АЛГОРИТМЫ СЖАТИЯ ДАННЫХ ПРИ ОБРАБОТКЕ ИЗОБРАЖЕНИЙ И ВИЗУАЛИЗАЦИИ НА СУПЕРКОМПЬЮТЕРАХ

С.В. Коробков

В настоящее время кластерные системы используются для решения сложных задач с большими объемами данных. Появляется актуальная задача по уменьшению времени, затрачиваемого на передачу данных, и уменьшению ресурсов, в том числе памяти, затрачиваемых на хранение таких данных. Задача уменьшения времени передачи данных может решаться путем сжатия передаваемых данных перед их передачей и декомпрессии после их приема. MPICH [1], одна из наиболее используемых реализаций технологии MPI [2], передачи данных между узлами вычислителя, не оговаривает наличие или отсутствие сжатия данных при передаче их через сеть [3]. Для хранения данных также можно применять сжатие. В то же время, реализация таких алгоритмов сжатия должна, с одной стороны предусматривать возможность проведения наиболее часто осуществляемых операций над такими сжатыми данными без больших временных затрат, и с другой стороны предоставлять возможность общего доступа к данным. Так же, в связи с особенностями организации вычислителей, скорость доступа к внешнему хранилищу данных невелика и сильно увеличивает время обработки данных. В тоже время, данные, предназначенные к обработке, удобно было бы загружать на разные процессоры параллельно, независимо друг от друга. Поэтому, требуется применять форматы файлов, поддерживающие параллельную запись и чтение. Поэтому, алгоритмы сжатия должны поддерживать такие форматы.

Визуализация научных данных и обработка изображений на суперкомпьютерах является неотъемлемой частью большинства крупномасштабных вычислительных экспериментов, проводимых на суперкомпьютерах. В частности, трехмерная визуализация с построением реалистичных изображений в реальном времени является наиболее актуальной проблемой в области суперкомпьютерной визуализации. При решении этой большой задачи возникает ряд подзадач, которые требуется решить максимально эффективно. Одной из таких подзадач является эффективное сжатие данных при обработке и передаче изображений. При этом возможно решение проблемы сжатия для двух случаев: сжатие при хранении вспомогательной информации и промежуточных результатов на одном вычислительном узле и сжатие при передаче данных между различными вычислительными узлами.

В различного рода алгоритмах обработки изображений и визуализации данных больших объемов применяются таблицы, предназначенные для ускорения работы алгоритма. Такие таблицы можно разделить на 2 класса: таблицы отношений, и таблицы классификации. Первые содержат значения 0 и 1, которые показывают, состоят ли данные объекты в отношении или нет. Таблицы второго типа содержат значения классов, к которым данный объект относится. Но данные и в тех и в других таблицах чаще всего имеют блочную структуру из-за гладкости обрабатываемого изображения.

Алгоритмы сжатия, применяемые в данной области должны обеспечивать высокую скорость сжатия и декомпрессии, для обеспечения быстрого доступа к сжатым данным. Другим вариантом для обеспечения быстрого доступа к данным является возможность локальной декомпрессии данных и доступа к ним.

Одним из быстрых алгоритмов сжатия является алгоритм LZO [4]. Этот алгоритм часто применяется как быстрый алгоритм сжатия, в том числе применяется для сжатия памяти. С этим алгоритмом будут сравниваются предлагаемые алгоритмы. Эффективность данного алгоритма показана в статье [5].

Рассмотрим следующий алгоритм архивирования табличных данных и доступа к ним:

Изначально данные создаются уже в заархивированном виде, и память выделятся только под хранение этих заархивированных данных. Данные в заархивированном состоянии представляют собой количество элементов в хранимых данных, и пары чисел, где X_i представляет собой относительный адрес первого элемента сжимаемого блока в разархивированном массиве, а Y_i количество подряд идущих одинаковых элементов. Такие пары не записываются для значения по умолчанию. То есть сжатые данные можно представить в виде массива, как представлено на рисунке 1.

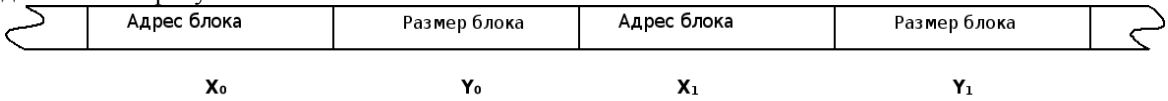

Рис. 1: Вид сжатых таблиц отношений.

Далее данные изменяются. Изменение происходит следующим образом:

- Если в изменяемом элементе раньше находилось значение по умолчанию, то необходимо провести следующие операции:

1. Создание новой пары чисел (x,y) для этого значения.
2. Находится место в массиве пар, в которое необходимо вставить данный блок. Все пары чисел должны быть упорядочены.
3. Созданная пара вставляется на нужное место.

4. Проверяются соседи вставленной пары на возможные слияния, и проводятся такие слияния.

- Если же в изменяемом элементе изменяется значение не по умолчанию, то у нас есть блок, к которому принадлежит адрес N, изменяемого значения, и соответствующая ему пара значений (x,y). Тогда необходимо разбить блок на 2 блока и исключить из них изменяемый элемент. И опять же провести слияния, если они возможны.

Слияние пар здесь происходит следующим образом:

1. Удаляются блоки длинны 0 если они есть.
2. Для соседних пар смотрится адрес блока и размер блока для правой пары и адрес блока для левой пары.
3. Если сумма адреса и размера блока для правой пары равна адресу блока для левой пары, то можно слить эти 2 пары в одну.
4. Проверка на слияние осуществляется только для изменяемых пар и пар, являющихся их непосредственными соседями.

Такого рода алгоритм позволяет хранить большие таблицы, существенно сократив объем памяти. В таблицах классификации же не получится выделить основное значение. Поэтому, кроме адреса блока и длинны блока, появляется третье значение, которое надо хранить. Но в таком случае, поскольку мы храним блоки одинаковых значений, для каждого из значений, выгодно не писать адрес блока, поскольку его довольно просто вычислить, просуммировав длины всех предыдущих блоков. Таким образом получается в архивированном состоянии хранится длина массива данных, и массив пар, где X_i – это количество подряд идущих одинаковых элементов, а Y_i – значение этих элементов. В результате, сжатые данные выглядят так, как представлено на рисунке 2.

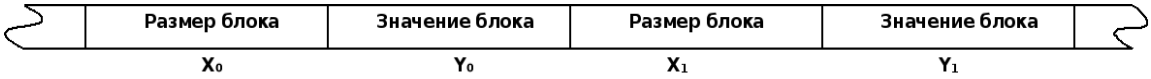

Рис. 2. Вид сжатых данных с классификационной информацией.

Действия, которые необходимо осуществить для изменения значения в таком массиве, мало отличаются от действий для предыдущего случая. То есть при изменении значения, находящегося в данных по адресу N, нам необходимо:

1. Создать новую пару для данного элемента.
2. Найти пару (x,y) соответствующую адресу N.
3. Разбить пару (x,y) на две, исключив из этой пары значение соответствующее N.
4. Проверить, чтобы размер пар не был равен нулю, а при наличии таких пар удалить их.
5. Записываются получившиеся пары в соответствующем порядке.
6. Проводятся слияния пар, если есть такая возможность.

Схематически процесс изменения значения по адресу N можно представить так, как представлено на рисунке 3. Здесь слева представлены несжатые данные, а справа представлены соответствующие им пары значений в сжатых данных. В заархивированном виде должны быть последовательно записаны три получившихся блока.

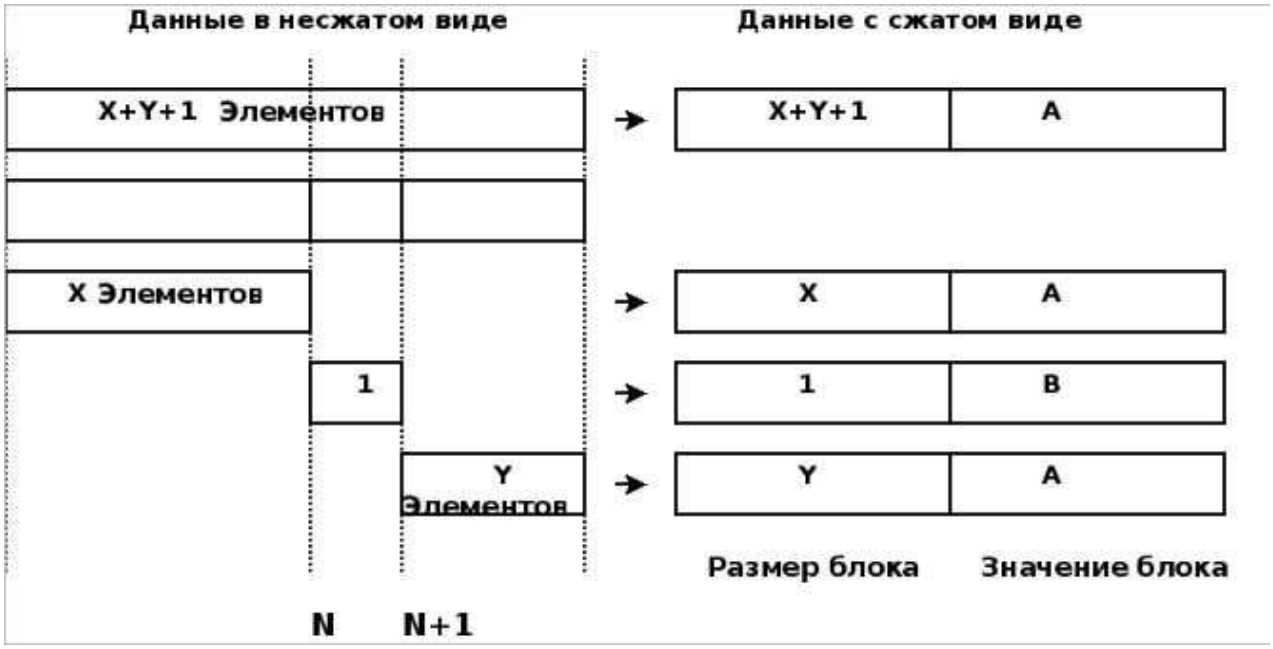

Рис. 3. Процесс изменения классификационных данных без декомпрессии.

Слияние пар в данном случае осуществляется просто проверкой значений блоков в соседних парах и в случае если значения одинаковые то данные пары можно объединить в одну, у которой размер блока является суммой размеров блоков объединяемых пар.

Такой алгоритм сжатия позволяет у всех элементов с одним значением менять значение на другое. Такое свойство очень полезно для таблиц хранящих классификацию.

Такого рода таблицы иногда требуется передать на другой вычислительный узел. Если же таблица хранится в несжатом виде, то сжать такую таблицу можно довольно быстро. Для этого необходимо, последовательно проходя по несжатым данным, записать все пары соответствующие этим блокам данных.

Для проверки эффективности алгоритмов использовались случайно сгенерированные данные блочной структуры. Для проверки эффективности алгоритма сжатия таблиц отношений применялись данные блочной структуры, состоящие из единиц и нулей. Средняя длина блока 20 элементов. Для проверки эффективности сжатия данных классификации использовались данные блочной структуры со значениями от 0 до 250. Средняя длина блока 120 элементов. Так же для сравнения алгоритм сжатия классификаций применялся и на таблицах отношений. Общая длина данных была взята 400 000 000 элементов. В качестве элемента данных бал взят байт. Таким образом размер несжатых данных: 400 МБ. Для тестирования передачи сжатых данных использовался установленный в МГУ имени М.В.Ломоносова вычислительный комплекс BlueGene /P.

Результаты оценки эффективности сжатия представлены в следующих таблицах. В таблице 1 приведены данные по сжатию данных классификаций предложенным алгоритмом для сжатия классификаций и алгоритмом LZO. В таблице 2 приведены данные по сжатию таблиц отношений 2 предложенными алгоритмами и алгоритмом LZO. В таблице 3 приведены время пересылки несжатых данных и суммарное время сжатия, пересылки и декомпрессии данных. Сжатие для пересылки осуществлялось предложенным алгоритмом для сжатия классификаций. Время во всех таблицах измеряется в секундах.

Таблица 1: Параметры сжатия данных классификации.

Алгоритм/данные	Коэффициент сжатия	Время на сжатие	Время на декомпрессию
Алгоритм сжатия классификаций	50.3224	3.43 сек	2.15 сек
LZO	21.0605	2.9 сек	1.1 сек

Таблица 2: Параметры сжатия таблиц отношений.

Алгоритм/данные	Коэффициент сжатия	Время на сжатие	Время затраченное на 400 доступов по записи	Время затраченное на 400 доступов по чтению
Алгоритм сжатия таблиц отношений	5.38403	4.84 сек	1.87 сек	0.00001 сек
Алгоритм сжатия классификаций	10.7237	6.17 сек	106.5 сек	100.19 сек
LZO	5.86319	7.19 сек	Не проводилось	Не проводилось

Таблица 3: Время передачи несжатых данных и время передачи сжатых данных, с учетом времени на сжатие и декомпрессию на BlueGene/P.

Алгоритмы	Не сжатые данные	Сжатые данные
Алгоритм сжатия классификаций	1.0667 сек	15.4667 сек
LZO	1.0667 сек	8.873 сек

Если посмотреть на данные, показанные в таблицах 1 и 2, то можно увидеть, что скорость сжатия и декомпрессии данных велика и сравнима с простым копированием памяти. Скорость копирования памяти на современных компьютерах составляет 300 МБ в секунду. В предлагаемом алгоритме скорость сжатия данных всего лишь в 3.5 раза ниже и достигает значения 80 МБ в секунду и даже больше. Как видно из таблицы 2 скорость доступа по чтению к памяти в сжатом виде довольно высока и всего лишь в несколько раз ниже скорости доступа к несжатой памяти. В то же время предлагаемый алгоритм обеспечивает достаточно высокую степень сжатия данных. Степень сжатия данных почти такая же, какую обеспечивает и выбранный для сравнения алгоритм LZO.

Из таблицы 1 видно, что предлагаемый алгоритм обеспечивает в 2 раза лучше коэффициент сжатия чем алгоритм LZO, при этом суммарно на сжатие и декомпрессию затрачивает времени не более чем в 2 раза больше. Несмотря на загруженность проводящей среды вычислитель, BlueGene /P показывает заявленную в [3] скорость в 5.1 ГБ в секунду. Данная скорость передачи, заявленная производителями, представляет собой скорость общей шины на которой находятся 32 вычислительных узла, что довольно легко вычислить поскольку в каждой группе из 32 вычислительных узлов 16 передавали данные и 16 принимали и за секунду каждый узел успел послать 400 МБ данных. Соответственно, при загруженной среде передачи, скорость передачи между узлами в одном вычислительном блоке составляет 400 МБ в секунду.

В статье [6] приведена формула, по которой можно определить максимальное количество времени, которое может быть затрачено на обработку каждого байта данных. Формула представляет собой ограничение на время обработки одного байта:

$$\grave{l}_1 \leqslant \frac{R-1}{R*BW} \qquad (2)$$

где: – время, затрачиваемое на сжатие каждого байта, R – коэффициент сжатия данных, BW – скорость связи узлов между друг другом в байтах в секунду. Соответственно подставив параметры вычислительной системы BlueGene /P и параметры сжатия получим следующее ограничение . В рассмотренных алгоритмах сжатия . По этим числам видно, что алгоритм должен работать не менее чем в 5.71428 раз быстрее, чем работает на данный момент.

В то же время, важно отметить, что при обработке данных большого объема блочной структуры, вряд ли часто будут требоваться эти данные в несжатом виде. Соответственно при передаче таких данных не будет требоваться сжимать и разжимать эти данные, поскольку данные уже будут находиться в сжатом виде.

В данной работе были реализованы алгоритмы сжатия для специфических данных и проведено исследование эффективности данных алгоритмов.

Можно выделить следующие основные результаты данной работы:

- Реализованы алгоритмы сжатия, обеспечивающие доступ к данным без их полной декомпрессии.
- Показана эффективность алгоритма сжатия в памяти, показана эффективность чтения данных без их полной декомпрессии.
- Показано, что предлагаемый алгоритм имеет преимущества по следующим параметрам: коэффициент сжатия и скорость доступа к сжатым данным.
- Показано, что предложенный алгоритм, применяемый при передаче данных по коммуникациям вычислителя BlueGene/P, не обеспечивает уменьшения времени работы программы. Причиной этому является небольшая вычислительная способность процессора в BlueGene/P , и в то же время очень быстрые коммуникации, которые обеспечивают быструю передачу даже не сжатых данных.

В дальнейшей работе планируется проверить эффективность работы предложенных алгоритмов в случае записи на внешние носители. Так же планируется исследовать возможность уменьшения времени доступа к сжатым данным. Для достижения уменьшения времени доступа к данным, предполагается использование кешей для уменьшения времени поиска пар соответствующих значениям. Так же планируется произвести исследование эффективности работы алгоритмов сжатия при передачи данных на других вычислительных системах. Для упрощения использования алгоритмов планируется создать библиотеку с разработанными алгоритмами и адаптировать ее для использования в библиотеке Parimpr визуализации и обработки изображений на суперкомпьютерах.

Работа выполнена в рамках НОЦ «Суперкомпьютерные технологии для решения задач обработки, хранения, передачи и защиты информации» при поддержке Федеральной целевой программы "Научные и научно-педагогические кадры инновационной России" на 2009 - 2013 годы и грантов РФФИ 08-07-00445-а, 09-07-12068-офи_м. Автор выражает благодарность своим научным руководителям, Поповой Нине Николаевне и Джосан Оксане Васильевне, и коллегам по кафедре Автоматизации систем вычислительных комплексов ВМК МГУ за ценные замечания по содержанию работы.

ЛИТЕРАТУРА:

1. W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. Parallel Computing, 22(6):789–828, September 1996.
2. MPI: A Message-Passing Interface Standard.
3. Carlos Sosa and Brant Knudson. IBM System Blue Gene Solution: Blue Gene/P Application Development. International Technical Support Organization, August 2009.
4. Lzo, http://www.oberhumer.com/opensource/lzo/.
5. Alejandro Calderón Rosa Filgueira, David E. Singh and Jesús Carretero. Compi: Enhancing mpi based applications performance and scalability using run-time compressio. In Recent Advances in Parallel Virtual Machine and Message Passing Interface, pages 207–218. Springer Berlin / Heidelberg, 2009.
6. Evan Speight Jian Ke, Martin Burtscher. Runtime compression of mpi messages to improve the performance and scalability of parallel applications. In Proceedings of the 2004 ACM/IEEE conference on Supercomputing, page 59, November 2004.