Д.В. Знаменский (ЗАО «МЦСТ»)

D. Znamenskiy

ВЫБОР ВАРИАНТОВ РЕАЛИЗАЦИИ СРЕДСТВ АППАРАТНОЙ ПОДДЕРЖКИ ВИРТУАЛИЗАЦИИ АРХИТЕКТУРЫ «ЭЛЬБРУС»

ALTERNATIVES OF HARDWARE VIRTUALIZATION SUPPORT IMPLEMENTATION FOR ELBRUS PROCESSOR ARCHITECTURE

Излагается подход к внедрению средств аппаратной поддержки виртуализации, основанный на стратегии развития виртуализации ПО «Эльбрус». Предложены режимы работы процессорного ядра «Эльбрус» с поддержкой виртуализации, варианты механизмов перехода между режимами, механизмы виртуализации локального контроллера прерываний. Приводится обзор и сравнение различных вариантов реализации двухуровневой трансляции адресов.

The paper describes an approach for hardware virtualization support implementation based on Elbrus software virtualization strategy. We suggest adding new processor core execution modes to support virtualization. We consider two mode switching mechanism alternatives and some local APIC virtualization techniques. We give a review and comparison of several nested address translation implementation options.

Ключевые слова: Эльбрус, виртуализация, гость, гипервизор, таблица страниц, виртуальные прерывания.

Keywords: Elbrus, virtualization, guest, hypervisor, page table, virtual interrupts.

Введение

Возрастающее значение приобретают технологии консолидации нагрузки и резервирования, а также сфера облачных вычислений. Их быстрое развитие обуславливает эволюцию технологий виртуализации, причём требования к производительности виртуализо-

ванных систем неуклонно возрастают. Практика показывает, что существенный прирост быстродействия можно обеспечить путём введения в архитектуру платформы средств аппаратной поддержки виртуализации. Различные варианты такой поддержки реализованы для архитектур x86, IA-64, ARM и SPARC.

Проблема актуальна и применительно к архитектуре «Эльбрус». С программной точки зрения виртуализацию в системы на этой основе планируется первоначально внедрять в форме паравиртуализации, при которой используется модифицированная гостевая операционная система (ОС), взаимодействующая с гипервизором через программно-аппаратный интерфейс. Поэтому в первую очередь необходимо обеспечить базовые средства аппаратной поддержки паравиртуализованных вычислений. В дальнейшем планируется расширить описанные базовые средства с целью повышения производительности виртуализованных систем и создания технической возможности перехода к полной виртуализации (запуску немодифицированных гостевых ОС).

В статье рассмотрены набор аппаратных средств поддержки виртуализации архитектуры «Эльбрус», а также возможные варианты его реализации.

1. Базовые средства аппаратной поддержки виртуализации

Режимы гостя и гипервизора

Анализируя существующий мировой опыт [1, 2, 3], можно принять за основу средств аппаратной поддержки виртуализации введение двух режимов работы процессорного ядра, один из которых предназначен для исполнения кода гипервизора, а другой – для исполнения кода гостевой ОС (рис. 1).

Предлагается ввести новые режимы ортогонально по отношению к существующим привилегированному и непривилегированному режимам, что позволит использовать в качестве гипервизора полноценную ОС с запущенными под ней пользовательскими («гиперпользовательскими») задачами, которые работают параллельно с гостевыми ОС. Пере-

ход из режима гипервизора в гостевой режим осуществляется по команде запуска гостевой ОС, обратные переходы выполняются либо при возникновении некоторых гостевых ситуаций, нарушающих виртуализацию (так называемый *перехват*), либо «добровольно» посредством поддержанного на уровне набора команд API (так называемый *гипервызов*). Средства управления гостевыми ОС (запуск и настройки поведения в гостевом режиме) доступны из привилегированного режима гипервизора (гиперпривилегированного режима).

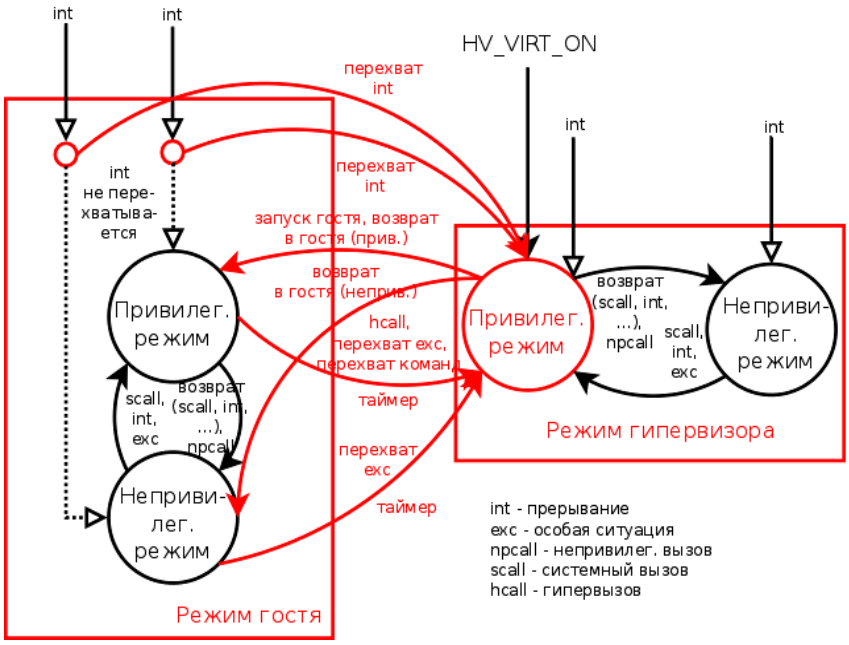

Рис. 1

Режимы работы процессора с поддержкой виртуализации

Работа в гостевом режиме

С точки зрения гостевой ОС гостевой режим работы процессорного ядра не отличается от работы ядра без поддержки виртуализации. Единственное программно видимое гостю расширение – команда гипервызова. При этом гипервизор может разрешить или запретить использование гипервызовов в гостевом режиме. Описанная настройка позволяет гипервизору либо «скрыть» от гостевой ОС наличие виртуализации, что подходит для техники полной виртуализации, либо оставить гостевой ОС возможность использования API взаимодействия с гипервизором (паравиртуализация).

В гостевом режиме предусмотрены следующие механизмы программно невидимого для гостя выхода в режим гипервизора:

• перехват нарушающих виртуализацию гостевых событий. К таким событиям относятся исполнение обращений к некоторым управляющим регистрам (например, регистрам MMU – Memory Management Unit), исполнение некоторых команд (работа с кэшами, выход из обработчика прерываний), а также прерывания и особые ситуации. Настройки механизма перехвата программно доступны в режиме гипервизора;

• срабатывание специального таймера, используемого планировщиком гипервизора;

• команда гипервызова, если она доступна в гостевом режиме.

Аппаратное переключение режимов

Возможны различные варианты реализации переходов между описанными выше режимами работы ядра. Переход состоит в переключении некоторого управляющего контекста ядра (подмножества управляющих регистров) и регистровых стеков. Предлагается сократить объём аппаратно переключаемого управляющего контекста, а часть состояния переключать программным образом в режиме гипервизора.

Для реализации переключения управляющего контекста ядра вводятся понятия архитектурного и теневого банков переключаемых регистров. К архитектурному банку относятся программно видимые, активные для данного режима управляющие регистры. Регистры теневого банка представляют собой копии архитектурных регистров, не активные в текущем режиме.

Возможны два варианта переключения режимов: с использованием памяти и теневого банка регистров, а также с использованием теневого банка регистров, но без использования памяти. В первом варианте управляющий контекст гостя хранится в физической памяти гипервизора в структуре некоторого заданного формата. Хранение состояния гостя в памяти используется в технологиях AMD SVM (Secure Virtual Machine) и Intel VT-x

(Virtualization Technology for x86). При входе в режим гостя происходит загрузка управляющего контекста на процессор, а при выходе в режим гипервизора – обратный процесс (откачка в память). Состояние гипервизора хранится на теневом банке регистров, при этом в режиме гипервизора теневые копии когерентны по записи относительно архитектурных регистров и программно недоступны гипервизору. Возможна также микропрограммная реализация переключения для этого варианта – она позволяет варьировать переключаемый контекст и поддержать произвольный формат области хранения гостевого состояния в памяти.

Во втором варианте реализации переключения режимов происходит «смена ролей» регистровых банков при переходах: архитектурный банк становится теневым, а теневой – архитектурным. К достоинствам этого варианта относятся отсутствие запросов в память при переключении и, как следствие, более высокая скорость переключения. Недостаток состоит в необходимости предоставления программного доступа к регистрам теневого банка в режиме гипервизора. Тем не менее, представляется, что с точки зрения формата команды «Эльбрус» резервы для предоставления такого доступа имеются.

Действие механизма переключения регистровых стеков зависит от направления переключения. При входе в режим гостя из гипервизора предлагается полностью откачивать регистровые стеки гипервизора в память, используя существующую spill-механику. При обратном переходе возможно как полное переключение с откачкой гостевых стеков, так и работа гипервизора на стеке гостя (без откачки). Второй вариант предполагает изменение существующей механики работы стека для режима гипервизора.

2. Расширения средств аппаратной поддержки виртуализации

Виртуализация локального APIC

Виртуализация LAPIC (локального контроллера APIC) в ядре процессора «Эльбрус» даст возможность повысить быстродействие системы в целом за счёт более тонкой

настройки перехвата прерываний и обслуживания части прерываний в гостевом режиме. Подобная виртуализация уже представлена в технологиях AMD SVM [2] и Intel VT-x [4] для архитектур x86. Планируется реализовать такие механизмы как:

- перехват прерываний на уровне LAPIC (маскируемых и немаскируемых) по векторам, задаваемым гипервизором в специальном регистре;
- маскирование физических прерываний с определёнными гипервизором векторами;
- команда End-Of-Interrupt с произвольным вектором прерывания в режиме гипервизора, которая позволяет гипервизору завершать обработку физических прерываний в ситуациях, когда гостевая ОС оставляет физические прерывания не обслуженными;
- инжекция виртуальных прерываний в гостевую ОС из гипервизора. С точки зрения гостевой ОС виртуальные прерывания не отличаются от физических. При совпадении типов физического и виртуального прерываний гонку выигрывает физическое прерывание, виртуальное прерывание выдаётся в гостевом режиме после окончания обслуживания физического. Возможно полное «отстранение» гостя от обслуживания физических прерываний при помощи механизма виртуализации страницы LAPIC, при котором обращения к физической странице LAPIC заменяются обращениями к виртуальной странице LAPIC. При этом в гостевом режиме выдаются только виртуальные прерывания, отображающиеся на виртуальной странице LAPIC, а физические прерывания обслуживаются в режиме гипервизора.

Двухуровневая трансляция адресов

Одним из наиболее существенных факторов, влияющих на производительность виртуализованных систем, является механизм трансляции гостевых адресов (физических и виртуальных) в физические адреса системы (гипервизора). Часто используемый при виртуализации механизм теневой таблицы страниц требует вмешательства гипервизора в работу гостевой ОС (например, при попытке модификации таблицы страниц), что суще-

ственно снижает быстродействие для целого класса задач, требующих активной работы с таблицами страниц. Поэтому для повышения быстродействия предлагается внедрить в архитектуру поддержку *двухуровневой* (*вложенной*) трансляции гостевых адресов, позволяющую избежать издержек, связанных с вмешательством гипервизора в работу гостевой ОС с гостевой таблицей страниц. По данным VMware [5, 6], прирост производительности для AMD-V (механизм RVI – Rapid Virtualization Indexing) составляет до 42%, для Intel VT-x (механизм EPT – Extended Page Tables) – до 48%.

При двухуровневой трансляции используются две аппаратно поддержанные таблицы страниц: гостевая таблица, предназначенная для трансляции виртуальных гостевых адресов в гостевые физические адреса, и специальная таблица гипервизора – для трансляции гостевых физических адресов в физические адреса гипервизора. В TLB (Translation Lookaside Buffer) при этом заносится полная трансляция адреса (из гостевого виртуального в системный физический). Во избежание сброса содержимого TLB при переключении режимов виртуализации каждая строка TLB расширяется идентификатором адресного пространства, уникальным для каждой гостевой ОС и использующимся как расширение к тэгу при вычислении попадания в TLB. Гостевая ОС самостоятельно обслуживает свою таблицу страниц, что позволяет избежать издержек, связанных с вмешательством гипервизора.

Возможны различные варианты реализации описанного механизма в зависимости от формата таблицы страниц гипервизора. Наиболее очевидный, реализованный в SVM и VT-x вариант – линейный многоуровневый формат, совпадающий с существующим форматом таблицы страниц. Существенным недостатком этого варианта является резкое возрастание как среднего, так и максимального количества обращений в память, связанных с поиском по таблицам страниц при промахе в TLB и кэшах таблиц. В общем случае количество обращений N для полного поиска по всем уровням обеих таблиц страниц задаётся формулой:

$$N = G\,(H + 1) + H, \qquad (1)$$

где G – количество уровней в гостевой таблице страниц, H – количество уровней в таблице страниц гипервизора. Для RVI «глубина» таблиц $H = G = 4$, что согласно (1) даёт $N = 24$.

Для снижения «стоимости» промаха по TLB и кэшам таблиц в механизме RVI реализованы [7] следующие оптимизации:

- занесение уровней обеих таблиц в кэш таблиц Page Walk Cache за исключением некоторых уровней с низкой степенью повторного использования;
- добавление Nested TLB (NTLB) – специального TLB на 16 строк, предназначенного для хранения трансляций гостевых физических адресов в физические адреса гипервизора. Попадание в NTLB для данного уровня гостевой таблицы страниц позволяет избежать поиска по таблице страниц гипервизора;
- преимущественное использование страниц памяти гипервизора размером 2 Мбайт.

Архитектура «Эльбрус» имеет свою специфику, которую нужно учитывать при реализации механизма двухуровневой трансляции адресов. Во-первых, используется 40-разрядный физический адрес, что даёт возможность вместо четырёх полноценных уровней таблицы страниц хранить в памяти лишь три уровня, а корневой уровень (level0) реализовать в виде двух регистров-указателей (по аналогии с расширением 32-разрядных физических адресов PAE – Physical Address Extension для x86). Таким образом, для архитектуры «Эльбрус» согласно формуле (1) получаем $G = 4$, $H = 3$, $N = 19$ (рис. 2).

Во-вторых, все уровни таблицы страниц, описывающей некоторое виртуальное пространство, загружаются в это же виртуальное пространство (кэшируются в TLB). Для таблицы гипервизора применение такого подхода оказывается затруднительно из-за существенно меньшего размера и большей плотности использования физического пространства гостя по сравнению с размером и плотностью использования виртуального пространства соответственно. Одно из решений состоит в хранении строк различных уровней таблицы в TLB с расширением строки TLB признаком таблицы страниц. Этот вариант прост

в реализации, однако может привести к возникновению значительного числа ассоциативных конфликтов в TLB, снижающих быстродействие системы. Поэтому с точки зрения производительности предпочтителен другой вариант – реализация отдельного небольшого кэша для таблицы страниц гипервизора. Кроме того, рассматривается вариант реализации Nested TLB, аналогичный Nested TLB в AMD SVM [7].

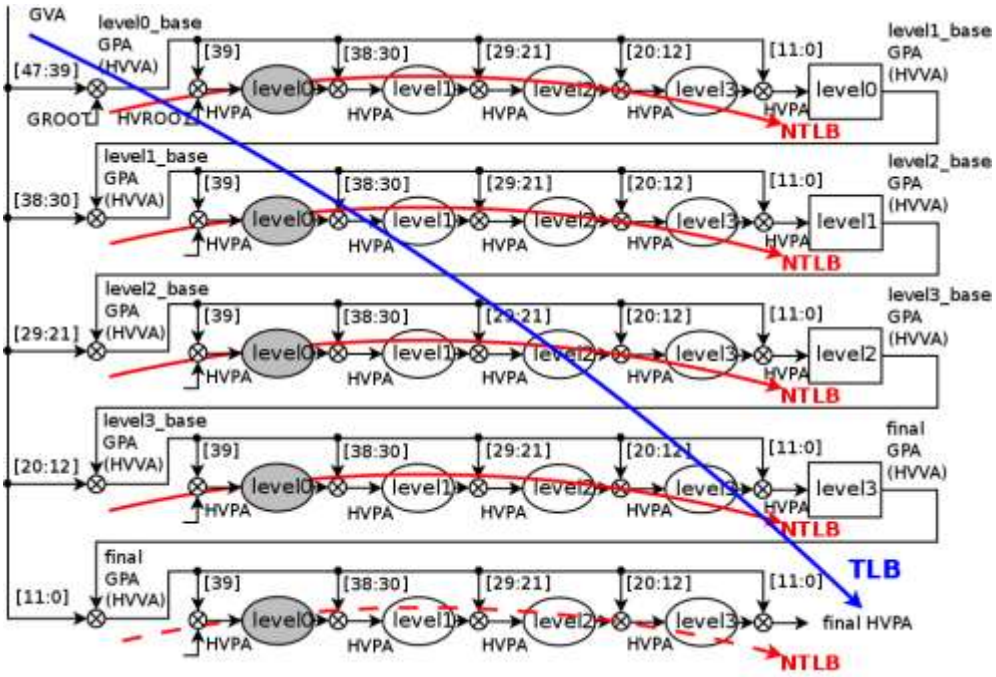

Рис. 2

Вариант реализации двухуровневой трансляции адресов с линейной таблицей гипервизора

Существуют альтернативные варианты форматов таблицы страниц гипервизора. Реализация плоской (flat) таблицы страниц [8] позволила бы сильно сократить время поиска: для описываемого случая согласно (1) получаем $G = 4$, $H = 1$, $N = 9$. В то же время, плоский формат таблицы страниц не подходит для большого, например 40-разрядного, физического пространства в силу неприемлемого размера таблицы. Однако для гостевых ОС, работающих в компактном 32-разрядном физическом пространстве, такой формат таблицы гипервизора даст существенный выигрыш в производительности. Кроме того, возможна реализация поддержки плоского формата таблицы параллельно с поддержкой любого другого формата таблицы страниц.

Хэшированная таблица страниц гипервизора [9], в отличие от плоской таблицы

страниц, реализуема для 40-разрядного физического пространства «Эльбрус». Количество уровней для хэшированной таблицы не определяется; аналогом этой характеристики является среднее количество коллизий на одну строку таблицы (при использовании техники chaining). Согласно [9] этот показатель можно минимизировать при наличии у гипервизора программного контроля над хэш-функцией путём её настройки в соответствии с картой гостевой физической памяти.

Заключение

В статье рассматриваются основанные на мировом опыте варианты аппаратной поддержки виртуализации архитектуры «Эльбрус». В дальнейшем планируется реализация некоторых из них исходя из требований ПО, прежде всего – ОС «Эльбрус». Кроме того, пока остаются открытыми вопросы виртуализации устройств ввода-вывода и поддержки бинарного транслятора.

Литература

1. Neiger et al. Intel Virtualization Technology: Hardware Support for Efficient Processor Virtualization. – Intel Technology Journal, Volume 10, Issue 03, 2006.

2. AMD64 Architecture Programmer`s Manual Volume 2: System Programming. – AMD, 2012.

3. Goodacre J. Hardware accelerated Virtualization in the ARM Cortex processors. – XenSummit Asia, 2011.

4. Intel 64 and IA-32 Architectures Software Developer`s Manual. Volume 3C: System Programming Guide, Part 3 – Intel, 2013.

5. Performance Evaluation of AMD RVI Hardware Assist. – VMware, 2008.

6. Performance Evaluation of Intel EPT Hardware Assist. – VMware, 2008.

7. Bhargava R., Serebrin B., Spadini F., Manne S. Accelerating Two-Dimensional Page

Walks for Virtualized Systems. – Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems, 2008.

8. Ahn J., Jin S., Huh J. Revisiting Hardware-Assisted Page Walks for Virtualized Systems. – Proceedings of the International Symposium on Computer Architecture, 2012.

9. Hoang G. et al. A Case for Alternative Nested Paging Models for Virtualized Systems. – Computer Architecture Letters, 2010.