УДК 004.738.5:519.682:519.872

Имитационное моделирование систем массового обслуживания в клиентских приложениях при использовании технологии «облачных» вычислений

© А.Ю. Быков, Е.В. Кожемякина, Ф.А. Панфилов

МГТУ им. Н.Э. Баумана, Москва, 105005, Россия

В технологии «облачных» вычислений обычно считается, что основные вычисления проводятся на стороне сервера, а клиент выполняет функции терминала. Однако иногда клиент обладает существенными вычислительными ресурсами, которые в этом случае простаивают. В статье рассмотрен подход к имитационному моделированию систем на клиентском компьютере через Web-браузер. Предложен интерпретатор для формального языка, похожего по синтаксису и семантике на GPSS. Интерпретатор реализован на языке Java в виде специального клиентского приложения — апплета, использующего библиотеку классов языка Java. Рассмотрен пример реализации имитационной модели типовой системы массового обслуживания. Приведен аналитический расчет модели. Представлены результаты имитационных экспериментов с моделью. Даны сравнительные временные оценки имитационных экспериментов на различных инструментальных средствах моделирования.

Ключевые слова: *имитационное моделирование, «облачные» вычисления, апплет, клиентское приложение, язык программирования Java.*

Введение. В настоящее время при разработке инструментальных программных средств имитационного моделирования наметилась тенденция перехода к «облачным» вычислениям [1], которые являются разновидностью распределенных вычислений и позволяют более эффективно использовать имеющиеся вычислительные ресурсы. В статье [2] представлено краткое описание библиотеки классов языка Java, которая позволяет создавать реализации имитационных моделей, способных выполняться как в специальных приложениях на языке Java для Web-сервера, называемых сервлетами, так и в клиентских приложениях на языке Java через Web-браузер, называемых апплетами, а также в обычных настольных приложениях. Эта библиотека классов основана на подходах, использованных в кроссплатформенной библиотеке функций языка Си++ [3, 4].

Обычно в «облачных» вычислениях большинство задач решается на стороне Web-сервера, так как вычислительные мощности клиента могут быть ограничены (например, клиентами могут быть различные мобильные устройства). Однако часто клиентами являются обычные

настольные персональные компьютеры (ПК) или ноутбуки, которые могут иметь вычислительные мощности, достаточные для решения некоторых задач. В этом случае некоторые задания могут выполняться на стороне клиента. Для более эффективного использования существующих вычислительных ресурсов необходимо решить задачу распределения заданий между клиентом и сервером в «облачных» вычислениях.

В данной статье рассмотрен подход, основанный на применении библиотеки классов языка Java к разработке реализаций имитационных моделей систем различного назначения, которые должны выполняться на стороне клиента, если клиентом является настольный ПК, ноутбук или, в некоторых случаях, планшетный компьютер. Также приведен пример реализации имитационной модели типовой системы массового обслуживания (СМО).

В [2] представлен пример реализации имитационной модели одноканальной СМО с неограниченной очередью, с пуассоновским потоком входящих заявок и пуассоновским потоком обслуживания. Реализация модели выполнена в виде апплета, встроенного в Web-страницу. При таком подходе к разработке моделей в сети Интернет требуется: разработать исходные коды для имитационной модели и апплета, определяющего интерфейс, на языке Java; скомпилировать исходные коды в байтовый код Java; далее с помощью тега языка HTML (HyperText Markup Language — язык разметки гипертекста) встроить исполняемые коды апплета (файлы с расширением class) на Web-страницу.

Такой подход неприемлем для обычного пользователя сети, работающего через Web-браузер, так как считается, что ему недоступен компилятор языка Java. Пользователь может работать в сети только с уже созданной реализацией модели и не может разрабатывать свои модели. В лучшем случае через интерфейс апплета он может менять различные исходные данные для модели.

Для обеспечения возможности построения реализаций моделей самим пользователем необходимо, чтобы клиентское приложение выполняло определенные функции некоторой среды разработки. В существующих инструментальных средах применяются два основных подхода к построению реализаций моделей: 1) построение графической схемы модели с помощью средств графического интерфейса пользователя, на основе которой затем создается исполняемый код; 2) запись реализации модели на формальном языке, далее трансляция кода в исполняемый код или интерпретация исходного кода.

Инструментальные среды, реализующие первый подход, являются, как правило, мощными интегрированными средами со средствами автоматизации разработки приложений. Они требуют значительных

вычислительных ресурсов и достаточно дороги. Непосредственная их реализация в виде клиентского приложения для Web-браузера является затруднительной.

Второй подход, основанный на записи кода модели на формальном языке, широко использовался в то время, когда не было операционных систем с графическим интерфейсом пользователя; он требует меньше вычислительных ресурсов. Примером такого подхода являются ранние среды на основе языка GPSS [5] без мощного графического интерфейса. При этом используются трансляторы или интерпретаторы соответствующих формальных языков. Подход, основанный на трансляции и (или) компиляции, достаточно сложно реализовать для Web-браузера. Остановимся подробнее на подходе, основанном на интерпретации.

Рассмотрим возможность реализации интерпретатора в виде клиентского приложения для Web-браузера, выполненного в виде Java-апплета, использующего библиотеку классов, представленную в [2]. Для удобства работы в качестве формального языка описания модели возьмем за основу синтаксис и семантику языка GPSS [5], который базируется на дискретно-событийном подходе, как и библиотека классов языка Java. Для выполнения основных функций имитационных моделей, базирующихся на дискретно-событийном подходе, реализован интерпретатор с минимальными функциями, базирующийся на синтаксисе языка GPSS с некоторыми ограничениями.

Описание разработанного клиентского приложения. Библиотека классов языка Java, описанная в [2], дополнена следующими *классами*: UniApplet, расширяющий стандартный класс javax.swing.JApplet (определяет интерфейс клиентского приложения); UniModel, расширяющий класс SimJava.Syst (определяет проведение экспериментов). Также добавлены вспомогательные классы: Block (для описания блока события модели); SaveValue (для задания сохраняемых величин); MyList и MyDeikstra (для перевода выражений внутри модели из инфиксную в постфиксную форму [6] по алгоритму Дейкстры при их обработке в ходе моделирования).

В модели на формальном языке, похожем на GPSS, могут быть использованы следующие *объекты*: каналы обслуживания; накопители (многоканальные устройства); очереди; гистограммы; сохраняемые величины. Также можно применять выражения со скобками, с вызовом заданных математических функций, вызовом функций для генерации случайных чисел. В выражениях можно использовать основные операции: арифметические, логические, операции сравнения.

Операторы в реализации модели могут быть трех типов: операторы-определения, например описание многоканальных устройств,

гистограмм и др.; операторы-блоки, задающие события модели, через которые проходят транзакты; управляющие операторы, или команды управления start, reset и др. Операторы имеют формат, похожий на формат операторов GPSS. Оператор может содержать метку, обязательное имя и в зависимости от типа оператора, возможно, параметры.

Апплет, как клиентское интернет-приложение, задан классом UniApplet. Апплет содержит следующие *элементы управления*: текстовая область для ввода исходного кода модели (объект класса TextArea); текстовая область для вывода блоков модели с номерами блоков после первичной обработки исходного кода и выявления ошибок (также объект класса TextArea); «кнопка» (объект класса JButton) для запуска выполнения экспериментов модели. Внешний вид апплета в браузере с заполненными текстовыми областями представлен на рис. 1.

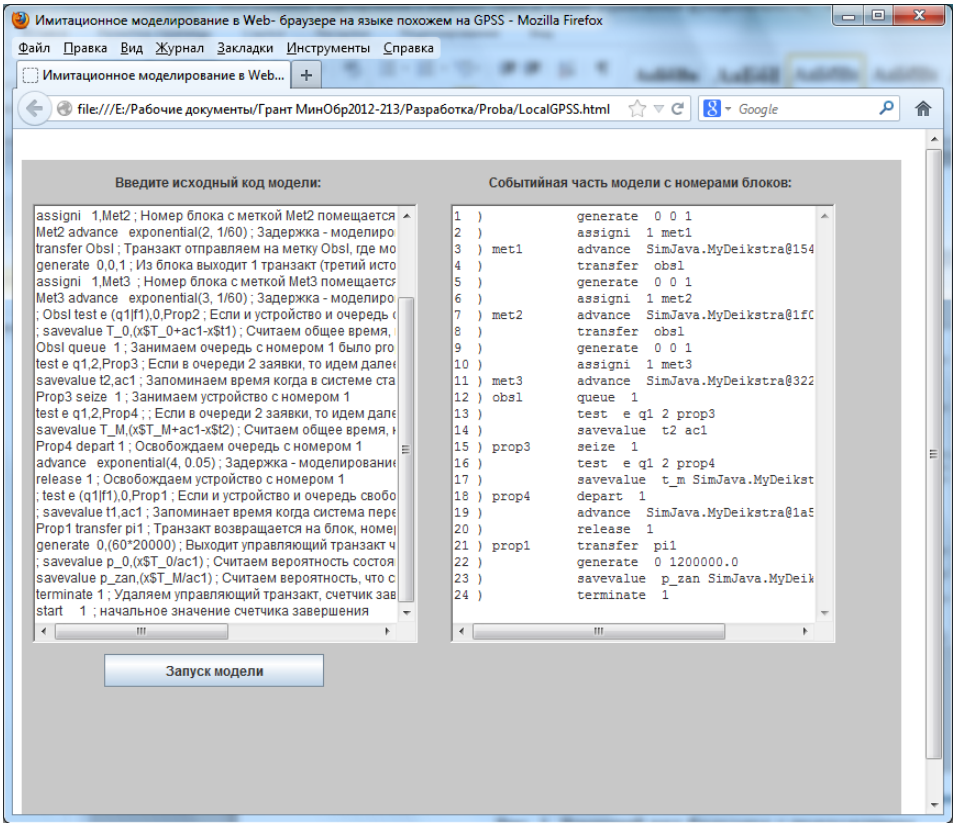

Рис. 1. Внешний вид браузера с приложением

Для выполнения имитационных экспериментов с моделью, заданной на языке, похожем на GPSS, используется отдельный класс языка Java — UniModel, расширяющий класс Syst [2]. Рассмотрим

основной алгоритм, реализуемый методами класса для проведения экспериментов.

Объект класса создается при нажатии кнопки «Запуск модели» внутри апплета. Вначале вызывается конструктор класса, который получает в качестве параметра массив строк, задающих модель на формальном языке. Конструктор получает ссылку на объект класса — контейнер для вывода результатов моделирования (класс, производный от класса java.awt.Container) и ссылку на объект класса TextArea для вывода блоков модели после их предварительной обработки. Конструктор инициализирует необходимые поля класса, вызывает метод start класса Syst для создания системной среды [2].

После создания системной среды вызывается метод createModelSreda для создания модельной среды. В этом методе в строках реализации модели выявляются все необходимые объекты системы моделирования: устройства обслуживания, накопители, очереди и т. д. Создаются они как объекты соответствующих классов библиотеки классов языка Java [2].

Далее в конструкторе вызывается метод createEvents, в котором создается событийная часть модели, при этом операторы-блоки заменяются на массив объектов класса Block. В этих объектах существуют поля для задания метки блока, названия блока и его параметров.

После создания конструктором объекта класса UniModel вызывается метод modeling, в котором запускается цикл моделирования событий на основе созданного массива ссылок на объекты класса Block. В цикле моделирования просматриваются ссылки на объекты класса Block; каждый объект задает выполнение одного события.

Для обработки каждого события вызывается метод uniBlock. Основу этого метода составляет оператор-переключатель, с его помощью для моделирования одного события вызывается метод соответствующего класса библиотеки классов языка Java [2].

Результаты моделирования выводятся в отдельное диалоговое окно, ссылка на контейнер которого (объект класса, производного от класса java.awt.Container) передается в качестве параметра конструктору класса UniModel.

Объект класса UniModel можно рассматривать как интерпретатор формального языка или некую «универсальную» модель, исходными данными для которой является код на формальном языке.

Приведем пример реализации модели типовой СМО и результаты, полученные при моделировании.

Реализация замкнутой системы массового обслуживания с одним каналом и *m* источниками заявок. Рассмотрим пример СМО, являющейся замкнутой, т. е. параметры входящего потока за-

явок зависят от состояния системы. Администратор обслуживает компьютерный зал, в котором находится три компьютера ($m = 3$). На работающем компьютере в среднем один раз в час происходит сбой (время между сбоями распределено по экспоненциальному закону, $\lambda = 1/60$ сбоев в минуту). Последствия сбоя администратор устраняет в среднем за 20 мин (время распределено по экспоненциальному закону, $\mu = 1/20$). Если сбой произошел, когда администратор устраняет последствия сбоя на другом компьютере, то компьютер становится в очередь на обслуживание.

Требуется провести имитационное моделирование СМО в течение 20 000 ч и определить параметры СМО: вероятность того, что все компьютеры работоспособны; вероятность того, что все компьютеры неработоспособны; загрузку канала обслуживания (администратора); параметры очереди (среднюю длину очереди и среднее время ожидания в очереди).

Данная задача имеет аналитическое решение [7]. Основные соотношения имеют следующий вид:

• вероятность того, что все компьютеры работоспособны (в системе нет ни одной заявки):

$$p_0 = \frac{1}{\sum\limits_{k=0}^{m} \frac{\rho^k m!}{(m-k)!}};$$

• вероятность того, что i компьютеров неработоспособны:

$$p_i = p_0 \rho^i \frac{m!}{(m-i)!}, \ i = 1, \ 2, \ ..., \ m;$$

• коэффициент загрузки канала (администратора)

$$K_{\text{загр}} = 1 - p_0;$$

• среднее число заявок (компьютеров) в очереди

$$\bar{Q}_{\text{очер}} = m - \frac{1 - p_0}{\rho} - (1 - p_0);$$

• среднее время нахождения заявки (компьютера) в очереди

$$\bar{T}_{\text{очер}} = \frac{\bar{Q}_{\text{очер}}}{(1 - p_0)\mu},$$

где $\rho = \lambda/\mu$ — приведенная интенсивность.

Подставляя заданные значения, получаем результаты:

$$p_0 \approx 0,346;\ \ p_m \approx 0,077;\ \ K_{\text{загр}} \approx 0,654;$$
$$\overline{Q}_{\text{очер}} \approx 0,385;\ \ \overline{T}_{\text{очер}} \approx 11,7647 \text{ мин.}$$

Исходный код модели на формальном языке, похожем на GPSS, с комментариями представлен в листинге 1; комментарии в каждой строчке следуют после символа «;».

Листинг 1

```
; Замкнутая СМО с m источниками заявок
generate 0,0,1; Из блока выходит 1 транзакт (первый источник)
assigni 1,Met1; Номер блока с меткой Met1 помещается в первый целочисленный параметр транзакта
Met1 advance exponential(1, 1/60); Задержка - моделирование работы без сбоев
transfer Obsl; Транзакт отправляем на метку Obsl, где моделируется сбой
generate 0,0,1; Из блока выходит 1 транзакт (второй источник)
assigni 1,Met2; Номер блока с меткой Met2 помещается в параметр транзакта
Met2 advance exponential(2, 1/60); Задержка - моделирование работы без сбоев
transfer Obsl; Транзакт отправляем на метку Obsl, где моделируется сбой
generate 0,0,1; Из блока выходит 1 транзакт (третий источник)
assigni 1,Met3; Номер блока с меткой Met3 помещается в параметр транзакта
Met3 advance exponential(3, 1/60); Задержка - моделирование работы без сбоев
Obsl queue 1; Занимаем очередь с номером 1
test e q1,2,Prop3; Если в очереди 2 заявки, то идем далее, иначе - на метку Prop3
savevalue t2,ac1; Запоминаем время, когда в системе стало максимальное число заявок
Prop3 seize 1; Занимаем устройство с номером 1
test e q1,2,Prop4; Если в очереди 2 заявки, то идем далее, иначе - на метку Prop4
savevalue T_M,(x$T_M+ac1-x$t2); Считаем общее время, когда в системе было максимальное число заявок
Prop4 depart 1; Освобождаем очередь с номером 1
advance exponential(4, 0.05); Задержка - моделирование устранения сбоя
release 1; Освобождаем устройство с номером 1
Prop1 transfer pi1; Транзакт возвращается на блок, номер которого в первом целочисленном параметре
```

generate 0,(60*20000); Выходит управляющий транзакт через 20 тыс. часов
savevalue p_zan,(x$T_M/ac1); Считаем вероятность, что система занята
terminate 1; Удаляем управляющий транзакт, счетчик завершения уменьшается на 1
start 1; Начальное значение счетчика завершения

Результаты моделирования представлены на рис. 2.

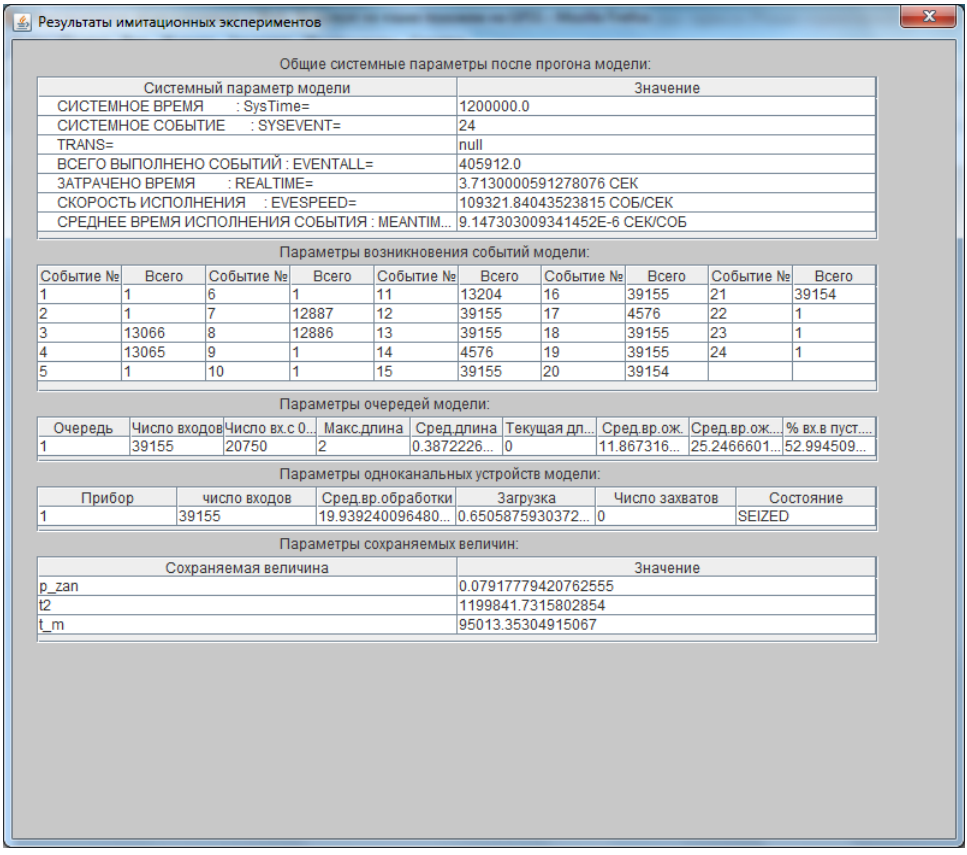

Рис. 2. Результаты моделирования

Как видно из рис. 2, полученные результаты с определенной погрешностью соответствуют расчетам на аналитической модели.

Сравнение с другими реализациями модели. С учетом того, что проведение имитационных экспериментов, как правило, требует много вычислительных ресурсов, представляет интерес сравнить временну́ю трудоемкость различных подходов при моделировании. Сравним длительность проведения имитационных экспериментов при моделировании одной и той же СМО, рассмотренной ранее, при тех же исходных данных и использовании следующих подходов (инструментальных средств моделирования):

• применение рассмотренной «универсальной» модели внутри апплета через браузер Firefox;

• применение рассмотренной «универсальной» модели внутри настольного Java-приложения с графическим интерфейсом пользователя (основу интерфейса вместо апплета составляет класс java.swing. JFrame);

• программирование реализации модели с прямым использованием библиотеки классов языка Java по аналогии с [2];

• использование языка GPSS (средство GPSS World).

Эксперименты проводились на одном и том же ноутбуке с процессором Intel®Core™ i5-2410 с тактовой частотой 2,3 ГГц под управлением операционной системы Windows 7 «Профессиональная». Примерная длительность проведения экспериментов с округлением до секунды представлена в таблице.

Временные затраты на моделирование при использовании различных инструментальных средств

Инструментальные средства моделирования	Время выполнения, с
«Универсальная» модель (интерпретатор) в апплете или настольном приложении	4
Java и библиотека классов в апплете или настольном приложении	1
GPSS World	1

Заключение. Разработанное клиентское приложение позволяет проводить моделирование систем, не привлекая вычислительные ресурсы Web-сервера. Однако данный подход уступает по производительности подходам с использованием языка Java и библиотеки классов или языка GPSS, поэтому он может применяться только для проведения экспериментов с моделями, не требующих больших вычислительных ресурсов и при наличии у клиентского компьютера современного микропроцессора.

Исследование выполнено при поддержке Министерства образования и науки Российской Федерации (соглашение № 14.B37.21.0401).

ЛИТЕРАТУРА

[1] Власов С.А., Девятков В.В., Кобелев Н.Б. Имитационные исследования: от классических технологий до облачных вычислений. *Сб. докл. Пятой (юбилейной) всерос. научно-практ. конф. по имитационному моделированию и его применению в науке и промышленности «Имитационное моделирование. Теория и практика» ИММОД-2011.* Санкт-Петербург, 2011, т. 1, с. 42–50.

[2] Быков А.Ю., Панфилов Ф.А., Сумарокова О.О. Имитационное моделирование с применением библиотеки классов языка Java, разработанной для «облачных» сервисов. *Инженерный журнал: наука и инновации,* 2013, № 2. URL: http://engjournal.ru/catalog/it/hidden/535.html (дата обращения 30.06.2013).

[3] Медведев Н.В., Быков А.Ю., Гришин Г.А. Имитационное моделирование систем массового обслуживания с использованием межплатформенной библиотеки функций языка Си++. *Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение,* 2005, № 4, с. 85–93.

[4] Журавлев А.М., Медведев Н.В., Быков А.Ю. Использование межплатформенной библиотеки функций языка Си++ для реализации имитационных моделей типовых систем массового обслуживания. *Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение,* 2008, № 4, с. 3–15.

[5] Шрайбер Т.Дж. *Моделирование на GPSS.* Москва, Машиностроение, 1980, 592 с.

[6] Карпов Ю.Г. *Теория и технология программирования. Основы построения трансляторов*. Санкт-Петербург, БХВ-Петербург, 2005, 272 с.

[7] Клейнрок Л. *Теория массового обслуживания*. Москва, Машиностроение, 1979, 432 с.

Статья поступила в редакцию 28.06.2013

Ссылку на эту статью просим оформлять следующим образом:

Быков А.Ю., Кожемякина Е.В., Панфилов Ф.А. Имитационное моделирование систем массового обслуживания в клиентских приложениях при использовании технологии «облачных» вычислений. *Инженерный журнал: наука и инновации,* 2013, вып. 11. URL: http://engjournal.ru/catalog/it/network/1001.html

Быков Александр Юрьевич родился в 1969 г., окончил ВИКИ им. А.Ф. Можайского в 1991 г. Канд. техн. наук, доцент кафедры «Информационная безопасность» МГТУ им. Н.Э. Баумана. Автор около 25 работ в области информационной безопасности и исследования систем обработки информации и управления. e-mail: abykov@bmstu.ru

Кожемякина Евгения Вадимовна родилась в 1993 г. Студентка 3-го курса кафедры «Информационная безопасность» МГТУ им. Н.Э. Баумана.

Панфилов Филипп Александрович родился в 1989 г. Ассистент кафедры «Информационная безопасность» МГТУ им. Н.Э. Баумана.