Применение сжатия информации при использовании многоядерных ускорителей для обработки баз данных*

К.Ю. Беседин, П.С. Костенецкий

Южно-Уральский государственный университет (НИУ)

Одной из проблем, возникающих при использовании многоядерных сопроцессоров Intel Xeon Phi для обработки баз данных, является необходимость передачи данных в память сопроцессора по шине PCI Express. В данной работе исследуется использование методов сжатия RLE, Null Suppression, LZSS и комбинации методов RLE и Null Suppression для ускорения обмена данными с сопроцессором. Была выполнена реализация выбранных методов для сопроцессора Intel Xeon Phi. Проведены вычислительные эксперименты, показавшие, что все рассмотренные методы сжатия позволяют повысить эффективность обработки баз данных на многоядерном процессоре при выполнении определенных условий относительно обрабатываемых данных. Также показано, что эффективность применения методов RLE и Null Suppression, а также их комбинации может быть дополнительно повышена за счет обработки данных в сжатом виде без предварительной распаковки.

1. Введение

Обработка сверхбольших объемов данных является одной из актуальных на сегодняшний день научных проблем. Одним из важных подходов к решению этой проблемы является использование параллельных СУБД [3, 5, 6, 15, 18]. В рамках данного подхода особую роль играет организация эффективной обработки запросов к базам данных. Использование для данной цели многоядерных сопроцессоров и графических ускорителей является одним из перспективных направлений исследований [1, 4]. Как графические ускорители, так и многоядерные сопроцессоры обладают рядом технических особенностей, которые необходимо принимать во внимание при разработке высокопроизводительных алгоритмов для данных устройств. Одной из таких ключевых особенностей является необходимость передачи данных по шине PCIe из основной памяти в память устройства и обратно. Поскольку пропускная способность шины PCIe в несколько раз ниже пропускной способности основной памяти, то такая передача данных считается одним из «узких мест» при программировании для GPU и многоядерных сопроцессоров [1, 2, 13, 14]. В данной ситуации существует несколько путей повышения производительности кода для данных устройств. Во-первых, разработчики оборудования и инструментального ПО к нему предоставляют различные средства, позволяющие эффективно использовать возможности устройств, например, возможность параллельно выполнять передачу данных и выполнения кода на устройстве или выполнять передачу только измененных элементов данных. Во-вторых, разработчики алгоритмов учитывают данную особенность и сводят к минимуму количество данных, передаваемых между GPU или многоядерным сопроцессором и основной памятью. Часть из вносимых таким образом изменений в алгоритмы не зависит от предметной области. К таким относятся, например, сохранение данных на устройстве или предпочтение вычисления значений данных их передаче из основной памяти. Некоторые же изменения специфичны для предметной области. Одним из примеров таких изменений можно считать применение сжатия данных в контексте систем баз данных.

*Работа выполнена при финансовой поддержке Минобрнауки РФ в рамках ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2014–2020 годы» (Соглашение № 14.574.21.0035).

Целью данного исследования является проверка возможности использования сжатия данных для сокращения времени их передачи из основной памяти в память сопроцессора. Также исследуется возможность обработки сжатых данных без предварительной распаковки. В рамках данной работы рассматриваются три используемых в СУБД метода сжатия данных: Run Length Encoding (RLE, кодирование длин серий), Null Suppression (подавление нулей) и LZSS (Lempel-Ziv-Storer-Szymanski). Представлены результаты исследования эффективности комбинаций методов сжатия RLE и Null Suppression. Разработаны параллельные алгоритмы, реализующие рассматриваемые методы сжатия. Выполнена реализация разработанных алгоритмов на языке C++ с использованием технологии OpenMP. Проведены вычислительные эксперименты, исследующие эффективность использования рассмотренных методов сжатия при передаче данных из основной памяти в память сопроцессора Intel Xeon Phi.

В разделе "Обзор работ по тематике исследования" приведен обзор работ, посвященных применению сжатия данных в СУБД и реализации методов сжатия на GPU. В разделе "Реализация методов сжатия" приведено описание выполненной реализации рассматриваемых методов сжатия. Раздел "Вычислительные эксперименты" описывает проведенные вычислительные эксперименты. Для каждого рассматриваемого метода сжатия представлена информация о ходе проведения экспериментов над этим методом, результаты экспериментов в виде графиков и сделанные на основе этих результатов выводы.

2. Обзор работ по тематике исследования

Сжатие данных является хорошо изученным и распространенным приемом, применяемым в системах баз данных, как для увеличения максимального объема хранимых данных, так и для повышения производительности за счет уменьшения объема данных, передаваемых во время операций ввода-вывода [11, 12, 16, 19]. Сжатие данных в поколоночных СУБД часто рассматривается отдельно. Особенности поколоночного хранения данных позволяют повысить как степень сжатия данных, так и скорость их обработки по сравнению с несжатыми данными [7–9].

Реализации алгоритмов сжатия данных на графических ускорителях посвящено несколько научных работ. В [21] рассмотрена реализация как алгоритмов сжатия с потерями (JPEG, vorbis), так и без потерь (LZ77). В работе [17] представлена реализация алгоритма LZSS. В [10] рассмотрено несколько алгоритмов сжатия в контексте систем баз данных. Исследований алгоритмов сжатия на Intel Xeon Phi до данного момента не проводилось.

Одним из перспективных подходов к использованию сжатия является использование комбинации из нескольких простых методов сжатия, таких как RLE, Null Suppression, Dictionary Coding и других. В работе [10] показано, что использование таких комбинаций для сжатия данных в контексте СУБД может обладать большей эффективностью, чем применение комбинируемых методов по отдельности.

3. Реализация методов сжатия

Для проведения исследований была выполнена реализация всех выбранных методов сжатия для сопроцессора Intel Xeon Phi. Для этого была использована технология OpenMP и язык программирования C++. Специальной реализации методов сжатия для центральных процессоров выполнено не было, так как этого не требовалось для достижения целей исследования. Также, исходя из целей исследования, был выбран offload-режим использования сопроцессора.

3.1. Алгоритм Null Suppression

Сжатие данных с использованием метода Null Suppression заключается в удалении нулей в старших битах каждого элемента сжимаемых данных. Число удаляемых нулей определяется тем числом нулей, которое можно удалить у элемента, чье значение занимает наибольшее число бит. Число нулей, удаленных при сжатии, сохраняется в качестве метаданных архива. Нами предлагается следующий алгоритм, реализующий данную схему сжатия:

1. определение числа удаляемых нулей (для простоты считается, что значение элемента, занимающего наибольшее число бит, известно заранее);
2. удаление определенного ранее числа нулей.

Для распаковки данных осуществляется обратная операция: добавление определенного числа нулей в начало каждого элемента сжатых данных. Как во время сжатия, так и во время распаковки, каждый элемент данных обрабатывается независимо от остальных. Это позволяет естественным образом реализовать параллельные версии этих алгоритмов.

3.2. Алгоритм RLE

Для сжатия данных по методу RLE предлагается использовать следующий алгоритм:

1. вычисление границ серий. В оперативной памяти формируется массив двоичных значений. Для элементов, которые завершают свою серию в этом массиве ставится 1, для остальных элементов — 0;
2. на основе данных о границах серии вычисляется их общее число и смещения серий в итоговом массиве значений серий;
3. вычисление длин серий и заполнение итоговых массивов значений серий и длин серий.

Распаковка данных, сжатых по методу RLE, происходит в два этапа:

1. вычисление объема распакованных данных и смещений в нем серий на основе массива длин серий;
2. на основе вычисленного на предыдущем этапе массива и массива значений серий заполняется итоговый массив с распакованными данными.

В отличие от метода Null Suppression, метод сжатия RLE не позволяет обрабатывать элементы сжимаемых данных независимо друг от друга. Для выполнения параллельного сжатия необходимо разделить последовательность сжимаемых данных на непересекающиеся подпоследовательности, которые можно обрабатывать независимо друг от друга. Распаковка данных, сжатых алгоритмом RLE, позволяет обрабатывать каждый элемент архива отдельно.

3.3. Алгоритм LZSS

В отличие от рассмотренных ранее методов сжатия, оригинальный алгоритм сжатия LZSS, описанный в [20] не позволяет выполнить параллельное сжатие и распаковку данных. Для обеспечения параллельного сжатия и распаковки, сжимаемые данные и сжатый архив разделяются на некоторое число отдельных сегментов, обработка которых может быть произведена независимо друг от друга.

Алгоритм состоит из следующих шагов:

1. распределение сжимаемых данных по отдельным сегментам;

2. параллельное сжатие каждого сегмента данных по отдельности с подсчетом размера сжатых сегментов;

3. подсчет общего размера сжатых данных и смещений сегментов в итоговом архиве;

4. копирование отдельно сжатых сегментов в итоговый архив;

5. добавление к архиву метаданных (размер словаря, число и размеры сегментов).

Распаковка данных осуществляется следующим образом:

1. вычисление общего размера распакованных данных на основе метаданных архива;

2. параллельная распаковка сжатых сегментов.

4. Вычислительные эксперименты

Вычислительные эксперименты производились на вычислительном кластере «Торнадо ЮУрГУ», характеристики которого представлены в табл. 1

Степень сжатия каждого из рассматриваемых методов зависит от определенных характеристик сжимаемых данных [10]. Для каждого из методов сжатия будет представлена такая характеристика и продемонстрировано ее влияние на эффективность использования метода.

Некоторые методы сжатия позволяют осуществлять обработку сжатых данных без предварительной распаковки [8]. Среди рассмотренных в данной работе методов, к ним относятся методы RLE, Null Suppression и комбинация RLE и Null Suppression. Для них были проведены соответствующие эксперименты.

Все схемы были протестированы на данных объемом 250 Мб, 500 Мб, 750 Мб, 1000 Мб, 1250 Мб и 1500 Мб. В данном разделе представлены графики для 1500 Мб. Результаты для других объемов данных аналогичны.

Во всех вычислительных экспериментах, сжимаемые данные представляют собой массивы из беззнаковых целых чисел типа `uint64_t`. Вычислительные эксперименты, тестирующие методы RLE и LZSS использовали одни и те же наборы тестовых данных. Для вычислительных экспериментов, тестирующих метод Null Suppression использовался отдельный набор тестовых данных.

Для каждого метода сжатия данных демонстрируется зависимость степени сжатия от выбранной характеристики сжимаемых данных. Также, для каждого метода сжатия сравниваются следующие показатели:

- время обработки сжатых данных с их распаковкой;
- время обработки данных в сжатом виде (если это возможно).

Процессор узла	Intel Xeon 5680 3.33 ГГц
Объем ОЗУ узла	24 / 48 Гб
Число процессоров в узле	2
Число узлов	480
Число узлов с сопроцессорами	384

Таблица 1. Суперкомпьютер «Торнадо ЮУрГУ»

Под временем обработки понимается сумма времени передачи данных, их распаковки (если требуется) и время выполнения над ними агрегатной функции - вычисления суммы При выполнении экспериментов считается, что данные находятся в оперативной памяти вычислительной системы в уже сжатом виде. Также для сравнения приведено время обработки тестовых данных в несжатом виде.

4.1. Метод RLE

Метод сжатия RLE эффективен в тех случаях, когда длины серий в сжимаемых данных достаточно велики, чтобы совокупность их закодированных представлений занимала меньший объем памяти, чем сами серии [19]. Это выполняется тогда, когда число серий достаточно мало. Это число зависит от конкретной реализации метода сжатия RLE и от объема сжимаемых данных. Иногда, для оценки эффективности метода сжатия используется не число серий, а их средняя длина [10]. В этом случае, как и с числом серий, для определения степени сжатия требуется учитывать особенности конкретной реализации метода и объем сжимаемых данных. Для упрощения представления результатов вычислительных экспериментов для разных объемов данных в качестве варьируемой характеристики используется отношение числа серий на общее число элементов в тестовых данных. В проведенных экспериментах отношение числа серий к общему объему данных варьировалось от 0.01 до 1.

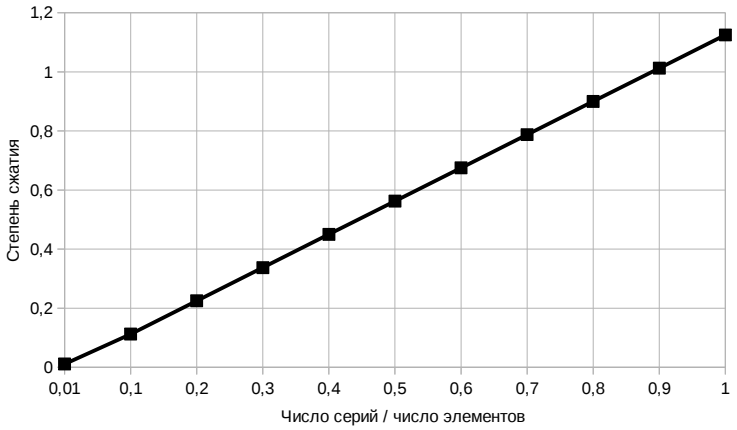

Рис. 1. Степень сжатия методом *RLE*

На рис. 1 показана зависимость степени сжатия 1500 Мб данных методом RLE от отношения числа серий к общему числу элементов данных. Из графика видно, что выполненная реализация метода эффективна, когда число серий не превышает 90% от числа сжимаемых элементов. На рис. 2 показана зависимость времени обработки 1500 Мб сжатых методом RLE данных от отношения числа серий к общему числу сжимаемых элементов. Из графика видно, что метод RLE позволяет ускорить обработку баз данных на сопроцессоре Intel Xeon Phi когда соотношение числа серий в сжимаемых данных к общему числу сжимаемых элементов не превышает 0,6 в случае предварительной распаковки сжатых данных, либо 0,9, в случае обработки данных в сжатом виде.

На основании полученных результатов можно сделать следующие выводы:

- метод сжатия RLE может быть эффективно использован при передаче данных в память сопроцессора Intel Xeon Phi если число серий в данных достаточно мало;
- обработка данных в сжатом виде позволяет дополнительно увеличить эффективность использования метода RLE.

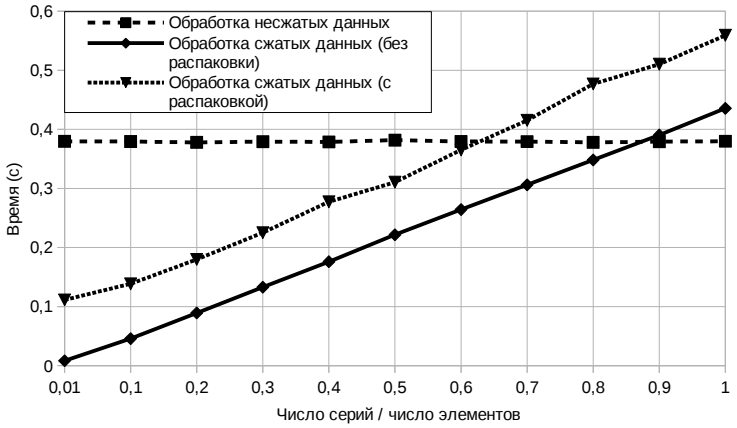

Рис. 2. Время обработки данных, сжатых методом *RLE*

4.2. Метод LZSS

Для описания зависимости эффективности сжатия данных методом LZSS от характеристик сжимаемых данных была использована та же характеристика, что и для метода RLE - отношение числа серий к общему числу сжимаемых данных. В отличие от методов RLE и Null Suppression, метод LZSS не позволяет производить обработку данных без их распаковки.

На рис. 3 показана зависимость степени сжатия методом LZSS от выбранной характеристики при обработке 1500 Мб данных. График показывает, что степень сжатия, обеспечиваемая выполненной реализацией метода LZSS, изменяется от 0,04 до 0,5. Видна зависимость степени сжатия от выбранной характеристики данных. Также следует отметить, что выполненная реализация LZSS в большинстве случаев обеспечивает более высокую, чем остальные методы степень сжатия.

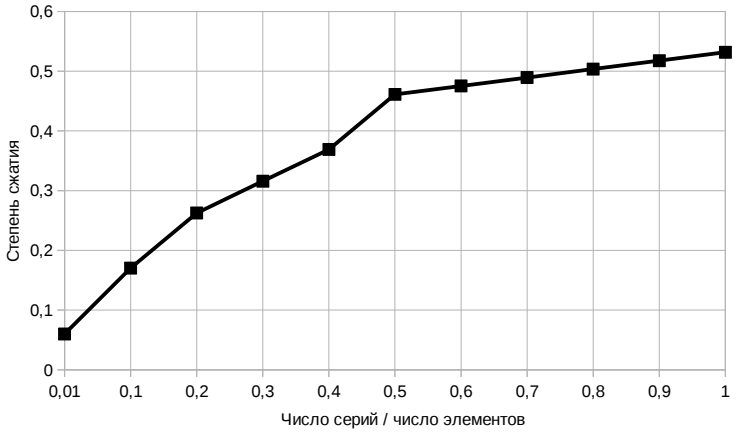

Рис. 3. Степень сжатия методом *LZSS*

На рис. 4 показана зависимость времени обработки данных, сжатых методом LZSS от отношения числа серий к общему числу сжимаемых элементов данных. Из графика видно, что время, требуемое на передачу сжатых данных, их распаковку и обработку, в большинстве случаев больше времени, требуемого на передачу и обработку несжатых данных. Это вызвано тем, что особенности метода сжатия LZSS не позволяют эффективно использовать векторные операции при распаковке данных.

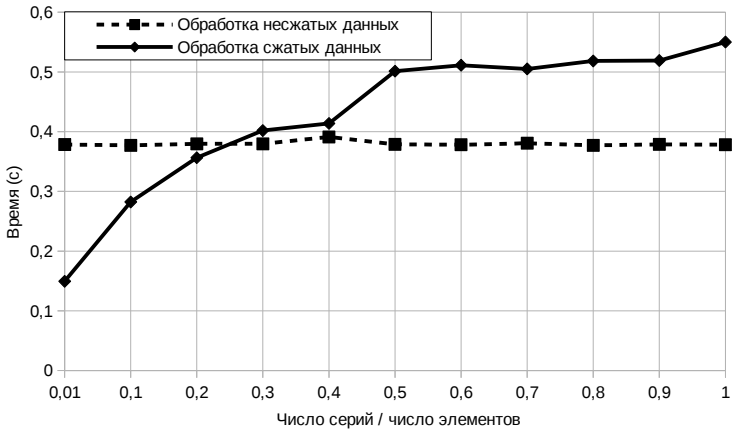

Рис. 4. Время обработки данных, сжатых методом *LZSS*

Исходя из результатов проведенных исследований, можно заключить, что метод сжатия LZSS может быть эффективно использован при передаче данных из основной памяти в память сопроцессора Intel Xeon Phi только в тех случаях, когда число серий в сжимаемых данных мало.

4.3. Null Suppression

Эффективность сжатия метода Null Suppression зависит только от значений сжимаемых данных. Для рассматриваемых наборов данных, степень сжатия будет определяться значением максимального элемента сжимаемых данных, которое будет варьироваться так, чтобы оно требовало 1, 2, 4 или 8 байт для сжатия.

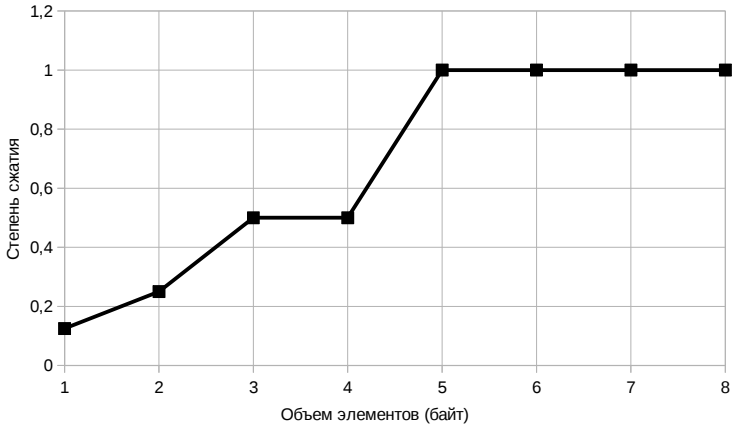

Рис. 5. Степень сжатия методом *Null Suppression*

На рис. 5 показана зависимость степени сжатия данных методом Null Suppression от максимального значения сжатых данных. В лучшем случае, степень сжатия составляет 0.125, а в худшем - 1. На рис. 6 показана зависимость времени обработки 1500 Мб данных, сжатых методом Null Suppression от значения максимального элемента этих данных. Из графика видно, что передача и обработка сжатых данных с их распаковкой эффективнее передачи и обработки несжатых данных в случае, когда максимальный элемент сжимаемых данных кодируется менее чем четырьмя байтами. Время передачи и обработки сжатых данных без их распаковки не превышает времени передачи и обработки несжатых данных.

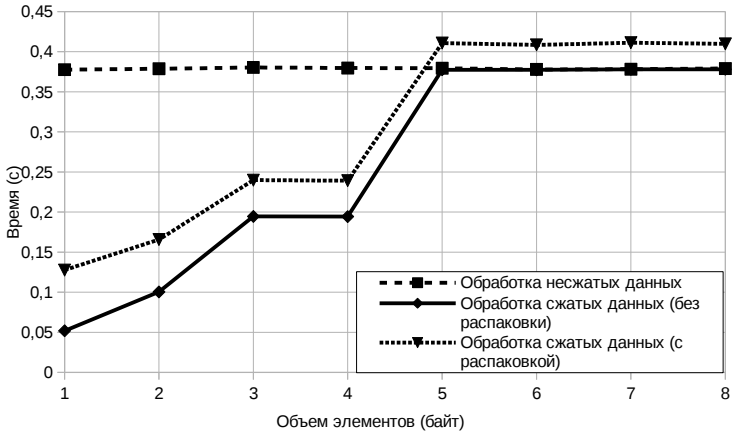

Рис. 6. Время обработки данных, сжатых методом *Null Suppression*

На основании проведенных экспериментов можно сделать следующие выводы:

- метод сжатия Null Suppression может быть эффективно использован для организации обработки баз данных на сопроцессоре, если значения элементов сжимаемых данных позволяют отбрасывать нулевые старшие байты без потерь данных;
- обработка данных в сжатом виде дополнительно повышает эффективность данного метода.

4.4. Комбинация RLE и Null Suppression

На данном этапе исследования рассматривался случай, когда метод Null Suppression применяется для повышения эффективности использования метода RLE. Соответственно, варьируемая характеристика совпадает с той, что была использована для экспериментов над методом RLE.

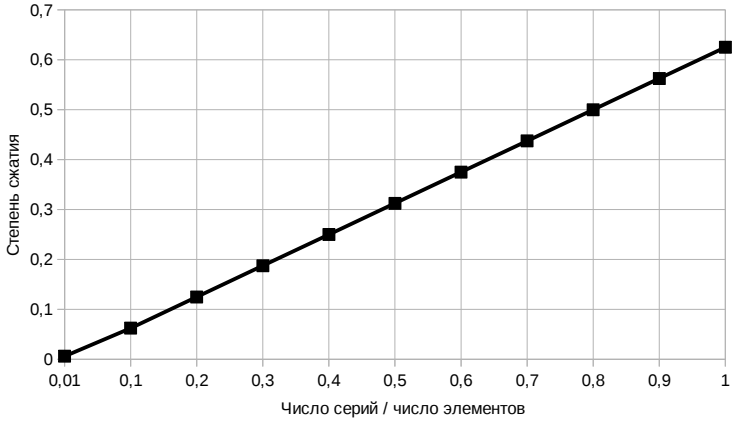

Рис. 7. Степень сжатия данных комбинацией методов RLE и Null Suppresion

На рис. 7 представлена зависимость степени сжатия данных от выбранной характеристики. С помощью данной комбинации удалось достичь степени сжатия от 0.00625 до 0.625. На рис. 8 представлена зависимость обработки данных, сжатых комбинаций методов RLE и Null Suppression. Видна зависимость времени обработки сжатых данных от выбранной характеристики. Также видно, что обработка данных в сжатом виде осуществляется быстрее, чем обработка данных с предварительной их распаковкой.

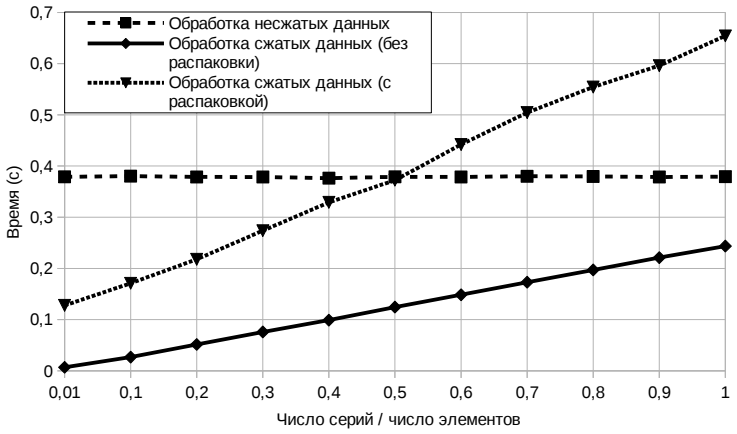

Рис. 8. Время обработки данных, сжатых комбинаций методов RLE и Null Suppresion

На основании данных экспериментов были сделаны следующие выводы:

- комбинация методов RLE и Null Suppression может быть использована для повышения эффективности метода RLE;
- обработка данных в сжатом виде дополнительно повышает эффективность данной комбинации методов.

Заключение

В рамках данной работы рассмотрено три метода сжатия данных: Run Length Encoding, Null Suppression и LZSS, а также комбинация методов RLE и Null Suppression. Были разработаны параллельные алгоритмы, реализующие рассматриваемые методы сжатия. Разработанные алгоритмы были реализованы на языке C++ с использованием технологии OpenMP. Проведены вычислительные эксперименты, показывающие, что все рассмотренные методы сжатия могут быть эффективно использованы для обработки баз данных на сопроцессоре Intel Xeon Phi в случае, когда обрабатываемые данные удовлетворяют определенным условиям. Кроме того, показано, что обработка данных в сжатом виде при использовании методов RLE и Null Suppression и их комбинации позволяет дополнительно увеличить эффективность применения данных методов.

Дальнейшими направлениями исследований будут:

- исследование эффективности использования других методов сжатия;
- исследование эффективности применения комбинаций из нескольких методов сжатия в контексте данной задачи.

Литература

1. Беседин К.Ю., Костенецкий П.С. Моделирование обработки запросов на гибридных вычислительных системах с многоядерными сопроцессорами и графическими ускорителями // Программные системы: теория и приложения: электрон. научн. журн. Института программных систем им. А.К. Айламазяна РАН. 2014. Т. 5, № 1(19), с. 91 – 110.
2. Беседин К.Ю., Костенецкий П.С. Применение многоядерных сопроцессоров в параллельных системах баз данных // Параллельные вычислительные технологии (ПаВТ'2013): труды международной научной конференции (1–5 апреля 2013 г., г. Челябинск). Челябинск: Издательский центр ЮУрГУ, 2013. с. 583

3. Костенецкий П.С., Соколинский Л.Б. Моделирование иерархических многопроцессорных систем баз данных // Программирование. Москва: МАИК “Наука/Интерпериодика”. Т. 39 № 1. 2013. с. 3–22.

4. Костенецкий П.С. Обработка запросов на кластерных вычислительных системах с многоядерными ускорителями // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2012. № 47 (306). С. 59–67.

5. Костенецкий П.С., Лепихов А.В., Соколинский Л.Б. Технологии параллельных систем баз данных для иерархических многопроцессорных сред // Автоматика и телемеханика. 2007. № 5. С. 112-125.

6. Соколинский Л.Б. Параллельные машины баз данных // Природа. 2001. № 8. С. 10–17.

7. Abadi D.J., Madden S.R., Ferreira M.C. Integrating Compression and Execution in Column-oriented Database Systems // ACM SIGMOD International Conference on Management of Data Chicago, IL, USA, June 26–29, 2006, Proceedings. ACM, 2006. – P. 671–682.

8. Abadi D.J., Madden S.R., Hachem N. Column-stores vs. Row-stores: How Different Are They Really? // ACM SIGMOD International Conference on Management of Data Vancouver, Canada, June 10–12, 2008, Proceedings. ACM, 2008. – P. 967–980.

9. Binnig C., Hildenbrand S., Färber F. Dictionary-based Order-preserving String Compression for Main Memory Column Stores // ACM SIGMOD International Conference on Management of Data Providence, Rhode Island, USA, June 29 – July 2, 2009, Proceedings. ACM, 2009. – P. 283–296.

10. Fang W., He B., Luo Q. Database Compression on Graphics Processors // 36th International Conference on Very Large Data Bases Singapore, September 13–17, 2010. Proceedings. VLDB Endowment, 2010. – P. 670–680.

11. Graefe G., Shapiro L. Data compression and database performance // ACM/IEEE-CS Symposium on Applied Computing, Kansas City, USA, April 3–5, 1991 Proceedings. IEEE, 1991. – P. 22–27.

12. Iyer B.R., Wilhite D. Data Compression Support in Databases // 20th International Conference on Very Large Data Bases Santiago de Chile, Chile, 12–15 September, 1994, Proceedings. Morgan Kaufmann Publishers Inc., 1994. – P. 695–704.

13. Jeffers, James Reinders Intel Xeon Phi coprocessor high-performance programming. USA: Elsevier, 2013.

14. Kirk D.B., Hwu W.W. Programming massively parallel processors. A hands-on approach. 2nd edition. USA: Elsevier, 2013.

15. Kostenetskiy P. S., Sokolinsky L. B. Analysis of Hierarchical Multiprocessor Database Systems // 2007 International Conference on High Performance Computing, Networking and Communication Systems (HPCNCS-07), Orlando, FL, USA. July 9-12 2007, Proceedings. ISRST, 2007. – P. 245-251.

16. Ng W.K, Ravishankar C.V. Block-Oriented Compression Techniques for Large Statistical Databases // IEEE Trans. on Knowl. and Data Eng. 1997. – Vol. 9, – No. 2 , – P. 314–328.

17. Ozsoy A., Swany M. CULZSS: LZSS Lossless Data Compression on CUDA // 2011 IEEE International Conference on Cluster Computing, Washington, DC, USA, 26–30 September, 2011. Proceedings. IEEE Computer Society, 2011. – P. 403–416.

18. Pan C., Zymbler M. Taming Elephants, or How to Embed Parallelism into PostgreSQL // Database and Expert Systems Applications — 24th International Conference, DEXA 2013, Prague, Czech Republic, August 26–29, 2013. Proceedings, Part I. Springer, 2013. Lecture Notes in Computer Science. Vol. 8055. P. 153–164.

19. Roth M.A., Van Horn S.J. Database Compression // ACM SIGMOD Record, 1993. – Vol. 22, – No. 3, – P. 31–39.

20. Storer, James A. and Szymanski, Thomas G. Data Compression via Textual Substitution // J. ACM, 1982. – Vol 29. – No. 4. – P. 928.–951.

21. Wu L., Storus M., Cross D. Cs315a: Final project cuda wuda shuda: Cuda compression project. Technical report. – USA: Stanford University, 2009 – 11 p.