На правах рукописи

Потапов Данила Романович

РАЗРАБОТКА И ИССЛЕДОВАНИЕ МЕТОДА И АЛГОРИТМОВ АДАПТАЦИИ АССОЦИАТИВНОГО КОНТЕЙНЕРА ДАННЫХ

Специальность 05.13.17 – теоретические основы информатики

АВТОРЕФЕРАТ

диссертации на соискание ученой степени
кандидата физико-математических наук

Воронеж – 2021

Работа выполнена в федеральном государственном бюджетном образовательном учреждении высшего образования «Воронежский государственный университет»

Научный руководитель:

Доктор физико-математических наук, профессор,
Артемов Михаил Анатольевич

Официальные оппоненты:

Добрица Вячеслав Порфирьевич, доктор физико-математических наук, профессор, федеральное государственное бюджетное образовательное учреждение высшего образования «Юго-Западный государственный университет», факультет фундаментальной и прикладной информатики, кафедра информационной безопасности, профессор

Горбунов Вячеслав Алексеевич, доктор физико-математических наук, профессор, федеральное государственное бюджетное образовательное учреждение высшего образования «Вологодский государственный университет», институт математики, естественных и компьютерных наук, кафедра автоматики и вычислительной техники, профессор

Ведущая организация:

Федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева»

Защита состоится «22» сентября 2021 г. в 15 ч. 10 мин. на заседании диссертационного совета Д 212.038.20 при ФГБОУ ВО «Воронежский государственный университет» по адресу: 394018, г.Воронеж, Университетская пл., 1, главный корпус, ауд. 333.

С диссертацией и авторефератом можно ознакомиться в научной библиотеке Воронежского государственного университета, а также на сайте http://www.science.vsu.ru/dissertations/9717/Диссертация_Потапов_Д.Р..pdf.

Автореферат разослан « » _____ 20__ г.

Ученый секретарь
диссертационного
совета Д 212.038.20,
доктор физико-
математических наук

С.А. Шабров

Введение

Актуальность работы

Во многих задачах обработки и хранения информации явно или неявно требуются эффективно работающие контейнеры данных (структуры данных). Существует много различных типов таких контейнеров, однако каждый из них является оптимальным только для ограниченного числа задач. При этом зачастую условия задачи меняются в процессе работы системы и, следовательно, для максимально эффективной работы системы периодически необходима смена типа контейнера. Таким образом, возникает необходимость создания самоадаптирующегося ассоциативного контейнера данных, который меняет логику своей работы в зависимости от нагрузки. Например, при большом количестве операций поиска контейнер работает на основе одной структуры данных, при большом количестве вставок – на основе другой, при определенном соотношении операций вставки, поиска и удаления – на основе третьей и т.д. Помимо обеспечения максимальной скорости работы должна использоваться та структура данных, которая занимает минимум дополнительной памяти.

Одним из очевидных случаев использования самоадаптирующихся контейнеров являются мобильные устройства, такие как планшетные ПК, смартфоны, мобильные телефоны, ноутбуки и др. Такие устройства в силу своих размеров имеют ограниченные ресурсы, существенно уступающие стационарным компьютерам. Ограниченными являются в первую очередь оперативная память и производительность процессора. Поэтому и появляется необходимость в адаптивных контейнерах данных. При изменяющейся нагрузке выбирается контейнер, оптимальный либо с точки зрения быстродействия, либо – минимальных затрат оперативной памяти.

Еще одним примером необходимости использования адаптивного контейнера данных является индекс базы данных. Данный объект используется для повышения эффективности поиска в базе данных, а, следовательно, для этого применяются структуры данных, оптимизированные для поиска. Однако перестроение таких контейнеров при большом количестве вставок требует много времени. Таким образом, использование различных структур данных в зависимости от нагрузки позволит увеличить эффективность индексов в базах данных.

Однако наибольший прирост производительности при использовании контейнеров, которые изменяют свое поведение при изменении условий работы или нагрузки, можно получить в условиях Highload (высоконагруженные системы), NoSQL (различные реализации хранилищ баз данных, которые существенно отличаются от реляционных и имеют возможность линейно масштабироваться) и Bigdata (структурированные и неструктурированные данные огромных объемов и методы работы с ними), т. к. именно в этих областях присутствует большое количество данных и большие нагрузки. В этом случае помимо основного контейнера зачастую используется такой метод оптимизации как кэширование. Таким образом, помимо смены основного контейнера, самоадаптирующийся контейнер данных может изменять такие параметры кэша как алгоритм вытеснения или размер кэша.

Данное исследование является актуальным, так как в настоящее время не существует универсального контейнера и алгоритма кэширования, в связи с чем существует необходимость динамически выбирать оптимальную структуру данных и алгоритм кэширования для каждого конкретного случая на основе различных критериев.

Степень разработанности темы

Проблемой оптимальной структуры данных занимались многие авторы. Алгоритмы и реализации одномерных контейнеров описаны в работах Г. Адельсон-Вельского, Е. Ландиса, Aragon C. R., Seidel R., Fredman M. L., Tarjan R. E. и др. Многомерные структуры данных

описаны в исследованиях Гулакова В.К., Трубакова А.О., Kriegel H., Seeger B., Bentley J. L., Six H.-W., Bentley J.L. и многих других авторов.

В этих исследованиях рассматриваются различные структуры данных, но ни в одном из исследований не рассмотрен универсальный контейнер данных, который был бы максимально эффективен в любых условиях и для любых нагрузок. Следовательно, разработка и реализация методов и алгоритмов самоадаптирующегося контейнера данных, является весьма актуальной задачей. Решению этой актуальной задачи посвящена данная диссертация.

Цель и задачи исследования

Цель данной работы состоит в разработке самоадаптирующегося контейнера данных. Для достижения этой цели необходимо решить следующие задачи:

1. Разработать метод адаптации ассоциативного контейнера данных.
2. Определить область применения существующих контейнеров данных и алгоритмов кэширования для реализации самоадаптирующегося контейнера данных.
3. Исследовать зависимость эффективности кэша от среднеквадратичного отклонения и соотношения скоростей хранилищ для нормального распределения.
4. Разработать и реализовать алгоритм определения параметров сложной нагрузки на самоадаптирующиеся контейнеры данных.
5. Разработать и реализовать алгоритм адаптивного кэширующего контейнера данных с использованием интервального статистического ряда.
6. Провести вычислительные эксперименты, подтверждающие эффективность предложенных алгоритмов.

Объект исследования – структуры данных и алгоритмы доступа к данным.
Предмет исследования – структура данных, которая изменяет логику работы в зависимости от нагрузки и условий работы.

Методология и методы исследования

В качестве теоретической и методологической основы диссертационного исследования использованы методы дискретной математики, системного анализа, математической статистики, теория разработки программного обеспечения.

Научная новизна

В диссертационной работе получены следующие результаты, отличающиеся научной новизной:

1. Предложен метод адаптации ассоциативного контейнера данных, позволяющий изменять структуру данных основного хранилища данных, алгоритм кэширования и размер кэша в зависимости от условий работы и нагрузки на контейнер.
2. Выявлена зависимость размера кэширующего контейнера от соотношения между скоростями работы кэша и основного хранилища и среднеквадратичного отклонения ключей при нормальном распределении, позволяющая определить минимальный по времени работы размер кэша.
3. Разработан алгоритм определения параметров сложной нагрузки, состоящей из смеси гауссовских распределений, отличающийся применением EM-алгоритма с использованием буфера.
4. Создан алгоритм адаптивного кэширующего контейнера данных с использованием интервального статистического ряда, позволяющий повысить процент попаданий в кэш.
5. На основе разработанных алгоритмов реализованы программные комплексы, позволяющие провести исследования их эффективности.

Степень достоверности результатов

Научные положения, теоретические выводы и практические рекомендации обоснованы корректным использованием математического аппарата и подтверждены вычислительными экспериментами на ЭВМ.

Теоретическая и практическая значимость

Теоретическая значимость заключается в создании метода и алгоритмов самоадаптирующегося контейнера данных и некоторых его модулей, а также в выявлении зависимости оптимального размера кэша от условий работы и распределения ключей. Практическая значимость заключается в программных реализациях разработанных в диссертации алгоритмов.

Апробация работы

Результаты работы были представлены в форме докладов на следующих конференциях: XX Международной конференции «Информатика: проблемы, методы, технологии», IPMT-2020 (Воронеж, 2020 г.), международных научно-технических конференциях «Актуальные проблемы прикладной математики, информатики и механики» (Воронеж, 2018 – 2019 гг.), XVI Международной научно-методической конференции: «Информатика: проблемы, методология, технологии» (Воронеж, 2016 г.), «Математическое и компьютерное моделирование, информационные технологии управления (МКМИТУ-2016)» (Воронеж, 2016 г.).

Публикации

Основные результаты работы опубликованы в 12 работах, 5 из которых опубликованы в рекомендуемых ВАК РФ рецензируемых научных изданиях. Кроме того, есть публикация за рубежом, проиндексированная в Scopus. Также автором было получено свидетельство о государственной регистрации программы для ЭВМ. Из совместных работ в диссертацию вошли только результаты, принадлежащие лично диссертанту.

Личный вклад автора

Все основные результаты, составившие диссертацию, получены автором лично. Соавторы публикаций по теме диссертации участвовали в обсуждении постановочной части решаемых задач и результатов вычислений по разработанным автором программам расчётов. В диссертации отсутствуют заимствованные материалы и результаты других авторов исследований, а когда по логике изложений такие сведения оказываются необходимыми, то они снабжены соответствующими ссылками на авторов и литературные источники.

Область исследования

Диссертационная работа соответствует следующим пунктам паспорта специальности 05.13.17 – Теоретические основы информатики:

1. Исследование, в том числе с помощью средств вычислительной техники, информационных процессов, информационных потребностей коллективных и индивидуальных пользователей.
2. Исследование информационных структур, разработка и анализ моделей информационных процессов и структур.
5. Разработка и исследование моделей и алгоритмов анализа данных, обнаружения закономерностей в данных и их извлечениях разработка и исследование методов и алгоритмов анализа текста, устной речи и изображений.

На защиту выносятся

1. Метод адаптации ассоциативного контейнера, позволяющий изменять структуру данных основного хранилища данных, алгоритм кэширования и размер кэша в зависимости от условий работы и нагрузки на контейнер.
2. Результат исследования зависимости оптимального размера кэша от соотношения скоростей хранилищ и среднеквадратичного отклонения ключей в нагрузке, определенной по нормальному распределению.
3. Алгоритм работы модуля определения параметров сложной нагрузки на контейнер, состоящей из смеси нормальных распределений.
4. Алгоритм работы адаптивного кэширующего контейнера данных с использованием интервального статистического ряда.
5. Программный комплекс, реализующий разработанные алгоритмы.

Структура и объём работы

Материал работы изложен на 137 страницах машинописного текста. Работа состоит из введения, четырех глав, выводов, списка литературы, содержит 33 рисунка, и 27 таблиц. Библиография включает 151 наименование.

Содержание работы

Во введении обоснована актуальность темы исследования, степень ее разработанности, приведены цель и задачи исследования, научная новизна, теоретическая и практическая значимость работы, методология и методы диссертационного исследования, положения, выносимые на защиту, степень достоверности и апробация результатов, а также основное содержание работы.

В первой главе представлен анализ и сравнение существующих методов построения контейнеров данных «ключ-значение» и методы их оптимизации (кэширование). Рассмотрены как одномерные, так и многомерные структуры данных. В результате проведенного в главе анализа выявлены основные ограничения, которые необходимо учитывать при реализации самоадаптирующихся ассоциативных контейнеров данных, основные сферы применения различных контейнеров при реализации самоадаптирующихся ассоциативных контейнеров данных, а также проанализированы существующие методы кэширования.

Вторая глава посвящена разработке способа задания самоадаптирующегося контейнера данных и его компонентов. Разработана структурная модель самоадаптирующегося контейнера, способ задания ассоциативного контейнера, нагрузки на контейнер и процесса применения нагрузки на контейнер. Кроме того, описан критерий выбора оптимального контейнера и выдвинуты некоторые предположения о зависимости оптимального размера кэша от среднеквадратичного отклонения ключей в нагрузке и соотношения скоростей основного и кэширующего контейнеров. Также описан способ задания нагрузки из смеси нормальных распределений и адаптивного кэширующего контейнера с использованием интервального статистического ряда.

Самоадаптирующийся контейнер данных (*SADC*) – ассоциативный контейнер, который меняет логику своей работы в зависимости от нагрузки *L* (поступающих к нему запросов, см. разделы 2.3 и 2.4). Кроме того, данный контейнер должен учитывать условия, в которых он работает – условия адаптации *C*.

Самоадаптирующийся контейнер данных состоит из:

1. Основного хранилища (ассоциативный контейнер *AS*);
2. Кэширующего ассоциативного контейнера *CS* размера *M*, реализующего алгоритм кэширования *A*;

3. Состояния самодаптирующегося контейнера данных в момент времени t q_t это комбинация из состояния ассоциативного контейнера данных *AS* на момент времени *t* и состояния кэширующего контейнера *CS* на момент времени *t*;
4. Модуля адаптации, который реализует некоторую функцию перехода f от одного состояния самоадаптирующегося контейнера к другому.

Основные элементы системы самоадаптирующегося контейнера данных изображены на рис. 1.

Рисунок 1. Общая схема самоадаптирующегося контейнера данных

Кроме того, разработан способ задания **ассоциативного контейнера** с помощью наборов и множеств:

$$AS = \langle O_t, OP, s_t \rangle, \quad (1)$$

где O_t – множество объектов в контейнере в момент времени t, OP – множество операций (отображений) над контейнером, s_t – внутреннее состояние контейнера в момент времени t.

$$O_t \subseteq O, \quad (2)$$

где O – множество всех возможных объектов, которые могут храниться в контейнере.

$$O_t = \{o^i\},\ i = 1, \dots, n_t, \quad (3)$$

где o^i – пара «ключ-значение», $n_t \in N$ – количество объектов в контейнере в момент времени t.

$$o^i = \langle k_i, v_i \rangle,\ i = 1, \dots, n_t, \quad (4)$$

где $k_i \in K$ – некоторый идентификатор (также называемый ключом), который однозначно определяет объект o^i , $v_i \in V$ – данные объекта o^i (также называемые значением).

$$s_t \in S, \quad (5)$$

где S – множество всех внутренних состояний контейнера.

$$OP = \{select, insert, remove\}, \quad (6)$$

где:

1. $select : K \times S \rightarrow O \times S$ это отображение, которое при получении идентификатора объекта возвращает объект, находящийся в множестве O и изменяет состояние контейнера s_t:

$$select(x, s_t) = \begin{cases} (o^x, s_{t+1}),\ (k_x = x) \wedge\ x \in \mathrm{K}_t \text{ (объект найден в контейнере)} \\ (\emptyset, s_{t+1}),\ x \notin \mathrm{K}_t \text{ (объект не найден в контейнере)} \end{cases}, \quad (7)$$

где K_t – множество всех ключей объектов из множества O_t, $x \in N$, $\mathrm{K}_t \subset K$.

2. $insert: O \times S \times\ O \rightarrow\ O \times S$ это отображение, которое добавляет объект к множеству объектов в контейнере:

$$insert(O_t, s_t, x) = (O_t \cup x, s_{t+1}) \quad (8)$$

3. $remove: O \times S \times\ K \rightarrow\ O \times S$ это отображение, которое удаляет объект из множества объектов в контейнере:

$$remove(O_t, s_t, x) = (O_t \setminus x, s_{t+1}) \quad (9)$$

В данной главе также предложен способ задания нагрузки на контейнер и процесса применения нагрузки на контейнер. **Нагрузкой L на контейнер данных** называется элемент множества:

$$L \in STL, \quad (10)$$

где STL – множество структурных нагрузок, определяемое по формуле:

$$STL\ =\ \{LB, LS\}, \quad (11)$$

где LB – блочная нагрузка (элементы в случайном порядке), а LS – последовательная нагрузка (кортеж элементов).

$$LB = \{e_1, \dots, e_n\} \quad (12)$$
$$LS = (e_1, \dots, e_n)$$

где $n \in N$ – количество элементов в нагрузке, а элементы нагрузки e_i это либо структурные нагрузки, либо базовые и задаются формулой:

$$e_i \in\ \{BL, STL\}, i = 1, \dots, n, \quad (13)$$

где BL – базовая нагрузка:

$$BL = (c, type), \quad (14)$$

где $c \in N$ – количество элементов в нагрузке, $type \in \{IL, RL, SL\}$ – тип нагрузки (IL – вставка в контейнер, RL – удаление из контейнера, SL – получение элемента контейнера). На основании данного подхода выведен способ задания **трассы** (детерминированной последовательности ключей, представляющей собой последовательность обращений с этими ключами к кэшу). В терминах нагрузки трасса T представляет собой:

$$T = L = LS =< BL(c, SL) >, \quad (15)$$

где c – количество элементов в трассе, а сами элементы трассы определяются отображением:

$$g(c) =\ \{k_1, \dots, k_c\}, \qquad k_i \in K, \qquad i = 1..c. \quad (16)$$

Процесс применения нагрузки на контейнер данных может быть представлен в виде набора:

$$AL = (AS, L, g, a), \quad (17)$$

где AS – контейнер данных, L – нагрузка, g – отображение для генерации ключей, a – отображение для применения нагрузки L к контейнеру AS.

$$g: N \rightarrow K$$
$$g(c) =\ \{k_1, \dots, k_c\},\ k_i \in K \quad (18)$$

$$a: O \times S \times STL \to O \times S$$

$$a(O_t, s_t, L) = \begin{cases} a(\dots a(a(O_t, s_t, e_1), e_2) \dots, e_n) | e_i \in L, i = 1..n, L \in STL \\ insert(\dots insert(insert(O_t, s_t, x_1), x_2), \dots, x_c) \\ \qquad | x_i \in g(c), L \in BL \wedge (type = IL) \\ remove(\dots remove(remove(O_t, s_t, x_1), x_2), \dots, x_c) \\ \qquad | x_i \in g(c), L \in BL \wedge (type = RL) \\ (O_t, select(O_t, \dots select(O_t, select(O_t, s_t, x_1.key), \dots, x_c.key) \\ \qquad | x_i \in g(c), L \in BL \wedge (type = SL) \end{cases}, \quad (19)$$

где O_t – множество объектов ассоциативного контейнера AS, s_t – внутреннее состояние контейнера AS, $\{insert, remove, select\}$ – множество операций контейнера AS.

На основе описанных выше наборов и множеств был разработан способ задания **самоадаптирующегося контейнера**:

$$SADC = \langle C, Q, q_t, f \rangle, \quad (20)$$

где C – множество условий адаптации, Q – множество состояний контейнера, $q_t \in Q$ – состояние контейнера в момент времени *t*, *f* – отображение перехода, которое при изменении нагрузки меняет состояние контейнера.

$$Q = \{q_i\}, i = 1..n \quad (21)$$

$$q_t = \langle AS_t, CS_t \rangle,$$

где $AS_t \in SAS$ – ассоциативный контейнер данных, хранящий данные в момент времени *t*, CS_t – кэширующий контейнер в момент времени *t*, *n* – количество всех возможных состояний контейнера.

$$CS_t = \langle A_t, M_t \rangle, \quad (22)$$

где $A_t \in A$ – алгоритм кэширования в момент времени *t*, $M_t \in N$ – размер кэша в момент времени *t*.

$$f: Q \times STL \times C \to Q \quad (23)$$

$$f(q_t, L, c) = q_{t+1},$$

Таким образом отображение *f* определяет, как будет изменяться самоадаптирующийся контейнер в зависимости от нагрузки *L* в условиях адаптации *C*. В общем случае может изменяться основной ассоциативный контейнер *AS*, алгоритм кэширования *A* и размер кэша *M*.

Также в диссертации описан критерий выбора оптимального контейнера. В данной работе основным критерием эффективности является время работы контейнера. Таким образом контейнер является оптимальным, если текущий набор параметров (AS_t, A_t, M_t) для нагрузки *L* в условиях адаптации *C* доставляет минимум функции времени работы контейнера:

$$t_{SADC}(AS, A, M, L, C) \to min \quad (24)$$

Следовательно, для построения оптимального контейнера необходимо решить задачу оптимизации 24, где *L*, *C* – независимые переменные, *AS*, *A*, *M* – зависимые (искомые) переменные, а функция t_{SADC} задана следующим образом:

$$t_{SADC}(AS, A, M, L, C) = t_{AS}(AS, A, M, L, C) + t_{CS}(A, M, L, C), \quad (25)$$

где $t_{AS}(AS, A, M, L, C)$ – время работы основного хранилища с применением ассоциативного контейнера AS при кэширующем контейнере с алгоритмом кэширования *A* и размером кэша *M* для нагрузки *L* в условиях *C*, $t_{CS}(A, M, L, C)$ – время работы кэша с алгоритмом кэширования *A* и размером кэша *M* для нагрузки *L* в условиях *C*.

Следовательно, если время работы при наборе текущих параметров (AS_t, A_t, M_t) для нагрузки *L* в условиях адаптации *C* не достигает минимума, то необходима адаптация.

Кроме этого во второй главе выдвинуты некоторые предположения о зависимости размера кэша M от среднеквадратичного отклонения σ и соотношения скоростей между хранилищами k, для которой выполняется критерий оптимальности 24. Зафиксировав алгоритм кэширования A_{const} и основное хранилище AS_{const}, а также взяв в качестве основного параметра трассы T среднеквадратичное отклонение σ и в качестве основного параметра условий адаптации C соотношение скоростей между хранилищами k, из формулы 24 получаем:

$$t(AS_{const}, A_{const}, M, \sigma, k) \to min, \tag{26}$$

где $t(AS_{const}, A_{const}, M, \sigma, k) =$

$$t_{AS}(AS_{const}, A_{const}, M, \sigma, k) + t_{CS}(A_{const}, M, \sigma, k),$$

$0 \le M \le M_{max}$, где M_{max} – доступный размер кэша.

Аналитически можно сделать несколько предположений:

Лемма 1. Существуют достаточно небольшие соотношения скоростей хранилищ, такие что использование кэша не имеет смысла (затраты на поиск в кэше больше, чем выгода от его использования):

$$\exists K_0 : \forall M > 0, \forall \sigma > 0, \forall k_0 \in K_0, \tag{27}$$

$$t(AS_{const}, A_{const}, M, \sigma, k_0) > t(AS_{const}, A_{const}, 0, \sigma, k_0) \Longrightarrow M_{opt} = 0.$$

Лемма 2. Для соотношений скоростей не попадающих под условия леммы 1 существует некоторый размер кэша M_0 такой что накладные расходы на кэш при $0 < M < M_0$ перекрывают выгоду от его использования, далее для размера M_0 эффективность применения кэша равна эффективности без кэша, и только при $M > M_0$ применение кэша становится эффективно.

$$\forall k \notin K_0, \forall \sigma \tag{28}$$

$$\exists M_0 : t(AS_{const}, A_{const}, M_0, \sigma, k_0) = t(AS_{const}, A_{const}, 0, \sigma, k_0)$$

Следствие 1. $M_{max} < M_0 \Rightarrow M_{opt} = 0.$

Лемма 3. Существуют k достаточно большие, такие что использование кэша уменьшает время работы контейнера для любого распределения ключей если позволяет максимально доступный размер кэша. В этом случае, оптимальный размер кэша будет стремиться к тому, чтобы вместить все уникальные элементы трассы T, что позволит уменьшить время обращения к основному хранилищу.

Однако в случае нормального распределения согласно математической статистике, в интервал $(-4\sigma, 4\sigma)$ помещается 99,997% всех уникальных ключей, а следовательно оптимальный размер кэша 8σ будет достаточным и дальнейшее увеличение размера кэша не будет давать существенного уменьшения времени работы контейнера.

$$\exists K_b : \forall M < 8\sigma < M_{max}, \forall \sigma > 0, \forall k_b \in K_b, \tag{29}$$

$$t(AS_{const}, A_{const}, M, \sigma, k_b) > t(AS_{const}, A_{const}, 8\sigma, \sigma, k_b) \Longrightarrow M_{opt} = N_T \approx 8\sigma,$$

где N_T – количество уникальных элементов в трассе T.

Лемма 4. $\forall k \in K_b : 8\sigma > M_{max} > M_0 \Rightarrow M_{opt} = M_{max}.$ (30)

Лемма 5. Существует некоторое множество соотношений скоростей хранилищ, для которых не выполняются условия лемм 1 и 3. В таком случае существует некоторая σ_0, для которой при $\sigma < \sigma_0$, оптимальным будет не использовать кэш, а при $\sigma > \sigma_0$ оптимальный размер кэша вычисляется аналогично леммам 3 и 4.

$$\forall k \in K_d = K/(K_0 \cup K_b) : M_{opt} = \begin{cases} 0, (\sigma < \sigma_0) \vee (M_{max} < M_0 \wedge \sigma > \sigma_0) \\ N_T \approx 8\sigma, 8\sigma < M_{max} \wedge \ \sigma > \sigma_0 \\ M_{max}, 8\sigma > M_{max} \wedge \sigma > \sigma_0 \end{cases} \tag{31}$$

Лемма 6. Основываюсь на предыдущих леммах и следствии, можно вывести формулу оптимального размера кэша в зависимости от среднеквадратичного отклонения ключей в трассе и соотношения скоростей хранилищ:

$$M_{opt}(k,\sigma)=\begin{cases}0,(k\in K_0)\vee(M_{max}<M_0)\vee(\sigma<\sigma_0\wedge k\in K_d)\\ 8\sigma,(k\in K_b\wedge M_0<8\sigma<M_{max})\vee\\ (k\in K_d\wedge M_0<8\sigma<M_{max}\wedge\sigma>\sigma_0)\\ M_{max},(k\in K_b\wedge 8\sigma>M_{max}>M_0)\vee\\ (k\in K_d\wedge 8\sigma>M_{max}>M_0\wedge\sigma>\sigma_0)\end{cases} \quad (32)$$

где k - соотношение скоростей хранилищ, σ – среднеквадратичное отклонение ключей в нагрузке на контейнер, K_0 – множество соотношений скоростей, для которых оптимальный размер кэша нулевой, K_b – множество соотношений скоростей, для которых оптимальный размер кэша равен максимально доступному, $K_d = K/(K_0 \cup K_b)$, M_{max}- максимально доступный размер кэша, M_0 – ненулевой размер кэша, для которого время работы равно времени работы при нулевом размере кэша, σ_0 – среднеквадратичное отклонение ключей в нагрузке при котором эффективность применения кэша становится больше чем затраты на его использование.

Помимо этого, в данной главе описан способ задания нагрузки из смеси нормальных распределений. Модель данных, в которой учитывается несколько одномерных нормальных распределений с различными весами называется смесью нормальных распределений (Gaussian mixture model - GMM). Для заданной выборки χ объемом N смесью p(x), состоящей из K компонент $N(\mu,\ \sigma^2)$ с весами π_k называется формула:

$$p(x)=\sum_{k=1}^{K}\pi_k N(\mu_k,\sigma_k^2), \quad (33)$$

$$\text{где } \sum_{k=1}^{K}\pi_k=1.$$

Т. о. из формулы 16:

$$g(c)=\ \{k_1,\dots,k_c\}, \quad (34)$$

$$\text{где } k_i=p(x), i=1..c.$$

Задача нахождения параметров таких распределений (среднеквадратичное отклонение σ и матожидание μ) сводится к задаче нахождения вектора параметров $\theta=(\pi_1,\dots,\pi_k,\mu_1,\dots,\mu_k,\sigma_1,\dots,\sigma_k)$, при котором функция правдоподобия 35 достигает максимума.

$$\ln p(\chi|\pi,\mu,)=\sum_{n=1}^{N}\ln\left\{\sum_{k=1}^{K}\pi_k N(x_n|\mu_k,\sigma_k^2)\right\} \quad (35)$$

В третьей главе представлены алгоритмы работы модулей самоадаптирующегося контейнера. Описана общая схема работы кэширующего модуля (рис. 2), используемого для исследования зависимости оптимального размера кэша от соотношения скоростей хранилищ и среднеквадратичного отклонения ключей в нагрузке. Алгоритм самоадаптирующегося контейнера данных, использующего программный кэш, был реализован с использованием архитектурного шаблона «кэш на стороне» (cache-aside). При данной архитектуре кэш располагается в оперативной памяти, а основное хранилище на жестком диске или на удаленном источнике, что позволяет программно изменять размеры кэша в процессе работы, чтобы обеспечить максимальную эффективность. Кэш и основное хранилище являются ассоциативными структурами данных, которые хранят пары «ключ-значение». Ключи k_i на данной схеме принадлежат объектам, сгенерированным по закону нормального распределения для некоторой трассы, а соотношение хранилищ k определяется по соотношению медленной и быстрой памяти.

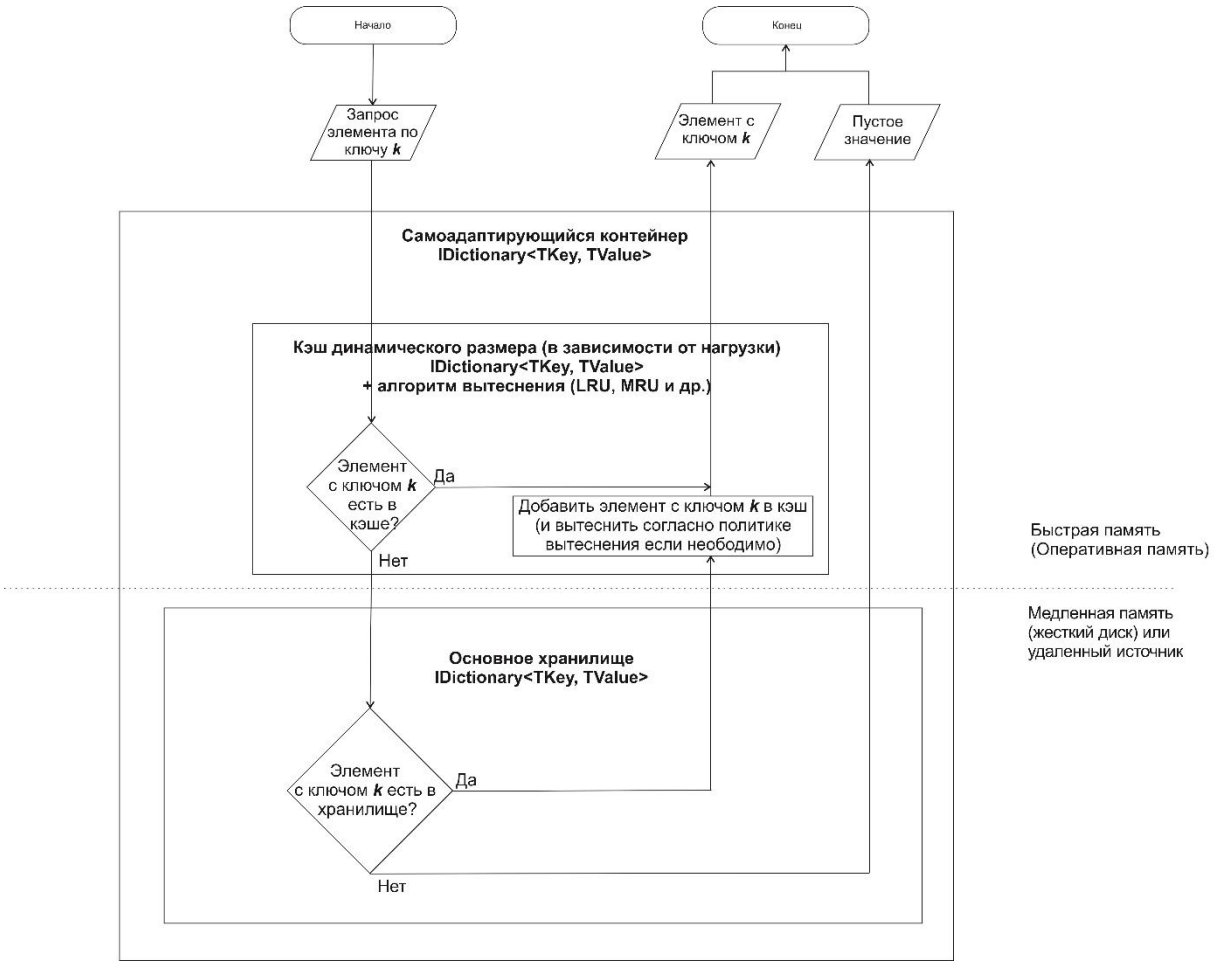

Рис. 2. Блок-схема алгоритма самоадаптирующегося контейнера данных с использованием кэша

Помимо этого, в данной главе представлен алгоритм работы **модуля определения параметров сложной нагрузки**, состоящей из смеси нормальных распределений. Для решения задачи ЗЗ из второй главы был выбран алгоритм EM (Expectation Maximization). Также проведен анализ методов инициализации данного алгоритма, т.к. EM алгоритм находит локальный экстремум функции правдоподобия, значение которого может оказаться гораздо ниже, чем глобальный максимум. Таким образом, в зависимости от выбора начального приближения алгоритм может сходиться к разным точкам. Были рассмотрены: случайный метод инициализации, rndEM, emEM, HAC и kmeans++. Для реализации модуля был выбран последний, т.к. он больше всего подходит для решения данной задачи. Кроме того, необходимо учитывать, что данные постоянно поступают в контейнер, зачастую в большом количестве и на большой скорости. Следовательно, необходимо применять специальные алгоритмы кластеризации на потоке. К таким алгоритмам относится EM алгоритм с использованием буфера.

Модуль самоадаптирующегося контейнера данных был реализован с применением данного алгоритма: сначала данные в режиме онлайн накапливаются в буфере, а затем в режиме оффлайн через некоторые заданные промежутки времени/заданное количество вновь прибывших элементов или по запросу контейнера (в случае если кластеризацию нужно провести немедленно) выполняется EM алгоритм кластеризации на буфере. Размер буфера является настраиваемым параметром и определяет, насколько быстро устаревают данные.

Также был разработан алгоритм **кэширующего контейнера данных с использованием интервального статистического ряда**:

1. Инициализация. В процессе инициализации основное хранилище разбивается на n_b блоков. В результате выполнения получается набор блоков, состоящий из элементов, каждый из которых содержит:

- Минимальное значение блока;
- Максимальное значение блока;
- Ключи основного хранилища, которые располагаются в интервале [минимальное значение блока; максимальное значение блока].

2. Построение нулевого интервального статистического ряда и очистка буфера. На основе полученного на шаге 1 набора блоков, строится нулевой интервальный статистический ряд, т.е. для каждого интервала выставляется значение 0. Также очищается буфер, содержащий l ключей запросов.
3. Обработка поступления запроса в кэширующий контейнер. Если в кэше есть такой ключ, то в ответ возвращается объект из кэша, если нет – из основного хранилища. При этом, если кол-во ключей в буфере не достигло значения l, то ключ запроса добавляется в буфер. Иначе выполняются шаги 4-5.
4. Заполнение и сортировка интервального статистического ряда по следующему правилу: для каждого интервала ряда подсчитывается сколько ключей из буфера попадает в интервал, далее интервальный статистический ряд упорядочивается по количеству ключей от большего к меньшему.
5. Заполнение кэша. В кэш помещаются все объекты, ключи которых попадают в интервалы, для которых произошло наибольшее число запросов за последнее количество запросов l по следующему правилу:
   - пока кэш не заполнен по интервальному статистическому ряду определяются интервалы с наибольшим числом попаданий;
   - по набору блоков, полученном в процессе инициализации, определяются все ключи основного хранилища, которые необходимо загрузить в кэш;
   - в кэш загружаются все объекты основного хранилища, у которых ключи совпадают с найденными на предыдущем этапе.

   Далее происходит переход к шагу 2.

Шаг 3 выполняется каждый раз при обращении к кэширующему контейнеру. Общий алгоритм работы кэширующего контейнера с использованием интервального статистического ряда изображен на рис. 3.

Четвертая глава посвящена вычислительным экспериментам. В результате экспериментов были доказаны леммы из главы 2 и найдены множества K_d, K_b, K_0, размер кэша M_0 и значение среднеквадратичного отклонения σ_0, для которых выполняются леммы, что позволило получить зависимость размера кэша от среднеквадратичного отклонения ключей в трассе и соотношения скоростей кэша и основного хранилища. Полученные закономерности могут быть использованы для динамического изменения размера кэша в зависимости от параметров нагрузки при реализации самоадаптирующегося контейнера данных.

Модуль определения параметров сложной нагрузки был протестирован для одного и нескольких кластеров, смещения и устаревания данных. Точность кластеризации модуля при использовании k-means++ составила 100% для одного кластера и 96% для нескольких. Также приведены результаты экспериментов реализации модуля со случайной инициализацией (точность кластеризации нескольких кластеров составила 16%). Кроме того, приведены результаты экспериментов по определению зависимости времени работы модуля от количества кластеров, количества итераций и количества ключей в буфере, которые позволяют прогнозировать время работы модуля. Исходя из результатов тестирования, сделано заключение, что данный модуль хорошо справляется с задачей определения параметров сложной нагрузки и может быть эффективно использован в самоадаптирующихся контейнерах данных.

Было проведено сравнение адаптивного кэширующего контейнера данных с использованием интервального статистического ряда с другими известными и эффективными алгоритмами кэширования – LRU, MRU, ARC, LIRS. Тестирование проводилось на различных искусственных трассах запросов и трассах, состоящих из реальных данных. В результате

экспериментов было выявлено, что адаптивный кэширующий контейнер данных с использованием интервального статистического ряда во многих ситуациях превосходит указанные выше алгоритмы по проценту попаданий в кэш, а, следовательно, является эффективным.

В **заключении** изложены основные результаты работы.

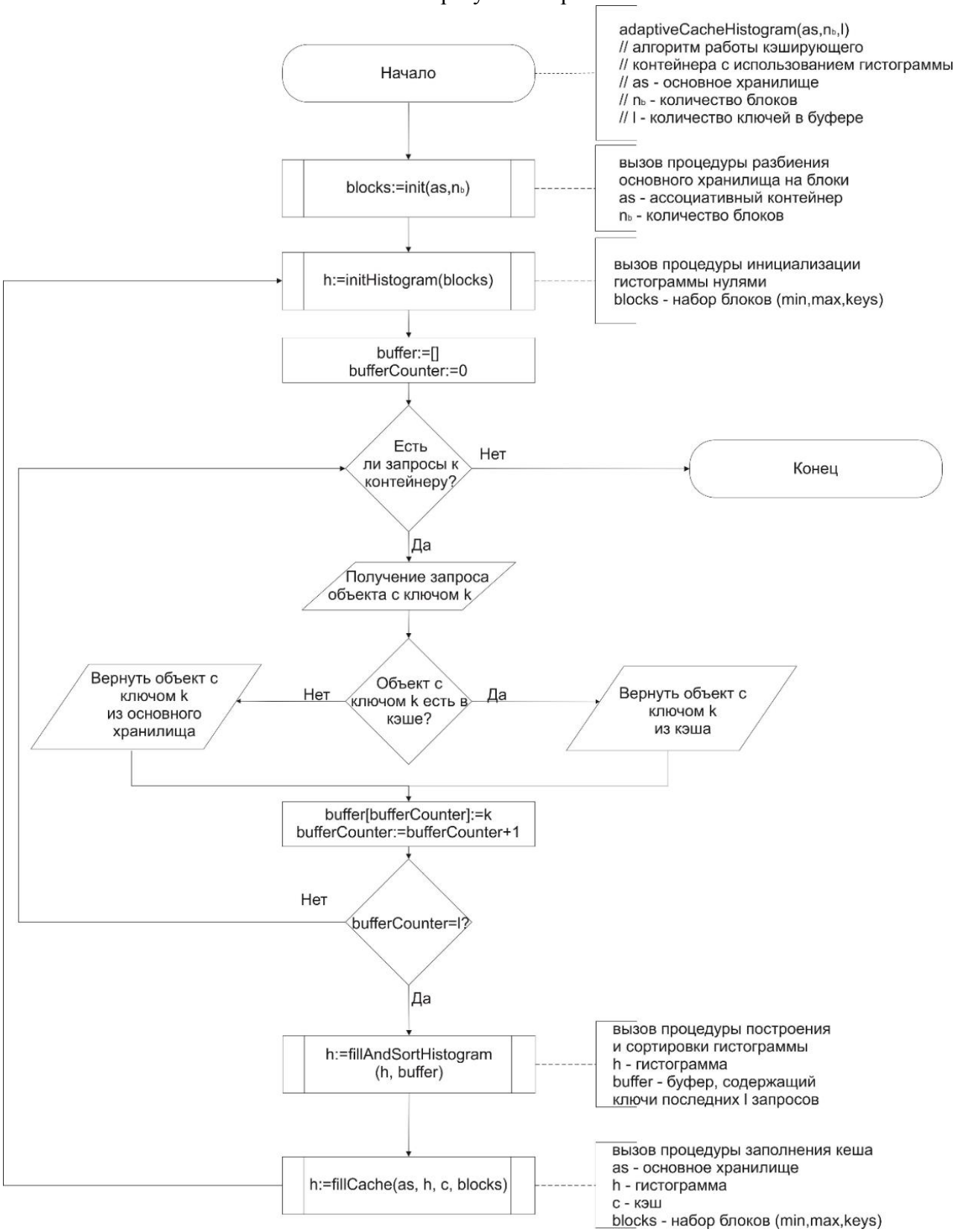

Рис 3. Блок-схема алгоритма кэширующего контейнера с использованием интервального статистического ряда

Основные результаты работы и выводы

1. Разработан метод адаптации ассоциативного контейнера данных.
2. Определены области применения одномерных и многомерных ассоциативных контейнеров данных и алгоритмов кэширования с целью использования в самоадаптирующемся контейнере данных.
3. При исследовании зависимости размера кэша от среднеквадратичного отклонения нормального распределения и соотношения скоростей хранилищ была получена формула оптимального по времени размера кэша.
4. Разработан и реализован алгоритм определения параметров сложной нагрузки на самоадаптирующиеся контейнеры данных, который позволил достигнуть 100% точности для одного кластера и 96% для нескольких кластеров.
5. Разработан и реализован алгоритм адаптивного кэширующего контейнера данных с использованием интервального статистического ряда. Проведено сравнение с другими известными и эффективными алгоритмами кэширования (LRU, MRU, ARC, LIRS). В результате экспериментов выявлено, что данный алгоритм во многих ситуациях превосходит данные алгоритмы по проценту попаданий в кэш.

Публикации, индексируемые в Scopus

1. Potapov D.R. Multidimensional data structures usage in adaptive data storages / D.R. Potapov // Journal of Physics: Conference Series. – 2019. –Vol. 1202. – 012020.

Публикации в изданиях, рекомендованных ВАК РФ

2. Потапов Д. Р. Обзор условий адаптации самоадаптирующихся ассоциативных контейнеров данных / Д. Р. Потапов, М.А. Артемов, Е.С. Барановский // Вестник ВГУ. Серия Системный анализ и информационные технологии. – 2017. – №1. – С. 112-119.
3. Потапов Д. Р. Обзор методов построения многомерных контейнеров данных «ключ-значение» для использования в самоадаптирующихся контейнерах данных / Д.Р. Потапов // Прикладная информатика. – 2018. – №2(74). – С. 69-82.
4. Обзор методов построения контейнеров данных «ключ-значение» для использования в самоадаптирующихся контейнерах данных / Потапов Д. Р., Артемов М.А., Барановский Е.С. Селезнев К.Е. // Кибернетика и программирование. – 2017. – №5. – С. 14-45.
5. Потапов Д. Р. Исследование эффективности применения кэша для использования в самоадаптирующихся контейнерах данных / Д.Р. Потапов // Информационные технологии. – 2019. – Т. 25. – №4. – С. 216-222.
6. Потапов Д. Р. Реализация модуля определения параметров сложной нагрузки на самоадаптирующиеся контейнеры данных / Д.Р. Потапов // ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ. – 2019. – №1. – С. 87-95.

Свидетельство о государственной регистрации программ для ЭВМ

7. Свидетельство о государственной регистрации программы для ЭВМ / Потапов Д.Р. «Средства визуального представления результатов анализа нагрузки и сравнения контейнеров данных» // Зарегистрировано в Реестре программ для ЭВМ № 2019613881 26.03.2019.

Прочие публикации автора

8. Потапов Д.Р. Визуальное представление результатов анализа и сравнения контейнеров данных / Д.Р. Потапов, К.Е. Селезнев // Информатика: проблемы, методология, технологии: Матер. XVI Международная научно-методическая конференция. – Воронеж: Воронежский государственный университет, 2016. – С. 123-128.

9. Потапов Д.Р. Визуальное сравнение и анализ контейнеров данных / Д.Р. Потапов // Математическое и компьютерное моделирование, информационные технологии управления: сб. тр. Школы для студентов, аспирантов и молодых ученых «МКМИТУ-2016». – Воронеж: Воронежский государственный университет, 2016. –
С. 178-181.
10. Потапов Д.Р. Разработка и реализация метода для анализа и сравнения контейнеров данных / Д.Р. Потапов // Вестник Факультета прикладной математики и механики. – 2016. – №5. – С. 30-36.
11. Потапов Д. Р. Анализ применения многомерных структур данных в самоадаптирующихся контейнерах данных / Д.Р. Потапов // Актуальные проблемы прикладной математики, информатики и механики: сб. тр. межд. науч.-техн. конф. – Воронеж: Воронежский государственный университет, 2017. – С. 436-442.
12. Потапов Д. Р. АНАЛИЗ ФУНКЦИИ РАСПРЕДЕЛЕНИЯ НАГРУЗКИ НА КЭШ-ПАМЯТЬ / Д.Р. Потапов // Актуальные проблемы прикладной математики, информатики и механики: сб. тр. межд. науч.-техн. конф. – Воронеж: Воронежский государственный университет, 2018. – С. 434-437.