Константин Лепихов

Полный realtime

В наше время большинство домашних компьютеров применяется совсем не в бытовых целях — уже стало обычным делом производить дома оцифровку видео или музыки в реальном масштабе времени (realtime), а не идти для этого в специализированную студию.

Это стало возможным в связи с увеличением вычислительных мощностей современных компьютеров и возможностей новых версий ОС. Например, ядро Linux версии 2.6 уже имеет поддержку так называемого soft-realtime, то есть работы в режиме вытесняющего, или приоритетного (preemptive), планирования, что помогает достичь хороших результатов при обслуживании большого количества процессов или во время выполнения активных дисковых операций. Под результатами подразумеваются такие характеристики как отклик (response) ядра на какой-либо системный вызов или задержка (latency) в его работе, связанная с нагрузкой или реализацией алгоритма. Более того, существует ряд патчей, которые позволяют ядру работать в режиме так называемого hard-realtime, то есть с гарантированным откликом системы при любой загрузке. Для простоты восприятия будем называть soft-realtime «легким» realtime, а hard — «тяжелым». И пусть вас не пугает частота употребления слова «патч» — ведь на них держится вся POSIX/Unix-система, а мы говорим здесь о его сердце, то есть ядре.

«Легкий» realtime

Патчи от Con Kolivas

Это наиболее популярные патчи, которые используются большинством дистрибутивов (как минимум это Debian, Gentoo, SUSE, Fedora, ALT Linux) в своих как бы realtime-ядрах. Еще их очень любят всевозможные «ядроделатели», например nitro sources или CKO patchset. Патчи -ck позволяют добиться большей «отзывчивости» ядра при работе со значительными объемами вычислений, передаче данных или активной деятельности системы ввода/вывода.

Их отличительные особенности:

- Полностью изменена работа планировщика ядра, теперь можно более гибко регулировать, что и как планировать и с каким приоритетом.
- Добавлен новый вид планирования — Batch (idle) scheduling, позволяющий перепланировать процессы таким образом, что они выполняются только во время простоя (idle) системы. Это очень удобно для таких задач как компиляция или длительных операций наподобие apt-get dist-upgrade. Грубо говоря, процесс выполняется «в фоне», не мешая другим.
- Добавлен новый вид планировщика — SCHED_ISO, позволяющий добиться soft-realtime, то есть возможностей работы приложения, приближенных к realtime (на самом деле там введено ограничение на загрузку процессора в режиме realtime). Этот планировщик доступен не только привилегированным процессам, но и обычным (non-root) приложениям.
- Интересная вещь — swap prefetch. Вместо бездумного сбрасывания страниц памяти в swap создает связанный список того, что уходит в файл подкачки, и, если система находится в состо-

нии простоя, возвращает эти страницы в память в обратном порядке. Главная идея такого поведения — если программа не использовалась долгое время и была переведена в swap, то вам не придется ждать ее долгого пробуждения с диска.

Патчи от MontaVista

Один из первых патчей, превращающих ядро действительно в realtime-preemptive (то есть с вытесняющей многозадачностью, приближенной к realtime). Он разработан Sven-Thorsten Dietrich на базе патча, написанного Инго Молнером (о нем мы расскажем чуть позже). Главная идея — запросы на прерывание помещаются в отдельные нити исполнения (threads), которые борются за процессор наравне с основными процессами. Как только такое прерывание добивается своей цели, его нить исполнения становится вытесняющей, то есть может быть вытеснена другим процессом, с большим приоритетом, или по истечении времени занятия ресурса, что позволяет обслужить больше прерываний, чем при обычном монопольном блокировании/разблокировании ядра.

Основная причина всех задержек в ядре — это критические секции в коде, защищенные механизмами блокирования (spinlocks). Поэтому рождается резонный вопрос: а почему не сделать и эти секции вытесняющими? Для ядер 2.4 уже есть реализация — это PMutex, разработанный группой исследователей из Мюнхенского университета, который и был взят за основу для ядра 2.6. Конечно, сразу же появились проблемы реализации — в версии PMutex был применен простейший механизм наследования приоритетов: как только процесс устанавливал флаг блокировки (mutex), его приоритет увеличивался, и он мог быстрее выполнить свою задачу и освободить mutex. Вместе с другими улучшениями это позволяло избежать многих проблем с искажением приоритетов при вытесняющем планировании. В версии от MontaVista был применен другой подход — все существующие механизмы блокирования были заменены на новые флаги блокировки (этот механизм был похож на существующую реализацию семафоров в ядре). То есть все критические секции в ядре стали вытесняющими. К тому же, перенеся все запросы на прерывание в отдельные нити исполнения, решили еще одну проблему: теперь не возникает тупиковых ситуаций, связанных с прерываниями.

К сожалению, не обошлось без ложки дегтя. Теперь надо пересмотреть весь код ядра, где используются блокировки, и проверить его работоспособность (если что-то не работает, вернуть семафоры и spinlock на место). К тому же не все компоненты ядра должны быть вытесняющими — например, сложно представить себе вытесняемый планировщик ядра. Но разработчики MontaVista дали хороший старт всем проектам soft-realtime, доказав, что ядро Linux вполне можно сделать таким, каким нужно.

Патчи от Инго Молнера

Успех MontaVista заинтересовал Инго Молнера, одного из известных «ядерных хакеров». История гласит, что он пропал на два дня и потом выложил свою реализацию, названную впоследствии voluntary preemption, а затем realtime preemption. Инго согласился с основной идеей патча от MontaVista, но с некоторыми изменениями:

- PMutex не применялся. Вместо него Инго использовал существующий механизм семафоров в ядре. Как он сам говорил впоследствии, объясняя свое решение, семафоры работают на всех архитектурах, а PMutex — только на x86. Было бы здорово реализовать наследование приоритетов через семафоры, тогда это будет доступно и решениям на базе семафоров, и решениям, переписанным из spinlock. Однако в реализации voluntary preemption этого так и не было сделано.
- Вместо переписывания кода на новый механизм блокировки был предложен интересный вариант — функции блокировки не изменялись, менялась только модель их поведения: теперь они могли быть обычными блокировками, или «хитрыми» семафорами.
- Инго выяснил, что довольно много блокировок должны использовать старый механизм, а не новые «полиморфные» блокировки. В его патче сохранилось около 90 «старых» блокировок (в версии MontaVista — 30), но благодаря «полимор-

Проект RTnetwork

Сетевой realtime

RTnetwork (www.rtnet.org) — это попытка решить проблемы задержек и коллизий в сети Ethernet при передаче данных с помощью сетевого стека, специально разработанного для условий строгого реального времени. Отличительные особенности:

- **реализация Open Source, основанная на открытых протоколах и стандартах;**
- **используется свой сетевой стек, отличный от стандартной реализации TCP/IP, интегрирован в инфраструктуру ядра Linux;**
- **используется Ethernet как транспортная технология;**
- **используется TDMA (Time Division Multiple Access) — улучшенный протокол для контроля передачи данных;**
- **используется специализированный заголовок для TDMA-передачи — RTmac;**
- **реализовано как расширение для RTAI (Real-Time Application Interface).**

Интеграция RTnetwork с FireWire

FireWire, также известная как IEEE-1397, — это высокопроизводительная последовательная шина для соединения разобщенных (heterogeneous) устройств. Предназначенная в основном для передачи данных в бытовой электронике (например, высокоскоростная передача видеоданных), она может быть с успехом применена в промышленных или научных целях. Поддержка RTnetwork для FireWire под ОС Linux во многом повторяет реализацию RTnetwork для сетевого стека. Общение FireWire стека и ядра RTnetwork реализовано через Ethernet-эмуляцию, то есть передаваемые пакеты преобразуются в Ethernet-пакеты для RTnetwork и FireWire-пакеты для стека FireWire.
Таким образом, FireWire работает так же, как и все остальные Ethernet-устройства в RTnetwork.

фичности» нового блокирования его патч получился гораздо меньше, чем у MontaVista.

Конечно, работа над патчем продолжается (и очень бурно — Инго выпускает по одной новой версии патча в неделю). Не все пока гладко, но надеемся, что все у него получится, а патч все-таки войдет в основное ядро Linux.

«Тяжелый» realtime

Теперь перейдем к проектам, цель которых — сделать из Linux настоящее realtime-ядро. «Интересно, чем их не устраивают, например, RT-патчи?» — спросите вы. А вот чем: все реализации soft-realtime не дают заданных (или детерминированных) параметров задержек, они лишь помогают их снизить. Realtime-система с негарантированными задержками, конечно, может сводить звуковые дорожки, но вот управлять атомной электростанцией, быть бортовой системой истребителя-перехватчика или просто супербыстрым IRC-спамботом — вряд ли. И вообще, настоящий realtime должен начинаться с отклика в несколько микросекунд. Существует пара проектов, которые реализуют hard-realtime на ядре Linux. Это RTLinux и RTAI/fusion.

RTLinux

Можно назвать этот проект паршивой овцой среди стада Open Source — реализация закрыта патентом на ПО. Основная идея RTLinux (и суть патента) — все аппаратное обеспечение функционирует под управлением маленького realtime-ядра, а само ядро Linux вместе с его процессами и задачами является лишь одним из низкоприоритетных процессов realtime-ядра. С микроядром можно общаться через специальный модуль, имеющий ограниченный набор команд. Для общения микроядра с прикладным ПО существуют так называемые каналы связи микроядра и ядра Linux. Таким образом, realtime-микроядро контролирует все аппаратное обеспечение, и можно гарантировать работу в реальном времени всей системы в целом.

RTAI/fusion

Разработчики из RTAI сначала хотели пойти по пути RTLinux — то есть изолировать основное ядро и ядро Linux друг от друга, но потом решение изменилось в сторону ядра Adeos. Это некое «суперядро», в котором realtime-ядро и классическая Linux выполняются как обычные процессы, что позволяем им свободно общаться между собой. То есть, мы используем множество вариантов связи между системами, чтобы realtime-компонент ответил первым на запросы оборудования, что является более гибким решением, нежели подход RTLinux. К сожалению, ядерный модуль для доступа к realtime-компонентам ядра все равно необходим, что усложняет реализацию пользовательского ПО.

Для решения этих проблем Филипп Джером, один из разработчиков ядра Adeos, предложил свою реализацию, названную RTAI/Fusion (впоследствии Xenomai). Она позволила применять обычные приложения Linux, использующие стандартные вызовы ядра, но при этом время отклика (response time) этих приложений было гарантированным, то есть приближенным к требованиям hard-realtime. RTAI/fusion может работать в двух режимах:

- «Усиленный» — гарантирует более низкие задержки, но сама программа должна зависеть только от возможностей, предоставляемых API RTAI. Некоторое количество системных вызовов ядра Linux реализуется RTAI, но не все.
- Когда выполняемый процесс запрашивает системный вызов, который не может быть реализован в «усиленном» режиме, то система переводится в следующий режим, называемый «защищенным». Он похож на реализации от MontaVista или Инго Молнера, то есть доступны все вызовы ядра, но задержки тоже выше. «Защищенный» режим запрещает все запросы на прерывание во время выполнения realtime-задачи, что может также увеличивать задержки.

Процессы могут выполняться как в «усиленном», так и в «защищенном» режиме — все зависит он требований этих процессов и реализации данных требований в RTAI.

Перспективы

Как правило, в заключение подобных обзоров пишут о том, когда тот или иной патч появится в основном ядре Linux, но пока ни один из описанных патчей (за исключением, может быть, модификаций -ck) не был принят большинством разработчиков ядра. Основные проблемы — это стабильность, которая может пострадать, и объем кода, который надо переписать. Возможно, все эти инициативы станут хорошим катализатором для открытия ветки 2.7, а может, и нет, так как на проверку этих патчей и их дальнейшее улучшение и развитие необходимо время. |

Дополнительная информация

Ссылки по теме

Цикл заметок о realtime-возможностях Linux на LWN:
- **http://lwn.net/Articles/106016;**
- **http://lwn.net/Articles/106010;**
- **http://lwn.net/Articles/129511;**
- **http://lwn.net/Articles/108216;**
- **http://lwn.net/Articles/96494;**
- **http://lwn.net/Articles/105866;**
- **http://lwn.net/Articles/129211.**

Подборка патчей от Con Kolivas для ядра 2.6:
- **http://members.optusnet.com.au/ckolivas/kernel.**

Подборка патчей от Con Kolivas для ядра 2.4:
- **www.plumlocosoft.com/kernel.**

Linux + PMutex для ядра 2.4:
- **http://inf3-www.informatik.unibw-muenchen.de/research/linux/mutex;**
- **http://inf3-www.informatik.unibw-muenchen.de/research/linux/hannover/automation_conf04.pdf.**

RTAI/fusion/Xenomai:
- **www.rtai.org;**
- **http://snail.fsffrance.org/www.xenomai.org;**
- **www.rtnet.org.**