DOI:

Виртуализация подсистемы прерываний микропроцессоров «Эльбрус»

С. Рыбаков[1], Р. Деменко[2]

УДК 004.414.23 | ВАК 05.13.05

В современных вычислительных системах широко используется виртуализация различных аппаратных ресурсов. Один из ключевых компонентов вычислительных комплексов, в значительной степени определяющий успешность их применения в различных приложениях, – подсистема обработки прерываний. В данной статье рассмотрены разработанные для архитектуры «Эльбрус» программно-аппаратные решения, направленные на повышение эффективности виртуализации подсистемы прерываний.

Средства виртуализации, реализованные в современных вычислительных системах, позволяют запустить на высокопроизводительном сервере несколько изолированных друг от друга виртуальных машин, эмулирующих реальные физические серверы со своими (гостевыми) операционными системами (далее для их наименования используется термин «гость»). Виртуализация операционных систем (ОС) дает возможность значительно повысить эффективность использования вычислительных ресурсов сервера.

Сохранение целостности такого подхода при разработке нового поколения микропроцессоров семейства «Эльбрус» естественно поставило вопрос об эффективной виртуализации подсистемы прерываний, прежде всего направленной на обеспечение минимального времени реакции на прерывания. Описанные в статье решения, ориентированные на выполнение этого требования, в основном базировались на аппаратной поддержке, обеспеченной с использованием специально разработанного аппаратного контроллера (Elbrus Programmable Interrupt Controller, EPIC) [1].

СТРУКТУРА ПОДСИСТЕМЫ ОБРАБОТКИ ПРЕРЫВАНИЙ

Контроллер прерываний состоит из нескольких компонентов:

- Core EPIC (CEPIC) – локальный компонент, уникальный для каждого процессорного ядра, содержит основные управляющие регистры;
- Processor EPIC (PREPIC) – общий для микропроцессора компонент;
- Input Output EPIC (IOEPIC) – внешний компонент, входит в состав встроенного контроллера периферийных устройств.

Компоненты контроллера прерываний взаимодействуют через сообщения специального формата (сообщения о прерывании, служебные сообщения). Внешние устройства формируют прерывания либо в виде MSI (Message Signaled Interrupt), которое представляет собой DMA-запись (Direct Memory Access) в выделенную область памяти, либо через сигнал, подключенный к IOEPIC (вход прерывания). Упрощенная схема доставки внешних прерываний состоит из следующих шагов (рис. 1):

1. Внешнее устройство формирует сообщение формата MSI, содержащее вектор прерывания и номер ядра, которому предназначено прерывание.
2. Хост-контроллер отлавливает MSI из потока DMA-обращений и передает сообщение в контроллер прерываний, где оно маршрутизируется до целевого CEPIC; прерывание размещается в регистре.
3. Контроллер прерываний выставляет сигнал в аппаратуру ядра о наличии необработанного прерывания, который запускает механизм обработки исключительных ситуаций. Программное обеспечение читает вектор прерывания из регистра CEPIC и вызывает соответствующую программу-обработчик. Обработка прерывания завершается записью в регистр CEPIC.

Виртуализация системы прерываний предполагает следующие действия:

- эмуляцию контроллера прерываний виртуальной машины;
- поддержку актуального состояния виртуального контроллера прерываний в соответствии с работой гостевой ОС;
- формирование и доставку виртуальных прерываний.

[1] АО «МЦСТ», инженер-программист 1-й категории, МИРЭА – Российский технологический университет (РТУ МИРЭА), аспирант, Stepan.A.Rybakov@mcst.ru.

[2] АО «МЦСТ», инженер 2-й категории, roman.dmnk@gmail.com.

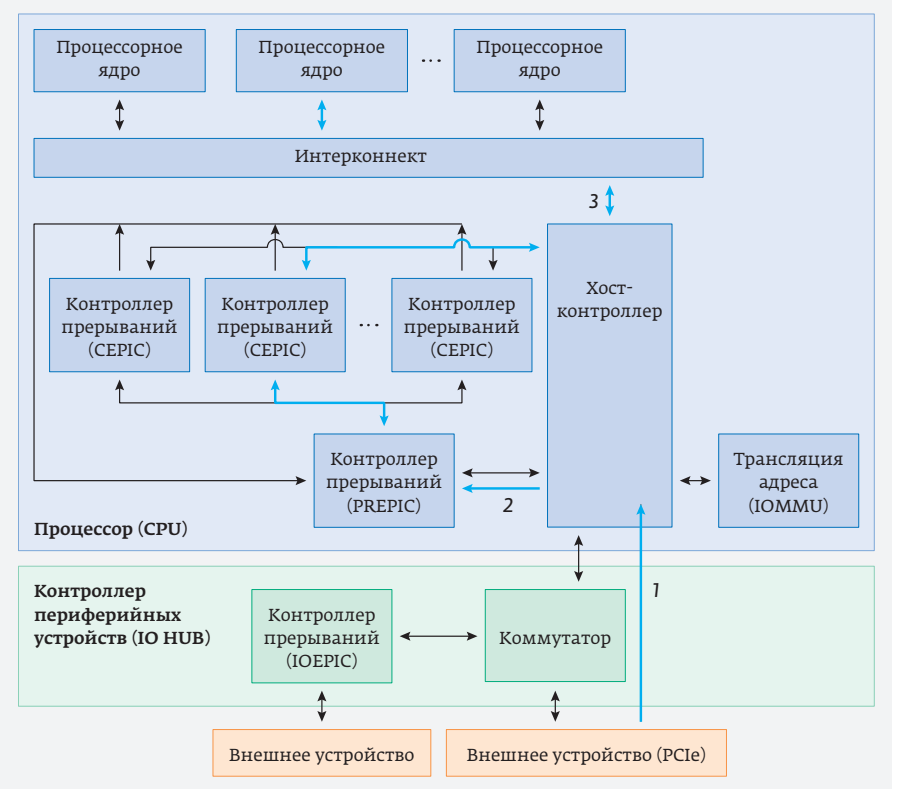

Рис. 1. Упрощенная схема доставки внешнего прерывания

ГИПЕРВИЗОР QEMU-KVM

Виртуализация операционных систем – это технология предоставления набора вычислительных ресурсов гостевым операционным системам с обеспечением их логической изоляции друг от друга. Гостевая ОС запускается внутри пользовательского приложения QEMU (Quick Emulator) [2], эмулирующего для гостя вычислительные ресурсы, оперативную память и устройства ввода-вывода. Поскольку каждому процессу QEMU соответствует одна виртуальная машина, для одновременной работы нескольких виртуальных машин запускается несколько независимых друг от друга процессов QEMU. При запуске в QEMU передается множество параметров, описывающих подлежащее эмуляции аппаратное обеспечение виртуальной машины. Примерами таких параметров являются: число виртуальных вычислительных ядер (vCPU), размер гостевой оперативной памяти и конкретные модели устройств ввода-вывода. Для эмуляции многоядерных систем гостевые ядра исполняются параллельно, в отдельных потоках QEMU. Эмуляция устройств ввода-вывода тоже осуществляется в отдельных потоках, асинхронно по отношению к потокам процессорных ядер vCPU.

Как и любое другое пользовательское приложение, QEMU взаимодействует с ядром операционной системы Linux посредством системных вызовов. Для начальной настройки и собственно запуска виртуальной машины QEMU использует модули ядра операционной системы Linux KVM (Kernel Virtual Machine) [3]. Передача управления из QEMU в KVM происходит с помощью специальных системных вызовов IOCTL (Input/Output Control). Поскольку QEMU – пользовательское приложение и запускается поверх ОС, пара QEMU-KVM является гипервизором 2-го типа. Это отличает QEMU-KVM от гипервизоров 1-го типа (например, Xen [4]), запускаемых напрямую на аппаратном обеспечении, без программной прослойки в виде ОС.

Рассмотрим подробнее процесс запуска гостевой ОС гипервизором (рис. 2). После первичной настройки виртуальной машины QEMU посредством специального вызова IOCTL передает управление в KVM, где происходят сохранение рабочего состояния (контекста) гипервизора и после этого, собственно, запуск гостя. Гостевая ОС продолжает работать до первого перехвата – принудительного возврата в гипервизор. Типичными причинами перехватов являются попытки гостя получить доступ к привилегированным ресурсам (например, регистрам устройства ввода-вывода), либо истечение выделенного гостю планировщиком кванта времени. После возврата в гипервизор KVM анализирует причину перехвата, и, если ее удается устранить локально, гость запускается снова. Если же необходима эмуляция ввода-вывода, то KVM возвращает управление QEMU, что требует дополнительной смены контекста. Таким образом, перехваты можно разделить на две группы: легковесные (обрабатываемые в KVM), и тяжеловесные (с выходом в QEMU).

ДОСТАВКА ГОСТЕВЫХ ПРЕРЫВАНИЙ БЕЗ АППАРАТНОЙ ПОДДЕРЖКИ ВИРТУАЛИЗАЦИИ

Для того чтобы проиллюстрировать необходимость аппаратных доработок по поддержке виртуализации в микропроцессорах «Эльбрус», рассмотрим алгоритм доставки и обработки внешних гостевых прерываний в системах без аппаратной поддержки (рис. 3). Источниками внешних прерываний являются эмулируемые в QEMU периферийные устройства.

QEMU с помощью специального вызова IOCTL передает информацию о прерывании из потока ввода-вывода

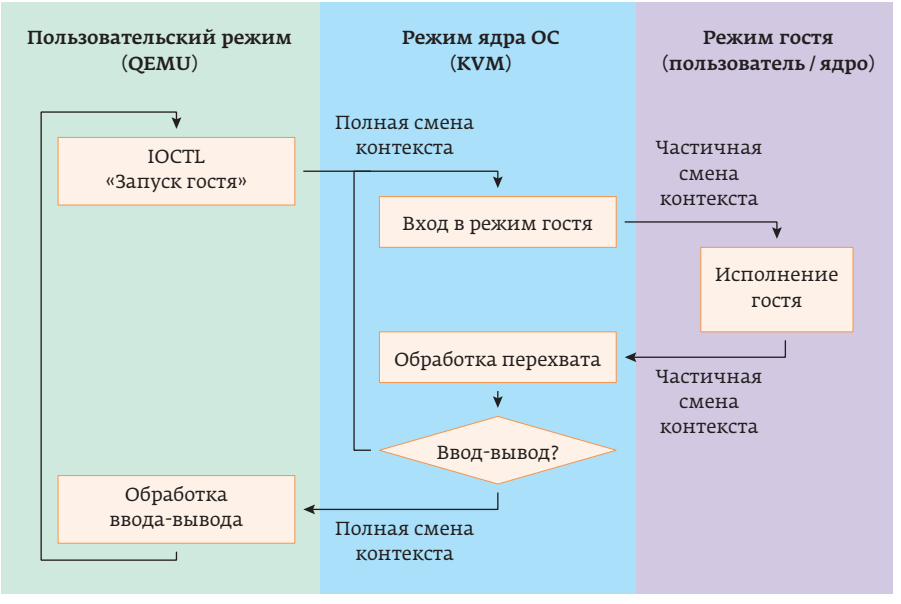

Рис. 2. Цикл запуска гостевой ОС гипервизором

в модуль KVM, который направляет прерывание в модель программируемого контроллера прерываний. После этого начинается процесс инжекции прерывания в гостевую ОС. Поскольку для инжекции гостевой процессор vCPU, которому адресовано прерывание, должен быть снят с исполнения, KVM проверяет, в работе ли это ядро в данный момент времени, и, при необходимости, приостанавливает его работу. После повторного входа в режим гостя начинается обработка прерывания. Во время обработки гость несколько раз обращается к регистрам контроллера прерываний (для определения пославшего прерывание устройства и для завершения обработки прерывания). Поскольку доступ к регистру контроллера – это привилегированная операция, каждое такое обращение перехватывается гипервизором, после чего происходит выход в QEMU и эмуляция поведения контроллера прерываний. Наконец, после успешной обработки прерывания, гостевая ОС возобновляет свое исполнение с прерванной инструкции.

Учитывая, что описанные перехваты происходят при доставке и обработке каждого гостевого прерывания, производительность гостевой системы значительно уступает невиртуализированной ОС.

Для устранения привносимых виртуализацией задержек (выделенных на рис. 3) был произведен ряд оптимизаций:

а) модель контроллера прерываний была перенесена из QEMU в KVM, что позволило уйти от тяжеловесных перехватов с выходом в QEMU;

б) был реализован алгоритм доставки прерываний гостевому ядру без снятия его с исполнения;

в) гостю был обеспечен доступ к регистрам программируемого контроллера прерываний без перехватов.

Первая из приведенных оптимизаций является чисто программной; для остальных в микропроцессорах «Эльбрус» нового поколения была введена аппаратная поддержка.

АППАРАТНАЯ ПОДДЕРЖКА ВИРТУАЛИЗАЦИИ ПОДСИСТЕМЫ ПРЕРЫВАНИЙ В МИКРОПРОЦЕССОРАХ «ЭЛЬБРУС»

Исключить перехват при обращении гостевой ОС к управляющим регистрам контроллера прерываний можно через реализацию в аппаратуре дополнительного набора регистров. Такие регистры используются виртуальным

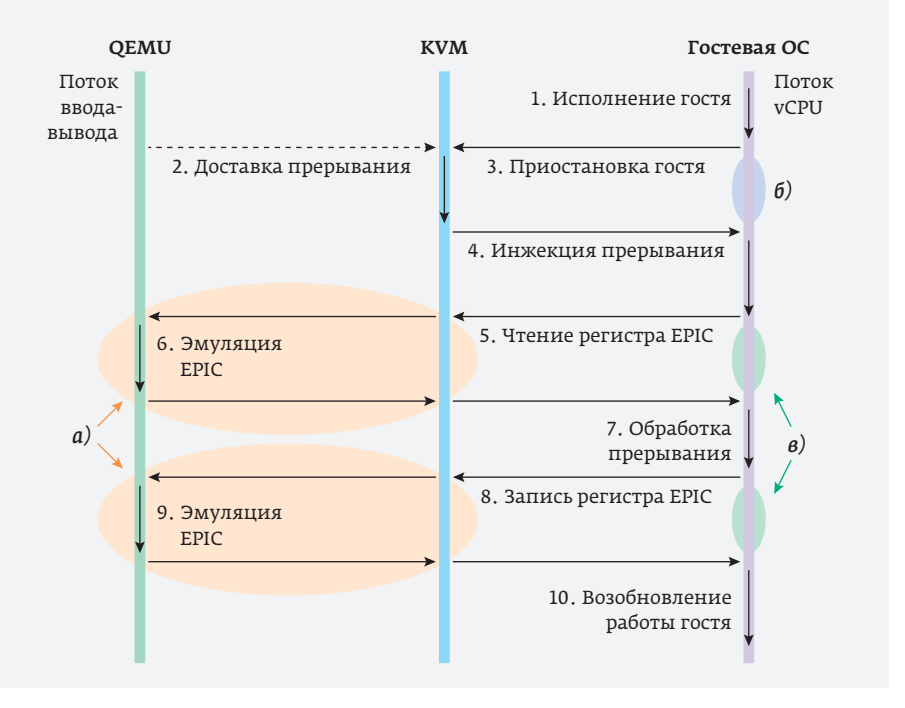

Рис. 3. Доставка и обработка гостевого прерывания без аппаратной поддержки виртуализации

ядром, которое исполняется в текущий момент на физическом ядре. При снятии виртуального ядра с исполнения гипервизор сохраняет состояние управляющих регистров гостя в память. При возобновлении работы гостевого ядра – восстанавливает сохраненное состояние.

При этом если в процессорном ядре в каждый момент времени исполняется либо гость, либо гипервизор, то в контроллере прерываний оба набора работают одновременно и независимо друг от друга. Виртуальное прерывание может прийти в CEPIC даже во время исполнения гипервизора. В этом случае гостевое прерывание ожидает в регистрах, пока не запустится исполнение гостя.

Обслуживание гостевого набора регистров осложняет процедуру переключения гостей из-за необходимости сохранять в память и восстанавливать из памяти состояния регистров контроллера прерываний, однако дает возможность обрабатывать гостевые запросы к управляющим регистрам контроллера прерываний без перехвата.

Ввиду того, что в аппаратуре появляются гостевые сообщения о прерываниях, для изоляции прерываний гипервизора от гостевых прерываний (а также для изоляции друг от друга сообщений о прерывании разных гостей) сообщения сопровождаются идентификатором виртуальной машины (guest_ID) и базовым адресом для записи гостевых прерываний в память (guest_base).

Контроллер PREPIC реализует таблицу соответствия виртуальных и физических ядер (Destination Address Table, DAT): для каждого физического ядра, на котором запущено виртуальное ядро, в таблице хранится идентификатор виртуальной машины (guest_ID) и номер виртуального ядра (guest_dst). Для сообщения о прерываниях, проходящих через PREPIC, осуществляется ассоциативный поиск.

Наличие в таблице DAT строки с парой {guest ID, guest dst} означает, что целевое виртуальное ядро активно, при этом известно на каком физическом ядре оно исполняется. Сообщение о прерывании перенаправляется в соответствующий CEPIC для записи в гостевые регистры.

Если в таблице DAT отсутствует строка с парой {guest ID, guest dst}, то исполнение целевого виртуального ядра отложено. Аппаратура формирует атомарную запись в память для сохранения прерывания в копии регистра CEPIC. Адрес для записи вычисляется из базового адреса (guest_base) и номера виртуального ядра (guest_dst).

При доставке межпроцессорных прерываний значения guest_ID и guest_base берутся из регистров CEPIC. При доставке внешних прерываний – как результат трансляции MSI-сообщений в блоке управления памятью для операций ввода-вывода (IOMMU). Соответствующие регистры CEPIC и элементы таблицы трансляции настраиваются гипервизором при постановке виртуального ядра на исполнение.

Подход к доставке виртуальных прерываний с реализацией полноценного набора управляющих регистров контроллера прерываний отличается от подхода Intel: в рамках технологии Virtualization Technology for Directed IO [5] гостевые запросы к регистрам контроллера прерываний сводятся к обращениям в оперативную память, при этом аппаратура эмулирует побочные эффекты при обращении к отдельным регистрам.

ПОДДЕРЖКА АКТУАЛЬНОГО СОСТОЯНИЯ ТАБЛИЦЫ СООТВЕТСТВИЯ ВИРТУАЛЬНЫХ И ФИЗИЧЕСКИХ ЯДЕР

Сложности подхода:

- сохранение и восстановление состояния CEPIC – не атомарные операции;
- состояние таблицы соответствия виртуальных и физических ядер должно быть одинаковым во всех компонентах PREPIC многопроцессорной системы, изменение строки в таблице – не атомарная операция;
- виртуальное прерывание может прийти в любой момент, в том числе в процессе снятия / запуска на исполнение виртуального ядра.

При снятии гостя с исполнения (после вычеркивания соответствующей строки в таблице DAT) в системе могут остаться сообщения о прерываниях, которые уже прошли процедуру поиска по таблице. Эти прерывания могут быть потеряны, если гипервизор начнет откачивать состояние в память до того, как прерывание дойдет до CEPIC.

При постановке гостя на исполнение (после обновления строки в DAT) в системе могут остаться прерывания в виде незавершенных записей в память. Эти прерывания могут быть потеряны, если гипервизор начнет восстанавливать состояние из памяти до того, как прерывания будут записаны в память.

Для исключения этих ситуаций аппаратура предоставляет гипервизору возможность точно определить моменты:

- доставки в целевой CEPIC всех сообщений о прерываниях, для которых уже найдено соответствующее физическое ядро;
- завершения всех связанных с доставкой прерывания отложенному виртуальному ядру DMA-операций, сформированных до изменения состояния строки в таблице.

Это достигается через программно-аппаратные механизмы, посредством которых происходит изменение строк в DAT:

1. Запись в регистр CEPIC формирует сообщение об изменении строки в DAT.

2. Служебное сообщение проходит через PREPIC вместе с потоком сообщений о прерываниях.
3. Каждый PREPIC выдает две квитанции: первая квитанция формируется, когда вычеркивается строка в DAT; вторая квитанция формируется, когда завершаются все DMA-записи для доставки прерываний, которые пришли в PREPIC до служебного сообщения.
4. CEPIC собирает все квитанции и сбрасывает статус.

С точки зрения программного обеспечения изменение строки в DAT происходит атомарно: запись в регистр CEPIC, затем чтение в цикле в ожидании сброса статуса.

ДОСТАВКА ГОСТЕВЫХ ПРЕРЫВАНИЙ С АППАРАТНОЙ ПОДДЕРЖКОЙ

Внешнее прерывание вновь генерируется эмулируемым в QEMU устройством и попадает в KVM. Перед инжекцией прерывания проверяется, в каком состоянии находится виртуальное ядро, которому предназначено прерывание, в данный момент времени – активном или нет. Переключение между этими состояниями с точки зрения доставляющего прерывание потока является атомарным. Если виртуальное ядро активно, то гипервизор инжектирует прерывание, задав регистр CEPIC. Если указанное виртуальное ядро снято с исполнения, то его контекст CEPIC был ранее сохранен в особую область памяти, и гипервизору достаточно взвести (установить равным единице) бит прерывания в этой области. Прерывание в этом случае будет доставлено сразу после следующей постановки гостя на исполнение.

Схема доставки межпроцессорных гостевых прерываний выглядит несколько иначе. Если рассмотренные ранее внешние прерывания генерировались эмулируемыми устройствами ввода-вывода в QEMU, то источником этого типа прерываний является гостевая ОС. Предположим, например, что гостевое ядро № 0 в ходе своего исполнения решает послать прерывание гостевому ядру № 1. Межпроцессорное прерывание формируется записью идентификатора ядра № 1 в гостевой регистр CEPIC. Обработка записи не требует участия гипервизора и исполняется аппаратно, без перехвата. Контроллер PREPIC затем проверяет, исполняется ли гостевое ядро № 1 в данный момент. Если исполняется – прерывание доставляется напрямую в гостевые регистры CEPIC. В противном случае прерывание доставляется через DMA-запись в сохраненный в оперативную память контекст отложенного ядра.

Вместо эмуляции некоторого устройства ввода-вывода гипервизор может передать гостю в пользование реальное физическое устройство. Такой подход, названный пробросом, или прямым назначением устройства гостю, позволяет добиться наилучшей производительности работы с устройством ввода-вывода. Однако, при передаче гостю контроля над устройством, у него появляется возможность по ошибке или намеренно генерировать прерывания с произвольным вектором [6]. Для защиты от такого поведения гостя были произведены доработки блока управления памятью для операций ввода-вывода, где прерывания транслируются перед их доставкой гостевой ОС. Трансляция в IOMMU гарантирует, что прерывания будут доставлены тому гостю, которому назначено устройство, и не повлияют на работу других гостей и гипервизора. Прошедшие трансляцию прерывания от внешнего устройства затем доставляются по аналогии с межпроцессорными прерываниями.

❊ ❊ ❊

Таким образом, реализация аппаратной поддержки виртуализации в новом поколении микропроцессоров семейства «Эльбрус» позволила избавиться от задержек в гостевых системах, связанных с эмуляцией аппаратного обеспечения. В новых проектах используется ряд программно-аппаратных механизмов, позволяющих сократить время доставки виртуальных прерываний: дублирование в аппаратуре управляющих регистров контроллера прерываний, аппаратная реализация таблицы соответствия виртуальных и физических ядер, механизмы для исключения потери прерываний при снятии/постановке виртуальной машины на исполнение.

В частности, эти механизмы позволяют доставлять виртуальные прерывания без приостановки исполнения гостя, а также избавиться от большинства перехватов при обращении гостем к регистрам контроллера прерываний. Благодаря программной оптимизации оставшиеся перехваты обращений к контроллеру обрабатываются модулем операционной системы Linux KVM, без тяжеловесных выходов в QEMU.

ЛИТЕРАТУРА

1. **Деменко Р. В., Трофимов В. Б.** Аппаратная поддержка виртуализации системы прерываний в микропроцессорах семейства «Эльбрус» // Вопросы радиоэлектроники. 2018. № 2. С. 40–44.
2. **Bellard F.** QEMU, a Fast and Portable Dynamic Translator // Proceedings of the FREENIX Track: 2005 USENIX Annual Technical Conference. P. 41–46.
3. **Kivity A., Kamay Y., Laor D. et al.** KVM: The Linux virtual machine monitor // Proceedings of the 2007 Ottawa Linux Symposium (OLS). 2007. P. 225–230.
4. **Pratt I., Fraser K., Hand S. et al.** Xen 3.0 and the art of virtualization // Proceedings of the 2005 Ottawa Linux Symposium (OLS). 2005. P. 65–77.
5. Intel Virtualization Technology for Directed I/O, Architecture Specification. – Intel, 2016.
6. **Bugnion E., Nieh J., Tsafrir D.** Hardware and Software Support for Virtualization // Synthesis Lectures on Computer Architecture. 2017.