(19) Евразийское патентное ведомство

(11) 044373 (13) B1

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ЕВРАЗИЙСКОМУ ПАТЕНТУ

(45) Дата публикации и выдачи патента
2023.08.22

(21) Номер заявки
202191500

(22) Дата подачи заявки
2021.06.28

(51) Int. Cl. ***H04L 45/00*** (2022.01)
H04L 41/00 (2022.01)
H04L 43/00 (2022.01)

(54) СПОСОБ И СИСТЕМА ЦИКЛИЧЕСКОГО РАСПРЕДЕЛЕННОГО АСИНХРОННОГО ОБМЕНА СООБЩЕНИЯМИ СО СЛАБОЙ СИНХРОНИЗАЦИЕЙ ДЛЯ РАБОТЫ С БОЛЬШИМИ ГРАФАМИ

(31) **2021105737**
(32) **2021.03.05**
(33) **RU**
(43) **2022.09.30**

(71)(73) Заявитель и патентовладелец:
ПУБЛИЧНОЕ АКЦИОНЕРНОЕ ОБЩЕСТВО "СБЕРБАНК РОССИИ" (ПАО СБЕРБАНК) (RU)

(72) Изобретатель:
Выборнов Валерий Викторович (RU)

(74) Представитель:
Герасин Б.В. (RU)

(56) US-A1-20110228788
US-A1-20050169193
US-B2-10177985
US-B1-10601671

(57) Изобретение относится к области информационных технологий, в частности к способам передачи информации между узлами сети с помощью циклического распределенного асинхронного обмена сообщениями со слабой синхронизацией, для работы с большими графами. Технический результат заключается в повышении эффективности и скорости передачи сообщений между узлами сети, что способствует быстрому решению прикладных задач в системе управления большими графами (FastGraph). Заявленный результат достигается за счет компьютерно-реализуемого способа циклического распределенного асинхронного обмена сообщениями со слабой синхронизацией, выполняемого с помощью процессора и содержащего этапы, на которых: a) определяют множество точек сетевого обмена сообщениями, где каждая точка представляет собой вычислительный узел; b) формируют модель графа для обмена сообщениями между точками сетевого обмена, в котором вершинами являются точки сетевого обмена, а ребрами - факт отправки сообщений; c) определяют точку сетевого обмена - координатора исполнения обмена сообщениями; d) осуществляют отправку сообщений от точки координатора по меньшей мере одной точке сетевого обмена; e) получают ответ от точки сетевого обмена, получившей сообщение на этапе d), причем упомянутый ответ содержит идентификаторы смежных точек сетевого обмена, в которые будут направлены сообщения от данной точки сетевого обмена, при этом упомянутая точка сетевого обмена формирует маршруты последующего обмена сообщениями; f) на точке координаторе осуществляют учет сформированных маршрутов обмена сообщениями для каждой точки сетевого обмена графа, на основании сообщений от каждой точки сетевого обмена; g) итеративно повторяют этапы e)-f) до того момента, пока точка координатор не сообщит, что точки сетевого обмена отсутствуют.

044373 B1

044373 B1

Область техники

Настоящее техническое решение относится к области информационных технологий, а также к области управления большими графовыми данными, в частности, к способам передачи графовой информации между узлами сети с помощью циклического распределенного асинхронного обмена сообщениями со слабой синхронизацией.

Уровень техники

Из уровня техники известен подход передачи сообщений с помощью построения графовой модели узлов обмена, например, различные реализации модели Bulk Synchronous Parallel (BSP) (https://en.wikipedia.org/wiki/Bulk_synchronous_parallel), в частности, модель, реализованная в Apache Giraph (http://giraph.apache.org/intro.html). Данная модель основана на алгоритме поиска кратчайшего пути между узлами обмена, для чего выполняется определение расстояния между вершинами в графовой структуре.

Известными решениями для поиска кратчайшего пути является - поиск кратчайшего пути между двумя вершинами взвешенного графа (Взвешенным называется граф, ребра которого имеют веса). Наиболее эффективным алгоритмом поиска кратчайшего пути на графах с не отрицательными весами является алгоритм Dijkstra (https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm). Алгоритм требует последовательного поиска пути от наиболее ближайшей (на текущий момент) вершины в первую очередь. Т.е. если известно, что путь может проходить через обе вершины, то алгоритм продолжает поиск с ближайшей из двух вершин.

Алгоритм CMMS (критерий -OUT) позволяет выполнять поиск кратчайшего пути в параллельном режиме. Алгоритм по определенному критерию выбирает подмножество вершин, через которые может проходить кратчайший пути. Далее эти вершины могут обрабатываться параллельно (см. https://people.mpi-inf.mpg.de/~mehlhorn/ftp/ParallelizationDijkstra.pdf).

Недостатки известных решений заключаются в том, что в решении Giraph вершина в рамках супершага один раз принимает сообщения, и один раз отправляет. В случае применения массивно параллельных вычислительных систем, в ряде случаев, сильная система синхронизации препятствует полной загрузке всех процессоров и увеличивает время работы алгоритма.

Таким образом, в заявленном решении предлагается совершать множественные пересылки между вершинами за счет более слабой синхронизации.

Сущность изобретения

Решаемой технической проблемой с помощью заявленного изобретения является создание нового метода циклического распределенного асинхронного обмена сообщениями между узлами сети со слабой синхронизацией, для работы с большими графами.

Технический результат заключается в повышении эффективности и скорости передачи сообщений между узлами сети, что способствует быстрому решению прикладных задач в системе управления большими графами (FastGraph).

Заявленный результат достигается за счет компьютерно-реализуемого способа циклического распределенного асинхронного обмена сообщениями со слабой синхронизацией, выполняемого с помощью процессора и содержащего этапы, на которых:

a) определяют множество точек сетевого обмена сообщениями, где каждая точка представляет собой вычислительный узел;

b) формируют модель графа для обмена сообщениями между точками сетевого обмена, в котором вершинами являются точки сетевого обмена, а ребрами - факт отправки сообщений;

c) определяют точку сетевого обмена - координатора исполнения обмена сообщениями;

d) осуществляют отправку сообщений от точки координатора по меньшей мере одной точке сетевого обмена;

e) получают ответ от точки сетевого обмена, получившей сообщение на этапе d), причем упомянутый ответ содержит идентификаторы смежных точек сетевого обмена, в которые будут направлены сообщения от данной точки сетевого обмена, при этом упомянутая точка сетевого обмена формирует маршруты последующего обмена сообщениями;

f) на точке координаторе осуществляют учет сформированных маршрутов обмена сообщениями для каждой точки сетевого обмена графа, на основании сообщений от каждой точки сетевого обмена;

g) итеративно повторяют этапы e)-f) до того момента, пока точка координатор не сообщит, что точки сетевого обмена отсутствуют.

В одном из частных примеров выполнения способа каждая вершина содержит хранилище состояний, в котором содержится информация о признаке посещения.

Заявленный технический результат достигается также за счет системы циклического распределенного асинхронного обмена сообщениями со слабой синхронизацией, содержащей по меньшей мере один процессор и по меньшей мере одно средство памяти, которое хранит машиночитаемые инструкции, которые при их исполнении процессором реализуют вышеуказанный способ.

Краткое описание фигур

Фиг. 1 иллюстрирует общий вид конической модели обмена сообщениями (пример реализации)

Фиг. 2 иллюстрирует блок-схему выполнения заявленного способа.

Фиг. 3 иллюстрирует пример вычислительной системы.

Осуществление изобретения

На фиг. 1 представлен пример реализации конической модели (100) обмена сообщениями. Точка (110) представляет узел - центр конуса, точка (150) - вершина конуса, точки (101)-(106) - узлы обмена данными, располагаемые на окружности конуса. В рамках реализации заявленного решения узлы конической модели обмениваются заданными типами сообщений, представленными в табл. 1.

Таблица 1. Тип сообщений

Аббревиатура	Расшифровка	Направление	Значение
CRAM	Cone Reverse Axis Message	Вершина конуса → Центр конуса	Обратное осевое сообщение. Инициирует первый конус в семействе.
CAM	Cone Axis Message	Центр конуса → Вершина конуса	Осевое сообщение. Обеспечивает отслеживание периферийных точек конуса.
CBM	Cone Base Message	Центр конуса → Окружность конуса	Основное сообщение. Переносит данные между точками распределенной системы и, опционально, порождает новые конусы.
CSM	Cone Side Message	Окружность конуса → Вершина конуса	Боковое сообщение. Набор CSM завершает каждый конус.

Необходимо отметить, что данный пример показывает лишь частную реализацию в виде конической архитектуры, которая может быть представлена различными вариациями графовой модели с сохранением заявленного функционала и основных вычислительных узлов решения (табл. 2).

Таблица 2. Описание узлов

Название	Описание
Коническая модель обмена сообщениями	Способ циклического распределенного асинхронного обмена сообщениями со слабой синхронизацией на примере модели графа в виде конуса.
Вычислительный узел (ВУ)	Контейнер ресурсов, участвующий в обмене сообщениями. Каждый ВУ представляет собой фрагмент (shard), объединяющий подмножество вершин графа, вместе с логикой обработки сообщений. Сообщения внутри каждого ВУ обрабатываются синхронно и в однопоточном режиме.
Вершина (конуса)	Менеджер (управляющий узел) обмена сообщениями. Инициирует семейства конусов (супершаги) и обрабатывает результаты, полученные от точек.
Конус	Цикл обмена сообщениями, состоящий из последовательности (CRAM|CBM) → N*CBM + CAM → N*CSM. Может порождать от 0 до N дочерних конусов. Можно также использовать название "макрошаг"
Семейство конусов	Набор последовательных конусов, порожденных одним CAM. Аналогичен супершагу в модели BSP.
Центр конуса	Точка, являющаяся адресатом CRAM в процессе данного конуса. Порождает CBM. Понятие "локально" для конуса.
Окружность конуса	Набор точек, являющихся получателями CBM в процессе данного конуса.

Узел сети, расположенный в вершине конуса (150) назначается координатором. С вершиной конуса (150) не ассоциируются никакие вершины (101-106, 110) графа. В координаторе хранится информация, необходимая для выполнения обмена и работы (специфично для процесса).

Узел (110), расположенный в центре конуса, может посылать сообщение самому себе (т.е. центр конуса может быть на периферии). Каждое следующее семейство конусов в рамках одной вершины может быть инициировано только по окончании предыдущего. Детектировать окончание семейства конусов является функцией фреймворка. Это единственная точка синхронизации (барьер) в конической модели.

Может быть несколько вершин конуса, осуществляющих операции параллельно. Порождаемые ими семейства конусов независимы друг от друга. В то же время они используют одни и те же точки, и обработка сообщений внутри точки происходит последовательно и синхронно.

На каждом из этапов исполнения алгоритма коническая модель обеспечивает взаимодействие между вычислительными узлами и координатором (150), в частности, отправления запросов, получение ответов, синхронизацию исполнения через ожидание ответов от всех узлов.

На фиг. 2 представлена блок-схема предлагаемого способа (200) циклического распределенного асинхронного обмена сообщениями со слабой синхронизацией. На первом этапе (201) определяют множество точек сетевого обмена сообщениями, по которым на этапе (202) формируются модель графа для обмена сообщениями между точками сетевого обмена, в котором вершинами являются точки сетевого обмена, а ребрами - факт отправки сообщений.

На этапе (203) определяют точку сетевого обмена - координатора исполнения обмена сообщениями

(150), т.е. вершину конуса - Pv, и инициирующие данные алгоритма dv для данной точки. Инициирующий конус в семействе конусов с помощью точки Pv имеет отличия от следующих в том, что он инициируется при помощи сообщения CRAM, в то время как последующие - при помощи сообщений CBM.

На этапе (203) при выполнении инициализации выполняется создание координатора исполнения (150) и заданного количества вычислительных узлов - центров конусов для осуществления процесса обмена сообщениями. На данном этапе (203) также определяется алгоритм распределения (партиционирования) графа, предназначенный для формирования вычислительных узлов, поскольку каждый вычислительный узел отвечает за обработку части графа - порождает свое семейство конусов. Примером такого алгоритма может служить алгоритм вычисления хеш-функции идентификатора mod K (http://personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/hashTable.htm).

На узле-координаторе (150) функция алгоритма Av вычисляет узел - центр конуса (110) Pc и ассоциированные данные dvc, что составляет сообщение CRAM:

$$Av(dv) \rightarrow (Pc, dvc).$$

Следующим шагом (204) является отправка координатором (150) сообщения CRAM точке сетевого обмена - центру конуса (110). Подробное описание шага представлено в разделе с описанием примера реализации способа на примере алгоритма CMMS-OUT.

На шаге (205) происходит следующее - на узле (110) (центр конуса) функция алгоритма Ac вычисляет список вершин (101)-(106) на периферии Pbn и ассоциированные данные dcbn, что составляет набор CBM; также вычисляется CAM, в который включается список периферийных вершин (необходим для отслеживания завершения семейства конусов в узле-координаторе):

Ac(dvc, d(Pc)) → {

(Pb1, dcb1),

(Pb2, dcb2),

...,

(PbN, dcbN),

(Pv, (dcv, Pb1, Pb2, ..., PbN))

}.

На шаге (206) происходит отправка CAM с данными из предыдущего пункта от точки сетевого обмена (центра конуса (110)) узлу координатору (150). Подробное описание данного шага представлено в разделе с описанием примера реализации способа на примере алгоритма CMMS-OUT.

На этапе (207) на точке координаторе (150) осуществляют учет сформированных маршрутов обмена сообщениями для каждой точки сетевого обмена графа (101)-(106), (110) на основании получаемых сообщений от каждой точки сетевого обмена. Принцип обработки зависит от процесса. Один из возможных принципов - вести две очереди - одна со списком всех точек сетевого обмена, которые необходимо посетить, а вторая очередь - со списком посещенных вершин.

На шаге (208) происходит проверка - посещены ли все точки сетевого обмена, завершены ли все параллельные работы. Если да, то обмен завершается. Если нет, происходит новая итерация и шаги (205-207) повторяются. Далее рассмотрим пример реализации способа (200) на примере алгоритма CMMS-OUT.

Приведем пример реализации способа (200) на примере алгоритма CMMS-OUT (см. фиг. 1). Алгоритм CMMS-OUT в конической модели описывает алгоритм поиска кратчайших путей (в классическом варианте однонаправленное), позволяющий находить кратчайшие пути и расстояния между некоторой вершиной (источником) и всеми остальными вершинами в графе - dist(n).

Алгоритм итеративный. Вершины делятся на непосещенные Q, посещенные и текущие C. Каждой вершине сопоставляется динамически (итеративно) обновляемое временное расстояние tdist(n) - минимальное из известных на данный момент расстояний от источника до данной вершины (предварительное расстояние). В посещенных вершинах tdist(n)=dist(n).

На первом шаге (итерации) tdist(s) = 0 для источника и tdist(n) = Infinity для всех остальных вершин n.

При каждом шаге (итерации) обновляется временное расстояние от текущих вершин до их инцидентов (вершин, к которым идут исходящие из них ребра), после чего текущие вершины объявляются посещенными. Следующие текущие вершины (следующие вершины для посещения) определяются условием:

$$C = \{ n \mid tdist(n) \leq \min \{ tdist(u) + w(u, z) \mid u \text{ in } Q, u \text{ connected to } z \} \}$$

Величина $L = \min \{ tdist(u) + w(u, z) \mid u \text{ in } Q, u \text{ connected to } z \}$ (которая является частью условия C) далее называется пороговым значением (threshold).

Для удобства описания опишем поиск кратчайшего пути из двух точек сетевого обмена - из точки

left (101) в точку right (106), то есть найти dist(right). Описание алгоритма будет соотнесено с этапами модели описанными выше (этапы из фиг. 2).

Первые этапы модели одинаковы для всех алгоритмов (и для CMMS-OUT в частности), и подготавливают возможность работы алгоритмов. Этапы (201)-(203) выполняются аналогично описанному выше способу реализации.

В координаторе (150) ведется две приоритезированные очереди точек, одна с приоритетом по порогу (threshold), и вторая с приоритетом по минимальному весу пути - minPathWeight = min {tdist по вершинам точки}.

В вершине конуса (cone vertex) помещается координатор. С вершиной конуса не ассоциируются никакие вершины графа.

В работе алгоритма участвуют точки сетевого обмена (вершины) и вычислительные узлы (ВУ). Вычислительный узел ассоциирован с набором вершин - то есть каждый ВУ включает в себя несколько вершин (точек сетевого обмена). Каждая из вершин (точка сетевого обмена) ассоциирована ровно с одним ВУ.

С каждым ВУ ассоциировано хранилище состояний вершин ВУ (специфическое для процесса). В хранилище для каждой вершины ВУ хранятся:

предварительное расстояние (tdist);

признак посещения;

значение $\text{minCEW}(u) = \min \{ w(u, z) \mid u \text{ connected to } z \}$ - minimum connected edge weight, определяющее минимум веса исходящего ребра.

Также в хранилище хранится текущее консолидированное для всех вершин ВУ пороговое значение (threshold).

Как уже отмечалось выше на этапе (203) происходит определение координатора исполнения, а также происходит партиционирование - выбор заданного количества (X) вычислительных узлов (ВУ), которые будут определять построение семейства конусов и будут обеспечивать параллельное исполнение алгоритма.

Координатор не знает где располагается стартовая точка вычисления кратчайшего пути (left) и финальная точка пути(right).

Для того чтобы найти стартовую точку left (для инициации алгоритма) координатор отправляет сообщение вычислительным узлам (ВУ) и получает ответ в каком вычислительном узле находится стартовая вершина.

Далее координатор на этапе (204) отправляет сообщение найденному ВУ в котором находится точка left.

На данном этапе (204) происходит отправка сообщения CRAM Init(left(101)) к точке сетевого обмена left (101) (начало пути). Вершина left (101) посещается (все пути к ней == 0). Это происходит при первой итерации алгоритма.

В дальнейшем координатор (150) (в дальнейших итерациях) на этапе (204) отправляет входные данные для этапа (205) - которые являются набором вершин, которые необходимо посетить в ходе выполнения этапа (205), которые распределены по различным ВУ, в парах с ассоциированным с вершиной весом пути к ней.

Таким образом координатор (150) может отправлять эти сообщения на этапе (204) нескольким ВУ параллельно, и соответственно шаги (205)-(206) будут выполняться в параллели для каждого ВУ. С каждым ВУ ассоциирован свой набор вершин.

На этапе (205) ВУ, получившая сообщение от координатора (150), формирует маршруты последующего обмена сообщениями. ВУ извлекает список смежных вершин и распределяет работу по добавлению этих вершин графа в очередь для посещения между соответствующим ВУ.

Каждый вычислительный узел должен рассчитать предел расстояния L для следующего шага и сообщить его координатору (150).

На этапе (205) происходит формирование на ВУ маршрутов последующего обмена сообщениями.

Для каждой точки сетевого обмена которая получила сообщение, отправленное на предыдущем этапе (204):

а) Ставится признак посещения (в хранилище состояний)

б) Выполняется макрошаг (Конус). Ассоциированная точка становится cone center.

в) Строится список вершин, куда идут ребра от данной вершины (incident nodes), в парах с ассоциированным с вершиной весом пути к ней (через текущую вершину). Применяется, при необходимости, фильтрация ребер и вершин.

г) По списку рассылается CBM запроса порогового значения ThresholdRequest([(node, pathWeight)]). Сообщения агрегируются по ВУ, к которым принадлежат целевые вершины.

Внутри каждой вершины (точки сетевого обмена) по получении ThresholdRequest:

1. В хранилище состояний вершин для каждой из непосещенных вершин:

а) Если нет состояния (вершина не была ранее видима), добавляется объект состояния для данной вершины. При открытии новой вершины вершина считается непосещенной, tdist равен бесконечности,

minCEW рассчитывается при инициализации состояния и далее не меняется.

б) производится обновление предварительных расстояний непосещенных вершин: если новое предварительное расстояние pathWeight меньше текущего tdist, новое становится текущим.

2. Вычисляется предложение по пороговому значению от ВУ: пороговое значение для всех непосещенных вершин из списка, принадлежащих данному вычислительному узлу (ВУ) (с использованием веса пути к ней и минимума веса исходящего ребра minCEW). Данная величина запоминается в хранилище.

После того как на этапе (205) сформированы предложения, они на этапе (206) отправляются координатору.

На этапе (206) на координатор высылается сообщение CAM VisitReport(thresholdOffer, minPathWeight) - информация по посещению ВУ, содержащая текущее пороговое значение для данного ВУ, взятое из хранилища, и со значением minPathWeight, равным минимуму tdist по вершинам ВУ.

Также на данном этапе от каждой вершины на координатор (150) отсылается сообщение CSM ThresholdOffer(thresholdOffer, minPathWeight) со значением предложения порогового значения thresholdOffer и со значением minPathWeight, равным минимуму tdist по вершинам ВУ.

На этапе (207) координатор (150) выбирает наименьший предел и дает команду вычислительным узлам посетить те вершин, которые попадают в данный предел. Координатор вычисляет пороговое значение L и посещает те вершины, которые попали в данный предел.

На следующем этапе (207) на координаторе (150) происходит учет информации о сформированных маршрутах обмена сообщениями, а именно:

а) Координатор по получении CAM VisitReport обновляет приоритезированные очереди точек;

б) В координаторе все сообщения CSM ThresholdOffer агрегируются и вычисляется эффективное пороговое значение. Обновляются приоритезированные очереди.

Также на этапе (207) происходит вычисление вершин для следующего посещения - из приоритезированной очереди в координаторе выделяются все ВУ, имеющие непосещенные вершины, удовлетворяющие пороговому значению.

Также после учета сформированных маршрутов на этапе (207) происходит проверка окончания работы алгоритма: если в ранее полученных маршрутах имеется информация о вершине right, то завершить алгоритм и вернуть ассоциированное с right значение tdist как результат работы алгоритма. Если данной вершины не обнаружено, то происходит переход на этап 208.

На этапе (208) происходит проверка - есть ли новые вершины для посещения (информация об этом была сформирована на предыдущем этапе). Если новых вершин нет, то завершить алгоритм с результатом бесконечность (целевая вершина right не найдена, т.е. нет никакого пути, идущего к ней из left).

Если вершины имеются, то дальше происходит итеративный повтор этапов (204_ - (207) алгоритма способа (200).

Переход к этапу (204) осуществляется исходя их того, что по каждой из выбранных вершин рассылается сообщение CRAM VisitNodes(threshold), далее на этапе (205) каждый ВУ отбирает вершины, соответствующие пороговому значению. Эти вершины объявляются выбранными. Весами пути к ним являются их текущие значения tdist.

Этапы (206)-(208) выполняются до того момента, пока не будет найдена искомая вершина, или же не обнаружится что пути из точки left в точку right нет, или же алгоритм не остановится по другому критерию остановки - например максимальное количество итераций.

На фиг. 3 представлен общий вид вычислительного устройства (300), пригодного для выполнения способа (200). Устройство (300) может представлять собой, например, сервер или иной тип вычислительного устройства, который может применяться для реализации заявленного способа.

В общем случае вычислительное устройство (300) содержит объединенные общей шиной информационного обмена один или несколько процессоров (301), средства памяти, такие как ОЗУ (302) и ПЗУ (303), интерфейсы ввода/вывода (304), устройства ввода/вывода (305), и устройство для сетевого взаимодействия (306).

Процессор (301) (или несколько процессоров, многоядерный процессор) могут выбираться из ассортимента устройств, широко применяемых в текущее время, например, компаний Intel™, AMD™, Apple™, Samsung Exynos™, MediaTEK™, Qualcomm Snapdragon™ и т.п. В качестве процессора (301) может также применяться графический процессор, например, Nvidia, AMD, Graphcore и пр.

ОЗУ (302) представляет собой оперативную память и предназначено для хранения исполняемых процессором (301) машиночитаемых инструкций для выполнение необходимых операций по логической обработке данных. ОЗУ (302), как правило, содержит исполняемые инструкции операционной системы и соответствующих программных компонент (приложения, программные модули и т.п.).

ПЗУ (303) представляет собой одно или более устройств постоянного хранения данных, например, жесткий диск (HDD), твердотельный накопитель данных (SSD), флэш-память (EEPROM, NAND и т.п.), оптические носители информации (CD-R/RW, DVD-R/RW, BlueRay Disc, MD) и др.

Для организации работы компонентов устройства (300) и организации работы внешних подключаемых устройств применяются различные виды интерфейсов В/В (304). Выбор соответствующих ин-

терфейсов зависит от конкретного исполнения вычислительного устройства, которые могут представлять собой, не ограничиваясь: PCI, AGP, PS/2, IrDa, FireWire, LPT, COM, SATA, IDE, Lightning, USB (2.0, 3.0, 3.1, micro, mini, type C), TRS/Audio jack (2.5, 3.5, 6.35), HDMI, DVI, VGA, Display Port, RJ45, RS232 и т.п. Для обеспечения взаимодействия пользователя с вычислительным устройством (300) применяются различные средства (305) В/В информации, например, клавиатура, дисплей (монитор), сенсорный дисплей, тач-пад, джойстик, манипулятор мышь, световое перо, стилус, сенсорная панель, трекбол, динамики, микрофон, средства дополненной реальности, оптические сенсоры, планшет, световые индикаторы, проектор, камера, средства биометрической идентификации (сканер сетчатки глаза, сканер отпечатков пальцев, модуль распознавания голоса) и т.п.

Средство сетевого взаимодействия (306) обеспечивает передачу данных устройством (300) посредством внутренней или внешней вычислительной сети, например, Интранет, Интернет, ЛВС и т.п. В качестве одного или более средств (306) может использоваться, но не ограничиваться: Ethernet карта, GSM модем, GPRS модем, LTE модем, 5G модем, модуль спутниковой связи, NFC модуль, Bluetooth и/или BLE модуль, Wi-Fi модуль и др.

Дополнительно могут применяться также средства спутниковой навигации в составе устройства (300), например, GPS, ГЛОНАСС, BeiDou, Galileo.

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.

ФОРМУЛА ИЗОБРЕТЕНИЯ

1. Компьютерно-реализуемый способ циклического распределенного асинхронного обмена сообщениями со слабой синхронизацией, выполняемый с помощью процессора и содержащий этапы, на которых:

a) определяют множество точек сетевого обмена сообщениями, где каждая точка связана с соответствующим вычислительным узлом (ВУ);

b) формируют модель графа для обмена сообщениями между точками сетевого обмена, в котором вершинами являются точки сетевого обмена, а ребрами - факт отправки сообщений между точками сетевого обмена;

c) определяют на полученном графе точку сетевого обмена - координатора исполнения обмена сообщениями, которая связана с вычислительным узлом (ВУ);

d) осуществляют отправку сообщений от точки координатора по меньшей мере одной точке сетевого обмена;

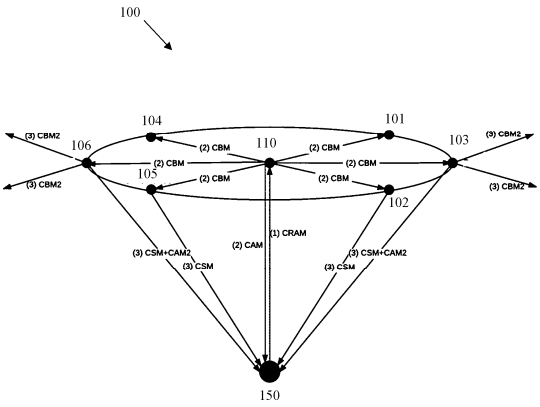
e) получают ответ от точки сетевого обмена, получившей сообщение на этапе d), причем упомянутый ответ содержит список идентификаторов смежных точек сетевого обмена, в которые будут направлены сообщения от данной точки сетевого обмена, при этом ВУ, связанный с упомянутой точкой сетевого обмена, формирует маршруты последующего обмена сообщениями, при этом ВУ извлекает список смежных точек сетевого обмена и распределяет работу по добавлению соответствующих вершин графа в очередь для посещения между ВУ, связанными с точками сетевого обмена из упомянутого списка;

f) на точке координаторе осуществляют учет сформированных маршрутов обмена сообщениями для каждой точки сетевого обмена графа, на основании сообщений от каждой точки сетевого обмена, и формируют команду на обмен сообщений между ВУ, определенными на этапе e);

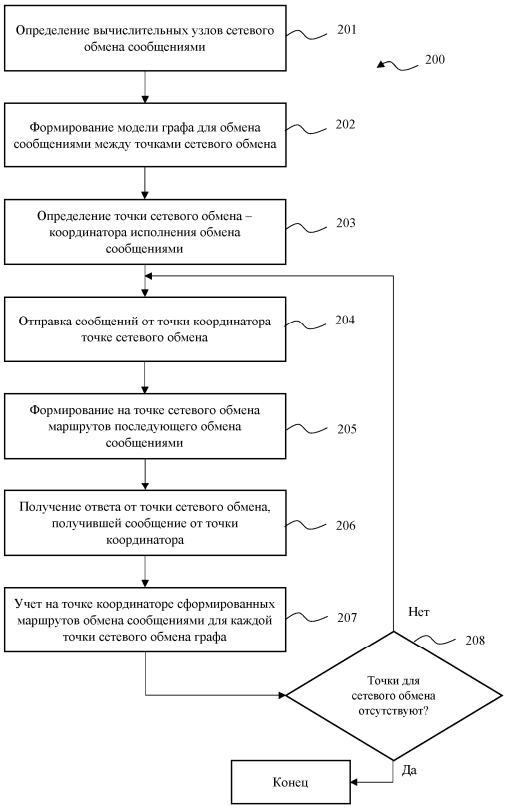
g) итеративно повторяют этапы e)-f) до того момента, пока точка координатор не сообщит, что точки сетевого обмена отсутствуют.

2. Способ по п.1, характеризующийся тем, что каждая вершина содержит хранилище состояний, в котором содержится информация о признаке посещения.

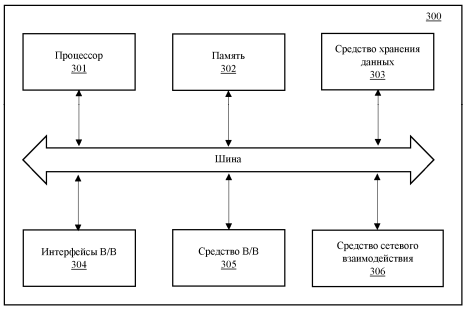
3. Система циклического распределенного асинхронного обмена сообщениями со слабой синхронизацией, содержащая по меньшей мере один процессор и по меньшей мере одно средство памяти, которое хранит машиночитаемые инструкции, которые при их исполнении процессором реализуют способ по любому из пп.1-2.



Фиг. 1



Фиг. 2



Фиг. 3