АНАЛИТИЧЕСКИЙ ОБЗОР КОМПАНИИ SUN MICROSYSTEMS

Оглавление

Введение

В наши дни концептуальный лозунг, провозглашенный когда-то компанией **Sun**, “Сеть - это компьютер” воспринимается уже как нечто само собой разумеющееся, а ведь менее двух десятков лет назад, когда на рынке только появились первые рабочие станции **Sun-1**, справедливость этого девиза не была столь несомненна. В те времена, когда господствовала идеология централизованной обработки информации на базе мэйнфреймов, фирма **Sun** предложила новый подход к развитию компьютерных технологий, заложив в его основу ряд принципов:

- Открытую и совместимую с широким спектром прикладного ПО операционную систему **Unix**, ориентированную на эффективную работу в сетях архитектуру аппаратного обеспечения.
- распределенную сетевую файловую систему **NFS**
- специализированные средства сетевого администрирования.

С течением времени эти составляющие развивались и трансформировались, и к началу 90-х базис технической концепции **Sun** оформился в прочную конструкцию из аппаратной **RISC**-платформы с архитектурой **SPARC**, операционной системы **Solaris**, сетевой файловой системы **NFS** и платформы сетевого администрирования **SunNet Manager**, место которого позднее заняло ПО **Solstice**. Но изначально выбранная стратегия, концентрирующаяся в лозунге “Сеть - это компьютер”, осталась практически неизменной, а верность ее отразилась в утверждении **Sun** на месте прочного лидера в сфере производства рабочих станций, серверов и сетевого программного обеспечения и характеризовалась взлетом продаж продукции **Sun** в начале 90-х годов.

История компании

Когда говорят о преуспевающих в бизнесе компаниях, то часто характеризуют годы их деятельности ключевыми событиями в их истории: крупные сделки, объявления важных продуктов, прибыльные годы. Однако для того, чтобы отследить этапы развития компании **Sun**, лучше всего обратить внимание в первую очередь на то, как принятые на раннем этапе стратегии воздействовали на всю компьютерную отрасль и как созданные **Sun** новые технологии стали органичной частью современной жизни компьютерной индустрии.

Компания **Sun** была создана в 1982 году в Стенфордском Университете, одном из лучших в мире учебных заведений в области компьютерных технологий. Аббревиатура **SUN** означала **Stanford University Network** - Сеть Стенфордского Университета, таким образом, уже в самом названии компании была заложена идея сетевых вычислений. Все началось с того, что студент-старшекурсник из Германии **Андреас Бехтольшейм (Andreas Bechtolsheim)** решил создать собственный компьютер из доступных недорогих компонентов. В продвижении получившегося продукта ему помогал студент в области экономики **Винод Хосла (Vinod Khosla)**, которому хватило деловой хватки, чтобы понять экономический потенциал работы **Бехтольшейма** и быстро собрать средства среди представителей Кремниевой Долины. Вскоре в **Sun** пришел и **Билл Джой (Bill Joy)**, который руководил разработкой **UNIX** в Калифорнийском университете в Беркли. И, наконец, возглавить работу компании

взялся друг **Винода** из Школы бизнеса **Скотт МакНили (Scott McNealy)**. В 1984 году Винод покинул **Sun, и Скотт МакНили стал президентом**.

Основатели компании разделяли общее мнение о необходимости использования в качестве стандартов своей работы идей открытых систем и сетевых вычислений. Решения, принятые на первом этапе, задали тон развития как самой компании **Sun** с ее инновационными проектами, так и всей компьютерной отрасли в целом. Сочетание готовых компонентов, стандартизованной и доступной операционной системы и несложного дизайна позволило компании **Sun Microsystems** достаточно быстро предложить на рынок мощную и доступную рабочую станцию для технических специалистов. При этом **Sun** пыталась использовать и те новые графические и сетевые технологии, которые ранее были доступны только в более дорогих и <закрытых> устройствах.

Изначально стратегия фирмы была направлена не столько на быстрые доходы, сколько на расширение рынка и проведение фундаментальных исследований и разработок. Тем самым, **Sun** принесла пользу не только себе, но и всей отрасли в целом, превратив операционную систему **Unix** в свою ключевую технологию, предложив отрасли модель сетевой файловой системы **NFS** и лицензируя архитектуру **SPARC** и операционную среду Solaris, чем создала почву для надежных инвестиций тысяч компаний.

Продукты и решения

Сегодня **Sun**, компания из списка **Fortune 500**, имеет высокую репутацию в деловых кругах, лидируя на рынке рабочих станций, серверов и в области технологий Internet и Intranet. Всего несколько лет назад компания **Sun** предложила технологию **Java**, которая сейчас получила массовое признание и широчайшее распространение. Sun фактически направляет развитие технологии Java, играющей сейчас ключевую роль в связке Intranet/Internet и в области сетевых технологий в целом.

Sun Microsystems была пионером на рынке рабочих станций со своей концепцией открытых систем. Продолжая это направление, сейчас компания концентрирует внимание на построении и управлении сетями предприятий. Объявив несколько лет назад новую серию серверов масштаба предприятия **Sun Enterprise 3000-6000** (сейчас выпускаются системы 3500-6500), Sun кроме рынка рабочих станций получила признание и на рынке серверов на базе Unix. Результат был закреплен выпуском в 1997 году сервера, обладающего возможностями систем класса большой ЭВМ - **Sun Enterprise 10000**, известного также под названием **StarFire**. Но не были забыты и компьютеры на базе **Unix** и в самой низкой ценовой категории. По ценам, сопоставимым с ценами на системы на базе процессоров **Intel, Sun Microsystems** предлагает заказчикам 2-х и 4-х процессорные серверы рабочей группы **Sun Enterprise 250** и **Sun Enterprise 450**, а также рабочие станции: **Sun Ultra 5**, **Sun Ultra 10**, обеспечивающие пользователям мощь и функциональную полноту **Unix** по ценам ПК. При этом особенность систем семейства **Ultra** состоит в том, что в разных классах машин часто используются одни и те же аппаратные компоненты, то есть фактически системы имеют модульную архитектуру, что позволяет упростить и унифицировать разработку программного обеспечения для них.

UltraSparc + Solaris

Одной из важнейших ветвей разработок компании в области ПО является операционная система **Solaris**. В 1991 году **Sun** объявила о выходе новой версии операционной системы **Unix - Solaris**, которая была основана на двух разновидностях **Unix: Berkeley 4.2/4.3 (BSD)** и **AT&T System V.** Выросшая из **SunOS**, операционная система Solaris, соответствуя многочисленным промышленным стандартам (**X/Open UNIX 95**, различные разделы **POSIX 1003.1, X11R6**), имеет одним из своих важных свойств высокую степень масштабируемости. Одна и та же операционная система используется и на однопроцессорных рабочих станциях, и на серверах масштаба рабочей группы, и на 64-х процессорном сервере **Sun Enterprise 10000**, сервере масштаба предприятия. **Solaris** обеспечивает почти линейный рост производительности при увеличении числа процессоров, подключенных к системе.

Сейчас **Sun** занимается также и разработкой кластерных архитектур. За счет объединения отдельных серверов **Ultra** в общую систему пользователи могут добиться значительного повышения производительности и надежности вычислительных систем. **Sun** предлагает два варианта кластерных решений. Во-первых, это система повышенной надежности **HA (High Availability)**, где дублируются все элементы, формирующие кластер. Таким образом, при выходе какой-либо машины из строя наиболее ответственное приложение запускается на резервном узле кластера. Во-вторых, это система **PDB (Parallel Database)**, которая предназначена для обеспечения параллельной работы реляционных баз данных на составляющих кластер компьютерах.

Java и Java-технологии

Один из путей воплощения в жизнь главенствующей концепции **Sun** "Сеть - это компьютер", смысл которой состоит в использовании для информационных вычислений не только отдельно взятого компьютера с ограниченными ресурсами, но и практически безграничных ресурсов сети, к которым данный компьютер имеет доступ, воплотился в идее создания сетевого компьютера. Согласно ей все большие задачи должны решаться на мощных серверах, а клиент должен иметь доступ только к вводу данных и к результатам.

Начальные затраты на сетевой компьютер невелики, прочие расходы, например на эксплуатацию и модернизацию аппаратного и программного обеспечения, также заметно ниже расходов на содержание обычной рабочей станции или персонального компьютера. Невысокие цены связаны с тем, что сетевые компьютеры не нуждаются в специальном администрировании. Так как, например, большинство сетевых компьютеров останутся без собственных устройств постоянной памяти, все пользовательские данные и конфигурационная информация будут храниться на серверах. Таким образом, при модернизации программного обеспечения или при установке новых приложений все замены происходят только в одном месте - на сервере. Централизованное администрирование и конфигурация могут обеспечить значительную экономию для фирм, имеющих тысячи клиентских машин. **Java-технологии** позволяют размещать приложения на сервере и делать их доступными для любого клиента независимо от его платформы. Это быстрее традиционных путей

распространения программного обеспечения и заметно дешевле модернизации большого числа персональных компьютеров.

Концепция сетевого компьютера очень тесно связана с разработанным компанией **Sun** языком **Java** и **Java-технологиями** в целом. При вычислениях по модели **Java** задачи администрирования и модернизации с уровня клиентов переходят на централизованные серверы. Для поддержки работы с Java серверы должны поддерживать взаимодействие с клиентами **Java** и запуск приложений **Java**. Так, например, **Sun** выпускает сервера **Netra** и **Ultra Enterprise**, которые удовлетворяют подобным требованиям. Так как основная доля расчетов при работе с тонким клиентом, каким и является сетевой компьютер, выполняется на сервере, необходимо заботиться о повышении пропускной способности серверов и их масштабируемости с точки зрения производительности и емкости.

В настоящее время **Sun** предлагает новый вариант сетевого компьютера **SunRay**, который должен постепенно вытеснить обычные PC, хотя на этот раз компания готова и к равнодушному отношению потенциальных клиентов. Это уже не первая попытка **Sun**: первый сетевой компьютер, разработанный **Sun**, - **JavaStation** - был представлен около двух лет назад, но тогда его погубило глобальное снижение цен на персональные компьютеры, многочисленные задержки с выпуском самого продукта и медленная работа **Java**. Теперь появилось много нового: концепция **Hot Desk**, которая позволяет запускать приложения на серверах **Sun**, а также поддерживает работу и с **Windows NT**, пакет офисных приложений **StarOffice** от **Star Division**, а также поддержка **смарт-кард** для удаленной работы и администрирования. Кроме того, изменилась и сама стратегия распространения сетевых компьютеров. Теперь **Sun** собирается не продавать, а сдавать новые сетевые компьютеры в аренду примерно за 10 долларов в месяц.

В 1995 году компания **Sun Microsystems** ввела в оборот термин , обозначив им корпоративную сетевую инфраструктуру, построенную по принципам технологий **Internet** и в частности **WWW**. Появление такой новой технологии вытекло из проекта по разработке универсального интерфейса для бытовых приборов. Начальная цель так и не была достигнута, но зато из этого проекта зародилось многое из того, без чего сейчас трудно представить себе **Internet-технологии**. Так, предложенная концепция **Intranet** во многом базируется на технологии **Java**, которая была создана для разработки мобильных приложений для **WWW**.

Итак, язык **Java** родился в ходе осуществления проекта по созданию передового программного обеспечения для различных бытовых приборов. Сначала проект реализовывался на **C++**, однако в определенный момент разработчики осознали необходимость изменения самого инструмента программирования для борьбы с возникающими в ходе работы проблемами. Стало очевидно, что нужен платформонезависимый язык программирования, позволяющий создавать программы, которые не требовали бы компиляции на каждой новой аппаратной архитектуре, и которые можно было бы использовать на различных процессорах в различных операционных системах.

Рождению языка **Java** предшествовал интересный случай. Разработчик программного обеспечения **Патрик Нотон**, один из сотрудников компании, понял, что больше не в состоянии поддерживать сотни различных интерфейсов используемых в компании программ, и сообщил исполнительному директору **Sun** и своему другу **Скотту МакНили**, что собирается покинуть компанию. В ответ на это **МакНили**

попросил **Нотона** составить список причин, вынуждающих его совершить этот шаг, а также предложить возможные способы решения возникших проблем. Нотон не рассчитывал на то, что его письмо получит хоть какой-то отклик, тем не менее он составил его, раскритиковав многие программные разработки в **Sun**, в частности разрабатываемую в тот момент архитектуру ПО **NeWS**. Однако это письмо было разослано всем ведущим инженерам **Sun Microsystems**, которые сразу же откликнулись, поддержав идеи своего коллеги. Обращение получило одобрение и у высшего начальства **Sun: Билла Джоя**, одного из основателей **Sun Microsystems**, и **Джеймса Гослинга**, начальника **Нотона**. В результате было принято решение начать разработку чего-то нового и необыкновенного. Этим занялась команда разработчиков под кодовым названием **Green**, которая в качестве приложения своих сил выбрала исследование бытовых устройств, таких как **Nintendo Game Boys** и устройства дистанционного управления. Необходимо было найти средство, с помощью которого можно было бы установить взаимодействие между этими разнородными устройствами, учитывая, что все они - видеомагнитофоны, проигрыватели лазерных дисков, стереосистемы и другое оборудование - реализованы на разных процессорах. Поэтому при разработке ПО для таких устройств необходимо было бы учитывать конкретные аппаратные особенности каждого из них, быть зажатым в рамках предоставленных аппаратных средств. Эти факторы стимулировали появление нового подхода к программированию ПО, который позволил бы сглаживать существующие отличия архитектур и который должен был стать ведущим на рынке бытовой электроники. И разработчики группы **Green** приступили к созданию нового средства разработки программ, объектно-ориентированного языка программирования, который был назван **Oak** в честь дуба, росшего под окном **Гослинга**. Вскоре **Sun** преобразовала команду **Green** в компанию **First Person**. Компания имела интересную концепцию, но все-таки никак не могла найти ей применение. После ряда неудач взор был обращен на недавно зародившуюся в Internet мировую паутину - **World Wide Web**. Тогда было предложено использовать язык **Oak** для создания Internet-приложений. Так **Oak** стал самостоятельным продуктом, оторвавшись от мира бытовой электроники. Был разработан и специальный **Oak-браузер**, названный **WebRunner**. В 1995 году **Sun Microsystems** объявила о выходе нового продукта, переименовав **Oak** в **Java**, объяснение чему можно найти разве что в особой любви программистов к кофе, а браузер **WebRunner** был переименован в HotJava.

Если говорить коротко, то **Java** - это не только язык, но платформа разработки приложений, включающая в себя простой, переносимый, интерпретируемый, высокопроизводительный и объектно-ориентированный язык программирования, а также среду исполнения.

Один из основных принципов **Java** состоит в том, что операционная система отделена от разработки приложений. Иными словами, разрабатываемый с помощью **Java** код не должен зависеть от платформы, на которой он разрабатывается и выполняется. А чем более код независим от платформы, тем более он мобилен и переносим. Для обеспечения платформонезависимости была специфицирована виртуальная **Java-машина**, на которой должны выполняться **Java-программы**, определены ее архитектура, система команд и представление данных. Исходные тексты **Java-программ** транслируются в коды этой машины. Это не машинные коды, подобные производимым компиляторами языка **C**, а так называемые байт-коды - высокоуровневые машинно-независимые коды для абстрактной машины, которая состоит из интерпретатора **Java** и исполняющей системы. Набор байт-кодов **Java** легко не только интерпретировать, но и достаточно эффективно компилировать <на лету> в машинные коды той платформы, на которой работает виртуальная машина **Java**. При

этом байт-коды содержат избыточную информацию, позволяющую проверять их на безопасность исполнения. Концепция виртуальной машины обеспечивает то, что при появлении новой аппаратно-программной платформы в переносе на нее будет нуждаться только **Java-машина**, а все программы, написанные на **Java**, изменений не потребуют. Стандарт **Java** определяет также, что при редактировании внешних связей программы прозрачным для пользователя образом может осуществляться поиск необходимых объектов не только на локальной машине, но и на других компьютерах, доступных в сети.

На первых порах многие рассматривали **Java** просто как средство оживления **Web-страниц**. Однако появились и другие способы разнообразия содержимого **WWW**, например, анимированные GIF-изображения, программы **JavaScript**, специально разрабатываемые для браузеров **plug-ins**. А коньком **Java** стала считаться именно возможность осуществления мобильных вычислений. Программы, работающие на любой платформе и свободно передающиеся по сети, оказались весьма удобны. Надо отметить, что идеи мобильного кода в сети **Internet** существовали уже давно, однако для их осуществления не находилось подходящей среды. На роль такой среды серьезно претендовала система **X Window**, однако она не имела необходимого для этого универсального интерфейса. Тем не менее мобильный код имеет и ряд существенных недостатков, основные из которых состоят в безопасности передаваемого кода. Во-первых, исходная программа может изначально содержать <программные закладки>, вирусы или просто ошибки. Во-вторых, возникает опасность несанкционированного доступа при использовании мобильного кода. Для того чтобы избавиться от возможных проблем, связанных с ошибками в программном коде, в **Java** удалили адресную арифметику и ввели специальный режим работы с памятью. При этом программа запускается в отдельной, виртуальной машине со своим адресным пространством. Кроме того, слежение за процессом выделения и освобождения динамической памяти в **Java** ведет уже не программист, но сама среда. Программисту не нужно заботиться о своевременном освобождении более не используемой им части динамической памяти, об этом позаботится “сборщик мусора” **виртуальной** машины. Главной задачей этих мер была борьба с наиболее распространенными ошибками: переполнением строковых констант фиксированной длины, переполнением стека при вызове подпрограмм, выделением и освобождением памяти во время работы программы. Разумеется, этот подход не мог не сказаться на эффективности кода. Однако такой недостаток компенсируется потенциально большей устойчивостью системы. Для обеспечения безопасности, связанной с возможностью несанкционированного доступа при использовании мобильного кода, в спецификацию **Java** были введены ограничения на использования кода в сети. В первую очередь это относится к разработке и использованию распределенных информационных систем. Ограничения вводятся на получение и передачу данных и кода на хосты, отличные от того, с которого запущен **Java-аплет**, что приводит к необходимости применять особые сервера-посредники **(proxy)** для связывания всех необходимых компонентов.

Java-технологии получили особенно широкое применение в инфраструктуре сетей **Intranet**. Термин **Intranet** используется для обозначения использования технологий **Internet** во внутренних корпоративных сетях. При этом в основе **Intranet** лежит идея эффективного совместного использования информации через единый тип интерфейса пользователя - браузер **Internet**. Использование парадигмы браузера **Internet** упрощает работу пользователей в сети, позволяет избежать проблем, связанных с обновлением программного обеспечения на рабочих местах пользователей, улучшает внутреннее взаимодействие систем в корпоративной сети в целом. Использующиеся в **Intranet** принципы web-навигации и поиска облегчают

пользователям процесс сбора и анализа информации. Одним из преимуществ использования **Java-технологий** в корпоративных сетях является предоставление пользователям возможности входить в систему из любого места **Intranet** и получать при этом доступ к своей родной рабочей среде. В такой модели централизованного управления информацией стоимость администрирования систем значительно снижается.

Итак, в основе **Java-вычислений** лежит модель клиент/сервер, в которой программный код **Java** загружается динамически с сервера по требованию клиента. Приложения **Java** могут работать в любом месте, где установлено программное обеспечение виртуальной машины **Java**. Этим достигается платформонезависимость и мобильность кода **Java**. Так, например, они могут исполняться в любом браузере с поддержкой Java. Именно это свойство позволяет добиться постепенной миграции на тонкие и, следовательно, легче управляемые клиенты - специальные устройства вроде **JavaStation** или недавно анонсированной компанией **Sun** станции **SunRay**.. Такие устройства - сетевые компьютеры - вместо традиционной ОС содержат простую систему, например **JavaOS**, которая обеспечивает работу виртуальной машины Java. **JavaOS** и виртуальная машина Java могут храниться как на клиенте во флэш-памяти, так и загружаться из сети. Данные пользователей и конфигурационная информация для клиентов хранятся на серверах, что обеспечивает централизованное администрирование и возможность доступа к необходимым данным из любой точки корпоративной сети. Архитектура **JavaOS** представляет собой совокупность микроядра и диспетчера памяти, драйверов устройств, виртуальной машины Java, системы **JavaOS Graphics** и **JavaOS Windowing**, сетевых классов и средств поддержки всех интерфейсов прикладного программирования **(API) Java**. Исполняемые на клиенте приложения взаимодействуют с серверами посредством стандартных сетевых протоколов, но могут использоваться также более сложные протоколы, например протокол **JDBC**, который обеспечивает **SQL**-ориентированное подключение к базам данных. Более сложные, многоуровневые приложения могут быть построены с использованием распределенных объектов, которые имеют возможность взаимодействовать при помощи протоколов **CORBA**. Следует отметить, что платформонезависимость Java позволяет использовать **Java-технологии** не только в традиционных настольных средах, но и в многочисленных устройствах, например в интеллектуальных телефонах со встроенными дисплеями, цифровых ассистентах **(PDA)** как, например, **PalmPilot**, в различных приставках, кассовых устройствах и т.д.

Технология Java руководствуется девизом **"Write Once, Run Anywhere"** ("Пишем один раз, используем везде"), устанавливая этим стандарт кросс-платформенной совместимости. Такой принцип разработки ПО позволяет оптимизировать процесс создания и внедрения приложений, позволяет сократить циклы разработки новых продуктов. Язык **Java** и модульные интерфейсы прикладного программирования **JavaBeans** не привязывают разработчиков и пользователей к какой-либо одной платформе. Однако нужно отметить, что **Java** позволяет делать вставки кода, написанного на других языках, например на **C++**. Поэтому для того, чтобы предоставить гарантию портируемости программ на **Java**, компания **Sun** предложила инициативу **100% Pure Java** (Стопроцентно чистая Java). Эта инициатива определяет набор интерфейсов **API**, которые обеспечивают максимальную кросс-платформную совместимость разработанных с их помощью приложений Java. Изначально Java поддерживает высокий уровень платформенной независимости, безопасности кода, простоты загрузки удаленных классов, а также “сборку мусора”. Однако при попытках связать код Java с кодами исходной платформы эти особенности могут быть утрачены. При использовании платформозависимых методов уже нет гарантии того, что код будет

безопасен и не нарушит устойчивость системы. Иногда может потребоваться включение методов исходной платформы для доступа к системным ресурсам, которые не поддерживаются Java. В таких случаях можно создавать интерфейс для связи между платформозависимыми кодами и кодами **Java**, однако такое приложение уже не сможет считаться стопроцентно соответствующим стандартам кросс-платформенной совместимости Java. Для поддержки возможности максимальной портируемости код программы должен быть целиком написан только на языке Java без каких-либо вставок на других языках, например **C++**. Приложения, претендующие на соответствие определению 100% **Pure Java**, должны проходить специальную сертификацию. Могут быть сертифицированы и части приложений, которые пишутся с использованием кодов Java и иных кодов. В этом случае могут быть сертифицированы те части приложения, которые отвечают определению 100% **Pure Java**. Таковы, например, приложения типа клиент/сервер, клиентская часть которых полностью написана на Java.

Важнейшей основой разработки приложений на **Java** является интерфейс прикладного программирования **Java API**. Он обеспечивает взаимодействие компонентов, аплетов и приложений Java, а также определяет набор ключевых интерфейсов, используемых разработчиками для написания приложений. **Java API** состоит из двух компонентов: базового интерфейса **Java Core API** и интерфейса стандартных расширений **Java Standard Extension API**. Приложения, удовлетворяющие стандартам **100% Pure Java**, должны полностью соответствовать базовым спецификациям интерфейсов прикладного программирования **Java (Java Core API)**.

Относительно недавно компания **Sun** выпустила новую версию платформы **Java - Java 2**, прежде именовавшуюся **Java 1.2**. В этой версии значительно повышена производительность среды исполнения приложений, реализована новая гибкая модель безопасности и включен расширенный набор программных интерфейсов **(API)**. Теперь интерфейс **Java Core API** состоит из пятнадцати библиотек классов, охватывающих основные технологии **Java: applet, awt, beans, io, lang, math, net, rmi, security, sql, text, util, accessibility, swing, corba**. Существовавшая до этого модель безопасности, работающая по принципу - песочница, то есть изолирующая работающее в виртуальной машине приложение от реальной операционной системы, была расширена таким образом, чтобы разработчики, администраторы и пользователи имели возможность определять собственную гибкую политику безопасности, или набор правил, обеспечивающих контроль над действиями приложений и аплетов. Полномочия Java-программ могут изменяться в зависимости от источника получения Java-программы, что определяет уровень доступа этой программы к ресурсам системы. Java 2 включает в себя также реализацию набора программных интерфейсов **JFC - Java Foundation Classes**, которые позволяют решать большое число стандартных задач прикладного программирования. В **Java 2** включен и механизм интеграции с рядом существующих технологий сетевого программирования. Так, например, В **Java 2** включен модуль **Java IDL**, который обеспечивает возможность удаленных взаимодействий по стандартам CORBA.

Одна из ветвей раскидистого дерева **Java-технологий** от **Sun** - технология **JavaBeans**, которая реализует концепцию платформнонезависимой модульной архитектуры программного обеспечения для среды разработки **Java**. Модулем **JavaBeans** может почти любая часть программной системы: от элемента графического интерфейса пользователя до крупного элемента прикладной программы с большим количеством функций. Преимущество технологии **JavaBeans** состоит в заложенном в ее основу принципе модульности, что означает, что один и тот же код может быть

повторно использован в различных приложениях. Так, например, текстовый редактор, реализованный в виде **JavaBeans-компонента**, может найти применение в широком спектре разрабатывающихся приложений без переписывания его кода. Следуя принципам Java, технология **JavaBeans** не зависит от платформ, поэтому решения на ее основе могут использоваться и на традиционных компьютерах, и на иных архитектурах, например в сотовых телефонах или микрокомпьютерах типа "персональный помощник".

Модульное ПО **JavaBeans** может интегрироваться и в другие модули, такие как **ActiveX**. Такая совместимость предоставляется при помощи разработанного компанией **Sun** средства управления **ActiveX (JavaBeans Bridge for ActiveX)**. Данное средство дает возможность **JavaBeans** работать в традиционных оболочках **ActiveX**, таких как **Microsoft Office** и **VisualBasic**.

Одна из новых технологий, на которую **Sun** возлагает большие надежды и в которой, возможно, видит фундамент сетевых вычислений следующего столетия, носит название **Jini**. Принципиально **Jini** представляет собой программную архитектуру на основе языка **Java**, которая позволяет распространять службы операционной системы по всей сети. Говоря иначе, **Jini** - это часть программной среды **Java**, которая работает на основе виртуальной машины **Java** и позволяет устройствам динамически подключаться к сети и пользоваться возможностями всех имеющихся в ней остальных устройств, если те зарегистрировали специальным образом свои услуги и согласны предоставлять их сетевому сообществу. В перспективе Sun видит возможность подключения устройств в сеть и моментальное установление ими связи с остальными устройствами в сети без использования каких-либо дополнительных драйверов. Для начала свободного общения устройства со своими <коллегами> будет достаточно просто включить его в сеть.

Проект **Jini** начал развиваться параллельно с ростом **Java-технологий**, зародившись в 1994 году, когда была анонсирована **Java**. Основная цель проекта состояла в создании сетевой инфраструктуры, которая позволила бы обеспечить возможность простой интеграции разнородных устройств в сети. Неудивительно, что интерес к данному проекту сразу проявили многие компании, занимающиеся бытовой электроникой, периферийными компьютерными устройствами, предоставлением сетевых услуг, поставкой компьютерных систем и вообще связанные с использованием передовых компьютерных технологий, так как рыночный спрос на технологию, которая упрощает работу устройств в компьютерной сети и делает эту сеть более функциональной, очевиден. Сейчас многие компании сотрудничают с Sun в разработке и тестировании технологии **Jini**, анализируя и испытывая ее в решении своих задач.

В основе концепции создания технологии **Jini** лежала следующая идея: работа большинства современных информационных и вычислительных систем базируется на компьютерных сетях, которые сейчас создаются и развиваются повсюду. При этом оборудование, работающее в сети, а также сетевое программное обеспечение бывает нелегко заставить слаженно работать без дополнительных финансовых затрат. Особые неудобства это доставляет пользователю, которому нужна простая и надежная сеть, не требующая постоянной заботы и поддержи ее функционирования, не заставляющая тратить много сил на подключение новых устройств. К примеру, преимущества разрабатываемой компанией **Sun** технологии **Jini** можно проиллюстрировать следующим образом: при подключении в сеть устройства, например, **PalmPilot**, об этом сразу становится известно остальным устройствам, работающим в сети: карманный компьютер автоматически превратится в ее часть. В то же время компьютер

может получить информацию о других сетевых устройствах и предоставляемых ими услугах. Например, пользователь сможет беспрепятственно обращаться к данным на одних компьютерах, обрабатывать их с помощью программ на других, а затем распечатать информацию на ближайшем принтере, не используя при этом никаких дополнительных драйверов, а просто обращаясь к услугам, предоставляемым устройствами, подключенными к сети и использующими технологию **Jini**. Такая автоматическая интеграция должна быть реализована для всех электронных устройств, от персонального компьютера до сотового телефона.

Технически принципы работы технологии **Jini** состоят из следующих действий: при подключении к сети устройства происходит его автоматическая регистрация в поисковой службе **Lookup Service** с помощью сервисов обнаружения и присоединения **Discovery and Join Service**, тем самым устройство вступает в так называемую "федерацию". Взаимодействие сетевых объектов осуществляется с использованием механизма Вызова Удаленных Методов **(Remote Method Invocation, RMI)** - высокоуровнего способа поддержки распределенных вычислений платформой Java. Спецификации RMI дают возможность удаленным устройствам объявлять набор услуг, которые они могут предоставить (это называется объявлением интерфейса удаленного объекта в терминах Java). Когда клиентская система желает воспользоваться услугой, зарегистрированной в поисковой службе, поисковая служба сообщает этой клиентской системе адрес устройства, где эта услуга непосредственно доступна (в терминах Java, возвращает ссылку на удаленный объект, реализующий нужный интерфейс). Модель лизинга Leasing определяет принципы удаления перечней зарегистрированных услуг с доски объявлений и заключения контрактов, а список прав доступа **Access Control List** данной услуги определяет, кто именно может ею воспользоваться.

Правда, данная концепция архитектуры распределенных вычислений вряд ли представляет собой нечто совершенно новое. **Дэвид Джелернтер**, профессор Йелльского университета, в свое время выдвинул идею распределенной архитектуры в проекте **Linda**. Позже похожие идеи развивались в **Lucent Technologies** в рамках технологии **Inferno**, в IBM в проекте **T Space**, в национальной лаборатории **Оак-Риджа** в проекте **Parallel Virtual Machine**. Microsoft также строит планы относительно разработки распределенной операционной системы, получившей название **Project Millennium**.

Теоретически технология **Jini** может быть применена к любым устройствам или программам, работающим в сети. Jini - это не новая сетевая операционная система, а сетевая инфраструктура, созданная на основе технологии **Java** и делающая попытку вывести распределенные сетевые вычисления на новый уровень. **Jini** разрешает проблему несовместимости сетевого оборудования, а также обеспечивает эффективный механизм взаимодействия вычислительных устройств и приложений в сети.

Технология .com

Сейчас **Sun** переходит на новый виток развития сетевых технологий и, в частности, электронного бизнеса и коммерции, выдвигая концепцию ".com". ".Com" - это более глобальное понятие, чем простой перенос операций в сеть. По словам **Sun**, использование принципов ".com" позволит компаниям улучшить контакты со своими клиентами, поставщиками, партнерами и сотрудниками. Относительная новизна этой

идеи состоит в том, что теперь компьютерные компании, и в первую очередь Sun, намерены предлагать свои услуги еще более интегрировано, чтобы избавить своих клиентов от необходимости самостоятельной настройки и наладки приобретенного аппаратного и программного обеспечения. Пользователю нужно получить услуги, и это главное. Основное внимание пользователя будет сосредоточено исключительно на его непосредственных задачах, будь то предоставление каких-либо услуг посредством компьютерных технологий, торговля, производство. В таком видении компьютерная индустрия по форме приближается по форме к телефонной сети, где потребители услуг весьма слабо связаны с механизмом обеспечения их связи, а все что им нужно - это поднять трубку, услышать гудок и набрать номер. По сути, концепция ".com" явилась закономерным этапом развития услуг **Sun Microsystems**, особенно если учесть значительное развитие сетевого бизнеса и коммерции в последние годы. Как говорится, сейчас наступает новая эра невидимых вычислений, оставившая позади персональные компьютеры.

Одной из основ ".com" стала недавно выпущенная новая версия операционной системы **Solaris - Solaris 8**. Операционная среда **Solaris 8** - потенциальный стандарт эпохи ".com" - это продукт, построенный с использованием гибкой и масштабируемой архитектуры и обладающий расширенной по сравнению с предыдущими версиями системы функциональностью. Данная версия системы позволяет значительно увеличить производительность Web-серверов за счет использования **NCA (Network Cache Accelerator)**. По данным Sun приложения **Java** в ряде случаев увеличивают производительность в несколько раз благодаря использованию подсистемы **Java HotSpot**. Производительность базы данных **Oracle** по сведениям Sun возрастает на 40%. Вместе с выпуском новой версии **Solaris** компания Sun внедряет новую бизнес-модель, которая включает в себя свободный доступ к операционной системе **Solaris**, в том числе и к исходному коду ОС, и бесплатные лицензии на программное обеспечение для конечных пользователей.

Объединяя свои аппаратные платформы и программные технологии, такие как **Java** и **Jini**, а также новую операционную систему **Solaris 8**, и называя обновленный комплексный подход к предоставлению услуг концепцией ".com", Sun пытается сделать новый шаг в развитии сетевого бизнеса в частности и предоставлении сетевых услуг в целом.

Вывод

Развитие компании **Sun Microsystems** идет в ногу с эволюцией сетевых вычислений. Заложив основу бурного расцвета сетевых технологий, Sun остается одной из ведущих компаний в этом секторе рынка. Сейчас ни у кого уже не вызывает сомнений тот факт, что современный компьютер немыслим без сети, что подтверждает выдвинутый при основании Sun лозунг "Сеть - это компьютер". Сегодня компания Sun продолжает свои исследования и предлагает на компьютерный рынок технологии, которые позволят пользователям не только компьютеров, но и самых разнообразных электронных устройств, получить доступ к мощи современных сетевых вычислительных систем с простотой, достойной технологий следующего тысячелетия.