УДК 681.51

НГУЕН ТХАНЬ ФЫОНГ

КЛАСТЕРНЫЕ АЛГОРИТМЫ ПРЕДВАРИТЕЛЬНОЙ ОБРАБОТКИ КОСМИЧЕСКИХ ИЗОБРАЖЕНИЙ

Abstract: *In article cluster algorithms of space images pre-processing are proposed. Problem of parallel computations during space images filtration is considered, and the choice of the most effective method of image segmentation and distribution of tasks between processors of high-performance computers are justified. The efficiency of algorithms was demonstrated during the experiments that were run on cluster of Glushkov Institute of Cybernetics of NASU.*
Key words: *cluster algorithms, parallel computations, image processing, image filtration.*

Анотація: *У статті запропоновано кластерні алгоритми попередньої обробки космічних зображень. Розглядається задача розпаралелювання обчислень при розв'язанні задачі фільтрації космічних зображень, та обґрунтовується найбільш ефективний спосіб розбиття зображення та розподілення задач між процесами паралельної обчислювальної системи. Ефективність отриманих алгоритмів доведено шляхом виконання експериментів на кластері Інституту кібернетики ім. В.М. Глушкова НАНУ.*
Ключові слова: *кластерні алгоритми, паралельні обчислювання, обробка зображень, фільтрація зображень.*

Аннотация: *В статье предложены кластерные алгоритмы предварительной обработки космических изображений. Рассматривается задача распараллеливания вычислений при решении задачи фильтрации космических изображений, и обосновывается наиболее эффективный способ разбиения изображения и распределения задач между процессорами параллельной вычислительной системы. Эффективность полученных алгоритмов иллюстрируется результатами компьютерного моделирования на кластере Института кибернетики им. В.М. Глушкова НАНУ.*
Ключевые слова: *кластерные алгоритмы, параллельные вычисления, обработка изображений, фильтрация изображений.*

1. Введение

Одна из самых важных задач при работе с изображениями связана с их предварительной обработкой, т.е. выделением и фильтрацией шума. При этом в процессе фильтрации необходимо обеспечить максимальное сохранение деталей изображения. Фильтрация искаженных пикселей относится к группе низкоуровневых операций обработки изображения [1]. При последовательной обработке каждого пикселя время обработки изображений достаточно велико, что неприемлемо для решения различных прикладных задач в реальном времени. Эту проблему можно решить с привлечением высокопроизводительных параллельных вычислительных машин [2].

В данной работе рассматривается задача фильтрации изображений с космического аппарата (КА) Meteosat. Он представляет собой геостационарный спутник, поставляющий информацию для решения гидрометеорологических задач и некоторых задач наблюдения Земли из космоса. На снимках КА Meteosat шум представляет собой композицию импульсного шума и горизонтальных полос с равномерно распределенным импульсным шумом. Особенностью этих снимков является высокая частота их поступления (каждые полчаса) и большой объем: один снимок КА Meteosat-7 занимает до 10 Мбайт, а один снимок КА Meteosat второго поколения (MSG – Meteosat Second Generation) – до 100 Мбайт. Поэтому решение прикладных задач предварительной и тематической обработки данных этого спутника невозможно без привлечения больших вычислительных ресурсов.

В статье также рассматривается и задача распараллеливания вычислений при решении задачи фильтрации космических изображений и обосновывается наиболее эффективный способ

разбиения изображения, распределения задач между процессорами параллельной вычислительной системы. Приводятся алгоритмы параллельных вычислений, используемые для решения задач фильтрации. Эффективность полученных алгоритмов иллюстрируется результатами компьютерного моделирования на кластере Института кибернетики имени В.М. Глушкова НАНУ, в состав которого входит 32 процессора Xeon 2.66 GHz. Полученные результаты можно обобщить и на обработку любых изображений высокого разрешения, а не только для фильтрации космических снимков.

2. Принципы параллельных вычислений

В этом разделе кратко описаны основные принципы построения параллельных вычислительных систем, модели параллельных вычислений и инструментальные средства, которые поддерживают разработку параллельных программ.

2.1. Структура параллельных вычислительных систем

Существуют две основные архитектуры систем параллельных вычислений: на базе симметричного мультипроцессора (СМП) и массовых параллельных процессоров (МПП) [3].

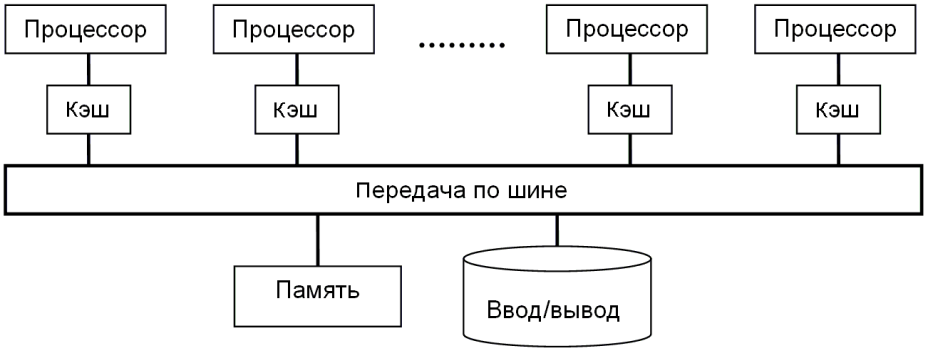

Рис. 1. СМП-архитектура

Система на основе симметричного мультипроцессора (СМП) (рис. 1) использует разделяемые ресурсы, например, память, подсистему ввода/вывода и т.д. При этом все процессоры имеют доступ к разделяемому ресурсу по принципу “симметрии” (“симметрия” означает, что ни один из процессоров не выполняет роль “операционного”) [3]. СМП-система обеспечивает механизм объединения данных, которые содержатся в локальной сверхоперативной памяти (кэш). Связь между процессорами и разделяемой памятью обеспечивает шина. Такое соединение компьютеров называется параллельным с разделяемой памятью.

Альтернативу СМП-системам составляют архитектуры на основе массовых параллельных процессоров МПП (рис. 2). Такая система состоит из отдельных компьютеров, называемых узлами, которые соединены между собой быстродействующей вычислительной сетью. У каждого узла есть свои процессор, память, подсистемы ввода/вывода. Внутреннее сетевое соединение позволяет осуществлять передачу данных между узлами. Такое соединение компьютеров называется параллельным с распределенной памятью.

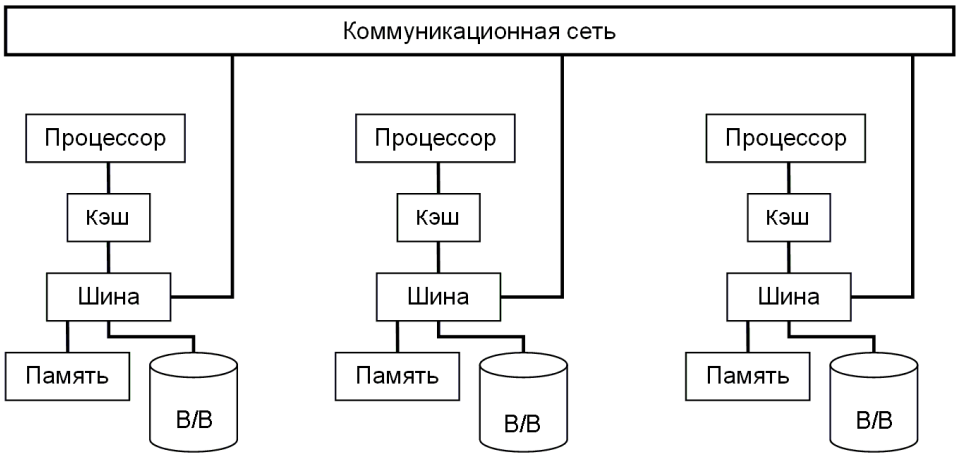

Рис. 2. МПП-архитектура

При выполнении параллельных вычислений в МПП-системе ресурсы сети используются по мере необходимости и ограничиваются только функциональными возможностями сети.

2.2. Модели параллельных вычислений

Существуют две общепринятые модели параллельных вычислений (рис. 3). Их отличие состоит в организации доступа к памяти процессора. В модели с разделяемой памятью каждый из потоков управления имеет доступ ко всей памяти. В модели с распределённой памятью память процессора доступна только для действующего процесса.

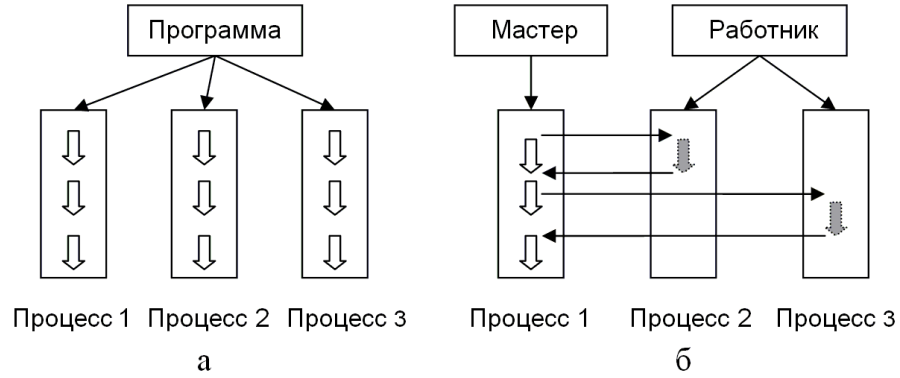

Рис. 3. Модели ЕПМД (а) и МПМД (б)

Примером реализации модели с распределённой памятью является модель передачи сообщений. Существуют два вида этой модели: единая программа со множеством данных – ЕПМД (рис. 3а) и множественные программы со множественными данными МПМД. В ЕПМД действует только одна программа, и каждый процесс соответствует выполнению кода этой программы на различных множествах данных. В модели МПМД разным процессам соответствуют различные программы, но процессы “сотрудничают” между собой для решения единой проблемы. Типичным примером модели МПМД является архитектура “мастер/работник” (рис. 3б): “мастер” отправляет работы “работнику” и несколько “работников” обслуживают одного “мастера”.

2.3. Интерфейс передачи сообщений MPI

Интерфейс передачи сообщений MPI является библиотекой функций и макросов, которые могут быть использованы в программах, написанных на языках программирования C, FORTRAN и C++ [4, 5]. MPI предназначен для программ, основанных на модели передачи сообщений [6]. Приведем три отличительных особенности этой библиотеки [7].

1. Платформенная независимость, т.е. одинаковый исходный код может выполняться на всем разнообразии машин, для которых можно использовать библиотеку MPI.

2. Возможность прозрачного выполнения в неоднородной среде. MPI обеспечивает виртуальную вычислительную машину, которая «скрывает» архитектурные различия. MPI по необходимости автоматически преобразует типы данных и использует нужный коммуникационный протокол.

3. Эффективная реализация на машинах с различными характеристиками. MPI избегает конкретного описания выполнения операций заменяя его логическим описанием. Поэтому существует возможность использования преимуществ различных машин.

3. Последовательный алгоритм фильтрации импульсного шума

В настоящее время одним из наиболее эффективных алгоритмов устранения изолированных пикселей шума (при низкой вероятности искажения изображения) является пороговый булев фильтр – ПБФ [8]. Однако на космических изображениях иногда появляются целые полосы зашумленных пикселей. Тогда для корректировки изображений целесообразно использовать сочетание фильтра ПБФ и предложенного в [9] эффективного метода фильтрации полос шума. В данной статье мы не будем детально рассматривать эти методы, а приведем лишь краткое описание их применения.

На *первом шаге* алгоритма выполняются поиск и удаление полос импульсного шума. При этом результаты выполнения этой операции передаются для обработки на последующем шаге. На *втором шаге* восстановление искаженных пикселей происходит только после фазы детектирования карты зашумленных пикселей. Кроме того, для повышения качества фильтрации этот шаг выполняется итеративно четыре раза. (Данное число было получено экспериментальным путем; при увеличении числа итераций происходило сглаживание изображения.) При этом результаты текущей итерации являются входными данными для последующей.

Среди популярных парадигм параллельных вычислений наиболее эффективными являются модели параллелизма данных и параллелизма задачи [10], которые относятся к моделям ЕПМД и МПМД соответственно. Следует отметить, что иногда сочетание параллелизма данных и задачи в одной программе повышает быстродействие больших вычислительных систем по сравнению с использованием каждого из подходов в отдельности [11]. Однако для решения поставленной задачи мы ограничимся рамками модели параллелизма данных. Это объясняется следующими причинами:

1. С точки зрения алгоритма, *первый* и *второй шаги* (до некоторой степени) могут существовать независимо, поэтому можно реализовать и модель параллелизма задачи. Но при этом возникают две проблемы:

– эксперименты в [9] показали, что предложенный алгоритм достигает максимальной эффективности только в том случае, когда *первый шаг* выполняется вначале, а *второму шагу* передается его результат;

– в модели параллелизма задачи существенно усложняется процесс балансировки нагрузки (корректного распределения заданий между процессами обработки с целью увеличения общей производительности [2]). Львиную долю времени обработки на каждом шаге занимает поиск полос импульсного шума и искаженных пикселей, так как они обрабатывают все пиксели изображения (включая сортировку соседних пикселей обрабатываемого пикселя). В свою очередь, при удалении полос импульсного шума и искаженных пикселей работа происходит только с незначительным количеством данных (< 0.5% от всего изображения). Поскольку поиск и удаление искаженных пикселей выполняется итеративно четыре раза, обеспечение равномерного распределения заданий между процессами является непростой задачей.

2. С технической стороны, предложенный алгоритм фильтрации использует две из трех основных “низкоуровневых операций обработки изображения” [11]. Поэтому естественным выглядит подход на основе параллелизма данных, когда много операций применяется к каждому отдельному элементу данных.

И, наконец, объем обрабатываемых данных в данной задаче не настолько велик, чтобы сочетание параллелизма данных и задачи показало свои преимущества.

4. Параллельный алгоритм фильтрации импульсного шума

Итак, в соответствии с проведенным выше анализом, для параллельной реализации алгоритма фильтрации импульсного шума целесообразно использовать модель параллелизма данных. В процессе обработки вся поверхность изображения сканируется окном одинакового размера $n \times n$ пикселей, где n – нечетное число. Обработку пикселей каждого окна можно выполнять параллельно в рамках отдельного процесса. Однако после каждой итерации обработки необходимо обеспечить обмен «краевой» информацией между соседними процессами. Это нежелательный побочный эффект распараллеливания, требующий связи между параллельно работающими процессами через связывающую шину (для отправки и получения информации между процессами, участвующими в параллельной обработке). Алгоритм фильтрации состоит в следующем:

1. Выполняется поиск полосы шума. Если она найдена, то удаляется.
2. Далее итеративно (четыре раза) выполняется каждый из следующих этапов:
    a) получение и отправка краевых данных соседним процессам;
    b) детектирование искаженных пикселей;
    c) восстановление обнаруженных искаженных пикселей.

Этот параллельный алгоритм основан на усовершенствованной модели ЕПМД, в которой применяются элементы архитектуры “мастер/работник”, соответствующей модели МПМД. Процесс “мастер” распределяет части изображения между процессами “работник”. Однако в данном случае процесс “мастер” также играет роль процесса “работник” и получает свою часть изображения.

Каждый процесс “работник” обрабатывает свои данные, после чего процесс “мастер” собирает в рамках одного изображения части данных процессов “работник”.

На *первом шаге* выполнения алгоритма каждому процессу присваивается свой ранг. На *втором шаге* процесс с рангом 0 (мастер) считывает изображение из файла. На *третьем шаге* части изображения отправляются всем процессам, включая процесс с рангом 0 (теперь он играет роль работника). На *четвертом шаге* каждый процесс обрабатывает свои данные, используя вышеописанный алгоритм фильтрации. По завершении этого шага ни один из процессов не обладает полными данными, и уже на *пятом шаге* все части изображения собираются и передаются процессу с рангом 0 (мастер). Этот процесс записывает обрабатываемое изображение в файл (*шестой шаг*).

При разработке программы с распараллеливанием выполнения задач, необходимо придерживаться следующих принципов [3]:

1. Максимальное увеличение последовательных фрагментов программы, которые можно выполнять параллельно.

2. Сбалансированность объема работ параллельных процессов.

3. Уменьшение времени обмена информацией.

В контексте поставленной задачи выполнение второго принципа зависит от того, существует ли полоса импульсного шума на части изображения, поступающей на обработку каждому процессору. Однако, как сказано выше, количество составляющих полосу искаженных пикселей незначительно (< 0.5 %), поэтому время обработки полосы тоже незначительно по сравнению со временем детектирования. Иначе говоря, второй принцип выполняется в том случае, когда размеры частей изображения, отправляемых различным процессорам, одинаковы.

Для уменьшения времени коммуникации необходимо минимизировать количество передаваемых данных (трафик) и число сеансов связи. Доступ к данным будет эффективнее, если они записаны в памяти последовательно, а не фрагментированы по всему носителю информации.

Будем также считать, что ширина космического изображения больше, чем его высота.

Рассмотрим четыре возможных способа разбиения данных изображения.

4.1. Разбиение изображения по горизонтали

Это самое естественное разбиение (рис. 4), поскольку данные в массиве обычно хранятся по строкам.

Серые зоны на рис. 4б содержат информацию о границах. Этой информацией обмениваются соседние процессы. Стрелки на рис. 4б указывают направление обмена данными (эти обозначения будут использоваться и далее).

При использовании такого подхода для поддержки процесса обмена "краевыми" данными существуют и некоторые ограничения. Во-первых, не обеспечивается сбалансированность объемов работ параллельных процессов, а, во-вторых, количество передаваемых между процессами данных максимально.

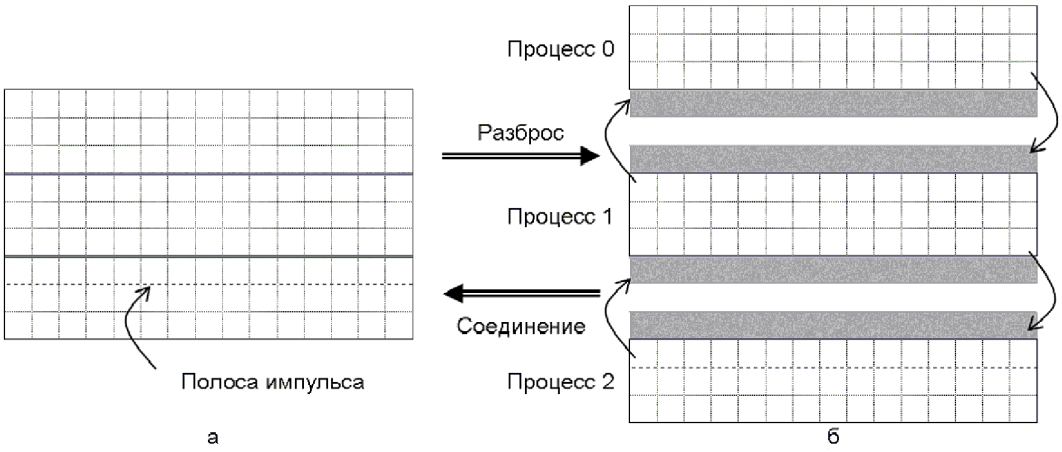

Рис. 4. Разбиение исходного изображения по горизонтали: а) исходное изображение; б) разбиение частей изображения по процессам

4.2. Поворот и горизонтальное разбиение изображения

Для устранения перечисленных выше недостатков до начала обработки изображения повернем его на 90°. При этом выполняется принцип сбалансированности объемов работ параллельных процессов, а также уменьшается количество данных, передаваемых между процессами. Следует также отметить, что при подобном разбиении для хранения данных все еще можно воспользоваться преимуществом хранения элементов массива по строкам (рис. 5).

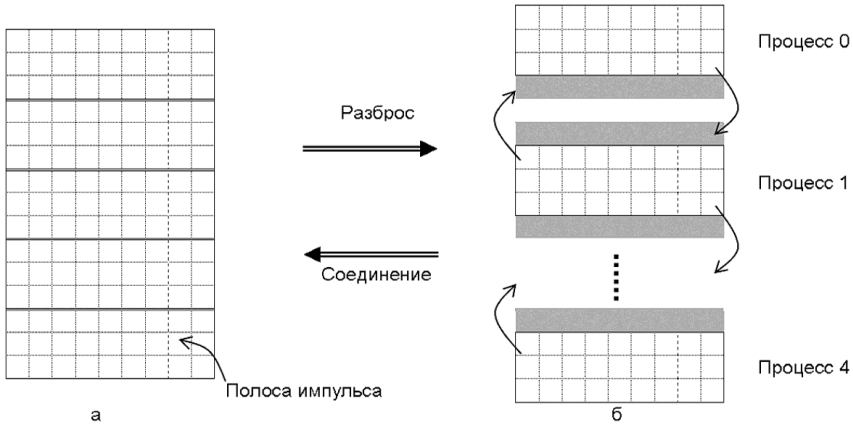

Рис. 5. Поворот исходного изображения на 90° и горизонтальное разбиение: а) повернутое изображение; б) части изображения, соответствующие разным процессам

Недостатком этого подхода являются временные затраты на поворот изображения до и после устранения шума.

4.3. Вертикальное разбиение изображения

Если считать, что интервал времени, необходимый для поворота изображения, является значительным и постоянным, тогда вполне логично вернуться к подходу, описанному в разд. 4.1, однако на этот раз исходное изображение предлагается разделить на вертикальные полосы (рис. 6).

При этом гарантированно выполняются два из трех перечисленных выше принципов: сбалансированность объемов работ параллельных процессов и уменьшение количества передаваемых данных. Недостатком этого подхода является необходимость обработки фрагментированной информации (поскольку фрагменты разных строк хранятся в памяти не подряд).

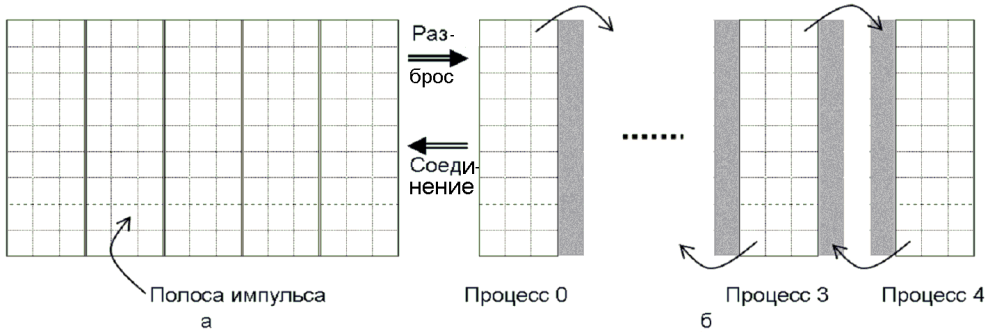

Рис. 6. Вертикальное разбиение исходного изображения: а) исходное изображение; б) части изображения, обрабатываемые разными процессами

С целью повышения эффективности были разработаны специальные методы для отправки/получения фрагментов изображения между процессами “мастер” и “работник”, а также для обмена граничными данными между соседними процессами.

4.4. Разбиение изображения по квадратам

Преимущество данного подхода заключается в том, что максимально уменьшается количество информации при обмене «границами» между соседними процессами. Поскольку при равных площадях периметр прямоугольника больше, чем периметр квадрата, было принято решение разбить исходное изображение на квадраты. Однако следует отметить, что этот подход в реализации является самым трудным из четырех предложенных (рис. 7).

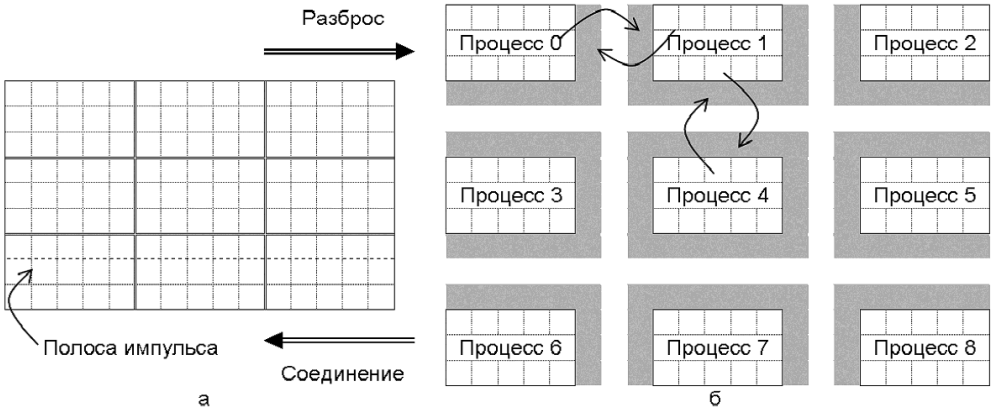

Рис. 7. Разбиение исходного изображения на квадраты: а) исходное изображение; б) части изображения, соответствующие различным процессам

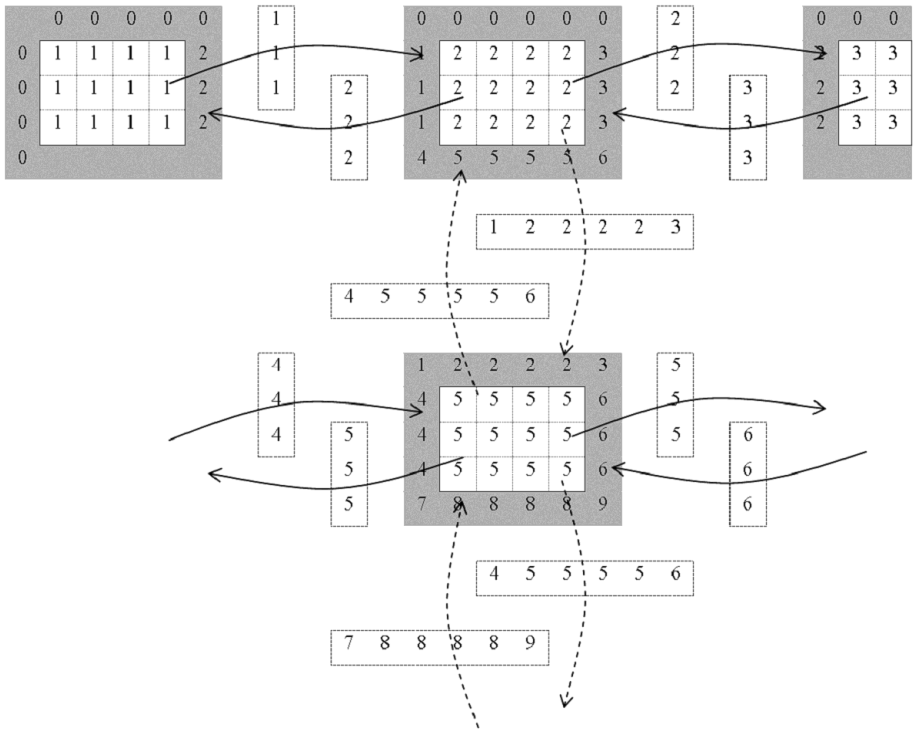

Рис. 8. Отправка/получение данных о границах. Сначала выполняется обмен данными «по горизонтали», а затем «по вертикали»

При таком подходе количество операций передачи информации о границах возрастает и составляет 4 операции обмена «по горизонтали» + 4 операции обмена «по вертикали» + 4 операции обмена элементами, расположенными по углам фрагмента изображения, соответствующего одному процессу. Применение подхода, предложенного в [3], дало возможность устранить передачу четырех угловых элементов (рис. 8).

5. Результаты экспериментов

Экспериментальные исследования проводились на кластере с 32 процессорами Xeon 2.66 GHz, который функционирует под управлением операционной системы Linux. В процессе экспериментов на кластере выполнялся процесс фильтрации большого космического изображения размером 100 Мбайт. Полученные результаты представлены на рис. 9.

При разделении изображения по квадратам массив логических процессоров формировался автоматически. Это было сделано с целью уменьшить максимально возможное количество пикселей, которые будут участвовать в фазе обмена информацией между процессами.

На основе полученных результатов можно сформулировать следующие выводы:

1. При горизонтальном разбиении изображения были получены наихудшие результаты. Дело в том, что в этом случае количество элементов, участвующих в процессе обмена информацией о границах, превышает количество элементов при использовании других типов разбиения. Этим и объясняется увеличение времени, требуемого для обработки изображения.

2. Время обработки при повороте, горизонтальном и вертикальном разбиении изображения приблизительно совпадает. Поворот изображения выполняется в оперативной памяти и по

сравнению с основным процессом требует меньше времени. Однако, если размер изображения достаточно большой, то его вертикальное разбиение является достаточно рациональным.

3. В большинстве случаев при вертикальном разбиении изображения требуется меньше времени, чем при его разбиении по квадратам (рис. 10). Основная причина является чисто технической. Внутри кластера отдельные вычислительные узлы (процессоры) связаны с использованием интерфейса Gigabit Ethernet. Это означает, что в единицу времени обмен информацией может происходить только между двумя процессами. Так, при разбиении изображения по квадратам количество таких «обменов» в два раза больше, чем при вертикальном разбиении. Именно поэтому с таким разбиением и связаны повышенные временные затраты.

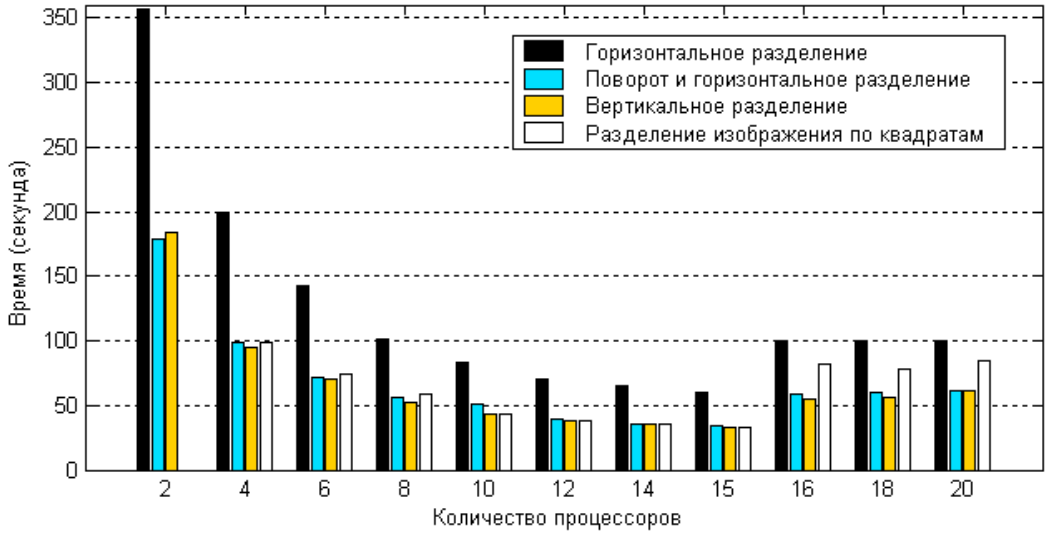

Рис. 9. Сравнительные результаты экспериментов, полученных на кластерном компьютере

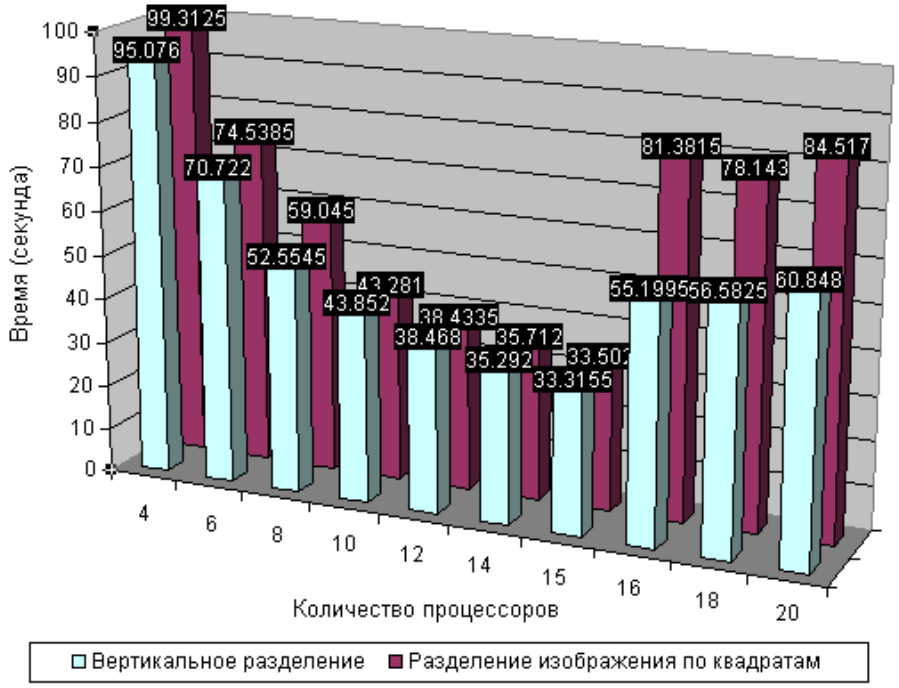

Рис. 10. Сравнение результатов обработки изображения при вертикальном разбиении и разбиении по квадратам

4. Как видно из рис. 10, минимальное время обработки изображения достигается при использовании 15 процессоров. При увеличении их количества эффективность рассмотренных методов снижается. Это связано с тем, что при увеличении числа процессов возрастает объем информации, которым обмениваются отдельные процессоры. Однако следует ожидать, что при переходе на специализированный интерфейс, например, SCI [13, 14], уменьшения производительности при увеличении числа процессоров не произойдет.

Космическое изображение КА Метеосат до и после фильтрации шума показано на рис. 11.

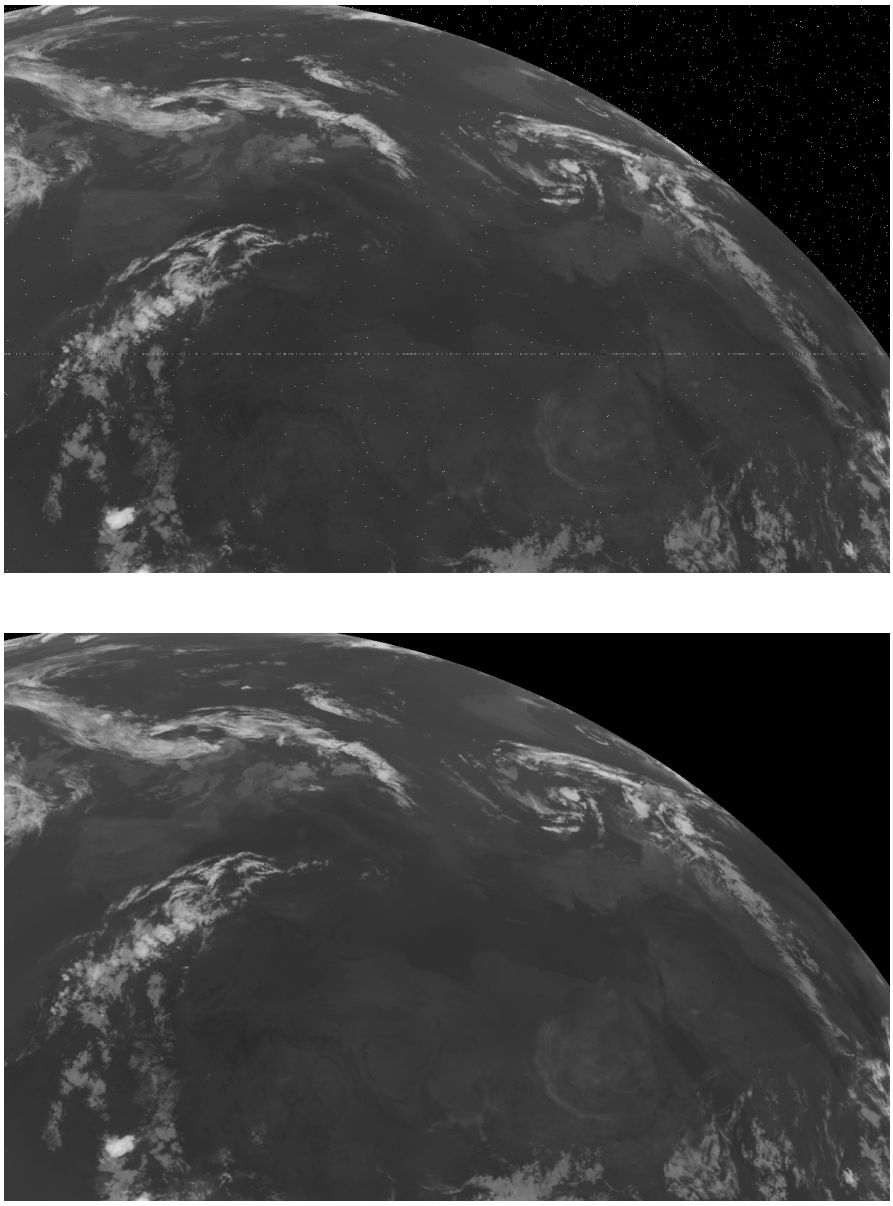

Рис. 11. Космический снимок КА Метеосат до (вверху) и после (внизу) фильтрации шума

6. Заключение

В данной статье предложен и обоснован параллельный алгоритм фильтрации изображений, искаженных импульсным шумом. Среди прочих достоинств для параллельной обработки данных характерны простота, естественность и эффективность. Если ставится задача оптимизации процесса решения задачи по времени, то лучше всего использовать вертикальное разбиение

изображения. Несмотря на то, что реализация этого подхода несколько сложнее остальных схем разбиения, результаты экспериментов показали, что для этого требуются наименьшие временные затраты. И, наоборот, если основным критерием является простота программной реализации, следует применять подход, связанный с поворотом и горизонтальным разбиением изображения. В этом случае отправку/получение частей изображения между основным и подчиненными процессами можно упростить, воспользовавшись парой функций MPI_Scatter и MPI_Gather из библиотеки функций MPI.

Предлагаемый алгоритм можно эффективно использовать при обработке мультиспектральных космических изображений большого объема в реальном времени. Описанный в данной статье алгоритм в дальнейшем планируется использовать в качестве модуля предварительной обработки изображений в системе определения динамики облачности на основе снимков КА Meteosat второго поколения. Кроме того, предложенные подходы к распараллеливанию изображения можно применять и для распараллеливания процесса решения тематических задач, в том числе построения маски облачности и оптических потоков. Учитывая большой объем изображений (до 100 MB), высокую частоту их поступления (каждые 15 минут), а также сложность алгоритмов определения динамики облачности, параллельную реализацию этих алгоритмов целесообразно выполнять не в локальной сети, а на кластере.

СПИСОК ЛИТЕРАТУРЫ

1. Nicolescu C., Jonker P. Parallel low-level image processing on a distributed-memory system // Proc. Workshop on Parallel and Distributed Methods for Image Processing (held in conjuction with IPDPS'2000, Cancun, Mexico, May 1-5). – 2000. – P. 226 – 233.
2. Seinstra F.J. User Transparent Parallel Image Processing // PhD Thesis. – Intelligent Sensory Information System. – University of Amsterdam, the Netherlands. – 2003 (http://www.science.uva.nl/~fjseins/Papers/Thesis/).
3. Aoyama Y., Nakano J. RS/6000 SP: Practical MPI Programming // RS/6000 Technical Support Center. – IBM Japan. – 1999 (http://www.sdsc.edu/user_services/datastar/guide/docs/RS6000_Practical_MPI_Programming.pdf).
4. Message Passing Interface Forum. MPI: A message-passing interface standard // Technical Report: UT-CS-94-230. – 1994 (http://scicomp.ewha.ac.kr/netlib/tennessee/ut-cs-94-230.ps).
5. Pacheco P.S. A user's guide to MPI // Technical report, Department of Mathematics. — University of San Francisco, California. – 1998 (http://www.cs.ucsb.edu/~gilbert/cs240aSpr2004/docs/MPIusersguide.pdf).
6. Gropp W.D. Issues in Accurate and Reliable Use of Parallel Computing in Numerical Programs // Preprint ANL/MCS-P1193-0804, Argonne National Laboratory. – 2004 (ftp://info.mcs.anl.gov/pub/tech_reports/reports/P1193.pdf).
7. MPI: The Complete Reference / M. Snir, S. Otto, S. Huss-Lederman, D. Walker, J. Dongarra. – The MPI Core, MIT Press, Boston (USA), 1998. – Vol. 1. – 450 p.
8. Aizenberg I., Astola J., Butakoff C., Egiazarian K., Paliy D. Effective Detection and Elimination of Impulse Noise with a Minimal Image Smoothing // Proc. of IEEE International Conf. on Image Processing. – 2003. – P. 357 – 360.
9. Нгуен Т.Ф. Простой и эффективный метод обнаружения и устранения полос импульсного шума на изображениях // Проблемы управления и информатики. – 2004. – № 5. – С. 125 – 130.
10. Fissgus U., Rauber T., Rünger G. A Framework for Generating Task Parallel Programs // The 7th Symposium on the Frontiers of Massively Parallel Computation. – Annapolis, (USA). – 1999. – P. 72 – 80.
11. Nicolescu C., Jonker P. A Data and Task Parallel Image Processing Environment // Parallel Computing. – 2002. – Vol. 28, Issue 7-8. – P. 945 – 965.
12. Gropp W.D., Lusk E., Doss N., Skjellum A. A High-Performance, Portable Implementation of the MPI Message Passing Interface // Parallel Computing. – 1996. – Vol. 22, Issue 6. – P. 789 – 828.
13. IEEE Standard for Scalable Coherent Interface (SCI) // IEEE Standard 1596–1992, IEEE Standards Board. –1993. – 490 p.
14. Hellwagner H., Reinefeld A. Scalable Coherent Interface: Technology and Applications // Proc. of SCI Europe ’98 . – 1998. – P. 165 –175.