РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 286 595** (13) **C2**

(51) МПК
G06F 9/44 (2006.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ,
ПАТЕНТАМ И ТОВАРНЫМ ЗНАКАМ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21), (22) Заявка: **2003118641/09, 26.10.2001**

(24) Дата начала отсчета срока действия патента:
26.10.2001

(30) Конвенционный приоритет:
24.11.2000 US 09/721,695

(43) Дата публикации заявки: **10.12.2004**

(45) Опубликовано: **27.10.2006 Бюл. № 30**

(56) Список документов, цитированных в отчете о поиске: **RU 2147378 C1, 10.04.2000. RU 2042193 C1, 20.08.1995. US 6108744 A, 22.08.2000. US 6092144 A, 18.07.2000.**

(85) Дата перевода заявки PCT на национальную фазу:
24.06.2003

(86) Заявка PCT:
US 01/51441 (26.10.2001)

(87) Публикация PCT:
WO 03/007105 (23.01.2003)

Адрес для переписки:
129010, Москва, ул. Б. Спасская, 25, стр.3, ООО "Юридическая фирма Городисский и Партнеры", пат.пов. Ю.Д.Кузнецову, рег.№ 595

(72) Автор(ы):
ФАЙНБЕРГ Мэттью А. (US)

(73) Патентообладатель(и):
КАТАРОН ПРОДАКШНЗ, ИНК. (US)

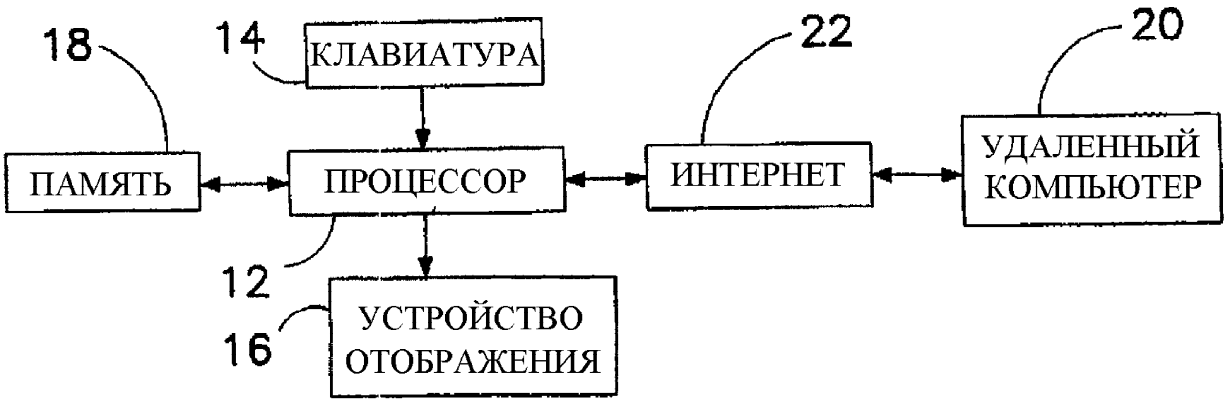
RU 2286595 C2

(54) РЕАЛИЗАЦИЯ КОМПЬЮТЕРНОЙ МНОГОЗАДАЧНОСТИ ЧЕРЕЗ ВИРТУАЛЬНУЮ ОРГАНИЗАЦИЮ ПОТОЧНОЙ ОБРАБОТКИ

(57) Реферат:

Изобретение относится к способу и устройству для одновременного выполнения многочисленных задач на компьютере. Технический результат заключается в расширении функциональных возможностей за счет организации многопоточной обработки. При работе компьютера множество команд байт-кода или псевдокода сохраняют в памяти компьютера. Для каждого из множества задач или заданий, которые выполняет компьютер, автоматически создается соответствующий виртуальный поток выполняемых контекстных данных, который включает в себя местоположение в памяти следующей одной из команд псевдокода, которые должны выполняться при выполнении соответствующей задачи или задания и значения любых требуемых локальных переменных. Каждую задачу или задание обрабатывают в соответствующей последовательности квантов времени или интервалов времени обработки под управлением соответствующего виртуального потока. 5 н. и 40 з.п. ф-лы, 11 ил.

RU 2286595 C2



ФИГ.1

RU 2286595 C2

RU 2286595 C2

RUSSIAN FEDERATION

(19) **RU** (11) **2 286 595** (13) **C2**

(51) Int. Cl.
G06F 9/44 (2006.01)

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY,
PATENTS AND TRADEMARKS

(12) ABSTRACT OF INVENTION

(21), (22) Application: **2003118641/09, 26.10.2001**

(24) Effective date for property rights: **26.10.2001**

(30) Priority:
24.11.2000 US 09/721,695

(43) Application published: **10.12.2004**

(45) Date of publication: **27.10.2006 Bull. 30**

(85) Commencement of national phase: **24.06.2003**

(86) PCT application:
US 01/51441 (26.10.2001)

(87) PCT publication:
WO 03/007105 (23.01.2003)

Mail address:
129010, Moskva, ul. B. Spasskaja, 25, str.3, OOO "Juridicheskaja firma Gorodisskij i Partnery", pat.pov. Ju.D.Kuznetsovu, reg.№ 595

(72) Inventor(s):
FAJNBERG Mehtt'ju A. (US)

(73) Proprietor(s):
KATARON PRODAKShNZ, INK. (US)

RU 2286595 C2

(54) METHOD FOR REALIZATION OF COMPUTER MULTI-TASK SYSTEM THROUGH VIRTUAL ORGANIZATION OF THREAD PROCESSING

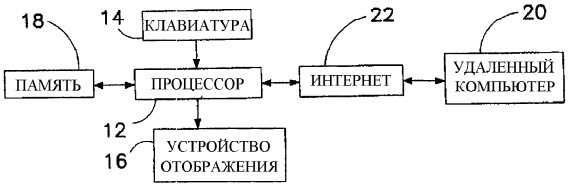
(57) Abstract:

FIELD: computer engineering, possible use for simultaneous execution of multiple tasks on a computer.

SUBSTANCE: during operation of computer a set of commands of byte-code or pseudo-code is stored in computer memory. For each one of a set of tasks or problems executed by computer, generated automatically is an aperture virtual thread of executed context data, which includes: position in memory of next one of pseudo-code commands, which should be executed during execution of appropriate task or problem and values of any required local variables. Each task or problem is processed in appropriate sequence of time quanta or processing time intervals under control of appropriate virtual thread.

EFFECT: expanded functional capabilities, due to organization of multithread processing.

5 cl, 11 dwg



ФИГ.1

RU 2286595 C2

ПРЕДШЕСТВУЮЩИЙ УРОВЕНЬ ТЕХНИКИ

Настоящее изобретение относится к способу и соответствующему устройству для одновременного выполнения многочисленных задач на компьютере.

Наиболее современные операционные системы имеют собственную многозадачность или возможность организации многопоточной обработки, то есть возможность организации многопоточной обработки, встроенной в операционную систему. Известными исключениями являются версии операционной системы Макинтош (Macintosh) (MacOS), предшествующие ОС X (OS X), которые обладают небольшими возможностями или возможностями без организации многопоточной обработки. К сожалению, выполненные возможности организации многопоточной обработки отличаются в зависимости от операционной системы и платформы аппаратных средств. Многочисленные платформы устанавливают пределы на общее количество потоков, которые могут существовать одновременно, и некоторые платформы вообще не позволяют организовать многопоточную обработку.

Для того чтобы правильно понять приведенные проблемы организации многопоточной обработки в программном обеспечении и их решения, необходимо понять общий подход к организации многопоточной обработки и специфический подход, используемый большинством систем организации поточной обработки с собственной платформой.

Для того чтобы сразу выполнить многочисленные задачи обработки, очевидным решением является обеспечение многочисленных наборов схем обработки в компьютерной системе. Однако обычный настольный компьютер имеет только один процессор, и даже мощные рабочие станции имеют не более одного - четырех процессоров.

Решением на основе программного обеспечения является квантование времени, то есть разделение времени процессора на ряд очень маленьких частей (квантов), и закрепление за каждым квантом, в свою очередь, различной задачи или потока. Как правило, каждый поток может выполнять работу в интервале от 3 мс до 30 мс в зависимости от операционной системы, и по истечении этого времени работа потока приостанавливается, после чего другой поток может выполнять работу. Операционная система обычно использует прерывание процессора по таймеру для периодического прерывания текущего выполняемого потока и запуска планировщика потоков операционной системы, часть программного обеспечения которого сохраняет состояние текущего потока или контекст выполнения, выбирает новый поток для выполнения работы, восстанавливает сохраненный контекст выполнения нового потока и затем позволяет процессору продолжить нормальное выполнение своих функций. Этот процесс называется контекстным переключением.

В дополнение к происходящему, когда квант времени истек, может также происходить контекстное переключение, если поток входит в состояние ожидания, то есть в состояние, в котором поток не должен ничего выполнять до тех пор, пока не произойдет специфическое событие. Когда поток входит в такое состояние, активизируется планировщик потока, и контекстное переключение происходит так, чтобы некоторый другой поток мог использовать остаток кванта времени.

Типичное событие, которое вызывает переход потока в состояние ожидания, происходит тогда, когда поток пытается обратиться к памяти, которая была разбита на страницы на диске. Операционная система приостанавливает поток до тех пор, пока система памяти не будет иметь шанс выполнить разбиение на страницы в памяти. Другими событиями, вызывающими переход потока в состояние ожидания, являются проверка потока для пользовательского ввода и попытка потока выполнить считывание с диска. В последнем случае операционная система приостанавливает поток до тех пор, пока не завершится считывание диска, позволяя другим потокам выполнять задачи обработки, в то время как первый поток ожидает считывания данных с диска. Еще одно событие, которое может стимулировать переход потока в состояние ожидания, происходит тогда, когда поток специфическим образом предоставляет остаток своего кванта времени. Это может происходить в случае, если, например, поток в течение некоторого времени не выполняет

никакой работы.

Так как контекстные переключения могут происходить с большой частотой, это критично для контекстного переключения, работающего чрезвычайно быстро. В многочисленных операционных системах установлены пределы на число потоков, которые могут существовать в системе. Windows 95 имеет максимально число, равное приблизительно 150-200 потокам перед тем, как система станет нестабильной, тогда как BeOS имеет максимум 4096 потоков на один процессор. Такое ограничение по числу потоков является результатом предварительного выделения операционной системой (по причинам производительности) при загрузке системы участка памяти фиксированного размера для таблицы потоков.

Стандартные неинтерпретируемые языки программирования компилируют исходный код, удобный для восприятия человеком, в машиночитаемый код или код на машинном языке, непосредственно считываемый с помощью процессора.

Интерпретируемый язык, с другой стороны, компилирует код, удобный для восприятия человеком, в код, считываемый интерпретатором, или в байт-код. Программа, реализованная программно, которая называется интерпретатором и написана на машинном языке, далее считывает байт-код и выдает команды процессору на выполнение соответствующих операций.

Основное преимущество интерпретируемого языка заключается в том, что байт-код можно выполнить так, чтобы машина была независимой, позволяя при этом выполнять программу, написанную на языке и скомпилированную в байт-код, на любой операционной системе и платформе аппаратных средств, для которой был написан интерпретатор.

При разработке интерпретируемого языка, который должен работать одинаковым образом на всех платформах, надежда на собственную организацию многопоточной обработки платформы в лучшем случае может быть проблематичной. Цель межплатформенного интерпретируемого языка программирования состоит в том, чтобы сделать возможным разработку программы на языке на одной платформе и затем выполнять эту программу без изменений на любой другой платформе, поддерживаемой языком. Язык Джава (Java) является одним примером попытки создать такой язык.

К сожалению, если такое приложение будет многопоточным, то использование собственных возможностей организации многопоточной обработки незамедлительно ограничит платформы, на которых может выполняться приложение. Прикладная программа незамедлительно прекращает работу большинства версий MacOS (которые не обладают способностью организации поточной обработки), и в зависимости от того, сколько требуется одновременно выполняемых потоков прикладной программы, это может помешать работе Windows 95 (максимум около 150 - 200 потоков), BeOS (максимум 4096 потоков на один процессор) или других платформ.

СУЩНОСТЬ ИЗОБРЕТЕНИЯ

Настоящее изобретение нацелено на решение вышеупомянутых проблем и обеспечение возможности многозадачности в компьютерах, имеющих различные платформы и различные операционные системы. В соответствии с изобретением независимое от платформы средство организации многопоточной обработки выполнено посредством интерпретатора, т.е. программы, реализованной программно, которая интерпретирует команды, которые составляют интерпретируемый язык программирования. Интерпретатор осуществляет многозадачность путем создания, поддержки и обработки в соответствии с виртуальными потоками. Это решение устраняет зависимость от возможности организации многопоточной обработки на собственной платформе и обеспечивает надежное средство межплатформенной организации многопоточной обработки с использованием интерпретируемого языка программирования.

В типичном интерпретируемом языке программирования каждая команда в программе должна считываться с помощью интерпретатора и поступать в процессор. Это означает, что интерпретируемая программа будет обычно выполняться более медленно, чем программа на машинном языке. Добавление кода в интерпретатор для проверки таймера

или счетчика при каждой машинной кодированной команде, как это делают при известной организации многопоточной обработки или квантовании времени, и выполнение контекстного переключения, когда необходимо, сильно воздействует на производительность интерпретируемой программы.

Соответственно, настоящее изобретение предусматривает выполнение контекстных переключений только между последовательными командами псевдокода, а не во время выполнения псевдокодовых команд. Таким образом, компьютер, использующий многозадачность или многопоточность, как это раскрыто здесь, повторно выполняет команды, эквивалентные многочисленным командам на машинном языке без проверки таймера или счетчика.

Способ работы компьютера содержит, согласно настоящему изобретению, этапы, в соответствии с которыми сохраняют в компьютерной памяти множество псевдокодовых команд, причем, по меньшей мере, некоторые из псевдокодовых команд содержат множество команд в машинных кодах, и для каждого из множества задач или заданий, подлежащих выполнению с помощью компьютера, автоматически создают соответствующий виртуальный поток выполняемых контекстных данных, включающий в себя (a) ячейку памяти следующей одной из псевдокодовых команд, которые должны выполняться при выполнении соответствующей задачи или задания, и (b) значения любых локальных переменных, которые требуются для выполнения соответствующей задачи или задания. Каждая из множества задач или заданий влечет за собой выполнение соответствующей одной из псевдокодовых команд, содержащих множество команд на машинном языке. Способ дополнительно содержит этапы, в соответствии с которыми обрабатывают каждую из задач или заданий в соответствующей последовательности квантов времени или интервалов времени обработки под управлением соответствующего виртуального потока и в каждом контекстном переключении между различными виртуальными потоками совершают такое контекстное переключение только после полного выполнения текущего выполнения одной из псевдокодовых команд.

Вместо принятия известного подхода "виртуальная машина" имитации мелкомодульных команд на машинном языке (подход, используемый Java) виртуальная организация поточной обработки согласно настоящему изобретению использует интерпретируемый язык с более грубыми командами, то есть каждая команда выполняет намного большую задачу.

Например, на машинном языке (или интерпретируемый язык, который имитирует машинный язык) рисование прямоугольника, который может состоять из нескольких сотен или тысячи команд, причем каждая команда устанавливает цвет одного из пикселей внутри прямоугольника.

При виртуальной организации поточной обработки одиночная команда используется для рисования прямоугольника, и интерпретатор обрабатывает остальное на машинном языке. Это означает, что проверки для потенциального контекстного переключения, выполненные после каждой команды, необходимо выполнять только один раз для всей операции рисования прямоугольника, а не после каждого пикселя.

Результатом является то, что интерпретатор может производить многопоточную обработку без серьезного отрицательного воздействия на производительность.

В соответствии с другим признаком настоящего изобретения каждая из виртуальных потоков является частью соответствующего связанного списка виртуальных потоков. Каждый из виртуальных потоков включает в себя указатель на следующий виртуальный поток в соответствующем связанном списке. Компьютерный способ дополнительно содержит для каждого контекстного переключения между различными виртуальными потоками справочные данные указателя выполняемого виртуального потока для определения идентификационной информации следующего виртуального потока, который будет выполняться.

Общей проблемой, с которой сталкиваются при осуществлении многопоточной обработки на собственной платформе, является ограничение числа потоков (как активных,

так и неактивных), которые могут одновременно существовать в системе.

В Windows 95/98, например, если общее количество потоков во всех приложениях достигает приблизительно 150-200, то система становится сильно неустойчивой. Поведение, которое проявляет испытательная система, сталкивающаяся с таким состоянием работы, включает в себя спонтанные перезагрузки, случайный ввод данных с клавиатуры, случайное перемещение мыши, нарушение работы запоминающих устройств и случайные сбои при работе приложений.

Одним результатом более грубых (крупных) наборов команд, используемых при виртуальной организации поточной обработки, является то, что время контекстного переключения не является критическим. Это означает, что, принимая дополнительное время, которое требуется для сохранения потоков в связанном списке, в качестве приемлемого, результатом является то, что виртуальная организация поточной обработки допускает очень большое число потоков, ограниченных только общей памятью, имеющейся в системе.

Настоящее изобретение предусматривает то, что виртуальные потоки, которые управляют или осуществляют промежуточное выполнение задач или заданий с помощью компьютера, сохраняются в множестве связанных списков, включающих в себя список неактивных виртуальных потоков, список активных виртуальных потоков и список поставленных в очередь виртуальных потоков. Компьютерный способ дополнительно содержит периодическое перемещение, по меньшей мере, одного виртуального потока из списка поставленных в очередь виртуальных потоков в список активных виртуальных потоков. Перемещение виртуального потока из списка поставленных в очередь виртуальных потоков в список активных виртуальных потоков в общем включает в себя (a) установку флага (семафора) для блокировки списка поставленных в очередь виртуальных потоков, (b) последующее изменение указателей в (i) перемещенном виртуальном потоке, (ii), по меньшей мере, одном виртуальном потоке первоначально в списке активных виртуальных потоков и (iii), по меньшей мере, одном виртуальном потоке, остающемся в списке поставленных в очередь виртуальных потоков, и (c) после этого возвращают в исходное положение или сбрасывают флаг для того, чтобы разрешить доступ к списку поставленных в очередь виртуальных потоков.

В соответствии с другим признаком настоящего изобретения каждый из виртуальных потоков включает в себя флаг (семафор), при этом компьютерный способ дополнительно содержит установку этого флага выбранного одного из виртуальных потоков, последующее изменение данных в выбранном виртуальном потоке и после этого возврат в исходное положение или сброс флага для того, чтобы разрешить доступ к выбранному виртуальному потоку. Установка флага выбранного потока, изменение данных и возврат в исходное положение или сброс флага можно выполнить в ответ на сообщение из другого одного из виртуальных потоков. Модификация данных обычно включает в себя изменение указателя выбранного виртуального потока.

В соответствии с другим признаком настоящего изобретения каждому из виртуальных потоков назначают очередь сообщений, при этом компьютерный способ дополнительно содержит ввод сообщения в очередь сообщений выбранного одного из виртуальных потоков во время выполнения задачи или задания в соответствии с еще одним из виртуальных потоков. Эти потоки могут соответствовать соответствующим задачам или заданиям, полученным из различных программ приложений, посредством чего ввод сообщения в очередь сообщений выбранного одного из виртуальных потоков осуществляет передачу данных между различными программами приложений. В другом приложении обмена сообщениями между потоками выбранный поток и другой поток являются посредническими или интерфейсными потоками на различных компьютерах. В этом случае ввод сообщения в очередь сообщений включает в себя передачу сообщения по линии связи между компьютерами. Линия связи может представлять собой частную компьютерную сеть или, например, глобальную компьютерную сеть, известную как Интернет.

Как предполагалось выше, создание виртуальных потоков, обработка задач или заданий в соответствующей последовательности квантов времени или интервалов времени обработки и совершение контекстных переключений включают в себя все работу компьютера под программой интерпретатора. Изобретение также предусматривает выполнение множества этапов программы интерпретатора на компьютере, при этом каждый этап соответствует собственному потоку. Каждый собственный поток создает соответствующий набор виртуальных потоков контекстных данных выполнения, обрабатывает каждую из множества задач или заданий в соответствующей последовательности квантов времени или интервалов времени обработки под управлением соответствующего виртуального потока и в каждом контекстном переключении между различными виртуальными потоками совершает такое контекстное переключение только после полного выполнения выполняющейся одной из псевдокодовых команд.

Выполнение многочисленных собственных потоков предпочтительно ограничено небольшим числом потоков на одиночном процессоре, например один или два потока. Там, где процессор обладает способностью внутренней организации многопоточной обработки, это ограничение освобождает другие потоки на основе собственной платформы для обработки других прикладных программ.

Так как виртуальная поточная обработка допускает неограниченное число создаваемых потоков и так как потоки имеют очень маленькие накладные расходы, программа, написанная на языке, который использует виртуальную организацию поточной обработки, может иметь преимущество в уникальном подходе программирования.

Этот подход программирования включает в себя использование большого числа потоков - один для каждого устройства пользовательского интерфейса на экране. Кнопка, например, имеет свой собственный поток. Полоса прокрутки имеет четыре потока - по одному для каждой кнопки, один для центральной полосы и один для главного потока. Потоки также не ограничены устройствами пользовательского интерфейса; например, программа сервера может создавать один поток для поддержки каждого клиентского запроса.

Типичное приложение может иметь в пределах от сотни потоков до нескольких тысяч потоков в зависимости от характера приложения. При выполнении в системе многочисленных заявок это позволяет быстро превысить возможности собственной организации поточной обработки Windows 95 (150-200 потоков), и это будет отрицательно влиять на производительность даже на платформах с большими или неограниченными возможностями организации поточной обработки. С другой стороны, виртуальная поточная обработка специально разработана для того, чтобы иметь дело с этими проблемами, делая возможным выполнение многочисленных приложений с десятками тысяч потоков без каких-либо проблем с производительностью. Такое широкое использование потоков значительно упрощает создание сложного приложения, потому что код интерфейса пользователя не должен хранить путь от сотен устройств интерфейса пользователя - каждый поток выполняет простую программу, которая сохраняет путь одиночного устройства интерфейса пользователя, за который этот поток отвечает.

Это приводит в результате к меньшим программам, которые проще создать, легче отлаживать и поддерживать.

В соответствии с другим признаком настоящего изобретения там, где выбранный один из виртуальных потоков находится в неактивном состоянии (например, в связанном списке неактивных потоков), компьютерный способ дополнительно содержит этапы, в соответствии с которыми формируют сообщение в ответ на ввод из источника, расположенного вне компьютера, производят вставку сообщения в очередь сообщений для выбранного виртуального потока, изменяют состояние выбранного потока из неактивного состояния в активное состояние, после этого осуществляют доступ к очереди сообщений для того, чтобы получить сообщение во время кванта времени или интервала времени обработки, назначенного для выбранного потока. Этот процесс является процессом, который используется для сдвига виртуального потока из неработающего или неактивного

состояния в активное состояние в соответствии с появлением события, подходящего для соответствующего потока. Это событие можно сформировать с помощью источника, расположенного вне компьютера, например с помощью оператора, нажимающего клавишу клавиатуры, или с помощью установления связи из удаленного компьютера.

Опосредованную интерпретатором виртуальную поточную обработку, согласно настоящему изобретению, можно располагать по приоритетам среди различных задач или заданий с помощью любого подходящего метода. Там, где каждый из виртуальных потоков включает в себя приоритет потока, компьютерный способ дополнительно содержит автоматическое обращение к приоритетам потоков в множестве виртуальных потоков для того, чтобы определить относительные свойства и изменение последовательности потоков в соответствии с определенными относительными приоритетами. В одной методике расположения по приоритетам данный поток, имеющий приоритет, который является общим числом, большим, чем приоритет второго потока, соответствует числу квантов времени или интервалов времени обработки, которое представляет собой общее число, большее, чем число квантов времени или интервалов времени обработки, соответствующих второму потоку.

Расположение по приоритетам виртуальных потоков (и, следовательно, их соответствующих задач) обеспечивает возможность также распределения загрузки обработки среди различных собственных потоков, где используется более одного собственного потока. Поток позволяет выделить задачу перераспределения виртуальных потоков из собственных потоков, имеющих больший, чем средний, приоритет задач для собственных потоков, имеющих приоритет, меньший, чем средний приоритет потоков. В целом, сдвиг потоков ограничен активными потоками.

Как описано ниже, задачи или задания, обработанные в соответствующей последовательности квантов времени или интервалов времени обработки под управлением соответствующих виртуальных потоков, включают в себя объекты управления, изображаемые на устройстве отображения компьютера, причем каждый из объектов составляет отдельную задачу или задание, назначенное соответствующему одному из виртуальных потоков. Обработанные задачи или задания, назначенные соответствующим виртуальным потокам интерпретатором, согласно настоящему изобретению дополнительно включают в себя контроль нажатия клавиш на клавиатуре компьютера. Каждая из клавиш образует отдельную задачу или задание, назначенное соответствующему одному из виртуальных потоков.

Временные интервалы или интервалы времени обработки измеряют предпочтительно путем подсчета последовательно выполненных псевдокодовых команд. Компьютерный способ дополнительно содержит для каждого из множества квантов времени или интервалов времени обработки завершение (прерывание) соответствующего интервала времени или интервал времени обработки после подсчета предопределенного числа последовательно выполненных псевдокодовых команд.

Многозадачный компьютер содержит в соответствии с конкретным вариантом осуществления настоящего изобретения память, устройство отображения, внешнее устройство ввода и, по меньшей мере, один процессор, подсоединенный в рабочем состоянии к памяти, устройству отображения и внешнему устройству ввода, при этом процессор имеет компилятор для преобразования команд исходной программы, введенной оператором, в байт-кодовые или псевдокодовые команды, причем компилятор связан в рабочем состоянии (оперативно) с памятью для предоставления возможности хранения в ней байт-кодовых или псевдокодовых команд. Процессор также имеет интерпретатор для выполнения байт-кодовых или псевдокодовых команд. Память хранит первый связанный список неактивных виртуальных потоков, второй связанный список активных виртуальных потоков и третий связанный список поставленных в очередь, или ожидающих, виртуальных потоков. Каждый из потоков включает в себя данные состояния или контекста, флаг (семафор) и указатель следующего потока в соответствующем списке. Интерпретатор подсоединен в рабочем состоянии (оперативно) к внешнему устройству ввода для

распознавания события, выработанного внешним устройством ввода, и подсоединен в рабочем состоянии (оперативно) к памяти (a) для перемещения, по меньшей мере, одного из неактивных виртуальных потоков из первого связанного списка в третий связанный список, (b) для перемещения поставленных в очередь или ожидающих виртуальных потоков из третьего связанного списка во второй связанный список, (c) для выполнения команд согласно данным состояния и контекста различных виртуальных потоков во втором связанном списке в последовательных квантах времени или интервалах времени обработки в соответствии с предопределенным планировщиком заданий по приоритетам. Интерпретатор подсоединен в рабочем состоянии (оперативно) к устройству отображения частично для изменения объекта на устройстве отображения в ответ на команды, определенные с помощью соответствующего активного виртуального потока во втором связанном списке.

Память может дополнительно хранить четвертый связанный список собственных потоков. В том случае, когда интерпретатор представляет собой один из множества вариантов (экземпляров) общего интерпретатора, при этом каждый из вариантов общего интерпретатора соответствует соответствующему одному из собственных потоков. Кроме того, второй связанный список является одним из множества связанных списков активного потока, причем каждый из собственных потоков связан с помощью соответствующего указателя с соответствующим одним из связанных списков активного потока, хотя третий связанный список является одним из множества связанных списков поставленных в очередь потоков, при этом каждый из собственных потоков связан с помощью соответствующего указателя с соответствующим одним из связанных списков поставленного в очередь потока.

В соответствии с другой конкретной особенностью настоящего изобретения интерпретатор включает в себя запрограммированную схему для перемещения виртуального потока из первого собственного потока, имеющего загрузку тяжелее обычной, во второй собственный поток, имеющий загрузку легче обычной.

Список или таблица неактивных виртуальных потоков предпочтительно включают в себя множество потоков, назначенных соответствующим клавишам клавиатуры для обработки срабатываний соответствующих клавиш. Список или таблица неактивных потоков может дополнительно включать в себя множество потоков, назначенных соответствующим объектам в отображаемом изображении для обработки изменений вида соответствующих объектов.

Там, где интерпретатор включает в себя модуль контекстного переключения и счетчик команд, модуль контекстного переключения подсоединен в рабочем состоянии (оперативно) к памяти и счетчику команд для контекстного переключения из выполняемого активного потока второго связанного списка в следующий активный поток во втором связанном списке после выполнения предопределенного числа байт-кода или псевдокодовых команд согласно выполняемому активному потоку.

Каждый из виртуальных потоков включает в себя местоположение в памяти следующей команды для выполнения в соответствующем потоке, значения любых локальных переменных для соответствующего потока и приоритет выполнения для соответствующего потока.

В соответствии с другими особенностями настоящего изобретения память сохраняет множество очередей сообщений, назначенных соответствующим потокам, а также сохраняет, по меньшей мере, один промежуточный или интерфейсный поток, имеющий контекст выполнения для поддержания связи с удаленным компьютером через линию связи. Там, где линия связи представляет собой компьютерную сеть, такую как Интернет, промежуточный или интерфейсный поток содержит адрес памяти, указывающий программу сетевого протокола.

Многозадачный компьютер содержит в соответствии с другим вариантом осуществления настоящего изобретения память, сохраняющую данные состояния и контекста многочисленных потоков или задач, и интерпретатор для выполнения

последовательностей байт-кодовых команд, каждая из которых состоит из многочисленных шагов набора команд, причем интерпретатор запрограммирован так, чтобы определять соответствующий виртуальный поток для каждой задачи, которая будет выполняться с помощью компьютера, выполнять байт-кодовые команды соответствующего текущего потока, выбранного из числа виртуальных потоков во время каждого кванта времени из ряда последовательных квантов времени, и выполнять контекстное переключение из одного из упомянутых виртуальных потоков в другой из виртуальных потоков только после выполнения одной из байт-кодовых команд.

Различные преимущества настоящего изобретения будут очевидны из их описания.

КРАТКОЕ ОПИСАНИЕ ЧЕРТЕЖЕЙ

Фиг.1 изображает блок-схему компьютерной системы, обладающей способностью виртуальной организации поточной обработки, согласно настоящему изобретению.

Фиг.2 изображает блок-схему выбранных компонентов процессора (фиг.1), на которой показаны соединения этих компонентов с другими элементами системы (фиг.1).

Фиг.3 изображает блок-схему выбранных компонентов интерпретатора, показанного на фиг.2.

Фиг.4 изображает схему данных состояния потока, сохраненных в памяти, показывающую структуру связанного списка данных.

Фиг.5A и 5B изображают этапы алгоритма, показывающие выбранные операции, которые выполняет интерпретатор (фиг.2).

Фиг.6 изображает блок-схему, показывающую связь между двумя компьютерами, использующими виртуальную поточную обработку настоящего изобретения.

ОПРЕДЕЛЕНИЯ

Термин "многозадачность", используемый здесь, относится к одновременному выполнению многочисленных задач с помощью компьютера.

Термин "псевдокод", используемый здесь, относится к компьютерным командам, скомпилированным для выполнения с помощью интерпретатора. Интерпретатор представляет собой программу, которая служит для трансляции в программы в псевдокодах машинного языка и выполнения указанных операций, когда они транслируются. "Псевдокод" не относится к аппаратным средствам конкретного компьютера и требует преобразования в код, используемый компьютером перед тем, как можно будет использовать программу. Многочисленные псевдокодовые команды влекут за собой выполнение многочисленных команд на машинном языке. Псевдокод иногда называется "байт-кодом".

Термин "задача" или "задание" используется здесь для обозначения любой функции, выполняемой с помощью компьютера. Задачи или задания могут изменяться по своему масштабу от простой операции, такой как изменение содержимого регистра процессора, до больших и сложных операций, требующих выполнения многочисленных псевдокодовых команд. Примеры задач или заданий включают в себя (a) контроль внешних устройств пользовательского ввода, таких как клавиатуры и отдельные ее клавиши, (b) создание и изменение объектов на мониторе или устройстве отображения, таких как кнопки меню, окна, полосы прокрутки, пиктограммы и фоновые изображения, (c) связь с удаленными компьютерами по сети или другой линии связи, (d) прикладные программы, такие как текстовый процессор, электронные таблицы, мультимедийный проигрыватель, калькулятор и так далее, и (e) различные компоненты или функции прикладных программ, такие как редактирование, печать, проверка орфографии и другие функции текстового процессора.

Термин "квант времени" или "интервал времени обработки" используется здесь для обозначения отрезка времени работы процессора. В известных многозадачных компьютерах все кванты времени равны по длительности, которая измеряется с помощью масштаба времени или прерывания таймера. В соответствии с настоящим раскрытием изобретения кванты времени или интервалы времени обработки измеряются с помощью таймера, как в известных многозадачных компьютерах, или подсчета команд. В последнем альтернативном случае все кванты времени или интервалы времени обработки

необязательно равны по длительности.

Используемый здесь термин "поток" относится к контексту выполнения для реализации или осуществления задания или задачи с помощью компьютера, при этом контекст выполнения используется или сопровождается в последовательности квантов времени или интервалов времени обработки. Термин "виртуальный поток", используемый здесь, относится к потоку, который создается, сохраняется, изменяется, обрабатывается и сопровождается (отслеживается) интерпретатором. Термин "собственный поток" используется здесь для обозначения потока, встроенного в операционную систему конкретного компьютера. Там, где операционная система компьютера имеет многочисленные возможности собственной организации поточной обработки, может быть использовано множество собственных потоков, при этом каждый такой собственный поток выполняет соответствующий вариант интерпретатора.

Термин "флаг" (семафор), который используется здесь, относится к объекту с возможностью блокировки, который можно установить или блокировать с помощью одного потока в данный момент времени для того, чтобы предотвратить доступ другим потокам к программе, виртуальному потоку, области памяти или другому компоненту компьютерной системы. Флаг используется с помощью реализаций собственной организации многопоточной обработки для синхронизации доступа к данным, которые совместно используются между потоками. Использование флагов важно потому, что такое использование предотвращает конфликты между многочисленными потоками, пытающимися изменить одни и те же данные в одно и то же время. Флаг используется для представления совместно используемых данных, при этом поток должен блокировать флаг перед попыткой доступа к данным.

"Связанный список" представляет собой структуру, используемую обычно в индустрии программных средств, где каждая запись в списке содержит адрес памяти следующей записи в списке. Это соединение позволяет вставлять или удалять записи из списка без перемещения других записей в списке. Удаление элемента просто приводит к изменению предшествующего элемента так, чтобы предшествующий элемент указывал адрес последующего элемента, таким образом освобождая память, используемую элементом.

Термин "контекстное переключение" используется здесь для обозначения процесса, в котором поток, выполняемый в текущий момент времени, прерывается, при этом состояние потока или контекст выполнения сохраняется, новый поток выбирается для выполнения, сохраненный контекст выполнения нового потока восстанавливается и следует непосредственно за следующими компьютерными операциями.

ОПИСАНИЕ ПРЕДПОЧТИТЕЛЬНЫХ ВАРИАНТОВ ОСУЩЕСТВЛЕНИЯ

Как показано на фиг.1, компьютерная система включает в себя процессор 12, клавиатуру 14, устройство 16 отображения и память 18. Процессор 12 подсоединен к удаленному компьютеру 20 через компьютерную сеть, такую как Интернет 22. Как показано на фиг.2, процессор 12 включает в себя интерпретатор 24, выполненный обычно в виде цифровых схем, характерных для компьютера и изменяемых посредством программирования для обеспечения многочисленных функций компьютера, которые включают в себя интерпретацию действий на клавиатуре и управление устройством 16 отображения и, в частности, внешний вид объектов на нем в ответ на ввод команд пользователем через клавиатуру 14 или в ответ на сообщения, получаемые из компьютера 20 по Интернету 22. Процессор 12 также включает в себя компилятор 26 (который может быть частью интерпретатора 24) для преобразования исходного текста, созданного человеком, в байт-код или псевдокод, который запоминается в памяти 18.

Как показано на фиг.3, интерпретатор 24 включает в себя блок 28 выполнения кода, подсоединенный в рабочем состоянии к памяти 18 для считывания байт-кода и выполнения операций в соответствии с байт-кодом. Интерпретатор 24 дополнительно включает в себя счетчик 30 команд, подсоединенный к блоку 28 выполнения кода для отслеживания числа байт-кодовых команд, обработанных за текущий квант времени или интервал времени обработки. Счетчик 30 подсоединен в рабочем состоянии к модулю 32 контекстного

переключения, подсоединенного, в свою очередь, к блоку 28 выполнения для введения изменения в контекст выполнения интерпретатора 24 после подсчета предопределенного числа байт-кодовых команд с помощью блока 28. Переключение выполнения может происходить раньше, то есть перед завершением подсчета, при определенных обстоятельствах, например, если введено состояние ожидания.

В большей части следующего обсуждения, предполагается, что интерпретатор 24 имеет дело только с компилируемым байт-кодом. В действительности, компилятор 26 (который может быть элементом интерпретатора или отдельной программой) должен транслировать исходный код, который может считать человек, в байт-код. Компиляторы являются стандартными в индустрии программных средств (C, C++, Бейсик (Basic), Паскаль (Pascal), Java и многие другие языки необходимо компилировать перед их выполнением), и методы написания компиляторов являются общеизвестными для многих программистов. Соответственно, дальнейшее обсуждение компилятора 26 будет опущено из настоящего раскрытия.

Интерпретатор 24 осуществляет многозадачность через создание и связывание виртуальных потоков и выполнение задач в соответствующей последовательности квантов времени или интервалов времени обработки в соответствии с или под управлением соответствующих виртуальных потоков. В дальнейшем предполагается, что интерпретатор 24 является интерпретатором, работающим на основе стека. На практике, виртуальная поточная обработка должна работать с любым типом интерпретатора, работающего со стеком или нет.

Рассмотрим следующий байт-код, который рисует прямоугольник с координатами x=10, y=10 и x=20, y=20 на экране устройства 16 отображения компьютера:

	ПАРАМЕТР КОМАНДЫ
ПОМЕСТИТЬ В СТЕК ЦЕЛОЕ ЧИСЛО	10
ПОМЕСТИТЬ В СТЕК ЦЕЛОЕ ЧИСЛО	10
ПОМЕСТИТЬ В СТЕК ЦЕЛОЕ ЧИСЛО	20
ПОМЕСТИТЬ В СТЕК ЦЕЛОЕ ЧИСЛО	20
РИСОВАТЬ ПРЯМОУГОЛЬНИК	

На языке, основанном на работе со стеком, первые четыре команды помещают значения 10, 10, 20 и 20 в стек. Команда РИСОВАТЬ БЛОК удаляет верхние четыре значения из стека и использует их как координаты для того, чтобы рисовать прямоугольник.

Эта команда байт-кода будет использоваться в качестве образцовой программы в следующих обсуждениях интерпретатора 24 и виртуальной поточной обработки.

Виртуальные потоки

Виртуальный поток является в основном контекстом выполнения и ничем больше. Контекст выполнения состоит из (a) местоположения в памяти следующей команды для выполнения в потоке, (b) значений любых локальных переменных для потока, (c) стека вызовов для потока, (d) приоритета потока и других признаков и (e) любых других данных, которые язык программирования должен хранить для каждого потока, таких как данные о состоянии ошибок. Байт-код не является частью потока, при этом несколько потоков могут одновременно выполнять один и тот же байт-код. Поток просто поддерживает указатель следующей команды байт-кода в памяти, который он будет выполнять.

Поток может находиться в одном из четырех состояний:

(1) Неактивное: Неактивные виртуальные потоки являются потоками, которые временно не выполняют никакой работы. Например, поток, который ожидает ввода пользователя, такого как нажатие клавиши, является неактивным, как и поток, который ожидает окончания работы таймера. Неактивные потоки хранятся в списке отдельно от активных потоков и не занимают какого-либо времени процессора.

(2) Ожидание в очереди: Виртуальный поток готов стать активным и назначен одному из собственных потоков, который запускает интерпретатор, но собственный поток занят выполнением команды и не может переместить виртуальный поток в его активный список

до тех пор, пока не закончится команда.

(3) Активный: Виртуальный поток находится в активном списке собственного потока и является одним из виртуальных потоков, который принимает кванты времени из собственного потока.

(4) Текущее: Текущий виртуальный поток является всегда также активным виртуальным потоком. Текущий виртуальный поток является виртуальным потоком, который выполняется в текущий момент времени, то есть собственный поток предоставляет ему квант времени, и он "находится" в пределах этого кванта времени.

Связанные списки потока

Как показано на фиг.4, интерпретатор 24 запоминает состояние потока и контекстные данные в памяти 18 в виде набора связанных списков. Существует два основных связанных списка: таблица 34 неактивных потоков и таблица 36 собственных потоков. Указатели 38 и 40 в первой записи каждого из этих списков 34 и 36 запоминаются в виде глобальных переменных в интерпретаторе 24.

Таблица 34 неактивных потоков хранит список всех неактивных виртуальных потоков 42 в системе. Эти потоки 42 остаются в неактивной таблице 34 до тех пор, пока не произойдут соответствующие события, которые вызывают повторную активизацию потоков. Каждая запись или виртуальный поток 42 в неактивной таблице 34 содержит состояние потока и контекстные данные 44, флаг 46, используемый для управления доступом к данным состояния и контекста потока, и указатель 48, содержащий адрес памяти следующей записи в списке.

Таблица 36 собственных потоков содержит запись для каждого собственного потока 50, который выполняет вариант интерпретатора 24 и который может принимать виртуальные потоки для выполнения. На платформах, таких как некоторые версии MacOS, где отсутствует способность собственной организации поточной обработки, существует только одна запись в таблице 36 собственных потоков. Каждая запись 50 в таблице 36 собственных потоков содержит один флаг 52, указатель 54 на связанный список 56 активных виртуальных потоков 58, другой указатель 60 на связанный список 62 поставленных в очередь или ожидающих виртуальных потоков 64 и другой указатель 66 для следующей записи в списке 36 собственных потоков. Связанные списки 56 и 62 виртуальных потоков 58 и 64 используют тот же самый формат, как и связанный список 34 неактивных виртуальных потоков 42, хотя отдельные активные потоки 58 и отдельные поставленные в очередь потоки 64 имеют одну и ту же структуру, как и неактивные потоки 42. Каждый собственный поток 50 периодически перемещает потоки из соответствующего связанного списка 62 поставленных в очередь виртуальных потоков 64 в соответствующий связанный список 56 активных виртуальных потоков 58. Собственный поток 50 не использует свои поставленные в очередь потоки 64, хотя собственный поток выполняет команды (через блок 28 выполнения), поэтому соответствующий связанный список 62 может быть блокирован, и потоки могут быть помещены в очередь без блокировки активного списка 56 собственных потоков, таким образом улучшая производительность мультипроцессорных систем.

Флаг (семафор) 52 используется для синхронизации доступа к соответствующему связанному списку 62 поставленных в очередь виртуальных потоков 64. Собственный поток 50 (владелец или другой собственный поток) должен иметь блокировку на соответствующем флаге 52 перед тем, как может быть получен доступ к соответствующей очереди 62. Остальная часть структуры потока (фиг.4) не требует флага, потому что единственный поток с возможностью доступа является соответствующим собственным потоком.

Интерпретатор 24 расходует большую часть своего времени (как это делает любой интерпретатор) в цикле, таком как показан на фиг.5А и 5В. Внутри этого цикла интерпретатор 24 выполняет ряд операций для того, чтобы поддержать виртуальную поточную обработку.

Задачи поддержки ОС

Одной из задач, которую должен выполнять периодически интерпретатор 24, является поддержка 70 операционной системы. Детали этой задачи изменяются в зависимости от платформы. Как правило, только один собственный поток 50 должен выполнять задачи 70 поддержки, хотя другие собственные потоки просто выполняют команду байт-кода. На некоторых платформах полностью отдельный собственный поток можно использовать для поддержки, и все собственные потоки 50 в таблице 36 собственных потоков могут быть выделены выполнению байт-кода. С другой стороны, платформы без возможности собственный организации поточной обработки (то есть системы кооперативной многозадачности, такие как версии MacOS до появления OS X) должны периодически выполнять задачи поддержки для того, чтобы разрешить другим задачам выполняться в системное время.

Типичные задачи, выполняемые как часть задач 70 поддержки операционной системы, включают в себя выполнение одной итерации событийного цикла операционной системы (такой как вызов ПолучитьСледующееСобытие() (GetNextEvent()) на платформах MacOS или СчитатьСообщение()/ПолучитьСобытие()/ПеревестиСообщение()/ ОтправитьСообщение() (PeekMessage()/GetMessageO/Translate Message()/DispatchMessage()) на платформах Windows).

Обработка события

После выполнения задач 70 поддержки интерпретатор 24 и, в частности, его блок 28 выполнения команд делает запрос 72 относительно того, находятся ли какие-либо виртуальные потоки в списке 56 активных потоков собственного потока 50. Там, где имеется, по меньшей мере, один активный поток 58 в списке 56 активных потоков, интерпретатор 24 предпринимает активную проверку 74 событий. Там, где активные потоки 58 отсутствуют в списке 56, интерпретатор 24 ожидает событие на этапе 76, таким образом предоставляя время процессора для других приложений в системе.

Когда событие имеет место, событие кодируется в виде сообщения и помещается в очередь сообщений для соответствующего виртуального потока. У каждого виртуального потока может быть предусмотрена своя собственная очередь сообщений. Если релевантный виртуальный поток является неактивным потоком 42, то его необходимо повторно активизировать, что включает в себя процедуру нахождения собственного потока 50, для которого неактивный поток должен быть назначен как поставленный в очередь или ожидающий поток 64.

Как показано на фиг.5А, если проверка 74 события приводит к обнаружению события, что устанавливается интерпретатором 24 на этапе 78 принятия решений, интерпретатор 24 идентифицирует принимаемый поток на этапе 80 и затем блокирует очередь сообщений принимаемого потока на этапе 82. Эта блокировка также предпринимается интерпретатором 24 после приема уведомления о событии на этапе 76. Далее интерпретатор 24 исследует на этапе 84, является ли принимаемый поток неактивным потоком. Если принимаемый поток является неактивным потоком 42, что определяет интерпретатор 24 на этапе 84 проверки, глобальный список 34 неактивных потоков блокируется (флаг не показан) на этапе 86. Принимаемый поток затем удаляется из связанного списка 34 неактивных потоков 42 на этапе 88. Это удаление обычно влечет за собой изменение указателя 48 виртуального потока, непосредственно предшествующего принимаемому потоку в списке 34 неактивных потоков с тем, чтобы этот указатель идентифицировал виртуальный поток непосредственно после принимаемого потока в списке 34 неактивных потоков.

После удаления принимаемого потока из связанного списка 34 неактивных потоков на этапе 88 интерпретатор 24 блокирует этот список 34 на этапе 90. На следующем этапе 92 интерпретатор 24 сканирует или просматривает связанный список 36 собственных потоков 50 для того, чтобы найти собственный поток с самой малой загрузкой. Загрузку собственного потока можно вычислить в виде количества виртуальных потоков, назначенных собственному потоку, хотя в общем случае лучше вычислить загрузку более точно путем суммирования приоритетов всех собственных потоков, назначенных

физическому потоку (то есть все собственные потоки в активном списке и очереди физического потока). Флаг 52 очереди собственного потока 50, выбранный как имеющий самую малую загрузку, блокируется с помощью интерпретатора 24 на этапе 94. Принимаемый виртуальный поток, только что удаленный из списка 34 неактивных потоков, затем добавляется в список 62 поставленных в очередь потоков выбранного собственного потока 50 на этапе 96, и соответствующий флаг 52 затем разблокируется на этапе 98. Добавление принимаемого виртуального потока в список 56 поставленных в очередь потоков влечет за собой модификацию двух указателей, в принимаемом потоке и в потоке, непосредственно предшествующем принимаемому потоку в списке 56 поставленных в очередь потоков, после вставки в него принимаемого потока.

Если принимаемый поток является ожидающим или неактивным потоком 42, что определяет интерпретатор 24 на этапе 84 проверки, интерпретатор приступает к программе 100 для переноса поставленного в очередь потока 64 из связанного списка 62 в связанный список 56 активных потоков. Программу 100 также выполняет интерпретатор 24 после переноса принимаемого потока из списка 34 неактивных потоков в список 56 поставленных в очередь потоков наименее занятого собственного потока 50. На первом этапе 102 программы 100 интерпретатор 24 блокирует флаг 52 очереди собственного потока. Интерпретатор 24 затем проверяет на этапе 104, содержит ли список 62 поставленных в очередь потоков, по меньшей мере, один виртуальный поток 64. Если содержит, то первый поток в очереди удаляется из очереди на этапе 106 и добавляется в соответствующий список 56 активных потоков на этапе 108. И снова это перемещение виртуального потока из одного списка в другой просто влечет за собой изменение трех указателей - одного указателя перемещенного потока и других указателей непосредственно предшествующих потоков в двух списках. После перемещения потока интерпретатор 24 решает на этапе 110 принятия решения, находится ли вновь перемещенный виртуальный поток в группе с более высоким приоритетом, чем выполняемый поток. Если это так, то контекстное переключение выполняется интерпретатором 24 и, в частности, модулем 32 (фиг.3) на этапе 112 с тем, чтобы вновь перемещенный поток стал потоком, выполняемым в текущий момент времени. Флаг 52 очереди затем разблокируется на этапе 114.

Обработка таймеров

Обработка таймеров не входит в алгоритм, иллюстрируемый на фиг.5А и 5В, так как обработка таймера не является критической частью виртуальной поточной обработки. Однако интерпретируемый язык должен, как правило, обеспечивать разработчиков средствами установки таймеров.

Наиболее эффективный способ реализации таймеров с виртуальной организацией поточной обработки состоит в том, чтобы сохранить в памяти 18 глобальный список всех таймеров. Список необходимо сортировать так, чтобы таймеры, действие которых закончится быстрее всего, появились в начале списка.

Каждый раз, когда главный цикл интерпретатора (фиг.5А и 5В) производит итерацию, обычно во время обработки событий, интерпретатор 24 должен проверить первую запись в списке таймеров для того, чтобы было видно, закончилась ли эта запись (если нет, то действие другого таймера не закончилось, потому что таймеры, задействованные позже, в списке заканчивают свое действие после первого таймера в списке). Если время работы таймера истекло, то его необходимо удалить из списка, и должно быть выработано событие. Событие вызывает активизацию соответствующего виртуального потока (если он уже является неактивным), и этот поток будет искать событие окончания времени действия таймера, когда поток проверяет свою очередь.

Если отсутствуют виртуальные потоки, назначенные собственному потоку, и, следовательно, собственный поток ожидает события (этап 76) вместо проверки наличия событий (этап 74), то интерпретатор 24 должен проверять таймеры перед ожиданием событий. Если в списке имеются какие-либо таймеры, то интерпретатор должен установить таймер операционной системы так, чтобы операционная система выработала событие для

снятия ожидания после истечения времени действия таймера.

Важно обратить также внимание на то, что различные операционные системы имеют различные степени точности в своих таймерах. Если оставшееся время на первом таймере в списке меньше, чем точность таймера операционной системы, интерпретатор 24 не может ждать сообщения, но должен вместо этого проверить сообщения с тем, чтобы событие истечения времени было выработано с необходимой точностью.

Сообщения между потоками

Как правило, виртуальные потоки имеют потребность в некоторых средствах связи для того, чтобы они могли осуществлять обмен данных друг с другом.

Собственные потоки в известных многозадачных компьютерных системах обычно используют совместно данные путем помещения данных в память с тем, чтобы данные были доступны для всех потоков, и последующим блокированием доступа к памяти для предотвращения одновременного доступа многочисленных потоков к данным. Виртуальная организация поточной обработки, описанная здесь, использует отличный подход, который представляет собой обмен сообщений между потоками. Сообщение состоит из идентификатора сообщения, который однозначно идентифицирует тип сообщения (например, строка "перемещение-мыши" может быть идентификатором сообщения), и блока данных. Блок данных может иметь любой формат и может иметь любой размер (хотя огромные блоки с размером в несколько мегабайт не рекомендуются по причинам производительности; в таких случаях предпочтительны серии меньших по размеру сообщений).

Сообщение может быть выработано в ответ на внешнее событие, принятое из операционной системы (включая события пользовательского ввода, например перемещение мыши, нажатия клавиш и так далее), или в ответ на команду в байт-коде виртуального потока.

Сегмент обработки событий алгоритма на фиг.5А показывает то, как сообщение добавляется в очередь сообщений потоков в ответ на событие. Та же самая методика используется для добавления сообщения в очередь сообщений потока в ответ на команду байт-кода, но с одним исключением: в случае, когда поток помещает сообщение в свою собственную очередь сообщений, необходимо действовать осторожно во избежание блокировки частей контекста потока, которые уже были блокированы в ходе выполнения команды байт-кода, при этом попытка блокировки в обоих местах может привести к тупиковым ситуациям в зависимости от реализации флагов на платформе.

Перемещение потоков из очереди активности в активный список

Когда виртуальный поток активизируется в ответ на сообщение или событие, виртуальный поток помещается в список 62 поставленных в очередь потоков соответствующего собственного потока 50, а не непосредственно в список 56 активных потоков этого собственного потока. Это сделано потому, что к активному списку 56 собственных потоков необходимо иметь доступ непосредственно только с помощью собственного потока с тем, чтобы список 56 не был заблокирован. Уход от необходимости блокировки активного списка 56 собственных потоков и блокировка только списка 62 поставленных в очередь потоков улучшает производительность, так как собственный поток 50 может быть занят выполнением команды, и собственный поток, который выполняет активизацию (который может представлять собой отличный поток, выполняющийся асинхронно на другом процессоре), не должен ждать выполнения команды для завершения.

Поэтому собственный поток отвечает за периодически перемещающиеся потоки из своей очереди активности в свой активный список.

Контекстное переключение

Каждый собственный поток 50 сохраняет указатель (например, указатель 54) в виртуальном потоке, выполняющемся в текущий момент времени, в записи собственного потока в списке 36 собственных потоков. Контекстное переключение заключается просто в изменении этого указателя для того, чтобы указать на другой виртуальный поток 58 в

списке 56 активных потоков. Код выполнения команды, который выполняется с помощью блока 28 выполнения команд (фиг.3), использует контекст потока в записи списка виртуальных потоков в адресе этого указателя, поэтому другое действие необязательно для контекстного переключения. Это означает, что контекстное переключение действует очень быстро, а выполнение команд имеет тенденцию к небольшому замедлению, чем обычно, из-за признака косвенности указателя.

Приоритеты потоков

Каждый виртуальный поток 42, 58, 64 имеет назначенный приоритет. Если многочисленные потоки активны в одно и то же время, то есть если список 56 активных потоков содержит больше одного потока 58, то поток с более высоким приоритетом будет получать больше времени процессора.

Одно из ключевых применений приоритетов потока заключается в том, чтобы предоставить приоритет ("старшинство") потокам, которые отвечают за пользовательский ввод. Этот приоритет позволяет интерпретатору 24, например, модифицировать объект на устройстве 16 отображения непосредственно после нажатия оператором клавиши на клавиатуре 14. Таким образом, оператор получает непосредственную обратную связь из компьютера, показывающего, что его или ее команда была получена и обрабатывается. Таким образом, пользователь знает, что клавиатура 14 и процессор 11 работают и не были приостановлены или, по-другому, заблокированы.

Для примера рассмотрим приложение с кнопкой "Печать", показанной в виде объекта на устройстве 16 отображения. Кнопка, которая представляет собой устройство пользовательского ввода, назначается своему собственному виртуальному потоку 42, 56, 64. Поток кнопки расходует большую часть своего времени в списке 34 неактивных потоков. Когда пользователь щелкает на кнопке (щелчок по кнопке мыши представляет собой событие пользовательского ввода), поток активизируется. Поток должен затем обновить изображение кнопки так, чтобы она выглядела "нажатой", после чего поток посылает сообщение в некоторый другой поток для уведомления этого другого потока о том, что кнопка была нажата, при этом другой поток будет затем делать то, что необходимо, например печатать документ.

При наличии потока кнопки с назначенным более высоким приоритетом, чем приоритеты других задач приложения, пользователь может быть уверен, что когда он или она щелкает на кнопке, то будет немедленный визуальный ответ (кнопка обновляет изображение, чтобы выглядеть "нажатой"), даже если система будет занята обработкой других задач.

Простая система приоритетов, воплощенная в алгоритме на фиг.5А и 5В, работает посредством назначения каждому потоку численного значения для его приоритета. Как правило, это значение находится в пределах от 0 до 100, с большими значениями, показывающими более высокие приоритеты, но использован может быть любой диапазон значений. В этой простой системе приоритетов данный активный поток 58 не будет занимать какое-либо время процессора, если имеются какие-либо потоки с более высокими приоритетами в соответствующем списке 56 активных потоков. Каждый собственный поток 50 в этой системе сохраняет "след" от уровня приоритета виртуального потока 58 с самым высоким приоритетом в своем активном списке 56 (этот уровень приоритета можно называть самым высоким активным приоритетом). Когда собственный поток 50 выполняет контекстное переключение и должен произвести выбор нового виртуального потока 56, чтобы стать текущим виртуальным потоком, собственный поток 50 всегда выбирает следующий виртуальный поток в списке 56, который имеет самый высокий активный приоритет и запускается с начала списка, когда собственный поток достигает конца списка. Каждый собственный поток 50 также сохраняет "след" количества активных виртуальных потоков 58 с самым высоким активным приоритетом.

В этой простой системе приоритетов всякий раз, когда собственный поток 50 перемещает виртуальный поток из соответствующего списка 62 поставленных в очередь потоков в соответствующий список 56 активных потоков, если этот виртуальный поток имеет более высокий приоритет, чем самый высокий активный приоритет, этот самый

высокий активный приоритет становится равным приоритету нового потока, при этом подсчет потоков с самым высоким активным приоритетом установлен равным 1, и контекстное переключение выполняется так, чтобы сделать новый поток текущим виртуальным потоком. Если новый виртуальный поток имеет приоритет, равный самому высокому активному приоритету, подсчет потоков с самым высоким активным приоритетом просто увеличивается.

Всякий раз, когда активный виртуальный поток 58 завершается или становится неактивным, соответствующий собственный поток 50 уменьшает подсчет потоков с наивысшим уровнем активного приоритета. Если подсчет достигает нуля, собственный поток 50 просматривает свой активный список 56 для того, чтобы определить новый самый высокий активный приоритет и новый подсчет потоков для этого приоритета, и выполняет контекстное переключение для того, чтобы сделать соответствующий поток новым текущим виртуальным потоком.

И, наконец, если уровень приоритета активного виртуального потока 58 увеличивается и новый приоритет выше, чем текущий самый высокий активный приоритет, самый высокий приоритет должен быть настроен, и этот поток должен стать новым текущим потоком, и аналогично, если уровень приоритета активного виртуального потока 58 уменьшается, если этот виртуальный поток имел раньше самый высокий уровень приоритета, соответствующий собственный поток 50 должен просматривать свой активный список 56 для того, чтобы определить новый самый высокий активный приоритет и новый подсчет потоков при этом приоритете, и затем выполняет контекстное переключение для того, чтобы сделать соответствующий поток новым текущим виртуальным потоком. Уровни приоритетов можно изменять в результате команд, которые выполняются с помощью интерпретатора 24.

Усовершенствованные приоритеты потоков

Более усовершенствованную систему можно использовать для приоритетов потоков, но совершенно необязательно для работы при виртуальной организации поточной обработки. Эта более усовершенствованная система позволяет вычислить время процессора для потока 58, даже если поток с более высоким приоритетом является активным.

Это выполняют с использованием групп приоритетов.

В типичной реализации приоритетом потока может быть значение между -9999 и +9999 включительно. Группа приоритетов потока равна приоритету потока, разделенному на 100 без дробных частей, например:

Группа приоритетов	Наименьший приоритет в группе	Наивысший приоритет в группе
-3	-399	-300
-2	-299	-200
-1	-199	-100
0	-99	+99
1	100	199
2	200	299
3	300	399

Правила для простой системы приоритетов потоков, описанной в предыдущем разделе, все еще используются, но эти правила применяются вместо групп приоритетов. Таким образом, данный активный поток 58 не будет получать какое-либо время процессора, если имеется активный поток в группе с более высоким приоритетом. Однако потоки 58 внутри одной и той же группы приоритетов будут получать время процессора на основании своих приоритетов внутри группы. Приоритеты потоков являются относительными. При заданных двух приоритетах P и Q поток приоритета P получит один квант времени для каждого кванта времени (Q-P)+1, которые принимает поток с приоритетом Q. Поэтому поток с приоритетом N получит один квант времени из каждых 4 квантов времени, которые получает поток с приоритетом N+3.

Например, рассмотрим следующий набор потоков:

ИД (ID) потока	Приоритет потока
A	50

B	120
C	121
D	122
E	124

Если все эти потоки являются одновременно активными, то будет выполняться поток A, так как он находится в группе с более низким приоритетом по сравнению с другими потоками. Оставшимся потокам будут выделяться кванты времени следующим образом:

BCD EEE D EEE C D EEE D EEE B C D EEE D EEE C D EEE D EEE

Другими словами, из каждых 38 квантов времени поток B будет получать 2 кванта времени поток C - 4, поток D - 8 и поток E - 24.

Это выделение квантов времени реализовано за счет поддержки счетчика 116 пропусков для каждого активного виртуального потока 58 (см. фиг.3). Каждый счетчик 116 имеет начальное значение, равное нулю. Всякий раз, когда происходит контекст переключения, и новый поток 58 должен быть выбран для выполнения, выбранный поток будет, естественно, находиться в группе с самым высоким приоритетом. Однако выбранный поток не может быть потоком с самым высоким приоритетом, который является к тому же активным, при этом другой поток с самым высоким приоритетом может находиться в той же самой группе приоритетов. Поэтому, если H - приоритет активного потока 58 с самым высоким приоритетом, и P - приоритет активного потока, который был выбран в качестве нового текущего виртуального потока для операции контекстного переключения, и S - значение счетчика пропусков (установленного первоначально на ноль) для нового текущего виртуального потока, и если $S \geq -P$, то контекстное переключение будет происходить нормально. В противном случае значение соответствующего счетчика 116 пропусков увеличивается, поток пропускается, и другой активный поток 58 выбирается как текущий поток.

Эта процедура изображена на фиг.5A. Интерпретатор 24 производит сначала проверку 118 относительно того, имеются ли в соответствующем списке 56 активных виртуальных потоков 58 какие-либо потоки из одной и той же группы приоритетов, как и текущий поток. Положительный результат приводит к тому, что интерпретатор 24 должен выбрать следующий поток в текущей группе приоритетов на этапе 120. Как описано выше, затем на этапе 122 интерпретатор 24 определяет приоритет H активного виртуального потока с самым высоким приоритетом в текущей группе приоритетов, приоритет P выбранного потока и отсчет S пропуска выбранного потока. На следующем этапе 124 принятия решения интерпретатор 24 производит запрос относительно того, является ли отсчет S пропуска больше или равным разности между приоритетом H активного виртуального потока с самым высоким приоритетом в текущей группе приоритетов и приоритетом P выбранного потока. Если отсчет S пропуска больше или равен разности H-P, то интерпретатор 24 производит сброс счетчика 116 пропусков выбранного потока обратно в ноль на этапе 126 и контекстное переключение на этапе 128. Выбранный поток становится теперь потоком, выполняемым в текущий момент времени. Если отсчет S пропуска меньше, чем разность H-P, интерпретатор 24 увеличивает содержимое счетчика 116 пропусков выбранного потока на этапе 130 и переходит на этап 120 для выбора другого потока в текущей группе приоритетов из списка 56 активных потоков.

Если все это основывается на относительных приоритетах потоков 58, а не на абсолютных приоритетах, то одинаковое число операций пропуска будет иметь место как для пары потоков с приоритетами 10 и 20, так и для пары потоков с приоритетами 510 и 520 - в любом случае, разность приоритетов равна 10, поэтому поток с низким приоритетом получит один квант из каждых десяти квантов времени, которые получает поток с самым высоким приоритетом.

Выполнение команд

В течение кванта времени активного виртуального потока 58 интерпретатор 24, а более конкретно блок 28 выполнения команд собственного потока 50, которому этот виртуальный поток был повторно назначен, производит считывание (этап 132) и выполнение (этап 134)

команд из виртуального потока настолько быстро, насколько это возможно. После выполнения каждой команды на этапе 134 собственный поток 50 (то есть соответствующий вариант интерпретатора 24) делает ряд проверок 136, 138, 140 для определения того, становится ли текущий виртуальный поток неактивным в результате команды, завершается ли текущий поток в результате команды, которая была только что выполнена, или истек ли квант времени виртуального потока. Если любое из этих условий истинно, то собственный поток 50 или соответствующий интерпретатор 24 прекращает выполнение команд из этого виртуального потока до тех пор, пока этот поток не станет активным или не будет назначен новый квант времени.

Квант времени или интервал времени обработки можно измерить путем использования таймера или счетчика команд, причем последний отслеживается с помощью счетчика 30 команд (фиг.3). В общем, лучше использовать подсчет команд, потому что непроизводительные затраты на каждую команду намного ниже. Проверка того, истек ли квант времени, заключается просто в увеличении переменной счетчика, с последующей проверкой, прошла ли переменная счетчика максимальное число команд для кванта времени. Сравнение подсчета команд с предопределенным максимальным значением подсчета можно предпринять с помощью модуля 32 контекстного переключения (фиг.3).

Путем выбора правильного размера для кванта времени распознают, что более продолжительные кванты времени делают более эффективным использование процессора 12 (таким образом добиваясь более быстрой работы), но уменьшают число контекстных переключений, которые могут происходить в пределах данного периода времени, что может привести в результате к ошибочной работе интерфейсов пользователя. В зависимости от лежащей в основе операционной системы, платформы аппаратных средств и характера приложения могут иметь смысл различные размеры квантов времени. Как правило, значение между 20 и 200 командами на один квант времени работает хорошо. Выбор кванта времени слишком маленьким (от 1 до 3 команд) сильно влияет на производительность, а выбор кванта времени слишком большим (миллионы команд), по существу, делает недееспособной организацию многопоточной обработки, особенно для любого вида приложения с графическим интерфейсом пользователя.

Если собственный поток 50, то есть соответствующий вариант (экземпляр) интерпретатора 24, обнаруживает при проверке 136, что поток, выполняемый в текущий момент времени, стал неактивным в результате своей последней выполненной команды, то поток удаляют из соответствующего списка 56 активных потоков на этапе 142. Это удаление требует настройки указателя активного потока, который непосредственно предшествует удаленному потоку в списке 56 активных потоков. Флаг 52 очереди из соответствующего собственного потока 50 затем блокируют на этапе 144, вновь неактивный поток вставляют в список 34 неактивных потоков на этапе 146, и флаг 52 разблокируют на этапе 148. Вставка потока в список 34 неактивных потоков влечет за собой настройку указателей вставленного потока и потока, непосредственно предшествующего вставленному потоку в списке 34 неактивных потоков.

Если собственный поток 50, то есть соответствующий вариант интерпретатора 24, обнаруживает при проверке 138, что поток, выполняемый в текущий момент времени, завершил свою задачу или работу в результате последней выполненной команды, поток удаляется из активного списка 56 на этапе 150. Кроме того, высвобождаются ресурсы, которые использовались завершенным потоком.

После перемещения потока из активного списка 56 в неактивный список 34 на этапах 142, 144, 146, 148 или после удаления завершенного потока из активного списка 56 на этапе 150 интерпретатор 24 релевантного собственного потока 50 проводит на этапе 152 проверку того, имеются ли какие-либо потоки, оставленные в активном списке 56 в группе с тем же самым приоритетом, как и текущий поток. Если это так, то интерпретатор 24 переходит к выполнению задач поддержки на этапе 70 (фиг.5А). Если нет, то интерпретатор 24 или собственный поток 50 просматривает активный список 56 на этапе 154 для того, чтобы определить приоритет потока с самым высоким приоритетом в

списке. На следующих этапах 156 и 158 поток с самым высоким приоритетом становится текущим потоком, и подсчитывают число потоков в активном списке 56 с самым высоким уровнем приоритета. Интерпретатор 24 возвращается к выполнению задач поддержки на этапе 70.

После того как было определено на этапах 136 и 138, что текущий поток не стал неактивным или не завершил свою задачу после выполнения последней команды байт-кода, интерпретатор 24 производит запрос на этапе 160 принятия решения, изменилась ли группа приоритетов текущего потока в результате команды, которая была только что выполнена. Если нет, то предпринимают проверку 140 для того, чтобы определить, истек ли квант времени. Если это так, то интерпретатор 24 на этапе 162 запрашивает, увеличилась или уменьшились группа приоритетов текущего потока. В случае увеличения число потоков в группе с самым высоким приоритетом вновь устанавливается равным единице на этапе 164. В случае уменьшения подсчет потоков в группе с самым высоким приоритетом уменьшается на этапе 166. Интерпретатор 24 проводит исследование на этапе 168 принятия решения 166, имеются ли какие-либо активные потоки, оставшиеся в группе с самым высоким приоритетом. Отрицательный результат этой проверки приводит к тому, что интерпретатор 24 просматривает активные потоки на этапе 170 для того, чтобы определить новую группу с самым высоким приоритетом и подсчитать число потоков в этой группе. Интерпретатор 24 затем производит контекстное переключение на этапе 172, при этом активный поток 58 с самым высоким приоритетом становится текущим потоком.

Положительный результат проверки или этапа 168 принятия решения приводит к тому, что интерпретатор 24 переходит непосредственно к этапу 172 контекстного переключения и после этого этапа к считыванию и выполнению этапов 132 и 134.

Балансировка загрузки потоков

Как установлено выше, собственных потоков 50 может быть много, причем каждый из них выполняет команды из связанного списка 56 активных виртуальных потоков 58. Когда виртуальный поток становится активным и назначен собственному потоку 50, система пытается назначать его собственному потоку с самой малой загрузкой. Это позволяет сохранить виртуальные потоки равномерно сбалансированными между собственными потоками.

Так как активные потоки завершаются или становятся неактивными, они необязательно будут выполняться равномерным способом - в системе с двумя собственными потоками, каждый из назначенных десяти виртуальных потоков, весьма вероятно для шести виртуальных потоков в одном собственном потоке, должен полностью завершаться или становиться неактивным, оставляя два собственных потока - один с четырьмя виртуальными потоками, а другой с десятью.

Желательно избегать несбалансированных или неравных загрузок среди собственных потоков 50, потому что такое состояние не позволяет эффективно использовать процессоры на многопроцессорной машине. Для того чтобы решить эту проблему, виртуальная организация поточной обработки, проводимая интерпретатором 24, использует метод под названием "балансировка потоков". Основной принцип балансировки потоков заключается в том, что интерпретатор 24 должен периодически проверять набор выполняемых виртуальных потоков 58 и перераспределять их для поддержания сбалансированного распределения виртуальных потоков среди собственных потоков 50. Балансировку потоков необходимо проводить с особой тщательностью во избежание создания собственных потоков 50, которые останавливаются для ожидания того, чтобы заблокированные данные стали доступным.

Любой собственный поток может выполнять балансировку потоков. Каждый собственный поток 50 поддерживает счетчик (не показан), который увеличивает собственный поток каждый раз, когда поток завершает выполнение полного набора квантов времени (то есть собственный поток, который достиг конца своего списка 56 активных потоков 58 и снова стартовал с начала). Когда этот счетчик достигает определенного значения, которое выбирают в соответствии с платформой, ожидаемым набором приложений и

предпочтениями оператора, счетчик сбрасывается в ноль, и собственный поток 50 пытается выполнить балансировку потоков.

Когда наступает время для собственного потока 50, чтобы выполнить необходимую балансировку потоков, следует пропустить операцию балансировки, если другой собственный поток находится уже в процессе выполнения балансировки. Координация балансировки потоков среди нескольких собственных потоков 50 выполняется через использование глобального флага (не показано). Собственный поток 50 должен иметь блокировку на этом флаге для того, чтобы выполнить балансировку потоков. Когда собственный поток 50 получает доступ к глобальному флагу и определяет, что он заблокирован, собственный поток не ждет флага, чтобы стать разблокированным, а вместо этого пропускает балансировку потоков.

В том случае, если собственный поток 50 имеет блокировку по флагу балансировки, собственный поток должен повторить итерацию по таблице 36 собственных потоков дважды. Каждый собственный поток 50 поддерживает значение загрузки потока (описанное выше, которое обычно равно сумме приоритетов всех активных потоков). Во время первого прохождения таблицы 36 собственных потоков поток 50, который выполняет балансировку, суммирует значения загрузки всех собственных потоков. Эта сумма всех собственных загрузок потока затем делится на число собственных потоков 50 для того, чтобы получить среднюю загрузку. Во время второго прохождения списка собственных потоков, если любой собственный поток 50 имеет загрузку, которая больше средней загрузки, этот активный список 56 собственных потоков блокируется, и виртуальные потоки удаляют из активного списка, начиная с потоков с низкими приоритетами, и перемещают по направлению к потокам с высокими приоритетами до тех пор, пока загрузка собственного потока не будет равна или меньше средней загрузки.

Эти виртуальные потоки затем назначают собственным потокам, как будто они только что стали активными, с использованием обычного алгоритма (определяют собственный поток с самой малой загрузкой и назначают ему виртуальный поток).

Потоки, имеющие имя

Когда приложение выполняет команду, которая создает новый поток, приложение обычно принимает уникальный идентификатор для потока, который был только что создан. Эти идентификаторы представляют собой обычно уникальные числовые коды, которые назначаются динамическим способом, когда потоки уже созданы, и они могут использоваться для определения целевых адресов при передаче сообщения между потоками.

Однако для одного приложения иногда требуется осуществить обмен сообщениями с другим приложением, которое уже выполняется. В этом случае, динамически назначенный числовой ИД целевого потока не доступен отправляющему приложению.

Поэтому рекомендуется, чтобы интерпретируемый язык выполнял команду для присвоения имени потоку или выполнял такую функциональную часть команды, которая создает новый поток. Имя потока представляет собой строку текста, который может использовать для уникальной идентификации потока. Как правило, разработчик приложений будет использовать некоторый идентификатор, который является уникальным для него (например, порядковый номер их пакета программ для разработки) и объединенный с подробным именем для приложения (такого как "Белая доска" ("Whiteboard")), для того, чтобы создать уникальное имя потока, которое они могут использовать для этого приложения.

Когда приложение должно посылать сообщение в другой поток, оно может сделать это с помощью динамически назначенного уникального числового ИД потока или имени потока, если было присвоено имя.

Сообщения потоков и организация сети

Сообщения, передаваемые между потоками, представляют собой мощное средство для поддержания связи между компонентами приложения, особенно между элементами пользовательского интерфейса, такими как кнопки и полосы прокрутки, и с именами

потоков они образуют мощный механизм для поддержания связи между приложениями, выполняемыми на одном компьютере.

Кроме того, сообщения потока позволяют сформировать мощное средство связи между приложениями на различных компьютерах. Если это реализовано правильно, то будет ясно ("прозрачно") для приложений и разработчиков приложений, передается ли сообщение потока в поток, имеющий имя, на локальном компьютере, или поток, имеющий имя, на удаленном компьютере не должен влиять на характер команд, необходимых для передачи сообщения.

Эта связь между приложениями на различных компьютерах выполняется путем использования промежуточных (прокси)потоков 174 и 176 (см. фиг.6). Когда поток 178 на локальном компьютере 180 должен установить связь с потоком 182 на удаленном компьютере 184, поток 178 на локальном компьютере 180 должен выполнить команду для подсоединения к удаленному потоку 182. Когда команда выполняется, создается новый локальный промежуточный поток 174. Этот промежуточный поток 174 выполняет модуль кода, который подсоединяется к удаленному компьютеру 184. Удаленный компьютер 184 принимает соединение и создает свой собственный промежуточный (прокси)поток 176. Промежуточные потоки 174 и 176 затем поддерживают связь друг с другом по сети 186 с использованием стандартных сетевых протоколов, таких как протокол управления передачей/Интернет-протокол (ПУП/ИП (TCP/IP)).

Локальный поток 178, который выполнил первоначальную команду соединения, теперь принимает ИД локального промежуточного потока 174. Локальный поток 178 может использовать этот ИД, как если бы он имел ИД удаленного целевого потока 182, то есть локальный поток 178 позволяет использовать ИД локального промежуточного потока 174 в качестве адреса для сообщений между потоками, которые нацелены на удаленной поток 182.

Каждый раз, когда локальный промежуточный (прокси)поток 174 принимает сообщение, он создает представление этого сообщения в двоичном буфере (не показано) и посылает этот буфер по сети 186 в удаленный промежуточный (прокси)поток 176 с использованием стандартного протокола. Удаленный промежуточный поток 176 затем транслирует представление двоичного буфера сообщения обратно в стандартное сообщение и направляет сообщение дальше в удаленный поток 182.

Та же самая система используется для передачи сообщений в другом направлении - если удаленный промежуточный поток 176 принимает сообщение, то сообщение транслируется и передается по сети в локальный промежуточный поток 174, который направляет его по направлению к локальному потоку 178.

Следует понимать, что модули процессора, раскрытые здесь, могут представлять собой "жестко встроенные" компоненты или обычные компьютерные схемы, изменяемые путем программирования для выполнения указанных функций. Таким образом, интерпретатор можно реализовать с помощью цифровой схемы, которую изменяют посредством программного обеспечения, для того чтобы скомпилировать исходную программу пользователя в виде псевдокода, создать виртуальные потоки для осуществления задач или заданий, обработать команды псевдокодов, выбранные в соответствии с выполняемым виртуальным потоком и так далее.

Кроме того, следует понимать, что приведенное здесь описание интерпретатора 24 и его работы применимо к любому варианту интерпретатора, который выполняется, то есть к различным собственным потокам 50.

Хотя изобретение было описано на примере конкретных вариантов осуществления и приложений, специалистам будет ясно в свете этого описания, что можно выработать дополнительные варианты осуществления и модификации без отклонения от сущности или превышения объема заявленного изобретения. Соответственно, следует понимать, что чертежи и их описание приведены здесь в качестве примера для облегчения понимания настоящего изобретения и не должны быть истолкованы для ограничения его объема.

Формула изобретения

1. Способ работы многозадачного компьютера, содержащий этапы, в соответствии с которыми сохраняют в памяти компьютера множество команд псевдокода, причем, по меньшей мере, некоторые из упомянутых команд псевдокода содержат множество команд в кодах машины, для каждого из множества задач или заданий, которые должны быть выполнены с помощью компьютера, автоматически создают соответствующий виртуальный поток данных контекста выполнения, включающий в себя (a) местоположение в памяти очередной одной из команд псевдокода, которые должны быть выполнены при выполнении соответствующей задачи или задания, и (b) значения любых локальных переменных, требуемых для выполнения соответствующей задачи или задания, при этом множество задач или заданий, каждая из которых влечет за собой выполнение соответствующей одной из команд псевдокода, содержит множество команд на машинном языке, обрабатывают каждую из задач или заданий в соответствующей последовательности квантов времени или интервалов времени обработки под управлением соответствующего виртуального потока, и при каждом контекстном переключении между различными виртуальными потоками предпринимают контекстное переключение только после завершения выполнения одной из команд псевдокода, выполняющейся в текущий момент времени.

2. Способ по п.1, по которому каждый из виртуальных потоков является частью соответствующего связанного списка виртуальных потоков, при этом каждый из виртуальных потоков дополнительно включает в себя указатель на следующий виртуальный поток в соответствующем связанном списке, дополнительно содержащий этап, в соответствии с которым для каждого контекстного переключения между различными виртуальными потоками обращаются к указателю выполняемого в настоящий момент виртуального потока для того, чтобы определить идентификационные данные следующего виртуального потока, который должен выполняться.

3. Способ по п.2, по которому соответствующий связанный список является одним из множества связанных списков виртуальных потоков, при этом один из связанных списков представляет собой список неактивных виртуальных потоков, другой из связанных списков представляет собой список активных виртуальных потоков, следующий один из связанных списков представляет собой список поставленных в очередь виртуальных потоков, дополнительно содержащий этап, в соответствии с которым периодически перемещают, по меньшей мере, один виртуальный поток из списка поставленных в очередь виртуальных потоков в список активных виртуальных потоков.

4. Способ по п.3, по которому этап перемещения виртуального потока из списка поставленных в очередь виртуальных потоков в список активных виртуальных потоков включает в себя подэтапы, в соответствии с которыми устанавливают флаг для блокировки списка поставленных в очередь виртуальных потоков, затем изменяют указатели: в перемещенном виртуальном потоке, в, по меньшей мере, одном виртуальном потоке, который изначально находился в списке активных виртуальных потоков, и в, по меньшей мере, в одном виртуальном потоке, который остается в списке поставленных в очередь виртуальных потоков, и после этого сбрасывают или разблокируют флаг для того, чтобы разрешить доступ к списку поставленных в очередь виртуальных потоков.

5. Способ по п.1, по которому каждый из виртуальных потоков дополнительно включает в себя флаг, дополнительно содержащий этапы, в соответствии с которыми устанавливают флаг выбранного одного из виртуальных потоков, затем изменяют данные в выбранном одном из виртуальных потоков, и после этого сбрасывают или разблокируют флаг, чтобы разрешить доступ к выбранному одному из виртуальных потоков.

6. Способ по п.5, по которому установку флага выбранного одного из виртуальных потоков, изменение данных и сброс или разблокировку флага выбранного одного из виртуальных потоков выполняют в ответ на сообщение из другого одного из виртуальных потоков.

7. Способ по п.5, по которому каждый из виртуальных потоков является частью

соответствующего связанного списка виртуальных потоков, при этом каждый из виртуальных потоков дополнительно включает в себя указатель на следующий виртуальный поток в соответствующем связанном списке, изменение данных включает в себя изменение указателя выбранного одного из виртуальных потоков.

8. Способ по п.1, по которому каждому из виртуальных потоков назначается очередь сообщений, дополнительно содержащий этап ввода сообщения в очередь сообщений выбранного одного из виртуальных потоков во время выполнения задачи или задания, соответствующей другому одному из виртуальных потоков.

9. Способ по п.8, по которому упомянутый выбранный один из виртуальных потоков и упомянутый другой один из виртуальных потоков соответствуют соответствующим задачам или заданиям, полученным из различных прикладных программ, посредством чего при вводе сообщения в очередь сообщений выбранного одного из виртуальных потоков осуществляется передача данных между различными прикладными программами.

10. Способ по п.8, по которому выбранный один из виртуальных потоков и другой один из виртуальных потоков являются промежуточными или интерфейсными потоками на различных компьютерах, при этом ввод сообщения в очередь сообщений включает в себя передачу упомянутого сообщения по линии связи между компьютерами.

11. Способ по п.1, по которому создание виртуальных потоков, обработка задач или заданий в соответствующей последовательности квантов времени или интервалов времени обработки и совершение контекстных переключений включают в себя работу компьютера согласно программе интерпретатора.

12. Способ по п.11, дополнительно содержащий этап выполнения множества экземпляров программы интерпретатора на компьютере, причем каждый экземпляр соответствует собственному потоку, при этом каждый собственный поток создает соответствующий набор виртуальных потоков данных контекста выполнения, обрабатывает каждую из множества задач или заданий в соответствующей последовательности квантов времени или интервалов времени обработки под управлением соответствующего виртуального потока, и при каждом контекстном переключении между различными виртуальными потоками предпринимает такое контекстное переключение только после завершения выполнения выполняемой в текущий момент времени одной из команд псевдокода.

13. Способ по п.12, дополнительно содержащий перемещение виртуального потока из первого собственного потока, имеющего загрузку выше среднего уровня, во второй собственный поток, имеющий загрузку ниже среднего уровня.

14. Способ по п.13, по которому перемещение виртуального потока включает в себя этапы, в соответствии с которыми определение средней загрузки по всем собственным потокам путем суммирования значений загрузки потока для собственных потоков и деления на число потоков, и для каждого из собственных потоков сравнение соответствующего значения загрузки потока со средней загрузкой для определения относительной загрузки.

15. Способ по п.1, по которому виртуальные потоки включают в себя первый промежуточный поток для осуществления обмена со вторым промежуточным потоком на другом компьютере через линию связи компьютерной сети, обработку обмена с другим компьютером, включающую в себя использование стандартных сетевых протоколов под управлением первого промежуточного потока.

16. Способ по п.15, по которому каждому из виртуальных потоков, включающих в себя первый промежуточный поток, назначается соответствующая очередь сообщений, дополнительно содержащий ввод сообщения в очередь сообщений первого промежуточного потока для выполнения передачи данных в другой компьютер по линии связи компьютерной сети.

17. Способ по п.1, по которому выбранный один из виртуальных потоков находится в неактивном состоянии, дополнительно содержащий этапы, в соответствии с которыми формируют сообщение в ответ на ввод из источника, расположенного вне компьютера, вставляют сообщение в очередь сообщений для выбранного одного из виртуальных

потоков, переводят выбранный один из виртуальных потоков из неактивного состояния в активное состояние и после этого вставляют сообщения в очередь сообщений и изменяют состояние выбранного одного из виртуальных потоков, осуществляют доступ к очереди сообщений для получения сообщения во время кванта времени или интервала времени обработки, назначенного выбранному одному из виртуальных потоков.

18. Способ по п.1, по которому каждый из виртуальных потоков дополнительно включает в себя приоритет потока, дополнительно содержащий этап автоматического обращения к приоритетам потоков в множестве виртуальных потоков для определения относительных приоритетов и изменения последовательности потоков в соответствии с определенными относительными приоритетами.

19. Способ по п.1, по которому задачи или задания, обработанные в соответствующей последовательности квантов времени или интервалов времени обработки под управлением соответствующих виртуальных потоков, включают в себя этапы, в соответствии с которыми управляют объектами, которые отображаются на устройстве отображения компьютера, при этом каждый из объектов относится к отдельной задаче или заданию, назначенному соответствующему одному из виртуальных потоков, и контролируют срабатывание клавиш на клавиатуре компьютера, при этом каждая из клавиш относится к отдельной задаче или заданию, назначенному соответствующему одному из виртуальных потоков.

20. Способ по п.1, по которому кванты времени или интервалы времени обработки измеряют путем подсчета последовательно выполненных команд псевдокода, дополнительно содержит этап, в соответствии с которым для каждого из множества квантов времени или интервалов времени обработки завершают соответствующий квант времени или интервал времени обработки после подсчета предопределенного числа последовательно выполненных команд псевдокода.

21. Многозадачный компьютер, содержащий память, устройство отображения, периферийное устройство ввода, по меньшей мере, один процессор, оперативно подсоединенный к памяти, устройству отображения и периферийному устройству ввода, при этом процессор содержит компилятор для преобразования команд исходного кода, введенного оператором, в байт-коды или команды псевдокода, причем компилятор оперативно связан с памятью для обеспечения возможности хранения в ней байт-кода или команд псевдокода, и интерпретатор для выполнения байт-кода или команд псевдокода, при этом память хранит первый связанный список неактивных виртуальных потоков, второй связанный список активных виртуальных потоков и третий связанный список поставленных в очередь или ожидающих виртуальных потоков, причем каждый из потоков включает в себя контекст или данные состояния, флаг и указатель на следующий поток в соответствующем списке, упомянутый интерпретатор оперативно подсоединен к периферийному устройству ввода для распознавания события, сформированного периферийным устройством ввода, при этом интерпретатор оперативно подсоединен к памяти для перемещения, по меньшей мере, одного из неактивных виртуальных потоков из первого связанного списка в третий связанный список, для перемещения поставленных в очередь или ожидающих виртуальных потоков из третьего связанного списка во второй связанный список, для выполнения команд в соответствии с контекстом и данными состояния различных виртуальных потоков во втором связанном списке в последовательных квантах времени или интервалах времени обработки в соответствии с предопределенным списком приоритетов, при этом интерпретатор оперативно подсоединен к устройству отображения, в частности, для изменения объекта на устройстве отображения в ответ на команды, определенные соответствующим активным виртуальным потоком во втором связанном списке.

22. Компьютер по п.21, в котором память дополнительно хранит четвертый связанный список собственных потоков, интерпретатор представляет собой один из множества экземпляров общего интерпретатора, причем каждый из экземпляров общего интерпретатора соответствует соответствующему одному из собственных потоков, второй связанный список является одним из множества связанных списков активных потоков, при

этом каждый из собственных потоков связан с помощью соответствующего указателя с соответствующим одним из связанных списков активных потоков, и третий связанный список является одним из множества связанных списков поставленных в очередь потоков, причем каждый из собственных потоков связан с помощью соответствующего указателя с соответствующим одним из связанных списков поставленных в очередь потоков.

23. Компьютер по п.22, в котором каждый активный поток включает в себя флаг для разрешения блокировки соответствующего потока с помощью одного собственного потока для предотвращения доступа к соответствующему потоку другими собственными потоками.

24. Компьютер по п.22, по которому интерпретатор включает в себя средство, выполняющее перемещение виртуального потока из первого собственного потока, имеющего загрузку выше среднего уровня, во второй собственный поток, имеющий загрузку ниже среднего уровня.

25. Компьютер по п.21, в котором список неактивных виртуальных потоков включает в себя множество потоков, назначенных соответствующим клавишам клавиатуры для обработки срабатываний соответствующих клавиш.

26. Компьютер по п.21, в котором список неактивных потоков включает в себя множество потоков, назначенных соответствующим объектам в изображении на устройстве отображения для обработки изменений во внешнем виде соответствующих объектов.

27. Компьютер по п.21, в котором интерпретатор включает в себя модуль контекстного переключения и счетчик команд, причем модуль контекстного переключения оперативно подсоединен к памяти и счетчику команд для выполнения контекстного переключения от выполняемого в текущий момент времени активного потока второго связанного списка на следующий активный поток во втором связанном списке после выполнения предопределенного числа команд байт-кода или псевдокода в соответствии с выполняемым в текущий момент времени активным потоком.

28. Компьютер по п.21, в котором каждый из виртуальных потоков включает в себя местоположение в памяти следующей команды для выполнения в соответствующем потоке, значения любых локальных переменных для соответствующего потока и приоритет выполнения для соответствующего потока.

29. Компьютер по п.21, в котором память хранит множество очередей сообщений, назначенных соответствующим одним из упомянутых потоков.

30. Компьютер по п.21, в котором память хранит, по меньшей мере, один промежуточный или интерфейсный поток, имеющий контекст выполнения для выполнения связи с удаленным компьютером через линию связи, при этом промежуточный или интерфейсный поток содержит адрес памяти, указывающий программу сетевого протокола.

31. Способ осуществления многозадачной работы компьютера, имеющего интерпретатор для выполнения последовательности команд байт-кода, каждая из которых состоит из многочисленных этапов машинных кодов, содержащий для каждой задачи из множества задач, которые должны выполняться на компьютере, используют интерпретатор для определения соответствующего виртуального потока, во время каждого кванта времени из ряда последовательных квантов времени выполняют команды байт-кода соответствующего текущего потока, выбранного из числа виртуальных потоков, и выполняют контекстное переключение с одного из виртуальных потоков на другой из виртуальных потоков только после выполнения одной из команд байт-кода.

32. Способ по п.31, по которому каждый из виртуальных потоков является частью соответствующего связанного списка виртуальных потоков, при этом каждый из виртуальных потоков дополнительно включает в себя указатель на следующий виртуальный поток в соответствующем связанном списке, дополнительно содержащий этап, в соответствии с которым для каждого контекстного переключения между различными виртуальными потоками обращаются к указателю выполняющегося в текущий момент времени виртуального потока для определения идентификационной информации следующего виртуального потока, который должен выполняться.

33. Способ по п.32, по которому соответствующий связанный список является одним из

множества связанных списков виртуальных потоков, при этом один из связанных списков представляет собой список неактивных виртуальных потоков, другой из связанных списков представляет собой список активных виртуальных потоков, следующий один из связанных списков представляет собой список поставленных в очередь виртуальных потоков, дополнительно содержащий периодическое перемещение, по меньшей мере, одного виртуального потока из списка поставленных в очередь виртуальных потоков в список активных виртуальных потоков.

34. Способ по п.33, по которому перемещение виртуального потока из списка поставленных в очередь виртуальных потоков в список активных виртуальных потоков включает в себя этапы, в соответствии с которыми устанавливают флаг для блокировки списка поставленных в очередь виртуальных потоков, затем изменяют указатели: в перемещенном виртуальном потоке, в, по меньшей мере, одном виртуальном потоке сначала в списке активных виртуальных потоков и в, по меньшей мере, одном виртуальном потоке, оставшемся в списке поставленных в очередь виртуальных потоков, после этого выполняют сброс или разблокировку флага для того, чтобы разрешить доступ к списку поставленных в очередь виртуальных потоков.

35. Способ по п.31, по которому каждый из виртуальных потоков дополнительно включает в себя флаг, дополнительно содержащий этапы, в соответствии с которыми устанавливают флаг выбранного одного из виртуальных потоков, затем изменяют данные в выбранном одном из виртуальных потоков, и после этого выполняют сброс или разблокировку флага для того, чтобы разрешить доступ к выбранному одному из виртуальных потоков.

36. Способ по п.35, по которому установку флага выбранного одного из виртуальных потоков, изменение данных и сброс или разблокировку флага выбранного одного из виртуальных потоков выполняют в ответ на сообщение от другого одного из виртуальных потоков.

37. Способ по п.31, по которому каждому из виртуальных потоков назначается очередь сообщений, дополнительно содержащий этап ввода сообщения в очередь сообщений выбранного одного из виртуальных потоков во время выполнения задачи или задания в соответствии с другим одним из виртуальных потоков.

38. Способ по п.31, по которому виртуальные потоки включают в себя первый промежуточный поток для выполнения обмена со вторым промежуточным потоком на другом компьютере через линию связи, дополнительно содержащий обработку команд байт-кода согласно первому промежуточному потоку для передачи сообщения во второй промежуточный поток по упомянутой линии связи.

39. Способ по п.31, по которому каждый из виртуальных потоков дополнительно включает в себя приоритет потока, дополнительно содержащий автоматическое обращение к приоритетам потоков в множестве виртуальных потоков для определения относительных приоритетов и изменение последовательности потоков в соответствии с определенными относительными приоритетами.

40. Способ по п.31, по которому кванты времени или интервалы времени обработки измеряют путем подсчета последовательно выполненных команд псевдокода, дополнительно содержащий для каждого из множества квантов времени или интервалов времени обработки завершение соответствующего кванта времени или интервала времени обработки после подсчета предопределенного числа последовательно выполненных команд псевдокода.

41. Компьютер, реализующий обработку множества задач, содержащий память, хранящую состояние и контекстные данные многочисленных потоков или задач, интерпретатор для выполнения последовательности команд байт-кода, каждая из которых состоит из множества этапов машинных кодов, после извлечения их из упомянутой памяти, при этом интерпретатор запрограммирован на определение соответствующего виртуального потока для каждой задачи, которая должна выполняться с помощью компьютера, на выполнение команд байт-кода соответствующего текущего потока,

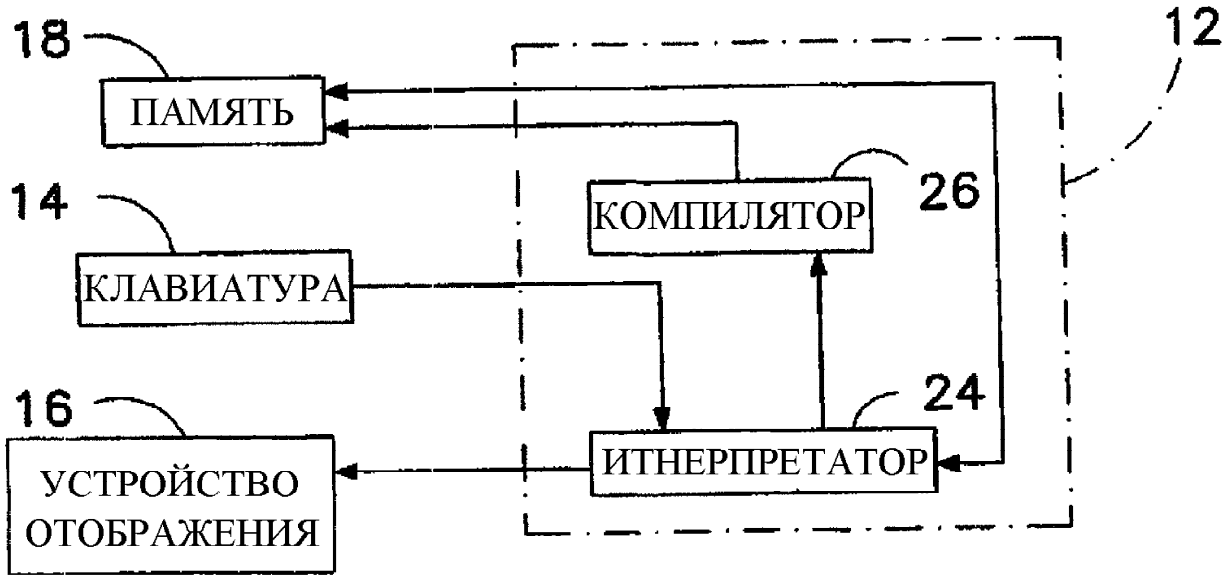
выбранного из числа виртуальных потоков во время каждого кванта времени из ряда последовательных квантов времени, и на выполнение контекстного переключения с одного из виртуальных потоков на другой из виртуальных потоков только после выполнения одной из команд байт-кода.

42. Компьютер по п.41, в котором каждый из виртуальных потоков является частью соответствующего связанного списка виртуальных потоков, при этом каждый из виртуальных потоков дополнительно включает в себя указатель на следующий виртуальный поток в соответствующем связанном списке, причем интерпретатор дополнительно запрограммирован на обращение, для каждого контекстного переключения между различными виртуальными потоками, к указателю выполняемого в текущий момент времени виртуального потока для определения идентификационной информации следующего виртуального потока, который должен выполняться.

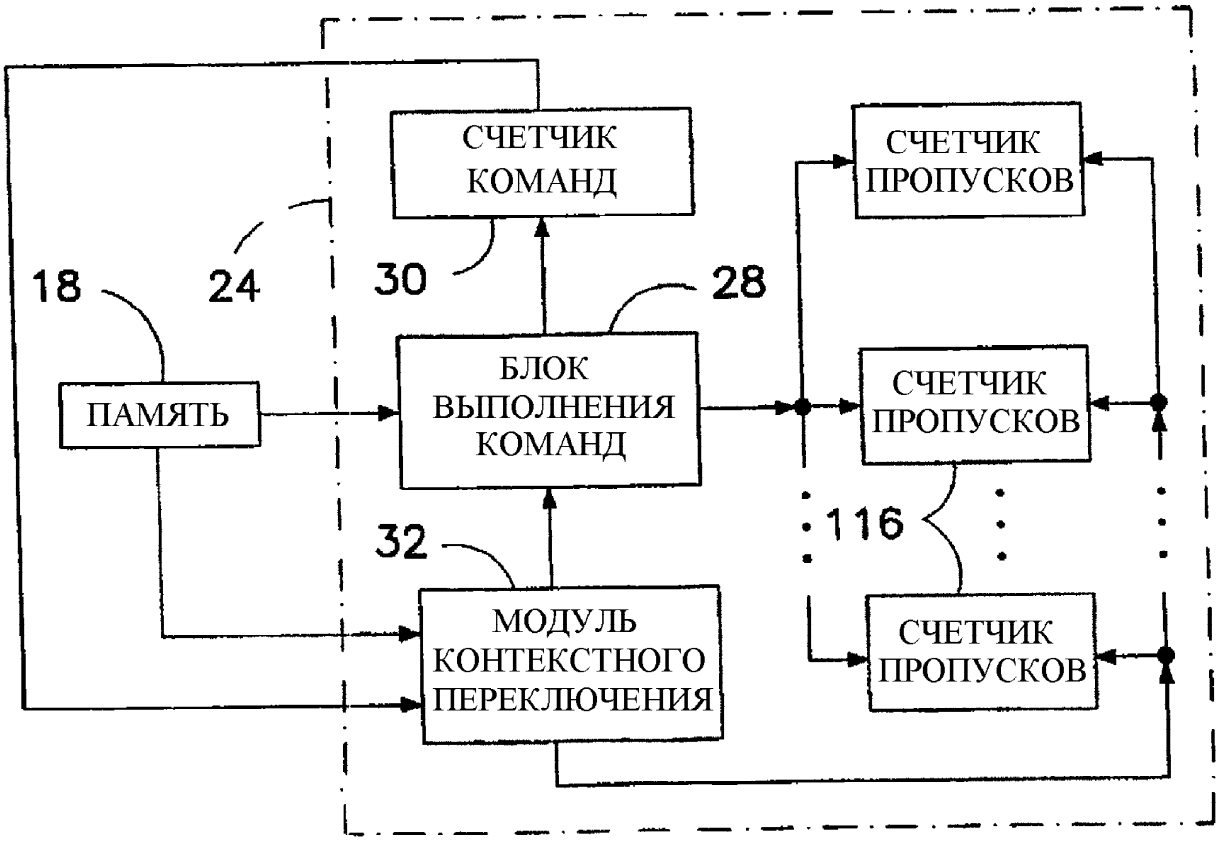
43. Компьютер по п.42, в котором соответствующий связанный список представляет собой один из множества связанных списков виртуальных потоков, один из связанных списков представляет собой один список неактивных виртуальных потоков, другой из связанных списков представляет собой список активных виртуальных потоков, другой один из связанных списков представляет собой список поставленных в очередь виртуальных потоков, при этом интерпретатор дополнительно запрограммирован на периодическое перемещение, по меньшей мере, одного из виртуальных потоков из списка поставленных в очередь виртуальных потоков в список активных виртуальных потоков.

44. Способ управления работой компьютера, содержащий этапы, в соответствии с которыми осуществляют запуск таймера компьютера для формирования последовательности квантов времени или интервалов времени обработки, компилируют входной пользовательский исходный код в команды байт-кода или псевдокода, каждая из которых соответствует множеству команд машинного кода, обеспечивают работу интерпретатора компьютера так, чтобы назначать вычислительные задачи соответствующим виртуальным потокам, назначают вычислительные задачи виртуальным потокам, включая идентификацию и сохранение состояния и контекстных данных для каждой из вычислительных задач, в каждом из квантов времени дополнительно обеспечивают работу интерпретатора для выполнения выбранных одних из команд байт-кода или псевдокода в соответствии с состоянием и контекстными данными текущего одного из виртуальных потоков, после выполнения каждой последующей одной из выбранных команд байт-кода или псевдокода и только после такого выполнения дополнительно обеспечивают работу интерпретатора для проверки того, истек ли предопределенный интервал, начиная с момента выполнения команд в соответствии с текущим одним из виртуальных потоков, и после определения истечения предопределенного интервала обеспечивают работу интерпретатора для выполнения контекстного переключения.

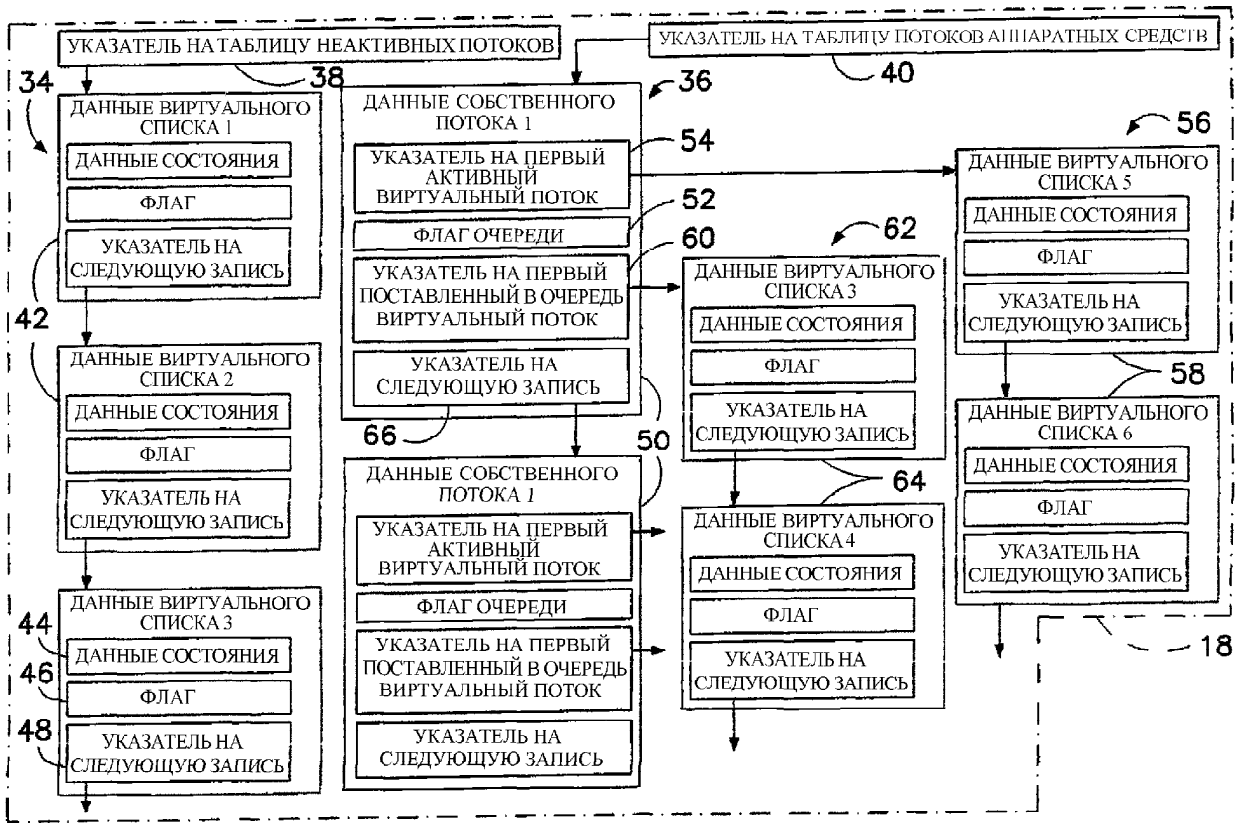
45. Способ по п.44, по которому задачи, назначенные соответствующим одним из виртуальных программных потоков, включают в себя этапы, в соответствии с которыми управляют объектами, появляющимися в изображении на экране устройства отображения, контролируют ввод оператора, выполняют процедуры прикладных программ, выполняют программы компьютерной поддержки, осуществляют связь с удаленными компьютерами через компьютерную сеть и вычисляют локальные переменные.


18
ПАМЯТЬ
12
14
КЛАВИАТУРА
КОМПИЛЯТОР
26
16
УСТРОЙСТВО ОТОБРАЖЕНИЯ
24
ИТНЕРПРЕТАТОР

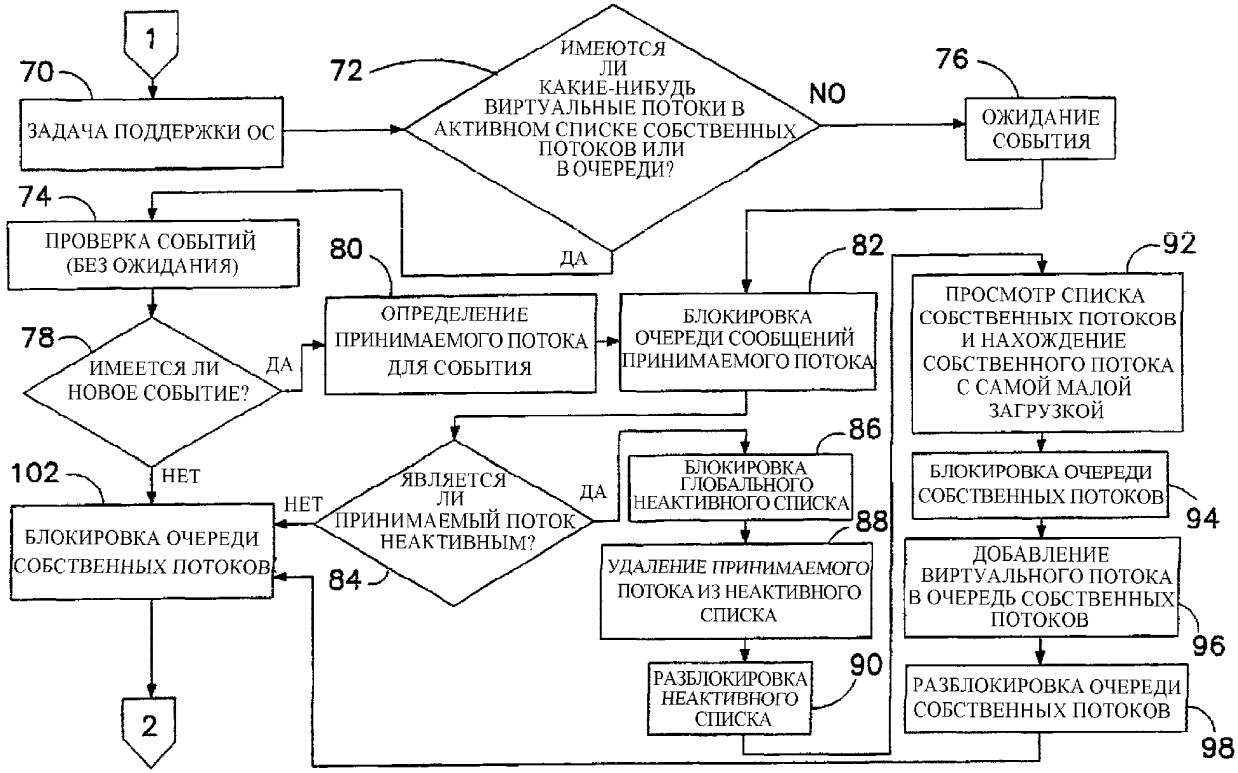
ФИГ.2


СЧЕТЧИК КОМАНД
СЧЕТЧИК ПРОПУСКОВ
24
30
28
18
ПАМЯТЬ
БЛОК ВЫПОЛНЕНИЯ КОМАНД
СЧЕТЧИК ПРОПУСКОВ
116
32
МОДУЛЬ КОНТЕКСТНОГО ПЕРЕКЛЮЧЕНИЯ
СЧЕТЧИК ПРОПУСКОВ

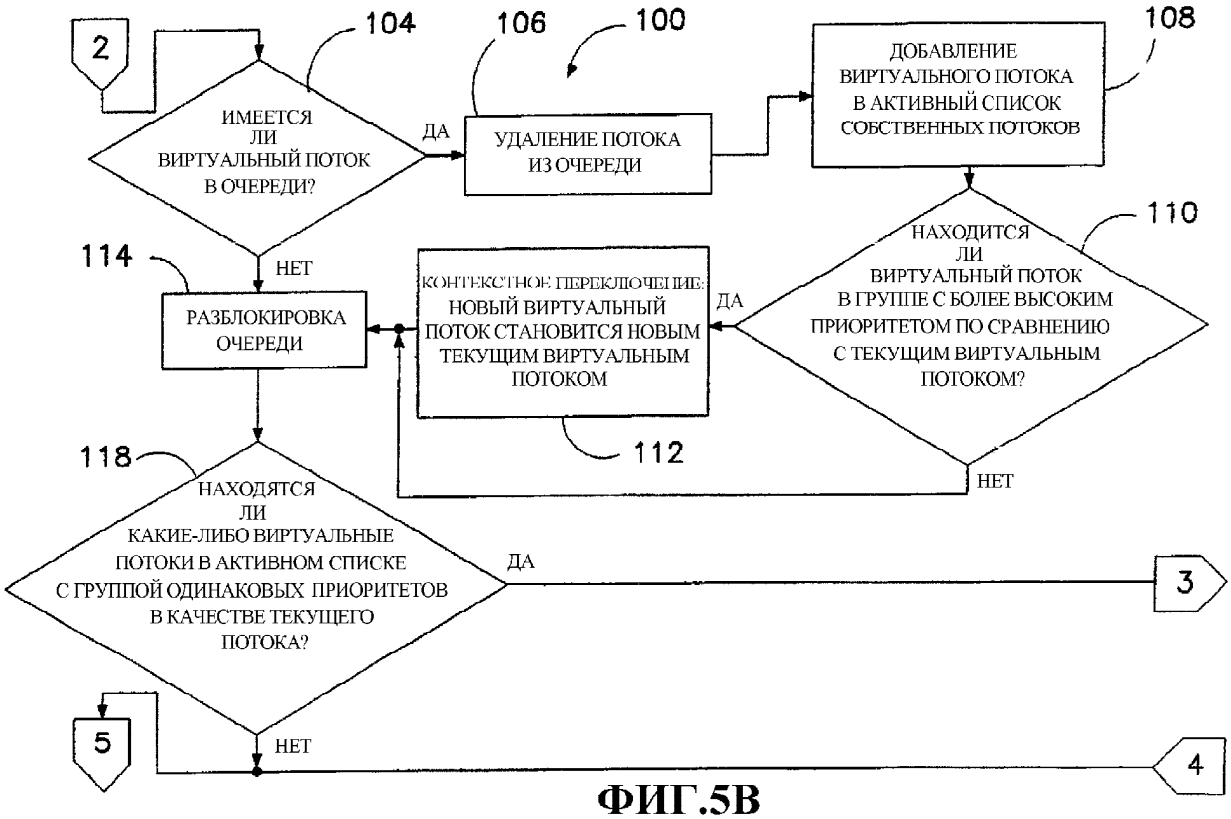
ФИГ.3


УКАЗАТЕЛЬ НА ТАБЛИЦУ НЕАКТИВНЫХ ПОТОКОВ
38
УКАЗАТЕЛЬ НА ТАБЛИЦУ ПОТОКОВ АППАРАТНЫХ СРЕДСТВ
40
34
ДАННЫЕ ВИРТУАЛЬНОГО СПИСКА 1
ДАННЫЕ СОСТОЯНИЯ
ФЛАГ
УКАЗАТЕЛЬ НА СЛЕДУЮЩУЮ ЗАПИСЬ
42
ДАННЫЕ ВИРТУАЛЬНОГО СПИСКА 2
ДАННЫЕ СОСТОЯНИЯ
ФЛАГ
УКАЗАТЕЛЬ НА СЛЕДУЮЩУЮ ЗАПИСЬ
ДАННЫЕ ВИРТУАЛЬНОГО СПИСКА 3
44
ДАННЫЕ СОСТОЯНИЯ
46
ФЛАГ
48
УКАЗАТЕЛЬ НА СЛЕДУЮЩУЮ ЗАПИСЬ
36
ДАННЫЕ СОБСТВЕННОГО ПОТОКА 1
54
УКАЗАТЕЛЬ НА ПЕРВЫЙ АКТИВНЫЙ ВИРТУАЛЬНЫЙ ПОТОК
52
ФЛАГ ОЧЕРЕДИ
60
УКАЗАТЕЛЬ НА ПЕРВЫЙ ПОСТАВЛЕННЫЙ В ОЧЕРЕДЬ ВИРТУАЛЬНЫЙ ПОТОК
УКАЗАТЕЛЬ НА СЛЕДУЮЩУЮ ЗАПИСЬ
66
ДАННЫЕ СОБСТВЕННОГО ПОТОКА 1
50
УКАЗАТЕЛЬ НА ПЕРВЫЙ АКТИВНЫЙ ВИРТУАЛЬНЫЙ ПОТОК
ФЛАГ ОЧЕРЕДИ
УКАЗАТЕЛЬ НА ПЕРВЫЙ ПОСТАВЛЕННЫЙ В ОЧЕРЕДЬ ВИРТУАЛЬНЫЙ ПОТОК
УКАЗАТЕЛЬ НА СЛЕДУЮЩУЮ ЗАПИСЬ
62
ДАННЫЕ ВИРТУАЛЬНОГО СПИСКА 3
ДАННЫЕ СОСТОЯНИЯ
ФЛАГ
УКАЗАТЕЛЬ НА СЛЕДУЮЩУЮ ЗАПИСЬ
64
ДАННЫЕ ВИРТУАЛЬНОГО СПИСКА 4
ДАННЫЕ СОСТОЯНИЯ
ФЛАГ
УКАЗАТЕЛЬ НА СЛЕДУЮЩУЮ ЗАПИСЬ
56
ДАННЫЕ ВИРТУАЛЬНОГО СПИСКА 5
ДАННЫЕ СОСТОЯНИЯ
ФЛАГ
УКАЗАТЕЛЬ НА СЛЕДУЮЩУЮ ЗАПИСЬ
58
ДАННЫЕ ВИРТУАЛЬНОГО СПИСКА 6
ДАННЫЕ СОСТОЯНИЯ
ФЛАГ
УКАЗАТЕЛЬ НА СЛЕДУЮЩУЮ ЗАПИСЬ
18

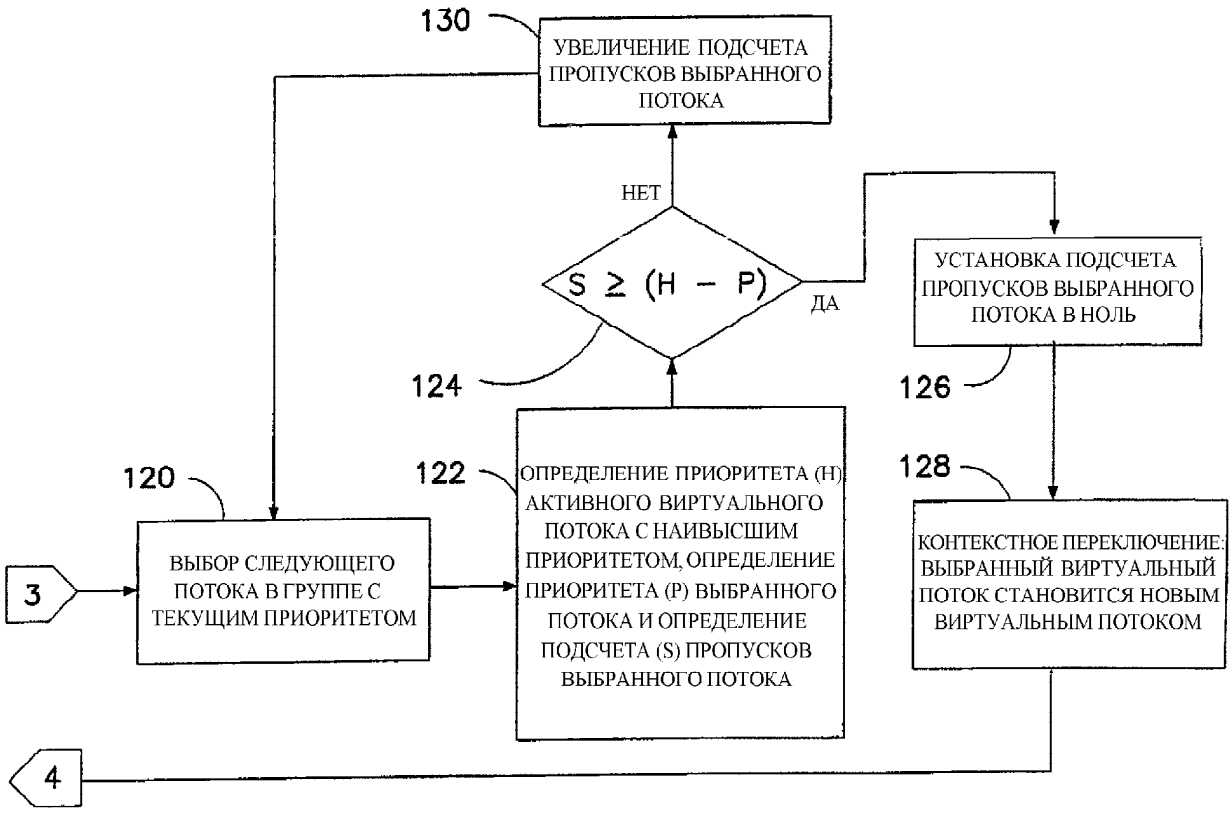
ФИГ.4


1
70
ЗАДАЧА ПОДДЕРЖКИ ОС
72
ИМЕЮТСЯ ЛИ КАКИЕ-НИБУДЬ ВИРТУАЛЬНЫЕ ПОТОКИ В АКТИВНОМ СПИСКЕ СОБСТВЕННЫХ ПОТОКОВ ИЛИ В ОЧЕРЕДИ?
NO
76
ОЖИДАНИЕ СОБЫТИЯ
74
ПРОВЕРКА СОБЫТИЙ (БЕЗ ОЖИДАНИЯ)
ДА
80
ОПРЕДЕЛЕНИЕ ПРИНИМАЕМОГО ПОТОКА ДЛЯ СОБЫТИЯ
82
БЛОКИРОВКА ОЧЕРЕДИ СООБЩЕНИЙ ПРИНИМАЕМОГО ПОТОКА
92
ПРОСМОТР СПИСКА СОБСТВЕННЫХ ПОТОКОВ И НАХОЖДЕНИЕ СОБСТВЕННОГО ПОТОКА С САМОЙ МАЛОЙ ЗАГРУЗКОЙ
78
ИМЕЕТСЯ ЛИ НОВОЕ СОБЫТИЕ?
ДА
НЕТ
86
БЛОКИРОВКА ГЛОБАЛЬНОГО НЕАКТИВНОГО СПИСКА
94
БЛОКИРОВКА ОЧЕРЕДИ СОБСТВЕННЫХ ПОТОКОВ
102
БЛОКИРОВКА ОЧЕРЕДИ СОБСТВЕННЫХ ПОТОКОВ
НЕТ
ЯВЛЯЕТСЯ ЛИ ПРИНИМАЕМЫЙ ПОТОК НЕАКТИВНЫМ?
ДА
84
88
УДАЛЕНИЕ ПРИНИМАЕМОГО ПОТОКА ИЗ НЕАКТИВНОГО СПИСКА
ДОБАВЛЕНИЕ ВИРТУАЛЬНОГО ПОТОКА В ОЧЕРЕДЬ СОБСТВЕННЫХ ПОТОКОВ
96
90
РАЗБЛОКИРОВКА НЕАКТИВНОГО СПИСКА
РАЗБЛОКИРОВКА ОЧЕРЕДИ СОБСТВЕННЫХ ПОТОКОВ
98
2

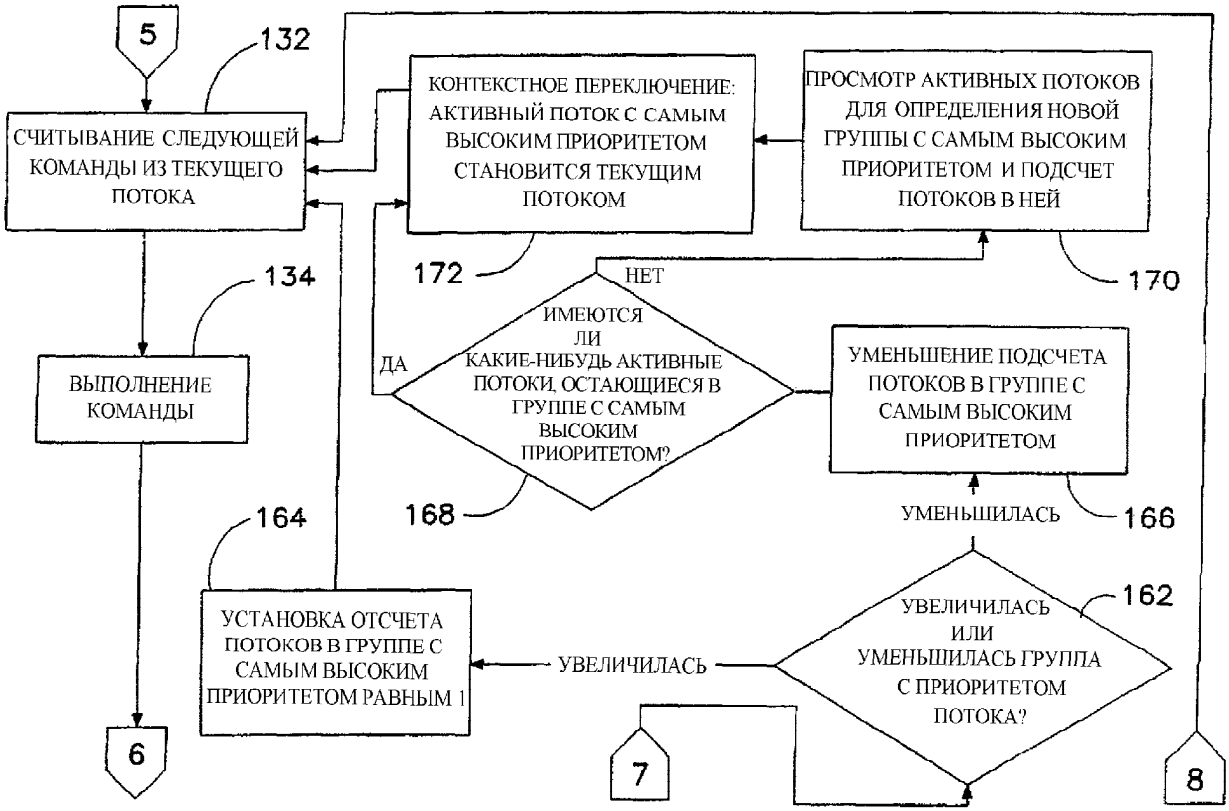
ФИГ.5А


2
104
106
100
108
ИМЕЕТСЯ ЛИ ВИРТУАЛЬНЫЙ ПОТОК В ОЧЕРЕДИ?
ДА
УДАЛЕНИЕ ПОТОКА ИЗ ОЧЕРЕДИ
ДОБАВЛЕНИЕ ВИРТУАЛЬНОГО ПОТОКА В АКТИВНЫЙ СПИСОК СОБСТВЕННЫХ ПОТОКОВ
110
НАХОДИТСЯ ЛИ ВИРТУАЛЬНЫЙ ПОТОК В ГРУППЕ С БОЛЕЕ ВЫСОКИМ ПРИОРИТЕТОМ ПО СРАВНЕНИЮ С ТЕКУЩИМ ВИРТУАЛЬНЫМ ПОТОКОМ?
ДА
НЕТ
114
НЕТ
РАЗБЛОКИРОВКА ОЧЕРЕДИ
КОНТЕКСТНОЕ ПЕРЕКЛЮЧЕНИЕ: НОВЫЙ ВИРТУАЛЬНЫЙ ПОТОК СТАНОВИТСЯ НОВЫМ ТЕКУЩИМ ВИРТУАЛЬНЫМ ПОТОКОМ
112
118
НАХОДЯТСЯ ЛИ КАКИЕ-ЛИБО ВИРТУАЛЬНЫЕ ПОТОКИ В АКТИВНОМ СПИСКЕ С ГРУППОЙ ОДИНАКОВЫХ ПРИОРИТЕТОВ В КАЧЕСТВЕ ТЕКУЩЕГО ПОТОКА?
ДА
3
5
НЕТ
4

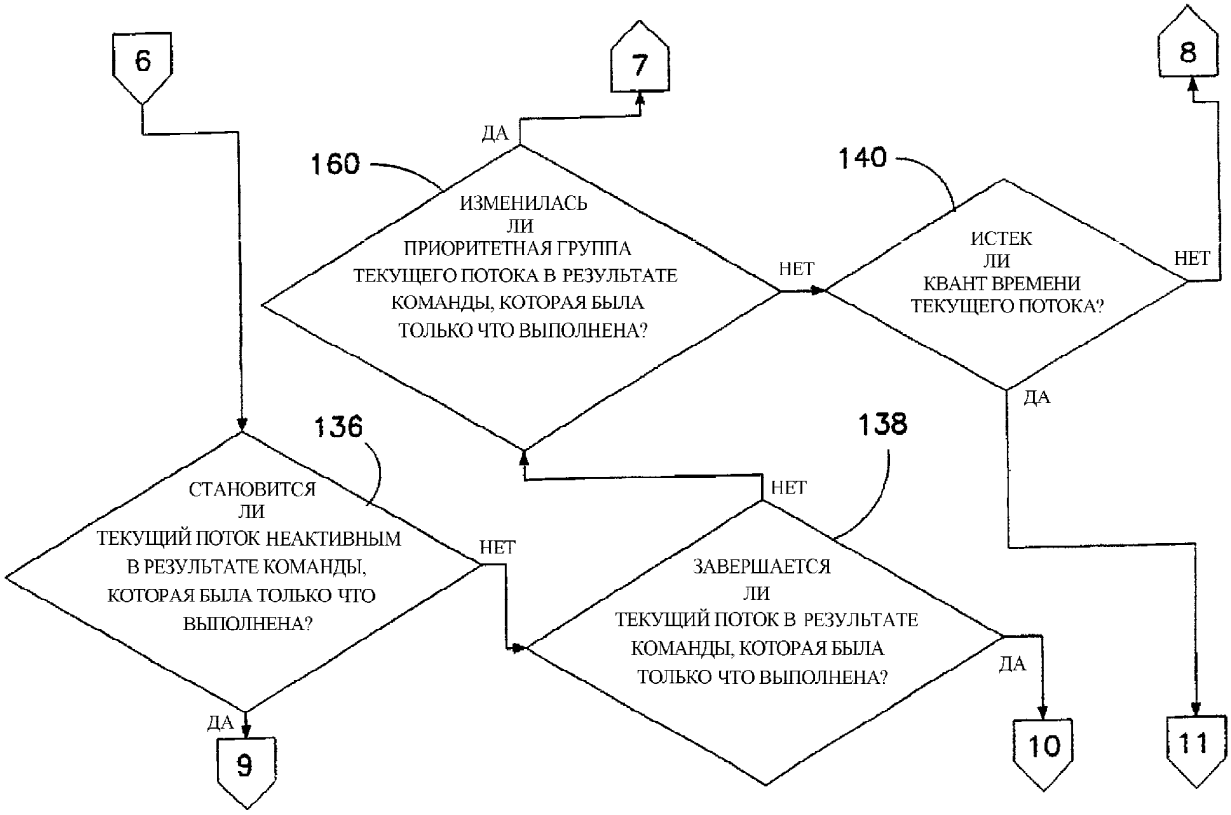
ФИГ.5В


130
УВЕЛИЧЕНИЕ ПОДСЧЕТА ПРОПУСКОВ ВЫБРАННОГО ПОТОКА
НЕТ
S ≥ (H – P)
ДА
УСТАНОВКА ПОДСЧЕТА ПРОПУСКОВ ВЫБРАННОГО ПОТОКА В НОЛЬ
124
126
120
122
128
3
ВЫБОР СЛЕДУЮЩЕГО ПОТОКА В ГРУППЕ С ТЕКУЩИМ ПРИОРИТЕТОМ
ОПРЕДЕЛЕНИЕ ПРИОРИТЕТА (H) АКТИВНОГО ВИРТУАЛЬНОГО ПОТОКА С НАИВЫСШИМ ПРИОРИТЕТОМ, ОПРЕДЕЛЕНИЕ ПРИОРИТЕТА (P) ВЫБРАННОГО ПОТОКА И ОПРЕДЕЛЕНИЕ ПОДСЧЕТА (S) ПРОПУСКОВ ВЫБРАННОГО ПОТОКА
КОНТЕКСТНОЕ ПЕРЕКЛЮЧЕНИЕ: ВЫБРАННЫЙ ВИРТУАЛЬНЫЙ ПОТОК СТАНОВИТСЯ НОВЫМ ВИРТУАЛЬНЫМ ПОТОКОМ
4

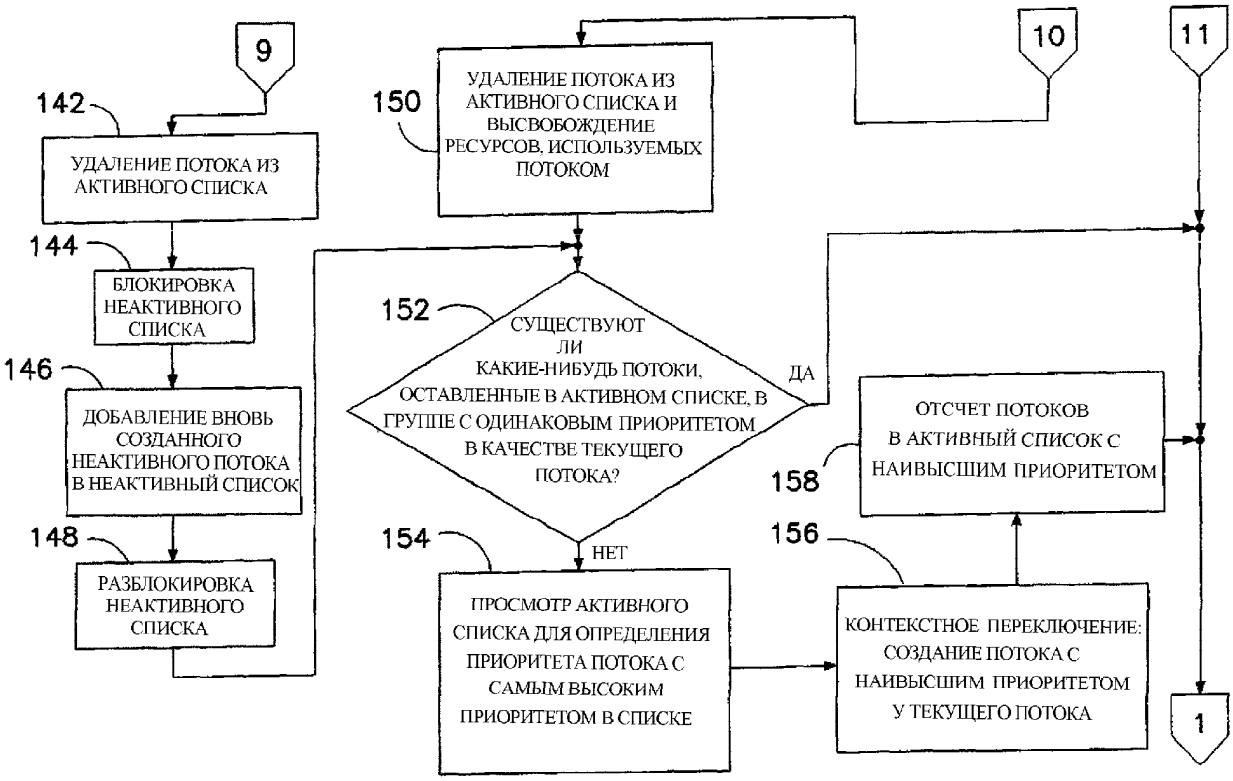
ФИГ.5С


5
132
СЧИТЫВАНИЕ СЛЕДУЮЩЕЙ КОМАНДЫ ИЗ ТЕКУЩЕГО ПОТОКА
КОНТЕКСТНОЕ ПЕРЕКЛЮЧЕНИЕ: АКТИВНЫЙ ПОТОК С САМЫМ ВЫСОКИМ ПРИОРИТЕТОМ СТАНОВИТСЯ ТЕКУЩИМ ПОТОКОМ
ПРОСМОТР АКТИВНЫХ ПОТОКОВ ДЛЯ ОПРЕДЕЛЕНИЯ НОВОЙ ГРУППЫ С САМЫМ ВЫСОКИМ ПРИОРИТЕТОМ И ПОДСЧЕТ ПОТОКОВ В НЕЙ
172
170
НЕТ
134
ВЫПОЛНЕНИЕ КОМАНДЫ
ДА
ИМЕЮТСЯ ЛИ КАКИЕ-НИБУДЬ АКТИВНЫЕ ПОТОКИ, ОСТАЮЩИЕСЯ В ГРУППЕ С САМЫМ ВЫСОКИМ ПРИОРИТЕТОМ?
УМЕНЬШЕНИЕ ПОДСЧЕТА ПОТОКОВ В ГРУППЕ С САМЫМ ВЫСОКИМ ПРИОРИТЕТОМ
168
166
УМЕНЬШИЛАСЬ
164
УСТАНОВКА ОТСЧЕТА ПОТОКОВ В ГРУППЕ С САМЫМ ВЫСОКИМ ПРИОРИТЕТОМ РАВНЫМ 1
УВЕЛИЧИЛАСЬ
УВЕЛИЧИЛАСЬ ИЛИ УМЕНЬШИЛАСЬ ГРУППА С ПРИОРИТЕТОМ ПОТОКА?
162
6
7
8

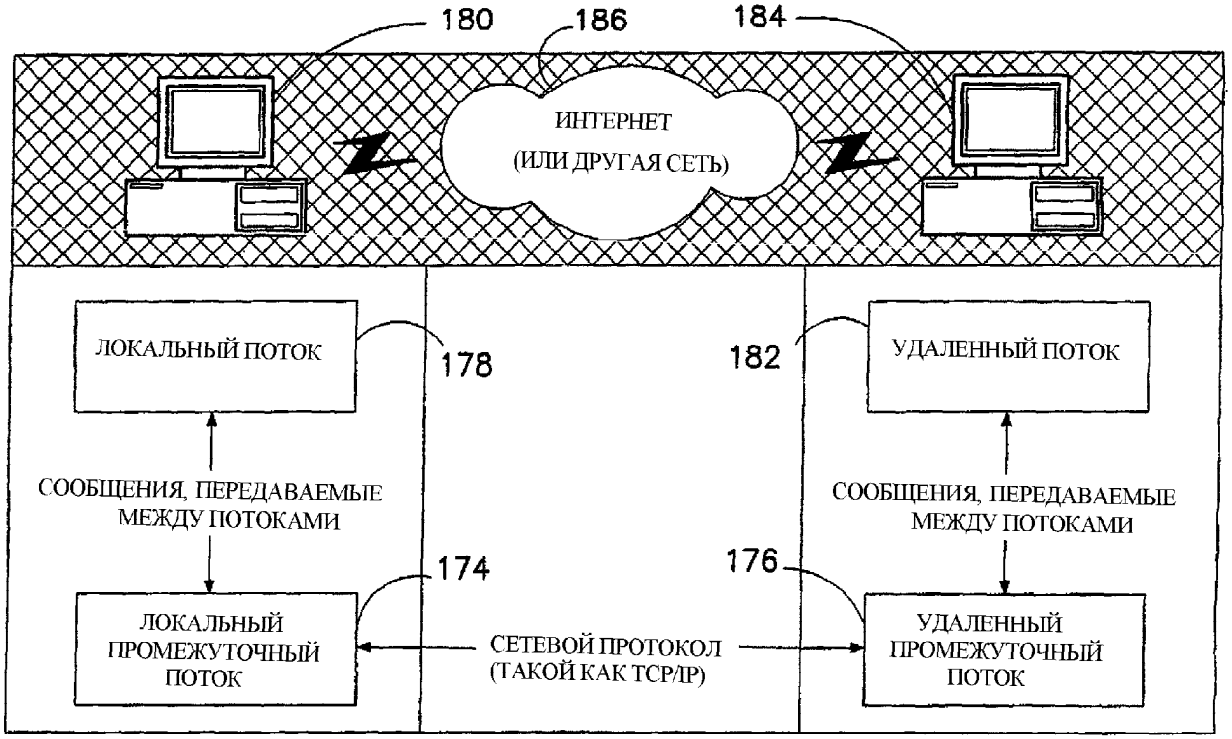
ФИГ.5D


6
7
8
ДА
160
ИЗМЕНИЛАСЬ ЛИ ПРИОРИТЕТНАЯ ГРУППА ТЕКУЩЕГО ПОТОКА В РЕЗУЛЬТАТЕ КОМАНДЫ, КОТОРАЯ БЫЛА ТОЛЬКО ЧТО ВЫПОЛНЕНА?
НЕТ
140
ИСТЕК ЛИ КВАНТ ВРЕМЕНИ ТЕКУЩЕГО ПОТОКА?
НЕТ
ДА
136
СТАНОВИТСЯ ЛИ ТЕКУЩИЙ ПОТОК НЕАКТИВНЫМ В РЕЗУЛЬТАТЕ КОМАНДЫ, КОТОРАЯ БЫЛА ТОЛЬКО ЧТО ВЫПОЛНЕНА?
НЕТ
138
НЕТ
ЗАВЕРШАЕТСЯ ЛИ ТЕКУЩИЙ ПОТОК В РЕЗУЛЬТАТЕ КОМАНДЫ, КОТОРАЯ БЫЛА ТОЛЬКО ЧТО ВЫПОЛНЕНА?
ДА
ДА
9
10
11

ФИГ.5E


9
142
УДАЛЕНИЕ ПОТОКА ИЗ АКТИВНОГО СПИСКА
144
БЛОКИРОВКА НЕАКТИВНОГО СПИСКА
146
ДОБАВЛЕНИЕ ВНОВЬ СОЗДАННОГО НЕАКТИВНОГО ПОТОКА В НЕАКТИВНЫЙ СПИСОК
148
РАЗБЛОКИРОВКА НЕАКТИВНОГО СПИСКА
150
УДАЛЕНИЕ ПОТОКА ИЗ АКТИВНОГО СПИСКА И ВЫСВОБОЖДЕНИЕ РЕСУРСОВ, ИСПОЛЬЗУЕМЫХ ПОТОКОМ
10
11
152
СУЩЕСТВУЮТ ЛИ КАКИЕ-НИБУДЬ ПОТОКИ, ОСТАВЛЕННЫЕ В АКТИВНОМ СПИСКЕ, В ГРУППЕ С ОДИНАКОВЫМ ПРИОРИТЕТОМ В КАЧЕСТВЕ ТЕКУЩЕГО ПОТОКА?
ДА
НЕТ
154
ПРОСМОТР АКТИВНОГО СПИСКА ДЛЯ ОПРЕДЕЛЕНИЯ ПРИОРИТЕТА ПОТОКА С САМЫМ ВЫСОКИМ ПРИОРИТЕТОМ В СПИСКЕ
156
КОНТЕКСТНОЕ ПЕРЕКЛЮЧЕНИЕ: СОЗДАНИЕ ПОТОКА С НАИВЫСШИМ ПРИОРИТЕТОМ У ТЕКУЩЕГО ПОТОКА
158
ОТСЧЕТ ПОТОКОВ В АКТИВНЫЙ СПИСОК С НАИВЫСШИМ ПРИОРИТЕТОМ
1

ФИГ.5F


180
186
184
ИНТЕРНЕТ
(ИЛИ ДРУГАЯ СЕТЬ)
ЛОКАЛЬНЫЙ ПОТОК
178
182
УДАЛЕННЫЙ ПОТОК
СООБЩЕНИЯ, ПЕРЕДАВАЕМЫЕ МЕЖДУ ПОТОКАМИ
СООБЩЕНИЯ, ПЕРЕДАВАЕМЫЕ МЕЖДУ ПОТОКАМИ
174
176
ЛОКАЛЬНЫЙ ПРОМЕЖУТОЧНЫЙ ПОТОК
СЕТЕВОЙ ПРОТОКОЛ (ТАКОЙ КАК TCP/IP)
УДАЛЕННЫЙ ПРОМЕЖУТОЧНЫЙ ПОТОК

ФИГ.6