стик заготовки в требуемые показатели качества готовой машиностроительной продукции. Поток по качеству характеризуется следующими параметрами (дальше потоковые характеристики или параметры потока), соответствующими нормативным значениям показателей качества изделия: параметры шероховатости поверхности, микрогеометрия, размеры, соответствие формы, микротвердость и т. п.

Библиографические ссылки

1. *Гличев А. В.* Качество, эффективность, нравственность : учеб. пособие. – М. : Премиум Инжиниринг, 2009. – 358 с. : ил.

2. *Канке А. А.*, *Кошевая И. П.* Основы логистики : учеб. пособие. – М. : КНОРУС, 2010. – 576 с.

3. *Гиссин В. И.* Управление качеством. – 2-е изд. – М. : МарТ ; Ростов н/Д : МарТ, 2003. – 400 с.

Y. G. Gushyan, Togliatti State University

Logistic Approach to Quality Construction of Machinery Parts

The paper considers the problem of ensuring the quality of manufacturing machine parts and the necessity of a systematic approach to management of manufacturing quality. Possibility of applying a logistic approach to management of manufactured machine parts quality is substantiated.

Key words: quality management, logistics approach to quality management, technological support for machine parts, technological heredity.

УДК [004.032.34+004.451.2].942:004.738.2

М. В. Тюлькин, аспирант, Пермский национальный исследовательский политехнический университет
И. В. Капгер, Пермская печатная фабрика (филиал ФГУП «ГОЗНАК»)
Е. Л. Кротова, кандидат физико-математических наук, Пермский национальный исследовательский политехнический университет
Л. Н. Кротов, доктор физико-математических наук, профессор, Пермский национальный исследовательский политехнический университет

РАЗРАБОТКА АРХИТЕКТУРЫ И ОРГАНИЗАЦИЯ ИНФОРМАЦИОННЫХ ПОТОКОВ В *COMET*-СЕРВЕРАХ ДЛЯ *WEB*-ПРИЛОЖЕНИЙ МОДЕЛИ *COMET* СО СХЕМОЙ ВЗАИМОДЕЙСТВИЯ *WEBSOCKET*. ОПИСАНИЕ *COMET*-СЕРВЕРА

Предпринимается попытка выделить основные составляющие элементы Comet-сервера, такие как различные информационные структуры данных и обрабатывающие их вычислительные потоки, а также предлагаются различные архитектурные решения взаимодействия между данными элементами посредством организации информационных потоков в зависимости от задач, решаемых Comet-сервером в условиях высокой нагрузки.

Ключевые слова: архитектура программ, высоконагружаемое приложение, информационный обмен, вычислительный поток, Comet-сервер, сокет-сервер, сокет, сокеты Беркли, клиент-серверное приложение.

Современные тенденции развития web-приложений (далее приложений), такие как перенос Desktop-приложений и построение высоконагружаемых проектов в Web, нацеленных на работу с большой аудиторией пользователей и включающих интенсивный информационный обмен внутри себя, предъявляют к архитектуре web-приложений, а в частности к информационную обмену внутри приложения, следующие новые требования [1, с. 2]:

1) оповещение клиента сервером о наличии новой информации;

2) поступление информации к клиенту с минимальной задержкой по времени между появлением данной информации и окончанием доставки;

3) соблюдение хронологии появления информации при ее передаче;

4) защита передаваемых данных от модификации и пассивного перехвата (при необходимости).

Классическая модель web-приложений, работающая по схеме «запрос – ответ» с использованием HTTP-протокола в качестве основного транспорта данных между клиентом и сервером [2], либо удовлетворяет данные требования не в полной мере, причем малоэффективно и ресурсозатратно [1, с. 2], либо не удовлетворяет вообще, когда следование данным требованиям критично.

Наиболее эффективным решением данной задачи является переход устройства web-приложения от классической модели к так называемой Comet-модели, которая предусматривает различные схемы взаимодействия, реализующие выполнение данных требований. Наиболее легко внедряемой и перспек-

Получено 18.05.12
Работа выполнена при финансовой поддержке Министерства образования Пермского края и при финансовой поддержке гранта РФФИ – Урал № 11-01-96015.

тивной является схема взаимодействия *WebSocket Streaming*, которая предполагает введение в информационный обмен между клиентом и сервером третьей стороны, называемой Comet-сервером.

Браузеру пользователя, чтобы начать работу с приложением, а следовательно, вступить в информационный обмен, необходимо послать запрос к серверу (далее web-серверу), в ответ на который web-сервер генерирует и пересылает браузеру пользователя экземпляр клиента приложения (клиентской части – frontend) с включенными в него данными о статусе пользователя в приложении для Comet-сервера. Статусные данные включают в себя следующее:

– идентификатор пользователя;

– информационные каналы, на которые подписан клиент пользователя (например, основной чат, чат друзей, канал сервисной информации);

– штамп даты и времени, когда пользователь начал работу с приложением, т. е. отправил запрос на получение экземпляра клиента web-серверу.

Web-сервер не может передать данные о пользователе Comet-серверу напрямую, поскольку ввиду особенности построения компьютерных сетей Comet-сервер не может определить, какой подключившийся клиент принадлежит данному пользователю [3, с. 40]. Поэтому клиент в браузере пользователя на этапе инициализации самостоятельно устанавливает соединение с Comet-сервером и пересылает данные о статусе. Поскольку между первым запросом к web-серверу и подключением клиента к Comet-серверу могли произойти события, требующие уведомления клиента, то Comet-сервер ведет историю сообщений некоторого объема и при подключении клиента высылает ему архивные сообщения в той хронологии, в какой они были получены от web-сервера. Важно отметить, что за сеанс работы с приложением статус пользователя может меняться, и поэтому пересылка статуса клиентом также может происходить несколько раз, в том числе и несколько раз может требоваться выдача истории сообщений в зависимости от логики работы приложения.

Когда на стороне сервера возникает событие, требующее оповещения клиента, он формирует соответствующее сообщение с информацией, необходимой клиенту, и пересылает его Comet-серверу. Данное сообщение характеризуется следующими данными:

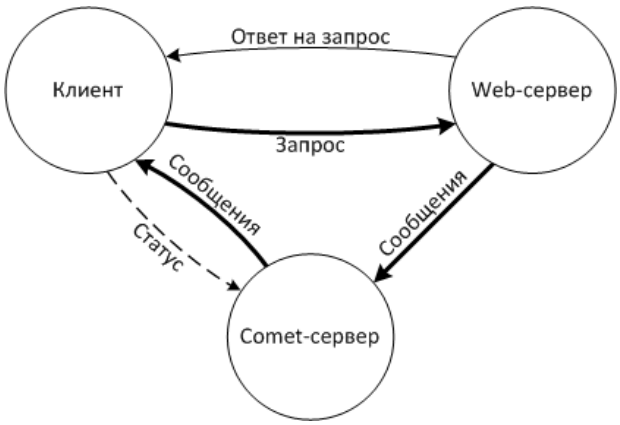
– список идентификаторов пользователей, которым предназначено данное сообщение, или условная метка широковещательной рассылки;

– список информационных каналов, на которые должно быть выслано данное сообщение.

Получив сообщение от web-сервера, Comet-сервер запоминает метку даты и времени прихода сообщения и добавляет его в историю. Затем проверяет соответствие данного сообщения статусу пользователя клиента и пересылает его, если статус соответствует.

Данный информационный обмен иллюстрирует следующий рисунок.

Архитектура Comet-сервера может быть различной в зависимости от решаемых web-приложением задач, но, как правило, все задачи такого рода должны решаться в условиях высокой нагрузки. При этом организация эффективного и быстрого информационного обмена внутри Comet-сервера между его элементами посредством информационных потоков в таких условиях превращается в нетривиальную задачу.



Информационный обмен в Comet-риложении

Задачи и функции Comet-сервера

Comet-сервер является программным обеспечением (далее ПО), которое выполняется на ЭВМ сервера, при этом, как правило, на этой же ЭВМ запущен web-сервер. Его основная задача – это рассылка информации от web-сервера клиентам.

Помимо рассылки сообщений Comet-сервер должен обрабатывать запросы от web-серверов и клиентов, не касающиеся пересылки, такие как подключение и отключение, обработка статусной информации клиентов, пересылка истории сообщений.

Также Comet-серверу необходимо поддерживать и обслуживать большое число сетевых соединений ввиду большой аудитории пользователей приложения, чья активность как правило неравномерно распределена во времени, а имеет пики и спады.

Поэтому Comet-сервер должен эффективно взаимодействовать со структурированными данными пользователей и истории сообщений, чтобы обеспечить максимальную производительность даже во время пиков нагрузки. Поскольку Comet-сервер как элемент взаимодействия в приложении скрыт от конечного пользователя, то пользователь не должен замечать его влияния, в данном случае отрицательного – торможение работы приложения.

При выполнении всех этих требований архитектура Comet-сервера должна максимально эффективно использовать архитектуру серверных ЭВМ, чтобы равномерно распределить нагрузку и по возможности свести ее к минимуму.

Дополнительно Comet-сервер должен предоставлять какой-либо интерфейс для взаимодействия с нешним управляющим элементом (например, стороннее ПО или человек), через который осуществля-

ется передача статусной информации о работе Comet-сервера и прием управляющей информации.

Элементы Comet-сервера

Рассмотренные выше задачи и функции Comet-сервера определяют элементы его архитектуры. Поэтому попробуем выделить основные элементы программы, между которыми будет происходить взаимодействие:

– потоки выполнения – потоки выполняющие процесс (алгоритм) Comet-сервера. На один процесс Comet-сервера операционная система (далее ОС) выделяет один основной поток, который в последующем может породить дочерние потоки. Один поток может выполняться только на одном вычислительном ядре процессора ЭВМ;

– области данных в оперативной памяти. Поскольку во время работы Comet-сервер пропускает через себя большое количество информации малого объема, то для временного хранения данной информации наиболее подходит ОЗУ ЭВМ;

– сокеты – программные интерфейсы для обмена информацией между удаленными процессами, реализованные посредством сокетов Беркли.

Отметим, что сокеты должны быть переведены в неблокирующий режим, чтобы во время работы с сокетом локального потока удаленный поток мог посылать или принимать данные, т. е. блокировки такого ресурса, как сокет существуют только относительно локальных потоков.

Забегая вперед, до рассмотрения возможных архитектур Comet-сервера, введем следующие обозначения, которые будут использоваться на рисунках, иллюстрирующих соответствующую архитектуру:

а) T_i – вычислительный поток, выполняющийся в рамках процесса Comet-сервера. Индекс i указывает на то, какую функцию данный поток выполняет. Он может быть единственным или множественным и принимает следующие значения:

1) M – основной поток, который стартует при запуске Comet-сервера в системе, инициализирует все структуры данных и интерфейсы взаимодействия, открывает сокеты, порождает дочерние потоки;

2) I – поток, обслуживающий внешний интерфейс взаимодействия;

3) K – поток, обслуживающий клиентов (подключение и отключение клиентов, обработка статусной информации);

4) S – поток, обрабатывающий запросы серверов, анализирует сообщения от сервера и добавляет их в историю;

5) D – сервисный поток, обслуживающий Comet-сервер, удаляет старые сообщения из истории и информационный «мусор», обслуживает сервисные структуры данных;

6) Ch – поток, осуществляющий хронологическую пересылку сообщений клиентам в тех моделях архитектуры, где это необходимо;

б) Ss_i – слушающий сокет на определенном порту хоста. Дополнительно мы будем считать, что с данным сокетом сопряжена структура данных, хранящая все адреса интерфейсов (дескрипторы сокетов) для связи с отдельными клиентами или серверами. Индекс i указывает на функцию данного сокета и может быть единственным или множественным. Сокет принимает следующие значения:

1) K – слушающий сокет, обрабатывающий подключения клиентов;

2) S – слушающий сокет, обрабатывающий подключения серверов;

в) S_i – сокет, доступ, к которому осуществляется через его дескриптор. Индекс i указывает на функцию данного сокета и может быть только единственным. Индекс принимает следующие значения:

1) K – сокет, обменивающийся информацией с клиентом;

2) S – сокет, обменивающийся информацией с сервером;

г) I – интерфейс взаимодействия с внешним управляющим элементом;

д) s – структура статусных данных, отражает все аспекты текущего состояния Comet-сервера;

е) K – область данных клиентов, в которой хранятся статусные и служебные данные о клиентах, подключенных к Comet-серверу. Отметим, что к перечню служебных данных относится и дескриптор сокета клиента;

ж) H – область архивных сообщений или история сообщений, где хранится определенный объем сообщений, разобранных на составляющие (список пользователей, список каналов, метка времени, текст сообщения), полученных Comet-сервером от web-сервера;

з) C_h – область архивных сообщений с соблюдением хронологии, полностью идентична предыдущей структуре данных за исключением того, что в один момент времени из данной структуры рассылается только одно сообщение. Будет упоминаться для специфичных архитектур Comet-сервера, в которых возможна рассылка сообщений из истории с нарушением хронологии их появления, в остальных случаях замещается предыдущей структурой данных;

и) T – область данных контроля запущенных потоков, структура данных сервисного характера. Она необходима для контроля запущенных потоков для архитектур, в которых порождение дочерних потоков зависит от внешних факторов.

Также на всех рисунках, иллюстрирующих соответствующую архитектуру, будут обозначаться информационные потоки от ресурса к вычислительному потоку (процедура чтения) и от вычислительного потока – к ресурсу (процедура записи):

а) жирной пунктирной линией – блокирующая процедура, не длительная во времени;

б) простой пунктирной линией – неблокирующая процедура, не длительная во времени;

в) жирной сплошной линией – блокирующая процедура, длительная во времени;

г) простой сплошной линией – неблокирующая процедура, длительная во времени.

Все процедуры чтения к областям данных являются длительными, поскольку сопряжены с поиском необходимой информации. А все процедуры записи – короткими, поскольку модифицируют уже найденный элемент или дописывают в конец области новый. Блокировка структуры потоком предусматривает следующие действия:

– если структура заблокирована другим потоком, то переход в состояние ожидания пока структура не будет разблокирована, т.е. пока другой поток не закончит с ней работу;

– блокировка структуры для любого другого потока;

– работа со структурой – запись или чтение;

– разблокировка структуры.

Блокировка структуры необходима в том случае, если возможно чтение или запись данных из структуры одним потоком, в то время как другой поток может осуществлять запись в данную структуру. Блокировки порождают простои в работе потоков, поэтому должны сводиться к минимуму.

Библиографические ссылки

1. *Crane D.*, *McCarthy P.* Comet and Reverse Ajax: The Next-Generation Ajax 2.0, 2008. – 142 с.

2. Hypertext Transfer Protocol -- HTTP/1.1. Июнь 1999. – URL: http://www.w3.org/Protocols/rfc2616/rfc2616.html (дата обращения: 18.02.2011).

3. *Снейдер Й.* Эффективное программирование TCP/IP. Библиотека программиста. – СПб. : Питер, 2002. – 320 с. : ил.

M. V. Tyulkin, Post-graduate, Perm National Research Polytechnic University
I. V. Kapger, Perm Printing Factory (branch) of the Federal State Unitary Enterprise "GOZNAK"
E. L. Krotova, PhD (Physics and Mathematics), Associate Professor, Perm National Research Polytechnic University
L. N. Krotov, DSc (Physics and Mathematics), Professor, Perm National Research Polytechnic University

Architecture Development and Organization of Data Flows in Comet-Servers for Comet Web-Applications with "Web-Socket Streaming" Interaction Scheme. Part 1 – Description of Comet-Server

The paper offers a highlight of the main constituent elements of Comet-Server, such as various data structures and processing computation flows. Different architectural solutions are also proposed on interaction between these elements through organization of data flows depending on tasks solved by Comet-server under high loads.

Key words: software architecture, high-loaded application, data exchange, computation flow, Comet-server, Socket-server, Socket, Berkeley sockets, client-server application.

УДК 681.32

С. Ф. Тюрин, доктор технических наук, профессор, Пермский национальный исследовательский политехнический университет
О. А. Громов, аспирант, Пермский национальный исследовательский политехнический университет
А. Н. Каменских, студент, Пермский национальный исследовательский политехнический университет

ПРОГРАММНЫЙ КОМПЛЕКС ИССЛЕДОВАНИЯ МЕТОДОВ ПОВЫШЕНИЯ НАДЕЖНОСТИ

Предлагается алгоритм поиска оптимальной структурной схемы надежности системы. Приводится пример расчета структурной схемы надежности. Описывается программа, которая позволяет реализовать данный алгоритм и исследовать возможные способы повышения надежности системы для заданных параметров.

Ключевые слова: структурная схема надежности, метод наискорейшего спуска, мажоритирование, резервирование, функционально полный толерантный элемент.

При синтезе современных сложных информационно-управляющих систем всегда возникает вопрос выбора оптимальной структуры. При этом проектируемая система должна реализовывать все функции, заданные в техническом задании, и иметь минимальную стоимость, чтобы обеспечить конкурентоспособность устройства. Задача выбора оптимальной структуры также остро стоит для проектировщиков систем специального назначения. В подобных системах большое значение имеет надежность системы, и поэтому в процессе разработки новых устройств как правило приходится увеличивать количество аппаратуры, чтобы обеспечить нужную наработку на отказ. Однако увеличение объема аппаратуры приводит к ухудшению многих других параметров, например, таких, как стоимость устройства, мощность, рассеиваемая устройством, массогабаритные характеристики и т. д.

Получено 17.01.12