УДК 004.023
https://doi.org/10.38086/2522-9613-2019-1-76-80

Алгоритм оптимизации выбора сжимаемых данных

Е. В. Моженкова, магистр технических наук, старший преподаватель кафедры микропроцессорных систем и сетей ИИТ

E-mail: Elena.Mozhenkova@gmail.com

УО «Белорусский государственный университет информатики и радиоэлектроники», ул. Козлова, д. 28, 220037, г. Минск, Республика Беларусь

Т. О. Титовец, студент кафедры кафедры информационных систем и технологий ИИТ

E-mail: nefelim4ag@gmail.com

УО «Белорусский государственный университет информатики и радиоэлектроники», ул. Козлова, д. 28, 220037, г. Минск, Республика Беларусь

А. И. Парамонов, , к. т. н., доцент, доцент кафедры программного обеспечения информационных технологий

E-mail: anton_paramonov@tut.by

УО «Белорусский государственный университет информатики и радиоэлектроники», ул. П. Бровки, д. 6, 220013, г. Минск, Республика Беларусь

Аннотация. В статье рассмотрена проблема эффективности работы алгоритмов сжатия данных. Выполнен анализ эффективности скоростных характеристик сжатия данных файловой системы Btrfs ядра Linux. Предложен алгоритм оптимизации выбора данных для сжатия в режиме реального времени. Описанный алгоритм с допустимой точностью дает ответ на вопрос— необходимо ли сжимать данные. Это позволяет значительно сократить использование вычислительных мощностей. Определены основные характеристики производительности описанного алгоритма.

Ключевые слова: технологии хранения данных; сжатие данных; Btrfs; Linux

Для цитирования: Моженкова, Е. В. Алгоритм оптимизации выбора сжимаемых данных / Е. В. Моженкова, Т. О. Титовец, А.И. Парамонов // Цифровая трансформация. – 2019. – № 1 (6). – С. 76–80. https://doi.org/10.38086/2522-9613-2019-1-76-80

Optimization Algorithm for Selecting Compressible Data

E. V. Mozhenkova, Master of Technical Sciences, Senior Lecturer of Microprocessor Systems and Networks Department of IIT

E-mail: Elena.Mozhenkova@gmail.com

Belarusian State University of Informatics and Radioelectronics, 28 Kozlova Str., 220037 Minsk, Republic of Belarus

T. O. Titovets, Student of Information Systems and Technologies Department of IIT

E-mail: nefelim4ag@gmail.com

Belarusian State University of Informatics and Radioelectronics, 28 Kozlova Str., 220037 Minsk, Republic of Belarus

A. I. Paramonov, Candidate of Sciences (Technology), Associate Professor of Information Technology Software Department

E-mail: anton_paramonov@tut.by

Belarusian State University of Informatics and Radioelectronics, 6 P. Brovka Str., 220013 Minsk, Republic of Belarus

Abstract. The article deals with the problem of the efficiency of data compression algorithms. The analysis of the effectiveness of the speed characteristics of data compression of the Btrfs file system of the Linux kernel has been performed. An algorithm for optimizing the choice of data for real-time compression is proposed. The described algorithm gives the answer with an acceptable accuracy whether it is necessary to compress the data. This can significantly reduce the use of computing power. The main performance characteristics of the described algorithm are determined.
Key words: data storage technologies; data compression; Btrfs; Linux
For citation: Mozhenkova E. V., Titovets T. O., Paramonov A. I. Optimization Algorithm for Selecting Compressible Data Operation of Building. *Cifrovaja transformacija* [Digital transformation], 2019, 1 (6), pp. 76–80 (in Russian). https://doi.org/10.38086/2522-9613-2019-1-76-80

Введение. Объемы цифровых данных, которые необходимо хранить, ежедневно растут с большой скоростью. Существующие технологии хранения данных уже в полной мере не удовлетворяют запросам потребителей и нуждаются не только в модернизации, но и в разработке принципиально новых подходов для обеспечения большей эффективности. Сегодня их разработка является одной из актуальных задач ИТ.

Системы хранения данных от компаний IBM, HP, NetApp решают эту проблему через механизм сжатия данных в процессе записи. Такую возможность предоставляют и некоторые современные файловые системы (NTFS, Btrfs, ZFS), которые поддерживаются в различных операционных системах [1, 2].

При оценке эффективности работы сжатия, записи и чтения данных учитываются характеристики центрального процессора и время доступа к диску. Поэтому для обеспечения заданного уровня качества обслуживания QoS (скорость/время доступа), степеней сжатия и других характеристик приходится применять различные алгоритмы хранения и сжатия данных в файловой системе и системах хранения данных.

В работе выполняется анализ эффективности скоростных характеристик процесса сжатия данных и предлагаются пути решения по оптимизации работы алгоритмов сжатия в режиме реального времени.

Основная часть. Рассмотрим возможности оптимизации работы файловой системы Btrfs ядра Linux в части сжатия данных. Файловая система Btrfs, аналогично другим, может сжимать данные в буфере во время записи информации на диск либо пережимать их после записи. В последнем случае появляется вероятность возникновения ошибки вида false negative (не сжимать сжимаемое) [3]. Степень и алгоритм сжатия файла зависит от структуры данных: конфигурационные файлы — легко сжимаемы, в отличие от медиа данных, которые не поддаются дальнейшему сжатию lossless алгоритмами без потерь [4]. Поэтому для дальнейшего исследования предлагается разделить данные на две категории: «легко сжимаемые» (например, нули) и «трудно сжимаемые» (например, шифрованные или уже сжатые файлы).

Любые данные перед сжатием разбиваются на блоки 8–128 килобайт, чтобы при обращении к байту в пределах сжатой части данных, распаковывать не весь файл, а только соответствующий контейнер. Такой подход ограничивает объемы данных, которые подаются на вход алгоритма сжатия для обработки. Несжимаемые данные, будут записаны на диск «как есть», а сжимаемые — в сжатом контейнере.

Файловая система Btrfs использует следующие библиотеки сжатия: ZLIB, LZOX1, ZSTD [5]. Каждая из этих библиотек применяет различные алгоритмы сжатия и требует разное количество времени на выполнение. Учитывая тот факт, что сжатие происходит непосредственно при записи, время, которое потратит алгоритм, играет существенную роль, так как увеличивается время отклика операционной системы при записи данных на диск. Алгоритмы сжатия не гарантируют определенную скорость сжатия, и как следствие — нагрузка на центральный процессор изменяется в зависимости от характера входных данных. Эти алгоритмы меняют свое поведение в зависимости от ключа, определяющего «насколько сильно стараться сжать данные». В работе предлагается абстрагироваться от конкретного алгоритма сжатия и анализировать структуру сжимаемых файлов с целью оптимизации запуска алгоритма сжатия путем определения возможности сжатия данных.

Эффективность скоростных характеристик сжатия данных анализируется на примере двух файлов размером 1 мегабайт: один файл заполнен нулями, другой — случайными числами. Характеристики исследуемых файлов представлены в таблице 1. Сжатие данных вы-

Таблица 1. Характеристики исследуемых файлов
Table 1. Characteristics of the files under investigation

Формат контейнера архива	Уровень сжатия	Содержимое файла	Время сжатия (мс)
Xz	-9	Zeroes	~50
Xz	-9	Random	~700

полняется в формате контейнера для архивов Xz, использующего алгоритм сжатия LZMA2. Уровень сжатия позволяет снизить размер данных, но время архивации может значительно увеличиваться.

На основе анализа времени сжатия файлов, можно сделать вывод, что попытки сжатия заведомо несжимаемых данных (категория «трудно сжимаемых») требует в 14 (~700/50) раз больше вычислительных ресурсов компьютера. Это позволило сформулировать задачу оптимизации алгоритма сжатия: по исходным данным определить, стоит ли их сжимать, еще до начала запуска процесса сжатия. При этом, для достижения эффекта необходимо, чтобы механизм определения возможности сжатия данных работал значительно быстрее, чем сам алгоритм сжатия.

Для файловой системы Btrfs предложен и реализован алгоритм быстрого определения сжимаемости данных, который включает следующие шаги:

1. Сэмплирование (выборка) входного потока данных. Метод удобен для профилирования на наборах данных и не требует затрат времени. Выбираются данные блоками размером 16 байт с интервалом 256 байт.

2. Проверка на повторяющийся паттерн для гарантированного сжатия.

3. Подсчет количества типов символов, который позволит заранее определить структуру данных — относится она к разряду легко сжимаемых или требующих дополнительного анализа.

4. Подсчет количества символов составляющих основную массу (90 %) объема входных данных — определение распределения типов байтов по гистограмме. Если распределение не является близким к непрерывному равномерному распределению, а ближе к нормальному, то данные с большой вероятностью сжимаемы.

5. В случае, если данные сложно отнести к обоим типам распределения, требуется дополнительный анализ — подсчет уровня энтропии по Шеннону (расчет количества бит, необходимый для кодирования входной последовательности).

Предложенный алгоритм многоуровневый. Основное его назначение — выдать ответ с допустимой точностью и как можно быстрее, необходимо ли сжимать данные.

Рассмотрим применение алгоритма на примере входного массива байт размером до 128 килобайт. Обрабатывать их целиком трудозатратно, поэтому применяем выборку методом Systematic sampling (систематическая выборка), т. к. он предсказуем, удобен для профилирования на наборах данных и не требует дополнительных вычислений (затрат времени) при реализации [6]. Выбираем блоки данных с интервалом 256 байт, размером 16 байт (131 072 / 256 = 512 блоков, 512 × 16 = 8192 байт выборки).

Объем выборки для работы алгоритма по входным данным определяется следующим образом. Желательно, чтобы каждый элемент поля встречался хотя бы 5 раз. В случае с несжимаемыми данными на входе — типы байтов будут равномерно распределены. Определяется размер сэмпла: 256 × 5 — 1280 байт.

Для работы других алгоритмов, в частности энтропии Шеннона, используется массив счетчиков. В качестве базового блока используется байт — это 2^8 = 256 вариантов набора бит. Этот массив будет отображением конечного поля Галуа с порядком 256.

Если взять выборку данных, то даже при минимальных вычислительных ресурсах можно быстро проверить состоит ли счетчик из одинаковых байтов (делением на 2) и затем проверкой совпадения частей (функция *memcmp*()). Если части совпадут, то производится расчет, сколько раз встречается каждый байт в выборке (это дешевая операция, т. к. байт можно использовать как адрес в массиве счетчиков, что приводит к линейной сложности поиска нужного счетчика O (1)).

При наличии конечного поля Галуа, которое заполнено уникальными элементами,

и при известном числе вхождений каждого элемента входные данные анализируются последовательно таким образом (в порядке увеличения вычислительной сложности применяемых алгоритмов):

1. Расчет количества элементов из 256 возможных, которые содержатся в массиве (т. е. проверить, сколько счетчиков имеет не нулевое значение). Данные можно отнести к разряду легко сжимаемых или требующих дополнительного анализа, если знать это значение.

Например, если количество уникальных символов лежит в интервале 0–64, то с очень большой вероятностью это текст (или аналогичные по сложности сжатия данные). Представим, что данные будут сжаты с использованием кода Хаффмана. Поскольку набор входных данных состоит из 64 элементов или меньше, то их можно закодировать с использованием $\log_2(64) = 6$ бит, что с высокой уверенностью дает возможность сжать данные минимум на 25 % ($100 - \log_2(64) \times 100 / \log_2(256)$). В случае если количество элементов лежит не в этом диапазоне нельзя вернуть ответ «Данные сжимаемы». Как следствие — нужен дополнительный анализ.

2. Массив счетчиков представляется в виде гистограммы, и на базе этого проводится дополнительный анализ — поиск пиков и определение насколько эти пики выражены.

В качестве алгоритма реализации выполняются следующее действия:

a) счетчики сортируются по убыванию, чтобы свести к минимуму сложность анализа;

b) расчет количества счетчиков, сумма элементов которых приведет к преодолению порога в 90 % от общего количества байтов в выборке. В зависимости от «выраженности» пиков (т. е. от разброса их значений) будет изменяться количество элементов, которое требуется для преодоления порога.

Эмпирическим путем была выявлена закономерность соответствия частоты повторения символа в интервалах сжимаемости данных:

– 0–64 — данные будут легко сжиматься;

– 200–256 — гистограмма демонстрирует распределение близкое к равномерному. Так как данные имеют равномерное распределение, они с большой вероятностью не сжимаемы.

При использовании кода Хаффмана для сжатия, ярко выраженные пики могут быть закодированы малым количеством байт, а остальная часть данных оставлена как есть. Алгоритм может сообщить: «Попробуй сжать данные», «Данные не сжимаемы».

3. Использование метода оценки энтропии по Шеннону. На данном этапе уже известно число элементов в выборке (всего N байт), и число повторений каждого элемента (байт с символом n). Следовательно, необходимо посчитать среднюю энтропию выборки:

$$p_i = n/N, \quad (1)$$

$$S(x) = \sum_{i=0}^{255}(p_i \times \log_2(p_i)) = p_0 \times \log_2(p_0) + p_1 \times$$

$$\times \log_2(p_1) + \ldots + p_{255} \times \log(p_{255}) \quad (2)$$

Получено значение энтропии в вещественном диапазоне 0–8. Эмпирическим путем установлено, что данные с энтропией, которая лежит в диапазоне 0–5.5, будут легко сжимаемы. Данные в интервале 5.5–6.5 можно попробовать сжать, для интервала 6.5–8 данные не сжимаемы.

В результате работы описанных алгоритмов на выходе возвращается значение, которое выражает, на каком из этапов и по какому критерию было решено, что данные стоит сжимать, а в случае с несжимаемыми данными просто сообщает об этом.

Для предложенного подхода можно сформулировать следующие характеристики производительности:

– требует памяти ~ 256 × 4 (4 байта под счетчик) 1 КБ + 8 КБ (выборка) = 9 КБ (вместе с кодом и переменными ~11 КБ, с более быстрой сортировкой ~12 КБ);

– потоковая производительность (от плохо сжимаемых до хорошо сжимаемых):

a) для сортировок inplace (heap sort): 3500–11 000 МБ/с;

b) для Radix сортировки (требует +1 КБ памяти): 6000–11 000 МБ/с.

Заключение. В результате прикладного исследования была разработана методика определения сжимаемости данных за счет дополнительного анализа структуры данных. Это позволяет учитывать содержание файлов и на его основе принимать решение о необходимости использования алгоритма сжатия. Алгоритм определения сжимаемости данных закодирован для файловой системы Btrfs ядра Linux и позволяет оптимизировать эффективность работы операционной системы при выполнении операций записи данных. Список измененных файлов относительно предыдущей версии обновления и описание внесенных доработок зафиксирован в репозитории Linux Git [7]. Алгоритм прошел этап тестирования и выпущен в ядре Linux версии 4.15 [8].

Список литературы

1. Трубачева, С. И. Особенности построения файловых систем / С. И. Трубачева // Вестник Волжского университета им. В. Н. Татищева. – 2013.– № 4 (22). – С 10–22.
2. Conway A. File Systems Fated for Senescence? Nonsense, Says Science! / 15th USENIX Conference on File and Storage Technologies: February 27 – March 2, 2017. – Santa Clara, CA, USA [Electronic resource]. – Mode of access: https://www.usenix.org/system/files/conference/fast17/fast17-conway.pdf. – Date of access: 22.02.2019.
3. Nisbet R., Miner G., Yale K. J. D. Model Evaluation and Enhancement. Handbook of Statistical Analysis and Data Mining Applications. – 2018. – PP. 215–233.
4. Ключеня, В. В. Процессор ДКП для систем компрессии мультимедиа данных без потерь и с потерями / В. В. Ключеня // Информационные технологии и системы 2013 (ИТС 2013): материалы международной научной конференции, БГУИР, Минск, Беларусь, 23 октября 2013 г. / редкол.: Л. Ю. Шилин [и др.]. – Минск: БГУИР, 2013. – С. 186–187.
5. Mohan J., Kadekod R., Chidambaram V. Analyzing IO Amplification in Linux File Systems / Department of Computer Science, University of Texas at Austin [Electronic resource]. – 2017. – Mode of access: https://arxiv.org/pdf/1707.08514.pdf. – Date of access: 22.02.2019.
6. Sayed A. Mostafa, Ibrahim A. Ahmad. Recent developments in systematic sampling // Journal of Statistical Theory and Practice. – 2017. – PP. 290–310.
7. Kernel.org git repositories [Electronic resource] // Btrfs kernel development. – Mode of access: https://git.kernel.org/pub/scm/linux/kernel/git/kdave/linux.git/log/?h=for-next&qt=author&q=nefelim4ag. – Date of access: 22.02.2019.
8. Linux 4.15 [Electronic resource] // Linus Torvalds. – Mode of access: https://lkml.org/lkml/2018/1/28/173. – Date of access: 22.02.2019.

References

1. Trubacheva S. I. Osobennosti postroeniya fajlovy'x system. Vestnik Volzhskogo universiteta im. V. N. Tatishheva [Bulletin of the Volzhsky University named after V. N. Tatishchev]. Tol'yatti, 2013, pp 10–22.
2. Conway A., Bakshi A. File Systems Fated for Senescence? Nonsense, Says Science! 15th USENIX Conference on File and Storage Technologies (February 27–March 2), 2017, Santa Clara, CA, USA. Available at: https://www.usenix.org/system/files/conference/fast17/fast17-conway.pdf (accessed: 22.02.2019).
3. Nisbet R., Miner G., Yale K. J. D. Model Evaluation and Enhancement. Handbook of Statistical Analysis and Data Mining Applications, 2018, pp. 215–233.
4. Klyuchenya V. V. Processor DKP dlya sistem kompressii mul'timedia danny'x bez poter' i s poteryami. Informacionny'e texnologii i sistemy' 2013 (ITS 2013): materialy mezhdunarodnoj nauchnoj konferencii, BGUIR, Minsk, Belarus', 23 oktyabrya 2013 g. [Information Technologies and Systems 2013 (ITS 2013): Materials of the International Scientific Conference, BSUIR, Minsk, 24th October 2013], 2013, pp. 186–187.
5. Mohan J., Kadekodi R., Chidambaram V. Analyzing IO Amplification in Linux File Systems. University of Texas at Austin. 2017. Available at: https://arxiv.org/pdf/1707.08514.pdf (accessed: 22.02.2019).
6. Sayed A. Mostafa, Ibrahim A. Ahmad. Recent developments in systematic sampling. Journal of Statistical Theory and Practice, 2017, pp. 290–310.
7. Kernel.org git repositories. Available at: https://git.kernel.org/pub/scm/linux/kernel/git/kdave/linux.git/log/?h=for-next&qt=author&q=nefelim4ag (accessed: 22.02.2019).
8. Linux 4.15. Available at: https://lkml.org/lkml/2018/1/28/173 (accessed: 22.02.2019).

Received: 27.02.2019 *Поступила: 27.02.2019*