

Система портов FreeBSD как средство управления парком машин хостинга

A draft.

Станислав Седов <stas@FreeBSD.org>
Deglitch Networks

23 ноября 2008 г.

Аннотация

Данная статья повествует о том, как система портов FreeBSD может быть успешно применена для управления программным обеспечением машин хостинга, если Вы используете ОС FreeBSD в своей инфраструктуре, как мы. Какая-то часть из изложенных идей также может быть применена и к другим системам. Я постараюсь не сильно вникать в детали, но возможно для понимания некоторого материала потребуется обладать некоторыми навыками работы с FreeBSD.

1 Архитектура современного хостинга

Ни для кого не секрет, что современные площадки крупных хостинговых провайдеров представляют собой достаточно сложные распределённые системы. Инфраструктура состоит из большого числа серверов различной функциональности, которые тем не менее, служат одной задаче: обеспечение работы виртуального хостинга. Основными компонентами являются сервера HTTP, сервера СУБД (таких как MySQL, PostgreSQL) и shell-сервера, предоставляющие доступ к серверу по протоколу ssh. В зависимости от архитектуры конкретного хостинга, возможно наличие каких-то других серверов. Таких серверов обычно очень много - от нескольких десятков для малых провайдеров, до сотен и тысяч для самых крупных. И так как вся система внедряется не за один раз, а установка всех этих машин распределена по времени, то все эти многочисленные сервера отличаются друг от друга по аппаратной части, версии установленной на них операционной системы, и, как следствие, конфигурации. Ситуация такова, что ни один, ни даже несколько человек уже не могут удержать в голове все конфигурации всех машин хостинговой площадки. Всё это становится кошмаром администратора. Между тем, обновление и унификация системного ПО на серверах не всегда приемлемо ввиду больших рисков, неизбежных перебоев в работе и т.д.

Другая, не менее важная, часть проблемы заключается в том, что достаточно большая часть используемого ПО подвергается локальным модификациям. Действительно, для построения полноценной системы виртуального хостинга требуется обеспечить безопасность расположенных на хостинге данных (как в плане надёжности хранения, так и защиты от атак), повысить скорость и пропускную способность всей системы (а

это напрямую сказывается на прибыли хостинговой компании), приспособить работу многих программ к работе в порой весьма специфических условиях. Например, в компании хостинговые телесистемы мы используем достаточно сильно модифицированные версии ОС FreeBSD, Apache, Squid, vsftpd, php, кроме того применяется целый ряд специфичных для нашей площадки приложений, утилит, модулей apache и т.д. Всё это также требуется поддерживать для всех используемых версий ОС, архитектур, конфигураций и т.д.

2 Управление ПО системы виртуального хостинга

Когда-то, давным давно, когда в нашей компании ещё использовался Линукс, ввиду сложившихся традиций мы не использовали никаких пакетных менеджеров вообще. Существовал внутренний CVS репозиторий, в котором хранились патчи к используемому ПО и внутренние разработки, и сборка всех этих программ производилась вручную. В конечном итоге это привело к полной неспособности обновлять ПО на серверах из-за того, что их конфигурация отличалась. Приходилось на полном серьёзе производить сборку новых приложений непосредственно на хостинговых машинах. Часто это приводило с боям из-за человеческих ошибок, было невозможно делать быстрые обновления связанные с безопасностью, так как сборки от разработчиков ОС не могли у нас работать из-за проблем с зависимостями и, часто, из-за наших внутренних модификаций ПО. В общем, это было катастрофой. Притом, что число HTTP серверов у нас тогда еле достигало десяти и ещё нескольких серверов СУБД.

На текущий момент мы используем систему портов FreeBSD для установки и поддержки всех хостинговых серверов, включая сервера СУБД. Что это такое, и в чём преимущества такого подхода?

3 Система портов FreeBSD

Я думаю, многие из здесь присутствующих слышали об ОС FreeBSD и имеют какое-то представление о ней, так как именно эта система используется многими хостинг-провайдерами. FreeBSD представляет из себя полноценную ОС, в отличие от Линукс, и включает в себя помимо ядра все основные утилиты UNIX, а также текстовые редакторы, компиляторы, приложения для работы с электронной почтой и т.д. Но для реальных задач зачастую данной функциональности недостаточно, и возникает необходимость в установке стороннего ПО. Именно для этих целей и была создана т.н. коллекция портов FreeBSD, или Ports Tree, или просто ports. По сути, порты - это специальный framework, написанный на языке make, и состоящий из двух основных частей: 1) собственно, самого frameworkа, размещённого в дереве в директории Mk/ и представляющего из себя набор файлов обеспечивающих работу как всей системы в целом, так и отдельных её подчастей. и 2) самих портов, содержащих внутри себя Makefile'ы, описывающие процедуру установки конкретного приложения. Также там могут быть какие-то вспомогательные скрипты, файлы, патчи к данному приложению, если они требуются. Важной особенностью здесь является то, что процедура сборки может быть модифицирована с учётом специфики конкретного окружения с помощью переменных окружения. Вы можете поменять какие-то настройки

собираемого приложения, включить/выключить какие-то его возможности, убрать ненужные зависимости (например, x11). Кроме того, дополнительные патчи могут быть добавлены в files/, если нужно.

Одна и та же система портов используется в FreeBSD как для установки приложений в системе, так и для сборки пакетов. Пакет представляет собой архив, содержащий уже скомпилированные файлы приложения и инструкции по его установке/регистрации в системы. Пакеты никто не создаёт руками, они строятся автоматически системой портов, для этого служит `target make package`. После установки пакеты регистрируют свои зависимости и список файлов в директории `/var/db/pkg`. По сути, приложения установленные из пакетов и портов после инсталляции ничем не отличаются.

Для сборки пакетов FreeBSD служит кластер под названием PointyHat, который регулярно производит сборку всех пакетов под все поддерживаемые архитектуры и ревизии ОС. Время сборки всех портов может занимать от нескольких часов для amd64, до нескольких недель для sparc64. ПО, используемое для сборки, доступно в открытом виде. Кроме того, существует специальная версия пакетов этих скриптов предназначенная для личного использования под названием tinderbox. Разработчики портов часто используют её для тестирования своих изменений перед непосредственно коммитом в основное дерево. Функциональность этой системы аналогична скриптам PointyHat и она может быть использована для сборки пакетов на основе локально-модифицированного дерева портов. Всё это открывает колоссальные возможности для организации системы сборки/тестирования ПО хостингового кластера, как стороннего, так и разработанного локально.

4 Адаптация к задачам управления системой массового хостинга

Как это работает у нас? Во-первых, мы создали и подключили к сборке раздел `local` системы портов. В этой категории мы разместили собственное ПО, которое мы используем на хостинге. Далее, в систему портов были интегрированы все наши патчи к стороннему ПО путём простого добавления этих файлов к существующим портам. Кроме того, были подобраны значения KNOBS необходимые для удовлетворения наших потребностей. Там, где это было необходимо, были произведены необходимые модификации в системе портов, где-то добавлены новые KNOBS и т.д. Мы возвратили все улучшения, которые были бы полезны другим, назад в основное дерево FreeBSD, для того, чтобы снизить стоимость дальнейшей поддержки локальных модификаций. Далее были собраны и сконфигурированы jail'ы для всех используемых версий ОС и архитектур. Они были получены непосредственно сборкой из соответствующих каждой используемой ветке исходных текстов. К счастью, мы используем всего 2 архитектуры, поэтому для сборки удалось обойтись одной машиной. Для этого используется сервер с Core2Quad, работающий под управлением FreeBSD 8-CURRENT/amd64 с 4G ОЗУ. Для сборки пакетов под архитектуру i386 применяется эмулятор QEMU. Для ускорения сборки все временные директории, директория сборки и т.д. размещаются на `tmpfs`, хранящейся в памяти, кроме того применяется `ssache`. Время сборки всех наших пакетов для amd64 порядка 3,5 часов, для i386 - порядка 12 часов. Директория `ssache` занимает порядка 1Gb. Замечу, что эти пакеты отличаются от официальных FreeBSD помимо

локальных патчей ещё и тем, что сборка их происходит на нашей версии FreeBSD, где может отличаться ABI от релизной версии.

5 Интеграция с основным деревом FreeBSD

Важной задачей такой системы также является интеграция с основным деревом FreeBSD, так как необходимо достаточно быстро вносить изменения основной ветки в нашу ветку. Для облегчения этого процесса мы используем систему контроля версий Mercurial. Вообще говоря, она используется как основной внутренний репозиторий компании (наряду с CVS, который использовался ранее). Mercurial (или Hg) - это распределённая система контроля версий, которая построена на идее транзакций. Т.е. репозиторий Mercurial, вообще говоря, не содержит внутри себя никаких файлов, директорий и т.д. Это просто набор диффов к исходному его состоянию, когда он пуст. Определённый последовательный набор таких транзакций и образует из себя репозиторий, последняя наложенная на рабочую директорию транзакция по сути является версией исходного кода в этой рабочей директории. Распределённость Mercurial заключается в том, что в каком-то эти наборы диффов можно синхронизировать между репозиториями, таким образом вести разработку параллельно. Эта его особенность и помогает отслеживать изменения в стороннем дереве FreeBSD и интегрировать их в произвольном порядке в наше дерево. Для этих целей мы создали по копии репозитория для каждой ветки FreeBSD, которые мы используем, и для основного дерева портов FreeBSD. Эти репозитории постоянно по крону синхронизируются с основным деревом FreeBSD, по сути представляя из себя эталонные копии. Наши репозитории, на основе которых строятся наши образы ОС и пакеты, получают путём "стягивания" изменений с эталонных репозиториях. Так как Mercurial помнит, откуда пришло то, или иное изменение, при этом не возникает конфликтов, которые были бы неизбежны в случае классических VCS. Вообще говоря, в случае CVS вам бы пришлось проделывать всю процедуру интеграции вручную, чем мы долгое время развлекались в проекте FreeBSD. Обычно, мы помечаем какую-то версию своего дерева внутренним тегом релиза, и вносим в него только те изменения основного дерева, которые критичны для нас (например, security fixes). Таким образом, наши пользователи защищены от возможных сбоев при обычных обновлениях. Кроме того, это позволяет быстро вернуться к той, или иной версии. Масштабные обновления мы обычно производим во время процедуры подготовки к релизу дерева FreeBSD, так как в этот момент оно наиболее стабильно.

6 Результаты

Что это даёт в итоге? Во-первых, нужные изменения могут попасть на сервера достаточно быстро: в случае обычных security фиксов это время колеблется в пределах нескольких десятков минут (за это время успевают собраться все затронутые пакеты (нет необходимости пересобирать всё), и пакет может быть обновлён на всех серверах); во-вторых, мы можем достаточно легко поддерживать несколько различных конфигураций на десятках серверов без особых проблем, и быстро интегрировать все обновления в наше дерево автоматически (фактически, 2 версии ОС на 2-х архитектурах только для хостинга); в-третьих, мы можем использовать

встроенные в FreeBSD средства для работы с пакетами, утилиты для их установки, поиску уязвимостей (VuXML). VuXML - система аудита безопасности портов FreeBSD, и она позволяет получать информацию о всех уязвимых пакетах на системе в автоматическом режиме посредством специальных утилит.

7 Дальнейшие усовершенствования

Система не идеальна, и никогда не будет. Хотелось бы побольше автоматизации, систему проверки качества собранных релизов (сейчас проверяем на новых серверах вручную).