электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.942

Дискретно-событийное имитационное моделирование

Леонтьев А.В., *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»

научный руководитель: Чёрненький М.В., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»
chernen@bmstu.ru

Основные понятия

Имитационное моделирование – это метод, позволяющий строить модели, описывающие процессы так, как они проходили бы в действительности. Такую модель можно «проиграть» во времени как для одного испытания, так и заданного их множества. При этом результаты будут определяться случайным характером процессов. По этим данным можно получить достаточно устойчивую статистику.

Дискретно-событийное имитационное моделирование – это подвид имитационного моделирования, в основе которого лежат события, которые выполняются в дискретном модельном времени.

Событие – это именованная сущность. На срабатывание события можно подписаться, а также возможно инициировать срабатывание события.

Модельное время – это неотрицательное число, увеличивающееся, либо не изменяющееся между обработкой событий. Реальное процессорное время никоим образом не связано с эти понятием, в процессе обработки события модельное время не изменяется.

Класс одновременных событий (КОС) – это множество событий, обработка которых осуществляется при неизменном значении модельного времени.

Подходы к реализации

Существуют два основных подхода к реализации: с использованием распараллеливания (выполнение процесса симуляции на нескольких потоках, процессах или компьютерах) и без.

В первом случае несколько процессов или событий, составляющих один КОС, могут обрабатываться одновременно в прямом смысле этого слова. При первом взгляде

кажется, что это даст сильный прирост производительности, ведь несколько событий могут обрабатываться параллельно. Но это не совсем так. Дело в том, что как правило, симулируемые процессы связаны между собой и конкурируют за какие-то ресурсы. Причём существуют два типа таких ресурсов: те, что связаны непосредственно с предметной областью, например, многоканальные устройства и очереди, и те, которые связаны с реализацией моделирующей системы, например, очереди сообщений, очередь событий и другие общие объекты. Следует понимать, что переключение контекста процессора не является мгновенным, и при очень большом количестве потоков, заблокированных в ожидании общего ресурса, большую часть времени процессор будет занят переключением потоков.

Во втором случае все события исполняются последовательно. Что касается событий, составляющих один КОС, то они также исполняются последовательно, однако модельное время остаётся неизменным. Модель не должна зависеть от порядка обработки событий одного КОС.

Суть алгоритма симуляции состоит в последовательном извлечении событий из очереди, присвоении модельному времени значения метки времени этого события и вызова обработчиков события.

Основные компоненты реализации

Очередь событий

Симуляция начинается с того, что в эту очередь кладётся первое событие, инициирующее все последующие. Симуляция завершается, когда эта очередь опустеет, либо истечёт указанный период симуляции. События могут быть как срочными, так и запланированными к выполнению в будущем. Обработка событий в очереди должна учитывать временную метку события. Таким образом события в очереди должны быть приоритезированы по времени. Для её реализации хорошо подходит двоичная куча. Сложность добавления события в очередь и извлечения оттуда составляет $O(\log n)$, где n - количество событий в очереди.

События

События характеризуются названием, меткой времени, когда оно должно быть обработано и множеством обработчиков - процессов. Событие можно создать, подписаться на его обработку и инициировать её.

Обработчики событий

Обработчики событий - это функции, которые вызываются при срабатывании связанного с ними события.

Модельное время

Модельное время - это неотрицательное число, изменяющееся в процессе симуляции. В начальный момент времени (при обработке стартового события) оно равно нулю.

Алгоритм изменения модельного времени крайне прост: извлекаем из очереди ближайшее событие и присваиваем значения его метки времени модельному времени.

Состояние

Состояние - это множество объектов, отражающих текущие значения свойств процессов и модели вообще.

Генератор случайных чисел

Моменты возникновение событий и их обработки являются случайными во времени. Для этого используются генераторы случайных или псевдослучайных чисел (ГПСЧ). Для моделирования при использовании ГПСЧ важно, чтобы период генерации был как можно больше. В качестве генератора можно использовать алгоритм «Вихрь Мерсенна». Его период равен числу Мерсенна, которое равно $2^{19937} - 1$.

Статистика

Целью симуляции является сбор статистики: динамика размеров очередей, времени обработки и т.д. При реализации статистики важно понимать, что сохранить в памяти все данные и обработать их по завершению симуляции, как правило, нельзя, так как их объём будет слишком велик. То есть данные нужно обрабатывать «на лету» прямо в процессе симуляции.

Существующие реализации

Рассмотрим две программных реализации событийно-ориентированных систем имитационного моделирования.

simpy

simpy - это библиотека, написанная на языке Python. Её основное преимущество состоит в применении генераторов, что позволяет существенно упростить код.

Например, вот так выглядит код, изменяющий цвет светофора каждые 10 единиц времени:

```
while True:
color = 'green'
yield env.timeout(10)
color = 'red'
```

По сути, присвоение красного цвета представляет собой обработчик события временной задержки на 10 единиц, однако код остаётся линейным.

Кроме базовых понятий, таких как Процесс и Событие данная библиотека предоставляет пользователю набор ресурсов, за доступ к которым могут конкурировать процессы. К ним относятся:

Resource - многоканальное устройство с очередью;

Container - ёмкость, в которую можно либо добавить некоторое количество ресурса (например, бензин, при моделировании заправки), либо наоборот - изъять, причём обе операции формируют очереди запросов;

Store - аналогичен Container, но положить и достать из него можно реальные объекты Python.

simjs

simjs - это библиотека для языка JavaScript, она аналогична **simpy**, однако на момент её реализации в JS ещё не было генераторов, и поэтому код моделей, написанных с использованием этой библиотеки несколько сложнее, чем для **simpy**. Вместо ключевого слова **yield** здесь используются функции обратного вызова (callbacks). Кроме тех возможностей, которые есть в simpy, в simjs также есть несколько классов для работы со статистикой:

DataSeries - представляет последовательность измерений с весом;

TimeSeries - DataSeries, где в качестве веса используется интервал времени, между измерениями;

Population - подсчитывает характеристики очереди, используя DataSeries и TimeSeries.

Реализация макета

В рамках дипломной квалификационной работы мною была разработана библиотека дискретно-событийного имитационного моделирования, а также веб-сервис, позволяющий производить имитационное моделирование систем массового обслуживания (СМО).

Библиотека разработана на языке Java и аналогична по набору сущностей и принципу работы с simjs и simpy. Однако Java является гораздо более производительной (в плане вычислений) платформой в сравнении с NodeJS и Python.

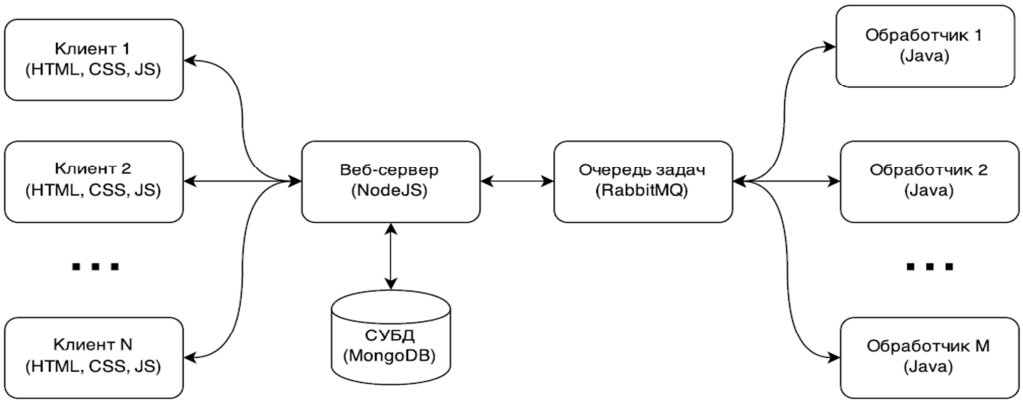
Веб-сервер, принимающий запросы на симуляцию от пользователя написан на NodeJS с использованием фреймворка express. Сервер выполняет 2 задачи: взаимодействие с клиентским приложением и распределение задач на множество серверов-обработчиков. Взаимодействие с серверами-обработчиками происходит через несколько очередей в RabbitMQ.

Сервера-обработчики принимают задачу на симуляцию, проверяют их на корректность, создают модель и выполняют симуляцию на одном из потоков, используя библиотеку.

Клиентский интерфейс представляет собой HTML5 приложение. Основным компонентом которого является canvas, на котором пользователь создаёт модель. В модель могут входить блоки: Генератор, Очередь, Разветвитель, Терминатор. Все блоки имеют свой набор параметров и название.

Генератор характеризуется частотой создания заявок. Очередь представляет собой совокупность собственно очереди и обслуживающего устройства и характеризуется частотой обслуживания, максимальным размером очереди и количеством каналов обслуживающего устройства. Разветвитель имеет два выходных канала и характеризуется вероятностью перенаправления заявки в первый канал. Терминатор служит для уничтожения заявок и сбора статистики.

Архитектура сервиса представлена на рисунке ниже.



Основным достоинством такой архитектуры является распределение задач по симуляции на несколько компьютеров и на несколько потоков на каждом компьютере.

Заключение

В настоящее время существует множество библиотек для современных прикладных языков программирования, предоставляющих широкие возможности для имитационного моделирования.

Список литературы

1. Чёрненький В.М. Понятие и свойства дискретного процесса функционирования системы // Наука и образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2011. № 11. Режим доступа: http://technomag.edu.ru/doc/292510.html (дата обращения 29.03.2015).
2. Чёрненький В.М. Теоретические основы построение имитационного процесса: учебное пособие по дисциплине «Описание параллельных процессов». Режим доступа: http://e-learning.bmstu.ru/portal_iu5/pluginfile.php/644/mod_page/content/7/index_2_2_2.htm (дата обращения: 29.03.2015).
3. Чёрненький В.М. Теоретические основы описания процессов функционирования дискретных систем. Учебное пособие по дисциплине «Описание параллельных процессов». Режим доступа: http://e-learning.bmstu.ru/portal_iu5/pluginfile.php/644/mod_page/content/7/index_oop.htm (дата обращения: 29.03.2015).
4. Документация по библиотеке simpy. Режим доступа: https://simpy.readthedocs.org/en/latest/ (дата обращения: 29.03.2015).
5. Документация к библиотеке simjs. Режим доступа: http://simjs.com (дата обращения: 29.03.2015).

6. Статья «Discrete_event_simulation». Википедия, свободная энциклопедия. Режим доступа: http://en.wikipedia.org/wiki/Discrete_event_simulation (дата обращения: 29.03.2015).