ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ

УДК 004.22, 534.26-534.231.2

А.А. Чусов, Л.Г. Стаценко, Ю.В. Миргородская

ЧУСОВ АНДРЕЙ АЛЕКСАНДРОВИЧ – кандидат технических наук, старший преподаватель кандидат наук кафедры электроники и средств связи Инженерной школы (Дальневосточный федеральный университет, Владивосток).
E-mail: Lpsztemp@gmail.com
СТАЦЕНКО ЛЮБОВЬ ГРИГОРЬЕВНА – доктор физико-математических наук, профессор, заведующая кафедрой электроники и средств связи Инженерной школы (Дальневосточный федеральный университет, Владивосток).
E-mail: lu-sta@mail.ru
МИРГОРОДСКАЯ ЮЛИЯ ВЯЧЕСЛАВОВНА – заведующая лабораториями кафедры электроники и средств связи Инженерной школы (Дальневосточный федеральный университет, Владивосток).
E-mail: yuliya_pascal@mail.ru

Параллельное имитационное моделирование физических полей автоматизированными распределенными системами

Представлены основные принципы создания систем CAM (Computer-Aided Modeling), позволяющих проводить автоматизированное имитационное моделирование и анализ различных физических полей на одной вычислительной платформе, представляющей собой распределенные и параллельные компьютеры. Такие CAM должны быть гибкими (адаптируемыми к решению задач моделирования физических полей различных типов), расширяемыми на произвольное множество вычислительных узлов в зависимости от требований ко времени, точности и стоимости моделирования. Также системы должны поддерживать параллельную многопользовательскую работу над одним или различными объектами в одной распределенной среде моделирования. Представлен метод выделения и архитектура предметно-независимых подсистем CAM, реализующих функции по параллельному обслуживанию множества подсистем моделирования, так, чтобы обеспечивалось снижение системной сложности CAM при высоких требованиях к адекватности модели и низких временных и стоимостных издержках при моделировании.

Ключевые слова: компьютерное моделирование, параллельно-распределенные вычислительные системы, моделирование физических полей.

Parallel simulation modelling of physical fields by means of the automated distribution systems. Andrey A. Chusov, Candidate of Technical Science, Senior Lecturer, Lubov G. Statsenko, Doctor of Sciences in Physics and Mathematics, Professor, Head of Department, Yulia V. Mirgorodskaya, Head of Laboratories, Department of Electronics and Communication Facilities, Far Eastern Federal University, Vladivostok, Russia.

Presented are the basic principles of creating CAM (Computer-Aided Modelling) systems, which enable one to carry out the automated simulation modelling and the analysis of various physical fields on the same computing platform representing distributed and parallel computers. Such CAM systems should be adaptable to solve the tasks of modelling the physical fields of various types and expandable to cover an arbitrary quantity of computation nodes depending on time efficiency, accuracy, and cost requirements. They must keep a multiuser work on one or various projects operating in the same distributed simulation environment. Presented is also the method of extraction and the structure of independent CMA systems performing the functions of the parallel maintenance of the multiple modelling subsystems in order to reduce the system complexity of the CMA under the conditions of high requirements for the adequateness of the model and low time and cost indexes when modeling.

Key words: computer-aided modeling, parallel and distributed computer systems, physical fields simulation.

Введение

Адекватность результатов модельных исследований в большой степени определяется степенью дискретной аппроксимации моделируемого объекта и используемыми методами моделирования из конкретной предметной области. Эти два управляющих параметра в свою очередь определяются временными, пространственными, стоимостными требованиями, которые предъявляются к модельному эксперименту. Известный статистический закон Мура, определяющий удваивание скорости вычислений машинами каждые 18 месяцев, в настоящее время может выполняться лишь с применением множества вычислителей, работающих параллельно в рамках одной системы и решающих одну задачу. В связи с этим одним из направлений работ, проводимых отечественными и зарубежными исследователями, является поиск методов распараллеливания процесса моделирования, дающих возможность задавать сколь угодно близкую к реальности модель, использовать более сложные с точки зрения вычислений методы. Наиболее известным и применяемым подходом к управлению процессом моделирования в параллельных вычислительных средах является параллельное дискретно-событийное моделирование (PDES). Этот подход определяет некоторый разделяемый между параллельными логическими процессами список элементов моделирования, которые определяются специалистом предметной области исходя из требований к минимальной пространственно-временной взаимозависимости элементов. Другой широко используемый метод конечных элементов – позволяющий аппроксимировать моделируемое пространство на некоторый набор функций таким образом, что решение задачи моделирования в конечном счете сводится к хорошо распараллеливаемому решению системы линейных алгебраических уравнений, состоящей из сколь угодно большого количества элементов.

Современные системы компьютерного моделирования направлены лишь на решение узкоспециальных задач, накладывая при этом значительные ограничения на диапазоны допустимых входных и управляющих параметров. Большинство таких систем не обладают достаточной гибкостью, чтобы позволить пользователю управлять такими показателями эффективности моделирования, как временные, стоимостные издержки при моделировании, адекватность результатов моделируемому объекту. Использование современных параллельно-распределенных компьютерных систем требует перестроения архитектур программных систем таким образом, чтобы их подсистемы могли функционировать с наименьшей взаимозависимостью во времени на минимально возможном уровне системной иерархии. Это дает удобную возможность для минимизации указанных ограничений систем компьютерного моделирования путем разделения предметно-ориентированных подсистем на различные независимые компоненты и их дальнейшего объединения в более крупные надсистемы, иногда называемые федерациями [12, 14, 20], а также разделения процесса моделирования на множество элементарных логических процессов, которые могут выполняться параллельно друг другу [3, 18].

Первый подход используется в таких технологиях, как HLA (High Level Architecture – архитектура высокого уровня) [20] и DIS (Distributed Interaction Simulation) [12, 14]. Он определяет способ и набор протоколов, используемых для объединения различных САМ в федерации для решения в ней множества различных предметных задач. Однако при этом не определяется архитектура предметно-независимых обслуживающих подсистем. Их выделение и детализированное описание позволяет снизить общую системную сложность систем моделирования, устранить необходимость создания и развертывания подсистем отдельно для каждой предметной области или решаемой задачи. Применительно к системам моделирования и анализа физических полей к таким подсистемам можно отнести подсистемы визуализации, подсистемы, предоставляющие клиентам необходимые в процессе анализа данные, подсистемы, обеспечивающие многопользовательское управление процессом анализа и системой САМ в целом, подсистемы, обеспечивающие безопасность САМ.

Второй подход, называемый параллельным дискретно-событийным моделированием (PDES – Parallel Discrete Event Simulation) [3, 18], определяет принципы управления и взаимодействия параллельно выполняемых логических процессов как компонентов процесса моделирования. Последний задается набором переменных состояний, изменяющихся во время выполнения компонентов. Логические процессы параллельно выполняют компоненты моделирования при генерации событий, содержащих временные метки и объединенные в разделяемый список событий.

В связи с этим основная цель работы заключается в разработке и реализации комплексного подхода к созданию архитектур систем компьютерного анализа, моделирования и визуализации физических полей различного типа с ориентацией на параллельные и распределенные среды с тем, чтобы повысить допустимую сложность анализа и таким образом обеспечить высокую точность и скорость работы.

Архитектура распределенной САМ на верхнем уровне

Исходя из указанной выше цели сформулировано восемь базовых принципов, определяющих требования к системе моделирования и ее архитектуре. Основное требование к системам моделирования, определяющее их структуру, заключается в гибкости для решения задач различных предметных областей. Комплексные модельные исследования могут требовать одновременной работы более чем одного пользователя над одним или множеством проектов [12, 14, 20]. Поэтому второе требование заключается в поддержке проведения различных процедур анализа одновременно множеством пользователей с использованием одной САМ.

Принцип 1. Здесь подразумевается неограниченный рост числа решаемых САМ задач по проведению моделирования с учетом роста числа имитационных экспериментов, который обусловлен высокими требованиями к адекватности модели, а также с учетом задач по обслуживанию терминалов, с которых осуществляется управление процессом. Необходимо также обеспечение расширяемости системы на множество параллельно-распределенных моделирующих и обслуживающих подсистем. Допустимый уровень параллелизма при этом может быть точно определен лишь для конкретной задачи моделирования, однако обеспечить параллелизм обслуживающих подсистем САМ можно без рассмотрения определенной предметной области и без детализации решаемых задач. Тем не менее существуют некоторые подходы [3, 18] к частичному определению способов распараллеливания непосредственно задачи моделирования без рассмотрения конкретной предметной области. Эти три базовых положения составляют принцип 1 построения распределенной САМ.

На рис. 1 приведена верхнеуровневая структурно-контекстная диаграмма системы моделирования. Выделяются следующие обязательные компоненты систем, архитектура которых является независимой от конкретной задачи анализа: графическая система, банк данных системы, подсистема управления доступными вычислительными узлами, подсистема

пользовательского интерфейса, имеющие общую реализацию. Кроме этого определяются подсистемы анализа, реализуемые разработчиком CAM для решения задач моделирования конкретных физических полей на основе требований к формату результатов моделирования и с использованием средств поддержки разработчика, предоставляемых общими компонентами.

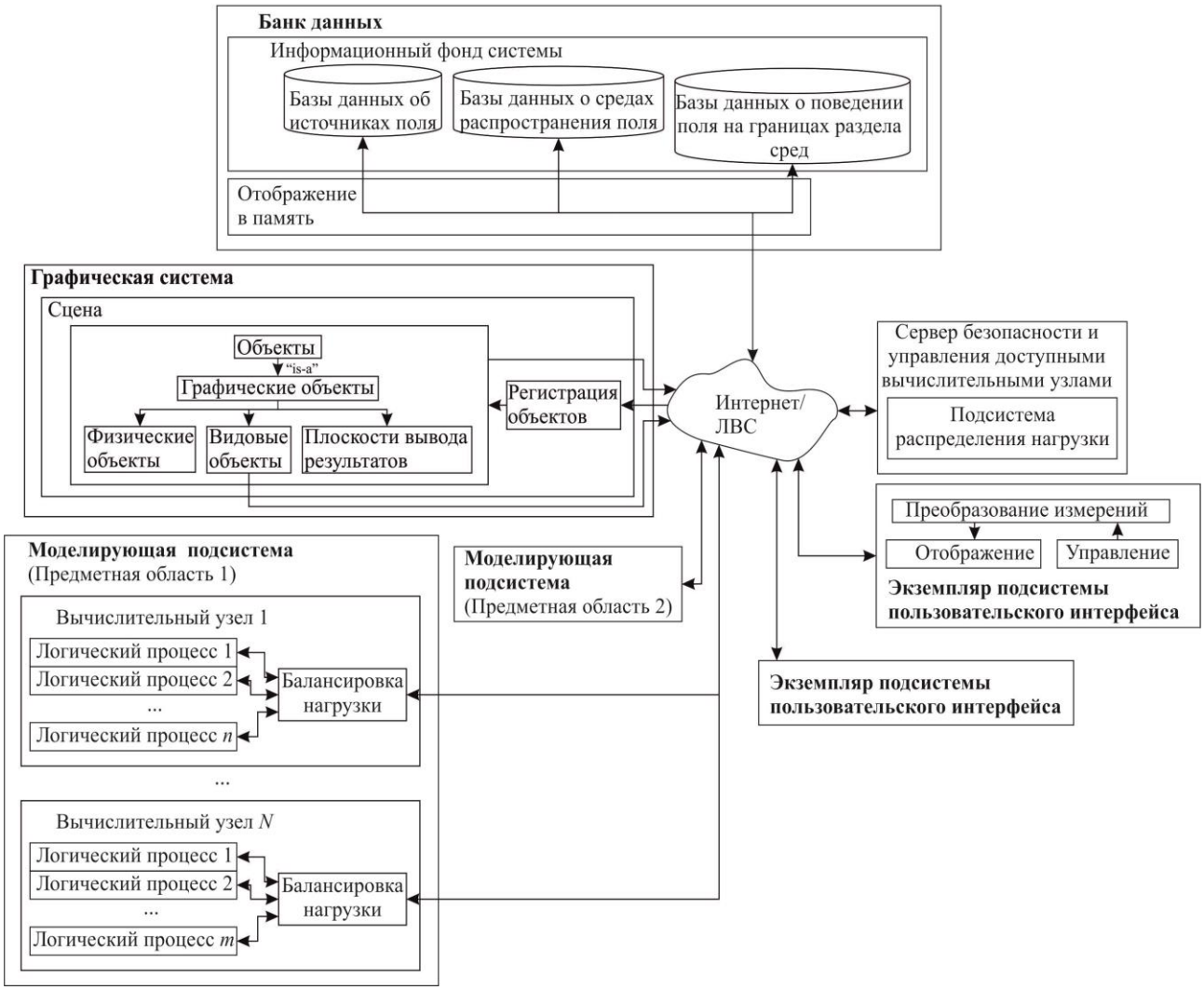

Рис. 1. Верхнеуровневая структурно-контекстная диаграмма распределенной CAM

Принцип 2 определяет подсистему пользовательского интерфейса, предоставляющую средства человеко-машинного взаимодействия и средства взаимодействия с внешними по отношению к CAM системами. Любое взаимодействие пользователя либо внешней по отношению к CAM системы должно обеспечиваться средствами системы пользовательского интерфейса. Каждый пользователь должен взаимодействовать с собственным экземпляром подсистемы. Подсистема управления – проектирующая подсистема, которая необходима для предоставления пользователю интерфейса для ввода параметров моделирования и анализа в систему: объекты, степень их аппроксимации, физические размеры, которым соответствует моделируемое пространство, параметры среды распространения моделируемого физического поля, а также положение секущих плоскостей, на которых рассчитываются распределения характеристик поля. Через подсистему отображения системы пользовательского интерфейса должны возвращаться результаты работы системы, показанные на рис. 2.

Рис. 2. Верхнеуровневая функционально-контекстная диаграмма САМ

Принцип 3. Централизованное управление узлами распределенной САМ, а также политика безопасности в САМ должны реализовываться сервером безопасности и управления доступными вычислительными узлами, задачей которого является хранение и предоставление адресов всех узлов системы, а также гарантия их подлинности и их авторизация. Вопрос защиты и контроля доступа подсистем распределенной информационной системы, включая подсистемы пользовательского интерфейса и, соответственно, пользователей, является актуальным [9, 22] при распределении вычислений по открытой сети.

При развертывании САМ системный администратор регистрирует узлы, на которых развернуты необходимые подсистемы САМ, на сервере безопасности и управления доступными вычислительными узлами. При необходимости, если отсутствует поддержка или разрешение политикой безопасности на удаленную регистрацию дополнительных узлов и пользователей, при развертывании системы также должны быть зарегистрированы пользователи системы и набор узлов, осуществляющих функции подсистем пользовательского интерфейса.

Запуск САМ должен заключаться в запуске сервера безопасности и управления доступными вычислительными узлами, который осуществляет опрос и аутентификацию всех обслуживающих и анализирующих подсистем, за исключением подсистем пользовательского интерфейса, по известным ему адресам и с использованием ключей, полученных во время регистрации. После запуска сервер должен производить циклическое прослушивание своих каналов на предмет подключения недоступных ранее узлов и удаленных подсистем пользовательского интерфейса.

Этой же либо выделенной отдельно подсистемой должно осуществляться распределение нагрузки между аутентифицированными подсистемами предметной области. При этом возможно использование следующего протокола. Для каждой отдельно взятой предметной области создается реестр адресов анализирующих подсистем, не занятых выполнением анализа (рис. 3). Также для каждой предметной области подсистемой создается список задач по выполнению отдельного имитационного эксперимента. Во время проведения процедуры анализа, если реестр адресов подсистем предметной области не пуст, параметры одного имитационного эксперимента из совокупности задач, которые необходимо выполнить для

осуществления анализа, передаются по одному из адресов из реестра. Если реестр пуст, параметры заносятся в список задач, выполнение которых осуществляется по мере освобождения подсистем предметной области. Подобные схемы управления нагрузкой в параллельных и распределенных системах имитационного моделирования являются предметом исследований в работах [3, 18], посвященных принципам PDES. В соответствующих терминах сервер осуществляет хранение списка событий, которыми являются сообщения о появлении задач по проведению имитационного эксперимента с заданными параметрами.

<table>
<tr><td colspan="5">Реестр свободных подсистем предметной области</td></tr>
<tr><td rowspan="3">Идентификатор предметной области 1</td><td>Адрес узла 1 анализирующей подсистемы</td><td>Адрес узла 2 анализирующей подсистемы</td><td>…</td><td>Адрес узла N_1 анализирующей подсистемы</td></tr>
<tr><td colspan="4">Параметры логических процессов</td></tr>
<tr><td>Параметры имитационного эксперимента: $(I, f, \theta_i, \varphi_j)_1$</td><td>Параметры имитационного эксперимента: $(I, f, \theta_i, \varphi_j)_2$</td><td>Параметры имитационного эксперимента: $(I, f, \theta_i, \varphi_j)_3$</td><td>…</td></tr>
<tr><td colspan="5">…</td></tr>
<tr><td rowspan="3">Идентификатор предметной области M</td><td>Адрес узла 1 анализирующей подсистемы</td><td>Адрес узла 2 анализирующей подсистемы</td><td>…</td><td>Адрес узла N_M анализирующей подсистемы</td></tr>
<tr><td colspan="4">Параметры логических процессов</td></tr>
<tr><td>Параметры имитационного эксперимента: $(I, f, \theta_i, \varphi_j)_1$</td><td>Параметры имитационного эксперимента: $(I, f, \theta_i, \varphi_j)_2$</td><td>Параметры имитационного эксперимента: $(I, f, \theta_i, \varphi_j)_3$</td><td>…</td></tr>
</table>

Рис. 3. Разделяемые реестр подсистемы управления вычислительными узлами и список событий

Принцип 4. Подсистема преобразования измерений необходима для реализации алгоритма взаимного отображения координат физического пространства на внутренние единицы измерения с тем, чтобы все значения могли быть заданы в единой разрядной сетке и в едином порядке следования байт, задаваемыми при создании модели и при развертывании САМ. Этой подсистемой должны определяться цены деления и пределы измерений в системе. Вызовы, осуществляемые пользователями через подсистему управления, должны обрабатываться данной подсистемой. Данная подсистема может быть выделена в отдельный узел для централизованного обслуживания запросов с подсистем пользовательского интерфейса. В этом случае целесообразно ее включение в графическую подсистему, предоставляющую, как показано ниже, функции централизованного геометрического моделирования. Также подсистемы преобразования измерений могут быть включены в подсистемы пользовательского интерфейса, как показано на рис. 1, для снижения нагрузки на централизованные узлы. В этом случае разрядность внутренних единиц измерения должна быть получена через обращение к графической подсистеме.

Принцип 5. Исходя из требований, определенных принципом 1, необходимо выделение отдельной подсистемы, выполняющей централизованное геометрическое моделирование пространства для последующего проведения различных имитационных экспериментов. Выделяется подсистема, определенная как графическая, которая хранит и задает графические параметры модели в системе, а также выполняет графическое отображение модели независимо от конкретных задач моделирования. Этой подсистемой реализуется компонент-сцена,

который предоставляет клиентский доступ к элементам геометрической модели. Задача видового компонента – обеспечение графического отображения элементов модели, определенных в другом (произвольном) архитектурном модуле. Это отображение предоставляется клиентам видового компонента по запросу в произвольные моменты времени. Дальнейшая подготовка кадра изображения осуществляется на стороне клиента средствами подсистем визуализации системы пользовательского интерфейса. Это дает возможность использовать полученное в результате видового преобразования изображение в качестве графической подложки, на которую может быть наложен слой, задающий графическое представление распределения энергетических характеристик поля.

Принцип 6. Разделяемый доступ компонентов САМ к постоянным справочным данным, необходимым при расчетах, должен предоставляться отдельной обслуживающей подсистемой – банком данных САМ. Информационное наполнение банка составляют данные, необходимые для проведения каждого имитационного эксперимента: сведения о типичных источниках поля, о физических свойствах среды распределения поля для заданного набора внешних условий, количественные сведения, характеризующие поведение поля на границах раздела сред.

Принцип 7. Целью проведения математического моделирования поведения физических полей в большинстве случаев является анализ распределения энергетических характеристик этих полей, таких как интенсивность, в простом или комплексном пространстве по заданному набору контрольных точек. Поэтому для систем САМ определен формат вывода результатов моделирования как набор матриц энергетических характеристик поля по заданному набору секущих плоскостей (рис. 2). Эти матрицы могут быть поданы на вход определенной разработчиком подсистемы, например подсистемы визуализации (рис. 1) системы пользовательского интерфейса, осуществляющей визуализацию распределения поля по этим матрицам. При этом в качестве графической подложки могут быть использованы результаты отображения геометрической модели, являющиеся выходом графической подсистемы.

Критерий эффективности САМ определяется тремя взаимозависимыми показателями [2, 12, 18]: временем анализа, требуемой точностью результатов, а также максимальной стоимостью системы. Законченный вид функции критерия должен определяться при разработке и развертывании реализованной системы САМ. Степень повышения точности вычислений задается доступными временными ресурсами на конечной аппаратной платформе путем понижения шагов дискретизации и повышения числа имитационных экспериментов, а также возможным введением более сложных аналитических и численных методов. Высвобождение временных ресурсов, частично или полностью затрачиваемых на повышение точности и, соответственно, общей вычислительной сложности процесса анализа, осуществляется введением параллелизма вычислений (см. принцип 3). Противоречивым показателем эффективности являются налагаемые ограничения по стоимости развертывания системы анализа и времени вычислений.

Принцип 8 определяет следующее минимальное разделение пользователей САМ по ролям (рис. 2). Минимальными правами, необходимыми для проведения моделирования и анализа поля с использованием заданных набора подсистем, настроек безопасности и существующим наполнением банка данных, должны обладать пользователи системы. Для внесения изменений в информационное наполнение банка данных определена роль администратора банка данных с соответствующими правами на изменение его содержимого. Оператор подсистем безопасности САМ должен иметь возможность внесения изменений в настройки безопасности как отдельных подсистем, так и системы в целом: определение ролей и прав доступа пользователей, управление протоколами регистрации, аутентификации и авторизации узлов и пользователей системы. Роль администратора системы должна объединять привилегии других ролей, а также включать привилегии на добавление в систему дополнитель-

ных узлов. Система должна поддерживать изменение данного набора ролей, включая его расширение и изменение прав доступа к системе для каждой из ролей.

При запуске пользователем экземпляра подсистемы пользовательского интерфейса производится процедура инициализации подсистемы в САМ в соответствии с протоколом, основанным на принципе 3. Узел, на котором развернута подсистема пользовательского интерфейса, и пользователь проходят процедуры аутентификации (либо регистрации, если это разрешено политикой безопасности и поддерживается соответствующим протоколом) и авторизации, обращаясь по известному адресу к серверу безопасности. Далее посредством обращения к подсистеме управления вычислительными узлами подсистема интерфейса пользователя получает список адресов и идентификаторов доступных узлов, на которых развернуты графические подсистемы САМ. Далее пользователь задает моделируемый объект путем создания элементов геометрической модели и регистрации их в графической подсистеме. После создания геометрической модели и присваивания физических параметров ее элементам адрес соответствующего компонента-сцены подается подсистемой пользовательского интерфейса на вход подсистемы предметной области, предоставленной подсистемой управления нагрузкой. Последняя регистрирует входящие в нее свободные узлы, осуществляющие параллельное имитационное моделирование и анализ, в описанном выше реестре подсистемы управления информационной нагрузкой и, в случае появления новых компонентов имитационного анализа, передает параметры, полностью описывающие эти компоненты, подсистеме управления нагрузкой. Независимое вычисление распределения поля для каждого модельного эксперимента, для каждого направления излучения и для каждого источника позволяет разделить процесс моделирования на набор параллельных единиц выполнения. Соответствующая диаграмма потоков данных показана на рис. 4.

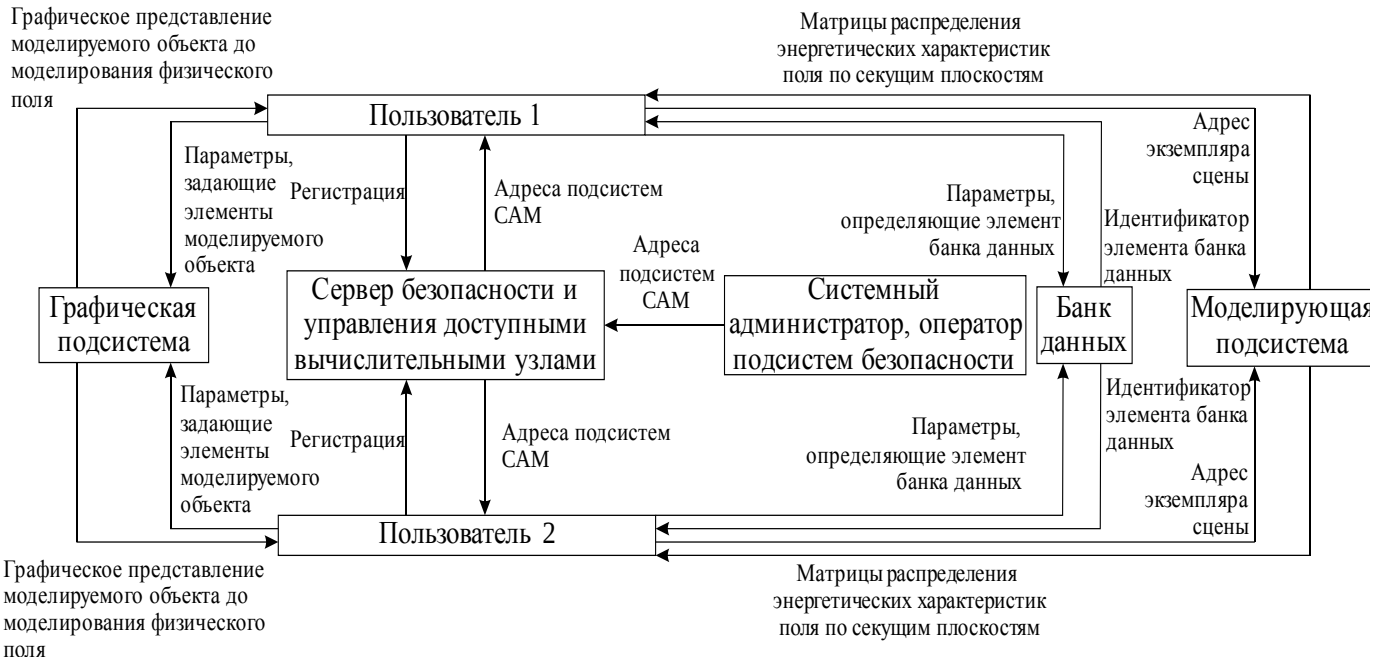

Рис. 4. Верхнеуровневая диаграмма потоков данных

Реализация системы управления банком данных с уменьшенной состязательностью за доступ к разделяемым данным

Основным фактором, прямо влияющим на структуру системы управления банком данных (СУБД) (рис. 5) в представленной реализации, является необходимость обеспечения одновременного синхронизированного доступа множества клиентов – подсистем (рис. 1) распределенной системы – к большому объему данных банка. В связи с этим имеет смысл

использовать такой способ физической организации данных в хранилище отдельной базы данных, когда эти данные предваряются хеш-таблицей, где значения сгруппированы в древовидную структуру. Структура дерева выбирается такой, при которой ее ветви и листья обладают наименьшей взаимозависимостью при выполнении операций над ними. Последнее дает возможность свести к минимуму использование блокирующих механизмов синхронизации. При этом допускается предположение о высокой вероятности такого доступа к данным, при котором они остаются неизменными. Это дает возможность использовать механизмы, обеспечивающие наименьшую затратность при синхронизации доступа к банку клиентов. Ввиду того что в соответствии с принципом 6, на описываемом уровне абстракции тип хранимых данных не определен, дополнительным требованием является обеспечение СУБД обобщенности хранимых элементов вплоть до внешних интерфейсов.

Структура СУБД определяет две подсистемы (рис. 5). Первая – подсистема хранения и представления данных на физическом уровне. Она предоставляет синхронизированный доступ к единицам физической модели хранения, реализует контроль доступа принципалов к данным посредством использования разграничительных списков контроля доступа (DACL – Discretionary Access Control List).

Вторая – серверная подсистема – объединяет множество хранилищ и предоставляет доступ к данным такой объединяющей логической модели организации данных, обеспечивая внешнюю синхронизацию и защиту доступа к логической модели, реализует клиент-серверное взаимодействие с помощью протоколов удаленного вызова процедур (RPC – Remote Procedure Call) и механизмов межпроцессного взаимодействия (IPC – Interprocess communication).

Каждый файл хранит отдельное дерево. Дерево, хранимое в файле, состоит из элементов трех типов: супергруппа, группа и элемент. Элементы последних двух типов имеют имена. Хеш-значения, по которым эти элементы идентифицируются, являются значениями детерминированной функции от имени. Каждый элемент дерева, кроме корня, сопровождается указателем, под которым здесь подразумевается значение, состоящее из хеш-идентификатора *ID* фиксированной длины и сдвига данных элемента, в байтах, относительно начала файла *FP*.

Корнем дерева файла является супергруппа. Супергруппа представляет собой упорядоченный по идентификаторам вектор $S_G = \{P_G\}$ из фиксированного числа N_G указателей на группы и их имен. Элементы супергруппы подразделяются на три категории. Первыми являются те элементы, имена которых имеют действительные значения, т.е. элементы, содержащие реально существующие группы. В том случае, если группа пуста и не имеет зарезервированного под элементы пространства, файловый указатель *FP* на данные группы, хранимый в указателе супергруппы, имеет служебное значение FP = -1. В предположении, что платформа для представления отрицательных чисел использует дополнительный код, на рисунке и далее максимальное значение, соответствующее заданной разрядности, обозначается как -1. Если группа пуста, но имеет зарезервированное пространство, все ее элементы, представляющие собой указатели, имеют FP = -1. Последними в супергруппу входят элементы с идентификаторами, имеющими служебное значение -1. Соответствующие группы рассматриваются системой как несуществующие. Вместе с тем среди них также есть деление на категории и упорядочение по файловым указателям: первыми идут элементы, имеющие зарезервированное пространство ($FP \neq$ -1). При создании клиентом новой группы такие элементы имеют приоритет при выборе места для хранения указателя новой группы. Вектор S_G оканчивается возможными пустыми элементами, оба поля указателей которых имеют значения -1.

Данные группы представляют собой односторонние связанные списки из векторов фиксированного размера N_E, состоящих из указателей. Первые $N_E - 1$ указателей являются указателями на фактические данные и имеют идентификаторы, образуемые детерминирова-

но от имен элементов с помощью хеш-функции. Структура данных таких элементов, образующих листы дерева, представлена тремя полями: имя элемента, которое должно быть уникальным в пределах всей группы (по причине упорядоченности элементов по соответствующим хеш-значениям); четырехбайтовый размер данных элемента, в байтах, диапазон значений которого ограничен сверху максимальным размером, указанным в заголовке файла; буфер данных элемента размером, указанным в заголовке, содержащий заданное предыдущим полем количество байтов данных элемента.

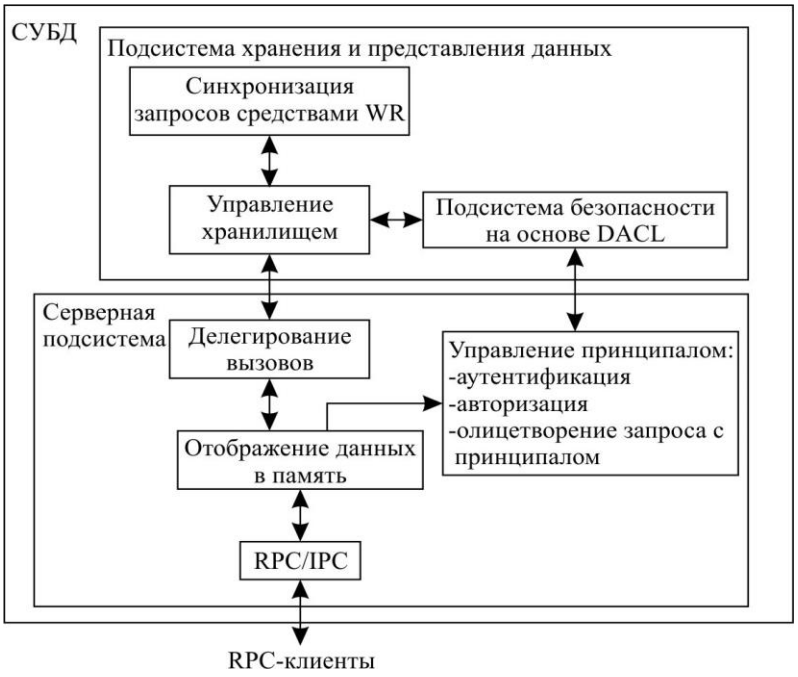

Рис. 5. Структурная диаграмма СУБД

Последний элемент вектора указывает на следующее звено списка группы. Этот элемент имеет значение идентификатора, равное идентификатору группы, и файловый сдвиг на данные следующего звена. Смысл служебного значения -1 полей указателя элемента тот же, что и для группового указателя. Упорядочение указателей элементов ведется по всему групповому списку.

Для работы с элементами дерева клиенты используют дескриптор подсистемы представления данных на физическом уровне, содержащий дескриптор файла, который фактически хранит данные, а также имя и дескриптор экземпляра механизма синхронизации, реализующий модель синхронизации «читающий-пишущий» (Write-Read, далее обозначается как WR). Оригинальная реализация данного механизма описана ниже. Экземпляр с заданным именем осуществляет синхронизацию всех потоков выполнения, открывших либо создавших этот экземпляр. Имя является значением детерминированной функции от абсолютного имени хранилища. Описанная структура файла создана такой, чтобы выделить три основных иерархических уровня дерева хеш-значений: супергруппу, группу и элемент, и обеспечить между ними наименьшую временную взаимозависимость. Кроме того, при такой организации структуры операции на одном уровне иерархии наименьшим образом влияют на другие уровни, что позволяет проводить эффективную синхронизацию разделяемого параллельного доступа к описанному дереву.

Синхронизация параллельно-распределенного доступа к разделяемым ресурсам

В представленной архитектуре CAM содержит разделяемые ресурсы, параллельный доступ к которым осуществляется множеством клиентов. Поэтому одной из целей исследования является разработка механизма синхронизации единиц выполнения, реализующего модель WR и накладывающего минимальные временные издержки, вызванные блокирующей синхронизацией и состязательностью.

P-схемы [1] алгоритмов, реализующих протоколы вхождения, выхода и безопасного динамического включения/выключения синхронизации механизмом, см. на рис. 6. Механизм использует три примитива синхронизации: мьютекс, фьютекс (объект критической секции) и автоматически сбрасываемое событие. Эти механизмы реализуются во всех современных многозадачных операционных системах и являются встроенными средствами многих современных языков программирования [15, разд. 30; 16, разд. 15.12]. Кроме того, реализации данных примитивов опубликованы в [17, 19, 21]. Это делает реализацию WR переносимой.

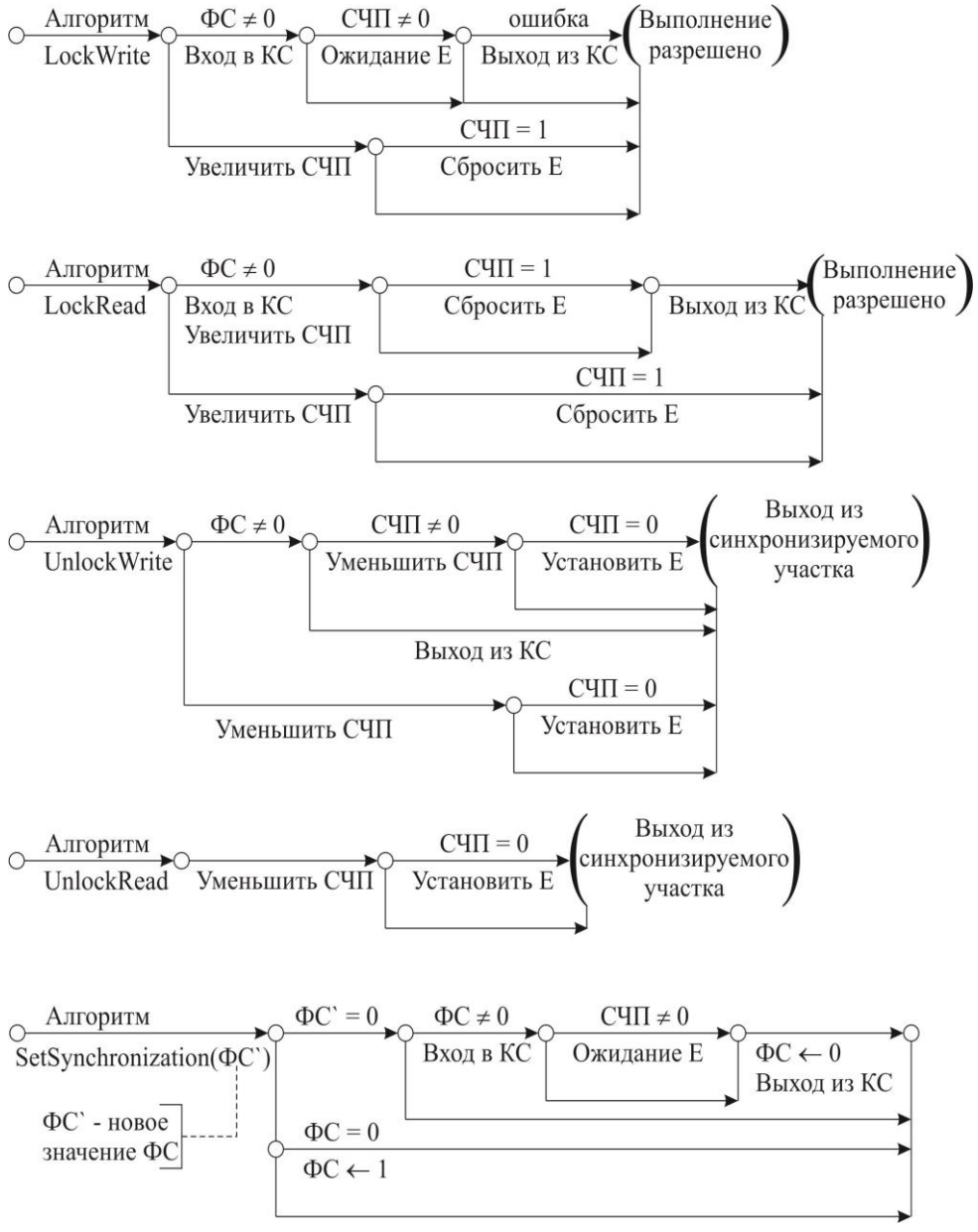

Рис. 6. P-схемы алгоритмов, реализующих протоколы WR. КС – мьютекс или фьютекс, E – автоматически сбрасываемое событие, ФС – флаг синхронизации, СЧП – атомарный счетчик читающих потоков

В своей основе механизм опирается на пять алгоритмов, представленных на рис. 6:

LockWrite – алгоритм вхождения в синхронизируемый участок кода на запись. Это означает, что все потоки, одновременно с вошедшим пытающиеся завладеть тем же экземпляром механизма WR, блокируются на время нахождения в участке исходного потока.

LockRead – алгоритм вхождения потоком в синхронизируемый участок кода на чтение. Это исключает одновременное вхождение другими потоками в участки, синхронизируемые посредством того же экземпляра WR, на запись. Однако читающие потоки выполняются беспрепятственно.

UnlockWrite – алгоритм освобождения экземпляра WR, занятого на запись потоком, выполняющим данный алгоритм.

UnlockRead – алгоритм освобождения экземпляра WR, занятого на чтение.

SetSynchronization – механизм безопасного динамического включения/отключения механизма синхронизации WR. Состояние механизма задается флагом, указанным параметрически.

Для доказательства того, что представленные алгоритмы корректны, введем следующие априорные условия:

1) элементарные операции над описанными выше примитивами синхронизации производятся корректно и атомарно;

2) примитив КС корректно блокирует все потоки, кроме владеющего им до момента выхода последнего из КС;

3) операции инкремента и декремента СЧП являются атомарными; это может быть обеспечено функциями взаимоблокировки [8, 13] либо посредством отдельного примитива КС;

4) также атомарными являются операции присваивания и сравнения СЧП и ФС;

5) в процессе синхронизации приведенные выше алгоритмы вызываются клиентами в корректном порядке: за вызовом LockWrite в том же потоке следует вызов UnlockWrite, за вызовом LockRead – вызов UnlockRead; алгоритм SetSynchronization может быть вызван произвольно, также отсутствуют вложенные участки синхронизации.

Далее для краткости последующего изложения введем следующие определения.

1. Истинно пишущий поток – поток, полностью выполнивший алгоритм LockWrite по ветвям рис. 6 при установленном ФС, но еще не выполнивший алгоритм UnlockWrite.

2. Псевдопишущий поток – поток, выполнивший алгоритм LockWrite по ветви, соответствующей сброшенному ФС, но еще не выполнивший алгоритм UnlockWrite.

3. Псевдо- и истинно читающие потоки определяются аналогично для алгоритмов LockRead и UnlockRead.

Лемма 1. Невозможно существование истинно пишущего потока, если СЧП $\neq 0$.

Действительно, так как сброс ФС возможен только в той же КС, в которой по определению находится истинно пишущий поток, для всех других потоков, использующих данный экземпляр WR, ФС также будет установлен. Как иллюстрирует рис. 6, единственным алгоритмом, который увеличивает СЧП в ветви, соответствующей ФС $\neq 0$, является LockRead. Однако последний делает это, находясь в КС экземпляра WR. Как свидетельствует рис. 6, если СЧП $\neq 0$, поток блокируется в ожидании сигнала от события E, которое установлено, только в том случае, если СЧП $= 0$ (обратная импликация). Это событие может быть установлено в алгоритмах UnlockWrite и UnlockRead, если в результате декремента значение СЧП упало до 0. Лемма доказана.

Лемма 2. Выход из КС в процессе выполнения псевдопишущим потоком алгоритма UnlockWrite невозможен.

По определению псевдопишущие потоки, выполняя алгоритм LockWrite – инкрементируют СЧП. Поэтому при наличии псевдопишущего потока при ФС ≠ 0 СЧП также не равен 0. В этих условиях, как и в случае ФС = 0, выполнение псевдопишущим потоком ветви, освобождающей КС, невозможно, что показано на рис. 6. Лемма доказана.

Теорема 1. Алгоритмы LockWrite и UnlockWrite корректно обеспечивают исключение одновременного выполнения синхронизируемого ими участка кода, если флаг синхронизации установлен в момент вхождения. Если флаг сброшен во время выполнения потоком алгоритма LockWrite, механизм не выполняет взаимного исключения потоков независимо от того, устанавливается ли ФС в процессе выполнения синхронизируемого участка кода клиента.

Если на момент вызова LockWrite флаг ФС установлен, единственно возможный путь безошибочного выполнения алгоритма – через вхождение в КС. В случае отсутствия внутренних ошибок алгоритма (на которые должным образом реагирует клиент) выход из КС в рамках LockWrite не производится. Следовательно, взаимное исключение потоков происходит на данном этапе, а истинно пишущий поток всегда владеет КС. При существовании такого потока, как показано леммой 1, ФС всегда имеет ненулевое значение, а СЧП – всегда нулевое. Поэтому единственный возможный путь выполнения алгоритма UnlockWrite пишущим потоком – через выход из КС. Следовательно, пары вызовов LockWrite-UnlockWrite при ФС ≠ 0 всегда корректно обеспечивают синхронизацию потоков посредством их взаимного исключения.

Если до выполнения LockWrite КС ФС был сброшен, единственный путь выполнения алгоритма – через инкремент СЧП и последующее завершение алгоритма. По определению поток, выполняющий данную ветвь, является псевдо пишущим. Согласно лемме 2, он не может выйти из КС, что является корректным поведением. Поэтому, согласно рис. 6, единственными путями выполнения алгоритма UnlockWrite, независимо от значения ФС, для псевдопишущего потока являются пути, на которых производится декремент СЧП и возможна установка события.

Теорема доказана.

Теорема 2. Истинно читающие потоки не могут выполняться одновременно с истинно пишущими, но могут выполняться одновременно друг с другом. Алгоритмы LockRead и UnlockRead корректно выполняют последовательность действий независимо от динамического включения/отключения ФС.

Истинно читающие потоки, как иллюстрирует рис. 6, в начале выполнения алгоритма LockRead увеличивают СЧП. Из леммы 1 следует, что при этом существование истинно пишущих потоков невозможно.

Рисунок 6 показывает, что независимо от значения ФС, т.е. от ветви, по которой идет выполнение алгоритма LockRead, синхронизируемый клиентский участок кода не защищается КС. Поэтому этот участок кода может выполняться параллельно множеством истинно читающих потоков, а также потоков, в процессе выполнения не использующих КС: псевдопишущих и псевдочитающих.

Теорема доказана.

Из теорем 1 и 2 и их доказательств следует, что алгоритм SetSynchronization безопасен для многопоточного динамического изменения значения ФС, т.е. для включения и отключения механизма WR.

Кроме данных теорем корректность алгоритмов подтвердили результаты численных экспериментов, которые заключались в создании от 2-х до 512-и потоков выполнения, ис-

пользующих механизм WR с различной задержкой в синхронизируемом участке кода, а также с различным соотношением читающих и пишущих потоков.

Целью создания данного механизма было необходимое расширение функционала по сравнению с аналогами, а также повышение вычислительной эффективности составляющих его алгоритмов. Решение этих двух задач заключается в компромиссе. Механизм может работать в двух режимах — базовом и расширенном. Базовый режим наиболее быстр, так как в этом случае к ядру операционной системы обращение происходит только при работе с событием. В этом случае в качестве примитива, реализующего взаимное исключение потоков, используется более быстрый объект критической секции кода (фьютекс). В расширенном режиме механизм может быть использован несколькими клиентами из разных процессов (в случае именованных экземпляров механизма). Для именованных экземпляров механизма WR действуют пространства имен объектов ядра. Также в расширенном режиме можно задать конечный интервал времени ожидания освобождения критической секции кода на чтение и запись, а также проводить это ожидание в дежурном режиме (поддержка асинхронного вызова процедур) [8, 13].

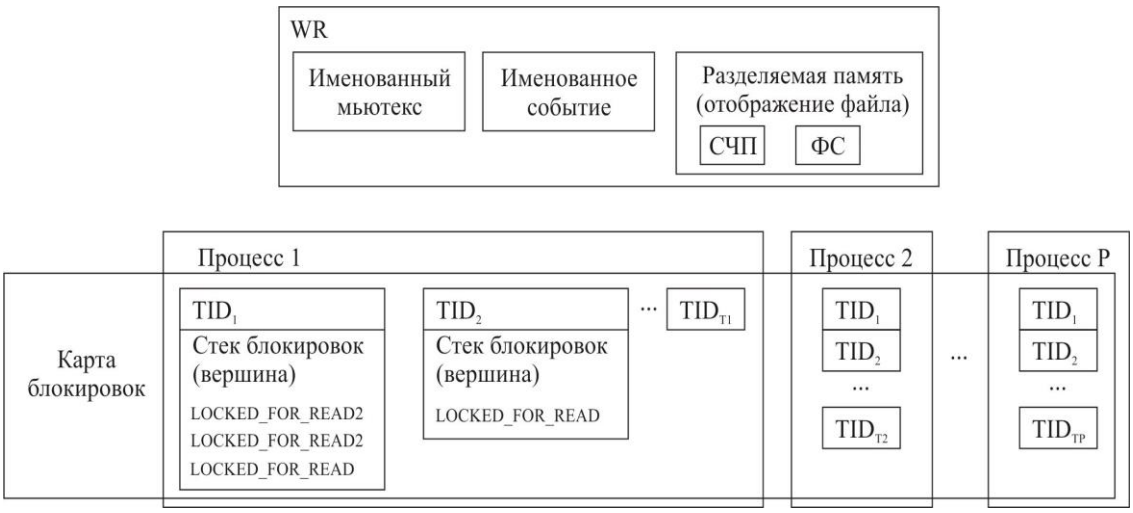

Рис. 7. Состояние памяти и единиц выполнения для WR в расширенном режиме

Состояния клиентских потоков выполнения фиксируются в структурах данных, принадлежащих клиентским процессам. Такие структуры в совокупности являются распределенным между клиентскими процессами ассоциативным массивом, ключевыми значениями которого являются идентификаторы потоков (TID – Thread Identifier), использующих механизм WR. Массив содержит стеки, включающие типы блокировок, захваченных потоками. Это дает возможность, во-первых, безопасного выхода из синхронизируемых участков кода, во-вторых – безопасного многократного вложенного вхождения в синхронизируемые WR секции, как показано на рис. 7, для потока TID_1 процесса 1.

Интерфейс описанного алгоритма задается следующими процедурами, в свою очередь вызывающими базовые функции, реализующие алгоритмы (рис. 6). EnterLockForRead и EnterLockForWrite – базовые процедуры входа в синхронизируемые секции кода соответственно на чтение и запись. EnterLockForReadEx и EnterLockForWriteEx – аналогичные процедуры для экземпляров WR в расширенном режиме. Последние две процедуры предоставляют описанные выше возможности для механизмов, работающих в расширенном режиме. LeaveWRLock – процедура выхода из синхронизируемой секции кода.

Использование WR для синхронизации разделяемого доступа к банку данных системы

На рис. 8 представлено пять типов операций подсистемы управления данными с точки зрения многопоточной синхронизации. При этом используются два иерархических уровня синхронизации доступа к элементам дерева хеш-значений. Первый уровень управляется посредством именованного экземпляра WR. Этот экземпляр создается при запуске подсистемы управления данными с именем, являющимся функцией от имени файла-хранилища, а его дескриптор, на рис. 8 обозначенный как hLock, содержится в структуре дескриптора подсистемы. Этот экземпляр используется для защиты вектора супергруппы. Второй уровень синхронизации задается набором экземпляров, работающих в расширенном режиме. На рис. 8 этот набор задается дескрипторами hGroupLock, hGroup1Lock, …, hGroupXLock. Задача этих экземпляров состоит в обеспечении синхронизации обращений к группам и элементам этих групп. Эти экземпляры также являются именованными. Имя является детерминированной функцией от базового имени синхронизации (имя экземпляра hLock) и идентификатора группы.

Первые два типа операций – изменения и чтения на уровне супергруппы – являются простейшими случаями использования WR, заключающимися в выполнении соответствующих операций при нахождении в базовой КС на запись и чтение соответственно. Значения FP полей супергруппы защищаются следующими двумя типами. Третий и четвертый типы обеспечивают синхронизацию операций уровня групп. При этом в первую очередь обеспечивается вхождение в базовую КС на чтение, что защищает супергруппу от изменений во время вызовов, не блокируя вызовы, не изменяющие супергруппу. Вложенным является вхождение в КС для защиты структуры групп. К данным типам не относятся операции, результаты которых – изменения имен и идентификаторов групп, поскольку, ввиду необходимости сохранения упорядоченности соответствующих указателей супергруппы, последняя изменяется, что требует применения синхронизации 1-го типа.

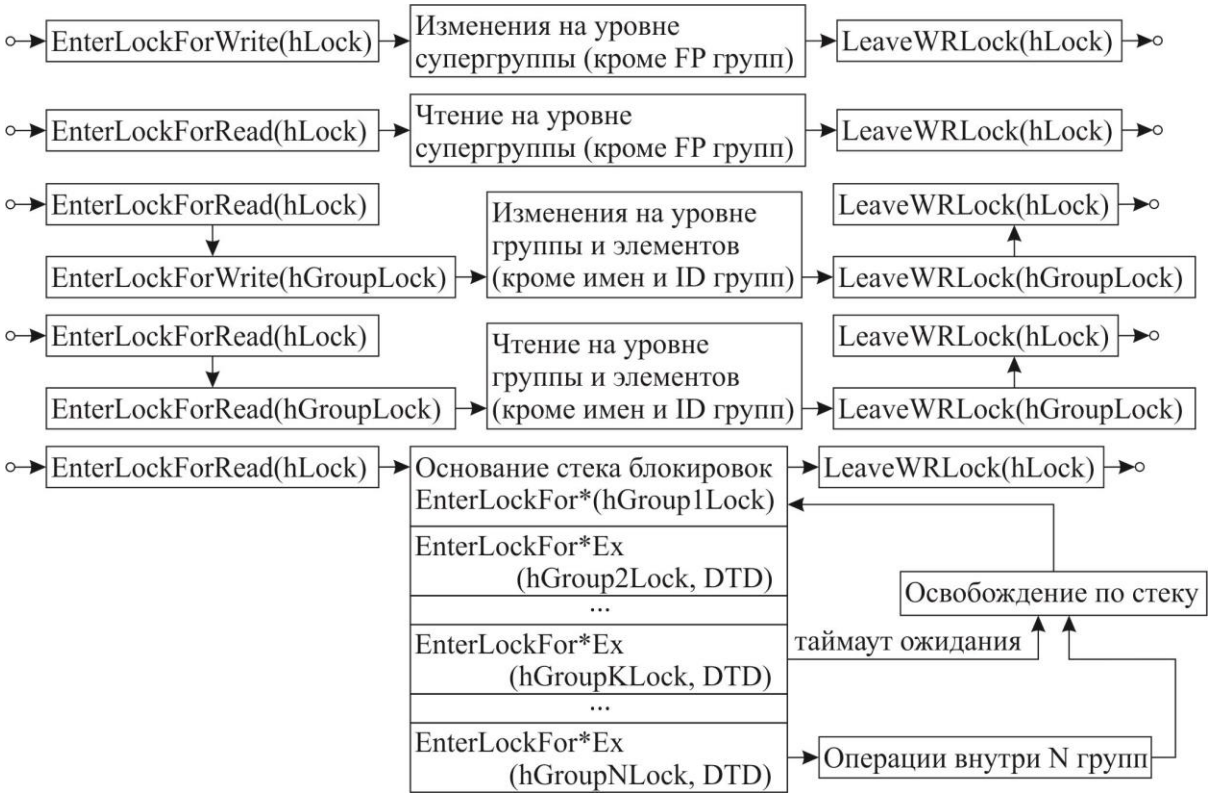

Рис. 8. Использование WR для синхронизации доступа к банку данных

Последний, пятый, тип синхронизации защищает дерево при операциях, изменяющих структуры/элементы одновременно нескольких групп. К таким операциям относится, например, перемещение элементов из одной группы в другую. При этом использование безвременной блокировки участвующих в операции групп может привести к взаимоблокировке выполняющихся потоков. Поэтому схема синхронизации (рис. 8) использует конечное ожидание освобождения экземпляров WR соответствующих групп. Время ожидания подбирается эмпирически для конкретных внешних условий: аппаратной и программной среды, в которой работает подсистема, наиболее вероятного количества клиентов, одновременно выполняющих операции, требующие синхронизации данного типа, наиболее вероятный тип операций файловой подсистемы и их частоту. Время должно быть выбрано так, чтобы в этих условиях обеспечить минимальную среднюю длительность взаимоблокировок. На рис. 8 это время обозначается как DTD (Deadlock Timeout Detection), которое подается на вход алгоритмов EnterLockForReadEx и EnterLockForWriteEx. При такой синхронизации экземплярами WR групп применяется стековая структура соответствующих вызовов. При этом в случае возникновения таймаута ожидания освобождения WR одной из групп предполагается возникновение условий состязательности. В этом случае осуществляется последовательное освобождение WR групп верх по стеку, после чего производится следующая итерация цикла входа в КС WR групп. Максимальное количество итераций цикла может быть как конечным, так и бесконечным. Кроме того, в ряде случаев (при длительных операциях внутри КС и большом количестве клиентов, одновременно производящих такие операции) общая производительность повышается при введении ненулевой временнòй задержки между итерациями. В случае если ситуации таймаута не возникает, после выполнения потоком защищаемых операций он освобождает занятые КС вверх по стеку, после чего завершает выполнение функции.

Эффективность использования синхронизации мьютексами и различными реализациями механизма WR

Выше показано, что СУБД, как и другие компоненты параллельно распределенной системы моделирования, активно используют механизм синхронизации WR для защиты разделяемых данных при параллельном доступе, хотя подходящие по функциональности аналоги также могут быть использованы в системе для синхронизации. Более важным для проектировщика законченной системы моделирования является выбор удобного для практического использования метода аналитической оценки эффективности. Существующие методы оценки не рассматривают такие специфичные для многопоточной среды выполнения источники временных издержек, как состязательность потоков, собственная сложность алгоритмов управления потоками выполнения. Предложенный метод оценки основывается на построении вероятностной модели вхождения параллельных алгоритмов в синхронизируемые участки.

Пусть p_W и p_R – вероятности того, что произвольно выбранная операция исходной, последовательной формы записи алгоритма будет выполняться в секции кода, синхронизируемой соответственно на запись и чтение. Пусть T – общее число потоков, осуществляющих выполнение алгоритма в параллельной записи. Если T не превышает числа вычислителей (процессоров) в параллельной среде выполнения, то время выполнения алгоритмов в параллельной форме может быть оценено с большой точностью с использованием формулы (1).

$$O_T\left(p_R, p_W, T, n\right) = \max\left(\begin{array}{l} p_W\left(O_s(n) + \varepsilon_W(T)\right) + \dfrac{p_R\left(O_S(n) + \varepsilon_R(T)\right)}{T}, \\ \dfrac{1 - p_R - p_W}{T} O_S(n) \end{array}\right) \quad (1)$$

Здесь используются следующие дополнительные обозначения: $O_S(n)$ – временная сложность алгоритма в исходной последовательной форме; ε_W и ε_R – функции временных издержек, вызванных состязательностью потоков выполнения и общей собственной сложностью алгоритмов синхронизации. В [8, 13] и независимо эмпирически в [10, 11] авторами статьи показано, что при условии непревышения числом потоков числа вычислителей функции $\varepsilon_W(T)$ и $\varepsilon_R(T)$ могут быть выражены линейно с большой точностью. Коэффициенты пропорциональности и смещения зависимостей являются значениями, которые зависят от вычислительной платформы, на которой выполняется анализируемый алгоритм. Поэтому указанные значения должны вычисляться эмпирически для целевой платформы. Левая часть записи (1) имеет минимум функции $T_{кр}$ по T. Им является критическое число потоков, с которого издержки ε_W и ε_R начинают преобладать над выигрышем во времени, получаемым при распараллеливании. Значение, получаемое дифферен-цированием левой части (1), показано в (2), где C_R – смещение ε_R и k_W – коэффициент ε_W:

$$T_{кр} = \sqrt{\frac{C_R + O_S p_R}{p_W k_W}} \qquad (2)$$

Графическое представление (1) показано на рис. 9 для случая, когда $p_R + p_W = 1$. На рис. 10 приведено графическое представление результатов численных исследований предложенного механизма WR и аналога QReadWriteLock, предоставляемого C++ компилятором Nokia Qt, по получению зависимости (1). Очевидна применимость описанного метода оценки временны̀х издержек для синхронизации посредством мьютексов, фьютексов, семафоров с единичным максимальным числом пропускаемых потоков, критических секций и других реализаций примитивов синхронизации, реализующих взаимное исключение. В этих случаях следует рассматривать (1) для $p_R = 0$.

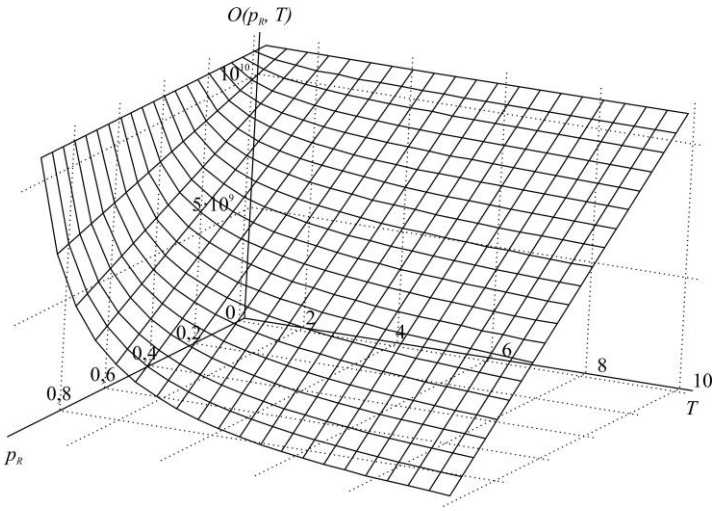

Рис. 9. Графическое представление (1)

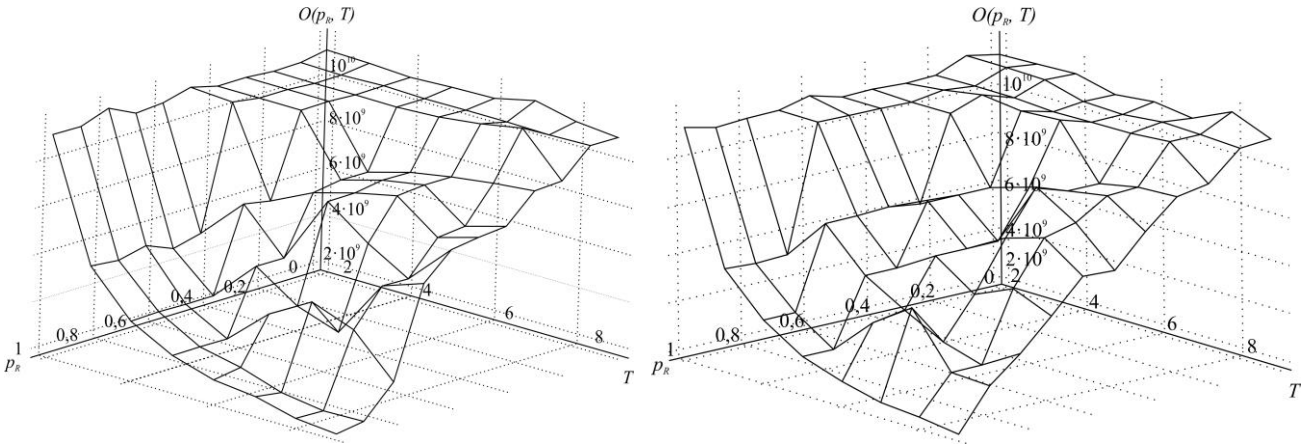

Рис. 10. Зависимость времени выполнения, полученная эмпирически профилированием параллельных алгоритмов, использующих для синхронизации предложенную реализацию WR (слева) и реализацию Nokia Qt (справа)

Заключение

Итак, нами были разработаны общие принципы синтеза архитектур компьютерных систем моделирования физических полей. Системы моделирования, созданные в соответствии с этими принципами, позволяют повысить сложность модели и снизить время вычислений за счет параллельно распределенных компьютерных сред. Разработан механизм синхронизации единиц выполнения, позволяющий снизить общее время вычислений. В статье показано его использование для снижения числа блокировок и состязательности между клиентами за доступ к банку данных, управляемому описанной СУБД. Другой пример применения механизма для реализации несколько иной модели вложенной синхронизации приведен в [10].

Результаты работы были использованы для создания законченных систем моделирования. Одна из них является системой моделирования акустических полей в замкнутых помещениях и подробно описана в [7, 11]. Другая реализация используется в САМ, выполняющей моделирование акустических полей в стохастических подводных волноводах. Она применяется для мониторинга экологической обстановки в океанических акваториях путем анализа распределения поля скорости звука по глубине и сравнения результатов натурных экспериментов с ожидаемым распределением, полученным путем моделирования. Последняя система моделирования описана в [4–6].

СПИСОК ЛИТЕРАТУРЫ

1. Единая система программной документации. Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения: ГОСТ 19.005-85. Изд. июнь 1986, введ. 01.07.1986. М.: Изд-во стандартов, 1988.

2. Норенков И.П. Основы автоматизированного проектирования: учебник для вузов. 4-е изд. М.: Изд-во МГТУ им. Н.Э. Баумана, 2002. 336 с.

3. Савенков К.О. Масштабирование дискретно-событийных имитационных моделей: дис. ... канд. физ.-мат. наук. М., 2007. 110 с.

4. Сальников Б.А., Сальникова Е.Н., Стаценко Л.Г., Чусов А.А. Об устойчивости лучевых траекторий распространения звука в случайно-неоднородном подводном волноводе // Акустические измерения и стандартизация. Ультразвук и ультразвуковая технология. Атмосферная акустика. Акустика океана: сб. тр. науч. конф. «Сессия Научного совета РАН по акустике и XXV сессия Российского акустического общества». Т. 2. М.: ГЕОС, 2012. С. 190–193.

5. Сальников Б.А., Сальникова Е.Н., Стаценко Л.Г., Чусов А.А. Результаты использования методов стохастического моделирования для расчёта зональной структуры акустического поля в двухканальном океаническом волноводе // Акустические измерения и стандартизация. Ультразвук и ультразвуковая технология. Атмосферная акустика. Акустика океана: сб. тр. науч. конф. «Сессия Научного совета РАН по акустике и XXV сессия Российского акустического общества». Т. 2. М.: ГЕОС, 2012. С. 194–196.

6. Чусов А.А. Использование методов распараллеленных и распределённых вычислений для оперативного прогноза уровня стохастичности поля скорости звука при проведении акустических экспериментов по дальнему распространению // Акустические измерения и стандартизация. Ультразвук и ультразвуковая технология. Атмосферная акустика. Акустика океана: Сб. тр. науч. конф. «Сессия Научного совета РАН по акустике и XXV сессия Российского акустического общества». Т. 2. М.: ГЕОС, 2012. С. 322–325.

7. Чусов А.А., Миргородская Ю.В. Программно-алгоритмическое обеспечение «Acoustic Modeler +» для акустического расчета и озвучения помещений // Вологдинские чтения. Владивосток: Изд-во ДВГТУ, 2009. С. 85.

8. Beveridge J., Wiener R., Multithreading applications in Win32. The complete guide to threads, Amsterdam, Addison-Wesley Developers Press, 1997. 397 p.

9. Chusov A.A. The TCP/IP cryptographic client server system for transmitting voice and text // Труды IX международного форума студентов, аспирантов и молодых ученых стран АТР. Владивосток: Изд-во ДВГТУ, 2009. С. 98–103.

10. Chusov A.A., Kovylin A.A., Statsenko L.G., Mirgorodskaya Yu.V., Parallel search for signals with specified cross- and autocorrelation properties on multiprocessor platforms. Radioelectronics and Communications Systems, New-York, Allerton Press, 2011;54(8):425–431.

11. Chusov A.A., Statsenko L.G., Ageeva A.A., Creation of flexible architectures for distributed analysis of physical fields. Pacific Science Review, 2012;14(1):5–9.

12. Hakiri A., Berthou P., Gayraud T., Addressing the Challenge of Distributed Interactive Simulation With Data Distribution Service, The Computing Research Repository (CoRR) abs/1008.3759, 9 p., 2010, URL: http://arxiv.org/ftp/arxiv/papers/1008/1008.3759.pdf

13. Hart J.M., Windows system programming. 4th ed., Addison-Wesley Professional, New-York, 2010, 656 p.

14. IEEE Std 1278.1a-1998, IEEE Standard for Distributed Interactive Simulation Application Protocols, IEEE Computer Society, New-York, 1998.

15. ISO/IEC 14882-2011: Programming languages – C++.

16. ISO/IEC 23270-2006: Programming languages – C#.

17. Lamport L., A new solution of Dijkstra's concurrent programming problem, *Commununications of the ACM*, 1974;17(8):453–455.

18. Liu J., Parallel Discrete-Event Simulation, *School of Computing and Information Sciences, Florida International University*, Feb. 2009, 20 p., URL: http://users.cis.fiu.edu/~liux/research/papers/pdes-eorms09.pdf

19. Peterson G.L., Concurrency and complexity, Tech. Rep. TR59, Computer Science Dep., *Univ. of Rochester*, Rochester, N.Y., Aug., 1979.

20. Reid M.R., An Evaluation of The High Level Architecture (HLA) as a Framework for NASA Modeling and Simulation, The 25th NASA Software Engineering Workshop, Goddard Space Flight Center, Maryland, Nov. 2000, 16 p., URL: http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20010016107_2001018403.pdf

21. Rivest R.L., Pratt V.R., The mutual exclusion problem for unreliable processes: Preliminary report, 17th Annual Symposium on Foundations of Computer Science, Proceedings. Houston, Tex., 1976, p. 1–8.

22. Schneier B., Applied Cryptography. 2nd ed., John Wiley & Sons, 1996, 758 p.

REFERENCES

1. The State Standard 19.005-85: Unification system of program documentation. P-schemes of algorithms and programs. Graphical designations and execution rules, 1988 (in Russ.). [Edinaja sistema programmnoj dokumentacii. P-shemy algoritmov i programm. Oboznachenija uslovnye graficheskie i pravila vypolnenija: GOST 19.005-85. Izd. ijun' 1986, vved. 1986.07.01. M.: Izd-vo standartov, 1988]. (in Russ.).

2. Norenkov I.P. Fundamentals of CAD: Methodology of designing automated systems, Bauman MSTU Publ., Moscow, 2002,. 336 p. (in Russ.). [Norenkov I.P. Osnovy avtomatizirovannogo proektirovanija: uchebnik dlja vuzov. 4-e izd. M.: Izd-vo MGTU im. N.Je. Baumana, 2002. 336 s.]. (in Russ.).

3. Savenkov K.O. Scaling discrete-event simulation models. PhD dissertation (Physics and mathematics), Moscow, Moscow State University, 2007 (in Russ.). [Savenkov K.O. Masshtabirovanie diskretno-sobytijnyh imitacionnyh modelej: dis. ... kand. fiz.-mat. nauk. M., 2007. 110 s]. (in Russ.).

4. Salnikov B.A., Salnikova E.N., Statsenko L.G., Chusov A.A. Stability of sound ray distribution paths in a randomly inhomogeneous underwater waveguide // Acoustical Measurements and Standardization, Ultrasound And Ultrasound Technology, Atmospheric Acoustics, Ocean Acoustics: Proceedings of the Science Conference “Session of the Scientific Council of Russian Academy of Science on Acoustics and XXV Session of the Russian Acoustical Society”, Moscow, GEOS, 2012; Vol. 2, p. 190–193. (in Russ.).

5. Salnikov B.A., Salnikova E.N., Statsenko L.G., Chusov A.A. Results of applying methods of stochastic modeling for calculating a zonal structure of an acoustic field in a two-channel oceanic waveguide // Acoustical Measurements and Standardization, Ultrasound and Ultrasound Technology, Atmospheric Acoustics, Ocean Acoustics: Proceedings of the Science Conference “Session of the Scientific Council of Russian Academy of Science on Acoustics and XXV Session of the Russian Acoustical Society”, Moscow, GEOS, 2012; Vol. 2, p. 194-196. (in Russ.).

6. Chusov A.A. Applying methods of parallel and distributed computations for online-forecasting of sound velocity field stochasticity levels during acoustical experiments of long-distance propagation // Acoustical Measurements and Standardization, Ultrasound And Ultrasound Technology, Atmospheric Acoustics, Ocean Acoustics: Proceedings of the Science Conference “Session of the Scientific Council of Russian Academy of Science on Acoustics and XXV Session of the Russian Acoustical Society”, Moscow, GEOS, 2012; Vol. 2:322–325. (in Russ.).

7. Chusov A.A., Mirgorodskaya Yu.V. Using software tool “Acoustic Modeler +” to calculate acoustics and insonifications of rooms // Proceedings of the “Vologdinskie chteniya” conference, Vladivostok, Far-Eastern National Technical University, 2009; P. 85 (in Russ.). [Chusov A.A., Mirgorodskaja Ju.V. Programmno-algoritmicheskoe obespechenie «Acoustic Modeler +» dlja akusticheskogo rascheta i ozvuchenija pomeshhenij // Vologdinskie chtenija. Vladivostok: Izdatel'stvo DVGTU, 2009. S. 85.]. (in Russ.).

8. Beveridge J., Wiener R. Multithreading applications in Win32. The complete guide to threads, Amsterdam, Addison-Wesley Developers Press, 1997. 397 p.

9. Chusov A.A. The TCP/IP cryptographic client server system for transmitting voice and text // Works of the IX international forum of students, postgraduate students and young scientists of the Pacific Rim, Vladivostok, 2009. P. 98-103 (in Russ.). [Chusov A.A. The TCP/IP cryptographic client server system for transmitting voice and text // Trudy IX mezhdunarodnogo foruma studentov, aspirantov i molodyh uchenyh stran ATR. Vladivostok: Izd-vo DVGTU, 2009. S. 98–103].

10. Chusov A.A., Kovylin A.A., Statsenko L.G., Mirgorodskaya Yu.V. Parallel search for signals with specified cross- and autocorrelation properties on multiprocessor platforms. Radioelectronics and Communications Systems, New-York, Allerton Press, 2011;54(8):425–431.

11. Chusov A.A., Statsenko L.G., Ageeva A.A. Creation of flexible architectures for distributed analysis of physical fields. Pacific Science Review, 2012;14(1):5–9.

12. Hakiri A., Berthou P., Gayraud T., Addressing the Challenge of Distributed Interactive Simulation With Data Distribution Service, The Computing Research Repository (CoRR) abs/1008.3759, 9 p., 2010, Retrieved January 20, 2014, http://arxiv.org/ftp/arxiv/papers/1008/1008.3759.pdf .

13. Hart J.M., Windows system programming. 4th ed., Addison-Wesley Professional, New-York, 2010, 656 p.

14. IEEE Std 1278.1a-1998, IEEE Standard for Distributed Interactive Simulation Application Protocols, IEEE Computer Society, New-York, 1998.

15. ISO/IEC 14882-2011: Programming languages – C++.

16. ISO/IEC 23270-2006: Programming languages – C#.Lamport L., A new solution of Dijkstra's concurrent programming problem, *Commununications of the ACM*, 1974;17(8):453–455.

17. Lamport L., A new solution of Dijkstra's concurrent programming problem, *Commununications of the ACM*, 1974;17(8):453–455.

18. Liu J., Parallel Discrete-Event Simulation, *School of Computing and Information Sciences - Florida International University*, Feb. 2009, 20p., Retrieved January 20, 2014, http://users.cis.fiu.edu/~liux/research/papers/pdes-eorms09.pdf .

19. Peterson G.L., Concurrency and complexity, Tech. Rep. TR59, Computer Science Dep., *Univ. of Rochester*, Rochester, N.Y., Aug., 1979.

20. Reid M.R., An Evaluation of The High Level Architecture (HLA) as a Framework for NASA Modeling and Simulation, The 25th NASA Software Engineering Workshop, Goddard Space Flight Center, Maryland, Nov. 2000, 16 p., Retreived January 20, 2014, http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20010016107_2001018403.pdf

21. Rivest R.L., Pratt V.R., The mutual exclusion problem for unreliable processes: Preliminary report, 17th Annual Symposium on Foundations of Computer Science, Proceedings. Houston, Tex., 1976, p. 1–8.

22. Schneier B., Applied Cryptography. 2nd ed., John Wiley & Sons, 1996, 758 p.