Микропроцессорные и цифровые системы
2005. №2(10)

УДК 004.451

© 2005 г. **Н.Ю. Сорокин,**
О.М. Ермаков,
Ю.Г. Ля
(Тихоокеанский государственный университет, Хабаровск)

ТЕСТИРОВАНИЕ РАСШИРЕНИЙ РЕАЛЬНОГО ВРЕМЕНИ ОПЕРАЦИОННОЙ СИСТЕМЫ LINUX

В работе приводится обзор распространенных расширений реального времени для операционной системы Linux. Основным объектом исследований являются латентность переключения процессов в ядрах Linux ветвей 2.4.x и 2.6.x, а также свойства расширений реального времени для этих ядер.

Введение

Особую важность и популярность в настоящее время приобретают операционные системы (ОС), основанные на открытом исходном коде (open source systems), поскольку они не являются коммерческими и доступны широкому кругу программистов с целью их постоянного усовершенствования. Одним из примеров таких операционных систем является ОС Linux – операционная система, которая уже широко распространена по всему миру. Однако использование таких систем в решении задач автоматизации и контроля, а также во встраиваемых системах весьма ограниченно ввиду того, что ядро такой операционной системы не поддерживает выполнение задач реального времени и имеет значительное число ограничений – например, отсутствие строгих временных характеристик планирования задач и др.

Целью исследований в области операционных систем реального времени на базе систем с открытым исходным кодом ставится адаптация таких систем к задачам управления и контроля. Основными задачами являются: детальное изучение характеристик и свойств компонентов ядра операционной системы Linux ветвей 2.4.x и 2.6.x, изучение и улучшение программных надстроек над стандартными ядрами, – например, KURT, RTAI и RTLinux.

Определим ряд терминов, используемых в данной статье.

ОСОН (*GPOS – General Purpose Operating Systems*) – *Операционные Системы Общего Назначения* – операционные системы, направленные на решение широкого круга задач. При этом решение узкоспециализированных задач либо задач, требующих спецификации четких временных параметров со стороны операционной системы, с помощью данных систем невозможно либо неприемлемо.

ОСРВ (*RTOS – Real-Time Operating Systems*) – *Операционные Системы Реального Времени* – операционные системы, способные обеспечить выполнение задачи в отведенное время [1]. Такие системы соответствуют следующим требованиям [2]:

детерминизм – свойство системы, заключающееся в выполнении операции в фиксированное, предопределенное время или в пределах предопределенных интервалов времени;

чувствительность – требование, определяющее количество времени, необходимое операционной системе для обслуживания прерывания после его распознавания;

управление со стороны пользователя – возможность тонкой настройки приоритетов задач, разделять задачи на жесткие и мягкие, определять использование страничной организации памяти или свопинг процессов и т.д.;

надежность – требование к системе, определяющее ее защищенность как от внутренних, так и от внешних ошибок, сбоев и отказов;

восстановление после сбоев – характеристика системы, которая описывает способность сохранить максимальную функциональность и не потерять данные при сбое.

Детерминизм и чувствительность в совокупности образуют время отклика на внешнее событие. Требования ко времени отклика являются критическими для ОСРВ. За основу исследований в данной работе мы взяли одну из составляющих времени отклика – латентность. Назовем ***л****атентностью* меру, определяющую время между поступлением прерывания и его обработкой. Наряду с этим, *порог латентности (deadline)* – отметка, задаваемая пользователем, превышение которой латентностью нежелательно, иногда невозможно.

Известно, что ОСРВ делятся на два основных класса. *ОСЖРВ* (*HRTOS – Hard Real-Time Operating Systems*) – *Операционные Системы Жесткого Реального Времени* – системы, которые гарантируют окончание выполнения задачи в строго отведенные ей временные рамки, их несоблюдение ведет к краху системы. Примером краха системы может служить несвоевременный отклик закрылок самолета при нажатии на штурвал, отказ аппарата системы жизнеобеспечения и т.д. Разработчики ОСЖРВ предоставляют гарантии по поводу того, что данная латентность не превысит порога латентности. Такие гарантии можно дать, лишь предсказав время выполнения элементарных функций ядра. При этом гарантии такого рода де-

лают невозможным присутствие в ОСЖРВ виртуальной памяти, так как время выполнения системных функций, работающих с вторичными устройствами, зачастую трудно прогнозируемо.

ОСМРВ (*SRTOS – Soft Real-Time Operating Systems*) – *Операционные Системы Мягкого Реального Времени* – задачи в такой системе менее требовательны к латентности. Это означает, что цель ОСМРВ – обеспечить латентность в пределах порога, превышение его не ведет к краху системы. Потенциальными задачами для таких ОСРВ могут быть мультимедиа приложения, Интернет приложения, работающие с потоковым звуком или видео. Превышение порога латентности чаще всего проявляется в виде искажения картинки, звука. Задачи реального времени в такой системе имеют наивысший приоритет по сравнению с рядовыми процессами. Таким образом, менее привилегированные процессы могут длительное время простаивать в очереди, т.е. наблюдается эффект голодания. Если же латентность у такого процесса высока слишком часто, система не попадает под определение ОСМРВ. Задачи реального времени сосуществуют с рядовыми задачами, поэтому в ОСМРВ часто возникает проблема передачи управления привилегированным задачам.

Модификации ОС Linux

Изначально ОСРВ разрабатывались как отдельные специализированные продукты, которые выполняли узкий круг задач. Такие ОСРВ как QNX проектировались с нуля для решения задач реального времени, чаще всего различных задач контроля технологических процессов. ОС такого рода предоставляют четкие гарантии, свойственные ОСРВ, но накладывают ограничения на использование технологий, свойственных ОСОН, – таких, например, как своп и использование сетевых интерфейсов. ОС такого рода справляются со специализированными задачами наилучшим образом, но такие решения довольно дороги и требуют квалифицированного персонала. Существует большой класс задач, в которых необходимо задание четкого времени отклика системы, но приобретение специализированной ОС нерентабельно.

Около 10 лет назад начали появляться ОСРВ – решения на основе систем с открытым исходным кодом. Данные расширения позволяют добиться желаемого результата с минимумом финансовых затрат. На сегодняшний день существует немало разработок, как платных, так и бесплатных, способных улучшить характеристики ОСОН, тем самым приблизив их по возможностям к ОСРВ. Некоторые продукты нацелены на изменение какой-то одной характеристики, другие – на изменение множества характеристик, системы в целом. Наибольшее число таких продуктов разрабатываются под ОС Linux. Из всего разнообразия подобных продуктов мы протестировали наиболее популярные расширения ОС Linux: RTLinux, RTAI, KURT.

RTLinux. Эта разработка является собственностью компании FSMLabs [3] и имеет как коммерческий вариант, так и бесплатный, распространяемый под двумя лицензиями – GPL и OPL. RTLinux выполнен в виде подгружаемого модуля, который представляет собой второе ядро, подменяющее стандартное POSIX-ядро Linux в наиболее ответственных участках.

RTLinux реализует режим “жесткого” реального времени. При такой схеме стандартное ядро ОС Linux запускается как низкоприоритетная задача ядра реального времени. Проблема с запрещением Linux – прерываний (ядро Linux – невытесняемое) решена путем имитации обработчиков прерываний Linux в ядре реального времени. Все аппаратные прерывания перехватываются ядром операционной системы реального времени. При возникновении прерывания RT-ядро перехватывает его. Если прерывание вызвано задачей реального времени, ядро сохраняет состояние “обычной” Linux и немедленно запускает задачу реального времени. В противном случае, если в обработчике реального времени установлено, что он будет разделять это прерывание с Linux, обработчику присваивается состояние ожидания. Если ОС Linux потребовала разрешить прерывания, то прерывания, находящиеся в состоянии ожидания, эмулируются. Ядро RTLinux спроектировано таким образом, что ядру реального времени никогда не приходится ожидать освобождения ресурса, занятого Linux-процессом.

RTAI. Этот проект был основан в Департаменте аэрокосмического проектирования Миланского политехнического института (DIAPM – Department of Aerospace Engineering of Politecnico di Milano), который спустя некоторое время был передан сообществу разработчиков систем с открытым исходным кодом (open source community). RTAI – Real-Time Application Interface – это расширение “жесткого” реального времени ядра ОС Linux [4]. Решение RTAI сходно с RTLinux и не имеет серьезных архитектурных отличий. Однако существует несколько различий на уровне изменения кода стандартного ядра. В системе RTLinux большинство изменений произведено непосредственно в исходных кодах ядра, что усложняет поддержку данного продукта – такую как обновление, поиск ошибок и т.д. RTAI ограничивает такие изменения путем добавления уровня аппаратной абстракции (hardware abstraction layer – HAL), состоящего из структур указателей на векторы прерываний и функций разрешения/запрещения прерываний.

KURT – Kanzas University Real-Time Linux – разрабатывается Information and Telecommunication Technology Center (ITTC) в Канзасском университете, распространяется по лицензии GPL [5]. KURT – расширение “мягкого” реального времени, позволяющее планировать процессы реального времени с разрешением в 10 микросекунд. Предусматривает два режима: нормальный и режим реального времени, т.е. процесс, использующий возможности KURT, в любой момент времени может находиться в

одном из этих режимов. Это расширение выполнено в виде отдельного модуля стандартного ядра ОС Linux, играющего роль дополнительного планировщика. Изменения кода ядра затрагивают не только планировщик, но и системный таймер: добавляется второй таймер, позволяющий довести время переключения контекста задач до 1 мс. Подход, используемый этой разработкой, дает возможность использовать режим реального времени, применяя весь API Linux, а не ограничиваясь лишь интерфейсами библиотеки.

Тестирование ядер ОС Linux и их модификаций

Целью проведения тестов было измерение одного из показателей, определяющих, насколько ОС Linux и его модификации соответствуют понятию “операционная система реального времени” – латентности.

Был написан тест, замеряющий латентность, с помощью которого мы исследовали следующие версии ядер ОС Linux: Linux 2.4.20 (vanilla), Linux 2.6.12.2 (vanilla), Linux 2.6.10 (с патчем RTAI), Linux 2.4.18 (с патчем KURT). Другие расширения реального времени не были протестированы ввиду недостатков при подключении их к стандартному ядру и отсутствия документации. Для каждой версии ядра проводилось три теста по 320000 замеров каждый.

Технология замеров состояла в следующем:

1) засекалось текущее время с точностью до наносекунды;
2) задача переводилась в спящий режим путем вызова функции nanosleep();
3) снова засекалось текущее время;
4) проводилось сравнение времени, которое было выставлено при вызове функции nanosleep(), и времени, которое процесс реально “спал” – разница между этими значениями и составляла искомую латентность.

```
for( int i = 0; i < 320000; i++ )
{
    int cur_time_1 = lat_gettime();
    nanosleep( 40000 );
    int cur_time_2 = lat_gettime();
    int latency = ( cur_time_2 – cur_time_1 ) – 40000;
}
```

Тесты проводились при максимальной загрузке системы, создаваемой искусственно, следующими средствами: 3 копии процесса “head” и 3 копии процесса “cat”, которые имитировали дисковую активность, также утилиты из специализированного пакета stress-kernel.

На основании проведенных тестов составлялась статистика по следующим параметрам:

1) общее количество замеров;
2) средняя латентность за тест (preemption latency average);

3) количество замеров, превысивших порог латентности, заданный на уровне 10 мс (deadline misses (10 ms));

4) наихудший (максимальный) замер (preemption latency maximum).

Все тесты запускались на системе со следующей конфигурацией: CPU Intel® Celeron® 2.40GHz, DDR 512Mb; ОС: Linux Slackware 9.0; glib 1.2.10, glib2 2.2.1, glibc 2.3.1, gcc 3.2.2.

В свою очередь на основании статистики строились приведенные (усредненные по значениям) графики.

Как видно из графика (рис. 1), средняя латентность находится на уровне 10 ms, данные показатели характерны для ветви 2.4.x [7]. Порог латентности превышен в 46.48% случаев. Ядро с такими показателями невозможно использовать в качестве альтернативы ОСРВ.

Результаты другого теста показали, что латентность в ветви 2.6.x существенно уменьшена (рис. 2, 3). Изменения и нововведения в данной ветви позволили понизить средний показатель латентности примерно в семь раз. Порог латентности не был превышен ни разу. Отсутствуют пики, т.е. резкие скачки значения латентности, что свидетельствует об общей стабильности системы.

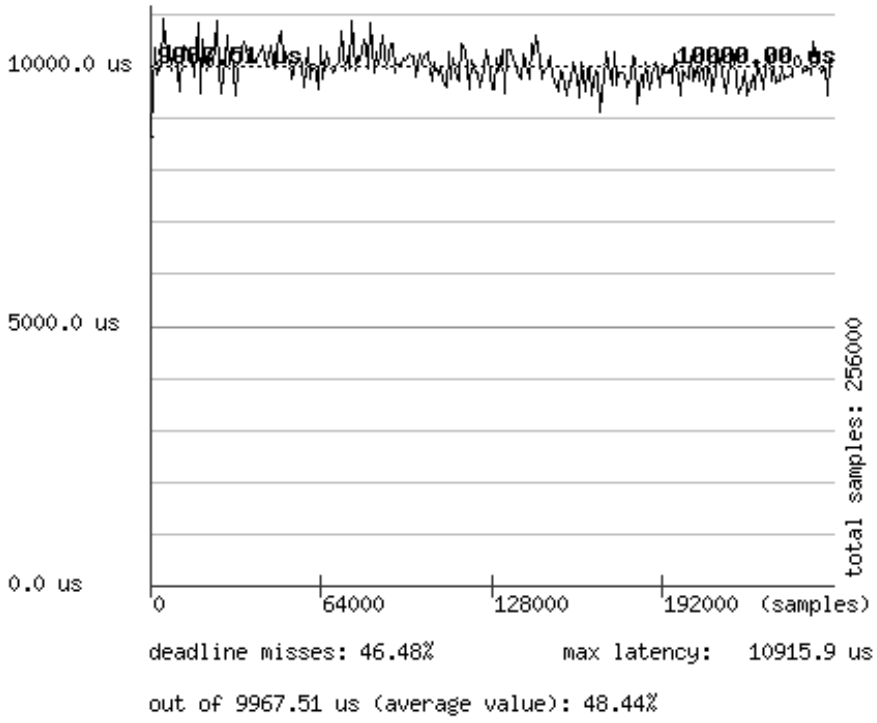

Рис. 1. Результаты теста для ядра 2.4.20 (vanilla).

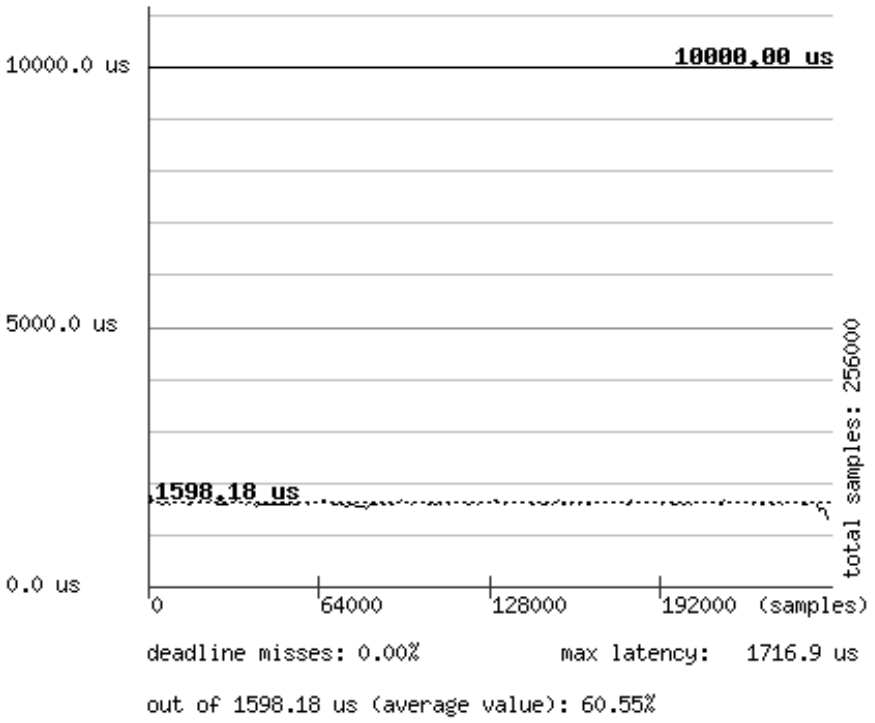

Рис. 2. Результаты теста для ядра 2.6.12 (vanilla).

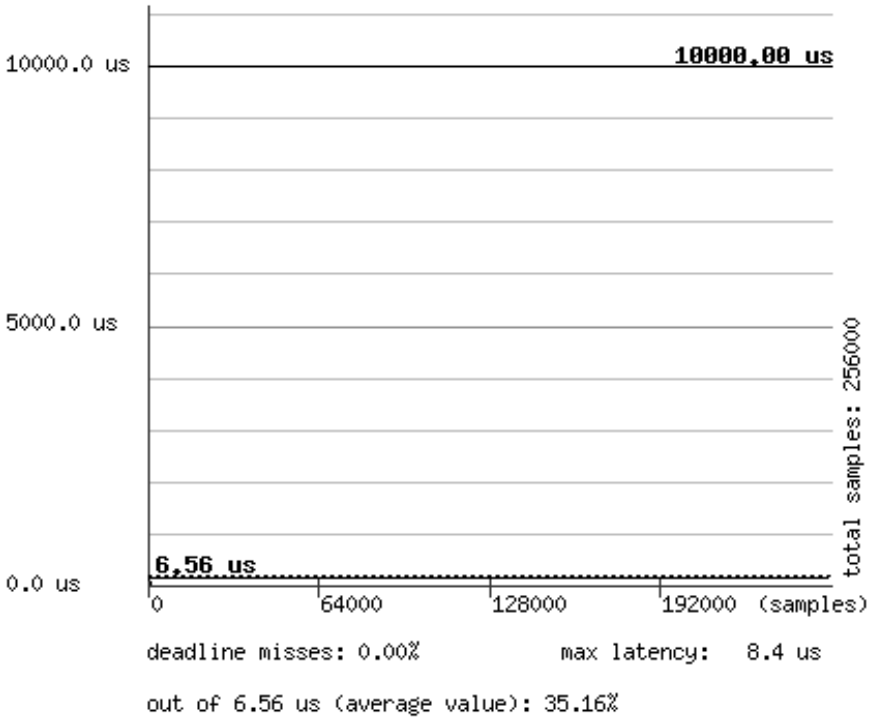

Рис. 3. Результаты теста для ядра 2.6.10 (с патчем RTAI).

Надстройка RTAI над ядром версии 2.6.10 позволила добиться заметного снижения латентности. Помимо того, что порог латентности не

был превышен ни разу, показатели оригинальной ветви 2.6.x были также заметно улучшены. Такая низкая латентность приближается к уровню ОСРВ, но при условии калибровки системы RTAI.

Была протестирована версия ядра 2.4.18 с патчем KURT, позволившим использовать второй планировщик реального времени. Результаты тестов оказались неудовлетворительными ввиду того, что превышение порога латентности наблюдалось на протяжении всего теста. Ниже представлена сводная таблица результатов всех проведенных тестов.

Версия ядра	2.4.18 (KURT)	2.4.20 (vanilla)	2.6.12 (vanilla)	2.6.10 (RTAI)
Максимальная латентность, мкс	108099.1	10915.9	1716.9	8.4
Средняя латентность, мкс	70604.12	9967.51	1598.18	6.56
Количество замеров превысивших среднюю латентность	48%	48%	60%	35%
Количество замеров превысивших порог латентности в 10000 мкс	100%	46%	0%	0%

Заключение

Проведенное исследование показало, что нововведения в ветви 2.6.x позволили добиться значительного снижения латентности за счет таких изменений как использование нового O(1)-планировщика процессов, поддержка механизма вытесняемости ядра, улучшенная система виртуальной памяти и т.д. Однако улучшить характеристик ядер версий 2.4.x и 2.6.x можно, используя различные расширения реального времени. Например, расширение RTAI позволяет улучшить результаты латентности в несколько раз по сравнению с оригинальной версией 2.6.x. При этом тестирование популярного расширения KURT для ядра 2.4.18 не принесло ожидаемых результатов по причине отсутствия постоянной поддержки со стороны производителя для современного аппаратного и программного обеспечения.

ЛИТЕРАТУРА

1. Стандарт POSIX 1003.1. http://www.unix.org/version3/ieee_std.html.
2. *Столлингс В.* Операционные системы. Лондон: изд-во Вильямс, 2002.
3. FSMLabs, Inc. RTLinuxFree. http://www.fsmlabs.com.
4. DIAPM. RTAI. http://www.rtai.org.
5. Kansas University. KURT. http://www.ittc.ku.edu/kurt.
6. *Love R.* Introducing the 2.6 Kernel. http://linuxjournal.com/article/6530, 2003.
7. *Clark Williams, Red Hat*. Inc. Linux Scheduler Latency. http://www.linuxdevices.com/files/article027/rh-rtpaper.pdf.

Статья представлена к публикации членом редколлегии Чье Ен Уном.