

Передовые технологии операционной системы Solaris

Краткий конспект лекций для включения в курсы “Операционные системы”, “Информатика”, “Основы информационных технологий”, “Программирование” и другие, где информация о современных технологиях и их конкретных реализациях может быть полезна.

Предлагается преподавателям для использования без ограничений: можно свободно копировать части данного документа или весь текст целиком, включать его в любые материалы, ссылаться или не ссылаться на источник – на усмотрение преподавателя. При использовании информации для подготовки лабораторных работ рекомендуется проверить версию программного обеспечения, установленного в лаборатории – данные материалы разработаны на основе опыта эксплуатации Solaris Express Developer Edition 5/07 и Solaris 10 11/06. Версии Solaris, более ранние, чем Solaris 10 11/06, могут не иметь части описанных особенностей.

Документ подготовил Филипп Торчинский, консультант по Solaris компании Sun Microsystems. Автор благодарит Сергея Пикалева, Виктора Латушкина, Леонида Леняшина, Марию Тишкову, Питера Карлссона (Peter Karlsson), Даниила Ландау, Макса Брунинга (Max Bruning) и Михаила Жирнова за участие в обсуждении материалов.

Лекция 1. Программное обеспечение с открытым кодом. Базовые сведения о Solaris.

Введение. Программное обеспечение с открытым кодом.

Программным обеспечением (ПО) с открытым кодом называется такое ПО, которое:

- свободно распространяется
- доступно в виде исходного кода (текста программ)
- разрешено модифицировать

Кроме этого, существует ПО, использующее открытые стандарты (т.е. стандарты, которые свободно распространяются и описание которых доступно). Это ПО может не иметь открытого кода, но при этом разработчикам другого ПО легко обеспечить взаимодействие с ним, потому что стандарты и протоколы обмена данными опубликованы и доступны всем.

Также различают бесплатное ПО, которое называют также свободным (по принципу распространения – свободно, без оплаты).

ПО с открытым кодом пользуется заслуженной популярностью во всем мире, так как оно:

- гарантирует независимость пользователя от производителей (исходный код остается у пользователя, даже если производитель перестанет поддерживать свой продукт);
- дает возможность каждому пользователю лично убедиться в отсутствии постороннего по отношению к заявленной функциональности или вредоносного кода;
- обладает большей гибкостью: пользователь может модифицировать исходный код для адаптации ПО к своим нуждам, как самостоятельно, так и поручив эту работу независимым разработчикам;
- дешевле обходится в эксплуатации: начальная стоимость равна нулю, стоимость поддержки обычно ниже, чем стоимость поддержки закрытого ПО, широкое открытое для общения сообщество пользователей позволяет тратить меньше денег на обучение и решение проблем.

В мире распространены самые разные программные пакеты, и одна из проблем тех, кто ими пользуется – совместимость форматов, в котором эти программные пакеты сохраняют данные. Современное ПО как с открытым, так и с закрытым кодом обычно разрабатывается

так, чтобы быть совместимым по форматам данных как с другими открытыми программными пакетами, так и с разработками компаний, продающих свои продукты с закрытым кодом. Например, пакет офисных программ StarOffice 8¹, функционально практически идентичный другим широко распространенным офисным пакетам, использует для хранения данных открытый стандарт, но допускает сохранение и импорт данных в закрытых форматах других офисных пакетов.

Существуют десятки тысяч программных пакетов с открытым кодом; однако большое влияние на распространение современных технологий оказывают, как правило, немногие из них. Среди них выделяются *операционные системы с открытым кодом*. Подавляющее большинство таких систем происходят от операционной системы UNIX, первая версия которой была разработана в 1969 году. Примером такой системы является Solaris. История этой системы началась в 1982 году, и ее код был открыт в июне 2005 года, поэтому системой с открытым кодом могут считаться версии Solaris 10, начиная с версии 11/06².

Дистрибутивы, основанные на открытом коде Solaris

Solaris работает на платформах SPARC, x86 и x64 – с процессорами от Sun, AMD и Intel, на компьютерах любого масштаба – ноутбуках, десктопах, рабочих станциях, серверах и в кластерах. Как и другие системы UNIX, Solaris в силу архитектурных особенностей потенциально менее подвержен вирусам, чем широко распространенные системы других архитектур. На практике подтвержденных случаев заражения Solaris 10 вирусами не зарегистрировано вообще.

Код Solaris 10 сейчас доступен для всех на сайте opensolaris.org. После того, как код ОС Solaris был открыт, и был анонсирован проект OpenSolaris, смысл которого – в усовершенствовании базовой системы, появилось несколько дистрибутивов, основанных на этом коде:

- Solaris Express Community Edition – самый свежий двоичный (без исходных кодов в поставке) дистрибутив для тех, кто хочет испробовать самые новые технологии, еще не включенные в другие дистрибутивы, и еще не опубликованные в сборнике кодов на opensolaris.org. Обновляется раз в две недели. Этот дистрибутив еще называют Solaris Nevada, build XX, указывая соответствующий номер выпуска. Работает на платформах SPARC, x86/x64. На его основе создается другой дистрибутив, Solaris Express Developer Edition;
- Solaris Express Developer Edition – тестируемый Sun Microsystems дистрибутив, распространяется для x86, однако может быть собран из компонент для SPARC. Выпускается раз в три месяца;
- BeleniX – дистрибутив, созданный с использованием исходного кода, открытого в рамках проекта OpenSolaris. Сейчас распространяется в виде загружаемой с CD системы (LiveCD), однако разработчики рассчитывают сделать из него полноценный дистрибутив с установкой на диск;
- marTux – первый не связанный с выпусками Solaris Express и сообществом разработчиков Solaris Express дистрибутив для SPARC;

1 StarOffice 8 – это ПО с закрытым кодом, использующее открытые стандарты. На данный момент продается за деньги, однако для образовательных и исследовательских организаций бесплатен для всех поддерживаемых ОС – Solaris, Linux и Windows. Кроме того, StarOffice 8 входит в дистрибутив Solaris Express Developer Edition, и поэтому оказывается бесплатным для всех пользователей этой ОС. Пользователи Windows могут его скачать и использовать бесплатно в составе Google Pack от Google.

2 В ОС Solaris версии нумеруются в виде ММ/YY, т.е. месяц/год, 11/06 означает выпуск, вышедший в ноябре 2006 года – прим. авт.

- Nexenta – распространяемая по лицензии GNU система, разработанная на основе открытого кода ядра Solaris. Nexenta объединяет ядро Solaris и разнообразные открытые приложения;
- SchilliX, основанный на коде Solaris дистрибутив на Live CD для x86/x64/EM64T.

Помимо упомянутых дистрибутивов, развиваемых широким сообществом разработчиков, как связанных с компанией Sun Microsystems, так и совершенно независимых от нее, надо отметить собственно Solaris (текущая новейшая версия на осень 2007 года – Solaris 10 update 4, также именуемая Solaris 10 08/07). Компания Sun внимательно изучает опыт разработчиков и пользователей Solaris Express и обеспечивает перенос (портирование) в очередные выпуски Solaris новых технологий и функциональностей, опробованных в Solaris Express.

Отличия Solaris от других операционных систем

Чем Solaris отличается от других систем, происходящих от UNIX? Ее удобнее администрировать, чем многие другие, в ней возможно делегировать пользователям только в действительности необходимые им права, что делает ее более безопасной, и, наконец, только в Solaris полностью реализованы несколько свежих технологий, изобретенных Sun Microsystems.

Коротко, вот основные преимущества Solaris перед другими операционными системами с открытым кодом:

- уникальный механизм виртуализации
- сверхнадежная транзакционная файловая система, легкая в управлении
- гибкое делегирование прав с помощью ролей (RBAC – role-based access control)
- легкое управление запуском и настройкой служб
- динамическая трассировка программ
- устойчивость к нагрузке - в частности, благодаря современному планировщику задач
- улучшенное быстродействие стека TCP/IP
- обратная совместимость по исполняемым кодам на 10 версий назад
- централизованная поддержка от производителя

А теперь разберем самые интересные преимущества подробнее.

Встроенный механизм виртуализации – возможность запустить в рамках одной операционной системы до 8192 виртуальных систем (так называемых контейнеров или “зон”). Каждая зона работает независимо от других, со своим собственным участком дискового пространства, своим уникальным IP-адресом, и даже своим собственным пользователем root. Зону можно ограничить в использовании оперативной памяти и процессорной мощности, что выгодно отличает зоны от похожих по назначению “клеток” (jails) во FreeBSD. Более того, в зонах специального типа можно запускать двоичные приложения для Linux, в том числе Oracle для Linux – без какой-либо модификации. Это достигнуто за счет измененного механизма обработки системных вызовов в ядре, позволяющего исполнять их таким же образом, каким они выполняются в другой системе.

Все зоны разделяют между собой одно ядро, работающее в главной, “глобальной” зоне, и это значительно снижает накладные расходы зон на планирование задач, делая это решение самым эффективным среди всех вариантов виртуализации серверов (а еще есть как минимум Xen, VMware и Qemu – с которыми Solaris 10 тоже полностью совместим).

Совершенно новая 128-разрядная транзакционная файловая система ZFS – по архитектуре и применяемым техническим решениям. В ней нет традиционных томов, равно и проблем с недостатком свободного места или нехваткой индексных дескрипторов. Благодаря транзакционному механизму даже при неожиданном отключении питания занудной проверки диска после старта не последует: все незаконченные транзакции тихо исчезнут. ZFS легко и логично администрируется, имеет новый, совместимый с NFSv4 набор прав доступа (схожий с применяемым в NTFS), в частности, наследование прав доступа от родительского каталога. Другие впечатляющие новинки – сквозное контрольное суммирование не только данных, но и метаданных, мгновенное создание “снимков” (образов всей файловой системы), незаменимое для резервного копирования “на ходу”, программное зеркалирование, RAID с переменным размером полос, возможность двойной избыточности программных RAID-массивов RAID-Z (этого обычно нет и в аппаратной реализации избыточности), и долгожданный рост максимального размера файлов и файлового хранилища: теперь максимальный размер файла равен 2^{64} байт.

В других системах UNIX для того, чтобы доверенный пользователь мог выполнить какую-то административную задачу (скажем, добавить пользователя), ему либо сообщают пароль root, либо он использует sudo. И то, и другое решение неидеально. Строгие правила безопасности и печальный опыт поколений требуют, чтобы пароль привилегированного пользователя был известен минимальному количеству людей (в идеале - одному человеку и еще записан на бумаге у него в сейфе). А sudo всего лишь позволяет исполнять определенные программы, в то время как по сути надо делегировать выполнение определенных обязанностей. Тонкая настройка прав доступа в Solaris позволяет делегировать обычным пользователям строго необходимые им права, например “добавлять принтер”, “выбирать сеть wi-fi для подключения”. Ничего лишнего: только то, что требуется. Добавляющий только принтеры не сможет добавить пользователя, а выбирающий сеть для подключения не сможет установить новый ключ для шифрования в протоколе WEP.

Архитектура Solaris препятствует созданию вирусов для этой системы, а надежная подсистема безопасности делает крайне маловероятным запуск вредоносных программ в Solaris. Четкое разделение прав между обычными и привилегированными пользователями гарантирует, что ошибочные действия пользователей не нанесут вреда системе в целом. В настоящее время вирусы в Solaris не выявлены, а специфические уязвимости быстро устраняются разработчиками.

Для быстрого, безопасного и автоматического обновления компонентов Solaris компания Sun Microsystems разработала Sun Online Connection – механизм обновлений для зарегистрированных пользователей. Для его использования необходимо зарегистрироваться на сайте, и разумеется его можно отключить, если загрузка обновлений стала невозможна (например, снизилась пропускная способность канала связи или приходится экономить деньги на трафике).

Уникальное средство управления службами (SMF – Service Management Facility) позволяет отслеживать зависимости служб друг от друга, отслеживать проблемы при их запуске и управлять запуском и остановом. Это средство унифицирует работу всех служб – как запускаемых с помощью init.d, так и независимо работающих серверов типа Apache. Это новшество значительно облегчает работу администратора. Такого средства пока нет ни в одной другой системе UNIX.

Для более опытных системных администраторов и разработчиков есть еще динамический трассировщик DTrace, позволяющий отследить цепочку системных вызовов любой программы или ядра, а также померять время, которое процессор потратил на выполнение

любой группы вызовов. Функциональность DTrace встроена в ядро и не требует перекомпиляции программ для отслеживания их кода.

Лекция 2. Многопоточность. Вариант Solaris.

Общие сведения о многопоточности

Единицей исполнения в любой операционной системе является процесс. Процесс – это совокупность исполняемого кода, данных, связанных с этим кодом, и информации, надобной для исполнения этого кода операционной системой. Существуют операционные системы, в которых после запуска одного процесса другой не может быть запущен до тех пор, пока не завершится первый. Такие ОС называются однозадачными и встречаются в современном мире редко. Большинство ОС – многозадачные, т.е. способны обеспечить одновременное исполнение нескольких процессов (как правило, нескольких десятков тысяч).

В многозадачной системе всегда есть планировщик задач – часть ядра ОС, которая передает управление от одного процесса другому в соответствии с алгоритмом переключения. Для определения процесса, которому надо передать управление, используется формула, в которую входит приоритет процесса – параметр, который придается любому процессу в системе.

Каждый процесс в большинстве ОС (в частности, Solaris, Linux, других системах UNIX и в современных версиях ОС Windows) выполняется в своем собственном адресном пространстве. Процесс не может обратиться к адресному пространству другого процесса, за исключением специального случая использования разделяемой памяти – механизма межпроцессного взаимодействия, который обеспечивается в некоторых ОС. Это сделано для того, чтобы сбои в работе одного процесса не могли повлиять на остальные процессы в системе. Процессы в многозадачной многопользовательской системе могут принадлежать разным пользователям и выполняются независимо друг от друга.

Когда процесс ждет завершения внешнего по отношению к нему события (обработки прерывания, ввода-вывода и пр.) возникает блокировка – код процесса не исполняется, а сам процесс находится в состоянии ожидания. Процесс может состоять из нескольких логических частей. Скажем, моделирование движения молекул может состоять из вычисления их координат в каждый момент времени, записи этой информации на диск и визуализации на экране. В этом случае приостановка вычислений на время записи информации на диск нежелательна.

На помощь приходит механизм многопоточности: внутри одного процесса создается несколько потоков. Поток (thread) – это тоже исполняемый код, как и процесс, но в отличие от процесса, поток разделяет свое адресное пространство с другими потоками этого же процесса. При программировании многопоточного приложения разработчику приходится самому решать, как именно он будет синхронизировать выполнение нескольких потоков внутри своей программы и как обеспечивать доступ к совместно используемой памяти.

Современные средства разработки предоставляют несколько механизмов синхронизации, например, исключения (mutex), блокировки, семафоры. В Solaris для многопоточного программирования можно использовать библиотеку POSIX threads (libpthreads) или разработанную ранее компанией Sun библиотеку libthreads (она сохраняется в Solaris для совместимости с предыдущими версиями).

Кроме удобного распараллеливания работ внутри одного процесса механизм потоков позволяет повысить производительность: управление большим количеством потоков для ядра проще, чем управление таким же количеством процессов, так как при переключении от

одного процесса к другому ядро должно выполнить больше операций, чем при переключении от потока к потоку.

В компьютерах с несколькими процессорами (а также с многоядерными процессорами, некоторые из которых обрабатывают несколько потоков команд одновременно, например, UltraSPARC T2 от Sun – 64 потока) многопоточность позволяет исполнять разные потоки на разных процессорах, что во многих случаях дает значительный прирост производительности.

Реализация многопоточности в Solaris

Задача обеспечения многопоточности сводится к созданию оптимального алгоритма управления потоками. С точки зрения ядра Solaris, поток – это единица диспетчеризации, т.е. именно потоки конкурируют между собой за доступ к ресурсам (прежде всего, к процессору).

Диспетчер (также его иногда называют планировщиком) – это часть ядра, ответственная за переключение между потоками, выполняемое в соответствии с приоритетами потоков.

В Solaris наряду с процессом и потоком вводятся понятия потока пользователя (user thread), потока ядра (kernel thread) и легковесного процесса (LWP, lightweight process). Поток ядра в Solaris – это как раз и есть единица диспетчеризации. Поток ядра описывается в ядре небольшой по размеру структурой (“структурой” в понимании языка C, т.е. набором переменных). Поток ядра может быть связан с легковесным потоком, который, в свою очередь, всегда связан с одним потоком пользователя. Кроме этого, поток ядра может быть инициирован самим ядром, и тогда с ним не связан никакой легковесный процесс. Рисунок 1 показывает соотношение потока пользователя, потока ядра и легковесного процесса. На рисунке также указано, как называются структуры, описывающие процесс, поток пользователя, легковесный процесс и поток ядра соответственно.

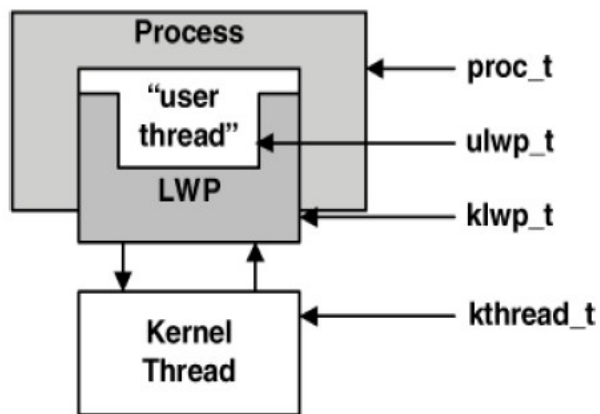


Рис. 1. Соотношение между потоками пользователя и потоками ядра в Solaris 10

Такая организация многопоточности введена в Solaris начиная с версии Solaris 10, потому что на современном оборудовании она более выигрышна, чем использованная в предыдущих версиях Solaris. Ранее к одному легковесному процессу могло привязываться несколько потоков пользователя (рисунок 2), что усложняло управление процессами на уровне библиотеки потоков libthreads и требовало отдельного планировщика потоков пользователя на уровне процесса. Эта схема была эффективной в течение многих лет, так как затраты процессорного времени на создание нового потока ядра были достаточно высоки. С ростом мощности процессоров, появлением многоядерных процессоров, удешевлением компонент в целом оптимизация управления потоками стала более важной, чем экономия на создании новых потоков ядра.

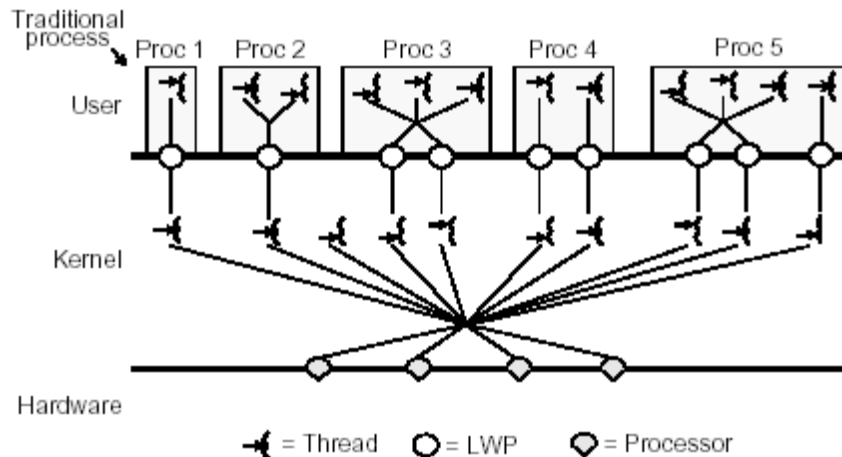


Рис. 2. Соотношение между потоками пользователя и потоками ядра в Solaris до версии 9 включительно

Лекция 3. Виртуализация операционных систем

Общие сведения о виртуализации

Под виртуализацией понимают возможность работы нескольких экземпляров операционных систем на одном физическом компьютере одновременно. Так, программное обеспечение компании VMware давно уже позволяет установить и запустить на одном и том же компьютере несколько операционных систем. Однако, это не очень экономно: ресурсы тратятся на хост-систему (т.е. ту, в которой установлено ПО VMware), а также на переключение между несколькими различными операционными системами. Кроме этого, существуют еще и накладные расходы на диспетчеризацию потоков внутри каждой из операционных систем.

Виртуализация необходима, когда надо на одном и том же компьютере проверить работу какого-то приложения в разных системах, или когда вместо двух нагруженных наполовину серверов вы сочли эффективным установить один сервер, чтобы он был загружен на 100%. Кроме этого, если вы просто хотите посмотреть на работу новой для вас системы, ее удобнее установить в качестве одной из нескольких в среде отдельной виртуальной машины, чем делать множественную загрузку (когда при старте компьютера вы вручную выбираете, какую систему запускать).

Кроме этого, виртуализация позволяет отделить одну запущенную систему от другой так, чтобы они не имели возможности влиять друг на друга. Возможность обмениваться данными сохраняется: две виртуальные машины могут это делать через сеть – так, как если бы они работали на двух разных компьютерах, подключенных к сети.

Особенности виртуализации в Solaris

В Solaris реализована уникальная схема виртуализации, которая пока нигде больше не применяется: внутри одного экземпляра операционной системы можно запустить множество неглобальных зон (т.е. “виртуальных” экземпляров системы). Первичный экземпляр Solaris, стартующий при включении компьютера, называется глобальной зоной, внутри нее располагаются неглобальные зоны. Таким образом, в пределах одного физического компьютера можно запустить до 8192 независимых друг от друга экземпляров Solaris.

Важно, что все зоны разделяют одно общее ядро, а сами друг друга “не видят”. Обработка системных вызовов приложений из всех зон одним ядром значительно снижает накладные расходы на переключение между разными виртуальными системами, по сравнению с VMware.

Кроме приложений Solaris в отдельной специальной Linux-зоне можно выполнять приложения Linux – без всякой модификации. В ней поддерживаются приложения, исполняемые файлы которых были скомпилированы и собраны для Red Hat Linux 3.5 или CentOS 3.0. Важно, чтобы приложение требовало версию ядра Linux не выше 2.4 и разделяемые библиотеки тех версий, которые присутствовали в указанных системах Linux. Известно, что авторами sunhelp.ru была реализована установка Debian Linux в Linux-зону Solaris.

Для исполнения приложений Linux в Linux-зоне Solaris следует прежде создать и настроить эту зону – зону специального типа (“маркированную” или “типизированную” зону, также называемую BrandZ). После этого зону следует “установить”, используя образ файловой системы “родной” системы Linux (например, Red Hat 3.5). Образ можно создать самому или скачать со страницы <http://opensolaris.org/os/community/brandz/>

Идея создания “маркированных” зон в будущем теоретически позволяет виртуализировать внутри Solaris различные операционные системы. На практике кроме Linux-зон уже реализован проект Etude, который сделал возможным запуск приложений Solaris 8 в отдельной зоне. Идея проекта состоит в том, чтобы выполнить миграцию масштабного приложения или их группы с Solaris 8 без всяких настроек, посредством простого переноса всей файловой системы старого сервера, работающего под управлением Solaris 8 в типизированную зону на новом сервере, работающем уже под Solaris 10.

Зонная (или, что то же самое, контейнерная) схема виртуализации позволяет совершенно изолировать приложения друг от друга, запуская их в разных зонах. Это дает возможность:

- 1) создать идеальную лабораторную среду – например, для каждого выполняющего лабораторную работу выделить его собственную виртуальную систему и дать ему все права на управление этой системой
- 2) разделить между собой приложения разных пользователей – например, хостинг-провайдер может каждому клиенту дать полный доступ к его – клиента – собственной виртуальной системе, включая доступ по ssh.
- 3) ограничить нагрузку на сервер, которую создает приложение или группа приложений, например, поместить СУБД в одну зону, а веб-сервер – в другую, и сбалансировать нагрузку на память и процессор для зон, указав предельно допустимые значения для каждой из них.

Для создания зон и управления ими следует использовать команды `zoneadm` или `zonectl`, а для того, чтобы “войти в зону” (подобно тому, как мы соединяемся с удаленным компьютером по ssh) можно либо с помощью команды `zlogin`, либо (если в зоне настроен сетевой интерфейс) с помощью `ssh`.

При создании неглобальной зоны вначале выполняется настройка параметров зоны – с помощью команды `zonectl`, как показано ниже. После этого выполняется установка зоны – командой `zoneadm -z <имя зоны> install`. При настройке можно указать, следует ли устанавливать все файлы зоны с “наследованием” части каталогов от глобальной зоны. По умолчанию установка производится с наследованием (так называемый режим `sparse root`), что позволяет экономить место. Настройки зоны легко посмотреть в файле `/etc/zones/<имя зоны>.xml`. При установке без наследования (`whole root`) внутрь зоны будут копироваться все системные каталоги. По умолчанию в режиме `sparse root` не копируются `/lib`, `/platform`, `/sbin` и `/usr`, поэтому общий объем системных файлов внутри зоны (если после установки системы

не установлены дополнительные пакеты) составляет примерно 100 Мб. Для установки зоны в режиме whole root надо вместо подкоманды create команды zonecfg дать команду create -b, или просто отредактировать файл /etc/zones/<имя зоны>.xml так, чтобы в нем не были указаны каталоги с параметром inherited-pkg-dir.

Рассмотрим пример создания двух неглобальных зон.

```
# zonecfg -z testzone
testzone: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:testzone> create
zonecfg:testzone> set zonepath=/export/home/testzone
zonecfg:testzone> add net
zonecfg:testzone:net> set address=172.16.2.1
zonecfg:testzone:net> set physical=bge0
zonecfg:testzone:net> end
zonecfg:testzone> commit
zonecfg:testzone> exit
# zoneadm -z testzone install
Preparing to install zone <testzone>.
Creating list of files to copy from the global zone.
Copying <28831> files to the zone.
Initializing zone product registry.
Determining zone package initialization order.
Preparing to initialize <1115> packages on the zone.
Initialized <1115> packages on zone.
Zone <testzone> is initialized.
Installation of these packages generated warnings: <ITstaroffice
SUNWcsu SUNWcsr SUNWsshdr SUNWhea>
The file
</export/home/testzone/root/var/sadm/system/logs/install_log>
contains a log of the zone installation.
```

В зависимости от количества установленных в глобальной зоне пакетов, изменяется и размер файловой системы зоны. Новые пакеты можно установить в какую-либо одну зону (как глобальную, так и неглобальную) или во все зоны одновременно. Как именно будет установлен пакет, зависит от следующего:

- указан ли параметр -G при запуске pkgadd
- установлены ли переменные SUNW_PKG_ALLZONES и SUNW_PKG_HOLLOW в файле информации о пакете (pkginfo – смотри man pkginfo)
- в какой зоне запущена программа pkgadd – в глобальной или неглобальной

Важно: после установки зоны в нее надо первый раз зайти с помощью команды `zlogin -C <имя зоны>`, так как в этот момент будет настроен механизм SMF в этой зоне (так же, как происходит при первой загрузке только что установленной системы).

Лекция 4. Транзакционная файловая система ZFS: основные принципы

Революционным прорывом в мир файловых систем принципиально нового уровня стала файловая система ZFS, которая поддерживается в Solaris 10 (начиная с update 2) и современных выпусках Solaris Express. Кроме этих систем, поддержка ZFS портирована во FreeBSD (с некоторыми ограничениями), и декларировано намерение компании Apple включить ее поддержку в следующие версии Mac OS X (начиная с версии Leopard).

Традиционные файловые системы, которые используют в UNIX (например, UFS или ext3) имеют ряд недостатков, которые стали особенно заметны с ростом количества и размера дисковых накопителей. К ним относятся:

- относительная трудность администрирования (необходимость разбивать диск на разделы, настраивать монтирование этих разделов, управлять доступом и квотами каждого из них, расширять файловую систему и т.п.)
- ограничение по максимальному размеру файлов и разделов, которое становится препятствием с ростом объема накопленных данных
- сложность и недостаточная надежность резервирования, сложность резервного копирования
- сложность управления массивом дисков из нескольких десятков, сотен или тем более тысяч штук (а в серьезных системах это совершенно реальный объем – скажем, для визуального моделирования видеofilмов или процессов реального мира)
- значительные затраты времени на проверку и восстановление данных (например, при аварийной перезагрузке файлового сервера в любой системе UNIX без ZFS).

Поэтому ZFS проектировалась прежде всего для того, чтобы повысить надежность хранения данных и облегчить управление дисковой подсистемой.

Какие принципы положены в основу ZFS

Основных принципов ZFS три:

1. объединение всего доступного дискового пространства в пул
2. сквозной контроль целостности данных
3. транзакционность

Все диски, доступные системе, объединяются в единый пул, и уже из него по запросу выделяется пространство для отдельных файловых систем. Такой же принцип положен в основу виртуальной памяти – когда приложение запрашивает у операционной системы один мегабайт памяти для размещения данных, ему безразлично, в какой области памяти для него будет выделен этот мегабайт. В ZFS при необходимости расширить пул в него просто добавляется новый диск – одной командой. Необходимости объединять разделы дисков в тома больше нет.

Для каждого блока данных и метаданных (метаданные – это служебная информация, которая описывает собственно данные: их расположение на диске, свойства файлов и других объектов в системе, права доступа к ним и пр.) выполняется контрольное суммирование,

причем контрольная сумма блока хранится в родительском блоке метаданных – чтобы физически разделить места хранения информации и контрольной суммы для этой информации. Еще несколько лет назад такой сквозной контроль данных посчитали бы избыточным и дорогостоящим (в смысле вычислительных ресурсов). Сегодня, когда скорость процессоров по сравнению со скоростью дисков возросла на порядки, оказалось, что это совсем недорого. Используемые в ZFS алгоритмы на современном процессоре класса Opteron позволяют вычислять контрольные суммы со скоростью 2—8 Гбайт/с.

В файловой системе ZFS данные всегда хранятся в целостном и непротиворечивом состоянии, так как запись на диск выполняется группами транзакций. Уже записанные на диск данные и метаданные при их изменении не перезаписываются, а пишутся в свободное пространство диска (такой метод записи обозначают термином “копирование при записи, сору-оп-write). По окончании группы транзакций происходит атомарная операция – запись нового uber-блока (главного блока файловой системы) – как показано на рис. 3. Это подтверждает завершение группы транзакций. С этого момента блоки, которые до начала группы транзакций были заняты данными, изменяющимися в ходе транзакции, считаются

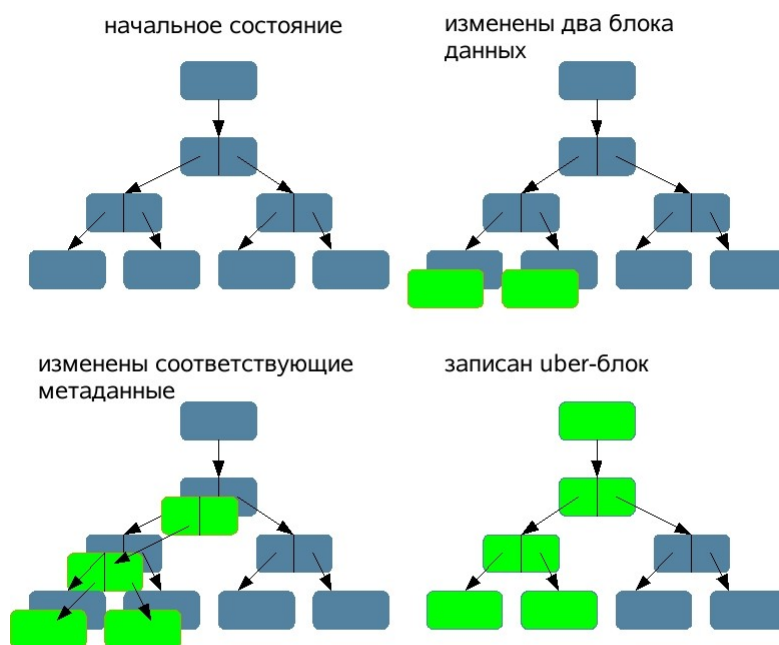


Рис 3. Порядок записи изменяемых данных и метаданных в ZFS.

освобожденными. Если в ходе транзакции произошел сбой (например, отключилось питание), диск после восстановления питания оказывается в том же состоянии, в котором был до начала транзакции.

Такая организация файловой системы не только повышает надежность, но и увеличивает ее производительность, так как низкоуровневый драйвер файловой системы теперь сам заботится о порядке записи данных на диск в ходе транзакции и способен его оптимизировать.

Средства обеспечения надежности ZFS

Контрольные суммы

Одно из отличий ZFS от других файловых систем состоит в том, что 256-битные

контрольные суммы блоков сохраняются не вместе с блоками, а в блоке адресации верхнего уровня, т.е. вместе с адресом блока хранится контрольная сумма адресуемого блока данных. Контрольные суммы считаются по любому объекту, не только по блоку данных, но и по блоку адресации, и если в блоке адресации из-за сбоя при хранении оказался неверный адрес, то система сама определит, что адрес неверный и ошибочные данные просто не будут выданы по запросу.

Таким образом, даже если никакого резервирования вы не предусмотрели, ZFS с большей вероятностью, чем традиционные файловые системы, сможет определить при считывании данных, что они повредились между моментами записи и считывания, и вы избежите получения заведомо неверной информации. Обычно этого мало: хотелось бы считывать в точности те данные, что были записаны – независимо от проблем с дисками, контроллерами и драйверами.

Хранение данных с избыточностью (RAID-Z)

Для этого в ZFS предусмотрена реализация RAID-Z – это организация программного RAID-массива с блоками переменной длины. Переменная длина дает значительный выигрыш в производительности, но меняет алгоритм восстановления данных, поскольку каждый блок в массиве может иметь разную длину. Кроме этого, в отличие от традиционных массивов RAID, в ZFS можно организовать не только однократный, но и двукратный контроль четности, и это защищает от одновременного сбоя двух физических носителей.

Легко видеть, что ZFS в самом деле гарантирует сохранность данных, если у вас достаточно надежное оборудование – по крайней мере один диск в запасе. Кстати, ZFS поддерживает и диски “горячего резерва” (spare disks), на которые ничего не записывается до момента, когда один из рабочих дисков RAID-Z выйдет из строя. Зато как только понадобится, сбойный диск будет исключен из пула дисков, а диск горячего резерва немедленно включен туда.

Для поддержки зеркалирования в ZFS реализована процедура ресинхронизации (resilvering) после замены сбойного диска исправным. Процедура выполняется автоматически.

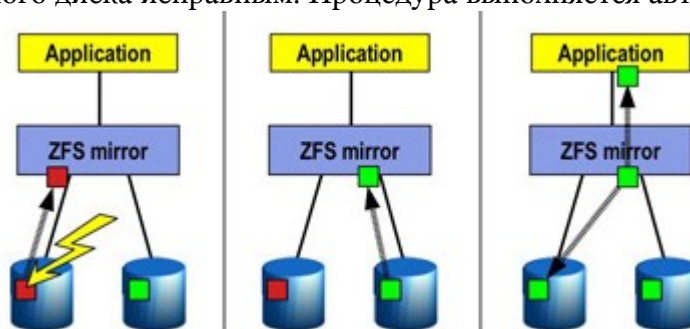


Рис. 4. Самовосстановление данных в ZFS

Если же диск не дает сбоев в целом, но какая-то часть данных оказалась повреждена (т.е. контрольные суммы не соответствуют данным), то происходит автоматическая запись на сбойный диск данных с “правильного” диска, как показано на рис. 4. Ошибочный блок данных показан красным цветом, корректный – зеленым. Вначале при выполнении чтения обнаруживается ошибочный блок (расхождение с контрольной суммой), приложению передается корректный блок данных с другого диска в зеркале, и затем сбойный блок на первом диске замещается корректным.

Резервирование метаданных

Все метаданные в ZFS пишутся на диск в несколько разных мест, для большей отказоустойчивости. В зависимости от ценности метаданных, делается три или – как минимум – две копии. В отличие от UFS, где на диск записывается несколько копий только суперблока, здесь делаются копии всех метаданных, причем если в пуле несколько устройств, ZFS старается разнести копии метаданных по разным устройствам.

Таким образом, для надежного хранения данных в ZFS предусмотрены:

- транзакционность
- контрольные суммы в метаданных
- программный RAID (RAID-Z), типов “зеркало”, “RAID с однократной четностью”, “RAID с двукратной четностью”
- диски горячего резерва для RAID-Z
- “самовосстановление” данных, когда они записываются с избыточностью (т.е. с контролем четности или зеркалированием).
- Тройное резервирование метаданных

Масштабируемость ZFS

Современные 64-разрядные файловые системы рассчитаны на хранение большого объема информации, но уже подсчитано, что при существующей динамике накопления данных их ресурс будет исчерпан в течение 10-15 лет. ZFS – это первая в мире 128-разрядная файловая система. Максимальный размер файла, который можно сохранить в ZFS – 2^{64} байт, т.е. 16 экзбайт. Это в 18 миллиардов миллиардов ($18,4 \times 10^{18}$) раз больше, чем позволяют современные 64-битные файловые системы. Максимальный размер одной файловой системы ZFS такой же - 2^{64} байт. Если создавать по 1000 файлов в секунду, то для достижения максимального количества файлов в ZFS (2^{48} штук) понадобится примерно 9000 лет. Нетрудно видеть, что ZFS действительно создавалась так, чтобы избежать скорого достижения лимитов: автор проекта ZFS Джеф Бонвик (Jeff Bonwick) подсчитал, что на 100%-е заполнение одной файловой системы ZFS потребуется больше энергии, чем для того, чтобы вскипятить весь Мировой океан!

Благодаря тому, что все объекты в ZFS создаются динамически (включая все возможные метаданные), знакомых по прежним файловым системам ограничений, связанных с количеством индексных дескрипторов, в ZFS нет: существует всего одно ограничение – физический объем диска, и это ограничение легко преодолеть, просто добавив еще один диск в пул.

Удобство администрирования ZFS

ZFS поддерживает создание снимков (мгновенных копий файловой системы), и благодаря этому делает возможным резервное копирование всей файловой системы “на лету”, даже во время интенсивной работы пользователей с файловой системой. Кроме снимков возможно создавать “клоны” файловых систем – их снимки, доступные для записи. Управление всеми возможностями ZFS реализовано через команды `zfs` и `zpool`.

В ZFS поддерживается назначение прав доступа по стандарту NFSv4, подобное тому, которое принято в файловой системе NTFS, когда тот или иной уровень доступа к объекту может быть назначен многим группам, и права доступа наследуются дочерними файлами и подкаталогами текущего каталога.

Дополнительную информацию о ZFS можно получить на веб-сайтах developers.sun.ru,

opensolaris.org и (на русском языке) в статье по адресу http://pcmag.ru/solutions/detail_print.php?ID=9141&print=Y

Лекция 5. Инструменты Sun Studio для разработки программ в Solaris и Linux

Современному разработчику программ для работы необходим достаточно большой набор инструментов: удобное средство работы с проектами, включающими много файлов, удобный текстовый редактор, компиляторы с разных языков, отладчик, профилировщик, компоновщик, то есть не только средства разработки, но и мощные средства эффективного выявления ошибок и проблем – таких, как ошибки выделения памяти, ошибки синхронизации, доступа к ресурсам и т.д. – и все это желательно иметь возможность запускать как из командной строки в простом текстовом интерфейсе, так и с использованием

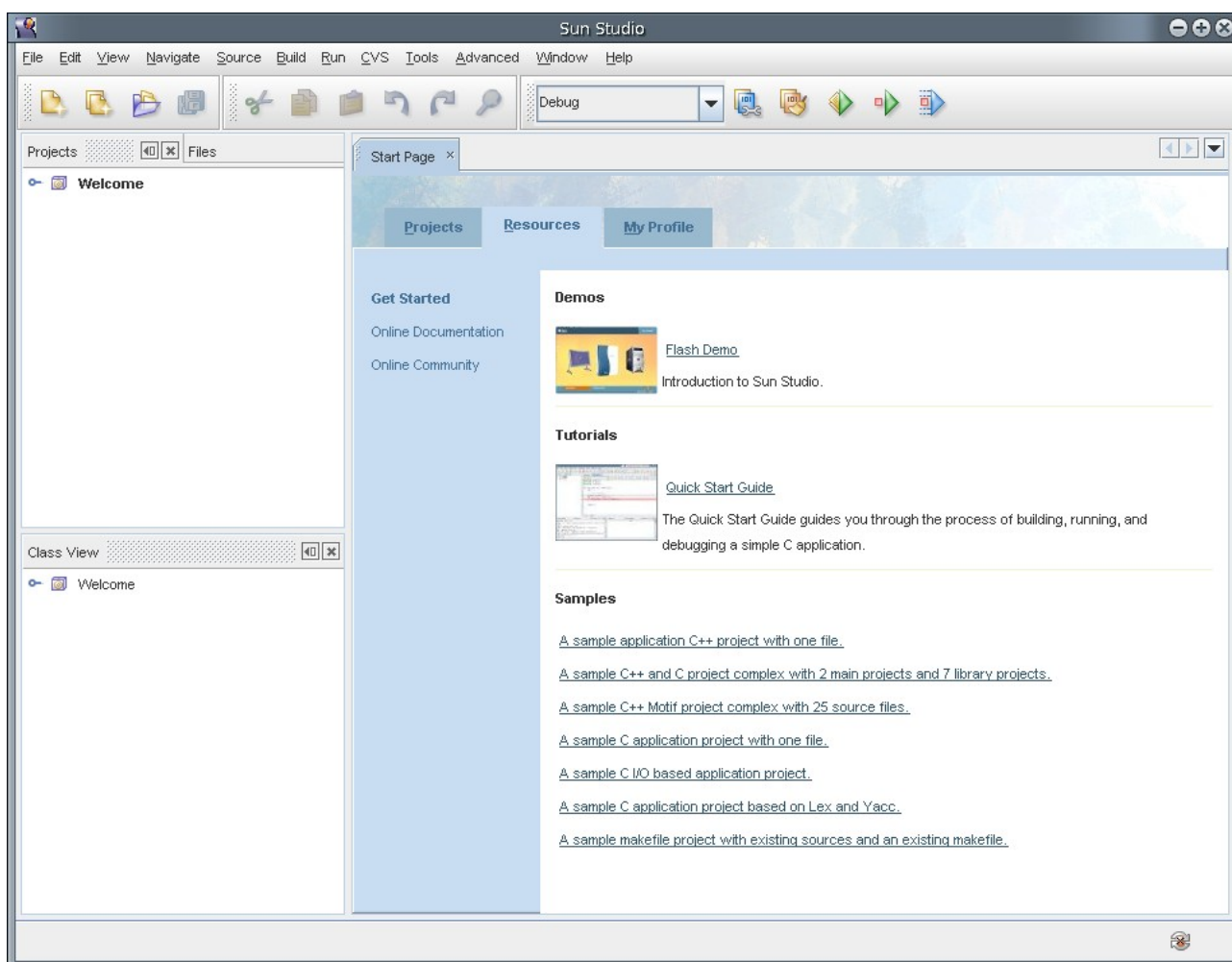


Рис. 5. Экран приветствия Sun Studio 12

графической интегрированной среды.

В старых версиях UNIX таких средств просто не существовало, одной из первых стала интегрированная среда программирования в Digital UNIX. К сожалению, эта среда была забыта вместе с самой ОС Digital после разделения и продажи компании. В настоящее время в мире UNIX есть несколько интегрированных сред, однако Sun Studio (по состоянию на осень 2007 года – текущая версия Sun Studio 12) среди них выделяется уникальной функциональностью, не говоря уже о включенных в среду фирменных компиляторах C/C++/FORTRAN от Sun. Пакет Sun Studio бесплатен и доступен для Linux и Solaris, он

входит в дистрибутив Solaris Express Developer Edition или может быть загружен с сайта developers.sun.ru. Дистрибутив Sun Studio 12 занимает примерно 275 Мб (в зависимости от версии и платформы). В ряде случаев компиляторы пакета (благодаря встроенному в Sun Studio оптимизатору кода) создают код, который в разы быстрее кода, созданного компилятором gcc.

На первый взгляд IDE Sun Studio очень напоминает Microsoft Visual Studio или Eclipse.

Интегрированная среда разработки построена на фундаменте NetBeans 5.5 – популярной бесплатной IDE для Java с открытым кодом. Sun Studio поставляется с компиляторами, утилитой `dmake` и отладчиком `dbx` от Sun, имеет встроенный профилировщик, анализатор потоков для многопоточных приложений и утилиту контроля памяти времени выполнения.

К уникальным функциям Sun Studio относятся отладчик, позволяющий присоединяться к отдельному потоку в процессе, анализатор потоков на предмет потенциальных конфликтов (`data race detection tool`). В состав IDE Sun Studio входит обозреватель классов C++. Мощный встроенный редактор кода с поддержкой языков поддерживает автодополнение кода, свертывание кода, переход к объявлению, маркировку синтаксиса, форматирование отступами. Быстрый синтаксический анализатор исходного кода не требует компиляции

Поддержка проектов на C/C++ включает в себя:

- поддержку различных типов файлов, используемых при разработке C и C++ приложений, таких как файлы сборки (`makefile`), файлы заголовков, `shell`-скрипты и файлы исходных кодов
- поддержку работы с проектами на основе файлов сборки (`makefile`)
- наличие шаблонов для типовых проектов, позволяющих разработчикам сосредоточиться на содержательной работе, а именно:
 - проект на C/C++ с существующим кодом
 - C/C++ приложение
 - динамическая библиотека
 - статическая библиотека

Кроме этого, среда Sun Studio поддерживает работу с файлами сборки (`Makefile`):

- мастер проектов допускает импорт существующих файлов сборки для создания новых проектов
- мастер файлов сборки позволяет указывать компиляторы, определения для препроцессора, параметры компиляции и т.п.
- для большинства типов проектов файлы сборки создаются автоматически
- свойства проекта позволяют определять цели и настройки и управлять ими

Отладчик пакета Sun Studio поддерживает как родной формат отладочной информации `dbx`, так и формат `gdb`; объединен со встроенным редактором, поддерживает несколько сеансов одновременно, поддерживает отладку по файлу дампа памяти (`core`) и позволяет подключиться к выполняющейся программе/процессу; `dbx` умеет одновременно отлаживать программы на Java, C/C++ и Fortran, что делает его незаменимым при отладке Java-кода с `native`-библиотеками.