А.И. Миков, Н.З. Нгуен

Кубанский государственный университет

alexander_mikov@mail.ru, duycoi90@gmail.com

ИССЛЕДОВАНИЕ ПАРАЛЛЕЛЬНЫХ ПРОЦЕССОВ ПРИ ИМИТАЦИОННОМ МОДЕЛИРОВАНИИ СЕТЕЙ БОЛЬШОГО МАСШТАБА

Введение

Процесс имитационного моделирования дискретных систем является довольно затратным по времени. Для получения достаточно статистически устойчивых оценок параметров исследуемой системы требуется провести много имитационных экспериментов. При сложной архитектуре системы, например сети массового обслуживания большого размера, обрабатываются десятки и сотни миллионов событий, поэтому, очень важной является *оптимизация построения симулятора* – основной системной программы, обеспечивающей управление очередями событий. Ускорению работы может способствовать распараллеливание процессов при наличии нескольких устройств обработки.

В данной работе рассматриваются *событийно-ориентированные системы моделирования*, в которых

- продвижение модельного времени происходит от события к событию;
- произошедшие события определяют последовательность смены состояний, связанных с наступлением этих событий.

При анализе работы систем имитационного моделирования различают [1, 2]: *физическое* время, используемое в моделируемой физической системе, *модельное* время (представление физического времени в модели) и *процессорное* время (время работы симулятора).

Во время работы имитационной модели формируется *список событий*, упорядоченных *по возрастанию времени* их наступления. Кро-

ме моментов времени их наступления события характеризуются *причинно-следственными связями* между ними, т.е. существуют события, зависящие друг от друга, и, как правило, они не могут обрабатываться одновременно (параллельно), а только в порядке, определяемом соответствующими причинно-следственными связями. Таким образом, в любой момент процессорного времени в системе имитационного моделирования имеется *множество запланированных событий*, на котором определены два отношения: линейное *отношение предшествования по времени* и частичное *отношение причинно-следственных связей*.

Если система имитационного моделирования исполняется на компьютере с n процессорами, то это множество должно быть разделено, по возможности, на n равных частей для достижения максимального ускорения процесса. Основная сложность заключается в том, что множество динамично, оно изменяется с каждым новым обработанным событием, поэтому алгоритм распределения событий на обработку по процессорам должен быть очень экономичным.

В работе приводятся три различных алгоритма распределения, и производится анализ их эффективности на примерах сетей массового обслуживания (СМО) определённой конфигурации.

Алгоритм и модель

Принцип алгоритма распределения событий для обработки по n процессорам заключается в следующем. В начальный момент модельного времени в системе одновременно появляется определенное количество событий, запись о каждом из них имеет следующие поля: *номер* (идентификатор) *события*; *время наступления события*; *порядковый номер заявки*, к которому это событие относится; *номер узла*, на котором эта заявка обрабатывается. Эти записи образуют *список запланированных событий*, упорядоченных по моменту их появления. Так как нужно обработать эти события по порядку, то очевидно, что должен быть определен *таймер модельного времени* для фиксации текущего модельного времени. Следует отметить, что условием, вызывающим приращение таймера, является наступление времени «ближайшего события». Ближайшее событие – это то событие, возникновение которого запланировано на момент времени, равный минимальному значению модельного времени в списке запланированных событий. Это делается для того, чтобы избежать обработки промежуточных моментов времени, на которые не планируется выполнение никаких событий.

После получения списка событий в некоторый определенный момент времени t необходима их обработка n процессорами. Также нужно отметить, что обработка событий происходит в процессорном вре-

мени, которое не связано однозначно с модельным временем. Приращение таймера процессорного времени происходит тогда, когда все события в момент *t* ещё не будут обработаны, а также, когда все процессоры в этот момент заняты. Кроме этого нужно учесть, что в один и тот же момент модельного времени могут появиться события, обладающие причинно-следственной связью. Следовательно, приращение можно также получить, если оставшиеся в списке события зависят от тех, которые в данный момент находятся в обработке. Моделирование завершится в тот момент, когда количество обработанных заявок превысит некоторое число, заданное пользователем.

Имея *n* процессоров для обработки событий, можно определить следующие методы их распределения в системе:

1. Все *n* процессоров – общие для всех узлов модели.
2. Модель делится на *k* частей (в зависимости от количества узлов *m*). Имеющиеся процессоры распределяется поровну между этими частями, т.е. на каждый узел приходится *n/m* процессоров.
3. Узлы имеют одинаковое количество процессоров на обработку, т.е. имея *m* узлов и *n* процессоров, получаем, что каждый узел имеет по *n/m* процессоров для обработки.

Используя эти методы, сравним результаты работы соответствующих им алгоритмов для различных моделей.

Определим класс моделей для последующего анализа.

Рассмотрим класс сетей СМО с ожиданием (неограниченной очередью), размерностью *k*×*k* следующего вида (рис. 1), где круг – это узел или сервер, прямоугольник – очередь с неограниченным количеством мест для ожидания.

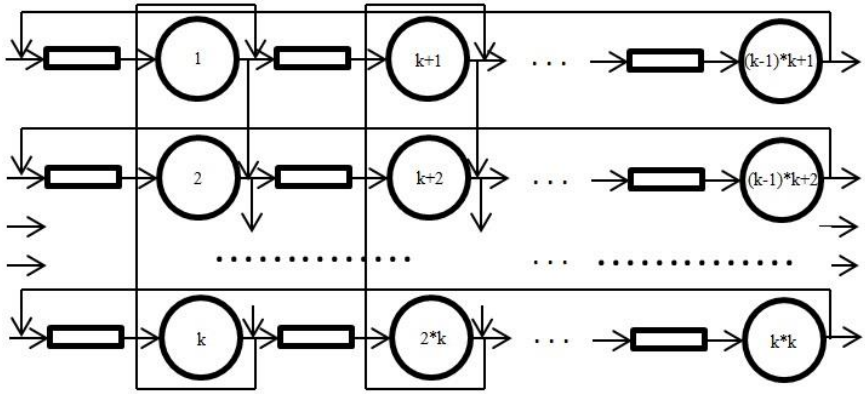

Рис. 1. Структура СМО размерности *k*×*k* с ожиданием

Следует отметить, что каждый узел имеет три вида событий, связанныз с обслуживанией на нём заявки, при наступлении которых в какой-либо момент времени система изменяет свое состояние: приход заявки; начало обработки заявки; конец обработки заявки.

Каждый *i*-й узел данной сети имеет два пути передачи заявки на последующую ее обработку, подчиняясь следующему закону:

$$W_{1,2}(i) = \begin{cases} (k+i;\, i+1-k), \text{если } i \ mod \ k = 0 \\ (i-(k-1)\times k;\, 0), \text{если } i > (k-1)\times k \\ (k+i;\, k+i+1), \text{в противном случае} \end{cases}.$$

Здесь $W_{1,2}(i)$ – функция перехода, значением которой является пара чисел, эквивалентных соответствующим узлам системы, где ‘0’ означает выход из системы. Следует отметить, что передача заявки происходит по одному из двух путей с равной вероятностью.

Проведем анализ поведения данного класса систем размерности $k \times k$ ($k = \overline{2,6}$) при использовании трех методов и при обработке 1 млн. заявок. На рис. 2 приведены графики зависимости времени *t* обработки заявок от количества процессоров *n*. Время указано в миллионах условных единиц.

Исследуя данные графики можно утверждать, что метод 1 работает лучше, чем два остальных алгоритма, и стремится к минимальному значению времени обработки быстрее, нежели другие. Для структур с $k = 2$, $k = 4$, $k = 6$ видно, что при увеличении числа процессоров в 2 раза, имея изначально 1 процессор, используя метод 1, скорость выполнения программы можно увеличить приблизительно в 2 раза, тогда как метод 2 даёт уменьшение времени только в 1.5 раза. Также несложно заметить, что минимальное время выполнения можно получить при наличии в системе количества процессоров, равного удвоенному количеству узлов (серверов) в структуре.

Следует отметить, что важной частью исследования является также анализ загрузки и дисбаланса процессоров в течение всего времени моделирования (рис. 3).

Здесь видно, что при использовании метода 1 и метода 2 проявляется большой дисбаланс нагрузки, тогда как метод 3 даёт довольно незначительный дисбаланс. Из этого следует, что метод 3 лучше использует ресурсы имеющихся процессоров. Но значительный минус этого метода в том, что он не может работать при количестве процессоров меньшем количества серверов в структуре сети, так как он может работать только в том случае, когда в каждом сервере находится как минимум 1 процессор для обработки событий.

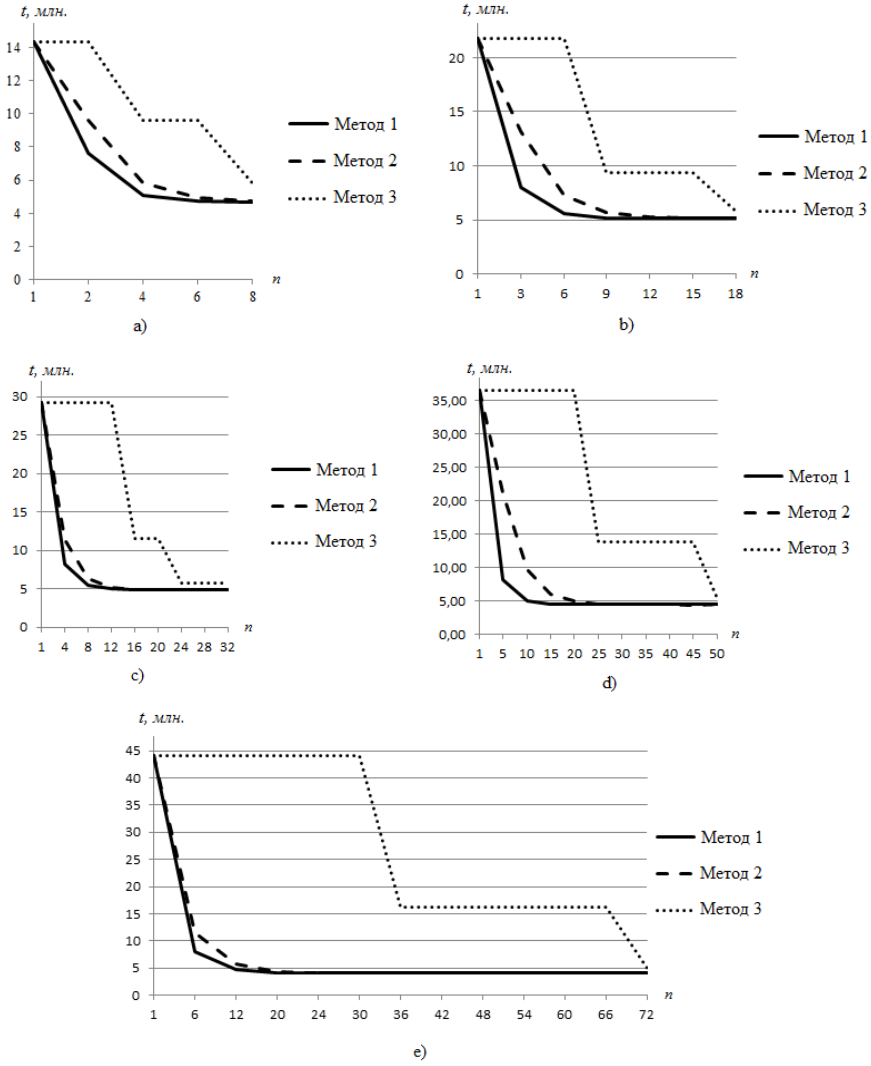

Рис. 2. Зависимости времени обработки от количества процессоров для систем размерности a) 2×2, b) 3×3, c) 4×4, d) 5×5, e) 6×6

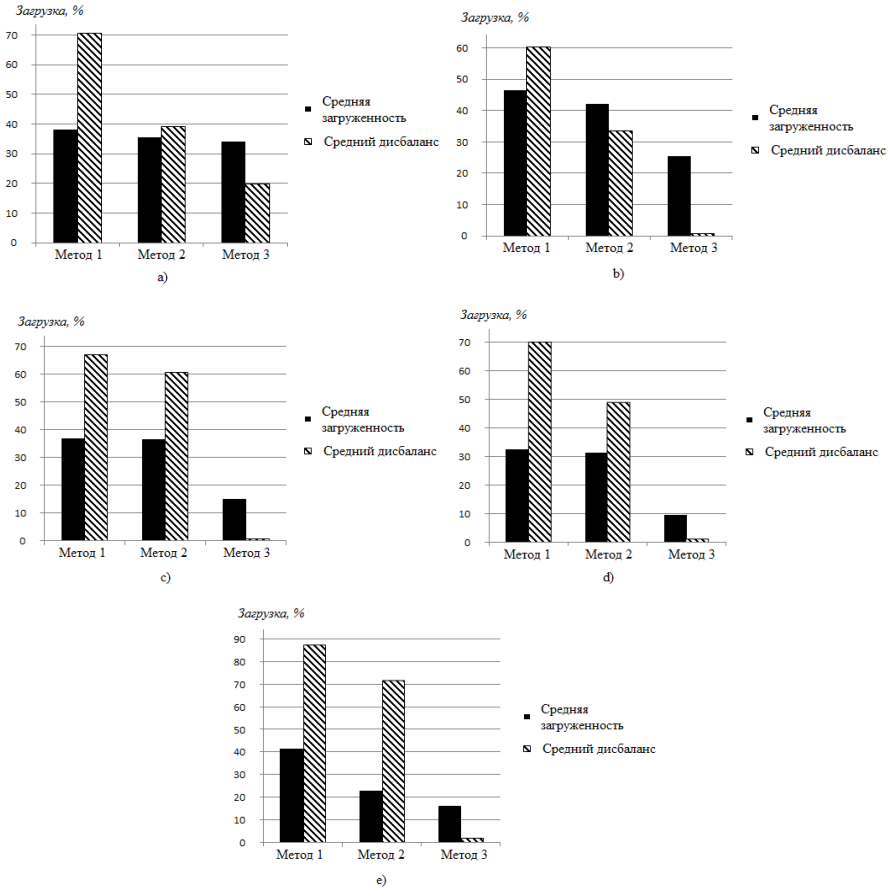

Рис. 3. Диаграммы средней загрузки и дисбаланса процессоров при моделировании систем различной размерности:
a) 2×2, b) 3×3, c) 4×4, d) 5×5, e) 6×6

Анализируя данные, приведённые в табл. 1, заметим, что обслуживание одного миллиона заявок вычислительная структура с параметром $k = 6$ выполнит быстрее, чем другие. При этом нужно учесть, что она обрабатывает событий больше, нежели другие структуры, меньшие по масштабу. Кроме того, нельзя оставить без внимания тот факт, что количество ресурсов, затрачиваемых на обработку, соответственно выше.

Таблица 1. Объем работы для систем разной размерности

Параметр структуры, k	Количество событий, обработанных в системе, млн.	Минимальное время моделирования, млн. ед.
2	14,34	4,648
3	21,74	5,115
4	29,18	4,889
5	36,58	4,452
6	44,03	4,051

Заметим, что при обработке 1 млн. заявок получаем уменьшение времени по отношению к последовательной обработке: при $k = 2$ в 3 раза; при $k = 3$ в 4,25 раза; при $k = 4$ в 5,96 раза; при $k = 5$ в 8,21 раза; при $k = 6$ в 10,86 раза.

Заключение

В данной работе представлен сравнительный анализ трех методов обработки списка событий, планируемых во время работы имитационной модели СМО. Было обнаружено, что при малом количестве процессоров (меньшем хотя бы удвоенного количества серверов в модели) лучшие результаты даёт использование метода 1, т.к. он позволяет получить минимальное время обработки с наименьшими затратами ресурсов. В то же время для получения лучшего баланса загрузки процессоров с минимальными затратами метод 2 подходит лучше. При наличии большого количества процессоров все три метода достигают минимального предела времени работы модели, и для обеспечения наилучшего баланса загрузки процессоров следует использовать метод 3. Просчитав результаты работы системы, начиная со структуры 3×3, можно сделать вывод о том, что при увеличении масштаба сети мы получаем большую производительность.

Библиографический список

1. *Миков А.И., Замятина Е.Б.* Инструментальные средства имитационного моделирования для анализа бизнес-процессов и управления рисками // Информатизация и связь. 2011. № 3. С. 14–16.
2. *Замятина Е.Б., Миков А.И.* Программные средства системы имитации Triad.NET для обеспечения ее адаптируемости и открытости // Информатизация и связь. 2012. №5. С. 130–133.