Методика и инструментарий проведения нагрузочного тестирования

А. Ю. Долгоруков[1], Д. С. Порай[2]

Одной из задач, возникающей в процессе создания корпоративных информационных систем, является проверка работоспособности комплекса при одновременной работе заявленного количества пользователей. В данной статье описываются методика и инструментарий, позволяющие решить эту задачу для широкого спектра информационных систем.

1. Введение

При разработке больших программных комплексов одной из важных задач является нагрузочное тестирование в условиях промышленной эксплуатации. Обычно условия промышленной эксплуатации включают в себя следующие параметры:

1) объем базы данных,
2) количество одновременно работающих пользователей,
3) характер операций, выполняемых пользователями,
4) количество и частота выполнения операций.

В случае однопользовательских систем или систем с двумя-тремя пользователями нагрузочное тестирование можно организовать силами нескольких тестеров. Для систем, с которыми должны работать 50–100 пользователей, провести тестирование человеческими силами оказывается невозможно по нескольким причинам:

1. Собрать одновременно большое количество тестеров на длительное время невозможно как организационно, так и финансово.
2. Обеспечить круглосуточную работу большого количества тестеров также затруднительно.
3) Обеспечить повторяемость эксперимента практически невозможно.

[1] 117312, г. Москва, пр. 60-летия Октября, д. 9, ООО «Когнитивные технологии», yuron@cognitive.ru.
[2] 117312, г. Москва, пр. 60-летия Октября, д. 9, ИСА РАН, dmip@cs.isa.ru.

Поэтому для таких задач крайне важными являются средства автоматического проведения нагрузочного тестирования.

На практике средства автоматического нагрузочного тестирования оказываются тесно связанными с тестируемой системой и их трудно использовать повторно в других проектах (в этом смысле в области тестирования web-приложений ситуация несколько лучше, чем в области классических приложений).

В рамках данной работы были создана методика проведения нагрузочного тестирования и инструментарий, облегчающий процесс создания нагрузочных тестов. Далее будут описаны основные принципы этой методики и инструментарий. Эти принципы были проверены на практике в нескольких проектах. Один из проектов — подсистема Workflow продукта Евфрат-Документооборот (далее Евфрат-ДО) — будет использован в качестве примера [4].

Описываемая методика является обобщением технологии, разработанной консорциумом Transaction Processing Performance Council (TPC) для тестирования производительности систем «сервер + реляционная СУБД» [2]. Разработанный инструментарий использует библиотеку CppUnit, исходно разработанную для создания тестов уровня модуля (unit test) [1].

2. Методика проведения нагрузочного тестирования

2.1. Постановка задачи

Рассмотрим схему, иллюстрирующую общие принципы тестирования производительности или, иначе говоря, определения приемлемости показателей эффективности информационной системы (см. рис. 1).

Объектом тестирования является информационная система (далее ИС).

Задачей тестирования является получение показателей эффективности (производительности) ИС и определение приемлемости этих показателей для предполагаемых режимов эксплуатации ИС.

Исходными данными для тестирования является предполагаемый режим эксплуатации ИС, который определяет:

- Бизнес процессы организации, реализуемые с помощью ИС, определяющие какие цепочки действий (операций, транзакций) и какими пользователями (ролями) выполняются.
- Временны́е характеристики выполнения бизнес процессов и отдельных операций, определяющие:
  - частоту (интенсивность) и/или вероятностный закон распределения по времени выполнения бизнес процессов и/или отдельных операций группами пользователей и/или отдельными пользователями,

– минимальное время необходимое пользователю для ввода параметров (пауза перед) каждой операции и среднее время размышления или оценки результатов (пауза после) выполнения каждой операции,
– комфортное 90 % и/или максимально допустимое время ожидания ответа (или результата выполнения) каждой операции.

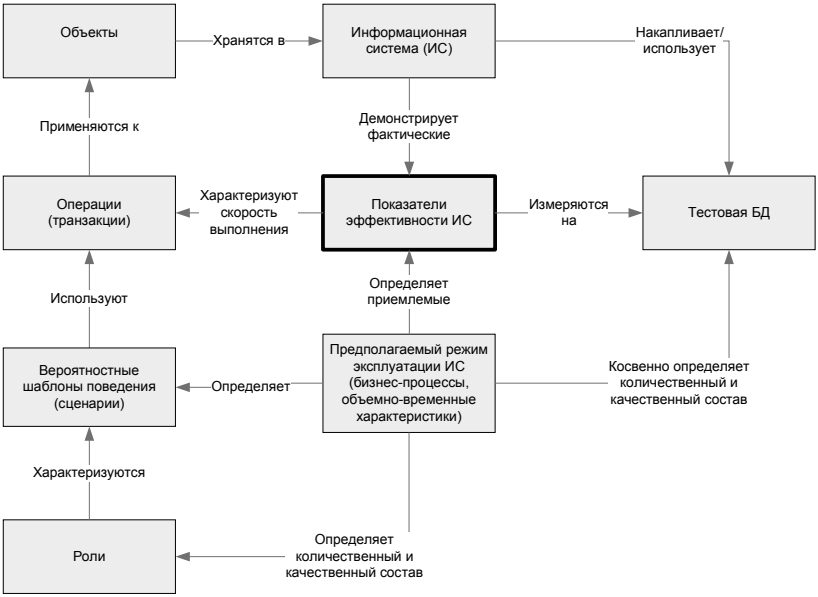

Рис. 1. Диаграмма определения приемлемости показателей эффективности информационной системы

- Предполагаемое количество пользователей ИС (и, частично, организационную структуру).
- Расчетный срок эксплуатации ИС (или период «очистки» БД ИС, связанный с регулярным «сбрасыванием» данных в архив).

В частности, на основании этих данных можно определить (подсчитать) необходимый качественный и количественный состав тестовой БД.

Требования к интенсивности выполнения операций и бизнес процессов не должны противоречить требованиям к времени размышления пользователей, т. е. быть выполнимы при теоретически минимальных нулевых временах ожидания ответа операций ИС (по сути, это означает, что сотрудники организации должны быть теоретически в состоянии выполнить задачи своей организации при идеально работающей ИС).

Временные характеристики, расчетное количество пользователей и срок эксплуатации ИС принято также объединять под общим названием «объемно-временные характеристики».

В результате тестирования должны быть получены ответы на все или некоторые из приведенных ниже вопросов:

- При фиксированном режиме эксплуатации, каковы фактическое минимальное, 90 % и максимальное время ожидания ответа (или результата выполнения) каждой операции? Удовлетворяют ли эти показатели представленным требованиям?
- Максимально допустимые режимы эксплуатации, при которых 90 % и максимальное время ожидания ответа (или результата) выполнения каждой операции удовлетворяет представленным требованиям, такие как:
  - максимальное количество пользователей при фиксированной интенсивности выполнения бизнес процессов и сроке эксплуатации ИС;
  - максимальная интенсивность выполнения бизнес процессов при фиксированном количестве пользователей и сроке эксплуатации ИС;
  - максимальный срок эксплуатации ИС при фиксированном количестве пользователей и интенсивности выполнения бизнес процессов.

2.2. Моделирование режима эксплуатации ИС

Для оценки фактических показателей эффективности ИС применяется моделирование (эмуляция) предполагаемого режима эксплуатации.

Моделирование проводится с помощью автоматических сценариев, эмулирующих работу пользователей ИС на тестовой БД в течение некоторого оценочного временно́го интервала.

Количественный и качественный состав тестовой БД, а также величина оценочного временно́го интервала определятся в соответствии с объемно-временными характеристиками ИС.

Эмуляция осуществляется по следующим принципам (см. рис. 2):

1. Главной составной частью процесса эмуляции является сценарий (последовательность технологических действий по обработке информации).
2. Сценарий соответствует одной роли пользователя.
3. Сценарий состоит из нескольких шагов. Каждый шаг соответствует определенной технологической операции: поиск документов, получение списка поручений, открытие документа и т. д.
4. Выполнение сценария заключается в последовательном выполнении его шагов. При этом эмулируется следующее поведение пользователя:
   4.1. Пользователь инициирует выполнение технологической операции.
   4.2. Пользователь изучает полученные результаты.

4.3. На основании полученной информации пользователь инициирует выполнение следующей операции.

5. Изучение пользователем результатов выполнения операции реализуется задержкой на случайную величину в определенном диапазоне.

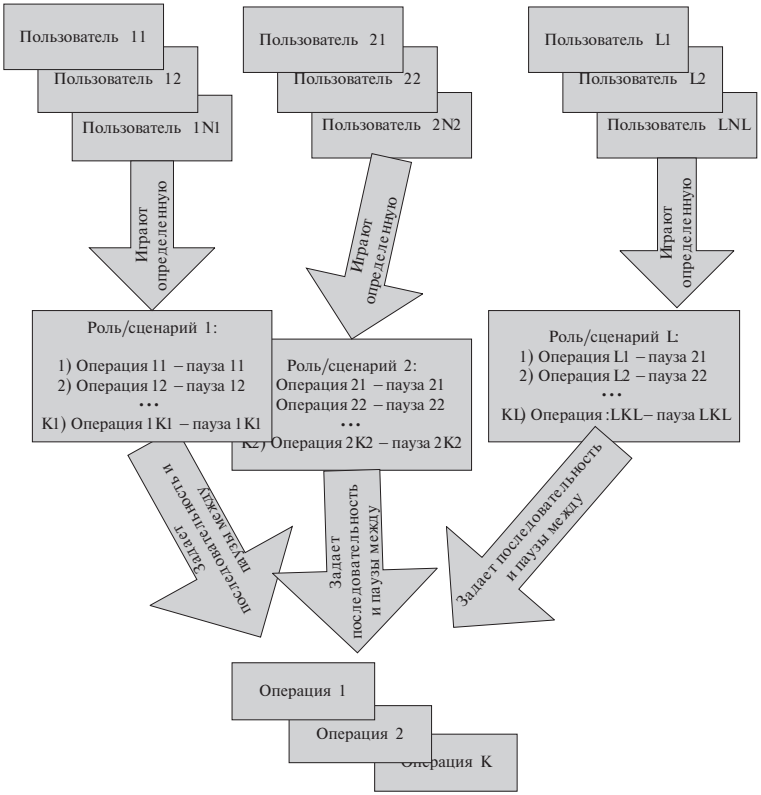

Рис. 2. Моделирование режима эксплуатации ИС

Для выполнения сценариев была разработана программа-эмулятор, которая позволяет выполнить указанный ей набор сценариев с описанным режимом эксплуатации. Программа-эмулятор будет подробно описана далее.

2.3. Использование методики

Чтобы организовать проведение нагрузочного тестирования в новом проекте, необходимо выполнить следующие шаги:

1. Выполнить подготовительные операции:
   1.1. Выделить роли пользователей, работающих с системой.
   1.2. Выделить операции, которые выполняет каждая из ролей.
   1.3. Определить временные характеристики каждой операции.
   1.4. Зафиксировать предполагаемое количество пользователей.
   1.5. Определить объем тестовой базы данных.
2. Выполнить необходимое программирование:
   2.1. Сформировать библиотеку (DLL) в виде расширяемого модуля (plug-in) библиотеки CppUnit.
   2.2. Запрограммировать каждый сценарий, соответствующий роли, в виде набора тестов CppUnit (test suite). При этом каждому шагу сценария должен соответствовать тест CppUnit (test).
3. Сформировать конфигурационный .INI файл с описанием временных характеристик каждой операции.
4. Выполнить накачку тестовой БД необходимого объема.
5. Выполнить тестовые запуски полученного модуля с помощью программы-эмулятора.

3. Инструментарий проведения нагрузочного тестирования

3.1. Общее описание программы-эмулятора UILoadTest

Программа предназначена для создания нагрузки на сервер без необходимости привлечения большого количества единиц средств вычислительной техники и большого количества пользователей (см. рис. 3). Данная задача решается эмуляцией работы нескольких пользователей с одной рабочей станции в автоматическом режиме.

3.2. Входные данные для программы-эмулятора

Входные данные программы-эмулятора описываются конфигурационным .INI файлом (см. табл. 1). В этом файле название сценария совпадает с названием класса CppUnit, реализующего соответствующий «test suite», а название шага сценария совпадает с названием функции CppUnit, реализующей соответствующий тест.

Наряду с параметрами, указанными в таблице, файл может содержать дополнительные секции, которые используются сценариями.

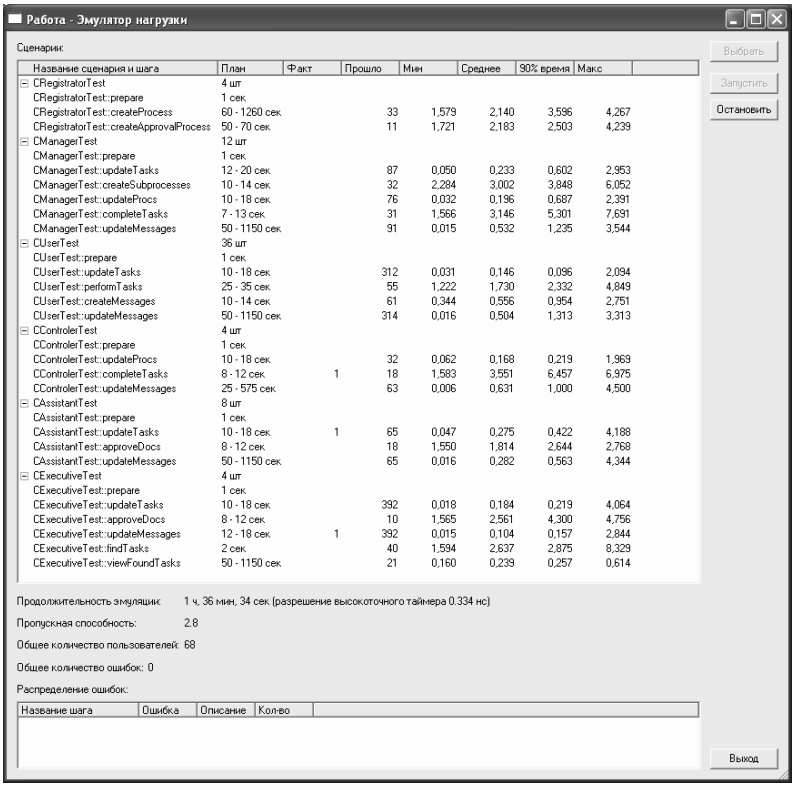

Рис. 3. Главное окно программы UILoadTest

Таблица 1

Конфигурационный .INI файл

Секция	Ключ	Описание
Options	LibraryName	Название динамической библиотеки (DLL-файла) с расширяемым модулем CppUnit.
Options	UsersCount	Количество сценариев, запускаемых на данной рабочей станции. Если не указано, то запускаются все сценарии.
Options	StartWaitTime	Минимальное время (в секундах) между двумя последовательными запусками сценариев. Если не указано, то одна секунда.

Окончание таблицы 1

Секция	Ключ	Описание
Название сценария	UsersCount	Количество одновременно работающих сценариев данного вида.
Название шага сценария	MinWaitTime	Минимальное время ожидания после успешного выполнения шага сценария
Название шага сценария	MaxWaitTime	Максимальное время ожидания после успешного выполнения шага сценария
Название шага сценария	IncludeInTPM	Включать ли данный шаг сценария в вычисление «пропускной способности» ИС

Пример конфигурационной файла (фрагмент) для нагрузочного тестирования системы Евфрат-ДО:

```
[Options]
LibraryName=WorkflowTest.dll
[Connection]
ServerName=192.168.0.5:17170
AdminLogin=sysadmin
FilialsCnt=4
[CRegistratorTest]
UsersCount=4
[CRegistratorTest::createProcess]
IncludeInTPM=1
MinWaitTime=60
MaxWaitTime=1260
[CRegistratorTest::createApprovalProcess]
IncludeInTPM=1
MinWaitTime=50
MaxWaitTime=70
[CManagerTest]
UsersCount=12
[CManagerTest::updateTasks]
MinWaitTime=12
MaxWaitTime=20
[CManagerTest::createSubprocesses]
IncludeInTPM=1
MinWaitTime=10
MaxWaitTime=14
```

В данном примере секция [`Connection`] описывает параметры подключения к серверу, которые используются сценариями из библиотеки WorkflowTest.dll.

3.3. Фазы работы программы

В целом процесс эмуляции можно разбить на три фазы:

1. Запуск сценариев. Программа осуществляет последовательный запуск необходимого количества сценариев.
2. Собственно эмуляция.
3. Остановка сценариев. Программа осуществляет остановку сценариев после выполнения ими очередного шага, т. е. процесс остановки занимает некоторое время.

Запуск сценариев выполняется так, что каждый сценарий выполняется в отдельном процессе (не потоке). Это позволяет исключить влияние сценариев друг на друга, а также эффекты кэширования.

Первым шагом всех сценариев является операция «подготовка» (prepare). Она выполняет подключение к базе данных. Для избежания конфликтов на этой стадии (например, когда требуется, чтобы каждый пользователь может войти в систему с одного рабочего места) запуск сценариев производится строго последовательно — сначала запускается первый сценарий, затем программа ждет, когда он подключится к базе данных, и лишь после этого запускается следующий сценарий.

3.4. Выходные информационные отчеты

По окончании работы программа-эмулятор формирует несколько файлов с отчетами:

1. Results.csv — результаты выполнения всех шагов сценариев с указанием названия шага, момента времени, в которое был выполнен шаг, и продолжительности выполнения шага.
2. Failures.csv — список ошибок, произошедших в процессе эмуляции с указанием сценария, в котором произошла ошибка, момента времени, в который был выполнен шаг сценария, и текста ошибки.
3. Report.html — сводный отчет о выполненной эмуляции.

Сводный отчет содержит следующие характеристики:

1. Название компьютера, на котором выполнялась эмуляция.
2. Время начала эмуляции. Началом эмуляции считается момент времени, когда количество одновременно работающих пользователей достигло необходимой величины.
3. Время окончания эмуляции. Временем окончания эмуляции считается момент, когда пользователь нажал кнопку «Остановить» в диалоге программы.

4. Продолжительность эмуляции. Интервал времени между «Временем начала эмуляции» и «Временем окончания эмуляции».
5. Пропускная способность системы. Среднее количество учетных операций, выполненных системой за одну минуту. Какие именно операции являются учетными, определяется .INI файлом. В случае системы Евфрат-ДО учетной операцией являлись все операции, вызывающие изменения БД, например, регистрации нового документа или заявки о готовности поручения.
6. Количество одновременно работающих пользователей.

Также сводный отчет содержит показатели по всем шагам сценариев:

1. Количество успешно выполненных шагов.
2. Минимальное время выполнения шагов.
3. Среднее время выполнения шагов. Подсчитывается как суммарное время успешно пройденных шагов, деленное на количество успешно пройденных шагов.
4. 90 %-е время выполнения шагов. Подсчитывается как максимальное время выполнения шагов после отбрасывания 10 % (от количества шагов) самых худших результатов.
5. Максимальное время выполнения шагов.

4. Пример. Документооборот малого предприятия

4.1. Типовые бизнес процессы

Были выделены следующие типовые бизнес процессы, обеспечиваемые подсистемой Workflow в составе Евфрат-ДО:

A. Исполнение документа по наложенным резолюциям.
B. Выборочный контроль исполнения.
C. Согласование и визирование документа.
D. Обмен пользовательскими сообщениями (рассылка документов).

Ниже приводятся описания этих бизнес процессов (сценариев).

4.4.1. Исполнение документа по наложенным резолюциям

Инициируется регистратором.

A1. На основании резолюций руководства, проставленных на документе, регистратор документа создает одно или несколько параллельных или последовательных поручений соответствующим исполнителям, возможно, с указанием срока исполнения. Контроль над исполнением

поручений передается контролеру организации, который также контролирует срок исполнения всего документа.

A2. Исполнитель получает назначенное ему поручение и принимает его к исполнению. Ответственный исполнитель по данному ему поручению может создать одно или несколько подчиненных поручений другим исполнителям (контролером подчиненных поручений становится сам ответственный исполнитель).

A3. При приближении и/или наступлении срока исполнения поручения исполнитель получает соответствующие напоминания.

A4. По окончанию работы, исполнитель заявляет о готовности поручения и, возможно, отправляет отчет об исполнении контролеру.

A5. Ответственный исполнитель получает отчеты исполнителей об исполнении созданных им подчиненных поручений и снимает их с контроля. Если сняты с контроля все подчиненные поручения основного поручения, то ответственный исполнитель заявляет контролеру о готовности основного поручения.

A6. Контролер получает отчеты исполнителей и, либо снимает поручения с контроля, либо возвращает их на доработку обратно исполнителям.

A7. При приближении и/или наступлении срока исполнения всего документа контролер получает соответствующие напоминания.

A8. В случае необходимости (распоряжения руководства, производственной необходимости или затруднений с исполнением документа), контролер может изменить исполнителя, срок исполнения поручения, отменить или добавить новые поручения по документу. Также возможна отмена исполнения всего документа.

A9. Когда контролер получает уведомление, что все поручения по документу исполнены, он снимает весь документ с контроля.

4.4.2. Выборочный контроль исполнения

Инициируется руководителем.

B1. Руководитель выполняет поиск поручений или контрольных документов (процессов) по исполнителям, срокам исполнения, состоянию готовности и прочим атрибутам. Или строит один из стандартных отчетов по исполнительской дисциплине за заданный интервал времени.

B2. Руководитель выборочно просматривает контрольные карточки (КК) найденных документов, состоящие из всех поручений по данному документу с информацией о ходе исполнения этих поручений.

4.4.3. Согласование и визирование документа

Инициируется регистратором.

C1. В зависимости от вида документа, регистратор документа создает одно или несколько параллельных или последовательных поручений-согласований соответствующим заместителям руководителя, а также завершающее поручение-визирование руководителю. (Также возможен выбор маршрута согласования/визирования по имеющемуся шаблону, созданному в Дизайнере маршрутов.)

C2. Заместитель руководителя получает отправленное ему поручение-согласование и отмечает свое положительное или отрицательное мнение по данному документу.

C3. В зависимости от выбранного порядка согласования документа, руководитель получает поручение-визирование после всех положительно (или, необязательно положительных) завершенных согласованиях. И принимает положительное или отрицательное решение по данному документу.

4.4.4. Обмен пользовательскими сообщениями (рассылка документов)

Инициируется каждым пользователем.

D1. Отправитель формирует сообщение (возможно, с одним или несколькими документами) и отправляет его одному или нескольким получателям.

D2. Получатель получает и просматривает сообщение.

D3. Отправитель получает уведомление о доставке и/или прочтении сообщения (если он запрашивал эти уведомления при отправке сообщения)

4.2. Требуемые или расчетные объемно-временные характеристики

Евфрат-ДО предназначен для работы в средних организациях с количеством сотрудников 10–20–50–100–200 человек.

Расчетный срок эксплуатации 1–3–5 лет (далее должно следовать «сбрасывание» данных в архивную БД).

Интенсивность работы пользователей (в плане нагрузки на подсистему Workflow) можно оценить в количестве созданных и/или исполненных одним пользователем поручений или согласований: от 5 до 50 в день (от одного в 1,5 часа до одного в 10 минут) (см. табл. 2).

Таблица 2

Предположительное относительное распределение частоты выполнения бизнес процессов и их среднестатистические параметры.

Бизнес процесс	Предположительная относительная частота выполнения	Предположительные среднестатистические параметры
A. Исполнение документа по наложенным резолюциям	10	1–5 поручений по одному документу; 1–2 дня — срок исполнения поручений;
B. Выборочный контроль исполнения	1–2	Поиск просроченных поручений. Просмотр КК документов у трех из них.
C. Согласование и визирование документа	2–5	2–4 согласования и/или визирования на один документ
D. Обмен пользовательскими сообщениями	1–2 на каждое поручение	1–2 получателя одного сообщения

4.3. Результаты тестирования

Далее приведен пример отчета.

Конфигурация

Параметр	Значение
Название компьютера	YURONTEST
Конфигурационный файл	D:\Testing\WFIndepTest_f7_x2.ini
Библиотека со сценариями	WorkflowTest.dll

Результаты в целом

Параметр	Значение
Начало эмуляции	2006-05-24 13:30:24
Окончание эмуляции	2006-05-24 16:03:26
Продолжительность эмуляции	2 ч, 33 мин, 2 сек (разрешение высокоточного таймера 0.334 нс)
Пропускная способность	12.1
Количество пользователей	119
Количество ошибок	0

Результаты по шагам сценариев

Название сценария и шага	Параметры	Выполнено	Минимальное время	Среднее время	90 % время	Максимальное время
CRegistratorTest	7 шт					
::createProcess	30–630 сек	182	0.408	0.753	0.969	4.799
::createApprovalProcess	50–70 сек	60	0.659	1.121	1.597	5.27
CManagerTest	21 шт					
::updateTasks	12–20 сек	540	0.04	0.183	0.377	4.987
::createSubprocesses	10–14 сек	188	0.848	1.226	1.568	5.254
::updateProcs	10–18 сек	379	0.042	0.176	0.406	3.501
::completeTasks	7–13 сек	235	0.613	1.524	2.208	8.91
::updateMessages	30–570 сек	595	0.022	0.181	0.36	6.015
CUserTest	63 шт					
::updateTasks	10–18 сек	1777	0.016	0.174	0.374	6.25
::performTasks	25–35 сек	370	0.504	0.843	1.115	7.536
::createMessages	10–14 сек	407	0.016	0.127	0.203	3.922
::updateMessages	30–570 сек	1776	0.009	0.184	0.36	4.58
CControlerTest	7 шт					
::updateProcs	10–18 сек	56	0.062	0.17	0.312	0.766
::completeTasks	8–12 сек	244	0.534	1.445	2.083	8.958
::updateMessages	15–285 сек	400	0.025	0.202	0.469	3.807
CAssistantTest	14 шт					
::updateTasks	10–18 сек	408	0.029	0.186	0.375	4.813
::approveDocs	8–12 сек	116	0.55	0.957	1.362	6.565
::updateMessages	30–570 сек	408	0.015	0.13	0.241	4.188
CExecutiveTest	7 шт					
::updateTasksEarly	1 сек	0	0	0	0	0
::updateTasks	10–18 сек	201	0.03	0.159	0.36	3.859
::approveDocs	8–12 сек	57	0.628	0.854	1.003	5.037
::updateMessages	30–570 сек	198	0.016	0.112	0.172	3.754
::findTasks	2 сек	17	0.955	4.067	19.485	26.33
::viewFoundTasks	12–18 сек	92	0.222	0.297	0.332	1.71

5. Другие применения нагрузочных тестов

Созданные для проведения нагрузочного тестирования сценарии реализуют возможности автоматического выполнения функциональности ИС. И с их помощью можно решать не только задачи собственно нагрузочного тестирования, но и некоторые смежные задачи (см. табл. 3):

- Замер скорости операций. Решение этой задачи позволяет установить времена выполнения операций в самом благоприятном режиме, а именно однопользовательском режиме. Эти времена особенно интересны в процессе оптимизации информационной системы. Можно сравнить времена до и после выполнения оптимизации и оценить, насколько успешной была проведенная работа.

Таблица 3

Задачи, решаемые с помощью нагрузочных тестов

Задачи/ Общие приемы решения	Тестирование производительности			Тестирование функциональности
	Нагрузочное тестирование	Замер скорости операций	Заполнение тестовой БД	
Моделирование бизнес процессов	Одновременная работа множества сценариев	Последовательное выполнение сценариев	Последовательное выполнение сценариев в нужной пропорции	Последовательное выполнение сценариев
Циклические повторения	2 часа одновременной работы	10 раз (берется среднее)	N раз (требуемый объем БД)	2 раза (для включения краевых эффектов)
Датчик случайных чисел	Поведение пользователей	—	Для равномерности заполнения БД	—
Автоматическое создание пользователей	—	—	Требуемое количество	При автоматическом тестировании после сборки

- Заполнение тестовой БД. Чтобы получить репрезентативные результаты, нагрузочное тестирование необходимо выполнять на заполненной базе данных. Заполнение можно осуществить с помощью отдельной программы, написанной специально для этих целей. Но лучшие

результаты будут при заполнении с помощью тех же самых сценариев, выполняемых в последовательном режиме без временных задержек между отдельными шагами.

- Тестирование функциональности. Созданные сценарии можно применять и для регулярной (например, каждую ночь) автоматической проверки функциональности системы. При этом сценарии должны запускаться автоматически после компиляции системы. Они должны выполняться в порядке, в котором данные от одного сценария будут передаваться к следующему. В этом случае запуск может производиться и на пустой базе.

Кроме того, нагрузочные тесты можно применять для оценки производительности оборудования — при каком количестве пользователей и объеме базы данных на имеющемся оборудовании можно достичь приемлемых времен ожидания ответа [3].

6. Заключение

В данной работе сформулирована задача проведения нагрузочного тестирования. Разработаны методика и инструментарий, позволяющие проводить нагрузочное тестирование. Важно отметить, что разработанная методика позволяет решить задачу проведения нагрузочного тестирования для широкого круга информационных систем. Инструментарий представляет собой программу UILoadTest, обеспечивающую параллельную работу необходимого числа автоматических сценариев. Описанная методика проверена на практике в процессе проведения работ по оптимизации подсистемы Workflow программного комплекса Евфрат-Документооборот.

Литература

1. CppUnit — C++ port of JUnit. Сайт проекта библиотеки CPP Unit (http://sourceforge.net/projects/cppunit/).
2. TPC BENCHMARK™ C. Standard Specification. Revision 5.6 / Transaction Processing Performance Council, December 2005 (http://www.tpc.org/tpcc/spec/tpcc_current.pdf).
3. *Еременко А., Шашков Р.* Разработка бизнес-приложений в Microsoft Business Solutions — Axapta версии 3.0. Альпина Бизнес Букс, 2005.
4. *Даниленко А. Ю., Минкин Ю. И.* Анализ основных принципов построения и особенностей защиты информации в системах электронного документооборота // Документооборот. Концепции и инструментарий (Сборник трудов ИСА РАН). М.: УРСС, 2004.