Журнал "Открытые системы", #11-12, 1999 год

Издательство "Открытые системы" (www.osp.ru)

Постоянный адрес статьи: http://www.osp.ru/os/1999/11-12/017.htm

Универсальный сервер Unis

Дмитрий Волков

16.11.1999

Ресурсы, накопленные за годы существования СУБД часто оказываются невостребованы из-за проблем обеспечения широкого доступа к ним через Web. Сервер Unis, более трех лет являющийся технологической основой разработки информационных систем в компании «Демос-Интернет», призван облегчить эту задачу путем реализации функций универсального шлюза для выхода во Всемирную паутину. Основная цель разработки — реализация интеллектуального алгоритма обработки запросов и распределения вычислительных ресурсов, обеспечивающего управление качеством сервиса, а не только его предоставление как такового. В статье рассмотрены основные принципы внутренней организации сервера, особенности его реализации и использования для удержания показателей качества (например, времени ответа) в некоторых заданных пределах.

Проблема обеспечения качества сервиса наиболее остро стоит перед разработчиками информационных систем, работающих в режиме реального времени. Например, время ответа системы обслуживания биржевых торгов должно быть минимальным, чтобы оставить брокеру как можно больше времени на принятие решения. Уменьшить время ответа можно, увеличив вычислительные мощности, обслуживающие систему, но эти мощности будут востребованы лишь на короткий период торговой сессии, все остальное время они будут простаивать. Проблема может быть решена путем размещения на одних и тех же вычислительных ресурсах различных информационных систем с разными приоритетами, например, разместить на одном сервере систему on-line торгов и систему доступа к архивной информации, менее требовательную ко времени ответа. В этом случае, на время торговой сессии максимальное количество ресурсов должно быть предоставлено системе торгов, в остальное время ресурсы будут использованы системой доступа к архиву.

Большинство существующих на сегодняшний день технологий не предусматривают реализацию подобных саморегулирующихся систем и тем более не позволяют сформулировать требования в терминах показателей качества, например: «время ответа системы X не должно превышать 1 секунды, а вероятность отказа системы Y обязана быть на более 3%». Универсальный сервер Unis предоставляет разработчикам подобный интерфейс управления системой в целом. Эта возможность обеспечена использованием в Unis оригинальных алгоритмов, основанных на методах теории массового обслуживания [**1**] и позволяющих в реальном времени оценивать и корректировать показатели качества сервиса.

Общая схема организации

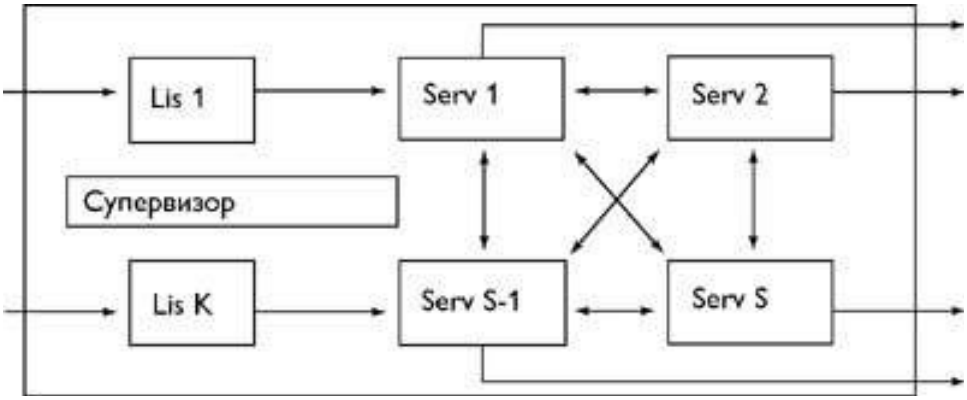

Рис.1. Схема универсального сервера

Схема универсального сервера представлена на рис.1. Как видно из рисунка, сервер состоит из модулей трех типов: управляющий модуль (Supervisor), приемники (Lis 1 — Lis K) и сервисы (Serv 1 — Serv S). В задачи управляющего модуля входят обеспечение запуска и остановки сервера, распределение ресурсов компьютера между модулями, сбор и визуализация статистики. Приемник обеспечивает получение запроса от клиента и передачу его одному из сервисов. Задача сервиса — принять клиентский запрос от приемника, обработать и либо сообщить клиенту о завершении обработки (успешном или ошибочном), либо передать частично обработанный запрос другому сервису. Все сервисы имеют одинаковую структуру (рис.2), различия между ними заключаются лишь в логике функционирования обработчиков.

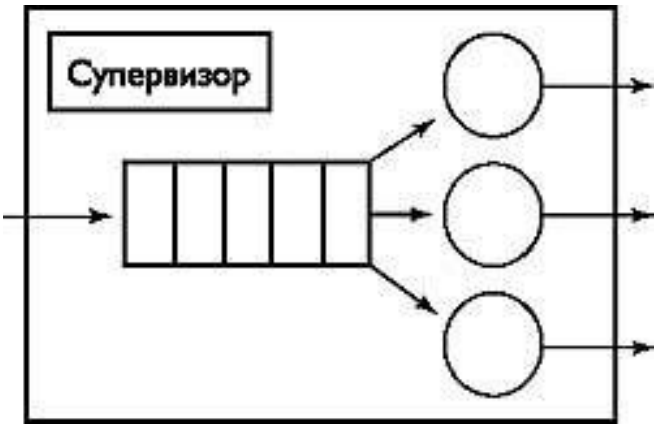

Рис.2. Схема организации сервиса

Сервис состоит из управляющего модуля, набора равноправных обработчиков и единой для всех обработчиков очереди. Запрос попадает в очередь, после чего, если имеется свободный в данное время обработчик, то запрос поступает к нему, если же свободного обработчика нет, то запрос находится в очереди до тех пор, пока не освободится один из обработчиков или управляющий модуль не примет решения о необходимости увеличения числа обработчиков. После завершения обработки запрос либо покидает систему, либо передается для дальнейшего обслуживания другому сервису.

Если посмотреть на Unis с точки зрения архитектуры клиент-сервер, то мы увидим, что это несколько серверных компонентов, объединенных единым блоком управления. Подобная схема реализации универсального сервера имеет целый ряд достоинств. Во-первых, она хорошо подходит для реализации на объектно-ориентированном языке программирования, что, в свою очередь, ведет к существенному снижению трудозатрат на расширение функциональности сервера. Фактически, для добавления нового сервиса необходимо лишь создать модуль обработчика, вся остальная инфраструктура уже существует и едина для всех сервисов. Второе достоинство в том, что распределение ресурсов компьютера между сервисами происходит на уровне модуля управления сервера, что позволяет реализовать более гибкую схему распределения, чем в случае, когда сервисы — это независимые процессы и распределение ресурсов

происходит на уровне операционной системы. Кроме того, благодаря наличию связей между сервисами внутри сервера, значительно снижается накладной расход на обработку комплексных запросов, то есть таких запросов, в обработке которых участвуют два или более сервисов.

Обмен данными между клиентом и сервером происходит по протоколу TCP/IP — началом сеанса работы можно считать организацию нового TCP/IP соединения между клиентом и приемником сервера. Задача приемника в этом случае — просто принять соединение и, не производя никаких действий, передать его на обработку специально выделенному сервису-диспетчеру. Заметим, что диспетчер выделен лишь с точки зрения логики его работы, в остальном — это такой же сервис, как и все остальные. В задачи диспетчера входят (в порядке исполнения).

1. Безусловная проверка возможности соединения (допустимо ли вообще соединение с IP адреса клиента).
2. Получение заголовка запроса (кто осуществил соединение и к какому сервису требуется обращение, при необходимости — авторизация клиента).
3. Условная проверка возможности соединения (имеет ли доступ этот клиент к запрашиваемому сервису).
4. Вычисление приоритета запроса, исходя из внутренних правил и пожеланий клиента.
5. Постановка запроса в очередь на обслуживание требуемым сервисом в соответствии с вычисленным приоритетом.

Задача сервиса в данной конфигурации — извлечь из очереди запрос клиента, обработать его, возвратить клиенту результат и либо завершить обработку запроса, либо передать его (поставить в очередь) на дальнейшую обработку другому сервису.

Применение системы

Настройка сервера на конкретные задачи производится с помощью конфигурационного файла, состоящего из трех основных секций:

- общие настройки системы;
- определение сервисов;
- определение приемников.

Ниже приведен конфигурационный файл системы Unis, настроенной на обработку запросов к СУБД Oracle и MySQL:

```
# Общие настройки
# Максимальное количество обработчиков в системе
MAX_ACTIVE_SERVERS      80
# Приоритет сервиса по умолчанию
DEFAULT_PRIORITY        10
# Log файл и уровень детализации лога
LOG_FILE                /www/unis/main.log
LOG_LEVEL               2
# Файл, в который будет помещаться статистика системы
STAT_FILE               /www/unis/stat.log
# Файл, в который будет помещаться PID процесса
PID_FILE                /www/unis/unis.pid

# Определение сервисов SQL Server for ORACLE
```

```
# Начало описания сервиса
#  @SERVER — ключевое слово
#  SQL — тип севриса
#  ora_wsql — имя сервиса
@SERVER:SQL             ora_wsql
PRIORITY                10
LOG_FILE                /www/unis/wsql.log
LOG_LEVEL               2
# Тип интерфейса к БД и параметры соединения с БД
SQL_SERVER              ORACLE
CONNECT_STRING          user/passwd
# Максимальное, минимальное и стартовое число обработчиков
MAX_SERVERS             20
MIN_SERVERS             10
START_SERVERS           15
# Максимальная длина очереди
QUEUE_LEN               48
# С каких адресов разрешены запросы
ALLOW_IP                127.0.0.1, 194.87.0.*

# Еще один сервис
@SERVER:SQL             mysql_wsql
PRIORITY                10
LOG_FILE                /www/unis/wsql.log
LOG_LEVEL               2
SQL_SERVER              MYSQL
CONNECT_STRING          user/passwd
MAX_SERVERS             5
MIN_SERVERS             1
START_SERVERS           1
QUEUE_LEN               20
ALLOW_IP                127.0.0.1, 194.87.0.*

# Определение диспетчера и приемника для сервисов
# Совпадение имен говорит о том, что приемник отправляет поступившие
# запросы на обработку диспетчера Dispatcher for WSQL
@SERVER:DISPATCHER      dsp_wsql
PRIORITY                10
LOG_FILE                 /www/unis/wsql.log
LOG_LEVEL               2
MAX_SERVERS             6
MIN_SERVERS             4
START_SERVERS           4
QUEUE_LEN               128
# TCP Listener for WSQL
@LISTENER:TCP           dsp_wsql
PORT                    17000
```

API взаимодействия клиента и сервера состоит из двух типов вызовов: общие для всех сервисов и реализующих доступ к функциям конкретного сервиса. При использовании Unis в качестве менеджера запросов к СУБД, к общим вызовам относятся unis_connect и unis_disconnect, обеспечивающие соединение с сервреом и завершение работы, а к функциональным — набор вызовов обеспечивающих обработку SQL запросов к СУБД:

- unis_cur_open,
- unis_cur_close,
- unis_cur_bind,
- unis_cur_exec,
- unis_cur_fetch,
- unis_do_sql.

Ниже приведен фрагмент кода клиентской программы (php3 [2]), обеспечивающий доступ к информации из СУБД.

```
// открываем простейшее соединение с сервером
$conn=unis_connect(«host.domain.ru:17000:ora_wsql»);
if ($conn) {
     // открываем курсор с sql предложением
     $cur=unis_cur_open(«select id,name from names»,5,10);
     if ($cur) {
          // выбираем записи с 5 по 15
          while (unis_cur_fetch($cur,&$res))
               echo(«ID: $res[0]\nName: $res[1]\n»);
          // закрываем курсор
          unis_cur_close($cur);
     }
     // производим обновление
     if (unis_do_sql(«update counter set val=val+1»))
          echo(«Успешное обновление»);
     // закрываем соединение с сервером
     unis_disconnect($conn);
}
```

Заметим, что изменение имени сервиса ora_wsql на mysql_wsql приведет к тому, что работа будет осуществляться не с Oracle, а с MySQL без изменения всего остального кода — API клиента инвариантен относительно конкретной СУБД.

Наряду с предложенным API существуют и другие, например, широко используемый стандарт ODBC [3]. Разработка собственного API была обусловлена необходимостью максимально оптимизировать обработку запросов (в частности, минимизировать сетевой обмен) и предоставить разработчику приложений максимально простой интерфейс. Опыт эксплуатации систем, использующих Unis (например, Web-серверы Russia On the Net (http://www.ru), JOB.RU (http://www.job.ru), «Из рук в руки» (http://www.izrukvruki.ru)), показывает, что в большинстве случаев предложенной функциональности вполне достаточно для разработки приложений, однако с целью расширения области применимости универсального сервера, сейчас ведется работа по интеграции в Unis поддержки ODBC, JDBC [4] и механизмов управления распределенными транзакциями, соответствующего стандарту X/Open XA [5].

Дополнительные возможности

При обычном использовании Unis обеспечивает оптимальное распределение вычислительных ресурсов между собственными сервисами согласно алгоритмам, подробно описанным в [1]. Кроме того, предлагается несколько дополнительных возможностей, обеспечивающих управление качеством сервиса. Одной из таких возможностей является определение в конфигурационном файле наиболее приоритетных сервисов с гарантированным временем ответа. Следует заметить, что согласно теории массового обслуживания, методы которой используются в алгоритмах управления Unis, гарантировать время ответа можно только при достаточной вычислительной мощности, то есть всегда существует то предельное значение плотности входного потока запросов, до которого можно удержать время ответа в заданных границах. При превышении этого значения можно гарантировать время ответа, но при этом не гарантируется ответ как таковой (то есть система отвергает часть запросов без попытки их обработки). По умолчанию, Unis стремится минимизировать вероятность отказа системы, однако это можно изменить, добавив в конфигурацию сервиса строку:

```
MAX_RESPONSE_TIME N[%]
```

N — это либо максимально допустимое время ответа системы в секундах, либо (если указан %) — процентное соотношение времени ответа и времени обслуживания (время ответа = время обслуживания + время нахождения в очереди).

Другая дополнительная возможность Unis — балансировка нагрузки между сервисами. Эта возможность используется, как правило, при построении распределенных информационных систем, работающих с потоками запросов высокой плотности. Например, в конфигурации сервера определены два сервиса: dbs_1 и dbs_2, реализующие одинаковую функциональность, но использующие для этого, например, СУБД физически расположенные на разных машинах. Добавим в конфигурацию еще один псевдосервис dbs и направим весь входной поток запросов на него:

```
@ALIAS dbs
dbs_1 40
dbs_2 60
```

При этом, входной поток запросов будет распределяться между сервисами dbs_1 и dbs_2, причем на dbs_1 придется 40% нагрузки, а на dbs_2 — 60%. Следует отметить, что Unis, благодаря наличию информации о показателях качества сервисов, балансирует нагрузку, а не просто входной поток запросов. В реальной ситуации может получиться, что сервис dbs_2 обрабатывает один запрос из десяти, но при этом из-за меньшей имеющейся у него в наличии вычислительной мощности его загруженность будет больше, чем у dbs_1.

Особенности реализации

Сервер реализован на языке C++ и рассчитан на работу под управлением одного из диалектов ОС UNIX: Sun/Intel Solaris, HP UX, FreeBSD, Linux. Все основные компоненты системы имеют базовые классы. Управление компонентами и взаимодействие между ними осуществляется набором методов, присутствующих в базовых классах. Изначально система не обладает какой-либо функциональностью (нет определенных сервисов) и представляет собой лишь готовую для наполнения оболочку. Расширение функциональности производится за счет порождения на основе базового класса обработчика нового класса, наследующего все методы для интеграции с системой и имеющего новые, актуальные для его собственной внутренней логики, методы и структуры данных. Описания базового класса обработчика и порожденного на его основе класса, для обработки запросов в СУБД имеют следующий вид:

```
// Базовый класс
class Server {
    protected:
        Log *slog;                      // поддержка логов
        Queue *squeue;                  // очередь
    public:
        int err_code;                   // код последней ошибки
        int status;                     // статус
        char *ident_string;             // параметры инициализации
        pthread_mutex_t s_lock;         // lock для эксклюзивного использо-я
        virtual ~Server() {};
        virtual int start()=0;          // функция для запуска обработчика
        virtual void stop()=0;          // функция для остановки обработчика
        virtual void timeout_stop()=0;  // функция для немедленной остановки
};

// Порожденный класс для доступа к СУБД
```

```
class Server_SQL : public Server {
     private:
          SQL *sql_s;                    // поддержка SQL
          int make_replay();             // собственно обработка запросов
     public:
          Server_SQL(int, char *, Log *, Queue *, int);
          ~Server_SQL();
          int start();
          void stop();
          void timeout_stop();
};
```

Из приведенного описания классов видно, что внедрение в систему новых сервисов достаточно простая операция — необходимо лишь надлежащим образом оформить процедуры старт/стоп и собственно обработки запросов. Аналогичные базовые классы определены и для остальных компонентов системы.

Следует отметить, что внутренняя организация системы во многом соответствует технологиям «объектного мира», таким как CORBA [**6**] и EJB [**7**], а в некоторых случаях просто повторяет их, с точностью до синтаксиса определения объектов. Сейчас ведутся работы по более плотной интеграции с этими технологиями, в частности последняя версия Unis содержит диспетчер, позволяющий принимать запросы по протоколу IIOP [**6**] с некоторыми ограничениями наименование и идентификацию объектов.

Заключение

Универсальный сервер Unis изначально создавался для решения конкретных практических задач, стоявших при разработке высокопроизводительных информационных систем. Благодаря особенностям своей внутренней организации, система является открытой для расширения функциональности и в настоящий момент ведется активная работа по реализации модулей, обеспечивающих поддержку дополнительных протоколов и стандартов, что позволит расширить сферу применимости универсального сервера в уже существующих информационных системах. Основным достоинством системы можно считать наличие универсальной оболочки, обеспечивающей управление вычислительными ресурсами, внутрь которой встраивается функциональный модуль, оптимизированный под конкретную задачу. Именно это сочетание позволяет добиться максимальной производительности информационных систем даже на небольших вычислительных мощностях, на которых использование фирменного программного обеспечения, реализующего большую, но обычно частично востребованную в конкретной системе, функциональность, приводит к серьезной деградации производительности.

Unis уже более трех лет используется в качестве технологической основы разработки информационных систем в компании «Демос-Интернет», в частности, для обеспечения работы Web-серверов Russia On the Net (*http://www.ru*), JOB.RU (*http://www.job.ru*), «Из рук в руки» (*http://www.izrukvruki.ru*). Кроме того, Unis используется в корпоративных информационных системах таких организаций, как Российский Центр Испытаний и Сертификации (Ростест-Москва), РосНИИРОС (ИС Координационной Группы домена .RU *http://www.cgroup.ru*).

Дмитрий Волков — сотрудник компании «Демос-Интернет». С ним можно связаться по электронной почте: vdv@vdv.ru

Литература

[1] Волков Д. «Концепция универсального сервера для архитектуры клиент-сервер».

[2] «PHP3 Manual». PHP Documentation Group, 1998. http://www.php.net

[3] Robert Signore, John Creamer, Michael O. Stegman. « The Odbc Solution : Open Database Connectivity in Distributed Environments». McGraw Hill, 1995

[4] Graham Hamilton, Rick Cattell. «JDBC: A Java SQL API». «. Sun Microsystems, Inc., 1998

[5] «Distributed TP: The XA Specification». Open Group Technical Standard, 1992

[6] «The Common Object Request Broker: Architecture and Specification». Object Management Group Inc., 1999

[7] Vlada Matena, Mark Hapner. «Enterprise JavaBeans Specification, v1.1». Sun Microsystems, Inc., 1999