ОСОБЕННОСТИ РЕАЛИЗАЦИИ ЯДРА СИСТЕМЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ SIMULAB

Е. С. Ершов (Омск)

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.

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 [1]. Таким образом, моделирование сложных систем может потребовать значительных затрат процессорного времени.

Сокращение процессорного времени может быть достигнуто благодаря эффективному и полноценному применению аппаратных ресурсов ЭВМ. В частности, решить проблемы ресурсоемкости и вычислительных затрат позволит разработка новых способов построения алгоритмов имитационного моделирования 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 реали-

зуется 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 [2] 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 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 перечисленных 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 самобалансирующих бинарных деревьев поиска (self-balancing binary search trees). Вызвано это прежде всего тем, что и вставка (сортировка узлов осуществляется автоматически), и удаление, и поиск узла занимают 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) [3], 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».

Приведем 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. 1, *а*). 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 моделирования сначала прове-

ряет 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. 1, *б*), 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. 1, *в*), 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.

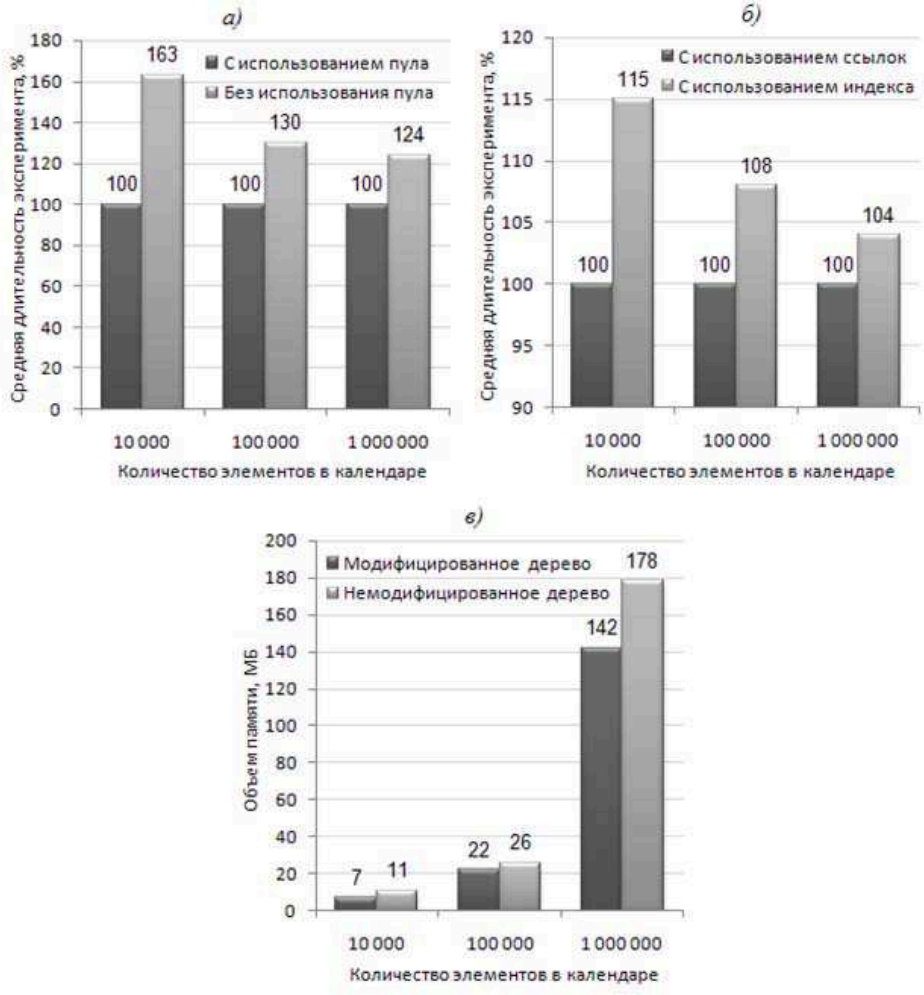

Рис. 1. Эффективность реализаций календаря событий с использованием пула элементов (*а*), с использованием ссылок на экземпляры агентов (*б*) и с модификацией красно-черного дерева (*в*)

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.).

Приведем 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.). 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. 3, *а*);

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

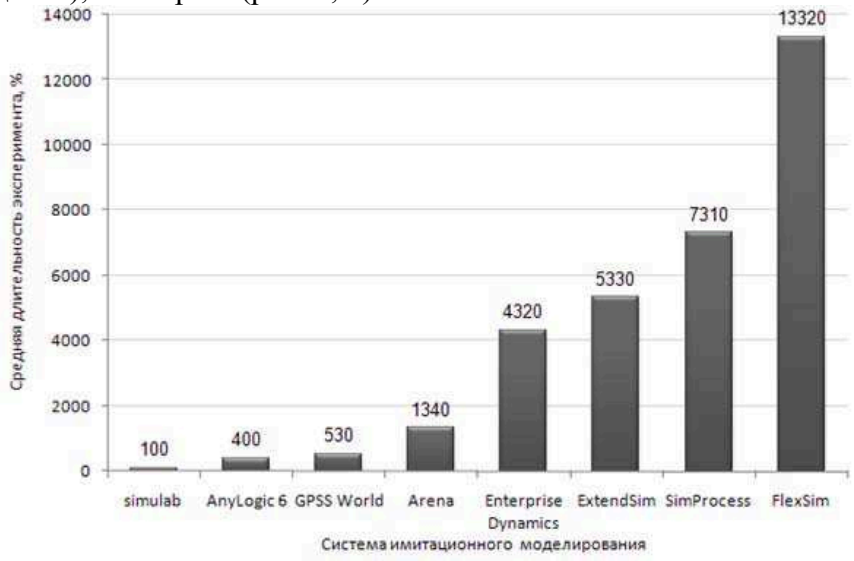

Рис. 2. Сравнение Simulab с современными системами ИМ по быстродействию в batch-режиме процессного моделирования

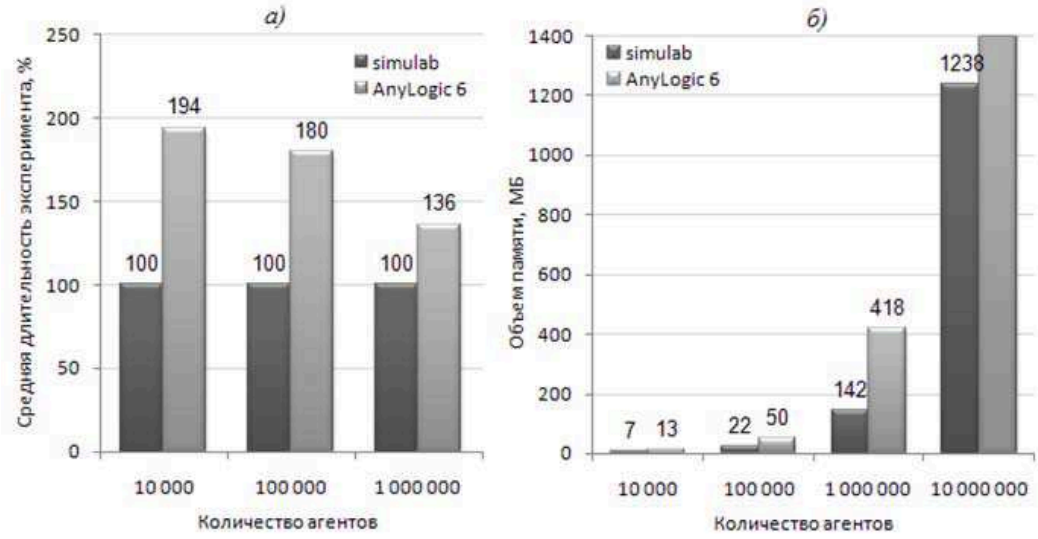

Рис. 3. Сравнение Simulab с СИМ AnyLogic 6 по быстродействию в агентном моделировании (а) и по использованию оперативной памяти (б)

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

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 СИМ на базе сетей массового обслуживания;

– значительно упростить разработку СППР на основе имитационного моделирования.

Литература

1. **Окольнишников В. В.** Представление времени в имитационном моделировании//Вычислительные технологии. Сибирское отделение РАН, 2005. Том 10. №5. С. 57–77.
2. **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.
3. **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.