РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 435 200** (13) **C2**

(51) МПК
G06F 9/24 (2006.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ,
ПАТЕНТАМ И ТОВАРНЫМ ЗНАКАМ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21)(22) Заявка: **2008113217/08, 03.10.2006**

(24) Дата начала отсчета срока действия патента:
03.10.2006

Приоритет(ы):
(30) Конвенционный приоритет:
06.10.2005 US 11/245,629

(43) Дата публикации заявки: **10.10.2009** Бюл. № 28

(45) Опубликовано: **27.11.2011** Бюл. № **33**

(56) Список документов, цитированных в отчете о поиске: **US 2003/0233534 A1, 18.12.2003. US 2004/0003223 A1, 01.01.2004. US 2004/0153694 A1, 05.08.2004. RU 2166791 C1, 10.05.2001.**

(85) Дата начала рассмотрения заявки PCT на национальной фазе: **04.04.2008**

(86) Заявка PCT:
US 2006/039127 (03.10.2006)

(87) Публикация заявки PCT:
WO 2007/044516 (19.04.2007)

Адрес для переписки:
129090, Москва, ул. Б.Спасская, 25, стр.3, ООО "Юридическая фирма Городисский и Партнеры", пат.пов. А.В.Мищу, рег.№ 364

(72) Автор(ы):
ТСАНГ Майкл Х. (US)

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

RU 2435200 C2

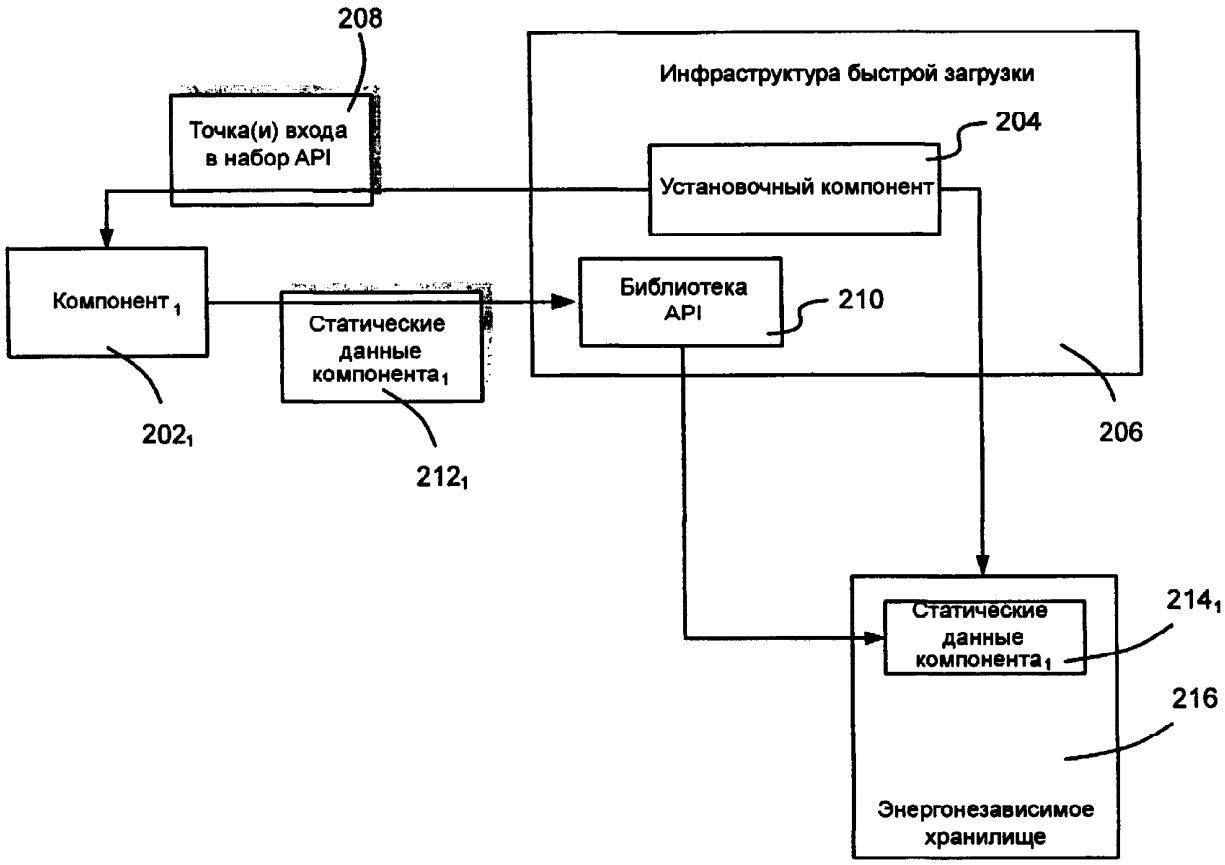
RU 2435200 C2

(54) БЫСТРАЯ ЗАГРУЗКА ОПЕРАЦИОННОЙ СИСТЕМЫ ИЗ ВЫКЛЮЧЕННОГО СОСТОЯНИЯ

(57) Реферат:

Изобретение относится к способам загрузки компьютерных систем. Техническим результатом является уменьшение времени загрузки компьютерной системы. Описан механизм быстрой загрузки, который, в целом, действует путем сохранения статических данных и/или кода для компонента системы, и, далее, путем предоставления компоненту системы доступа к статическим данным и/или коду в течение последующей загрузки устройства. Статические данные и/или код одного или более компонентов могут быть переведены из энергонезависимой памяти в энергозависимую память, благодаря чему при последующих перезагрузках этим компонентам не требуется повторно вычислять их сохраненные статические данные. Инфраструктура быстрой загрузки может включать в себя набор интерфейсов, и она предоставляет первый механизм, который сохраняет статические данные и/или код для компонента системы, и второй механизм, который предоставляет компоненту системы

доступ к статическим данным и/или коду. Инфраструктура быстрой загрузки может также предоставлять компоненту способ для аннулирования статических данных и/или кода, а также фоновый механизм, который собирает статические данные и/или код от компонента системы. 3 н. и 17 з.п. ф-лы, 7 ил.



ФИГ. 2

RU 2435200 C2

RU 2435200 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY,
PATENTS AND TRADEMARKS

(19) **RU** (11) **2 435 200** (13) **C2**

(51) Int. Cl.
G06F 9/24 (2006.01)

(12) ABSTRACT OF INVENTION

(21)(22) Application: **2008113217/08, 03.10.2006**

(24) Effective date for property rights:
03.10.2006

Priority:
(30) Priority:
06.10.2005 US 11/245,629

(43) Application published: **10.10.2009** Bull. 28

(45) Date of publication: **27.11.2011 Bull. 33**

(85) Commencement of national phase: **04.04.2008**

(86) PCT application:
US 2006/039127 (03.10.2006)

(87) PCT publication:
WO 2007/044516 (19.04.2007)

Mail address:
129090, Moskva, ul. B.Spasskaja, 25, str.3, OOO "Juridicheskaja firma Gorodisskij i Partnery", pat.pov. A.V.Mitsu, reg.№ 364

(72) Inventor(s):
TSANG Majkl Kh. (US)

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

RU 2435200 C2

RU 2435200 C2

(54) FAST BOOTING OPERATING SYSTEM FROM OFF STATE

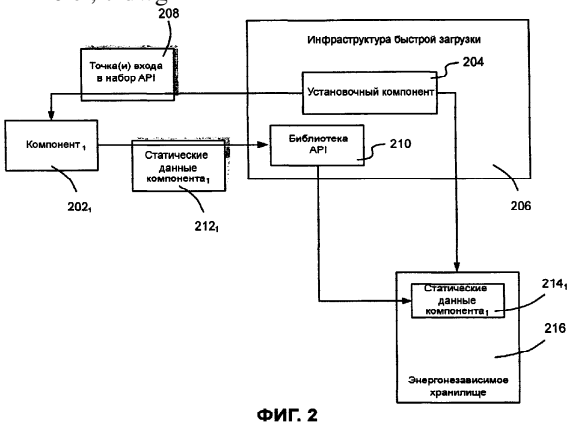
(57) Abstract:

FIELD: information technology.

SUBSTANCE: described is a fast boot mechanism which generally operates by storing static data and/or code for a system component, and then providing the system component with access to the static data and/or code during a subsequent device boot. Static data and/or code of one or more components may be transferred from non-volatile memory to volatile memory, whereby subsequent reboots need not have the components re-compute their stored static data. A fast boot infrastructure may include an interface set, and provides a first mechanism which stores static data and/or code for a system component, and a second mechanism which provides the system component with access to the static data and/ or code. The fast boot infrastructure may also provide the component with a way to invalidate static data and/or code, along with a background mechanism that collects static data and/or code from the system component.

EFFECT: faster booting of a computer system.

20 cl, 7 dwg



ФИГ. 2

Уровень техники

В связи с популяризацией мобильных устройств, увеличивается потребность в более длительном времени автономной работы и мгновенной доступности. Принимая во внимание сложность полноценной операционной системы, такой как операционная система семейства Microsoft Windows®, удовлетворение этой потребности вызывает значительные проблемы. Например, типичный мобильный персональный компьютер загружается из выключенного состояния (например, соответствующего состоянию S5 согласно стандарту ACPI (Advanced Configuration and Power Interface), примерно за двадцать-сорок секунд, что, безусловно, нельзя рассматривать как моментально доступное устройство.

Было предпринято множество попыток реализации различных механизмов для более быстрого приведения компьютерной системы обратно в рабочее состояние. Например, согласно стандарту ACPI было определено состояние ожидания S3/Standby, в котором состояние системы, по существу, фиксируется в системной памяти, посредством чего достигается соответствующая характеристика мгновенного включения. Тем не менее, состояние S3 расходует заряд батареи, что является проблемой для многих пользователей и производителей, и, по меньшей мере, вследствие этой причины использование режима S3 является недостаточным решением для многих пользователей. Также среди некоторых пользователей и производителей есть озабоченность тем, что состояние S3 может деградировать в течение времени, что, вероятно, имеет место из-за различных драйверов и служб, которые получаются из различных источников разработки и которые могут образовывать критический путь возобновления кода.

Также определено состояние ожидания S4/Hibernate, которое, в общем, переносит содержимое системой памяти в файл на жестком диске, чтобы предоставить возможность системе возобновить работу несколько быстрее, между тем сохраняя данные в случае сбоя питания. Преимущество состояния S4 заключается в том, что не расходуется энергия, но этому состоянию присущи те же проблемы стабильности и потенциальной потери данных, что и в случае состояния S3. Сверх того, возобновление работы из состояния S4 в среднем занимает приблизительно пятнадцать секунд, что также не может рассматриваться решением "мгновенного включения".

Даже для систем настольных компьютеров, которые не зависят от батарей, функция быстрого включения становится очень востребованной. Например, поскольку многие компьютерные системы развиваются в направлении предоставления развлекательных услуг, в частности, домашние компьютерные системы, увеличивается необходимость в компьютерных устройствах, которые действуют больше как потребительские электронные товары, чтобы было возможно продавать компьютерные продукты на конкурентном рынке компьютерных изделий и потребительской электроники. Например, в отличие от вышеупомянутого периода в двадцать-сорок секунд, необходимого для последовательности холодной загрузки, обычные потребительские электронные устройства могут быть включены и приведены в рабочее состояние в течение нескольких секунд.

В целом, для компьютерных систем желательна более быстрая загрузка из полностью выключенного состояния. Такая быстрая загрузка должна быть независима от других типов (например, S3 и S4) операций возобновления работы и дополнять их, поскольку такие спящие состояния сохраняют состояния системы, которые важны в других пользовательских сценариях.

Раскрытие изобретения

Различные аспекты настоящего изобретения нацелены на механизм быстрой загрузки, который, в целом, действует путем сохранения статических данных для компонента системы, и, далее, предоставления компоненту системы доступа к статическим данным в течение последующей загрузки устройства. Например, после установки или инсталляции образ быстрой загрузки, содержащий статические данные и, возможно, код одного или более компонентов, может быть переписан из энергонезависимой памяти в энергозависимую память (например, ОЗУ). Например, размещение ветви кода загрузки в энергонезависимой памяти имеет преимущество, заключающееся в пропуске относительно длительного времени раскрутки вращающегося носителя, как например, в случае жесткого диска, для которого типичное время раскрутки может составлять примерно пять секунд. При последующих перезагрузках, каждому компоненту системы, который сохранил статические данные, предоставляется доступ к его соответствующим статическим данным, благодаря чему большая часть данных не нуждается в повторном перечислении во время каждой загрузки, что делает последующие перезагрузки более быстрыми.

Проиллюстрирована инфраструктура быстрой загрузки, включающая в себя первый механизм, который сохраняет статические данные и/или код для компонента системы в первый момент времени, и второй механизм, который в течение загрузки системы устройства предоставляет компоненту системы доступ к статическим данным во второй момент времени, который следует за первым моментом времени. Инфраструктура быстрой загрузки может включать в себя набор интерфейсов, посредством которого компонент (клиент) системы может сохранять статические данные и/или код, получать доступ к статическим данным и аннулировать статические данные (например, в случае их изменения). Инфраструктура быстрой загрузки может также использовать фоновый механизм, который собирает статические данные и/или код от компонента системы, например, после загрузки системы.

Другие преимущества будут очевидны из следующего осуществления изобретения и прилагаемых чертежей.

Краткое описание чертежей

Настоящее изобретение проиллюстрировано в качестве примера в прилагаемых чертежах, в которых одинаковые ссылочные номера обозначают схожие элементы и в которых:

Фиг.1 - иллюстративный пример вычислительного окружения общего назначения, в котором могут быть внедрены различные аспекты настоящего изобретения.

Фиг.2 - структурная схема, иллюстрирующая пример архитектуры для сохранения статических данных, например, во время установки компонента системы.

Фиг.3 - структурная схема, иллюстрирующая пример архитектуры для предоставления компоненту системы доступа к его статическим данным, например, во время операции загрузки.

Фиг.4A и 4B - структурные схемы, иллюстрирующие пример архитектуры для аннулирования статических данных компонента при изменении и для изменения статических данных после изменения, соответственно.

Фиг.5 - структурная схема, иллюстрирующая пример архитектуры для получения статических данных компонента системы в течение фоновой операции.

Фиг.6 - структурная схема, иллюстрирующая пример процесса быстрой загрузки с жесткого диска или энергонезависимого ОЗУ.

Осуществление изобретения

Пример рабочего окружения

Фиг.1 иллюстрирует пример подходящего вычислительного окружения 100 системы, в которой может быть осуществлено настоящее изобретение. Вычислительное окружение 100 системы является лишь одним примером подходящего вычислительного окружения, и оно не предназначено для определения ограничений объема использования или функциональных возможностей настоящего изобретения. Кроме того, вычислительное окружение 100 не должно быть интерпретировано как имеющее зависимость или требования, относящиеся к какому-либо компоненту или комбинациям компонентов, проиллюстрированных в примере рабочего окружения 100.

Изобретение может быть реализовано со множеством других окружений или конфигураций вычислительной системы общего назначения или специального назначения. Примеры известных вычислительных систем, окружений и/или конфигураций, которые могут быть подходящими для использования с настоящим изобретением, включают в себя, но не ограничиваются перечисленным: персональные компьютеры, серверные компьютеры, карманные или портативные устройства, планшетные устройства, многопроцессорные системы, системы на микропроцессорах, телевизионные приставки, программируемая потребительская электроника, сетевые персональные компьютеры, миникомпьютеры, мэйнфреймы, распределенные вычислительные окружения, которые включают в себя любые из упомянутых систем или устройств, и т.п.

Настоящее изобретение может быть описано в общем контексте выполняемых компьютером команд, таких как программные модули, которые выполняются компьютером. В общем, программные модули включают в себя процедуры, программы, объекты, компоненты, структуры данных и т.п., которые выполняют конкретные задачи или осуществляют конкретные абстрактные типы данных. Настоящее изобретение может быть применено в распределенных вычислительных окружениях, где задачи выполняются посредством удаленных устройств обработки, которые объединены через сеть связи. В распределенном вычислительном окружении программные модули могут быть расположены как в среде хранения локального компьютера, так и в среде хранения удаленного компьютера, включая устройства памяти.

Ссылаясь на Фиг.1, пример системы для осуществления настоящего изобретения включает в себя вычислительное устройство общего назначения в форме компьютера 110. Компоненты компьютера 110 могут включать в себя, но не ограничены перечисленным, процессор 120, системную память 130 и системную шину 121, которая соединяет различные компоненты системы (в том числе, соединяет системную память) с процессором 120. Системная шина 121 может быть любого типа из ряда типов структур шин, включающих в себя шину памяти или контроллер памяти, периферийную шину и локальную шину, используя любую архитектуру из разнообразия архитектур шин. В качестве примера и не ограничиваясь перечисленным, подобные архитектуры включают в себя шину стандарта Industry Standard Architecture (ISA), шину стандарта Micro Channel Architecture (MCA), шину стандарта Enhanced ISA (EISA), локальную шину стандарта Video Electronics Standards Association (VESA) и шину стандарта Peripheral Component Interconnect (PCI), также известную как шина расширения.

Компьютер 110, как правило, включает в себя ряд машиночитаемых носителей.

Машиночитаемые носители могут представлять собой любые доступные средства, к которым компьютер 110 может выполнить доступ, и они включают в себя как энергозависимые, так и энергонезависимые носители, съемные и несъемные носители. В качестве примера, но не ограничиваясь перечисленным, машиночитаемые носители могут содержать компьютерные средства хранения и средства связи. Компьютерное средство хранения включает в себя энергозависимое, энергонезависимое, съемное и несъемное средство, реализованное посредством какого-либо способа или технологии для хранения информации, такой как машиночитаемые команды, структуры данных, программные модули и другие данные. Компьютерное средство хранения включает в себя, но не ограничено перечисленным, ОЗУ, ПЗУ, ЭСППЗУ, флэш-память или другую технологию памяти, диски CD-ROM, цифровые универсальные диски (DVD) или иные оптические дисковые носители, магнитные кассеты, магнитные ленты, магнитные дисковые носители или другие магнитные запоминающие устройства, или любое другое средство, которое может быть использовано, чтобы хранить желаемую информацию, и к которой может быть выполнен доступ компьютером 110. Средство связи, как правило, заключает в себя машиночитаемые инструкции, структуры данных, программные модули или другие данные в виде модулированного сигнала данных, такого как несущая волна или другой механизм передачи, и включает в себя любое средство доставки информации. Термин "модулированный сигнал данных" означает сигнал, у которого одна или более характеристик установлены или изменены таким образом, чтобы кодировать в сигнал информацию. В качестве примера, но не ограничиваясь перечисленным, средство связи включает в себя проводное средство, такое как проводная сеть или прямое проводное соединение, и беспроводное средство, такое как акустическое, радиочастотное, инфракрасное и другие беспроводные средства. Комбинации из каких-либо вышеперечисленных типов также входят в объем понятия машиночитаемый носитель.

Системная память 130 включает в себя компьютерное средство хранения в форме энергозависимой и/или энергонезависимой памяти, такой как ПЗУ 131 и ОЗУ 132. Базовая система 133 ввода/вывода (BIOS), содержащая базовые процедуры, которые помогают передавать информацию между элементами в компьютере 110, как например во время загрузки, как правило, хранится в ПЗУ 131. ОЗУ 132, как правило, содержит данные и/или программные модули, которые непосредственно доступны и/или задействованы процессором 120. В качестве примера, но не ограничиваясь этим, Фиг.1 иллюстрирует операционную систему 134, прикладные программы 135, другие программные модули 136 и программные данные 137.

Компьютер 110 может также включать в себя другое съемное/несъемное энергозависимое/энергонезависимое компьютерное средство хранения. Исключительно в качестве примера, Фиг.1 иллюстрирует привод 141 жесткого диска, который считывает с или записывает на несъемный, энергонезависимый магнитный носитель, привод 151 магнитного диска, который считывает с или записывает на съемный, энергонезависимый магнитный диск 152, и привод 155 оптического диска, который считывает с или записывает на съемный, энергонезависимый оптический диск 156, такой как CD-ROM или другой оптический носитель информации. Другие съемные/несъемные, энергозависимые/энергонезависимые компьютерные носители информации, которые могут быть использованы в примере рабочего окружения, включают в себя, но не ограничиваются перечисленным, кассеты с магнитной лентой, карты флэш-памяти, цифровые универсальные диски, цифровые видео ленты, твердотельные ОЗУ, твердотельные ПЗУ и т.п. Привод 141 жесткого диска, как

правило, соединен с системной шиной 121 через интерфейс несъемной памяти, такой как интерфейс 140, а привод 151 магнитного диска и привод 155 оптического диска, как правило, соединены с системой шиной 121 посредством интерфейса съемной памяти, такого как интерфейс 150.

Приводы и связанные с ними компьютерные запоминающие носители, описанные выше и проиллюстрированные на Фиг.1, предоставляют хранение машиночитаемых инструкций, структур данных, программных модулей и других данных для компьютера 110. На Фиг.1, например, привод 141 жесткого диска проиллюстрирован, как хранящий операционную систему 144, прикладные программы 145, другие программные модули 146 и программные данные 147. Следует отметить, что эти компоненты могут быть такими же, как операционная система 134, прикладные программы 135, другие программные модули 136 и программные данные 137 или же отличаться от них. Операционная система 144, прикладные программы 145, другие программные модули 146 и программные данные 147 обозначены различными номерами, чтобы проиллюстрировать, что, по меньшей мере, они представляют собой различные копии. Пользователь может вводить команды и информацию в компьютер 110 через устройства ввода, такие как планшет или электронный дигитайзер 164, микрофон 163, клавиатура 162 и указывающее устройство 161, такое как мышь, трекбол или сенсорная панель. Другие устройства ввода, которые не показаны на Фиг.1, могут включать в себя джойстик, игровой планшет, спутниковую антенну, сканер или т.п. Эти и другие устройства ввода часто соединяются с процессором 120 через интерфейс 160 ввода пользователя, который соединен с системной шиной, но они могут также быть соединены посредством другого интерфейса и структур шины, такой как параллельный порт, игровой порт или универсальная последовательная шина (USB). Монитор 191 или другой тип устройства отображения также соединен с системной шиной 121 посредством интерфейса, такого как видео интерфейс 190. Монитор 191 также может быть интегрирован с сенсорной панелью и т.п. Следует отметить, что монитор и/или сенсорная панель могут быть физически соединены с корпусом, в котором размещается вычислительное устройство 110, такое как персональный компьютер планшетного типа. В добавление, компьютеры, такие как вычислительное устройство 110, могут также включать в себя другие периферийные устройства вывода, такие как громкоговорители 195 и принтер 196, которые могут быть соединены через интерфейс 194 периферийных устройств вывода и т.п.

Компьютер 110 может работать в сетевом окружении, используя логические соединения с одним или более удаленными компьютерами, такими как удаленный компьютер 180. Удаленный компьютер 180 может быть персональным компьютером, сервером, маршрутизатором, сетевым персональным компьютером, устройством одноранговового узла или другим обычным сетевым узлом, и он, как правило, включает в себя многие или все элементы, описанные выше относительно компьютера 110, хотя на Фиг.1 проиллюстрировано только устройство 181 памяти. Логические соединения, изображенные на Фиг.1, включают в себя локальную сеть (Local Area Network, LAN) 171 и глобальную сеть (Wide Area Network, WAN) 173, но могут также включать в себя другие сети. Подобные сетевые окружения типичны для контор, компьютерных сетей масштаба предприятия, интранета и Интернета.

При использовании в сетевом окружении локальной сети компьютер 110 соединен с локальной сетью 171 через сетевой интерфейс или адаптер 170. При использовании в сетевом окружении глобальной сети компьютер 110, как правило, включает в себя

модем 172 или иное средство для установления связи через глобальную сеть 173, такую как Интернет. Модем 172, который может быть внутренним или внешним, может быть соединен с системной шиной 121 посредством интерфейса 160 ввода пользователя, или иного подходящего механизма. В сетевом окружении программные модули, изображенные относительно компьютера 110, или их части могут храниться в удаленном устройстве памяти. В качестве примера, но не ограничиваясь этим, Фиг.1 иллюстрирует удаленные прикладные программы 185 как находящиеся в устройстве 181 памяти. Очевидно, что показанные сетевые соединения представляют собой лишь примеры, и могут быть использованы другие средства для установления линии связи между компьютерами.

Быстрая загрузка

Различные аспекты описанной здесь технологии нацелены на технологию, которая быстро загружает вычислительное устройство из выключенного состояния, что также включает в себя быструю перезагрузку вычислительного устройства. В одном варианте осуществления часть технологии может считывать данные с твердотельного энергонезависимого устройства хранения типа ОЗУ, что выполняется быстрее, чем извлечение данных с диска, в частности, когда диску требуется раскрутиться из выключенного состояния. Тем не менее, как описано ниже, описанная здесь технология быстрой загрузки обычно также предоставляет более быструю загрузку (относительно существующих технологий) с жесткого диска. По существу, любой из упомянутых здесь примеров не является ограничивающим, и настоящее изобретение может быть использовано различными способами, которые в целом обеспечивают выгоды и преимущества в вычислении.

В общем, при каждой загрузке до состояния полностью работоспособной операционной системы вычислительное устройство каждый раз выполняет большую часть одной и той же инициализации. Некоторые из этих повторяемых инициализаций могут занимать относительно большое время. Описанный здесь механизм быстрой загрузки делает попытку уменьшить количество этих требующих много времени инициализаций путем выполнения, по меньшей мере, части этих инициализаций только один раз, например, при начальной установке вычислительной машины, и последующего сохранения получающихся в результате инициализированных состояний в энергонезависимой форме. Путем быстрого восстановления этих сохраненных состояний при последующих загрузках, вместо их повторного вычисления каждых раз, большая часть времени загрузки устраняется.

Следует отметить, что компоненты, используемые в течение загрузки, имеют как динамические данные, так и статические данные, причем в использованном здесь значении термин "статические данные" обозначает данные, которые обычно не меняются между загрузками, хотя иногда они могут изменяться. Очевидно, что сбор и восстановление данных состояния, которые являются динамическими, приведет к повреждению системы. Например, файловая система содержит метаданные, такие как битовые карты кластера, и когда система загружается и используется, создаются новые файлы и стираются старые файлы, вследствие чего ранее полученная битовая карта кластера будет некорректна для текущего состояния.

В общем, описанная здесь технология быстрой загрузки предоставляет возможность компонентам, которые действуют во время загрузки, сохранять и восстанавливать их статические данные, между тем позволяя им перечислять свои динамические данные, как и раньше. Поскольку отдельным компонентам известно, какие статические и динамические состояния они имеют, эти компоненты могут

идентифицировать свои статические состояния и выполнить динамическую инициализацию/перечисление данных динамического состояния, между тем просто восстанавливая свои данные статического состояния во время процедуры быстрой загрузки. Таким образом, устраняется большая часть повторяющейся инициализации, которая возникает во время последовательности медленной загрузки.

Некоторые компоненты могут быть унаследованными компонентами, которые написаны не для идентификации и/или сохранения статических данных, или они могут принять решение не участвовать в заданной схеме быстрой загрузки по другим причинам. Для предоставления возможности смешивания участвующих и не участвующих компонентов предоставлен механизм Интерфейса Прикладной Программы (Application Program Interface, API), так что путем вызова API участвовать будут только те компоненты, которым известно о быстрой загрузке, тогда как не участвующие унаследованные компоненты и т.п. проигнорируют предоставленную функциональную возможность.

Сверх того, поскольку компоненты могут иметь взаимозависимости, инициализация имеет место в нескольких фазах. Такая многофазная инициализация уже структурирована в операционных системах семейства Windows®. Например, в ядре NT, различные компоненты ядра (например, объект, безопасность, процесс, PnP, исполнение, ядро, управление памятью, регистр и т.п.) имеют точки входа инициализации, которые вызываются на различных фазах инициализации. На каждой фазе инициализации компонент определяет, что нужно инициализировать, на основании зависимостей от инициализации других компонентов в предшествующих фазах.

Чтобы принять участие в быстрой загрузке, на каждой фазе инициализации компонент может определить, какие из его инициализаций занимают время и/или можно ли выполнить инициализацию единожды и сохранить результаты. Эта информация может быть предоставлена в инфраструктуру быстрой загрузки в различные моменты времени, например, когда компонент инсталлируется или когда операционная система компьютера и ее компоненты устанавливаются в первый раз. Для этой цели, инсталлятор устройства или т.п. перечисляет компоненты ядра для информации быстрой загрузки и вызывает менеджер памяти/компоненты файловой системы, чтобы сохранить эту информацию состояния в энергонезависимой форме.

Например, как в целом показано на Фиг.2, каждый компонент (например, 202_1) ядра может быть вызван установочным компонентом 204 (связанным с инсталлятором устройства или включающим в себя инсталлятор устройства) инфраструктуры 206 быстрой загрузки в конце установки операционной системы. В одном варианте осуществления вызов предоставляет компоненту 202_1 данные 208, идентифицирующие точку (или точки) входа, например, интерфейса прикладной программы (API), в библиотеку 210 API. Посредством этой точки входа компонент 202_1 может указать ячейки памяти, которые содержат код и/или статические состояния, которые могут быть сохранены, посредством чего эти ячейки памяти могут быть восстановлены при последующих сессиях быстрой загрузки. Альтернативно, компонент 202_1 может напрямую предоставить данные (например, в структуре данных, понятной компоненту 202_1) в ответ на такой вызов. Ячейки памяти и/или код или сами данные представлены на Фиг.2 блоками "статические данные", обозначенными номерами 212_1 (как передаваемые данные) и 214_1 (как сохраненные данные). Статический код и/или данные 214_1, содержащие образ быстрой загрузки, сохраняются в неком энергонезависимом хранилище 216, таком как

энергонезависимое ОЗУ, или, альтернативно, могут быть сохранены целиком или частично в средстве хранения другого типа, таком как жесткий диск.

После сохранения этих данных, при последующих загрузках, когда каждый компонент ядра вызывается для инициализации, например, на различных фазах, эти ячейки памяти (или сами данные) доступны для использования компонентом. В результате, по меньшей мере, определенные части инициализации, требующей много времени, каждого участвующего компонента могут быть пропущены. Следует отметить, что для не участвующих компонентов обычная инициализация не может быть пропущена, вследствие чего этот компонент потратит столько же времени, сколько он тратит при загрузке обычной длительности.

В одном варианте осуществления в различных компонентах предоставлены новые интерфейсы для компонентов ядра, а также новая поддерживающая инфраструктура в различных компонентах. Один интерфейс компонента ядра предоставлен для возможности вызова каждого компонента ядра, так что каждый упомянутый компонент может в ответ предоставить перечень блоков памяти, которые содержат код и/или данные (или сам код/данные). Эти блоки памяти будут сохранены в энергонезависимом хранилище 216, которое может содержать файл образа состояния быстрой загрузки на жестком диске в обычных вычислительных устройствах или в хранилище типа энергонезависимого ОЗУ или т.п. в будущих вычислительных устройствах.

Для построения образа состояния быстрой загрузки этот интерфейс для сохранения статических данных может быть вызван инсталлятором устройства, например, посредством инициации программой установки операционной системы. Этот интерфейс может быть вызван при установке нового компонента. В ответственности каждого участвующего компонента входит определение того, какие данные должны быть сохранены для сокращения времени загрузки при последующих быстрых загрузках. Например, при разработке каждый участвующий компонент может выполнить профилирование кода на своей ветви кода инициализации, чтобы идентифицировать узкое место загрузки.

Еще один интерфейс предоставлен, чтобы предоставить возможность компоненту получать перечень блоков памяти, которые компонент запросил сохранить, например, на каждой фазе. Например, этот интерфейс может соответствовать модифицированной версии существующего интерфейса инициализации фазы, уже присутствующего в операционных системах семейства Windows®. Способность извлекать статические данные состояния предоставляет возможность компоненту пропустить некоторую часть его инициализаций, которые требуют много времени. Также предоставлен поддерживающий интерфейс, посредством которого компонент может выяснить, выполняется ли в текущее время быстрая загрузка. Этот интерфейс предоставлен так, чтобы компоненту было известно, когда он должен использовать или не использовать кэшированную информацию из сохраненной информации состояния быстрой загрузки. Сверх того, компоненты ядра, такие как компонент менеджера памяти, и возможно другие компоненты, могут экспортировать новые полезные интерфейсы API, которые облегчают идентификацию и/или сохранение блоков памяти состояния быстрой загрузки. Например, существующие интерфейсы API распределения памяти могут быть расширены, чтобы обеспечить возможность маркирования распределений памяти, которые содержат информацию состояния быстрой загрузки.

Фиг.3 иллюстрирует пример, в котором сохраненные в энергонезависимом

хранилище данные используются при последующей операции быстрой загрузки. В общем, когда в течение быстрой загрузки вызывается участвующий компонент, этот компонент распознает флаг 318 быстрой загрузки (например, установленный в памяти посредством компонента 319 загрузчика для загрузки с жесткого диска или посредством BIOS для загрузки с энергонезависимого ОЗУ). Тогда, этот компонент (например, 202_1) вызывает библиотеку 210 API инфраструктуры 206 быстрой загрузки, чтобы принять свои данные состояния, предварительно кэшированные для текущей фазы загрузки. Следует отметить, что этот компонент может фактически принять указатель на свои статические данные, распакованные или иным образом переданные в ОЗУ 132 из энергонезависимого хранилища 216. Например, некоторые или все статические данные компонента могут быть сохранены как образ быстрой загрузки, который распаковывается или иным образом передается в ОЗУ 132 при запуске, и посредством вызова API каждому компоненту предоставляется туда доступ.

Каждый компонент (например, 202_1-202_n) вызывается в свою очередь, по одному на каждую фазу, и эти участвующие компоненты могут, таким образом, получить доступ к своим статическим данным и перечислить свои динамические данные. Это проиллюстрировано на Фиг.3 посредством данных 314_1 для фазы 1 компонента 202_1, к которым компонент 202_1 выполняет доступ в ОЗУ 132 при извлечении из энергонезависимого хранилища 216. Код 320 инфраструктуры быстрой загрузки может обработать упомянутое извлечение при вызове библиотеки 210 API. Те компоненты (например, унаследованные), которые не участвуют в схеме быстрой загрузки, не выполняют обратный вызов, также как и компоненты, которые не имеют данных, кэшированных для определенной фазы, хотя такие компоненты, которым известно о быстрой загрузке, могут альтернативно ответить "для этой фазы не требуется данных" или т.п.

Ссылаясь на Фиг.4А, в библиотеке 210 API предоставлен еще один интерфейс, посредством которого компонент (например, 202_1) может уведомить код 206 инфраструктуры быстрой загрузки аннулировать часть или весь сохраненный образ состояния, поскольку его (как правило) статическое состояние были изменено. Например, пользователь может изменить, как правило, статические данные компонента посредством действия в панели управления и/или регистре. Компонент 202_1 несет ответственность за обнаружение изменений его данных. В одном варианте осуществления аннулированные данные 430 не будут возвращены, вследствие чего при следующей загрузке компоненту 202_1 потребуется перечислить статические данные и динамические данные, как будто в операции медленной загрузки, и снова кэшировать действительные данные.

Альтернативно, как показано на Фиг.4В посредством новых статических данных 446_1 состояния, которые передаются и сохраняются (данные 448_1 состояния), аннулирование кэшированных данных быстрой загрузки может инициировать операцию повторного захвата состояния, например, посредством бесшумной/фоновой цепочки 560 (например, Фиг.5), которая, например, может действовать аналогично захвату состояния, выполняемому процессом установки операционной системы, как описано выше. В любом случае, когда возникает изменение, как правило, статических данных, по меньшей мере, измененная часть ранее кэшированных данных состояния больше не будет использоваться.

Подобная фоновая операция, в целом проиллюстрированная на Фиг.5, требуется не только тогда, когда компонент обнаруживает изменение, но и в других случаях. Например, часть текущего времени загрузки расходуется на задержку из-за поиска

устройств (например, жестких дисков IDE), которые могли быть добавлены пользователем. При опросе каждому такому устройству (в компьютер с двумя каналами IDE может быть установлено от одного до четырех таких устройств) дается примерно одна секунда на ответ. Однако внутренние устройства не часто добавляют или удаляют, и, соответственно, вместо опроса каждого устройства при загрузке, изменение может быть обнаружено посредством последующей фоновой операции 560.

Кроме того, возможно, что компонент не будет написан должным образом, и он не сможет корректно обнаруживать изменения своих статических данных. Для устранения нарушения целостности данных может использоваться фоновая цепочка/процесс 560 (Фиг.5), который время от времени запрашивает (например, по случайному или циклическому принципу), чтобы каждый компонент повторно перечислил свои статические данные. Сверх того, любой или все компоненты время от времени могут быть подвержены полному перечислению медленной загрузки вместо быстрой загрузки, и пользователь или иной механизм (например, утилита) может запросить выполнение медленной загрузки, если есть подозрение нарушения целостности статических данных, например, исходя из сбоя компонента при аннулировании измененных данных.

Как описано выше, информация состояния быстрой загрузки может храниться в некоторой форме твердотельного энергонезависимого хранилища, такого как энергонезависимое ОЗУ. Для таких устройств может быть установлен новый стандартизированный интерфейс BIOS, чтобы указывать, как BIOS будет извлекать и восстанавливать какую-либо необходимую информацию состояния быстрой загрузки из энергонезависимого ОЗУ в основную память, например, до того, как операционная система принимает управление.

Фиг.6 иллюстрирует пример различия ветви кода между системами, которые имеют образ состояния быстрой загрузки, сохраненный как файл на жестком диске, и новыми системами, которые имеют образ состояния быстрой загрузки, сохраненный в энергонезависимом ОЗУ. Следует отметить, что вычислительному устройству (например, его BIOS) будет известно, загружается ли оно с энергонезависимого ОЗУ или диска.

Этап 602 представляет загрузку с жесткого диска, которая начинается с загрузки и выполнения процесса загрузчика, например, NTLDR. Загрузчик проверяет действительный файл спящего режима на этапе 604, и если такой файл присутствует, то загрузчик возобновляет работу со спящего режима (режима S4 стандарта ACPI). При отсутствии такого файла, загрузчик переходит к этапу 608, где проверяет наличие сохраненного образа быстрой загрузки.

Если на этапе 608 такой образ не удается найти, то система выполняет медленную (нормальную) загрузку. Следует отметить, что все же возможно выполнить быструю загрузку в следующий раз, например, путем получения информации быстрой загрузки от участвующих компонентов в фоновой операции, как описано выше. Например, это может иметь место, если образ быстрой загрузки был удален по какой-то причине.

Если вместо этого на этапе 608 процесс загрузки находит образ быстрой загрузки, то он восстанавливает информацию быстрой загрузки, например, используя вызов BIOS, чтобы загрузить образ в оперативную память.

Этап 614 представляет распаковку (или передачу иным способом) образа в его конечное место в ОЗУ 132. Следует отметить, что для будущих машин, которые содержат энергонезависимое ОЗУ как, по меньшей мере, часть своего постоянного хранилища, BIOS действует на этапах 612 и 614, чтобы выполнить загрузку без

этапов 602-610. Тогда, на этапе 616 код (например, диспетчер) быстрой загрузки вызывает каждый компонент системы для его инициализации, как в целом описано со ссылкой на Фиг.3.

Теоретически, при наличии образа состояния быстрой загрузки, сохраненного в энергонезависимом ОЗУ, и участии большинства или всех компонентов ядра, время быстрой загрузки из выключенного состояния может приближаться к времени возобновления S3 (как правило, это время для восстановления образа из энергонезависимого ОЗУ + время возобновления S3 + дополнительная инициализация динамического состояния). С учетом существующей технологии, это время может составлять порядка пяти секунд (при условии наличия энергонезависимого ОЗУ объемом 65 МБ непосредственно на PCI-шине, когда извлечение всего содержимого в ОЗУ занимает менее одной секунды).

Несмотря на то что настоящее изобретение может быть подвержено различным модификациям и возможны альтернативные конструкции, выше на чертежах показаны и подробно описаны его конкретные варианты осуществления. Тем не менее, следует понимать, что нет какого-либо намерения ограничить изобретение раскрытыми конкретными формами, и напротив, изобретение должно охватывать все модификации, альтернативные конструкции и эквиваленты, входящие в рамки сущности и объема настоящего изобретения.

Формула изобретения

1. Материальный машиночитаемый носитель, содержащий машиночитаемые инструкции, которые при выполнении на вычислительном устройстве осуществляют этапы для выполнения быстрой загрузки вычислительного устройства из полностью выключенного состояния, причем этапы заключаются в том, что:

принимают статические данные и/или код из компонента системы в упомянутом вычислительном устройстве;

сохраняют статические данные и/или код в энергонезависимом носителе;

в течение последующей загрузки вычислительного устройства из полностью выключенного состояния указывают компоненту системы, что быстрая загрузка исполняется в текущий момент, и предоставляют компоненту системы доступ к статическим данным и/или коду; и в ответ на указание, компонент системы осуществляет доступ к статическим данным и/или коду так, что компонент системы использует статические данные и/или код для восстановления состояния компонента системы, причем в результате быстрой загрузки вычислительное устройство находится в полностью рабочем состоянии.

2. Машиночитаемый носитель по п.1, причем этап предоставления компоненту системы доступа к статическим данным и/или коду содержит предоставление компоненту системы точки входа в набор интерфейсов прикладного программирования инфраструктуры быстрой загрузки.

3. Машиночитаемый носитель по п.1, причем статические данные и/или код сохранены в образе в твердотельной памяти.

4. Машиночитаемый носитель по п.1, причем статические данные и/или код сохранены в образе на жестком диске.

5. Машиночитаемый носитель по п.1, причем этап предоставления компоненту системы доступа к статическим данным и/или коду содержит вызов компонента системы для инициализации в течение последующей загрузки.

6. Машиночитаемый носитель по п.5, причем этап предоставления компоненту

системы доступа к статическим данным и/или коду содержит вызов компонента системы в течение, по меньшей мере, двух фаз из множества фаз загрузки.

7. Машиночитаемый носитель по п.1, дополнительно содержащий машиночитаемые инструкции, которые при исполнении на вычислительном устройстве выполняют прием в инфраструктуре быстрой загрузки вызова от компонента системы, причем вызов предоставляет инфраструктуре быстрой загрузки доступ к статическим данным и/или коду для сохранения с целью использования при последующей загрузке.

8. Машиночитаемый носитель по п.1, дополнительно содержащий машиночитаемые инструкции, которые при исполнении на вычислительном устройстве выполняют прием в инфраструктуре быстрой загрузки вызова от компонента системы, причем вызов соответствует аннулированию статических данных с тем, чтобы статические данные в текущем состоянии не использовались при последующей загрузке.

9. Машиночитаемый носитель по п.1, дополнительно содержащий машиночитаемые инструкции, которые при исполнении на вычислительном устройстве обеспечивают осуществление связи с компонентом системы, работающим в фоновом режиме, чтобы предоставить инфраструктуре быстрой загрузки доступ к статическим данным и/или коду для сохранения для использования при последующей загрузке.

10. Способ осуществления быстрой загрузки вычислительного устройства из полностью выключенного состояния, содержащий этапы, на которых:

принимают статические данные и/или код из компонента системы в упомянутом вычислительном устройстве;

сохраняют статические данные и/или код в энергонезависимом носителе;

в течение последующей загрузки вычислительного устройства из полностью выключенного состояния указывают компоненту системы, что быстрая загрузка исполняется в текущий момент, и предоставляют компоненту системы доступ к статическим данным и/или коду; и в ответ на указание, компонент системы осуществляет доступ к статическим данным и/или коду так, что компонент системы использует статические данные и/или код для восстановления состояния компонента системы, причем в результате быстрой загрузки вычислительное устройство находится в полностью рабочем состоянии.

11. Способ по п.10, в котором статические данные и/или код принимаются в течение операции установки.

12. Способ по п.10, дополнительно содержащий этап, на котором, как часть восстановления состояния компонента системы, перечисляют динамические данные в компоненте системы.

13. Способ по п.10, в котором этап предоставления компоненту системы доступа к статическим данным и/или коду содержит передачу статических данных и/или кода из энергонезависимого носителя в энергозависимое устройство памяти.

14. Способ по п.10, в котором этап предоставления компоненту системы доступа к статическим данным и/или коду содержит предоставление компоненту системы точки входа в набор интерфейсов прикладного программирования инфраструктуры быстрой загрузки, и в котором этап указания компоненту системы, что быстрая загрузка исполняется в текущий момент, содержит установку флага, указывающего быструю загрузку.

15. Способ по п.10, в котором этап предоставления компоненту системы доступа к статическим данным содержит вызов компонента системы в течение, по меньшей мере, двух фаз из множества фаз загрузки.

16. Система в вычислительном устройстве, которая выполнена с возможностью

осуществления быстрой загрузки из полностью выключенного состояния, содержащая:

компонент системы и инфраструктуру быстрой загрузки, причем инфраструктура быстрой загрузки включает в себя:

первый механизм для осуществления:

приема статических данных и/или кода из компонента системы в вычислительном устройстве и

сохранения статических данных и/или кода в энергонезависимом носителе; и

второй механизм для осуществления:

в течение последующей загрузки вычислительного устройства из полностью выключенного состояния указания компоненту системы, что быстрая загрузка исполняется в текущий момент, и

предоставления компоненту системы доступа к статическим данным и/или коду; и

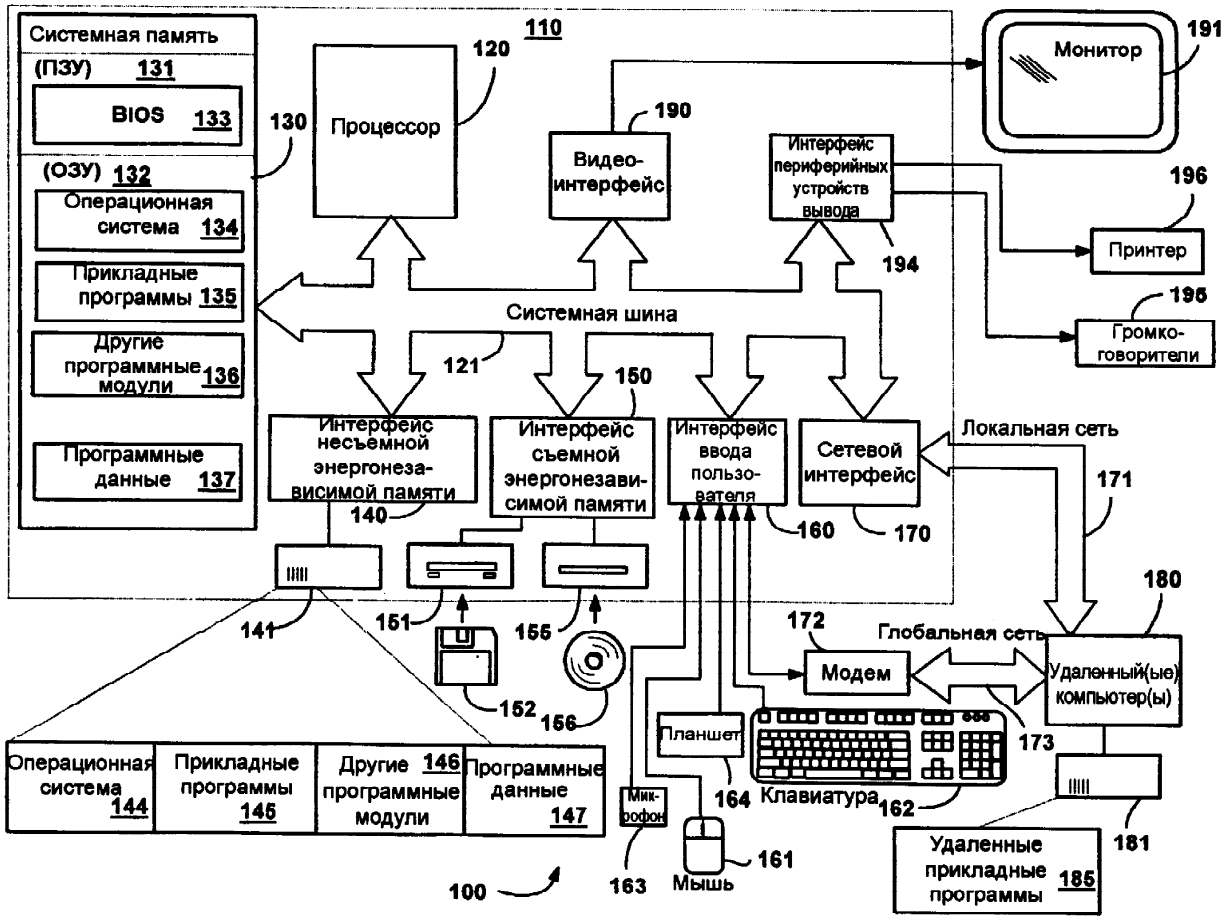
при этом в ответ на прием указания из второго механизма компонент системы осуществляет доступ к статическим данным и/или коду так, что компонент системы использует статические данные и/или код для восстановления состояния компонента системы, причем в результате быстрой загрузки вычислительное устройство находится в полностью рабочем состоянии.

17. Система по п.16, в которой первый механизм содержит установочный компонент, который регистрирует компонент системы, включающий в себя предоставление компоненту системы набора интерфейсов для приема статических данных и/или кода.

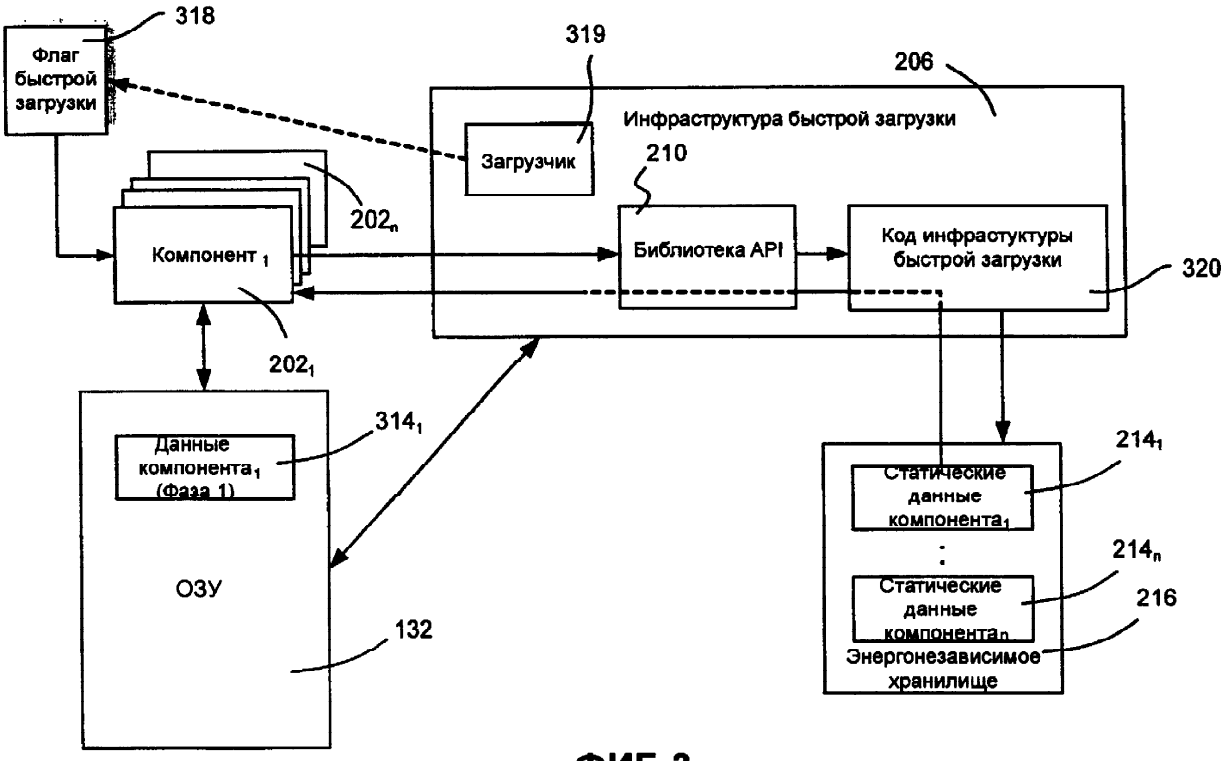
18. Система по п.17, в которой второй механизм содержит код, который осуществляет связь с компонентом системы через один интерфейс из упомянутого набора интерфейсов.

19. Система по п.17, в которой инфраструктура быстрой загрузки содержит третий механизм для аннулирования статических данных.

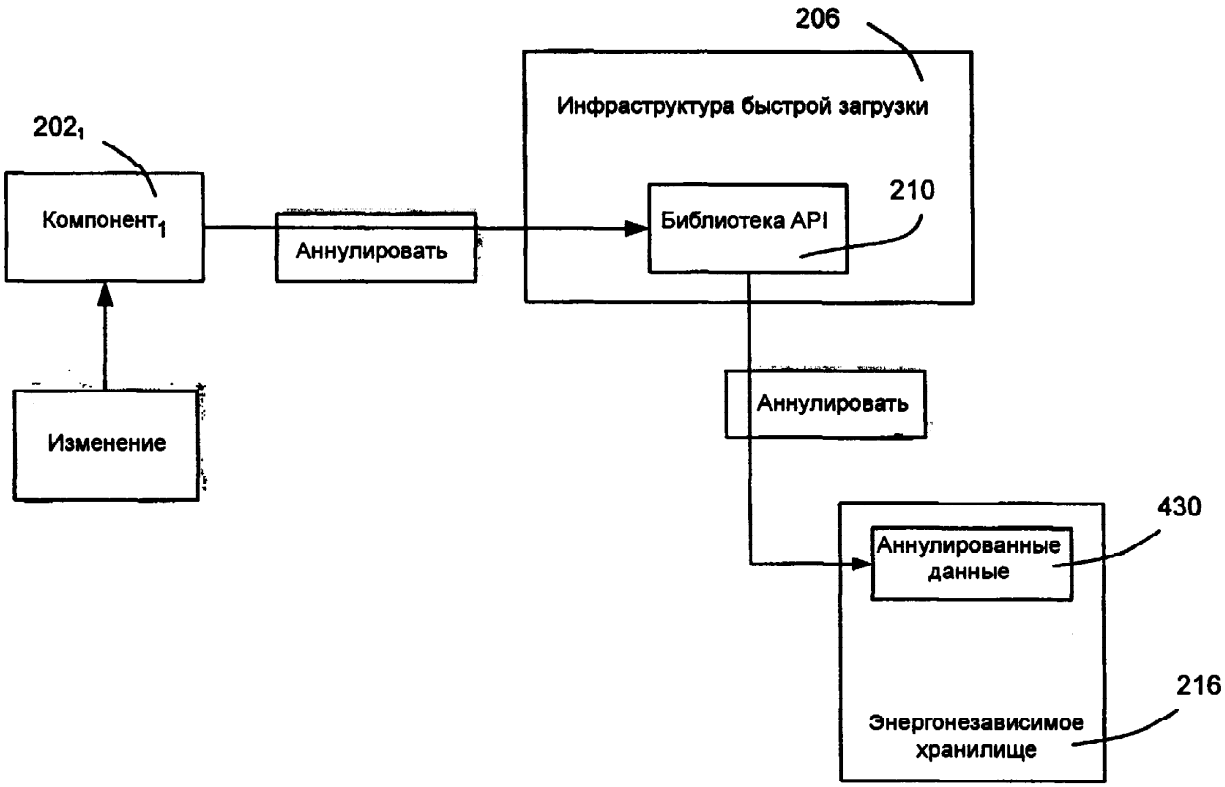
20. Система по п.17, в которой инфраструктура быстрой загрузки содержит фоновый механизм для сбора статических данных и/или кода от компонента системы.


Системная память
(ПЗУ) 131
BIOS 133
(ОЗУ) 132
Операционная система 134
Прикладные программы 135
Другие программные модули 136
Программные данные 137
130
110
Процессор
120
Видео-интерфейс
190
Монитор
191
Интерфейс периферийных устройств вывода
194
196
Принтер
195
Громко-говорители
Системная шина
121
150
Интерфейс несъемной энергонезависимой памяти
140
Интерфейс съемной энергонезависимой памяти
Интерфейс ввода пользователя
160
Сетевой интерфейс
170
Локальная сеть
171
180
141
151
155
172
Глобальная сеть
Модем
Удаленный(ые) компьютер(ы)
152
156
Планшет
164
Клавиатура
162
173
Операционная система 144
Прикладные программы 145
Другие программные модули 146
Программные данные 147
Микрофон
163
Мышь
161
181
Удаленные прикладные программы 185
100

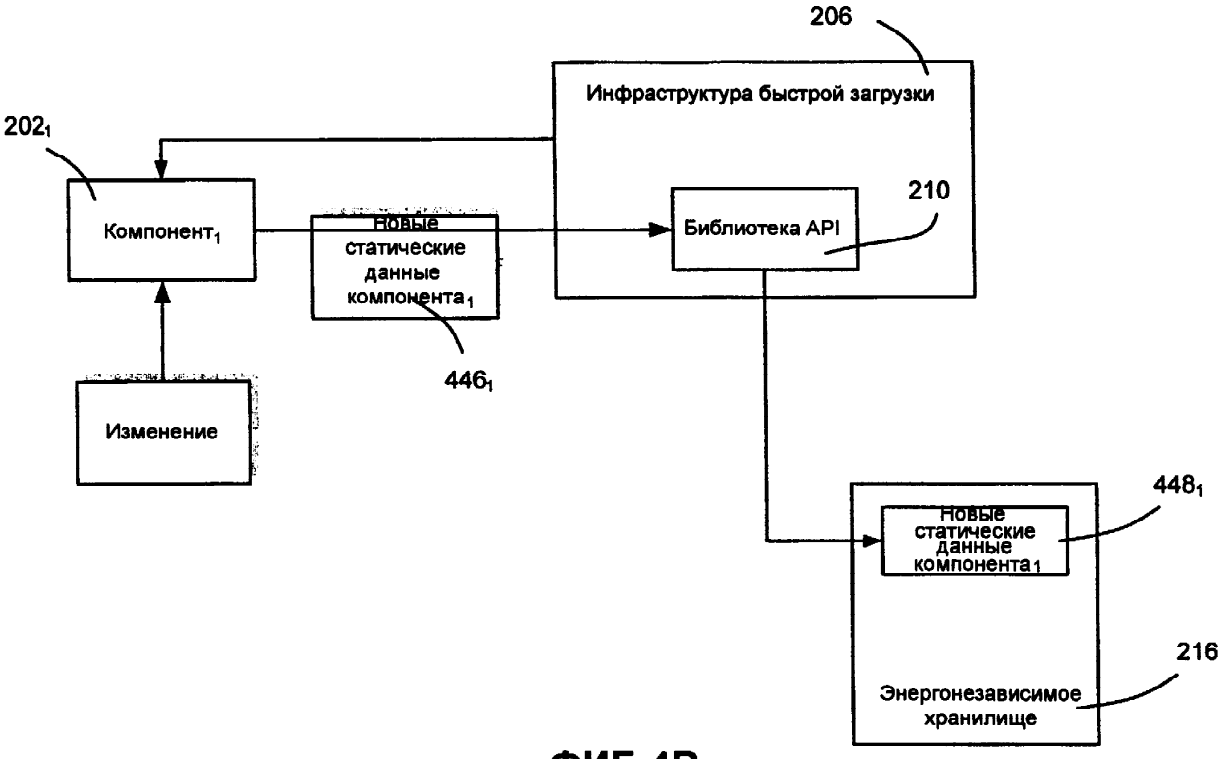
ФИГ. 1


318
Флаг быстрой загрузки
319
206
Инфраструктура быстрой загрузки
Загрузчик
210
202_n
Компонент 1
Библиотека API
Код инфрастуктуры быстрой загрузки
320
202_1
314_1
Данные компонента$_1$ (Фаза 1)
ОЗУ
132
214_1
Статические данные компонента$_1$
214_n
Статические данные компонента$_n$
216
Энергонезависимое хранилище

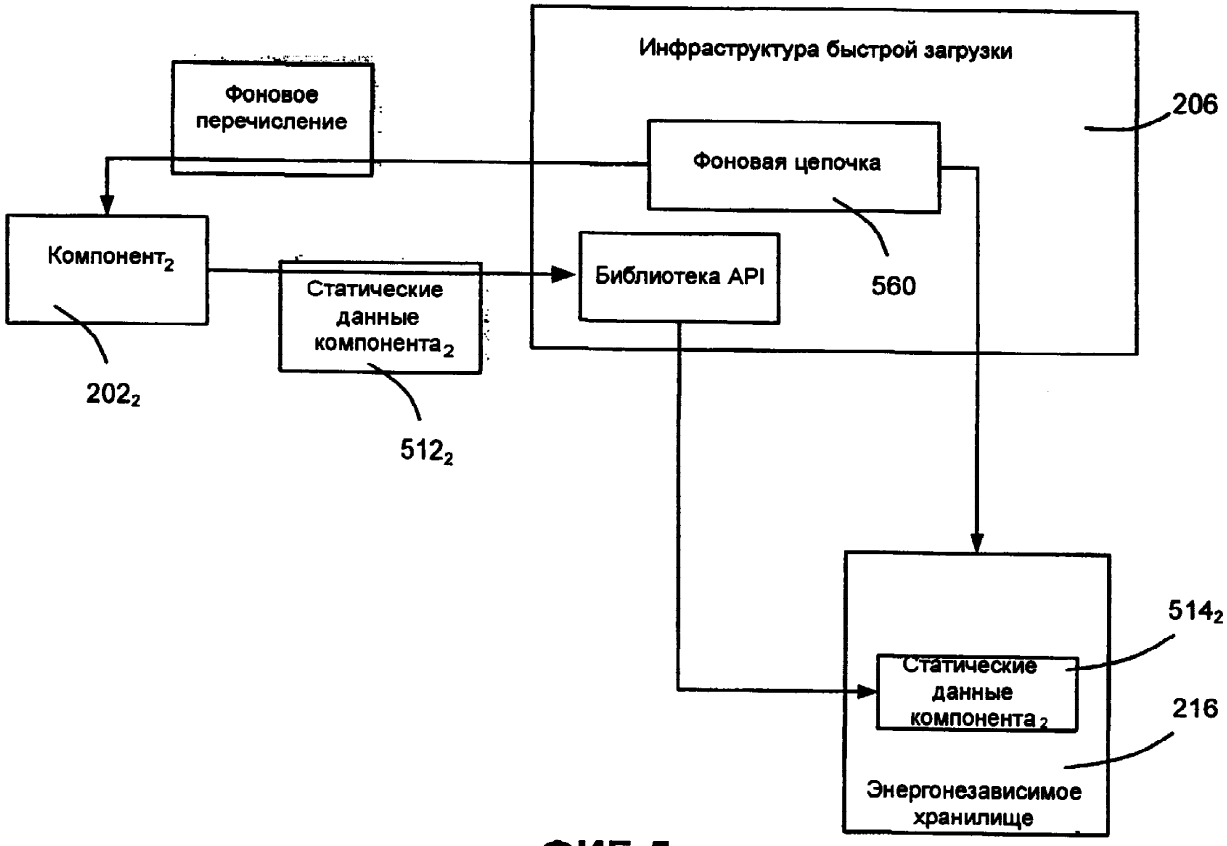
ФИГ. 3


206
Инфраструктура быстрой загрузки
202₁
Компонент₁
Аннулировать
Библиотека API
210
Аннулировать
Изменение
430
Аннулированные данные
216
Энергонезависимое хранилище

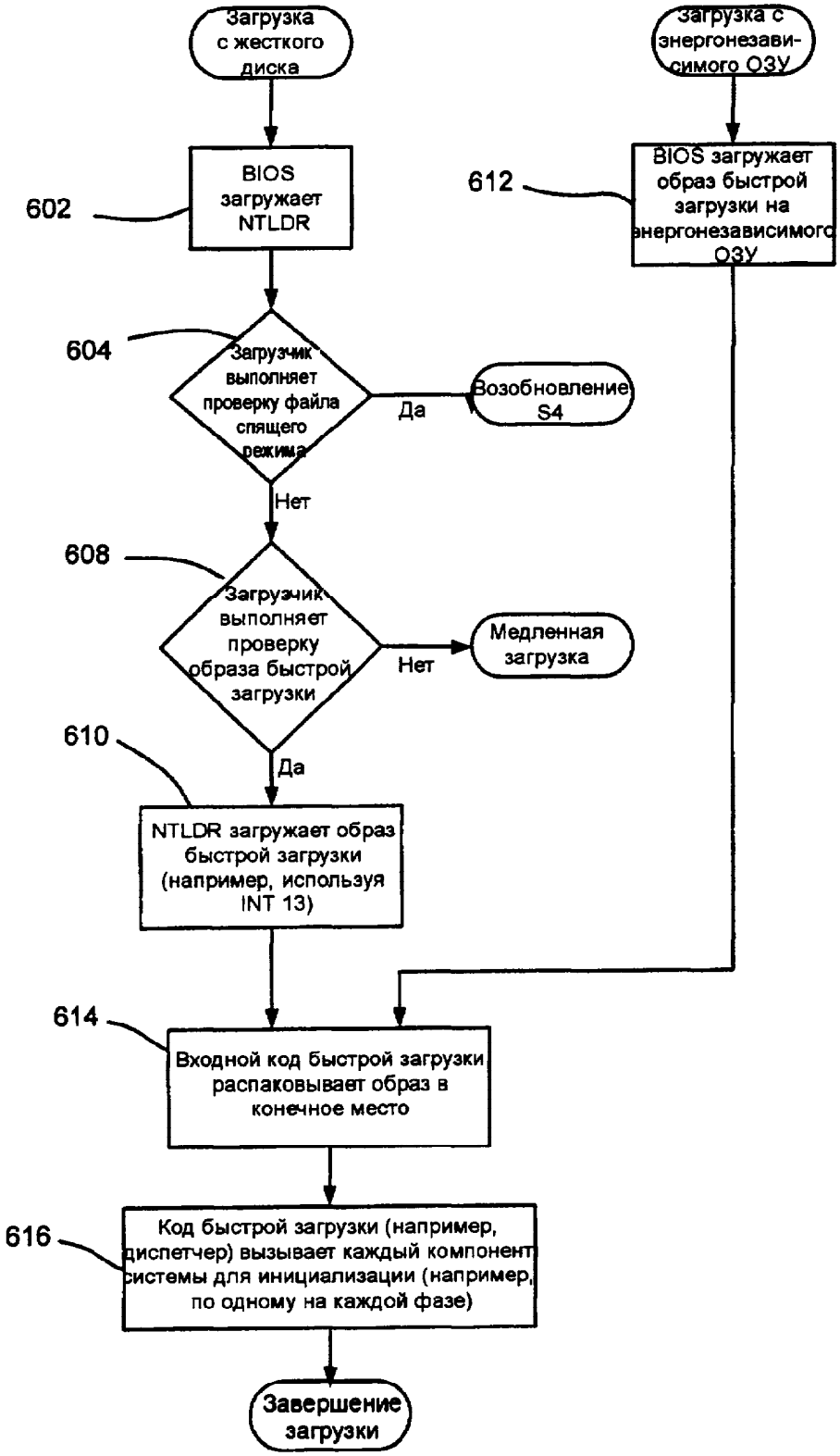
ФИГ. 4A


206
Инфраструктура быстрой загрузки
202₁
Компонент₁
Новые статические данные компонента₁
446₁
Библиотека API
210
Изменение
448₁
Новые статические данные компонента₁
216
Энергонезависимое хранилище

ФИГ. 4B


Инфраструктура быстрой загрузки
206
Фоновое перечисление
Фоновая цепочка
Компонент2
Статические данные компонента2
Библиотека API
560
2022
5122
5142
Статические данные компонента2
216
Энергонезависимое хранилище

ФИГ. 5


Загрузка с жесткого диска
Загрузка с энергонезависимого ОЗУ
BIOS загружает NTLDR
602
BIOS загружает образ быстрой загрузки на энергонезависимого ОЗУ
612
604
Загрузчик выполняет проверку файла спящего режима
Да
Возобновление S4
Нет
608
Загрузчик выполняет проверку образа быстрой загрузки
Нет
Медленная загрузка
Да
610
NTLDR загружает образ быстрой загрузки (например, используя INT 13)
614
Входной код быстрой загрузки распаковывает образ в конечное место
616
Код быстрой загрузки (например, диспетчер) вызывает каждый компонент системы для инициализации (например, по одному на каждой фазе)
Завершение загрузки

ФИГ. 6