УДК 004.94+004.05

М.А. Керенцева, Т.В. Мартыненко, Е.О. Савкова
Донецкий национальный технический университет, г. Донецк
Кафедра автоматизированных систем управления
E-mail: marinakerentseva@gmail.com, mart@kita.dgtu.donetsk.ua, helen-savkova@rambler.ru

РАЗРАБОТКА ГРАФО-АНАЛИТИЧЕСКОЙ МОДЕЛИ УЧЕТА НАГРУЗОК ВЕБ-БАЗИРОВАННЫХ СИСТЕМ

Аннотация
Керенцева М. А., Мартыненко Т. В., Савкова Е. О. Разработка графо-аналитической модели учета нагрузок веб-базированных систем. *Проанализированы принципы производительности и определены наиболее проблемные компоненты веб-базированных систем. Выполнен анализ типов нагрузки на веб-сервера и способов их оптимизации. Произведена классификация видов нагрузки на сервера по принадлежности к компонентам веб-системы. Для каждого класса предложены решения по снижению нагрузки. Разработана графо-аналитическая модель учета нагрузки веб-систем.*
Ключевые слова*: веб-базированная система, нагрузка, производительность, веб-сервер, веб-контейнер, база данных, граф, система нагрузочного тестирования.*

Введение. В настоящее время все большее количество компаний и организаций начинают общаться со своими клиентами с помощью Web. Организация электронного документооборота предприятия включает:

1) отслеживание работы с архивными документами;
2) взаимодействие с клиентскими документами;
3) банковские системы управления счетами;
4) системы по организации бизнес-поездок;
5) управление бухгалтерией и др.

Крупные компании, филиалы которых часто располагаятся в разных городах, требуют реализации управления внутренними процессами организации посредством внедрения беб-систем. Для безотказной работы веб-системы необходимо прогнозировать ее максимальную производительность (количество запросов, обрабатываемых в единицу времени), при которой система выдерживает пиковые нагрузки. Нагрузка на веб-систему определяет время отклика на запросы клиентов, что является одним из основных показателей эффективности ее работы. В настоящее время существует два основных способа повышения быстродействия веб-системы [1, 2]:

1. Программный.
   1.1. Разработка эффективных приложений для Web (быстродействие, используемые ресурсы)
   1.2. Оптимизация и модернизация программных компонентов Web-сервера
   1.3. Технология кэширования в среде Web
2. Аппаратный.
   2.1. Увеличение полосы пропускания.
   2.2. Установление высокопроизводительного сетевого оборудования
   2.3. Использование системы балансировки нагрузки программных продуктов. Система балансировки программных продуктов подразумевает распределение нагрузки по нескольким серверам, что повышает отказоустойчивость Web-базированной системы.

Целью данной работы является определение наиболее проблемных компонентов и фрагментов веб-базированных систем на основе анализа их производительности.

Постановка задачи. Типичный процесс обслуживания запроса к веб-серверу предусматривает выполнение следующих шагов [3]:

1. Клиент инициирует запрос к серверу.
2. Браузер устанавливает соединение с сервером.
3. Веб-сервер создает новый поток/процесс для обработки запроса.
4. Если клиент запросил динамический контент (например, отправил запрос к скрипту), веб-сервер создает отдельный процесс или запускает модуль обработки. В результате обработки запроса формируется web-страница, которая отправляется клиенту.
5. Если клиент запросил статический файл, то сервер просто отправляет этот файл клиенту.
6. Браузер клиента получает ответ, закрывает соединение с сервером и отображает ответ.

На рис. 1 представлен процесс работы типичной Web-системы.

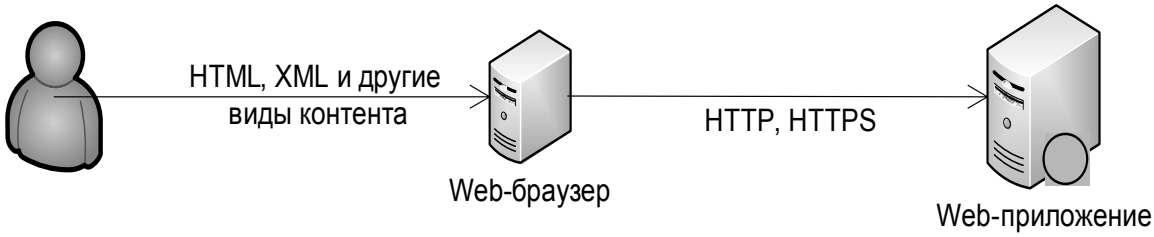

Рисунок 1 – Диаграмма работы Web-системы

Обращение к входящим в систему комонентам может быть неоднократным. При этом загрузка сервера может быть постоянной или динамически изменяющейся.

Предлагается представить обращение одного клиента к веб-системе в виде ориентированного взвешенного графа [4], где все пути являются простыми (рис.2).

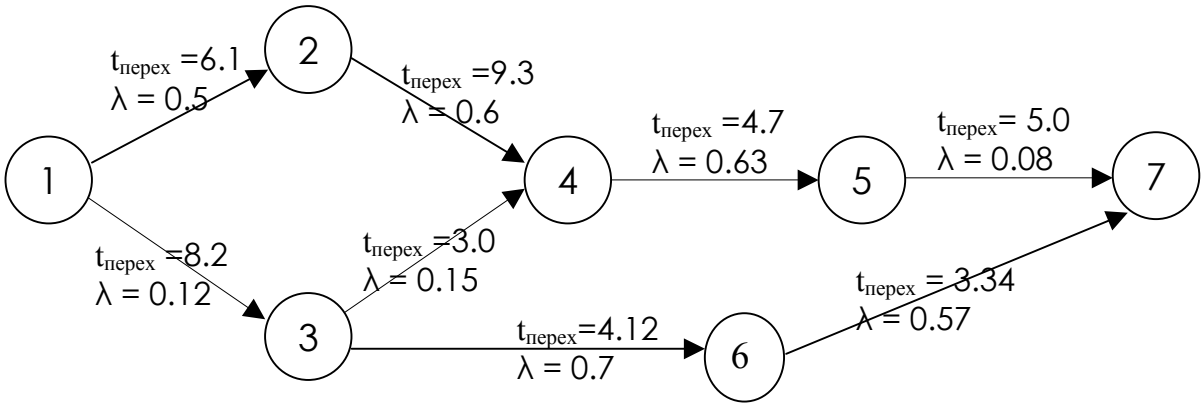

Рисунок 2 – Граф переходов запросов клиента веб-системы

Каждая вершина соответствует моменту получения ответа веб-базированной системы на запрос клиента (назовем это *состоянием системы*). Ребра графа отображают переходы между состояниями веб-системы. Анализ загруженности системы может быть выполнен на основе конечного числа сценариев, из которых для построения графа используются наиболее часто выполняемые. Такая упрощенная схема принята для возможности тестирования и

выявления критических переходов. Критическими будем называть переходы, на обработку которых система затратила наибольшее время.

Каждый переход характеризуется вектором из двух параметров:

1) временем загрузки сервера $t_{перех}$.

2) весовым коэффициентом λ, отражающим относительную частоту осуществления данного перехода. Данный параметр задается аналитиком тестируемой веб-системы.

Пример значений параметров приведен на рис. 2.

Классификация времени загрузки веб-базированной системы. Со стороны серверной части взаимодействие с клиентом можно характеризовать следующей последовательностью действий [3]:

1) прием от клиента запросов во время сессии;

2) передача их при необходимости в другой код (возможно, с удаленным доступом), для обработки запроса и обращения к данным;

3) генерация результатов для отображения в браузере.

Схема соединения с компонентами такой веб-базированной системы, содержащей один веб-сервер (кластера и фермы веб-серверов в данной работе не рассматриваются), представлена на рис. 3. и отображает ребро между двумя вершинами графа.

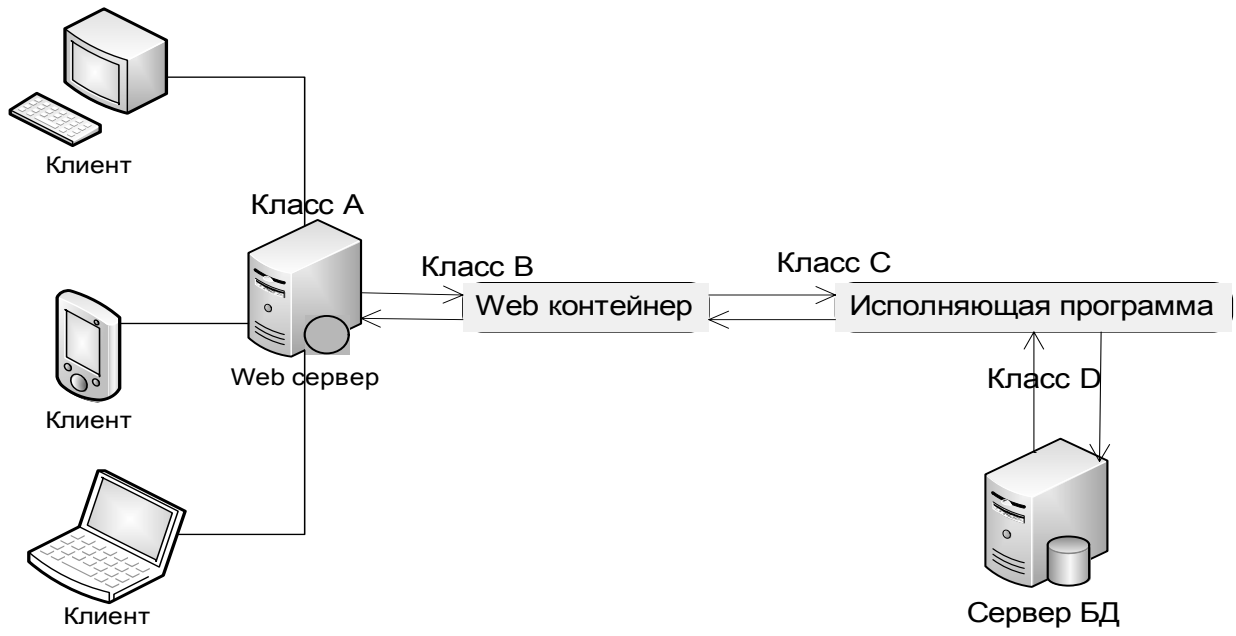

Рисунок 3 – Упрощенная схема web-соединения

На схеме процесс web-соединения разделен на ключевые операции, для обозначения которых предложен следующий классификатор:

Класс A – поступление запроса от клиента к веб-серверу или получение ответа.

Класс B – передача запроса/ответа между веб-сервером и веб-контейнером. сборка/разборка ответа/запроса веб-контейнером.

Класс C – передача разобранного запроса исполняющей программе, ее работа.

Класс D – обращения к базе данных (БД).

Каждая из указанных операций характеризуется набором параметров. Одним из основных параметров является время выполнения операции. Обозначим время выполнения операции для класса A: t_A, для класса B – t_B и т.д. Общее время перехода будет суммарным показателем выполнения всех операций:

$$t_{перех} = t_A + t_B + t_C, \quad (1)$$

$$t_A = |t_{вс} - t_з|, \quad (2)$$

где $t_{вс}$ – момент получения запроса (отправки ответа) веб-сервером;

$t_з$ – момент отправки запроса (получения ответа) клиентом.

$$t_B = |t_к - t'_{вс}|, \quad (3)$$

где $t_к$ – момент отправки запроса в исполняющую процедуру (либо получения ответа от исполняющей процедуры);

$t'_{вс}$ – время получения ответа (отправки запроса) между веб-сервером и веб-контейнером.

Следует отметить, что знак выражения $t_к - t'_{вс}$ определяется направлением операции (запрос или ответ), поэтому в (3) предагается использовать абсолютную величину времени выполнения.

Класс D является подклассом класса C, поскольку время работы исполняющей программы включает в себя время обращений к базе данных:

$$t_C = t'_C + t_D, \quad (4)$$

где t_D – суммарное время обращений к базе данных,

t'_C – время работы исполняющей программы без обращений к базе данных.

С учетом (4), формулу (1) можно записать в виде:

$$t_{перех} = t_A + t_B + t'_C + t_D \quad (5)$$

Для снижения загрузки веб-базированной системы необходимо:

Этап 1. Определить критические переходы (рис. 2).

Этап 2. Выделить классы, требующие минимизации.

Этап 3. Выбрать средства снижения нагрузки.

Этап 1. *Для реализации 1-го этапа* необходимо выполнить моделирование множества обращений клиентов к веб-системе и определение наиболее критического перехода с максимальным временем выполнения, для чего предлагается использовать *систему нагрузочного тестирования* [5]. Для оценки всех показателей системы используется критерий Производительности (Performance) испытания, который представляет собой Нагрузочный (Load-testing) тест и тест Устойчивости (Stress). Тестирование заключается в имитации формирования запросов к веб-системе от нескольких тысяч посетителей с формированием кратковременных пиков нагрузки (в автоматическом режиме моделируется прохождение по ссылкам веб-системы одновременно многими виртуальными пользователями). Система нагрузочного тестирования выдает такие показатели как: время отклика системы, количество байт, участвовавших в транзакции, коды ответов сервера и т.п.

Для исследования одного пути на критичность используется отдельная группа потоков (Thread Group), в настройках которой задается выбранный сценарий. Каждое состояние системы (вершина графа на рис. 2) реализуется посредством добавления образца запроса (Sampler).

Введем показатель значимости j-го перехода μ_j ($j = \overline{0,k}$, где k – общее количество переходов):

$$\mu_j = \bar{t}_j * \lambda_j, \quad (6)$$

где λ_j – весовой коэффициент; $\bar{t}_j$ – усредненное время j-го перехода:

$$\bar{t}_j = \frac{1}{n}\sum_{i=1}^{n} t_{перех_i}, \quad (7)$$

где n – количество запусков сценария на выполнение; $t_{перех_i}$ – время загрузки сервера на j-м переходе.

Получив показатели значимости μ_j всех переходов по формуле (6), выбирается наиболее критический путем ранжирования вектора μ.

Этап 2. *При определении класса, требующего минимизации* (значение параметра *max{tj}*) производится тестирование выбранного критического перехода для получения средних значений $\overline{t_A}$, $\overline{t_B}$ и $\overline{t_C}$. При этом выполняемый сценарий будет состоять из одного образца запроса – того, который является конечным состоянием критического перехода.

$$\overline{t_A} = \frac{1}{l}\sum_{i=1}^{l} t_A , \tag{8}$$

где *l* – количество запусков сценария на выполнение.

Аналогично определяются средние значения t для классов B и C. Значения t_A, t_B и t_C определяются по формулам (2-4). Для получения параметров t'_C и t_D внутри кода исполняющей программы необходимо встроить счетчики времени, которые для каждого запроса будут измерять время, ориентируясь на текущую сессию пользователя.

Получив средние значения всех классов, необходимо выбрать критический класс – класс, имеющий максимальное значение параметра t.

Этап 3. *Принятие решения по способу снижения нагрузки* зависит от полученного на этапе 2 критического класса.

Для класса A используются аппаратные способы снижения нагрузки такие как: увеличение полосы пропускания, улучшение сетевого оборудования и т.д.

Для класса B решением может быть замена связки сервер-контейнер (аппаратное).

Для класса C – модификация программного кода и/или оптимизация структуры БД и работы ней.

Поскольку уменьшение значений t_A и t_B требуют внешнего вмешательства (что не всегда может быть достигнуто на практике), более детально рассмотрим снижение нагрузки для классов C и D. Для этого необходимо проделать шаги 1-3, но выполнять ранжирование временных показателей значимости μ с учетом только классов C и D. Для этого необходимо модифицировать формулу (5):

$$t_{перех} = t'_C + t_D \tag{9}$$

Тогда

$$\mu_j = \bar{t}_j * \lambda_j = \frac{1}{n}\sum_{i=1}^{n} t_{перехi} * \lambda_j = \frac{1}{n}\sum_{i=1}^{n} (t'_{Ci} + t_{Di}) * \lambda_j , \tag{10}$$

где λ_j – весовой коэффициент перехода;

n – количество запусков сценария на выполнение;

t_{Di} – время обращений к базе данных на i-том запуске,

t'_{Ci} – время работы исполняющей программы без обращений к базе данных на i-том запуске.

Выделив таким образом «узкие места», необходимо принимать решения по снижению нагрузки:

- Модификация кода исполняющей программы.
- Использование или внедрение более эффективных серверных технологий.
- Оптимизация запросов к БД
  - Кэширование виджетов – полезных списков и ссылок, которые отображаются на каждой странице сайта. Информация в виджетах обновляется не так часто и ее можно кэшировать.

  - Составление списка активных запросов к базе. Запросы, время выполнения у которых самое большое, необходимо программно оптимизировать, либо кэшировать (если они еще не кэшированы).
  - Сокращение числа сложных запросов.
- Организация обращений к сессиям. Сессии являются файловым хранилищем (файлы сессий хранятся на жестком диске сервера), поэтому обращение к ним следует минимизировать.
- Формирование кластеров БД.
- Другие решения: специфичные для отдельно взятого типа базы данных и позволяющие снизить количество обращений к ней, улучшить функционирование и качество программных средств.

Результаты тестирования модели. Для проведения экспериментальных исследований были выбраны следующие характеристики веб-системы:

1) Application Server: Apache Tomcat 6.0;
2) DB: SQL Server;
3) OS: Windows 2003 Server;
4) Java: JDK 6.0.12.

В работе использована система тестирования *Apache JMeter* [5] (jakarta.apache.org/jmeter), которая является Java-приложением с открытым кодом и предназначена для нагрузочного тестирования веб-приложений, их отдельных компонентов (скриптов, сервлетов, Java объектов и др.), FTP-серверов, баз данных (с использованием JDBC) и сети.

Для того, чтобы иметь возможность логировать и обрабатывать полученную информацию, выбран тип слушателя под названием «Simple Data Writer», который позволяет осуществить запись результатов теста в файл.

Оценка общего времени t_C производилась внедрением фильтра в web приложение, оценка времени t_D происходила в момент создания и разрушения соединения с базой данных.

В результате исследований наиболее критичным оказался Класс D. Для устранения «узкого места» была выполнена оптимизация запросов к базе данных.

В критичных участках система была перепроектирована таким образом, чтобы вместо использования технологии Hibernate были осуществлены запросы к БД непосредственно через JDBC.

Данное решение позволило сократить время выполнения критичных участков веб-системы на 21%.

Выводы. В работе исследована задача снижения нагрузки веб-серверов с учетом характеристик компонентов, специфических для конкретной системы.

Разработана графо-аналитическая модель учета нагрузки веб-базированных систем, которая включает граф переходов запросов клиента и математическую формализацию состояний системы.

Выполнена классификация типов нагрузки серверов по принадлежности к компонентам веб-системы.

Для каждого класса предложены варианты решения по снижению нагрузки.

При развитии модели возможно добавление параметров для вершин графа, которые определяют время нахождения системы в данном состоянии. Возможны доработки модели для оптимизации отдельных классов нагрузки.

Разработанная модель может использоваться для прогнозирования стрессоустойчивости серверов.

Литература:

1. ТАО ЧЖОУ “Системы балансировки нагрузки Web-серверов”, Журнал "Windows 2000 Magazine", #03/2000
2. Богомолов С.В., Спирягин В.И. «Создание структурно оптимальной системы управления веб-сайтом» [Электр. ресурс], Режим доступа к изд. : http://xpoint.ru/know-how/Articles/ModelCMS
3. James McGovern, Rahim Adatia, Yakov Fain. “Java™ 2 Enterprise Edition 1.4 Bible”, Published by Wiley Publishing, Inc. 10475 Crosspoint Boulevar Indianapolis, IN 46256, www.wiley.com Copyright © 2003 by Wiley Publishing, Inc., Indianapolis, Indiana
4. Уилсон Р. Введение в теорию графов. Пер с англ. М.: Мир, 1977. 208с. [Электр. ресурс], Режим доступа к изд. : http://eqworld.ipmnet.ru/ru/library/books/Uilson1977ru.djvu
5. “Apache JMeter” Copyright © 1999-2009, Apache Software Foundation, URL: http://jakarta.apache.org/jmeter/

Abstract

Kerentseva M.A., Martynenko T.V., Savkova E.O. Development graph-analytic model accounting loads of web based systems. *Performance web based systems is analyzed, identified the most problematic components. The analysis of the server load principles and ways of its optimization is performed. Implemented research work in reducing the problem of load web servers, taking into account the factors and components, specified for a particular system. Completed classification of types of load on the servers belonging to the components of a web system. For each class offered a possible solution to reduce the load. A graph-analytical model accounting load web based systems designed.*

Key words*: web-based system, load, performance, Web server, Web container, database, graph, system load testing.*

Анотація

Керенцева М. А., Мартиненко Т. В., Савкова О. Й. Розробка графо-аналітичної моделі обліку навантажень веб-базованих систем. *Проаналізовано принципи продуктивності та визначені найбільш проблемні компоненти веб-базованих систем. Виконано аналіз типів навантаження на веб-сервера і способів їх оптимізації. Здійснена класифікація видів навантаження на сервери за належністю до компонентів веб-системи. Для кожного класу пропонуються можливі рішення щодо зниження навантаження. Розроблено графо-аналітичну модель обліку навантаження веб-систем.*

Ключові слова: *веб-базована система, навантаження, продуктивність, веб-сервер, веб-контейнер, база даних, граф, система навантажувального тестування.*

Здано в редакцію: 26.03.2010р.

Рекомендовано до друку: д.т.н, проф. Зорі А.А.