РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 829 301**(13) **C1**

(51) МПК
G06F 17/40 (2006.01)
G06F 9/50 (2006.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(52) СПК
G06F 17/40 (2024.08); G06F 9/50 (2024.08)

(21)(22) Заявка: **2024117711, 26.06.2024**

(24) Дата начала отсчета срока действия патента:
26.06.2024

Дата регистрации:
30.10.2024

Приоритет(ы):
(22) Дата подачи заявки: **26.06.2024**

(45) Опубликовано: **30.10.2024** Бюл. № 31

Адрес для переписки:
121059, Москва, вн. тер. г. Муниципальный Округ Дорогомилово, ул. Киевская, 7, корп. 2, этаж 11, КОМНАТА 20/Р3, АО "Софит"

(72) Автор(ы):
Дубейко Вячеслав Анатольевич (RU)

(73) Патентообладатель(и):
АКЦИОНЕРНОЕ ОБЩЕСТВО "СОФИТ" (RU)

(56) Список документов, цитированных в отчете о поиске: **US 2021/0133913 A1, 06.05.2021. US 2021/0117360 A1, 22.04.2021. US 11650849 B2, 16.05.2023. US 2021/0141552 A1, 13.05.2021. RU 2322690 C2, 20.04.2008.**

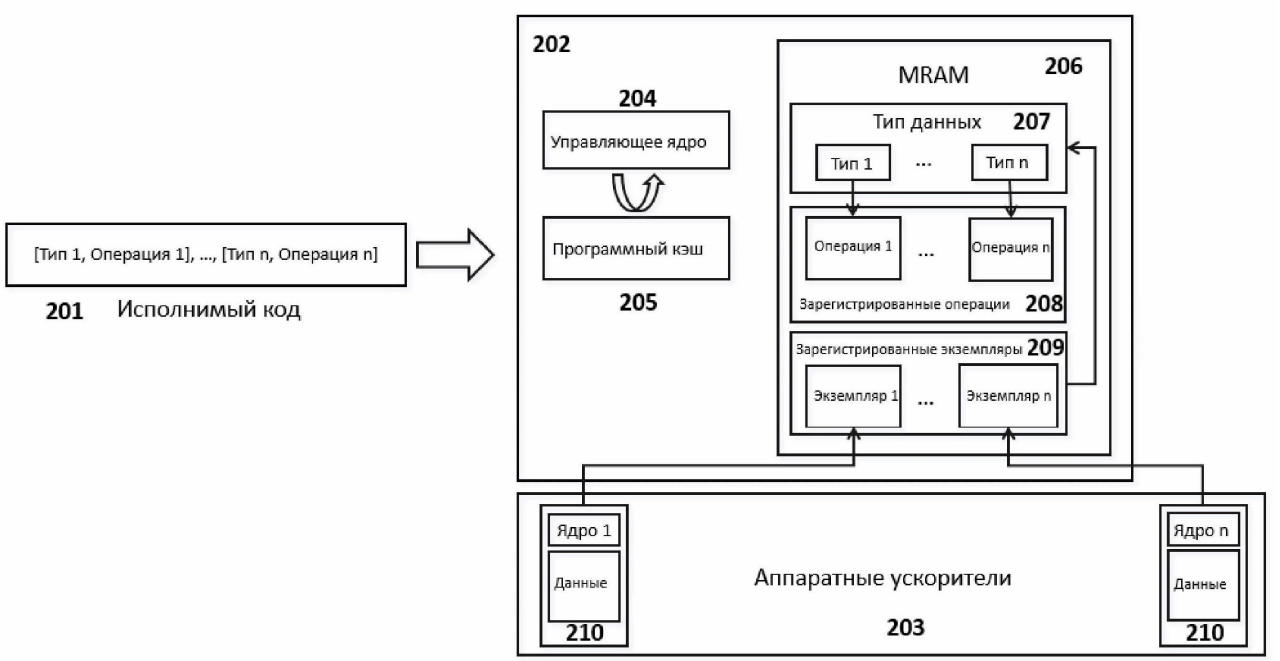
(54) СПОСОБ И СИСТЕМА ОБРАБОТКИ ДАННЫХ

(57) Реферат:

Изобретение относится к вычислительной технике. Технический результат заключается в повышении производительности и скорости обработки данных в компьютерной системе. Указанный результат достигается благодаря осуществлению способа обработки данных, в котором: производится инициализация дезагрегированного процессора, в результате которой: управляющее ядро осуществляет проверку состояния персистентной памяти на наличие метаданных; производится инициализация аппаратных ускорителей и опрос ускорителей для предоставления списка доступных операций; осуществляется поиск всех доступных аппаратных ускорителей, в результате которого: производится проверка присутствия экземпляров аппаратных ускорителей, которые уже зарегистрированы в таблице экземпляров; производится удаление из таблицы экземпляров аппаратных ускорителей, которые не были обнаружены; и осуществляется запрос операций, поддерживаемых каждым новым найденным аппаратным ускорителем; осуществляют регистрацию кода запрошенных операций в таблице зарегистрированных операций, типа ассоциированных данных в таблице типов данных и идентификатора экземпляра аппаратного ускорителя в таблице зарегистрированных экземпляров аппаратных ускорителей с помощью управляющего ядра; и выполняют обработку данных. 2 н. и 12 з.п. ф-лы, 3 ил.

RU 2829301 C1

RU 2829301 C1



Фиг. 1

RU 2829301 C1

RU 2829301 C1

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY

(19) **RU** (11) **2 829 301**(13) **C1**

(51) Int. Cl.
G06F 17/40 (2006.01)
G06F 9/50 (2006.01)

(12) ABSTRACT OF INVENTION

(52) CPC
G06F 17/40 (2024.08); G06F 9/50 (2024.08)

(21)(22) Application: **2024117711, 26.06.2024**

(24) Effective date for property rights:
26.06.2024

Registration date:
30.10.2024

Priority:
(22) Date of filing: **26.06.2024**

(45) Date of publication: **30.10.2024** Bull. № **31**

Mail address:
121059, Moskva, vn. ter. g. Munitsipalnyj Okrug Dorogomilovo, ul. Kievskaya, 7, korp. 2, etazh 11, KOMNATA 20/R3, AO "Sofit"

(72) Inventor(s):
Dubeiko Viacheslav Anatolevich (RU)

(73) Proprietor(s):
AKTsIONERNOE OBShchESTVO "SOFIT" (RU)

(54) DATA PROCESSING METHOD AND SYSTEM

(57) Abstract:

FIELD: computer engineering.

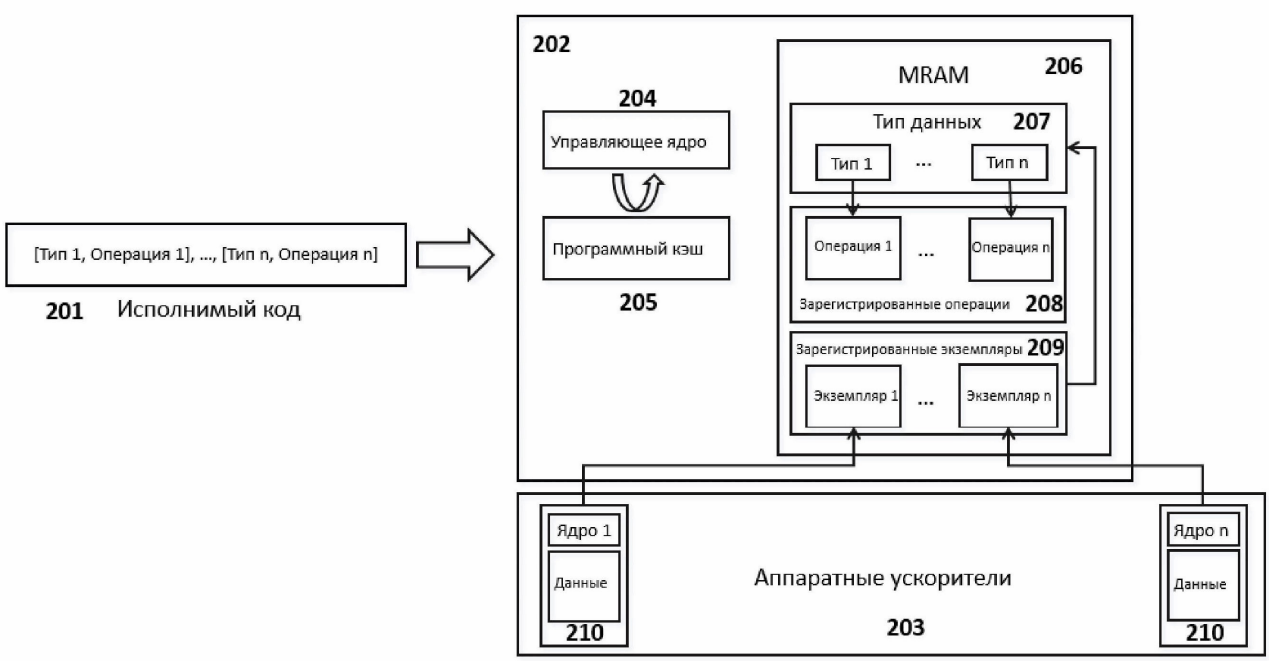
SUBSTANCE: result is achieved due to implementation of data processing method, in which: disaggregated processor is initialized, as a result of which: control core checks the state of persistent memory for presence of metadata; hardware accelerators are initialized and accelerators are polled to provide a list of available operations; searching for all available hardware accelerators, as a result of which: checking the presence of instances of hardware accelerators, which are already registered in the instance table; hardware accelerator instances that were not detected are deleted from the table; and performing a request for operations supported by each newly found hardware accelerator; performing the requested operations code registration in the registered operations table, the associated data type in the data types table and the hardware accelerator instance identifier in the hardware accelerator registered instances table using the control core; and performing data processing.

EFFECT: high efficiency and speed of data processing in a computer system.

14 cl, 3 dwg

RU 2829301 C1

RU 2829301 C1



Фиг. 1

RU 2829301 C1

RU 2829301 C1

ОБЛАСТЬ ТЕХНИКИ

[1] Заявленное техническое решение в общем относится к области вычислительной техники, а в частности к способу и системе обработки данных.

УРОВЕНЬ ТЕХНИКИ

[2] Современная вычислительная парадигма базируется на парадигме машины Тьюринга и архитектуре фон Неймана. Множество исследований показали, что широко используемая вычислительная парадигма достигла своих пределов совершенствования и обладает большим количеством недостатков, не позволяющих существенно увеличить производительность вычислений. В качестве таких ключевых недостатков можно упомянуть cache coherence problem (проблема консистентности состояния данных в основной памяти и кешах центрального процессора), memory wall problem (проблема опережения производительности вычислительного ядра процессора пропускной способности шины взаимодействия с памятью (DRAM)), data moving problem (проблема копирования или обмена данными между персистентной памятью, основной памятью (DRAM) системы и кешами процессора), power consumption problem (проблема избыточного потребления энергии), throughput bottleneck (проблема ограниченной пропускной способности интерфейсов взаимодействия), Big Data problem (или проблема обработки больших данных).

[3] Современная вычислительная парадигма является алгоритмо-ориентированной и задействует несколько типов памяти (регистры CPU, DRAM, персистентная память). Исполнимый код и пользовательские данные хранятся в персистентной памяти, которая может быть представлена HDD/SSD или другими технологиями или устройствами перситентного хранения данных. Однако, ядро процессора не способно напрямую обращаться и работать с данными в персистентной памяти, поскольку этот тип памяти не является байт-адресуемым (за исключением некоторых типов памяти, например, NOR флеш или типов памяти, поддерживающих XiP (eXecute-in-Place) механизм). Поэтому любые данные (а также исполнительный код) прежде всего должны быть скопированы в DRAM, а в случае их модификации быть сохранены обратно в персистентную память.

[4] Такой подход был достаточно эффективен в момент создания парадигмы, но в настоящее время такой подход вносит существенные накладные расходы (или понижение производительности вычислений) в окружении быстрой персистентной памяти и современных многоядерных процессоров. Более того, емкость (или объемы хранимых данных) современных HDD/SSD на порядки больше чем общий размер DRAM в одном экземпляре вычислительной системы, объемы обрабатываемых данных экспоненциально растут, что приводит к большим объемам обмена данными между персистентной памятью и DRAM, в результате это приводит к деградации производительности обработки данных в многопоточном окружении, исполняемом на многоядерных процессорах.

[5] Более того, DRAM гораздо медленнее, чем ядро процессора, поэтому исполнимый код и обрабатываемые данные приходится копировать в L1/L2/L3 кеш процессора для непосредственного исполнения и обработки. В условиях многопоточного окружения процессору приходится переключать контекст между задачами, что приводит к деградации производительности исполнения.

[6] Современные процессоры являются многоядерными и у каждого ядра есть свой кеш, а также кеш разделяемый несколькими ядрами процессора. Если какие-либо данные в кеше данных определенного ядра были модифицированы, то приходится использовать специальный cache coherence protocol с целью синхронизировать состояние данных в

DRAM и кешах конкретных ядер процессора. Что в итоге приводит к существенной деградации производительности для большинства современных типов алгоритмов и вычислительных нагрузок на компьютерную систему.

[7] Современная вычислительная парадигма обладает рядом особенностей, которые в результате развития элементной базы и компьютерной архитектуры привели к целому ряду проблем, которые делают невозможным существенное улучшение производительности современных компьютерных систем. Все это требует предложения альтернативных решений, способных решить проблему существенного улучшения производительности компьютерных систем.

[8] Современные системы хранения данных представляют собой комплексную архитектуру, включающую множество распределенных и взаимодействующих программных компонентов и большой пул устройств персистентной памяти, представляющих собой первичное место хранения Big Data. Такие объемы данных (петабайты или экзабайты) не могут быть обработаны в разумное время одной даже достаточно мощной компьютерной системой, в силу ограниченного объема DRAM и вышеуказанных причин деградации производительности в рамках одной системы. Поэтому обычно большие объемы данных (или/и распределенных данных) обрабатываются множеством компьютерных систем одновременно посредством разделения массива данных на порции и распределения таких порций между системами. К примеру, такая парадигма используется в рамках MapReduce подхода.

[9] Однако, в рамках одной конкретной компьютерной системы обработка порции данных все-равно страдает от вышеописанных проблем, приводящих к общей деградации производительности и невозможности существенного улучшения производительности обработки данных. То есть, cache coherence problem, memory wall problem, data moving problem, throughput bottleneck играют существенную роль в рамках одной системы. Более того, распределённая обработка данных добавляет дополнительные причины деградации производительности (ограничение пропускной способности сети, синхронизация доступа разных систем к разделяемым порциям данных, и тому подобное).

[10] Однако, можно отметить одну важную деталь. Любые данные хранятся в персистентной памяти и если мы хотим вести обработку данных, то любые данные должны быть скопированы в DRAM системы, после чего они становятся доступны ядрам CPU для обработки. Но гораздо более эффективной может быть data-centric модель, когда данные могут обрабатываться непосредственно в месте хранения данных, либо рядом с персистентной памятью посредством аппаратных ускорителей, например. С одной стороны, нет необходимости копировать данные в DRAM хоста, с другой стороны, такой подход обеспечивает глубокое распараллеливание потоков обработки массива данных.

[11] Такой подход может быть проиллюстрирован на основе клиент-серверной архитектуры системы управления базой данных. В данной архитектуре клиенты посылают SQL запросы на сторону сервера, который обладает знанием где сохранены данные и выполняет запросы клиентов посредством извлечения или обновления данных, которые удовлетворяют требованиям запросов клиентов. Классический подход предполагает, что сервер должен использовать доступную DRAM для извлечения данных из персистентной памяти, провести обработку данных в DRAM на основе полученных запросов и переслать результаты обработки данных на сторону клиента или сохранить модифицированные данные в персистентной памяти. Классический подход обладает всеми перечисленными выше недостатками, которые существенно

понижают производительность операций с данными.

[12] Следовательно, недостатком известных решений в данной области техники является отсутствие возможности существенно увеличить производительность компьютерной системы.

РАСКРЫТИЕ ИЗОБРЕТЕНИЯ

[13] В заявленном техническом решении предлагается новый подход к выполнению обработки данных в компьютерной системе.

[14] Таким образом, решается техническая проблема отсутствия возможности повышения производительности компьютерной системы.

[15] Техническим результатом, достигающимся при решении данной проблемы, является повышение производительности и скорости обработки данных в компьютерной системе.

[16] Указанный технический результат достигается благодаря осуществлению способа обработки данных, содержащего этапы, на которых:

- производится инициализация дезагрегированного процессора, посредством инициализации и перевода управляющего ядра в состояние готовности исполнения кода и к готовности добавления новых команд, в результате которой:
  - управляющее ядро осуществляет проверку состояния персистентной памяти на наличие метаданных, причем:
    - если персистентная память не содержит метаданных или содержит случайный набор данных, то управляющее ядро инициализирует персистентную память посредством заполнения ее нулями и подготовки исходного состояния метаданных;
    - если консистентные метаданные были обнаружены в персистентной памяти, то дезагрегированный процессор инициализируется;
- производится инициализация аппаратных ускорителей и опрос каждого экземпляра ускорителей для предоставления списка доступных операций;
- осуществляется поиск, с помощью управляющего ядра, всех доступных аппаратных ускорителей, в результате которого:
  - производится проверка присутствия экземпляров аппаратных ускорителей, которые уже зарегистрированы в таблице экземпляров;
  - производится удаление из таблицы экземпляров аппаратных ускорителей, которые не были обнаружены; и
  - осуществляется запрос операций, поддерживаемых каждым новым найденным аппаратным ускорителем;
- осуществляют регистрацию кода запрошенных операций в таблице зарегистрированных операций, типа ассоциированных данных в таблице типов данных и идентификатора экземпляра аппаратного ускорителя в таблице зарегистрированных экземпляров аппаратных ускорителей с помощью управляющего ядра; и
- выполняют обработку данных за счет исполнения кода с помощью аппаратных ускорителей.

[17] В одном из частных вариантов реализации способа управляющее ядро представляет собой конечный автомат.

[18] В другом частном варианте реализации способа осуществление поиска дополнительно содержит этап, на котором осуществляется опрос управляющим ядром новых аппаратных ускорителей и регистрация их в таблице экземпляров.

[19] В другом частном варианте реализации способа персистентная память представляет собой память MRAM.

[20] В другом частном варинте реализации способа персистентная память

представляет собой память ReRAM.

[21] В другом частном варианте реализации способа персистентная память представляет собой память FeRAM.

[22] В другом частном варианте реализации способа персистентная память представляет собой память PRAM.

[23] Кроме того, заявленный технический результат достигается за счет системы обработки данных, содержащей:

- дезагрегированный процессор, содержащий:

○ персистентную память, выполненную с возможностью хранения метаданных и таблицу экземпляров аппаратных ускорителей, таблицу кодов операций, таблицу типов данных;

○ управляющее ядро, выполненное с возможностью

- проверки состояния персистентной памяти на наличие метаданных, причем:

○ если персистентная память не содержит метаданных или содержит случайный набор данных, то управляющее ядро инициализирует персистентную память посредством заполнения ее нулями и подготовки исходного состояния метаданных;

○ если консистентные метаданные были обнаружены в персистентной памяти, то дезагрегированный процессор инициализируется;

- регистрации кода поддерживаемых ускорителем операций в таблице зарегистрированных операций, типа ассоциированных данных в таблице типов данных и идентификатора экземпляра аппаратного ускорителя в таблице зарегистрированных экземпляров аппаратных ускорителей;

○ программный кэш, выполненный с возможностью временного хранения программного кода и команд на время их исполнения аппаратными ускорителями;

- аппаратные ускорители, выполненные с возможностью обработки данных.

[24] В одном из частных вариантов реализации системы персистентная память представляет собой память MRAM.

[25] В другом частном варинте реализации системы персистентная память представляет собой память ReRAM.

[26] В другом частном варианте реализации системы персистентная память представляет собой память FeRAM.

[27] В другом частном варианте реализации системы персистентная память представляет собой память PRAM.

[28] В другом частном варианте реализации системы программный кэш представляет собой память SRAM.

[29] В другом частном варианте реализации системы программный кэш представляет собой память eDRAM.

КРАТКОЕ ОПИСАНИЕ ЧЕРТЕЖЕЙ

[30] Фиг. 1 иллюстрирует архитектуру дезагрегированного процессора.

[31] Фиг. 2 иллюстрирует пример реализации системы обработки данных базирующуюся на архитектуре дезагрегированного процессора.

[32] Фиг. 3 иллюстрирует блок-схему заявленного способа.

ОСУЩЕСТВЛЕНИЕ ИЗОБРЕТЕНИЯ

[33] Ниже будут описаны понятия и термины, необходимые для понимания данного технического решения.

[34] Конечный автомат (КА) - математическая абстракция, модель дискретного устройства, имеющего один вход, один выход и в каждый момент времени находящегося в одном состоянии из множества возможных. Является частным случаем абстрактного

дискретного автомата, число возможных внутренних состояний которого конечно.

[35] При работе на вход КА последовательно поступают входные воздействия, а на выходе КА формирует выходные сигналы. Обычно под входными воздействиями принимают подачу на вход автомата символов одного алфавита, а на выход КА в процессе работы выдаёт символы в общем случае другого, возможно даже не пересекающегося со входным, алфавита.

[36] Cache coherence problem - Проблема согласованности кэша возникает, когда несколько кэшей хранят копии одних и тех же данных в основной памяти системы, и изменения, сделанные в одном кэше, необходимо распространить на состоянии данных в основной памяти системы и на другие кэши. Несоблюдение согласованности кэша может привести к повреждению данных и некорректному поведению программы.

[37] Memory wall problem – проблема опережения производительности процессора пропускной способности памяти.

[38] Data moving problem – проблема копирования или обмена данными между персистентной памятью, основной памятью (DRAM) системы и кешами процессора.

[39] Power consumption problem – проблема роста энергопотребления при увеличении вычислительных мощностей.

[40] Throughput bottleneck – проблема ограниченной пропускной способности интерфейсов взаимодействия.

[41] Big Data problem – проблема обработки больших данных в рамках современных баз данных и систем хранения данных.

[42] Hardware ускоритель – аппаратный ускоритель.

[43] В данном решении под системой подразумевается компьютерная система, ЭВМ (электронно-вычислительная машина), ЧПУ (числовое программное управление), ПЛК (программируемый логический контроллер), компьютеризированные системы управления и любые другие устройства, способные выполнять заданную, чётко определённую последовательность вычислительных операций (действий, инструкций).

[44] Под устройством обработки команд подразумевается электронный блок либо интегральная схема (микропроцессор), исполняющая машинные инструкции (программы).

[45] Устройство обработки команд считывает и выполняет машинные инструкции (программы) с одного или более устройства хранения данных, например, таких устройств, как оперативно запоминающие устройства (ОЗУ) и/или постоянные запоминающие устройства (ПЗУ). В качестве ПЗУ могут выступать, но, не ограничиваясь, жесткие диски (HDD), флеш-память, твердотельные накопители (SSD), оптические носители данных (CD, DVD, BD, MD и т.п.) и др.

[46] Программа - последовательность инструкций, предназначенных для исполнения устройством управления вычислительной машины или устройством обработки команд.

[47] Дезагрегированная архитектура давно уже стала довольно популярной парадигмой эффективного использования ресурсов в рамках дата-центров. Идея состоит в том, чтобы на основе высокоскоростных коммуникаций агрегировать доступные ресурсы (CPU, memory, storage device) по мере необходимости в них для конкретных вычислений и работы с данными. Такая парадигма легко позволяет выделять, освобождать и гибко переконфигурировать имеющиеся ресурсы с целью эффективно проводить вычисления и работу с данными.

[48] Предлагаемое изобретение предназначено для решения вышеописанных проблем (cache coherence problem, memory wall problem, data moving problem, throughput bottleneck) посредством дезагрегированной архитектуры процессора, способной вести обработку

данных посредством аппаратных ускорителей в месте хранения данных. Суть идеи состоит в том, что все пространство персистентной памяти разбивается на фрагменты и каждый фрагмент имеет индивидуальный аппаратный ускоритель, способный исполнить один или несколько вариантов преобразования данных.

[49] Функционал каждого аппаратного ускорителя (или набор поддерживаемых команд) регистрируется в таблице доступных команд дезагрегированного процессора и связывается с определенным типом данных. После этого дезагрегированный процессор способен координировать работу массива аппаратных ускорителей посредством параллельной отправки команд аппаратным ускорителям, зарегистрированных для определенного типа данных. Основными достоинствами предлагаемой архитектуры являются:

1) Система команд дезагрегированного процессора может легко изменяться без необходимости каких-либо изменений в аппаратной архитектуре (современные CPU обладают фиксированной системой команд).

2) Дезагрегированный процессор не исполняет команды, а всего лишь доставляет их к месту хранения данных для исполнения посредством аппаратных ускорителей (можно использовать CXL протокол для доставки команд, к примеру), что устраняет такие проблемы как coherence problem, memory wall problem, data moving problem, throughput bottleneck посредством обработки данных непосредственно в месте хранения.

3) Обработка данных может происходить параллельно и одновременно большим количеством аппаратных ускорителей, что может существенно увеличивать производительность обработки данных.

4) Процессору не обязательно знать местонахождение данных, обработка данных может осуществляться посредством обобщенных алгоритмов, поскольку команды на обработку могут применяться к определенным типам данных и доставляться параллельно всем аппаратным ускорителям, обладающим данными определенного типа.

5) Дезагрегированная архитектура используется для гибкого синтеза архитектуры процессора прямо в процессе исполнения.

6) Предлагается новая модель представления данных, которая может быть эффективно использована для классификации данных и глубоко параллельной обработки.

[50] Современные процессоры имеют фиксированную систему команд. Это означает, что современный CPU представляет собой фиксированный набор функциональных блоков, каждый из которых задействуется на исполнение если машинный код исполняемого приложения содержит код операции, выбирающий соответствующий функциональный блок.

[51] Это означает что как код операции, так и обрабатываемые данные должны быть доставлены в кэш процессора для исполнения и обработки. Набор функциональных блоков не может быть расширен или изменен без изменения схемотехнической схемы процессора. То есть, это означает что для разработки и производства нового процессора с измененной или расширенной системой команд придется потратить значительные ресурсы.

[52] Более того, именно такая архитектура является источником проблем (cache coherence problem, memory wall problem, data moving problem, throughput bottleneck) не позволяющим существенно улучшить производительность компьютерных систем.

[53] Можно представить, что CPU может исходно не обладать системой команд или иметь исходно пустую таблицу инструкций. То есть, процессор может иметь таблицу

инструкций, в которой можно зарегистрировать некоторый функционал, логику или набор команд, реализуемых посредством аппаратного ускорителя или некоторого ядра общего назначения (способного исполнить программу или микропрограмму). Таблица представляет собой гибкий механизм формирования и изменения набора инструкций или функционала процессора посредством регистрации/дерегистрации функциональных блоков, представленных в виде аппаратных ускорителей в дезагрегированной архитектуре системы.

[54] Точнее такая таблица может быть представлена в виде трех компонентов: (1) тип данных, (2) список инструкций, (3) список экземпляров зарегистрированных функциональных блоков, аппаратных ускорителей или программных реализаций некоторого функционала. Список экземпляров нужен для ассоциации инструкции с экземпляром функционала и доставки задания или инструкции для исполнения этим функционалом. Список инструкций представляет собой доступную систему команд или функционала дезагрегированного процессора. А список типов данных ассоциирует конкретную инструкцию с конкретным типом данных.

[55] Любая команда или инструкция процессора применима к определенному типу данных. То есть, вполне возможно агрегировать определенный тип данных и хранить данные одного типа в рамках одного контейнера или другого подобного представления. С данными определенного типа может быть ассоциирован аппаратный ускоритель или другая «железная» логика, способная обрабатывать данные или применять одну операцию или набор операций к данным определенного типа.

[56] Таким образом, процессор, получая некоторую инструкцию, способен найти в таблице зарегистрированных экземпляров всех получателей, способных исполнить подобную инструкцию и применить операцию обработки данных ко всем элементам данных, принадлежащих определённому контейнеру данных определенного типа. То есть, получив инструкцию, процессор находит все экземпляры аппаратных ускорителей, ассоциированных с конкретным типом данных и рассылает им задание на проведение обработки данных.

[57] Исполнение может протекать в асинхронной или синхронной манере. Дезагрегированный процессор может дожидаться окончания исполнения обработки данных на всех задействованных аппаратных ускорителях перед отсылкой следующей инструкции (синхронный режим), либо он может отправлять группу операций и дожидаться окончания исполнения группы операций (ассинхронный режим).

[58] Одной из особенностей предлагаемого решения является способность одной инструкции быть одновременно доставленной и быть исполненной множеством аппаратных ускорителей, ассоциированных с определенным типом данных. То есть, в предлагаемом подходе параллельная обработка данных является частью парадигмы и не требует специальных усилий по разработке программного кода.

[59] Процессор сам осуществляет масштабирование исполнения некоторой логики или функционала на все множество зарегистрированных аппаратных ускорителей, работающих с данными определенного типа. То есть, любая логика может быть реализована в алгоритме общего назначения, которая для каждой инструкции может быть динамически масштабирована дезагрегированным процессором на некоторое множество исполнительных ядер без необходимости изменять логику или код программы под некоторое окружение или многопоточное исполнение. С одной стороны, это может существенно упростить разработку логики приложений, а с другой стороны эффективность и производительность исполнения может эффективно масштабироваться на все множество аппаратных ускорителей.

[60] Логика работы дезагрегированного процессора может выглядеть как обработка последовательности инструкций, также как это работает для современных CPU. Разница состоит в том, что в случае дезагрегированного процессора инструкция представляет собой зарегистрированную логику аппаратного ускорителя (которая может представлять довольно сложную и легко переконфигурируемую логику), которая исполняется аппаратным ускорителем, а не самим процессором.

[61] Как показано на Фиг. 1 реализация дезагрегированного процессора может базироваться на MRAM памяти (206) или любой другой байт адресуемой персистентной памяти, то есть, MRAM память будет служить тем пространством, в котором будут регистрироваться функциональные блоки (аппаратные ускорители (203)), которые и будут составлять доступный набор операций процессора.

[62] Фактически, необходимо иметь: (1) пространство для регистрации конкретных экземпляров функциональных блоков (209) (или аппаратных ускорителей); (2) пространство для регистрации кодов операции (208), которые ассоциируют зарегистрированные экземпляры функциональных блоков с идентификационным кодом операции; (3) таблица типов данных (207).

[63] Каждый код операции может быть применен к определенному типу данных, поскольку каждый аппаратный ускоритель обладает некоторым набором данных определенного типа или класса (210). То есть, либо код операции подразумевает некоторый тип данных, либо бинарный код может содержать информацию к какому типу данных нужно применить операцию.

[64] Дезагрегированный процессор (202) может обладать специальным управляющим ядром (204) и программным кэшем (205). Задачей управляющего ядра является обслуживание запросов по регистрации функциональных блоков или непосредственно обнаружения и регистрации функциональных блоков, тем самым формируя систему команд процессора.

[65] Программный кэш необходим для временного хранения бинарного кода программы (201), которую нужно исполнить посредством делегирования команд аппаратным ускорителям управляющим ядром дезагрегированного процессора. В момент старта компьютерной системы происходит обнаружение всех аппаратных ускорителей или функциональных блоков, присутствующих в системе, и их последующая регистрация. Технически, операция регистрации может производиться самим дезагрегированным процессором, аппаратные ускорители могут обнаруживать дезагрегированный процессор и регистрировать себя сами, либо firmware логика или логика некоторого приложения может осуществлять обнаружение, распределение между дезагрегированными процессорами и регистрацию аппаратных ускорителей.

[66] После того как функциональные блоки зарегистрированы в дезагрегированном процессоре, то он способен исполнять обработку данных на множестве аппаратных ускорителей. MRAM представляет собой персистентную память и это означает что система команд дезагрегированного процессора будет существовать даже после осуществления рестарта системы.

[67] Для изменения системы команд процессора необходимо сначала исключить или удалить некоторую зарегистрированную операцию и зарегистрировать другую. Поскольку MRAM память обладает хорошей долговечностью, то операции изменения системы команд можно повторять многократно.

[68] Система команд дезагрегированного процессора может включать несколько типов операций:

(A) Результат операции с данными сохраняется в том же самом массиве данных.

(B) Операция производится с двумя массивами данных и результат может сохраняться в первый или второй массив данных.

(C) Операция производится с двумя массивами данных и результат сохраняется в третий массив данных.

[69] Обычно приложение ожидает фиксированную систему команд процессора, и компилятор преобразовывает логику приложения с языка высокого уровня на машинный язык. Дезагрегированный процессор не обладает фиксированной системой команд и потому нуждается в специализированном подходе компиляции приложения и специализированном представлении хранения данных.

[70] Можно предложить подход, при котором логика обычного приложения может быть проанализирована классическим компилятором или методами машинного обучения с целью выделения примитивов алгоритмов, которые могут быть синтезированы FPGA логикой в аппаратных ускорителях. Таким образом, примитивы алгоритмов будут реализованы в аппаратных ускорителях, а логика обычного приложения будет преобразована в логику приложения, способного быть исполненным на дезагрегированном процессоре.

[71] Другой подход может заключаться в том, что может быть сформирован массив аппаратных ускорителей с известным набором типов данных и набором функций, которые могут быть исполнены на аппаратных ускорителях. И логика приложения может быть написана программистом или нейронной сетью в виде обобщенного алгоритма, который может быть исполнен дезагрегированным процессором.

[72] На Фиг. 2 представлено описание потока исполнения в рамках архитектуры дезагрегированного процессора. Архитектура включает дезагрегированный процессор (301), пул ядер аппаратных ускорителей (308), пул памяти для хранения данных (302). Для реализации своей функциональности дезагрегированный процессор содержит таблицу зарегистрированных команд (303) и таблицу зарегистрированных экземпляров ядер аппаратных ускорителей (304).

[73] Таблицы (303, 304) могут храниться в памяти персистентного типа, позволяющую обращаться к хранимым данным с различной гранулярностью (которая не может быть меньше одного байта), способную выдержать многочисленные циклы перезаписи без деградации основных физических характеристик памяти и гарантировать короткое время обращения или записи (не хуже чем DRAM). К примеру, MRAM может такой памятью, способной реализовать функционал таблиц в архитектуре дезагрегированного процессора. Если не требуется сохранять систему команд в постоянной памяти, то таблицы могут представлены SRAM памятью, а система команд дезагрегированного процессора может формироваться при каждом старте и инициализации системы.

[74] Пул памяти (302) может быть представлена как массивом DRAM памяти (памятью временного хранения), так и быстрой персистентной памятью на базе SSD носителей памяти, или гибридной системой хранения данных, включающей в себя комбинацию HDD, SSD, DRAM памяти. Этот пул памяти может быть массивом памяти, разделяемой пулом аппаратных ускорителей (308), либо каждый аппаратный ускоритель имеет некоторый объем памяти (представленный посредством DRAM, SSD) на своей плате расширения, которые объединяются в агрегированное пространство памяти (302).

[75] Каждый аппаратный ускоритель может быть представлен PCIe платой расширения или в любом другом форм-факторе, использующий другой интерфейс взаимодействия (например U.2, CXL). Таким образом, пул аппаратных ускорителей (308) может быть представлен некоторым набором плат расширения (например, PCIe плата расширения), каждая из которых может содержать одно или несколько

исполнительных ядер (306). Аппаратный ускоритель должен быть способен проинформировать запрашивающую систему о наборе поддерживаемых операций и поддерживать протокол регистрации поддерживаемых команд в таблице дезагрегированного процессора (301).

[76] Дезагрегированный процессор (301) может получить код операции из DRAM или программного кэша на основании содержимого специального регистра, хранящего указатель на текущий код операции, подлежащий исполнению. Прежде всего код операции должен быть найден в таблице зарегистрированных операций (303). Если код операции не найден в таблице, то это вызывает остановку исполнения программного кода и генерацию процессором исключения, которое должно быть обработано операционной системой. Если код операции найден в таблице (303), то необходимо найти экземпляры аппаратных ускорителей, способных исполнить операцию, в таблице зарегистрированных аппаратных ускорителей (304). Если таблица (304) не содержит идентификаторов ускорителей для текущего кода операции, то это вызывает остановку исполнения программного кода и генерацию процессором исключения, которое должно быть обработано операционной системой. Код операции доставляется каждому зарегистрированному экземпляру аппаратного ускорителя в пуле (305). Код операции ассоциирован с типом данных, поэтому каждый аппаратный ускоритель может выбрать ядро или ядра (306), ассоциированные с определенным типом данных. В конечном итоге, каждое ядро (306) осуществляет обработку данных (307) в рамках пула (305). По окончанию исполнения обработки данных каждое ядро (306) в рамках пула (305) информирует управляющее ядро дезагрегированного процессора (301) об окончании процесса обработки данных.

[77] Как показано на Фиг. 3 способ обработки данных (100) состоит из нескольких этапов.

[78] На этапе (101) производится инициализация дезагрегированного процессора, посредством инициализации и перевода управляющего ядра в состояние готовности регистрации новых команд и маршрутизации или доставки зарегистрированных команд на исполнение аппаратными ускорителями или другими реализациями функциональных блоков.

[79] На данном этапе управляющее ядро осуществляет проверку состояния персистентной памяти на наличие метаданных, причем:

- если персистентная память не содержит метаданных или содержит случайный набор данных, то управляющее ядро инициализирует персистентную память посредством заполнения ее нулями и подготовки исходного состояния метаданных;
- если консистентные метаданные были обнаружены в персистентной памяти, то дезагрегированный процессор заканчивает этап инициализации.

[80] В одном из вариантов реализации изобретения управляющее ядро представляет собой конечный автомат.

[81] В другом частном варианте реализации изобретения персистентная память представляет собой память MRAM.

[82] В другом частном варинте реализации способа персистентная память представляет собой память ReRAM.

[83] В другом частном варианте реализации способа персистентная память представляет собой память FeRAM.

[84] В другом частном варианте реализации способа персистентная память представляет собой память PRAM.

[85] Далее на этапе (102) производится инициализация аппаратных ускорителей,

проверка состояния функциональных блоков и формирование списка поддерживаемых команд и типов данных, которые могут быть зарегистрированы дезагрегированным процессором.

[86] В одном из вариантов реализации изобретения если управляющее ядро осуществляет регистрацию, то сперва происходит инициализация аппаратных ускорителей, а потом управляющее ядро обнаруживает и регистрирует эти ускорители посредством их опроса.

[87] В другом из вариантов реализации изобретения аппаратные ускорители могут обращаться к управляющему ядру после инициализации с целью регистрации.

[88] В другом из вариантов реализации изобретения инициализация управляющего ядра и аппаратных ускорителей происходит без регистрации, а исполнение некоторого программного кода осуществляется посредством широковещательной доставки команд программного кода всем имеющимся аппаратным ускорителям в системе, либо обнаружения аппаратных ускорителей управляющим ядром и доставкой команд управляющим ядром на исполнение обнаруженным аппаратным ускорителям.

[89] Далее на этапе (103) осуществляется поиск, с помощью управляющего ядра, всех доступных аппаратных ускорителей.

[90] На данном этапе производится проверка присутствия экземпляров аппаратных ускорителей, которые уже зарегистрированы в таблице экземпляров, производится удаление из таблицы экземпляров аппаратных ускорителей, которые не были обнаружены; и осуществляется запрос поддерживаемых операций, для каждого нового экземпляра найденного аппаратного ускорителя.

[91] В одном из вариантов реализации изобретения осуществление поиска дополнительно содержит этап, на котором осуществляется опрос управляющим ядром новых аппаратных ускорителей и регистрация их в таблице экземпляров.

[92] Далее на этапе (104) осуществляют регистрацию кодов операций, поддерживаемых аппаратным ускорителем, в таблице зарегистрированных операций, типа данных ассоциированных с кодом операции в таблице типов данных и идентификатора экземпляра аппаратного ускорителя в таблице зарегистрированных экземпляров аппаратных ускорителей с помощью управляющего ядра дезагрегированного процессора.

[93] И на этапе (105) выполняют обработку данных за счет исполнения кода с помощью аппаратных ускорителей.

[94] Заявленный способ выполнения обработки данных реализуется с помощью системы обработки данных, содержащей:

- дезагрегированный процессор, содержащий:

○ персистентную память, выполненную с возможностью хранения метаданных и таблицу экземпляров аппаратных ускорителей, таблицу кодов операций, таблицу типов данных;

○ управляющее ядро, выполненное с возможностью

- проверки состояния персистентной памяти на наличие метаданных, причем:

○ если персистентная память не содержит метаданных или содержит случайный набор данных, то управляющее ядро инициализирует персистентную память посредством заполнения ее нулями и подготовки исходного состояния метаданных;

○ если консистентные метаданные были обнаружены в персистентной памяти, то дезагрегированный процессор инициализируется;

- регистрации кода поддерживаемых ускорителем операций в таблице зарегистрированных операций, типа ассоциированных данных в таблице типов данных

и идентификатора экземпляра аппаратного ускорителя в таблице зарегистрированных экземпляров аппаратных ускорителей;

○ программный кэш, выполненный с возможностью временного хранения программного кода и команд на время их исполнения аппаратными ускорителями;

- аппаратные ускорители, выполненные с возможностью обработки данных.

[95] В одном из частных вариантов реализации системы персистентная память представляет собой память MRAM.

[96] В другом частном варинте реализации системы персистентная память представляет собой память ReRAM.

[97] В другом частном варианте реализации системы персистентная память представляет собой память FeRAM.

[98] В другом частном варианте реализации системы персистентная память представляет собой память PRAM.

[99] В другом частном варианте реализации системы программный кэш представляет собой память SRAM.

[100] В другом частном варианте реализации системы программный кэш представляет собой память eDRAM.

[101] Представленные материалы заявки раскрывают предпочтительные примеры реализации технического решения и не должны трактоваться как ограничивающие иные, частные примеры его воплощения, не выходящие за пределы испрашиваемой правовой охраны, которые являются очевидными для специалистов соответствующей области техники.

(57) Формула изобретения

1. Способ обработки данных, содержащий этапы, на которых:

- производится инициализация дезагрегированного процессора посредством инициализации и перевода управляющего ядра в состояние готовности исполнения кода и к готовности добавления новых команд, в результате которой:

○ управляющее ядро осуществляет проверку состояния персистентной памяти на наличие метаданных, причем:

▪ если персистентная память не содержит метаданных или содержит случайный набор данных, то управляющее ядро инициализирует персистентную память посредством заполнения ее нулями и подготовки исходного состояния метаданных;

▪ если консистентные метаданные были обнаружены в персистентной памяти, то дезагрегированный процессор инициализируется;

- производится инициализация аппаратных ускорителей и опрос каждого экземпляра ускорителей для предоставления списка доступных операций;

- осуществляется поиск, с помощью управляющего ядра, всех доступных аппаратных ускорителей, в результате которого:

○ производится проверка присутствия экземпляров аппаратных ускорителей, которые уже зарегистрированы в таблице экземпляров;

○ производится удаление из таблицы экземпляров аппаратных ускорителей, которые не были обнаружены; и

○ осуществляется запрос операций, поддерживаемых каждым новым найденным аппаратным ускорителем;

- осуществляют регистрацию кода запрошенных операций в таблице зарегистрированных операций, типа ассоциированных данных в таблице типов данных и идентификатора экземпляра аппаратного ускорителя в таблице зарегистрированных

экземпляров аппаратных ускорителей с помощью управляющего ядра; и

- выполняют обработку данных за счет исполнения кода с помощью аппаратных ускорителей.

2. Способ по п.1, характеризующийся тем, что управляющее ядро представляет собой конечный автомат.

3. Способ по п.1, характеризующийся тем, что осуществление поиска дополнительно содержит этап, на котором осуществляется опрос управляющим ядром новых аппаратных ускорителей и регистрация их в таблице экземпляров.

4. Способ по п.1, характеризующийся тем, что персистентная память представляет собой память MRAM.

5. Способ по п.1, характеризующийся тем, что персистентная память представляет собой память ReRAM.

6. Способ по п.1, характеризующийся тем, что персистентная память представляет собой память FeRAM.

7. Способ по п.1, характеризующийся тем, что персистентная память представляет собой память PRAM.

8. Система обработки данных, содержащая:

- дезагрегированный процессор, содержащий:

○ персистентную память, выполненную с возможностью хранения метаданных, и таблицу экземпляров аппаратных ускорителей, таблицу кодов операций, таблицу типов данных;

○ управляющее ядро, выполненное с возможностью

- проверки состояния персистентной памяти на наличие метаданных, причем:

○ если персистентная память не содержит метаданных или содержит случайный набор данных, то управляющее ядро инициализирует персистентную память посредством заполнения ее нулями и подготовки исходного состояния метаданных;

○ если консистентные метаданные были обнаружены в персистентной памяти, то дезагрегированный процессор инициализируется;

- регистрации кода поддерживаемых ускорителем операций в таблице зарегистрированных операций, типа ассоциированных данных в таблице типов данных и идентификатора экземпляра аппаратного ускорителя в таблице зарегистрированных экземпляров аппаратных ускорителей;

○ программный кэш, выполненный с возможностью временного хранения программного кода и команд на время их исполнения аппаратными ускорителями;

- аппаратные ускорители, выполненные с возможностью обработки данных.

9. Система по п.8, характеризующаяся тем, что персистентная память представляет собой память MRAM.

10. Система по п.8, характеризующаяся тем, что персистентная память представляет собой память ReRAM.

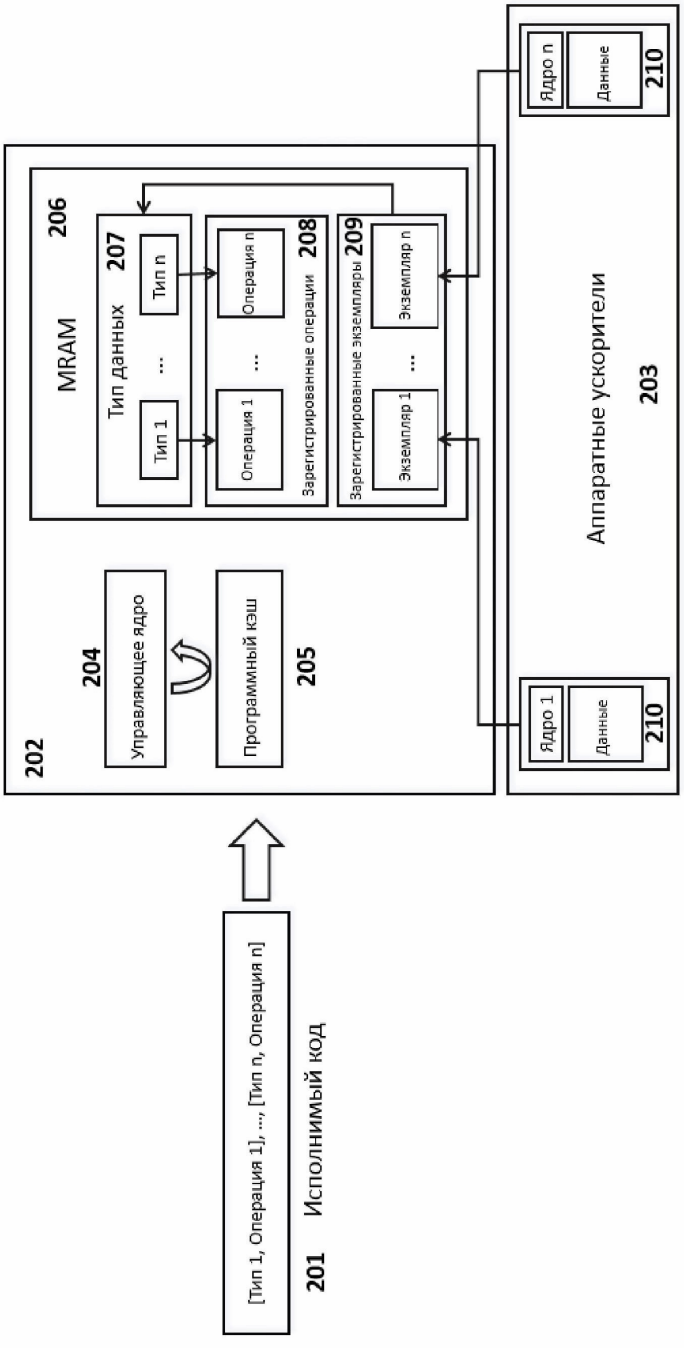
11. Система по п.8, характеризующаяся тем, что персистентная память представляет собой память FeRAM.

12. Система по п.8, характеризующаяся тем, что персистентная память представляет собой память PRAM.

13. Система по п.8, характеризующаяся тем, что программный кэш представляет собой память SRAM.

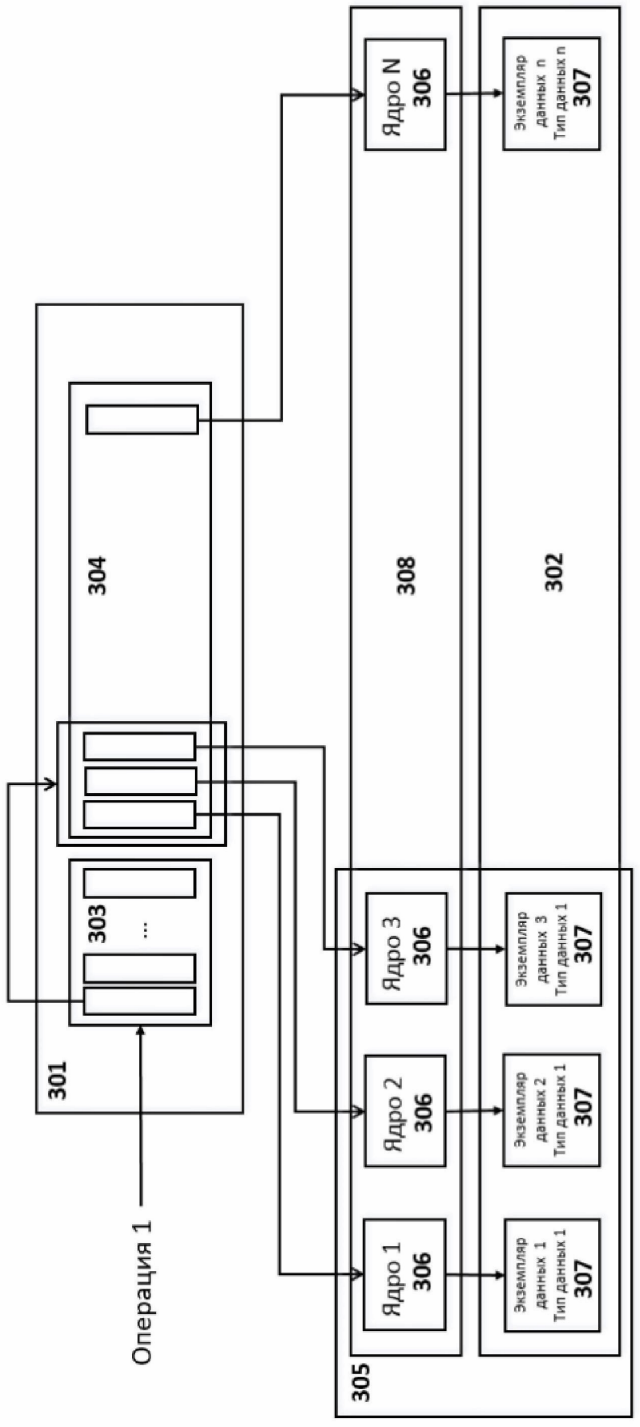
14. Система по п.8, характеризующаяся тем, что программный кэш представляет собой память eDRAM.

1


[Тип 1, Операция 1], ..., [Тип n, Операция n]
201 Исполнимый код
202
MRAM 206
Тип данных 207
Тип 1
...
Тип n
Операция 1
...
Операция n
Зарегистрированные операции 208
Экземпляр 1
...
Экземпляр n
Зарегистрированные экземпляры 209
204
Управляющее ядро
Программный кэш
205
Аппаратные ускорители
203
Ядро 1
Данные
210
Ядро n
Данные
210

Фиг. 1

2


Операция 1
301
303
...
304
305
308
302
Ядро 1
306
Ядро 2
306
Ядро 3
306
Ядро N
306
Экземпляр данных 1
Тип данных 1
307
Экземпляр данных 2
Тип данных 1
307
Экземпляр данных 3
Тип данных 1
307
Экземпляр данных n
Тип данных n
307

Фиг. 2

3


100
101
Инициализация дезагрегированного процессора.
102
Инициализация аппаратных ускорителей.
103
Поиск всех доступных аппаратных ускорителей.
104
Регистрация кода запрошенных операций.
105
Выполнение обработки данных.

Фиг .3