АРХИТЕКТУРА СИСТЕМЫ ОБРАБОТКИ БОЛЬШИХ ОБЪЕМОВ ИЗОБРАЖЕНИЙ С АВТОМАТИЧЕСКИМ РАСПАРАЛЛЕЛИВАНИЕМ

А.В. Созыкин, М.Л. Гольдштейн

Введение. В результате проведения научных исследований, например, таких, как астрономические наблюдения, работа с современным медицинским сканирующим оборудованием, физические эксперименты и мн.др. часто создаются изображения. Эти изображения затем подвергаются компьютерной обработке для дальнейшей интерпретации и получения научных результатов. При проведении исследований на современном мировом уровне формируется большой объем изображений, который может измеряться сотнями и тысячами гигабайт. Эффективная обработка таких объемов возможна только на параллельных вычислительных системах.

Популярные в настоящее время параллельные архитектуры плохо подходят для обработки больших объемов данных, в том числе изображений. Применяемый практически повсеместно подход к разработке параллельных программ на основе передачи сообщений с использованием MPI обладает высокой сложностью и зачастую является избыточным, т.к. изображения, как правило, могут обрабатываться независимо друг от друга. Системы хранения, применяемые в параллельных вычислительных системах, отличаются высокой стоимостью, но при этом часто не обладают достаточной производительностью.

Разработка архитектуры и программных средств для обработки больших объемов изображений на крупномасштабных параллельных вычислительных системах с учетом устранения перечисленных недостатков является актуальной задачей. Основные требования к такой архитектуре – это возможность автоматического распараллеливания последовательной программы и распределенная файловая система для хранения данных на внутренних дисках серверов.

Целью данной работы является создание архитектуры системы обработки изображений больших объемов (СОИБО) с автоматическим распараллеливанием. СОИБО должна предоставлять прикладному программисту простой программный интерфейс, с использованием которого он создает последовательную программу, обрабатывающую одно изображение. Применение этой программы к большому числу изображений в параллельном режиме обеспечивается СОИБО. Для достижения данной цели необходимо выполнить следующие задачи:

- анализ популярных в настоящее время технологий параллельного программирования и выбор технологии, на которой будет строиться СОИБО;
- разработка архитектуры СОИБО на базе выбранной технологии параллельного программирования;
- тестовая реализация предложенной архитектуры СОИБО.

Выбор технологии параллельного программирования. При выборе технологии параллельного программирования необходимо учитывать следующие требования к СОИБО:

- обработка каждого изображения или группы изображений должна происходить независимо друг от друга;
- поддержка распределенной файловой системы;
- автоматическое распараллеливание последовательной программы;
- поддержка работы на кластере из серверов стандартной архитектуры.

Сравнение популярных технологий параллельного программирования с учетом перечисленных выше требований приведено в табл. 1.

Таблица 1. Сравнение технологий параллельного программирования

	MPI	**OpenMP**	**MapReduce**
Автоматическое распараллеливание	Нет	Есть (директивы компилятора)	Есть
Архитектуры оборудования	SMP, кластер, MPP	SMP	SMP, кластер
Распределенная файловая система	Нет	Нет	Есть
Механизмы восстановления в результате отказов оборудования	Нет	Нет	Есть
Алгоритм	Любой	Любой	MapReduce

На основе анализа табл. 1. можно сделать вывод, что для создания СОИБО лучше всего подходит технология MapReduce [1]. MapReduce включает распределенную файловую систему [2], обеспечивает возможность автоматического распараллеливания как на одном многопроцессорном сервере, так и на кластере, предоставляет защиту от сбоев оборудования. Существенным ограничением технологии является поддержка только одного алгоритма MapReduce, ориентированного на параллельное выполнение независимых задач.

Однако изображения или группы изображений могут обрабатываться независимо, поэтому применение MapReduce может быть оправдано. Наиболее популярной реализацией технологии MapReduce является система Apache Hadoop [3].

Применение распределенной файловой системы совместно с Hadoop для хранения изображений обеспечивает ряд преимуществ по сравнению с альтернативными возможностями организации хранения данных, такими как сетевая файловая система (NFS или CIFS) и параллельная файловая система (Lustre или GPFS). Во-первых, распределенная файловая система позволяет хранить данные на внутренних дисках серверов кластера, что обеспечивает возможность увеличения дисковой емкости путем добавления в кластер новых узлов без необходимости установки дорогостоящей и сложной в сопровождении дисковой системы хранения данных. Во-вторых, система запуска задач Hadoop интегрирована с распределенной файловой системой и пытается запускать задачи на тех серверах, где храняться данные для обработки (подход "перемещение вычислений к данным"). При больших объемах данных такой подход существенно повышает производительность обработки. В-третьих, распределенная файловая система обладает высокой пропускной способностью, т.к. чтение и запись могут выполняться через сетевые интерфейсы всех серверов кластера в параллельном режиме. Пропускную способность распределенной файловой системы, как и ее дисковую емкость, можно легко увеличить, добавляя в кластер новые узлы. Дисковые системы хранения, напротив, ограничены по количеству интерфейсов ввода-вывода, и увеличение количества этих интерфейсов, как правило, обходится дорого.

Hadoop широко применяется для решения научных задач обработки изображений, например, для определение положения снимка местности [4], обработки данных астрономических наблюдений [5], результатов дистанционного зондирования [6] и пространсвенных данных [7]. Хотя Hadoop хорошо подходит для создания СОИБО, он обладает следующими недостатками:

- разработка программ для Hadoop требует знания деталей его внутреннего устройства и алгоритма MapReduce;
- в Hadoop отсутвует встроенная поддержка работы с изображениями.

Предлагаемая архитектура СОИБО позволяет преодолеть оба описанных выше недостатка Hadoop путем изоляция от прикладного программиста, занимающегося обработкой изображений, деталей функционирования Hadoop, связанных с параллельностью и загрузкой изображений из распределенной файловой системы. Для этого предоставляется простой API для работы с изображением, которое уже находится в памяти и готово к обработке. Примером такого подхода может служить система HIPI [8].

Архитектура. Схема предлагаемой логической архитектуры СОИБО приведена на рис. 1. В верхней части схемы показаны компоненты Hadoop, используемые в системе, а в нижней части новые компоненты СОИБО.

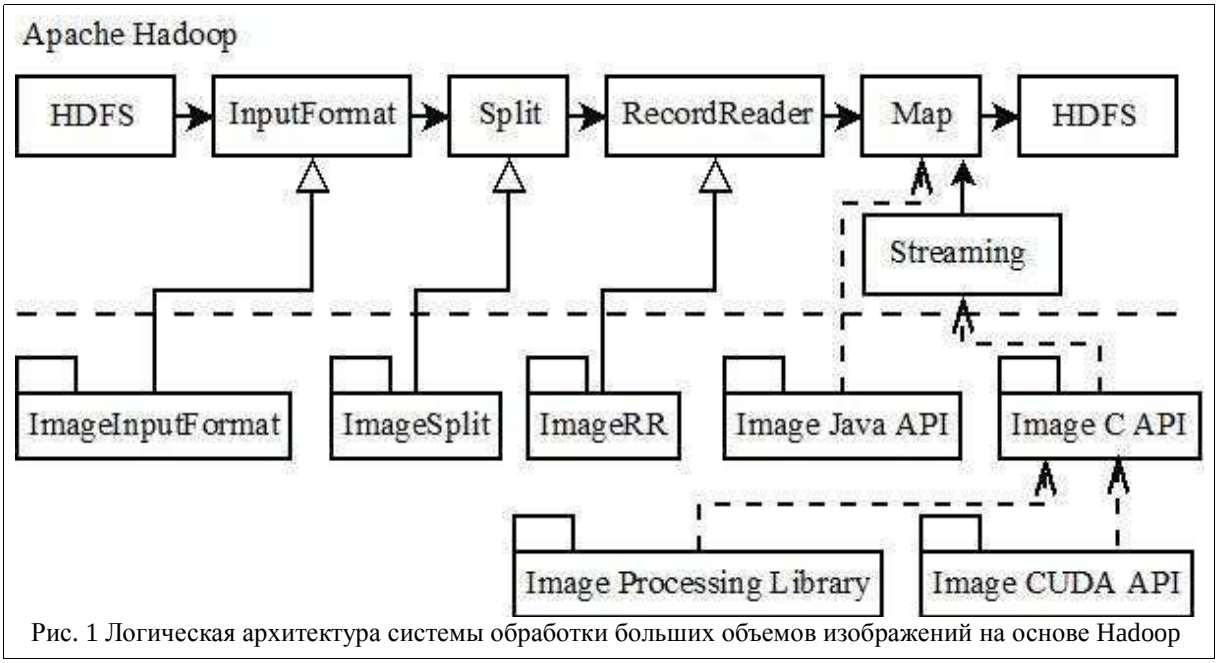

Рис. 1 Логическая архитектура системы обработки больших объемов изображений на основе Hadoop

Изображения, которые нужно обработать, записываются в распределенную файловую систему HDFS. СОИБО содержит пакеты, читающие изображения в различных форматах: ImageInputFormat включает классы, задающие формат чтения изображений, ImageSplit – классы, разбивающее изображение на части, а ImageRR – классы, выполняющее чтение изображений в популярных форматах. Использование ImageSplit не является обязательным и рекомендуется только если размер изображения превышает размер блока HDFS (по-умолчанию 64МБ), в противном случае возможно снижение производительности.

Обработка изображений выполняется в функции Map, результаты записываются в HDFS, функция Reduce не используется. Предоставляется два интерфейса работы с изображениями: Java API и C API. Java API использует стандартные возможности Hadoop, а C API основан на Hadoop Streaming. Java API предоставляет возможность быстрого создания обработчиков изображений, особенно программистам, знакомым с Hadoop, а C API обеспечивает высокую производительность.

На основе C API предлагается создать набор готовых обработчиков изображений, реализующих, например, преобразование Фурье, масочную фильтрацию, изменение размеров и т.п. Готовые обработчики включаются в пакет ImageProcessingLibrary и могут применяться без дополнительного программирования в параллельном режиме. Высокопроизводительная реализация обработчиков возможна с использованием библиотек Intel MKL, Intel Integrated Performance Primitives и AMD Core Math Library.

Пакет ImageCudaAPI позволяет разрабатывать программы для обработки изображений, работающих на графических процессорах производства компании NVIDIA.

Схема архитектуры развертывания СОИБО представлена на рис. 2.

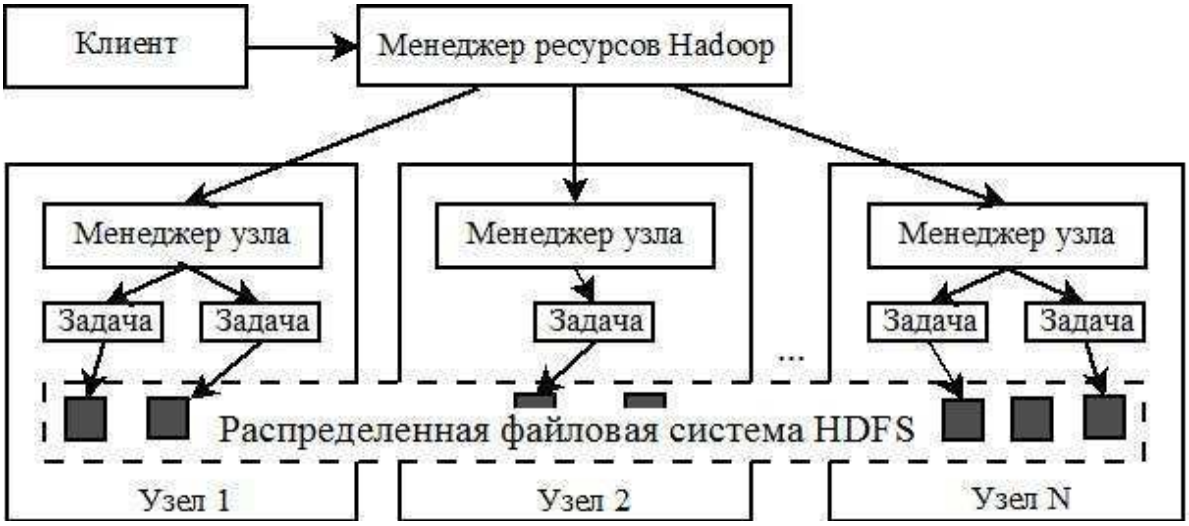

Рис. 2 Архитектура развертывания системы обработки больших объемов изображений на основе Apache Hadoop

СОИБО работает на кластере серверов стандартной архитектуры, управляемом Hadoop. Изображения хранятся на внутренних дисках серверов кластера, логически объединенных в единую распределенную файловую систему HDFS. Запуск задач обработки изображений выполняется с помощью менеджера ресурсов Hadoop, который распределяет задачи по серверам кластера. При этом каждый сервер кластера используется как для хранения изображений, так и для выполнения задач обработки изображений. Менеджер ресурсов Hadoop распределяет задачи таким образом, чтобы они выполнялись на узлах, которые содержат изображения, подлежащие обработке.

Задача представляет собой последовательную программу, обрабатывающую одно изображение (или группу связанных изображений), разработанную с использованием Image C API или Image Java API, или использующую готовый обработчик из пакета Image Processing Library (см. рис. 1). Параллельное применение этой программы к большому количеству изображений выполняется автоматически средствами Hadoop.

Текущая реализация. В настоящее время выполнена первая очередь реализации предложенной архитектуры, включающая ImageInputFormat и ImageRR для формата BMP, а также базовый вариант Image Java API. Выполнено развертывание системы на кластере Apache Hadoop из 4-х узлов Fujitsu-Siemens RX 220, в каждом по 2 процессора AMD Opteron 285, 8 ГБ памяти, жесткий диск 250 ГБ SATA.

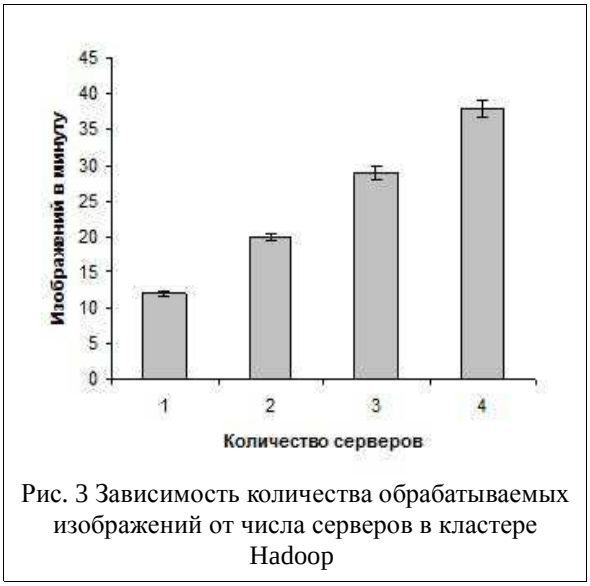

Рис. 3 Зависимость количества обрабатываемых изображений от числа серверов в кластере Hadoop

Выполненная реализация практически применена для обработки изображений от системы Particle Image Velocimetry [9], источник данных — проект PIV Challenge [10]. В процессе обработки вычисляется поле скорости потока с использованием стандартного кросскорреляционного алгоритма. Зависимость количества изображений, обработанных в минуту, от числа серверов в кластере Hadoop, приведено на рис. 3 (представлены средние значения с доверительными интервалами, количество экспериментов 10 шт. для каждого числа серверов). Обрабатывались изображения в формате BMP, размер одного изображения 250 КБ, общий объем изображений 500 ГБ.

Как видно из рис. 3., текущая реализация СОИБО масштабируется почти линейно с ростом числа серверов в кластере Hadoop. Следует отметить, что данная реализация выполнена на Java, и с переходом на C и Hadoop Streaming производительность обработки изображений может быть увеличена.

Заключение. Предложена архитектура системы обработки больших объемов изображений на основе MapReduce с автоматическим распараллеливанием. Особенностью системы является изоляция от прикладного программиста, занимающегося обработкой изображений, деталей внутреннего устройства Hadoop, связанных с организацией параллельной работы и загрузкой изображений. Прикладному програмисту предоставляется простой API (C или Java) для работы с изображением, уже загруженным в память. Программист разрабатывает последовательную функцию, обрабатывающую только одно изображение, а запуск этой функции для обработки большого объема изображений в параллельном режиме выполняется системой автоматически.

Тестовая реализация и практическое применение для обработки изображений от системы PIV подтвердили работоспособность предлагаемой архитектуры.

Направлениями дальнейших работ является:

- реализация Image C API для повышения производительности обработки изображений;
- реализация библиотеки готовых обработчиков изображений Image Processing Library;
- поддержка обработки изображений на GPU.

Работа поддержана грантом УрО РАН 12-П-1-1029

ЛИТЕРАТУРА:

1. Jeffrey Dean, Sanjay Ghemawat: MapReduce: simplified data processing on large clusters. Commun. ACM 51(1): 107-113 (2008).
2. Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google File System // 19th ACM Symposium on Operating Systems Principles, Lake George, NY, October, 2003
3. Apache Hadoop [Электронный ресурс]. – http://hadoop.apache.org/ (дата обращения 29.05.2012)
4. James Hays, Alexei A. Efros. IM2GPS: estimating geographic information from a single image. Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2008.
5. K. Wiley, A. Connolly, J. Gardner, S. Krughoff, M. Balazinska, B. Howe, Y. Kwon, Y. Bu. Astronomy in the Cloud: Using MapReduce for Image Co-Addition // Publications of the Astronomical Society of the Pacific. - 2011. - V.123, №901. - P. 366-380.
6. Almeer, M.H. Cloud Hadoop Map Reduce For Remote Sensing Image Analysis // Journal of Emerging Trends in Computing and Information Sciences. - 2012. - V.3, № 4. - P. 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. Springer-Verlag Berlin, Heidelberg. - 2009. - P. 302-319.

8. Chris Sweeney, Liu Liu, Sean Arietta, Jason Lawrence. HIPI: A Hadoop Image Processing Interface for Image-based MapReduce Tasks. Chris. University of Virginia. 2011.
9. Adrian, R.J.; Westerweel, J. Particle Image Velocimetry. Cambridge University Press. 2011. ISBN 978-0-521-44008-0
10. PIV Challenge [Электронный ресурс]. URL: http://www.pivchallenge.org (дата обращения 29.05.2012)