Г. Т. БАЛАКАЕВА, А. С. АКТЫМБАЕВА

РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА ДЛЯ МОДЕЛИРОВАНИЯ СИСТЕМЫ МАССОВОГО ОБСЛУЖИВАНИЯ

Описана имитационная модель системы массового обслуживания с одним устройством обслуживания на языке программирования JAVA, который, располагая развитым механизмом многопоточности, позволяет достаточно просто описывать разветвленные параллельные процессы с элементами синхронизации. Подробно изложен алгоритм событийно-ориентированной симуляции, приведены результаты моделирования системы с одним устройством обслуживания.

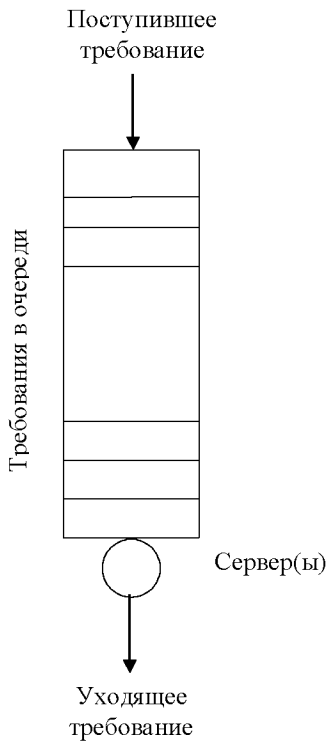
Во многих областях практической деятельности мы сталкиваемся с необходимостью пребывания в состоянии ожидания. Подобные ситуации возникают в очередях в билетных кассах, в крупных аэропортах, при ожидании обслуживающим персоналом самолетов разрешения на взлет или посадку, на телефонных станциях в ожидании освобождения линии абонента, в ремонтных цехах в ожидании ремонта станков и оборудования, на складах снабженческо-сбытовых организаций в ожидании разгрузки или погрузки транспортных средств, в банках при обслуживании очередей клиентов, в системах управления запасами. Во всех перечисленных случаях имеем дело с массовостью и обслуживанием. Изучением таких ситуаций занимается теория массового обслуживания.

Сегодня метод имитационного моделирования является одним из самых мощных и наиболее эффективных методов исследования процессов и систем самой различной природы и степени сложности. Сущность его состоит в написании компьютерной программы, имитирующей процесс функционирования системы, и проведении экспериментов на этой программе для получения статистических характеристик моделируемой системы. Используя результаты имитационного моделирования, можно описать поведение системы, оценить влияние различных параметров системы на ее характеристики, выявить преимущества и недостатки предлагаемых изменений, прогнозировать поведение системы.

Дискретно-событийное моделирование используется для построения модели, отражающей

развитие системы во времени, когда состояния переменных меняются мгновенно в конкретные моменты времени, т. е. система может изменяться только в исчислимое количество моментов времени. В такие моменты времени происходят события, при этом событие определяется как мгновенное возникновение, которое может изменить состояние системы [1].

В данной статье рассматривается моделирование системы массового обслуживания с одним устройством обслуживания (рис.).



Модель очереди

Когда требование поступает, а устройство свободно, обслуживание начинается немедленно. Время обслуживания T_1, T_2, ... следующих требований представлено независимыми одинаково распределенными случайными величинами, на которые не влияют интервалы времени обслуживания. Если при поступлении требования устройство занято, требование становится в очередь. По завершении обслуживания требования устройство выбирает требование из очереди (если такая имеется) по принципу FIFO.

Имитационная программа имеет три основных цели:

1) генерирование действующих дорожек для изучения системы;

2) собрание статистики о различных интересующих нас величин;

3) получение хорошей оценки получаемых желаемых измерений (величин).

Среди универсальных пакетов имитационного моделирования хорошо известны Arena, Extend, AweSim, GPSS/H, Micro Saint, MODSIM III, SES/workbench, SIMPLE ++, SIMUL8, SLX, Taylor Enterprise Dynamics.

В настоящее время возрастает потребность в интеллектуальных системах управления, отличающихся должной гибкостью работы, обеспечить которую без использования элементов моделирования сложно. Особенно эффективным является использование встроенных имитационных моделей. Среди особых требований к таким системам, кроме общих требований, предъявляемых к программному обеспечению систем управления, можно выделить следующие:

1) удобство описания моделируемых процессов;

2) кроссплатформенность;

3) высокое быстродействие.

Перечисленные требования могут быть вполне обеспечены применением платформы JAVA. Язык JAVA, располагая развитым механизмом многопоточности, позволяет достаточно просто описывать разветвленные параллельные процессы с элементами синхронизации. Кроме того, данному языку внутренне присуща кроссплатформенность за счет использования виртуальной JAVA-машины.

В качестве аппарата формализации в системе используются сети СМО.

Встроенные сетевые возможности JAVA позволяют обеспечить распределенность процесса моделирования в сети. Особенно удобным в этом плане является механизм сохранения объектов в поток. В частности, этот механизм может использоваться при моделировании с целью предсказания реакции системы на формируемый системой управления поток управляющих команд. В определенный момент времени (перед отсылкой управляющих команд) происходит сохранение модели в потоки; изменяются ключевые параметры модели, которые в различных вариантах передаются в распределенную вычислительную среду. По результатам моделирования выбирается наиболее подходящий вариант потока управления. Многократность итераций при принятии

решений обеспечивается механизмом откатов и воспроизведением сохраненной модели из потока.

Начнем с событийно-ориентированного подхода. В событийно-ориентированном моделировании процедура ассоциируется с каждым типом события в системе: оно представляет действия, необходимые для управления этим типом события, и вызывается каждый раз, как только случается событие этого типа. Предположим, что имеется M типов событий *1, 2, ... , M*, и пусть соответствующие процедуры будут обозначаться как $P_1, P_2, \ldots, P_M$. Симулятор содержит в себе то, что *вызывается в «списке событий». Пусть* $(T_1, T_2, \ldots, T_M)$ – время событий, где T_i – время, когда следующий тип события i собирается произойти. Приращение времени представляется нахождением наименьшего T_i и берется это время за величину текущего времени, после этого вызывается соответствующая процедура P_i.

Основные шаги алгоритма:

1) инициализация списка событий и переменных;

2) вызов процедуры *CLOCK*;

3) анализ полученной статистики и представление отчета.

Опишем алгоритмы событийно-ориентированной симуляции для нашей системы. В программе имеется четыре типа событий: поступление требования, завершение обслуживания, неисправность устройства и восстановление завершения. Мы назовем эти события соответственно *следующим образом: ARRIVAL, SERVICE,* BREAK, REPAIR. Имеется входящее требование *JOB* с единственным аттрибутом *ARRIVALTIME*, регистрирующая время. Когда требование поступает в систему, эта переменная будет использоваться для определения ответного времени для требования. Представим лист событий и очередь требований как связанный список записей.

Действия процедур времени и событий:

• *Процедура CLOCK*:

while TIME < PERIOD

{

○ установить *TIME* на время возникновения первого события в списке событий, то есть $T_i = \min(T_1, T_2, \ldots, T_M)$;

○ удалить это событие из списка;

○ вызвать соответствующую процедуру события P_i.

}

• Процедура *ARRIVAL:*

○ создать новое требование (*JOB*) с *ARRIVALTIME*, установить на *TIME* и перенести конец очереди требований;

○ сгенерировать интервал времени между поступлениями из соответствующих распределений и внести следующее событие в соответствующую позицию в списке событий;

○ если очередь обнаружена пустой и сервер в рабочем состоянии, то сгенерировать время обслуживания из соответствующего распределения и внести событие завершения обслуживания в соответствующую позицию в списке событий.

• Процедура *SERVICE:*

○ определить ответное время первого требования (*JOB*) в очереди и сохранить его;

○ увеличить число завершенных требований на один;

○ удалить первое требование из очереди;

○ если очередь не пуста, сгенерировать время обслуживания из соответствующего распределения и внести событие завершения обслуживания на соответствующую позицию в списке событий.

• Процедура *BREAK*:

○ если событие завершения обслуживания запланировано, то удалить его из списка событий;

○ сгенерировать восстановление интервала из соответствующего распределения и внести событие завершения восстановления в соответствующую позицию в списке событий;

○ установить флаг для указания нерабочего сервера.

• Процедура *REPAIR*:

○ сгенерировать действующий период из соответствующего распределения и внести аварийное событие в правильную позицию в списке событий;

○ если очередь не пуста, сгенерировать время обслуживания с соответствующего распределения и ввести событие завершения обслуживания в правильную позицию в списке событий;

○ восстановить (сбросить) флаг для указания рабочего сервера.

Результаты моделирования системы с одним уст ройст вом обслуж ивания. Вначале мы должны определить трафик интенсивности (иногда называют степень загрузки или интенсивность поступления требований). Для этого следует разделить среднее время между поступлениями λ на

среднее время обслуживания μ. Для устойчивой системы среднее время обслуживания всегда должно быть больше, чем среднее время между поступлениями. Поэтому ρ всегда должно быть меньше единицы. После долгого периода времени среднее время обслуживания всегда должно превышать время поступления:

$$\rho = \frac{\lambda}{\mu}. \qquad (1)$$

Среднее число требований в системе (N) вычисляется следующим уравнением:

$$N = \frac{\rho}{1-\rho}. \qquad (2)$$

Общее время ожидания вычисляется по следующей формуле (включая время обслуживания):

$$T = \frac{1}{\mu - \lambda}. \qquad (3)$$

Таким образом, проведено моделирование системы массового обслуживания как основного метода исследования сложных систем с разветвленным процессом функционирования. Разработана программа на языке программирования JAVA, получены результаты, которые дают возможность провести моделирование системы массового обслуживания типа M/M/1, что позволяет перейти к решению следующих задач.

ЛИТЕРАТУРА

1. *Лоу А.М., Кельтон В.Д.* Имитационное моделирование. СПб.: Питер, 2004. 847 с.

2. *Mitrani I.* Simulation techniques for discrete event systems. Cambridge university press, University of Newcastle upon Tyne, Computing Laboratory.

3. http://www.eventhelix.com/RealtimeMantra/CongestionControl/m_m_1_queue.htm

4. *Alexopoulos C., Seila A.* Output Data Analysis, in handbook of simulation. New York: John Wiley, 1998. 564 c.

Казахстанско-Британский технический университет, г. Алматы;

Казахский национальный университет им. аль-Фараби, г. Алматы

Поступила 20.10.06г.