GPSS-model to interact.

Keywords: simulation of complex systems, distributed simulation modeling, time synchronization algorithms.