#### 2.1.3. Язык манипуляции данными (ЯМД)

*Язык манипуляции данными (ЯМД*) обеспечивает эффективные команды манипуляции сетевой системой базы данных. ЯМД позволяет пользователям выполнять над базой данных операции в целях получения информации, создания отчетов, а также обновления и изменения содержимого записей.

Основные команды ЯМД можно классифицировать следующим образом: команды передвижения, команды извлечения, команды обновления записей, команды обновления наборов.

Табл.2. Основные типы команд ЯМД.

|  |  |  |
| --- | --- | --- |
| № | Наименование типа команд | Назначение |
| *1* | *Команды передвижения.* | Команды, применяемые для поиска записей базы данных. |
| *2* | *Команды извлечения.* | Команды, применяемые для извлечения записей базы данных. |
| *3* | *Команды обновления записей.* | Команды, применяемые для изменения значений записей. |
| *4* | *Команды обновления наборов.* | Команды, применяемые для добавления, изменения или удаления экземпляров наборов. |

**Заключение**

Процесс преобразования функциональной модели в реляционную включает создание реляционной таблицы для каждого объектного множества модели. Атрибуты объектного множества становятся атрибутами реляционной таблицы. Если в функциональной модели существует ключевой атрибут, то он может использоваться в качестве ключа реляционной таблицы. В противном случае ключевой атрибут таблицы может быть создан аналитиком. Однако, лучше всего, если такой атрибут естественным образом возникает из моделируемого приложения. Отношения один-к-одному и один-ко-многим преобразуются в реляционную модель путем превращения их в атрибуты соответствующей таблицы. Отношения многие-ко-многим соответствуют многозначным атрибутам и преобразуются в четвертую нормальную форму путем создания ключа из двух столбцов, соответствующих ключам двух объектных множеств, участвующих в отношении. Конкретизирующие множества преобразуются путем создания отдельных реляционных таблиц, ключи которых совпадают с ключами обобщающих объектных множеств. Рекурсивные отношения также можно смоделировать, создав новое смысловое имя атрибута, обозначающее отношение.

#### 2.2. Архитектуры реализации корпоративных информационных систем.

При построении корпоративных информационных сетей, как правило, используются две базовые архитектуры: Клиент-сервер и Интернет/Интранет. В чем же преимущества и недостатки использования каждой из данных архитектур и когда их применение оправдано? Найти ответы на эти вопросы мы постараемся в данном разделе.

Одной из самых распространенных на сегодня архитектур построения корпоративных информационных систем является архитектура КЛИЕНТ-СЕРВЕР.

В реализованной по данной архитектуре информационной сети клиенту предоставлен широкий спектр приложений и инструментов разработки, которые ориентированы на максимальное использование вычислительных возможностей клиентских рабочих мест, используя ресурсы сервера в основном для хранения и обмена документами, а также для выхода во внешнюю среду. Для тех программных систем, которые имеют разделение на клиентскую и серверную части, применение данной архитектуры позволяет лучше защитить серверную часть приложений, при этом, предоставляя возможность приложениям либо непосредственно адресоваться к другим серверным приложениям, либо маршрутизировать запросы к ним. Средством (инструментарием) для реализации клиентских модулей для ОС Windows в данном случае является, как правило, Delpfi.

Однако при этом частые обращения клиента к серверу снижают производительность работы сети, кроме этого приходится решать вопросы безопасной работы в сети, так как приложения и данные распределены между различными клиентами. Распределенный характер построения системы обуславливает сложность ее настройки и сопровождения. Чем сложнее структура сети, построенной по архитектуре КЛИЕНТ-СЕРВЕР, тем выше вероятность отказа любого из ее компонентов.

В последнее время все большее развитие получает архитектура Интернет/Интранет. В основе реализации корпоративных информационных систем на базе данной архитектуры лежит принцип "открытой архитектуры", что во многом определяет независимость реализации корпоративной системы от конкретного производителя. Все программное обеспечение таких систем реализуется в виде аплетов или сервлетов (программ написанных на языке JAVA) или в виде cgi модулей (программ написанных как правило на Perl или С).

Основными экономическими преимуществами данной архитектуры являются:

* относительно низкие затраты на внедрение и эксплуатацию;
* высокая способность к интеграции существующих гетерогенных информационных ресурсов корпораций;
* повышение уровня эффективности использования оборудования (сохранение инвестиций).
* прикладные программные средства доступны с любого рабочего места, имеющего соответствующие права доступа;
* минимальный состав программно-технических средств на клиентском рабочем месте (теоретически необходима лишь программа просмотра - броузер и общесистемное ПО);
* минимальные затраты на настройку и сопровождение клиентских рабочих мест, что позволяет реализовывать системы с тысячами пользователей (причем многие из которых могут работать за удаленными терминалами).

В общем случае АИС, реализованная с использованием данной архитектуры включает Web-узлы с интерактивным информационным наполнением, реализованных при помощи технологий Java, JavaBeans и JavaScript, взаимодействующих с предметной базой данных, с одной стороны, и с клиентским местом с другой. База данных, в свою очередь, является источником информации для интерактивных приложений реального времени.

По запросу клиента WEB узел осуществляет следующие операции (рис.7):

* Отправляет ASCII коды HTML страниц (или VRML документов), включающие при необходимости элементы javaScript;
* Отсылает двоичный код запрошенного ресурса (изображения, адио-, видеофайла, архива и т.п.);
* Отсылает байт коды JAVA апплетов.
* Принимает конкретную информацию от пользователя (результат заполнения активной формы, или статистическую информацию запрошенную CGI скриптом);
* Осуществляет заполнение базы данных;
* Принимает сообщения от пользователя и регламентирует доступ к ресурсам Web узла на основе анализа принятой информации (проверка паролей и т.п.);
* Принимает информацию от пользователя и в зависимости от нее динамически формирует HTML страницы, либо VRML документы, обращаясь, при необходимости, к базам данных и существующим на WEB узле HTML страницам и VRML документам.

После того, как клиент получил ответ WEB сервера, он осуществляет следующие операции:

* визуализирует HTML страницу либо VRML документ в окне броузера;
* интерпретирует команды JavaScript, модифицирует образ HTML страницы и т.п.;
* интерпретируя байт коды JAVA апплетов, позволяет загружать и выполнять активные приложения;
* ведет диалог с пользователем, заполняющим формы, и создает новые запросы к WEB серверу;
* с помощью утилит воспроизводит коды аудио и видео файлов, поддерживает мультимедийные средства;
* обеспечивает моделирование виртуальной реальности просматривая VRML документы.

Перечисленные задачи WEB клиента обеспечиваются возможностями броузера и специализированным программным обеспечением (утилитами), размещенными на рабочей станции клиента. Следует отметить и тот факт, что жестких стандартов на построение WEB клиента пока нет и его компонентный состав может различаться.

На сегодняшний день известны и широко применяются три основных технологии создания интерактивного взаимодействия с пользователем в Web. Первый путь заключается в использовании Стандартного Интерфейса Шлюза (Commom Gateway Interface) - CGI. Второй - включение JavaScript - сценариев в тело Web-страниц. И наконец самый мощный, предоставляющий практически неограниченные возможности способ - применение технологии Java (ипользование Java-аплетов).

**CGI** - это механизм для выбора, обработки и форматирования информации. Возможность взаимодействия, обеспечиваемая CGI, предоставляется во многих формах, но в основном это динамический доступ к информации, содержащейся в базах данных. Например, многие узлы применяют CGI для того, чтобы пользователи могли запрашивать базы данных и получать ответы в виде динамически сформированных Web-страниц (рис.17).

Имеются в виду узлы, предоставляющие доступ к базам данных, средствам поиска, и даже информационные системы, предающие сообщения в ответ на ввод пользователя. Все эти узлы используют CGI, чтобы принять ввод пользователя и передать его с сервера Web базе данных. База данных обрабатывает запрос и возвращает ответ серверу, который в свою очередь пересылает его опять броузеру для отображения. Без СGI база данных этого не смогла бы. Данный интерфейс можно считать посредником между броузером, сервером и любой информацией которая должна передаваться между ними.

В отличии от HTML, CGI не является языком описания документов. Собственно, это и не язык вообще; это стандарт. Он просто определяет, как серверы Web передают информацию, используя приложения, исполняемые на сервере. Это способ расширения возможностей сервера Web без преобразования при этом его самого. Подобно тому как броузер Web обращается к вспомогательным приложениям для обработки информации, которую он не понимает, CGI предоставляет серверу Web возможность преложить работу на другие приложения, такие как базы данных и средства поиска.

При написании программы шлюза (которая может конвертировать ввод из одной системы в другую) CGI позволяет использовать почти любой язык программирования. Способность использовать при написании программы шлюза любой язык, даже язык сценариев, чрезвычайно важна. Самыми популярными языками являются shell, Perl, C и С++. Сценарием традиционно называют программу, которая выполняется с помощью интерпретатора, выполняющего каждую строку программы по мере ее считывания.

Последовательность действий при взаимодействии клиента с программой запущенной на Web-сервере можно сформулировать как следующая последовательность шагов.

1. Броузер принимает введенную пользователем информацию, как правило с помощью форы.
2. Броузер помещает введенную пользователем информацию в URL, указывающий имя и местоположение сценария CGI, который требуется ввести в действие.
3. Броузер подключается к серверу Web и запрашивает URL. Сервер определяет, что URL должен ввести в действие сценарий CGI, и запускает указанный сценарий.
4. Сценарий CGI выполняется, обрабатывая все передаваемые ему данные.
5. Сценарий CGI динамически формирует Web-страницу и возвращает результат серверу.
6. Сервер возвращает результат клиенту.
7. Броузер отображает результат пользователю

Это является упрощенной схемой взаимодействия между броузером, сервером и сценарием CGI. Наибольшую популярность CGI - сценарии нашли при использовании в качестве обработчиков форм, средства доступа к базам данных, средства осуществления локального и глобального поиска, шлюзовых протоколов.

Наибольшей мощью в реализации клиентского программного обеспечения обладают аплеты - программы написанные на языке JAVA. В узком смысле слова Java - это объектно-ориентированный язык, напоминающий C++, но более простой для освоения и использования. В более широком смысле Java - это целая технология программирования, изначально рассчитанная на интеграцию с Web-сервисом, то есть на использование в сетевой среде. Поскольку Web-навигаторы существуют практически для всех аппаратно-программных платформ, Java-среда должна быть как можно более мобильной, в идеале полностью независимой от платформы.

С целью решения перечисленных проблем были приняты, помимо интеграции с Web-навигатором, два других важнейших постулата.

1. Была специфицирована виртуальная Java-машина (JVM), на которой должны выполняться (интерпретироваться) Java-программы. Определены архитектура, представление элементов данных и система команд Java-машины. Исходные Java-тексты транслируются в коды этой машины. Тем самым, при появлении новой аппаратно-программной платформы в портировании будет нуждаться только Java-машина; все программы, написанные на Java, пойдут без изменений.
2. Определено, что при редактировании внешних связей Java-программы и при работе Web-навигатора прозрачным для пользователя образом может осуществляться поиск необходимых объектов не только на локальной машине, но и на других компьютерах, доступных по сети (в частности, на WWW-сервере). Найденные объекты загружаются, а их методы выполняются затем на машине пользователя.

Несомненно, между двумя сформулированными положениями существует тесная связь. В компилируемой среде трудно абстрагироваться от аппаратных особенностей компьютера, как трудно (хотя и можно) реализовать прозрачную динамическую загрузку по сети. С другой стороны, прием объектов извне требует повышенной осторожности при работе с ними, а, значит, и со всеми Java-программами. Принимать необходимые меры безопасности проще всего в интерпретируемой, а не компилируемой среде. Вообще, мобильность, динамизм и безопасность - спутники интерпретатора, а не компилятора.

Принятые решения делают Java-среду идеальным средством разработки интерактивных клиентских компонентов (апплетов) Web-систем. Особо отметим прозрачную для пользователя динамическую загрузку объектов по сети. Из этого вытекает такое важнейшее достоинство, как нулевая стоимость администрирования клиентских систем, написанных на Java. Достаточно обновить версию объекта на сервере, после чего клиент автоматически получит именно ее, а не старый вариант. Без этого реальная работа с развитой сетевой инфраструктурой практически невозможна.

Стандартный реляционный доступ к данным очень важен для программ на Java, потому что Java-апплеты по природе своей не являются монолитными, самодостаточными программами. Будучи модульными, апплеты должны получать информацию из хранилищ данных, обрабатывать ее и записывать обратно для последующей обработки другими апплетами. Монолитные программы могут себе позволить иметь собственные схемы обработки данных, но Java-апплеты, пересекающие границы операционных систем и компьютерных сетей, нуждаются в опубликовании открытых схем доступа к данным.

Интерфейс JDBC (Java Database Connectivity - связанность баз данных Java) является первой попыткой реализации доступа к данным из программ Java, не зависящего от платформы и базы данных. В версии JDK 1.1 JDBC является составной частью основного Java API.

JDBC - это набор реляционных объектов и методов взаимодействия с источниками данных. Программа на языке Java открывает связь с таблицей, создает объект оператор, передает через него операторы SQL системе управления базой данных получает результаты и служебную информацию о них. В типичном случае файлы .class JDBC и апплет/приложение на языке Java находятся на компьютере клиенте. Хотя они могут быть загружены из сети, для минимизации задержек во время выполнения лучше иметь классы JDBC у клиента. Система управления базой данных (CУБД) и источник данных обычно расположены на удаленном сервере.

На рисунке 19 показаны различные варианты реализаций связи JDBC с базой данных. Апплет/приложение взаимодействует с JDBC в системе клиента, драйвер отвечает за обмен информацией с базой данных через сеть.

Классы JDBC находятся в пакете java.sql.\*. Все программы Java используют объекты и методы из этого пакета для чтения и записи в источник данных. Программе, использующей JDBC, требуется драйвер к источнику данных, с которым она будет взаимодействовать. Этот драйвер может быть написан на другом (не Java) языке программирования, или он может являться программой на языке Java, которая общается с сервером, используя RPC (Remote Procedure Call) - удаленный вызов процедур или HTTP. Обе схемы приведены на рис.19. Драйвер JDBC может быть библиотекой на другом (не Java), как программа сопряжения ODBC - JDBC, или классом Java, который общается через сеть с сервером базы данных, используя RPC или HTTP.

Допускается, что приложение будет иметь дело с несколькими источниками данных, возможно, с неоднородными. По этой причине у JDBC есть диспетчер драйверов, чьи обязанность заключаются в управлении драйверами и предоставлении программе списка загруженных драйверов.

Хотя словосочетание "База данных" входит в расшифровку аббревиатуры JDBC, форма, содержание и расположение данных не интересуют программу Java, использующую JDBC, поскольку существует драйвер к этим данным.

**Сопряжение JDBC - ODBC**

В качестве составной части JDBC поставляется драйвер для доступа из JDBC к источникам данных ODBC (Open Database Connectivity), и называется "программа сопряжения JDBC - ODBC". Эта программа сопряжения реализована в виде JdbcOdbc.class и является библиотекой для доступа к драйверу ODBC.

Поскольку JDBC конструктивно близок к ODBC, программа сопряжения является несложной надстройкой над JDBC. На внутреннем уровне этот драйвер отображает методы Java в вызовы ODBC и тем самым взаимодействует с любым ODBC - драйвером. Достоинство такой программы сопряжения состоит в том, что JDBC имеет доступ к любым базам данных, поскольку ODBC - драйверы распространены очень широко.

В соответствии с правилами Internet JDBC идентифицирует базу данных при помощи URL, который имеет форму:

*jdbc:<субпротокол>:<имя, связанное с СУБД или Протоколом>*

У баз данных в Internet/intranet "имя" может содержать сетевой URL

*//<имя хоста>:<порт>/..*

<*субпротокол*> может быть любым именем, которое понимает база данных. Имя субпротокола "odbc" зарезервированно для источников данных формата ODBC. Типичный JDBC URL для базы данных ODBC выглядит следующим образом:

*jdbc:odbc:<DNS - имя ODBC>;User=<имя пользователя>; PW=<пароль>*

**Внутреннее устройство JDBC - приложения**

Чтобы обработать информацию из базы данных, информационно-обучающая система на языке Java выполняет ряд шагов. На рис.20 показаны основные объекты JDBC, методы и последовательность выполнения, Во-первых, программа вызывает метод *getConnection (),* чтобы получить объект *Connection.*Затем она создает объект *Statement* и подготавливает оператор SQL.

Оператор SQL может быть выполнен немедленно (объект *Statement*), а может быть откомпилирован (объект *PreparedStatement*) или представлен в виде вызова процедуры (объект *CallableStatement*). Когда выполняется метод *executeQuery(),* возвращается объект *ResultSet*. Операторы SQL, такие как updatе или delete не возвращают *ResultSet*. Для таких операторов используется метод *executeUpdate().* Он возвращает целое, указывающее количество рядов, затронутых оператором SQL.

*ResultSet* содержит ряды данных и анализируетcя методом *next()*. Если приложение обрабатывает транзакции, можно пользоваться методами *rollback()* и *commit()* для отмены или подтверждения изменений, внесенных оператором

SQL.

**Примеры запроса и модификации базы данных с использованием JDBC**

Данный пример иллюстрирует как при помощи SQL - опрератора SELECT составляется список всех студентов из базы данных. Ниже приводятся шаги, которые необходимы для выполнения этого задания при помощи API JDBC. Каждый шаг имеет форму текста на языке Java с комментариями.

// описать методы и переменные

public void ListStudents () throws SQLException

{

int i, noOfColumns;

String stNo, stFName, stLName;

// инициализировать и загрузить драйвер JDBC-ODBC

Class.forName ("jdbc.odbc.JdbcOdbcDriver");

// создать объект Connection

Connection ex1Con = DriverManager.getConnection (

"jdbc:odbc:StudentDB;uid="admin";pw="sa"");

// создать простой объект Statement

Statement ex1Stmt = ex1Con.createStatement ();

// Создать строку SQL, передать ее СУБД и

// выполнить SQL-оператор

ResultSet ex1rs = ex1Stmt.executeQuery (

"SELECT StudentNumber, FirstName, LastName FROM Students");

// Обработать каждый ряд и вывести результат на консоль

System.out.println ("Student Number First Name Last Name");

while (ex1rs.next())

{

stNo = ex1rs.getString (1);

stFName = ex1rs.getString (2);

stLName = ex1rs.getString (3);

System.out.println (stNo, stFName, stLName);

}

}

В следующем примере поле firstName таблицы Students изменяется. Доступ осуществляется через поле StudentNumber.

*// описать методы, переменные и параметры*

public void UpdateStudentName (String stFName, String stLName, String stNo)

throws SQLException

{

int retValue;

//инициализировать и загрузить драйвер JDBC-ODBC

Class.forName ("jdbc.odbc.JdbcOdbcDriver");

// создать объект Connection

Connection ex1Con = DriverManager.getConnection (

"jdbc:odbc:StudentDB;uid="admin";pw="sa"");

// создать простой объект Statement

Statement ex1Stmt = ex1Con.createStatement ();

// Создать строку SQL, передать ее СУБД и

// выполнить SQL-оператор

String SQLBuffer = "UPDATE Students SET FirstName =" +

stFName + ", lastName =" + stLName +

"WHERE StudentNumber = " + stNo;

retValue = ex1Stmt.executeUpdate (SQLBuffer);

System.out.println ("Модифицированно " + retValue +

" строк в базе данных.")

}

Рис.21. Интерфейс для регистрации пользователя в АИС.

Таким образом, взаимодействие с базами данных из Java также отличается простотой и гибкостью, связанной с эффективной реализацией JDBC API. В сочетании со своей природной платформо-независимостью, Java предоставляет уникальный инструмент для создания интерактивных распределенных информационно-обучающих систем на база Internet/Intranet - технологий.

Основными сложностями при реализации корпоративных систем на базе данной архитектуры являются:

* отсутствие многих популярных приложений и средств разработки реализованных в виде JAVA аплетов;
* относительное высокое время компиляции аплетов на клиентских местах (временно);
* вопросы безопасной работы в сети.

#### 2.2.1. Сравнительные исследования типовых серверных платформ.

Выбирая платформу для АИС, нужно учитывать множество аспектов. На решение влияют соображения, связанные с надежностью (кластеризация и балансировка нагрузки), среды разработки, работы над содержанием узла и защиты информации. Результаты тестирования различных платформ широко представлены в периодической печати, представим здесь лишь некоторые обобщения материалов тестирования [].

При проведении испытаний оценивались Solaris 2.6, Windows NT Server 4 и Red Hat Linux 6.02 (ядро 2.2.11) при эксплуатации четырех web-серверов, занимающих ведущие позиции в мире: Microsoft Internet Information Server 4 (IIS), Netscape Enterprise Server 3.61, Web Server 2.1 корпорации Sun и Stronghold Web Server 2.4.1 (популярный вариант Web-сервера Apache с функциями защиты от несанкционированного доступа) (на тестах использовались триал версии указанного программного обеспечения). Все платформы были испытаны с помощью новой версии эталонного теста WebBench отделения Ziff-Davis Benchmark Operation.

#### 2.2.1.2. Особенности функционирования АИС на платформе Sun.

Solaris - это современная операционная система UNIX клона. Примечательно, что она, опережая свое время, позволяет работать с 64-разрядными прикладными программами и имеет собственные расширения, которые помогают ей выдерживать высокие пользовательские нагрузки Web-узлов с интенсивным обменом информацией. В Solaris также предусмотрены замечательные возможности применения серверных прикладных программ и средств разработки сторонних производителей.

Ключевой момент для понимания различий между платформами Linux, Microsoft и Sun - способ, которым серверные программы каждой из них обрабатывают большое число подключений. Обычно это делается в многопотоковом режиме. Многопотоковый режим возникает, когда прикладная программа (также называемая процессом) содержит множество небольших блоков исполняемого кода, работающих независимо друг от друга и, возможно, одновременно на разных процессорах. Эти потоки могут совместно пользоваться ресурсами и представляют собой способ организации программы, позволяющий одновременно выполнять несколько задач.

Модель потоков Solaris весьма сложна. Она состоит из потоков на уровне ядра (kthreads) - реальных объектов, передаваемых отдельному процессору; потоков на пользовательском уровне и промежуточной структуры, называемой облегченным (lightweight) процессом. Это позволяет тонко управлять структурой прикладной программы и реализации в ней прикладной многозадачности.

**Stronghhold на платформе Solaris**

Создатели Web-сервера Stronghold (и Apache, основы Stronghold) считают, что многопотоковые программы обычно менее надежны, чем "монолитные". Такое различие стратегий объясняет значительные расхождения показателей производительности, поскольку и Sun Web Server 2.1, и Netscape Enterprise используют второй процессор, установленный в испытательных системах. Поэтому Stronghold, в зависимости от прикладного ПО, не столь эффективно использует оборудование Sun, содержащее до 64 процессоров.

**Netscape на платформе Solaris**

Netscape Enterprise Server 3.61 - Web-сервер, избранный для реализации большинства крупных узлов на основе Solaris, в том числе и корпорации Sun. Инструментальные средства фирмы Netscape, а также предлагаемые независимыми производителями, способствуют разработке сложных прикладных программ для Web с помощью сценариев на языках JavaScript, CORBA, Java.

Еще одна важнейшая система, стоящая за добротными программами для Web на серверах Netscape, - сервер прикладных программ Netscape Application Server (NAS). Сервер NAS - среда программирования для объектов на языках C++ и Java - обеспечивает масштабируемость и устойчивость к сбоям прикладных программ. В NAS имеются инструменты для создания многоуровневых программ, объединяющих HTML и запросы к базам данных на серверах NAS.

**Sun Web Server**

Sun Web Server (SWS) обеспечивает разработку программ, конечно же, на языке Java. На SWS можно использовать сервлеты и разнообразные возможности, такие как CORBA. Сервлеты (servlet) - это Java-программы, запускаемые на сервере и, подобно CGI, передающие сверстанные HTML-страницы браузеру. Для сервлет существует собственный API к функциям рабочей среды сервера. В SWS также предусмотрена возможность использования серверных Java-страниц (Java Server Pages) - способа обращения к серверным функциям Java со страниц Web и из CGI-программ.

При соответствующем использовании Web-серверов на платформе Solaris, эта операционная система на многопроцессорных станциях превосходит по производительности Windows NT. Такого результата достигла Sun Microsystems благодаря использованию Solaris Network Cache and Accelerator (SNCA) - мощного механизма кэширования для Web-сервера. SWS победил в испытаниях при обслуживании статических страниц. При выполнении динамических CGI-испытаний Netscape на платформе Solaris превзошел и SWS, и IIS для Windows NT.

#### 2.2.1.3. Особенности функционирования АИС на платформе Microsoft.

**Microsoft Windows NT Server**

Windows NT 4 Server и Internet Information Server (IIS) являются исключительно коммерческой web-платформой, разработанной компанией Microsoft. Данная ОС имеет удобный интуитивно понятный интерфейс взаимодействия с пользователем, что делает её довольно привлекательной для использования. Windows NT 4 Server оснащена службой балансировки нагрузки (Windows NT Load Balancing Services), которая позволяет создавать группу серверов и распределять нагрузку между ними. Пользователи при этом видят только один IP-адрес и полагают, что существует только один сервер. Однако служба Load Balancing Services - это неполноценная кластерная система, поэтому она не способна обеспечить такое высокое быстродействие, как настоящий кластер. Windows NT не может работать с мощными аппаратными и программными средствами кластеров, в том числе с собственной службой Microsoft Cluster Service, продуктами серии Infinity компании IBM и продуктами NonStop производства Compaq. У Microsoft есть продукты всех этапов разработки для Web, однако обычно их заменяют изделиями других фирм. Пакет Allaire ColdFusion 4.0, как среда разработки для Web, - отличный пример этого.

**Netscape Enterprise на платформе Windows NT**

Netscape Enterprise в среде Windows NT представляет собой Web-сервер, ориентированный на большие нагрузки. Для него имеется множество моделей программирования. Например, помимо общепринятых моделей разработки HTML и CGI в продукте Netscape предусмотрены возможности работы с JavaScript на стороне сервера. Почти все функции сервера Netscape для Solaris работают и на платформе Windows NT.

При тестировании на производительность IIS показал неплохие результаты. Скорость при работе IIS достигнута за счет хорошо организованной обработкой файлового ввода-вывода. Дополняет обработку сообщений в Windows NT возможность асинхронного ввода-вывода, позволяющая обрабатывать запрос одновременно с выполнением операций ввода-вывода в файл или ЛВС. Подобная функция имеется в Solaris, но до сих пор не полностью реализована в Linux. По результатам теста IIS проигрывает SWW при обработке статических страниц, а Netscape Enterprise на платформе NT оказался менее производительным во всех режимах, чем на платформе Solaris.

#### 2.2.1.4. Особенности функционирования АИС на основе Linux.

Все больше растет популярность Linux и её респектабельность как платформы разработки для Web и корпоративных сред. Linux характеризуется рядом преимуществ, таких как широкое сообщество разработчиков открытого кода, поддержка многих моделей комплектующих, и, главная особенность состоит в том, что Linux полностью бесплатная ОС. Linux является разновидностью Unix и изначально создавалась для работы в сетях. В каждой новой версии Linux появляются некоторые усовершенствования, направленные на повышение масштабируемости и производительности серверных прикладных программ.

**Apache и Stronghold**

Для тестов в среде Linux был использован Stronghold Web Server 2.4.1 компании C2Net. Stronghold - это сервер с возможностями применения технологии SSL, в основе которого лежит Web-сервер Apache. Сервер Stronghold обладает всеми преимуществами Apache, в том числе мощными средствами обеспечения работы с виртуальными базовыми машинами (способность одного web-сервера обслуживать несколько машин одновременно).

Платформа Stronghold, подобно Linux, имеет заслуженную репутацию надежной и стабильной системы. Но Stronghold - и, следовательно, Apache - не оптимизированы для многопроцессорных сред. Поэтому Web-узлы, основанные на серверах Apache, лучше масштабировать путем добавления серверов, а не процессоров.

Напротив, IIS и Netscape Enterprise имеют многопотоковую архитектуру, которая масштабируется на несколько процессоров одного сервера. При испытаниях на многопроцессорных станциях они, как правило, обгоняли Stronghold.

Apache позволяет тонко настраивать ряд параметров (такие как число процессов, доступных клиентам). Для Apache, как и для других серверов, есть механизм работы сервлетам (Apache Jserv). Механизм работы с сервлетами встраивается в Apache в виде модуля и работает с любой совместимой с JDK 1.1 виртуальной Java-машиной. По производительности дуэт Apache-Linux оказался оптимальным для однопроцессорных систем. По обработке статических страниц он немного уступал SWW и IIS, а по стабильности работы превосходил серверы на платформеWindows NT.

Linux - это функциональность UNIX + пользовательско-ориентированный интерфейс Windows-систем. Большая часть поддерживаемого Linux оборудования - это то, что пользователи реально у себя имеют. Как в результате оказалось - большая часть популярной периферии для 80386/80486 поддерживается (действительно, Linux поддерживает оборудование, которое в ряде случаев не поддерживают некоторые коммерческие UNIX). Хотя некоторые достаточно экзотические устройства пока не поддерживаются.

Важным вопросом при создании АИС является обеспечение жизнестойкости и надежности работы информационных серверов. В качестве иллюстрации эффективности платформы приведем расчет параметров для сервера с 25000 посетителей в день []. Подсчет загрузки: 24ч\*60мин\*60 сек=86400 секунд в сутках, если каждый посетитель берет с сервера по 10 документов (\*.html + графика) то при равномерном распределении загрузки получается 3 обращения к серверу в секунду. Реальное распределение трафика носит характер кривой Гаусса либо синусоиды (в зависимости от содержания сервера), в максимумах которых загрузка достигает 10-20 обр/с. Для нормальной работы такому серверу необходимо около 400 Mb RAM, при хорошей настройке - не менее 200.

При правильной конфигурации сервера не рекомендуется использовать swap. все должно помещаться в оперативной памяти (имеется ввиду, что у сервера swap-область быть должна, но она обязана быть пустой). Для предотвращения перегрузок рекомендуется пользоваться несколькими правилами, снижающими загрузку сервера. Придерживаясь этих правил можно съэкономить около 30% ресурсов сервера.

1. Для статической информации всегда ставить last-modified в атрибут выдачу CGI-скриптов - документ без временного штампа не сохраняется в локальном кэше, и постоянно перезаписывается при просмотре.
2. CGI программы хранить в любом каталоге кроме /CGI-BIN/, т.к. proxy-серверы не кэшируют файлы, находящиеся в этих каталогах, и каждый раз вынуждены обращаться к вам на сервер.
3. Устанавливать поле last-modified у русского apache с автоматическим определением кодировки, чтобы на proxy-серверах не оставались файлы в некорректной кодировке.
4. Не применять авторедирект по чарсету в русском apache.
5. Не использовать фреймы, т.к. вместо одного файла появляется минимум 3.
6. Не использовать анимированные \*.gif, т.к. некоторые NN обращаются к серверу перед каждым циклом.
7. 404 код не делать cgi-скриптом, 404 код не делать "красивым" - с графическими изображениями и указаниями на прочие разделы, т.к. сошедший с ума робот собирает невероятное количество 404 ошибок, зацикливаясь в них на веки.
8. Создать на сервере файл robot.txt, т.к. это самый запрашиваемый документ на сервере, и иначе порождает массу 404 (см. п. 7). А также разумные роботы слушаются запретов в этом файле, что уменьшает нагрузку на сервер.
9. Не ставить баннеры наверху страницы, т.к. баннер сверху отнимает 1-2 реквеста из 4-х и в итоге грузится вперед тормозя ваши сайтовые картинки.
10. При вызове баннера не обращаться каждый раз к CGI, а подставлять вместо случайного числа любое число, что можно сделать, например, получив дату на JavaScript.
11. Вызывать баннеры программами на Си, т.к. Perl работает медленнее.
12. На одной машине должен размещаться только информационный сервер, не одновременно с почтой и др. сервисами.

На сегодня архитектура Internet/Intranet, в том числе и на платформе LINUX, уже используется при построении корпоративных ИС для решения задач автоматизации управления банками, управления проектированием, управления ТП, АСУ ТП, электронной коммерции, оперативной информации по курсу валют и акций и т.п.