DOI:

Аппаратная поддержка виртуализации архитектуры «Эльбрус»

Д. Знаменский[1], А. Блинников[2]

УДК 004.414.23 | ВАК 05.13.15

Виртуализация вычислительных ресурсов – одна из ключевых технологий, используемых в современных компьютерных системах, позволяющая значительно повысить степень утилизации оборудования и уровень безопасности, в частности, при организации облачных вычислений. Программная реализация этой технологии ведет к существенному снижению производительности системы по сравнению с нативной. Потери можно снизить, если передать часть программных механизмов аппаратуре. Для наиболее распространенных архитектур внедрение и развитие средств аппаратной поддержки виртуализации являются общей тенденцией, и архитектура микропроцессоров «Эльбрус» не исключение. В статье описаны основные средства аппаратной поддержки виртуализации, реализованные в версии 6 системы команд «Эльбрус».

ПРОГРАММНАЯ И АППАРАТНО ПОДДЕРЖИВАЕМАЯ ВИРТУАЛИЗАЦИЯ

С точки зрения контроля над аппаратными ресурсами (процессорным временем, памятью и внешними устройствами) в традиционной (нативной) вычислительной системе можно выделить два слоя программного обеспечения: прикладные программы и операционную систему (ОС), составляющую основу системного ПО. Роль последнего состоит в управлении аппаратурой и выделении ресурсов прикладному ПО. Виртуализация добавляет еще один, более привилегированный слой – монитор виртуальных машин, или гипервизор. Он также решает задачу управления ресурсами, но уже в отношении системного ПО, создавая последнему виртуальную аппаратную среду (виртуальную машину, ВМ). При этом ее ОС становится промежуточным слоем системного ПО (гостем) и не обладает полным контролем над аппаратными ресурсами. Гостевая ОС может учитывать наличие виртуализации, составляя логическую связку с гипервизором (паравиртуализация), или работать независимо от ее наличия (полная виртуализация). В обоих случаях задача гипервизора по виртуализации аппаратных ресурсов для гостевых ОС существенно сложнее задачи ОС хост-машины (нативной ОС) по распределению ресурсов между приложениями, так как, в отличие от последних, гостевые ОС привилегированы, архитектурно зависимы и имеют прямой доступ к аппаратуре. При этом объем виртуализуемых аппаратных ресурсов, как правило, больше объема ресурсов, заказываемых приложением у нативной ОС. Все это неизбежно приводит к значительному усложнению гипервизора относительно классической ОС и к существенным потерям быстродействия системы.

Изначально виртуализация была чисто программной технологией, основанной на исполнении кодов ВМ посредством двоичной трансляции или использовании паравиртуализованных гостей [1]. Можно выделить ряд узких мест программной виртуализации с точки зрения производительности и функциональности, а именно:

- необходимость двоичной трансляции продиктована общей привилегированной средой исполнения кодов гостевых ОС и гипервизора. Определенные последовательности привилегированных инструкций эмулируются гипервизором, что вносит значительные накладные расходы;
- работа нескольких ВМ на одном процессоре вносит дополнительные издержки на программное переключение их архитектурных состояний и усложняет гипервизор;
- дополнительные сложности могут возникать из-за неатомарности переключения: обработка внешних событий в неконсистентном состоянии машины должна быть исключена;
- блок управления памятью (Memory Management Unit, MMU) является общим для гипервизора и гостевых ОС. Так как гипервизор управляет системной памятью нативной машины, на него так или иначе ложится ответственность по управлению памятью приложений, работающих под гостевыми ОС. Кроме того,

[1] АО «МЦСТ», старший инженер, тел.: +7 499 135-62-22, znamen_d@mcst.ru.

[2] АО «МЦСТ», старший инженер, тел.: +7 499 135-33-61, blinnik_a@mcst.ru.

нарушается аппаратная изоляция адресных пространств гостей и гипервизора;
- гипервизор несет полную ответственность за распределение ВМ по физическим процессорам системы и управление периферией, что вызывает дополнительные издержки на обработку прерываний и эмуляцию внешних устройств.

Используя принцип разгрузки и ускорения ПО за счет переноса функциональности в аппаратуру, можно сформулировать список требований к архитектурным расширениям, направленным на преодоление перечисленных трудностей:
- режимы работы процессора, предназначенные для изоляции сред исполнения гипервизора и ВМ, должны допускать замену двоичной трансляции прямым исполнением кодов ВМ с дополнительными мерами по ограничению контроля последних над аппаратными ресурсами в пользу гипервизора;
- переключение между режимами должно осуществляться через исполнение специальных команд либо в ответ на определенные события (например, особые ситуации);
- атомарность переключения базового архитектурного состояния процессора должна ускорить переключение, упростить ПО и свести к минимуму или полностью исключить неконсистентность состояния аппаратуры при переключениях;
- аппаратная виртуализация ресурсов в режиме исполнения ВМ должна осуществляться в виде правомочной подмены ресурса, косвенного доступа к ресурсу через гипервизор и др.;
- виртуализация памяти ВМ прозрачным для нее способом должна по возможности обеспечить наименьшее вмешательство гипервизора в гостевые механизмы управления памятью и минимизировать потери на адресную трансляцию;
- механизмы виртуализации внешних устройств (ВУ) и прерываний, работающие в связке с виртуализацией процессора, должны учитывать вводимое архитектурное разделение слоев гостевого ПО и гипервизора и предоставлять техники абстрагирования ресурсов, которые позволят снизить участие гипервизора в работе гостевого ПО и упростят задачу доставки гостевых прерываний и распределения ВУ между ВМ.

В соответствии с этими требованиями в ряде ведущих зарубежных платформ, начиная с x86, были внедрены аппаратные средства, направленные на поддержку различных аспектов технологии виртуализации. Работа в этом направлении проведена и применительно к архитектуре «Эльбрус».

РЕЖИМЫ ПРИВИЛЕГИРОВАННОСТИ И ПЕРЕХОДЫ МЕЖДУ НИМИ

В версиях системы команд (СК) «Эльбрус», вплоть до 5-й включительно, были определены два режима привилегированности: пользовательский, предназначенный для прикладных программ, и привилегированный – для системного ПО, переходы между которыми происходят при исполнении и возврате из системных вызовов (команды scall и return), а также при обработке исключительных ситуаций и прерываний.

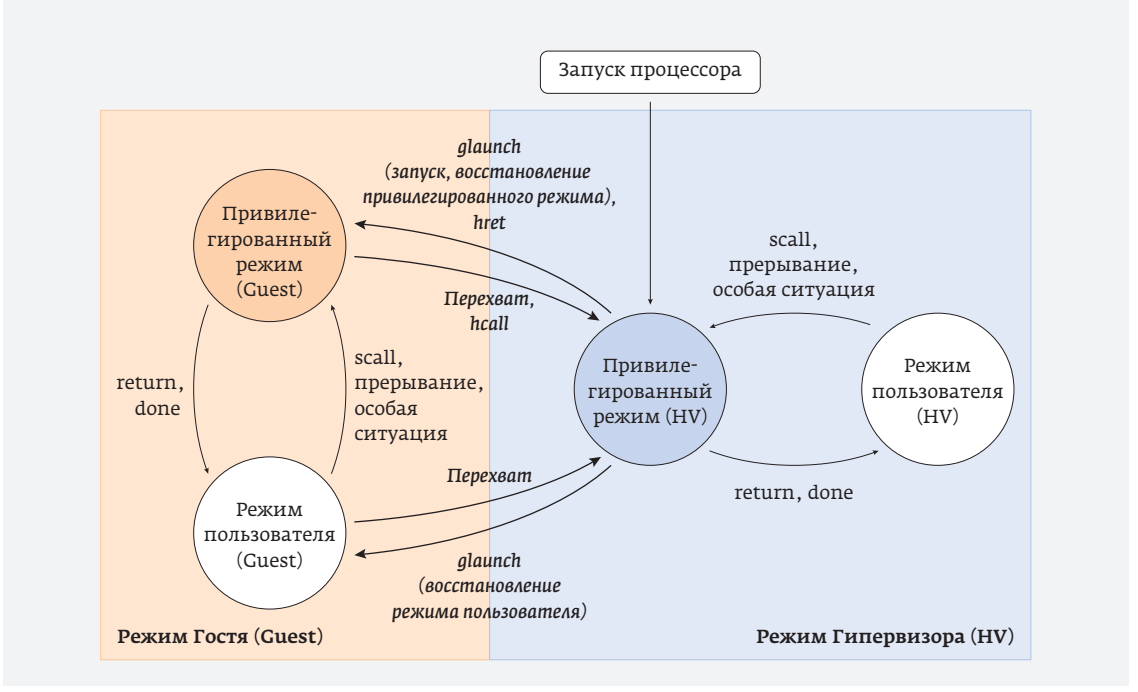

Рис. 1. Режимы работы процессора «Эльбрус» с поддержкой виртуализации

В версии 6 вводится разбиение на два дополнительных режима – режим гостя и режим гипервизора (рис. 1), в каждом из которых имеется свой привилегированный и пользовательский режимы. При этом с точки зрения доступной для ПО функциональности гостевые режимы практически не отличаются от невиртуализованных режимов СК «Эльбрус» версии 5, за исключением поддержки паравиртуализации. Все управление средствами поддержки виртуализации, а также переход в режим гостя доступны только в привилегированном режиме гипервизора, предназначенном для исполнения ядра ОС хост-машины. Пользовательский режим гипервизора эквивалентен невиртуализованному пользовательскому режиму. Указанные свойства позволяют поддерживать гипервизоры любого типа, в том числе тонкий гипервизор (такой, как Xen) и комбинированный гипервизор на основе ОС хост-машины, в частности Linux KVM, реализованный для архитектуры «Эльбрус». В последнем случае пользователем ОС может быть как обычное приложение, так и эмулятор QEMU, используемый в связке с Linux KVM для эмуляции гостевых устройств ввода-вывода.

Для вводимых в версии 6 СК режимов доступны дополнительные переходы. Первичный запуск гостевой ОС, а также возобновление ее работы осуществляются специальной командой glaunch. Выбор целевого гостевого режима (привилегированного или пользовательского) определяется архитектурным состоянием запускаемой ВМ. Обратный переход осуществляется по событию перехвата (interception) только в привилегированный режим гипервизора. Таким образом, события glaunch и interception образуют логическую связку запуска и остановки гостя, которая является основной для полной виртуализации. Кроме того, доступна еще одна логическая связка переходов гипервызовов (команда hcall) – возврат из гипервызова (команда hret), предназначенная для паравиртуализации и действующая в обратном направлении: гипервызов переводит процессор из привилегированного режима гостя в привилегированный режим гипервизора, а возврат из гипервызова совершает обратный переход. Использование гипервызовов позволяет снизить издержки на переключение архитектурного контекста для паравиртуализованных гостевых ОС.

РЕЖИМ ГИПЕРВИЗОРА

Далее, если не оговорено иное, будем рассматривать привилегированный режим гипервизора. В этом режиме доступны расширения СК, предназначенные для управления виртуализацией:

- команды передачи управления гостевому режиму glaunch и hret;
- дополнительные поля управляющих и статусных регистров;
- регистр-идентификатор гостя GID (Guest ID), используемый для маркировки гостевых адресных пространств и маршрутизации прерываний;
- комплект регистров так называемого теневого контекста, используемый при переключениях между гипервизором и гостем;
- регистры точки входа в гипервызов из гостевого режима;
- управление гостевым режимом с помощью регистров VIRT_CTRL_CU и VIRT_CTRL_MU для настройки перехватов, входа в гостевой режим, виртуализации таблиц страниц и др.;
- буферы перехватов устройства управления (INTC_INFO_CU) и подсистемы памяти (INTC_INFO_MU);
- управление гостевым временем;
- дополнительная функциональность регистров информации о прерываниях и др.

АТОМАРНОЕ ПЕРЕКЛЮЧЕНИЕ КОНТЕКСТОВ

Существуют различные способы реализации атомарного переключения контекстов. В системах Intel [2] и AMD [3] вводится атомарное аппаратное переключение через структуру данных в системной памяти (VMCS – для Intel VMX, VMCB – для AMD SVM). По команде переключения между гостем и монитором виртуальных машин аппаратура сохраняет состояние последнего в структуре данных и загружает гостевое состояние из этой структуры. При выходе из гостевого режима в монитор виртуальных машин выполняются обратные действия. Для архитектуры «Эльбрус» выбран другой подход, а именно – дублирование подмножества архитектурных регистров машины. Оригинальный комплект дублированных регистров называется активным контекстом, а комплект, вводимый для поддержки виртуализации – теневым контекстом. Каждый комплект включает в себя:

- некоторые управляющие регистры;
- указатели стека процедур и стека связующей информации процедурных переходов, а также дополнительный набор указателей стеков BACKUP-области (см. далее в разделе «Алгоритм переключения»);
- дескрипторы модулей компиляции ОС для функционирования в защищенном режиме архитектуры «Эльбрус»;
- часть контекста MMU, в частности, управляющий регистр MMU, базовый адрес таблицы страниц ОС, регистр-идентификатор задачи (Process ID).

Пассивным контекстом называется часть архитектурного состояния, не входящая в активный контекст, а именно, часть состояния машины, не имеющая аппаратной теневой копии. При входе в гостевой режим (например, по команде glaunch) активный и теневой контексты атомарно меняются ролями: активный становится теневым, а теневой – активным. Состав переключаемого

контекста определяется необходимостью функционирования гипервизора во время постановки ВМ на процессор: программная загрузка состояния ВМ из памяти на регистры теневого контекста не должна влиять на активное состояние и механизмы работы гипервизора. Собственный пассивный контекст гипервизор сохраняет программно. В этом смысле реализация переключаемых контекстов в СК «Эльбрус» ближе к варианту MIPS VM [5], где вводится полноценная копия гостевого состояния для Coprocessor 0. С точки зрения СК описанный механизм требует дополнительных кодировок регистров, однако упрощает аппаратную реализацию переключения и ускоряет его благодаря отсутствию аппаратных обращений в память для сохранения контекста.

ПЕРЕХВАТЫ В ГОСТЕВОМ РЕЖИМЕ

Гостевая ОС имеет ограниченные права по сравнению с нативной. Ограничения касаются, в частности, управления состоянием процессора, системной памятью и прерываниями. Одна из ключевых задач гипервизора – регулирование правомочности доступа к перечисленным ресурсам со стороны запущенных гостевых ОС. По этой причине обработка некоторых событий или прямое выполнение определенных привилегированных действий гостем должны блокироваться и вызывать немедленную передачу управления гипервизору с сохранением сведений о причине выхода. Описанная механика называется перехватом (VM-Exit – в терминологии Intel VMX, interception или #VMEXT – в терминологии AMD SVM, Hypervisor trap – в терминологии ARM).

В архитектуре «Эльбрус» определены следующие классы перехватов:

- чтение и запись управляющих и статусных регистров ядра и MMU;
- часть особых ситуаций (exc_data_page, exc_instr_page, и др.) и внешние прерывания (маскируемые и немаскируемые);
- операции очистки TLB (Translation Lookaside Buffer) и кеш-памяти;
- обращения по физическому адресу;
- обращение к виртуализационным ресурсам.

Совокупность перечисленных перехватов позволяет виртуализовать ресурсы процессора прозрачным для гостя способом.

Виртуализация внешних прерываний допускает грубую настройку с перехватом всех прерываний и более тонкую, не требующую перехвата прерываний гостя. Тонкая настройка опирается на виртуализацию подсистемы прерываний на основе контроллера EPIC [6] и подсистемы ввода-вывода [7], которая позволяет отличать прерывания, предназначенные гипервизору и гостю, на уровне аппаратуры и предоставлять ВМ изолированный доступ к внешним устройствам.

В архитектуру введено особое событие уровня гипервизора, вызывающее перехват, – срабатывание специального таймера вытеснения гостя (G_PREEMPT_TMR). Таймер программируется гипервизором и позволяет аппаратно отсчитывать квант времени, выделенный гостю для исполнения на данном физическом ядре процессора.

АЛГОРИТМ ПЕРЕКЛЮЧЕНИЯ

Наиболее полное переключение состояния процессора происходит при выполнении команды glaunch и перехвата. Алгоритм переключения для glaunch включает следующие действия:

1. откачка аппаратных верхушек процедурного и связующего стеков в память гипервизора;
2. подкачка гостевых аппаратных верхушек процедурного и связующего стеков из BACKUP-областей.
3. «обмен» активного и теневого контекстов ядра и MMU. С этого момента аппаратура работает в гостевом режиме. Становятся активны механизмы виртуализации таблиц страниц (теневая таблица страниц или двухуровневая трансляция);
4. повторное выполнение перехваченных гостевых операций (детали механизма описаны далее);
5. при изъятии гипервизором прерываний и особых ситуаций – выход из гостевого обработчика прерываний;
6. передача управления гостевому коду.

Независимо от причины, перехват исключительных ситуаций и прерываний всегда вызывает вход в обработчик прерываний гостевого режима с выделением окна стека связующей информации. Обратный переход по команде glaunch может выполняться по сценариям, предполагающим различные варианты действий гипервизора, отличающиеся исполнением шага 5.

Алгоритм переключения при перехвате представляет собой обратную последовательность шагов, дополненную сохранением специфической информации о перехвате.

ГИПЕРВЫЗОВЫ

Для поддержки паравиртуализации введен механизм гипервызовов, опирающийся на команду hcall (аналог инструкции VMMCALL в AMD SVM) и обратную команду hret. Паравиртуализованная гостевая ОС с помощью команды hcall может явно вызывать гипервизор, передавая ему параметры. При этом полной откачки гостевых стеков, как при перехвате, не требуется, что значительно сокращает издержки на переключение и позволяет широко использовать «легкие» гипервызовы. Подобно системным вызовам из режима пользователя, для команды hcall определен набор точек входа, которые явно задаются гостем.

СОХРАНЕНИЕ И ПОВТОРНОЕ ВЫПОЛНЕНИЕ ПЕРЕХВАЧЕННЫХ ОПЕРАЦИЙ

Перехват операции всегда предполагает ответные действия гипервизора. К таковым можно отнести устранение причины перехвата или модификацию поведения операции. Для разбора перехваченной ситуации и ответных действий гипервизору требуется информация о гостевых операциях и событиях, случившихся в момент перехвата. Кроме того, перехват операции (если он не аварийный), будучи прозрачным для гостя, должен завершаться фактическим ее выполнением в гостевом контексте в неизменном или виртуализованном виде. В СК «Эльбрус» введены буферы перехватов INTC_INFO, используемые для сохранения и повторного выполнения перехваченных операций. Буфер INTC_INFO_CU соответствует операциям доступа к управляющим регистрам CU, прерываниям и особым ситуациям, а буфер INTC_INFO_MU – операциям доступа к управляющим регистрам MU и обращениям в память. Сохранение информации и программный доступ к буферам организованы по принципу FIFO, а их объем достаточен для хранения максимального числа невыполненных заявок. Механика повторного выполнения работает в обратную сторону: гипервизор загружает соответствующий буфер данными операции (в неизменном виде либо подменяя ее аргументы при необходимости), а само повторное выполнение происходит на шаге 4 алгоритма glaunch и является прозрачным для гостя.

ТЕНЕВАЯ ТРАНСЛЯЦИЯ АДРЕСОВ

Для эффективной поддержки паравиртуализации в архитектуру «Эльбрус» введен программно-аппаратный механизм теневой трансляции адресов (рис. 2). Гостевая ОС формирует и видит свою таблицу страниц. Но в действительности она не используется при трансляции виртуальных адресов и даже не видна аппаратуре. Гипервизор полностью заменяет ее на так называемую теневую таблицу страниц (Shadow Page Table), которая используется для трансляции адресов устройством TLU (Table Look-up Unit). Есть существенное ограничение: структура и формат элементов теневой таблицы полностью совпадают с таблицей гостевой ОС, а алгоритмы аппаратного прохода по таблицам страниц должны быть полностью идентичны. Из этого вытекают два преимущества теневой трансляции адресов:

- для аппаратной поддержки требуются минимальные изменения;
- нет дополнительных накладных расходов на трансляцию адреса.

Этот механизм в перспективе может быть использован для эмуляции работы с любыми видами таблиц страниц гостевой ВМ через замену на таблицу, которую поддерживает платформа «Эльбрус». Отметим, что программный механизм SPT традиционно применяется для различных архитектур (например, virtual TLB в терминологии Intel [2]), однако для архитектуры «Эльбрус» режим функционирования с теневой таблицей страниц (ТС) определен аппаратно.

В результате прохода по теневой таблице информация о трансляции виртуального адреса в физический заносится в TLB. Помимо этого, в архитектуре Эльбрус предусмотрен кеш для хранения PTE (Page Table Entry) промежуточных уровней – PWC (Page Walk Cache). Трансляция адреса начинается с поиска PTE предпоследнего уровня в PWC, и в случае нахождения остается выполнить лишь одно обращение к таблице страниц последнего уровня. Виртуальный адрес в TLB, PWC и кешах первого уровня кода и данных расширен контекстом, образуемым идентификатором гостя (GID) и идентификатором процесса (PID), который позволяет осуществить защиту памяти между гипервизором и различными гостевыми ВМ, а также между различными процессами ОС и пользователей.

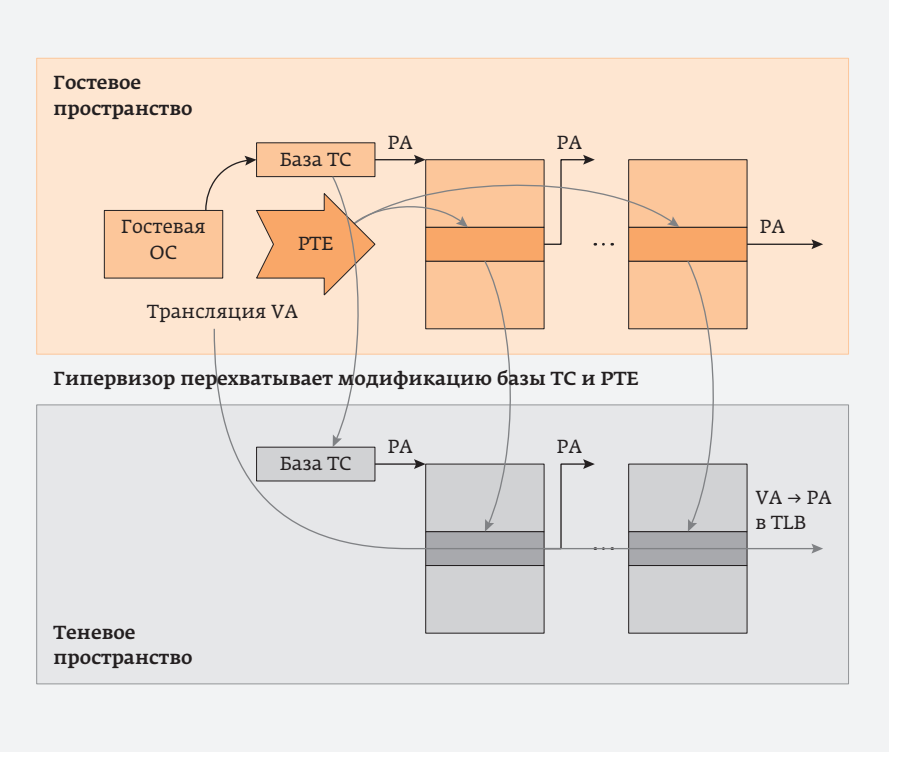

Рис. 2. Схема теневой трансляции адресов

При переключении задач и гостевых ВМ очищать TLB, PWC и прочие кеши не требуется.

Для переноса изменений из ТС гостя в теневую таблицу и наоборот организованы перехваты ошибок трансляции (page faults). Обработка данных page fault в ОС, как правило, завершается выделением страницы в оперативной памяти (для page miss) либо разрешением доступа по записи (для nwrite page, ОС используют технику copy-on-write) [8]. Информация о перехваченной операции, включая виртуальный адрес, сохраняется в буфере INTC_INFO_MU. По этому адресу гипервизор может определить ход трансляции по ТС и проанализировать состояние PTE в гостевой и теневой таблицах.

Процесс поиска изменений в таблице страниц гостя может быть достаточно долгим, особенно если гость выполнил операцию полной очистки TLB. Как правило, это используется после больших изменений в ТС. В этом случае невозможно быстро локализовать изменения, поскольку неизвестны виртуальные адреса. Поэтому в архитектуре «Эльбрус» применяется широко распространенная оптимизация – гипервизор отображает таблицу гостя в теневой таблице, навсегда закрыв доступ по записи [9]. Любая модификация гостевых PTE перехватывается, при этом в INTC_INFO_MU сохраняется адрес PTE и записываемое значение.

ГОСТЕВЫЕ ФИЗИЧЕСКИЕ АДРЕСА

Теневые таблицы страниц могут использоваться для трансляции не только виртуальных, но и физических адресов, когда гостевая ОС находится на стадии инициализации и еще не создавала свои ТС. Архитектура «Эльбрус» предоставляет возможность системному ПО выполнять обращения в память непосредственно по физическому адресу с помощью специальных операций, которые широко применяются в драйверах устройств. Если режим трансляции виртуальных адресов уже включен, то для таких обращений теневые таблицы уже не применимы. Поэтому в аппаратуре дополнительно введен механизм трансляции гостевых физических адресов GPA (Guest Physical Address). Трансляция осуществляется по таблице страниц физических адресов гостя GP PT (Guest Physical Page Table), которую формирует гипервизор. В результате прохода по GP PT информация о трансляции GPA в физический адрес (для лучшего понимания принято говорить физический адрес хоста – HPA) заносится в TLB.

ДВУХУРОВНЕВАЯ ТРАНСЛЯЦИЯ АДРЕСОВ

Важнейшим шагом на пути к полной виртуализации является снятие ограничений на поддержку гипервизором гостевой таблицы страниц ТС. В пределе гипервизор не должен вмешиваться в механизм трансляции гостевых виртуальных адресов, может не знать деталей устройства гостевого механизма трансляции и того, включен ли этот механизм у гостя. Гостевой механизм может быть произвольным и даже не поддержанным аппаратно. Такая независимость достигается введением отдельного этапа (уровня) трансляции гостевого физического адреса в физический адрес хоста. При использовании гостем виртуальной памяти два этапа трансляции адреса функционируют независимо. Отметим, что наилучшая производительность все же достигается при выборе гостем штатного архитектурного механизма трансляции адреса. Соответственно, в трансляции адреса участвуют две независимые таблицы страниц.

На первом этапе выполняется трансляция виртуального адреса (VA) в гостевой физический адрес (GPA) по таблице страниц гостевой ОС (таблица 1-го уровня). Если гость обращается по физическому адресу (GPA), то этот этап пропускается. На втором этапе выполняется трансляция гостевого физического адреса (GPA) в физической адрес хоста (HPA) по таблице GP PT, которую формирует гипервизор (таблица 2-го уровня). Схожий принцип использован в механизмах EPT [2], Nested Paging [3] и Two-Stage Address Translation [4]. Таким образом, область ответственности гостевой ОС и гипервизора строго разграничены и никак не пересекаются, что и является главным преимуществом:

- гипервизор не вмешивается в формирование гостевой ТС и не должен знать что-либо о ее организации;
- не нужно следить за гостевой таблицей страниц – большой выигрыш по отношению к теневой трансляции.

К недостаткам двухуровневой трансляции адресов по отношению к теневой трансляции относятся более дорогостоящая аппаратная реализация и дополнительный этап трансляции, причем для каждого уровня ТС гостя, что в пределе может увеличить суммарное количество шагов трансляции адреса на порядок. На данный момент в мировом сообществе продолжаются исследования по применению различных видов таблиц страниц в качестве GP PT для сокращения накладных расходов (в частности [10]), но однозначных выводов сейчас сделать нельзя. Поэтому в архитектуре «Эльбрус», как и в x86-платформах, в качестве GP PT выбрана обычная многоуровневая таблица страниц. В результате прохода по таблицам страниц в TLB заносится трансляция VA или GPA в HPA. Для компенсации накладных расходов в TLU также кешируются PTE гостя промежуточных уровней в PWC, которые содержат трансляцию в HPA, то есть уже после выполнения второго этапа трансляции, а также GP PTE промежуточных уровней в GPC (Guest Physical Cache). Весь цикл двухуровневой трансляции, представленный на рис. 3, который выполняется в случае промаха в TLB, в лучшем случае предполагает одно обращение к PTE гостя (#20) и одно обращение к GP PTE (#24).

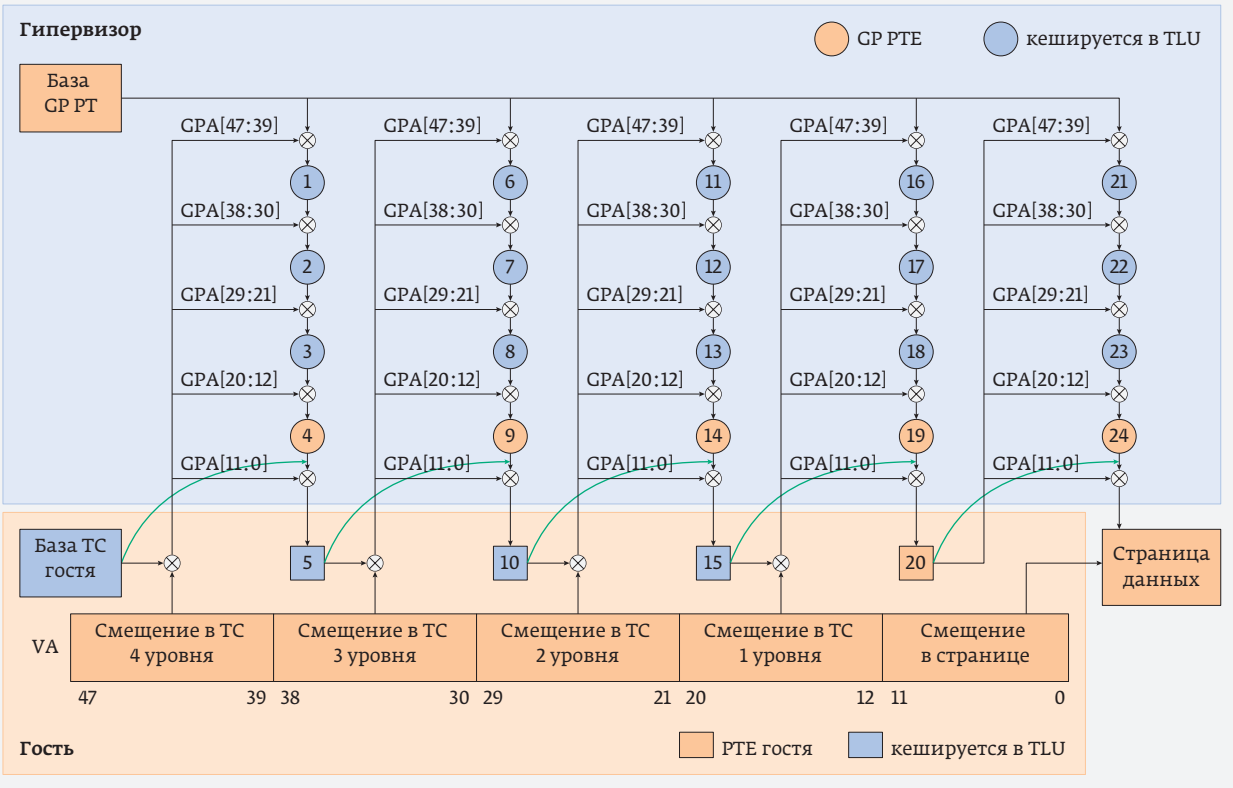

Рис. 3. Схема двухуровневой трансляции адресов

❖ ❖ ❖

В статье представлена совокупность расширений аппаратной поддержки виртуализации системы команд «Эльбрус», включающая режимы работы, переходы между режимами и средства виртуализации основных ресурсов процессора. Функциональность вводимых архитектурных расширений сопоставима с аналогичными решениями для архитектур x86, ARM и MIPS. Дальнейшее развитие средств аппаратной поддержки виртуализации направлено на повышение производительности и расширение функциональности виртуализации подсистемы памяти.

ЛИТЕРАТУРА

1. VMWare Understanding Full Virtualization, Paravirtualization and Hardware Assist. p. 4. March 11, 2008. – www.vmware.com.
2. Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3C: System Programming Guide, Part 3, Order Number: 326019–060US, September 2016. – www.intel.ru.
3. AMD64 Architecture Programmer's Manual Volume 2: System Programming, rev.3.33, AMD Corp. 2020. – www.amd.com.
4. Arm Instruction Set Version 1.0 Reference Guide, Issue 0100-00, 25 October 2018. – https://static.docs.arm.com.
5. MIPS64 Architecture for Programmers Volume IV-i: Virtualization Module of the MIPS64 Architecture, Document Number: MD00847 Revision 1.06, December 10, 2013. – www.mips.com
6. **Деменко Р. В., Трофимов В. Б.** Аппаратная поддержка виртуализации системы прерываний в микропроцессорах семейства «Эльбрус» // Вопросы радиоэлектроники. 2018. № 2. С. 40–44.
7. **Поляков Н. Ю.** Виртуализация подсистемы ввода-вывода микропроцессоров архитектуры Эльбрус. 12 марта 2020. – www.mcst.ru.
8. **Doeppner T.** Operating systems in depth: design and programming, published by John Wiley & Sons, 2011, pp. 310–313. – www.pdfdrive.com.
9. **Bugnion E., Nieh J., Tsafrir D.** Hardware and software support for virtualization, Morgan & Claypool Publishers, 2017, p. 61. – https://pdfs.semanticscholar.org.
10. **Yaniv I., Tsafrir D.** Hash, don't cache (the page table), SIGMETRIX, 2016. – https://cs.technion.ac.il.