УНИВЕРСИТЕТ ИТМО
НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
май–июнь 2015 Том 15 № 3 ISSN 2226-1494 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
May–June 2015 Vol. 15 No 3 ISSN 2226-1494 http://ntv.ifmo.ru/en

УДК 004.627

АНАЛИЗ И ОЦЕНКА МИНИМАЛЬНОГО УРОВНЯ ПРЕФИКСНОГО ДЕРЕВА В СИСТЕМЕ БЕСХЕШЕВОЙ ДЕДУПЛИКАЦИИ

М.А. Жуков[a], Д.Б. Афанасьев[a]

[a] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация
Адрес для переписки: fenderst11@gmail.com

Информация о статье
Поступила в редакцию 17.12.14, принята к печати 16.02.15
doi:10.17586/2226-1494-2015-15-3-470-475
Язык статьи – русский

Ссылка для цитирования: Жуков М.А., Афанасьев Д.Б. Анализ и оценка минимального уровня префиксного дерева в системе бесхешевой дедупликации // Научно-технический вестник информационных технологий, механики и оптики. 2015. Т. 15. № 3. С. 470–475.

Аннотация

Предмет исследования. Предложен метод ограничения минимального уровня префиксного дерева в системе бесхешевой дедупликации данных.

Метод. Сущность предлагаемого метода заключается в принудительном заполнении префиксного дерева до определенного минимального уровня. Использование предлагаемого метода позволяет снизить количество коллизий на нижних уровнях префиксного дерева. Максимальный теоретический прирост производительности соответствует доле коллизий от общего количества операций чтения данных с носителя. Применение метода ограничения минимального уровня префиксного дерева увеличивает объем метаданных на объем новых структур, содержащих один элемент.

Основные результаты. Результаты работы подтверждены данными вычислительного эксперимента бесхешевой дедупликации на наборе данных объемом 528 ГБ. Анализ процесса показал, что 99% времени выполнения занимает позиционирование головок жестких дисков. Причиной этого является распределение блоков на носителе в случайном порядке. На экспериментальном наборе данных применение метода ограничения минимального уровня префиксного дерева может увеличить производительность на 16%, а возрастание объема метаданных составит 49%. Общий объем метаданных будет меньше на 34%, чем при применении метода хешевой дедупликации с использованием алгоритма MD5, и на 17% меньше, чем с использованием алгоритма Tiger192. Полученные результаты подтверждают эффективность предложенного метода.

Практическая значимость. Предложенный метод позволяет увеличить производительность процесса за счет сокращения количества коллизий при построении префиксного дерева. Результаты представляют практическую значимость для специалистов, занимающихся разработкой системы бесхешевой дедупликации данных.

Ключевые слова
бесхешевая дедупликация, увеличение производительности дедупликации, префиксное дерево, операции ввода-вывода, объем метаданных, эксперимент бесхешевой дедупликации данных.

ANALYSIS AND ESTIMATION OF THE TRIE MINIMUM LEVEL IN NON-HASH DEDUPLICATION SYSTEM

M.A. Zhukov[a], D.B. Afanasiev[a]

[a] ITMO University, Saint Petersburg, 197101, Russian Federation
Corresponding author: fenderst11@gmail.com

Article info
Received 17.12.14, accepted 16.02.15
doi:10.17586/2226-1494-2015-15-3-470-475
Article in Russian

For citation: Zhukov M.A., Afanasiev D.B. Analysis and estimation of the trie minimum level in non-hash deduplication system. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2015, vol.15, no. 3, pp. 470–475.

Abstract

Subject of research. The paper deals with a method of restriction for the trie minimum level in non-hash data deduplication system.

Method. The subject matter of the method lies in forcibly completing the trie to a specific minimum level. The proposed method makes it possible to increase performance of the process by reducing the number of collisions at the lower levels of the trie. The maximum theoretical performance growth corresponds to the share of collisions in the total number of data read operations from the storage medium. Proposed method application increases the metadata size to the amount of new structures containing one element.

Main results. The results of the work have been proved by the data of computational experiment with non-hash deduplication on 528 GB data set. The process analysis has shown that 99% of the execution time is taken to head positioning of hard-drives. The reason is a random distribution of the blocks on the storage medium. Application of the method of minimum level restriction for the trie in non-hash data deduplication system on the experimental data set gives the possibility to increase performance maximum by 16% and the increase of metadata size is 49%. The total amount of metadata is 34% less than with hash-based deduplication using the MD5 algorithm, and is 17% less than using Tiger192 algorithm. These results confirm the effectiveness of the proposed method.
Practical relevance. The proposed method increases the performance of deduplication process by reducing the number of collisions in the trie construction. The results are of practical importance for professionals involved in the development of non-hash data deduplication methods.

Keywords
non-hash-based deduplication, deduplication performance increase, trie, prefix tree, I/O operations, metadata size, non-hash deduplication experiment.

Введение

Дедупликация – это технология, ориентированная на исключение избыточности в наборах данных путем замены повторяющихся данных ссылками на уже существующие данные, обеспечивая сокращение хранимой на носителе информации [1, 2]. Традиционные методы дедупликации [2–7] и их реализации [1, 8–12] используют для верификации данных значения результатов хеш-функций блоков данных [1, 2]. Этим методам присущи две проблемы:

- высокие требования к объему оперативной памяти для хранения метаданных;
- возможность совпадений результатов хеш-функции различных блоков, называемых хеш-коллизиями.

Из-за возможности возникновения хеш-коллизий требуется прямая сверка очередного блока данных с блоком данных на носителе, что также имеет прямое влияние на производительность.

Бесхешевая дедупликация данных практически не рассмотрена в литературе [2, 8]. Был предложен метод бесхешевой дедупликации с верификацией блоков в структуре данных префиксного дерева [13, 14].

Принцип работы системы бесхешевой дедупликации с использованием верификации в структуре префиксного дерева [15–17] описан ниже.

При поиске по префиксному дереву осуществляется последовательный перебор префиксов проверяемого блока данных. Поиск по дереву осуществляется до обнаружения ссылки на блок данных на носителе или отсутствия ссылки на очередной уровень. В последнем случае в соответствующую ветвь дерева добавляется ссылка на проверяемый блок. При обнаружении ссылки на блок данных будет произведено чтение этого блока и сравнение с проверяемым блоком данных. В случае их совпадения в логическом представлении данных будет записана ссылка на уже существующий блок по данному адресу. В противном случае будет выполнена операция записи блока в хранилище и созданы новые уровни в префиксном дереве до уровня первого расхождения префикса в этих блоках данных. Последняя структура новой ветви будет содержать две ссылки в хранилище по соответствующим определенным префиксам.

В случае обнаружения ссылки на блок в префиксном дереве производится прямая сверка блока. В случае совпадения блока данный блок считается дедуплицированным. Проверка таких блоков также присуща методам хешевой дедупликации. В случае несовпадения блока данная ситуация именуется коллизией и присуща только предложенному методу префиксной дедупликации.

С ростом объема дедуплицируемых данных методом бесхешевой дедупликации с использованием префиксного дерева скорость процесса падает по причине увеличения количества коллизий и идентичных блоков. Таким образом, узким местом становится система ввода-вывода, загруженная в процессе построения префиксного дерева. По результату анализа производимых операций в ходе процесса бесхешевой дедупликации было выявлено, что около 99% времени тратится на ожидание позиционирования головок жестких дисков при операциях ввода-вывода, так как проверяемые блоки распределены на носителе в случайном порядке.

Целью работы является разработка и оценка метода увеличения производительности процесса рассматриваемой бесхешевой дедупликации данных путем сокращения количества коллизий. Предложен метод сокращения количества коллизий ограничением минимального уровня построения дерева.

Ввиду того, что большинство ссылок на блоки данных находятся на начальных уровнях дерева, коллизии в процессе увеличения объема данных также преобладают на начальных уровнях. Сокращение количества коллизий позволит увеличить производительность процесса бесхешевой дедупликации. Были поставлены задачи определения количества коллизий, количества ссылок на блоки в префиксном дереве и динамики изменения количества ссылок на блоки в префиксном дереве по мере роста объема обработанных данных.

Описание эксперимента

Для решения поставленных задач был произведен эксперимент бесхешевой дедупликации данных. Исходным набором данных являлась резервная копия систем инфраструктуры реального предприятия.

Программа бесхешевой дедупликации была запущена на выполнение на этом наборе данных с размером блока 512 Б со сбором статистической информации с шагом 16 ГБ дедуплицированных данных. Эксперимент был прекращен при достижении 528 ГБ дедуплицированных данных. Информация включала в себя количество структур каждого типа, используемых в реализации, количество цепочек в дереве разной длины, количество значений ссылок на блок данных на каждом уровне дерева, время выполнения процесса, количество коллизий и количество дедуплицированных блоков.

Тип структуры	Количество ссылок	Размер структуры, Б	Количество структур	Занимаемый объем, МБ
0	1	12	415 876 931	4 759
1	2	20	120 759 285	2 303
2	4	40	15 811 187	603
3	8	76	2 130 836	154
4	16	148	651 341	91
5	32	292	304 402	84
6	64	580	15 016 086	8 305
7	128	1156	1 819 538	2 005
8	256	2052	121 017	236

Таблица. Типы используемых структур и занимаемый ими объем

В реализации бесхешевой дедупликации использовались фиксированные размеры структур разного типа, содержащие в себе количество ссылок, равное степени 2, т.е. от 0 до 8. Был использован префикс размером 1 Б. Типы используемых структур и итоговое количество структур каждого типа, полученных в результате проведения эксперимента, описаны в таблице. Для осуществления процесса дедупликации программа использовала один поток.

Эксперимент проводился на системе со следующими характеристиками:
процессор – Intel Xeon E5-2643 10M Cache, 3,3 ГГц;
операционная система – Oracle Solaris 10u11 x64;
объем памяти – 64 ГБ;
частота памяти – 1600 МГц;
дисковая подсистема – RAID нулевого уровня из 4 жестких дисков Seagate ST1000DM003 1TB 7200rpm.

Анализ полученных данных

В ходе эксперимента была собрана статистическая информация о количестве ссылок на каждом уровне дерева с шагом в 16 ГБ дедуплицированных данных. В конце эксперимента на 528 ГБ был получен коэффициент дедупликации, равный 1,69, что соответствует 896 ГБ исходных данных. Изменение скорости процесса при увеличении объема дедуплицированных данных наглядно отображено на рис. 1.

Рис. 1. Изменение скорости процесса дедупликации с увеличением объема обработанных данных

На наборе данных основная часть (72,8%) всех ссылок на блоки хранилища располагалась на третьем уровне дерева. На первых уровнях дерева – с 0 по 8 – содержится 96,73% всех ссылок на блоки. Изменения количества ссылок на блоки хранилища для первых восьми уровней по мере роста объема обработанных данных показаны на графике на рис. 2. Снижение количества ссылок на втором уровне до нуля на 192 ГБ дедуплицированных данных свидетельствует о возникновении как минимум 5642398 коллизий в ходе работы системы.

Как было показано ранее, операции чтения имеют место в случаях возникновения коллизий и в случаях проверки существующего блока. Наглядно изменение соотношения количества коллизий и деду-

плицированных блоков от общего количества блоков отображено на рис. 3. На 528 ГБ дедуплицированных данных коллизии составляют 8% от общего числа прочитанных блоков, а проверка повторяющихся блоков – 41,1%. Таким образом, для последовательного чтения очередных 16 ГБ исходных данных требуется около 36 с, при этом необходимо проверить около 1,8 млн блоков (суммарное количество коллизий и повторяющихся блоков), для чего, если не учитывать кеширование операционной системы, потребуется около 155000 с. Таким образом, основную часть времени занимает проверка коллизий и повторяющихся блоков.

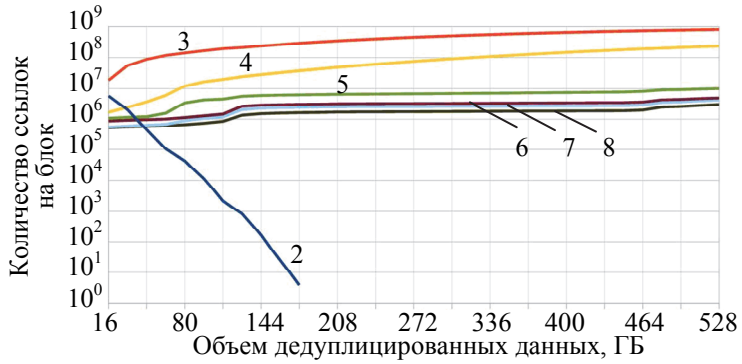

Рис. 2. Изменение количества ссылок на блоки хранилища на 2–8 уровнях дерева

Следует отметить, что при использовании традиционных методов дедупликации на основе хеширования также требуется проверять повторяющиеся блоки ввиду существования хеш-коллизий.

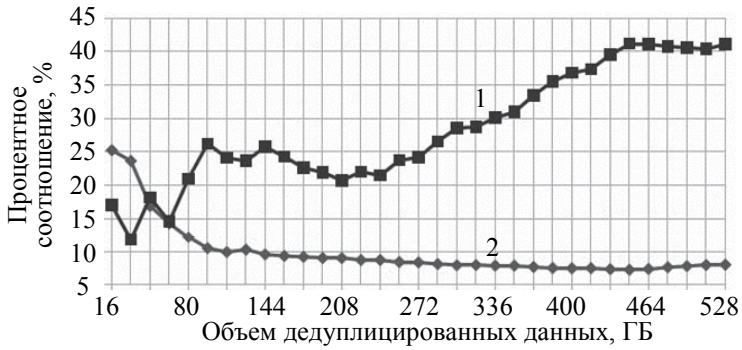

Рис. 3. Изменение соотношения количества коллизий и дедуплицированных данных

Оценить максимально возможный прирост производительности от сокращения количества коллизий можно, оценив соотношение количества коллизий от всех операций чтения с жесткого диска. На наборе данных максимальный теоретический прирост производительности может составить около 16%. Дисперсия количества ссылок на блоки данных по мере роста дедуплицированных данных на разных уровнях префиксного дерева (рис. 4) наглядно отображает преобладающий разброс количества ссылок на начальных уровнях, что свидетельствует о большом количестве коллизий на начальных уровнях, ведущем к снижению производительности.

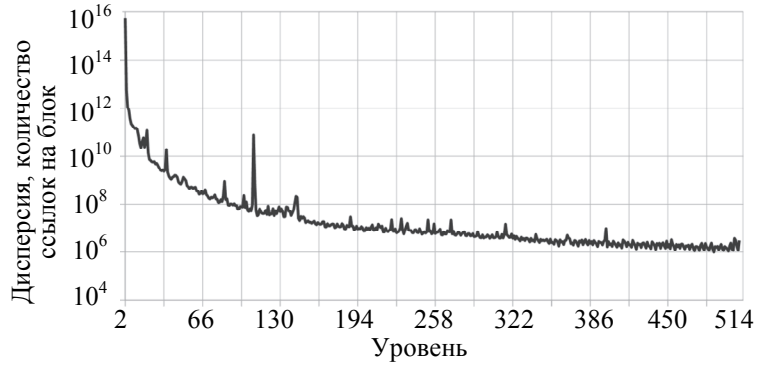

Рис. 4. Дисперсия количества ссылок на блок на разных уровнях

Оценка ограничения минимального уровня дерева

Построение дерева с определенного минимального уровня позволит избежать части коллизий, возникающих на нижних уровнях префиксного дерева, снизив тем самым нагрузку на систему ввода-вывода в ущерб потребляемому объему оперативной памяти для хранения метаданных, необходимой для работы системы дедупликации. В частности, перенесение на один уровень в глубину порождает дополнительную структуру типа 0 (один элемент). Объем, на который увеличится объем метаданных при задании определенного минимального уровня, может быть оценен формулой определения увеличения объема метаданных при задании определенного минимального уровня построения дерева:

$$D(l) = h_0 \times \sum_{i=0}^{l} N_i \times (l - i),$$

где h_0 – размер структуры, содержащей один элемент; l – заданный минимальный уровень построения дерева; N_i – количество элементов с ссылкой на блок данных на i-ом уровне.

Применяя данную формулу к полученной информации о количестве ссылок на блок на 528 ГБ дедуплицированных данных, при минимальном уровне построения, равным 4, получим, что системе бесхешевой дедупликации потребуется дополнительно 9,2 ГБ памяти для хранения метаданных, что составляет 49,7% исходного объема метаданных размером 18,5 ГБ. Суммарный объем метаданных в этом случае составит 27,7 ГБ, что на 34% меньше, чем объем метаданных на этом же наборе данных при хешевом подходе с использованием алгоритма хеширования MD5 [16, 18] (из расчета 32 Б – хеш-сумма, 8 Б – ссылка на блок) и на 17% меньше, чем при хешевом подходе с использованием алгоритма Tiger192 [16] (из расчета 24 Б – хеш-сумма, 8 Б – ссылка на блок).

Использование на наборе данных в качестве минимального уровня построения дерева значения 3 не увеличит общий объем метаданных, сократив как минимум 5,5 млн коллизий (величина, равная максимальному количеству ссылок на блок на предыдущих уровнях). Сокращение коллизий на этом наборе данных позволит сократить время процесса максимум на 16,3% (процент коллизий от общего числа операций чтения блока).

Заключение

В работе предложен метод ограничения минимального уровня дерева в системе бесхешевой дедупликации данных, позволяющий увеличить производительность процесса за счет сокращения количества коллизий при построении префиксного дерева. Произведена оценка предложенного метода. Определено фактическое и максимально возможное увеличение производительности при использовании ограничения минимального уровня дерева на экспериментальном наборе данных. Произведено сравнение объем,а требуемого метаданными системы бесхешевой дедупликации, с объемом метаданных при хешевой дедупликации. Полученные результаты подтверждают эффективность использования ограничения минимального уровня. Предложенный метод будет использован в разрабатываемой системе бесхешевой дедупликации данных. Сформирован задел для дальнейших исследований по заблаговременному определению ограничения минимального уровня, экспериментальному подтверждению эффективности предложенного метода.

Литература

1. Щербинин А. Решения по дедупликации данных // Storage News. 2008. № 2 (35). С. 2–7.
2. Черняк Л. Просто о сложностях дедупликации // Открытые системы. СУБД. 2013. № 3. С. 54–55.
3. Meyer D.T, Bolosky W.J. A study of practical deduplication // ACM Transactions on Storage. 2012. V. 7. N 4. Art. 14. doi: 10.1145/2078861.2078864
4. Meister D. Advanced Data Deduplication Techniques and their Application, PhD dissertation. Mainz, Johannes Gutenberg University, 2013. 230 p.
5. Bhagwat D., Eshghi K., Long D.D.E., Lillibridge M. Extreme binning: scalable, parallel deduplication for chunk-based file backup // Proc. 17th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). London, UK, 2009. P. 237–245. doi: 10.1109/MASCOT.2009.5366623
6. Казаков В.Г., Федосин С.А., Плотникова Н.П. Способ адаптивной дедупликации с применением многоуровневого индекса размещения копируемых блоков данных // Фундаментальные исследования. 2013. № 8–6. С. 1322–1325.
7. Maddodi S., Attigeri G.V., Karunakar A.K. Data deduplication techniques and analysis // Proc. 3rd Int. Conf. on Emerging Trends in Engineering and Technology (ICETET 2010). Goa, India, 2010. P. 664–668. doi: 10.1109/ICETET.2010.42
8. Orlando K., Bautista M.M., Mejia J.R.M., Langnor R.G. IBM ProtecTIER Implementation and Best Practices Guide. IBM Redbooks, 2014. 578 p.
9. Biplob D., Sudipta S., Jin L. ChunkStash: speeding up inline storage deduplication using flash memory // Proc. 2010 USENIX Annual Technical Conference. 2010. P. 16.

10. Osuna A., Balogh E., Galante de Carvalho R.A., Javier R.F., Mann Z. Implementing IBM Storage Data Deduplication Solutions. IBM Redbooks, 2011. 322 p.
11. Sheu R.-K., Yuan S.-M., Lo W.-T., Ku C.-I. Design and implementation of file deduplication framework on HDFS // International Journal of Distributed Sensor Networks. 2014. V. 2014. Art. 561340. doi: 10.1155/2014/561340
12. Srinivasan K., Bisson T., Goodson G., Voruganti K. iDedup: latency-aware, inline data deduplication for primary storage // Proc. 10th USENIX Conference on File and Storage Technologies (FAST 2012). San Jose, USA, 2012. P. 299–312.
13. Кнут Д.Э. Искусство программирования. Т. 3. Сортировка и поиск. 2 изд. М.: Вильямс, 2001. 824 с.
14. Ахо А.В., Хопкрофт Д.Э., Ульман Д.Д. Структуры данных и алгоритмы. М.: Вильямс, 2000. 382 с.
15. Жуков М.А., Афанасьев Д.Б. Верификация блоков данных в системе бесхешевой дедупликации // Сб. тезисов докладов II конгресса молодых ученых. Вып. 1. СПб.: НИУ ИТМО, 2013. С. 78.
16. Жуков М.А., Афанасьев Д.Б. Хранение метаданных блоков в структуре данных префиксного дерева // Сборник трудов молодых ученых и сотрудников кафедры ВТ. Вып. 5. СПб.: НИУ ИТМО, 2014. С. 12–15.
17. Жуков М.А., Афанасьев Д.Б. Порядок построения префиксного дерева в системе бесхешевой дедупликации // Труды XXI Всероссийской научно-методической конференции Телематика'2014. Санкт-Петербург, 2014. С. 89.
18. Жуков М.А. Настройка параметров дедупликации // Сборник тезисов докладов конгресса молодых ученых. Вып. 1. Санкт-Петербург, 2012. С. 58.

Жуков Максим Андреевич – студент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, fenderst11@gmail.com

Афанасьев Дмитрий Борисович – ассистент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, dima@cs.ifmo.ru

Maxim A. Zhukov – student, ITMO University, Saint Petersburg, 197101, Russian Federation, fenderst11@gmail.com

Dmitry B. Afanasiev – assistant, ITMO University, Saint Petersburg, 197101, Russian Federation, dima@cs.ifmo.ru