ПАРАЛЛЕЛЬНАЯ ОБРАБОТКА ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ MAPREDUCE И HADOOP

А.В. Созыкин[1,2], Д.А. Усталов[1,2], М.А. Черноскутов[1,2]

[1] *Институт математики и механики УрО РАН*
[2] *Уральский федеральный университет*

Введение. В настоящее время задачи обработки больших объёмов изображений все чаще встречаются в научных проектах, анализе Интернет, медицинских исследованиях и т. п. Объёмы современных коллекций изображений таковы, что их невозможно обработать на одном компьютере за приемлемое время. Для распараллеливания обработки изображений хорошо подходит модель MapReduce [1], т. к. во многих задачах изображения (или группы изображений) обрабатываются независимо.

Однако применение MapReduce для обработки изображений затруднено тем, что самая популярная реализация этой технологии — Apache Hadoop [2], ориентирована преимущественно на обработку текстов и не содержит средств для работы с изображениями.

В статье представлена система MIPR (MapReduce Image Processing), обеспечивающая возможность обработки изображений в Hadoop.

Поток данных в Hadoop. В модели MapReduce прикладной программист разрабатывает только функции Map и Reduce, все остальное обеспечивает среда выполнения. Схема потока данных среды выполнения Hadoop показана на рис. 1. Сначала необходимо прочитать данные из распределенной файловой системы HDFS [3]. Для этого Hadoop использует InputFormat и RecordReader. InputFormat определяет формат входных данных и отвечает за разделение данных на части, каждая из которых обрабатывается отдельным процессом Map. RecordReader обеспечивает чтение данных из файлов и разбиение их на записи, состоящие из пар «ключ-значение».

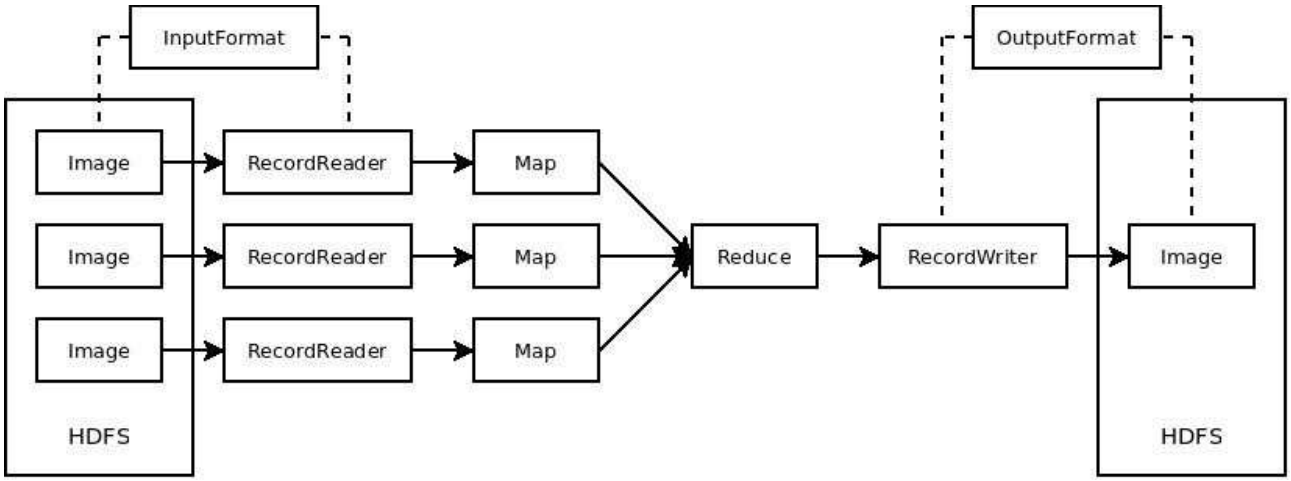

Рис. 1. Схема потока данных среды выполнения Hadoop

Затем данные должны быть представлены во внутреннем формате Hadoop, пригодном для сериализации. Hadoop преобразует данные в поток байт для передачи по сети или временного хранения на диске. Наиболее популярная технология сериализации — Hadoop Writable.

После окончания обработки MapReduce, данные должны быть записаны в HDFS. Для записи данных Hadoop использует OutputFormat и RecordWriter. OutputFormat определяет тип и структуру выходных файлов, а RecordWriter обеспечивает запись пар «ключ-значение» в эти файлы.

Таким образом, чтобы встроить обработку изображений в поток данных среды выполнения Hadoop, необходимо разработать:

- Средства для чтения изображений из HDFS (InputFormat, RecordReader).
- Внутреннее представление изображений в Hadoop с поддержкой сериализации.
- Средства для записи изображений в HDFS (OutputFormat, RecordWriter).

Использование Hadoop и MapReduce для обработки изображений. Hadoop использовался для обработки изображений в нескольких научных проектах [4-9]. При этом в большинстве проектов разработчики самостоятельно реализовывали работу с изображения в Hadoop.

Среда выполнения Hadoop устроена сложно, расширять её тяжело, поэтому многие разработчики пытались применять для обработки изображений стандартные возможности Hadoop. В большинстве случаев с изображениями работали как с массивом байт. В каждой функции Map и Reduce массив байт конвертировался в изображение, выполнялась обработки, а затем обратная конвертация в массив байт. Такой подход не очень удобен, но как правило эффективен. Встречаются и менее эффективные и удобные способы использования Hadoop для обработки изображений. Например, в работе [8] авторы преобразуют снимки земной поверхности со спутника в текстовый формат, где каждая строка представляет один пиксел изображения и содержит коды цветов в текстовом виде.

Существуют две готовые системы, которые можно применить для обработки изображений в Hadoop: HIPI (Hadoop Image Processing Interface) [10] и OpenIMAJ (Open Intelligent Multimedia Analysis for Java) [11]. HIPI создана специально для обеспечения возможности обработки изображений в Hadoop. OpenIMAJ - это система анализа изображений и видео на Java, часть функциональности которой реализована с поддержкой Hadoop.

Система HIPI включает реализацию полного потока данных для обработки изображений в Hadoop: InputFormat и OtputFormat, соответствующие им RecordReader и RecordWriter, а также внутреннее представление изображения в Hadoop. В качестве последнего используется FloatImage на основе простого массива. Для сериализации FloatImage реализует интерфейс Writable. Также FloatImage включает небольшой набор готовых операций, которые можно выполнять с изображением: преобразование цветного изображения в градации серого и наоборот, изменение размера, обрезка и т. п. В качестве входных данных система HIPI использует HipiImageBundle — файл специально разработанного формата, в котором объединены несколько изображений для повышения эффективности их хранения в HDFS, а также производительности обработки в MapReduce. HipiImageBundle, кроме самих изображений, включает дескрипторы, по которым можно фильтровать изображения при обработке в Hadoop, а также индекс для быстрого поиска нужных изображений. HipiImageBundle обеспечивает более высокую производительность обработки изображений в MapReduce, по сравнению со стандартными средствами Hadoop. Недостатками HIPI является низкая функциональность по обработке изображений: прикладному программисту предоставляется простой массив с пикселами, всю обработку которого необходимо реализовывать самостоятельно. Также к недостаткам можно отнести использование не стандартного формата хранения изображений HipiImageBundle, что затрудняет обработку изображений, если она должна состоять из нескольких частей, выполняемых различным программным обеспечением.

Система OpenIMAJ предоставляет больше возможностей по обработке и анализу изображений, чем HIPI. OpenIMAJ включает реализацию алгоритмов компьютерного зрения (поиск лиц, выделение SIFT дескрипторов, характерных областей и др.), кластеризации, классификации и т. п. Часть из этих алгоритмов реализована с возможностью использования Hadoop. В качестве входных данных используется стандартный для Hadoop формат Sequence файл. В отличие от HIPI, OpenIMAJ не включает средства чтения и записи изображений из HDFS и встроенного сериализуемого представления изображений в Hadoop. Вместо этого OpenIMAJ представляет изображение в виде массива байт, который требует конвертации в пригодный для обработки вид. Недостатком OpenIMAJ является то, что эта система предоставляет только набор готовых обработчиков изображений с использованием Hadoop, а не средства разработки обработчиков изображений.

Система MIPR реализует предложенную ранее архитектуру обработки больших объёмов изображений с использованием MapReduce и Hadoop [12]. Основные компоненты MIPR:

- Средства для ввода/вывода и внутреннего представления изображений в Hadoop.
- Программный интерфейс обработки изображений в Hadoop.
- Библиотека готовых обработчиков изображений.

MIPR обеспечивает возможность использования *представления изображений* в трёх форматах:

- Стандартный для Java формат BufferedImage.
- Форматы изображений OpenIMAJ: MBFImage (цветное) и FImage (оттенки серого).
- Массив пикселов.

В качестве технологии сериализации используется Hadoop Writable. Диаграмма классов внутренних представлений изображений в MIPR показана на рис. 2.

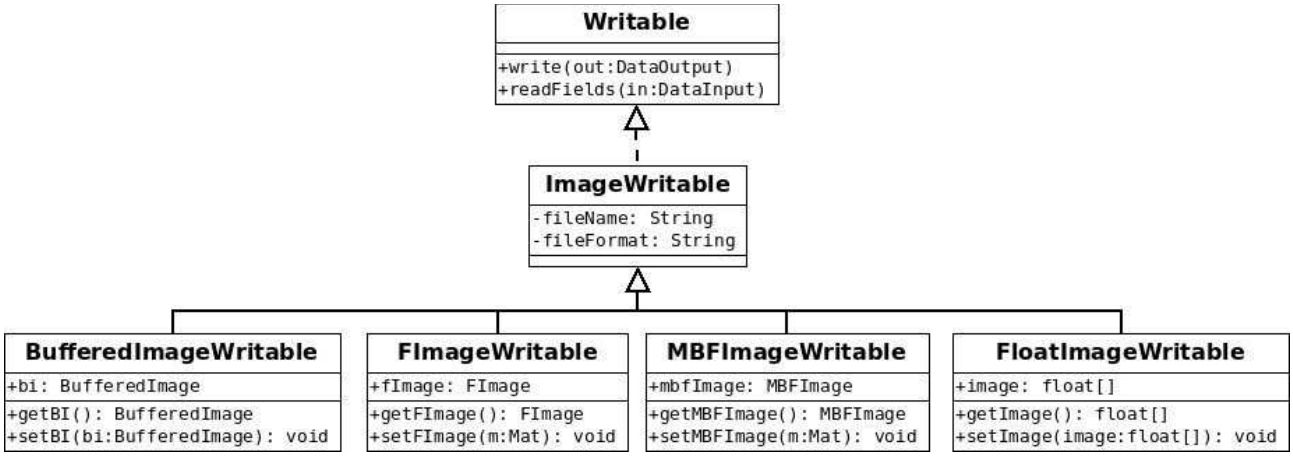

Рис. 2. Диаграмма классов представления изображений в системе MIPR

Средства ввода-вывода изображений в MIPR включают InputFormat, RecordReader, OutputFormat и RecordWriter для каждого формата изображений MIPR (см. рис. 2.). Логика работы средств ввода-вывода для всех форматов одинаковая. Изображения читаются из файла целиком и передаются в функцию Map. Записываются изображения в HDFS в виде отдельных файлов или упакованными в Sequence или Map файлы.

Для обработки изображений в Hadoop достаточно иметь форматы представления изображений, реализующих интерфейс Writable, и соответствующие им средства ввода-вывода. Но это низкоуровневые возможности, использование которых требует знания внутреннего устройства Hadoop и технологии MapReduce.

Для прикладных программистов, являющихся специалистами в обработке и анализе изображений, был предложен *программный интерфейс обработки изображений в Hadoop.* Интерфейс изолирует от прикладного программиста детали внутреннего устройства Hadoop и позволяет создавать функции, обрабатывающие только одно изображение (или группу связанных изображений), которое уже загружено в память. Все остальное берет на себя система MIPR, которая предоставляет:

- Набор MapReduce драйверов для запуска задач обработки изображений в разных форматах. Драйвер устанавливает нужный формат изображения, соответствующие ему форматы ввода-вывод и Map-класс. В настоящее время поддерживаются только задачи без Reduce.
- Реализации классов Map для различных форматов изображений. Классы извлекают изображение из внутреннего представления Hadoop и вызывают пользовательскую функцию обработчика изображения. Результирующее изображение преобразуется во внутренне представление Hadoop. Каждый процесс Map выдаёт одно изображение.
- Средства для упаковки отдельных изображений в Sequence файлы и средства извлечения изображений из Sequence файлов.

Библиотека готовых обработчиков изображений включает в себя реализации часто используемых операций на основе Hadoop, таких как преобразование форматов и размеров изображений, извлечение SIFT-дескрипторов, обнаружение лиц и т.п.

Особенности реализации системы MIPR. Основная проблема, с которой столкнулись при реализации MIPR — это плохая работа Hadoop с большим количеством файлов маленького размера. Проблема вызваны следующими причинами:

- Размер изображения намного меньше блока данных HDFS (по умолчанию 64 МБ). Для загрузки каждого изображения выполняется отдельная операция чтения, в результате которой загружается небольшой объем данных, гораздо меньше чем размер блока. Скорость чтения низкая.
- Для каждого файла с изображением хранится отдельная запись в памяти Hadoop NameNode. Когда изображений много, то NameNode требуется много памяти и поиск в таблице файлов занимает много времени.

Существуют разные подходы к решению проблемы маленьких файлов. Hadoop предоставляет штатные средства, позволяющие объединить несколько маленьких файлов в один большой: архивы Hadoop, Sequence файлы и Map файлы. В системе HIPI для этих целей разработан собственный формат данных HipiImageBundle. Система OpenIMAJ использует Sequence файлы. MIPR также обеспечивает возможность использования Sequence файлов.

Упаковка изображений в один большой файл не всегда является удобной. Как правило, обработка изображений состоит из нескольких этапов, и некоторые этапы требуют на входе или выдают на выход отдельные изображения, а не один упакованный файл. Для таких ситуаций в MIPR использован подход на основе CombineFileInputFromat. Это специальный формат чтения входных данных в Hadoop, позволяющий читать несколько небольших файлов на диске за одну операцию, что значительно увеличивает скорость чтения.

Специализированные InputFormat на основе CombineFileInputFromat были разработаны для каждого формата изображений MIPR (см. рис. 2).

Использование CombineFileInputFromat не полностью решает проблему маленьких файлов. По прежнему остаётся большое числа записей в NameNode, что отрицательно влияет на производительность. Поэтому желательно использовать Sequence файлы всегда, когда есть такая возможность.

Эксперименты. Для оценки работы системы MIPR был проведён ряд экспериментов. В качестве набора данных использовалась коллекция MIRFLICKR-1M [13], содержащая 1 миллион изображений, загруженных с Flickr, общим объёмом 118 ГБ. Целью экспериментов было оценить масштабируемость системы MIPR и эффективность работы с большим количеством маленьких изображений. Эксперименты проводились на кластере Hadoop, состоящем из шести узлов (пять вычислительных и один управляющий). Параметры узлов кластера: операционная система Linux CentOS 6.5, процессоры 2xAMD Opteron 2218, 8 ГБ оперативной памяти, диск 500 ГБ, Hadoop 2.2.

Для оценки масштабируемости системы MIPR были реализованы несколько операций, описанные в табл. 1. Во всех экспериментах изображения хранились в Sequence файлах.

Таблица 1. Операции с изображениями в экспериментах с MIPR

№	Операция	Представление изображения	Библиотека
1	Преобразование формата из Jpeg в PNG	BufferedImageWritable	Java 2D
2	Выделение границ фильтром Собеля	BufferedImageWritable	Java 2D
3	Извлечение SIFT-дескрипторов	MBFImageWtitable	OpenIMAJ
4	Обнаружение лиц	MBFImageWtitable	OpenIMAJ

На рис. 3. представлена зависимость времени обработки изображений от количества узлов в кластере. Эксперименты выполнялись над всей коллекцией данных MIRFLICKR-1M.

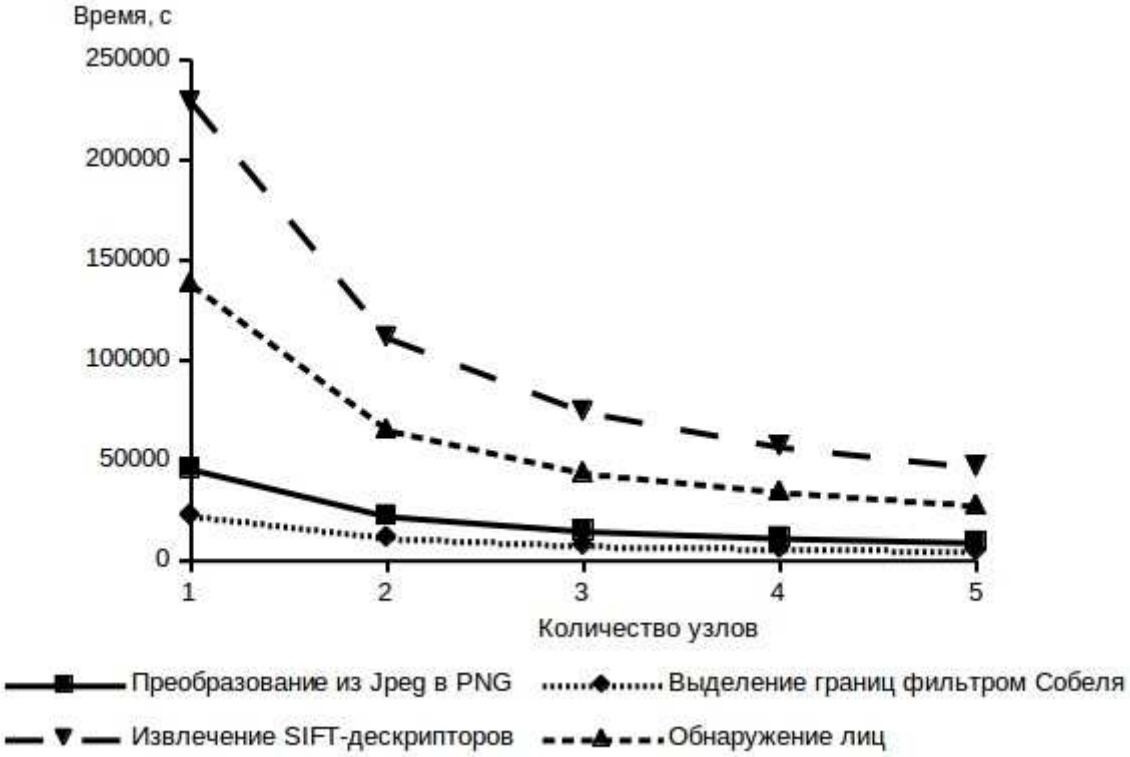

Рис.3. Зависимость времени выполнения обработки изображения от количества узлов в кластере

На рис. 4. представлен график ускорения обработки изображений в зависимости от количества узлов в кластере.

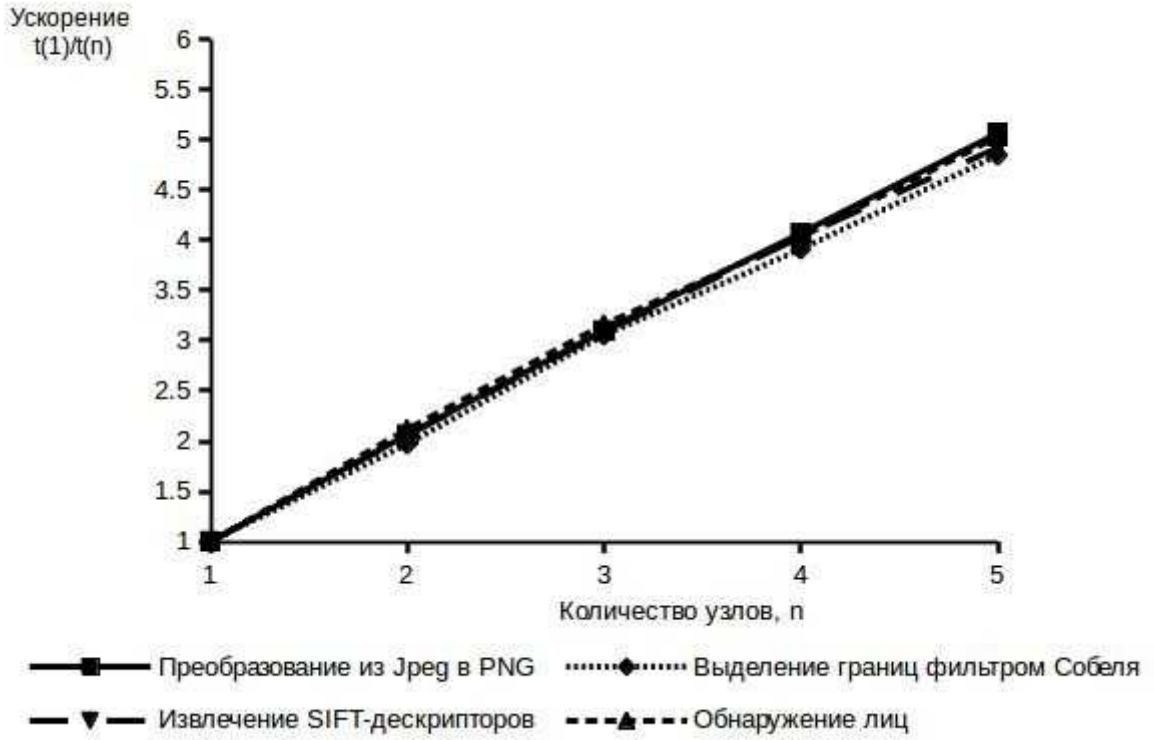

Рис.4. Ускорение обработки изображений в зависимости от количетсва узлов кластера, где t(1) — время выполнения на одном узле, t(n) — время выполнения на n узлах

На рис. 5. представлена зависимость времени выполнения операций по обработке изображений от объёма коллекции (логарифмическая шкала). Эксперименты выполнялись на пяти узлах кластера.

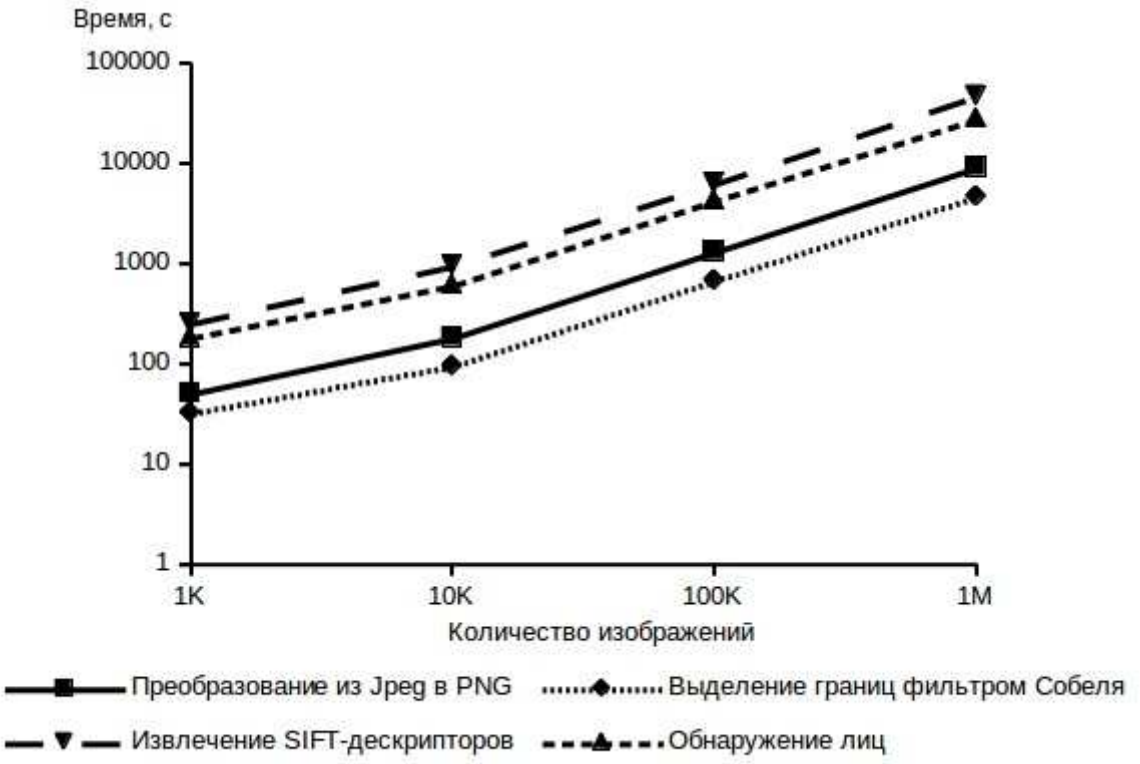

Рис.5. Зависимость времени выполнения операций по обработке изображения от размера коллекции

На основе анализа рис. 3-5 можно сделать вывод, что система MIPR обладает почти линейной масштабируемостью как по количеству узлов в кластере, так и по объёму обрабатываемых данных. Это ожидаемый результат, т. к. изображения в коллекции обрабатываются независимо.

Для оценки эффективности работы MIPR с большим количеством маленьких файлов был выполнен эксперимент по преобразованию коллекции MIRFLICKR-1M, содержащей цветные изображения, в формат оттенков серого. Данная операция выбрана потому, что разработчики HIPI использовали её для оценки производительности работы HipiImageBundle. Результаты экспериментов приведены в табл.2.

Таблица 2. Преобразование цветных изображений в оттенки серого

№	Система	Формат изображений	Время, с
1	HIPI	HipiImageBundle	7 350
2	OpenIMAJ	Sequence файл	9 230
3	MIPR	Маленькие файлы	98 700
4	MIPR, CombineFileInputFromat	Маленькие файлы	10 800
5	MIPR	Sequence файл	9 120

Время, приведённое в табл.2, не учитывает время создания Sequence файлов и HipiImageBundle. Следует отметить, что в исходном тесте производительности HIPI не выполнялась запись изображений в оттенках серого в HDFS, т. к. она в HIPI не реализована. Чтобы получить результаты пригодные для сравнения, пришлось модифицировать код теста HIPI и добавить запись в HDFS исходного цветного изображения.

По данным табл.2 видно, что применение CombineFileInputFromat позволило сократить время обработки маленьких изображений почти в 10 раз. Применение Sequence файлов позволяет получить ещё больший выигрыш в производительности. Причём эффект от применения Sequence файлов примерно одинаковый как в системе MIPR, так и в OpenIMAJ. Эффективнее всего работает система HIPI за счёт специализированного формата представления изображений HipiImageBundle.

Сравнение MIPR с аналогами. Система MIPR обладает следующими преимуществами по сравнения с аналогами MIPR и OpenIMAJ:

- Широкий набор форматов внутреннего представления изображений в Hadoop. Это позволяет прикладному программисту использовать возможности различных библиотек для обработки изображений.

- Поддержка обработки изображений как в одном большом файле, так и в большом количестве маленьких файлов. Разница в производительности при использовании CombineFileInputFromat составляет 15%. HIPI может обрабатывать файлы только в формате HipiImageBundle, а OpenIMAJ — только в Sequence файлах.
- Предоставление средств для ввода/вывода и внутреннего представления изображений в Hadoop, программного интерфейса обработки изображений в Hadoop и библиотеки готовых обработчиков изображений. Система HIPI включает только первые два компонента, а OpenIMAJ — только третий.

Производительность систем можно оценить на основе данных табл. 2. При использовании Sequence файлов производительность MIPR и OpenIMAJ примерно одинакова. Производительность системы HIPI выше за счёт использования специализированного формата представления файлов HipiImageBundle. Однако данный формат является специфическим для системы HIPI и не может быть прочитан другими системами.

Заключение. В статье представлена система MIPR, предназначенная для автоматизации параллельной обработки изображений с помощью технологии MapReduce. Система расширяет функциональность Hadoop, обеспечивая возможность использования изображений в MapReduce программах. Кроме этого, MIPR предоставляет программный интерфейс обработки изображений, скрывающий от прикладного программиста детали внутреннего устройства Hadoop, и набор готовых к использования параллельных обработчиков изображений.

Тестирование системы MIPR показало её почти линейное масштабирование, как с ростом количества узлов в кластере, так и с ростом объёма обрабатываемых данных.

В качестве направлений дальнейшего развития можно выделить:

- Реализация поддержки в MIPR изображений в формате OpenCV и соответвующих им средств ввода-вывода.
- Реализация программного интерфейса обработки изображений на Python и C++.
- Расширение возможностей библиотеки готовых обработчиков изображений.

Благодарности.

Работа поддержана грантом РФФИ 14-07-31324 мол_а, а также Программой Президиума УрО РАН, проект 12-П-1-1029.

ЛИТЕРАТУРА:

1. J. Dean, S. Ghemawat: MapReduce: simplified data processing on large clusters. // Commun. ACM 51(1), 2008. pp. 107-113.
2. Apache Hadoop [Электронный ресурс]. URL: http://hadoop.apache.org (Дата обращения 29.05.2014).
3. K. Shvachko, H. Kuang, S. Radia, R. Chansler. The Hadoop Distributed File System. // Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST '10). IEEE Computer Society, Washington, DC, USA, pp. 1-10.
4. D. Moise, D. Shestakov, G. Gudmundsson, L. Amsaleg. Indexing and searching 100M images with Map-Reduce // Proceedings of the 3rd ACM conference on International conference on multimedia retrieval, pp. 17–24, ACM, New York (2013).
5. K. Wiley, A. Connolly, S. Krugho, J. Gardner, M. Balazinska, B. Howe, Y. Kwon, Y. Bu. Astronomical Image Processing with Hadoop // Astronomical Data Analysis Software and Systems XX. ASP Conference Proceedings 2011, vol. 442, pp. 93–98.
6. M. H. Almeer. Cloud Hadoop Map Reduce For Remote Sensing Image Analysis. // Journal of Emerging Trends in Computing and Information Sciences, 2012, vol. 3, no. 4, pp. 637–644.
7. A. Cary, Z. Sun, V. Hristidis, N. Rishe. Experiences on Processing Spatial Data with MapReduce // Proceedings of the 21st International Conference on Scientific and Statistical Database Management, 2009, pp. 302–319.
8. Z. Lv, Y. Hu, H. Zhong, J. Wu, B. Li, H. Zhao. Parallel k-means clustering of remote sensing images based on mapreduce // The 2010 International Conference on Web Information Systems and Mining. Springer-Verlag Berlin, Heidelberg, 2010. pp. 162–170.
9. B. White, T. Yeh, J. Lin, L. Davis. Web-scale computer vision using mapreduce for multimedia data mining // Proceedings of the Tenth International Workshop on Multimedia Data Mining, MDMKDD 10, New York, NY, USA, 2010. ACM. pp. 9:1–9:10.
10. S. Chris, L. Liu, A. Sean, L. Jason: HIPI: A hadoop image processing interface for image-based map reduce tasks, B.S. Thesis. University of Virginia, Department of Computer Science, 2011.
11. J. S. Hare, S. Samangooei, D. P. Dupplaw. OpenIMAJ and ImageTerrier: Java libraries and tools for scalable multimedia analysis and indexing of images. // Proceedings of the 19th ACM international conference on Multimedia (MM '11). ACM, New York, NY, USA. pp. 691-694
12. А. В. Созыкин, М. Л. Голдьштейн. Архитектура системы обработки больших объемов изображений с автоматическим распараллеливанием // Научный сервис в сети Интернет: поиск новых решений: Труды

Международной суперкомпьютерной конференции (17-22 сентября 2012 г., г. Новороссийск). — М.: Изд-во МГУ, 2012. — С.58-62.
13. M. Huiskes, B. Thomee, M. Lew: New Trends and Ideas in Visual Concept Detection. // ACM International Conference on Multimedia Information Retrieval (MIR'10) 2010, Philadelphia, USA. ACM, New York, USA. pp. 527–536.