Для цитирования: Аблакатов М. А. Управление энергопотреблением сервера семейства «Эльбрус», основанное на отключении процессорных ядер по данным подсистемы балансировки загрузки // Вопросы радиоэлектроники. 2017. № 3. С. 48–51.
УДК 004.451.8

М. А. Аблакатов[1]

[1] АО «МЦСТ»

УПРАВЛЕНИЕ ЭНЕРГОПОТРЕБЛЕНИЕМ СЕРВЕРА СЕМЕЙСТВА «ЭЛЬБРУС», ОСНОВАННОЕ НА ОТКЛЮЧЕНИИ ПРОЦЕССОРНЫХ ЯДЕР ПО ДАННЫМ ПОДСИСТЕМЫ БАЛАНСИРОВКИ ЗАГРУЗКИ

Статья посвящена автоматизированной системе мониторинга нагрузки и отключения процессорных ядер, спроектированной в составе ОС Linux для снижения энергопотребления серверных платформ на базе микропроцессоров «Эльбрус-4С». Целью является создание пользовательского демона и интерфейса передачи информации о задачах на процессорных ядрах из пространства ядра в пространство пользователя. Проблема решается при использовании технологии горячего отключения процессорных ядер HotPlug.

Ключевые слова: *ядро Linux, управление энергопотреблением, Эльбрус-4С.*

Введение

Существенное значение для серверных платформ имеет проблема энергопотребления в состоянии простоя, вызванного временным отсутствием подлежащих выполнению задач. Ранее, в работе [1], было показано, что отключение конвейера и синхроимпульсов процессорного ядра могут снизить потребление им энергии на 5,7 и 28% соответственно по сравнению с работой в пустом цикле. Опираясь на эти показатели, данная работа ставит целью создание автоматизированной системы мониторинга нагрузки и выключения процессорных ядер для снижения энергопотребления серверных платформ на базе микропроцессоров типа «Эльбрус-4С».

Ключевым элементом системы является пользовательское приложение – демон (*daemon*), работающее в фоновом режиме. Используя *Sysfs* – пользовательский интерфейс взаимодействия с ядром Linux, демон последовательно инициирует логическую и аппаратную стадии отключения процессорного ядра (*HotPlug*) при отсутствии на нем работающих либо спящих пользовательских процессов в течение заданного регулируемого интервала времени. Их количество отслеживается в пространстве ядра подсистемами планировщика задач и балансировщика нагрузки и хранится в структурах данных ядра. Доступ пользовательского демона к этим данным был обеспечен через интерфейс *Sysfs*. При логическом отключении процессор удаляется из маски активных процессоров, его пользовательские и системные потоки переносятся на активное ядро, а потоки категории *per-cpu kthreads,* имеющие копию на каждом ядре, приостанавливаются. В процессе аппаратного отключения запрещается обработка прерываний на контроллере прерываний LAPIC (Local Advanced Programmable Controller) этого ядра и отключается подача на него синхроимпульса.

Реализация системы

Доступ к данным о загруженности ядра

Ввиду того, что в многозадачном режиме ОС Linux каждая задача выполняется на определенном процессорном ядре, достижение максимальной производительности требует обеспечения оптимального распределения задач между ядрами. Эту функцию в ОС выполняет балансировщик загрузки (*load-balancing*), который делает вывод о загруженности каждого ядра, возможности его отключения или необходимости включения, регулярно отслеживая следующие параметры:

- *nr_running* – число задач, находящихся в очереди на исполнение, в т.ч. исполняющаяся задача;
- *nr_sleeping* – число задач, находящихся в состоянии ожидания TASK_INTERRUPTABLE или TASK_UNINTERRUPTABLE.

В разработанной системе эти параметры представлены в виде файлов */sys/device/system/cpu/cpuX/nr_running* и */sys/device/system/cpu/cpuX/nr_sleeping,* доступ к которым из пространства пользователя обеспечивается через интерфейс в *Sysfs*.

Демон в пространстве пользователя

При создании демона в пространстве пользователя была поставлена задача – отслеживать приведенные выше параметры загрузки процессорных ядер через интерфейс в *Sysfs* и при необходимости, в зависимости от их значений, включать либо выключать ядра через другой интерфейс в *Sysfs* (*/sys/device/system/cpu/cpuX/online,* где *X* – номер процессорного ядра).

Отключение процессорных ядер производится в случае, если на процессорном ядре отсутствуют исполняющиеся, ожидающие исполнения или спящие задачи в течение последних *SLEEP_THRESHOLD* секунд. Нулевое процессорное ядро не подлежит отключению.

Когда количество ожидающих исполнения задач на каком-либо процессорном ядре превышает параметр *WAKE_THRESHOLD*, предлагается включить любое спящее процессорное ядро, если такое присутствует в системе.

Отключение обработки прерываний таймера

В данной работе термином «*HotPlug*» обозначается аппаратно-программный процесс, выполняющий включение/отключение ядер без необходимости перезагрузки системы или ее существенного выхода из рабочего режима. Реализация *HotPlug* в ядре Linux ОС «Эльбрус» не позволяла корректно выполнять отключение процессорных ядер, т.к. после перевода в отключенное состояние они тут же из него выходили при поступлении прерывания с таймера на контроллер LAPIC процессорного ядра. В связи с этим возникла необходимость в процессе исполнения аппаратной части *HotPlug* запрещать прерывания с таймера для отключаемого процессорного ядра и, соответственно, снимать запрет для включаемого ядра. Это выполняется путем обнуления и установки определенного бита в таблице LVT контроллера.

Потоки ядра

Потоки ядра (*kernel threads*, или *kthreads*) – это способ представления в пространстве ядра фоновых задач, которые обрабатывают асинхронные события либо спят в ожидании события. Потоки ядра исполняются в пространстве ядра и имеют доступ к его функциям и структурам данных. Подобно пользовательским потокам, потоки ядра монополизируют процессорные ядра в результате вытесняющего планирования задач.

В ядре Linux имеются наборы потоков ядра, исполняющих задачи, которые напрямую относятся к процессорным ядрам (*per-cpu kthreads*), – например, обработку приходящих на них прерываний. Копии этих потоков существуют для каждого ядра, и при отключении ядра, на котором исполняется такой поток, он не переносится на другое активное ядро, а должен быть заморожен до тех пор, пока отключенное ядро не продолжит работу.

В ядре Linux для таких потоков существует интерфейс *percpu_thread*. В него входят функции добавления и удаления *per-cpu* потоков ядра – *int smpboot_register_percpu_thread()* и *void smpboot_unregister_percpu_thread,* а также тип структуры *struct smp_hotplug_thread,* которой они описываются.

Нетривиальная проблема возникла в связи с тем, что в ядро Linux ОС «Эльбрус» был ранее добавлен поток ядра *kfree_hw_stacksd*, который отвечает за регулярный запуск функции очистки аппаратных стеков на каждом процессорном ядре. По логике своей работы он является *per-cpu* потоком ядра, однако создавался как обычный поток ядра. Вследствие этого в SMP-системе при отключении процессорного ядра данный поток мигрировал с него на другое рабочее ядро, несмотря на установленную афинность только к текущему ядру. Это обусловлено тем, что, если ядру Linux не удается перенести поток ядра ни на одно из работающих ядер из-за установленных параметров афинности, оно их игнорирует. Подобное поведение вызывало зависание системы после выключения процессорного ядра с имеющимся на нем потоком *kfree_hw_stacksd*.

Поэтому для корректной работы потоков *kfree_hw_stacksd* в SMP-системе было необходимо изменить их реализацию, используя *percpu_thread* интерфейс. Листинг новой реализации приведен ниже.

```
static DEFINE_PER_CPU(struct list_head, hw_stacks_to_
free_list);
static DEFINE_PER_CPU(struct task_struct *, kfree_hw_
stacks_task);
static void kfree_hw_stacksd(unsigned int cpu)
{
    while (kthread_should_park() == 0) {
        set_current_state(TASK_INTERRUPTIBLE);
        schedule();
        __set_current_state(TASK_RUNNING);
        free_queued_hw_stacks(cpu);
    }
}
static int kfree_hw_stacksd_should_run(unsigned int cpu)
{
    return cpu_online(cpu);
}
static struct smp_hotplug_thread hw_stacks_threads = {
    .store = &kfree_hw_stacks_task,
    .thread_should_run = kfree_hw_stacksd_should_run,
    .thread_fn = kfree_hw_stacksd,
    .thread_comm = «khwstacksd/%u»,
```

Таблица. Результаты эксперимента по отключению микропроцессорных ядер

Количество включенных процессорных ядер	4	3	2	1
Действующее напряжение питания ВК, В	220	220	220	220
Действующий ток питания ВК, мА	325	304	290	276
Потребляемая ВК мощность, Вт	72	67	64	61
Снижение энергопотребления ВК,%	0	7	11	15

```
};
static int create_kfree_hw_stacks_tasks()
{
    BUG_ON(smpboot_register_percpu_thread(&hw_
    stacks_threads));
    return 0;
}
arch_initcall(create_kfree_hw_stacks_tasks);
```

Результаты

Для оценки эффективности данного решения был проведен эксперимент на вычислительном комплексе (ВК) на базе четырехъядерного микропроцессора E2S, в ходе которого измерялось потребление мощности ВК в зависимости от количества активных микропроцессорных ядер. Во время измерений при отсутствии подлежащих выполнению на них задач отключались микропроцессорные ядра. Результатом измерений являлся процент снижения потребления мощности ВК с отключенными ядрами по сравнению с потреблением, когда микропропроцессорные ядра работают в пустом цикле. Результаты данного эксперимента приведены в таблице.

Выводы

В статье описаны программные средства, введенные в ОС Linux для минимизации потребления мощности процессорными ядрами микропроцессора «Эльбрус-4С» в состоянии простоя.

Использование данных программных средств даже в системе на четырехъядерном микропроцессоре позволяет снизить потребление мощности системой на 15%. Стоит отметить, что это снижение потребления мощности всего вычислительного комплекса, включающего и другие потребляющие мощность узлы помимо микропроцессора, что позволяет говорить о том, что полученные результаты не противоречат заявленным в [1] оценкам в 27%. Кажется логичным заявление о том, что на серверной платформе «Эльбрус», реализованной на четырех восьмиядерных процессорах, процент сокращения потребления мощности ВК будет превышать 15%.

Дальнейшая работа в этой области предполагает реализацию новых алгоритмов балансировки задач между процессорными ядрами в составе подсистемы балансировщика загрузки в ОС Linux.

СПИСОК ЛИТЕРАТУРЫ

1. Кравцунов Е.М., Семенихин С.В. Управление энергопотреблением СНК «Эльбрус-2С+» в состоянии простоя процессорного ядра // Вопросы радиоэлектроники. 2013. Т. 4. № 3. С. 143–157.

ИНФОРМАЦИЯ ОБ АВТОРЕ

Аблакатов Михаил Андреевич, инженер-программист, АО «МЦСТ», 119334, Москва, ул. Вавилова, д.24, тел.: 8 (499) 135-33-21, e-mail: mikhail.a.ablakatov@mcst.ru.

For citation: Ablakatov M.A. «Elbrus» server power management based on a CPU HotPlugging and load balancing subsystem. Voprosy radioelektroniki, 2017, no. 3, pp. 48–51.

M.A. Ablakatov

«ELBRUS» SERVER POWER MANAGEMENT BASED ON A CPU HOTPLUGGING AND LOAD BALANCING SUBSYSTEM

The work is dedicated to the development of an automated system as part of the Linux operating system for monitoring the CPUs load and CPU HotPlugging to reduce power consumption of server platforms based on the «Elbrus-4C» microprocessor. The goal of system design is to create a user daemon and data transfer interface on the number of tasks on the processor cores from kernel space to user space. This problem is solved by using the HotPlug technology.

Keywords: Linux kernel, power management, Elbrus-4C.

REFERENCES

1. Kravtcunov E. M., Semenikhin S. V. Power Management of «Elbrus-2S+» SoC in a state of idle CPU. *Voprosy radioelektroniki,* 2013, vol. 4, no.3, pp. 143–157.

AUTHOR

Ablakatov Mikhail, software engineer, JSC «MCST», 24, Vavilova st., Moscow, 119334, Russian Federation, tel.: +7 (499) 135-33-21, e-mail: mikhail.a.ablakatov@mcst.ru.