Научная статья
УДК004:519.854
DOI:10.31854/1813-324X-2023-9-2-112-127

Иерархическая модель и алгоритм оптимизации решений при распределенном хранении и обработке данных

Кирилл Викторович Кротов, krotov_k1@mail.ru

Севастопольский государственный университет,
Севастополь, 299053, Российская Федерация

Аннотация: *Задача оптимизации распределенного хранения и обработки данных является трудноразрешимой за ограниченное время. В связи с этим для ее решения применен иерархический подход, предусматривающий представление обобщенной задачи в виде совокупности иерархически упорядоченных подзадач, для каждой из которых на соответствующем ей уровне иерархии определяются локально оптимальные решения. Для оптимизации решений по распределенному хранению и обработке данных сформирована модель процесса в виде совокупности иерархически упорядоченных компонент, а также математическая модель иерархической игры, представляющая собой способ оптимизации решений на уровнях иерархии. С целью определения эффективных решений на уровнях иерархии разработан алгоритм локальной оптимизации решений, в основу которого положены генетические алгоритмы. Построение расписаний обработки данных, назначенных на вычислительные устройства, реализуется с использованием предложенной эвристической процедуры. Применение разработанных моделей процесса распределенного хранения и обработки данных, модели иерархический игры и алгоритмов оптимизации решений позволили значительно увеличить размерность задачи, учесть при оптимизации решений на уровнях иерархии параметры, характеризующие каналы передачи данных, минимизировать количество неиспользованных ресурсов.*

Ключевые слова: *иерархическая игра, ограничения на объемы распределенных устройств хранения, оптимизация решений по распределенному хранению и распределенной обработке данных, генетические алгоритмы.*

Ссылка для цитирования: Кротов К.В. Иерархическая модель и алгоритм оптимизации решений при распределенном хранении и обработке данных // Труды учебных заведений связи. 2023. Т. 9. № 2. С. 112–127. DOI:10.31854/1813-324X-2023-9-2-112-127

Hierarchical Model and Decision Optimization Algorithm for Distributed Data Storage and Processing

Kirill Krotov, krotov_k1@mail.ru

[1]Sevastopol State University
Sevastopol, 299053, Russian Federation

Abstract: *The task of optimizing distributed data storage and processing is difficult to solve in a limited time. In this regard, a hierarchical approach has been applied to solve it, which provides for the presentation of a generalized problem in the form of a set of hierarchically ordered subtasks, for each of which locally optimal solutions are determined at the appropriate hierarchy level. To optimize solutions for distributed data storage and processing, a process model has been formed, presented in the form of a set of hierarchically ordered components, a mathematical model of a hierarchical game, which is a way to optimize solutions at hierarchy levels. In order to determine effective solutions at hierarchy levels, an algorithm for local optimization of solutions based on genetic algorithms*

has been developed. The construction of data processing schedules assigned to computing devices is implemented using the proposed heuristic procedure. The application of the developed models of the distributed data storage and processing process, hierarchical game models and algorithms for optimizing solutions made it possible to significantly increase the dimension of the problem, take into account the parameters characterizing data transmission channels when optimizing solutions at hierarchy levels, and minimize the amount of unused resources.

Keywords: *hierarchical game, restrictions on the volume of distributed storage devices, optimization of solutions for distributed storage and distributed data processing, genetic algorithms*

For citation: Krotov K. Hierarchical Model and Decision Optimization Algorithm for Distributed Data Storage and Processing. *Proc. of Telecom. Universities.* 2023;9(2):112–127. (in Russ.) DOI:10.31854/1813-324X-2023-9-2-112-127

Введение

В настоящее время активно развивается подход к обработке данных, предполагающий, что устройства их хранения и выполнения вычислений с ними являются географически распределенными. Указанный подход к обработке данных предусматривает распределенное их хранение и распределенную их обработку, в смысле – выполнение вычислений (РХ-РО). Предполагается, что для решения некоторого пула задач обработки данных выделяются ресурсы ограниченного объема как для их хранения, так и для вычислений. Также следует учитывать, что хранение данных на устройствах и их обработка связаны с взиманием платы за оказываемые услуги. Отсюда рассматриваемая постановка задачи типа «РХ-РО» характеризуется особенностями, связанными с наличием ограничений на ресурсы, а также с назначением стоимости за оказание соответствующих услуг.

В то же время немаловажным фактором является эффективное использование ресурсов, реализующих передачу данных между устройствами хранения и обработки, которое также характеризуется стоимостными показателями. Таким образом, реализация РХ-РО обеспечивается использованием трех видов ресурсов. В связи с необходимостью эффективного применения ограниченных ресурсов, а также с целью минимизации финансовых затрат требуется оптимизировать их (данных) распределение по устройствам хранения и назначение вычислительных устройств (ВУ) для их обработки.

Современное состояние научных исследований, направленных на планирование распределенного выполнения заданий (обработки данных на распределенных ВУ) связано в основном с применением различных эвристических подходов. В [1] рассматривается применение известных эвристик для выбора задания, назначаемого на устройство в GRID-системах: «задание с наименьшим временем обработки – первым» (SPTF, *аббр. от англ.* Shortest-Processing-Time-First), «короткое задание – первым» (SJF, *аббр. от англ.* Short-Job-First), «задание с самым длинным временем обработки – первым» (LPTF, *аббр. от англ.* Longest-Processing-Time-First), «задание самого большого размера – первым» (LSF, *аббр. от англ.* Largest-Size-First), «задание с самым ранним директивным сроком окончания выполнения – первым» (EDF, *аббр. от англ.* Earliest-Deadline-First). Также в [1] предложен способ упорядочивания заданий в очереди на выполнение с использованием специальных метрик, выражения для вычисления которых предусматривают использование параметров: директивного срока, длительности обработки данных, значения текущего времени. Кроме того, в [1] рассматривается применение эвристик для выбора устройства, на которое может быть назначено задание из головы очереди: эвристика «выбор наиболее подходящего» (*от англ.* Best-Fit) выбирает узел, который имеет наименьшее количество доступных ресурсов и может выполнить обработку рассматриваемых данных; эвристика «выбор первого подходящего» (*от англ.* First-Fit) планирует задания по первому подходящему ресурсу из списка доступных; эвристика «самый быстрый ресурс – первый» (FRF, *аббр. от англ.* Fastest-Resource-First) выбирает самый быстрый ресурс из списка доступных и назначает его для выполнения задания; эвристика «минимально загруженный ресурс – первый» (MLF, *аббр. от англ.* Min-Loaded-First) выбирает ресурс, имеющий максимальную неиспользуемую мощность процессора.

Развитию эвристического подхода к планированию в GRID-системах посвящена работа [2], в которой рассматриваются эвристики: 1) UDA (*аббр. от англ.* User-Defined-Attributes, определяемый пользователем атрибут) – назначает устройству задачу с наилучшим ожидаемым временем ее выполнения, независимо от его (устройства) доступности; 2) алгоритм Min-Min–задача с самым минимальным временем выполнения назначается на соответствующий ресурс; 3) алгоритм Max-Min–задача с максимальным (среди минимальных) временем выполнения назначается на соответствующий ресурс.

В работе [3] также рассматривается применение эвристик Min-Min и Max-Min к планированию в GRID-системах; анализируются эвристические правила:

1) «минимальное время выполнения – первым» (MET, *аббр. от англ.* Minimum-Execution-Time), ко-

торое назначает задание устройству, имеющему наименьшее время выполнения для этого задания;

2) «самое длинное задание – на самый быстрый ресурс» (LJFR, *аббр. от англ.* Longest-Job-to-Fastest-Resource);

3) «самое короткое задание – на самый быстрый ресурс» (SJFR, *аббр. от англ.* Shortest-Job-to-Fastest-Resource);

4) «минимальная относительная стоимость выполнения задания – первой» (RC, *аббр. от англ.* Relative-Cost) – предполагает вычисление значений параметров статической и динамической относительной стоимости и упорядочивание заданий в соответствии с этими значениями.

Исследованию рассмотренных выше эвристик (в том числе MET, MLF, FRF, Min-Min и Max-Min) применительно к планированию в GRID-системах посвящена работа [4]. Также упоминается о возможности применения метаэвристических алгоритмов (генетические алгоритмы, муравьиные колонии, имитация отжига) к решению задач планирования в рассмотренных системах. Однако непосредственная адаптация этих алгоритмов для решения указанных задач планирования в этой работе не рассматривается.

В работе [5] рассматриваются различные политики (эвристические правила) назначения ресурсов заданиям в облачных средах. Это политика «первого доступного кэша» (FCA, *аббр. от англ.* First-Cache-Available), политика «максимального количества попаданий кэша» (MCH, *аббр. от англ.* Max-Cache-Hit), «максимальное количество вычислений» (MCU, *аббр. от англ.* Max-Compute-Util) и «хорошее вычисление кэша» (GCC, *аббр. от англ.* Good-Cache-Computing). Политика FCA игнорирует информацию о местоположении данных при выборе исполнителя для задачи; она обеспечивает выбор первого доступного исполнителя (устройства) и не предоставляет исполнителю никакой информации о расположении данных, необходимых для задачи; исполнитель должен получать все эти данные из постоянного хранилища при каждом доступе. Политика MCH использует информацию о расположении данных для отправки каждой задачи исполнителю с наибольшим объемом данных, необходимых для ее решения. Если исполнитель занят, то отправка откладывается до тех пор, пока он не станет доступным. Политика MCU использует информацию о местоположении данных, пытаясь максимизировать использование ресурсов даже при потенциально более высоких затратах на перемещение данных. Она назначает задачу доступному исполнителю, предпочитая исполнителей, находящихся «ближе» к необходимым для ее решения данным. Политика GCC – гибридная политика по отношению к MCH и MCU; она устанавливает порог минимального использования процессора с целью определения необходимости использования одной из указанных политик (MCH или MCU).

В работе [6] наряду с уже рассмотренными эвристиками используются следующие правила для планирования выполнения заданий в вычислительных кластерах: «наименее подходящий» (WF, *аббр. от англ.* Worst-Fit); случайное назначение (RF, *аббр. от англ.* Random-Fit), а также методы предварительного резервирования и обратного заполнения. Метод предварительного резервирования использует информацию о времени выполнения, предоставленную пользователем, чтобы зарезервировать ресурсы процессора и памяти, и соответственно сгенерировать расписание. Метод обратного заполнения является улучшением алгоритма планирования с полным распределением задач по ресурсам.

В работе [7] рассматривается способ определения количества задач, входящих в задания (работы), назначаемые для выполнения на разных ВУ. При определении указанного количества задач учитывается степень параллелизма (то есть количество параллельно выполняющихся задач, входящих в задание/работу), а также производительность (скорость выполнения вычислений) как отдельных процессоров, так и производительность всей системы. Определенные таким образом совокупности задач назначаются соответствующим ВУ для выполнения.

Рассмотренный в [8] алгоритм обеспечивает назначение виртуальных машин для выполнения заданий с учетом их (заданий) бюджета. Алгоритм предполагает первоначально формирование списка виртуальных машин, которые могут быть назначены для выполнения заданий. Для каждой виртуальной машины из этого списка с учетом характеристик заданий (количество операций в программах, размеры входного и выходного файлов), а также характеристик самих виртуальных машин (пропускные способности каналов, связывающих устройства, объемы оперативной и постоянной памяти, выделяемых для хранения данных, стоимости передачи данных по каналам и хранения) определяется стоимость их (заданий) выполнения на соответствующих устройствах. В том случае, если стоимость выполнения задания на виртуальной машине не превышает выделенного для него бюджета, то рассматриваемая машина назначается для реализации вычислений с этим заданием. Здесь стоимость выполнения заданий на виртуальных машинах (и формула, позволяющая вычислить эту стоимость) является своего рода эвристикой, используемой при планировании.

В [9] предложен алгоритм назначения ВУ в облачной среде с учетом заданных директивных сроков окончания выполнения заданий и ограничений на бюджеты, связанные с их выполнением. Задан

граф следования заданий при реализации вычислительного процесса (граф передачи управления между заданиями при реализации вычислительного процесса). Алгоритм предусматривает первоначальное распределение заданий по уровням независимого (параллельного) выполнения – формирование соответствующей формы, в соответствии с которой реализуется назначение заданий каждого уровня (при движении сверху-вниз по полученной форме) на ВУ с учетом бюджета выполнения и с последующим определением сроков их завершения. При распределении заданий по устройствам учитываются только временные и стоимостные параметры их выполнения. Рассмотренный алгоритм не предусматривает оптимизации решений и может быть отнесен к эвристическим.

В целом выполненный анализ существующих методов планирования распределенного выполнения заданий на ВУ позволил определить следующие их недостатки:

– рассмотренные алгоритмы и методы реализуют планирование выполнения заданий на устройствах с использованием эвристического подхода (эвристических правил и алгоритмов), то есть не предусматривают поиска приближенно оптимальных или локально оптимальных решений;

– при распределении заданий на устройства большинство алгоритмов учитывают только один вид ресурсов – процессорное время, но не учитывают ресурсы других видов – каналы передачи данных и хранилища данных (в том числе ограниченного объема).

В связи с этим разработка новых методов и алгоритмов планирования распределенного хранения и распределенной обработки данных при учете передачи данных между устройствами и учете стоимостных характеристик реализации указанных операций является актуальной.

Задача оптимизации решений по размещению данных на устройствах хранения ограниченного объема, по назначению ВУ для их обработки (с учетом использования каналов передача между указанными устройствами) является NP-трудной [1]. В связи с этим необходима разработка приближенных методов оптимизации решений или методов локальной оптимизации решений. Одним из возможных способов решения указанной задачи оптимизации является иерархический подход [10, 11], который предусматривает выделение в обобщенной задаче оптимизации совокупности подзадач, каждая из которых решается на назначенном ей уровне иерархии; реализация указанного подхода предполагает:

– определенный порядок формирования решений – первоначально решение формируется на вышестоящем уровне, затем на нижестоящем уровне выбирается лучшее (в частном случае, локально-оптимальное);

– необходимость обмена решениями между уровнями – решение, сформированное на вышестоящем уровне, передается на нижестоящий для определения локально-оптимального решения, которое передается на вышестоящий для вычисления на его основе оценки оптимальности;

– зависимость оценки оптимальности решений на вышестоящем уровне от решения на нижестоящем – обобщенное оптимальное решение задачи формируется путем определения оптимальных решений на каждом из уровней иерархии.

В соответствии с указанными особенностями иерархического подхода для решения сформулированной задачи может быть применен аппарат теории иерархических игр. В связи с этим получение эффективных решений по размещению данных в хранилищах, по назначению ВУ для их обработки обеспечивается разработкой:

– математической модели процесса распределенных хранения и обработки данных (при ограничениях на размеры хранилищ и установлении стоимостных характеристик по оказанию услуг, связанных с хранением, обработкой и передачей данных);

– математической модели иерархической игры оптимизации решений по распределенному хранению и распределенной обработке данных как способа определения эффективных (в частном случае, локально-оптимальных) решений;

– метода (алгоритма) поиска локально-оптимальных решений на каждом из уровней иерархии;

– алгоритма построения расписаний обработки данных, назначенных на соответствующие ВУ.

Таким образом, основное предназначение выполняемого исследования состоит в значительном увеличении размерности рассматриваемых задач, а также в гарантированном получении эффективных решений при различных значениях их (задач) входных данных.

1. Математическое моделирование процессов и синтез математической модели оптимизации решений по распределенному хранению и обработке данных

Для синтеза математической модели процесса распределенного хранения и обработки данных в рассмотрение введены обозначения для входных данных и параметров задачи:

1) i – идентификатор типа данных, распределенное хранение и обработка которых выполняется в системе ($i = \overline{1, N}$);

2) d_i – объем данных i-го типа, которые должны быть размещены в устройствах хранения ($i = \overline{1, N}$);

3) m – идентификатор устройства распределенного хранения данных ($m = \overline{1, M}$);

4) v_m – размер m-го хранилища данных ($m = \overline{1, M}$), являющийся задаваемым при решении задачи оптимизации; $V = (v_m | \ m = \overline{1, M})$– вектор-строка размеров m-х хранилищ данных;

5) s_m – стоимость хранения единицы данных в единицу времени на m-м устройстве хранения; $S = (s_m | \ m = \overline{1, M})^T$ – вектор-столбец стоимостей хранения данных разных типов в единицу времени на m-х устройствах;

6) sh_m – размер штрафа за единицу неиспользованного ресурса на m-ом устройстве хранения; $Sh = (sh_m | \ m = \overline{1, M})$– вектор-строка штрафов за единицу неиспользованного ресурса на m-х устройствах;

7) l – идентификатор устройства распределенной обработки данных ($l = \overline{1, L}$);

8) t_{il}– длительность обработки данных i-го типа на l-м ВУ ($i = \overline{1, N}; l = \overline{1, L}$); $T = \|t_{il}\|_{N \times L}$– матрица длительностей обработки данных i-х типов на l-х ВУ;

9) w_l – стоимость единицы времени обработки данных на l-м ВУ ($l = \overline{1, L}$);

10) c_{ml} – пропускная способность канала передачи данных между m-м устройством хранения и l-м устройством обработки (в том случае, если между m-м устройством хранения и l-м устройством обработки отсутствует канал передачи данных, то $c_{ml} = 0$); $C = \|c_{ml}\|_{M \times L}$ – матрица пропускных способностей каналов передачи данных между m-ми устройствами хранения и l-ми устройствами обработки;

11) b_{ml} – длина канала передачи данных между m-м устройством хранения и l-м устройством обработки ($b_{ml} = 0$, если канал передачи данных отсутствует); $B = \|b_{ml}\|_{M \times L}$ – матрица длин каналов передачи данных между m-ми устройствами хранения и l-ми устройствами обработки;

12) $b_{\max}$ – максимальная длина канала передачи данных, используемая для вычисления значений коэффициентов длин каналов θ_{ml} ($m = \overline{1, M}$; $l = \overline{1, L}$);

13) θ_{ml} – коэффициент длины канала передачи данных между m-м устройством хранения и l-м устройством обработки (вычисляется в соответствии с формулой: $\theta_{ml} = (b_{ml}/b_{\max})$); $\Theta = \|\theta_{ml}\|_{M \times L}$ – матрица коэффициентов длин каналов передачи данных между m-ми устройствами хранения и l-ми устройствами обработки;

14) q_{ml} – стоимость передачи единицы данных между на m-м устройством хранения и l-м устройством обработки ($m = \overline{1, M}; l = \overline{1, L}$); $Q = \|q_{ml}\|_{M \times L}$ – матрица стоимостей передачи единицы данных между m-ми устройствами хранения и l-ми ВУ.

Постановка задачи предполагает введение условия $\sum_{i=1}^{N} d_i \leq \sum_{m=1}^{M} v_m$, предусматривающего, что все данные будут распределены для хранения по устройствам. Необходимость построения расписания загрузки на устройства хранения в связи с этим отсутствует. Однако при распределении данных i-х типов по ВУ требуется построение расписаний их (данных) обработки на устройствах, так как на его основе осуществляется расчет стоимостных показателей, связанных с хранением.

В соответствии с иерархическим подходом [10, 11] к математическому моделированию и оптимизации процессов распределенного хранения и обработки данных выполнена декомпозиция обобщенной функции системы на совокупность иерархически упорядоченных подфункций, которые распределены по уровням следующим образом:

1) верхний уровень – моделирование и оптимизация распределенного хранения данных на устройствах;

2) нижний уровень – моделирование и оптимизация распределенной обработки данных на ВУ; также на нижнем уровне реализуется построение расписаний обработки на устройствах назначенных данных.

Построение в соответствии с выполненной декомпозицией математической модели процессов хранения и обработки данных предполагает введение в рассмотрение иерархически упорядоченных компонент, которые содержат следующие матрицы:

компонента верхнего уровня

– матрицу назначений $P = \|p_{im}\|_{N \times M}$, элементы которой $p_{im} = 1$, если данные i-го типа размещены для хранения на m-м устройстве, и $p_{im} = 0$, если данные i-го типа не размещены для хранения на m-м устройстве;

– матрицу $T^{mem} = \|t_{im}^{mem}\|_{N \times M}$ интервалов времени хранения данных i-х типов на m-х устройствах хранения ($i = \overline{1, N}; m = \overline{1, M}$); элемент матрицы $t_{im}^{mem} \neq 0$ в том случае, если для соответствующих индексов i и m элемент p_{im}матрицы P равен 1 ($p_{im} = 1$), элемент матрицы $t_{im}^{mem} = 0$ в том случае, если для соответствующих индексов i и m элемент p_{im} матрицы P равен 0 ($p_{im} = 0$);

компонента нижнего уровня

– матрицу назначений $R = \|r_{li}\|_{L \times N}$, элементы которой $r_{li} = 1$, если данные i-го типа назначены для обработки на l-м ВУ, и $r_{li} = 0$, если данные i-го типа не назначены для обработки на l-м устройстве;

– матрицу $T^0 = \left\|t_{il}^0\right\|_{N \times L}$моментов времени начала обработки данных i-х типов на l-х ВУ (элемент t_{il}^0 матрицы T^0 – это момент времени начала обработки данных i-го типа на l-м ВУ ($i = \overline{1, N}; l = \overline{1, L}$); $t_{il}^0 \neq 0$ в том случае, если для соответствующих индексов i и l элемент r_{li} матрицы R равен 1

($r_{li} = 1$), $t_{il}^0 = 0$ в том случае, если для соответствующих индексов i и l элемент r_{li} матрицы R равен 0 ($r_{li} = 0$)).

Элементы t_{il}^0 матрицы T^0 соответствуют определенным порядкам обработки данных i-х типов на l-х ВУ, формируемым с использованием разработанной эвристической процедуры. То есть матрица T^0 соответствует расписаниям обработки данных i-х типов на l-х ВУ.

В этом случае математическая модель процесса распределенного хранения и распределенной обработки данных имеет вид кортежей:

– *компонента верхнего уровня* – кортеж вида $[P, T^{mem}]$, соответствующий распределению данных для хранения по устройствам и характеризующий интервалы времени хранения их на этих устройствах;

– *компонента нижнего уровня* – кортеж вида $[R, T^0]$ соответствующий распределению данных на ВУ и характеризующий моменты времени начала их обработки на этих устройствах (иными словами – расписание распределенной обработки данных на ВУ).

Компоненты математической модели процесса распределенного хранения и обработки данных соответствуют решениям, оптимизируемым на каждом из уровней иерархии. На верхнем уровне оптимизируется решение по распределенному хранению данных и по интервалам времени, в течение которых данные определенных типов будут храниться на соответствующих устройствах. На нижнем уровне оптимизируется решение по распределенной обработке данных и по моментам времени начала их обработки на ВУ, которые соответствуют расписаниям выполнения операций с данными. Применение теоретико-игрового подхода предполагает назначение на верхнем уровне ведущего игрока, который выполняет оптимизацию решений по распределенному хранению данных, и назначение на нижнем уровне ведомого игрока, который выполняет оптимизацию решений по распределенной обработке данных (при условии, что расписания обработки данных, формируемые с использованием эвристической процедуры, однозначно соответствуют распределению обработки данных по ВУ).

Особенностью реализации теоретико-игрового подхода к иерархической оптимизации решений по распределенному хранению и обработке данных (особенностями взаимодействия игроков при оптимизации решений на уровнях игры) являются [12–14]:

1) передача решения по распределенному хранению данных с верхнего уровня на нижний с целью формирования на его основе решения по распределенной обработке данных;

2) формирование для полученного с верхнего уровня решения по распределенному хранению данных локально-оптимального решения по распределенной обработке данных на соответствующем ему нижнем уровне;

3) передача локально-оптимального решения по распределенной обработке данных с нижнего уровня на верхний с целью оценки оптимальности решения по распределенному хранению данных.

В соответствии с указанными особенностями взаимодействия игроков и введенными обозначениями для решений, оптимизируемым на каждом из уровней, модель иерархической игры представлена в следующем общем виде [12–14]:

1) верхний уровень:

$$\min_{[P,T^{mem}]\in N_1} f_1([P, T^{mem}], [R, T^0]*);$$

2) нижний уровень:

$$\min_{[R,T^0]\in N_2([P,T^{mem}])} f_2([P, T^{mem}], [R, T^0]),$$

где N_1, $N_2([P, T^{mem}])$ – множество решений по распределенному хранению данных и по их обработке, соответствующее сформированному решению $[P, T^{mem}]$ по распределенному хранению данных, полученному с верхнего уровня.

Построение математической модели иерархической игры реализуется в предположении, что на верхнем уровне оптимизация решений выполняется с учетом внешней цели функционирования системы, которая требует минимизации финансовых затрат на хранение, передачу и обработку данных (минимизацию стоимости использования ресурсов). Оптимизация решений по распределенной обработке данных на нижнем уровне выполняется с точки зрения внутренней цели функционирования системы, предусматривающей минимизацию длительностей передачи и обработки данных. На основе указанных особенностей формирование критериев на уровнях иерархической игры оптимизации решений по распределенному хранению и обработке данных (при учете передачи данных между устройствами) выполняется в соответствии с рассматриваемыми ниже рассуждениями.

Интервал времени передачи данных одного i-го типа между m-м устройством, на котором реализуется их хранение, и l-м ВУ, на котором реализуется их обработка, определяется следующим образом:

$$\sum_{m=1}^{M}\sum_{l=1}^{L} p_{im} r_{li} \frac{d_i}{c_{ml}}.$$

В представленном выражении не учитывается длина канала связи, который используется для передачи данных. В рассмотрение введен коэффициент θ_{ml}, выступающий в роли веса, позволяющего учесть длины каналов связи в формируемых

оценках критерия при сравнении решений по распределенной обработке данных.

В этом случае оценка времени передачи данных одного i-го типа между m-м устройством, на котором реализуется их хранение, и l-м ВУ, на котором реализуется их обработка (с учетом веса, характеризующего длину канала связи), определяется выражением:

$$\sum_{m=1}^{M}\sum_{l=1}^{L} p_{im} r_{li} \frac{d_i}{c_{ml}} \theta_{ml}.$$

Тогда обобщенная оценка времени передачи данных всех n типов между устройствами, на которых реализуется их хранение, и устройствами, на которых реализуется их обработка (с учетом весов, соответствующих длинам каналов передачи данных), определяется выражением:

$$\sum_{i=1}^{N}\sum_{m=1}^{M}\sum_{l=1}^{L} p_{im} r_{li} \frac{d_i}{c_{ml}} \theta_{ml}.$$

Интервал времени выполнения вычислений с данными i-х типов, обработка которых назначена на одном l-м устройстве, определяется следующим образом:

$$\sum_{i=1}^{N} r_{li} t_{il}.$$

При условии, что обработку данных на всех устройствах требуется распределить равномерно (равномерная загрузка ВУ обработкой данных), интервал времени, соответствующий окончанию обработки данных всех N типов на всех параллельно функционирующих устройствах, определяется выражением:

$$\max_{l=\overline{1,L}} \sum_{i=1}^{N} r_{li} t_{il}.$$

На нижнем уровне иерархической игры необходимо минимизировать общие временные затраты на передачу данных между устройствами хранения и обработки, а также временные затраты на обработку данных (с целью минимизации общего времени использования ресурсов обработки и передачи данных).

Указанные характеристики процесса распределенной обработки данных определяются выражением следующего вида:

$$\sum_{i=1}^{N}\sum_{m=1}^{M}\sum_{l=1}^{L} p_{im} r_{li} \frac{d_i}{c_{ml}} \theta_{ml} + \max_{l=\overline{1,L}} \sum_{i=1}^{N} r_{li} t_{il}. \qquad (1)$$

Выражение (1) используется в качестве критерия оптимальности решений по распределенной обработке данных всех N типов на L параллельно функционирующих устройствах.

Определение интервалов времени t_{im}^{mem} хранения данных i-х типов на m-х устройствах ($i = \overline{1,N}; m = \overline{1,M}$) реализуется в предположении, что все данные N типов распределяются по устройствам хранения одновременно в момент времени, равный 0. Тогда интервал времени хранения данных i-го типа на m-м устройстве определяется с учетом момента времени t_{il}^{0} начала обработки данных на l-м ВУ и интервала времени передачи всего объема данных этого типа (в количестве d_i) с m-го устройства на l-е устройство. То есть момент времени окончания хранения данных i-го типа на m-м устройстве (и, соответственно, интервал времени t_{im}^{mem} их хранения на этом устройстве) определяется как разность между моментом времени начала обработки данных на l-м устройстве и интервалом времени передачи данных по каналу, соединяющему m-е устройство (где хранятся данные) и l-е устройство (где они обрабатываются).

Интервал времени передачи данных между m-м устройством хранения и l-м устройством обработки определяется как сумма интервала времени выставления данных в канал в заданном их объеме d_i (при известной пропускной способности c_{ml} канала, по которому передаются данные) и интервала времени передачи данных по каналу. С целью определения интервала времени передачи данных по каналу в рассмотрение введена константа S_{tr}, соответствующая скорости распространения сигнала в канале ($S_{tr} = 2 * 10^5$км/с).

Тогда для данных i-го типа, размещенных для хранения на m-м устройстве и обрабатываемых на l-м устройстве, время их передачи по каналу будет определено следующим образом:

$$\sum_{l=1}^{L} p_{im} \cdot r_{li} \cdot \frac{b_{ml}}{S_{tr}},$$

где b_{ml} – длина канала между m-м устройством (где данные размещены для хранения) и l-м устройством, на котором данные обрабатываются.

Интервал времени выставления данных i-го типа, размещенных для хранения на m-м устройстве, в канал, соединяющий это устройство и l-е устройство, на котором они обрабатываются, определяется выражением вида:

$$\sum_{l=1}^{L} p_{im} \cdot r_{li} \cdot \frac{d_i}{c_{ml}}.$$

Тогда общий интервал времени выставления данных i-го типа (количество которых равно d_i) в канал с пропускной способностью c_{ml} и передачи этих данных по каналу, характеризуемому длиной b_{ml}, определяется следующим образом (при зафиксированном значении типа данных i и индексе m хранилища, на котором они находятся):

$$\sum_{l=1}^{L} p_{im} r_{li} \left(\frac{d_i}{c_{ml}} + \frac{b_{ml}}{S_{tr}} \right).$$

Значение t_{il}^0 для данных i-го типа, обрабатываемых на l-м устройстве, определяется в соответствии с расписанием, формируемом с использованием эвристической процедуры.

Тогда интервал времени t_{im}^{mem} хранения данных i-го типа на m-м устройстве определяется следующим образом (значения идентификатора i типа данных и идентификатора устройства хранения m зафиксированы):

$$t_{im}^{mem} = \sum_{l=1}^{L} \left(t_{il}^0 r_{li} - p_{im} r_{li} \left(\frac{d_i}{c_{ml}} + \frac{b_{ml}}{S_{tr}} \right) \right),$$

где $i = \overline{1, N}, m = \overline{1, M}$.

С целью синтеза критерия оптимальности решений на верхнем уровне иерархической игры введена в рассмотрение матрица $P' = \|p'_{mi}\|_{M \times N}$, элементы которой определяются следующим образом: $P' = P^T$ (где T – символ операции транспонирования матрицы P). То есть $p'_{mi} = p_{im}$ $(m = \overline{1, M}; i = \overline{1, N})$.

Тогда для данных i-го типа, хранящихся на m-м устройстве, стоимость хранения единицы данных в течение интервала времени t_{im}^{mem} определяется выражением вида:

$$\sum_{m=1}^{M} t_{im}^{mem} \cdot p'_{mi} s_m.$$

В силу истинности условия:

$$\sum_{i=1}^{N} d_i \le \sum_{m=1}^{M} v_m$$

все данные будут распределены для хранения по устройствам.

Тогда стоимость хранения всех данных n типов на устройствах определяется выражением вида:

$$\sum_{i=1}^{N} d_i \left(\sum_{m=1}^{M} t_{im}^{mem} p'_{mi} s_m \right).$$

где d_i – это количество данных i-х типов, хранящихся на m-х устройствах.

Стоимость выполнения единицы вычислительных операций на l-ом устройстве обозначена через w_l, а длительность выполнения всех операций на l-м устройстве с назначенными на него данными определяется выражением:

$$\sum_{i=1}^{N} r_{li} t_{il}.$$

Тогда стоимость выполнения всех вычислительных операций на отдельном l-м устройстве определяется выражением:

$$w_l \sum_{i=1}^{N} r_{li} t_{il},$$

а стоимость реализации вычислительных операций с данными на всех L устройствах обработки:

$$\sum_{l=1}^{L} w_l \sum_{i=1}^{N} r_{li} t_{il}.$$

Аналогичным образом затраты на передачу единицы данных разных i-х типов между некоторым m-м устройством хранения, на котором они размещены, и некоторым l-м устройством обработки, на котором они обрабатываются, определяются выражением:

$$\sum_{i=1}^{N} p_{im} r_{li} q_{ml},$$

а затраты на передачу всего объема d_i данных i-х типов между m-м устройством хранения и l-м устройством обработки определяется выражением:

$$\sum_{i=1}^{N} p_{im} r_{li} d_i q_{ml}.$$

Выражение для определения суммарных затрат на передачу данных i-х типов $(i = \overline{1, N})$ от разных m-х устройств хранения на различные l-е устройства ~~для~~ обработки имеет вид:

$$\sum_{i=1}^{N} \sum_{m=1}^{M} \sum_{l=1}^{L} p_{im} r_{li} d_i q_{ml}.$$

Количество использованного ресурса на некотором m-м устройстве для хранения данных i-го типа определяется выражением:

$$\sum_{i=1}^{N} p'_{mi} d_i,$$

где p'_{mi} – элемент матрицы P' $(P' = P^T)$.

Количество неиспользованного ресурса на m-м устройстве хранения определяется выражением:

$$v_m - \sum_{i=1}^{N} p'_{mi} d_i.$$

Тогда размер штрафа за неиспользованный ресурс хранения данных на m-м устройстве определяется выражением:

$$s_m \left(v_m - \sum_{i=1}^{N} p'_{mi} d_i \right).$$

Итоговое выражение для определения суммарных штрафов за неиспользуемые ресурсы на всех устройствах хранения имеет вид:

$$\sum_{m=1}^{M} s_m \left(v_m - \sum_{i=1}^{N} p'_{mi} d_i \right).$$

Выражение для суммарных затрат на распределенное хранение и обработку данных, передачу данных между устройствами хранения и обработки при учете штрафов за неиспользование ограниченных ресурсов их хранения, полученное на основе синтезированных выше выражений, имеет вид:

$$\sum_{i=1}^{N} d_i \left(\sum_{m=1}^{M} t_{im}^{mem} p'_{mi} s_m \right) + \sum_{l=1}^{L} w_l \sum_{i=1}^{N} r_{li} t_{il} + \sum_{i=1}^{N} \sum_{m=1}^{M} \sum_{l=1}^{L} p_{im} r_{li} d_i q_{ml} + \sum_{m=1}^{M} s_m \left(v_m - \sum_{i=1}^{N} p'_{mi} d_i \right). \quad (2)$$

С учетом выражений (1, 2) для критериев оптимальности решений по распределенному хранению и обработке данных получена математическая модель иерархической игры оптимизации указанных решений в следующем виде:

1) верхний уровень – минимизация затрат на хранение, передачу, обработку данных и штрафов за не полное использование ограниченных ресурсов:

$$\min \ f_1 \ , \quad (3)$$

где

$$f_1 = \sum_{i=1}^{N} d_i \left(\sum_{m=1}^{M} t_{im}^{mem} p'_{mi} s_m \right) + \sum_{l=1}^{L} w_l \sum_{i=1}^{N} r_{li} t_{il} + \sum_{i=1}^{N} \sum_{m=1}^{M} \sum_{l=1}^{L} p_{im} r_{li} d_i q_{ml} + \sum_{m=1}^{M} s_m \left(v_m - \sum_{i=1}^{N} p'_{mi} d_i \right);$$

2) нижний уровень – минимизация времени передачи данных и времени обработки:

$$\min \ f_2 \ , \quad (4)$$

где

$$f_2 = \sum_{i=1}^{N} \sum_{m=1}^{M} \sum_{l=1}^{L} p_{im} r_{li} \frac{d_i}{c_{ml}} \theta_{ml} + \max_{l=\overline{1,L}} \sum_{i=1}^{N} r_{li} t_{il};$$

3) ограничения на множества допустимых решений на верхнем и нижнем уровне:

$$\sum_{m=1}^{M} p_{im} = 1 \ (i = \overline{1,N}), \quad (5)$$

$$\sum_{l=1}^{L} r_{li} = 1 \ (i = \overline{1,N}), \quad (6)$$

$$\sum_{i=1}^{N} d_i \cdot p_{im} \le v_m \ (m = \overline{1,M}). \quad (7)$$

Ограничение (5) предусматривает, что данные i-го типа размещаются только на одном из M устройств их хранения. Ограничение (6) предусматривает, что данные i-го типа назначаются только на одно из LВУ для их обработки. Ограничение (7) предусматривает, что количество данных разных типов, хранящихся на m-м устройстве, не превышает допустимого (заданного) объема хранилища.

Таким образом, математическая модель иерархической игры вида (3–7) представляет собой способ оптимизации решений по распределенному хранению и распределенной обработке данных, реализуемый в двухуровневой системе.

2. Генетический алгоритм оптимизации решений по распределенному хранению и обработке данных

На нижнем уровне иерархии системы оптимизации решений по распределенному хранению и обработке данных обеспечивается достижение внутренней цели функционирования, предусматривающей эффективное использование ресурсов (минимизацию временных затрат). Поэтому распределение обработки данных по ВУ реализуется с учетом решения по распределенному хранению данных таким образом, чтобы обеспечить минимизацию времени использования каналов передачи данных. Для минимизации времени хранения данных на устройствах расписание (порядок) их обработки формируется в соответствии с эвристическим алгоритмом, предусматривающим, что среди всех данных, закрепленных для выполнения на соответствующем ВУ, в первую очередь обрабатываются данные большого объема. Тем самым обеспечивается минимизация времени использования ресурса хранения. Минимизация использования ресурсов обработки и передачи данных гарантируется видом сформированного критерия.

С целью оптимизации решений по распределенному хранению и обработке данных применен аппарат генетических алгоритмов [15, 16]. Реализация генетических алгоритмов для оптимизации решений на соответствующих уровнях иерархии предусматривает первоначальную разработку способа их (решений) кодирования и последующую разработку генетических операторов, которые позволяют получить локально-оптимальные решения на соответствующих уровнях иерархической игры.

Кодирование решений по распределенной обработке данных с целью их оптимизации на нижнем уровне иерархической игры выполняется следующим образом. Одному решению по распределенной обработке данных соответствует хромосома, состоящая из n генов (где n – количество типов данных, обрабатываемых в системе). Каждый i-й ген $(i = \overline{1,N})$ соответствует закреплению данных i-го типа для обработки за одним из L ВУ. Обозначим значение i-го гена через l_i $(i = \overline{1,N})$, тогда при формировании начального решения (хромосомы) выполняется инициализация $l_i = l$, где l – номер ВУ, на котором выполняется обработка данных i-го типа $(l = \overline{1,L})$.

Для кодирования решений по распределенному хранению данных с целью их оптимизации на верхнем уровне иерархической игры каждому i-му гену $(i = \overline{1,N})$ ставится в соответствие номер устройства m $(m = \overline{1,M})$, на котором размещаются данные i-го типа. Значение i-го гена $(i = \overline{1,N})$ обозначим через m_i, тогда при формировании некоторого решения (хромосомы) выполняется инициализация $m_i = m$, где m – номер устройства $(m = \overline{1,M})$, на котором выполняется хранение данных i-го типа $(i = \overline{1,N})$.

Популяция (множество) хромосом, соответствующих решениям по размещению данных на устройствах хранения, имеет вид:

$$\mathrm{H}_m = \{h_{m_u} | u = \overline{1,U}\},$$

где U – количество хромосом в популяции; h_{m_u} – u-я хромосома, представленная в виде вектора $h_{m_u} = (m_1^u, m_2^u, .., m_n^u)$; m_i^u – значение i-го гена $(i = \overline{1,N})$ в u-й хромосоме $(u = \overline{1,U})$, при условии, что $m_i^u \in \{1,2,\dots,M\}$.

Популяция (множество) хромосом, соответствующих решениям по назначению обработки данных на соответствующих ВУ, имеет вид:

$$\mathrm{H}_l = \{h_{l_u} | u = \overline{1,U}\},$$

где h_{l_u} – u-я хромосома, представленная в виде вектора $h_{l_u} = (l_1^u, l_2^u, .., l_n^u)$; l_i^u – значение i-го гена $(i = \overline{1,N})$ в u-й хромосоме $(u = \overline{1,U})$, такое, что $l_i^u \in \{1,2,\dots,L\}$.

В соответствии с представленным способом кодирования решений формируется начальная популяция путем генерации значений соответствующих генов разных хромосом (при условии выполнения ограничения (7)). Дальнейшая оптимизация решений предусматривает их изменение на каждом из уровней иерархии в иерархической игре в соответствии с разработанными генетическими операторами. Для поиска локально оптимальных решений по распределенному хранению и обработке данных использованы следующие генетические операторы:

– оператор селекции с целью выбора родительских хромосом для их последующего скрещивания;

– оператор кроссоверинга (скрещивания), позволяющий получать новые хромосомы (решения) путем наследования свойств родительских хромосом (решений);

– оператор мутации, позволяющий трансформировать решения, полученные в результате скрещивания, случайным образом.

В качестве оператора селекции родительских хромосом используется метод, основанный на принципе колеса рулетки. Так как на каждом из уровней в иерархический игре реализуется минимизация соответствующих этим уровням критериев, поэтому разработан способ определения сектора колеса рулетки, соответствующего рассматриваемому решению (способ определения величины подинтервала в интервале [0;1]), который ставится в соответствие рассматриваемой хромосоме (решению) при определении возможности ее скрещивания с другими хромосомами.

Для каждой хромосомы $h_{m_u} \in H_m$ (аналогично для хромосомы $h_{l_u} \in H_l$) первоначально определяется ее оценка $O_{m_u}^1$, используемая в дальнейшем при вычислении величины сектора, в соответствии с выражением вида:

$$O_{m_u}^1 = \frac{f_1^u}{\sum_{u=1}^{U} f_1^u},$$

для хромосом, соответствующих решениям по распределению данных на ВУ для обработки:

$$O_{l_u}^1 = \frac{f_2^u}{\sum_{u=1}^{U} f_2^u},$$

где f_1^u и f_2^u – значения критериев f_1 и f_2 на верхнем и нижнем уровнях иерархической игры для некоторых u-х решений (u-х хромосом, входящих в популяции)).

На основе значений $O_{m_u}^1$ (для решений на верхнем уровне) и $O_{l_u}^1$ (для решений на нижнем уровне) определяются значения для оценок $O_{m_u}^2$ и $O_{l_u}^2$ u-х хромосом $(u = \overline{1,U})$ в соответствии с выражениями вида:

$$O_{m_u}^2 = \frac{1}{1 + O_{m_u}^1};\ O_{l_u}^2 = \frac{1}{1 + O_{l_u}^1}.$$

Оценка P_{m_u} и и P_{l_u} величины сектора колеса рулетки для u-х хромосом $(u = \overline{1,U})$ на верхнем и нижнем уровнях иерархической игры определяются в соответствии с выражениями вида:

$$P_{m_u} = \frac{O_{m_u}^2}{\sum_{u=1}^{U} O_{m_u}^2};\ P_{l_u} = \frac{O_{l_u}^2}{\sum_{u=1}^{U} O_{l_u}^2}.$$

Вычисленные с использованием полученных выражений значения величин сектора колеса рулетки используются для определения размеров подинтервалов в интервале [0;1], закрепляемых за соответствующими хромосомами. Генерация случайных чисел и определение их принадлежности соответствующим подинтервалам позволяют идентифицировать родительские хромосомы, используемые для скрещивания. После формирования множества родительских хромосом, используемых для скрещивания, в нем необходимо идентифицировать пары, к которым будет применен оператор кроссоверинга. Способами выбора пар родительских хромосом в соответствующем множестве для их непосредственного скрещивания (применения оператора кроссоверинга) являются [15, 16]:

а) случайный выбор – пары родительских хромосом выбираются случайным образом;

б) селективный выбор – выбираются такие хромосомы, которым соответствуют значения критерия оптимальности решений выше среднего значения для всей популяции;

в) инбридинг, при котором первая родительская хромосома выбирается случайным образом, а вторая – исходя из максимальной похожести на первую;

г) аутбридинг, при котором первая родительская хромосома выбирается случайно, а вторая – исходя из максимального отличия от первой.

С целью исключения возможности преждевременного попадания в «ловушку» локального экстремума для выбора пар родительских хромосом использован генотипный аутбридинг.

Идентификация различия между текущей рассматриваемой хромосомой $h_{m_u} \in H_m (h_{l_u} \in H_l$ в популяции $H_l)$, выбираемой случайным образом, и хромосомой $h_{m_j} \in H_m$, с которой будет выполняться скрещивание, определяется в соответствии с условием следующего вида:

$$\max_j \sum_{i=1}^{N} \left| m_i^u - m_i^j \right| ; \ \max_j \sum_{i=1}^{N} \left| l_i^u - l_i^j \right|,$$

где $j = \overline{1, u-1} \& j = \overline{u+1, U})$.

Представленное условие определяет максимальное различие в генах в хромосоме, выбранной случайно, и в хромосоме, с ней скрещиваемой.

Для выбранных таким образом пар хромосом случайным образом определяются 2 точки скрещивания (номера генов в хромосомах, являющихся граничными при реализации скрещивания). Индексы (номера) генов, являющихся границами участков хромосом при скрещивании обозначены через q_1 (первая точка) и q_2 (вторая точка). Таким образом, в генах с определенными таким образом номерами реализуется двухточечное скрещивание пары родительских хромосом.

Если родительские хромосомы имеют вид:

$$h_{m_u} = (m_1^u, m_2^u, \dots, m_{q_1-1}^u, m_{q_1}^u, m_{q_1+1}^u, \dots, \dots m_{q_2-1}^u, m_{q_2}^u, m_{q_2+1}^u, \dots, m_n^u)$$

и

$$h_{m_j} = (m_1^j, m_2^j, \dots, m_{q_1-1}^j, m_{q_1}^j, m_{q_1+1}^j, \dots, \dots, m_{q_2-1}^j, m_{q_2}^j, m_{q_2+1}^j, \dots, m_n^j),$$

то в результате реализации оператора двухточечного кроссоверинга пара дочерних хромосом будет иметь вид:

$$h'_{m_u} = (m_1^u, m_2^u, \dots, m_{q_1-1}^u, m_{q_1}^j, m_{q_1+1}^j, \dots, \dots, m_{q_1-1}^j, m_{q_1}^u, m_{q_1+1}^u, \dots, m_n^u)$$

и

$$h'_{m_j} = (m_1^j, m_2^j, \dots, m_{q_1-1}^j, m_{q_1}^u, m_{q_1+1}^u, \dots, \dots, m_{q_1-1}^u, m_{q_1}^j, m_{q_1+1}^j, \dots, m_n^j).$$

Аналогичные рассуждения применены при скрещивании хромосом h_{l_u} и h_{l_j} на нижнем уровне иерархии игры при оптимизации решений по назначению ВУ для обработки данных n типов.

Полученные описанным способом дочерние хромосомы подвергаются на следующем этапе воздействию оператора мутации. Стохастический характер оператора мутации предусматривает, что каждый ген будет подвержен ее (мутации) воздействию с определенной вероятностью (обозначенной как $P_м$). Воздействие оператора мутации на ген предусматривает либо увеличение на 1 его значения, либо уменьшения на 1 его значения. Характер операции (увеличение либо уменьшение значений генов) также определяется стохастически.

Синтезированный генетический алгоритм используется для поиска локально оптимального решения по закреплению обработки данных i-х типов ($i = \overline{1, N}$) за l-ми ВУ ($l = \overline{1, L}$) (лучшей хромосомы в сформированной финальной популяции, сгенерированной для текущего решения по распределенному хранению данных). То есть это локально оптимальное решение соответствует текущему решению по распределенному хранению данных, полученному с верхнего уровня. Оно передается на верхний уровень с целью оценки оптимальности решения по распределенному хранению данных. Для решений по распределенному хранению данных на верхнем уровне (хромосом, входящих в популяцию, рассматриваемую на верхнем уровне) реализуется повторное формирование новых решений с использованием рассмотренных выше генетических операторов (формирование новой популяции хромосом, соответствующих решениям по распределенному хранению данных).

Поиск локально оптимальных решений по распределенному хранению данных и их распреде-

ленной обработке реализуется для заданного количества поколений популяций. После чего в финальной популяции на верхнем уровне выбирается лучшее (локально оптимальное) решение (с минимальным значением критерия на этом уровне), а на нижнем уровне идентифицируется лучшее решение в популяции, соответствующей этому локально оптимальному решению на верхнем уровне.

Оптимизация решений по распределенному хранению и обработке данных (на верхнем и нижнем уровне иерархической игры, соответственно) выполняется при следующих параметрах генетических алгоритмов: общее количество хромосом в популяции U = 60; разрыв поколений T = 0,5 (количество наиболее приспособленных хромосом, участвующих в селекции), количество поколений равно 30. Вероятность мутации генов определяется по формуле $P_м = 1/(10 \cdot N)$, где N – количество типов данных, распределенное хранение и обработка которых реализуется в системе.

3. Эвристический алгоритм построения расписаний обработки данных на ВУ

Поиск локально оптимальных решений по распределенному хранению и обработке данных реализуется при условии минимизации стоимости использования ресурсов (в том числе минимизации стоимости передачи данных по каналам, связывающим устройства хранения и обработки). С целью минимизации интервалов времени хранения данных на m-х устройствах (и как следствие – минимизации стоимости хранения) предложен эвристический алгоритм, позволяющий формировать расписания обработки данных на назначенных для этого ВУ. Таким образом, построение расписаний обработки данных на ВУ, на которых она назначена, реализуется с учетом требования минимизации времени нахождения данных в хранилищах, что обеспечивает минимизацию стоимости хранения. Эвристический алгоритм построения расписаний предусматривает, что среди всех назначенных на устройство для обработки данных в первую очередь будут обрабатываться те, размер которых является максимальным (то есть обработка данных на устройстве осуществляется в порядке не убывания их объемов $d_i (i = \overline{1,N})$).

Для обоснования эвристической процедуры формирования порядков реализации распределенной обработки данных на каждом из ВУ в рассмотрение введены:

1) множества $N^l (l = \overline{1,L})$ типов данных, назначенных для выполнения указанной операции на эти устройства (множества формируются в соответствии с видом матрицы R с учетом значений ее элементов $r_{li} = 1$ в каждой l-й строке $(l = \overline{1,L})$);

2) переменные $Pred^l$ $(l = \overline{1,L})$, предназначенные для хранения идентификатора типа данных, которые рассматривались на предшествующей итерации алгоритма и для которых было вычислено значение t_{il}^0 элемента матрицы T^0, соответствующей расписаниям обработки данных на l-х приборах (первоначально переменные инициализируются значением 0).

Определение значений t_{il}^0 $(i = \overline{1,N}, l = \overline{1,L})$ моментов времени начала выполнения операций с данными i-х типов на l-х ВУ (то есть построение расписаний обработки данных на l-х устройствах) реализуется в предположении, что к моменту времени начала интерпретации расписания данные всех N типов уже распределены по устройствам хранения. В том случае, если данные некоторых i-х типов занимают $(j = 1)$-е позиции в последовательностях их обработки на l-х приборах (значения переменных $Pred^l = 0 (l = \overline{1,L})$), тогда для них предполагается, что $t_{im}^{mem} = 0$. В этом случае момент времени начала обработки данных определяется на основе интервала времени их передачи между m-м устройством, где они размещены, и l-м ВУ, на котором они обрабатываются (определяется интервалом времени выставления данных в канал и интервалом времени передачи данных по каналу).

Вычисление значения t_{il}^0 для данных i-го типа в $(j = 1)$-й позиции их обработки на l-м приборе реализуется в соответствии с выражением:

$$t_{il}^0 = \sum_{m=1}^{M} p_{im} r_{li} \left(\frac{d_i}{c_{ml}} + \frac{b_{ml}}{S_{tr}} \right). \quad (8)$$

В том случае, если для данных i-го типа их позиция в последовательности обработки на l-ом приборе $j \neq 1$, то значение t_{il}^0 определяется выражением:

$$t_{il}^0 = t_{i'l}^0 + \sum_{l=1}^{L} t_{i'l} r_{li'}, \quad (9)$$

где i' – тип данных, которые занимают в последовательности их обработки на l-м приборе предшествующую $(j-1)$-ю позицию по отношению к j-й позиции данных рассматриваемого i-го типа (номер типа данных использован на предыдущей итерации алгоритма построения расписания для инициализации переменной $Pred^l$). Перед началом интерпретации процедуры построения расписаний и определения значений t_{il}^0 для данных i-х типов, закрепленных за соответствующими приборами, элементы матрицы T^0 инициализированы значением 0.

Эвристический алгоритм построения расписания обработки данных, назначенных на l-е устройство, а также определения моментов времени t_{il}^0

начала обработки данных i-х типов на этом устройстве, имеет следующий порядок шагов.

Шаг 1. Переменную $\mathrm{Pr}ed^l$ инициализировать значением 0.

Шаг 2. Определить в множестве N^l тип данных i', которому соответствует условие: $\max_i(d_i | \ i \in N^l)$.

Шаг 3. Если $\mathrm{Pr}ed^l = 0$, то для рассматриваемого i'-го типа данных определить значение $t^0_{i'l}$ с использованием выражения (8); если $\mathrm{Pr}ed^l \neq 0$, то для рассматриваемого i'-го типа данных определить значение $t^0_{i'l}$ с использованием выражения (9), с учетом данных i-го типа, идентификатором которых инициализирована переменная $\mathrm{Pr}ed^l$ на предыдущей итерации алгоритма, с использованием значения t^0_{il}, определяемого для данных этого типа в матрице T^0).

Шаг 4. Модифицировать множество N^l и значение переменной $\mathrm{Pr}ed^l$: $N^l = N^l \backslash \{i'\}$; $\mathrm{Pr}ed^l = i'$.

Шаг 5. Если $N^l \neq \emptyset$, то перейти на пункт 2; если $N^l = \emptyset$, то перейти на пункт 6.

Шаг 6. Останов алгоритма.

Результатом интерпретации рассмотренного алгоритма являются последовательности выполнения заданий на l-х приборах, которым соответствует матрица T^0 моментов времени t^0_{il} начала обработки данных i-х типов на этих приборах. Матрица T^0 совместно с матрицей назначений R передается на верхний уровень с целью вычисления на их основе оценки критерия f_1 текущего рассматриваемого решения $[P, T^{mem}]$ по распределению данных по хранилищам.

4. Анализ эффективности применения метода многоуровневой оптимизации решений по распределенному хранению и обработке данных

С целью исследования эффективности применения предложенного метода многоуровневой оптимизации решений по распределенном хранению и обработке данных значения параметров задачи задаются в соответствии с следующими обозначениями:

$\min(d_i)$ – минимальный объем данных i-х типов ($i = \overline{1, N}$), обрабатываемых в системе;

$\max(d_i)$ – максимальный объем данных i-х типов ($i = \overline{1, N}$), обрабатываемых в системе;

$\max(d_i)/\min(d_i)$ – отношение максимального объема данных i-х типов ($i = \overline{1, N}$) к минимальному объему данных i-х типов ($i = \overline{1, N}$), обрабатываемых в системе, которое определяет неоднородность объемов данных различных типов, распределяемых по устройствам хранения;

$\min(t_{il})$ – минимальная длительность обработки данных i-х типов ($i = \overline{1, N}$) на l-х ВУ ($l = \overline{1, L}$);

$\max(t_{il})$ – максимальная длительность обработки данных i-х типов ($i = \overline{1, N}$) на l-х ВУ ($l = \overline{1, L}$);

$\max(t_{il})/\min(t_{il})$ – отношение максимальной длительности обработки данных i-х типов ($i = \overline{1, N}$) на l-х ВУ ($l = \overline{1, L}$) к минимальной, определяющее неоднородность длительностей обработки данных на различных ВУ.

При проведении исследований значения $\min(d_i)$ и $\min(t_{il})$ задавались равными 10. Также при проведении исследований значения отношений $\max(d_i)/\min(d_i)$ и $\max(t_{il})/\min(t_{il})$ задавались равными: 1, 2, 4, 8, 16. Исследования проводились при фиксированных значениях длин каналов передачи данных между устройствами хранения и обработки, а также их пропускных способностей.

С целью анализа эффективности применения метода многоуровневой оптимизации решений по распределенному хранению и обработке данных использован эвристический алгоритм SIM упаковки в контейнеры [17]; метод формируется путем оптимизации с использованием рассмотренного генетического алгоритма. Эффективность применения метода многоуровневой оптимизации характеризуется степенью снижения затрат на хранение, обработку, передачу данных и штрафов за неиспользованные ресурсы для решений, полученного с использованием эвристического алгоритма SIM, и для решений, полученных путем оптимизации.

Снижение суммарных затрат при использовании метода многоуровневой оптимизации оценивалась путем сравнения:

1) значения критерия f_1 на верхнем уровне, обозначенного как f_1^{SIM}, полученного для решения, сформированного с использованием эвристического алгоритма SIM упаковки в контейнеры [17], и соответствующего ему локально оптимального решения по распределенной обработке данных;

2) значения критерия f_1 на верхнем уровне, полученного после многоуровневой оптимизации (обозначенного как f_1^o).

Снижение затрат на хранение, обработку, передачу данных и штрафов при оптимизации решений определено в соответствии с выражением:

$$\frac{f_1^{SIM} - f_1^o}{f_1^{SIM}}.$$

Снижение суммарных затрат на хранение, обработку, передачу данных и штрафов, получаемое при использовании метода многоуровневой оптимизации, представлено на рисунке 1 (слева M = 5, L = 5; справа M = 10, L = 10).

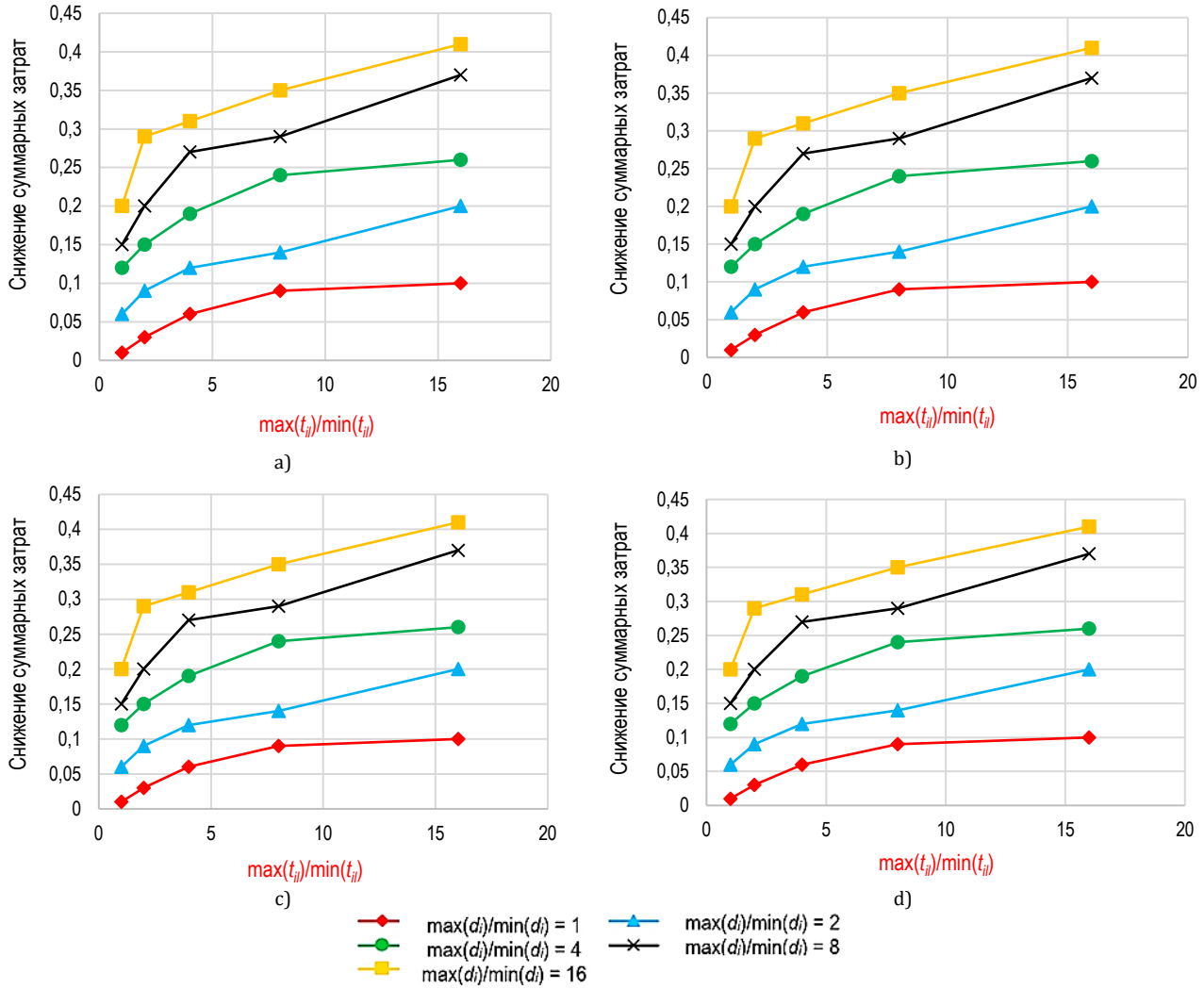

Рис. 1. Снижение суммарных затрат на хранение, обработку, передачу данных и штрафов для решений, полученных с использованием алгоритма SIM и метода многоуровневой оптимизации для n = 50 (a, c) и для n = 100 (b, d)

Fig. 1. Reduction of Total Costs for Storage, Processing, Data Transmission and Penalties for Solutions Obtained Using the SIM Algorithm and the Multilevel Optimization Method for n = 50 (a, c) and for n = 100 (b, d)

Анализ результатов исследований позволил сделать следующие выводы, касающиеся применения предложенного метода многоуровневой оптимизации решений по распределенному хранению и обработке данных:

1) увеличение количества устройств хранения M и обработки данных L (при неизменном количестве типов данных n) обуславливает снижение эффективности применении разработанного метода;

2) увеличение количества типов данных n, для которых реализуется распределенное хранение и обработка (при неизменном количестве ВУ) обуславливает увеличение эффективности применения разработанного метода;

3) увеличение неоднородности объемов хранимых данных (значения $\max(d_i)/\min(d_i)$) обуславливает более значительный рост эффективности применения метода по сравнению с увеличением неоднородности длительностей обработки данных на устройствах (значения $\max(t_{il})/\min(t_{il})$).

Заключение

В работе решена задача математического моделирования и оптимизации процессов распределенного хранения и обработки данных с учетом ограничений на ресурсы и взаимодействия между устройствами. Разработана иерархическая модель процесса распределенного хранения и обработки данных, представляющая собой совокупность компонент на соответствующих уровнях. Компонента модели верхнего уровня соответствует составляющей процесса, связанной с распределенным хранением данных, компонента нижнего уровня – с составляющей процесса, связанной с распределенной их обработкой вычислительными устройствами.

Для определения эффективного размещения данных с целью их распределенного хранения и обработки применена декомпозиция обобщенной задачи оптимизации на совокупность иерархически упорядоченных подзадач, для каждой из кото-

рых на соответствующем ей уровне идентифицируется локально-оптимальное решение. На основе определенной таким образом совокупности иерархически упорядоченных подзадач разработана математическая модель иерархической игры идентификации локально оптимальных решений на соответствующих уровнях. Предложенная модель игры позволяет реализовать совместную оптимизацию размещения данных при их распределенном хранении (на верхнем уровне) и их закрепления за вычислительными устройствами для распределенной обработки (на нижнем уровне). Критерий на верхнем уровне соответствует суммарным затратам на хранение, обработку, передачу данных, а также штрафы за неполное использование ограниченных ресурсов хранения. Критерий на нижнем уровне соответствует суммарным длительностям передачи данных по каналам между устройствами хранения и обработки, а также их обработки на соответствующих вычислительных устройствах. Особенностью математической модели процесса распределенного хранения и обработки данных, модели иерархической игры оптимизации решений является учет длительностей интервалов времени их (данных) пребывания на устройствах хранения. Эта характеристика процесса определяемых в соответствии с сформированными на нижнем уровне расписаниями обработки, а также длительностями интервалов времени выставления данных в каналы и их передачи по каналам.

В качестве способа оптимизации решений по распределенному хранению и обработке данных на каждом из уровней иерархии использованы генетические алгоритмы. Выполнена разработка способов кодирования решений на каждом из уровней, а также генетических операторов, позволяющих осуществлять поиск локально оптимальных решений.

С целью построения расписаний обработки данных, закрепленных за соответствующими вычислительными устройствами, разработан эвристический алгоритм, предусматривающий упорядочивание обработки данных с точки зрения не убывания их объема. Использование разработанного эвристического алгоритма построения расписаний обработки данных на устройствах обеспечивает минимизацию интервалов времени их нахождения на устройствах хранения и, как следствие, минимизацию затрат на хранение. Выполненные исследования эффективности применения разработанного метода многоуровневой оптимизации решений по распределенным хранению и обработки данных показали, что его использование позволяет на 10–60 % снизить суммарные финансовые затраты на выполнения указанных операций, а также на операции передачи данных и штрафы за неполное использование ограниченных ресурсов.

Практическая ценность полученных результатов состоит в разработке алгоритмов оптимизации решений по распределенному хранению и обработке данных, которые могут быть непосредственно применены при планировании вычислений в кластерных и облачных системах.

Список источников

1. Prajapati H.B., Shah V.A. Scheduling in Grid Computing Environment // Proceedings of the Fourth International Conference on Advanced Computing & Communication Technologies (Rohtak, India, 08–09 February 2014). IEEE, 2014. PP. 315–324. DOI:10.1109/ACCT.2014.32

2. Bhatia M.K. Task Scheduling in Grid Computing: A Review // Advances in Computational Sciences and Technology. 2017. Vol. 10. Iss. 6. PP. 1707–1714.

3. Xhafa F., Barolli L., DurresiA. Batch mode scheduling in grid systems // International Journal of Web and Grid Services. 2007. Vol. 3. Iss. 1. PP. 19–37. DOI:10.1504/IJWGS.2007.012635

4. Khan M. Design and Analysis of Security Aware Scheduling in Grid Computing Environment // International Journal of Computer Science and Information Technology Research (IJCSITR). 2013. Vol. 1.Iss. 1. PP. 42–50.

5. Naresh U. Study on Many-Task-Computing using Data Aware Scheduling in Cloud Computing // International Journal of Innovations & Advancement in Computer Science (IJIACS). 2017. Vol. 6. Iss. 9. PP. 360–366.

6. Mahajan S., Kaur R. A Concern towards Job scheduling in Cluster Computing // International Journal of Computer Engineering in Research Trends. 2015. Vol. 2. Iss. 6. PP. 392–394.

7. Abawajy J.H. Dynamic Parallel Job Scheduling in Multi-cluster Computing Systems // Proceedings of the 4th International Conference of Computer Science (ICCS 2004, Kraków, Poland, 6–9 June 2004). Lecture Notes in Computer Science. Vol. 3036. Berlin, Heidelberg: Springer, 2004. PP. 27–34. DOI:10.1007/978-3-540-24685-5_4

8. Alworafi M.A., Dhari A., El-Booz Sh.A., Mallappa S. Budget-aware task scheduling technique for efficient management of cloud resources // International Journal High Performance Computing and Networking. 2019. Vol. 14. Iss. 4. PP. 453–465. DOI:10.1504/IJHPCN.2019.102352

9. Arabnejad V., Bubendorfer K., Ng B. Budget and Deadline Aware e-Science Workflow Scheduling in Clouds // IEEE Transactions on Parallel and Distributed Systems. 2019. Vol. 30. Iss. 1. PP. 29–44. DOI:10.1109/TPDS.2018.2849396

10. Месарович М., Мако Д., Такахара И. Теория иерархических многоуровневых систем. М.: Из-во «Мир», 1973. 344 с.

11. Воронин А.А., Мишин С.П. Оптимальные иерархические структуры. М.: ИПУ РАН, 2003. 214 с.

12. Губко М.В., Новиков Д.А. Теория игр в управлении организационными системами. М.: Институт проблем управления им. В.А. Трапезникова, 2005. 138 с.

13. Бурков В.Н., Коргин Н.А., Новиков Д.А. Введение в теорию управления организационными системами. М.: Либроком, 2009. 264 с.
14. Бусыгин В.П., Желободько Е.В., Коковин С. Г., Цыплаков А.А. Микроэкономический анализ несовершенных рынков. Новосибирск: Новосиб. гос. ун-т, 1999. 132 с.
15. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы. М.: Физматлит, 2006. 320 с.
16. Курейчик В.М. Генетические алгоритмы и их применение. Таганрог: Таганрогское РТУ, 2002. 244 с.
17. Смирнов А.В. О задаче упаковки в контейнеры // Успехи математических наук.1991. Т. 46. № 4. С. 173–174.

References

1. Prajapati H.B., Shah V.A. Scheduling in Grid Computing Environment. *Proceedings of the Fourth International Conference on Advanced Computing & Communication Technologie, 08–09 February 2014s, Rohtak, India*. IEEE; 2014. p.315–324. DOI:10.1109/ACCT.2014.32
2. Bhatia M.K. Task Scheduling in Grid Computing: A Review. *Advances in Computational Sciences and Technology*. 2017;10(6):1707–1714.
3. Xhafa F., Barolli L., Durresi A. Batch mode scheduling in grid systems. *International Journal of Web and Grid Services*. 2007;3(1):19–37. DOI:10.1504/IJWGS.2007.012635
4. Khan M. Design and Analysis of Security Aware Scheduling in Grid Computing Environment. *International Journal of Computer Science and Information Technology Research (IJCSITR)*. 2013;1(1):42–50.
5. Naresh U. Study on Many-Task-Computing using Data Aware Scheduling in Cloud Computing. *International Journal of Innovations & Advancement in Computer Science (IJIACS).* 2017;6(9):360–366.
6. Mahajan S., Kaur R. A Concern towards Job scheduling in Cluster Computing. *International Journal of Computer Engineering in Research Trends.* 2015;2(6):392–394.
7. Abawajy J.H. Dynamic Parallel Job Scheduling in Multi-cluster Computing Systems. *Proceedings of the 4th International Conference of Computer Science, ICCS 2004, 6–9 June 2004, Kraków, Poland. Lecture Notes in Computer Science.* Berlin, Heidelberg: Springer; 2004. *vol.3036*. p. 27–34. DOI:10.1007/978-3-540-24685-5_4
8. Alworafi M.A., Dhari A., El-Booz Sh.A., Mallappa S. Budget-aware task scheduling technique for efficient management of cloud resources. *International Journal High Performance Computing and Networking*. 2019;14(4):453–465. DOI:10.1504/IJHPCN.2019.102352
9. Arabnejad V., Bubendorfer K., Ng B. Budget and Deadline Aware e-Science Workflow Scheduling in Clouds. *IEEE Transactions on Parallel and Distributed Systems*. 2019;30(1):29–44. DOI:10.1109/TPDS.2018.2849396
10. Mesarovich M., Mako D., Takahara I. *Theory of Hierarchical Multilevel Systems*. Moscow: Mir Publ.; 1973. 344 p. (in Russ.)
11. Voronin A.A., Mishin S.P. *Optimal Hierarchical Structures*. Moscow: V.A. Trapeznikov Institute of Management Problems Publ.; 2003. 214 p. (in Russ.)
12. Gubko M.V., Novikov D.A. *Game Theory in the Management of Organizational Systems.* Moscow: V.A. Trapeznikov Institute of Management Problems Publ.; 2005. 138 p. (in Russ.)
13. Burkov V.N., Korgin N.A., Novikov D.A. *Introduction to the Theory of Management of Organizational Systems*. Moscow: Librocom Publ.; 2009. 264 p.(in Russ.)
14. Busygin V.P., Zhelobodko E.V., Kokovin S.G., Tsyplakov A.A. *Microeconomic Analysis of Imperfect Markets*. Novosibirsk: Novosibirsk State University Publ.; 1999. 132 p. (in Russ.)
15. Gladkov L.A., Kureychik V.V., Kureychik V.M. *Genetic Algorithms*. M.: Fizmatlit Publ.; 2006. 320 p. (in Russ.)
16. Kureychik V.M. *Genetic Algorithms and Their Application*. Taganrog: Taganrog Radio Engineering University Publ.; 2002. 244 p. (in Russ.)
17. Smirnov A.V. On the Problem of Packaging in Containers. *Successes of Mathematical Sciences.* 1991;46(4):173–174. (in Russ.)

Статья поступила в редакцию 28.11.2022; одобрена после рецензирования 23.01.2023; принята к публикации 26.01.2023.
The article was submitted 28.11.2022; approved after reviewing 23.01.2023; accepted for publication 26.01.2023.

Информация об авторе:

КРОТОВ Кирилл Викторович
доктор технических наук, доцент, доцент кафедры «Информационные системы» Севастопольского государственного университета
https://orcid.org/0000-0002-9670-6141