КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ
COMPUTER SIMULATION

УДК 004.75 **DOI: 10.18413/2518-1092-2026-11-2-0-10**

Воскобойников И.С.
Гвоздевский И.Н.
Булгаков В.Д.

МЕТОД АДАПТИВНОГО ФОРМИРОВАНИЯ БЛОКОВ И ИНДЕКСАЦИИ НЕСТРУКТУРИРОВАННЫХ ДАННЫХ В ДЕЦЕНТРАЛИЗОВАННЫХ СИСТЕМАХ ХРАНЕНИЯ

Белгородский государственный технологический университет им. В.Г. Шухова
ул. Костюкова, 46, г. Белгород, 308012, Россия

e-mail: ilia.voskoboinikov@mail.ru

Аннотация

В статье рассматривается задача обработки и индексирования неструктурированных текстовых данных в децентрализованных системах хранения. Проведён анализ существующих подходов, основанных на статических параметрах формирования блоков, выявлены их ограничения, связанные с отсутствием учёта динамических характеристик распределённой среды, что приводит к увеличению сетевой нагрузки и снижению эффективности поиска.

Предложен метод адаптивного формирования блоков и индексирования данных, учитывающий интенсивность входного потока, уровень загрузки сети и количество активных узлов. Описана архитектура системы, включающая модули предобработки, агрегации данных, индексирования и распределённого хранения на основе распределённых хеш-таблиц. Разработана математическая модель и алгоритм, обеспечивающие динамическое управление параметрами формирования блоков.

Проведён теоретический анализ работы алгоритма, показано влияние ключевых параметров системы на процессы формирования блоков. Выполнено сравнение с фиксированным методом, продемонстрированы преимущества адаптивного подхода с точки зрения снижения сетевых издержек, повышения масштабируемости и устойчивости к изменению нагрузки.

Работа может быть использована при разработке распределённых систем хранения, поисковых платформ и систем потоковой обработки данных. Полученные результаты создают основу для дальнейших исследований адаптивных и гибридных методов обработки неструктурированных данных в децентрализованных средах.

Ключевые слова: неструктурированные данные; децентрализованные системы хранения; адаптивное формирование блоков; распределённая индексация; DHT; обработка текстовых данных; масштабируемость; оптимизация

Для цитирования: Воскобойников И.С., Гвоздевский И.Н., Булгаков В.Д. Метод адаптивного формирования блоков и индексации неструктурированных данных в децентрализованных системах хранения // Научный результат. Информационные технологии. – Т.11, №2, 2026. – С. 121-134. DOI: 10.18413/2518-1092-2026-11-2-0-10

Voskoboinikov I.S.
Gvozdevsky I.N.
Bulgakov V.D.

METHOD OF ADAPTIVE BLOCK FORMATION AND INDEXING OF UNSTRUCTURED DATA IN DECENTRALIZED STORAGE SYSTEMS

Belgorod State Technological University named after V.G. Shukhov
46 Kostyukova St., Belgorod, 308012, Russia

e-mail: ilia.voskoboinikov@mail.ru

Abstract
The paper addresses the problem of processing and indexing unstructured textual data in decentralized storage systems. An analysis of existing approaches based on static block formation parameters is conducted, revealing their limitations related to the lack of consideration of dynamic characteristics of distributed environments, which leads to increased network overhead and reduced search efficiency.
An adaptive method for block formation and data indexing is proposed, taking into account input data rate, network load, and the number of active nodes. The system architecture is described, including modules for data preprocessing, aggregation, indexing, and distributed storage based on distributed hash tables. A mathematical model and an algorithm are developed to provide dynamic control of block formation parameters.
A theoretical analysis of the proposed algorithm is performed, demonstrating the influence of key system parameters on block formation. A comparison with a fixed block formation method is presented, showing the advantages of the adaptive approach in terms of reduced network costs, improved scalability, and robustness under varying load conditions.
The proposed method can be applied in the design of distributed storage systems, decentralized search platforms, and stream processing systems. The results provide a foundation for further research on adaptive and hybrid methods for processing unstructured data in decentralized environments.
Keywords: unstructured data; decentralized storage systems; adaptive block formation; distributed indexing; DHT; text processing; scalability; optimization
For citation: Voskoboinikov I.S., Gvozdevsky I.N., Bulgakov V.D. Method of Adaptive Block Formation and Indexing of Unstructured Data in Decentralized Storage Systems // Research result. Information technologies. – T.11, №2, 2026. – P. 121-134. DOI: 10.18413/2518-1092-2026-11-2-0-10

ВВЕДЕНИЕ

Существующие методы обработки и поиска неструктурированных данных, как правило, разрабатывались для централизованных систем и предполагают наличие глобального индекса, централизованных вычислительных ресурсов и стабильной инфраструктуры хранения [7]. В условиях децентрализации такие предпосылки не выполняются, что требует адаптации классических подходов к новым архитектурным ограничениям. В частности, методы распределённой индексации позволяют частично решить проблему поиска данных за счёт размещения индексных структур в распределённой среде [8]. Однако такие подходы, как правило, используют статические параметры обработки и не учитывают динамику изменения состояния сети. Однако такие подходы, как правило, используют статические параметры обработки и не учитывают динамику изменений состояния сети.

В ряде работ, посвящённых децентрализованным системам хранения, основной акцент сделан на вопросах функционирования базовой инфраструктуры – в частности, рассматриваются алгоритмы маршрутизации в одноранговых сетях, механизмы репликации данных, а также подходы к обеспечению их целостности и согласованности [9, 10].

При этом аспекты, связанные с адаптацией параметров обработки и индексирования неструктурированных данных, как правило, рассматриваются эпизодически и не образуют целостного методического подхода.

Анализ существующих решений показывает, что в них отсутствует единый механизм, позволяющий учитывать изменения характеристик системы – таких как интенсивность поступления данных, текущая загрузка сети и изменение числа активных узлов – при формировании блоков и построении индексных структур.

На практике это приводит к ряду эффектов: увеличению объёма сетевого взаимодействия, росту задержек при выполнении поисковых операций и снижению эффективности использования ресурсов распределённой среды. Особенно явно данные проблемы проявляются в условиях нестабильной нагрузки, когда параметры системы изменяются во времени.

В связи с этим возникает необходимость разработки подходов, ориентированных на динамическую настройку параметров обработки данных. Речь идёт о переходе от фиксированных схем к моделям, способным изменять своё поведение в зависимости от текущего состояния распределённой системы.

Целью работы является разработка метода адаптивного формирования блоков и индексации неструктурированных данных в децентрализованных системах хранения, обеспечивающего снижение сетевой нагрузки и повышение эффективности выполнения поисковых операций.

Следует отметить, что большинство современных решений ориентированы на оптимизацию отдельных аспектов (хранение, маршрутизация), тогда как комплексные методы адаптивного управления процессами обработки данных остаются недостаточно разработанными.

Научная новизна работы заключается в разработке адаптивной модели формирования блоков, учитывающей одновременно интенсивность потока данных, загрузку сети и количество активных узлов, а также в интеграции данной модели с механизмами распределённой индексации в DHT-системах.

АРХИТЕКТУРА СИСТЕМЫ

В рамках исследования рассматривается децентрализованная система обработки и хранения неструктурированных данных, построенная на основе одноранговой (peer-to-peer) сети с использованием распределённой хеш-таблицы (DHT). Архитектура системы ориентирована на обеспечение масштабируемости, отказоустойчивости и эффективной обработки текстовых данных в условиях динамически изменяющейся сетевой среды [4–6].

Предлагаемая архитектура включает в себя совокупность взаимодействующих компонентов, реализующих полный цикл обработки данных – от их поступления до индексирования и размещения в распределённом хранилище. В отличие от традиционных централизованных систем, каждый узел сети выполняет как функции хранения, так и функции обработки данных, что соответствует принципам децентрализованных вычислений [9].

В архитектуре выделяются следующие функциональные модули:

1. Модуль приёма данных (Data Ingestion Module)

Данный модуль обеспечивает поступление неструктурированных текстовых данных в систему из внешних источников. Источниками могут выступать пользовательские запросы, потоки данных, файловые системы или распределённые хранилища. Поток данных характеризуется интенсивностью поступления (классическая модель потока заявок [14]), которая может быть описана как:

$$\lambda(t) = \frac{dN(t)}{dt},$$

где

$\lambda(t)$ – интенсивность поступления данных (документов/единицу времени);

$N(t)$ – кумулятивное количество поступивших документов к моменту времени t;

t – время.

2. Модуль предобработки данных (Preprocessing Module)

На данном этапе осуществляется преобразование неструктурированных текстовых данных в форму, пригодную для индексирования. Предобработка включает токенизацию, нормализацию,

удаление стоп-слов и лемматизацию, что соответствует классическим подходам обработки текстовой информации [7]. Результатом работы модуля является множество термов:

$$\mathrm{T(D_j)} = \left\{\mathrm{t_{j1}}, \mathrm{t_{j2}}, \ldots, \mathrm{t_{jm_j}}\right\},$$

где

D_j – j-й документ;

$\mathrm{T}(D_j)$ – множество термов документа;

t_{ji} – i-й терм документа D_j;

$\mathrm{m_j}$ – количество термов в документе.

3. Модуль адаптивного формирования блоков (Adaptive Block Formation Module)

Ключевым элементом предлагаемой архитектуры является модуль формирования блоков данных, обеспечивающий агрегирование отдельных документов в блоки перед их индексированием и размещением в системе. Блок определяется как:

$$B_k = \{D_1, D_2, \ldots, D_k\},$$

где

$\mathrm{B_k}$ – k-й сформированный блок;

$\mathrm{D_i}$ – i-й документ;

k – количество документов в блоке.

В отличие от существующих подходов, где размер блока фиксирован, в данной работе предполагается использование адаптивного механизма, учитывающего текущее состояние системы (нагрузку сети, интенсивность поступления данных и количество узлов).

4. Модуль индексирования (Indexing Module)

После формирования блока выполняется построение индексной структуры, обеспечивающей эффективный поиск данных. Индекс блока формируется на основе объединения термов входящих в него документов:

$$\mathrm{I(B_k)} = \bigcup_{\mathrm{i=1}}^{\mathrm{k}} \mathrm{T(D_i)},$$

где

$I(B_k)$ – индекс блока;

$\mathrm{T}(D_i)$ – множество термов документа.

Полученный индекс сопоставляется с идентификатором блока, формируемым с использованием хеш-функции:

$$\mathrm{CID_k} = \mathrm{H(B_k)},$$

где

$\mathrm{CID_k}$– контент-адресуемый идентификатор блока;

$\mathrm{H}(\cdot)$ – криптографическая хеш-функция [4],

$\mathrm{B_k}$– k-й сформированный блок.

5. Модуль распределённого хранения (Distributed Storage Module)

Хранение блоков и соответствующих индексных структур осуществляется в распределённой хеш-таблице. Каждому ключу (CID) сопоставляется узел сети, ответственный за его хранение. Распределение данных выполняется на основе метрики расстояния (например, XOR-метрики в алгоритме Kademlia) [5]. Функция размещения может быть представлена в виде:

$$Node(CID_k) = \arg\min_{n \in \mathcal{N}} d\,(CID_k, n)$$

где

$\mathcal{N}$ – множество узлов сети;

n – конкретный узел;

$\mathrm{d}(\cdot,\cdot)$ – метрика расстояния (например XOR);

$Node(CID_k)$ – узел, где хранится блок.

6. Модуль поиска (Query Processing Module)

Поиск данных осуществляется посредством обращения к распределённому индексу. Запрос преобразуется в набор термов, после чего выполняется поиск соответствующих блоков в DHT. Результатом является множество идентификаторов:

$$R(Q) = \{CID_1, CID_2, \dots, CID_p\}$$

где

Q – поисковый запрос;

$R(Q)$ – результат поиска;

CID_p– найденные блоки;

p – количество найденных блоков.

Общую архитектуру системы можно представить в виде следующей схемы (см.рисунок 1).

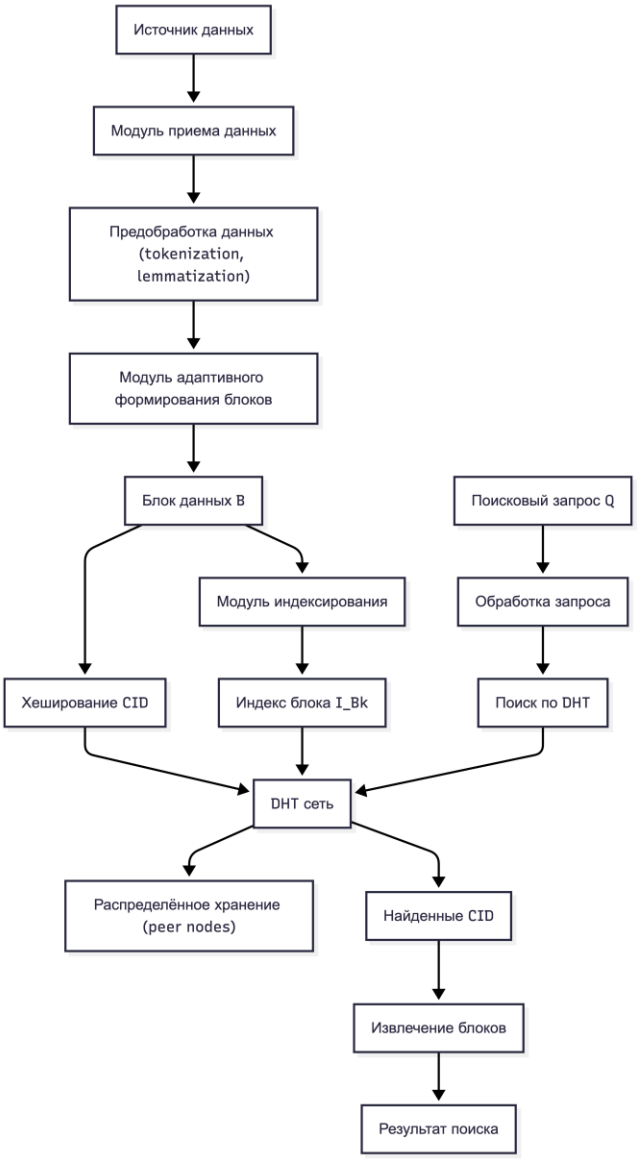

Рис. 1. Общая архитектура системы

Fig. 1. General architecture of the system

В отличие от существующих решений, предложенная архитектура обладает следующими характеристиками:

- Отсутствие централизованного узла управления.
- Совмещение функций хранения и обработки на уровне узлов.
- Использование контент-адресуемого хранения.

- Введение модуля адаптивного формирования блоков как ключевого элемента системы.
- Возможность динамической настройки параметров обработки данных.

Предложенная архитектура может быть реализована как поверх существующих протоколов, таких как Kademlia или Chord, без существенной модификации их базовых механизмов маршрутизации.

МАТЕМАТИЧЕСКАЯ МОДЕЛЬ АДАПТИВНОГО ФОРМИРОВАНИЯ БЛОКОВ

В условиях децентрализованных систем хранения эффективность обработки неструктурированных данных во многом определяется параметрами формирования блоков и стратегией их индексирования. В традиционных подходах размер блока и частота его формирования задаются статически, что не учитывает динамику изменения характеристик системы, таких как интенсивность поступления данных, загрузка сети и количество активных узлов [9, 12]. Это приводит к неэффективному использованию ресурсов и увеличению времени отклика системы.

В работе предлагается математическая модель адаптивного формирования блоков, основанная на учёте динамических параметров распределённой среды. Подобные подходы к адаптивному управлению параметрами системы рассматриваются в работах по динамическому распределению ресурсов [11]. Для формализации задачи вводится следующий набор параметров:

- $\lambda(t)$ – интенсивность поступления данных;
- $L(t) \ni [0, 1]$ – уровень загрузки сети;
- t_{acc} – время накопления блока;
- B_k – формируемый блок данных;
- $|B_k|$ – размер блока (количество документов);
- B_{max} – максимально допустимый размер блока;
- $N(t)$ – количество активных узлов в сети момент времени t;
- T_{max} – максимально допустимое время формирования блока.

В рамках предлагаемой модели предполагается, что функции $\lambda(t)$, $L(t)$, $N(t)$ являются кусочно-непрерывными и могут оцениваться на основе данных мониторинга распределённой системы. Следует отметить, что точность их определения зависит от используемых механизмов сбора метрик и может изменяться во времени.

Указанные параметры одновременно характеризуют как свойства входного потока данных, так и текущее состояние сети. Формирование блока осуществляется при выполнении одного из следующих условий:

$$|B_k| \geq B_{max} \bigvee t_{acc} \geq T_{max}.$$

Подобное условие соответствует широко применяемым стратегиям буферизации в системах потоковой обработки данных, однако в данном случае она используется в качестве базового ограничения.

В отличие от фиксированных подходов, в работе предлагается определять размер блока как функцию от динамических параметров системы:

$$B_{opt}(t) = \alpha * \frac{\lambda(t)}{L(t) + \varepsilon} * N(t),$$

где

$B_{opt}(t)$ – оптимальный размер блока;

α – коэффициент масштабирования;

ε – малая положительная константа, предотвращающая деления на ноль;

$T(t)$ – количество узлов сети.

Предложенная функция отражает интуитивно ожидаемое поведение системы: при увеличении интенсивности входного потока размер блока возрастает, тогда как рост загрузки сети приводит к его уменьшению. Увеличение числа узлов, напротив, позволяет системе обрабатывать более крупные блоки за счёт распределения нагрузки.

Для учета времени формирования блока, вводится следующая функция:

$$T_{opt}(t) = \beta * \frac{1}{\lambda(t)} * (1 + L(t)),$$

где

$T_{opt}(t)$ – оптимальное время формирования блока;

β – коэффициент масштабирования времени.

Она отражает поведение системы при различных режимах работы. При увеличении интенсивности входного потока интервал накопления сокращается, что позволяет быстрее формировать блоки. В условиях высокой сетевой нагрузки, напротив, допускается увеличение времени накопления, что снижает частоту передачи данных.

Процесс формирования блоков можно рассматривать с позиции минимизации совокупных затрат, включающих сетевые издержки и время поиска:

$$\min_{B_k} (C_{net}(B_k) + T_{search}(B_k)),$$

где

$C_{net}(B_k)$ – суммарные затраты сети;

$T_{search}(B_k)$ – время поиска данных.

В рамках принятой модели предполагается, что:

$$C_{net}(B_k) \sim |B_k| * \log N(t)$$
$$T_{search}(B_k) \sim \log |I(B_k)|,$$

где

$I(B_k)$ – индекс блока.

С учётом введённых зависимостей условия формирования блока принимают вид:

$$|B_k| \geq B_{opt}(t) \vee t_{acc} \geq T_{opt}(t).$$

Увеличение размера блока приводит к снижению сетевых издержек, однако одновременно усложняет процедуры поиска, что требует выбора компромиссного режима функционирования системы.

Следует отметить, что предложенная модель имеет ряд ограничений:

- параметры $\lambda(t)$, $L(t)$, $N(t)$ считаются известными и измеримыми;
- не учитывается семантическая структура данных;
- не рассматриваются задержки, связанные с репликацией данных;
- предполагается однородность узлов сети.

Несмотря на указанные ограничения, модель позволяет формализовать процесс адаптивного формирования блоков и может служить основой для дальнейших исследований.

АЛГОРИТМ АДАПТИВНОГО ФОРМИРОВАНИЯ БЛОКОВ И ИНДЕКСАЦИИ

Предложенная математическая модель адаптивного формирования блоков реализуется в виде алгоритма, обеспечивающего динамическое управление процессом агрегации данных и их индексирования в условиях децентрализованной среды. Алгоритм учитывает текущие характеристики системы, включая интенсивность входного потока данных, загрузку сети и количество активных узлов.

Алгоритм (см. рисунок 2) функционирует в потоковом режиме и включает следующие основные этапы:

1. Приём и буферизация входных данных;
2. Предобработка текстовой информации;
3. Динамическое формирование блока;
4. Проверка условий завершения блока;
5. Построение индексной структуры;
6. Хеширование блока и размещение в распределённой сети;
7. Обновление параметров системы.

В отличие от статических подходов, предложенный алгоритм реализует механизм адаптации параметров $B_{opt}(t)$ и $T_{opt}(t)$, определённых в предыдущем разделе.

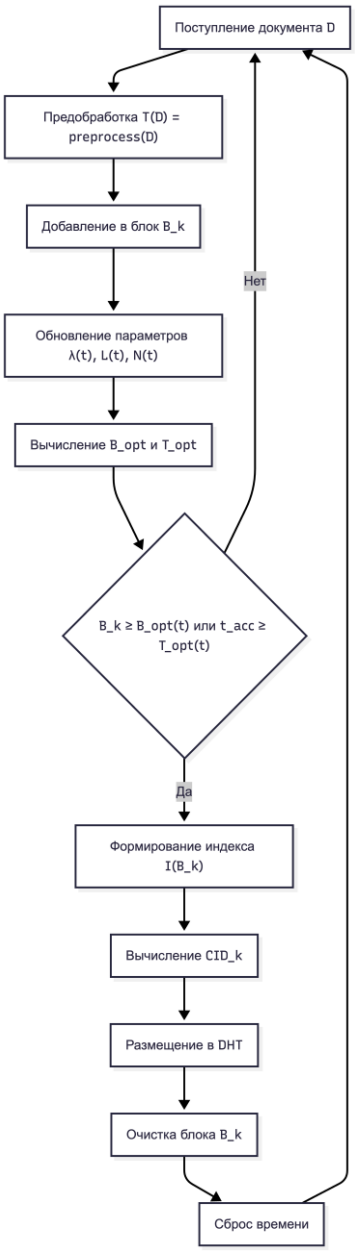

Рис. 2. Алгоритм адаптивного формирования блоков и индексации
Fig. 2. Algorithm of adaptive block formation and indexing

Алгоритм имеет линейную вычислительную сложность O(n) по числу обрабатываемых документов в блоке.

Предложенный алгоритм обладает следующими преимуществами:

- адаптация к динамике сети;
- снижение количества сетевых операций;
- уменьшение избыточной индексации;
- повышение масштабируемости системы;
- гибкость настройки параметров.

Ограничения алгоритма:

- требуется оценка параметров λ(t), L(t), N(t);

- не учитываются приоритеты документов;
- отсутствует учёт семантической значимости данных;
- предполагается синхронность обновления параметров.

ДОКАЗАТЕЛЬСТВО КОРРЕКТНОСТИ РАБОТЫ АЛГОРИТМА

Для подтверждения работоспособности предложенного алгоритма адаптивного формирования блоков рассмотрим теоретический сценарий функционирования системы при различных параметрах входного потока и состояния сети.

Пусть: $\lambda(t)=50$ док/с, $L(t)=0.5$, $N(t)=10$, $\alpha=2$, $\beta=1$, $\varepsilon=0.01$

Расчёт адаптивного размера блока:

$$B_{opt}(t) = 2 * \frac{50}{0.5 + 0.01} * 10 \approx 1961 .$$

Расчёт адаптивного времени:

$$T_{opt}(t) = 1 * \frac{1}{50} * (1 + 0.5) = 0.03$$

Формирование блока происходит при выполнении условий:

$$|B_k| \geq B_{opt}(t) \vee t_{acc} \geq T_{opt}(t) .$$

За время $T_{opt}(t)$ в систему поступает документов:

$$\lambda(t)* T_{opt}(t)= 50*0.03 = 1.5 \approx 2$$

Следовательно:

- условие по времени выполняется быстрее;
- блок формируется по времени, а не по размеру.

Сценарий изменения нагрузки сети.

Пусть: $L(t)=0.9$.

Новый размер блока:

$$B_{opt}(t) = 2 * \frac{50}{0.9 + 0.01} * 10 \approx 1099 .$$

Новое время:

$$T_{opt}(t) = 1 * \frac{1}{50} * (1 + 0.9) = 0.038.$$

Увеличение $L(t)$ приводит к уменьшению $B_{opt}(t)$ и увеличению $T_{opt}(t)$. Это соответствует целям алгоритма:

- снижение сетевой нагрузки;
- уменьшение объёма передаваемых данных.

Сценарий при увеличении числа узлов:

Пусть $N(t) = 20$, тогда новый размер блока:

$$B_{opt}(t) = 2 * \frac{50}{0.5 + 0.01} * 20 \approx 3921.$$

Полученный результат указывает на то, что при росте числа узлов система допускает формирование более крупных блоков. Это объясняется расширением доступных вычислительных и сетевых ресурсов, что позволяет перераспределять нагрузку между участниками сети.

Проведённые расчёты позволяют сделать следующие выводы:

- изменение входных параметров отражается на размере блока ожидаемым образом;
- достигается согласование между нагрузкой на сеть и производительностью обработки;
- поведение системы остаётся устойчивым при варьировании параметров.

С учётом полученных результатов можно считать, что предложенный алгоритм является пригодным для использования в распределённых средах, где характеристики нагрузки изменяются во времени.

СРАВНЕНИЕ С ФИКСИРОВАННЫМ МЕТОДОМ

В качестве базового подхода рассматривается метод с фиксированным размером блока:

$$B_{fix} = 1000,$$

где

B_{fix} – заранее заданный размер блока, не зависящий от параметров системы.

Подобные решения широко применяются на практике благодаря простоте реализации, однако их использование ограничено в условиях изменяющейся нагрузки. Отсутствие зависимости от параметров среды приводит к тому, что система не адаптируется к текущему состоянию сети и входного потока данных.

В предлагаемом методе размер блока определяется динамически и, в рассматриваемом примере, изменяется в диапазоне:

$$B_{opt}(t) \in [1099,3921],$$

что зависит от значений $\lambda(t)$, $L(t)$ и $N(t)$.

Сравнительные характеристики двух подходов приведены в таблице.

Таблица

Сравнение методов

Table

Comparison of methods

Параметр	Фиксированный метод	Адаптивный метод
Размер блока	1000	1099-3921
Учет нагрузки L(t)	Нет	Да
Учет потока данных λ(t)	Нет	Да
Учет числа узлов N(t)	Нет	Да
Гибкость	Низка	Высокая
Адаптация к динамике среды	Отсутствует	Реализована
Сетевые издержки	Фиксированные, не оптимизируются	Снижаются за счет адаптации
Время отклика	Нестабильное при изменении нагрузки	Стабилизируется

По результатам проведенного сравнения можно увидеть, что фиксированный метод не учитывает изменение параметров распределенной системы, поэтому эффективность его применения будет зависеть от текущего состояния системы. При высокой интенсивности потока это может приводить к увеличению числа операций записи и индексирования, тогда как при низкой нагрузке возрастает время формирования блоков.

Исходя из нашего сравнения, можно сделать вывод, что адаптивный метод более устойчив к изменению параметров среды и обеспечивает более предсказуемое поведение системы при различных режимах нагрузки.

СЦЕНАРИИ ВНЕДРЕНИЯ АЛГОРИТМА

Предложенный алгоритм адаптивного формирования блоков и индексирования неструктурированных данных может быть применён в различных классах децентрализованных систем, где наблюдается высокая динамика потоков данных и ограниченность сетевых ресурсов. Ниже рассмотрены ключевые сценарии его внедрения.

1. Децентрализованные системы хранения данных (IPFS-подобные системы)

Сценарий внедрения:

В системах контент-адресуемого хранения, таких как IPFS и аналогичные решения, данные распределяются по узлам сети и идентифицируются с использованием хеш-функций [4]. При

обработке большого объёма неструктурированных текстовых данных возникает проблема избыточной фрагментации и увеличения количества мелких блоков.

Рекомендации по применению.

Использование предлагаемого алгоритма целесообразно в следующих случаях:

при высокой интенсивности поступления данных $\lambda(t)$;

- при увеличении количества мелких файлов и документов;
- при необходимости оптимизации распределённого индексирования.

Алгоритм позволяет:

- агрегировать данные в адаптивные блоки;
- уменьшить количество операций записи в DHT;
- повысить эффективность поиска за счёт укрупнённых индексных структур.

Потенциальные проблемы и вызовы:

- увеличение времени доступа к отдельным документам внутри блока;
- необходимость адаптации существующих механизмов chunking;
- дополнительные вычислительные затраты на адаптацию параметров.

Возможные решения:

- использование многоуровневой индексации;
- внедрение кэширования популярных данных;
- оптимизация параметров α и β под конкретную нагрузку.

2. Децентрализованные поисковые системы

Сценарий внедрения:

В распределённых системах поиска текстовых данных (например, P2P search engines) индексирование является одной из наиболее ресурсоёмких операций [8]. При этом статические методы формирования индексов приводят к увеличению задержек и неравномерной нагрузке на узлы.

Рекомендации по применению.

Алгоритм рекомендуется использовать:

- при необходимости масштабирования поисковой системы;
- при высокой вариативности запросов;
- в условиях неоднородной загрузки узлов.

Использование адаптивного формирования блоков позволяет:

- уменьшить количество индексных структур;
- снизить время поиска за счёт агрегированных индексов;
- обеспечить баланс нагрузки между узлами.

Потенциальные проблемы:

- рост размера индексных структур;
- сложность обновления индексов;
- возможное увеличение латентности при доступе к данным.

Возможные решения:

- применение инкрементального индексирования;
- использование распределённых кэшей;
- оптимизация структуры индекса.

3. Системы потоковой обработки данных

Сценарий внедрения:

В системах потоковой обработки (stream processing), таких как распределённые аналитические платформы, данные поступают непрерывно и требуют оперативной обработки. В таких условиях фиксированные параметры буферизации могут приводить к перегрузке системы. Современные системы потоковой обработки данных, такие как описанные в [12, 13], используют динамическое управление окнами обработки, что концептуально близко к предлагаемому подходу.

Рекомендации по применению.

Алгоритм эффективен:
- при переменной интенсивности потоков данных;
- в системах реального времени;
- при ограниченных сетевых ресурсах.

Адаптивная модель позволяет:
- динамически изменять размер буфера (блока);
- предотвращать перегрузку сети;
- оптимизировать задержки обработки.

Потенциальные проблемы:
- необходимость точной оценки параметров λ(t), L(t);
- сложность интеграции с существующими streaming-фреймворками.

Возможные решения:
- использование методов мониторинга нагрузки;
- внедрение механизмов прогнозирования потока данных;
- интеграция с системами управления потоками.

РЕКОМЕНДАЦИИ ПО ИНТЕГРАЦИИ АЛГОРИТМА

Интеграция алгоритма адаптивного формирования блоков в существующие децентрализованные системы требует поэтапного подхода и учёта архитектурных особенностей конкретной платформы.

1. Условия целесообразности внедрения.

Использование алгоритма рекомендуется в следующих случаях:
- высокая интенсивность входного потока данных;
- наличие перегрузок сети (высокие значения L(t));
- необходимость масштабирования системы;
- большое количество мелких данных.

2. Случаи, когда внедрение нецелесообразно

Алгоритм может быть избыточным:
- в системах с низкой нагрузкой;
- при малом объёме данных;
- в централизованных архитектурах с фиксированными ресурсами.

3. Этапы интеграции алгоритма

Этап 1: Анализ системы:
- определение текущих параметров: λ(t), L(t), N(t);
- выявление узких мест;
- оценка требований к производительности.

Этап 2: Модификация архитектуры:
- внедрение модуля адаптивного формирования блоков;
- интеграция с системой индексирования;
- обеспечение совместимости с DHT.

Этап 3: Реализация мониторинга:
- сбор метрик нагрузки сети;
- оценка интенсивности потока данных;
- отслеживание производительности.

Этап 4: Настройка параметров модели:
- выбор коэффициентов α, β;
- определение пороговых значений;
- проведение тестирования.

Этап 5: Тестирование:
- нагрузочное тестирование;
- проверка устойчивости системы;
- анализ времени отклика.

Этап 6: Развёртывание:
- поэтапное внедрение алгоритма;
- обновление узлов сети;
- контроль корректности работы.

Этап 7: Мониторинг и оптимизация:
- анализ ключевых метрик (время формирования блока, нагрузка сети, скорость поиска);
- корректировка параметров модели.

4. Практические рекомендации:
- использовать адаптивный алгоритм совместно с кэшированием;
- учитывать неоднородность узлов сети;
- применять гибкую настройку параметров;
- проводить регулярную оптимизацию модели.

ЗАКЛЮЧЕНИЕ

В работе предложен метод адаптивного формирования блоков и индексирования неструктурированных данных в децентрализованных системах хранения, учитывающий динамические характеристики распределённой среды, включая интенсивность поступления данных, загрузку сети и количество активных узлов.

Разработанная архитектура системы объединяет этапы предобработки, агрегации, индексирования и последующего размещения данных в распределённой среде. В отличие от ряда существующих решений, в основе предложенного подхода лежит динамическое определение параметров формирования блоков, включая их размер и время накопления, что позволяет учитывать текущее состояние сети и характеристики входного потока данных.

В рамках исследования предложена математическая модель, описывающая механизм адаптации указанных параметров, а также реализующий её алгоритм. Теоретический анализ показал, что алгоритм корректно функционирует при различных сценариях. При этом достигается согласование между уровнем сетевой нагрузки и эффективностью поиска.

Рассмотренные варианты применения и предложенные рекомендации по внедрению подтверждают, что метод может использоваться в различных типах децентрализованных систем – от распределённых хранилищ до поисковых платформ и систем потоковой обработки данных.

Предложенный подход может быть использован как основа для повышения эффективности обработки неструктурированных данных в распределённых средах хранения, особенно в условиях изменяющейся нагрузки и неоднородности сети.

Список литературы

1. Gandomi A., Haider M. Beyond the hype: Big data concepts, methods, and analytics // International Journal of Information Management. – 2015. – Vol. 35, No. 2. – P. 137–144.
2. Михнев И.П. Цифровые технологии Big Data в современном высшем образовании: технологии поиска и обработки неструктурированной информации // Материалы конференции. – Новосибирск, 2019. – С. 326–329.
3. Фирова Д.В., Барышникова М.Ю. Обзор методов извлечения данных из неструктурированных документов // Матрица научного познания. – 2022. – № 2-1. – С. 56–71.
4. Benet J. IPFS – Content Addressed, Versioned, P2P File System // arXiv preprint arXiv:1407.3561, 2014.
5. Maymounkov P., Mazieres D. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric // IPTPS. – 2002. – P. 53–65.
6. Tanenbaum A.S., van Steen M. Distributed Systems: Principles and Paradigms. – Pearson, 2007.
7. Manning C.D., Raghavan P., Schütze H. Introduction to Information Retrieval. – Cambridge University Press, 2008.

8. Xu Y., Chen L. Efficient Indexing and Query Processing in Distributed Text Retrieval Systems // IEEE Access. – 2020. – Vol. 8. – P. 112345–112357.
9. Li J., Chen X. Decentralized Storage Systems: Architecture and Challenges // IEEE Access. – 2020. – Vol. 8. – P. 227093–227105.
10. Stoica I. et al. Chord: A Scalable Peer-to-Peer Lookup Protocol // IEEE/ACM Transactions on Networking. – 2003. – Vol. 11. – P. 17–32.
11. Wang S., Li X. Dynamic Resource Allocation in Distributed Networks // IEEE Transactions on Cloud Computing. – 2022. – Vol. 10. – P. 45–58.
12. Carbone P. et al. Apache Flink: Stream and Batch Processing in a Single Engine // IEEE Data Engineering Bulletin. – 2015.
13. Akidau T. et al. The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale Systems // VLDB. – 2015.
14. Kleinrock L. Queueing systems. Volume 1: Theory. – New York: Wiley, 1975. – 417 p.

References

1. Gandomi A., Haider M. Beyond the hype: Big data concepts, methods, and analytics // International Journal of Information Management. – 2015. – Vol. 35, No. 2. – P. 137–144.
2. Mikhnev I.P. Digital technologies of Big Data in modern higher education: technologies of search and processing of unstructured information // Conference proceedings. Novosibirsk, 2019. – pp. 326-329.
3. Firova D.V., Baryshnikova M.Y. Review of data extraction methods from unstructured documents // Matrix of scientific knowledge. - 2022. – No. 2-1. – pp. 56-71.
4. Benet J. IPFS – Content Addressed, Versioned, P2P File System // arXiv preprint arXiv:1407.3561, 2014.
5. Maymounkov P., Mazieres D. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric // IPTPS. – 2002. – P. 53–65.
6. Tanenbaum A.S., van Steen M. Distributed Systems: Principles and Paradigms. – Pearson, 2007.
7. Manning C.D., Raghavan P., Schütze H. Introduction to Information Retrieval. – Cambridge University Press, 2008.
8. Xu Y., Chen L. Efficient Indexing and Query Processing in Distributed Text Retrieval Systems // IEEE Access. – 2020. – Vol. 8. – P. 112345–112357.
9. Li J., Chen X. Decentralized Storage Systems: Architecture and Challenges // IEEE Access. – 2020. – Vol. 8. – P. 227093–227105.
10. Stoica I. et al. Chord: A Scalable Peer-to-Peer Lookup Protocol // IEEE/ACM Transactions on Networking. – 2003. – Vol. 11. – P. 17–32.
11. Wang S., Li X. Dynamic Resource Allocation in Distributed Networks // IEEE Transactions on Cloud Computing. – 2022. – Vol. 10. – P. 45–58.
12. Carbone P. et al. Apache Flink: Stream and Batch Processing in a Single Engine // IEEE Data Engineering Bulletin. – 2015.
13. Akidau T. et al. The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale Systems // VLDB. – 2015.
14. Kleinrock L. Queueing systems. Volume 1: Theory. – New York: Wiley, 1975. – 417 p.

Воскобойников Илья Сергеевич, соискатель по специальности 2.3.1. Системный анализ, управление и обработка информации, статистика, Белгородский государственный технологический университет им. В.Г. Шухова, г. Белгород, Россия
Гвоздевский Игорь Николаевич, кандидат технических наук, доцент, начальник управления информатизации, Белгородский государственный технологический университет им. В.Г. Шухова, г. Белгород, Россия
Булгаков Владислав Дмитриевич, соискатель по специальности 2.3.1. Системный анализ, управление и обработка информации, статистика, Белгородский государственный технологический университет им. В.Г. Шухова, г. Белгород, Россия

Voskoboinikov Ilia Sergeevich, Applicant in the Specialty 2.3.1. System analysis, management and information processing, statistics, Belgorod State Technological University named after V.G. Shukhov, Belgorod, Russia
Gvozdevsky Igor Nikolaevich, Candidate of Technical Sciences, Associate Professor, Head of the Informatization Department of the Belgorod State Technological University named after V.G. Shukhov, Belgorod, Russia
Bulgakov Vladislav Dmitrievich, Applicant in the Specialty 2.3.1. System analysis, management and information processing, statistics, Belgorod State Technological University named after V.G. Shukhov, Belgorod, Russia