Российская академия наук
Сибирское отделение
Институт систем информатики
им. А. П. Ершова

В. С. Рыжов

НЕКОТОРЫЕ АСПЕКТЫ ПРОЕКТИРОВАНИЯ АРХИТЕКТУРЫ КРУПНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Препринт
106

Новосибирск 2003

В работе формулируются основные требования, которые должны учитываться при проектировании крупных информационных систем: развиваемость, масштабируемость и безопасность; рассматриваются проблемы, встающие перед разработчиками и администраторами при создании таких систем. Предлагаемое комплексное аппаратно-программное архитектурное решение, полученное на основе обобщения опыта разработки большого числа приложений, может использоваться в качестве рекомендаций разработчикам крупных систем.

Siberian Division of the Russian Academy of Sciences
A. P. Ershov Institute of Informatics Systems

Vladimir S. Ryzhov

SOME ASPECTS OF ARCHITECTURAL DESIGN FOR SCALABLE INFORMATION SYSTEMS

Preprint
106

Novosibirsk 2003

The requirements that should be taken into account while designing scalable information systems are considered. An integrated solution on hardware and software architecture is proposed. It has a number of advantages in the aspect of scalability, competitive access, security and continuous development of a live system. The solution is a result of summarizing the experience of development of a large number of information systems.

1. КРУПНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ: ЧТО СКРЫВАЕТСЯ ПОД ЭТИМ ПОНЯТИЕМ

В первую очередь следует договориться, что мы подразумеваем под крупными информационными системами (ИС). Крупными ИС мы будем называть системы, предоставляющие услуги большому количеству пользователей, т. е. обеспечивающие одновременный доступ примерно 100-200 пользователям и поддерживающие более 100 тысяч зарегистрированных пользователей, при этом средний объем данных каждого пользователя равен около 10Мб, т. е. суммарное количество хранимых данных более 1 терабайта (Тб). В качестве примера крупных систем можно привести такие информационные порталы, как Yahoo.com, Amazon.com, Yandex.ru.

Высокая стоимость современных ИС и их влияние на эффективность деятельности предприятий вынуждают вкладывать немалые средства в технологии, дающие уверенность в качестве конечного результата. Поэтому понятно желание найти средства, универсально решающие все или почти все проблемы построения крупных систем. К сожалению, таких средств не существует. Каждый метод и поддерживающие его инструментальные средства имеют свою область применения, свои недостатки и ограничения. Иными словами, предлагаемое в данной работе архитектурное решение призвано решить только те проблемы, на которых акцентируется внимание в этой статье.

Однако избежать внедрения новых средств не удастся. Чем крупнее и сложнее система, тем выше риск ее неудачного построения. Одним из способов снижения риска является формализация процесса построения систем, помогающая добиться успеха и качества. Излишне говорить о важности такого результата для компаний, деятельность которых сильно зависит от функционирования компьютерных систем. А чтобы освоение новых методов проходило эффективно, необходимо четко видеть их преимущества и недостатки.

Встречается как чрезмерно оптимистическое отношение к технологиям программной инженерии, так и их недооценка. В этой области существуют свои заблуждения и мифы: например, миф об абсолютном преимуществе дорогих интегрированных средств. Попробуем прояснить ситуацию.

2. ТРЕБОВАНИЯ К АРХИТЕКТУРЕ КРУПНЫХ СИСТЕМ

Тот факт, что услугами системы пользуются постоянно, требует от системы непрерывной бесперебойной работы. Этого можно достичь только путем создания распределенной системы с резервным копированием, необходимым для максимально быстрого восстановления в случае сбоев в работе аппаратной или программной частей. Поскольку на создание крупной системы затрачиваются значительные ресурсы, переписывание системы с нуля было бы невыгодным. Между тем, фантастический прогресс в этой сфере требует своевременного перехода на новые технологии. Следовательно, система должна предоставлять возможность внесения в нее изменений на протяжении всего периода ее существования.

В настоящее время уровень развития сетевых технологий предъявляет серьезные требования к безопасности хранимых данных, что заставляет использовать в системе новейшие технологии, отвечающие этим требованиям.

При проектировании подобных систем немаловажную роль играет и ценовой аспект, требующий от системы хорошей масштабируемости. Поскольку каждая система проходит определенный путь развития, в начале которого число пользователей системы невелико, экономически целесообразно не поддерживать 100 тысяч пользователей с самого начала, но заложить возможность постепенного наращивания ресурсов системы по мере необходимости.

2.1. Комплексный подход к программной и аппаратной частям крупных систем

При проектировании крупных систем недостаточно продумать только программную часть будущей системы. В отличие от малых систем, где проектирование зачастую сводится к продумыванию программной части и обеспечению ее переносимости на широкий круг платформ, проектирование крупных систем требует комплексного подхода как к программной, так и аппаратной частям. При хорошем взаимодействии между этими частями можно получить немалый выигрыш не только в производительности, но и в стоимости системы, а также заблаговременно решить некоторые проблемы, которые могут возникнуть по мере роста и развития системы.

2.2. Безопасность

Проблемы безопасности ИС возникли достаточно давно [1, 2, 3]. Формулируя коротко, система должна быть безопасной и гарантировать сохранность данных и их защиту от несанкционированного доступа. Сотни тысяч клиентов никогда не станут пользоваться Интернет-системой, если их личные данные могут стать достоянием общественности или быть утерянными в результате сбоя из-за плохо продуманной системы защиты информации.

2.2.1. Устойчивость к несанкционированному доступу

Эксперты в области компьютерной безопасности постоянно обнаруживают веб-сайты, на которых возможен несанкционированный доступ к номерам кредитных карт их клиентов, личным данным и другой информации, не подлежащей разглашению. Передавая информацию на сервера, пользователи сети Интернет рискуют, что кто-то воспользуется их кредитными картами или получит доступ к их конфиденциальным данным, причем даже без ведома владельцев серверов.

ИС должна обеспечивать соответствующую идентификацию и авторизацию клиентов. Кроме того, все транзакции должны производиться по каналам, исключающим возможность прослушивания.

2.2.2. Устойчивость к отказу аппаратуры

Аспект устойчивости к отказу оборудования особенно важен, если речь идет о крупных системах [4]. Должна быть предусмотрена возможность моментального переключения на резервные сервера в случае отказа аппаратуры. Следует учесть все: возможность пожара, затопления, отключения электричества в местах расположения серверов. Отсюда, как минимум, вытекает требование разнесения рабочего и резервного серверов на значительное расстояние, например, в разные страны, для обеспечения бессбойной работы сервера ИС даже в случае стихийного бедствия. Кроме того, архитектурное решение должно поддерживать периодическое копирование текущего состояния системы с целью мгновенного восстановления системы в случае отказа аппаратных средств.

2.3. Масштабируемость

Наиболее важным свойством крупных ИС является возможность непрерывного наращивания аппаратуры по мере их роста и развития. Следует

учитывать возрастание количества пользователей системы, увеличение размера пользовательской информации, а также рост числа конкурентных пользователей. При этом вследствие нелинейного и независимого роста перечисленных параметров и с учетом требования экономического характера необходима возможность независимого наращивания именно тех единиц оборудования, которые обеспечивали бы увеличение нужного параметра. Здесь также приходится принимать во внимание ценовые аспекты. Так, например, стоимость СУБД зависит от числа процессоров в машине, на которой установлена база данных. Отсюда немедленно следует решение — разнести на разные машины исполнение бизнес-логики, которое требует большой процессорной мощности, и хранение данных, требующее большого дискового пространства.

2.4. Развиваемость

Под развиваемостью мы будем понимать возможность производить изменения в уже работающей системе, функционирование которой не должно прекращаться ни на минуту. Эта характеристика очень важна для крупных ИС, если принимать во внимание стремительный прогресс в области Интернет-технологий. В статье В.Черняка, посвященной проблемам построения Интернет-порталов [5], жизненный цикл портала изображается следующей схемой.

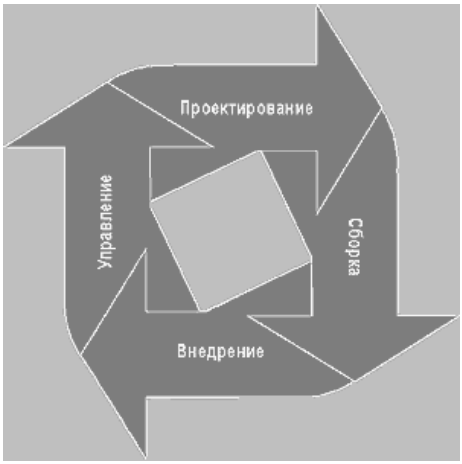

Рис.1. Жизненный цикл портала

Как видно из рис. 1, процесс Проектирование — Сборка — Внедрение — Управление представляет собой замкнутый цикл, повторяющийся на протяжении жизни портала (ИС).

Требование развиваемости системы ставит перед ее разработчиком следующие задачи:

- архитектурная поддержка перехода к новой версии без остановки работы уже работающей системы;
- конвертирование данных, имеющихся в старой базе данных, в новую структуру базы данных.

3. АРХИТЕКТУРНОЕ РЕШЕНИЕ

Нами предложено комплексное аппаратно-программное архитектурное решение, имеющее ряд преимуществ в отношении аспектов масштабируемости, конкурентного доступа, безопасности и непрерывного развития живой системы. Схема распределения компонентов ИС согласно предлагаемой архитектурной модели по серверам представлена на рис. 2.

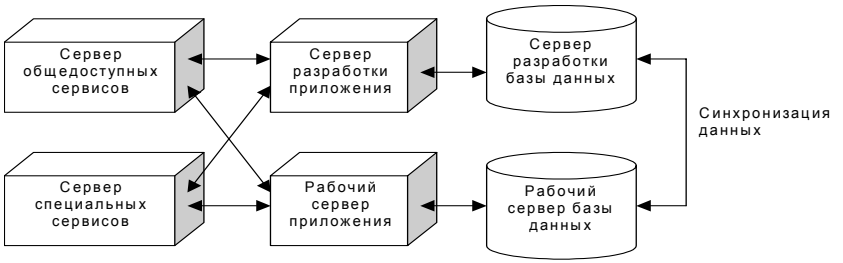

Рис.2. Архитектурное решение оборудования

3.1. Сервер общедоступных сервисов

Этот сервер предназначается для общедоступных сервисов таких, как веб-сервер, DNS-сервер или почтовый сервер. Это единственный элемент

системы, который является общедоступным из внешнего мира через Интернет. Следовательно, этот компьютер должен быть очень тщательно сконфигурирован. Сервер общедоступных сервисов не содержит никакой клиентской информации, на нем нет никакой обработки бизнес-логики, кроме трансляции HTML-файлов, подготовленных сервером приложений. Кроме того, этот сервер может исполнять роль DNS-сервера или почтового сервера. Фактически, этот компьютер служит firewall для остальных частей системы, которые связаны между собой в локальную сеть и невидимы снаружи. Таким образом, система располагается в отдельном Интранет-сегменте.

Сервер общедоступных сервисов критичен к доступу конкурентных пользователей. По мере роста числа пользователей растет и число одновременно работающих пользователей. Увеличение числа компьютеров с установленными на них веб-серверами обеспечивает поддержку масштабирования системы до требуемого уровня конкурентного доступа. Добавление веб-серверов в работающую систему может быть выполнено без прекращения работы системы. Распределение нагрузки на сервера можно реализовать с помощью динамической выдачи IP-адресов при трансляции запросов DNS-сервером, как это делается, например, на Microsoft.com. Различные веб-сервера желательно располагать на различных каналах, чтобы не было проблем с перегрузкой каналов.

3.2. Сервер специальных сервисов

Через этот сервер обеспечивается доступ сервисов третьих сторон, используемых приложением (таких, например, как электронные деньги). Кроме того, этот сервер обеспечивает доступ разработчиков к системе. Доступ разработчиков и третьих сторон разрешен только со строго ограниченного набора IP-адресов. Обмен информацией с программным обеспечением третьих сторон осуществляется с использованием протокола SSL или других криптографических алгоритмов во избежание прослушивания каналов связи с серверами третьих сторон. Эта часть системы очень существенна, поскольку именно эти каналы используется для передачи конфиденциальной информации пользователей (например, номеров кредитных карт) на сервера третьих сторон для проведения финансовых транзакций. Кроме того, через этот сервер может быть осуществлен доступ к базе данных и рабочим серверам, что также требует особой тщательности при конфигурации сервера, поскольку изнутри система не имеет мощной защиты от несанкционированного доступа.

3.3. Рабочий сервер приложения и сервер разработки приложения

На рабочем сервере приложения установлена действующая версия приложения. На сервере разработки приложения установлена версия приложения, находящаяся в стадии разработки. Когда возникает необходимость запуска новой версии приложения, эти сервера переключаются: сервер разработки становится действующим сервером, а действующий — сервером разработки. Для переключения системному администратору достаточно выполнить следующие действия (см. рис. 3):

1) перестроить конфигурацию сервера разработки в интересах соблюдения безопасности — остановить и деинсталлировать приложения и сервисы, использовавшиеся для разработки и не нужные для работы сервера;
2) произвести переключение серверов приложения, меняя местами внутренние IP-адреса сервера разработки приложения и рабочего сервера приложения.

В том случае, если в ходе разработки изменилась структура базы данных, требуется произвести также и переключение серверов баз данных. Для этого системный администратор выполняет следующие дополнительные действия:

3) переводит рабочую систему в режим “только чтение”;
4) транслирует данные из работающей базы в базу разработки;
5) производит переключение серверов баз данных, меняя местами внутренние IP-адреса сервера разработки базы данных и рабочего сервера базы данных.

Описанный процесс позволяет перейти к новой версии приложения на работающей системе, практически не прекращая ни на минуту ее работу.

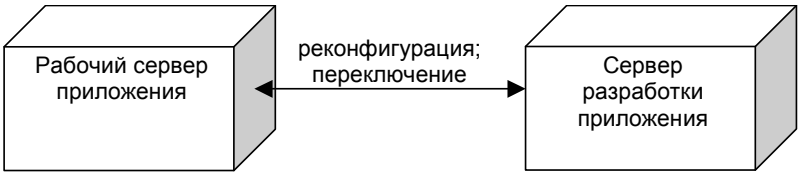

Рис. 3. Переключение серверов приложения

3.4. Рабочий сервер базы данных и сервер разработки базы данных

Рабочий сервер базы данных используется как сервер базы данных работающего приложения. Он содержит два экземпляра базы данных: первая — рабочая база данных, вторая — зеркальная копия разрабатываемой базы данных (рис. 4).

Сервер разработки базы данных используется для развития базы данных. Он также содержит два экземпляра базы данных: первая — база данных, находящаяся в стадии разработки; она может иметь структуру, отличную от структуры текущей базы данных; вторая — зеркальная копия рабочей базы данных (рис. 4).

Когда возникает необходимость запуска новой версии базы данных, разработчики системы запускают процесс конвертации данных из рабочей базы данных (с актуальными данными) в новую структуру базы данных. После того, как конвертация данных закончится, и новая база данных будет заполнена актуальными данными, администратор системы осуществит переключение, меняя местами внутренние IP-адреса. Это позволяет перейти к новой структуре базы данных на работающей системе, не прекращая функционирования системы.

Предложенная конфигурация позволит также использовать зеркальную копию рабочей базы данных на сервере разработки в качестве резервной копии в случае сбоя программного обеспечения или аппаратуры. Если происходит такой сбой, администратор просто меняет местами рабочий сервер и сервер разработки, запустив при этом резервную копию приложения на сервере разработки и сменив его IP-адрес на IP-адрес рабочего сервера. Таким образом, сервер разработки становится рабочим сервером.

Периодическое зеркальное копирование осуществляется средствами используемой СУБД в соответствии с настройками. При использовании хорошей СУБД это самый удобный и быстрый способ создания резервной копии.

3.5. Синхронизация данных

Жизненный цикл системы представлен на рис. 4. Регулярное зеркальное копирование обеспечивает наличие точной копии рабочей базы данных с актуальными данными (B) на сервере разработки базы данных. Когда готова новая версия структуры базы данных (C), начинается процесс конвертации данных из (B) в (C). Конвертация данных выполняется с целью синхронизации данных на рабочем сервере и сервере разработки. После того как процесс синхронизации данных закончен, свежая версия базы данных заполняется актуальными данными. Администратор системы осуществляет переключение, меняя местами сервера баз данных: рабочий сервер базы данных становится сервером разработки, а сервер разработки базы данных становится рабочим сервером.

Когда потребуется запустить новую версию базы данных в следующий раз, производится конвертация данных из базы данных (D), являющейся зеркальной копией рабочей базы данных, в базу данных (A), которая будет являться разрабатываемой базой данных.

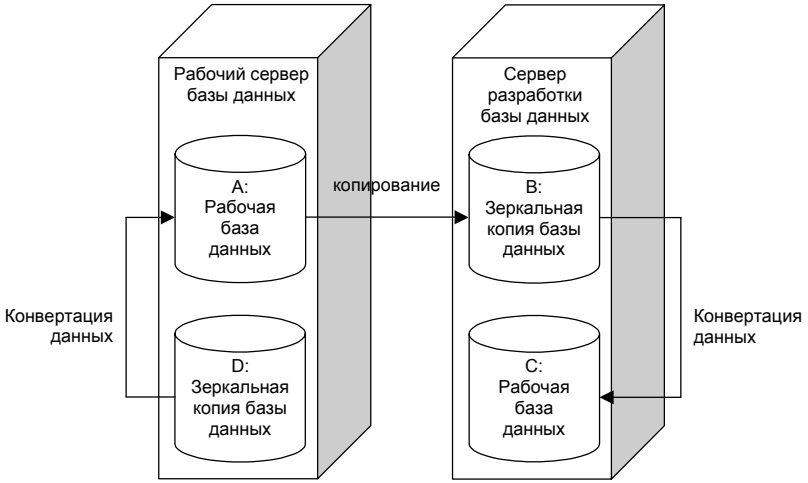

Рис. 4. Жизненный цикл системы

3.6. Универсальная конвертация данных

В процессе разработки приложения структура базы данных может изменяться — появляются новые таблицы и поля, изменяются типы полей и т.д. Но в работающей системе зарегистрировано множество пользователей, и работающая база данных заполнена данными, которые нельзя потерять. Возникает проблема конвертации актуальных данных в новую структуру базы данных.

Был реализован метод универсальной конвертации данных путем трансляции данных в XML. Процесс конвертации данных представлен на рис. 5 и заключается в следующем: данные конвертируются в XML (1). Затем полученный XML-код конвертируется в новый XML-код с использованием специально созданного XSL-модуля, в котором содержится бизнес-логика конвертации. Полученный XML (2) затем транслируется в данные новой базы данных.

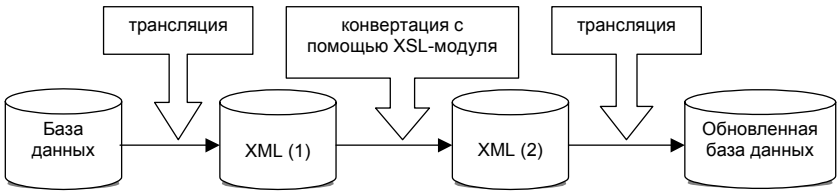

Рис. 5. Процесс конвертации данных

4. БЕЗОПАСНОСТЬ

Рассмотрим предложенную модель архитектуры с точки зрения безопасности. С учетом аспектов безопасности схема, изображенная на рис.2, будет иметь вид, показанный на рис. 6.

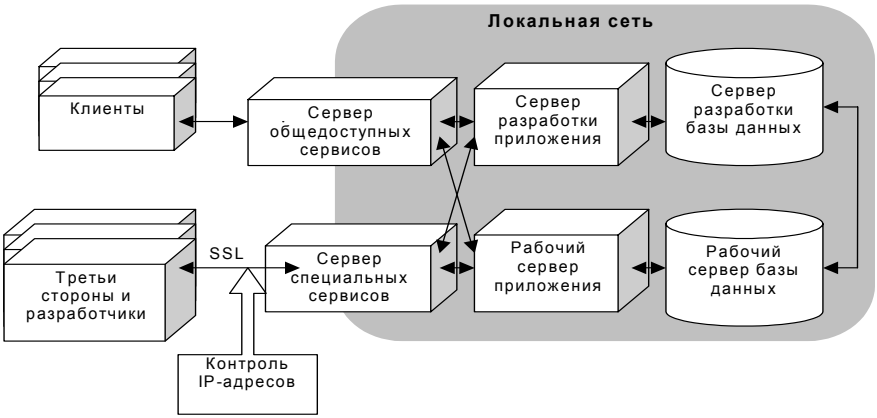

Рис. 6. Модель архитектуры с точки зрения безопасности

4.1. Устойчивость к несанкционированному доступу

4.1.1. Идентификация и авторизация

В рамках предложенной модели будут рассматриваться парольно-защищенные системы. Пользователи системы открывают сессию с помощью процедуры входа в систему, которая требует передачи пароля для аутентификации пользователя. Предполагается, что на протяжении всей сессии только авторизованное лицо взаимодействует с системой. Чтобы завершить сессию, пользователь может воспользоваться процедурой выхода из системы, но если пользователь не взаимодействует с системой в течение определенного времени (которое задается настройками системы), сессия закрывается автоматически, и чтобы продолжить работу с системой, пользователь должен снова пройти процедуру входа в систему для открытия новой сессии. Это делается для того, чтобы уменьшить вероятность несанкционированного доступа с правами пользователя той машины, на которой открыта сессия, в случае, если пользователь забыл выполнить процедуру выхода из системы.

Во время осуществления связи между сервером специальных сервисов и серверами третьих сторон (или серверами разработчиков) для идентификации используются IP-адреса этих внешних серверов. Любое внешнее под-

ключение, за исключением компьютеров, имеющих IP-адреса из фиксированного набора, запрещено на уровне ядра (рис. 6).

4.1.2. Безопасность каналов передачи данных

Во избежание прослушивания каналов во время передачи таких важных данных, как пароль, для передачи используется протокол SSL (рис. 6). Этот протокол используется при всех подключениях к серверу специальных сервисов. При подключении к серверу общедоступных сервисов протокол SSL используется в случаях, когда предполагается передача конфиденциальной пользовательской информации.

4.1.3. Сессия и объект сессии

Поскольку использование cookie-файлов для хранения данных сессии недопустимо с точки зрения безопасности, от него пришлось отказаться. Для этих целей был реализован собственный объект сессии, который основан на передаче со всеми интерактивными страницами уникального ключа, который передается обратно на сервер при всех запросах.

4.2. Устойчивость к сбою аппаратуры

Предложенная модель архитектуры имеет высокий уровень надежности в отношении сбоя аппаратуры. Непрерывное зеркальное копирование работающей базы данных обеспечивает наличие точной копии базы данных с актуальными данными на сервере разработки (рис. 4). Если произошел сбой оборудования, имеется возможность немедленного переключения на сервер разработки. Такая модель архитектуры позволяет разнести сервер разработки и рабочий сервер на достаточное расстояние для обеспечения непрерывной работы даже в случае стихийного бедствия.

5. МАСШТАБИРУЕМОСТЬ

Предложенная модель архитектуры обеспечивает высокий уровень масштабируемости системы на разных уровнях благодаря гибкости расположения программного обеспечения. Относительно независимые части системы позволяют наращивать аппаратную часть там, где возникают, по мере эксплуатации системы, узкие места. При появлении симптомов нехватки ресурсов в каком-либо из узлов, можно сразу же нарастить их именно в этом узле.

5.1. Конкурентный доступ

По мере роста числа пользователей, осуществляющих одновременный доступ к системе, возрастает нагрузка на веб-сервер. Когда система начинает систематически исчерпывать возможность новых подключений, к ней могут быть добавлены новые компьютеры с инсталлированными на них веб-серверами. Все компьютеры продолжают оставаться в локальной сети, но могут быть размещены в местах, значительно удаленных друг от друга. Предпочтительно добавлять новые веб-сервера на другие, ранее не использовавшиеся каналы, чтобы не столкнуться с проблемой заполнения канала. Конечно, если ширина канала позволяет, можно добавлять несколько веб-серверов на один канал. В этом случае схема архитектуры будет такой, как показано на рис. 7.

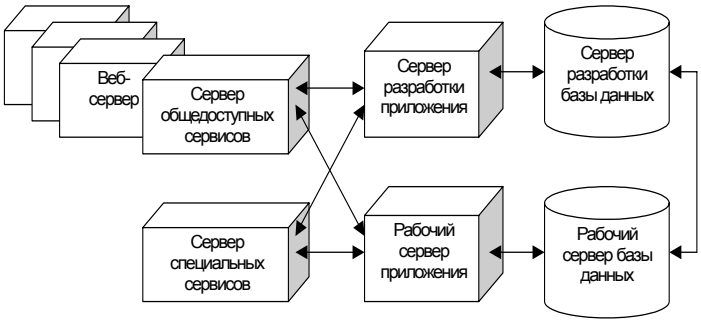

Рис. 7. Повышение числа конкурентных пользователей

5.2. Масштабируемость на уровне базы данных

С ростом числа зарегистрированных пользователей системы возрастает объем хранимых данных. Предложенная архитектурная модель предоставляет возможность добавлять дисковое пространство к серверу базы данных (рис. 8). При этом в случае нехватки процессорных ресурсов для обработки бизнес-логики на рабочем сервере приложения можно добавить процессоры именно в этот сервер, не трогая сервер базы данных. Такой подход позволяет существенно сэкономить на стоимости СУБД, поскольку ее стоимость в значительной мере зависит от количества процессоров.

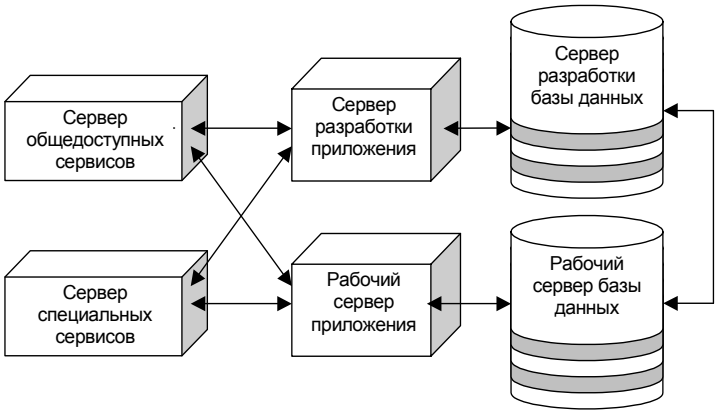

Рис. 8. Наращивание мощности сервера базы данных

5.3. Масштабируемость на уровне приложения

Предложенное архитектурное решение дает возможность добавлять в систему новые процессоры (или использовать кластеры), чтобы усилить мощность той части, что отвечает за исполнение бизнес логики (рис. 9). Поскольку сервер приложения и сервер базы данных разнесены на разные компьютеры, добавление новых процессоров никак не скажется на стоимости используемой СУБД.

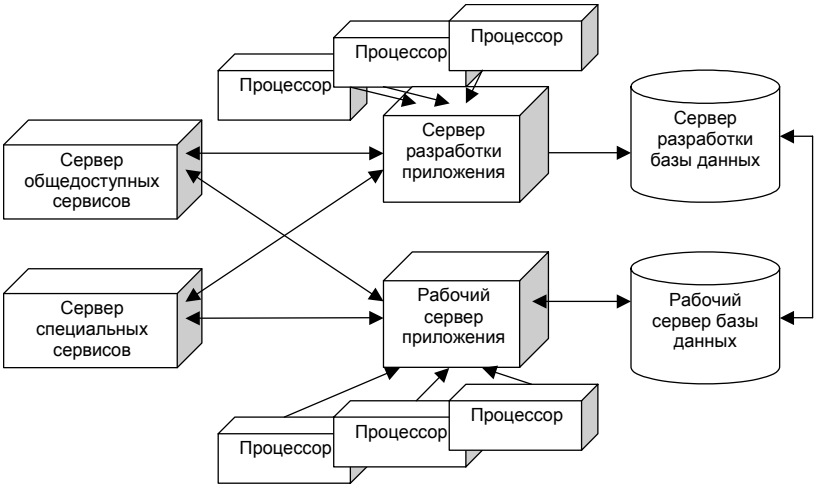

Рис. 9. Наращивание мощности сервера приложения

6. ЗАКЛЮЧЕНИЕ

Сравнивая предложенную архитектурную модель с теми наработками, которые сейчас широко используются в мире, можно выделить несколько характеризующих ее отличительных моментов.

В отличие от архитектуры ИС, используемой порталом Yahoo.com, в предлагаемой модели система располагается целиком в одном отдельном Интранет-сегменте. Это способствует цельности системы, что позволяет избежать ситуаций, наблюдаемых временами на Yahoo.com, когда некоторая часть системы недоступна по той или иной причине. Архитектура Yahoo построена на базе большого количества разнородных блоков, находящихся на отдельных аппаратных комплексах. По этой причине обновление версий происходит не поэтапно, а стохастически. Порой в обновленной версии можно встретить куски со старым внешним видом, либо плохо совместимые с новой версией.

Архитектура портала HotMail.com базируется на основе одного мощного компьютера и не имеет активной копии. Это приводит в случае сбоев к долговременным остановкам и частичной потере данных из-за того, что данные восстанавливаются из резервной копии, получаемой периодически.

После сбоя производится исправление ошибки, приведшей к сбою, и только после этого возобновляется работа портала.

Система, на которой базируется портал Amazon.com, имеет активную копию, но процесс обновления версии приложения является весьма дорогостоящей операцией, поскольку для новой версии всегда собирается новый аппаратный комплекс, на котором и происходит подготовка новой версии. Когда версия готова, производится переключение IP-адресов на новый аппаратный комплекс. Следует признать, что такой подход имеет и положительный момент: аппаратура всегда соответствует реалиям времени и позволяет программистам ориентироваться на новые достижения в области аппаратного обеспечения. Но очевидно, что при этом подходе становятся невозможными частые оперативные обновления системы, что бывает особенно важно на ранних стадиях развития ИС.

Подводя итоги, можно сделать вывод, что данная архитектурная модель ориентирована прежде всего на возможность легкого развития системы. Наличие двух параллельных структур аппаратной части системы позволяет быстро и без серьезных проблем переходить к новой версии системы, когда это потребуется. Такой переход может быть выполнен путем переключения с сервера разработки на рабочий сервер и наоборот, что производится без остановки работы системы. Аппаратное разнесение отдельных сегментов системы таких, как база данных, сервер приложения, сервер специальных сервисов, сервер общедоступных сервисов, дает возможность гибкого наращивания ресурсов в сегменте, в котором ощущается их нехватка. Кроме того, разделение серверов специальных и общедоступных сервисов позволяет осуществлять дополнительные меры для повышения безопасности системы. Использование резервного копирования на два различных носителя увеличивает устойчивость системы к потере данных.

СПИСОК ЛИТЕРАТУРЫ

1. **The Generally** Accepted System Security Principles (GSSP). Exposure Draft. — GSSP Draft Sub-committee, 1994.
2. **Гайкович В., Першин А.** Безопасность электронных банковских систем. — М.: Единая Европа, 1994.
3. **Левин В.К.** Защита информации в информационно-вычислительных системах и сетях // Программирование. — 1994. — С. 5–16.
4. **Галатенко В.** Информационная безопасность — обзор основных положений. // Открытые системы. — 1996. — № 3 (17). — С. 42–45.

5. **Черняк Л.** Порталы и жизненные циклы // Открытые системы. — 2002. — № 2. В Интернет: http://www.osp.ru/os/2002/02/060.htm.

Рыжов В. С.

НЕКОТОРЫЕ АСПЕКТЫ ПРОЕКТИРОВАНИЯ АРХИТЕКТУРЫ КРУПНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Препринт
106

Рукопись поступила в редакцию 06.02.03
Рецензент А. В. Быстров
Редактор З. В. Скок

Подписано в печать 01.04.03
Формат бумаги 60 × 84 1/16 Объем 1.1 уч.-изд.л., 1.3 п.л.
Тираж 50 экз.

НФ ООО ИПО “Эмари” РИЦ, 630090, г. Новосибирск, пр. Акад. Лаврентьева, 6