изводствами) / А.М. Пуртов ; рук. работы В.А. Шапцев. — Омск: ИИТПМ СО РАН, 1994. — 138 с.

8. Смелянский, Р.Л. Об одной вероятностной модели программ / Р.Л. Смелянский, А.Г. Бахмуров, Д. Гурьев // Программирование. — 1986. — № 6. — 45—64.

ДУБЫНИН Дмитрий Геннадьевич, советник управления связи и безопасности министерства промышленной политики, транспорта и связи Омской области, аспирант.
Адрес для переписки:e-mail: ddg@omskportal.ru

Статья поступила в редакцию 13.05.2010 г.

УДК 681.3.06

Е. С. ЕРШОВ

Омский государственный технический университет

ОСОБЕННОСТИ РЕАЛИЗАЦИИ ЯДРА СИСТЕМЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ SIMULAB

В статье рассмотрены особенности реализации ядра системы имитационного моделирования Simulab в режиме виртуального модельного времени. В частности, предлагаются методы повышения скорости работы с календарем событий, а также способы эффективного использования оперативной памяти.

Ключевые слова: дискретно-событийное моделирование, агентное моделирование, эффективность.

1. Введение

В настоящее время существует большое количество различных систем имитационного моделирования (СИМ), как узкоспециализированных, так и многоцелевых. В данной статье будет рассматриваться система многоподходного имитационного моделирования Simulab, разработанная группой сотрудников кафедры АСОИУ ОмГТУ. В задачи автора статьи, в частности, входило проектирование и реализация ядра и среды разработки моделей СИМ Simulab. Среди особенностей Simulab можно выделить следующие:

1) Simulab реализует подходы системной динамики, динамических систем, процессного дискретно-событийного и агентного моделирования [1], используя один язык моделирования в единой графической среде разработки;

2) графическая среда разработки моделей значительно ускоряет процесс создания моделей;

3) архитектура системы Simulab позволяет подключать ее библиотеки к любой СИМ, обладающей открытым программным интерфейсом (AnyLogic 6, Repast Symphony и т.п.);

4) использование Simulab по сравнению с аналогами позволяет существенно сократить затраты времени и оперативной памяти на проведение имитационных экспериментов (ИЭ);

5) при проведении реплицированных экспериментов (под репликацией эксперимента подразумевается выполнение нескольких «прогонов» стохастической модели для одного набора значений параметров с дальнейшим усреднением результатов) Simulab полностью использует ресурсы компьютеров, оснащенных многоядерными процессорами, что также позволяет сократить временные затраты в 2—3 раза;

6) Simulab позволяет решать задачи оптимизации имитационных моделей (ИМ), построенных на базе сетей массового обслуживания [2];

7) как сама среда моделирования, так и все ее библиотеки реализованы на языке программирования Java, что позволяет пользователю работать с Simulab в любой операционной системе и на различных по архитектуре процессорах, для которых существует виртуальная Java-машина (Java Runtime Environment / JRE). Более того, пользователю, работающему на удаленном компьютере, предоставляется возможность запускать из сети (Intranet/Internet) как саму среду Simulab, так и созданные в ней модели, без необходимости устанавливать у себя какое-либо программное обеспечение (кроме Java Runtime Environment), при этом процесс моделирования может происходить как на компьютере пользователя, так и на удаленном сервере;

8) в качестве внутреннего языка также используется язык программирования Java. Таким образом, пользователю, знакомому с языками Java, C++ или C# нет необходимости изучать какой-либо специализированный язык моделирования, который зачастую во многом будет ограничивать его возможности. Такой подход позволяет работать с имитационной моделью на уровне языка объектно-ориентированного программирования и пользоваться всеми возможностями языка Java. Кроме того, появляется возможность компилировать модель как в исполняемый файл, так и в апплет, не требующие для запуска среды моделирования;

9) редактор форм, входящий в состав Simulab, поддерживает большой набор интерактивных элементов (меток, кнопок, полей ввода и т.д.) и позволяет создавать интерфейсы любой степени сложности, а также связывать их с ИМ.

Созданные интерфейсы могут использоваться с целью:

— создания управляющих оболочек с возможностью всестороннего контроля над процессом моделирования;

— формирования сложных отчетов, содержащих данные о результатах работы имитационной модели;

— создания сложных оптимизирующих подсистем;

— создания комплексных систем поддержки принятия решений (СППР) на основе имитационных моделей.

Набор стандартных управляющих элементов может быть расширен пользователем. Также поддерживаются средства анализа данных и большой набор элементов бизнес-графики, спроектированных для эффективной обработки и презентации результатов моделирования: статистики, наборы данных, графики, диаграммы, гистограммы;

10) Simulab обладает расширенным набором структур для моделирования больших сетей (таких как Интернет, сеть телефонных переговоров и т.д.) [3];

11) алгоритмы генерации графов, используемые в Simulab, работают в десятки раз быстрее алгоритмов, реализованных в популярной среде для работы с графами Java Universal Network/Graph Framework (JUNG) [4] и среде моделирования AnyLogic 6 [3];

12) 2D/3D анимация в Simulab создается автоматически уже при составлении структуры модели. В общем случае степень «реальности» виртуальной 3D модели, созданной в Simulab, определяется степенью детализации используемых 3D объектов, наличием соответствующего звукового окружения, специальных эффектов и т.п. Так как процессы имитационного моделирования и 3D визуализации являются достаточно ресурсоемкими, пользователю предоставляется возможность создания 2D/3D анимации не только во время имитационного эксперимента (на одном компьютере или нескольких компьютерах, объединенных в сеть), но и при пост-обработке, что значительно снижает нагрузку на компьютер.

2. Особенности реализации ядра СИМ Simulab

Основная задача, которая ставилась при разработке ядра (программы-планировщика для организации выполнения событий имитационной модели в хронологическом порядке) СИМ Simulab—минимизация времени выполнения имитационной модели на компьютере (процессорного времени).

Процессорное время в первую очередь зависит от частоты изменений в моделируемой системе за моделируемый период времени, а также от объема вычислений, связанных с обработкой событий [5]. Таким образом, моделирование сложных систем может потребовать значительных затрат процессорного времени.

Сокращение процессорного времени может быть достигнуто за счет эффективного и полноценного применения аппаратных ресурсов ЭВМ. В частности, решить проблемы ресурсоемкости и вычислительных затрат при имитационном моделировании позволит разработка новых способов построения алгоритмов ИМ и применение эффективных методов программирования.

Одной из важных составляющих ядра любой СИМ является календарь событий (очередь событий, event queue), состоящий в общем случае из элементов, каждый из которых содержит как минимум два поля:

— ссылку на запланированное событие (e_i);

— модельное время, на которое запланировано событие (t_i).

Таким образом, можно сказать, что календарь событий — это список элементов l_i, где $l_i = (e_i, t_i)$.

Ядро СИМ выбирает из календаря событий событие с минимальным временем, которое впоследствии становится текущим модельным временем. Далее ядро присваивает системной переменной с текущим модельным временем минимальное значение времени, на которое запланировано выполнение события и передает ему управление. Если сразу несколько событий запланированы на одно и то же время, то они выполняются последовательно друг за другом, системное время не изменяется до тех пор, пока все эти события не будут обработаны. Далее происходит удаление выполненных событий из календаря событий. Для того чтобы поиск был эффективным, календарь событий упорядочивают по возрастанию времени.

Обработка очередного события e_i также может порождать новое событие e_j, выполнение которого запланировано на некоторый момент времени t_j. Таким образом, в календаре событий появляется новый элемент $l_j = (e_j, t_j)$. В этом случае ядро системы моделирования помещает новый элемент в календарь событий, сохраняя при этом упорядоченность по возрастанию.

Основная проблема дискретно-событийного и агентного моделирования — эффективное использование календаря событий. Традиционно календарь событий реализуется на основе связных списков (linked lists), основное преимущество которых — вставка и удаление элементов занимает время $t = O(1)$, что делает их незаменимыми в случаях, когда необходима быстрая вставка элементов в начало или конец списка. Однако, в случае с календарем событий, проявляется и основной недостаток связных списков — время поиска места для вставки (для сохранения хронологической упорядоченности) пропорционально количеству элементов в данном списке. Большинство алгоритмов, предложенных в [6] и использующих связные списки, различными способами пытаются обойти этот недостаток, например:

1) предпринимается поиск с начала и конца списка;

2) для различных типов событий используют разные списки;

3) список с дополнительным указателем на его середину (элемент равноудаленный от начала и конца списка): при этом сначала происходит сравнение с нужным элементом, а потом поиск происходит в нужной половине списка (с начала и конца подсписка);

4) список событий делится на подсписки, которые соответствуют интервалам системного времени, при этом длины всех интервалов равны фиксированному значению, которое задается пользователем. При планировании нового события вычисляется его индекс в массиве указателей, после чего новый элемент списка событий может быть размещен в соответствующем его подсписке.

К сожалению, использование вышеперечисленных модификаций связного списка не только не приводит к значительному приросту производительности, но и усложняет алгоритм обработки списка.

Как показывает практика, при реализации календаря событий наиболее эффективными по сравнению со связными списками являются алгоритмы на основе самобалансирующихся бинарных деревьев поиска (self-balancing binary search trees). Вызвано это прежде всего тем, что и вставка (сортировка узлов осуществляется автоматически) и удаление

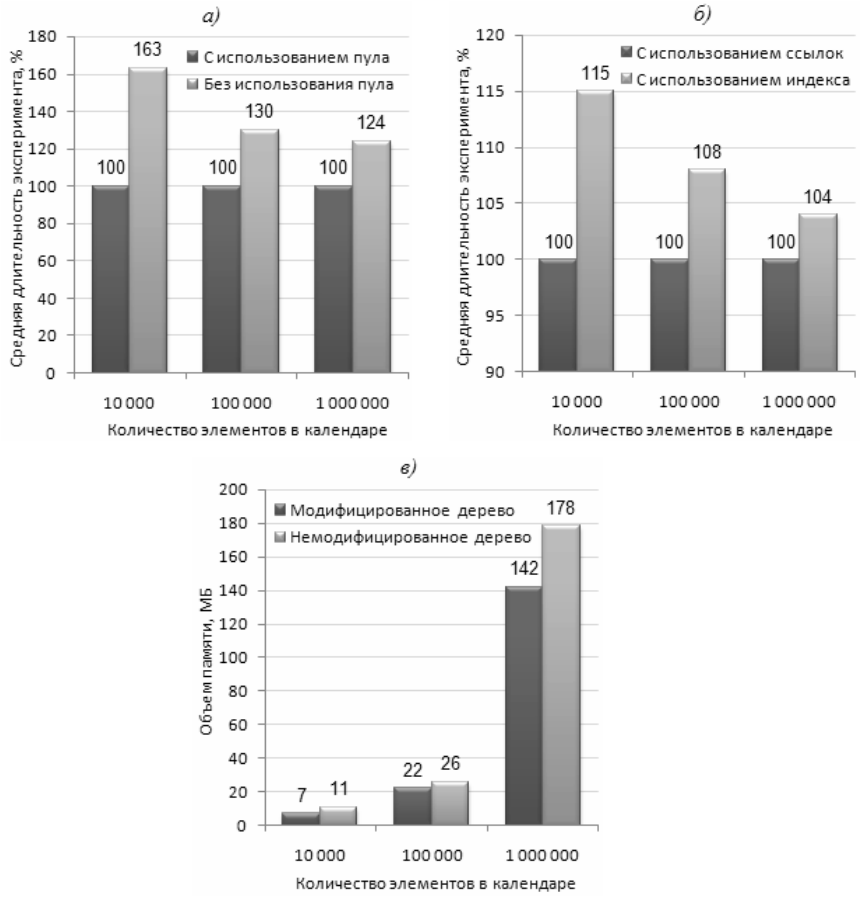

Рис. 1. Эффективность реализаций календаря событий с использованием пула элементов (а), с использованием ссылок на экземпляры агентов (б) и с модификацией красно-черного дерева (в)

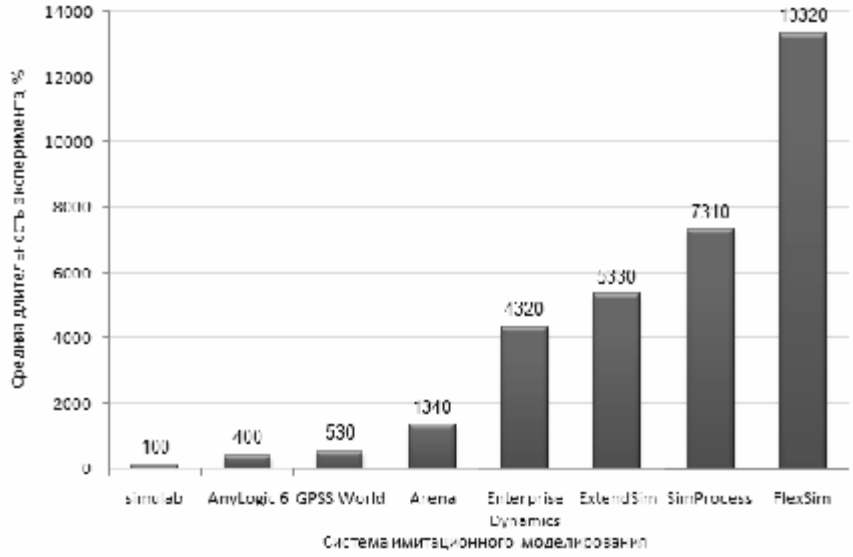

Рис. 2. Сравнение Simulab с современными системами ИМ по быстродействию в batch-режиме процессного моделирования

и поиск узла занимает логарифмическое время $t = O(log_2 N)$, где N — количество узлов в дереве (далее под узлом дерева будем подразумевать элемент календаря событий). Как правило, используют красно-черные деревья (red-black trees) [7], популярность которых связана с тем, что на них часто достигается лучшее соотношение между степенью сбалансированности (дерево сбалансировано, если путь от корня до каждого узла имеет длину, не превышающую $O(log_2 N)$) и сложностью ее поддержки. Сбалансированность достигается за счет введения дополнительного атрибута узла дерева — «цвет». Этот атрибут может принимать одно из двух возможных значений — «черный» или «красный». Красно-черное дерево обладает следующими свойствами:

— все листья черны;

— все потомки красных узлов черны (т.е. запрещена ситуация с двумя красными узлами подряд);

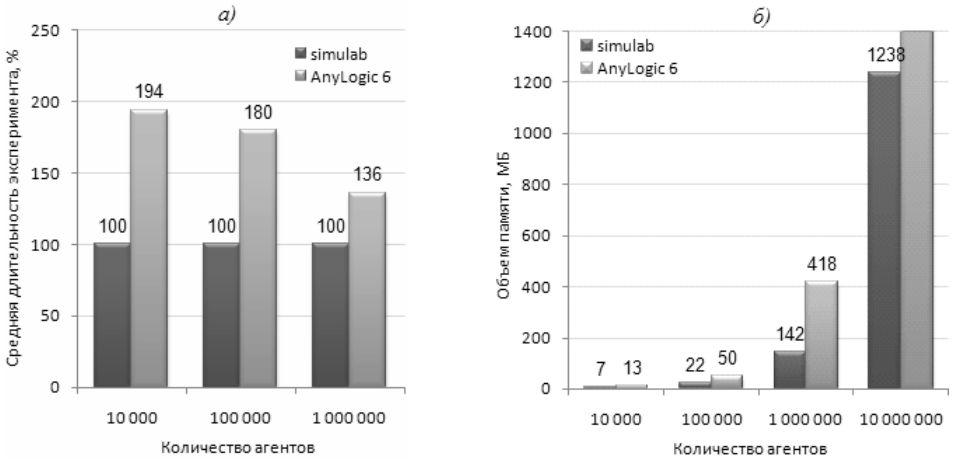

Рис. 3. Сравнение Simulab с системой ИМ AnyLogic 6 по быстродействию в агентном моделировании (а) и по использованию оперативной памяти (б)

— на всех ветвях дерева, ведущих от его корня к листьям, число чёрных узлов одинаково (что и обеспечивает сбалансированность дерева).

Далее приведено несколько приемов, использованных при реализации ядра СИМ Simulab, и в совокупности позволяющих в несколько раз повысить скорость работы с календарем событий на базе красно-черного дерева:

1) как было отмечено ранее, в общем случае время вставки нового узла в красно-черное дерево составляет $t = O(log_2 N)$ (что имеет место, например, при вставке узла с минимальным или максимальным значением ключа). Однако, используя специальные указатели на узлы дерева с минимальным и максимальным значениями ключа, можно сократить время вставки в начало/конец календаря событий до $t = O(1)$;

2) для календаря событий на каждом шаге моделирования характерно добавление и удаление одного или нескольких элементов. Необходимо отметить, что в этом случае происходит создание/удаление новых экземпляров объектов (элементов календаря), что сильно сказывается на быстродействии календаря событий (рис. 1а). Для решения этой проблемы предлагается применять специальный пул элементов, который используется следующим образом: при удалении элемента из календаря событий он не уничтожается, а помещается в специально выделенный для этого массив. При добавлении в календарь событий нового элемента, ядро системы моделиро-вания сначала проверяет наличие в пуле удаленных элементов, и если таковые имеются, то использует один из них, в противном случае – создает новый элемент;

3) в частном случае в задачи ядра СИМ также входит хранение массива заявок, находящихся в моделируемой системе, а также управление этим массивом. Однако доступ к заявкам по индексу сопряжен с дополнительными временными расходами (рис. 1б), вызванными постоянной проверкой на выход за границы массива. Для решения этой проблемы предлагается хранить в элементе календаря событий ссылки на экземпляр заявки, связанной с событием;

4) стандартная реализация красно-черного дерева не предусматривает возможности хранения узлов с одинаковым значением ключа, что приводит к необходимости реализовывать в каждом узле, по меньшей мере, связный список, предназначенный для хранения событий, запланированных на одно время. Это приводит к дополнительным затратам памяти (рис. 1в), поскольку мы вынуждены хранить ссылку на этот связный список, возможно, так никогда и не воспользовавшись им. Для решения этой проблемы предлагается модифицировать алгоритм вставки узлов в дерево таким образом, чтобы новые узлы со значением ключа, равным значению ключа существующего узла, рассматривались как узлы с меньшим значением ключа.

При реализации ядра СИМ также необходимо уделять внимание эффективному использованию оперативной памяти. Это особенно важно в агентном моделировании, поскольку при большом количестве (более миллиона) агентов календарь событий будет содержать миллионы элементов, при этом каждый агент будет обладать своими уникальными свойствами и автономным поведением (принятие решения в соответствии с некоторым набором правил, взаимодействие с окружающей средой и другими агентами, самоизменение и т.п.).

Далее приведено несколько приемов реализации класса агента, которые позволили значительно уменьшить затраты памяти в СИМ Simulab:

1) уменьшение количества полей, необходимых для описания базовых свойств агента. Очевидно, что при сокращении количества полей описывающих базовые свойства агента сократится и объем памяти, занимаемой всеми экземплярами данного агента. Добиться этого сокращения можно несколькими способами, например, следует отказаться от полей, значения которых при необходимости могут быть вычислены через другие поля.

Отказ от полей, которые не используются в текущем режиме моделирования, также позволит освободить память. Например, при моделировании в режиме виртуального модельного времени полностью отпадает необходимость в полях и методах, используемых только при визуализации агентов;

2) отказ от использования в качестве полей простых классов (таких как Point, Dimension и т.п.). Необходимо отметить, что, в этом случае при описании свойств агента в оперативной памяти будут храниться не только поля экземпляра класса, но и ссылка на этот экземпляр. Например, отказ от использования класса Point для описания текущего положения агента в непрерывном двухмерном пространстве и, соответственно, использование двух полей типа double, содержащих координаты, позволяет избежать неоправданных затрат памяти на хранение ссылок.

Как показали многочисленные эксперименты с дискретно-событийными, агентными и многоподходными моделями, ядро СИМ Simulab по сравнению с существующими аналогами позволяет:

— существенно сократить время на проведение ИЭ в режиме виртуального модельного времени: в 4 раза для дискретно-событийного моделирования (рис. 2) и в 1,5 — 2 раза для агентного моделирования (рис. 3а);

— сократить объем памяти, необходимой для проведения ИЭ (преимущественно в агентных моделях), в 2 — 3 раза (рис. 3б).

Нужно отметить, что данные результаты достигнуты при полной функциональной совместимости с аналогами и без использования каких-либо технологий распределенных вычислений.

3. Заключение

Использование СИМ Simulab (или отдельных ее библиотек в составе других СИМ) позволит разработчикам:

— сократить время на разработку ИМ для решения задач в области производства, логистики и цепочек поставок, в сфере обслуживания, пешеходных и транспортных потоков, информационно-телекоммуникационных сетей, социальных и экологических процессов и систем и т.п.;

— по сравнению с аналогами существенно сократить затраты времени и оперативной памяти на проведение ИЭ;

— полностью использовать ресурсы многоядерных процессоров при проведении реплицированных ИЭ;

— оптимизировать сложные ИМ на базе сетей массового обслуживания;

— значительно упростить разработку СППР на основе ИМ.

Библиографический список

1. Карпов, Ю.Г. Имитационное моделирование систем. Введение в моделирование с AnyLogic 5 / Ю. Г. Карпов. — СПб.: БХВ-Петербург, 2005. — 400 с.

2. Задорожный, В.Н. Градиентный метод и программа оптимизации сетей с очередями / В.Н. Задорожный, Е.С. Ершов // Омский научный вестник. — 2009. — № 2(80). — С. 205 — 210.

3. Ершов, Е.С. Система имитационного моделирования Simulab / Е.С. Ершов, Е.Б. Юдин // Имитационное моделирование. Теория и практика: матер. 4-й Всеросс. конф. ИММОД-2009. — СПб., 21 — 23 октября 2009 / ЦТСС. — СПб., 2009. — Т. 1. — С. 257 — 261.

4. Java Universal Network / Graph Framework Overview. — URL: http://jung.sourceforge.net/index.html. [Дата обращения: 04.05.2010].

5. Окольнишников, В.В. Представление времени в имитационном моделировании / В.В. Окольнишников // Вычислительные технологии. — Сибирское отделение РАН, 2005. — Т. 10. — № 5. — С. 57 — 77.

6. Миков, А.И. Распределенные системы и алгоритмы / А.И. Миков, Е.Б. Замятина. — URL: http://www.intuit.ru/department/algorithms/distrsa. Дата обращения: 04.05.2010.

7. Cormen, T. H. Introduction to Algorithms. Second Edition / T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. — MIT Press and McGraw-Hill, 2001. — P. 273 — 301.

ЕРШОВ Евгений Сергеевич, аспирант, ассистент кафедры «Автоматизированные системы обработки информации и управления».
Адрес для переписки: e-mail: eugene.ershov@mail.ru

Статья поступила в редакцию 04.06.2010 г.

Книжная полка

004.7/Х79

Хорев, П. Б. Методы и средства защиты информации в компьютерных системах [Текст]: учеб. пособие для вузов по направлению 230100 (654600) «Информатика и вычислительная техника» / П. Б. Хорев.—4-е изд., стер.— М.: Академия, 2008.—254, [1] с.: рис., табл.—(Высшее профессиональное образование).—Библиогр.: с. 251-252.— ISBN 978-5-7695-5118-5.

Основное внимание в учебном пособии уделено программным и программно-аппаратным методам и средствам защиты информации. Рассмотрены, в частности, модели безопасности, используемые в защищенных версиях операционной системы Windows (в том числе использование функций криптографического интерфейса приложений CryptoAPI) и операционных системах «клона» Unix.

004/М74

Могилев, А. В. Информатика [Текст]: учеб. пособие для вузов по пед. специальностям / А. В. Могилев, Н. И. Пак, Е. К. Хеннер; под ред. Е. К. Хеннера.—7-е изд., стер.—М.: Академия, 2009.—840, [1] с.: рис., табл.—(Высшее профессиональное образование).—ISBN 978-5-7695-6342-3.

Содержатся обширные сведения по теоретическим основам информатики, программному обеспечению, языкам и методам программирования, вычислительной технике, информационным системам, компьютерным сетям и телекоммуникациям, компьютерному моделированию и социальной информатике.

004/Ч-46

Черемушкин, А. В. Криптографические протоколы. Основные свойства и уязвимости [Текст]: учеб. пособие для вузов по специальности «Компьютерная безопасность»/А. В. Черемушкин.—М.: Академия , 2009.—271, [1] с.: рис., табл.—(Высшее профессиональное образование).—Библиогр.: с. 264-270.—ISBN 978-5-7695-5748-4.

Изложены основные принципы построения криптографических протоколов, подробно описаны свойства протоколов. Рассмотрено большое число примеров, демонстрирующих типовые уязвимости и их влияние на свойства протоколов.