РОССИЙСКАЯ ФЕДЕРАЦИЯ



ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(19) **RU** (11) **2 569 805**(13) **C2**

(51) МПК
G06F 12/02 (2006.01)

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21)(22) Заявка: **2011139151/08, 19.03.2010**

(24) Дата начала отсчета срока действия патента:
19.03.2010

Приоритет(ы):
(30) Конвенционный приоритет:
26.03.2009 US 12/412,272

(43) Дата публикации заявки: **27.03.2013** Бюл. № **9**

(45) Опубликовано: **27.11.2015** Бюл. № **33**

(56) Список документов, цитированных в отчете о поиске: **US 2005/0120160 A1, 02.06.2005. US 2006/0206891 A1, 14.09.2006. US 2006/0184938 A1, 17.08.2006. US 6075938 A, 13.06.2000. RU 2003123112 A, 27.02.2005.**

(85) Дата начала рассмотрения заявки PCT на национальной фазе: **23.09.2011**

(86) Заявка PCT:
US 2010/028034 (19.03.2010)

(87) Публикация заявки PCT:
WO 2010/111149 (30.09.2010)

Адрес для переписки:
129090, Москва, ул. Б. Спасская, 25, строение 3, ООО "Юридическая фирма Городисский и Партнеры"

(72) Автор(ы):
ОШИНС Джейкоб (US)

(73) Патентообладатель(и):
МАЙКРОСОФТ ТЕКНОЛОДЖИ ЛАЙСЕНСИНГ, ЭлЭлСи (US)

RU 2569805 C2

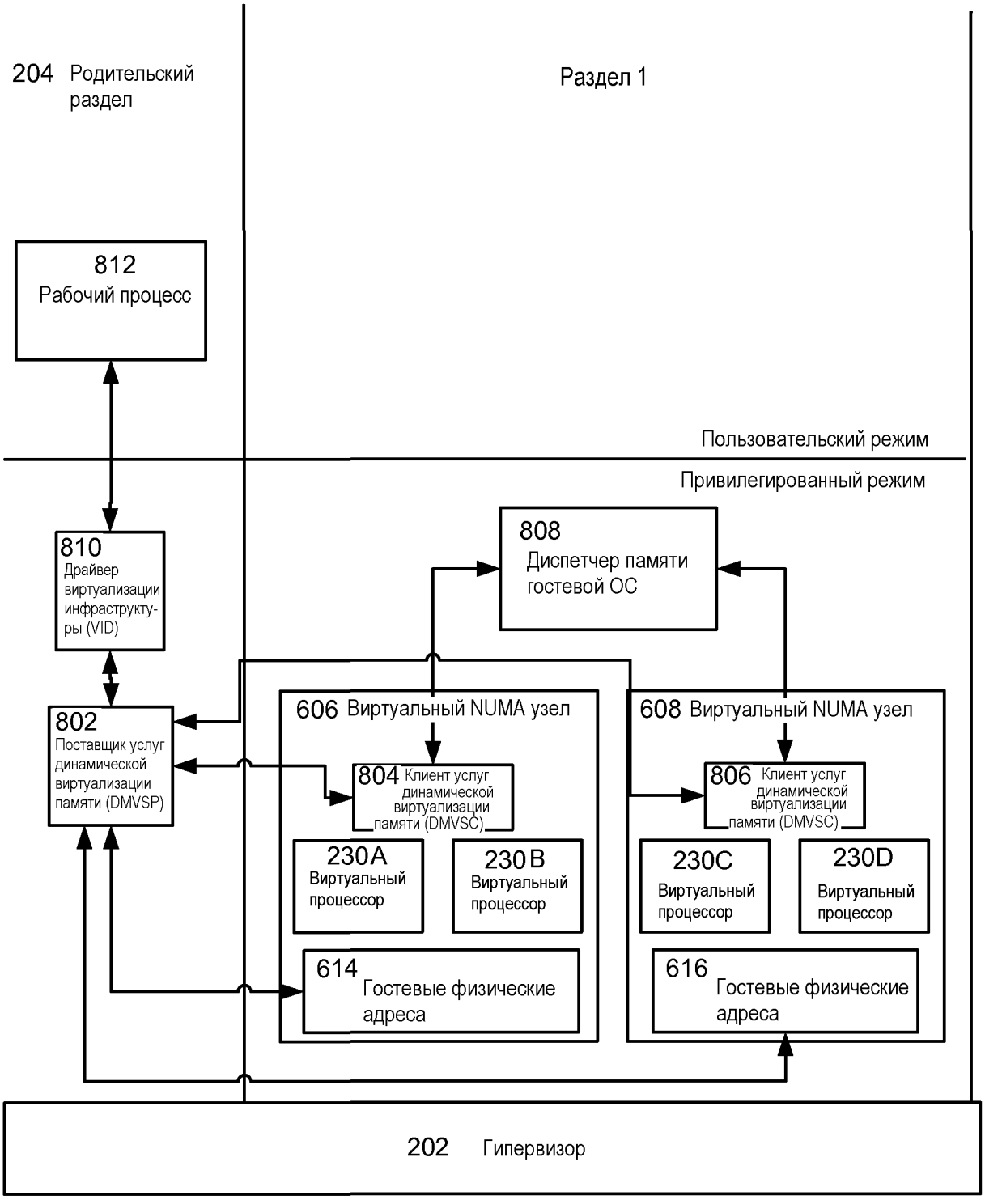
(54) ВИРТУАЛЬНАЯ АРХИТЕКТУРА НЕОДНОРОДНОЙ ПАМЯТИ ДЛЯ ВИРТУАЛЬНЫХ МАШИН

(57) Реферат:

Изобретение относится к области технологии виртуализации. Техническим результатом является повышение эффективности управления памятью для виртуальных машин. Способ управления памятью для виртуальных машин содержит этапы, на которых принимают запрос создания экземпляра виртуальной машины, причем запрос включает в себя характеристику виртуальной машины; выбирают топологию виртуальных узлов архитектуры неоднородной памяти (NUMA) для виртуальной машины на основе этой характеристики, причем топология виртуальных NUMA узлов включает в себя множество виртуальных NUMA узлов; создают экземпляр виртуальной машины в вычислительной системе, причем виртуальная машина включает в себя множество виртуальных NUMA узлов; и регулируют, на основе, по меньшей мере частично, нагрузки на память в конкретном виртуальном NUMA узле из множества виртуальных NUMA узлов, объем гостевой памяти, назначенной этому конкретному виртуальному NUMA узлу, на величину, равную второму объему гостевой памяти, причем вторым объемом гостевой памяти осуществляется обмен между упомянутым конкретным виртуальным

RU 2569805 C2

NUMA узлом и вторым виртуальным NUMA узлом из множества виртуальных NUMA узлов на основе, по меньшей мере частично, нагрузки на память во втором виртуальном NUMA узле. 3 н. и 16 з.п. ф-лы, 17 ил.



Фиг. 8

RU 2569805 C2
RU 2569805 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY

(19) **RU** (11) **2 569 805**(13) **C2**

(51) Int. Cl.
G06F 12/02 (2006.01)

(12) ABSTRACT OF INVENTION

(21)(22) Application: **2011139151/08, 19.03.2010**

(24) Effective date for property rights:
19.03.2010

Priority:
(30) Convention priority:
26.03.2009 US 12/412,272

(43) Application published: **27.03.2013** Bull. № **9**

(45) Date of publication: **27.11.2015** Bull. № **33**

(85) Commencement of national phase: **23.09.2011**

(86) PCT application:
US 2010/028034 (19.03.2010)

(87) PCT publication:
WO 2010/111149 (30.09.2010)

Mail address:
129090, Moskva, ul. B. Spasskaja, 25, stroenie 3, OOO "Juridicheskaja firma Gorodisskij i Partnery"

(72) Inventor(s):
OShINS Dzhejkob (US)

(73) Proprietor(s):
MAJKROSOFT TEKNOLODZhI LAJSENSING, EhlEhlSi (US)

RU 2569805 C2

(54) VIRTUAL NON-UNIFORM MEMORY ARCHITECTURE FOR VIRTUAL MACHINES

(57) Abstract:

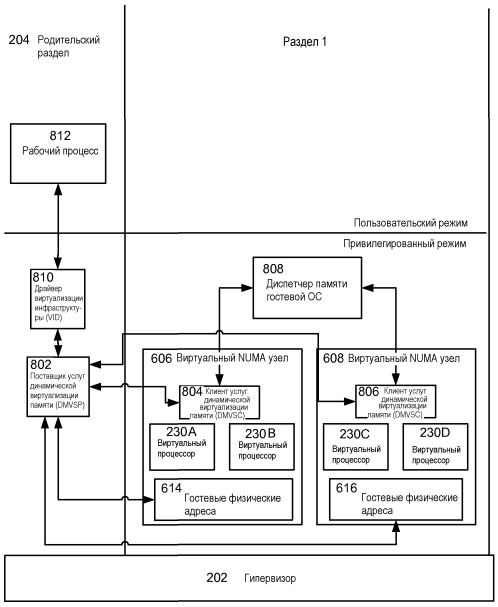
FIELD: physics, computer engineering.

SUBSTANCE: invention relates to virtualisation engineering. A method of managing memory for virtual machines comprises steps of receiving a request to instantiate a virtual machine, the request including a characteristic for the virtual machine; selecting a virtual non-uniform memory architecture (NUMA) node topology for the virtual machine based on the characteristic, the virtual NUMA node topology including a plurality of virtual NUMA nodes; instantiating the virtual machine in a computer system, the virtual machine including the plurality of virtual NUMA nodes; and adjusting, based at least in part on memory load in a specific virtual NUMA node of the plurality of virtual NUMA nodes, an amount of guest memory assigned to the specific virtual NUMA node, by a value equal to a second amount of guest memory, the second amount of guest memory enabling exchange between said specific virtual NUMA node and a second virtual NUMA node of the plurality of virtual NUMA nodes based at least in part on memory load in the second virtual NUMA node.

EFFECT: high efficiency of managing memory for virtual machines.

19 cl, 17 dwg

RU 2569805 C2



Фиг. 8

RU 2569805 C2

RU 2569805 C2

ПРЕДШЕСТВУЮЩИЙ УРОВЕНЬ ТЕХНИКИ

Технология виртуализации предусматривает разделение аппаратных ресурсов между многими разделами, каждый из которых может осуществлять хостинг для гостевой операционной системы. В целом, технология виртуальной машины может использоваться для объединения серверов и повышения их переносимости. Поскольку виртуальные машины становятся крупнее, и их рабочие нагрузки увеличиваются, способность легко осуществлять их объединение и/или перемещение с одной вычислительной системы на другую становится более трудной. Соответственно, являются желательными способы, предназначенные для повышения способности осуществлять объединение и/или перемещение более крупных виртуальных машин.

СУЩНОСТЬ ИЗОБРЕТЕНИЯ

Примерный вариант осуществления настоящего раскрытия описывает способ. В этом примере способ включает в себя, но не ограничивается указанным, прием запроса на создание экземпляра виртуальной машины, запрос включает в себя характеристику виртуальной машины; выбор для виртуальной машины топологии виртуального NUMA узла на основании характеристики, причем топология виртуального NUMA узла включает в себя множество виртуальных NUMA узлов; создание экземпляра виртуальной машины на вычислительной системе, причем виртуальная машина включает в себя множество виртуальных NUMA узлов; и подстройку, на основании нагрузки на память в конкретном виртуальном NUMA узле из множества, объема гостевой памяти, назначенной конкретному виртуальному NUMA узлу. В дополнение к вышеизложенному, другие аспекты описываются в пунктах формулы изобретения, на чертежах и тексте, образующем часть настоящего раскрытия.

Примерный вариант осуществления настоящего раскрытия описывает способ. В этом примере, способ включает в себя, но без ограничения указанным, исполнение виртуальной машины, причем виртуальная машина имеет топологию, которая включает в себя множество виртуальных NUMA узлов, при этом топология виртуальной машины формируется независимо от физической топологии вычислительной системы; определение нагрузки на память в каждом виртуальном NUMA узле из множества; и подстройку, на основании нагрузки на память в каждом виртуальном NUMA узле множества, гостевой памяти, назначенной, по меньшей мере, одному виртуальному NUMA узлу множества. В дополнение к вышеизложенному, другие аспекты описываются в пунктах формулы изобретения, на чертежах и в тексте, образующем часть настоящего раскрытия.

Примерный вариант осуществления настоящего раскрытия описывает способ. В этом примере, способ включает в себя, но без ограничения указанным, исполнение первой виртуальной машины, причем виртуальная машина имеет топологию, которая включает в себя множество виртуальных NUMA узлов, каждый виртуальный NUMA узел множества включает в себя виртуальный процессор и гостевые физические адреса, при этом топология виртуальной машины формируется независимо от физической топологии вычислительной системы; и добавление дополнительного виртуального процессора к виртуальному NUMA узлу множества. В дополнение к вышеизложенному, другие аспекты описаны в пунктах формулы изобретения, на чертежах и в тексте, образующем часть настоящего раскрытия.

Специалист в данной области техники может оценить, что один или несколько различных аспектов раскрытия могут включать в себя, без ограничения указанным, схемное и/или программное решение для осуществления упомянутых в документе аспектов настоящего раскрытия; схемное и/или программное решение могут быть

фактически любой комбинацией аппаратных средств, программного обеспечения и/или микропрограммного обеспечения, конфигурированной для осуществления упомянутых в документе аспектов в зависимости от проектных решений разработчика системы.

Вышеизложенное является кратким описанием и таким образом содержит, по необходимости, упрощения, обобщения и опущения подробностей. Специалисты в данной области техники оценят, что краткое описание является лишь иллюстративным и не подразумевается, что является ограничительным каким-либо образом.

КРАТКОЕ ОПИСАНИЕ ЧЕРТЕЖЕЙ

Фиг.1 - изображение примерной вычислительной системы, в которой могут быть реализованы аспекты настоящего раскрытия.

Фиг.2 - изображение операционной среды для применения на практике аспектов настоящего раскрытия.

Фиг.3 - изображение операционной среды для применения аспектов настоящего раскрытия.

Фиг.4 - изображение организации памяти в вариантах осуществления настоящего раскрытия.

Фиг.5 - изображение примерной операционной среды, применяющей аспекты настоящего раскрытия.

Фиг.6 - изображение примерной операционной среды, применяющей аспекты настоящего раскрытия.

Фиг.7 - изображение примерной операционной среды, применяющей аспекты настоящего раскрытия.

Фиг.8 - изображение примерной блок-схемы, иллюстрирующей аспекты настоящего раскрытия.

Фиг.9 - изображение последовательности операций для применения аспектов настоящего раскрытия.

Фиг.10 - изображение альтернативного варианта осуществления для последовательности операций по Фиг.9.

Фиг.11 - изображение последовательности операций для применения аспектов настоящего раскрытия.

Фиг.12 - изображение альтернативного варианта осуществления для последовательности операций по Фиг.11.

Фиг.13 - изображение альтернативного варианта осуществления для последовательности операций по Фиг.12.

Фиг.14 - изображение альтернативного варианта осуществления для последовательности операций по Фиг.12.

Фиг.15 - изображение последовательности операций для применения аспектов настоящего раскрытия.

Фиг.16 - изображение альтернативного варианта осуществления для последовательности операций по Фиг.15.

Фиг.17 - изображение альтернативного варианта осуществления для последовательности операций по Фиг.16.

ПОДРОБНОЕ ОПИСАНИЕ ИЗОБРЕТЕНИЯ

Варианты осуществления могут исполняться на одном или нескольких компьютерах. Фиг.1 и последующее обсуждение предназначены для обеспечения краткого общего описания подходящей вычислительной среды, в которой может быть реализовано раскрытие. Специалист в данной области техники может оценить, что вычислительная

система по Фиг.1 может в некоторых исполнениях реализовывать вычислительные системы 200, 300, 600, и 700. В этих примерных вариантах осуществления вычислительные системы могут включать в себя некоторые или все компоненты, описанные на Фиг.1, и схему в конфигурации с возможностью реализации аспектов настоящего раскрытия.

Термин «схема», используемый по всему раскрытию, может заключать в себе аппаратные компоненты, такие как контроллеры аппаратных прерываний, накопители на жестких дисках, сетевые адаптеры, графические процессоры, аппаратно реализованные кодеки видео/аудио, и микропрограммное/программное обеспечение, используемое для оперирования такими аппаратными средствами. В тех же или других вариантах осуществления термин «схема» может заключать в себе микропроцессоры, конфигурированные для выполнения функции(й) посредством микропрограммного обеспечения или коммутаторов, установленных некоторым образом. В тех же или других примерных вариантах осуществления термин «схема» может заключать в себе один или несколько логических процессоров, например, одно или несколько процессорных ядер (базовых средств) многоядерного универсального устройства обработки. Логический процессор(ы) в этом примере может быть конфигурирован посредством программных команд, реализующих логику, выполненную с возможностью выполнять функцию(и), которые загружаются из машинной памяти, например, оперативного запоминающего устройства (ОЗУ, RAM), постоянного запоминающего устройства (ПЗУ, ROM), микропрограммного обеспечения и/или виртуальной памяти. В примерных вариантах осуществления, где схема включает в себя комбинацию аппаратных средств и программного обеспечения, разработчик может записать реализующий логику исходный код, который впоследствии компилируется в машиночитаемый код, который может обрабатываться логическим процессором. Поскольку специалист в данной области техники может оценить, что уровень техники эволюционировал к позиции, где существует небольшое различие между аппаратными средствами, программным обеспечением или комбинацией аппаратного/программного обеспечения, выбор «аппаратные средства против программного обеспечения», чтобы осуществлять функции, является лишь выбором проектного решения. Таким образом, поскольку специалист в данной области техники может оценить, что программно реализованный процесс может быть преобразован в эквивалентную аппаратную структуру, и аппаратная структура может быть непосредственно преобразована в эквивалентный программно реализованный процесс, выбор «аппаратная реализация против программной реализации» является одной из альтернатив проектного решения и оставляется разработчику.

Теперь с обращением на Фиг.1, на ней изображена иллюстративная универсальная вычислительная система. Универсальная вычислительная система может включать в себя обычный компьютер 20 или подобный, включающий в состав логический процессор 21, системную память 22 и системную шину 23, которая связывает различные компоненты системы, включая системную память, с логическим процессором 21. Системная шина 23 может быть любой из нескольких типов шинных архитектур, включая шину памяти или контроллер памяти, периферийную шину, и локальную шину, использующую любую из разнообразия шинных архитектур. Системная память может включать в себя постоянное ЗУ (ПЗУ) 24 и оперативное ЗУ (ОЗУ) 25. Базовая система 26 ввода/вывода (БИОС, BIOS), содержащая основные программы, которые содействует передаче информации между компонентами внутри компьютера 20, например, в течение запуска, хранится в ПЗУ 24. Компьютер 20 может дополнительно включать в состав накопитель 27 на жестком диске для считывания с жесткого диска и записи на него (не

показано), накопитель 28 на магнитном диске для считывания со съемного магнитного диска 29 или записи на него, и накопитель 30 на оптическом диске для считывания со съемного оптического диска 31, такого как ПЗУ на компакт-диске (CD ROM), или записи на таковой, или другой оптический носитель. Накопитель 27 на жестком диске, накопитель 28 на магнитном диске и накопитель 30 на оптическом диске показаны соединенными с системной шиной 23 интерфейсом 32 накопителя на жестком диске, интерфейсом 33 накопителя на магнитном диске и интерфейсом 34 накопителя на оптическом диске, соответственно. Накопители и связанные с ними читаемые компьютером носители информации обеспечивают энергонезависимое ЗУ для читаемых компьютером команд, структур данных, программных модулей и других данных для компьютера 20. Хотя иллюстративная среда, описанная в документе, использует накопитель на жестком диске, съемный магнитный диск 29 и съемный оптический диск 31, специалисты в данной области техники должны оценить, что другие типы читаемых компьютером носителей информации, которые могут хранить данные, являющиеся доступными посредством компьютера, такие как магнитные кассеты, карты флэш-памяти, цифровые видеодиски, картриджи Бернулли, оперативные ЗУ (ОЗУ), постоянные ЗУ (ПЗУ) и т.п., также могут использоваться в иллюстративной среде. В целом, в некоторых вариантах осуществления, такие читаемые компьютером носители информации могут использоваться, чтобы хранить исполнимые процессором команды, реализующие аспекты настоящего раскрытия.

Ряд программных модулей может храниться на накопителе на жестком диске, магнитном диске 29, оптическом диске 31, в ПЗУ 24 или ОЗУ 25, включая операционную систему 35, одну или несколько прикладных программ 36, другие программные модули 37 и программные данные 38. Пользователь может вводить команды и информацию в компьютер 20 через устройства ввода данных, такие как клавиатура 40 и указательное устройство 42. Другие устройства ввода данных (не показано) могут включать в себя микрофон, джойстик, игровую панель, спутниковую антенну, сканер или подобное. Эти и другие устройства ввода данных обычно соединяются с логическим процессором 21 через интерфейс 46 последовательного порта, который связан с системной шиной, но могут соединяться посредством других интерфейсов, таких как параллельный порт, игровой порт или универсальная последовательная шина (USB). Дисплей 47 или другой тип устройства отображения также может соединяться с системной шиной 23 через интерфейс, такой как видеоадаптер 48. В дополнение к дисплею 47, компьютеры обычно включают в состав другие периферийные устройства вывода (не показано), такие как динамики и принтеры. Иллюстративная система по Фиг.1 также включает в состав контроллер 55, шину 56 интерфейса малых вычислительных систем (SCSI) и внешнее запоминающее устройство 62, подсоединенное к шине 56 SCSI.

Компьютер 20 может действовать в сетевой среде, используя логические соединения с одним или несколькими удаленными компьютерами, например, удаленным компьютером 49. Удаленный компьютер 49 может быть другим компьютером, сервером, маршрутизатором, сетевым ПК, одноранговым устройством или другим обычным сетевым узлом, и обычно может включать в себя многие или все компоненты, описанные выше относительно компьютера 20, хотя на Фиг.1 было показано только запоминающее устройство 50. Логические соединения, изображенные на Фиг.1, могут включать в себя локальную вычислительную сеть (LAN) 51 и глобальную вычислительную сеть (WAN) 52. Такие сетевые среды являются обычными в учреждениях, вычислительных сетях масштаба предприятия, внутрикорпоративных локальных сетях и сети Интернет.

При использовании в сетевой среде LAN компьютер 20 может быть соединен с

локальной сетью 51 через сетевой интерфейс или адаптер 53. При использовании в сетевой среде WAN компьютер 20 обычно может включать в состав модем 54 или другое средство для установления связи по сети WAN 52, такой как сеть Интернет. Модем 54, который может быть внутренним или внешним, может быть соединен с системной шиной 23 через интерфейс 46 последовательного порта. В сетевой среде программные модули, изображенные относящимися к персональному компьютеру 20, или их части, могут храниться на удаленном запоминающем устройстве. Будет оценено, что показанные сетевые соединения являются иллюстративными и могут использоваться другие средства установления линии связи между компьютерами. Кроме того, хотя предполагается, что многочисленные варианты осуществления настоящего раскрытия являются особенно хорошо подходящими для компьютеризированных систем, в этом документе отсутствует что-либо, предназначенное для ограничения раскрытия такими вариантами осуществления.

Теперь, что касается Фиг.2 и 3, на них изображены высокоуровневые блок-схемы вычислительных систем. Как показано фигурой чертежа, вычислительная система 200 может включать в состав физические аппаратные устройства, такие как устройства 208 хранения информации, например, накопитель на жестком диске, сетевую интерфейсную плату (NIC) 210, графическую плату 234, по меньшей мере, один логический процессор 212, и оперативную память (ОЗУ) 214. Вычислительная система 200 может также включать в состав компоненты такие же, как компьютер 20 по Фиг.1. Хотя проиллюстрирован один логический процессор, в других вариантах осуществления компьютерная система 200 может иметь множество логических процессоров, например, множество исполнительных ядер (процессора) на один процессор и/или множество процессоров, каждый из которых может иметь множество исполнительных ядер. Продолжая описание Фиг.2, изображен гипервизор 202, который в области техники может также называться «монитор виртуальной машины». Гипервизор 202 в изображенном варианте осуществления включает в себя исполнимые команды, предназначенные для управления и арбитража доступа к аппаратным средствам вычислительной системы 200. В общих чертах, гипервизор 202 может формировать среды исполнения (программ), называемые разделами, такие как дочерний раздел 1 - дочерний раздел N (где N - целое число больше 1). В вариантах осуществления дочерний раздел может рассматриваться базовой единицей изолированности, поддерживаемой гипервизором 202, то есть, каждый дочерний раздел может быть отображен на множество аппаратных ресурсов, например, память, устройства, такты логического процессора, и т.д., который находится под управлением гипервизора 202 и/или родительского раздела. В вариантах осуществления гипервизором 202 может быть автономный программный продукт, часть операционной системы, встроенной в микропрограммное обеспечение системной («материнской») платы, специализированные интегральные схемы или комбинация такового.

В изображенном примере вычислительная система 200 включает в себя родительский раздел 204, который также можно рассматривать как домен 0 в сообществе (разработчиков) систем с открытым исходным кодом. Родительский раздел 204 может конфигурироваться для предоставления ресурсов гостевым операционным системам, исполняющимся в дочерних разделах 1-N, путем использования поставщиков 228 услуг виртуализации (VSP), которые также известны как драйверы серверной части в сообществе использующих открытый исходный код. В этой примерной архитектуре родительский раздел 204 может управлять пропуском (являться шлюзом) доступа к базовому аппаратному обеспечению. В общих чертах, VSP 228 могут использоваться,

чтобы мультиплексировать интерфейсы к аппаратным ресурсам при посредстве клиентов услуг виртуализации (VSC), которые также известны как драйверы клиентской (передней) части в сообществе использующих открытый исходный код. Каждый дочерний раздел может включать в себя один или несколько виртуальных процессоров, например, виртуальные процессоры 230-232, так что гостевые операционные системы 220-222 могут управлять и планировать потоки для исполнения на них. В целом, виртуальные процессоры 230-232 являются исполнимыми командами и связанной с ними информацией состояния, которые обеспечивают представление физического процессора с конкретной архитектурой. Например, одна виртуальная машина может содержать виртуальный процессор, имеющий характеристики процессора Intel x86, тогда как другой виртуальный процессор может иметь характеристики процессора PowerPC. Виртуальные процессоры в этом примере могут отображаться на логические процессоры вычислительной системы так, что команды, которые выполняют виртуальные процессоры, будут поддержаны логическими процессорами. Таким образом, в этих примерных вариантах осуществления, многие виртуальные процессоры могут быть исполняющимися одновременно в то время, как, например, другой логический процессор исполняет команды гипервизора. В сущности говоря, и как проиллюстрировано фигурой чертежа, комбинацию виртуальных процессоров, различных VSC и памяти в разделе может рассматривать виртуальной машиной, такой как виртуальная машина 240 или 242.

В целом, гостевые операционные системы 220-222 могут включать в себя любую операционную систему, такую как, например, операционные системы от компаний Microsoft®, Apple®, сообщество использующих открытый исходный код, и т.д. Гостевые операционные системы могут включать в себя пользовательский/привилегированный режимы работы и могут иметь ядра операционной системы, которые могут включать в себя планировщики, диспетчеры памяти, и т.д. Каждая гостевая операционная система 220-222 может содержать связанные с ней файловые системы, которые могут содержать хранимые на них приложения, такие как серверы электронной коммерции, серверы электронной почты, и т.д., и гостевые операционные системы непосредственно. Гостевые операционные системы 220-222 могут планировать потоки для исполнения на виртуальных процессорах 230-232, и экземпляры таких приложений могут приводиться в исполнение.

Теперь с обращением на Фиг.3 иллюстрируется альтернативная архитектура, которая может использоваться. На Фиг.3 изображены компоненты, подобные таковым на Фиг.2, однако в этом примерном варианте осуществления, гипервизор 202 может включать в себя поставщики 228 услуг виртуализации и драйверы 224 устройств, и родительский раздел 204 может содержать вспомогательные программы 236 задания конфигурации. В этой архитектуре гипервизор 202 может выполнять такие же или сходные функции, как гипервизор 202 по Фиг.2. Гипервизор 202 по Фиг.3 может быть автономным программным продуктом, частью операционной системы, встроенной в микропрограммное обеспечение системной платы, или блок гипервизора 202 может быть реализован специализированными интегральными схемами. В этом примере родительский раздел 204 может содержать команды, которые могут использоваться, чтобы задавать конфигурацию гипервизора 202, однако запросы доступа к аппаратному обеспечению могут обрабатываться гипервизором 202 вместо того, чтобы пересылаться на родительский раздел 204.

Теперь с обращением к Фиг.4 иллюстрируется организация памяти в вариантах осуществления, которые включают в себя виртуальные машины. Например, вычислительная система, такая как вычислительная система 200, может иметь ОЗУ 214,

имеющее адреса памяти. Вместо предоставления отчета о системных физических адресах памяти на виртуальные машины, гипервизор 202 может представлять различные адреса для системных физических адресов, например, гостевые физические адреса (GPA), на диспетчеры памяти гостевых операционных систем. Гостевые операционные системы затем могут манипулировать гостевыми физическими адресами, и гипервизор 202 поддерживает взаимосвязи согласно адресам GPA и SPA. Как показано фигурой чертежа, в варианте осуществления адресы GPA и SPA могут быть организованы в виде блоков (данных) в памяти. В общих чертах, блок памяти может включать в себя одну или несколько страниц памяти. Взаимосвязи между адресами GPA и SPA могут поддерживаться таблицей теневых страниц, например, таковой, описанной в принадлежащей тому же правообладателю заявке на патент США за номером № 11/128665, озаглавленной "Enhanced Shadow Page Table Algorithms" (Усовершенствованные алгоритмы таблицы теневых страниц), содержание которой тем самым во всей полноте включено в документ путем ссылки. В действии, если гостевая операционная система хранит данные в виде адреса GPA в блоке 1, данные фактически могут храниться в другом SPA, таком как блок 6, в системе.

Вкратце на Фиг.5 изображена операционная среда для применения аспектов настоящего раскрытия. Например, некоторое число вычислительных систем 504-510 могут быть связаны вместе в центр 500 хранения и обработки данных 500 (дата-центр) (хотя изображены четыре вычислительные системы, специалист в данной области техники может оценить, что дата-центр 500 может включать в себя больше или меньше вычислительных систем). Изображенные вычислительные системы могут иметь различные топологии, и, кроме того, они могут иметь различные характеристики, например, различные объемы ОЗУ, различное быстродействие ОЗУ, различное количество логических процессоров и/или логические процессоры с различным быстродействием.

Система 502 управления может содержать компоненты, подобные таковым вычислительной системы 20 по Фиг.1, и/или вычислительных систем 200, 300, 600 или 700. То есть, в одном варианте осуществления система 502 управления может быть вычислительной системой, которая заключает в себе объект изобретения, описанный ниже по отношению к Фиг.6 или 7.

Продолжая общий краткий обзор чертежей, на Фиг.6 изображена вычислительная система 600, имеющая топологию симметричной многопроцессорной системы (SMP) или 'плоскую' топологию. В целом, SMP является архитектурой вычислительной системы, которая включает в состав множество процессоров, которые соединены с одной совместно используемой памятью. В этой организации, контроллер 602 памяти может управлять потоком данных в память и из нее. Доступ к памяти может быть однородным по отношению к каждому логическому процессору 212A-F, и каждый логический процессор может осуществлять доступ к полному диапазону (адресов) памяти, то есть, системным физическим адресам 622-632. Эта топология работает хорошо для вычислительных систем с относительно небольшим числом процессоров, но если вычислительная система включает много процессоров, все конкурирующие за доступ к шине совместно используемой памяти, рабочая характеристика системы может снижаться. Кроме того, сложность вычислительной системы значительно увеличивается, каковое в свою очередь повышает стоимость на один процессор.

Вкратце, вычислительная система 600 может включать в состав такие же или подобные компоненты, как компьютер 200 или 300. Как показано фигурой чертежа, вычислительная система 600 может содержать множество логических процессоров

212A-212F (хотя изображены шесть логических процессоров, вычислительная система может иметь больше или меньше), связанных вместе через контроллер 602 памяти, который управляет пропуском доступа к ОЗУ 214. Подобно описанному выше, каждый логический процессор 212A-212F может иметь различные характеристики, например, тактовые частоты, размер кэша, и т.д. В этой организации контроллер 602 памяти может управлять потоком данных в ОЗУ 214 и из него.

Может быть создан экземпляр гипервизора 202, и он может управлять аппаратными средствами вычислительной системы 600. Гипервизор 202 может управлять одной или несколькими виртуальными машинами 240-242, каждая из которых может содержать виртуальные NUMA узлы, такие как виртуальные NUMA узлы 606-612. Виртуальные NUMA узлы 606-612 могут использоваться для определения организации ресурсов виртуальной машины путем предоставления отчета о виртуальных топологиях на гостевые приложения или гостевые операционные системы, такие как гостевые операционные системы 220 и 222. Как показано фигурой чертежа, каждый виртуальный NUMA узел 606-612 может содержать один или несколько виртуальных процессоров 230A-D, 232A-D и гостевые физические адреса 614-616 и 618-620. В целом, гипервизор 202 может поддержать каждый виртуальный NUMA узел 606-612 с помощью одного или нескольких логических процессоров и системных физических адресов из ОЗУ 214. То есть, гипервизор 202 может задавать один или несколько логических процессоров в качестве «идеальных» процессоров, которые могут использоваться, чтобы исполнять потоки виртуальных процессоров.

Вкратце, на Фиг.7 изображена вычислительная система 700 с топологией, включающей в себя NUMA узлы 702-706. Вычислительные системы с NUMA узлами в целом могут рассматриваться как компьютеры, которые составлены из меньших вычислительных систем. В этом примере, каждый NUMA узел 606-612 может включать в себя один или несколько логических процессоров и локальную память. Память внутри NUMA узла считается локальной памятью, и память в других NUMA узлах считается удаленной памятью, поскольку только процессоры внутри узла соединены с одной и той же шиной памяти. NUMA узлы внутренне соединяются кэш-когерентными на уровне домена межсоединениями, которые дают возможность процессорам в одном NUMA узле осуществлять доступ к памяти в других NUMA узлах когерентным образом. Тем самым, системные физические адреса 622-632 являются однородными по отношению к каждому процессору. Или другими словами, системный физический адрес 20000 является одинаковым для каждого процессора в вычислительной системе. Различие состоит в том, что для некоторых процессоров адрес памяти, соответствующий 20000, является адресом локальной памяти, например, внутри их NUMA узла, а для других процессоров адрес памяти 20000 является удаленным, например, вне их NUMA узла. В целом, к локальной памяти можно осуществлять доступ быстрее, чем к удаленной памяти, и соотношение между временем «локальный против удаленного» доступа называется отношением NUMA. Отношение NUMA от 1 до 2 означает, что требуется в два раза больше тактов процессора, чтобы осуществить доступ к конкретному удаленному системному физическому адресу, чем к локальному системному физическому адресу. NUMA смягчает узкие места, обусловленные системами SMP, путем ограничения числа процессоров на одну любую шину памяти, и в целом является менее дорогим, чем вычислительная система SMP с тем же количеством логических процессоров.

Вычислительная система 700 может включать в себя такие же или подобные компоненты, что и компьютер 200 или 300. Как показано фигурой чертежа, в этой операционной среде вычислительная система 700 включает в себя три NUMA узла 702-

706 (хотя компьютер может иметь больше или меньше), соединенных межсоединениями 708. Как проиллюстрировано фигурой чертежа, число процессоров в рамках каждого NUMA узла может быть переменным, и каждый узел может иметь свое собственное ОЗУ.

Подобно Фиг.7, гипервизор 202 может управлять аппаратным обеспечением вычислительной системы 700. Когда загружаются гостевые операционные системы или цельные приложения, они могут выявлять топологии виртуальных машин 240 и 242, подобных описанным выше. Каждому виртуальному NUMA узлу 606-612 может быть назначен один или несколько идеальных процессоров и системные физические адреса из того же NUMA узла, который может использоваться для исполнения потоков виртуального процессора.

Хотя вычислительные системы 600 и 700 изображены включающими в состав две виртуальные машины 240 и 242, в других вариантах осуществления они могут исполнять больше или меньше виртуальных машин. Кроме того, хотя каждая виртуальная машина изображена с наличием двух виртуальных NUMA узлов, в других вариантах осуществления, виртуальные машины могут иметь больше или меньше виртуальных NUMA узлов. Также, хотя виртуальные NUMA узлы изображены имеющими два виртуальных процессора, в других вариантах осуществления виртуальные NUMA узлы могут содержать больше или меньше виртуальных процессоров. Кроме того, каждый виртуальный NUMA узел может иметь топологию, отличную от других виртуальных NUMA узлов, например, один виртуальный NUMA узел может содержать 4 виртуальных процессора и ОЗУ в 8 гигабайтов, тогда как другой виртуальный NUMA узел может содержать 2 виртуальных процессора и ОЗУ в 4 гигабайта.

На Фиг.8 изображена блок-схема среды, которая может использоваться в аспектах настоящего раскрытия. Как показано фигурой чертежа, иллюстрируется компонент, управляющий памятью, назначенной виртуальной машине, который может быть известным как поставщик 802 услуг динамической виртуализации памяти (DMVSP), и может использоваться, чтобы подстраивать объем памяти, доступной для виртуального NUMA узла. Как показано фигурой чертежа, DMVSP 802 может связываться с одним или несколькими драйверами динамического изменения выделяемой памяти (ballooning), которые могут быть известны как клиенты услуг виртуализации, а с клиентами 804 и/или 806 услуг динамической виртуализации памяти (DMVSC) (хотя изображен один DMVSC на каждый виртуальный NUMA узел, в других вариантах осуществления может использоваться один DMVSC на один раздел). В общих чертах, клиенты DMVSC 804 и/или 806 могут обеспечивать информацию, которая может использоваться DMVSP 802, чтобы подстраивать память виртуальных NUMA узлов, и каждый DMVSC может также помогать осуществлять передачу и возврат памяти от виртуального NUMA узла, с которым он связан. Клиенты DMVSC 804, 806, и DMVSP 802 могут обмениваться информацией посредством шины виртуализации, описанной в заявке на патент США за номером №11/128647, озаглавленной "Partition Bus" (Шина раздела), содержание которой включено в документ путем ссылки во всей полноте. Кроме того, дополнительные аспекты DMVSC-клиентов и DMVSP-поставщиков описаны в заявке на патент США за номером № 12/345469, озаглавленной "Dynamic Virtual Machine Memory Management" (Динамическое управление памятью виртуальной машины), содержание которой включено в документ путем ссылки во всей полноте.

Продолжая описание Фиг.8, система может включать в себя изображенный рабочий процесс 812 («исполнитель»), который может управлять дочерним разделом(ами). Рабочий процесс 812 может работать вместе с драйвером 810 виртуализации

инфраструктуры (VID), который может распределять память дочернего раздела. Например, VID 810 может устанавливать и удалять взаимосвязи между гостевыми физическими адресами и системными физическими адресами. На Фиг.8 также изображен раздел, которая может включать в себя гостевую операционную систему, такую как гостевая операционная система 220, которая может включать в себя диспетчер 808 памяти. В целом, диспетчер 808 памяти может распределять память приложениям по их запросу и освобождать память, когда она больше не требуется приложениям.

Нижеследующее является рядом блок-схем, изображающих реализации процессов. Для простоты понимания блок-схемы организованы так, что начальные блок-схемы представляют реализацию с точки зрения полной "общей картины", и последующие блок-схемы обеспечивают дальнейшие дополнения и/или подробности. Кроме того, специалист в данной области техники может оценить, что последовательность операций, изображенная пунктирными линиями, рассматривается необязательной.

Возвращаясь теперь к Фиг.9, на ней изображена последовательность операций для применения аспектов настоящего раскрытия, включающая операции 900-910. Последовательность операций начинается операцией 900, и операция 902 иллюстрирует прием запроса на создание экземпляра виртуальной машины, запрос включает характеристику виртуальной машины. Например, и со ссылкой на Фиг.6 или 7, гипервизор 202 может принимать запрос на создание виртуальной машины, такой как виртуальная машина 240. Например, запрос может приниматься от системы 502 управления, родительского раздела 204 по Фиг.2 или 3, и т.д. Запрос может быть запросом на новую виртуальную машину, или он может быть запросом на экземпляр предварительно сохраненной виртуальной машины. Если виртуальной машиной 240 является новая виртуальная машина, характеристики виртуальной машины, например, объем оперативной памяти, назначенной виртуальной машине, число виртуальных процессоров, или какой тип устройств ввода/вывода должна иметь виртуальная машина, могут быть заданы, например, администратором.

Продолжая описание Фиг.9, операция 904 показывает выбор топологии виртуального NUMA узла для виртуальной машины на основании характеристики, топология виртуального NUMA узла включает в себя множество виртуальных NUMA узлов. Например, процесс в родительском разделе 204 (и/или гипервизоре 202) может определять топологию для виртуальной машины 240 на основании принятой характеристики. Например, родительский раздел 204 может включать информацию, которая идентифицирует заданный по умолчанию размер для виртуального NUMA узла, такого как виртуальный NUMA узел 606. Процесс в родительском разделе 204 может использовать информацию, которая описывает заданный по умолчанию размер и требуемую характеристику, чтобы определить число виртуальных NUMA узлов для виртуальной машины 240. В конкретном примере, требуемой характеристикой может быть 6-процессорная виртуальная машина с ОЗУ в 10 гигабайтов. Если заданный по умолчанию размер виртуального NUMA узла включает 2 виртуальных процессора и ОЗУ в 4 гигабайта, то система 502 управления может сформировать конфигурационный файл, который указывает, что виртуальная машина 240 будет включать в себя 3 виртуальных NUMA узла.

В варианте осуществления заданный по умолчанию размер виртуального NUMA узла может задаваться администратором или системой 502 управления. Возвращаясь к Фиг.5, система 502 управления может исполнять одну или несколько программ, которые могут получать информацию, которая идентифицирует физическую топологию вычислительных систем 504-510, в дата-центре 500, например, информацию, которая

идентифицирует, сколько NUMA узлов (если таковые есть) содержит каждая вычислительная система 504-510, насколько велико ОЗУ, которое содержит каждая вычислительная система 504-510, быстродействие ОЗУ, как организовано ОЗУ, быстродействие процессоров, сколько процессорных ядер содержит каждый процессор, и т.д.

В целом, размер виртуального NUMA узла влияет на работу виртуальных машин в дата-центре 500. Например, если размер виртуального NUMA узла увеличивается, например, по памяти и/или процессорам, переносимость виртуального NUMA узла снижается. Или другими словами, «большие» виртуальные NUMA узлы могут затруднять перемещение виртуальной машины. Это происходит потому, что виртуальный NUMA узел должен назначаться или NUMA узлу, или вычислительной системе, которая имеет достаточно 'плоских' ресурсов, чтобы осуществлять виртуальный NUMA узел. Если, например, виртуальный NUMA узел является слишком большим, например, он содержит слишком большое ОЗУ или слишком много виртуальных процессоров, он не будет способным вмещаться в меньшие NUMA узлы в дата-центре 500, таким образом, ограничивая способность перемещения виртуальной машины. Кроме того, если более крупный виртуальный NUMA узел просто назначается на множество меньших NUMA узлов, рабочая характеристика виртуальной машины снизится вследствие различия, которое имеется между временами доступа к локальной памяти и удаленной памяти.

С другой стороны, если размер виртуального NUMA узла уменьшается, может оказываться неблагоприятное воздействие на рабочую характеристику гостевой операционной системы. Эта неэффективность может происходить, поскольку гостевая операционная система будет пытаться рассредоточить приложения, и свое собственное исполнение на отдельные виртуальные NUMA узлы. Гостевая операционная система будет связана ограничениями в этом случае, и рабочая характеристика снизится.

Соответственно, в системе 502 управления по варианту осуществления можно достигать баланса между переносимостью и эффективностью путем определения оптимального размера виртуального NUMA узла для дата-центра 500. Например, в варианте осуществления логический процессор системы 502 управления может исполнять программу и определять средний размер NUMA узла в дата-центре, например, среднее число логических процессоров, средний объем ОЗУ и т.д., и задавать, чтобы размер виртуального NUMA узла был одинаковым или меньше среднего NUMA узла в системе. В другом варианте осуществления программа может быть конфигурирована, чтобы задавать размер виртуального NUMA узла немного меньше наименьшего NUMA узла в дата-центре 500. В варианте осуществления, размер виртуального NUMA узла может задаваться являющимся слегка меньше среднего размера или наименьшего размера с тем, чтобы, если вычислительная система становится тяжело распределенной (по ресурсам), более одного виртуального NUMA узла может быть назначено на один NUMA узел. В конкретном примере, если наименьший NUMA узел содержит 4 логических процессора и ОЗУ в 8 гигабайтов, то размер виртуального NUMA узла можно задать, например, в 2 виртуальных процессора и ОЗУ в 4 гигабайта.

Операция 906 показывает создание на вычислительной системе экземпляра виртуальной машины, включающей в состав множество виртуальных NUMA узлов. В варианте осуществления, гипервизор 202 может исполняться логическим процессором, и может создаваться экземпляр виртуальной машины, содержащей множество виртуальных NUMA узлов. Например, и со ссылкой на Фиг.6 и/или Фиг.7, виртуальная машина 240, содержащая виртуальные NUMA узлы 606-608, может осуществляться компьютерной системой 600 или 700. То есть, VID 810 может поддержать гостевые

физические адреса виртуальной машины 240 с помощью системных физических адресов из ОЗУ и виртуальные процессоры - с помощью одного или нескольких логических процессоров. Например, блоки гостевых физических адресов 614 могут быть поддержаны блоками системных физических адресов 622, и блоки гостевых физических адресов 616 могут быть поддержаны блоками системных физических адресов 624. Потоки гипервизора затем могут планироваться на логические процессоры, поддерживающие виртуальные процессоры, и могут исполняться команды, указывающие виртуальные процессоры. Как показано Фиг.6 и 7, топология каждой из виртуальных машин может создаваться независимо от топологии базового аппаратного обеспечения. То есть, топология каждой виртуальной машины является «развязанной» от базовой физической топологии вычислительной системы, которая приводит ее в исполнение.

В варианте осуществления, BIOS виртуальной машины или микропрограммное обеспечение начальной загрузки могут описывать топологию виртуальной машины, например, содержит ли она виртуальные NUMA узлы, размер любых виртуальных NUMA узлов, и NUMA отношения для виртуальных NUMA узлов, для гостевой операционной системы для цельного приложения. Структура данных может быть обработана и гостевой операционной системой (ОС, OS) 220 или приложением, и ее могут использовать ОС или приложение, чтобы воспользоваться преимуществом наличия виртуальных NUMA узлов. Например, гостевая операционная система 220 может пытаться «привязать» потоки приложения, которое не осведомлено о NUMA, к виртуальному NUMA узлу так, чтобы исполнение приложения оставалось локальным. В другом примере программа управления базами данных, например, SQL-сервер, может распределять блокировки локально на виртуальные NUMA узлы, и база данных может расщеплять запросы считывания/записи по всем виртуальным NUMA узлам. В еще одном примере гостевая операционная система 220 может создавать пулы страниц для каждого виртуального NUMA узла в виртуальной машине.

Продолжая описание Фиг.9, операция 908 показывает подстройку, на основании нагрузки на память в конкретном виртуальном NUMA узле в множестве, объема гостевой памяти, назначенной конкретному виртуальному NUMA узлу. Например, логический процессор, для примера, логический процессор 212A по Фиг.6 или 7, может исполнять команды, указывающие на DMVSP 802, и может подстраивать объем гостевых физических адресов, доступных виртуальному NUMA узлу, такому как виртуальный NUMA узел 606. То есть, DMVSP 802 может исполняться, и может осуществляться передача или возврат памяти на основании нагрузки, которую испытывают виртуальные NUMA узлы.

В варианте осуществления, нагрузка на память может идентифицировать, как на рабочую характеристику гостевой операционной системы 220 влияет объем памяти, который доступен каждому виртуальному NUMA узлу 606-608. Эта информация может вычисляться в течение времени выполнения гостевой операционной системы 220, например, одним или несколькими DMVSC, такими как DMVSC 804 и/или 806, и посылаться на DMVSP 802. Например, нагрузка на память может быть представлена последовательностью значений, которые могут идентифицировать различный уровень нагрузки на память в виртуальном NUMA узле. Если ресурсы в виртуальном NUMA узле становятся более нагруженными, то есть, если объем памяти, требуемый для эффективного исполнения текущей рабочей нагрузки на виртуальном NUMA узле увеличивается, DMVSC 804 может изменить значение и передать эту информацию на DMVSP 802.

В варианте осуществления, информацию нагрузки на память может вычислять

DMVSC 804, исходя из информации, принятой от гостевой операционной системы 220. Например, DMVSC 804 может быть конфигурирован для приема информации страничной организации операционной системы от диспетчера 808 памяти относительно виртуального NUMA узла 606. Мониторинг интенсивности страничного обмена гостевой операционной системы может осуществляться двумя счетчиками, которые предоставляются диспетчером 808 памяти и диспетчером кэша, а именно: интенсивность страничного обмена и интенсивность ротации кэша.

В том же или другом варианте осуществления DMVSC 804 может принимать от диспетчера 808 памяти уведомления о физической памяти, связанные с виртуальным NUMA узлом 606, и использовать эту информацию, чтобы вычислять нагрузку на память виртуального NUMA узла 606. Например, диспетчер 808 памяти может выводить уведомления о высоком объеме памяти и уведомления о низком объеме памяти, на основании активности в гостевой операционной системе 220, которая связана с виртуальным NUMA узлом 606. Диспетчер 808 памяти может запускать эти уведомления на основании пороговой величины низкого объема памяти (LMT) и пороговой величины высокого объема памяти (HMT). В конкретном примерном варианте осуществления, заданным по умолчанию уровнем доступной памяти, который сигнализирует событие уведомления «низкий ресурс памяти», может быть приблизительно 32 Мбайт в расчете на 4 Гбайта, до максимального в 64 Мбайт. Заданным по умолчанию уровнем, который сигнализирует событие уведомления «высокий ресурс памяти», может быть, например, трехкратное заданное по умолчанию значение низкого объема памяти. Промежуточные уровни доступности памяти между этими двумя могут быть определены делением интервала между уровнями пороговой величиной высокого объема памяти и пороговой величины низкого объема памяти. Специалист в данной области техники может оценить, что эти значения являются иллюстративными, и что могут выполняться изменения без выхода за рамки существа раскрытия.

Эти уведомления, наряду с другими, может использовать DMVSC 804, чтобы вычислять нагрузку на память виртуального NUMA узла 606. Например, каждый уровень может быть связан со значением, например, 0-4, и если принимаются во внимание какие-либо другие счетчики производительности, они также могут быть связаны со значениями. Значения для каждого счетчика производительности затем могут использоваться, чтобы вычислять текущую нагрузку на память виртуального NUMA узла 606. В конкретном примере, нагрузка на память может вычисляться путем использования более высоких или более низких значений для счетчика производительности. В другом примере среднее значение счетчиков производительности может использоваться в качестве нагрузки на память. В еще одном варианте осуществления может использоваться более сложный алгоритм, чтобы вычислять нагрузку на память, который учитывает предыдущую оценку счетчика производительности и назначает каждому счетчику производительности скалярное значение, чтобы влиять на его относительный вес в вычислении.

При принятии решения осуществлять передачу памяти, DMVSP 802 может использовать разнообразие способов, одним из которых является режим «горячего добавления». Например, некоторые операционные системы могут поддерживать «горячее добавление», которое дает возможность добавления диапазонов адресов физической памяти к исполняющейся операционной системе, не требуя перезагрузки системы. То есть, диспетчер 808 памяти может быть конфигурирован для поддержки динамического добавления памяти к исполняющейся системе. В варианте осуществления «горячего добавления» DMVSC 804 может конфигурироваться для доступа к интерфейсу

«горячего добавления» диспетчера 808 памяти, и DMVSC 804 может посылать сообщение на гостевую операционную систему 220, которое описывает «добавляемые в горячем режиме» GPA и с каким виртуальным NUMA узлом они связаны. Диспетчер 808 памяти может затем сделать новую память доступной для гостевой операционной системы 220, драйверов, приложений, или любых других процессов, которые исполняются на виртуальном NUMA узле 606. Например, DMVSC 804 может принимать «добавляемые в горячем режиме» адреса памяти от DMVSP 802 после того, как VID 810 формирует взаимосвязи между адресами GPA и SPA.

Подобным образом режим «горячего удаления» может использоваться, чтобы осуществлять возврат адресов памяти от виртуального NUMA узла, такого как виртуальный NUMA узел 606. Например, DMVSC 804 может посылать сообщение на гостевую операционную систему 220, которое указывает, что память была удалена в режиме «горячего удаления». DMVSC 804 может запросить, чтобы диспетчер 808 памяти обеспечил блоки адресов GPA от виртуального NUMA узла 606 для удаления. В этом примере DMVSC 804 затем может вызвать интерфейс прикладного программирования (API) удаления, обеспечиваемый диспетчером 808 памяти, и удалить адреса GPA из гостевой операционной системы 220. В варианте осуществления, где используется «горячее удаление», память, которая удаляется, не идет в зачет гостевой текущей передачи памяти, и диспетчер 808 памяти может подстраивать свои внутренние счетчики, используя способы, подобные используемым операционной системой для удаления памяти, которая физически удаляется из системной платы.

В другом варианте осуществления, память может быть возвращена виртуальному NUMA узлу с использованием способа динамического изменения выделяемой памяти. То есть, память может быть возвращена путем развязки гостевых физических адресов в виртуальном NUMA 606 узле от физических адресов, которые поддерживают их. Например, логический процессор 212B может исполнять команды, указывающие на DMVSC 804, и может послать на диспетчер 808 памяти сообщение, запрашивающее резервирование диспетчером 808 памяти некоторого объема памяти для использования посредством DMVSC 804, например, один или несколько блоков памяти. Диспетчер 808 памяти может наложить блокировку памяти для монопольного использования в рамках DMVSC 804, и DMVSC 804 может послать адреса GPA памяти на DMVSP 802. В этом примере DMVSP 802 может посылать адреса GPA на VID 810, и VID 810 может удалить в таблице теневых страниц статьи отображения этих адресов GPA на адреса SPA. В этом примере, диспетчер 808 памяти может включать информацию, которая идентифицирует, что адреса GPA все еще действительны, однако фактически адреса GPA более не поддержаны системными физическими адресами. В этом примере диспетчер 808 памяти не будет использовать блокированные адреса GPA, и поддерживающие их адреса SPA могут повторно распределяться.

Гостевые физические адреса, которые были развязаны, могут повторно связываться с физическими адресами. В этом примере запрос на передачу страниц памяти может принимать VID 810, и VID 810 может получить адреса SPA, чтобы выполнить запрос, и послать диапазон адресов на DMVSP 802. В варианте осуществления VID 810 может быть конфигурирован для получения непрерывного диапазона адресов SPA, чтобы повышать эффективность системы. В этом примере VID 810 может определить, что гостевая операционная система 220 содержит адреса GPA, блокированные для монопольного использования посредством DMVSC 804, связанным с виртуальным NUMA узлом 606. VID 810 может создать взаимосвязи между блокированными адресами GPA и SPA и послать сообщение на DMVSP 802. DMVSP 802 затем может послать

сообщение на DMVSC 804, и DMVSC 804 может послать сообщение на диспетчер 808 памяти, указывая, что GPA могут быть разблокированы и возвращены в пул памяти в диспетчере 808 памяти, который связан с виртуальным NUMA узлом 606.

В варианте осуществления VID 810 может определять, использовать ли способ «горячего добавления» или динамического изменения выделяемой памяти в зависимости от того, изменялись ли динамически адреса GPA. Например, если VID 810 принимает адреса SPA для передачи памяти NUMA узлу 606, он может определить, блокировал ли DMVSC 804 какие-либо GPA. В случае, что имеются блокированные адреса GPA, VID 810 может поддержать их адресами SPA прежде выполнения «горячего добавления» памяти. Прежде, чем память передается виртуальному NUMA узлу 606, она может быть обнулена, и связанные с ней строки кэша могут быть очищены из соображений безопасности. Путем обнуления памяти не происходит утечка содержимого памяти, предварительно связанной с одним разделом, в другой раздел.

Возвращаясь теперь к Фиг.10, на ней изображен альтернативный вариант осуществления последовательности операций по Фиг.9, включающей дополнительные операции 1010-1020. Операция 1010 показывает определение, что нагрузка на память во втором виртуальном NUMA узле множества больше заранее заданного значения; и перемещение второго виртуального NUMA узла во второй NUMA узел вычислительной системы. Например, и возвращаясь к Фиг.7, в варианте осуществления нагрузка на память во втором виртуальном NUMA узле 608 может повыситься. То есть, DMVSP 802 может принять значение, указывающее нагрузку на память, которое указывает, что виртуальный NUMA узел 608 испытывает нагрузку. В этом примере виртуальная машина 240 или индивидуальный виртуальный NUMA узел 608 могут содержать целевое значение нагрузки, и текущее значение нагрузки может быть больше целевого значения, которое было задано администратором. Целевые значения нагрузки могут храниться в структуре данных, к которой можно осуществлять доступ DMVSP 802. Затем могут приниматься текущие значения нагрузки исполняющихся виртуальных машин или виртуальных NUMA узлов. DMVSP 802 может последовательно переходить по перечню исполняющихся виртуальных машин или виртуальных NUMA узлов и передавать память, чтобы уменьшить значения нагрузки на память до целевых значений, и осуществлять возврат памяти, чтобы увеличить нагрузку до целевых значений.

В примере, DMVSP 802 может быть конфигурирован для определения, что NUMA узел, в настоящий момент осуществляющий хостинг виртуальных NUMA узлов 606 и 608, например, NUMA узел 702, не является способным распределить достаточно памяти, чтобы получить целевые значения нагрузки на память для обоих виртуальных NUMA узлов. В этом примере DMVSP 802 может быть конфигурирован для посылки сигнала на гипервизор 202, и гипервизор 202 может быть конфигурирован для попытки смещения одного из виртуальных NUMA узлов из NUMA узла 702. Гипервизор 202 может проверить текущую рабочую нагрузку NUMA узлов 702-706 и определить, что, например, NUMA узел 704 может разместить виртуальный NUMA узел и распределить ему достаточно ресурсов для уменьшения нагрузки на память до целевого значения. В этом примере гипервизор 202 может быть конфигурирован для повторного назначения виртуального NUMA узла 608 на NUMA узел 704. То есть гипервизор 202, вместе с VID 810, может повторно отображать гостевые физические адреса 616 на системные физические адреса 714, и задавать логические процессоры 212 E и F в качестве идеальных процессоров для виртуальных процессоров 230 C и D.

Продолжая описание Фиг.10, операция 1012 иллюстрирует возврат, по меньшей мере, одного блока памяти в гостевой памяти от конкретного виртуального NUMA

узла; и передачу возвращенного, по меньшей мере, одного блока памяти гостевой памяти второму виртуальному NUMA узлу. Например, DMVSP 802 может быть конфигурирован для возврата памяти, например, от виртуального NUMA узла 606 и передачи памяти виртуальному NUMA узлу 608. В этом примере, виртуальные NUMA узлы 606 и 608 могут поддерживаться одиночным NUMA узлом, или 'плоской' архитектурой. В этом примерном варианте осуществления DMVSP 802 может пытаться освободить память из виртуального NUMA узла 606, когда, например, нет в наличии памяти, доступной для передачи виртуальному NUMA узлу 608. В другом примере, DMVSP 802 может быть конфигурирован для возврата памяти, например, от виртуального NUMA узла 610 и передачи памяти виртуальному NUMA узлу 608. То есть, память может заимствоваться из одной виртуальной машины и отдаваться другой.

В конкретном примере, и со ссылкой на Фиг.6, виртуальные NUMA узлы 606 и 608 могут отображаться на ресурсы вычислительной системы 600. В этом примере, DMVSP 802 может проверять другие виртуальные NUMA узлы, например, в порядке приоритета памяти, начиная, например, с низкоприоритетных виртуальных NUMA узлов в виртуальной машине 240 или виртуальной машине низшего приоритета. Если, например, выявлен виртуальный NUMA узел, такой как виртуальный NUMA узел 606, имеющий величину нагрузки на память, которая меньше целевой пороговой величины, DMVSP 802 может инициировать возврат памяти и удаление памяти из виртуального NUMA узла 606. После завершения возврата, может инициироваться операция передачи, и память может добавляться «в горячем режиме» к виртуальному NUMA узлу 608 или динамически измененные гостевые физические адреса могут быть развязаны с системными физическими адресами.

В конкретном примере, и со ссылкой на Фиг.7, DMVSP 802 может проверить другие виртуальные NUMA узлы, которые поддерживает тот же NUMA узел 702, например, в порядке приоритета памяти. Если, например, выявляется, что виртуальный NUMA узел на одном и том же NUMA узле с виртуальным NUMA узлом 608 имеет величину нагрузки на память, которая меньше целевой пороговой величины, DMVSP 802 может инициировать возврат памяти. После завершения возврата, может инициироваться операция передачи, и память может добавляться «в горячем режиме» к виртуальному NUMA узлу 608 или динамически измененные гостевые физические адреса могут повторно связываться с системными физическими адресами.

Продолжая описание Фиг.10, операция 1014 изображает определение, что, по меньшей мере, один блок гостевой памяти конкретного виртуального NUMA узла является развязанным с системной памятью; и отображение, по меньшей мере, одного блока памяти в гостевой памяти, по меньшей мере, на один блок памяти в системной памяти. Например, в варианте осуществления DMVSP 802 может исполняться логическим процессором, и может быть выполнено определение поддержать адреса GPA в виртуальном NUMA узле 606 адресами SPA 624. Например, адреса GPA может резервировать DMVSC 804, и адреса SPA могут повторно распределяться либо другому виртуальному NUMA узлу, либо родительскому разделу 204. В этом примере запрос на передачу страниц памяти может принимать VID 810, и VID 810 может получать адреса SPA, чтобы выполнить запрос, и посылать диапазон адресов на DMVSP 802. В варианте осуществления VID 810 может быть конфигурирован для получения непрерывного диапазона адресов SPA для того, чтобы повышать эффективность системы. В варианте осуществления NUMA, VID 810 может быть конфигурирован для получения непрерывного диапазона адресов SPA от того же NUMA узла, который исполняет виртуальный NUMA узел 606. VID 810 может создать взаимосвязи между

блокированными адресами GPA и адресами SPA и послать сообщение на DMVSP 802. DMVSP 802 может затем посылать сообщение на DMVSC 804, и DMVSC 804 может посылать на диспетчер 808 памяти сообщение, указывающее, что адреса GPA могут быть разблокированы и возвращены в пул памяти, связанный с виртуальным NUMA узлом 606.

Продолжая описание Фиг.10, операция 1016 иллюстрирует отображение конкретного виртуального NUMA узла на первый NUMA узел вычислительной системы; и перемещение конкретного виртуального NUMA узла на второй NUMA узел вычислительной системы. Например, и со ссылкой на Фиг.7, гостевая операционная система 220 может быть рассредоточена, по меньшей мере, по двум NUMA узлам, таким как NUMA узлы 702 и 704. Например, и со ссылкой на Фиг.7, гипервизор 202 может планировать виртуальные NUMA узлы 606 и 608 для исполнения на NUMA узле 702. В этом примере гипервизор 202 может принимать сигнал, который указывает, что NUMA узел 702 нагружен. Например, гостевая операционная система 220 может формировать сигналы, которые указывают, что виртуальные NUMA узлы 606 и 608 недостаточны по памяти. В этом примере, гипервизор 202 может быть конфигурирован для уменьшения рабочей нагрузки на нагруженном NUMA узле путем перемещения виртуального NUMA узла 608 из NUMA узла 702.

Продолжая описание Фиг.10, операция 1018 иллюстрирует добавление виртуального процессора к конкретному виртуальному NUMA узлу. Например, в варианте осуществления виртуальный процессор, такой как виртуальный процессор 230B, может добавляться в течение времени выполнения виртуальной машины 240 с использованием, например, режима процессора «горячее добавление». То есть, виртуальный NUMA узел 606 может иметь только один виртуальный процессор 230A в одной точке, и затем добавляется еще один. В варианте осуществления вновь добавленный процессор может быть назначен процессору, поддерживающему виртуальный процессор 230A, или может быть распределен другой логический процессор для исполнения потоков виртуального процессора 230B. В варианте осуществления NUMA, если другой логический процессор является используемым для поддержки виртуального процессора 230B, он может распределяться от того же NUMA узла 702, который поддерживает другие виртуальные процессоры в виртуальном NUMA узле 606.

Продолжая описание Фиг.10, операция 1020 иллюстрирует прием запроса на исполнение виртуального процессора виртуальной машины, причем виртуальный процессор назначен логическому процессору, логический процессор назначен NUMA узлу и виртуальный процессор назначен виртуальному NUMA узла; определение, что логический процессор не может исполнять виртуальный процессор; и выбор второго логического процессора для исполнения виртуального процессора, причем второй логический процессор из второго NUMA узла. Например, и со ссылкой на Фиг.7, в варианте осуществления гипервизор 202 может принимать запрос на исполнение потока виртуального процессора от виртуального процессора 230A и пытаться планировать поток на идеальный процессор 212A, например, процессор, поддерживающий виртуальный процессор 230A. В этом примере, гипервизор 202 может выявить, что логический процессор 212A является чрезмерно нагруженным (по ресурсам) и неспособным исполнять поток виртуального процессора. В этом случае, может исполняться гипервизор 202, и он может выбрать другой логический процессор для исполнения потока виртуального процессора. Например, гипервизор 202 может пытаться выбрать другой логический процессор на том же NUMA узле. Если, например, NUMA узел является чрезмерно нагруженным, гипервизор 202 может быть конфигурирован

для выбора удаленного процессора для исполнения виртуального процессора 230A, например, логического процессора 212E. В этом примере принятие решение относительно того, ожидать или планировать поток на удаленный узел, может выполняться с использованием NUMA отношения, связанного с NUMA узлом 704. Если NUMA отношение является низким, и вероятное ожидание идеального процессора является длительным, то может быть принято решение планировать поток на NUMA узел 704. С другой стороны, если NUMA отношение является высоким, и вероятное время ожидания является малым, то может быть принято решение ожидать.

Возвращаясь теперь к Фиг.11, на ней изображена последовательность операций для применения аспектов настоящего раскрытия, включающая операции 1100, 1102, 1104, и 1106. Сущность операции 1100 составляет последовательность операций, и операция 1102 показывает исполнение виртуальной машины, имеющей топологию, которая включает в себя множество виртуальных NUMA узлов, причем топология виртуальной машины формируется независимо от физической топологии вычислительной системы. Например, гипервизор 202 может исполнять виртуальную машину, имеющую множество виртуальных NUMA узлов. Как показано на Фиг.6, может быть создана виртуальная машина 240, которая включает в себя виртуальные NUMA узлы 606 и 608. Каждый из виртуальных NUMA узлов может иметь один или несколько виртуальных процессоров 230 A-D и гостевые физические адреса 614 и 616. В этом варианте осуществления виртуальные NUMA узлы 606 и 608 могут создаваться независимо от топологии базового аппаратного обеспечения. То есть, топология виртуальной машины является не связанной с базовым аппаратным обеспечением, например, как изображено на Фиг.6 и 7. Таким образом, в этом варианте осуществления, топология каждой виртуальной машины является развязанной от базовой физической топологии вычислительной системы, которая приводит ее в исполнение.

Продолжая описание Фиг.11, операция 1104 иллюстрирует определение нагрузки на память в каждом виртуальном NUMA узле множества. Например, и со ссылкой на Фиг.8, для каждого виртуального NUMA узла 606 и 608 нагрузка на память может быть получена, например, сформирована и/или принята. Информация нагрузки на память может идентифицировать, как на рабочую характеристику гостя влияет объем памяти, который доступен каждому виртуальному NUMA узлу 606-608. Эта информация может вычисляться в течение времени выполнения гостевой операционной системы 220, например, посредством одного или нескольких DMVSC, таких как DMVSC 804 и/или 806, и посылаться на DMVSP 802. То есть, в конкретном варианте осуществления логический процессор может исполнять команды, указывающие на DMVSC 804 или 806, и формировать информацию нагрузки на память для каждого виртуального NUMA узла. Эта информация затем может посылаться на DMVSP 802, например.

В примерном варианте осуществления информация нагрузки на память может включать в себя ряд значений, изменяющихся в диапазоне 0-4, и каждое значение может идентифицировать различный уровень нагрузки на память, который гостевая ОС испытывает, в зависимости от ресурсов виртуальных NUMA узлов 606-608. Поскольку гостевая операционная система становится более нагруженной, то есть, поскольку объем памяти, требуемый для эффективного исполнения текущей рабочей нагрузки, увеличивается, DMVSC 804 и 806 могут пересматривать свои значения и передавать эту информацию на DMVSP 802.

Продолжая описание Фиг.11, операция 1106 показывает подстройку, на основании нагрузки на память в каждом виртуальном NUMA узле множества, гостевой памяти, назначенной, по меньшей мере, одному виртуальному NUMA узлу множества. В варианте

осуществления, который включает в себя операцию 1206, логический процессор 212A может исполнять команды, указывающие DMVSP 802, и может подстраивать объемы гостевых физических адресов в виртуальном NUMA узле 606, например. То есть, DMVSP 802 может осуществлять передачу или возврат памяти на основании нагрузки на память в виртуальном NUMA узле 606, например, память может передаваться, если процессы, распределенные виртуальному NUMA узлу 606, являются нагруженными.

В варианте осуществления, когда логический процессор 212, конфигурированный посредством DMVSP 802, определяет, осуществлять передачу или возврат памяти, он может делать это на основе «по блоку памяти». Например, DMVSP 802 может передавать/возвращать блок памяти и проверять, каким образом изменяется состояние памяти. Если состояние памяти не изменилось, DMVSP 802 может передавать/возвращать другой блок памяти.

Возвращаясь теперь к Фиг.12, на ней изображен альтернативный вариант осуществления последовательности операций 1100 по Фиг.11, включающей операции 1208-1216. Как проиллюстрировано фигурой чертежа, операция 1208 показывает посылку виртуальной машины на вторую вычислительную систему. Например, и со ссылкой на Фиг.5, в одном варианте осуществления состояние виртуальной машины может храниться в одном или нескольких конфигурационных файлах и посылаться на другую вычислительную систему, например, посылаться от компьютера 504 на 506. Гипервизор вычислительной системы 506 может считывать файл или файлы и создавать экземпляр виртуальной машины.

Топология виртуальной машины влияет на способность перемещать и восстанавливать виртуальные машины. Конкретно, решение позволять, чтобы выявлялась топология базового аппаратного обеспечения, и влияние размера виртуальных NUMA узлов на то, насколько хорошо виртуальная машина будет исполнять и может ли она быть легко перемещаемой. Например, размер виртуальных NUMA узлов влияет на способность перемещения, например, если размер виртуального NUMA узла повышается, переносимость виртуального NUMA узла снижается, и если размер виртуального NUMA узла уменьшается, то же делает рабочая характеристика виртуальной машины. Кроме того, виртуальные машины, которые могут выявлять топологию базового компьютера, не могут легко перемещаться вследствие факта, что осведомленные («информированные») о NUMA операционные системы и приложения оптимизирует себя во время начальной загрузки на основании первой топологии, которую они выявляют, и эта оптимизация может не работать хорошо на компьютерах, на которые виртуальная машина может быть перемещена в будущем. Таким образом, путем объявления виртуальных NUMA узлов гостевой операционной системе при ее начальной загрузке, операционная система может быть оптимизирована для использования NUMA узлов. Путем установки размеров виртуальных NUMA узлов корректно виртуальная машина может быть оптимизирована для многих разнотипных вычислительных систем в дата-центре 500.

Например, и со ссылкой на Фиг.6, виртуальная машина 240 может включать в себя два или большее число виртуальных NUMA узлов 606 и 608. Гипервизор 202 может поддерживать виртуальные процессоры 230 A-D логическими процессорами 212 A-D. При начальной загрузке гостевой операционной системы 220 она может выявить виртуальные NUMA узлы 606 и 608, и может быть конфигурирована для оптимизации планирования и исполнения процессов, чтобы использовать виртуальные NUMA узлы 606 и 608. После некоторого времени виртуальная машина 240 может быть перемещена на вычислительную систему, имеющую физическую топологию, подобную изображенной

Фиг.7. Гипервизор 202 по Фиг.7 может поддержать виртуальные процессоры 230 A и B логическими процессорами 212 A и B и поддержать виртуальные процессоры 230 C и D логическими процессорами 212 E и F. Гостевая операционная система 220 может продолжать работать таким же образом, как если бы исполнялась на вычислительной системе по Фиг.6, даже если топология базового компьютера изменилась из SMP в NUMA.

Продолжая описание Фиг.12, операция 1210 показывает отображение первого виртуального NUMA узла в множестве на первый NUMA узел в вычислительной системы; и отображение второго виртуального NUMA узла в множестве на первый NUMA узел вычислительной системы. Например, и со ссылкой на Фиг.7, логический процессор, такой как логический процессор 212A, может исполнять команды гипервизора, и может «привязать» виртуальные NUMA узлы, такие как виртуальные NUMA узлы 606 и 608, к NUMA узлу, такому как NUMA узел 702. Более конкретно, логический процессор может поддержать гостевые физические адреса 614 и 616 системными физическими адресами из ОЗУ 214 в составе NUMA узла 702, и может поддержать виртуальные процессоры 230 A-D логическими процессорами 212 A-D.

Продолжая описание Фиг.12, операция 1212 показывает отображение первого виртуального NUMA узла множества на первый NUMA узел вычислительной системы; и отображение второго виртуального NUMA узла множества на второй NUMA узел вычислительной системы. Например, и со ссылкой на Фиг.7, логический процессор, такой как логический процессор 212A, может исполнять команды гипервизора и может назначать виртуальный NUMA узел 606 на NUMA узел 702 и назначать виртуальный NUMA узел 608 на NUMA узел 704. В этом примере, при исполнении гипервизора 202, планировщик гипервизора может планировать потоки от виртуальных процессоров 230 A-B на логические процессоры 212A-D и планировать потоки от виртуальных процессоров 230 C или D на логические процессоры 212 E-G.

Возвращаясь теперь к Фиг.13, на ней иллюстрируется альтернативный вариант осуществления последовательности операций по Фиг.12, включающей операцию 1314, которая показывает определение, что нагрузка на память во втором виртуальном NUMA узле больше заранее заданного значения; и перемещение второго виртуального NUMA узла на второй NUMA узел вычислительной системы. Например, и возвращаясь к Фиг.7, в варианте осуществления нагрузка на память во втором виртуальном NUMA узле 608 может повыситься. То есть, DMVSP 802 может принимать значение, указывающее нагрузку на память, которое указывает, что виртуальный NUMA узел 608 является нагруженным. В этом примере каждый виртуальный NUMA узел 606-612 и/или виртуальная машина 240-242, могут иметь целевое значение нагрузки, и текущее значение нагрузки для виртуального NUMA узла 608 может быть больше целевого значения, которое было задано администратором. Целевые значения нагрузки могут храниться в структуре данных, к которой может осуществлять доступ DMVSP 802. Затем могут приниматься текущие значения нагрузки исполняющихся виртуальных машин или виртуальных NUMA узлов. DMVSP 802 может последовательно переходить по перечню исполняющихся виртуальных машин или виртуальных NUMA узлов и осуществлять передачу памяти, чтобы уменьшить значения нагрузки на память до целевых значений, и возврат памяти, чтобы увеличить нагрузку до целевых значений.

Возвращаясь теперь к Фиг.14, на ней иллюстрируется альтернативный вариант осуществления последовательности операций по Фиг.12, включающей операцию 1416, которая показывает определение, что нагрузка на память второй виртуальной машины больше заранее заданного значения; и перемещение второго виртуального NUMA узла

виртуальной машины на первый NUMA узел вычислительной системы. В варианте осуществления, по меньшей мере, две виртуальные машины могут быть исполняющимися, например, виртуальные машины 240 и 242 из таковых. В этом примере виртуальные NUMA узлы виртуальной машины 240 могут отображаться, например, на NUMA узлы 702 и 704 по Фиг.7 и виртуальные NUMA узлы виртуальной машины 242 могут отображаться, например, на NUMA узел 706. В этом примере каждая виртуальная машина 240 и 242 и/или каждый виртуальный NUMA узел 606-612 могут иметь целевое значение нагрузки, которое может храниться в структуре данных, к которой может осуществлять доступ DMVSP 802. В этом примере нагрузка на память во второй виртуальной машине 242 может повыситься в результате активности в виртуальной машине, например, виртуальная машина 242 приняла много запросов на считывание/запись, и значение может быть принято посредством DMVSP 802. DMVSP 802 может принимать текущие значения нагрузки для исполняющихся виртуальных машин или виртуальных NUMA узлов и последовательно переходить по перечню исполняющихся виртуальных машин или виртуальных NUMA узлов, чтобы определить, может ли память передаваться виртуальной машине 242, чтобы смягчить нагрузку на память.

В ситуации, где нагрузку нельзя смягчить путем передачи или возврата памяти, DMVSP 802 может быть конфигурирован для посылки сигнала на гипервизор 202, и гипервизор 202 может пытаться повторно распределять ресурсы вычислительной системы, чтобы уменьшить нагрузку на память. Например, гипервизор 202 может проверить текущую рабочую нагрузку NUMA узлов 702-706 и определить, что, например, NUMA узел 702 может разместить виртуальные NUMA узлы из виртуальной машины 240, и повторно назначить виртуальный NUMA узел 608 на NUMA узел 702. То есть, гипервизор 202 вместе с VID 810 может повторно отображать гостевые физические адреса 616 на системные физические адреса 712, и задавать логические процессоры 212 A и D в качестве идеальных процессоров для виртуальных процессоров 230 C и D. Затем гипервизор может повторно отображать виртуальный NUMA узел 610 на NUMA узел 704 и подстраивать память внутри каждого виртуального NUMA узла 610-612 виртуальной машины 242, чтобы уменьшить ее нагрузку на память.

Возвращаясь теперь к Фиг.15, на ней иллюстрируется последовательность операций для применения аспектов настоящего раскрытия, включающая операции 1500, 1502, и 1504. Операция 1500 начинает последовательность операций, и операция 1502 показывает исполнение первой виртуальной машины с топологией, включающей в себя множество виртуальных NUMA узлов, каждый виртуальный NUMA узел множества включает в состав виртуальный процессор и гостевые физические адреса, причем топология виртуальной машины формируется независимо от физической топологии вычислительной системы. Например, гипервизор 202 по Фиг.7 может исполнять виртуальную машину 240, содержащую множество виртуальных NUMA узлов 608-610. Виртуальные NUMA узлы 606 и 608 могут каждый иметь один или несколько виртуальных процессоров 230 A-D и гостевые физические адреса 614 и 616. В этом варианте осуществления виртуальные NUMA узлы 606 и 608 отображаются на ресурсы вычислительной системы 700. Например, логические процессоры 212 A и B могут быть заданы в качестве идеальных процессоров для виртуального процессора 230 A и B, и гостевые физические адреса 614 могут быть поддержаны системными физическими адресами 710. Аналогично, логические процессоры 212 E и F могут быть заданы в качестве идеальных процессоров для виртуального процессора 230 C и D и гостевые физические адреса 616 могут отображаться на системные физические адреса 714. В этом

варианте осуществления виртуальные NUMA узлы 606 и 608 являются независимыми от топологии базового аппаратного обеспечения. То есть, топология виртуальной машины является не связанной с базовым аппаратным обеспечением, например, как изображено на Фиг.6 и 7. Таким образом, в этом варианте осуществления, топология виртуальной машины является развязанной от нижележащей физической топологии вычислительной системы, которая приводит ее в исполнение.

Продолжая описание Фиг.15, операция 1504 показывает добавление дополнительного виртуального процессора к виртуальному NUMA узлу в множестве. Например, в варианте осуществления дополнительный виртуальный процессор может быть добавлен к виртуальному NUMA узлу такому как, например, виртуальный NUMA узел 606. В этом примере виртуальный процессор, такой как виртуальный процессор, может быть добавлен в течение времени выполнения виртуальной машины 240 с использованием, например, режима процессора «горячее добавление». В варианте осуществления вновь добавленный процессор может быть назначен на процессор, поддерживающий виртуальный процессор 230A, или другой логический процессор может быть задан в качестве идеального процессора, чтобы исполнять потоки вновь добавленного виртуального процессора. В варианте осуществления NUMA логические процессоры могут распределяться от того же NUMA узла 702, поддерживающего виртуальный NUMA узел 606.

Возвращаясь теперь к Фиг.16, на ней изображен альтернативный вариант осуществления последовательности операций, изображенной на Фиг.15, включающей дополнительные операции 1606-1612. Операция 1606 показывает определение нагрузки на память в каждом виртуальном NUMA узле множества; и подстройку, на основании нагрузки на память в каждом виртуальном NUMA узле множества, гостевой памяти, назначенной, по меньшей мере, одному виртуальному NUMA узлу множества. Со ссылкой на Фиг.8, нагрузка на память для каждого виртуального NUMA узла 606 и 608 может быть получена, например, сформирована и/или принята. Информация нагрузки на память может идентифицировать, как на рабочую характеристику гостя влияет объем памяти, доступный каждому виртуальному NUMA узлу. Клиенты DMVSC 804 и 806 могут конфигурироваться для приема, например, информации уведомлений о физической памяти и/или страничной организации гостевой операционной системы от диспетчера 808 памяти и использования ее для вычисления нагрузки на память для каждого виртуального NUMA узла 606 и 608.

Продолжая пример, логический процессор 212A, например, может исполнять команды, указывающие DMVSP 802, и может подстраивать объем гостевых физических адресов в виртуальном NUMA узле 606, например. То есть, DMVSP 802 осуществляет передачу или возврат памяти для виртуального NUMA узла 606 на основании нагрузки на память, которую гостевая ОС испытывает, с учетом ресурсов внутри него.

Продолжая описание Фиг.16, операция 1608 (1606?) иллюстрирует определение нагрузки на память в каждом виртуальном NUMA узле множества; и подстройку, на основании нагрузки на память в каждом виртуальном NUMA узле множества, гостевой памяти, назначенной, по меньшей мере, одному виртуальному NUMA узлу множества. В варианте осуществления, который включает в себя этап 1706, гипервизор 202 может исполняться логическим процессором, и виртуальный процессор может быть удален из виртуального NUMA узла 606. Например, гипервизор 202 может осуществлять доступ к API «горячего удаления» гостевой операционной системы 220 и удаляет, например, виртуальный процессор 230B из виртуального NUMA узла 608.

Возвращаясь теперь к операции 1610, она показывает предоставление отчетов NUMA

отношений для множества виртуальных NUMA узлов на гостевую операционную систему. Например, в варианте осуществления гипервизор 202 может формировать NUMA отношения для виртуальных NUMA узлов 606-608, и эта информация может сообщаться на гостевую операционную систему 220 либо по Фиг.6, либо по Фиг.7. В варианте осуществления гипервизор 202 может создавать структуру данных в таблице микропрограммных средств виртуальной машины, которая указывает NUMA отношение для виртуальных NUMA узлов, и при начальной загрузке гостевой операционной системы 220 гость может считывать таблицу и использовать информацию, чтобы принимать решения планирования. Например, гостевая операционная система, или осведомленное о NUMA приложение могут использовать NUMA отношения, чтобы определить, использовать или не использовать ресурсы от удаленного NUMA узла. Например, операционная система может иметь ожидающий обработки поток, который готов к исполнению. Операционная система в этом примере также может быть конфигурирована для ожидания в течение некоторого времени, пока не освободится идеальный процессор, в противном случае она планирует поток на удаленный процессор, у которого NUMA отношение меньше заранее заданного значения. В этом случае время, которое планировщик согласен ожидать, зависит от NUMA отношения.

Возвращаясь теперь к операции 1612, она показывает перемещение виртуальной машины из первого NUMA узла в множество NUMA узлов. Например, в варианте осуществления команды гипервизора могут исполняться логическим процессором, и виртуальная машина 240 может отображаться на множество NUMA узлов 704 и 706. В этом примере вычислительная система 700 может находиться в условиях тяжелого использования, например, виртуальная машина 242 возможно использовала большинство из ресурсов, и поэтому виртуальная машина 242 была смещена с компьютера 700. В этой ситуации гипервизор 202 может повторно распределять ресурсы вычислительной системы 700 и повторно отображать виртуальную машину 240 на NUMA узлы 704 и 706.

Возвращаясь теперь к Фиг.17, на ней изображен альтернативный вариант осуществления последовательности операций по Фиг.16, включающей операции 1714 и 1716. В варианте осуществления операция 1606 может включать в себя операцию 1714, которая показывает возврат памяти от первого виртуального NUMA узла множества на основании определения, что текущая нагрузка на память виртуального NUMA узла ниже целевой пороговой величины. Например, в варианте осуществления DMVSP 802 может быть конфигурирован для возврата памяти от виртуального NUMA узла 606, если нагрузка на память виртуального NUMA узла 608 ниже целевой пороговой величины. Например, в варианте осуществления DMVSP 802 может пытаться освободить память из виртуального NUMA узла 606, например, если в родительском разделе нет доступной памяти, которая может передаваться, и виртуальный NUMA узел 608 испытывает недопустимую нагрузку. Если осуществляется возврат памяти от виртуального NUMA узла 606, на DMVSC 804 может посылаться асинхронное сообщение, указывающее ему осуществить возврат памяти. При ответе DMVSC 804, связанный с гостевой ОС 220, может указывать новую нагрузку на память в рамках виртуального NUMA узла 606. В некоторых случаях, нагрузка на память может быть увеличена в ответ на операцию удаления памяти.

Продолжая описание Фиг.17, операция 1716 показывает передачу памяти первому виртуальному NUMA узлу множества на основании определения, что текущая нагрузка на память гостевой операционной системы больше целевой пороговой величины. Например, в варианте осуществления DMVSP 802 может быть конфигурирован для

передачи памяти виртуальному NUMA узлу 606, если нагрузка на память виртуального NUMA узла 606 больше целевой пороговой величины. В этом случае, если память имеется в наличии, она может быть распределена виртуальному NUMA узлу 606. То есть, DMVSP 802 может получать информацию нагрузки на память, которая идентифицирует, как на рабочую характеристику виртуального NUMA узла 606 влияет имеющийся объем памяти, и добавлять память виртуальному NUMA узлу 606. В конкретном примере информация нагрузки на память может быть значением. В этом примере DMVSP 802 может сравнивать текущее значение нагрузки на память с таблицей информации, которая указывает минимальное значение для виртуального NUMA узла 606, и подстраивать память, пока нагрузка на память виртуального NUMA узла 606 не станет равной минимальной. Например, администратор может конфигурировать гостевую операционную систему, исполняющую критическое приложение, чтобы имела низкие минимальные значения для виртуальных NUMA узлов 606 и 608.

В предшествующем подробном описании изложены различные варианты осуществления систем и/или процессов посредством примеров и/или функциональных схем. Поскольку такие структурные схемы и/или примеры содержат одну или несколько функций и/или операций, специалистам в данной области техники будет понятно, что каждая функция и/или операция в рамках таких структурных схем или примеров может быть осуществлена отдельно и/или вместе посредством широкого ряда аппаратных средств, программного обеспечения, микропрограммного обеспечения или фактически любой комбинации таковых.

Хотя были показаны и описаны конкретные аспекты настоящего объекта изобретения, описанного в документе, специалистам в данной области техники будет очевидно, что на основании указаний в документе, могут выполняться изменения и модификации без выхода за рамки объема объекта изобретения, описанного в документе, и его более широких аспектов, и, следовательно, прилагаемая формула изобретения должна охватывать в рамках своего объема все такие изменения и модификации, которые находятся в рамках истинного существа и объема охраны объекта изобретения, описанного в документе.

Формула изобретения

1. Способ управления памятью для виртуальных машин, содержащий этапы, на которых:

принимают запрос создания экземпляра виртуальной машины, причем запрос включает в себя характеристику виртуальной машины;

выбирают топологию виртуальных узлов архитектуры неоднородной памяти (NUMA) для виртуальной машины на основе этой характеристики, причем топология виртуальных NUMA узлов включает в себя множество виртуальных NUMA узлов;

создают экземпляр виртуальной машины в вычислительной системе, причем виртуальная машина включает в себя множество виртуальных NUMA узлов; и

регулируют, на основе, по меньшей мере частично, нагрузки на память в конкретном виртуальном NUMA узле из множества виртуальных NUMA узлов, объем гостевой памяти, назначенной этому конкретному виртуальному NUMA узлу, на величину, равную второму объему гостевой памяти, причем вторым объемом гостевой памяти осуществляется обмен между упомянутым конкретным виртуальным NUMA узлом и вторым виртуальным NUMA узлом из множества виртуальных NUMA узлов на основе, по меньшей мере частично, нагрузки на память во втором виртуальном NUMA узле.

2. Способ по п. 1, дополнительно содержащий этапы, на которых:

определяют, что нагрузка на память во втором виртуальном NUMA узле из множества виртуальных NUMA узлов больше заранее заданного значения; и

перемещают второй виртуальный NUMA узел на второй NUMA узел вычислительной системы.

3. Способ по п. 1, в котором регулировка объема гостевой памяти на величину, равную второму объему гостевой памяти, дополнительно содержит:

возвращают по меньшей мере один блок памяти в гостевой памяти от упомянутого конкретного виртуального NUMA узла; и

передают возвращенный по меньшей мере один блок памяти в гостевой памяти второму виртуальному NUMA узлу.

4. Способ по п. 1, в котором регулировка объема гостевой памяти на величину, равную второму объему гостевой памяти, дополнительно содержит:

определяют, что по меньшей мере один блок гостевой памяти упомянутого конкретного виртуального NUMA узла является развязанным с системной памятью; и

соотносят этот по меньшей мере один блок памяти в гостевой памяти с по меньшей мере одним блоком памяти в системной памяти.

5. Способ по п. 1, дополнительно содержащий этапы, на которых:

соотносят упомянутый конкретный виртуальный NUMA узел с первым NUMA узлом вычислительной системы; и

перемещают упомянутый конкретный виртуальный NUMA узел на второй NUMA узел вычислительной системы.

6. Способ по п. 1, дополнительно содержащий этап, на котором добавляют виртуальный процессор к упомянутому конкретному виртуальному NUMA узлу.

7. Способ по п. 1, дополнительно содержащий этапы, на которых:

принимают запрос на исполнение виртуального процессора виртуальной машины, причем виртуальный процессор назначен логическому процессору, логический процессор назначен NUMA узлу и виртуальный процессор назначен одному из множества виртуальных NUMA узлов;

определяют, что логический процессор не может исполнять виртуальный процессор; и

выбирают второй логический процессор для исполнения виртуального процессора, причем второй логический процессор из другого NUMA узла.

8. Вычислительная система, выполненная с возможностью управления памятью для виртуальных машин, при этом вычислительная система содержит:

схему для исполнения виртуальной машины, причем виртуальная машина имеет топологию, которая включает в себя множество виртуальных узлов архитектуры неоднородной памяти (NUMA);

схему для определения нагрузки на память в каждом виртуальном NUMA узле из множества виртуальных NUMA узлов; и

схему для регулировки, на основе, по меньшей мере частично, упомянутой определенной нагрузки на память в конкретном виртуальном NUMA узле из множества виртуальных NUMA узлов, объема гостевой памяти, назначенной этому конкретному виртуальному NUMA узлу, на величину, равную второму объему гостевой памяти, причем вторым объемом гостевой памяти осуществляется обмен между упомянутым конкретным виртуальным NUMA узлом и другим виртуальным NUMA узлом из множества виртуальных NUMA узлов на основе, по меньшей мере частично, упомянутой определенной нагрузки на память в этом другом виртуальном NUMA узле.

9. Вычислительная система по п. 8, дополнительно содержащая схему для посылки

виртуальной машины на вторую вычислительную систему.

10. Вычислительная система по п. 8, дополнительно содержащая:

схему для соотнесения первого виртуального NUMA узла из множества виртуальных NUMA узлов с первым NUMA узлом вычислительной системы; и

схему для соотнесения второго виртуального NUMA узла из множества виртуальных NUMA узлов с первым NUMA узлом вычислительной системы.

11. Вычислительная система по п. 8, дополнительно содержащая:

схему для соотнесения первого виртуального NUMA узла из множества виртуальных NUMA узлов с первым NUMA узлом вычислительной системы; и

схему для соотнесения второго виртуального NUMA узла из множества виртуальных NUMA узлов со вторым NUMA узлом вычислительной системы.

12. Вычислительная система по п. 10, дополнительно содержащая:

схему для определения, что нагрузка на память во втором виртуальном NUMA узле больше заранее заданного значения; и

схему для перемещения второго виртуального NUMA узла на второй NUMA узел вычислительной системы.

13. Вычислительная система по п. 11, дополнительно содержащая:

схему для определения, что нагрузка на память второй виртуальной машины больше заранее заданного значения; и

схему для перемещения второго виртуального NUMA узла виртуальной машины на первый NUMA узел вычислительной системы.

14. Считываемый компьютером носитель информации, включающий в себя исполняемые процессором команды, при этом считываемый компьютером носитель информации содержит:

команды для исполнения первой виртуальной машины, причем первая виртуальная машина имеет топологию, которая включает в себя множество виртуальных узлов архитектуры неоднородной памяти (NUMA), и каждый виртуальный NUMA узел из множества виртуальных узлов NUMA включает в себя виртуальный процессор и гостевые физические адреса, причем топология первой виртуальной машины, которая включает в себя упомянутое множество виртуальных узлов NUMA, является независимой от того, содержит ли физическая топология вычислительной системы, исполняющей эту виртуальную машину, физические узлы NUMA;

команды для добавления дополнительного виртуального процессора к виртуальному NUMA узлу из множества виртуальных узлов NUMA во время исполнения первой виртуальной машины;

команды для определения нагрузки на память в множестве виртуальных NUMA узлов; и

команды для регулировки, на основе, по меньшей мере частично, упомянутой определенной нагрузки на память в конкретном виртуальном NUMA узле из множества виртуальных NUMA узлов, объема гостевой памяти, назначенной этому конкретному виртуальному NUMA узлу, на величину, равную второму объему гостевой памяти, причем вторым объемом гостевой памяти осуществляется обмен между упомянутым конкретным виртуальным NUMA узлом и другим виртуальным NUMA узлом из множества виртуальных NUMA узлов на основе, по меньшей мере частично, упомянутой определенной нагрузки на память в этом другом виртуальном NUMA узле.

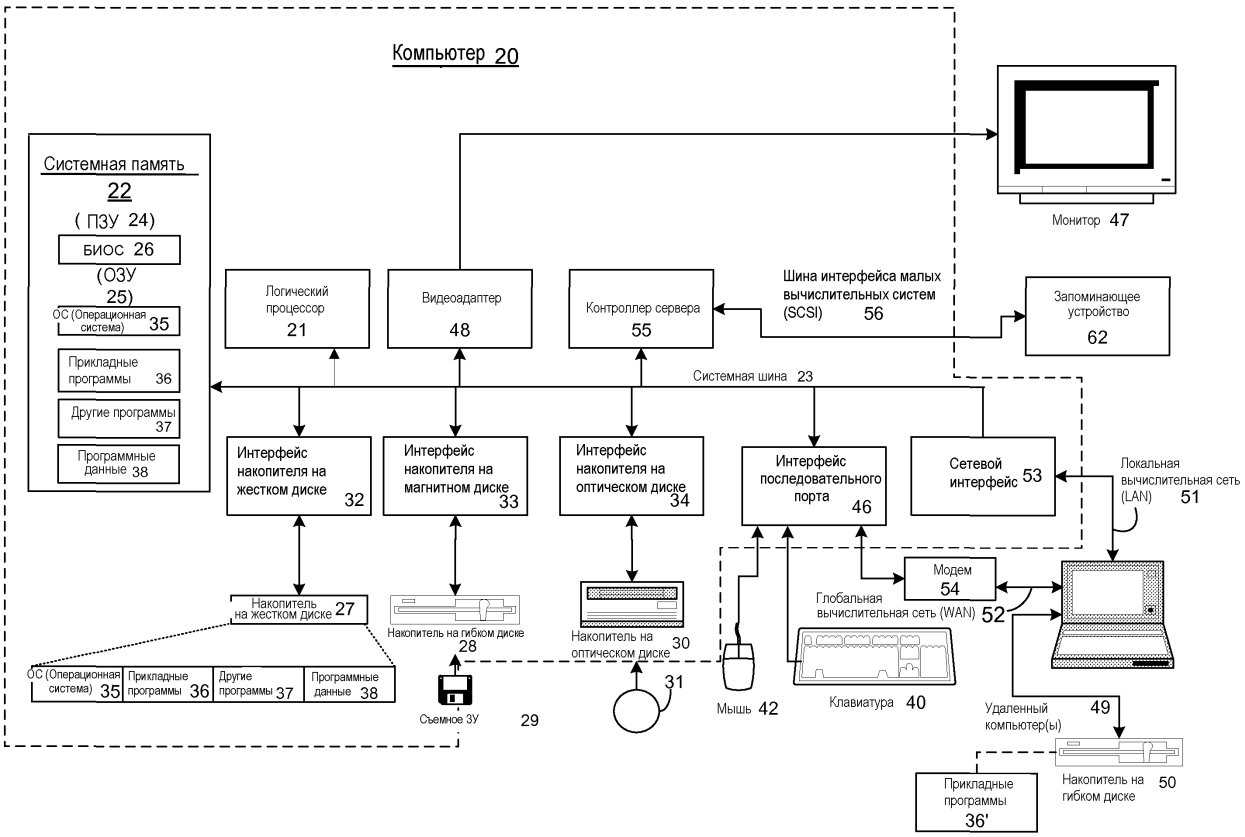
15. Считываемый компьютером носитель информации по п. 14, дополнительно содержащий команды для удаления виртуального процессора из виртуального NUMA узла из множества виртуальных NUMA узлов.

16. Считываемый компьютером носитель информации по п. 14, дополнительно содержащий команды для предоставления отчета о NUMA отношениях для множества виртуальных NUMA узлов на гостевую операционную систему.

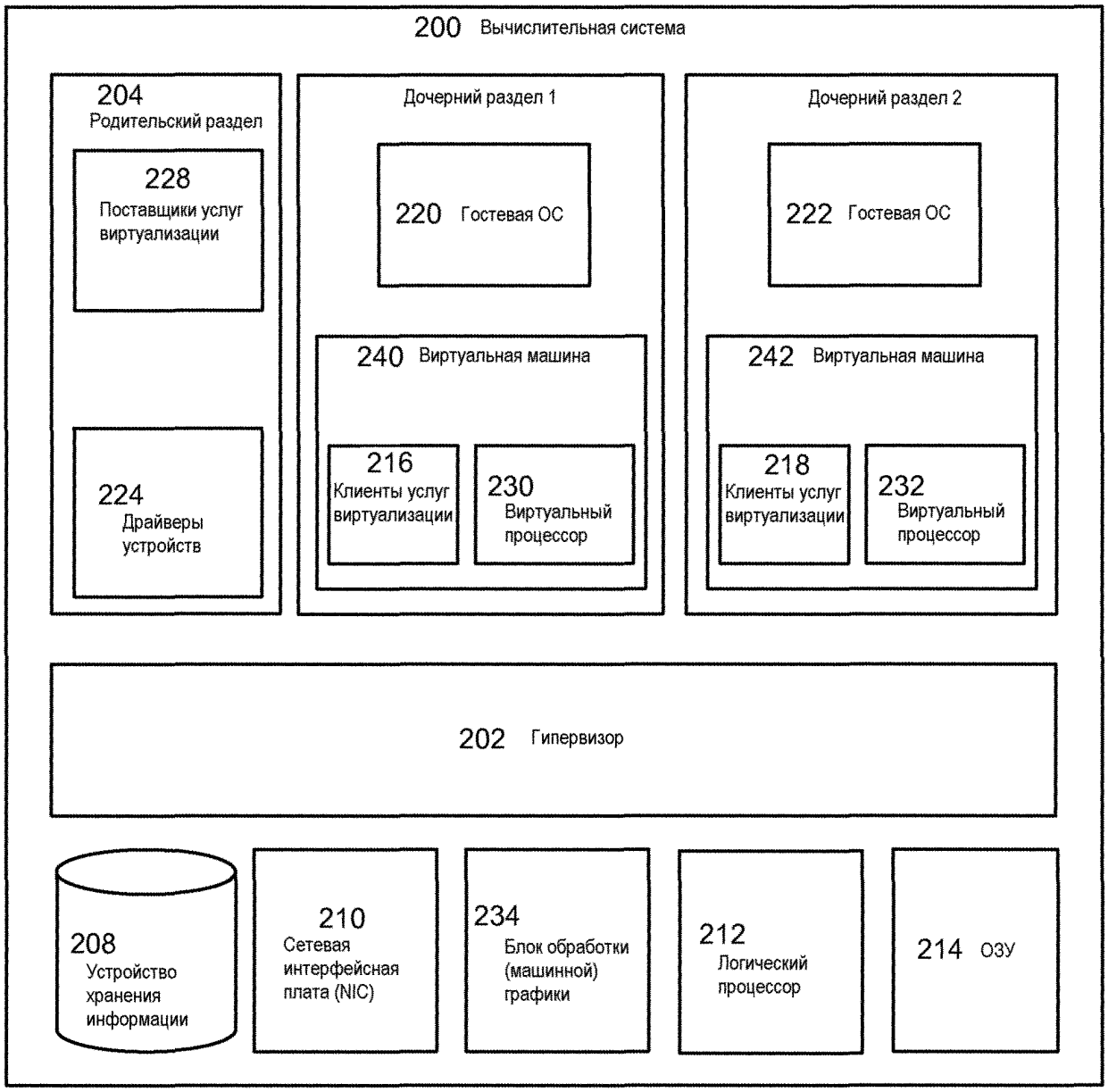
17. Считываемый компьютером носитель информации по п. 14, дополнительно содержащий команды для перемещения виртуальной машины из первого NUMA узла в множество NUMA узлов.

18. Считываемый компьютером носитель информации по п. 14, в котором команды для регулировки объема гостевой памяти дополнительно содержат команды для возврата памяти от первого виртуального NUMA узла из множества виртуальных NUMA узлов на основе определения того, что текущая нагрузка на память первого виртуального NUMA узла ниже целевой пороговой величины.

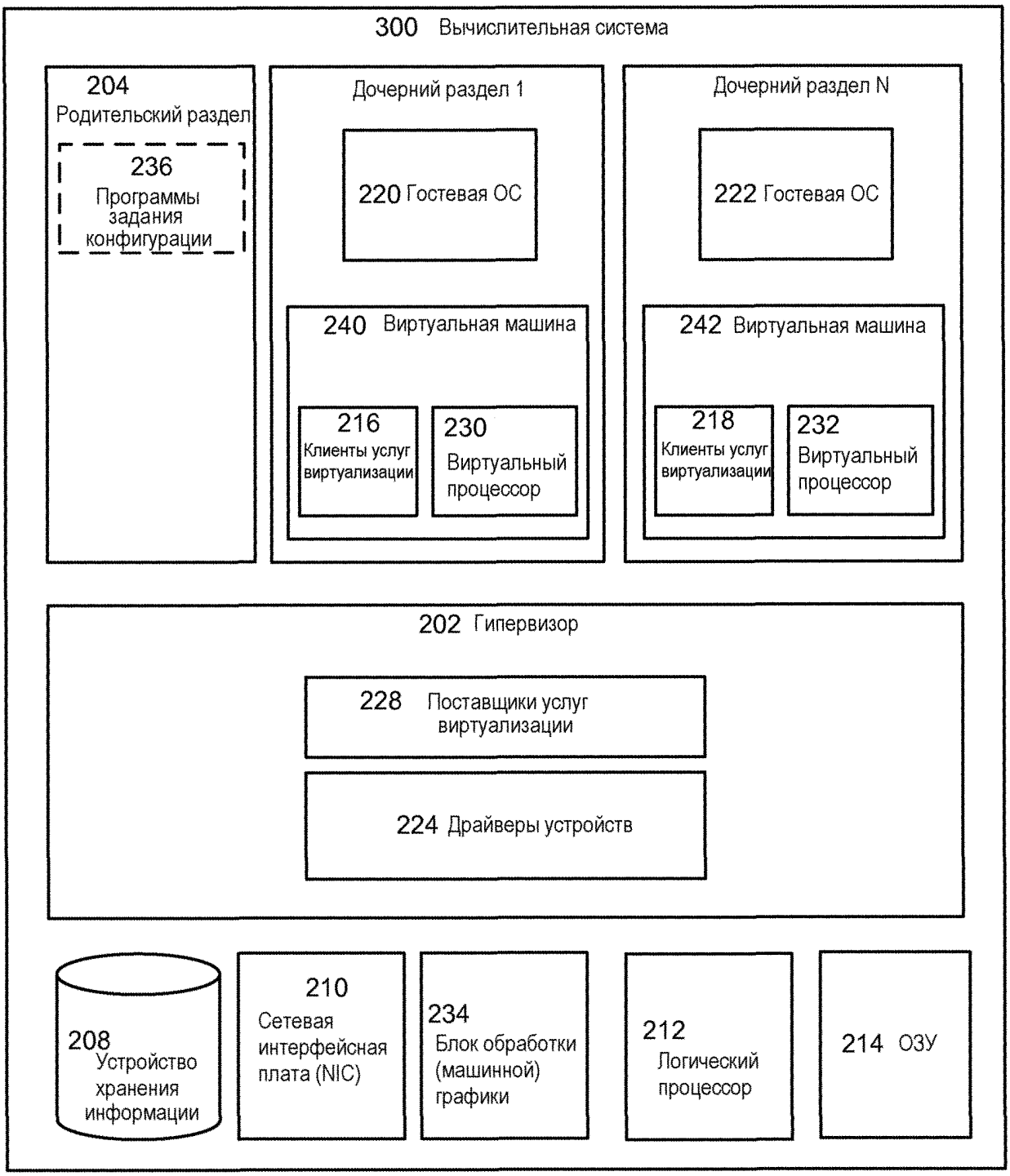
19. Считываемый компьютером носитель информации по п. 14, в котором команды для регулировки объема гостевой памяти дополнительно содержат команды для передачи памяти первому виртуальному NUMA узлу из множества виртуальных NUMA узлов на основе определения того, что текущая нагрузка на память гостевой операционной системы больше целевой пороговой величины.


Компьютер 20
Системная память 22
(ПЗУ 24)
БИОС 26
(ОЗУ 25)
ОС (Операционная система) 35
Прикладные программы 36
Другие программы 37
Программные данные 38
Логический процессор 21
Видеоадаптер 48
Контроллер сервера 55
Шина интерфейса малых вычислительных систем (SCSI) 56
Монитор 47
Запоминающее устройство 62
Системная шина 23
Интерфейс накопителя на жестком диске 32
Интерфейс накопителя на магнитном диске 33
Интерфейс накопителя на оптическом диске 34
Интерфейс последовательного порта 46
Сетевой интерфейс 53
Локальная вычислительная сеть (LAN) 51
Модем 54
Глобальная вычислительная сеть (WAN) 52
Накопитель на жестком диске 27
Накопитель на гибком диске 28
Накопитель на оптическом диске 30
Съемное ЗУ 29
Мышь 42
Клавиатура 40
Удаленный компьютер(ы) 49
Прикладные программы 36'
Накопитель на гибком диске 50

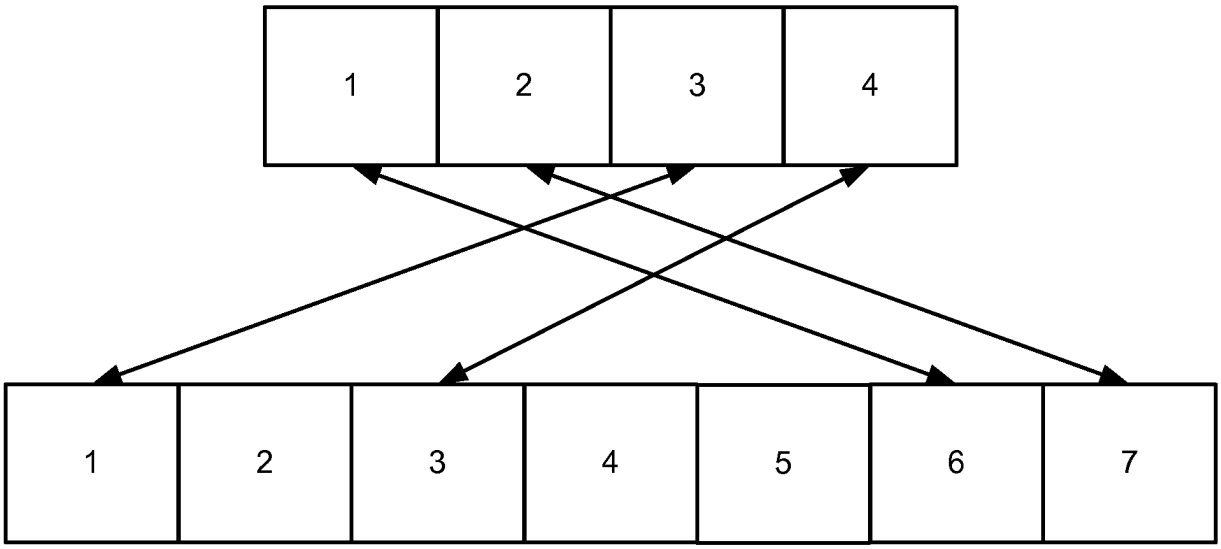
Фиг. 1


200 Вычислительная система
204 Родительский раздел
228 Поставщики услуг виртуализации
224 Драйверы устройств
Дочерний раздел 1
220 Гостевая ОС
240 Виртуальная машина
216 Клиенты услуг виртуализации
230 Виртуальный процессор
Дочерний раздел 2
222 Гостевая ОС
242 Виртуальная машина
218 Клиенты услуг виртуализации
232 Виртуальный процессор
202 Гипервизор
208 Устройство хранения информации
210 Сетевая интерфейсная плата (NIC)
234 Блок обработки (машинной) графики
212 Логический процессор
214 ОЗУ

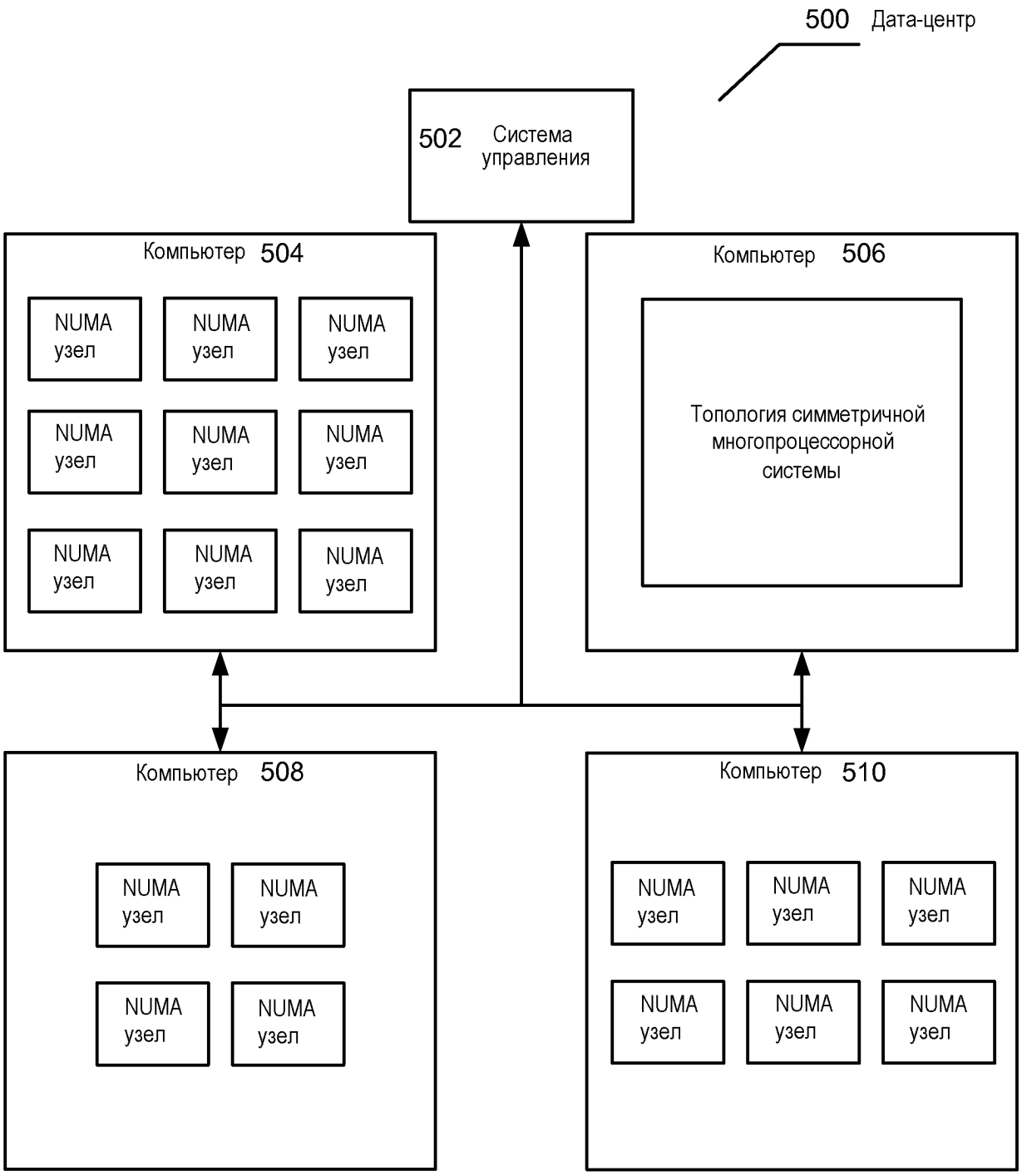
Фиг. 2


300 Вычислительная система
204
Родительский раздел
236
Программы задания конфигурации
Дочерний раздел 1
220 Гостевая ОС
240 Виртуальная машина
216
Клиенты услуг виртуализации
230
Виртуальный процессор
Дочерний раздел N
222 Гостевая ОС
242 Виртуальная машина
218
Клиенты услуг виртуализации
232
Виртуальный процессор
202 Гипервизор
228 Поставщики услуг виртуализации
224 Драйверы устройств
208
Устройство хранения информации
210
Сетевая интерфейсная плата (NIC)
234
Блок обработки (машинной) графики
212
Логический процессор
214 ОЗУ

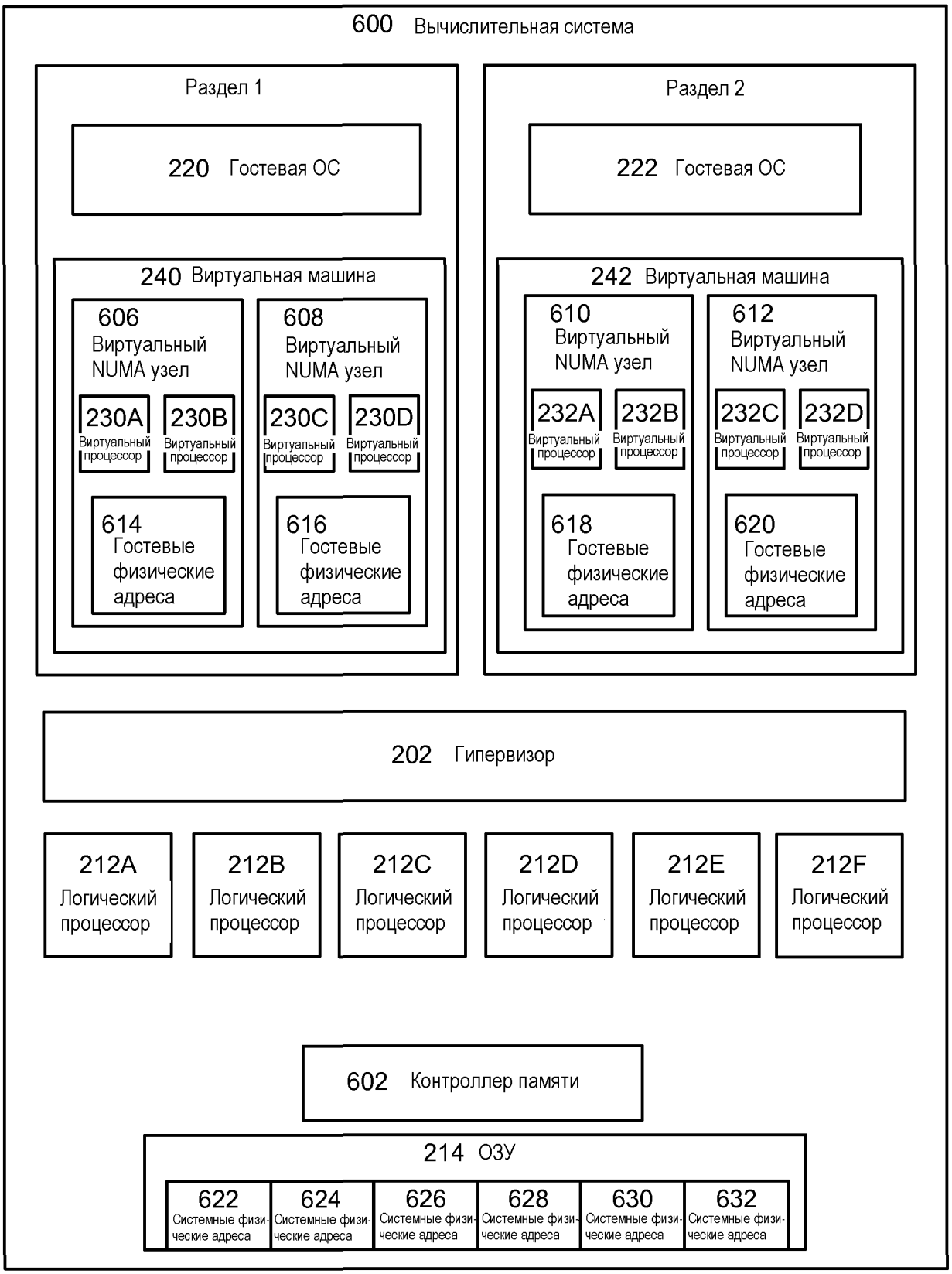
Фиг. 3


Гостевой физический адрес
1
2
3
4
1
2
3
4
5
6
7
Системный физический адрес

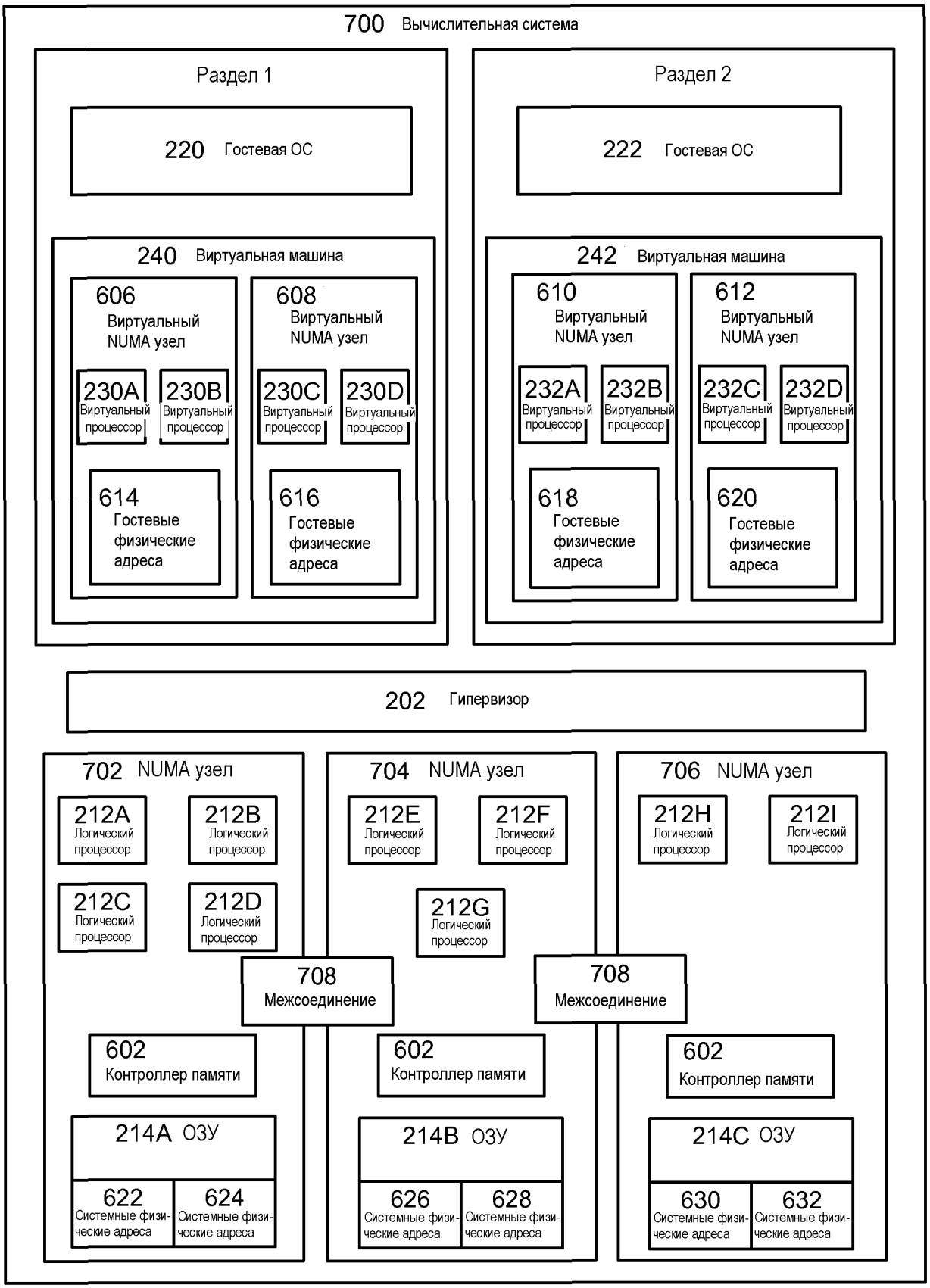
Фиг. 4


500 Дата-центр
502 Система управления
Компьютер 504
NUMA узел
NUMA узел
NUMA узел
NUMA узел
NUMA узел
NUMA узел
NUMA узел
NUMA узел
NUMA узел
Компьютер 506
Топология симметричной многопроцессорной системы
Компьютер 508
NUMA узел
NUMA узел
NUMA узел
NUMA узел
Компьютер 510
NUMA узел
NUMA узел
NUMA узел
NUMA узел
NUMA узел
NUMA узел

Фиг. 5


600 Вычислительная система
Раздел 1
220 Гостевая ОС
240 Виртуальная машина
606 Виртуальный NUMA узел
230A Виртуальный процессор
230B Виртуальный процессор
614 Гостевые физические адреса
608 Виртуальный NUMA узел
230C Виртуальный процессор
230D Виртуальный процессор
616 Гостевые физические адреса
Раздел 2
222 Гостевая ОС
242 Виртуальная машина
610 Виртуальный NUMA узел
232A Виртуальный процессор
232B Виртуальный процессор
618 Гостевые физические адреса
612 Виртуальный NUMA узел
232C Виртуальный процессор
232D Виртуальный процессор
620 Гостевые физические адреса
202 Гипервизор
212A Логический процессор
212B Логический процессор
212C Логический процессор
212D Логический процессор
212E Логический процессор
212F Логический процессор
602 Контроллер памяти
214 ОЗУ
622 Системные физические адреса
624 Системные физические адреса
626 Системные физические адреса
628 Системные физические адреса
630 Системные физические адреса
632 Системные физические адреса

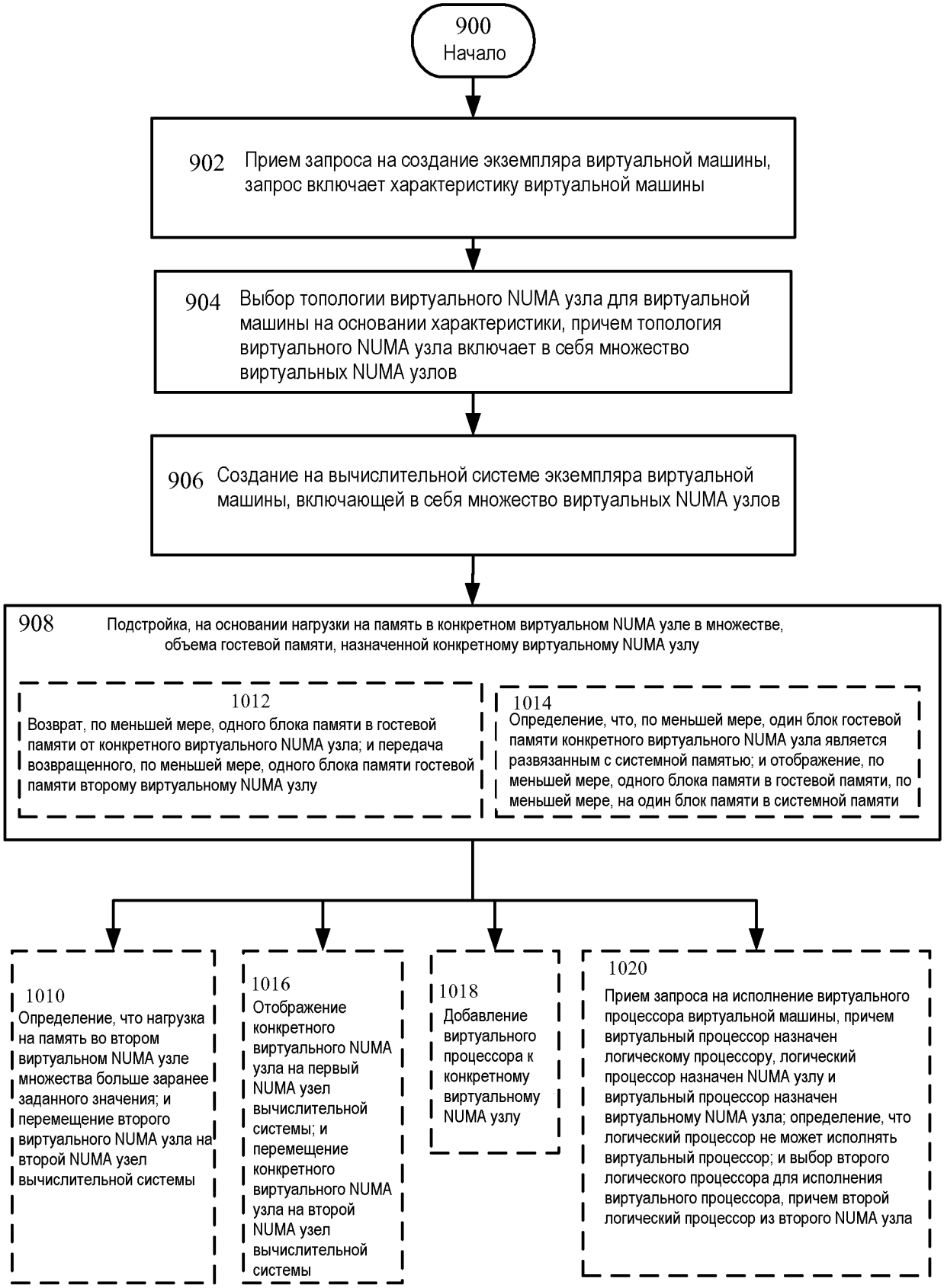
Фиг. 6


700 Вычислительная система
Раздел 1
220 Гостевая ОС
240 Виртуальная машина
606 Виртуальный NUMA узел
230A Виртуальный процессор
230B Виртуальный процессор
614 Гостевые физические адреса
608 Виртуальный NUMA узел
230C Виртуальный процессор
230D Виртуальный процессор
616 Гостевые физические адреса
Раздел 2
222 Гостевая ОС
242 Виртуальная машина
610 Виртуальный NUMA узел
232A Виртуальный процессор
232B Виртуальный процессор
618 Гостевые физические адреса
612 Виртуальный NUMA узел
232C Виртуальный процессор
232D Виртуальный процессор
620 Гостевые физические адреса
202 Гипервизор
702 NUMA узел
212A Логический процессор
212B Логический процессор
212C Логический процессор
212D Логический процессор
704 NUMA узел
212E Логический процессор
212F Логический процессор
212G Логический процессор
706 NUMA узел
212H Логический процессор
212I Логический процессор
708 Межсоединение
708 Межсоединение
602 Контроллер памяти
602 Контроллер памяти
602 Контроллер памяти
214A ОЗУ
622 Системные физические адреса
624 Системные физические адреса
214B ОЗУ
626 Системные физические адреса
628 Системные физические адреса
214C ОЗУ
630 Системные физические адреса
632 Системные физические адреса

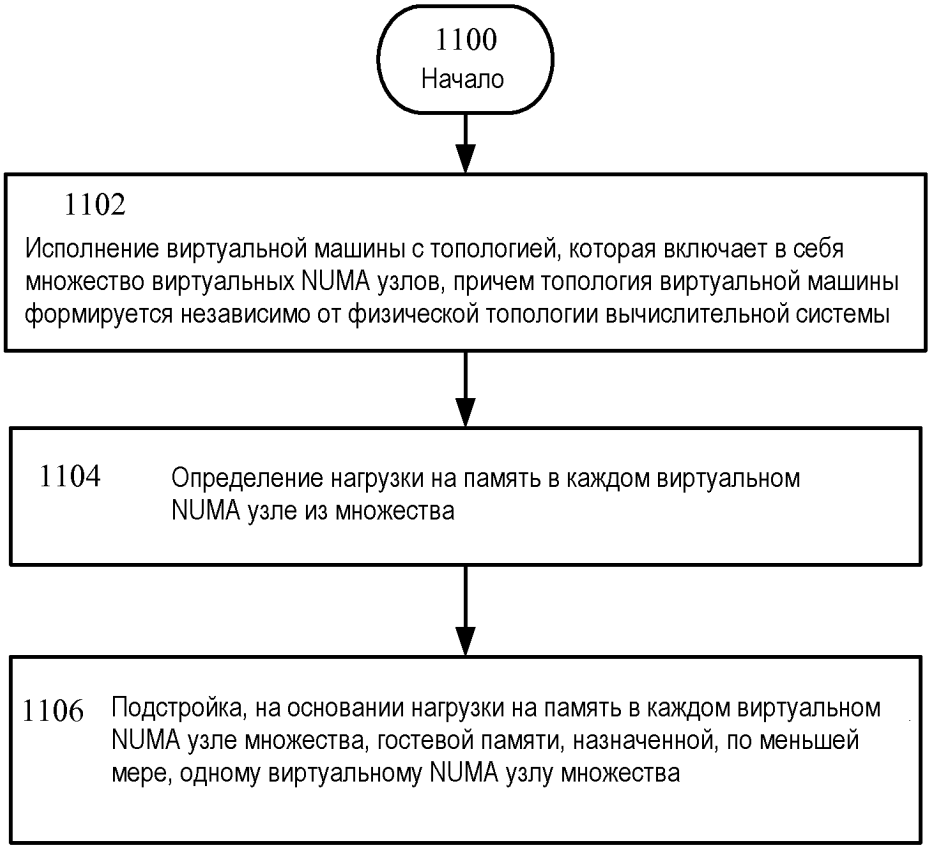
Фиг. 7


900
Начало
902
Прием запроса на создание экземпляра виртуальной машины, запрос включает в себя характеристику виртуальной машины
904
Выбор топологии виртуального NUMA узла для виртуальной машины на основании характеристики, причем топология виртуального NUMA узла включает в себя множество виртуальных NUMA узлов
906
Создание на вычислительной системе экземпляра виртуальной машины, включающей в себя состав множество виртуальных NUMA узлов
908
Подстройка, на основании нагрузки на память в конкретном виртуальном NUMA узле в множестве, объема гостевой памяти, назначенной конкретному виртуальному NUMA узлу

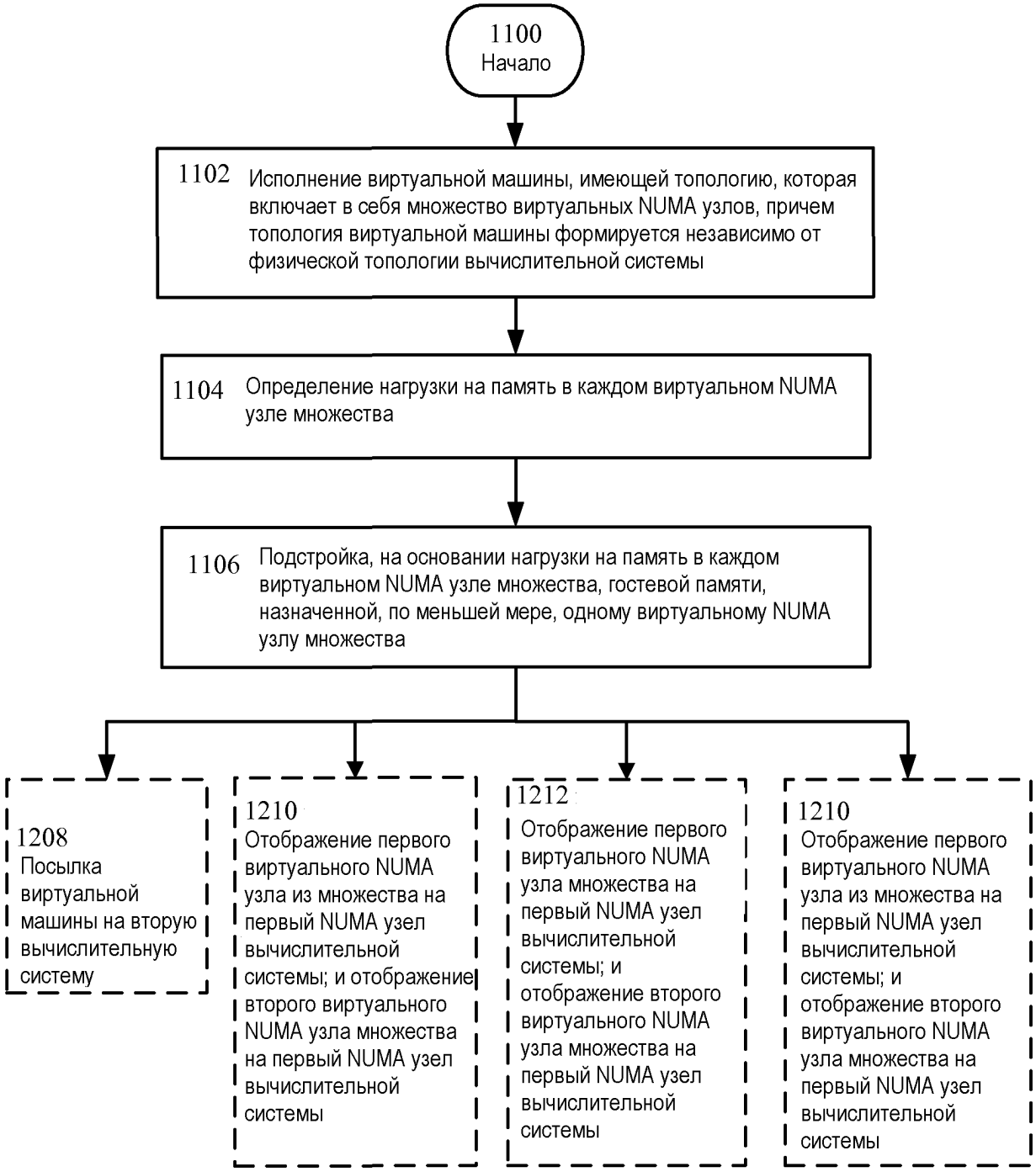
Фиг. 9


900
Начало
902 Прием запроса на создание экземпляра виртуальной машины, запрос включает характеристику виртуальной машины
904 Выбор топологии виртуального NUMA узла для виртуальной машины на основании характеристики, причем топология виртуального NUMA узла включает в себя множество виртуальных NUMA узлов
906 Создание на вычислительной системе экземпляра виртуальной машины, включающей в себя множество виртуальных NUMA узлов
908 Подстройка, на основании нагрузки на память в конкретном виртуальном NUMA узле в множестве, объема гостевой памяти, назначенной конкретному виртуальному NUMA узлу
1012 Возврат, по меньшей мере, одного блока памяти в гостевой памяти от конкретного виртуального NUMA узла; и передача возвращенного, по меньшей мере, одного блока памяти гостевой памяти второму виртуальному NUMA узлу
1014 Определение, что, по меньшей мере, один блок гостевой памяти конкретного виртуального NUMA узла является развязанным с системной памятью; и отображение, по меньшей мере, одного блока памяти в гостевой памяти, по меньшей мере, на один блок памяти в системной памяти
1010 Определение, что нагрузка на память во втором виртуальном NUMA узле множества больше заранее заданного значения; и перемещение второго виртуального NUMA узла на второй NUMA узел вычислительной системы
1016 Отображение конкретного виртуального NUMA узла на первый NUMA узел вычислительной системы; и перемещение конкретного виртуального NUMA узла на второй NUMA узел вычислительной системы
1018 Добавление виртуального процессора к конкретному виртуальному NUMA узлу
1020 Прием запроса на исполнение виртуального процессора виртуальной машины, причем виртуальный процессор назначен логическому процессору, логический процессор назначен NUMA узлу и виртуальный процессор назначен виртуальному NUMA узла; определение, что логический процессор не может исполнять виртуальный процессор; и выбор второго логического процессора для исполнения виртуального процессора, причем второй логический процессор из второго NUMA узла

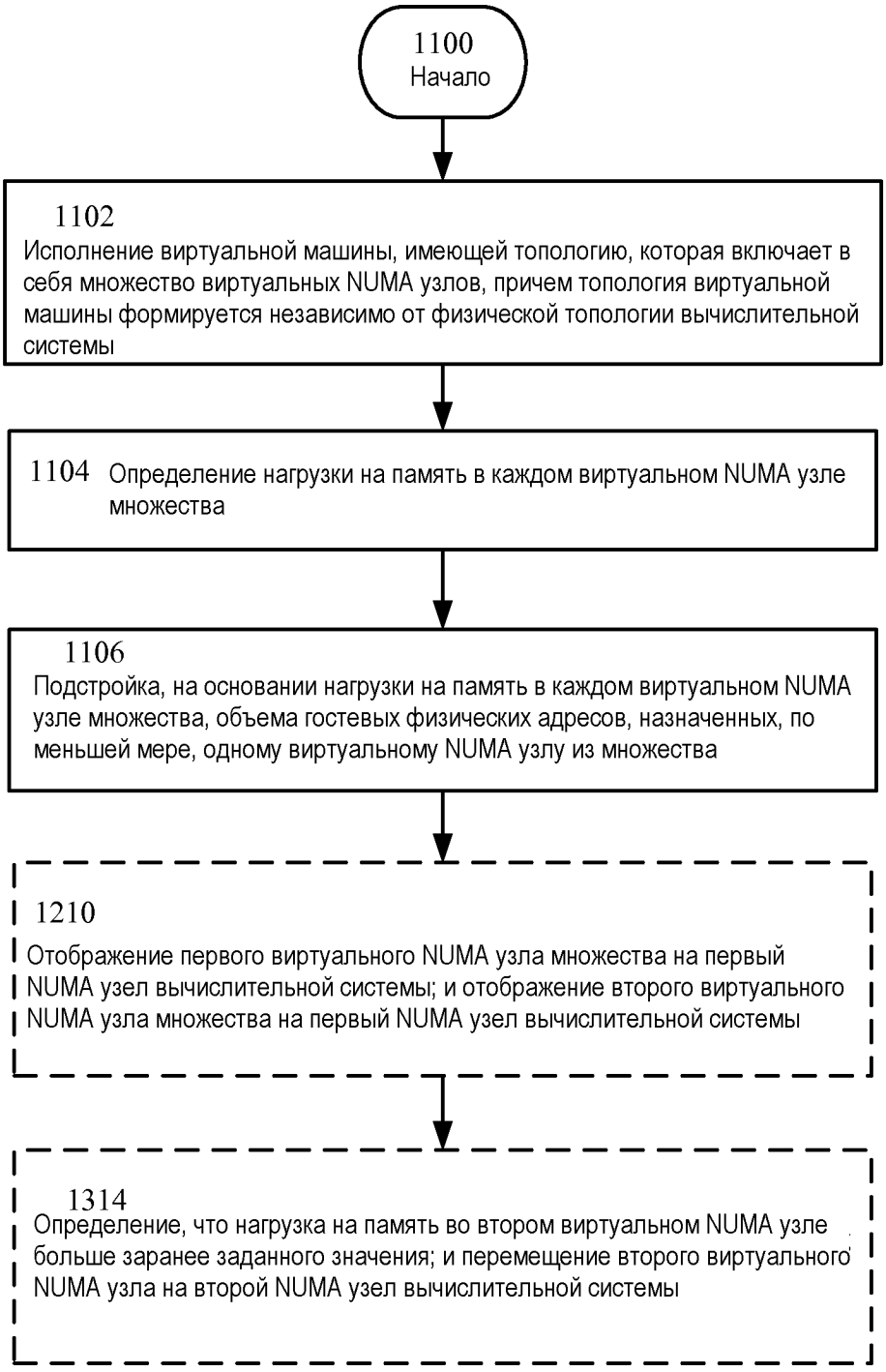
Фиг. 10


1100
Начало
1102
Исполнение виртуальной машины с топологией, которая включает в себя множество виртуальных NUMA узлов, причем топология виртуальной машины формируется независимо от физической топологии вычислительной системы
1104
Определение нагрузки на память в каждом виртуальном NUMA узле из множества
1106
Подстройка, на основании нагрузки на память в каждом виртуальном NUMA узле множества, гостевой памяти, назначенной, по меньшей мере, одному виртуальному NUMA узлу множества

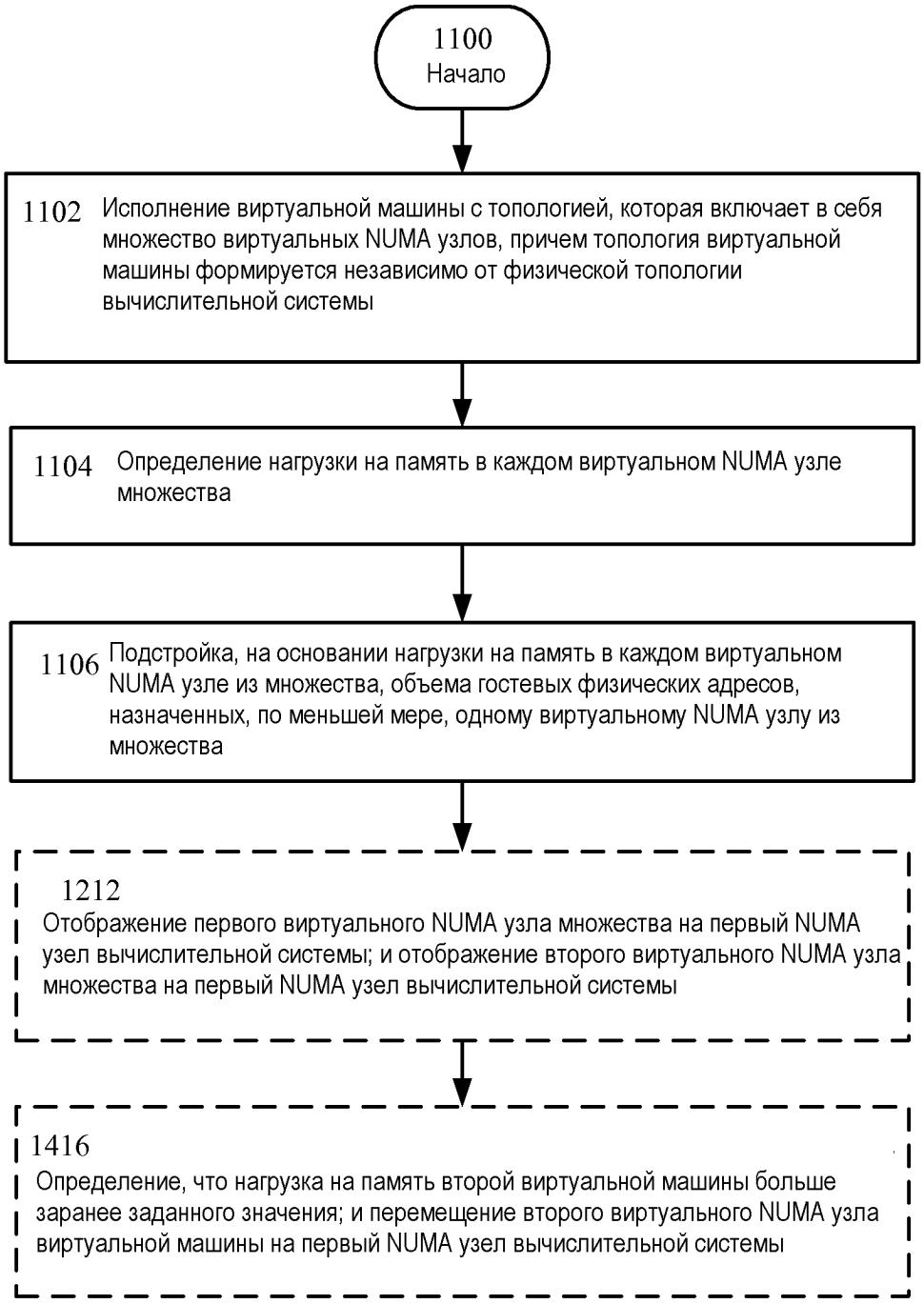
Фиг. 11


1100
Начало
1102
Исполнение виртуальной машины, имеющей топологию, которая включает в себя множество виртуальных NUMA узлов, причем топология виртуальной машины формируется независимо от физической топологии вычислительной системы
1104
Определение нагрузки на память в каждом виртуальном NUMA узле множества
1106
Подстройка, на основании нагрузки на память в каждом виртуальном NUMA узле множества, гостевой памяти, назначенной, по меньшей мере, одному виртуальному NUMA узлу множества
1208
Посылка виртуальной машины на вторую вычислительную систему
1210
Отображение первого виртуального NUMA узла из множества на первый NUMA узел вычислительной системы; и отображение второго виртуального NUMA узла множества на первый NUMA узел вычислительной системы
1212
Отображение первого виртуального NUMA узла множества на первый NUMA узел вычислительной системы; и отображение второго виртуального NUMA узла множества на первый NUMA узел вычислительной системы
1210
Отображение первого виртуального NUMA узла из множества на первый NUMA узел вычислительной системы; и отображение второго виртуального NUMA узла множества на первый NUMA узел вычислительной системы

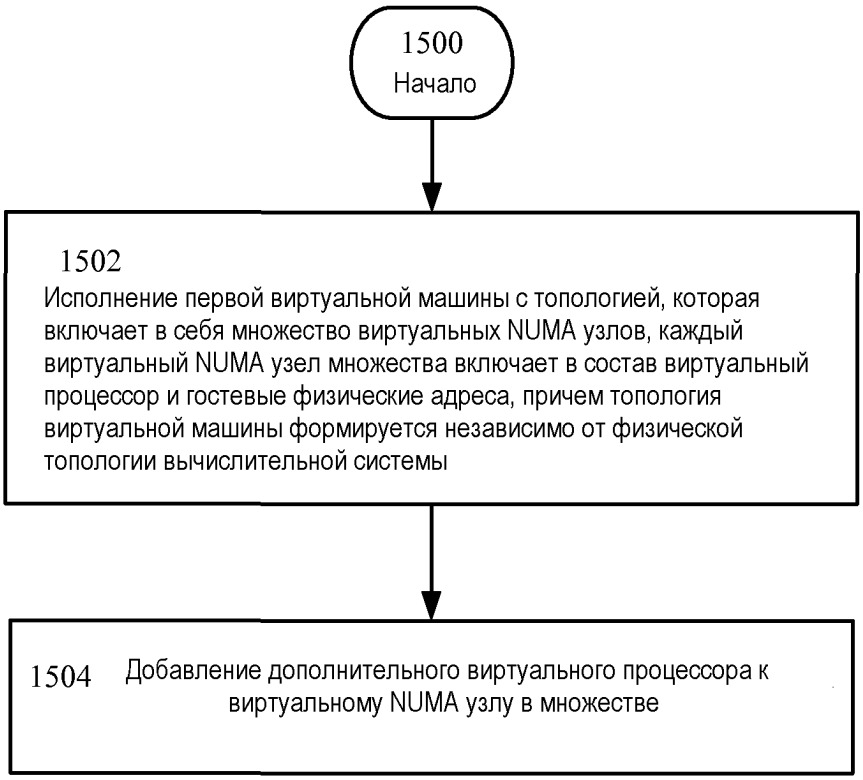
Фиг. 12


1100
Начало
1102
Исполнение виртуальной машины, имеющей топологию, которая включает в себя множество виртуальных NUMA узлов, причем топология виртуальной машины формируется независимо от физической топологии вычислительной системы
1104 Определение нагрузки на память в каждом виртуальном NUMA узле множества
1106
Подстройка, на основании нагрузки на память в каждом виртуальном NUMA узле множества, объема гостевых физических адресов, назначенных, по меньшей мере, одному виртуальному NUMA узлу из множества
1210
Отображение первого виртуального NUMA узла множества на первый NUMA узел вычислительной системы; и отображение второго виртуального NUMA узла множества на первый NUMA узел вычислительной системы
1314
Определение, что нагрузка на память во втором виртуальном NUMA узле больше заранее заданного значения; и перемещение второго виртуального NUMA узла на второй NUMA узел вычислительной системы

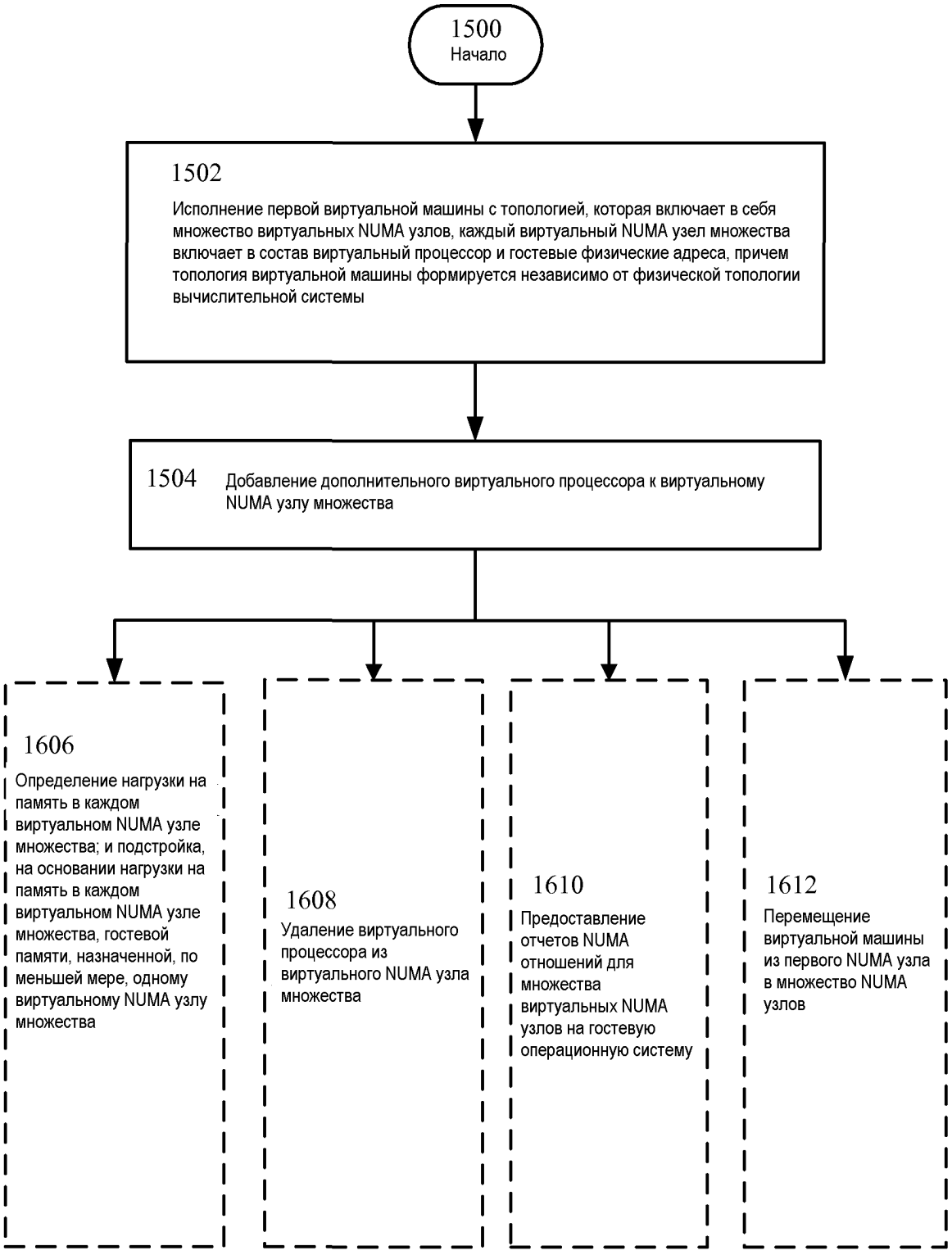
Фиг. 13


1100
Начало
1102
Исполнение виртуальной машины с топологией, которая включает в себя множество виртуальных NUMA узлов, причем топология виртуальной машины формируется независимо от физической топологии вычислительной системы
1104
Определение нагрузки на память в каждом виртуальном NUMA узле множества
1106
Подстройка, на основании нагрузки на память в каждом виртуальном NUMA узле из множества, объема гостевых физических адресов, назначенных, по меньшей мере, одному виртуальному NUMA узлу из множества
1212
Отображение первого виртуального NUMA узла множества на первый NUMA узел вычислительной системы; и отображение второго виртуального NUMA узла множества на первый NUMA узел вычислительной системы
1416
Определение, что нагрузка на память второй виртуальной машины больше заранее заданного значения; и перемещение второго виртуального NUMA узла виртуальной машины на первый NUMA узел вычислительной системы

Фиг. 14


1500
Начало
1502
Исполнение первой виртуальной машины с топологией, которая включает в себя множество виртуальных NUMA узлов, каждый виртуальный NUMA узел множества включает в состав виртуальный процессор и гостевые физические адреса, причем топология виртуальной машины формируется независимо от физической топологии вычислительной системы
1504
Добавление дополнительного виртуального процессора к виртуальному NUMA узлу в множестве

Фиг. 15


1500
Начало
1502
Исполнение первой виртуальной машины с топологией, которая включает в себя множество виртуальных NUMA узлов, каждый виртуальный NUMA узел множества включает в состав виртуальный процессор и гостевые физические адреса, причем топология виртуальной машины формируется независимо от физической топологии вычислительной системы
1504
Добавление дополнительного виртуального процессора к виртуальному NUMA узлу множества
1606
Определение нагрузки на память в каждом виртуальном NUMA узле множества; и подстройка, на основании нагрузки на память в каждом виртуальном NUMA узле множества, гостевой памяти, назначенной, по меньшей мере, одному виртуальному NUMA узлу множества
1608
Удаление виртуального процессора из виртуального NUMA узла множества
1610
Предоставление отчетов NUMA отношений для множества виртуальных NUMA узлов на гостевую операционную систему
1612
Перемещение виртуальной машины из первого NUMA узла в множество NUMA узлов

Фиг. 16


1500
Начало
1502
Исполнение первой виртуальной машины с топологией, которая включает в себя множество виртуальных NUMA узлов, каждый виртуальный NUMA узел множества включает в состав виртуальный процессор и гостевые физические адреса, причем топология виртуальной машины формируется независимо от физической топологии вычислительной системы
1504
Добавление дополнительного виртуального процессора к виртуальному NUMA узлу множества
1606
Определение нагрузки на память в каждом виртуальном NUMA узле множества; и подстройка, на основании нагрузки на память в каждом виртуальном NUMA узле множества, гостевой памяти, назначенной, по меньшей мере, одному виртуальному NUMA узлу множества
1714
Возврат памяти от первого виртуального NUMA узла множества на основании определения, что текущая нагрузка на память виртуального NUMA узла ниже целевой пороговой величины
1716
Передача памяти первому виртуальному NUMA узлу множества на основании определения, что текущая нагрузка на память гостевой операционной системы больше целевой пороговой величины

Фиг. 17