РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 357 280** (13) **C2**

(51) МПК
G06F 12/16 (2006.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ,
ПАТЕНТАМ И ТОВАРНЫМ ЗНАКАМ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21), (22) Заявка: **2004132538/09, 05.11.2004**

(24) Дата начала отсчета срока действия патента:
05.11.2004

(30) Конвенционный приоритет:
06.11.2003 US 10/702,863

(43) Дата публикации заявки: **10.04.2006**

(45) Опубликовано: **27.05.2009** Бюл. № **15**

(56) Список документов, цитированных в отчете о поиске: **US 4641274, 03.02.1987. WO 01/98951 A1, 27.12.2001. EP 0665670 A2, 02.08.1995. RU 2155373 C2, 27.08.2000. RU 2163726 C2, 27.02.2001.**

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

(72) Автор(ы):
ЛЮ Хай (US),
АНТОНОФФ Лаурен Н. (US)

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

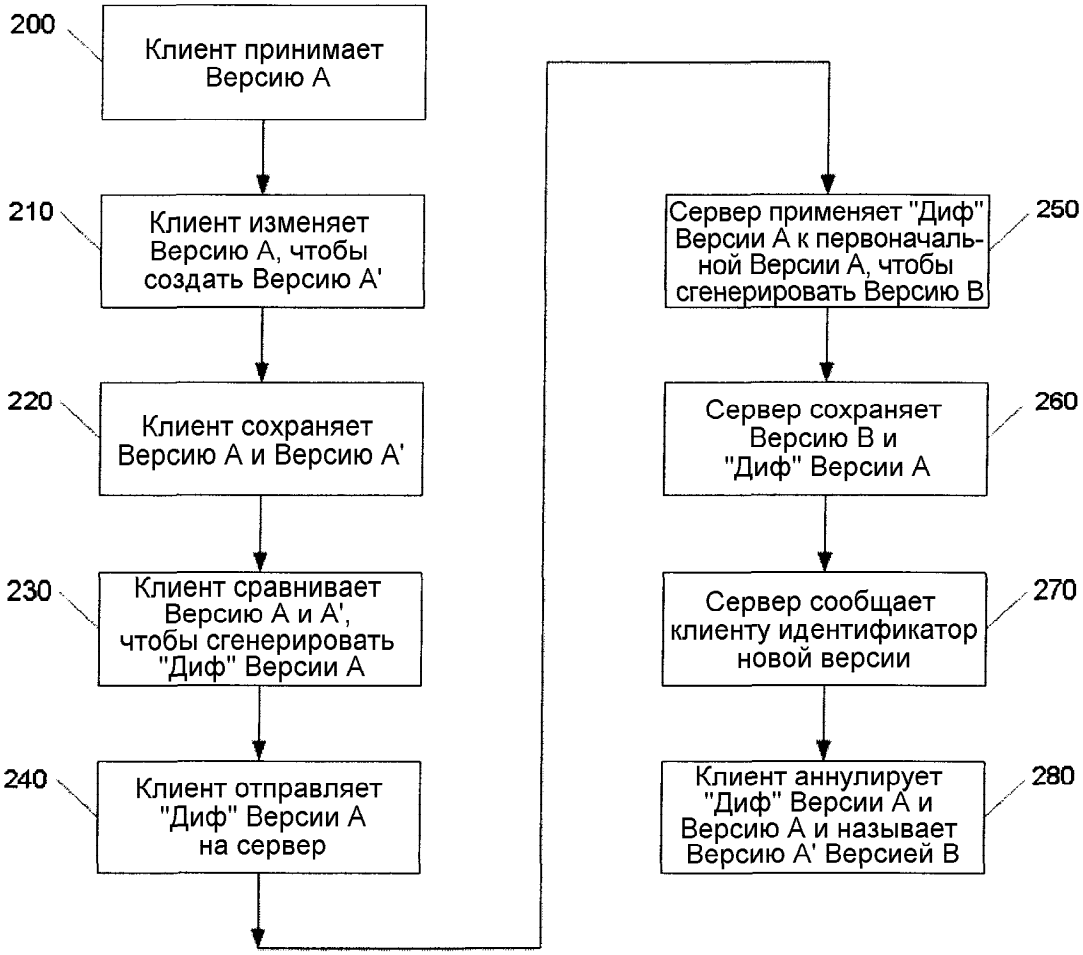
(54) ОПТИМИЗАЦИЯ РЕПЛИКАЦИИ ФАЙЛОВ С ИСПОЛЬЗОВАНИЕМ ДВОИЧНЫХ СРАВНЕНИЙ

(57) Реферат:

Изобретение относится к вычислительной технике. Техническим результатом является обеспечение синхронности копий файла, расположенных на клиенте и на сервере, при внесении в этот файл изменений. Данные сравниваются с предыдущей версией, известной как клиенту, так и серверу, и генерируется имеющее высокую степень сжатия представление различий между файлом и его предыдущей версией. Затем осуществляется передача этих различий или «дифов», при этом могут использоваться расширения протокола HTTP (Протокола Передачи Гипертекста). 2 н. и 22 з.п. ф-лы, 6 ил.

RU 2357280 C2

RU 2357280 C2



Фиг. 2

RU 2357280 C2

RU 2357280 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY,
PATENTS AND TRADEMARKS

(19) **RU** (11) **2 357 280** (13) **C2**

(51) Int. Cl.
G06F 12/16 (2006.01)

(12) ABSTRACT OF INVENTION

(21), (22) Application: **2004132538/09, 05.11.2004**

(24) Effective date for property rights:
05.11.2004

(30) Priority:
06.11.2003 US 10/702,863

(43) Application published: **10.04.2006**

(45) Date of publication: **27.05.2009 Bull. 15**

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

(72) Inventor(s):
LJu Khaj (US),
ANTONOFF Lauren N. (US)

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

(54) OPTIMISING REPLICATION OF FILES USING BINARY COMPARISONS

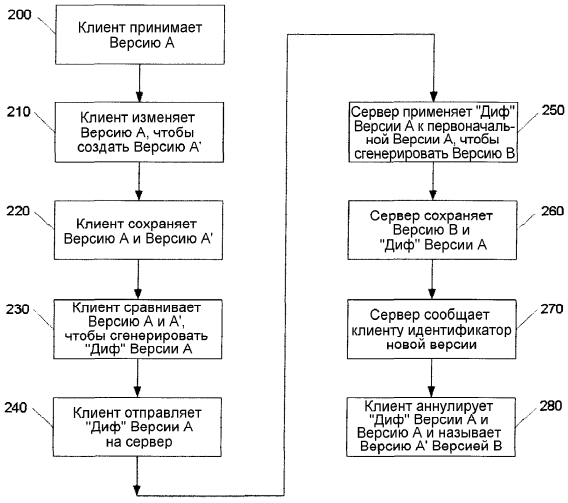
(57) Abstract:

FIELD: information technology.

SUBSTANCE: invention relates to computer engineering. Data is compared with the previous version, known to the client, and to the server, and the high compression presented in the difference between the file and its previous version is generated. Then the transmission of these differences or "diffs" is carried out, and the extension of the HTTP protocol (Hypertext transfer protocol) can be used,.

EFFECT: providing synchronism of the copies of the file, located on the client and the server, while introducing changes to the file.

24 cl, 6 dwg



Фиг. 2

RU 2357280 C2

RU 2357280 C2

ОБЛАСТЬ ТЕХНИКИ, К КОТОРОЙ ОТНОСИТСЯ ИЗОБРЕТЕНИЕ

Данное изобретение, в общем, относится к области репликации машинных файлов. Более конкретно, данное изобретение относится к репликации файлов с использованием двоичных сравнений.

ПРЕДШЕСТВУЮЩИЙ УРОВЕНЬ ТЕХНИКИ

Репликация позволяет осуществлять как локальный, так и удаленный доступ к данным за счет синхронного сопровождения клиентской и серверной версий файла или документа. При том, что эта функциональная возможность ценна и крайне необходима для приложений, она также весьма затратна, что связано с теми объемами данных, которые необходимо передавать между клиентами и сервером. Целые файлы и документы вместе с различными их версиями сохраняются на сервере и передаются между сервером и его клиентами. В силу этого многие системы репликации пытаются сберечь полосу пропускания, сжимая данные перед их передачей. Однако традиционные способы сжатия осуществляют кодирование данных всего файла, даже если большая часть этих данных была передана как часть предыдущей версии. Таким образом, даже небольшие изменения, внесенные в данные файла или документ, все-таки требуют сжатия и передачи всего файла или документа, несмотря на то, что большая часть этих данных уже находится в месте назначения в форме ранее полученной версии.

В связи с вышесказанным существует потребность в системах и способах, устраняющих ограничения и недостатки предшествующего уровня развития техники.

СУЩНОСТЬ ИЗОБРЕТЕНИЯ

Настоящее изобретение предлагает механизм для поддержания синхронности копий файла, расположенных на клиенте и на сервере, при внесении в этот файл изменений. Данные сравниваются с предыдущей версией, известной как клиенту, так и серверу, и генерируется имеющее высокую степень сжатия представление различий между файлом и его предыдущей версией.

Согласно одному варианту осуществления изобретения на клиенте производится прием и сохранение первой копии и второй копии базового файла. Обе копии являются идентичными - клиент принимает одну копию и сохраняет два экземпляра этой копии. Затем клиент вносит изменения в первую копию, и производится определение различия (такого как двоичное различие, т.е. различие, выраженное в двоичной форме) между измененной первой копией и второй копией. Различие передается на сервер, который сопровождает базовый файл. В случае если базовый файл на сервере является таким же, как базовый файл, который хранился на первом устройстве, сервер принимает различие; в противном же случае сервер отвергает различие.

Согласно аспектам изобретения, если вышеупомянутое различие отвергнуто сервером, то сервер передает клиенту второе различие. После этого клиент применяет второе различие ко второй копии базового файла, хранящейся на первом устройстве. Таким образом, клиентский базовый файл обновляется в соответствии с базовым файлом, который находится на сервере. После этого клиент может произвести изменения этого обновленного файла, сгенерировать новое различие и передать новое различие на сервер.

Дополнительные признаки и преимущества изобретения станут очевидными из следующего ниже подробного описания иллюстративных вариантов осуществления изобретения, которое приводится вместе со ссылками на сопровождающие его чертежи.

ПЕРЕЧЕНЬ ФИГУР ЧЕРТЕЖЕЙ

Предшествующее описание сущности изобретения, равно как и следующее ниже подробное описание предпочтительных вариантов осуществления изобретения более понятны при рассмотрении их совместно с прилагаемыми чертежами:

Фиг. 1 - структурная схема, изображающая иллюстративную вычислительную среду, в которой могут быть реализованы аспекты изобретения.

Фиг. 2 - блок-схема алгоритма иллюстративного способа сопровождения обновляемого файла согласно настоящему изобретению.

Фиг. 3 - блок-схема алгоритма другого иллюстративного способа сопровождения обновляемого файла согласно настоящему изобретению.

Фиг. 4 - структурная схема, изображающая иллюстративную систему, полезную для описания аспектов настоящего изобретения.

Фиг. 5 и 6 - блок-схемы алгоритма другого иллюстративного способа сопровождения обновляемого файла согласно настоящему изобретению.

ПОДРОБНОЕ ОПИСАНИЕ ПРЕДПОЧТИТЕЛЬНЫХ ВАРИАНТОВ ОСУЩЕСТВЛЕНИЯ ИЗОБЕТЕНИЯ

Краткий обзор

Настоящее изобретение относится к поддержанию синхронности локальных (также именуемых здесь «клиентскими») и расположенных на сервере копий файла при внесении в этот файл изменений. Иллюстративные системы и способы, описанные в данном документе, являются более эффективными, чем имеющиеся на данный момент технические приемы, и сводят к минимуму как требования к полосе пропускания, так и элемент времени при осуществлении синхронизации.

Данные сравниваются с предыдущей версией, известной как клиенту, так и серверу, и генерируется имеющее высокую степень сжатия представление различий между файлом и его предыдущей версией. Затем осуществляется передача этих различий или «дифов» («diff», сокращение от английского «difference» - разница), при этом могут использоваться расширения протокола HTTP (Протокола Передачи Гипертекста).

Иллюстративная вычислительная среда

Фиг.1 иллюстрирует пример подходящей среды 100 вычислительной системы, в которой может быть реализовано изобретение. Среда 100 вычислительной системы представляет собой только один пример подходящей вычислительной среды и не подразумевает наложение каких-либо ограничений на область использования или функциональные возможности данного изобретения. В равной степени вычислительная среда 100 не должна толковаться как имеющая какую-либо зависимость или требование в отношении какого-либо одного компонента или комбинации компонентов, изображенных в составе иллюстративной операционной среды 100.

Изобретение способно функционировать и с другими средами или конфигурациями вычислительных систем универсального или специального назначения. Примеры широко известных вычислительных систем, сред и/или конфигураций, которые могут быть пригодны для использования с настоящим изобретением, включают в себя, но не в ограничительном смысле, персональные компьютеры, компьютеры-серверы, переносные или портативные устройства, многопроцессорные системы, системы, основанные на микропроцессорах, телевизионные приставки, программируемое бытовое электронное оборудование, сетевые персональные компьютеры, миникомпьютеры, универсальные компьютеры (мейнфреймы), распределенные вычислительные среды, содержащие любые из вышеупомянутых систем или устройств,

и т.п.

Изобретение может быть описано в общем контексте машиноисполняемых команд, таких как программные модули, исполняемые компьютером. Обычно программные модули включают в себя процедуры, программы, объекты, компоненты, структуры данных и т.п., которые выполняют конкретные задачи или реализуют определенные абстрактные типы данных. Изобретение может быть также осуществлено в распределенных вычислительных средах, где задачи выполняются удаленными устройствами обработки данных, связанными через сеть связи или другую среду передачи данных. В распределенной вычислительной среде программные модули и другие данные могут быть расположены как в локальных, так и в удаленных компьютерных носителях информации, включая запоминающие устройства.

Согласно фиг. 1 иллюстративная система для осуществления изобретения включает в себя вычислительное устройство общего назначения в виде компьютера 110. Компоненты компьютера 110 могут содержать, но не в ограничительном смысле, процессор 120, системную память 130 и системную шину 121, соединяющую различные компоненты системы, включая системную память, с процессором 120. Системная шина 121 может относиться к любому из нескольких типов конструкций шины, включая шину памяти или контроллер памяти, периферийную шину и локальную шину, использующие любую из множества типов архитектуры шины. В качестве примера, но не ограничения, такие типы архитектуры включают в себя шину архитектуры, соответствующей промышленному стандарту (ISA), шину микроканальной архитектуры (MCA), усовершенствованную шину ISA (EISA), локальную шину Ассоциации по стандартам в области видеоэлектроники (VESA) и шину межсоединения периферийных компонентов (PCI), также известную как мезонинная шина.

Компьютер 110 обычно содержит разнообразные машиночитаемые носители информации. Машиночитаемые носители информации могут представлять собой любые имеющиеся в наличии носители информации, к которым компьютер 110 может осуществлять доступ, и включают в себя как энергозависимые, так и энергонезависимые носители информации, как съемные, так и несъемные носители информации. В качестве примера, но не ограничения, машиночитаемые носители информации могут включать в себя компьютерные носители информации и среды передачи данных. Компьютерные носители информации включают в себя как энергозависимые, так и энергонезависимые носители информации, как съемные, так и несъемные носители информации, реализованные любым способом или технологией для хранения информации, такой как машиночитаемые команды, структуры данных, программные модули или другие данные. Компьютерные носители информации включают в себя, но не в ограничительном смысле, оперативное запоминающее устройство (ОЗУ, RAM), постоянное запоминающее устройство (ПЗУ, ROM), электрически стираемое программируемое постоянное запоминающее устройство (ЭСППЗУ, EEPROM), флэш-память или память, использующую другую технологию, постоянное запоминающее устройство на компакт-диске (CD-ROM), универсальные цифровые диски (DVD) или другой оптический дисковый накопитель, магнитные кассеты, магнитную ленту, магнитный дисковый накопитель или другие магнитные запоминающие устройства, или любой другой носитель, который может использоваться для хранения необходимой информации, и к которому компьютер 110 может осуществлять доступ. Среды передачи данных обычно воплощают машиночитаемые команды, структуры данных, программные модули или другие

данные в сигнале, модулированном данными, таком как несущая, или другом транспортном механизме и включают в себя любые среды доставки информации. Термин «сигнал, модулированный данными» означает сигнал, в котором одна или более его характеристик установлены или изменены таким образом, чтобы обеспечить кодирование информации в сигнале. В качестве примера, но не ограничения, среды передачи данных включают в себя проводные среды, такие как проводная сеть или прямое кабельное соединение, и беспроводные среды, такие как акустическая, радиочастотная, инфракрасная и другие беспроводные среды передачи данных. Комбинации любых упомянутых выше сред также должны быть включены в диапазон машиночитаемых носителей информации.

Системная память 130 включает в себя компьютерный носитель информации в виде энергозависимой и/или энергонезависимой памяти, такой как постоянное запоминающее устройство (ПЗУ) 131 и оперативное запоминающее устройство (ОЗУ) 132. Базовая система 133 ввода/вывода (BIOS), содержащая базовые процедуры, способствующие передаче информации между элементами внутри компьютера 110, например, при запуске, обычно хранится в ПЗУ 131. ОЗУ 132 обычно содержит данные и/или программные модули, к которым процессор 120 может осуществить доступ немедленно или которыми он в текущий момент оперирует. В качестве примера, но не ограничения, фиг. 1 изображает операционную систему 134, прикладные программы 135, другие программные модули 136 и данные 137 программ.

Компьютер 110 может также содержать другие съемные/несъемные, энергозависимые/энергонезависимые компьютерные носители информации. Только в качестве примера, фиг. 1 изображает накопитель 141 на жестких магнитных дисках, который осуществляет считывание с несъемного, энергонезависимого магнитного носителя информации или запись на него, дисковод 151 для магнитного диска, который осуществляет считывание со съмного энергонезависимого магнитного диска 152 или запись на него, и дисковод 155 для оптического диска, который осуществляет считывание со съемного энергонезависимого оптического диска 156, такого как компакт-диск (CD-ROM) или другие оптические носители информации, или запись на него. Другие съемные/несъемные, энергозависимые/энергонезависимые компьютерные носители информации, которые могут быть использованы в иллюстративной операционной среде, включают в себя, но не в ограничительном смысле, кассеты с магнитной лентой, карточки флэш-памяти, универсальные цифровые диски, цифровую видеомагнитофонную ленту, твердотельное ОЗУ, твердотельное ПЗУ и т.д. Накопитель 141 на жестких магнитных дисках обычно подсоединен к системной шине 121 посредством интерфейса несъемной памяти, такого как интерфейс 140, а дисковод 151 для магнитного диска и дисковод 155 для оптического диска обычно подсоединены к системной шине 121 посредством интерфейса съемной памяти, такого как интерфейс 150.

Дисководы и соответствующие им компьютерные носители информации, описанные выше и изображенные на фиг. 1, обеспечивают хранение машиночитаемых команд, структур данных, программных модулей и других данных для компьютера 110. Например, на фиг. 1 накопитель 141 на жестких магнитных дисках изображен как хранящий операционную систему 144, прикладные программы 145, другие программные модули 146 и данные 147 программ. Следует отметить, что эти компоненты могут быть либо идентичными операционной системе 134, прикладным программам 135, другим программным модулям 136 и данным 137 программ, либо отличаться от них. На данной фигуре операционной системе 144, прикладным

программам 145, другим программным модулям 146 и данным 147 программ даны другие номера позиций для иллюстрации того факта, что они, как минимум, являются другими копиями. Пользователь может осуществлять ввод команд и информации в компьютер 110 посредством устройств ввода, таких как клавиатура 162 и координатно-указательное устройство 161, обычно определяемое как мышь, шаровой манипулятор или сенсорная панель. В число других устройств ввода (не показанных) могут входить микрофон, джойстик, игровая панель, спутниковая параболическая антенна, сканер и т.д. Часто эти и другие устройства ввода соединены с процессором 120 посредством интерфейса 160 пользовательского ввода, подсоединенного к системной шине, но они могут быть соединены с процессором посредством других конструкций интерфейса и шины, таких как параллельный порт, игровой порт или универсальная последовательная шина (USB). Также к системной шине 121 посредством интерфейса, такого как видеоинтерфейс 190, может быть подсоединен монитор 191 или другой тип устройства отображения. В дополнение к монитору компьютеры могут также содержать другие периферийные устройства вывода, такие как громкоговорители 197 и принтер 196, которые могут быть подсоединены посредством периферийного интерфейса 195 вывода.

Компьютер 110 может функционировать в сетевой среде, используя логические соединения с одним или большим количеством удаленных компьютеров, таких как удаленный компьютер 180. Удаленный компьютер 180 может быть персональным компьютером, сервером, маршрутизатором, сетевым PC, одноранговым устройством или другим узлом общей сети и обычно содержит многие или все элементы, описанные выше в отношении компьютера 110, хотя на фиг. 1 изображено только запоминающее устройство 181. Показанные логические соединения включают в себя локальную сеть (LAN) 171 и глобальную сеть (WAN) 173, но могут также включать в себя и другие сети. Такие сетевые среды часто используются в офисах, сетях масштаба предприятия, интрасетях и в сети Интернет.

При использовании в сетевой среде LAN компьютер 110 соединен с сетью LAN 171 посредством сетевого интерфейса или адаптера 170. При использовании в сетевой среде WAN компьютер 110 обычно включает в себя модем 172 или другие средства для установления связи через сеть WAN 173, такую как Интернет. Модем 172, который может быть внутренним или внешним, может быть подсоединен к системной шине 121 посредством интерфейса 160 пользовательского ввода или другого соответствующего механизма. В сетевой среде программные модули, показанные в отношении компьютера 110, или их части могут храниться в удаленном запоминающем устройстве. В качестве примера, но не ограничения, фиг. 1 изображает удаленно хранящиеся прикладные программы 185 как размещенные на запоминающем устройстве 181. Следует иметь в виду, что изображенные сетевые соединения являются иллюстративными и могут быть использованы и другие средства установления линии связи между компьютерами.

Иллюстративные распределенные вычислительные системы или архитектуры

В свете сближения персональных вычислений и сети Интернет были разработаны и разрабатываются в настоящий момент различные распределенные вычислительные системы. Как индивидуальным, так и корпоративным пользователям предоставляется обеспечивающий прозрачное взаимодействие интерфейс с поддержкой Web для приложений и вычислительных устройств, при этом производимые на компьютере операции делаются во все возрастающей степени ориентированными на использование сетевого браузера или работу в сети.

Например, платформа.NET компании Microsoft® включает в себя серверы, услуги блочно-модульной компоновки, такие как Web-ориентированное хранение данных, и загружаемое программное обеспечение устройств. Вообще говоря, платформа NET обеспечивает: (1) возможность организовать совместную работу целого ряда вычислительных устройств и автоматическое обновление и синхронизацию пользовательской информации на всех этих устройствах, (2) повышенную интерактивную способность Web-сайтов, достигнутую за счет большего использования расширяемого языка разметки XML, чем языка разметки гипертекста HTML, (3) услуги, предоставляемые в интерактивном режиме (режиме «он-лайн»), которые характеризуются настраиваемыми на индивидуальной основе доступом и доставкой пользователю продуктов и услуг из центральной исходной точки для управления различными приложениями, такими, например, как электронная почта, или программных средств, таких как пакет программ Office® компании Microsoft®, (4) централизованное хранение данных, которое повысит эффективность и простоту доступа к информации, равно как и уровень синхронизации информации между пользователями и устройствами, (5) возможность интеграции различных средств связи, таких как электронная почта, факсы и телефоны, (6) возможность для разработчиков создавать многократно используемые модули, благодаря чему повышается производительность и снижается количество ошибок программирования, и (7) также много других характеристик межплатформенной интеграции.

Хотя приводимые в данном документе варианты осуществления изобретения описываются в связи с программным обеспечением, находящимся в вычислительном устройстве, одна или более частей изобретения могут быть также реализованы посредством операционной системы, интерфейса прикладного программирования (API) или объекта-посредника между сопроцессором и запрашивающим объектом таким образом, что исполнение, поддержка услуг или доступ к услугам могут быть осуществлены средствами всех языков и служб платформы.NET, равно как и в других распределенных вычислительных системах.

Иллюстративные варианты осуществления изобретения

Фиг. 2 представляет собой блок-схему алгоритма иллюстративного способа сопровождения обновляемого файла согласно настоящему изобретению. В этом иллюстративном варианте осуществления изобретения клиент изменяет файл и пересылает изменения на сервер. На этапе 200 клиент принимает копию последней версии («Версии A») базового файла, который хранится на сервере. На этапе 210 клиент производит свои изменения в Версии A, чтобы создать Версию A'. На этапе 220 клиент сохраняет копию первоначальной Версии A и новой Версии A'. Таким образом, клиент сохраняет копию последнего известного состояния файла на сервере, даже если пользователь обновляет файл. Предполагается, что копия Версии A может быть сохранена у клиента либо до этапа 210, либо после этапа 210.

После этого на этапе 230 посредством сравнения Версий A и A' определяется различие или «диф». «Диф» представляет собой механизм, посредством которого производится сравнение двух версий файла с целью генерации сжатого «дифа», который может быть применен к более ранней версии файла для генерации более поздней версии файла. Нахождение различия может осуществляться при помощи любого способа, технической процедуры или системы, известных специалистам в данной области техники и предназначенных для определения различия между базовой формой и измененной формой. Предпочтительно, чтобы различие, которое

сгенерировано, являлось двоичным различием. Файл рассматривается как последовательность байтов. Для генерации двоичного различия путем определения различия между дубликатом или базовой копией и исправленной копией используется традиционный алгоритм сжатия. Это различие затем направляется на сервер, где оно отвергается или считается приемлемым. Различие будет отвергнуто, если базовый файл, хранящийся на сервере, изменился, и в этом случае различие бесполезно для сервера. Предполагается, что согласно настоящему изобретению может быть использовано любое средство или методика определения различия. Использование методики двоичного различия представлено здесь в иллюстративных целях.

В частности, на этапе 240 клиент отправляет «диф» на сервер. Сервер после проведения проверки, подтверждающей, что имеющаяся у него последняя версия базового файла не изменилась по сравнению с Версией А, которую клиент использовал для внесения изменений, применяет на этапе 250 «диф» к Версии А, чтобы сгенерировать новую, самую последнюю версию файла, Версию В. Проверка версий, осуществляемая сервером, более подробно описывается ниже в отношении фиг. 3-6.

На этапе 260 сервер сохраняет новую Версию В, равно как и представленный клиентом «диф» (последнее в необязательном порядке). Новая Версия В рассматривается как самое последнее обновление базового файла, а «диф» сохраняется для использования другими клиентами, которые могут вносить изменения в первоначальную Версию А, как это более подробно описывается ниже в отношении фиг. 3-6. Сервер, в необязательном порядке, сохраняет «диф» для того, чтобы предоставить другим клиентам возможность оптимизировать обновление. Если произведено несколько ревизий, то для того, чтобы перейти от более ранней версии к более поздней, может потребоваться несколько «дифов». Следует отметить, что «диф» может также оказаться полезным для тех клиентов, которые не намереваются производить изменения, но хотят прочесть последнюю Версию В и уже имеют Версию А.

На этапе 270 сервер сообщает клиенту идентификатор новой версии (например, «Версия В»). После этого клиент аннулирует «диф», который он определил на этапе 230, равно как и Версию А, которую он сохранял, а на этапе 280 клиент присваивает своей Версии А' идентификатор новой версии. Таким образом, клиент переименовывает Версию А' в Версию В.

Фиг. 3 представляет собой блок-схему алгоритма другого иллюстративного способа сопровождения обновляемого файла. В этом примере сервер предоставляет клиенту последние изменения в виде файла различий («диф-файла»). На этапе 300 клиент, имеющий Версию А файла, запрашивает обновление файла. Клиент мог бы сделать такой запрос по той, например, причине, что клиенту желательно произвести изменения в самой последней версии файла. Клиент сообщает серверу о том, что на клиенте имеется Версия А, и в ответ на это сервер на этапе 310 посылает клиенту «диф» для А. Сервер мог сохранить «диф» для А после обновления, произведенного предыдущим клиентом (например, на этапе 260 на фиг. 2). На этапе 320 клиент применяет «диф» Версии А к сохраняемой им Версии А, чтобы сгенерировать самую последнюю версию файла (например, «Версию В»).

Фиг. 4 представляет собой структурную схему, изображающую иллюстративную систему, полезную для описания аспектов настоящего изобретения, а на фиг. 5 и 6 показана блок-схема алгоритма иллюстративного способа сопровождения обновляемого файла для случая, когда два пользователя производят изменения в одном и том же базовом файле. В этом примере предположим, что сервер 400

сопровождает базовый файл (Версию А), а два клиента 410, 420 (именуемые здесь как клиенты 1 и 2, соответственно) оба намерены произвести изменения в одном и том же базовом файле.

На этапе 500 как клиент 1, так и клиент 2 запрашивают и получают последнюю версию («Версию А») базового файла от сервера 400 (то есть клиенты 1 и 2 загружают базовый файл). Предполагается, что клиенты 1 и 2 могут производить изменения в базовом файле параллельно или последовательно во времени. Тем не менее, только один клиент будет первым при передаче своих изменений первоначального базового файла на сервер. Эти-то изменения и будут применены к первоначальному базовому файлу. Таким образом, для первого из клиентов, который отправит различие на сервер, это различие считается сервером приемлемым. Различие следующего клиента, основанное на этом базовом файле, будет отвергнуто сервером. Следовательно, клиент, который отправляет свои изменения первоначального файла позже, должен сначала получить обновленный базовый файл, а затем произвести изменения в этом обновленном базовом файле, как это более подробно описывается ниже.

Предположим, что клиент 1 производит свои изменения первым, при этом способ проходит этапы, аналогичные этапам с 200 по 260, показанным на фиг. 2. Это означает, что клиент 1 производит на этапе 505 изменения в Версии А, чтобы создать Версию А'. На этапе 510 клиент 1 сохраняет копию первоначальной Версии А и новой Версии А'. Предполагается, что копия Версии А может быть сохранена у клиента 1 либо до этапа 505, либо после этапа 505. После этого на этапе 515 посредством сравнения Версий А и А' генерируется «диф» (предпочтительно, двоичный «диф»).

На этапе 520 клиент 1 отправляет это различие на сервер 400. Когда синхронизация переходит назад к серверу, клиент проверяет, поддерживает ли сервер механизм «дифов» и после этого загружает на сервер «диф» вместе с описывающей версию информацией, которая специфицирует версию первоначального файла. Сервер 400 после проведения проверки, подтверждающей, что имеющаяся у него последняя версия базового файла не изменилась по сравнению с Версией А, которую клиент использовал для внесения изменений, применяет на этапе 525 предоставленный клиентом 1 «диф» Версии А, чтобы сгенерировать новую, самую последнюю версию файла, Версию В.

На этапе 530 сервер сохраняет новую Версию В, равно как и представленный клиентом 1 «диф». Новая Версия В рассматривается как самое последнее обновление базового файла, а «диф» сохраняется для использования другими клиентами (например, клиентом 2), которые могут производить изменения в первоначальной Версии А.

Аналогично тому, как это делалось на этапах 270 и 280, хотя это и не показано на фиг. 5, сервер 400 сообщает клиенту 1 идентификатор новой версии (например, «Версия В»). После этого клиент 1 аннулирует определенный им «диф», равно как и сохраненную им Версию А, и клиент присваивает своей Версии А' идентификатор новой версии. Таким образом, клиент переименовывает Версию А' в Версию В.

Тем временем на этапе 535 клиент 2 производит изменение полученной им Версии А первоначального базового файла, чтобы создать новую версию, Верию А”. На этапе 540 клиент 2 сохраняет копию первоначальной Версии А и новую Версию А”. Предполагается, что копия Версии А может быть сохранена на клиенте 2 либо до этапа 540, либо после этапа 540. После этого на этапе 545 посредством сравнения Версий А и А” генерируется «диф».

На этапе 550 клиент 2 отправляет свой «диф» Версии А на сервер 400. Сервер 400

проводит проверку, чтобы определить, не изменился ли базовый файл, который хранится у него, по сравнению с базовым файлом, который клиент 2 использовал как базу для произведенных клиентом 2 изменений.

Если состояние базового файла на сервере было занесено в кэш в локальном запоминающем устройстве, связанном с запрашивающим клиентом, то производится сравнение файла, размещенного в локальном запоминающем устройстве, с состоянием соответствующего файла на сервере. Это сравнение производится с той целью, чтобы определить, является ли копия файла, хранящаяся в локальном запоминающем устройстве, самой последней версией или на сервере имеется более поздняя версия. Иначе говоря, это сравнение учитывает возможность того, что другой клиент изменил или обновил запрошенный файл за время, прошедшее с того момента, когда запрашивающий клиент получил эту копию файла. Следует отметить, что сравнение по возможности должно включать в себя передачу идентификатора, представляющего состояние файла, не требуя при этом осуществлять передачу всего файла между клиентом и сервером. Таким образом, сравнение устраняет сетевой трафик, который мог бы потребоваться в противном случае, и освобождает от необходимости передавать одну и ту же версию файла более чем один раз.

Таким образом, на этапе 555 сервер проводит проверку, чтобы убедиться в том, что имеющаяся у него последняя версия базового файла не изменилась по сравнению с Версией A, которую клиент использовал при внесении изменений. Если базовый файл не изменился, то сервер 400 применяет на этапе 590 предоставленный клиентом 2 «диф» к хранящемуся на сервере базовому файлу, чтобы сгенерировать новую, самую последнюю версию файла, которую сервер сохраняет наряду с «дифом», предоставленным клиентом 2. При этом сервер 400 сообщил бы клиенту 2 идентификатор новой версии, а клиент 2 затем аннулировал бы определенный им «диф», равно как и хранимую им Версию A и присвоил бы своей измененной Версии A” идентификатор новой версии.

Однако в данном примере клиент 1 уже предоставил изменения на сервер 400, так что базовый файл, хранящийся на сервере, изменился и стал Версией B. Клиент 2 не располагает копией Версии B и внсит свои изменения в Версию A файла. Следовательно, поскольку базовый файл изменился, сервер 400 на этапе 560 отвергает «диф», предоставленный клиентом 2, и отправляет клиенту 2 предоставленный клиентом 1 «диф» Версии A, который сервер 400 получил и сохранил ранее (на этапе 530).

На этапе 565 клиент 2 применяет «диф» Версии A, предоставленный клиентом 1, к хранящейся у него Версии A, чтобы получить самую последнюю хранящуюся на сервере версию файла (здесь Версию B). После этого клиент 2 на этапах 570 и 575, соответственно, определяет «диф» (различие) между самой последней версией и своей измененной Версией A” и отправляет этот «диф» на сервер 400. На этапе 585 сервер 400 применяет новый «диф» к хранящейся у него самой последней версии (Версии B), чтобы сгенерировать новую самую последнюю версию (здесь Версию C). На этапе 585 сервер 400 сохраняет новую самую последнюю версию, равно как только что полученный «диф». Аналогично тому, как это делалось на этапах 270 и 280, хотя и не показано на Фиг.5, сервер 400 сообщает клиенту 2 идентификатор новой версии (например «Версия C»). После этого клиент 2 аннулирует определенный им «диф», равно как и сохраненную им версию, и клиент присваивает своей Версии A” идентификатор новой версии. Таким образом, клиент переименовывает Версию A” в Версию C.

Предусматривается, чтобы вместо автоматического сохранения измененной версии на сервере пользователь, такой как администратор, мог бы определять то, каким образом изменения должны быть интегрированы в файл. Это может помочь избежать конфликтов контента (информационно значимого содержимого) с изменениями, произведенными предыдущим пользователем.

Следует отметить, что «диф» может быть определен либо до, либо после того, как сервер выразил свое согласие принять «диф». Таким образом, для повышения эффективности клиент может ждать до тех пор, пока сервер не укажет, что клиент произвел изменения в той же самой версии базового файла, которую сервер в настоящий момент сопровождает как самую последнюю версию. Только после этого клиент определял бы «диф» и предоставлял бы его серверу. Желательно, чтобы сервер не вычислял различие, а вместо этого лишь применял это различие.

Предусматривается ситуация, при которой один клиент производит несколько загрузок в сервер, прежде чем другой клиент соединяется с сервером, чтобы предоставить ему свои изменения. Например, предположим, что первоначальный базовый файл является Версией A. Затем клиент 1 производит изменения, и эти изменения принимаются как Версия B. Если клиент 1 производит поледующие изменения и предоставляет их серверу, то эта новая самая последняя версия будет сохранена как Версия C. Желательно, чтобы сервер сохранял различие между Версиями A и B и различие между Версиями B и C. При этом, когда другой клиент производит изменения, сервер будет отправлять этому клиенту различие между Версиями A и B и различие между Версиями B и C, предпочтительно в одном и том же сообщении. После этого клиент воссоздает Версию B, затем Версию C, определяет различие между Версией C и ее изменениями и предоставляет это различие серверу.

Если клиент, имеющий устаревшую версию, обращается к серверу за получением самой последней версии, то он сообщает серверу о том, какую версию он имеет, и если хранящиеся на сервере «дифы» восходят к дате этой версии, то клиенту возвращается соответствующий «диф» или «дифы» вместе с идентификатором текущей версии. Предпочтительно, чтобы сервер поддерживал все «дифы» между различными версиями, которые он получает во время обработки данных, что необходимо для того, чтобы обслуживать тех клиентов, которые могут все еще производить изменения в старых версиях, (то есть для того, чтобы иметь «обратную совместимость» со старыми версиями базового файла). Однако в некоторый момент времени сервер может удалить или иным образом ликвидировать ранее сохраненные «дифы», которые он поддерживал. Такое действие может быть продиктовано, например, датой или емкостью запоминающего устройства.

Желательно, чтобы для передачи «дифов» использовался Протокол передачи гипертекста (HTTP). В частности, расширения протокола могут использоваться для предупреждения сервера о том, что в этот момент производится передача «дифа», или что «диф» иным образом внедрен или вставлен в состав сообщения.

Протокол передачи гипертекста HTTP был создан в качестве стандартного механизма, посредством которого информация транспортируется по сетям, совместимым с протоколом TCP/IP (Протоколом управления передачей / Межсетевым протоколом), таким как сеть Интернет, интрасети и экстрасети. Более конкретно, протокол HTTP представляет собой протокол прикладного уровня, предназначенный для распределенных ориентированных на поддержку пользователя информационных систем гипермедиа (расширенного по сравнению с гипертекстом метода организации мультимедийной информации). Он представляет собой простой протокол общего

назначения, который посредством расширения его методов запроса, кодов ошибок и заголовков может быть использован для решения многих задач, выходящих за рамки его использования для передачи гипертекста, таких, например, задач как блоки преобразования имен и системы управления распределенными объектами. Его называют транспортным протоколом, поскольку в соответствии с его спецификациями осуществляется транспортировка информации, и его также называют протоколом типа «запрос-ответ», поскольку информация поступает в обмен на подачу клиентом запроса серверу, который генерирует ответ на этот запрос. Протокол HTTP в том значении, в котором он упоминается в данном документе, относится в широком смысле к любому стандарту HTTP, и с ним можно ознакомиться на web-сайте http://www.w3.org.

Общепринятым направлением использования протокола HTTP является транспортировка информации, сформатированной в соответствии с языком разметки. В таких случаях запрашиваемая информация при транспортировке в соответствии с протоколом HTTP обычно представлена в формате Языка разметки гипертекста (HTML). Однако появляются и другие стандартные языки разметки. Одним из таких языков разметки является Расширяемый язык разметки (XML). Язык XML описывает класс объектов данных, именуемых XML-документами, и частично описывает поведение компьютерных программ, которые их обрабатывают. Основное различие между HTML и XML заключается в том, что в первом из них информационное содержимое взаимоувязано с описанием компоновки содержимого, что, например, делает сложным их разделение. Напротив, в языке XML описание компоновки данных в памяти и логическая структура содержимого поддерживается отдельно от самого содержимого. Тем не менее, оба языка: как XML, так и HTML, представляют собой производные от языка разметки, известного как Стандартный обобщенный язык разметки (SGML). Язык XML в том значении, в котором он упоминается в данном документе, относится в широком смысле к любому стандарту XML, который описан на web-сайте http://www.w3.org.

Для поддержания обратной совместимости и возможностей взаимодействия, например, может быть использован расширенный заголовок Протокола передачи гипертекста HTTP в ответе OPTIONS (ОПЦИИ), что позволяет клиенту определить, что сервер поддерживает двоичные «дифы». Расширенный заголовок в запросах GET (запросах на ПОЛУЧЕНИЕ) уведомляет сервер о том, что клиент принимает «дифы».

Как клиент, так и сервер могут принять решение не использовать двоичный «диф». В некоторых случаях вполне может оказаться, что сообщение, передающее «диф», (например, заголовок двоичного «дифа» в протоколе HTTP) будет больше, чем сам файл. В таком случае может оказаться более предпочтительным пересылать между сервером и клиентами ам документ, а не «диф». Клиент может определить, что размер «дифа» больше, чем новый файл. Это может произойти, например, в том случае, если новый файл имеет 0 байтов. Сервер может принять решение аннулировать «дифы» для того, чтобы сберечь место в памяти. Если «диф» не используется, то пересылается файл целиком. Чтобы подать сигнал, что пересылается «диф», клиент, может вместе со своим запросом PUT (запросом на РАЗМЕЩЕНИЕ) направить расширенный заголовок (расширенные заголовки), указывающий (указывающие) на присутствие в теле сообщения двоичного «дифа» и сообщающий (сообщающие) номер версии базового файла, на основе которой сгенерирован «диф». Сервер отправляет вместе со своим ответом GET (ответом ПОЛУЧИ) расширенный заголовок (расширенные заголовки), указывающие на присутствие цепочки двоичных «дифов» и сообщающего

(сообщающих) номер версии базового файла и номер «дифов» в цепочке.

В случае, когда для приведения файла клиента в самое свежее состояние, требуется множество «дифов», сервер может принять решение либо соединить «дифы» вместе в виде цепочки в одном сообщении, либо, если сумма «дифов» больше, чем новая версия файла, отправить клиенту саму новую версию.

Предпочтительно, чтобы средство для вычисления и применения «дифов» имелось на клиентах и/или на сервере (серверах). Также предпочтительно, чтобы был реализован протокол для обнаружения «дифов» и управления «дифами»/версиями. Расширенные заголовки протокола HTTP позволяют клиенту и серверу выразить свою способность в области определения различий между файлами. Например, клиент отправляет вместе со своим запросом PUT расширенный заголовок (расширенные заголовки), указывающий (указывающие) на присутствие в теле сообщения двоичного «дифа» и сообщающий (сообщающие) номер версии базового файла, на основе которой сгенерирован «диф». Сервер отправляет вместе со своим ответом GET расширенный заголовок (расширенные заголовки), указывающие на присутствие цепочки двоичных «дифов», сообщающего (сообщающих) номер версии базового файла и номер «дифов» в цепочке.

Желательно, чтобы код сервера был способен управлять «дифами», применять «дифы» (используя вышеупомянутое средство), хранить их и высылать их клиенту, и предпочтительно, чтобы каждый клиент был способен сопровождать состояние сервера, генерировать «дифы», передавать «дифы» на сервер и применять полученные от сервера «дифы».

Репликация широко используется в большом разнообразии приложений, но стоимость и производительность этих систем представляют собой постоянную прблему. Настоящее изобретение превращает саму сущность репликации (известное состояние клиента/сервера) в рычаг, позволяющий резко повысить эффективность системы. Настоящее изобретение может применяться к системам, осуществляющим репликацию больших файлов, обновляемых на постоянной основе, таких как информационные продукты, допускающие доступ к базирующимся на сервере документам в автономном режиме.

Как было отмечено выше, хотя иллюстративные варианты осуществления настоящего изобретения были описаны в связи с различными вычислительными устройствами, концепции, лежащие в его основе, могут быть применены к любому вычислительному устройству или системе.

Различные методики, описанные в данном изобретении, могут быть реализованы в связи с аппаратными средствами или программными средствами, или там, где это целесообразно, в связи с комбинацией тех и других средств. Таким образом, способы и устройство, предлагаемые в настоящем изобретении, или определенные их аспекты или части могут принимать форму программного кода (то есть команд) воплощенных на материальных носителях информации, таких как накопители на гибких магнитных дисках, на компакт-дисках (CD-ROM), на жестких магнитных дисках или любом другом машиночитаемом носителе информации, причем, когда этот программный код загружен в машину, такую как компьютер, и исполнен этой машиной, эта машина превращается в устройство для осуществления данного изобретения. В случае исполнения программного кода на программируемых вычислительных машинах вычислительное устройство в общем случае будет включать в себя процессор, носитель информации, который может считываться процессором (включая энергозависимую и энергонезависимую память и/или запоминающие элементы), по

меньшей мере, одно устройство ввода и, по меньшей мере, одно устройство вывода. При необходимости программа (программы) могут быть реализованы на языке ассемблера или машинном языке. В любом случае язык может представлять собой транслируемый или интерпретируемый язык и может сочетаться с вариантами аппаратной реализации.

Способы и устройство, предлагаемые в настоящем изобретении, также могут быть осуществлены посредством коммуникаций, воплощенных в форме программного кода, который передается по некой среде передачи данных, как, например, по электрическим проводам или кабелям, по оптоволоконным каналам, причем, когда программный код принят, загружен и исполнен машиной, такой как стираемое программируемое постоянное запоминающее устройство (СППЗУ, EPROM), вентильная матрица, программируемое логическое устройство (PLD), компьютер-клиент и тому подобное, машина превращается в устройство для осуществления данного изобретения. При реализации на процессоре общего назначения программный код, объединяясь с процессором, образует уникальное устройство, которое производит операции, порождающие функциональные возможности настоящего изобретения. Кроме того, любые способы хранения информации, используемые в связи с настоящим изобретением, могут неизменно являться сочетанием аппаратных и программных средств.

Хотя настоящее изобретение было описано в связи с предпочтительными вариантами осуществления изобретения, показанными на различных фигурах, следует понимать, что могут быть использованы и другие аналогичные варианты осуществления или что в описанные варианты осуществления могут быть внесены изменения и дополнения для выполнения той же самой функции настоящего изобретения, и это не выходит за его рамки. В силу этого настоящее изобретение не должно сводиться к какому-либо одному варианту своего осуществления, но, наоборот, должно толковаться во всей широте и объеме, определенными в соответствии с прилагаемой формулой изобретения.

Формула изобретения

1. Способ сопровождения обновляемого файла, включающий в себя этапы, на которых

сохраняют первую копию и вторую копию базового файла на первом устройстве и на втором устройстве;

принимают первый набор изменений в отношении первой копии на первом устройстве и второй набор изменений в отношении первой копии на втором устройстве;

определяют первое различие между измененной первой копией и второй копией на первом устройстве и второе различие между измененной первой копией и второй копией на втором устройстве;

передают первое различие и второе различие на сервер;

принимают первое различие или второе различие первым по времени на сервере;

считают приемлемым различие, принятое первым по времени на сервере, если базовый файл, имеющийся на сервере, является таким же, как базовый файл, который был сохранен на устройстве, ассоциированном с различием, принятым первым по времени, в противном случае отклоняют это различие на сервере;

отклоняют различие, принятое вторым по времени на сервере; и

передают третье различие от сервера на устройство, ассоциированное с различием,

принятым вторым по времени, и применяют третье различие ко второй копии базового файла, сохраненной на упомянутом устройстве.

2. Способ по п.1, дополнительно включающий в себя этап, на котором принимают от сервера первую копию файла на первом устройстве и создают вторю копию файла на первом устройстве перед сохранением первой копии и второй копии базового файла на первом устройстве.

3. Способ по п.1, в котором при определении первого различия используют двоичные сравнения, проводимые между измененной первой копией и второй копией.

4. Способ по п.1, в котором первое различие является двоичной разницей.

5. Способ по п.1, в котором, если различие отклонено на сервере, то передают с сервера дополнительное различие на первое устройство и применяют это дополнительное различие ко второй копии базового файла, сохраненной на устройстве, ассоциированном с различием, принятым первым по времени.

6. Способ по п.1, дополнительно включающий в себя этап, на котором определяют, является ли базовый файл на сервере таким же, как базовый файл, который был сохранен на первом устройстве.

7. Способ по п.1, в котором при передаче первого различия используют расширения к Протоколу передачи гипертекста (HTTP).

8. Способ по п.1, дополнительно включающий в себя этапы, на которых

принимают от сервера первую копию базового файла на втором устройстве и

создают вторую копию базового файла на втором устройстве перед сохранением первой копии и второй копии базового файла на втором устройстве.

9. Способ по п.1, в котором при определении второго различия используют двоичные сравнения, проводимые между измененной первой копией и второй копией на втором устройстве.

10. Способ по п.1, в котором второе различие является двоичной разницей.

11. Способ по п.1, дополнительно включающий в себя этап, на котором определяют, является ли базовый файл на сервере таким же, как базовый файл, который был сохранен на втором устройстве.

12. Способ по п.1, в котором при передаче второго различия используют расширения к протоколу HTTP.

13. Машиночитаемый носитель информации с хранящимися на нем машиноисполняемыми командами для выполнения способа сопровождения обновляемого файла, включающего в себя

сохранение первой копии и второй копии базового файла на первом устройстве и на втором устройстве;

прием первого набора изменений в отношении первой копии на первом устройстве и второго набора изменений в отношении первой копии на втором устройстве;

определение первого различия между измененной первой копией и второй копией на первом устройстве и второго различия между измененной первой копией и второй копией на втором устройстве;

передачу первого различия и второго различия на сервер;

прием первого различия или второго различия первым по времени на сервере;

принятие различия, принятого первым по времени на сервере, если базовый файл, имеющийся на сервере, является таким же, как базовый файл, который был сохранен на устройстве, ассоциированном с различием, принятым первым по времени, в противном случае отклонение этого различия на сервере;

отклонение различия, принятого вторым по времени на сервере;

и

передачу третьего различия от сервера на устройство, ассоциированное с различием, принятым вторым по времени, и применение третьего различия ко второй копии базового файла, сохраненной на упомянутом устройстве.

14. Машиночитаемый носитель информации по п.13, дополнительно содержащий машиноисполняемые команды для приема от сервера первой копии файла на первом устройстве и создания второй копии файла на первом устройстве перед сохранением первой копии и второй копии базового файла на первом устройстве.

15. Машиночитаемый носитель информации по п.13, в котором определение первого различия содержит использование двоичных сравнений, проводимых между измененной первой копией и второй копией.

16. Машиночитаемый носитель информации по п.13, в котором первое различие является двоичной разницей.

17. Машиночитаемый носитель информации по п.13, дополнительно содержащий машиноисполняемые команды для передачи с сервера, если различие отклонено на сервере, дополнительного различия на первое устройство и применения этого дополнительного различия ко второй копии базового файла, сохраненной на устройстве, ассоциированном с различием, принятым первым по времени.

18. Машиночитаемый носитель информации по п.13, дополнительно содержащий машиноисполняемые команды для определения того, является ли базовый файл на сервере таким же, как базовый файл, который был сохранен на первом устройстве.

19. Машиночитаемый носитель информации по п.13, в котором передача первого различия включает в себя использование расширений к протоколу HTTP.

20. Машиночитаемый носитель информации по п.13, дополнительно содержащий машиноисполняемые команды для

приема от сервера первой копии базового файла на втором устройстве и

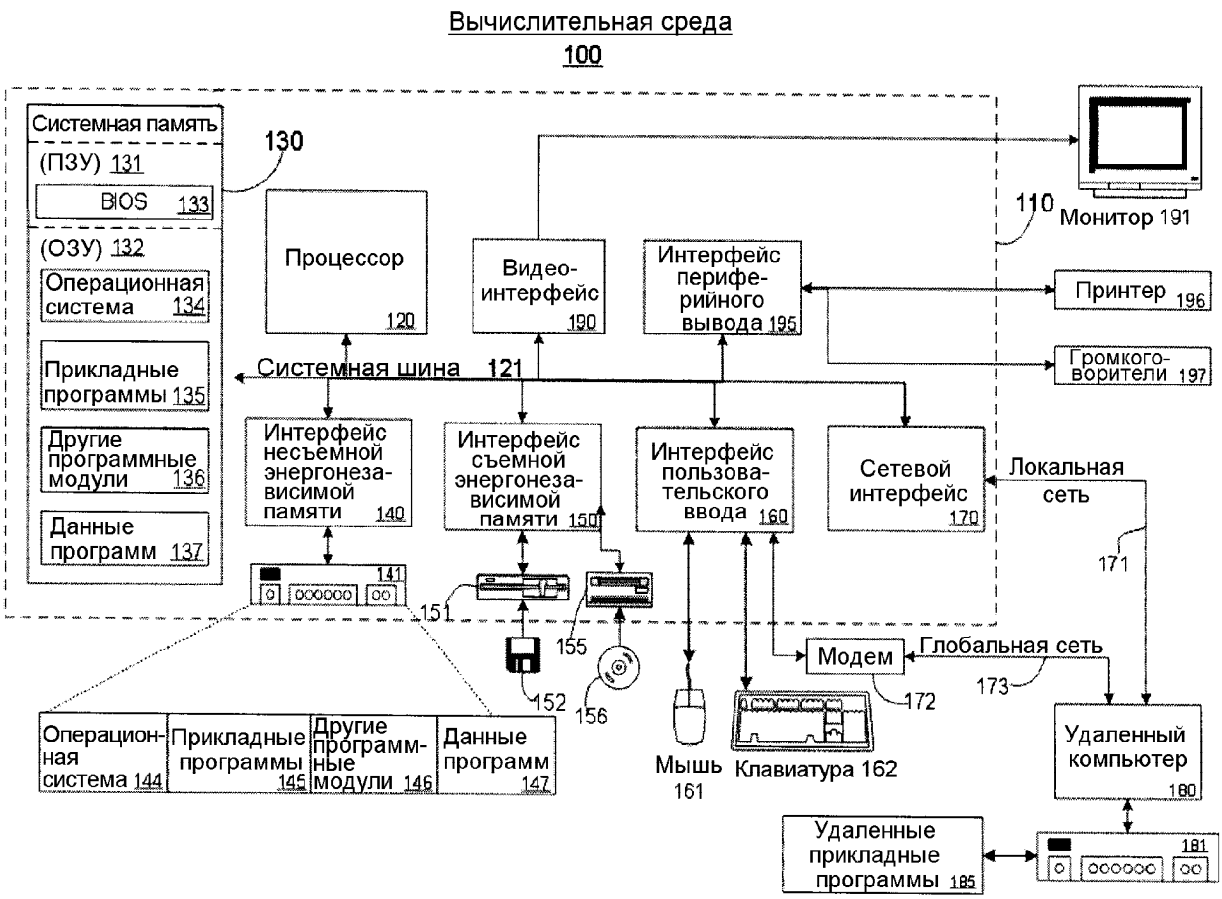
создания второй копии базового файла на втором устройстве перед сохранением первой копии и второй копии базового файла на втором устройстве.

21. Машиночитаемый носитель информации по п.13, в котором определение второго различия включает в себя использование двоичных сравнений, проводимых между измененной первой копией и второй копией на втором устройстве.

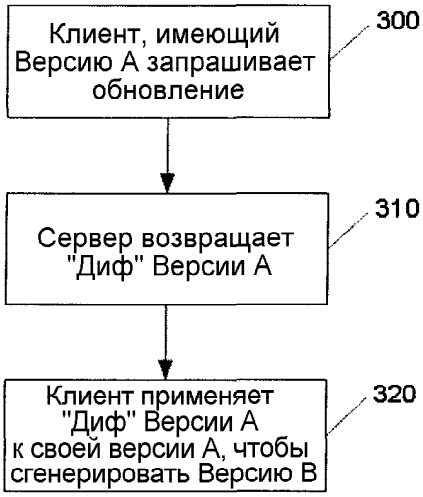
22. Машиночитаемый носитель по п.13, в котором второе различие является двоичной разницей.

23. Машиночитаемый носитель информации по п.13, дополнительно содержащий машиноисполняемые команды для определения того, является ли базовый файл на сервере таким же, как базовый файл, который был сохранен на втором устройстве.

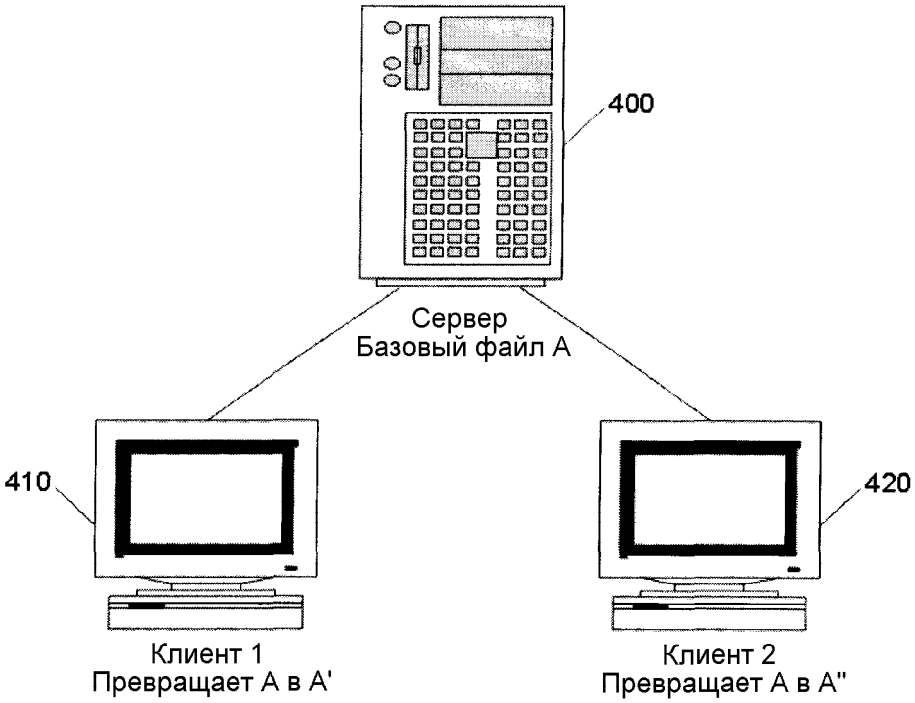
24. Машиночитаемый носитель информации по п.13, в котором передача второго различия включает в себя использование расширений к протоколу HTTP.


Вычислительная среда
100
Системная память
(ПЗУ) 131
BIOS 133
(ОЗУ) 132
Операционная система 134
Прикладные программы 135
Другие программные модули 136
Данные программ 137
130
Процессор 120
Видео-интерфейс 190
Интерфейс перифе-рийного вывода 195
Монитор 191
110
Принтер 196
Громкого-ворители 197
Системная шина 121
Интерфейс несъемной энергонеза-висимой памяти 140
Интерфейс съемной энергонеза-висимой памяти 150
Интерфейс пользова-тельского ввода 160
Сетевой интерфейс 170
Локальная сеть
171
141
151
155
152
156
Модем
Глобальная сеть
172
173
Операцион-ная система 144
Прикладные программы 145
Другие программ-ные модули 146
Данные программ 147
Мышь 161
Клавиатура 162
Удаленный компьютер 180
Удаленные прикладные программы 185
181

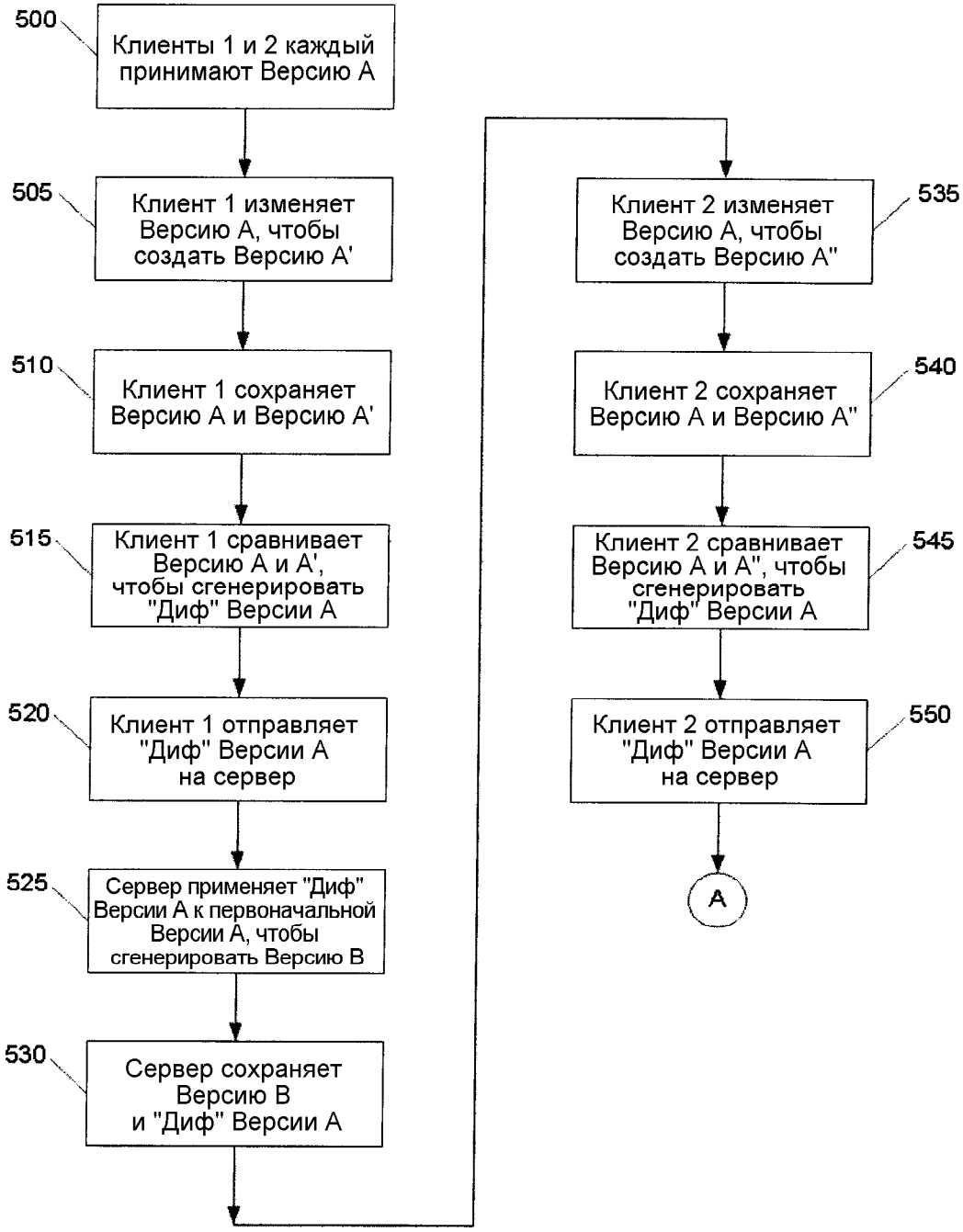
Фиг. 1


Клиент, имеющий Версию А запрашивает обновление
300
Сервер возвращает "Диф" Версии А
310
Клиент применяет "Диф" Версии А к своей версии А, чтобы сгенерировать Версию В
320

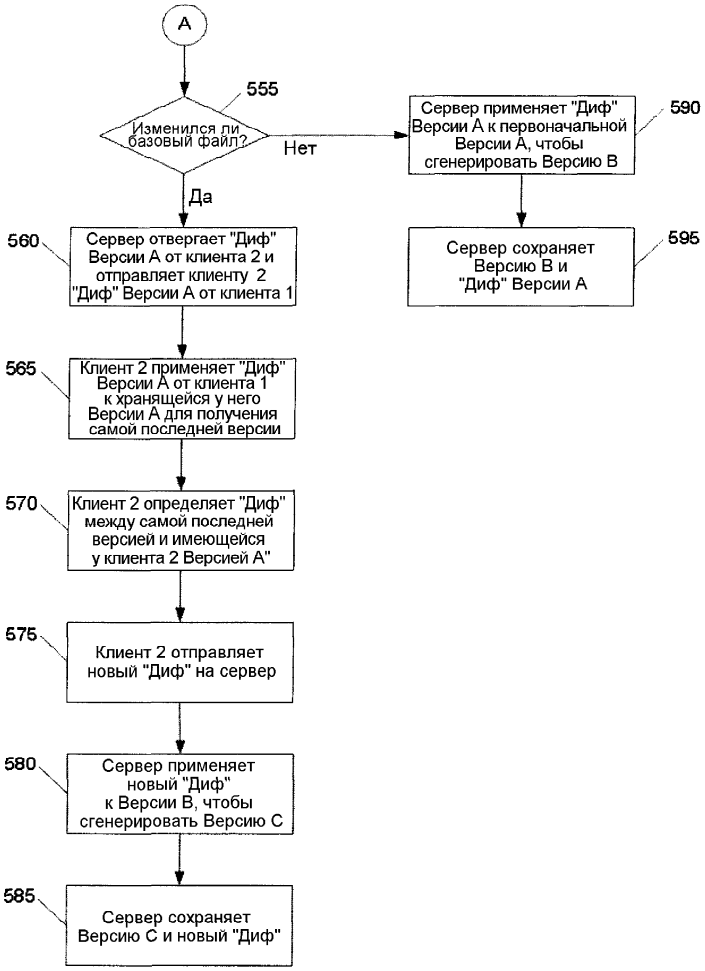
Фиг. 3


400
Сервер
Базовый файл A
410
420
Клиент 1
Превращает A в A'
Клиент 2
Превращает A в A''

Фиг. 4


500
Клиенты 1 и 2 каждый принимают Версию A
505
Клиент 1 изменяет Версию A, чтобы создать Версию A'
510
Клиент 1 сохраняет Версию A и Версию A'
515
Клиент 1 сравнивает Версию A и A', чтобы сгенерировать "Диф" Версии A
520
Клиент 1 отправляет "Диф" Версии A на сервер
525
Сервер применяет "Диф" Версии A к первоначальной Версии A, чтобы сгенерировать Версию B
530
Сервер сохраняет Версию B и "Диф" Версии A
535
Клиент 2 изменяет Версию A, чтобы создать Версию A"
540
Клиент 2 сохраняет Версию A и Версию A"
545
Клиент 2 сравнивает Версию A и A", чтобы сгенерировать "Диф" Версии A
550
Клиент 2 отправляет "Диф" Версии A на сервер
A

Фиг. 5


A
555
Изменился ли базовый файл?
Нет
Да
590
Сервер применяет "Диф" Версии А к первоначальной Версии А, чтобы сгенерировать Версию В
595
Сервер сохраняет Версию В и "Диф" Версии А
560
Сервер отвергает "Диф" Версии А от клиента 2 и отправляет клиенту 2 "Диф" Версии А от клиента 1
565
Клиент 2 применяет "Диф" Версии А от клиента 1 к хранящейся у него Версии А для получения самой последней версии
570
Клиент 2 определяет "Диф" между самой последней версией и имеющейся у клиента 2 Версией А"
575
Клиент 2 отправляет новый "Диф" на сервер
580
Сервер применяет новый "Диф" к Версии В, чтобы сгенерировать Версию С
585
Сервер сохраняет Версию С и новый "Диф"

Фиг. 6