Исследование возможности распараллеливания алгоритма Лемпеля-Зива-Уэлча (LZW) в импортозамещающих телекоммуникационных мультимедийных технологиях

Д.Ю. Васин [1]

[1] *Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского" (ННГУ), пр. Гагарина, 23, г. Нижний Новгород, 603022, Россия*

Аннотация

В статье рассмотрены результаты исследований автора в области последовательного/параллельного сжатия одномерных, адаптивных звуковых сигналов. Содержится обзор многих известных алгоритмов сжатия в том числе и базовый алгоритм LZW и его компоненты, а также представлен подробный анализ двух схем распараллеливания: схема последовательной сроки, разделенная на несколько частей для сжатия, и распараллеливание с использованием префиксного дерева с помощью расширения внутренних циклов. Исследована возможность распараллеливания алгоритма LZW для сжатия и оценки его производительности. Описана оценка ускорения по закону Амдала и оценка масштабирования за счет увеличения количества потоков и процессов. Представлены результаты экспериментов, сравнивающих последовательный и параллельный алгоритмы LZW с точки зрения степени сжатия, качества и вычислительной сложности.

Ключевые слова

Последовательное/параллельное сжатие одномерных звуковых сигналов; алгоритм Лемпеля-Зива-Уэлча (LZW); параллельные алгоритмы; мультимедийные технологии; импортозамещение в области телекоммуникационных технологий.

Study of the Possibility of Parallelization of the Lempel-Ziv-Welch (LZW) Algorithm in Import-Substituting Telecommunication Multimedia Technologies

D. Yu. Vasin [1]

[1] *National Research Lobachevsky State University of Nizhny Novgorod, Prospekt Gagarina, 23, Nizhny Novgorod, 603950, Russia*

Abstract

The article discusses the results of the author's research in the field of serial/parallel compression of one-dimensional, adaptive audio signals. An overview of many well-known compression algorithms is provided, including the basic LZW algorithm and its components, and a detailed analysis of two parallelization schemes is presented: a serial time scheme divided into several parts for compression, and parallelization using a prefix tree using expansion inner loops. The possibility of parallelization of the LZW algorithm for compression and evaluation of its performance is investigated. An estimation of acceleration according to Amdahl's law and an estimation of scaling due to an increase in the number of threads and processes are described. The results of experiments comparing serial and parallel LZW algorithms in terms of compression ratio, quality, and computational complexity are presented.

ГрафиКон 2023: 33-я Международная конференция по компьютерной графике и машинному зрению, 19-21 сентября 2023 г., Институт проблем управления им. В.А. Трапезникова Российской академии наук, г. Москва, Россия
EMAIL: dm04@list.ru (Д.Ю. Васин)
ORCID: 0000-0002–4341–2457 (Д. Ю. Васин)

Keywords
Serial/parallel compression of one-dimensional audio signals; Lempel-Ziv-Welch (LZW) algorithm; parallel algorithms; multimedia technologies; import substitution in the field of telecommunication technologies.

1. Введение

В последние годы наблюдается значительное сближение технологий сети Интернет с различными цифровыми технологиями обработки и передачи аудио и видео данных, что открывает новые возможности на активно развивающемся рынке мультимедиа. Это определяет интерес к проведению исследований, которые направлены на выявление недостатков имеющихся методов обработки цифровых данных и при необходимости определения путей их совершенствования для удовлетворения современных технологических нужд.

Существующие возможности цифровой обработки речевой и звуковой информации внедряются в различные виды приложений и существенно расширяют их спектр: наряду с широко распространенными системами телефонии, традиционного телерадиовещания и различных разновидностей систем специального назначения, появились и повсеместно внедряются самые различные приложения – видеотелефония и конференцсвязь, интернет-вещание и мобильные мультимедийные приложения, телевидение высокой четкости и цифровое кино. При этом, цифровые системы кодирования речи и звука, их передачи и приема позволяют достичь высочайшего уровня качества и предоставляют пользователю массу новых возможностей и новых видов услуг [1].

Конкретные виды аппаратной и программной реализации систем цифровой обработки речевой и звуковой информации определяются их исходными характеристиками, особенностями слухового восприятия и требованиями к качеству воспроизведения.

Однако, рост числа и совершенствование различных IT-технологий неизбежно приводит к значительному увеличению объемов данных, обрабатываемых и передаваемых по каналам связи. В течение последних десятилетий человечество создало столько же данных, сколько за всю предыдущую историю цивилизации. При этом темпы роста объема хранимых данных продолжают увеличиваться: по разным оценкам их удвоение происходит каждые 4 или 10 лет. В данном контексте представляются важными вопросы сжатия цифровых данных вообще и звуковой информации, в частности, которые в настоящее время продолжают оставаться весьма актуальными [2–13].

Компрессия аудиоданных является важнейшей составляющей различных телекоммуникационных мультимедийных систем. Особо выделим системы двойного назначения, функционирующие на основе современных протоколов передачи, гарантирующих целостность передаваемых данных.

Базовой характеристикой различных телекоммуникационных мультимедийных систем является быстродействие используемых алгоритмов компрессии, поскольку такие системы функционируют в реальном времени. Данная характеристика в значительной степени определяет число одновременно обслуживаемых пользователей, время работы мобильного устройства от батареи, стоимость специализированных вычислительных средств обработки аудио, возможность кодирования аудио в фоновом режиме при использовании ресурсов вычислителя для других задач.

Поскольку существующие алгоритмы аудио компрессии, как правило, обладают либо сравнительно малым коэффициентом сжатия при низкой трудоёмкости, либо характеризуются высоким коэффициентом сжатия при высокой трудоемкости, то при выборе конкретного алгоритма производится поиск компромисса между желаемым коэффициентом сжатия и трудоёмкостью алгоритма.

Особенно строгие ограничения на трудоёмкость накладываются в многоканальных системах, а также в системах, где необходимо параллельно осуществлять интенсивную обработку другой информации.

Рост сложных задач, решение которых связано с применением современных IT-технологий, ведет к необходимости использования параллельных вычислений, которые носят междисциплинарный характер. Это определяет возможности применения параллельных

методов в различных областях научно-практической деятельности, поскольку такие методы затрагивают такие области, как численные методы, структуры и алгоритмы обработки данных, аппаратное и программное обеспечение, системный анализ.

Целью любого параллелизма является повышение эффективности работы компьютера на различных уровнях: задач, данных, алгоритмов, инструкций, битов [14].

В целом, видеокодирование и/или кодирование неподвижных изображений вкупе с кодированием звука, позволяет сделать сеть передачи данных по-настоящему универсальной и использовать сетевой ресурс максимально полно. Под универсальностью понимаем увеличение видов источников информации, на базе которых могут быть построены новые сетевые мультимедийные услуги, а эффективное использование сетевого ресурса напрямую влияет на стоимость услуги и на гибкость внедрения такой услуги в существующей сетевой архитектуре.

Конечная цель данного исследования заключалась в исследовании возможности распараллеливания алгоритма LZW для сжатия и оценки его производительности и разработке программной реализации последовательного/параллельного сжатия одномерных, адаптивных звуковых сигналов. В результате планировалось получить информацию о производительности параллельных алгоритмов LZW и их потенциальных приложениях для сжатия аудиосигналов.

2. Цель сжатия, типы и классификация существующих методов и систем сжатия. Критерии оценки методов сжатия

Компрессия (сжатие) данных (data compression) – это процесс сокращения объема данных без потерь или с минимальной потерей информации. Путем применения алгоритмов и методов компрессии, существенно уменьшается размер данных, сохраняя при этом их суть и основные характеристики. Это может быть полезно во множестве ситуаций, начиная от экономии места на хранилище и ускорения передачи данных до минимизации затрат на интернет-трафик и повышения производительности систем обработки и анализа информации. Обратная процедура – *восстановление данных (распаковка, декомпрессия)*. Сжатие основано на устранении избыточности, содержащейся в исходных данных.

Главная цель алгоритмов компрессии данных – уменьшить размер данных, сохраняя при этом их информационную значимость. Компрессия данных активно применяется во множестве областей, включая сжатие аудио и видеофайлов, сжатие изображений, архивирование файлов и многое другое. В аудио и видео сжатие позволяет свести к минимуму размер файлов, необходимых для их хранения и передачи онлайн. Это особенно актуально для различных телекоммуникационных систем, где важно обеспечить высокое качество воспроизведения при минимальных затратах на трафик.

В основе разработки эффективных методов компрессии данных лежит модель избыточности (источника данных), т.е. для их эффективной компрессии необходимы предваряющие сведения о природе компрессируемых данных. При отсутствии указанной метаинформации об источнике невозможно разрабатывать преобразование, позволяющее уменьшить информационную избыточность сообщения. Различают статическую (неизменную) для всего сжимаемого сообщения модель избыточности и модель, формируемую (параметризуемую) на этапе сжатия (восстановления). Все существующие методы компрессии разделим на *адаптивные*, позволяющие на основе входных данных изменять модель избыточности информации и *неадаптивными,* применяемыми для работы с данными, обладающими хорошо определёнными и неизменными характеристиками. Как правило неадаптивными являются обычно узкоспециализированные алгоритмы.

Абсолютное большинство достаточно универсальных алгоритмов являются в той или иной мере адаптивными.

Простейшим примером избыточности является повторение в тексте отдельных слов (фрагментов). Повторяющийся фрагмент единожды кодируется, а в дальнейшем он всякий раз заменяется ссылкой на уже закодированный фрагмент с указанием его длины. Другой вид избыточности связан с тем, что некоторые символы в исходных данных встречаются чаще других. Устраняется этот вид избыточности применением энтропийного кодирование, смысл которого заключается в использовании кодов переменной длины: часто встречающиеся данные

заменяются более короткими кодовыми словами, а редко встречающиеся – более длинными. Сжатие данных, не обладающих свойством избыточности (белый шум), принципиально невозможно без потерь.

Сжатие звука является фундаментальной технологией в современной цифровой обработке аудиоданных. Оно позволяет значительно сократить объем звуковых файлов и потокового аудио, что существенно облегчает их передачу, хранение и обработку. Различные алгоритмы сжатия были разработаны для достижения оптимальной компромиссной точки между степенью сжатия и сохранением качества звука.

При компрессии звука различают статистическую и психоакустическую избыточность. Сокращение статистической избыточности основано на учете свойств самих звуковых сигналов, а психоакустической – на учете особенностей слухового восприятия звука. Устранение статистической избыточности звукового сигнала даже при достаточно сложных процедурах обработки позволяет в конечном итоге уменьшить изначально требуемую пропускную способность канала связи лишь на 15 … 40 % [2]. Такое сжатие не может обеспечить все возрастающие требования, причем не столько к количеству, как к качеству передаваемой информации.

Поэтому в последнее время в нашей стране и за рубежом уделяется большое внимание [1; 2] вопросам развития новых методов сжатия звука, основанных на использовании психоакустических свойств слуха человека. При этом сжатие производится не во временной, а в частотной области, для чего используются дискретные ортогональные преобразования (ДОП).

Кодирование с использованием психоакустического восприятия подразумевает способ преобразования потока данных, при котором кодированию подвергается только та часть звуковой информации, которую способно воспринять ухо человека, остальные же составляющие исходного сигнала можно отбросить, за счет чего и достигается эффект сжатия.

Основными требованиями к методам кодирования звуковой информации является разборчивость воспроизводимого сигнала и гарантированное достаточно хорошее качество звучания музыкальных и голосовых сообщений, существенно отличных друг от друга.

Методы сжатия разделим на два типа:

неискажающие (lossless) методы сжатия (сжатие без потерь, архивация данных) гарантируют, что декодированные данные будут в точности совпадать с исходными, т.к такие методы не уменьшают количество информации в сообщении, несмотря на уменьшение длины. Обычно используются для передачи и хранения различных данных, в которых искажения недопустимы или нежелательны (различная текстовая информация, документы, чертежи и т.п.).

Сжатие звука без потерь основано на использовании различных методов кодирования, включая алгоритмы словарного и словарно-кодового сжатия. Эти алгоритмы позволяют находить повторяющиеся последовательности и заменять их более компактными кодами. Они обеспечивают полное восстановление исходного звука, не теряя никакой информации.

Существует предел сжатия без потерь, зависящий от энтропии источника. Чем более предсказуемы получаемые данные, тем лучше их можно сжать. Случайная независимая равновероятная последовательность сжатию без потерь не поддаётся.

По теореме Шеннона об источнике шифрования (или теорема бесшумного шифрования) оптимальная длинна кода для символа равна log_bP, где b число символов, использованных при генерации выходного кода (b=2 для двоичного кода), P — вероятность появления входного символа, т.е., если вероятность появления элемента s_i равна p(s_i), то наиболее выгодно будет представить этот элемент log_2p(s_i) битами;

искажающие (lossy) методы сжатия (сжатие с потерями) могут искажать исходные данные (человеческая речь, музыкальные данные, изображения), например за счет удаления несущественной с точки зрения дальнейшего использования восстановленной части данных, после чего полное восстановление невозможно. Такая возможность возникает, только если распределение вероятностей на множестве сообщений не равномерное, например часть теоретически возможных в прежней кодировке сообщений на практике не встречается. Обладают значительно большей, чем сжатие без потерь, эффективностью, обычно применяются для сокращения объёма исходных данных, когда сокращение объема является более важным аспектом по сравнению с полным соответствием исходных и восстановленных данных.

Искажающие методы характеризуются двумя параметрами: скоростью сжатия *R[бит/отсчет]* и величиной искажений *D*, определяемых как $R = \frac{k}{n}$; $D = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - x_i^*\right)^2$, где величина *D* является мерой среднеквадратического различия между X* и X.

Пусть располагаем двумя системами разрушающего сжатия: первая – со скоростью и искажениями R_1 и D_1 соответственно и вторая – со скоростью R_2 и искажениями D_2. Установлено, что первая из них лучше, если $(R_1 < R_2) \wedge (D_1 < D_2)$. Т.к параметры R_i и D_i связаны обратной зависимостью, невозможно построить систему разрушающего сжатия, обеспечивающую одновременно снижение скорости *R* и уменьшение искажений *D*.

Из сказанного следует, что целью оптимизации системы сжатия с потерями может быть либо минимизация скорости при заданной величине искажений, либо получение наименьших искажений при заданной скорости сжатия.

Методы этой группы можно подразделить на три вида [15]:

1. кодирование непосредственно реализации аудиосигнала (Wave Form Codec);
2. измерение, кодирование и передача на приемную сторону параметров аудиосигнала, по которым уже на приемной стороне производится синтез этого (искусственного) аудиосигнала. Такие системы называют вокодерными (Source Codec);
3. гибридные способы кодирования, т.е. сочетание первого и второго способов кодирования.

Использование разрушающих методов определяется тремя факторами:

1. непосредственно генерация набора исходных сообщений происходит с некоторым, отличным от нуля, уровнем искажений и ошибок, поскольку сами источники данных, обладают ограниченным динамическим диапазоном, поэтому невозможно достичь абсолютной точности воспроизведения;
2. при передаче данных по каналам связи и их хранении наличие различного рода помех абсолютно неизбежно, следовательно, принятое (восстановленное) сообщение никогда абсолютно не совпадет с переданным. Таким образом, практически ни при каких условиях в реальности нельзя обеспечить абсолютно точную передачу, обусловленную наличием тех или иных помех в передающем тракте (в системе хранения);
3. все сообщения передаются и сохраняются для их восприятия и использования получателем (органами чувств человека, исполнительными механизмами и т.д.), которые тоже обладают конечной разрешающей способностью. А это значит, что получатель не заметит малого различия между абсолютно точным и приближенным значениями воспроизводимого сигнала. Порог чувствительности к искажениям различен у разных получателей, но он всегда есть.

Кодирование с разрушением учитывает эти доводы в пользу приближенного восстановления данных и позволяет получить за счет некоторой контролируемой по величине ошибки рекордные коэффициенты сжатия, на порядки превышающие степень сжатия для неразрушающих методов.

Выбор системы неразрушающего или разрушающего сжатия зависит от типа данных, подлежащих сжатию. Неискажающее сжатие, при всей привлекательности перспективы получить абсолютное совпадение исходных и восстановленных данных, имеет невысокую эффективность – коэффициенты такого сжатия редко превышают 3…5 (за исключением случаев кодирования данных с высокой степенью повторяемости одинаковых участков и т.п.).

В основе большинства методов разрушающего сжатия лежит кодирование не самих данных, а некоторых линейных преобразований от них, например, коэффициентов дискретного преобразования Фурье (ДПФ), или косинусного преобразования, преобразований Хаара, Уолша и т.п. Несмотря на значительное разнообразие алгоритмов компрессии цифровых аудиоданных, структура кодера может быть представлена в виде обобщенной схемы, показанной на рисунке 1:

Система сжатия данных состоит из кодера и декодера источника. Кодер сжимает (компрессирует) данные источника в сжатые данные, а декодер выполняет обратную функцию– восстанавливает данных источника из сжатых данных. При неискажающем сжатии

восстановленные данные абсолютно точно совпадают с исходными, а при искажающем – могут незначительно отличаться от них [16].

При разрушающем кодировании аудиоданных большинство используемых методов основаны на учете психоакустичесих свойств слуховой системы человека. В этой связи наиболее востребованными являются методы и алгоритмы, учитывающие различного рода маскировки, выявление отдельных деталей звучания, которыми можно пренебречь при восстановлении, эффективные алгоритмы передискретизации, переквантования и пр.[1]

Рисунок 1 – Структурная схема системы сжатия данных

Фактически можно выделить три основных этапа кодирования звука: фильтрация, применение психоакустической модели, квантование и кодирование. Реализациям практически каждого из этих этапов в существующих технологиях присущи определенные недостатки [17].

Так, в подавляющем большинстве алгоритмов во время фильтрации отсчеты сигнала делятся на субполосы равной ширины, что упрощает фильтры, но сильно контрастирует с особенностями слухового восприятия, которое зависит от частоты звука [17]. Кроме того, смежные фильтры должны идеально пропускать разные диапазоны частот. На практике они имеют существенное частотное перекрытие. Звук, состоящий из одного чистого тона, может попасть в два фильтра и породить сигналы, которые потом будут квантоваться в две подполосы вместо одной [16].

Основными свойствами какого-либо алгоритма сжатия данных являются:

- качество (коэффициент или степень) сжатия, т. е. отношение длины (в битах) сжатого представления данных к длине исходного представления;
- скорость кодирования и декодирования, определяемые временем, затрачиваемым на кодирование и декодирование данных;
- объем требуемой памяти.

В области сжатия данных существует *закон рычага,* указывающий на то, что алгоритмы, требующие больше ресурсов (времени и памяти), обычно достигают лучшего качества сжатия., а менее ресурсоемкие алгоритмы, как правило, уступают в качестве сжатия более ресурсоемким алгоритмам.

Таким образом, создание оптимального алгоритма сжатия данных с практической точки зрения является сложной задачей, требующей достижения высокого качества сжатия при ограниченном использовании ресурсов. Для этого необходимо обеспечить баланс между эффективностью сжатия и требуемыми ресурсами, для формирования оптимального соотношения между размером сжатых данных и используемыми ресурсами.

3. Идея базового алгоритма Лемпеля-Зива-Велча (LZW)

Алгоритм Лемпеля-Зива-Велча (LZW) является еще одним важным алгоритмом компрессии данных без потерь, был разработан *Абрахамом Лемпелем* и *Яковом Зивом* в 1977 году, а позже его улучшенная версия была предложена Терри Велчем.

Кодирование. LZW основывается на идее построения словаря, состоящего из уже встреченных входных данных строк. Он начинает с инициализации словаря базовыми символами, такими как отдельные символы или короткие строки. Затем LZW сканирует входные данные и сравнивает текущую строку с существующими элементами словаря. Если текущая строка уже присутствует в словаре, LZW переходит к следующему символу и повторяет процесс. Если текущая строка не найдена в словаре, LZW добавляет ее в словарь и заменяет текущую строку ссылкой на предыдущий подходящий элемент словаря. Это позволяет уменьшить объем данных, заменяя повторяющиеся последовательности более короткими ссылками. LZW продолжает сканирование и построение словаря до тех пор, пока не достигнет конца входных данных или до достижения заданного предела словаря. Затем полученные

ссылки и окончательный словарь используются для восстановления исходных данных при декомпрессии.

Этот подход значительно сокращает размер данных, делая их более компактными и эффективными для хранения и передачи.

Алгоритм LZW обладает временной сложностью *O(n)*, где *n* – длина входного сигнала, что делает его одним из самых эффективных алгоритмов сжатия, доступных на сегодняшний день. Низкая вычислительная сложность в сочетании с возможностью достижения высокой степени сжатия делает алгоритм LZW популярным выбором для приложений, связанных со сжатием аудиосигналов

Декодирование. Особенность LZW заключается в том, что для декомпрессии нам не надо сохранять таблицу строк в файл для распаковки. Алгоритм построен таким образом, что мы в состоянии восстановить таблицу строк, пользуясь только потоком кодов.

Теперь представим, что мы получили некоторое закодированное сообщение и нам нужно его декодировать. Для восстановления данных необходимо знать начальный словарь, а последующие записи словаря мы можем реконструировать уже на ходу, поскольку они являются просто конкатенацией предыдущих записей. Кроме того, в процессе кодировании и декодировании коды в словарь добавляются во время обработки одного и того же символа, т.е. это происходит “синхронно”.

При сжатии данных алгоритм LZW эффективно удаляет повторяющиеся участки информации и заменяет их более компактными ссылками. Благодаря этому, объем данных значительно сокращается без потери информации. Однако для корректной декомпрессии необходимо сохранить информацию о словаре или использовать специальные механизмы, позволяющие восстановить исходный словарь при декомпрессии.

Алгоритм LZW широко применяется для сжатия текстовых данных, а также в других областях, где встречается неравномерное распределение символов или повторяющиеся структуры данных. Он обладает высоким уровнем сжатия и высокой скоростью выполнения, что делает его особенно полезным для решения задач сжатия данных без потерь.

Преимущества базового алгоритма LZW:

- основное отличие и преимущество в однопроходности алгоритма;
- большая гибкость в представлении дробных частот встречаемости символов;
- позволяет сжимать данные с энтропией, меньшей 1 бита на кодируемый символ;
- обеспечивает почти оптимальную степень сжатия с точки зрения энтропийной оценки кодирования Шеннона;
- является блочным;
- демонстрирует большую эффективность в отношении данных с неравномерным распределением вероятностей кодируемых символов;
- легко взаимозаменяется и работает совместно с эффективным алгоритмом Хаффмана;
- алгоритма LZW прост в реализации, по сравнению с аналогами;
- не требует вычисления вероятностей встречаемости символов или кодов;
- алгоритм построен таким образом, что мы в состоянии восстановить таблицу строк, пользуясь только потоком кодов;
- имеет преимущество перед статическими методами сжатия;
- извлекает преимущества при повторяющихся цепочках данных.

Недостатки базового алгоритма LZW:

- малая эффективность при кодировании незначительного объёма данных;
- более эффективен в тех случаях, когда исходные данные содержат большое количество повторяющихся подстрок;
- алгоритм не проводит анализ входных данных поэтому не оптимален;
- алгоритм не проводит анализ входных данных.

4. Подготовка данных для получения начального решения

Классический алгоритм LZW предназначен для сжатия текстовых данных и не имеет прямой поддержки аудиофайлов. Однако, в целях сжатия аудиофайлов, можно использовать дополнительные преобразования, чтобы подготовить данные для алгоритма LZW. Кратко представим поэтапный процесс сжатия аудиофайла с использованием классического алгоритма LZW и дополнительных преобразований:

Предварительная обработка аудиофайла заключалась в осуществлении стандартных процедур нормализации громкости и фильтрации внешних шумов.

Преобразование аудиофайла в текстовые данные – для применения классического алгоритма LZW аудиофайл должен быть преобразован в текстовые данные. Одним из методов является преобразование аудио в последовательность чисел, используя, алгоритм *Pulse Code Modulation (PCM)*. PCM представляет аудиофайл в виде аналоговых сигналов, дискретизованных и квантованных в цифровую форму [18].

Импульсно-кодовая модуляция (ИКМ, pulse code modulation, PCM) используется для оцифровки аналоговых сигналов. Практически все виды аналоговых данных (видео, аудио (голос, музыка), телеметрия) допускают применение ИКМ.

При импульсно-кодовой модуляции аналоговый передаваемый сигнал преобразуется в цифровую форму посредством трёх операций: дискретизации по времени, квантования по амплитуде и кодирования.

Формат .PCM представляет аудиоданные в виде последовательности чисел, представляющих значения амплитуды сигнала на протяжении времени.

Для преобразования аудиосигнала формата .MP3 в файл формата .PCM используется «онлайн конвертер ASPOSE», в котором присутствует технология ИКМ [19].

Преобразование .PCM файла в бинарную строку – выполняется с помощью оригинальной утилиты на языке Python, разработанной в ходе реализации проекта.

5. Описание схем распараллеливания LZW

*Две схемы распараллеливания: р*аспараллеливание алгоритма LZW направлено на повышение его производительности за счет разбиения обработки на более мелкие задачи, которые можно выполнять одновременно. Существует две основные схемы распараллеливания для алгоритма LZW: распараллеливание последовательных строк и распараллеливание с использованием префиксного дерева (внутренние циклы).

Строка разделена на две части для сжатия – схема разбивает входную строку на несколько частей и выполняет сжатие каждой части параллельно. Затем сжатые части объединяются для формирования окончательного сжатого вывода. Эта схема относительно проста в реализации и может быть легко распараллелена, что делает ее привлекательным вариантом для аппаратных реализаций.

Распараллеливание внутренних циклов префиксного дерева – схема распараллеливает внутренние циклы алгоритма LZW, используя дерево префиксов для хранения словаря. Префиксное дерево — это тип древовидной структуры данных, в которой каждый узел представляет собой префикс строки. Это делает возможным параллельный поиск по словарю, что значительно сокращает время, необходимое для кодирования входного сигнала.

Обе схемы распараллеливания имеют свои преимущества и недостатки, и выбор схемы будет зависеть от конкретных требований приложения. Последовательное распараллеливание проще в реализации, но может иметь более низкую производительность, чем схема распараллеливания префиксного дерева, в то время как схема распараллеливания префиксного дерева имеет более высокую производительность, но ее сложнее реализовать.

Для оценки производительности параллельного алгоритма LZW используются следующие тестовые задания:

Сжатие бинарной строки. Первая тестовая задача включает сжатие бинарной строки с использованием алгоритма LZW. Бинарная строка может иметь разную длину и сложность, а производительность сжатия измеряется степенью сжатия и временем сжатия. Эксперимент

проводится на бинарной строке разной величины, результаты выполнения сравниваются в таблице. На вход поступает случайно сгенерированная бинарная строка, которая делится на несколько частей, после этого она сжимается с помощью алгоритма LZW. Сжатые части объединяются в одну строку.

Сжатие с помощью префиксного дерева. Во второй тестовой задаче рассматривается сжатие структуры данных, которая представляет ассоциативный массив с использованием алгоритма LZW. Для этого используется префиксное дерево, где каждое ребро помечено символом, а все ребра, исходящие из одного узла, имеют разные символы. Строки в этой структуре данных могут быть разной длины и сложности, и эффективность сжатия оценивается по степени сжатия и времени выполнения.

В процессе сжатия динамически создается словарь фраз, где определенным последовательностям символов присваиваются коды фиксированной длины. Алгоритм просматривает текст по одному символу, начиная слева. При обработке каждого символа алгоритм ищет наибольшую совпадающую с фразой из словаря строку X. Затем код этой фразы выводится, а строка XУ, где У - символ, следующий за X во входном сообщении, добавляется в словарь как новая фраза и присваивается ей код (при этом XУ еще не содержится в словаре). Символ У используется для начала следующей фразы.

Построение префиксного дерева, содержащего фразы из словаря, осуществляется путем чтения максимально длинной строки, начиная с корня дерева. Для добавления новой фразы XУ к найденному узлу необходимо создать нового сына с символом У, а код фразы X может быть представлен индексом узла в массиве, содержащем все узлы. Результаты выполнения сравниваются в таблице. Для достижения эффективности работы алгоритма применяется распараллеливание внутренних циклов.

Эти тестовые задания предназначены для оценки производительности алгоритма LZW в различных условиях и для обеспечения всестороннего понимания производительности и ограничений алгоритма. Выполняя эти тестовые задания, можно сравнить производительность параллельных и последовательных реализаций алгоритма LZW и определить области для улучшения.

6. Методика экспериментов последовательного и параллельного сжатия

Для оценки производительности последовательной и параллельной реализаций алгоритма LZW используются следующие экспериментальные методики:

Последовательная реализация. Последовательная реализация алгоритма LZW выполняется на одном ядре компьютера с использованием одного потока. Двоичная строка или звуковой сигнал сжимаются с использованием последовательной реализации, а производительность сжатия измеряется с точки зрения степени сжатия и времени сжатия.

Параллельная реализация. Параллельная реализация алгоритма LZW выполняется на компьютере с несколькими ядрами и потоками. Двоичная строка или звуковой сигнал сжимаются с использованием параллельной реализации, а производительность сжатия измеряется с точки зрения степени сжатия и времени сжатия. Количество потоков и процессов, используемых в параллельной реализации, можно варьировать для оценки масштабируемости алгоритма.

Эти экспериментальные методы предназначены для обеспечения справедливого сравнения последовательной и параллельной реализации алгоритма LZW. Выполняя обе реализации на одном компьютере с одинаковыми входными данными, можно точно сравнить производительность сжатия и выявить любые различия между двумя реализациями.

7. Анализ распараллеливания

Проанализируем распараллеливание алгоритма LZW с точки зрения ускорения и масштабирования. Проводя анализ распараллеливания, можно понять потенциальные

преимущества производительности, которые могут быть достигнуты за счет распараллеливания алгоритма LZW, и принять обоснованные решения об оптимальном подходе к распараллеливанию для данной задачи.

Оценка ускорения рассчитывается с использованием закона Амдала [18, 19], который предсказывает максимально возможное ускорение, которое может быть достигнуто за счет распараллеливания алгоритма. Согласно этому закону, ускорение выполнения программы за счет распараллеливания ее инструкций на множестве вычислителей ограничено временем, необходимым для выполнения ее последовательных инструкций. Принимая во внимание долю алгоритма, который может быть распараллелен, и количество используемых процессоров, закон Амдала может дать оценку максимально возможного ускорения. Закон Амдала иллюстрирует ограничение роста производительности вычислительной системы с увеличением количества вычислителей.

Оценка масштабирования выполняется путем увеличения количества потоков и процессов, используемых для распараллеливания. Она включает измерение производительности параллельного алгоритма LZW по мере увеличения числа потоков и процессов, используемых для распараллеливания. Цель состоит в том, чтобы определить оптимальное количество потоков и процессов, обеспечивающих наилучшую производительность, с учетом таких факторов, как доступные аппаратные ресурсы и вычислительная сложность алгоритма.

8. Результаты эксперимента

Результаты алгоритма LZW на бинарной строке содержатся в таблицах 1–2 и на рисунке 2.

Таблица 1 – Последовательное сжатие

Время сжатия (сек.)	Величина строки (симв.)	Величина сжатой строки (симв.)	Коэффициент сжатия	Кол-во ядер	Ускорение
1,759	10^6	74.026	13,5	1	–
23,345	10^7	575.182	17,4	1	–
257,214	10^8	4434.907	22,6	1	–

Таблица 2 – Параллельное сжатие

Время сжатия (сек.)	Величина строки (симв.)	Величина сжатой строки (симв.)	Коэффициент сжатия	Кол-во ядер	Ускорение
0,964	10^6	79.519	12,6	2	1.8
11,106	10^7	626,268	16	2	2,1
132,891	10^8	4620,417	21,6	2	1,9
0,515	10^6	85,658	11,7	4	3,4
5,998	10^7	674,533	14,8	4	3,9
78,241	10^8	4992,351	20	4	3,3
0,377	10^6	92,770	10,8	8	4,7
3,944	10^7	723,674	13,8	8	5,9
47,501	10^8	5577,31	17,9	8	5,4

Результаты на префиксном дереве приведены в таблицах 3–4 и на рисунке3.

Таблица 3 – Последовательное сжатие

Время сжатия (сек.)	Величина строки (симв.)	Величина сжатой строки (симв.)	Коэффициент сжатия	Кол-во ядер	Ускорение
49,8	10^8	3487,721	28,6	1	–
666,3	10^9	14304,223	69,9	1	–

Таблица 4 – Параллельное сжатие

Время сжатия (сек.)	Величина строки (симв.)	Величина сжатой строки (симв.)	Коэффициент сжатия	Кол-во ядер	Ускорение
25,7	10^8	3503,619	28,5	2	1.9
334,2	10^9	14635,695	68,3	2	2
14,4	10^8	3577,119	27,9	4	3,5
164,2	10^9	15823,117	63,2	4	4
12,2	10^8	3488,085	28,7	8	4,1
100,2	10^9	14471,249	69,1	8	6,6

Рисунок 2 –Закон Амдала для эксперимента с бинарной строкой

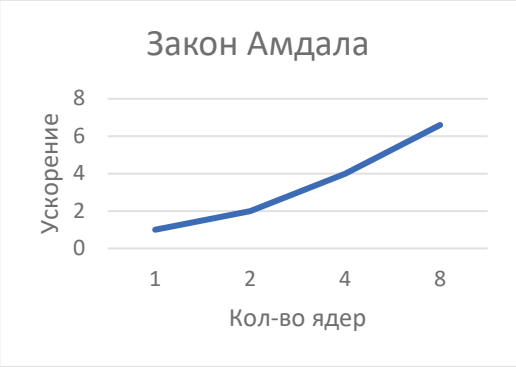

Рисунок 3 – Закон Амдала для эксперимента с префиксным деревом

При технических возможностях в 8 ядер процессора, можно оценить степень масштабируемости алгоритма и ускорение по закону Амдала. Алгоритм достаточно хорошо поддаётся масштабируемости. Можно сделать прогноз, что значительного ускорения не будет только на 4096 ядрах. По закону Амдала, эффективность распараллеливания данного алгоритма около 90%.

По закону Амдала, эффективность распараллеливания данного алгоритма, используя префиксное дерево, немного лучше, чем эффективность предыдущего.

9. Заключение

При анализе результатов эксперимента и их сравнении с теоретическими ожиданиями получилось многочисленное сходство.

Для оценки эффективности работы параллельной реализации алгоритма LZW, был проведен эксперимент сравнения времени выполнения последовательной и параллельной реализаций на разных входных данных: префиксном дереве и бинарной строке.

Результаты показали, что при параллельной реализации алгоритма LZW на префиксном дереве время сжатия в несколько раз лучше, чем в случае бинарной строки. Однако, время сжатия в параллельной реализации оказалось значительно больше, чем в последовательной реализации. Разница в размере сжатых данных между последовательной и параллельной реализациями была незначительной.

Таким образом, параллельная реализация алгоритма LZW на префиксном дереве более эффективна, чем параллельная реализация на бинарной строке. Однако, в общем случае, параллельная реализация алгоритма не всегда даёт выигрыш во времени выполнения, а может даже приводить к ухудшению.

Во всех экспериментах реализации алгоритма LZW расположим в порядке убывания их эффективности:

1) параллельное сжатие (префиксное дерево);
2) параллельное сжатие (бинарная строка);

3) последовательное сжатие (префиксное дерево);
4) последовательное сжатие (бинарная строка).

Алгоритм LZW представляет собой компромисс между скоростью и степенью сжатия, может породить много ответвлений и подобных реализаций в будущем.

10. Список источников

[1] Дворкович В.П., Дворкович А.В. Цифровые видеоинформационные системы (теория и практика). М.: Техносфера, 2012. – 1008 с

[2] Watkinson J. An Introduction to Digital Audio. Second Edition. Waltham, Massachusetts: Focal Press, 2013. 419 p.

[3] Заболотов В. А., Стефанова И. А. Сжатие аудиоданных на основе психоакустических свойств слуха человека. Естественные и математические науки в современном мире // Сб. ст. по матер. XLIII Междунар. науч.-практ. конф. № 6(41). Новосибирск: СибАК, 2016. С. 43–51.

[4] LameXP – Audio Encoder Front-End. URL: http://lamexp.sourceforge.net/ (дата обращения 16.06.2023).

[5] Ulacha G., Stasinski R. Context Lossless Coding of Audio Signals // Compression Conf. Snowbird: UT, 2013. P. 523–523.

[6] Monkey’s Audio Compression. URL: https://www.monkeysaudio.com/. (дата обращения 16.06.2023).

[7] Audio Compression. URL: https://www.monkeysaudio.com/. (дата обращения 16.06.2023).

[8] Nowak N., Zabierowski W. Methods of Sound Data Compression – Comparison of Different Standards // Радиоэлектроника и информатика. 2011. № 4. URL: https://cyberleninka.ru/article/n/methods-of-sound-data-compression-comparison-of-different-standards (дата обращения 16.06.2023).

[9] Luthfi F., Erwin S. Data audio compression lossless FLAC format to lossy audio MP3 format with Huffman Shift Coding algorithm // 2016 4 th Intern. Conf. on Information and Communication Technology (ICoICT), Bandung, 2016. P. 1–5.

[10] Rupali B., Patil Dr., Kulat K. D. Audio compression using dynamic Huffman and RLE coding // 2017 2nd Intern. Conf. on Communication and Electronics Systems (ICCES), Coimbatore, 2017. P. 160–162.

[11] Ковалгин Ю. А., Вологдин Э. И. Цифровое кодирование звуковых сигналов. СПб.: КОРОНА-принт, 2004. 240 с.

[12] Ковалгин Ю. А., Фадеева Д. Р. Исследование психоакустических моделей кодеков с компрессией цифровых аудиоданных // Современная наука: актуальные проблемы теории и практики. 2016. № 7. С. 29–38.

[13] Ковалгин Ю. А. Психоакустика и компрессия цифровых аудиоданных / СПбГУТ. СПб., 2013. 300 с.

[14] Ковалгин Ю. А., Вологдин Э. И. Аудиотехника / под ред. проф. Ю. А. Ковалгина. М.: Горячая линия – Телеком, 2013. 768 с.

[15] Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ – Петербург, 2002. - 608 с.: илл.

[16] Сэломон Д. Сжатие данных, изображений и звука. – М.: Техносфера, 2004. – 368 с.

[17] Ковалгин Ю.А, Вологдин Э.И. Цифровое кодирование звуковых сигналов. – СПб: КОРОНА-принт, 2004. – 240 с.

[18] URL: https://intellect.icu/zadachi-parallelnykh-vychislenij-parallelnye-vychisleniya-zakon-amdala-5595 (дата обращения 16.06.2023).

[19] URL: http://onreader.mdl.ru/MasteringConcurrencyInPython/content/Ch02.html. (дата обращения 16.06.2023).