УДК 004.627 doi: 10.15827/0236-235X.142.170-177 2024. Т. 37. № 2. С. 170–177

Сравнительный анализ методов сжатия табличных данных

К.В. Гарев [1, 2]✉

[1] ФБГУ «Российский центр научной информации», г. Москва, 119334, Россия
[2] Межведомственный суперкомпьютерный центр РАН, г. Москва, 119334, Россия

Ссылка для цитирования
Гарев К.В. Сравнительный анализ методов сжатия табличных данных // Программные продукты и системы. 2024. Т. 37. № 2. С. 170–177. doi: 10.15827/0236-235X.142.170-177
Информация о статье
Группа специальностей ВАК: 2.3.1
Поступила в редакцию: 20.12.2023 После доработки: 24.01.2024 Принята к публикации: 14.02.2024

Аннотация. В работе проанализированы методы компрессии табличных данных, используемых в рамках функционирования платформы «Инфраструктура научно-исследовательских данных» (Платформа ИНИД). Оператором платформы является Российский центр научной информации. Основная цель исследования – выявление наиболее оптимальных методик сжатия данных, способных быть интегрированными в структуру Платформы ИНИД для доработки функционала по обмену и управлению научными данными. В ходе работы проведен тщательный анализ пяти наиболее популярных методов сжатия данных, доступных для реализации с использованием программных средств Python: Deflate (gzip), LZMA, Bzip2, Brotli и Snappy. Для каждой из рассмотренных технологий сжатия выявлены их преимущества и недостатки с учетом специфики обработки табличных данных, в том числе коэффициента сжатия, скорости обработки и степени сохранности информации. Результаты данного исследования могут использоваться в практике обмена, хранения, обработки и анализа табличных данных в рамках Платформы ИНИД. Особое внимание уделено анализу преимуществ и недостатков каждого из методов сжатия. Это позволяет сформировать рекомендации по выбору наиболее подходящих технологий, соответствующих строгим требованиям к производительности, эффективности сжатия и надежности сохранности данных. Важным аспектом исследования является фокусировка на возможности оптимизации процессов внутри Платформы ИНИД, что направлено на повышение ее эффективности как инструмента в области работы с научными данными. Кроме того, данная работа способствует углублению понимания потенциала применения современных методов сжатия данных в контексте практик обмена научными данными, открывая перспективу для дальнейших исследований и разработок в этой области.

Ключевые слова: методы сжатия, алгоритмы сжатия, Deflate, LZMA, Bzip2, Brotli, Snappy, сравнительный анализ

Введение. В последнее время объемы генерируемых и обрабатываемых данных в цифровом мире демонстрируют бурный рост [1]. Это касается как индивидуальных пользователей, так и крупных организаций, сталкивающихся с необходимостью эффективного хранения и передачи информации. В таком контексте алгоритмы сжатия данных приобретают особую актуальность, поскольку позволяют существенно уменьшить размеры файлов и, как следствие, сократить расходы на хранение данных, а также ускорить процесс их передачи через сети.

Каждый из существующих алгоритмов сжатия имеет уникальные характеристики, определяющие их предназначение и области применения. От выбора метода сжатия и реализующего его алгоритма зависят не только конечный размер файла, но и скорость его сжатия-распаковки, что влияет на общую производительность системы. Такие алгоритмы, как LZ77 и LZ78, разработанные в конце 1970-х годов, положили начало целому поколению методов сжатия, каждый из которых по-своему оптимизирует процесс уменьшения размера данных без потерь [2].

В последующие годы были разработаны и внедрены такие алгоритмы, как Deflate, который был представлен как эффективное сочетание сжатия LZ77 с кодированием Хаффмана, Brotli, обеспечивающий более высокие коэффициенты сжатия для веб-данных, а также Snappy и LZO, ориентированные на максимальную скорость сжатия и распаковки. Каждый из этих алгоритмов имеет свое предназначение, будь то максимально возможное сжатие данных или их быстрая обработка без значительных временных затрат.

Данная работа посвящена сравнительному анализу современных методов сжатия данных. Цель исследования – оценить эффективность каждого метода с точки зрения коэффициента сжатия и скорости обработки для последующего использования в рамках функционирования платформы «Инфраструктура научно-исследовательских данных» (Платформа ИНИД) (https://data.rcsi.science/).

Описание алгоритмов сжатия

Цель сжатия – уменьшить размер файлов для экономии пространства на носителях ин-

формации и ускорения передачи данных по сети. Алгоритмы сжатия данных можно классифицировать на две большие категории: сжатие с потерями и без потерь. Каждый алгоритм имеет свои уникальные методы для достижения сжатия.

Сжатие без потерь позволяет точно восстановить исходные данные, в то время как сжатие с потерями приводит к уменьшению размера за счет невозможности точного восстановления оригинала, что приемлемо, например, в аудио, видео и изображениях. Сжатие без потерь использует такие методы, как кодирование Хаффмана, алгоритмы LZ77 и LZ78 и их производные (например, Deflate), а также преобразование Берроуза–Уилера с последующим кодированием длин серий (RLE) и кодированием Хаффмана (как в Bzip2).

Сжатие с потерями применяется в основном к мультимедийным данным, где небольшие потери качества могут быть незаметны человеку или приемлемы с точки зрения конечного использования. Примеры включают JPEG для изображений, MPEG для видео и MP3 для аудио.

Основой теории сжатия данных является понятие энтропии, введенное Клодом Шенноном [3]. Энтропия характеризует среднее количество информации (в битах), содержащееся в каждом символе данных, и определяет теоретический предел сжатия, который может быть достигнут без потерь для заданного набора данных. Алгоритмы сжатия стремятся приблизиться к этому пределу, уменьшая избыточность информации.

Рассмотрим наиболее известные алгоритмы сжатия.

Deflate – алгоритм сжатия данных, введенный Филом Кацем в 1993 году и используемый во многих форматах и протоколах, включая ZIP, gzip и PNG. Алгоритм сочетает в себе два основных метода сжатия – LZ77 (Lempel-Ziv 1977) и кодирование Хаффмана. Это позволяет эффективно сжимать данные, уменьшая их размер без потерь [4].

Рассмотрим подробнее основные шаги Deflate.

Первая часть Deflate – это алгоритм сжатия LZ77 (сжатие на базе словаря), в основе которого лежит процедура поиска и замены повторяющихся последовательностей символов в данных. LZ77 использует так называемый словарь – буфер, содержащий уже обработанные данные, для поиска повторяющихся последовательностей.

Прежде всего алгоритм ищет последовательности символов, которые ранее уже встречались в данных. Если такая последовательность найдена, она заменяется ссылкой на ее предыдущее вхождение в словарь. Эта ссылка обычно представляет собой пару значений: смещение назад по тексту до начала повторяющейся последовательности и ее длину. Для данных, которые не могут быть сжаты с помощью ссылок на словарь (то есть уникальные или не имеющие совпадений в словаре), используется их непосредственное представление.

После применения LZ77 к данным Deflate использует кодирование Хаффмана для дальнейшего сжатия. Это метод сжатия данных без потерь, который использует переменную длину кода для представления символов. Символы, встречающиеся чаще, кодируются более короткими кодами, а редкие – более длинными.

Таким образом, сначала на основе частоты встречаемости символов и последовательностей в сжимаемых данных строится дерево Хаффмана. Каждому символу или последовательности присваивается уникальный двоичный код в соответствии с его положением в дереве.

Затем данные кодируются с использованием полученных двоичных кодов, что приводит к их дальнейшему уменьшению в размере.

Итоговый сжатый поток данных в формате Deflate состоит из серии блоков, каждый из которых может быть закодирован независимо. Блоки могут использовать различные стратегии кодирования в зависимости от того, какой подход обеспечивает лучшее сжатие для конкретного набора данных. Deflate также позволяет использовать для кодирования фиксированное или динамическое дерево Хаффмана, что дает дополнительную гибкость в оптимизации процесса сжатия.

LZMA (*Lempel-Ziv-Markov chain Algorithm*) – алгоритм сжатия данных без потерь, который сочетает в себе методы словарного сжатия и адаптивное кодирование арифметическим кодом [5]. Он базируется на алгоритме LZ77 и расширяет его возможности, улучшая эффективность сжатия за счет использования более сложных моделей предсказания.

Процесс сжатия с использованием LZMA начинается с построения словаря входных данных. Затем алгоритм находит повторяющиеся последовательности байтов. Каждая повторяющаяся последовательность заменяется ссылкой на ее предыдущее вхождение в словарь. Этот процесс позволяет сократить размер данных за счет удаления избыточности информации.

После этапа словарного сжатия применяется адаптивное кодирование арифметическим кодом, используемое для дальнейшего умень-

шения размера данных. На этом этапе каждый символ заменяется кодом, который зависит от вероятности его появления в текущем контексте. Алгоритм адаптивно обновляет значения вероятностей появления символов на основе предыдущих символов, что позволяет ему эффективно сжимать данные с учетом их структуры.

Одним из ключевых преимуществ LZMA является высокий коэффициент сжатия, особенно для файлов большого размера или данных с высокой степенью избыточности. Однако этот высокий коэффициент сжатия достигается за счет использования более сложных алгоритмов, что может привести к увеличению времени сжатия и распаковки данных по сравнению с некоторыми другими методами сжатия.

Bzip2 – алгоритм сжатия данных без потерь, который использует алгоритм Берроуза–Уилера (Burrows–Wheeler Transform, BWT) в сочетании с кодированием длин серий (Run-Length Encoding, RLE) [6] и кодированием Хаффмана. Алгоритм разработан Джулианом Сьюардом, изначально был представлен в 1996 году и предназначен для сжатия текстовых данных с высоким коэффициентом сжатия при приемлемой скорости выполнения [7].

Процесс сжатия в Bzip2 включает несколько ключевых шагов. На первом этапе исходные данные переупорядочиваются таким образом, чтобы создать условия для более эффективного последующего сжатия. BWT группирует похожие символы вместе, делая текст более однородным и улучшая условия для следующих этапов сжатия. После применения BWT исходный текст может содержать серии повторяющихся символов. RLE сокращает эти серии до более компактного представления, записывая символ и количество его последовательных повторений. Затем осуществляется сортировка по контексту. Этот шаг помогает повысить эффективность последующего кодирования Хаффмана, дополнительно упорядочивая символы по их контексту. На последнем этапе применяется кодирование Хаффмана, которое заменяет часто встречающиеся символы или группы символов короткими кодами, а редкие – более длинными. Это дополнительно сжимает данные, основываясь на их частотности.

Алгоритм Bzip2 обрабатывает данные блоками, обычно размером в 900 КБ. Каждый блок сжимается независимо, что позволяет эффективно распараллелить процесс сжатия и распаковки, а также облегчает восстановление данных при частичном повреждении архива.

Brotli – алгоритм сжатия данных, разработанный Google и предназначенный в основном для сжатия веб-контента [8]. Он обеспечивает высокий коэффициент сжатия при сохранении приемлемой скорости обработки данных.

Процесс сжатия с использованием Brotli начинается с построения словаря входных данных, который затем используется для поиска повторяющихся последовательностей. После этого применяются различные методы адаптивного кодирования, такие как кодирование Хаффмана и предсказательное кодирование, для дальнейшего уменьшения размера данных. Алгоритм также может использовать контекстное моделирование для эффективного сжатия текстовых данных.

Одним из ключевых преимуществ Brotli является специальная оптимизация для веб-контента, что делает его особенно эффективным для сжатия HTML, CSS, JavaScript и других типов файлов, используемых на веб-страницах. Он также предлагает высокий коэффициент сжатия при высокой скорости обработки данных, что делает его привлекательным для использования в онлайн-приложениях и сервисах.

Snappy – алгоритм сжатия данных, также разработанный Google. Применяется в широком спектре приложений, где требуется высокая скорость сжатия и распаковки данных при относительно низком коэффициенте сжатия [9].

Процесс сжатия с использованием Snappy основан на алгоритме сжатия данных без потерь, обеспечивающем быстрое сжатие и распаковку данных путем замены повторяющихся последовательностей байтов ссылками на их предыдущие вхождения в словаре. Этот метод сжатия приводит к относительно низкому коэффициенту сжатия по сравнению с некоторыми другими алгоритмами, но при этом обеспечивает высокую скорость обработки данных.

Одним из ключевых преимуществ Snappy является высокая скорость сжатия и распаковки данных. Он оптимизирован для обеспечения максимальной производительности при сжатии и распаковке данных в реальном времени, что делает его идеальным для использования в онлайн-сервисах и приложениях, где скорость обработки данных критически важна.

Следует отметить, что Snappy может быть менее эффективным в сжатии данных с высокой степенью избыточности, таких как текстовые данные или данные с повторяющимися паттернами. Это следует учитывать при выборе метода сжатия в зависимости от конкретных характеристик данных и требований приложения.

Критерии сравнения алгоритмов сжатия

Для определения критериев отбора методов следует учитывать ряд аспектов.

1. *Коэффициент сжатия*. Это ключевой показатель эффективности метода сжатия. Важно выбрать методы сжатия, обеспечивающие наиболее высокий коэффициент, что позволит минимизировать объем хранимых и передаваемых данных, сокращая затраты на хранение и улучшая производительность системы. В данном контексте коэффициент сжатия определяется как отношение объема несжатого файла к объему сжатого.

2. *Сохранность данных*. Учитывая биологическую значимость данных о структурных мотивах белков, важно обеспечить сохранность информации при сжатии. Чтобы избежать искажений или потерь важных деталей структуры белков, предпочтительно использовать такие методы сжатия без потерь, как Deflate, LZMA и Bzip2.

3. *Скорость сжатия.* Важным критерием является скорость сжатия данных. Учитывая, что БД может содержать большие объемы информации, желательно выбирать методы сжатия, обеспечивающие приемлемую скорость обработки данных. Это позволит эффективно работать с данными и обеспечивать быстрый доступ к ним.

4. *Поддержка и совместимость.* Важно выбрать методы сжатия, которые хорошо поддерживаются в современных программных и аппаратных системах, а также имеют широкую совместимость с используемыми платформами и инструментами. Это обеспечит легкость интеграции выбранных методов существующей инфраструктурой и упростит их использование.

Тестовым набором данных, структура которого показана в таблице 1, является один из датасетов, представленных на Платформе ИНИД «База данных структурных мотивов белков: биологические и физико-химические свойства». Набор данных предназначен для проведения структурного анализа белков, ассоциированных с развитием заболеваний, в том числе аберрантных форм белков, которые образованы вследствие аминокислотных замен, модифицирования после синтеза. Набор данных может применяться для решения прикладных медико-биологических задач, таких как разработка новых подходов к диагностике заболеваний, изучение молекулярных основ патогенеза, выявление мишеней белкового происхождения для лекарственных средств и проектирование миметиков (белков с заданными свойствами). Набор данных содержит 3.96 млн аннотаций структурных мотивов в белковых структурах с указанием внутренних координат (http://data.rcsi.science/data-catalog/datasets/203/).

Сравнительный анализ алгоритмов сжатия

Сравниваемые алгоритмы были реализованы посредством использования python-библиотек. Вычислительный эксперимент проводился с использованием вычислительных мощностей виртуальной машины со следующими характеристиками: 32 ядра, Intel Xeon, 72 Гб оперативной памяти, сервер под управлением Linux Debian 11.

Результаты сравнительного анализа представлены в таблице 2.

Графическая интерпретация результатов сравнения алгоритмов представлена на рисунке.

Таким образом, на основе проведенного анализа и данных вычислительного эксперимента можно выделить преимущества и недостатки алгоритмов сжатия.

Deflate. Преимущества: баланс между скоростью и коэффициентом сжатия. Deflate обеспечивает относительно высокий коэффициент сжатия при сохранении приемлемой скорости сжатия и распаковки.

Широкая поддержка: поддерживается большинством программных и аппаратных систем, что делает его универсальным выбором для сжатия табличных данных.

Недостатки: не оптимизирован для табличных данных. Может быть не таким эффективным для специфических типов табличных данных, как другие алгоритмы, специально разработанные для этой цели.

LZMA. Преимущества: высокий коэффициент сжатия. LZMA обеспечивает один из самых высоких коэффициентов сжатия среди алгоритмов без потерь, что может быть критически важным для больших объемов табличных данных.

Эффективность сжатия: особенно эффективен для сжатия больших наборов данных с высокой избыточностью информации.

Недостатки: скорость сжатия и распаковки. LZMA обычно работает медленнее других алгоритмов, что может быть ограничивающим фактором для реальных приложений, требующих быстрого доступа к данным.

Bzip2. Преимущества: высокий коэффициент сжатия для определенных типов данных. Bzip2 часто превосходит gzip по коэффициенту

Таблица 1

Структура набора данных

Table 1

Dataset structure

Атрибут	Описание	Количество пропусков (NaN) (тш/%)	Формат
experiment_id	Идентификатор трехмерной структуры в Protein Data Bank	0	String
uniprot_id	Идентификатор белка в базе знаний UniProt	1330	String
protein_name	Название белка	101 949	String
chain_id	Идентификатор белковой цепи в экспериментальной структуре	98 257	String
resolution	Разрешение трехмерной структуры белка	2 712 626	Unmeric
experiment_type	Тип эксперимента, в котором получена трехмерная структура белка	0	String
biological_process	Биологический(е) процесс(ы), в которых участвует белок	2 436 633	String
molecular_function	Функция(и), которую выполняет белок в клетке/организме	2 006 228	String
fasta	Аминокислотная последовательность структурного мотива в формате однобуквенного кода	0	String
apfid	Уникальный идентификатор мотива, который содержит указание на экспериментальную структуру, позиции начала и конца мотива	0	String
motif_type	Тип структурного мотива	0	String
motif_len	Длина структурного мотива	0	Integer
motif_start	Позиция начала структурного мотива в цепи белка	1 567	Integer
motif_end	Позиция конца структурного мотива в цепи белка	1 567	Integer
organism	Название организма, которому принадлежит белок (англ.)	98 257	String

сжатия, особенно для файлов большого размера. Это делает его предпочтительным для сжатия объемных табличных данных.

Недостатки: скорость сжатия и распаковки, в сравнении с такими алгоритмами, как Snappy или Deflate, работает медленнее.

Использование ресурсов: Bzip2 может требовать больше оперативной памяти для сжатия и распаковки по сравнению с некоторыми другими алгоритмами, что важно учитывать при работе с ограниченными ресурсами или большими объемами данных.

Brotli. Преимущества: оптимизирован для веб-данных. Brotli был специально разработан для сжатия веб-контента, что делает его особенно подходящим для сжатия табличных данных, передаваемых через веб.

Высокий коэффициент сжатия при высокой скорости: обеспечивает отличный коэффициент сжатия, при этом оставаясь достаточно быстрым для онлайн-приложений.

Недостатки: совместимость и поддержка. Хотя поддержка Brotli растет, она все еще не так универсальна, как у более старых алгоритмов, таких как gzip.

Snappy. Преимущества: высокая скорость сжатия и распаковки. Snappy оптимизирован для быстрого сжатия и распаковки, что делает его подходящим алгоритмом для ситуаций, где время обработки критично.

Таблица 2

Сравнение алгоритмов сжатия

Table 2

Comparison of compression algorithms

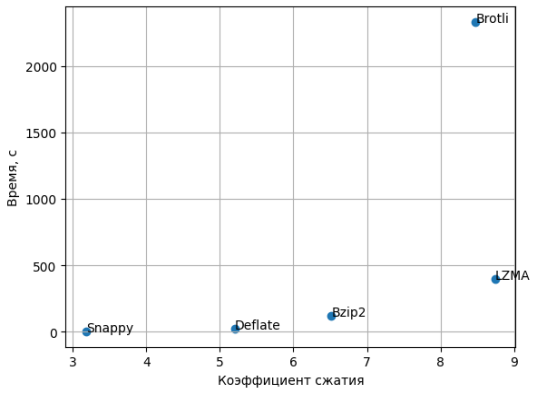
Метод сжатия	Алгоритм сжатия	Коэффициент сжатия	Время сжатия
LZ77 + Huffman	Deflate	5,203	0:00:25.282951
LZ77 + марковские цепи	LZMA	8,742	0:06:38.348505
BWT + RLE + Huffman	Bzip2	6,517	0:01:59.607255
	Brotli	8,476	0:38:52.891844
Поиск совпадений (как в LZ77)	Snappy	3,182	0:00:03.906618

Недостатки: коэффициент сжатия. Может не достигать такого же уровня сжатия, как некоторые другие алгоритмы, особенно для данных с высокой избыточностью.

Практическое применение

Использование оптимальных методов сжатия данных, выявленных в рамках исследования, в составе Платформы ИНИД и других информационных систем, оперирующих сверхбольшими объемами данных, позволит значительно оптимизировать процессы обмена, хранения и анализа табличных данных.

Интеграция результатов исследования в Платформу ИНИД обеспечит пользователям возможность эффективно управлять объемными наборами данных и уменьшить нагрузку на инфраструктуру. Благодаря выбору оптимальных методов сжатия, основанному на реальных данных и анализе их характеристик, платформа сможет предоставлять пользователям быстрый доступ к информации и обеспечивать высокую скорость обработки данных.



Графическое представление сравнения алгоритмов сжатия

Graphical representation of comparison of compression algorithms

Использование результатов исследования в Платформе ИНИД также способствует повышению ее конкурентоспособности и привлекательности для научного сообщества. Эффективное управление научными данными, обеспечиваемое оптимальными методами сжатия, будет способствовать развитию научных исследований в различных областях.

Заключение

В ходе исследования был проведен сравнительный анализ пяти наиболее распространенных методов сжатия данных – Deflate, LZMA, Bzip2, Brotli и Snappy с использованием табличных данных, представляющих БД структурных мотивов белков. Каждый метод был оценен по коэффициенту сжатия и времени сжатия для определения их эффективности и производительности в контексте данного исследования.

На основе полученных результатов можно сделать следующие выводы.

Deflate продемонстрировал приемлемый коэффициент и низкое время сжатия, что делает его подходящим для быстрой обработки данных с относительно хорошим коэффициентом сжатия.

LZMA показал высокий коэффициент сжатия, однако его время сжатия значительно превышает время других методов, что делает его менее подходящим для приложений, требующих быстродействия.

Bzip2 продемонстрировал хороший баланс между коэффициентом и временем сжатия, что делает его универсальным методом для различных приложений сжатия данных.

Brotli показал высокий коэффициент сжатия, но его время сжатия значительно превышает время других методов, что делает метод менее подходящим для сценариев, где требуется быстрая обработка данных.

Snappy продемонстрировал низкий коэффициент сжатия, однако обеспечил наименьшее время, что делает его подходящим для задач, где скорость обработки данных приоритетна.

Таким образом, выбор метода сжатия должен основываться на конкретных требованиях и характеристиках приложения, включая коэффициент сжатия, скорость обработки данных и доступные ресурсы.

Список литературы

1. Manyika J., Chui M., Brown B., Bughin J., Dobbs R. et al. Big Data: The Next Frontier for Innovation, Competition, and Productivity. McKinsey & Company, 2011, 146 p. URL: https://personal.utdallas.edu/~muratk/courses/cloud11f_files/MGI-full-report.pdf (дата обращения: 10.01.2024).
2. Choudhary S.M., Patel A.S., Parmar S.J. Study of LZ77 and LZ78 data compression techniques. IJESIT, 2015, vol. 4, no. 3, pp. 2319–5967.
3. Shannon C.E. A mathematical theory of communication. The Bell System Tech. J., 1948, vol. 27, no. 3, pp. 379–423. doi: 10.1002/j.1538-7305.1948.tb01338.x.
4. Oswal S., Singh A., Kumari K. Deflate compression algorithm. IJERGS, 2016, vol. 4, no. 1, pp. 430–436.
5. Adjeroh D., Bell T., Mukherjee A. The Burrows-Wheeler Transform: Data compression, Suffix Arrays, and Pattern Matching. NY, Springer Publ., 2008, 352 p. doi: 10.1007/978-0-387-78909-5.
6. Sarika S., Srilali S. Improved run length encoding scheme for efficient compression data rate. IJERA, 2013, vol. 3, no. 6, pp. 2248–9622.
7. Gupta A., Bansal A., Khanduja V. Modern lossless compression techniques: Review, comparison and analysis. Proc. ICECCT, 2017, pp. 1–8. doi: 10.1109/ICECCT.2017.8117850.
8. Alakuijala J., Farruggia A., Ferragina P. et al. Brotli: A general-purpose data compressor. ACM Transactions on Inform. Sys., 2019, vol. 37, pp. 1–30. doi: 10.1145/3231935.
9. Chen R., Xu L. SnappyR: A new high-speed lossless data compression algorithm. Proc. DCC, 2023, pp. 334–334. doi: 10.1109/DCC55655.2023.00052.

Software & Systems doi: 10.15827/0236-235X.142.170-177 2024, 37(2), pp. 170–177

Compression methods for tabular data: Comparative analysis

Kirill V. Garev [1, 2]✉

[1] Russian Center for Science Information, Moscow, 119334, Russian Federation
[2] Joint Supercomputer Center of RAS, Moscow, 119334, Russian Federation

For citation
Garev, K.V. (2024) 'Compression methods for tabular data: Comparative analysis', *Software & Systems*, 37(2), pp. 170–177 (in Russ.). doi: 10.15827/0236-235X.142.170-177
Article info
Received: 20.12.2023 After revision: 24.01.2024 Accepted: 14.02.2024

Abstract. This paper presents a comparative analysis of table data compression methods used within in terms of the Research Data Infrastructure platform (RDI Platform) operated by the Russian Center for Science Information. The main purpose of the study was to identify the most optimal data compression methods that can be integrated into the RDI Platform structure to enhance the functionality of data exchange and management. The author of the study has carried out a thorough analysis of the five most popular data compression methods available for implementation using the following Python software tools: Deflate (gzip), LZMA, Bzip2, Brotli, and Snappy. The author has analyzed advantages and disadvantages of each considered compression technology, taking into account the specifics of table data processing, including compression ratio, processing speed, and the degree of information preservation. The results of this study can contribute to the practice of exchanging, storing, processing, and analyzing table data in terms of the RDI Platform. The author paid particular attention to analyzing the advantages and disadvantages of each compression method, which allowed forming recommendations for choosing the most suitable technologies that meet strict requirements for performance, compression efficiency, and data preservation reliability. An important aspect of the study is focusing on the possibilities of optimizing processes within the RDI Platform, which increases its efficiency as a tool in the field of working with scientific data. Moreover, this work helps

deepen the understanding of the potential application of modern data compression methods in terms of scientific data exchange practice; it opens prospects for further research and development in this area.
Keywords: compression methods, compression algorithms, Deflate, LZMA, Bzip2, Brotli, Snappy, comparative analysis

References

1. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R. et al. (2011) *Big Data: The Next Frontier for Innovation, Competition, and Productivity*. McKinsey & Company, 146 p., available at: https://personal.utdallas.edu/~muratk/courses/cloud11f_files/MGI-full-report.pdf (accessed January 10, 2024).
2. Choudhary, S.M., Patel, A.S., Parmar, S.J. (2015) 'Study of LZ77 and LZ78 data compression techniques', *IJESIT*, 4(3), pp. 2319–5967.
3. Shannon, C.E. (1948) 'A mathematical theory of communication', *The Bell System Tech. J*., 27(3), pp. 379–423. doi: 10.1002/j.1538-7305.1948.tb01338.x.
4. Oswal, S., Singh, A., Kumari, K. (2016) 'Deflate compression algorithm', *IJERGS*, 4(1), pp. 430–436.
5. Adjeroh, D., Bell, T., Mukherjee, A. (2008) *The Burrows-Wheeler Transform: Data Compression, Suffix Arrays, and Pattern Matching*. NY: Springer Publ., 352 p. doi: 10.1007/978-0-387-78909-5.
6. Sarika, S., Srilali, S. (2013) 'Improved run length encoding scheme for efficient compression data rate', *IJERA*, 3(6), pp. 2248–9622.
7. Gupta, A., Bansal, A., Khanduja, V. (2017) 'Modern lossless compression techniques: Review, comparison and analysis', *Proc. ICECCT*, pp. 1–8. doi: 10.1109/ICECCT.2017.8117850.
8. Alakuijala, J., Farruggia, A., Ferragina, P. et al. (2019) 'Brotli: A general-purpose data compressor', *ACM Transactions on Inform. Sys*., 37, pp. 1–30. doi: 10.1145/3231935.
9. Chen, R., Xu, L. (2023) 'SnappyR: A new high-speed lossless data compression algorithm', *Proc. DCC*, pp. 334–334. doi: 10.1109/DCC55655.2023.00052.

Авторы
Гарев Кирилл Валерьевич [1, 2],
начальник управления, научный сотрудник,
garev.kv@gmail.com, kv@garev.ru

[1] ФБГУ «Российский центр научной информации»,
г. Москва, 119334, Россия
[2] Межведомственный суперкомпьютерный
центр РАН, г. Москва, 119334, Россия

Authors
Kirill V. Garev [1, 2], Head of the Department,
Research Associate,
garev.kv@gmail.com, kv@garev.ru

[1] Russian Center for Science Information,
Moscow, 119334, Russian Federation
[2] Joint Supercomputer Center of RAS,
Moscow, 119334, Russian Federation