РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 406 128** (13) **C2**

(51) МПК
G06F 17/00 (2006.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ,
ПАТЕНТАМ И ТОВАРНЫМ ЗНАКАМ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21), (22) Заявка: **2005136419/08, 23.11.2005**

(24) Дата начала отсчета срока действия патента:
23.11.2005

(30) Конвенционный приоритет:
30.12.2004 US 11/026,380

(43) Дата публикации заявки: **27.05.2007**

(45) Опубликовано: **10.12.2010** Бюл. № **34**

(56) Список документов, цитированных в отчете о поиске: **US 2005185207 A1, 25.08.2005. US 6496847 B1, 17.12.2002. RU 2192040 C2, 27.10.2002. RU 2193228 C2, 20.11.2002. US 2004268135 A1, 30.12.2004.**

Адрес для переписки:
129090, Москва, ул. Б.Спасская, 25, стр.3, ООО "Юридическая фирма Городисский и Партнеры", пат.пов. Ю.Д.Кузнецову, рег.№ 595

(72) Автор(ы):
БЛИТ Дэвид Р. (US)

(73) Патентообладатель(и):
МАЙКРОСОФТ КОРПОРЕЙШН (US)

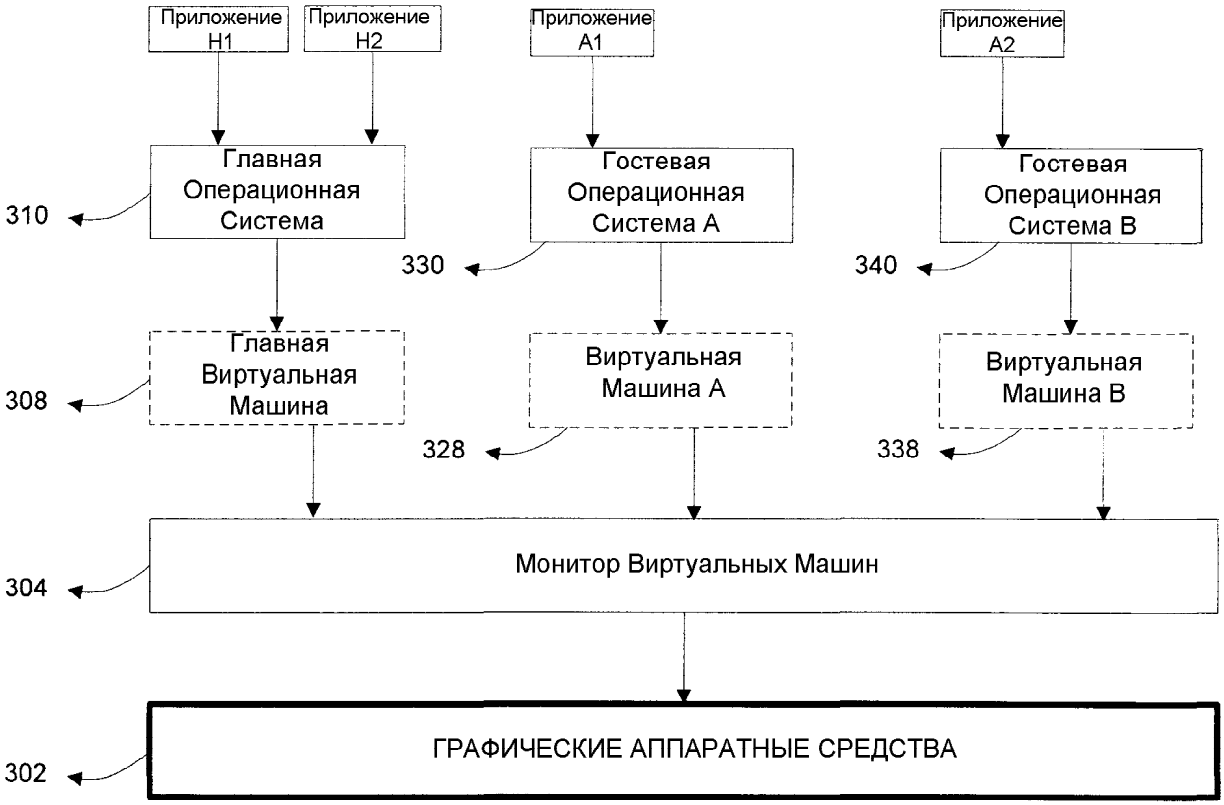
RU 2406128 C2

(54) СИСТЕМА И СПОСОБ ДЛЯ ВИРТУАЛИЗАЦИИ ГРАФИЧЕСКИХ ПОДСИСТЕМ

(57) Реферат:

Изобретение относится к системе и способу для применения виртуальных машин к графическим аппаратным средствам. Техническим результатом является повышение производительности за счет предоставления аппаратного доступа к ускорителям графических операций. В системе, в то время как код супервизора исполняется на центральном процессоре, фактические элементы графической работы исполняются непосредственно на графических аппаратных средствах, и код супервизора структурирован как графический монитор виртуальных машин. Совместимость приложений поддерживается за счет использования технологии монитора виртуальных машин для того, чтобы исполнять первую операционную систему, такую как первоначальная версия операционной системы, одновременно со второй операционной системой, такой как новая версия операционной системы, в отдельных виртуальных машинах. Технология монитора виртуальных машин, применяемая к главным процессорам, распространена на графические процессоры. Способ обеспечивает пользователю внешне единообразное впечатление при исполнении множественных приложений в различных виртуальных машинах. 4 н. и 15 з.п. ф-лы, 14 ил.

RU 2406128 C2



ФИГ. 3A

RU 2406128 C2

RU 2406128 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY,
PATENTS AND TRADEMARKS

(19) **RU** (11) **2 406 128** (13) **C2**

(51) Int. Cl.
G06F 17/00 (2006.01)

(12) ABSTRACT OF INVENTION

(21), (22) Application: **2005136419/08, 23.11.2005**

(24) Effective date for property rights:
23.11.2005

(30) Priority:
30.12.2004 US 11/026,380

(43) Application published: **27.05.2007**

(45) Date of publication: **10.12.2010 Bull. 34**

Mail address:
129090, Moskva, ul. B.Spasskaja, 25, str.3, OOO "Juridicheskaja firma Gorodisskij i Partnery", pat.pov. Ju.D.Kuznetsovu, reg.№ 595

(72) Inventor(s):
BLIT Dehvid R. (US)

(73) Proprietor(s):
MAJKROSOFT KORPOREJShN (US)

(54) SYSTEM AND METHOD OF VIRTUALISING GRAPHIC SUBSYSTEMS

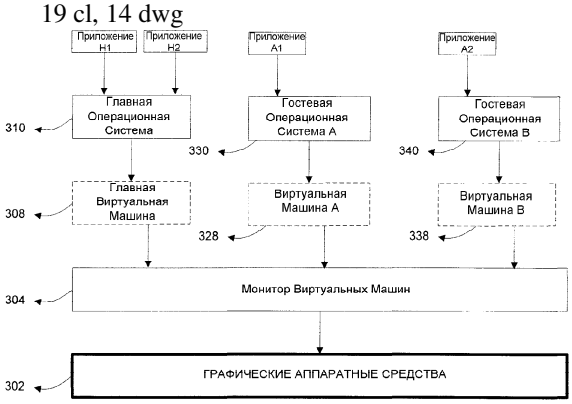
(57) Abstract:

FIELD: information technology.

SUBSTANCE: while supervisory code runs on the central processing unit, the actual graphics work items are run directly on the graphics hardware and the supervisory code is structured as a graphics virtual machine monitor. Application compatibility is retained using virtual machine monitor technology to run a first operating system such as an original operating system version, simultaneously with a second operating system, such as a new operating system version, in separate virtual machines. Virtual machine monitor technology applied to host processors is extended to graphics processors. The method provides a user with outward uniform impression while running multiple applications in different virtual machines.

EFFECT: high efficiency owing to provision for hardware access to graphic operations accelerators.

19 cl, 14 dwg



ФИГ. 3A

RU 2406128 C2

RU 2406128 C2

Часть описания данного патентного документа может содержать материал, который подпадает под охрану авторского права. Владелец авторского права не возражает против факсимильного воспроизведения кем бы то ни было данного патентного документа или патентного описания при его фигурировании в делах или документах Бюро по патентам и товарным знакам, но в иных случаях оставляет за собой все, какие ни есть, права по авторскому праву. К данному документу должно применяться следующее уведомление: Copyright © 2004, Microsoft Corp.

Область техники, к которой относится изобретение

Настоящее изобретение относится к системам, устройствам, способам, пользовательским интерфейсам, протоколам и интерфейсам прикладного программирования (API), предназначенным для виртуализации графических подсистем вычислительных устройств. Более конкретно, изобретение относится к виртуализированным архитектурам графических средств, которые предоставляют преимущества, связанные с совместимостью, защитой данных и управлением цифровыми правами.

Уровень техники

В порядке описания предшествующего уровня техники, касающегося состояния современного графического конвейера, отметим, что графические подсистемы обычно взаимодействуют с главным компьютером для выполнения от его имени определенных специализированных задач, например задач, которые требуют, фигурально выражаясь, много «грубой» вычислительной мощи, таких как подготовка выходных данных приложения для отображения на мониторе, или принтере, или другом устройстве. Например, для создания трехмерного компьютерного графического представления объекты, подлежащие изображению, представляются в виде математических моделей внутри компьютера, которые хорошо приспособлены для интенсивной математической обработки графическим процессором (GPU) графической подсистемы. К примеру, трехмерные модели могут состоять из геометрических точек в системе координат, состоящей из осей x, y и z, например, соответствующих ширине, высоте и толщине соответственно. Объекты определяются рядом точек, именуемых вершинами. Местоположение точки или вершины определяется ее координатами x, y и z (или другой системой координат). Согласно принятой в графике терминологии, одна вершина представляет собой точку, две вершины определяют прямую линию или отрезок прямой линии, и три вершины определяют треугольник, при этом все три понятия представляют собой «примитивы». При соединении трех или более этих точек образуется многоугольник, причем треугольник является простейшим многоугольником, который может быть использован для того, чтобы аппроксимировать трехмерную геометрию объекта и применять графические данные к трехмерной геометрии с тем, чтобы создавать разнообразие художественных и реалистических эффектов и трансформаций цветов. Графический конвейер, использующий разнообразные вычислительные субблоки, может обрабатывать вершины и другие потоки данных очень быстро и эффективно, так, чтобы, в конечном счете, представлять очень сложные трехмерные объекты в двухмерном пространстве отображения.

Таким образом, некоторые задачи, такие как визуализация (формирование визуального представления) и отображение трехмерной (3-D) графики на экране, обычно включают в себя много расчетов и вычислений, которые хорошо подходят для выполнения их графической подсистемой. В простой графической системе такие вычисления производятся в соответствии с некоторым уровнем взаимодействия или

совместной обработки данных со стороны центрального процессора (CPU) и графического процессора (GPU). Согласно приводимому в качестве примера сценарию, после того, как в центральном процессоре осуществлена обработка команд и произведены некоторые первоначальные вычисления, набор заданных своими координатами точек или вершин, которые определяют объект, подлежащий визуализации, сохраняется в видеопамяти для дальнейшей обработки графическим процессором в графическом конвейере. Если эти данные представляют собой, например, трехмерные графические данные, то блок мозаичного разбиения может разбить графические данные на простые многоугольники в соответствии с заранее заданными алгоритмами, разработанными для эффективного покрытия поверхности представляемого объекта - процесс известный как «мозаичное разбиение». На настоящий момент в большинстве графических конвейеров данные затем могут обрабатываться одним или более вычислительными субблоками графического процессора, такими как процедурные блоки затушевывания, в зависимости от команд, поступивших в графический процессор, который во взаимодействии с видеопамятью, то есть с буфером кадра, обрабатывает эти данные в соответствии с рабочими командами по этим данным и выводит эти данные туда, куда надлежащим образом указано, например, на устройство отображения или печатающее устройство.

Как показано на фиг.1 в отношении вычислительных систем, имеющих графические сопроцессорные системы, вычислительные системы делятся на главный центральный процессор и графические аппаратные средства. Центральный процессор способствует тому, чтобы делались вызовы графических интерфейсов прикладного программирования со стороны приложений и служб, которые запрашивают их использование. Традиционно, приложения и драйверы располагаются на стороне центрального процессора и информация из этих источников посылается в виде элементов работы в графический конвейер, такой как отображение данных на мониторе. Сначала, эта информация посылается из центрального процессора в графический процессор в виде, упакованном центральным процессором в соответствии с интерфейсами прикладного программирования. Затем информация из приложения ждет в памяти того момента, когда она будет запланирована на обработку и обработана вычислительными субблоками графического процессора, такими как блок (блоки) затушевывания вершин в случае, если обрабатываются трехмерные графические данные. После того как блок (блоки) затушевывания вершин заканчивает (заканчивают) свои операции, информация обычно выводится из блока (блоков) затушевывания вершин по специальному тракту данных в блок (блоки) затушевывания пикселей до того момента, как к ней будет осуществлен доступ блоком (блоками) затушевывания пикселей с целью ее дальнейшей обработки. После того как блок (блоки) затушевывания пикселей выполнили (закончили) свои операции, информация помещается в буфер кадра для вывода в виде развертки на устройство отображения или посылается назад в главный процессор для дальнейшей обработки.

Термин «буфер кадра» в современной архитектуре графических средств обычно относится к какой-либо памяти (обычно к видеопамяти, предусматриваемой для взаимодействия с графическим процессором), используемой в связи с процессами «растеризации» и/или процессами вывода из цифроаналогового преобразователя (DAC). В этом отношении отметим, что хотя термин «растеризация» иногда употребляется в более общем смысле, обычно под «растеризацией» понимают обработку данных, выполняемую в связи с обработкой пикселей, или обработку, выполняемую задающим механизмом, в графическом конвейере. С другой стороны,

развертка или вывод из цифроаналогового преобразователя (DAC) представляет собой процесс передачи сигналов на монитор или жидкокристаллический дисплей на основе содержимого буфера кадра.

Из-за специфики задачи и вовлеченных вычислительных ресурсов графические подсистемы компьютерных систем обычно используются для отображения графических объектов на экране устройства отображения. Цель трехмерной (3-D) компьютерной графики обычно состоит в том, чтобы создать двухмерные (2-D) изображения на экране компьютера, которые реалистически представляют объект или объекты в трех измерениях. В реальном мире объекты занимают три измерения, имея при этом реальную высоту, реальную ширину и реальную толщину. Примером двухмерного представления трехмерного пространства служит фотография. Трехмерная компьютерная графика, в общем, подобна фотографии в том, что она представляет трехмерный мир на двухмерном пространстве экрана компьютера, с тем исключением, что изображение, лежащее в основе этого, обычно смоделировано при помощи трехмерной геометрии и текстур поверхностей.

Изображения, созданные при помощи трехмерной компьютерной графики, используются в широком диапазоне приложений, от развлекательных видеоигр до тренажеров, имитирующих полет самолета, и предназначены для представления в реалистической манере того, как человек видит некоторую сцену в данный момент времени. Хорошо известные примеры трехмерной компьютерной графики включают в свое число специальные эффекты в голливудских фильмах, таких как Терминатор II (Terminator II), Парк юрского периода (Jurassic Park), История игрушки (Toy Story) и тому подобное. Одной из отраслей, которая испытала в последние несколько лет особенно бурный рост, является отрасль компьютерных игр, и современное поколение компьютерных игр применяет технологии трехмерной графики во все более возрастающих масштабах. В то же самое время скорость ведения игры становится все более и более быстрой. Это сочетание подпитывало истинную потребность в быстрой и гибкой визуализации трехмерной графики в относительно недорогих системах.

Однако в соответствии с этими быстро развивающимися тенденциями, связанными с графическими подсистемами, на повестку дня встали, соответственно, вопросы совместимости, защиты информации и управления правами. Поскольку каждое новое поколение графических процессоров и связанных с ними аппаратных средств вытесняет предыдущее поколение, то в дополнение к программным средствам должны развиваться и интерфейсы к конвейеру (например, DirectX8, DirectX9, GDI (Интерфейс графического устройства), GDI+ (Интерфейс графического устройства +) и т.д.), и если от них ожидается обработка вызовов, адресуемых «старым» интерфейсам, то требуется написание специального кода, который обеспечивает поддержку для более старых аппаратных средств. После более чем нескольких поколений вероятность того, что код графического интерфейса какой-либо одной конкретной операционной системы останется управляемым, уменьшается, а вероятность ошибок в обработке, вызванных несовместимыми вызовами функций увеличивается, поскольку укореняются эффекты ветвления кода, обеспечивающего совместимость. Дополнительную проблему представляет то, что развиваются полностью обособленные «родовые группы» кода графических интерфейсов, что вызвано существованием различных операционных систем или платформ (например, Macintosh, Linux, Windows и т.д.) или даже различных родовых групп, включенных в состав одной и той же платформы (например, OpenGL, GDI+, DirectX и т.д.). Таким образом, сохраняются проблемы, в силу которых различные версии операционных систем,

графических интерфейсов и связанных с ними аппаратных средств могут не поддерживаться в равной мере, и эти проблемы со временем становятся более сложными. Было бы желательным смягчить остроту этой проблемы без необходимости изменять составляющий ядро код, поставляемый с любой конкретной операционной системой или набором интерфейсов.

На том же самом фоне проблемы сохранности информации и защиты содержимого/управления цифровыми правами (DRM) становится более сложно решать в связи с графическим конвейером. При наличии различных операционных систем, интерфейсов и аппаратных средств отсутствует общий механизм, который обеспечивал бы сохранность данных или обеспечивал бы соблюдение условий защиты содержимого/управления цифровыми правами во всех известных конвейерах. Таким образом, требуется некоторый общий способ или уровень для решения вопросов сохранности данных и/или управления цифровыми правами, который работал бы независимо от графического конвейера.

Таким образом, было бы желательным виртуализировать графический конвейер компьютерной системы, предоставив виртуальную машину в качестве уровня взаимодействия, осуществляемого между главным процессором и графическим процессором. Кроме того, было бы желательным реализовать системы и способы, которые преодолевают недостатки, связанные с текущим положением дел в вопросах совместимости, сохранности информации и защиты содержимого/управления цифровыми правами, встречающиеся (недостатки) в случае, когда используются различные версии операционных систем или интерфейсов или различные архитектуры графических процессоров.

Раскрытие изобретения

Учитывая вышеозначенные и другие недостатки, имеющиеся в данной области техники, настоящее изобретение предусматривает системы и способы для применения виртуальных машин к пространству графических аппаратных средств. В различных вариантах осуществления изобретения, в то время как код супервизора исполняется на центральном процессоре, фактические элементы графической работы исполняются непосредственно на графических аппаратных средствах. В одном неограничивающем (не накладывающем ограничения на объем охраны изобретения) варианте осуществления изобретения код супервизора структурирован как графический монитор виртуальных машин.

Совместимость приложений поддерживается за счет использования технологии монитора виртуальных машин (VMM) для того, чтобы исполнять первую операционную систему (OS), такую как первоначальная версия операционной системы, одновременно со второй операционной системой, такой как новая версия операционной системы, в отдельных виртуальных машинах (VM). В одном варианте осуществления изобретения технология монитора виртуальных машин, применяемая к главным процессорам, распространена на графические процессоры (GPU) для того, чтобы предоставить аппаратный доступ к ускорителям графических операций, обеспечивая при этом, чтобы «унаследованные» приложения функционировали с полной производительностью. Изобретение также предоставляет способы сделать впечатление пользователя при исполнении множества приложений в различных виртуальных машинах внешне единообразным. В других аспектах изобретения посредством использования технологии монитора виртуальных машин виртуализированная архитектура графических средств по данному изобретению расширена таким образом, чтобы предоставлять «доверенные» службы и защиту

содержимого.

Другие преимущества и признаки изобретения описаны ниже.

Краткое описание чертежей

Системы и способы для виртуализации графических процессоров в соответствии с настоящим изобретением описываются далее со ссылкой на сопровождающие чертежи, на которых:

фиг.1 - блок-схема, иллюстрирующая реализацию графического конвейера на предшествующем уровне техники;

фиг.2A - блок-схема, представляющая разделение на логические уровни архитектуры аппаратных средств и программных средств для эмулированной операционной среды в некоторой компьютерной системе;

фиг.2B - блок-схема, представляющая виртуализированную вычислительную систему, в которой эмуляцию выполняет главная операционная система (либо непосредственно либо посредством гипервизора);

фиг.2C - блок-схема, представляющая альтернативную виртуализированную компьютерную систему, в которой эмуляцию выполняет монитор виртуальных машин, исполняемый параллельно с главной операционной системой;

фиг.3A - иллюстрация первой неограничивающей архитектуры для применения виртуальных машин к пространству графических аппаратных средств, благодаря которой код супервизора исполняется на центральном процессоре, а элементы графической работы обрабатываются непосредственно графическими аппаратными средствами;

фиг.3B - иллюстрация другой приводимой в качестве примера неограничивающей архитектуры, предусматриваемой для виртуализации графического конвейера в соответствии с изобретением;

фиг.4A - иллюстрация варианта осуществления изобретения, предназначенного для отображения «единственного рабочего стола единовременно» в соответствии с изобретением;

фиг.4B - иллюстрация альтернативного варианта осуществления «рабочий стол в окне», посредством которого все рабочие столы могут быть одновременно отображены в соответствии с изобретением;

фиг.4C - иллюстрация приводимой в качестве примера архитектуры для смешивания «доверенных» и «не наделенных доверием» данных приложений в соответствии с изобретением;

фиг.4D - иллюстрация расширения «доверенного» раздела за счет включения в него шлюза взаимодействия мультимедиа в качестве отдельного уровня для защиты содержимого;

фиг.4E - иллюстрация приводимого в качестве примера взаимодействия, при котором «унаследованное» приложение гостевой операционной системы, исполняемое на главной операционной системе, взаимодействует с компонентом ««унаследованные» службы» гостевой операционной системы с целью эмуляции «унаследованного» режима;

фиг.5 - блок-схема последовательности операций способа, представляющего неограничивающую последовательность операций по обработке графических данных в соответствии с одним или более вариантами осуществления виртуализированной архитектуры графических средств, предусматриваемой в соответствии с изобретением;

фиг.6A - блок-схема, представляющая приводимую в качестве примера сетевую среду, имеющую разнообразные вычислительные устройства, на которых может быть

реализовано настоящее изобретение; и

фиг.6В - блок-схема, представляющая приводимое в качестве примера неограничивающее вычислительное устройство, на котором может быть реализовано настоящее изобретение.

Осуществление изобретения

Обзор

Как было упомянуто, настоящее изобретение предусматривает системы и способы для применения виртуальных машин к пространству графических аппаратных средств. В различных вариантах осуществления изобретения в то время, как код супервизора, такого как графический монитор виртуальных машин, исполняется на центральном процессоре, фактические элементы графической работы исполняются непосредственно на графических аппаратных средствах.

В одном неограничивающем аспекте изобретения предусмотрены системы и способы, предназначенные для разделения функций графических аппаратных средств на привилегированные и непривилегированные операции и для эффективной виртуализации привилегированных операций при поддержании у пользователя впечатления единообразия. В другом неограничивающем аспекте изобретения достигнута эффективная виртуализация обработки аппаратных прерываний. В дополнительных аспектах изобретение предусматривает возможности по управлению ресурсами на уровне эмуляции для назначения разделам заранее заданного выделения ресурсов, таких как память, циклы обработки, место на экране и т.д., и/или для динамического назначения таких ресурсов в соответствии с параметрами/потребностями системы при перекрестной работе с приложениями и платформами.

Другой полезный результат этой архитектуры состоит в том, что настоящее изобретение позволяет виртуальному разделу иметь прямой доступ к графическим аппаратным средствам, не требуя внесения изменений в гостевую операционную систему или в приложения. Кроме того, из основной архитектуры по данному изобретению проистекает множество вариантов ее применения. Например, «унаследованный» код может быть оставлен в старых операционных системах при использовании для исполнения приложений, зависящих от «унаследованного» кода, экземпляра «унаследованной» операционной системы. Другим примером служит идея построения «доверенных» и «не наделенных доверием» служб, которые в свою очередь могут быть использованы для поддержки технологий вывода «доверенной» графики или защиты содержимого.

Другие системы и способы, предусмотренные в соответствии с изобретением, относятся к впечатлению пользователя. При наличии множественных разделов одно из впечатлений пользователя, предусматриваемых изобретением, представляет собой подход «рабочий стол в окне», при котором отображаемый рабочий стол для каждой гостевой операционной системы перенаправляется в окно (или в чередующийся вид рабочего стола) на устройстве отображения (или на множественных устройствах отображения в случае множественных мониторов, иногда называемых «мультимоном»). В других вариантах своего осуществления изобретение делает возможным впечатление пользователя, которое позволяет окнам из различных гостевых операционных систем быть смешанными на единственном унифицированном отображении, при этом производится идентификация «пикселей», связанных с каким-либо конкретным окном. В одном варианте своего осуществления изобретение обеспечивает идентификацию «пикселей» и их связей с окном посредством

добавления в гостевую операционную систему «вспомогательных программ» для идентификации частей различных окон. В других вариантах своего осуществления изобретение делает возможным взаимодействие с гостевым «администратором окна», обеспечивающее, чтобы незагороженные окна на основном рабочем столе были незагороженными и на виртуальном рабочем столе гостевых операционных систем так, чтобы они могли быть извлечены и отображены на основном рабочем столе. В других вариантах осуществления изобретения предусматривается поддержка графических аппаратных средств с тем, чтобы оказывать частичное или полное содействие в идентификации адресов ячеек памяти для пикселей из различных окон.

Изобретение также включает в себя варианты осуществления, предусматривающие дополнительную поддержку графических аппаратных средств с целью оказания содействия в управлении окнами на виртуальном рабочем столе. При том, что это - область, в которой будущие операционные системы могут оказать дополнительную поддержку, чтобы сделать эту задачу более легкой, ее идея состоит в том, чтобы сделать возможной структуру, которая идентифицирует индивидуальные окна из различных гостевых операционных систем, и чтобы сделать эти данные доступными для унифицированной службы преставления (также иногда именуемой механизмом (системой) составления рабочего стола), которая составляет вместе множественные источники. Этот механизм имеет доступ к состоянию памяти графических данных из множественных «гостей», при том, что «гости» обычно не могут видеть вызовы функций и действия друг друга.

Один подход, который делает возможным эффекты сложного, многоуровневого составления в отображаемых выходных данных в отношении содержимого из соответствующих гостевых операционных систем, заключается в том, чтобы механизм представления «вытягивал» содержимое окон из различных гостевых операционных систем с целью построения составного рабочего стола. Альтернативный подход заключается в том, чтобы позволить гостевым операционным системам осуществлять визуализацию непосредственно на единственном основном рабочем столе, то есть осуществлять визуализацию в этой памяти, но использовать при этом специальную поддержку аппаратных средств для ограничения тех частей рабочего стола, на которых могут рисовать эти гостевые операционные системы. Например, управление допустимыми областями может осуществляться с использованием отдельного привилегированного механизма, аналогичного механизму, используемому для аппаратного вырезания окна, при котором список прямоугольников или растр определяют маску, которая указывает, какие пиксели доступны для записи. Этот подход можно было бы реализовать в виде операции «выталкивания», при которой гостевые операционные системы осуществляют рисование непосредственно на рабочем столе, а механизм представления управляет тем, где располагается доступная (доступные) для записи область (области) для каждой гостевой операционной системы.

В разнообразных других вариантах своего осуществления изобретение предусматривает системы и способы для идентификации и определения местоположения пикселей окон, визуализируемых первой операционной системой, и окон, визуализируемых второй операционной системой, и представления этих окон либо по одному единовременно, либо объединенными некоторым образом вместе для отображения. Изобретение также включает в себя системы и способы для выделения и назначения физических ресурсов, таких как видеопамять или системная память, второй операционной системе таким образом, чтобы к назначенным ресурсам вторая

операционная система имела прямой доступ, а неназначенные ресурсы были ей недоступны. Кроме того, изобретение предусматривает системы и способы для перехватывания и эффективной эмуляции привилегированных операций.

Другие более подробные аспекты изобретения описаны ниже, но сначала в нижеследующем описании приводится общий обзор и некоторый общепринятый словарь по виртуальным машинам и связанной с ними терминологии в том виде, в котором эти термины стали известны в связи с технологиями виртуализации операционных систем и главного процессора («центрального процессора»). При этом приводится набор терминов, который специалист обычной квалификации в данной области техники может найти полезным при чтении следующего далее описания устройства, систем и способов для предоставления возможностей эмуляции на стороне графической подсистемы вычислительных систем в соответствии с изобретением.

Обзор виртуальных машин

Компьютеры включают в себя центральные процессоры (ЦП) общего назначения или «процессоры», которые разработаны для того, чтобы исполнять конкретный набор системных команд. Группа процессоров, которые имеют схожую архитектуру или конструктивные параметры, может рассматриваться как члены одного и того же семейства процессоров. Примеры современных семейств процессоров включают в себя семейство процессоров Motorola 680X0, изготавливаемое International Business Machines (IBM) или Motorola, Inc. из г. Феникс, Аризона (Phoenix, Arizona); семейство процессоров Intel 80X86, изготавливаемое Intel Corporation из Саннивэйл, Калифорния (Sunnyvale, California); и семейство процессоров PowerPC, которое изготавливает Motorola, Inc. и которое используется в компьютерах, изготавливаемых Apple Computer, Inc. из Купертино, Калифорния (Cupertino, California). Хотя группа процессоров может находиться в одном и том же семействе вследствие их схожей архитектуры и конструктивных соображений, внутри семейства процессоры могут в широких пределах различаться между собой по своей тактовой частоте и другим рабочим параметрам.

Каждое семейство микропроцессоров исполняет команды, которые являются уникальными для этого семейства процессоров. Совокупный набор команд, которые может исполнять процессор или семейство процессоров, известен как набор команд процессора. В качестве примера отметим, что набор команд, используемый семейством процессоров Intel 80X86, несовместим с набором команд, используемым семейством процессоров PowerPC. Набор команд Intel 80X86 основан на формате «Компьютер со сложным набором команд» (CISC). Набор команд Motorola PowerPC основан на формате «Компьютера с сокращенным набором команд» (RISC). CISC-процессоры используют большое количество команд, некоторые из которых могут выполнять довольно сложные функции, но которые требуют обычно много тактов для своего исполнения. RISC-процессоры используют меньший набор доступных команд для выполнения более простого набора функций, которые исполняются с гораздо более высокой скоростью.

Уникальность семейства процессоров среди компьютерных систем также обычно имеет своим результатом несовместимость среди других элементов архитектуры аппаратных средств компьютерных систем. Компьютерная система, изготовленная с процессором из семейства процессоров Intel 80X86, будет иметь архитектуру аппаратных средств, отличную от архитектуры аппаратных средств компьютерной системы, изготовленной с процессором из семейства процессоров PowerPC. Вследствие уникальности набора команд процессора и архитектуры аппаратных средств

компьютерной системы программы прикладного программного обеспечения обычно пишутся для исполнения на конкретной компьютерной системе, исполняющей конкретную операционную систему.

Вообще говоря, изготовители компьютеров пытаются максимизировать свою долю рынка, прилагая усилия для того, чтобы имелось больше, а не меньше приложений, исполняемых на семействе микропроцессоров, связанном с ассортиментной линией этих изготовителей компьютеров. Для расширения количества операционных систем и приложений, которые могут исполняться на некоторой компьютерной системе, получила развитие область технологии, при которой данный компьютер, имеющий один тип центрального процессора, именуемого главным, будет включать в себя программу-виртуализатор, которая позволяет главному компьютеру эмулировать команды некоторого не связанного с ним типа центрального процессора, именуемого «гостем». Таким образом, главный компьютер будет исполнять приложение, которое будет приводить к тому, что в ответ на данную гостевую команду (команду «гостя») будут вызываться одна или более команд главного процессора и таким образом главный компьютер может исполнять как программные средства, разработанные для его собственной архитектуры аппаратных средств, так и программные средства, написанные для компьютеров, имеющих не связанную с ним архитектуру аппаратных средств.

В качестве более конкретного примера, компьютерная система, изготовленная, например, в Apple Computer, может исполнять операционные системы и программы, написанные для компьютерных систем, основанных на PC. Кроме того, может иметься возможность использовать программы-виртуализаторы для исполнения параллельно на единственном центральном процессоре множественных несовместимых операционных систем. В этой последней ситуации, хотя каждая операционная система несовместима с другой, программы-виртуализаторы могут осуществлять хостинг каждой из нескольких операционных систем и тем самым позволяют исполнять несовместимые в иных случаях операционные системы параллельно на одной и той же главной компьютерной системе.

При эмуляции гостевой компьютерной системы на главной компьютерной системе гостевая компьютерная система называется «виртуальной машиной», поскольку гостевая компьютерная система существует в главной компьютерной системе только как чистое программное представление функционирования одной конкретной архитектуры аппаратных средств. Термины «виртуализатор», «эмулятор», «прямой исполнитель», «виртуальная машина» и «эмуляция процессора» иногда используются как взаимозаменяемые для обозначения способности имитировать или эмулировать архитектуру аппаратных средств всей компьютерной системы, используя для этого один или несколько подходов, которые известны и понятны специалистам в данной области техники. Кроме того, все случаи использования термина «эмуляция» в любой его форме подразумевают передачу этого широкого значения и не подразумевают проведения различия между концепциями эмуляции, связанными с исполнением команд, в противоположность непосредственному исполнению команд операционной системы в виртуальной машине. Таким образом, например, программное средство Virtual PC, созданное Connectix Corporation из Сан Матео, Калифорния (San Mateo, California), «эмулирует» (посредством эмуляции исполнения команд и/или непосредственного исполнения) весь компьютер, который включает в себя процессор Intel 80X86 Pentium и разнообразные компоненты материнской платы и карт, и функционирование этих компонентов «эмулируется» на виртуальной машине,

которая исполняется на главной машине. Программа-виртуализатор, исполняемая в программных средствах операционной системы и архитектуре аппаратных средств главного компьютера, такого как компьютерная система, имеющая процессор PowerPC, имитирует функционирование всей гостевой компьютерной системы.

Общий случай виртуализации позволяет одной процессорной архитектуре исполнять операционные системы и программы из других процессорных архитектур (например, программы для PowerPC Mac на архитектуре x86 Windows, и наоборот), но важным частным случаем является тот случай, когда лежащие в основе процессорные архитектуры являются одинаковыми (исполнение разнообразных версий x86 Linux или различных версий x86 Windows на x86). В этом последнем случае имеется потенциал для более эффективного исполнения гостевой операционной системы и ее приложений, поскольку лежащий в их основе набор команд является тем же самым. В таком случае гостевые команды допускаются к непосредственному исполнению на процессоре без потери при этом управления и оставления системы открытой для атак (то есть гостевая операционная система, как бы находится в «песочнице»). Как подробно описывается ниже, здесь на передний план выходят разделение команд на привилегированные в противоположность непривилегированным и технологии управления доступом к памяти. Для виртуализации, при которой имеется несоответствие архитектур (PowerPC <-> x86), могли бы быть использованы два подхода: покомандная эмуляция (сравнительно медленная) или трансляция из гостевого набора команд в собственный набор команд (более эффективный подход, но использует этап трансляции). Если используется эмуляция команд, то сделать среду робастной сравнительно просто; однако, если используется трансляция, то это отсылает нас назад к частному случаю, при котором процессорные архитектуры одинаковы.

Согласно изобретению производится виртуализация графических платформ и по этой причине приводимый в качестве примера сценарий, соответствующий применению изобретения к архитектурам графических средств, будет представлять собой эмуляцию карты ATI, использующей аппаратные средства NVidia. Для случая виртуализации графических средств изобретение в различных вариантах своего осуществления фокусируется на непосредственном исполнении команд из одной и той же архитектуры графических средств. В контексте графики эквивалента трансляции на самом деле не существует, или скорее эмуляция и трансляция сводятся к одному и тому же. Трансляция выполняется на процессоре по той причине, что после того, как код оттранслирован, этот код может исполняться множество раз, тем самым амортизируя стоимость трансляции. Однако вследствие того, каким образом структурированы графические приложения, эквивалент программы для трансляции (не учитывая блоки затушевывания) необязательно существует. Вместо этого поток команд (список элементов работы) постоянно регенерируется, так что необязательно, чтобы имелось какое-либо их повторное использование, которое могло бы амортизировать стоимость трансляции. В более сложном варианте осуществления изобретения списки элементов работы могли бы быть исследованы на предмет определения того, совпадают ли они с ранее оттранслированным списком.

Как описано в различных вариантах осуществления изобретения, приведенных ниже, согласно различным вариантам осуществления изобретения виртуализация при непосредственном исполнении гостевых команд (либо после трансляции, либо вследствие того, что используется такая же процессорная архитектура) имеет

выигрыш в производительности, если может быть обеспечена безопасность системы. Таким образом, в различных вариантах своего осуществления изобретение описывает системы и способы для управления доступом «гостя» к некоторым или всем лежащим в основе физическим ресурсам (памяти, устройствам и т.д.).

Программа-виртуализатор действует как звено обмена между архитектурой аппаратных средств главной машины и командами, передаваемыми программными средствами (например, операционными системами, приложениями и т.д.), исполняемыми внутри эмулированной среды. Эта программа-виртуализатор может представлять собой главную операционную систему (HOS), которая представляет собой операционную систему, исполняемую непосредственно на аппаратных средствах физического компьютера (и которая может содержать гипервизор, описанный более подробно ниже в данном документе). В качестве альтернативы, эмулированная среда могла бы также представлять собой монитор виртуальных машин (VMM), который представляет собой уровень программных средств, исполняемый непосредственно над уровнем аппаратных средств, возможно, при параллельном исполнении и функционируя в сопряжении с главной операционной системой, и который может осуществлять виртуализацию всех ресурсов главной машины (равно как и некоторых виртуальных ресурсов), выставляя интерфейсы, которые одинаковы с аппаратными средствами, виртуализацию которых осуществляет монитор виртуальных машин. Эта виртуализация предоставляет виртуализатору (равно как и самой главной компьютерной системе) возможность быть незамеченным уровнями операционных систем, исполняемых над ним.

Таким образом, эмуляция процессора предоставляет гостевой операционной системе возможность исполнения на виртуальной машине, созданной виртуализатором, исполняемым на главной компьютерной системе, содержащей как физические аппаратные средства, так и главную операционную систему.

С концептуальной точки зрения, компьютерные системы обычно содержат один или более уровней программных средств, исполняемых на служащем основанием системы уровне аппаратных средств. Это разделение на уровни сделано по причине абстракции. Благодаря определению интерфейса для данного уровня программных средств этот уровень может быть реализован иным образом посредством других уровней, расположенных над ним. В хорошо спроектированной компьютерной системе каждый уровень знает только о ближайшем находящемся под ним уровне (и полагается только на него). Это позволяет заменять уровень или «стопку» (множество примыкающих друг к другу уровней), не оказывая отрицательного воздействия на уровни, расположенные над вышеназванным уровнем или «стопкой». Например, программные приложения (верхние уровни) обычно полагаются на нижние уровни операционных систем (нижние уровни) в том, что касается записи файлов в некоторую форму постоянного запоминающего устройства, и этим приложениям не требуется понимать различие между записью данных на гибкий диск, в накопитель на жестких магнитных дисках или в сетевую папку. Если этот нижний уровень заменен новыми компонентами операционной системы, предназначенными для записи файлов, то функционирование программных приложений, расположенных на верхнем уровне, останется незатронутым.

Гибкость разделенных на уровни программных средств позволяет виртуальной машине (VM) представить слой виртуальных аппаратных средств, который на самом деле представляет собой еще один слой программных средств. Таким образом, виртуальная машина может создать для уровней программных средств,

расположенных над ней, иллюзию того, что эти уровни программных средств исполняются в своей собственной частной компьютерной системе, и, таким образом, виртуальные машины могут позволить множественным «гостевым системам» исполняться параллельно на единственной «главной системе». Этот уровень абстракции представлен на фиг.2А.

На фиг.2А показана схема, представляющая разделение на логические уровни архитектуры аппаратных средств и программных средств для эмулированной операционной среды в некоторой компьютерной системе. На этом чертеже программа (94) эмуляции исполняется непосредственно или опосредованно в архитектуре (92) физических аппаратных средств. Программа (94) эмуляции может представлять собой (a) монитор виртуальных машин, который исполняется наряду с главной операционной системой, (b) специализированную главную операционную систему, имеющую собственные эмуляционные возможности, или (c) главную операционную систему с компонентом «гипервизор», при этом упомянутый компонент «гипервизор» выполняет упомянутую эмуляцию. Программа (94) эмуляции эмулирует гостевую архитектуру (96) аппаратных средств (показанную в виде пунктирных линий для иллюстрации того факта, что этот компонент представляет собой «виртуальную машину», то есть аппаратные средства, которые на самом деле не существуют, но вместо этого эмулируются упомянутой программой (94) эмуляции). Гостевая операционная система (98) исполняется в упомянутой гостевой архитектуре (96) аппаратных средств, а программное приложение (100) исполняется в гостевой операционной системе (98). В эмулированной операционной среде, показанной на фиг.2А, и вследствие функционирования программы (94) эмуляции - программное приложение (100) может исполняться в компьютерной системе (90) даже в том случае, если программное приложение (100) разработано для исполнения в операционной системе, которая обычно несовместима с этой главной операционной системой и архитектурой (92) аппаратных средств.

Фиг.2В иллюстрирует виртуализированную вычислительную систему, содержащую уровень (104) программных средств главной операционной системы, исполняемый непосредственно над уровнем физических компьютерных аппаратных средств (102), причем главная операционная система (104) (главная ОС) предоставляет доступ к ресурсам физических компьютерных аппаратных средств (102), выставляя интерфейсы, которые одинаковы с теми аппаратными средствами, эмуляцию (или «виртуализацию») которых осуществляет главная операционная система, - что, в свою очередь, позволяет главной операционной системе оставаться незамеченной уровнями операционных систем, исполняемых над ней. И вновь, для того чтобы выполнять эмуляцию, главная операционная система (102) может представлять собой специально разработанную операционную систему с собственными эмуляционными возможностями или, в качестве альтернативы, она может представлять собой стандартную операционную систему со встроенным компонентом «гипервизором», предназначенным для выполнения эмуляции (не показанным на чертеже).

Обратимся вновь к фиг.2В, на которой над главной операционной системой (104) расположены две реализации виртуальных машин (VM): виртуальная машина А (108), которая, например, может представлять собой виртуализированный процессор Intel 386, и виртуальная машина В (110), которая, например, может представлять собой виртуализированную версию одного процессора из семейства процессоров Motorola 680X0. Над каждой виртуальной машиной (108) и (110) расположены гостевые операционные системы (гостевые ОС) А (112) и В (114) соответственно. Над гостевой

операционной системой A (112) исполняются два приложения: приложение A1 (116) и приложение A2 (118), а над гостевой операционной системой B (114) исполняется приложение B1 (120).

В отношении фиг.2B важно отметить, что виртуальная машина A (108) и виртуальная машина B (110) (которые показаны на чертеже пунктирными линиями) являются виртуализированными представлениями компьютерных аппаратных средств, которые существуют только в виде программных конструкций и которые стали возможны вследствие исполнения специализированной эмуляционной программы (эмуляционных программ), которая не только представляет виртуальную машину A (108) и виртуальную машину B (110) гостевой операционной системе A (112) и гостевой операционной системе B (114) соответственно, но которая также выполняет все содержащиеся в программных средствах этапы, необходимые для того, чтобы гостевая операционная система A (112) и гостевая операционная система B (114) опосредованно взаимодействовали с реальными физическими компьютерными аппаратными средствами (102).

Фиг.2C иллюстрирует альтернативную виртуализированную компьютерную систему, в которой эмуляцию выполняет монитор виртуальных машин (VMM) (104'), исполняемый наряду с главной операционной системой (104"). Для некоторых вариантов осуществления изобретения монитор виртуальных машин может представлять собой приложение, исполняемое над главной операционной системой (104) и взаимодействующее с компьютерными аппаратными средствами только через упомянутую главную операционную систему (104). В других вариантах осуществления изобретения, и как показано на Фиг.2C, монитор виртуальных машин может вместо этого содержать частично независимую программную систему, которая на некоторых уровнях опосредованно взаимодействует с компьютерными аппаратными средствами (102) через главную операционную систему (104), но на других уровнях монитор виртуальных машин непосредственно взаимодействует с компьютерными аппаратными средствами (102) (аналогично тому, как с компьютерными аппаратными средствами непосредственно взаимодействует главная операционная система). А в других еще вариантах осуществления изобретения монитор виртуальных машин может содержать полностью независимую систему программных средств, которая на всех уровнях непосредственно взаимодействует с компьютерными аппаратными средствами (102) (аналогично тому, как с компьютерными аппаратными средствами непосредственно взаимодействует главная операционная система), не используя при этом главную операционную систему (104) (хотя по-прежнему взаимодействуя с упомянутой главной операционной системой (104) постольку, поскольку это необходимо для координации использования упомянутых компьютерных аппаратных средств (102) и во избежание конфликтов и тому подобное).

Предусматривается, что все эти варианты реализации виртуальной машины образуют альтернативные варианты осуществления настоящего изобретения согласно приведенному здесь описанию, и ничто здесь не должно толковаться в смысле, ограничивающем изобретение каким-либо конкретным вариантом осуществления эмуляции. Кроме того, любая ссылка на взаимодействие между приложениями (116), (118) и (120) через виртуальную машину A (108) и/или виртуальную машину B (110), соответственно (предположительно в сценарии эмуляции аппаратных средств) должна толковаться как являющаяся на самом деле взаимодействием между приложениями (116), (118) и (120) и виртуализатором, который создал эту

виртуализацию. Подобным же образом, любая ссылка на взаимодействие между связанных с приложениями виртуальной машиной A (108) и/или виртуальной машиной B (110 и главной операционной системой (104) и/или компьютерными аппаратными средствами (102) (предположительно с целью непосредственного или опосредованного исполнения компьютерных команд на компьютерных аппаратных средствах (102)) должна толковаться как являющаяся на самом деле взаимодействием между виртуализатором, который создал эту виртуализацию, и главной операционной системой (104) и/или компьютерными аппаратными средствами (102) в зависимости от того, что уместно.

Системы и способы для виртуализации графических процессоров и графического конвейера

Как было упомянуто, настоящее изобретение предусматривает системы и способы для применения виртуальных машин к пространству графических аппаратных средств, благодаря чему код супервизора выполняется на центральном процессоре, а фактические элементы графической работы обрабатываются непосредственно графическими аппаратными средствами. На фиг.3A показана первая неограничивающая архитектура, которая иллюстрирует некоторые общие концепции. Например, как показано на иллюстрации, существует монитор (304) виртуальных машин, расположенный на уровне находящемся поверх уровня графических аппаратных средств (302) вычислительного устройства. Не накладывая ограничений, данный вариант осуществления изобретения рассматривает главную операционную систему как аналогичную гостевой операционной системе в том, что каждая из них получает виртуальную машину. Хостинг главной операционной системы (310), исполняющей приложение H1 и приложение H2, осуществляется виртуальной машиной (308). Хостинг гостевой операционной системы A (330), исполняющей приложение A1, осуществляется виртуальной машиной A (328). Хостинг гостевой операционной системы B (340), исполняющей приложение B1, осуществляется виртуальной машиной B (338). В одном неограничивающем варианте осуществления изобретения код супервизора структурирован как графический монитор виртуальных машин.

Цель изобретения состоит в том, чтобы предоставить непрекращающуюся поддержку для старых графических программных средств в более новых операционных системах и разнообразные виртуализированные архитектуры по данному изобретению достигают этой цели. Решения, которые переносят старые функциональные возможности вперед, в новую операционную систему, только усложняют новую операционную систему - результат, который можно избежать согласно архитектурам по данному изобретению. Решения, предусматриваемые в соответствии с изобретением, включают в себя исполнение виртуальной версии старой операционной системы наряду с новой операционной системой и поддержку старых графических программных средств только в экземпляре старой операционной системы. Это зиждется на понятии виртуальных машин и мониторов виртуальных машин, описанных в приведенном выше разделе и использованных в коммерчески доступных продуктах, таких как Virtual PC и VMware. Однако в этих основанных на PC архитектурах графические программные средства в «разделе» виртуальной операционной системы часто эмулируются и выполняются гораздо более медленно, чем в первоначальной невиртуализированной реализации. Поэтому в соответствии с данным изобретением стопке уровней графических программных средств предоставляется возможность продолжать иметь прямой доступ к лежащим в ее

основе графическим аппаратным средствам при исполнении ее под управлением старой (то есть гостевой) операционной системы. Это аналогично тому, что известно под названием Тип II монитора виртуальных машин, в котором гостевая операционная система может исполняться непосредственно на процессоре без дорогостоящей эмуляции посредством монитора виртуальных машин.

Для обеспечения того, чтобы множественные разделы операционных систем исполнялись на графических аппаратных средствах без ведома друг друга и не зная о том, что графические аппаратные средства были виртуализированы, требуется поддержка в графических аппаратных средствах. Поэтому в соответствии с вариантами осуществления изобретения управление ресурсами и контроль за тем, какой раздел исполняется на графических аппаратных средствах, осуществляется графическим монитором виртуальных машин. При использовании изобретения множественные гостевые операционные системы могут исполняться параллельно, и каждая из них может иметь (практически) полно-скоростной прямой доступ к графическим аппаратным средствам. Некоторые «гости» могут представлять собой «унаследованные» операционные системы, исполняющие более старое программное обеспечение, в то время как другие «гости» могут выполнять другие специфические задачи.

Расширение изобретения приводит к вариантам его осуществления, которые предусматривают защищенную обработку данных, причем защищенная обработка данных может производиться либо в «главной», либо в «гостевой» операционных системах. Достижение этого в соответствии с изобретением имеет положительный эффект, заключающийся в робастной и защищенной виртуализации графических аппаратных средств, такой, при которой гостевые операционные системы не могут вмешиваться в работу друг друга. Если гостевые операционные системы не могут вмешиваться в работу друг друга, то устраняется возможность жульнического манипулирования программными средствами в другой гостевой операционной системе без ведома пользователя. Система рассматривает монитор виртуальных машин в качестве «доверенной» базы или ««доверенного» ядра». В различных вариантах осуществления изобретения «доверенная» база может быть распространена на другую гостевую (или главную) операционную систему с тем, чтобы предоставить дополнительные «доверенные» графические службы (а не помещать все службы в монитор виртуальных машин или ядро). Эти службы могут включать в себя контроль за тем, что является видимым на устройстве отображения, например, посредством защищенного или «доверенного» администратора окна или механизма составления рабочего стола. Это позволяет исполнять смесь приложений в «доверенных» и «не наделенных доверием» разделах операционных систем при унифицированном выводе посредством единственного (или множественного) физического графического устройства.

Защищенная база может быть дополнительно распространена в соответствии с изобретением таким образом, чтобы предоставить более робастную систему для обеспечения защиты содержимого, например чтобы поддерживать цифровые права, прикрепленные к документу, видеоданным, аудиоданным и т.д. Такого рода система усиливает «доверенные» службы, действующие в «доверенном» мониторе виртуальных машин и «доверенном» разделе («доверенных» разделах) таким образом, чтобы обеспечивать применение цифровых прав. Однако пользовательский интерфейс и другие функции в приложениях воспроизведения (управление мультимедийными библиотеками, списки воспроизведения и т.д.) могут быть отделены и могут

исполняться в «не наделенном доверием» разделе, поддерживая при этом связь с защищенными службами в «доверенных» разделах.

По мере того, как заказчики переходили на новые операционные системы и новые версии операционных систем, совместимость главным образом обеспечивалась посредством кропотливого сохранения устаревших программных интерфейсов и режимов, что вело ко все возрастающим издержкам во время циклов разработки программного обеспечения. Сложность проектирования и реализации возрастает, а риск неумышленного нарушения «унаследованного» режима сохраняется на протяжении всего цикла разработки. В соответствии с этим изобретение предлагает альтернативную стратегию разработки, при которой «унаследованный» режим может быть стратегическим путем и благополучно удален из новой разработки при продолжающейся защите инвестиций заказчика.

Совместимость приложений сохраняется с использованием технологии монитора виртуальных машин (VMM) для одновременного исполнения первоначальной версии операционной системы с новой операционной системой в отдельных виртуальных машинах (VM). Традиционная технология монитора виртуальных машин главной машины распространена на графические процессоры (GPU) и позволяет осуществлять аппаратный доступ к ускорителям графических операций, что обеспечивает функционирование «унаследованных» приложений с полной производительностью. Изобретение также предусматривает способы, позволяющие сделать впечатление пользователя при исполнении множества приложений в различных виртуальных машинах внешне единообразным. В других аспектах изобретения посредством использования технологии монитора виртуальных машин виртуализированная архитектура графических средств по данному изобретению расширена таким образом, чтобы предоставлять «доверенные» службы и защиту содержимого/управление цифровыми правами.

Долгосрочный успех программных средств неизбежно приводит к «унаследованным» программным интерфейсам и режимам. Противоречащие друг другу требования новизны и обратной совместимости, предъявляемые к новым программным средствам, ведут к более сложным и более длительным циклам разработки программных средств. Попытки снизить возрастающую сложность за счет отбрасывания разрядов селекции в «унаследованных» программных средствах используют больше ресурсов для оценки решений, добавляют риск того, что будут приняты неправильные решения, и неизменно дают лишь минимально эффективный результат. Конечным результатом этого является низкая скорость появления новшеств и их внедрения в ключевых технологиях и фокусирование внимания на тех инновациях, которые не нарушают статус-кво.

Альтернативный подход заключается в том, чтобы охватить более старые версии программных средств операционной системы, перенося их в более поздние времена, но в то же самое время, держа их вне новой разработки. В соответствии с этим, изобретение использует монитор виртуальных машин для того, чтобы исполнять предыдущие версии операционной системы одновременно с самой последней версией операционной системы в отдельных виртуальных машинах, причем одновременно оставляя у пользователя приятное впечатление о том, как приложения, использующие более старые версии интерфейсов, единообразно или почти единообразно исполнялись в тех же самых средах, для которых они были разработаны, и вели себя тем же самым образом.

Прежде чем обрисовать различные варианты осуществления изобретения полезно

рассмотреть пример. В соответствии с ним предположим, что для второй версии некоторой операционной системы, скажем, OS_B, желательно освободиться от некоторого набора графических интерфейсов прикладного программирования, поддерживаемых в первой версии этой операционной системы, скажем, OS_A. Однако цель состоит в том, чтобы исполнять OS_B и OS_A одновременно, то есть приложения, использующие графические интерфейсы OS_A, исполняются на виртуальном «виде» OS_A, тогда как новые приложения исполняются на виде OS_B, оба из которых дают равным образом «хорошие» пользовательские впечатления. Определение того, что составляет «хорошее» пользовательское впечатление, несколько расплывчато, но некоторые основополагающие критерии, взятые в отдельности, могут включать в себя: (A) единообразный пользовательский ввод (клавиатура, мышь и т.д.) и вывод, осуществляемый приложением (на устройство отображения, печать, мультимедиа и т.д.) при использовании любого из видов операционной системы; (B) малое воспринимаемое различие в производительности или отсутствие воспринимаемого различия в производительности (один показатель, который может быть здесь применен, заключается в том, что производительность любого приложения OS_A, исполняемого на виде OS_A, должна отличаться не более чем на 5% от производительности того же самого приложения, исполняемого на собственной первой версии этой операционной системы без виртуального монитора); и (C) виртуальный вид OS_A должен использовать не измененные программные средства, которые использованы для первой версии этой операционной системы. Это является желательным по той причине, что как только «дверь открывается», позволяя вносить изменения в гостевую операционную систему, вновь возникают риски нарушения совместимости в результате ветвления, описанного в разделе «Предшествующий уровень техники». Один из способов ослабить это требование заключается в том, чтобы для обеспечения единообразного функционирования позволить дополнительным вспомогательным приложениям исполняться под управлением не измененной гостевой операционной системы.

Одним из вопросов, связанных с обеспечением минимального снижения производительности в пользовательском восприятии, является обеспечение аппаратного ускорения служб визуализации графических данных. Для интерфейса прикладного программирования, подобного интерфейсу графического устройства (GDI), может быть практически целесообразным эмулировать всю «стопку уровней» программных средств, но в случае более новых интерфейсов прикладного программирования, которые зависят от аппаратного ускорения, снижение производительности может быть недопустимо. Одна из возможностей заключается в том, чтобы перенаправить запросы на обработку графических данных из гостевой операционной системы в главную операционную систему, используя, например, программную эмуляцию/организацию прерываний нижнеуровневых интерфейсов аппаратных средств относящихся к хорошо известному графическому аппаратному устройству.

Хотя этот традиционный подход успешен в случае устройств с высоким уровнем абстракции (много работы, приходящейся на запрос) или простых устройств (например, последовательные порты), он не является практически целесообразным для современных графических аппаратных средств. Запросы на прерывание на этом уровне обычно слишком неэффективны (графические аппаратные средства спроектированы для приема входных данных со скоростью более чем 1Гб/с) и современные графические аппаратные средства, как правило, слишком сложны для

того, чтобы осуществлять надежную эмуляцию на этом уровне подробности (напомним, что современное графическое устройство имеет ~ 200 миллионов транзисторов). На каком-то этапе этот подход мог бы быть практически целесообразным для устройства, имеющего менее чем 10 миллионов транзисторов и для приложений с малой интенсивностью графических операций, но даже в этих обстоятельствах требуемый уровень производительности не был бы достигнут.

Другой возможностью является перехват потока протокола графических данных в некоторой более высокой (не зависящей от устройств) точке потока и перенаправление его в главную операционную систему. Хотя это и устраняет проблему, вызываемую эмуляцией сложного устройства, тем не менее, в результате могут возникнуть серьезные проблемы с производительностью в случае, если используются потоки протокола с высоким диапазоном рабочих частот. Другая проблема, которая может возникнуть в результате, относится к сценарию, в котором потоки протокола не поддаются перехвату без внесения значительных изменений в гостевую операционную систему. Один из подходов мог бы заключаться в замене библиотек динамических ссылок (DLL), содержащих графические интерфейсы прикладного программирования, на заглушки протокола для перенаправления потока в гостевую операционную систему, хотя и здесь все же имеются значительные проблемы, связанные с этим подходом, в случае, когда способы управления состоянием в первоначальных интерфейсах прикладного программирования не были разработаны с учетом этого подхода. Хотя любой интерфейс прикладного программирования, который был разработан для удаленного исполнения («удаленный») вероятно имеет приемлемый режим, который мог бы быть использован для достижения этой цели, не все графические и мультимедийные интерфейсы прикладного программирования были разработаны с учетом этого решения. И в этом случае данный подход требует изменения (замены) компонентов системного уровня и тем самым вносит риски, относящиеся к совместимости.

Подход различных вариантов осуществления изобретения заключается в том, чтобы вытолкнуть проблему на уровень аппаратных средств и продолжать позволять гостевой операционной системе иметь прямой доступ к аппаратным средствам, напоминающий технологии монитора виртуальных машин, используемые для центрального процессора. При эффективной реализации у этой технологии имеется несколько преимуществ. Первое преимущество заключается в устранении потребности во внесении изменений в гостевую операционную систему. Второе преимущество обеспечивает те же самые или чрезвычайно близкие уровни производительности, как если бы гостевая операционная система исполнялась непосредственно на этих аппаратных средствах. Следует также отметить, что эта схема позволяет «унаследованным» операционным системам поддерживать более новые графические аппаратные средства, пока поставщики графических аппаратных средств продолжают оказывать поддержку драйверами для новых аппаратных средств в «унаследованных» операционных системах. Таким образом, согласно разнообразным виртуализированным архитектурам по данному изобретению, созданы графические аппаратные средства, которые могут быть виртуализированы, гостевая операционная система и монитор виртуальных машин структурированы таким образом, чтобы эффективно эксплуатировать эти аппаратные средства, а представление (отображение) виртуализируется таким образом, что оно может быть интегрировано в составной рабочий стол.

Изобретение таким образом делает возможным доступ каждой виртуальной

машины к ресурсам визуализации графических данных, создает в графическом процессоре среду монитора виртуальных машин для сквозной виртуализации конвейера. В частности, это включает в себя идею эффективной виртуализации графического процессора для множественных графических клиентов. В различных вариантах осуществления изобретения архитектуры виртуализации по данному изобретению подчиняются одному или более требованиям из числа следующих требований: четко определенное состояние исполнения (контекст), поддержка переключения контекста с малой задержкой, защищенные виртуальные адресные пространства, управление памятью с подкачкой по обращению, и наличие режимов привилегированного исполнения. Эти требования используются для того, чтобы обеспечить эффективное, защищенное и робастное совместное использование аппаратных ресурсов графического процессора множественными клиентами (приложениями).

Для упрощения обсуждения виртуализации полезно обобщить степень поддержки виртуализации, которая предполагается в соответствии с изобретением. Простейшая аналогия заключается в том, что здесь должен быть достигнут приблизительно тот же самый уровень совместного использования графического процессора, как тот, что достигнут в современном центральном процессоре для совместного использования центрального процессора множественными клиентами (приложениями). Это является естественным следствием эволюции графического процессора, ведь, он представляет собой мощный вычислительный ресурс, который должен эффективно совместно использоваться множественными независимыми клиентами. Одно отличие от центрального процессора заключается в том, что графический процессор управляется рабочей программой, исполняемой на центральном процессоре, а не на графическом процессоре. В некоторых моделях драйверов, наоборот, графическая подсистема ядра действует как мини-диспетчер, предоставляя службы диспетчеризации и управления памятью и обеспечивая контроль над глобальными ресурсами, такими как конфигурация видеоустройства отображения, управление электропитанием и т.д. Этот мини-диспетчер функционирует независимо от диспетчера ядра, принимая свои собственные решения по диспетчеризации и управлению памятью в соответствии с алгоритмами и политикой, настроенной на требования клиентов графического процессора.

Коль скоро удовлетворены требования виртуализации, становится возможным предоставить драйверу клиента прямой доступ к аппаратным средствам графического процессора. Эти правила аналогичны тем, что предназначены для виртуализации других функций ядра, то есть являющийся гостевым драйвер не должен быть способен обнаруживать, что он исполняется на мониторе виртуальных машин и не должен быть способен обходить какие бы то ни было механизмы защиты, установленные монитором виртуальных машин. Для целей данного обсуждения принимается, что монитор виртуальных машин графического процессора вложен в ядро монитора виртуальных машин и для всех практических целей функционирует как драйвер в ядре монитора виртуальных машин. Графический монитор виртуальных машин может включать в себя часть, не зависящую от устройства, и часть, зависящую от устройства (последнюю предоставляет изготовитель графических аппаратных средств).

Фиг.3В иллюстрирует другую приводимую в качестве примера неограничивающую архитектуру, предусматриваемую для виртуализации графического конвейера в соответствии с изобретением. Монитор (304) виртуальных машин включает в себя

компонент «графический монитор виртуальных машин», который оперирует графическими функциональными возможностями и осуществляет управление графикой в соответствии с изобретением. Любая операционная система из числа главной операционной системы с главным графическим драйвером (HGF), гостевой операционной системы A с графическим драйвером (AGD), гостевой операционной системы B с графическим драйвером (BGD) может взаимодействовать с графическим монитором виртуальных машин через монитор (304) виртуальных машин.

Поскольку одним из мотивационных соображений в пользу использования монитора виртуальных машин является эффективность, то имеет смысл напрямую реализовать тип II графического монитора виртуальных машин с аппаратной поддержкой. Предоставление усовершенствованной модели драйвера отображения (DDM) добавляет аппаратную поддержку для виртуальных адресных пространств с контролируемым доступом, выгрузки и привилегированных операций. Один «лакмусовый» тест на эффективную поддержку монитора виртуальных машин заключается в исполнении имеющейся усовершенствованной модели драйвера отображения поверх графического монитора виртуальных машин без внесения каких-либо изменений в этот драйвер. Это означает, что операции ядра графической подсистемы, которые осуществляют инициализацию карты, устанавливают таблицы страниц, программируют контроллеры устройства отображения, устанавливают окно адресов шины PCI (шины межсоединения периферийных компонентов), манипулируют входными данными, интерпретируемыми как физические адреса и т.д., должны сгенерировать прерывания для монитора виртуальных машин для эффективной интерпретации. Прерывания должны перехватываться монитором виртуальных машин и направляться в соответствующий (гостевой) драйвер. Высокоскоростные непривилегированные операции, такие как операции визуализации, должны исполняться графическими аппаратными средствами без вмешательства со стороны монитора виртуальных машин.

Аппаратные средства с усовершенствованной моделью драйвера поддерживают множественные контексты исполнения, каждый с отдельным виртуальным адресным пространством и входной очередью (кольцевым буфером) для управления буферами входящих прямых адресов памяти потоков команд. Ресурсы локальной и системной физической памяти отображаются на виртуальное адресное пространство контекста посредством инициализации элементов таблицы страниц с тем, чтобы они указывали на физические адреса. О завершении операций потока команд подается сигнал с использованием записи в память (заграждающей метки) и прерываний.

Один из способов для управления защитой памяти представляет собой разделение физического адресного пространства локальной памяти графических данных и выделения фиксированного непрерывного раздела каждой виртуальной машине (гостевой операционной системе). Память за пределами выделенной невидима для виртуальной машины. Этот механизм мог бы быть реализован с использованием простого уровня косвенности посредством индексного и лимитирующего регистров, программируемых графическим монитором виртуальных машин. Это может потребовать осуществления контроля содержимого регистров конфигурации шины PCI (шины межсоединения периферийных компонентов), видимого для драйвера (считываний, уловленных монитором виртуальных машин). В этой схеме управления памятью каждый «гость» видит фиксированную выделенную ему память в течение времени жизни этого «гостя». Это устраняет необходимость для монитора виртуальных машин улавливать операции с таблицей страниц (которые могут быть

частыми), поскольку окно физического адресного пространства, видимое для аппаратных средств, тщательно контролируется монитором виртуальных машин. Это приемлемый первый подход к реализации поддержки виртуализированной памяти, поскольку он освобождает монитор виртуальных машин от детального управления памятью, хотя, как полагают, обеспечивает менее эффективное использование ресурсов памяти.

Можно представить, что потребуется более длительное время для того, чтобы получить монитор виртуальных машин, участвующий в операциях детального управления памятью. Эта часть становится более сложной по той причине, что операции, используемые драйвером для программирования элементов таблицы страниц, должны быть уловлены монитором виртуальных машин. В зависимости от целей, связанных с производительностью при работе по постраничной организации памяти, может потребоваться более тщательное проектирование конструкции аппаратных средств с целью повышения эффективности обнаружения и эмуляции этих операций. Например, может оказаться полезным спроектировать операцию (команды) отображения страниц в памяти таким образом, чтобы множественные отображения могли быть специфицированы как группа, и они могли быть легко распознаны и декодированы в качестве группы в графическом мониторе виртуальных машин. Это минимизирует количество прерываний посредством монитора виртуальных машин. Также монитор виртуальных машин может использовать («доверенные») службы в гостевом разделе для поддержки операций постраничной организации памяти для поддерживающего запоминающего устройства (другого уровня поддерживающего запоминающего устройства помимо того, что реализует гостевой драйвер модели драйвера отображения). Хотя эта схема является более сложной, она позволяет каждому «гостю» использовать все аппаратные ресурсы памяти.

Подход, описанный в связи с этими вариантами осуществления изобретения, предполагает использование программно-виртуализированных таблиц страниц. Подходом альтернативным по отношению к программно-виртуализированным таблицам страниц служит добавление еще одного набора привилегированных таблиц страниц, то есть еще одного уровня косвенности, в графические аппаратные средства для непосредственного использования монитором виртуальных машин. Это представляет собой обобщение подхода на основе индексного и лимитирующего регистров, описанного для фиксированных разделов, которое (обобщение) позволяет монитору виртуальных машин контролировать то, как графические аппаратные средства видят физический ресурс (физические ресурсы) на страничном уровне глубины детализации.

Обсуждение вопроса управления памятью до настоящего момента фокусировалось на управлении распределением локальной памяти графических данных графическим контекстам. Аналогичные требования имеются и в отношении отображения страниц системной памяти на аппаратные средства, поскольку здесь также используются адреса физической памяти. При этом важно то, каким образом процессорный монитор виртуальных машин управляет физической памятью. Обычно устройствами, которые используют физические адреса, управляют с особой тщательностью с тем, чтобы сохранять полный контроль за их отображениями. Простая схема с фиксированными разделами, предложенная для видеопамяти, могла бы быть распространена для работы с системной памятью, как и ранее, посредством добавления элементов управления графическими аппаратными средствами, ограничивающих окно доступной системной памяти. Однако это неприменимо в том случае, если

процессорный монитор виртуальных машин при обслуживании операций выделения памяти для «гостей» выделяет физическую память, не являющуюся непрерывной. Это означает, что для ресурсов системной памяти должна быть реализована более сложная схема подробного управления памятью (что предполагает в то же самое время принятие решения по локальной памяти графических данных). Вероятно, имеются и некоторые дополнительные допущения, принимаемые процессорным монитором виртуальных машин и касающиеся управления физической памятью и потенциальными взаимодействиями с графическим монитором виртуальных машин, которые должны быть тщательным образом исследованы.

В целях обеспечения более эффективной обработки прерываний, типы прерываний могли бы быть разделены на те, что должны обрабатываться монитором виртуальных машин, и те, что не должны им обрабатываться. Последняя категория могла бы включать в себя окончания команд и т.д., и они могут обрабатываться непосредственно гостевым драйвером.

Другой большой категорией, относящейся к управлению, является диспетчеризация обрабатывающих ресурсов. Усовершенствованная модель драйвера поддерживает выгрузку контекстов, так что в самой простой модели управления монитор виртуальных машин выделяет разделам интервалы времени, выгружая исполняемый в текущий момент графический контекст из исполняемого в текущий момент раздела. Современная усовершенствованная модель драйвера использует для задания нового списка исполняемых контекстов список исполнения с двойной буферизацией. При этом делается допущение о том, что невыполненных списков исполнения будет не более чем один. Виртуализированная модель диспетчеризации потребует некоторых изменений для того, чтобы упростить монитору виртуальных машин переключение разделов на границах списка исполнения и, кроме того, обеспечить, чтобы при переходе к новому разделу состояние контекста сохранялось в старом разделе и восстанавливалось из нового раздела (например, чтобы ресурсы памяти соответствующего раздела распределялись в надлежащие моменты времени во время сохранения и восстановления контекста). И в этом случае будут иметь место допущения и взаимодействия, связанные с диспетчером процессорного монитора виртуальных машин, которые должны быть рассмотрены.

Может быть реализована и более сложная схема диспетчеризации, в которой графический монитор виртуальных машин в большей степени осведомлен о рабочей нагрузке и приоритетах различных контекстов. Эта информация может быть сделана доступной для процессорного монитора виртуальных машин для принятия более осмысленных решений по диспетчеризации. Имеется несколько интересных вопросов, связанных с проектированием, касающихся того, может ли графический монитор виртуальных машин исполнять контекст из раздела, отличного от того, который в данный момент исполняет процессорный монитор виртуальных машин. Возможно, и может быть необходимо в целях максимизации использования аппаратных средств допускать некоторое перекрытие в обработке во время переходов между разделами. Однако это требует более тщательного рассмотрения того, каким образом управляются и распределяются ресурсы во время переходов.

Что касается виртуализации представления и взаимодействия, то в первом варианте осуществления изобретения, коль скоро произведена виртуализация графических аппаратных средств, каждый «гость» создает свой собственный рабочий стол и визуализирует окна из гостевых приложений на этом рабочем столе. Этот гостевой рабочий стол хранится в локальной памяти графических данных и в обычной рабочей

среде выводится в виде развертки на устройство отображения (электронно-лучевую трубку или плоскую панель). В виртуализированной рабочей среде содержимое множественных гостевых рабочих столов собирается вместе и отображается на единственном отображении. Имеется несколько подходов к формированию этого отображения, включающих в себя предоставление множественных виртуальных рабочих столов (единственный рабочий стол единовременно), предоставление рабочего стола в его собственном окне или единственный унифицированный рабочий стол.

В первом варианте осуществления изобретения единственный гостевой рабочий стол выбирается в качестве «текущего» рабочего стола и отображается. Используя некоторый элемент управления пользовательского интерфейса, пользователь может осуществлять выбор между различными «гостями», но единовременно отображается только один рабочий стол. Этот подход является самым простым, поскольку изображения индивидуальных гостевых рабочих столов могут быть отображены с легкостью. Фактически, элемент управления пользовательского интерфейса определяет то, какому «гостю» разрешено произвести установку сканирующих регистров контроллера устройства отображения.

Фиг.4А иллюстрирует вариант осуществления изобретения, в котором отображают «единственный рабочий стол единовременно». Можно видеть, что каждая виртуальная машина получает свою собственную область рабочего стола в локальной памяти графических данных, причем каждая область независима от других, благодаря чему для отображения выводится в виде развертки только «активный» рабочий стол, выбранный пользователем. Таким образом, «гости» способны осуществлять растеризацию своих соответствующих элементов работы и оставить их где-то в памяти, а механизм представления способен управлять тем, какая область памяти (содержащая биты, прошедшие растеризацию) отображается, что могло бы подразумевать дополнительную обработку растеризации с целью изменения размеров или перемещения содержимого.

По определению, этот первый вариант осуществления изобретения не позволяет множественным рабочим столам быть видимыми одновременно (хотя в конфигурации «мультимон» (множественных мониторов) различные контроллеры устройств отображения могли бы отображать рабочие столы от различных «гостей», а не все от одного и того же «гостя»). Альтернативным подходом является второй вариант осуществления изобретения, в котором индивидуальные рабочие столы отображаются в отдельных окнах-контейнерах на главном рабочем столе. Это использует службу, подлежащую исполнению в главной операционной системе и имеющую доступ к содержимому рабочего стола каждого «гостя». Эта служба выполняет операцию составления, формирующую главный рабочий стол, использующий гостевые рабочие столы в качестве входных данных. Для каждого гостевого рабочего стола используется окно-заместитель, управляющее размером и расположением этого рабочего стола. Содержимое гостевого рабочего стола может быть извлечено непосредственно из содержимого видеопамяти посредством перехвата изменений в установочные параметры контроллера устройства отображения для этого рабочего стола (используя их для определения адреса памяти, габаритного поля и других параметров). Составной главный рабочий стол формируется во многом таким же способом, как формируется типичный рабочий стол с использованием механизма составления рабочего стола (DCE) - также используя аппаратное графическое ускорение. Этот второй вариант осуществления может быть объединен с первым

вариантом для того, чтобы поддерживать режим с единственным прямым видом гостевого рабочего стола или режим с составным видом множественных рабочих столов в отдельных окнах.

Фиг.4B иллюстрирует альтернативный вариант осуществления «рабочий стол в окне», посредством которого все рабочие столы могут быть одновременно отображены механизмом представления с учетом загораживания. Например, главный рабочий стол мог бы по умолчанию быть окном рабочего стола, расположенным на заднем плане на отображении, а гостевые рабочие столы, каждый из которых отображается внутри своего собственного окна-контейнера, получают отдельное окно на отображении поверх окна рабочего стола, расположенного на заднем плане.

Этот второй вариант осуществления изобретения начинает вводить некоторые усложнения, связанные с координацией ввода при синхронизации положения и размера окон гостевого рабочего стола. По мере того, как в главном окне-контейнере производится манипуляция окнами для гостевого рабочего стола, эти взаимодействия улавливаются и направляются «гостю» для обработки. «Гость», например, его администратор окна, обновляет гостевой рабочий стол и в результате этого на главном рабочем столе отражаются новые места расположения окон. К этому типу координации ввода уже обращались в других продуктах виртуализации, которые реализуют принцип «Рабочий стол в окне», и он (этот тип координации) не должен быть затронут при виртуализации графических аппаратных средств.

Второй вариант осуществления изобретения представляет собой значительное усовершенствование, но по-прежнему было бы желательным смешивать окна от множественных гостевых рабочих столов, то есть скрывать тот факт, что имеются гостевые рабочие столы. Эта проблема сводится к нахождению содержимого индивидуального окна в каждом гостевом рабочем столе, но она более сложная, чем нахождение содержимого рабочего стола, поскольку информацию об индивидуальном окне не просто перехватить в потоке данных, посылаемых графическим аппаратным средствам. Однако в случае, если целенаправленно нарушить правило, касающееся внесения изменений в гостевую операционную систему, то имеется возможность добавить некоторые зацепки, способствующие идентификации пикселей (ячеек памяти) для индивидуальных окон. Имеется пара различных подходов, которые могут быть при этом использованы.

Один подход заключается в том, чтобы держать окна на гостевом рабочем столе организованными по принципу мозаики (без перекрытия), так что все их содержимое всегда доступно, поддерживая при этом информацию о размере и месте расположения каждого окна на гостевом рабочем столе, предназначенную для использования администратором/механизмом составления главного окна. Этот подход включает в себя ведение двух отдельных наборов информации об окнах: виртуальной информации (на главном рабочем столе) и информации о физическом месте расположения на гостевом рабочем столе. Он также усложняет координацию ввода, поскольку операции, которые манипулируют виртуальным окном, могут затрагивать или могут не затрагивать физическое окно (например, если виртуальное окно загорожено, то может оказаться полезным исправить это место на гостевом рабочем столе). Взаимодействия внутри виртуального окна переводятся во взаимодействия, соответствующие физическому окну (это могло бы быть также просто как коррекция входных координат).

Альтернативный подход заключается в том, чтобы переадресовать окна гостевого рабочего стола в память, которая более легко управляется. Это аналогично

переадресации, выполняемой для интерфейса графического устройства (GDI), в котором для каждого окна поддерживается отдельный «поддерживающий буфер». Содержимое этих поддерживающих буферов сделано доступным для администратора/механизма составления главного окна. В таком случае при взаимодействии пользователя с содержимым виртуального окна производится аналогичное отображение между виртуальными входными координатами и физическими входными координатами.

При некоторой доле предвидения и планирования имеется возможность формировать будущие операционные системы таким образом, чтобы они лучше способствовали таким перехватам в тех случаях, когда эти операционные системы исполняются как «гости». Расположение оболочки на каждом гостевом рабочем столе (если и когда они прекращают свое существование с точки зрения пользователя), обработка расширения оболочки, опции стартового меню, сокращения приложений, буфер вырезанного изображения и т.д. соединены в унифицированный вид рабочего стола. Это представляет собой вполне решаемые проблемы, но требует знания бульших подробностей о соответствующих вариантах реализации в каждом из «гостей» и, возможно, (статического) извлечения и соединения этой информации вместе с целью формирования унифицированного вида. Аналогичные вопросы возникают и для случая поддержки унифицированного доступа.

Аналогичные проблемы могут существовать и при обеспечении единообразного представления других частей многогостевой системы, например запоминающего устройства, установленных приложений, подключения к сети и т.д. Промежуточная стратегия экспонирования множественных «гостей» в виде рабочих столов, имеющих оконную организацию, с независимыми характерными отпечатками позволила бы сфокусироваться на задаче построения робастной инфраструктуры виртуализации. Параллельно имеется возможность обратиться к некоторым другим признакам, таким как «доверенная» обработка и защита содержимого в целях управления цифровыми правами.

В качестве основы для защищенной («доверенной») обработки графических данных вышеописанные виртуализированные графические системы сочетаются с другими моделями для защищенной обработки данных с использованием виртуальных машин. Характеристики виртуализированной графической среды с легкостью позволяют, чтобы смешивание «доверенной» и «не наделенной доверием» обработки данных производилось в одной и той же системе надежным образом при полной производительности. Такого рода модель требует, чтобы графический монитор виртуальных машин осуществлял робастную аутентификацию и начальную загрузку графических устройств. Это, конечно, опирается на другие «доверенные» компоненты в том, чтобы осуществлять защищенную начальную загрузку графического монитора виртуальных машин. Поскольку среда виртуализации осуществляет робастную защиту множественных «гостей» и главного раздела друг от друга, то имеется возможность создать «доверенный» раздел, который может производить «доверенную» обработку графических данных. Если администратор окна и механизм составления окна исполняются как службы в «доверенном» разделе, то они могут смешивать «доверенные» и «не наделенные доверием» данные приложений надежным образом, используя для этого ранее описанные способы представления.

Фиг.4С иллюстрирует приводимую в качестве примера архитектуру для смешивания «доверенных» и «не наделенных доверием» данных приложений в соответствии с изобретением. Как было описано, аналогично главной операционной

системе H с графическим драйвером HGD предусматривается «доверенная» операционная система T как «доверенный» раздел с графическим драйвером TGD. «Доверенная» операционная система T включает в себя компонент «администратор окна», или «механизм составления окна», который перед визуализацией посредством графического конвейера берет входные данные из кода графического монитора виртуальных машин в отношении любых ограничений на хранение и визуализацию «доверенных» данных в отличие от «не наделенных доверием» данных.

«Доверенная» служба нуждается в доступе к отображаемым данным отдельного гостевого рабочего стола, но эта информация не интерпретируется службой составления в качестве управляющих данных. Она используется как входные данные визуализации, поэтому риск атаки со стороны содержимого совместно используемых данных отсутствует. Если виртуализация графических аппаратных средств осуществлена правильно, то риск того, что код в одном разделе атакует другой раздел, отсутствует. Гостевые драйверы не могут ни захватить управление аппаратными средствами, ни вмешаться в работу других «гостей». Код приложения, исполняемый внутри графических аппаратных средств, не может вмешаться в работу других приложений в том же самом гостевом разделе, не говоря уж о других гостевых разделах.

Поддержка виртуализации по данному изобретению не устраняет необходимость в аутентификации аппаратных средств, робастном контроле отображаемых выходных данных или другие требования, предназначенные обеспечивать, чтобы отдельно взятый «доверенный» раздел мог надежно передавать отображаемые данные на устройство отображения, но она позволяет множеству разделов робастно совместно использовать графические аппаратные средства при малом воздействии на производительность.

Что касается основы для защиты содержимого во время обработки и отображения данных в соответствии с изобретением, то отметим, что применение цифровых прав для видеоданных с полным движением в настоящее время обеспечивается с использованием технологии «Защищенный тракт видеоданных», предназначенной для защиты содержимого. Защищенный тракт видеоданных объединяет в себе несколько из числа нижеследующих способов, обеспечивающих защищенную транспортировку и обработку видеоданных от исходного устройства до устройства отображения:

(1) среда ядра укрепляется таким образом, что запрещаются отладчики ядра и т.п., и только «доверенным» компонентам разрешается функционировать в режиме ядра («кольцо 0») при проигрывании защищенного содержимого;

(2) процессы пользовательского режима, обрабатывающие защищенное содержимое, дополнительно укрепляются против внешних атак (защищенная среда). Видеоданные, представленные в системе или памяти графических данных (или отображаемые из системы или из этой памяти), не доступны из других процессов;

(3) видеоданные, посылаемые по доступным для пользователей шинам, шифруются;

(4) графический драйвер должен быть сертифицирован в отношении проигрывания защищенного содержимого и, в свою очередь, производит оценку того, что графические аппаратные средства представляют собой допустимое синхронизирующее устройство для защищенного содержимого; и

(5) защита данных, выводимых на устройство отображения, поддерживается графическими аппаратными средствами (Macrovision, CGSM-A, HDCP) и может робастно управляться подсистемой воспроизведения мультимедийных данных.

Механизмы (1) и (2) могут быть заменены более общей базой защищенных

вычислений в соответствии с изобретением. Эта база защищенных вычислений должна, помимо того, предотвращать обманные действия посредством технологии виртуализации, делая невозможным нарушение процесса защищенной начальной загрузки. Механизмы (3) и (5) по-прежнему необходимы для того, чтобы продолжать обеспечивать защищенную обработку и транспортировку видеоданных по мере того, как они перемещаются из системной памяти через графические аппаратные средства в устройство отображения. Способы виртуализации графических аппаратных средств согласно изобретению не замещают никакие из специальных способов защиты содержимого, связанные с трактом графических данных; напротив, они служат основанием для построения «доверенных» графических служб. Эти «доверенные» службы могут быть расширены таким образом, чтобы включать в себя компоненты защищенной обработки мультимедийных данных (шлюз взаимодействия мультимедиа или для краткости MIG) и их исполнение в «безопасном» «доверенном» разделе, а не во враждебном гостевом разделе. Существующая инфраструктура защищенных видеоданных уже отделила защищенную обработку данных (передача прав, политика, декодирование, дешифрование) от незащищенной обработки данных (графический пользовательский интерфейс приложений воспроизведения и т.д.). Это разделение остается нетронутым и в виртуализированной среде, при этом защищенные компоненты исполняются в «доверенном» разделе (в другой (более хорошо) защищенной среде), а незащищенные компоненты остаются в «не наделенном доверием» гостевом разделе.

Таким образом, фиг.4D иллюстрирует расширение понятия «доверенного» раздела T, разработанного в связи с фиг.4C, за счет включения в него шлюза взаимодействия мультимедиа, в качестве отдельного уровня защиты содержимого, воспроизводимого приложением-проигрывателем мультимедиа, исполняемым в гостевой операционной системе A.

Аналогично тому, как продукт Virtual PC может исполнять большое количество операционных систем, виртуализированная графическая среда по данному изобретению способна поддерживать драйверы/модели драйверов для заранее заданного количества операционных систем или их версий.

Успех каждой операционной системы измеряется широтой приложений, которые она может исполнять и эффективностью этих приложений. Тщательно избегая стремления изменить гостевую операционную систему, можно получить возможность исполнять под управлением виртуализированной операционной системы любое приложение, которое исполняется под управлением этой операционной системы, исполняемой в ее собственном (не виртуальном) качестве. Места возможных различий заключаются в том, как разделяются ресурсы. Например, если для некоторого раздела сделано доступным только некоторое подмножество ресурсов (часть локальной памяти графических данных), то приложение может не функционировать надлежащим образом. В будущих операционных системах это, возможно, будет менее актуально, поскольку виртуализация «первого порядка» - виртуализация ресурсов для совместного использования множественными приложениями - станет нормой.

Стимул для движения по направлению к полностью виртуализированным графическим аппаратным средствам заключается для компаний - разработчиков программных средств в том, чтобы освободиться от проблем, связанных с «унаследованным» кодом, при этом по-прежнему обеспечивая практический способ для непрерывной совместимости приложений. В качестве побочного эффекта, изобретение также предоставляет прекрасную среду для «доверенной» обработки

графических данных и защиты содержимого.

Этот второй показатель успеха означает, что развивающиеся приложения должны прекратить использовать «унаследованный» код и должны развиваться с тем же успехом. Для больших баз кода приложений это может оказаться чрезвычайно трудным. Например, может оказаться невозможным оставить в прошлом программные средства интерфейса графического устройства как «унаследованный» код, в случае, если, например, «унаследованная» база кода приложений, которая эволюционирует вместе с версиями операционной системы, такая как база кода Office компании Microsoft, неразрывным образом зависит от него. В таких случаях, здесь, возможно, могли бы иметь место гибридные модели, которые можно изучить. Например, гостевые операционные системы могли бы рассматриваться как общий поставщик служб. Вновь используем в качестве примера интерфейс графического устройства, тогда «унаследованная» операционная система могла бы рассматриваться как поставляющая службу визуализации интерфейса графического устройства. Приложение использует механизм удаленного типа, который использует эту службу. Цель в такого рода ситуации необязательно состоит в том, чтобы исполнять приложение без каких-либо изменений кода, но скорее в том, чтобы внести минимум изменений, которые позволили бы приложению продолжать использовать «унаследованный» код интерфейса графического устройства в виде службы, действующей между разделами, при этом выигрывается время на полный переход с этого интерфейса прикладного программирования.

Фиг.4E иллюстрирует приводимое в качестве примера взаимодействие, при котором «унаследованное» приложение гостевой операционной системы A, исполняемое на главной операционной системе H, взаимодействует с компонентом ««унаследованные» службы» гостевой операционной системы с тем, чтобы все таки эмулировать «унаследованный» режим с точки зрения главной операционной системы H.

Фиг.5 иллюстрирует приводимую в качестве примера последовательность операций, в которой приложение, исполняемое в гостевой операционной системе, в соответствии с любой из виртуализированных архитектур, показанных или описанных в данной заявке, выдает команду, затрагивающую ресурсы графической подсистемы. На этапе 500 приложение, исполняемое в гостевой операционной системе, выводит элемент графической работы (например, нарисовать синий круг на экране в координате «x,y»). На этапе 510 гостевая операционная система обрабатывает этот запрос, посылая его в очередь графической работы и связанному с ней диспетчеру. На этапе 520, когда элемент графической работы готов (прошел диспетчеризацию), указатель на данные этого элемента графической работы передается графическими драйверами аппаратным средствам. На этапе 530, если элемент графической работы является привилегированной операцией, то графический монитор виртуальных машин принимает и обрабатывает этот запрос на работу, основываясь на политиках (гостевая операционная система по отношению к главной операционной системе, политика распределения памяти, тип команды и т.д.). Если элемент графической работы является непривилегированной операцией, то вместо этого на этапе 540 графические аппаратные средства приступают к непосредственной обработке этого запроса.

Таким образом, согласно изобретению осуществляется виртуализация аппаратных средств процессора по отношению к графическому ускорителю, имеющая своей целью обеспечение преимуществ высокопроизводительной визуализации для множественных

гостевых операционных систем, исполняемых в виртуализированной среде. Такого рода среда может быть использована для эффективного решения нескольких проблем: (A) устранение «унаследованного» кода в разработке новых операционных систем, поддерживая при этом совместимость приложений посредством исполнения «унаследованного» приложения в «унаследованной» операционной системе в среде виртуальной машины; (B) предоставление инфраструктуры, которая безопасно смешивает «доверенную» и «не наделенную доверием» обработку в одной и той же виртуализированной среде; и (C) усиление «доверенной» среды таким образом, чтобы она обеспечивала защиту содержимого для управления цифровыми правами.

Приводимые в качестве примера сетевые и распределенные среды

Специалист обычного уровня квалификации в данной области техники может понять, что изобретение может быть реализовано в связи с любым компьютером или другим клиентским или серверным устройством, которые могут быть размещены как часть компьютерной сети или в распределенной вычислительной среде. В этом отношении отметим, что настоящее изобретение относится к любой компьютерной системе или среде, имеющей любое количество памяти или запоминающих устройств, и любое количество приложений и процессов, происходящих в любом количестве запоминающих устройств или томов, которые могут быть использованы в связи с виртуализацией графического конвейера в соответствии с настоящим изобретением. Настоящее изобретение может применяться к среде с компьютерами-серверами и компьютерами-клиентами, размещенными в сетевой среде или в распределенной вычислительной среде, имеющими удаленное или локальное запоминающее устройство. Настоящее изобретение также может быть применено к автономным вычислительным устройствам, имеющим функциональные возможности языка программирования, способности к интерпретации и исполнению с целью генерирования, приема и передачи информации в связи с удаленными или локальными службами. В игровой среде графический конвейер особенно уместен по отношению к тем вычислительным устройствам, что функционируют в сети или в распределенной вычислительной среде, и по этой причине виртуализация графического конвейера в соответствии с настоящим изобретением может быть применена в этих средах с большой эффективностью.

Распределенные вычисления обеспечивают совместное использование компьютерных ресурсов и служб посредством обмена между вычислительными устройствами и системами. Эти ресурсы и службы включают в себя обмен информацией, кэш-память и запоминающее устройство на дисках для файлов. Распределенные вычисления используют преимущество сетевых соединений, позволяя клиентам усилить свою коллективную мощность для блага всего предприятия. В этом отношении отметим, что множество устройств может иметь приложения, объекты или ресурсы, которые могут подразумевать графические процессы по данному изобретению.

Фиг.6A приводит схематическое представление приводимой в качестве примера сетевой или распределенной вычислительной среды. Распределенная вычислительная среда содержит вычислительные объекты (10a), (10b) и т.д. и вычислительные объекты или устройства (610a), (610b), (610c) и т.д. Эти объекты могут содержать программы, способы, хранилища данных, программируемую логику и т.д. Объекты могут содержать части тех же самых или других устройств, таких как персональные цифровые секретари (PDA), аудио/видео устройства, MP3-плееры, персональные компьютеры и т.д. Каждый объект может поддерживать связь с другим объектом

посредством сети (14) связи. Эта сеть может сама содержать другие вычислительные объекты и вычислительные устройства, которые предоставляют службы системе, показанной на фиг.6А, и может сама представлять множественные соединенные между собой сети. В соответствии с одним аспектом изобретения, каждый объект 10a, 10b, и т.д. или (610a), (610b), (610c) и т.д. может содержать приложение, которое могло бы использовать интерфейс прикладного программирования, или другой объект, программные средства, микропрограммные средства и/или аппаратные средства для того, чтобы запрашивать использование процессов виртуализации графических средств по данному изобретению.

Можно также себе представить, что хостинг объекта, такого как 610c, может быть осуществлен на другом вычислительном устройстве 10a, 10b, и т.д. или (610a), (610b) и т.д. Таким образом, хотя изображенная физическая среда может показывать соединенные устройства как компьютеры, такая иллюстрация служит просто в качестве примера, и эта физическая среда может быть в качестве альтернативы изображена или описана как содержащая разнообразные цифровые устройства, такие как персональные цифровые секретари, телевизионные системы, MP3-плееры и т.д., программные объекты, такие как интерфейсы, COM-объекты (объекты модели компонентных объектов) и тому подобное.

Существует множество систем, компонентов и сетевых конфигураций, которые поддерживают распределенные вычислительные среды. Например, вычислительные системы могут быть соединены вместе посредством систем проводной и беспроводной связи. В настоящее время многие из сетей подсоединены к сети «Интернет», которая предоставляет инфраструктуру для широко распределенных вычислений и охватывает много различных сетей. Любая из этих инфраструктур может быть использована для приводимых в качестве примера вариантов связи, приспособленных для процессов виртуализации графических аппаратных средств по данному изобретению.

В бытовых сетевых средах имеется, по меньшей мере, четыре диспаратные сетевые транспортные среды, каждая из которых может поддерживать уникальный протокол, такие как линия электропитания, данные (как передаваемые беспроводным способом, так и по проводам), голос (например, телефон) и развлекательное мультимедиа. Большинство бытовых устройств управления, таких как выключатели света и бытовое радиоэлектронное и электрическое оборудования могут использовать линии электропитания для связи. Службы обработки данных могут попадать в дом по широкополосным каналам связи, а внутри дома доступ к ним осуществляется с использованием либо беспроводных (например, с использованием стандартов HomeRF (Бытовой радиочастотной связи) или 802.11B) или проводных (например, с использованием стандартов Home PNA, Cat 5, сети «Ethernet», даже линии электропитания) соединений. Трафик голосовых данных может попадать в дом либо по проводам (например, Cat 3), либо беспроводным способом (например, через сотовые телефонные аппараты) и может распространяться внутри дома с использованием проводной связи Cat 3. Развлекательное мультимедиа или другие графические данные могут попадать в дом либо через спутник, либо по кабелю и обычно распространяется в доме с использованием коаксиального кабеля. Стандарт IEEE 1394 (стандарт Института инженеров по электротехнике и радиоэлектронике) и DVI (Цифровое видео взаимодействие) также предусматривают цифровые межсоединения для кластеров мультимедийных устройств. Все эти сетевые среды и другие сетевые среды, которые могут появиться в случае, когда стандарты протоколов могут быть соединены между собой с целью образования сети, такой как

интрасеть, представляют собой среды, которые могут быть соединены с внешним миром посредством сети «Интернет». Короче говоря, существует многообразие диспаратных источников для хранения и передачи данных, и, следовательно, развиваясь, вычислительные устройства потребуют способов совместного использования данных, таких как данные, доступ к которым или использование которых характерно для программных объектов, использующих виртуализированные графические службы в соответствии с настоящим изобретением.

Термин «Интернет» обычно относится к совокупности сетей и шлюзов, которые используют набор TCP/IP протоколов, которые хорошо известны в области создания компьютерных сетей. TCP/IP представляет собой сокращение от «Протокол управления передачей/Межсетевой протокол». Сеть «Интернет» может быть описана как система географически распределенных удаленных компьютерных сетей, соединенных между собой посредством компьютеров, исполняющих сетевые протоколы, которые позволяют пользователям взаимодействовать и делиться между собой информацией по сети (сетям). По причине такого широко распространенного совместного использования информации удаленные сети, такие как сеть «Интернет», к настоящему моменту, в общем, эволюционировали в открытую систему, для которой разработчики могут разрабатывать программные приложения, предназначенные для выполнения специальных операций или служб, практически без ограничения.

Таким образом, сетевая инфраструктура делает возможным множество сетевых топологий, таких как архитектуры клиент/сервер, одноранговые или гибридные архитектуры. «Клиент» представляет собой класс или группу, которая использует службы другого класса или группы, с которыми он не связан. Таким образом, в ходе вычислений клиент представляет собой процесс, то есть, грубо говоря, набор команд или задач, который запрашивает службу, предоставляемую другой программой. Являющийся клиентом процесс использует запрошенную службу, не будучи при этом обязанным «знать» какие-либо подробности об этой другой программе или самой службе. В архитектуре клиент/сервер, особенно в сетевой системе, клиент обычно представляет собой компьютер, который осуществляет доступ к совместно используемым сетевым ресурсам, предоставляемым другим компьютером, например сервером. В примере, приведенном на фиг.6А, компьютеры (610a), (610b) и т.д. могут рассматриваться как клиенты, а компьютеры (10a), (10b) и т.д. могут рассматриваться как сервер, где сервер (10a), (10b) и т.д. поддерживает данные, которые затем реплицируются в компьютерах-клиентах (610a), (610b) и т.д., хотя в зависимости от обстоятельств любой компьютер может рассматриваться как клиент, сервер или и то и другое вместе взятое. Любое из этих вычислительных устройств может обрабатывать данные или запрашивать службы или задачи, которые могут подразумевать реализацию виртуализированных архитектур графических средств по данному изобретению.

Сервер обычно представляет собой удаленную вычислительную систему, доступ к которой осуществляется по удаленной или локальной сети, такой как сеть «Интернет». Являющийся клиентом процесс может быть активным в первой компьютерной системе, а сервер может быть активным в другой компьютерной системе, при этом они поддерживают связь друг с другом посредством среды передачи данных, обеспечивая тем самым распределенные функциональные возможности и позволяя множеству клиентов воспользоваться способностями сервера по сбору информации. Любые программные объекты, применяемые с целью использовать виртуализированную архитектуру (виртуализированные архитектуры) по данному изобретению, могут

быть распределены по множественным вычислительным устройствам или объектам.

Клиент (клиенты) и сервер (серверы) поддерживают связь друг с другом с использованием функциональных возможностей, предоставляемых уровнем (уровнями) протоколов. Например, Протокол передачи гипертекста (HTTP) представляет собой широко распространенный протокол, который используется в связи со «Всемирной паутиной» (WWW) или «Web». Обычно для идентификации компьютеров сервера или клиента друг для друга может быть использован адрес в компьютерной сети, такой как адрес по Межсетевому протоколу (IP-адрес) или другая ссылка, такая как Универсальный адрес ресурса (URL). Связь может быть обеспечена посредством среды передачи данных, например клиент (клиенты) и сервер (серверы) могут быть соединены друг с другом посредством TCP/IP-соединения (соединений), обеспечивающего (обеспечивающих) высокую пропускную способность.

Таким образом, фиг.6A иллюстрирует приводимую в качестве примера сетевую или распределенную среду, в которой сервер поддерживает связь с компьютерами-клиентами по сети/каналу связи и в которой может быть использовано настоящее изобретение. Если описать это более подробно, то ряд серверов (10a), (10b), и т.д. соединены посредством сети/канала (14) связи, которая/который могут представлять собой локальную сеть (LAN), глобальную сеть (WAN), интрасеть, сеть «Интернет» и т.д., с рядом клиентов или удаленных вычислительных устройств (610a), (610b), (610c), (610d), (610e) и т.д., таких как портативный компьютер, переносной компьютер, тонкий клиент, подключенное к сети бытовое радиоэлектронное или электрическое оборудование или другое устройство, такое как кассетный видеомагнитофон, телевизионный приемник, микроволновая печь, осветительный прибор, нагревательный прибор и подобные им устройства в соответствии с настоящим изобретением. Таким образом, предполагается, что настоящее изобретение может применяться к любому вычислительному устройству, в связи с которым желательно реализовать гостевые графические интерфейсы и операционные системы в соответствии с настоящим изобретением.

В сетевой среде, в которой сеть/канал (14) связи представляет собой сеть «Интернет», например серверы (10a), (10b) и т.д., могут представлять собой Web-серверы, с которыми клиенты (610a), (610b), (610c), (610d), (610e) и т.д. поддерживают связь посредством любого из ряда известных протоколов, таких как HTTP. Серверы (10a), (10b) и т.д. могут также служить в качестве клиентов (610a), (610b), (610c), (610d), (610e) и т.д., что может быть характерно для распределенной вычислительной среды.

Связь может быть проводной или беспроводной в зависимости от того, что уместно. Клиентские устройства (610a), (610b), (610c), (610d), (610e) и т.д. могут поддерживать или могут не поддерживать связь посредством сети/канала (14) связи и могут иметь связанные с ними независимые средства связи. Например, в случае телевизионного приемника или кассетного видеомагнитофона может иметься или может не иметься сетевой аспект управления этими приборами. Каждый компьютер-клиент (610a), (610b), (610c), (610d), (610e) и т.д. и компьютер-сервер (10a), (10b) и т.д. могут быть оснащены разнообразными прикладными программными модулями или объектами (635) и соединениями или доступом к различным типам элементов хранения или объектам, по которым могут храниться файлы или потоки данных или на которые часть (части) файлов или потоков данных могут загружаться, передаваться или перемещаться. Любой один или более компьютер из числа компьютеров (10a), (10b), (610a), (610b) и т.д. могут отвечать за поддержание и обновление базы (20) данных или

другого элемента хранения, такого как база данных или память (20) для хранения данных, обработанных в соответствии с изобретением. Таким образом, настоящее изобретение может быть использовано в компьютерной среде, имеющей компьютеры-клиенты (610a), (610b) и т.д., которые могут осуществлять доступ и взаимодействие с компьютерной сетью/каналом связи (14), и компьютеры-серверы (10a), (10b) и т.д., которые могут осуществлять взаимодействие с компьютерами-клиентами (610a), (610b) и т.д. и другими подобными устройствами, и базами (20) данных.

Приводимое в качестве примера вычислительное устройство

Фиг.6B и нижеследующее обсуждение имеют своей целью дать краткое общее описание подходящей вычислительной среды, в связи с которой может быть реализовано данное изобретение. Однако следует иметь в виду, что для использования в связи с настоящим изобретением предполагаются переносные, портативные и другие вычислительные устройства и вычислительные объекты всех видов, то есть всюду, где в вычислительной среде имеется графический процессор. Хотя ниже описан компьютер общего назначения, это представляет собой лишь один пример, и настоящее изобретение может быть реализовано с тонким клиентом, имеющим способность взаимодействовать и взаимосвязь с сетью/шиной. Таким образом, настоящее изобретение может быть реализовано в среде служб с сетевым хостингом, в которой подразумеваются очень малые или минимальные ресурсы у клиента, то есть в сетевой среде, в которой клиентское устройство служит просто интерфейсом сети/канала связи, таким как некоторый объект, помещенный в бытовое радиоэлектронное и электрическое оборудование. По сути дела, везде, где могут храниться данные или откуда данные могут быть извлечены или переданы в другой компьютер, имеется подходящая или пригодная среда для функционирования технологий виртуализации графических средств в соответствии с изобретением.

Хотя это и не является обязательным требованием, изобретение может быть реализовано в целом или в части посредством операционной системы, предназначенной для использования разработчиком служб для устройства или объекта и/или включенной в состав прикладных программных средств, которые функционируют в связи с виртуализированным графическим конвейером (виртуализированными графическими конвейерами) по данному изобретению. Программные средства могут быть описаны в общем контексте машиноисполняемых команд, таких как программные модули, исполняемые одним или более компьютерами, такими как клиентские рабочие станции, серверы или другие устройства. Вообще, программные модули включают в себя процедуры, программы, объекты, компоненты, структуры данных и т.п., которые выполняют конкретные задачи и/или реализуют определенные абстрактные типы данных. Обычно, функциональные возможности программных модулей могут быть объединены или распределены в различных вариантах осуществления изобретения. Кроме того, специалисты в данной области техники должны понимать, что изобретение может быть практически осуществлено и с другими конфигурациями компьютерных систем и протоколами. Другие хорошо известные вычислительные системы, среды и/или конфигурации, которые могут быть пригодны для использования с данным изобретением, включают в себя, но не в качестве ограничения: персональные компьютеры (PC), банкоматы, компьютеры-серверы, переносные или портативные устройства, многопроцессорные системы, системы, основанные на микропроцессорах, программируемую бытовую электронную аппаратуру, сетевые персональные компьютеры (PC), бытовое радиоэлектронное и электрическое оборудование,

световые приборы, элементы контроля окружающей среды, миникомпьютеры, универсальные компьютеры и тому подобное. Изобретение также может быть практически осуществлено в распределенных вычислительных средах, в которых задания выполняются удаленными устройствами обработки данных, связанными через сеть/канал связи или другую среду передачи данных. В распределенной вычислительной среде программные модули могут быть размещены как в локальных, так и в удаленных компьютерных носителях информации, включая запоминающие устройства, и узлы-клиенты могут в свою очередь вести себя как узлы-серверы.

Таким образом, фиг.6В иллюстрирует пример подходящей вычислительной системной среды (600), в которой может быть реализовано данное изобретение, хотя как было ясно сказано выше, вычислительная системная среда (600) представляет собой только один пример подходящей вычислительной среды и не имеет своей целью предлагать какое-либо ограничение в отношении объема использования или функциональных возможностей изобретения. Равным образом вычислительная системная среда (600) не должна интерпретироваться как имеющая какую-либо зависимость или требование, относящиеся к любому одному компоненту или сочетанию компонентов, проиллюстрированных в приводимой в качестве примера вычислительной системной среде (600).

Согласно фиг.6В приводимая в качестве примера система для реализации изобретения включает в себя вычислительное устройство общего назначения в форме компьютера (610). Компоненты компьютера (610) могут включать в себя, но не в смысле ограничения: процессор (620), системную память (630) и системную шину (621), которая соединяет различные компоненты системы, включая системную память, с процессором (620). Системная шина (621) может относиться к любому из нескольких типов структур шины, включая шину памяти или контроллер памяти, периферийную шину и локальную шину, использующие любую из множества архитектур шины. В качестве примера, но не ограничения, такие архитектуры включают в себя шину архитектуры отраслевого стандарта (ISA), шину микроканальной архитектуры (MCA), усовершенствованную шину ISA (EISA), локальную шину Ассоциации по стандартам в области видеоэлектроники (VESA) и шину межсоединения периферийных компонентов (PCI) (также известную как мезонинная шина) и ускоренную шину PCI (PCIe).

Компьютер (610) обычно включает в себя разнообразные машиночитаемые носители информации. Машиночитаемые носители информации могут представлять собой любые доступные носители информации, к которым может осуществлять доступ компьютер (610), и включают в себя как энергозависимые, так и энергонезависимые носители информации, как съемные, так и несъемные носители информации. В качестве примера, но не ограничения, машиночитаемые носители информации могут содержать компьютерные носители информации и среды передачи данных. Компьютерные носители информации включают в себя как энергозависимые, так и энергонезависимые, как съемные, так и несъемные носители информации, реализованные любым способом или технологией для хранения информации, такой как машиночитаемые команды, структуры данных, программные модули или другие данные. Компьютерные носители информации включают в себя, но не в ограничительном смысле, оперативное запоминающее устройство (ОЗУ, RAM), постоянное запоминающее устройство (ПЗУ, ROM), электрически стираемое программируемое постоянное запоминающее устройство (ЭСППЗУ, EEPROM), флэш-память или память, использующую другую технологию, постоянное запоминающее

устройство на компакт-диске (CD-ROM), универсальные цифровые диски (DVD) или другой оптический дисковый накопитель, магнитные кассеты, магнитную ленту, магнитный дисковый накопитель или другие магнитные запоминающие устройства или любой другой носитель, который может быть использован для хранения необходимой информации и к которому может осуществлять доступ компьютер (610). Среда передачи данных обычно воплощает машиночитаемые команды, структуры данных, программные модули или другие данные в сигнале, модулированном данными, таком как несущая или другой механизм переноса информации, и включает в себя любые среды доставки информации. Термин «сигнал, модулированный данными» означает сигнал, одна или более характеристик которого устанавливается или изменяется таким образом, чтобы обеспечить кодирование информации в сигнале. В качестве примера, но не ограничения, среда передачи данных включает в себя проводные среды, такие как проводная сеть или прямое кабельное соединение, и беспроводные среды, такие как акустическая, радиочастотная, инфракрасная и другие беспроводные среды. Комбинации любых упомянутых выше сред также должны быть включены в диапазон машиночитаемых носителей информации.

Системная память (630) включает в себя компьютерный носитель информации в виде энергозависимой и/или энергонезависимой памяти, такой как постоянное запоминающее устройство (ПЗУ, ROM) (631) и оперативное запоминающее устройство (ОЗУ, RAM) (632). Обычно в ПЗУ (631) хранится базовая система (633) ввода/вывода (BIOS), содержащая базовые процедуры, способствующие передаче информации между элементами внутри компьютера (610), например, при запуске. ОЗУ (632) обычно содержит данные и/или программные модули, к которым можно осуществить доступ немедленно, и/или которыми в текущий момент оперирует процессор (620). В качестве примера, но не ограничения, на фиг.6В проиллюстрированы операционная система (634), прикладные программы (635), другие программные модули (636) и данные (637) программ.

Компьютер (610) также может включать в себя и другие съемные/несъемные, энергозависимые/энергонезависимые компьютерные носители информации. Исключительно в качестве примера на фиг.6В проиллюстрированы накопитель (641) на жестких магнитных дисках, который осуществляет считывание с несъемных энергонезависимых магнитных носителей информации или запись на них, дисковод (651) для магнитного диска, который осуществляет считывание со съемного энергонезависимого магнитного диска (652) или запись на него, и дисковод (655) для оптического диска, который осуществляет считывание со съемного энергонезависимого оптического диска (656), такого как компакт-диск CD-ROM, или других оптических носителей информации или запись на них. Другие съемные/несъемные, энергозависимые/энергонезависимые компьютерные носители информации, которые могут быть использованы в этой приводимой в качестве примера операционной среде, включают в себя, но не в ограничительном смысле, кассеты с магнитной лентой, карточки флэш-памяти, универсальные цифровые диски, цифровую видеомагнитофонную ленту, твердотельное ОЗУ, твердотельное ПЗУ и тому подобное. Накопитель (641) на жестких магнитных дисках обычно подсоединен к системной шине (621) посредством интерфейса несъемной памяти, такого как интерфейс (640), а дисковод (651) для магнитного диска и дисковод (655) для оптического диска обычно подсоединены к системной шине (621) посредством интерфейса съемной памяти, такого как интерфейс (650).

Дисководы и соответствующие им компьютерные носители информации,

описанные выше и проиллюстрированные на фиг.6B, обеспечивают хранение машиночитаемых команд, структур данных, программных модулей и других данных для компьютера (610). Например, на фиг.6B накопитель (641) на жестких магнитных дисках проиллюстрирован как хранящий операционную систему (644), прикладные программы (645), другие программные модули (646) и данные (647) программ. Следует отметить, что эти компоненты могут либо быть идентичными операционной системе (634), прикладным программам (635), другим программным модулям (636) и данным (637) программ, либо отличаться от них. Операционной системе (644), прикладным программам (645), другим программным модулям (646) и данным (647) программ присвоены здесь другие номера для того, чтобы проиллюстрировать тот факт, что, как минимум, они являются другими копиями. Пользователь может осуществлять ввод команд и информации в компьютер (610) посредством устройств ввода, таких как клавиатура (662) и координатно-указательное устройство (661), обычно именуемое мышью, шаровым манипулятором или сенсорной панелью. В число других устройств ввода (не показанных на чертеже) могут входить микрофон, джойстик, игровая панель, спутниковая антенна, сканер или подобные им устройства. Часто эти и другие устройства ввода соединены с процессором (620) посредством интерфейса (660) пользовательского ввода, подсоединенного к системной шине (621), но они могут быть соединены с процессором посредством другого интерфейса и других структур шины, таких, например, как параллельный порт, игровой порт, или универсальная последовательная шина (USB). Это и есть типы структур, которые подвергаются виртуализации в архитектурах по данному изобретению. С системной шиной (621) также может быть соединен графический интерфейс (682), такой как один из интерфейсов, реализованных Northbridge. Northbridge представляет собой набор микросхем, который поддерживает связь с центральным процессором или главным процессором (620) и отвечает за такую связь, как связь через шины PCI, PCIe и порт ускоренной передачи графических данных (AGP). Изобретение предполагает как интегрированные (внутри Northbridge), так и дискретные (вне Northbridge) реализации графических средств. С графическим интерфейсом (682) могут поддерживать связь один или более графических процессоров (GPU) (684). В связи с этим отметим, что графические процессоры (684) обычно включают в себя внутрикристальную память, такую как память регистрового типа, и графические процессоры (684) поддерживают связь с видеопамятью (686). Однако графические процессоры (684) представляют собой лишь один пример сопроцессора, и, следовательно, в компьютер (610) могут быть включены разнообразные сопроцессорные устройства, и он может включать в себя разнообразные процедурные блоки затушевывания, такие как блоки затушевывания пикселей и вершин. Также к системной шине (626) подсоединен монитор (691) или другой тип устройства отображения, причем подсоединен посредством интерфейса, такого как видеоинтерфейс (690), который в свою очередь может поддерживать связь с видеопамятью (686). В дополнение к монитору (691) компьютеры могут также включать в себя другие периферийные устройства вывода, такие как громкоговорители (697) и принтер (696), которые могут быть подсоединены посредством периферийного интерфейса (695) вывода.

Компьютер (610) может функционировать в сетевой или распределенной среде, используя логические соединения с одним или более удаленными компьютерами, такими как удаленный компьютер (680). Удаленный компьютер (680) может представлять собой персональный компьютер, сервер, маршрутизатор, сетевой персональный компьютер (PC), одноранговое устройство или другой узел общей сети,

и обычно включает в себя многие или все элементы, описанные выше в отношении компьютера (610), хотя на фиг.6В проиллюстрировано только запоминающее устройство (681). Логические соединения, изображенные на фиг.6В, включают в себя локальную сеть (LAN) (671), и глобальную сеть (WAN) (673), но могут также включать в себя и другие сети/каналы связи. Такие сетевые среды часто используются в офисах, компьютерных сетях масштаба предприятия, интрасетях и в сети «Интернет».

При использовании в сетевой среде LAN компьютер (610) соединен с LAN (671) посредством сетевого интерфейса или адаптера (670). При использовании в сетевой среде WAN компьютер (610) обычно содержит модем (672) или другое средство для установления связи через сеть WAN (673), такую как сеть Интернет. Модем (672), который может быть внутренним или внешним, может быть подсоединен к системной шине (621) посредством интерфейса (660) пользовательского ввода или другого соответствующего механизма. В сетевой среде программные модули, изображенные в отношении компьютера (610), или их части могут храниться в удаленном запоминающем устройстве. В качестве примера, но не ограничения, фиг.6В изображает удаленные прикладные программы (685), как размещенные на запоминающем устройстве (681). Следует понимать, что показанные сетевые соединения приводятся в качестве примера и могут быть использованы и другие средства установления линии связи между компьютерами.

Существует множество способов реализации настоящего изобретения, например соответствующий интерфейс прикладного программирования, набор инструментальных средств, код драйвера, операционная система, средство управления, автономные или загружаемые программные объекты и т.д., которые предоставляют приложениям и службам возможность использовать виртуализированные архитектуру (архитектуры), системы и способы по данному изобретению. Изобретение рассматривает использование изобретения с точки зрения интерфейса прикладного программирования (или другого программного объекта), равно как и с точки зрения объекта программных средств или аппаратных средств, который вмещает в себе какую-либо из вышеупомянутых технологий в соответствии с изобретением. Таким образом, разнообразные варианты реализации изобретения, описанные в данной заявке, могут иметь аспекты, которые реализованы целиком в виде аппаратных средств, частично в виде аппаратных средств, а частично в виде программных средств, равно как и в виде программных средств.

Как было упомянуто выше, хотя приводимые в качестве примера варианты осуществления настоящего изобретения описаны в связи с разнообразными вычислительными устройствами и сетевыми архитектурами, лежащие в их основе концепции могут быть применены к любому вычислительному устройству или системе, в которой желательно использовать графический процессор. Например, разнообразные алгоритм (алгоритмы) и реализации аппаратных средств по данному изобретению могут быть применены к операционной системе вычислительного устройства, предоставленной как отдельный объект на этом устройстве, как часть другого объекта, как многократно используемое средство управления, как загружаемый с сервера объект, как «посредник» между устройством или объектом и сетью, как распределенный объект, как аппаратные средства, в памяти, как комбинация любых из вышеперечисленных вариантов и т.д. Специалист, имеющий обычную квалификацию в данной области техники, должен понимать, что имеется множество способов предоставления кода и номенклатуры объектов, которые обеспечивают те же самые, схожие или эквивалентные функциональные возможности,

как те, что обеспечены различными вариантами осуществления изобретения.

Как было упомянуто, описанные здесь разнообразные технологии могут быть реализованы в связи с программными средствами, аппаратными средствами или там, где это уместно, комбинацией обоих видов средств. Таким образом, способы и устройство по настоящему изобретению или некоторые их аспекты или части могут принимать форму программного кода, то есть команд, воплощенных в осязаемых носителях информации, таких как гибкие дискеты, ПЗУ на компакт-дисках (CD-ROM), накопители на жестких магнитных дисках или любой другой машиночитаемый носитель информации, причем, когда этот программный код загружается в машину, такую как компьютер, и исполняется ею, эта машина становится устройством для практического осуществления изобретения. В случае исполнения программного кода на программируемых компьютерах, вычислительное устройство обычно включает в себя процессор, носитель информации, могущий быть считанным компьютером, (включая энергозависимую и энергонезависимую память и/или запоминающие элементы), по меньшей мере, одно устройство ввода и, по меньшей мере, одно устройство вывода. Предпочтительно, чтобы одна или более программ, которые могут реализовывать или использовать технологии виртуализации графического процессора по настоящему изобретению, например, посредством использования интерфейса прикладного программирования, обрабатывающего данные, средств управления многократного использования или им подобных, были реализованы на процедурно- или объектно-ориентированном языке программирования высокого уровня для коммуникации с компьютерной системой. Однако эта программа (программы) может быть реализована (могут быть реализованы) на языке ассемблера или машинном языке и сочетаться с аппаратными реализациями.

Способы и устройство по настоящему изобретению также могут быть практически осуществлены посредством сообщений, воплощенных в форме программного кода, который передается в некоторой среде передачи данных, как например, по электрическим проводам или кабелям, по волоконной оптике или посредством любой другой формы передачи данных, причем, когда этот программный код принят и загружен в машину, такую как стираемое программируемое постоянное запоминающее устройство (СППЗУ, EPROM), вентильная матрица, программируемое логическое устройство (ПЛУ, PLD), компьютер-клиент и т.д., и исполняется ею, эта машина становится устройством для практического осуществления изобретения. При реализации на процессоре общего назначения программный код объединяется с процессором и предоставляет уникальное устройство, которое работает таким образом, чтобы создать функциональные возможности по настоящему изобретению. Кроме того, любая технология хранения данных, используемая в связи с настоящим изобретением, может неизменно быть комбинацией аппаратных средств и программных средств.

Несмотря на то, что настоящее изобретение было описано в связи с предпочтительными вариантами своего осуществления, показанными на различных фигурах, следует понимать, что могут быть использованы другие аналогичные варианты осуществления изобретения, а в описанные варианты осуществления могут быть внесены изменения и дополнения для выполнения той же самой функции по настоящему изобретению, и это не выходит за рамки изобретения. Например, хотя приводимые в качестве примера сетевые среды по данному изобретению описаны в контексте сетевой среды, такой как одноранговая сетевая среда, специалист в данной области техники должен признать, что настоящее изобретение этим не ограничено, и

что способы, описанные в настоящей заявке, могут применяться к любому вычислительному устройству или среде, таким как игровая консоль, переносной компьютер, портативный компьютер и т.д., как с проводной, так и с беспроводной связью, и могут быть применены к любому количеству таких вычислительных устройств, соединенных посредством сети связи и осуществляющих взаимодействие по этой сети. Помимо этого, следует подчеркнуть, что предполагается многообразие компьютерных платформ, включая операционные системы для переносных устройств и другие операционные системы, специализированные по применению, особенно в свете того, что количество сетевых устройств с беспроводной связью продолжает расти.

Несмотря на то, что предпочтительные варианты осуществления изобретения относятся к использованию настоящего изобретения в контексте графического конвейера, изобретение этим не ограничено, но, напротив, может быть равным образом реализовано для виртуализации второго специализированного процессора, взаимодействующего с главным процессором по другим основаниям. Кроме того, изобретение предполагает сценарий, в котором множественные экземпляры одной и той же версии или редакции операционной системы функционируют в отдельных виртуальных машинах в соответствии с изобретением. Можно иметь в виду, что виртуализация по данному изобретению не зависит от операций, для которых используется графический процессор. Также подразумевается, что изобретение применяется ко всем архитектурам графических процессоров, включая, но не в качестве ограничения, реализации со множественными графическими процессорами, равно как и варианты осуществления со множеством графических процессоров, которые создают иллюзию единственного интерфейса графического процессора. Также, помимо этого, настоящее изобретение может быть реализовано во множестве обрабатывающих микросхем или устройств, или с распределением по этому множеству, и хранение данных может быть аналогичным образом осуществлено с распределением по множеству устройств. Следовательно, настоящее изобретение не должно быть ограничено никаким единственным вариантом своего осуществления, но, напротив, должно толковаться во всей своей широте и объеме в соответствии с прилагаемой формулой изобретения.

Формула изобретения

1. Способ обработки вызовов графического интерфейса прикладного программирования (API) в вычислительной системе, имеющей главную операционную систему, первую гостевую операционную систему, хостинг которой осуществляется в первой виртуальной машине, и вторую гостевую операционную систему, хостинг которой осуществляется во второй виртуальной машине, включающий в себя этапы, на которых:

принимают первый вызов графического API для обработки графическим процессором (GPU) графической подсистемы вычислительной системы, причем первый вызов графического API имеет виртуальный адрес, соответствующий пространству реальных адресов графической подсистемы, и создан упомянутой первой виртуальной машиной; и

обрабатывают упомянутый вызов графического API графическим компонентом монитора виртуальной машины компонента монитора виртуальной машины, который дает возможность функционирования виртуальной машине, при этом графический компонент монитора виртуальной машины отделен от главной

операционной системы;

принимают второй вызов графического API для обработки GPU, причем второй вызов графического API имеет виртуальный адрес, соответствующий пространству реальных адресов графической подсистемы, и создан упомянутой второй виртуальной машиной;

обрабатывают упомянутый второй вызов графического API графическим компонентом монитора виртуальной машины;

составляют компонентом составления отображение, содержащее обработанный первый вызов графического API и обработанный второй вызов графического API, причем компонент составления имеет доступ к памяти GPU, в которой хранится обработанный первый вызов графического API, и памяти GPU, в которой хранится обработанный второй вызов графического API; и отображают отображение на устройстве отображения.

2. Способ по п.1, в котором этап обработки включает в себя этап, на котором:

выделяют, по меньшей мере, один физический ресурс, содержащий, по меньшей мере, один ресурс из числа системной памяти или видеопамяти, для вызова графического API.

3. Способ по п.2, в котором этап выделения, по меньшей мере, одного физического ресурса включает в себя этап, на котором выделяют отдельный, по меньшей мере, один ресурс из числа области видеопамяти или области системной памяти для вызовов графического API, созданных отдельными приложениями, исполняемыми на виртуальной машине.

4. Способ по п.1, в котором этап обработки включает в себя этап, на котором определяют, является ли вызов графического API привилегированным или непривилегированным, и применяют различные наборы правил для обработки привилегированного в отличие от непривилегированного вызова графического API.

5. Способ по п.1, в котором компонент составления содержит «доверенную» базу кода, созданную компонентом монитора виртуальной машины.

6. Способ по п.1, который также содержит этап, на котором

осуществляют обработку выходных данных «не наделенного доверием» приложения, исполняемого в первой виртуальной машине, включающую в себя использование, по меньшей мере, одной «доверенной» службы, исполняемой в третьей виртуальной машине для защиты упомянутых выходных данных.

7. Графический процессор, выполненный с возможностью реализации способа по п.1.

8. Вычислительная система для обработки вызовов интерфейса прикладного программирования (API), содержащая процессор;

главную операционную систему;

гостевую операционную систему, хостинг которой осуществляется в виртуальной машине;

графический процессор графической подсистемы и компонент монитора виртуальной машины, при этом компонент монитора виртуальной машины отделен от главной операционной системы и имеет графический компонент монитора виртуальной машины, при этом компонент монитора виртуальной машины дает возможность функционирования виртуальной машине, и графический компонент монитора виртуальной машины имеет прямой доступ к графической подсистеме и сконфигурирован с возможностью:

обеспечивать гостевую операционную систему пространством виртуальных

адресов для графической подсистемы, при этом пространство виртуальных адресов соответствует пространству физических адресов графической подсистемы;

принимать вызов графического API для обработки графическим процессором, при этом вызов графического API создан виртуальной машиной;

переводить адрес одного или более элементов работы из пространства виртуальных адресов гостевой операционной системы в соответствующее пространство физических адресов графической подсистемы;

передавать вызов графического API непосредственно в графическую подсистему для обработки графическим процессором;

идентифицировать результат, содержащий результат обработки вызова графического API графическим процессором, причем графический компонент монитора виртуальной машины имеет доступ к адресу результата обработки вызова графического API графическим процессором в памяти; и передавать результат в главную операционную систему для составления и отображения на устройстве отображения.

9. Вычислительная система по п.8, в которой графический компонент монитора виртуальной машины также сконфигурирован для выделения для вызова графического API, по меньшей мере, одного физического ресурса, содержащего, по меньшей мере, один ресурс из системной памяти или видеопамяти, во время обработки вызова графического API.

10. Вычислительная система по п.9, в которой выделение, по меньшей мере, одного физического ресурса включает в себя выделение отдельного, по меньшей мере, одного ресурса из области видеопамяти или области системной памяти для вызовов графического API, созданных отдельными приложениями, исполняемыми на виртуальной машине.

11. Вычислительная система по п.8, в которой графический компонент монитора виртуальной машины также сконфигурирован с возможностью определения, является ли вызов графического API привилегированным или непривилегированным, и применения различных наборов правил для обработки привилегированного в отличие от непривилегированного вызова графического API.

12. Вычислительная система по п.8, в которой главная операционная система содержит «доверенную» базу кода.

13. Вычислительная система по п.8, в которой графический компонент монитора виртуальной машины также сконфигурирован с возможностью обработки выходных данных «не наделенного доверием» приложения, исполняемого в виртуальной машине, включающей в себя использование, по меньшей мере, одной «доверенной» службы, исполняемой во второй виртуальной машине для защиты упомянутых выходных данных.

14. Машиночитаемый носитель данных, имеющий команды для исполнения компьютером для обработки вызовов интерфейса прикладного программирования (API) в вычислительной системе, имеющей главную операционную систему, первую операционную систему, хостинг которой осуществляется в первой виртуальной машине, и вторую операционную систему, хостинг которой осуществляется во второй виртуальной машине, каждая виртуальная машина имеет графический драйвер, соответствующий графической подсистеме вычислительной системы, первая операционная система имеет графический API, который отличается от графического API второй операционной системы, при этом команды для исполнения компьютером содержат команды для:

обеспечения первой операционной системы первым пространством виртуальных адресов для графической подсистемы, при этом первое пространство виртуальных адресов соответствует пространству адресов графической подсистемы;

обеспечения второй операционной системы вторым пространством виртуальных адресов для графической подсистемы, при этом второе пространство виртуальных адресов соответствует пространству адресов графической подсистемы;

приема вызова графического API от или графического драйвера первой операционной системы, или графического драйвера второй операционной системы, при этом графический драйвер переводит принятый вызов в графическом API, соответствующем операционной системе графического драйвера, в вызов графической подсистемы;

перевода виртуального адреса вызова графического API в соответствующий физический адрес и

обработки вызова графического API графическим компонентом монитора виртуальной машины, который дает возможность функционировать первой и второй виртуальным машинам, используя физический адрес вызова графической подсистемы, при этом компонент монитора виртуальной машины отделен от главной операционной системы и имеет прямой доступ к графической подсистеме.

15. Машиночитаемый носитель данных по п.14, в котором команды для исполнения компьютером также содержат команды для:

выделения, по меньшей мере, одного физического ресурса, содержащего, по меньшей мере, один ресурс из системной памяти или видеопамяти для упомянутого вызова графического API.

16. Машиночитаемый носитель данных по п.15, в котором выделение, по меньшей мере, одного физического ресурса включает в себя выделение отдельного, по меньшей мере, одного ресурса из области видеопамяти или области системной памяти для вызовов графического API, созданных отдельными приложениями, исполняемыми на первой виртуальной машине или второй первой виртуальной машине.

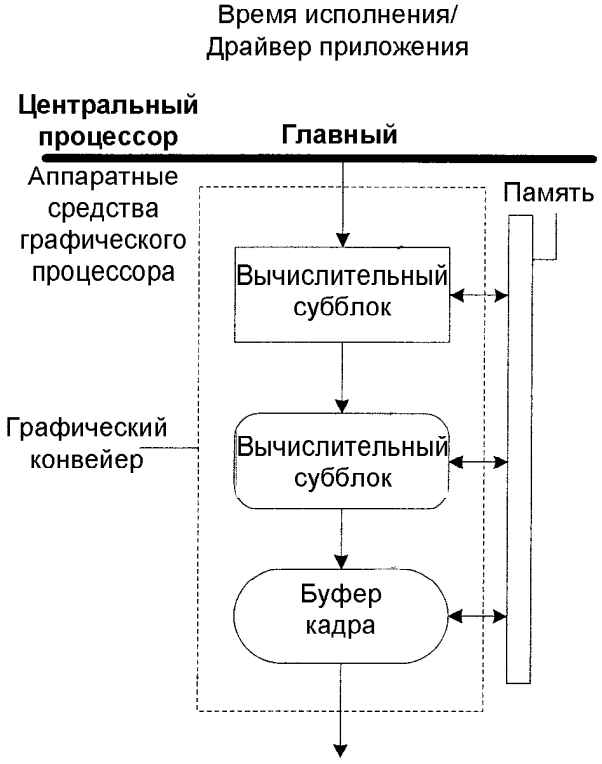
17. Машиночитаемый носитель данных по п.14, в котором обработка вызова графического API включает в себя определение, является ли вызов графического API привилегированным или непривилегированным, и применение различных наборов правил для обработки привилегированного в отличие от непривилегированного вызова графического API.

18. Машиночитаемый носитель данных по п.14, в котором команды для исполнения компьютером также содержат команды для

составления отображения выходных данных одного или более элементов работы из «доверенной» базы кода, созданной компонентом монитора виртуальной машины.

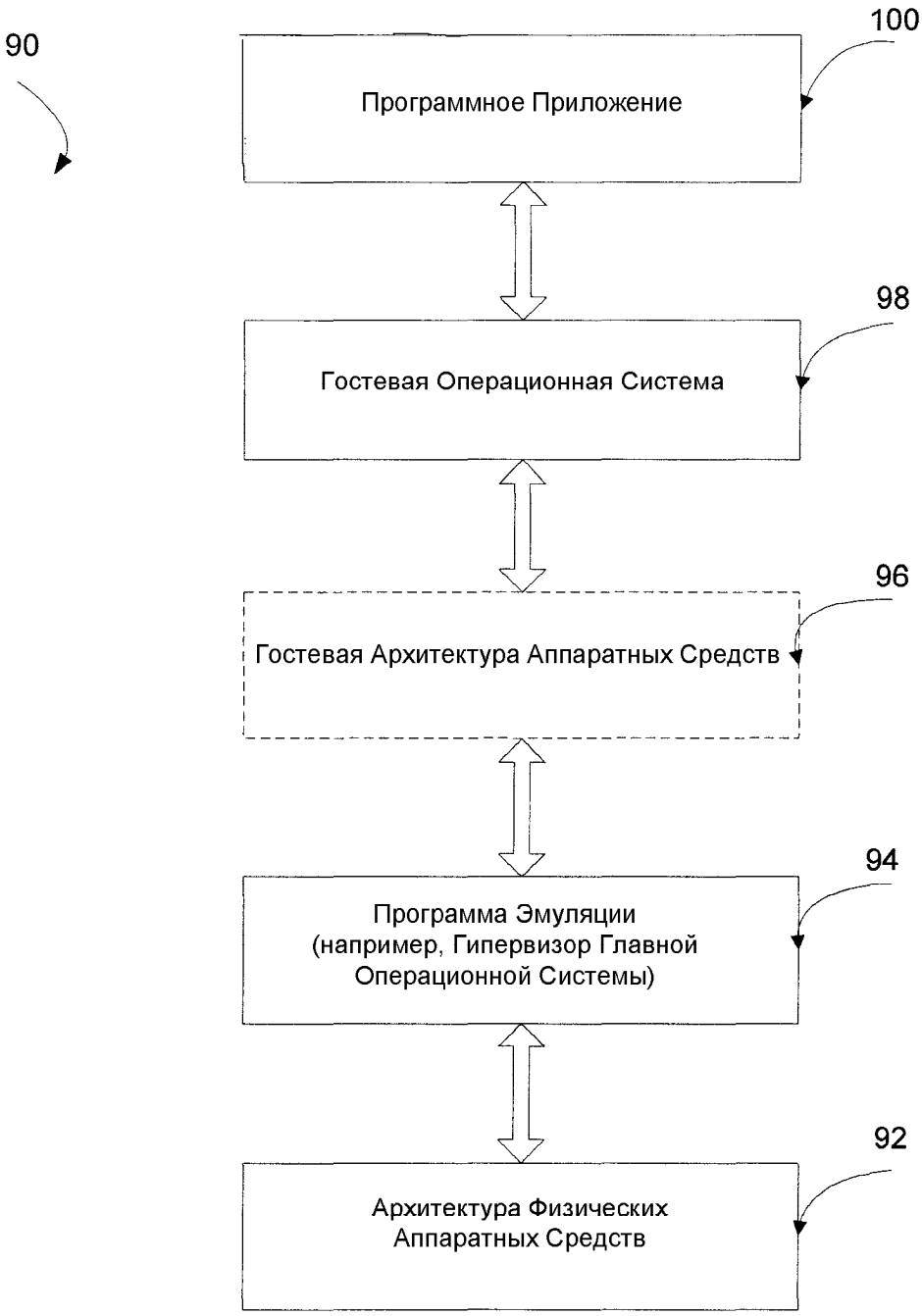
19. Машиночитаемый носитель данных по п.14, в котором команды для исполнения компьютером также содержат команды для

обработки выходных данных «не наделенного доверием» приложения, исполняемого в первой виртуальной машине, включающей в себя использование, по меньшей мере, одной «доверенной» службы, исполняемой во второй виртуальной машине для защиты упомянутых выходных данных.

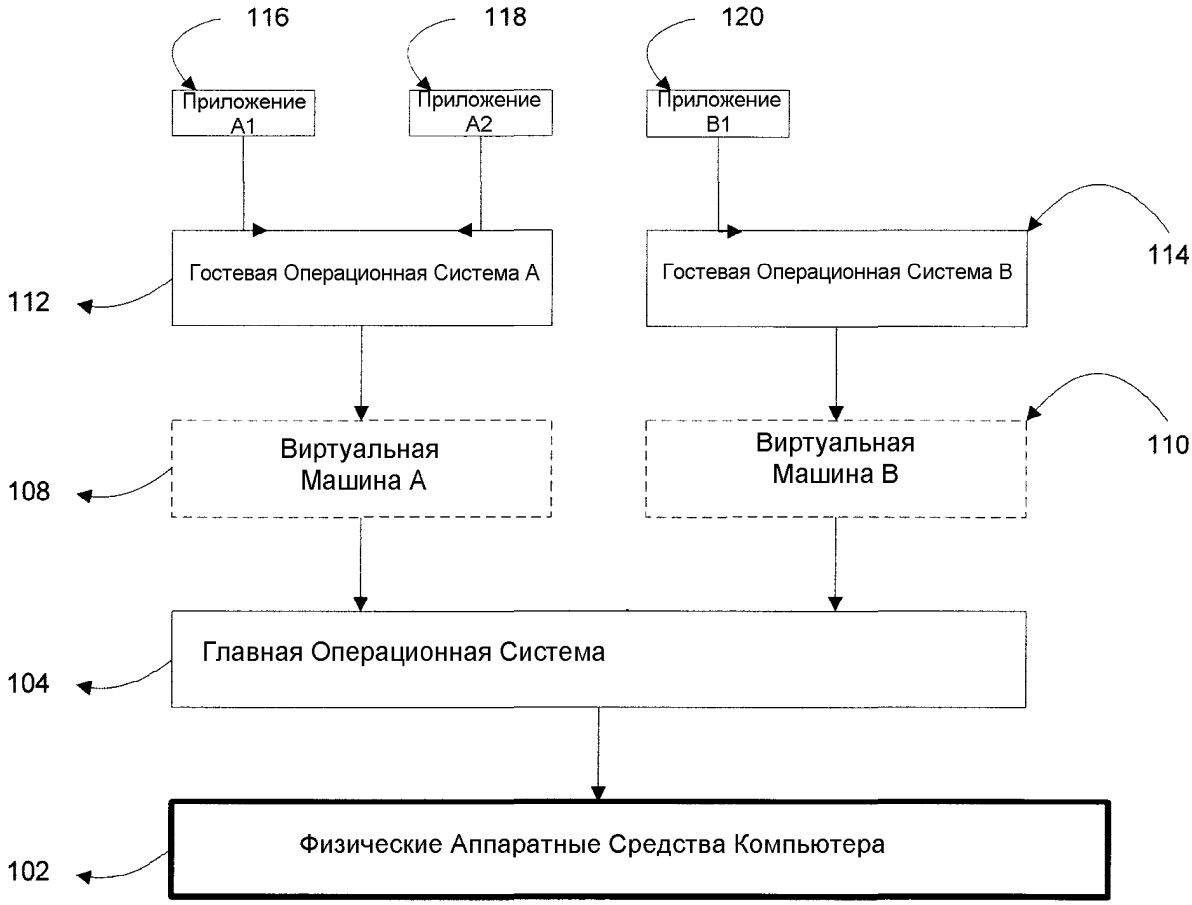


Предшествующий уровень техники

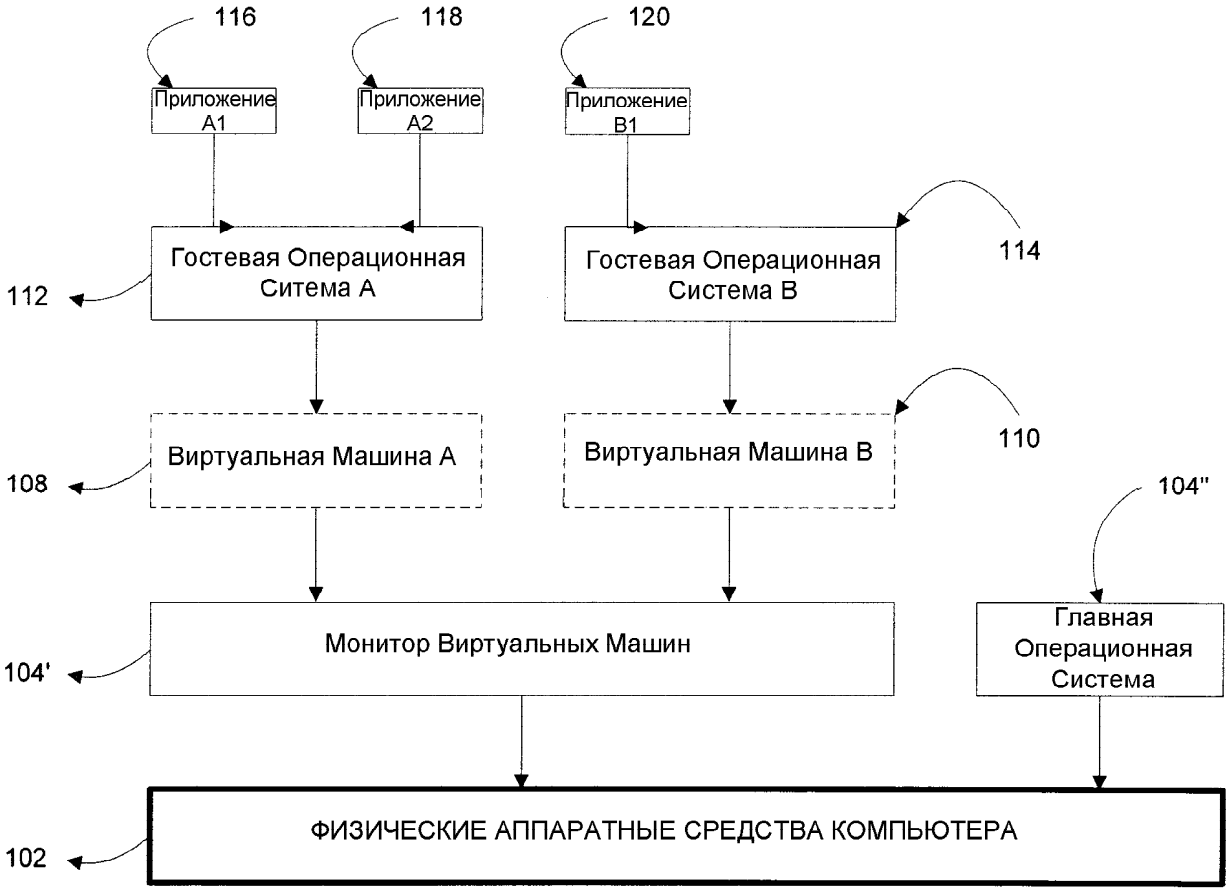
ФИГ. 1


90
100
Программное Приложение
98
Гостевая Операционная Система
96
Гостевая Архитектура Аппаратных Средств
94
Программа Эмуляции
(например, Гипервизор Главной
Операционной Системы)
92
Архитектура Физических
Аппаратных Средств

ФИГ. 2А


116
118
120
Приложение A1
Приложение A2
Приложение B1
Гостевая Операционная Система A
Гостевая Операционная Система B
114
112
Виртуальная Машина A
Виртуальная Машина B
110
108
Главная Операционная Система
104
Физические Аппаратные Средства Компьютера
102

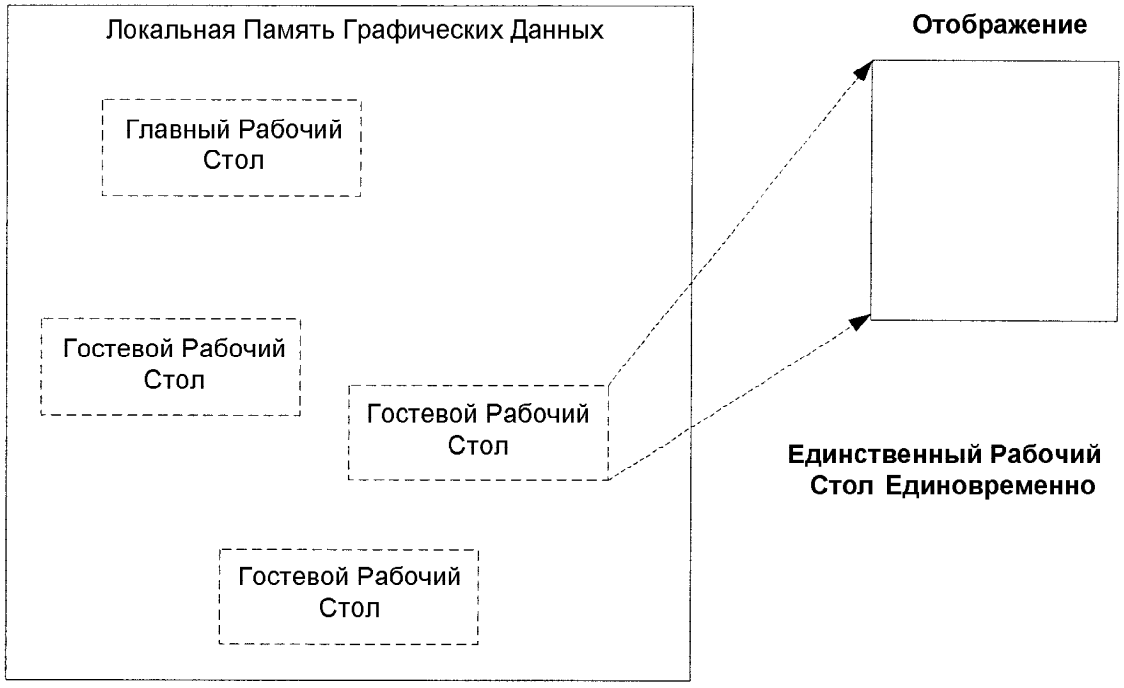
ФИГ. 2B


116
118
120
Приложение A1
Приложение A2
Приложение B1
Гостевая Операционная Ситема A
Гостевая Операционная Система B
114
112
Виртуальная Машина A
Виртуальная Машина B
110
108
104"
Монитор Виртуальных Машин
Главная Операционная Система
104'
ФИЗИЧЕСКИЕ АППАРАТНЫЕ СРЕДСТВА КОМПЬЮТЕРА
102

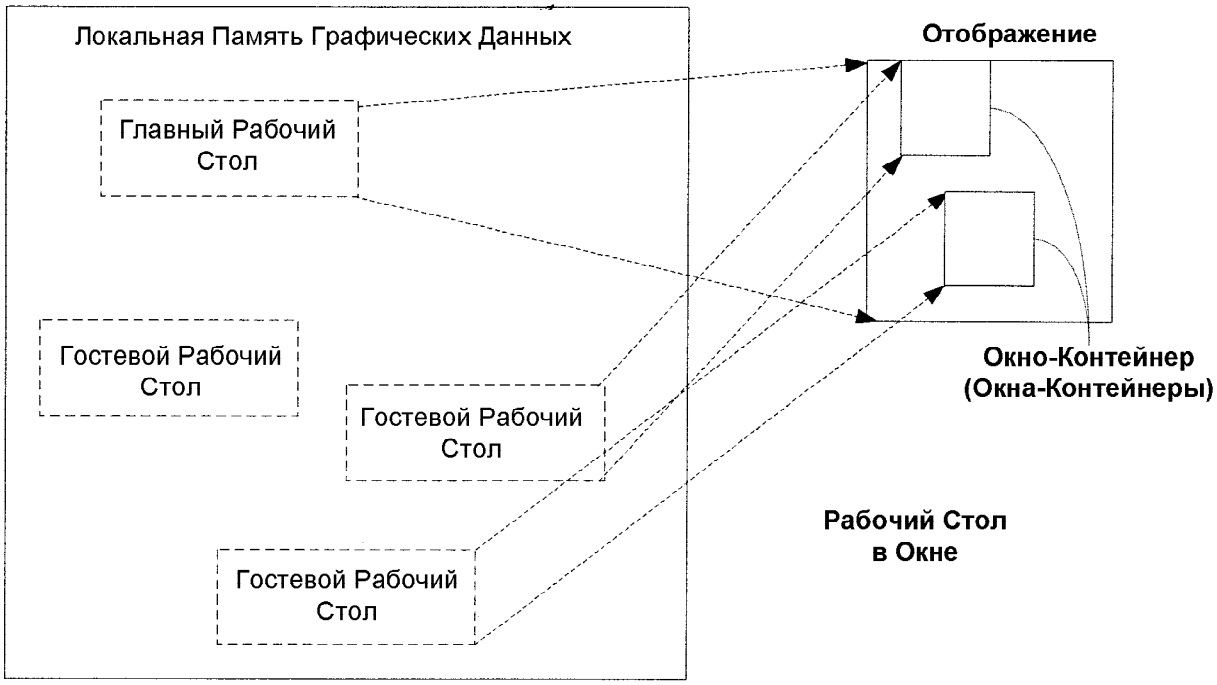
ФИГ. 2C

Главная Операционная Система
Графический драйвер HGD
Гостевая Операционная Система A
Графический драйвер AGD
Гостевая Операционная Система B
Графический драйвер BGD
Монитор Виртуальных Машин
304
Графический Монитор Виртуальных Машин
ГРАФИЧЕСКИЕ АППАРАТНЫЕ СРЕДСТВА
302

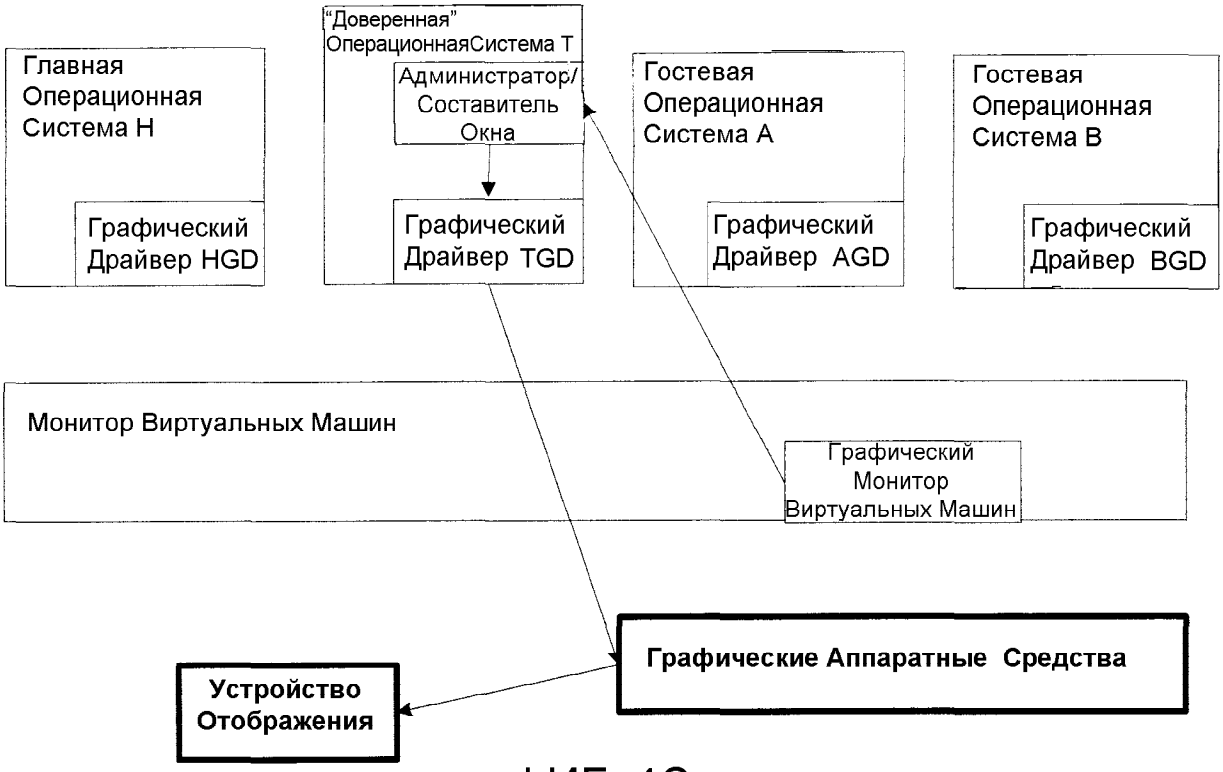
ФИГ. 3B


Локальная Память Графических Данных
Главный Рабочий Стол
Гостевой Рабочий Стол
Гостевой Рабочий Стол
Гостевой Рабочий Стол
Отображение
Единственный Рабочий Стол Единовременно

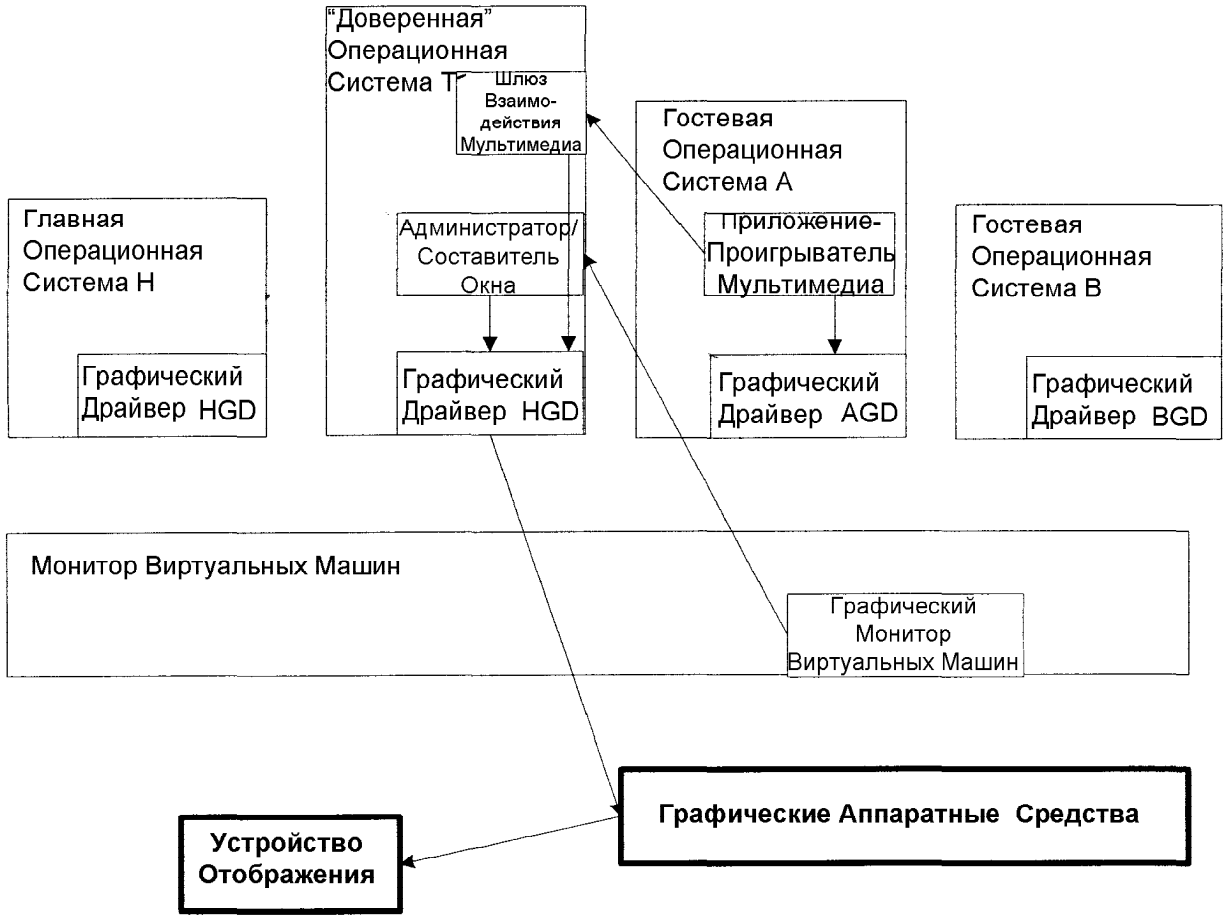
ФИГ. 4A


Локальная Память Графических Данных
Главный Рабочий Стол
Гостевой Рабочий Стол
Гостевой Рабочий Стол
Гостевой Рабочий Стол
Отображение
Окно-Контейнер (Окна-Контейнеры)
Рабочий Стол в Окне

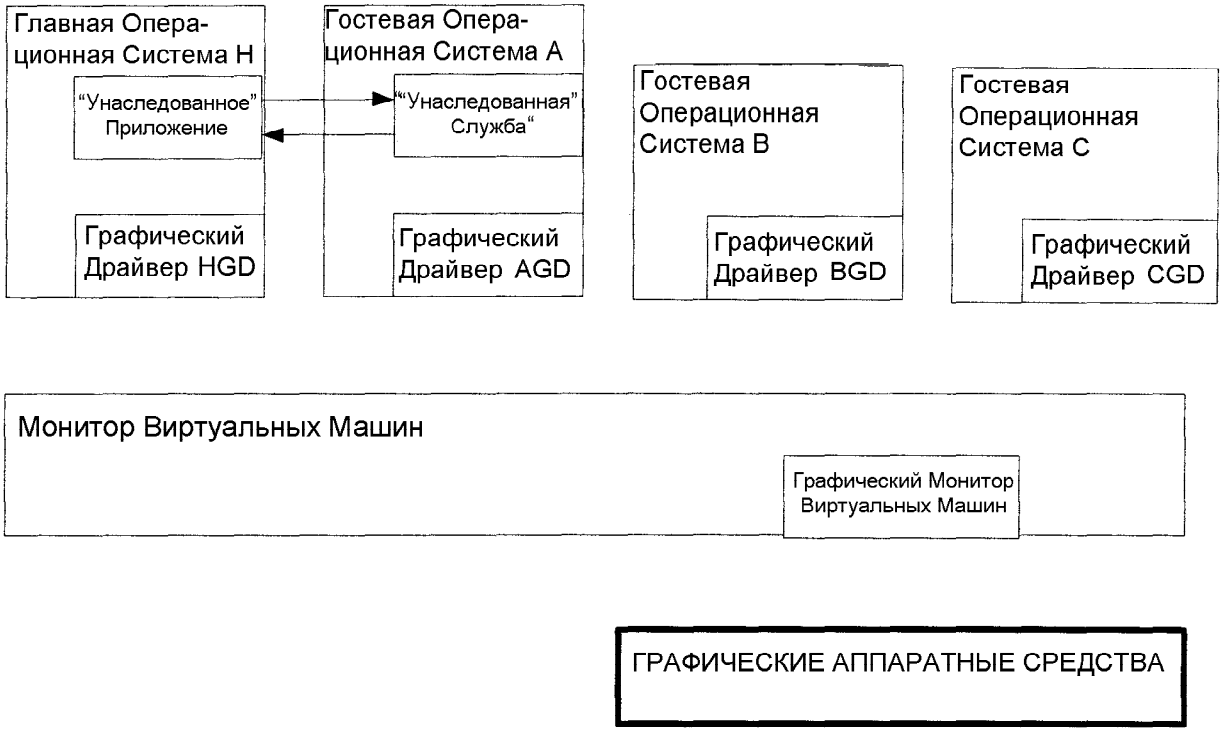
ФИГ. 4B


Главная Операционная Система H
Графический Драйвер HGD
"Доверенная" ОперационнаяСистема T
Администратор/ Составитель Окна
Графический Драйвер TGD
Гостевая Операционная Система A
Графический Драйвер AGD
Гостевая Операционная Система B
Графический Драйвер BGD
Монитор Виртуальных Машин
Графический Монитор Виртуальных Машин
Графические Аппаратные Средства
Устройство Отображения

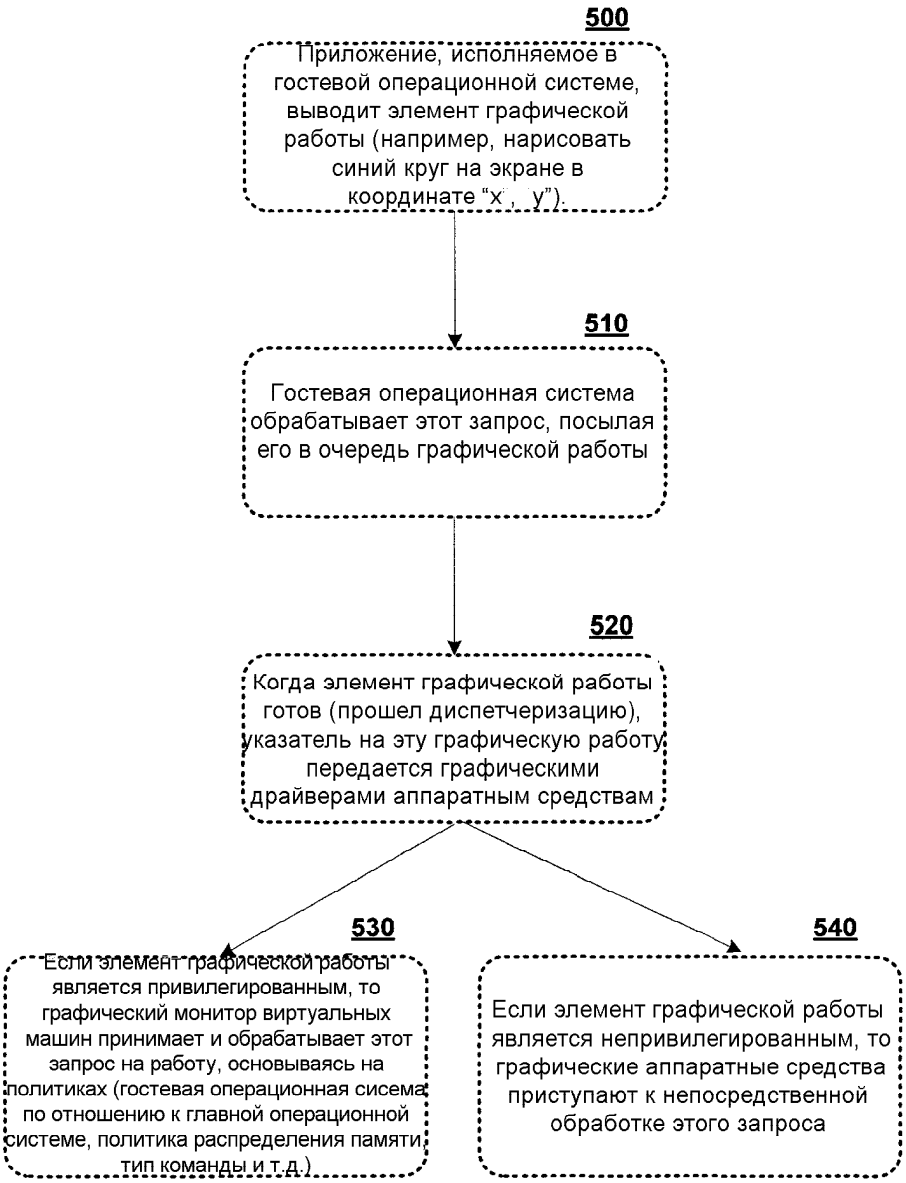
ФИГ. 4C


"Доверенная" Операционная Система T
Шлюз Взаимо-действия Мультимедиа
Гостевая Операционная Система A
Главная Операционная Система H
Администратор/ Составитель Окна
Приложение-Проигрыватель Мультимедиа
Гостевая Операционная Система B
Графический Драйвер HGD
Графический Драйвер HGD
Графический Драйвер AGD
Графический Драйвер BGD
Монитор Виртуальных Машин
Графический Монитор Виртуальных Машин
Графические Аппаратные Средства
Устройство Отображения

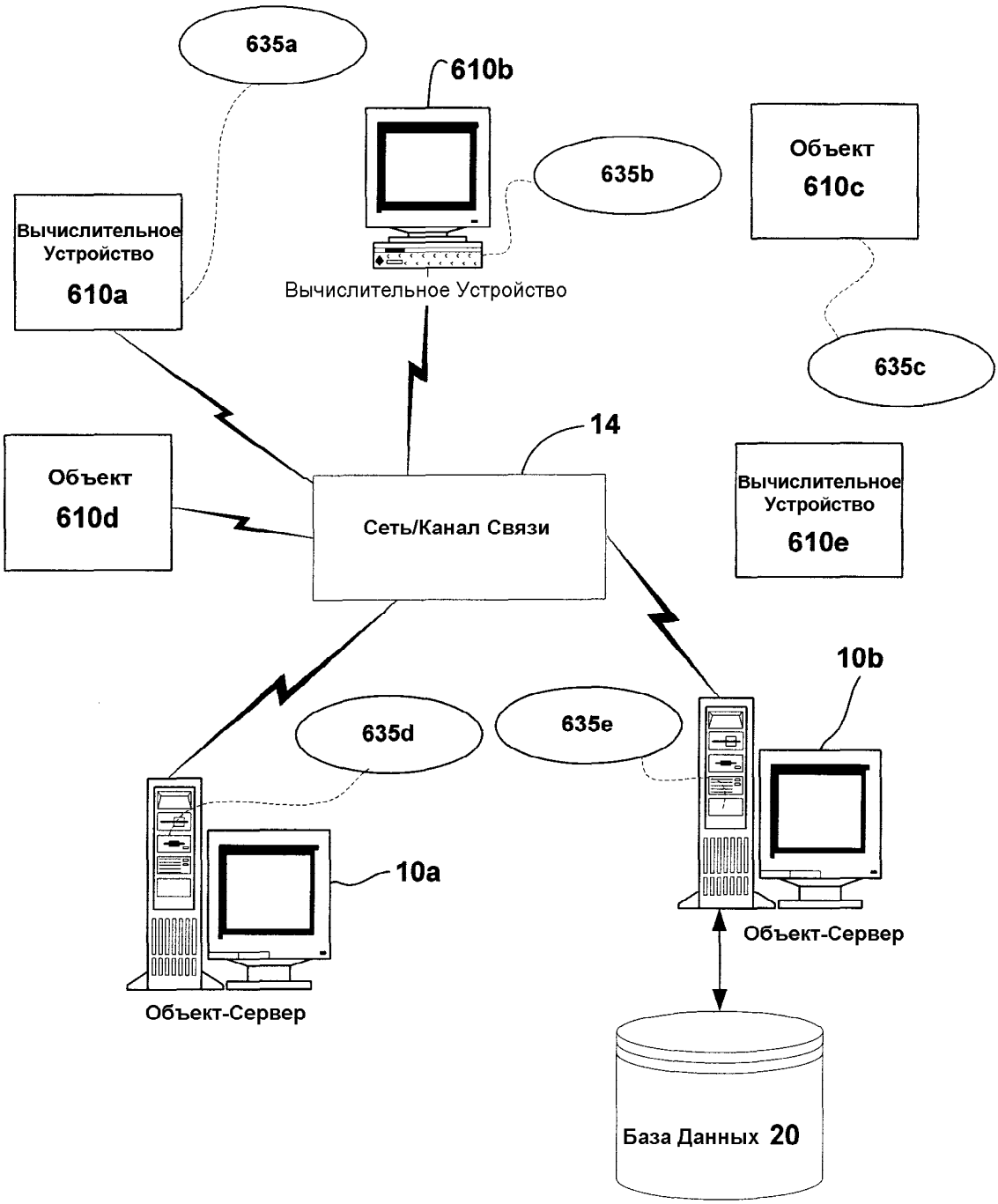
ФИГ. 4D


Главная Операционная Система H
Гостевая Операционная Система A
Гостевая Операционная Система B
Гостевая Операционная Система C
"Унаследованное" Приложение
"Унаследованная" Служба"
Графический Драйвер HGD
Графический Драйвер AGD
Графический Драйвер BGD
Графический Драйвер CGD
Монитор Виртуальных Машин
Графический Монитор Виртуальных Машин
ГРАФИЧЕСКИЕ АППАРАТНЫЕ СРЕДСТВА

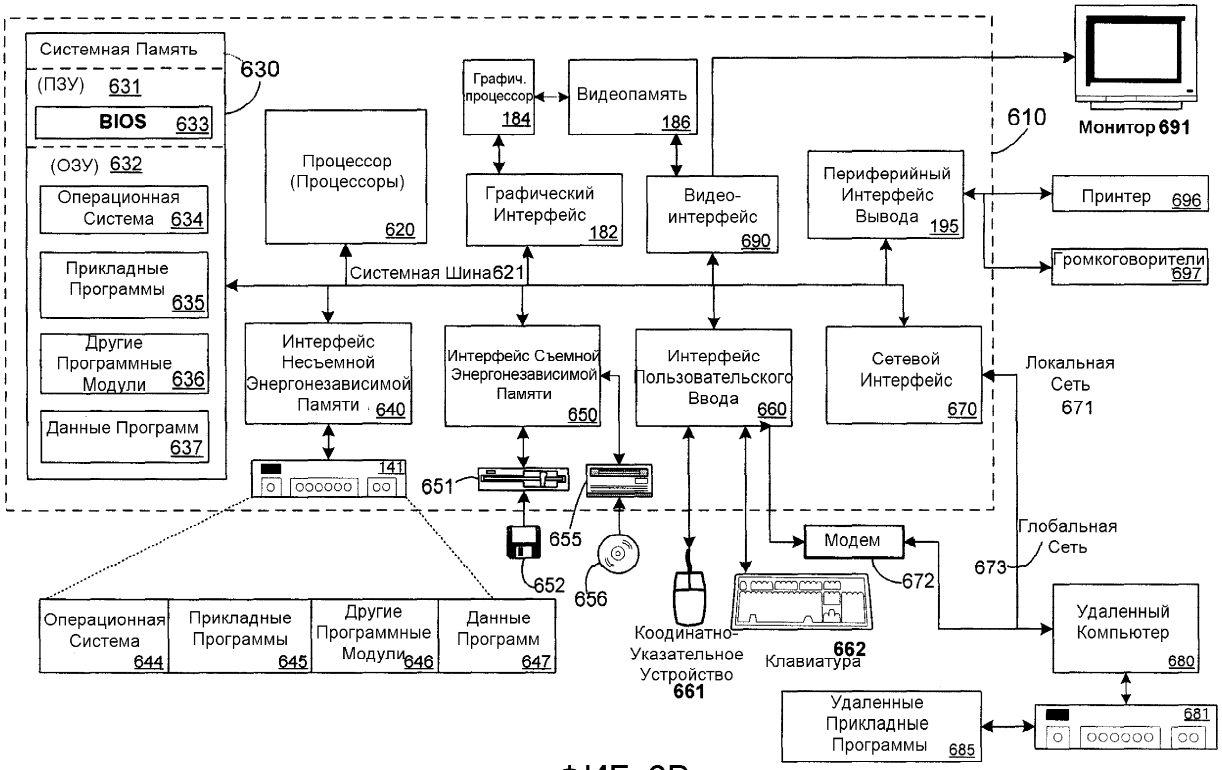
ФИГ. 4E


500
Приложение, исполняемое в гостевой операционной системе, выводит элемент графической работы (например, нарисовать синий круг на экране в координате "x , y").
510
Гостевая операционная система обрабатывает этот запрос, посылая его в очередь графической работы
520
Когда элемент графической работы готов (прошел диспетчеризацию), указатель на эту графическую работу передается графическими драйверами аппаратным средствам
530
Если элемент графической работы является привилегированным, то графический монитор виртуальных машин принимает и обрабатывает этот запрос на работу, основываясь на политиках (гостевая операционная сисема по отношению к главной операционной системе, политика распределения памяти, тип команды и т.д.)
540
Если элемент графической работы является непривилегированным, то графические аппаратные средства приступают к непосредственной обработке этого запроса

ФИГ. 5


635a
610b
635b
Объект
610c
Вычислительное Устройство
610a
Вычислительное Устройство
635c
14
Объект
610d
Сеть/Канал Связи
Вычислительное Устройство
610e
10b
635d
635e
10a
Объект-Сервер
Объект-Сервер
База Данных 20

ФИГ. 6А


Вычислительная Среда 600
Системная Память
(ПЗУ) 631
BIOS 633
(ОЗУ) 632
Операционная Система 634
Прикладные Программы 635
Другие Программные Модули 636
Данные Программ 637
630
Процессор (Процессоры) 620
Граф. процессор 184
Видеопамять 186
Графический Интерфейс 182
Видео-интерфейс 690
Периферийный Интерфейс Вывода 195
Монитор 691
610
Принтер 696
Громкоговорители 697
Системная Шина 621
Интерфейс Несъемной Энергонезависимой Памяти 640
Интерфейс Съемной Энергонезависимой Памяти 650
Интерфейс Пользовательского Ввода 660
Сетевой Интерфейс 670
Локальная Сеть 671
141
651
655
652
656
Модем
672
673
Глобальная Сеть
Операционная Система 644
Прикладные Программы 645
Другие Программные Модули 646
Данные Программ 647
Координатно-Указательное Устройство 661
Клавиатура 662
Удаленный Компьютер 680
Удаленные Прикладные Программы 685
681

ФИГ. 6В