Руслан Богатырев

Java и Juice: дуэль технологий ?!

Предварительный вариант опубликован в журнале "Компьютерра" (1996 г.)

Итак, что бы вы предпочли: чашечку ароматного черного кофе или стакан свежего яблочного сока? Когда как. Хотя... а причем здесь сок? Все очень просто: в пышущей ароматом чашечке кофе многие программисты с недавних пор сразу безошибочно угадывают символ Java. А стакан сока — это символ новой веб-технологии с именем Juice (в пер. на русский язык означает "сок"). И ныне она бросает смелый вызов самой Java.

Так что же такое Juice? И чем эта технология превосходит то, что нам знакомо по миру Java? Давайте попробуем в этом разобраться.

Технология Java, разработанная в стенах Sun Microsystems, опирается на новый язык объектно-ориентированного программирования с именем Java. Во главу угла здесь ставятся два основных принципа — независимость от программно-аппаратной платформы (ОС + процессор) и безопасность (надежность) создаваемых приложений. Первый принцип реализуется через механизм виртуальной Java-машины (Virtual Java Machine) и через недавно анонсированную операционную систему JavaOS. Второй достигается прежде всего за счет языка Java (строгая типизация, запрет адресной арифметики, пакеты, автоматическая сборка мусора). Уникальность и неповторимость Java — один из главных мифов, который связан с этим языком и технологией. Он настойчиво вбивается в умы программистов и пользователей на протяжении всех тех полутора лет, как официально появился на свет язык Java. Что ж, удивляться этому не приходится: подобные утверждения — всего лишь один из приемов в рекламной кампании по раскрутке Java-технологии. Факт остается фактом: в языке Java нет явных средств, обеспечивающих динамическую подгрузку и исполнение программных агентов на произвольной архитектуре. Это заложено в Java-технологию, а потому ряд уже существующих языков может быть безболезненно переориентирован на поддержку подобных схем!

Более того, в мире разрабатывались схожие технологии, которые лишь в силу своего экспериментального характера не были известны широкой общественности. Технология Juice — это один из весьма показательных примеров. Juice опирается не на Java, а на язык Oberon, который был разработан швейцарским профессором Никлаусом Виртом (Niklaus Wirth), автором языков Паскаль и Modula-2, еще в 1988 г. С тех пор язык Oberon прошел весьма серьезную обкатку, обзавелся более совершенной модификацией Oberon-2 и за последние пять лет распространился в виде коммерческих систем программирования на большинстве известных платформ (MS-DOS, Windows 3.x, Windows 95, Windows NT, OS/2 Warp, VAX/VMS, Alpha AXP OpenVMS, Linux, NeXT OS, IBM AIX, HP-UX, SunOS, Solaris, IRIX, Mac OS и др.). С точки зрения надежности и безопасности исполняемых приложений язык Oberon ни в чем не уступает языку Java. И запрет адресной арифметики, и строгая типизация, и модульная инкапсуляция, и автоматическая сборка мусора, и даже динамический контроль типов — все это было предусмотрено в языке с самого его рождения. А с точки зрения модульного и объектно-ориентированного программирования он выглядит куда более простым, лаконичным и сбалансированным.

Язык Oberon опирается на операционную систему с именем Oberon, которая так же, как и JavaOS, обеспечивает прослойку между приложениями и программно-аппаратной платформой. Система Oberon позволяет работать как на "голом" оборудовании, так и поверх уже существующей ОС. Помимо этого она обладает целым рядом новаций в области пользовательского интерфейса и в области интеграции программных компонентов и составных документов. Важным достоинством языка и системы Oberon является стабильность спецификаций (за прошедшие годы изменения были минимальными), полная открытость исходных текстов операционной системы, всех некоммерческих библиотек, компиляторов и конверторов. Для максимальной эффективности работы Oberon-приложений была построена Oberon-ориентированная архитектура, нашедшая свое воплощение в экспериментальных компьютерах Ceres. Наконец, ведущим центром поддержки Oberon является не какая-то промышленная фирма,

а один из известнейших университетских центров Европы — Институт компьютерных систем ETH в Цюрихе.

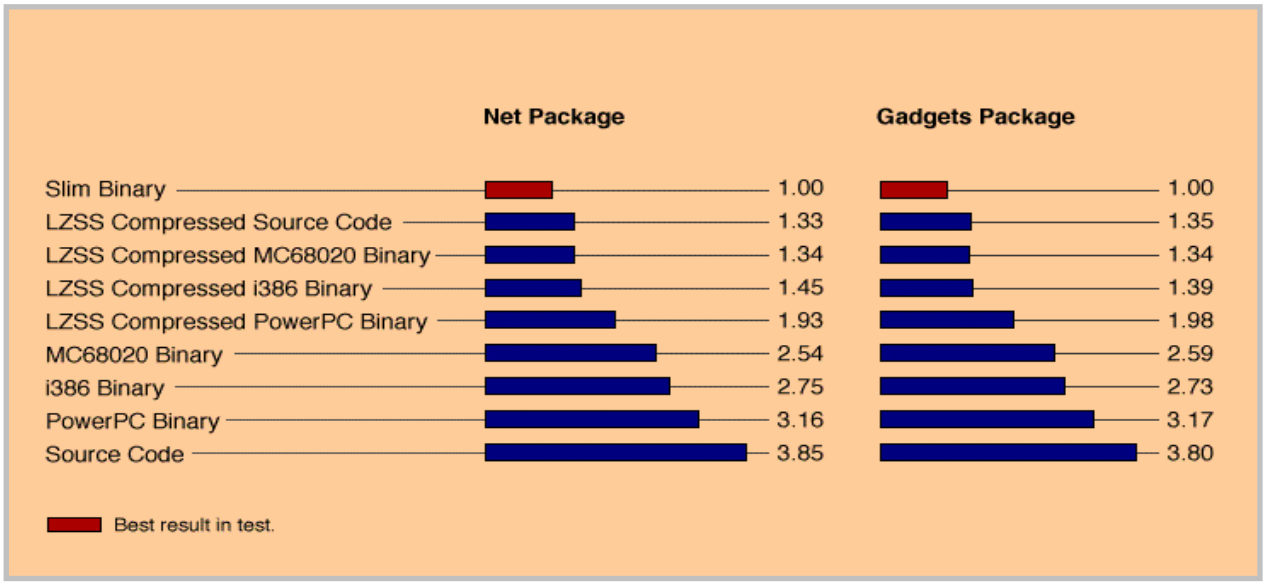
Все это замечательно, но каким же образом достигается независимость от платформы и как приложения на Oberon'е могут выполняться в среде Интернет/интранет ? Вот здесь-то и заключена изюминка технологии Juice. Да простит меня читатель за ряд технических подробностей, но если вам интересно разобраться в сути явления, придется немного сконцентрировать свое внимание. Давайте ненадолго вернемся к технологии Java.

Истинные приложения (applications) и мини-приложения (applets) языка Java перед своим выполнением проходят вполне определенный путь преобразований. Сначала исходный текст с помощью специального конвертора (Java Compiler) транслируется в байт-код Java. Затем этот линеаризированный объектный код попадает в виртуальную Java-машину, которая и осуществляет его привязку к конкретной платформе. Но это еще не все. Здесь включается в работу загрузчик байт-кодов (Bytecode Loader). Затем загруженный код проходит своеобразное "чистилище" — специальный верификатор байт-кодов (Bytecode Verifier). Ведь в ходе передачи по каналам связи Интернет/интранет возможно случайное или умышленное искажение объектного кода. Наконец, когда верификатор дает "добро", можно автоматически запускать интерпретатор байт-кода. На самом деле, истинная картина выглядит несколько сложнее. Но даже здесь легко видеть, что верификация кода и его интерпретация являются наиболее узкими местами в реализации технологии Java. Насколько неуязвимым (в смысле информационной безопасности) является верификатор байт-кода и как быстро может выполнять свою работу интерпретатор виртуальной машины? Для тщательной проверки верификатору придется проанализировать весь код, причем весьма замысловатым образом. Что касается интерпретатора, то он, как и следовало ожидать, работает крайне медленно. Это давно уже ни от кого не скрывают. Справедливости ради надо сказать, что здесь проблема уже частично решена за счет пост-компиляции: специальные Just-In-Time компиляторы разных фирм преобразуют байт-код в истинный исполняемый код конкретной платформы. Однако многого добиться столь замысловатый технологический путь все равно не позволяет.

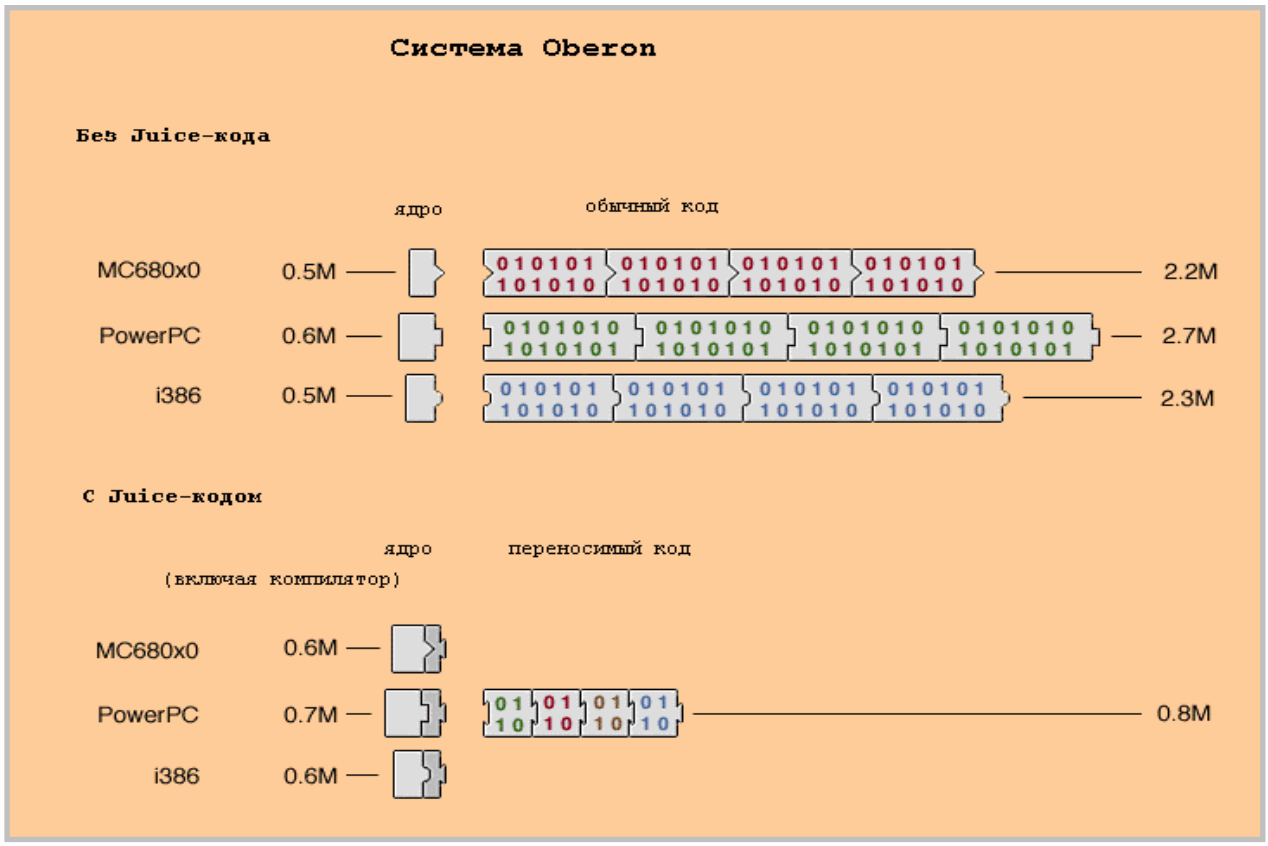
Что же касается технологии Juice, то она идет несколько иным путем. В основу положена динамическая кодогенерация по схеме Франца — один из побочных результатов проекта Oberon. В 1994 г. Михаэль Франц (Michael Franz) все в том же институте Вирта (Institute for Computer Systems, ETH Zurich) защитил диссертацию на тему "Динамическая (on-the-fly) кодогенерация — ключ к переносимому программному обеспечению". Идея состоит в том, что компилятор (причем не обязательно языка Oberon!) транслирует исходный текст в компактный промежуточный код древовидной структуры, построенный на принципе семантического словаря. В отличие от известного с начала 70-х годов байт-кода здесь, как ни странно, одновременно достигается куда более высокая плотность кода и при этом не теряется (как в случае байт-кода) высокоуровневая информация о структуре программы, которая крайне необходима для контроля целостности и для последующей эффективной оптимизации кода. Затем работает специальный загрузчик, который перед выполнением своих обычных функций осуществляет оптимизирующую генерацию загружаемого кода. Информационную безопасность обеспечивает специальный блок этого загрузчика. В отличие от верификатора Java ему не требуется выполнять огромную работу по анализу потоков данных и повторную работу по проверке целостности инструкций: за это отвечает Juice-дерево. На сегодняшний день уже реализованы и свободно доступны соответствующие программные расширения (plug-in) для наиболее распространенного сетевого навигатора Netscape Navigator. Причем это сделано для двух платформ: Apple PowerPC Macintosh и Windows 95 / Windows NT.

Главным идеологом технологии Juice является 32-летний Михаэль Франц. В среде Oberon-программистов он известен прежде всего как автор Macintosh и PowerMac-реализаций системы Oberon, а также как основной разработчик механизма OMI (Oberon Module Interchange) — важнейшей составляющей технологии Juice. Механизм OMI позволял осуществлять не просто динамическую кодогенерацию, он обеспечивал также смешанную динамическую загрузку OMI-кода и родного объектного кода данного компьютера. Здесь возникает вполне естественный вопрос: какой ценой удается добиться подобной гибкости? Она не столь высока, как в случае Java. Динамическая компиляция ни в чем не уступает лучшим оптимизирующим компиляторам и в этом смысле в отличие от виртуальных машин и пост-компиляции находится в более выгодном положении. Удивляет также и компактность кода, что особенно важно при его передаче по каналам связи и при подкачке с диска в оперативную память. Размер Juice-кода (так называемый slim-binary код) приблизительно в 4 раза меньше соответствующего исходного текста. И все же платить приходится: накладные расходы на динамическую генерацию кода возникают во время загрузки.

Как показали эксперименты, для достаточно объемных приложений время подобной загрузки возрастает от 1,3 до 5 раз в зависимости от приложения и оборудования. Однако, здесь гораздо показательней не абсолютная, а относительная оценка. Так, вся система Oberon загружается на Pentium/166 за 2,1 сек с обычным исполняемым кодом и за 3,8 сек с Juice-кодом. Взгляните на результаты тестов, которые проводились для двух наиболее объемных пакетов системы Oberon: это Gadgets (пользовательский интерфейс; 2,3 Мбайт исходного текста) и Net (сетевые функции, 0,7 Мбайт). На диаграмме указывается приведенный размер кода.



Еще одной платой за гибкость являются размеры кодогенерирующих загрузчиков. В среднем для разных платформ они составляют около 100 Кбайт.



Важно отметить, что технология Java позаимствовала для своей реализации принцип интерпретируемого байт-кода, который получил наибольшую известность благодаря разработке в начале 70-х годов P-кода, служившего для обеспечения переносимости Паскаль-программ на разные платформы. По иронии судьбы разработан этот код был группой ученых во главе с Никлаусом Виртом — крестным отцом технологии Juice.

Подведем некоторые итоги. Технология Juice, как и Java, призвана в первую очередь решать задачи не зависящего от платформы, быстрого и надежного исполнения самых разных приложений, прежде всего сетевых. В отличие от Java, которая изначально создавалась для встроенных приложений для бытовых приборов, Juice – более универсальная технология. Важно также и то, что уже имеется практическая реализация этой технологии для языка Oberon, который помимо своего возраста превосходит Java в ряде немаловажных языковых аспектов. В то же время Java имеет огромную поддержку в компьютерной индустрии и ныне с выходом JavaOS и активными работами по созданию специализированных Java-чипов (picoJava, microJava) обладает безусловным приоритетом перед своими возможными конкурентами.

Каково нынешнее состояние дел по технологии Juice? Об этом говорить пока рано: прошло около месяца с момента ее официального объявления. Но кое-что можно сообщить уже сейчас. Михаэль Франц в конце прошлого года покинул стены ETH и отправился из Европы в Америку, в Университет штата Калифорния в Ирвайне (University of California, Irvine). Здесь новоиспеченный профессор вместе с двумя своими студентами Томасом Кистлером (Thomas Kistler) и Мартином Бартшером (Martin Burtscher) и продолжает доведение технологии Juice. Права на товарный знак Juice Франц передал Попечительскому Совету Калифорнийского университета в Ирвайне.

Интересно, что первой реакцией некоторых программистов на анонс новой технологии, было недоумение по поводу столь простого и необычного названия. Оно и понятно: название, которое дается технологии, как и имя, которым нарекают человека, оказывает большое воздействие на всю дальнейшую судьбу. Сплетни быстро разносятся по свету. Вот уже и до нас дошли слухи из стана фирмы Borland, где родилась одна из первых реализаций пост-компилятора Java (AppAccelerator for Java) и где с особым пристрастием относятся ко всему, что так или иначе связано с Java. Первой реакцией их специалистов было восклицание: “Что за странное название!” Действительно, почему не Borneo, не Sumatra и даже не Mocca++ ? Правда, с самим названием Java, как теперь выясняется, вышла небольшая промашка. Американцы видно не могли знать, что Java — это не только название экзотического острова, но и весьма популярный в Париже 1900-х годов танец с участием путан и их сутенеров. А вот Juice — вещь безобидная, хоть и пришедшая из Старого Света. В переводе с английского помимо слова “сок” Juice обозначает также понятия “суть”, “основа”. Начальная буква “J” не только прозрачно намекает на связь с Java, но обладает еще и неким магическим действием. Как говаривал Винни-Пух в известном мультфильме, “жжж” — это неспроста! Еще раз посмотрите на фирменную эмблему Juice. На ней тщательно замаскирована явно яблочная основа этого сока. Тому тоже есть несложное объяснение. И связано оно с автором технологии.

Вглядитесь в его лицо и запомните дату 18 июля 1996 г. В этот день Михаэль Франц объявил в нескольких телеконференциях сети Интернет об официальном рождении новой технологии Juice. Что символично, произошло это в преддверии празднования на Руси яблочного спаса. Яблоко, хоть и надкушенное, как известно, символ фирмы Apple Computer, подарившей миру Macintosh-культуру. И вот один из преданных поклонников Macintosh наглядно доказал, что даже талантливый программист-одиночка в состоянии бросить серьезный вызов всесильной компьютерной индустрии. Правда, если за его спиной стоит настоящая школа и реальная поддержка коллег. И какая судьба ни постигнет теперь Juice, в лице Франца старушка-Европа в очередной раз утерла нос хваленой Америке. Наша страна в какой-то мере тоже приложила к этому руку. Ведь, по мнению авторитетных специалистов, именно у нас, в новосибирском Академгородке, разработан и продолжает совершенствоваться XDS — один из лучших коммерческих компиляторов языка Oberon-2, способный создавать удивительно компактный код для множества популярных компьютерных платформ!

В заключение давайте вновь вернемся к чашечке черного кофе и стакану яблочного сока, с которых мы и начали наше знакомство с технологией Juice. Сейчас уже основательно подзабыты многие полезные напитки, которые были в почете у наших мудрых предков. Среди них не последнее место занимает сидр, который одни называют яблочным вином, другие — яблочным напитком. Как мне удалось выяснить в одной из старинных книг, секрет его приготовления кроется не только в смешении сока двух разных сортов яблок (кислых и сладких), и не просто в отказе от использования сахара и воды. Все гораздо хитрее — в бутылку будущего сидра перед ее закупоркой обязательно кладут ровно две изюминки. Так вот, чтобы Juice из рядового яблочного сока превратился в настоящий сидр, нужны как раз-таки эти две изюминки. Одна из них — эффективная динамическая компиляция — у Juice уже имеется. Дело за малым — за реальной поддержкой технологии крупными компьютерными компаниями. Только тогда Juice сможет по-настоящему покорить мир и принести большую пользу многим и многим людям. Так давайте же пожелаем успеха всем ее нынешним и будущим разработчикам!