УДК 519.683.8

К.С. Курочка[1], В.А. Ковалев[2]

ОРГАНИЗАЦИЯ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ ПРИ ОБРАБОТКЕ ЦИФРОВЫХ БИОМЕДИЦИНСКИХ ИЗОБРАЖЕНИЙ

Рассматривается задача организации распределенных вычислений для вычислительно емких алгоритмов обработки биомедицинских изображений в локальных вычислительных сетях и в сети Интернет. Приводятся данные вычислительных экспериментов на примере задачи обработки изображений с использованием скользящего окна большого размера.

Введение

Решение современных задач обработки, анализа и распознавания биомедицинских изображений приводит к необходимости обработки сотен гигабайт информации, что накладывает значительные требования на используемую вычислительную технику, от которой требуется не только высокая суммарная производительность обрабатывающих процессоров, но и большая оперативная память, достаточная для одновременного хранения сотен, а порой и тысяч изображений [1]. Даже самые быстродействующие современные ЭВМ последовательной архитектуры не способны решать задачи такой сложности [2]. В связи с этим наиболее эффективным способом решения задач данного типа оказалось их распараллеливание и распределенная обработка.

1. Методы организации параллельных распределенных вычислений

Существуют специальные процессоры обработки графической информации, которые являются весьма дорогостоящими и малораспространенными. Поэтому для решения указанных выше задач широкое применение находят различные технологии распределенных вычислений. В настоящее время распространены следующие аппаратные средства организации параллельных вычислений:

– суперкомпьютеры с параллельной архитектурой векторных процессоров;
– симметричные мультипроцессорные системы;
– системы с массовым параллелизмом;
– кластерные грид-системы, созданные на основе локальных или глобальных сетей [3, 4].

В современных условиях наибольшее распространение получили кластерные системы, достоинством которых является то, что они позволяют для достижения необходимой производительности объединять в единые вычислительные системы компьютеры самого разного типа, начиная от персональных и заканчивая многопроцессорными суперкомпьютерами. Таким образом, при относительно низких затратах кластерные системы позволяют достигать производительность, равную полученной на специализированном суперкомпьютере.

Выделяют ряд моделей организации вычислительных систем [3, 5]:

– централизованные модели программирования с общедоступным состоянием. Обычно применяются в тесно связанных системах с синхронными языками и средствами реализации, предназначенными для машин с общей (совместно используемой) или гибридной (физически распределенной, но логически общей) памятью;

– модели передачи сообщений. Наиболее широкое распространение получил стандарт MPI (Message Passing Interface), обеспечивающий связь между ветвями параллельного приложения [6]. В рамках MPI для решения поставленной задачи разрабатывается одна программа, и эта единственная программа запускается на выполнение одновременно на всех имеющихся процессорах или системах. При этом каждый процесс может обрабатывать свой набор исходных данных. В MPI существует целое множество операций передачи данных. Они обеспечивают разные способы пересылки данных, реализуют практически все коммуникационные опера-

ции. Подобный способ организации параллельных вычислений получил наименование «одна программа – множество процессов»;

– модели удаленного вызова процедур (RPC) и удаленного вызова методов (RMI). Обеспечивают те же возможности, что и MPI, но структурируют взаимодействие между отправителем и получателем в виде конструкции языка и представляют собой механизмы для управления потоками команд и данных;

– гибридные модели. Применяются в основном в кламперах (кластерах симметричных многопроцессорных систем) и в грид-системах, позволяют организовать выполнение как в пределах, так и вне непосредственно доступных адресных пространств;

– одноранговые вычисления (P2P). Представляют собой разделение компьютерных ресурсов и сервисов путем прямого обмена между системами, которые пользуются преимуществом существующих вычислительных возможностей настольных систем и связностью их работы с сетями;

– объектные и компонентные модели представляют интерфейсы метаязыков для управления и организации взаимодействия между объектами, находящимися в распределенной системе (CORBA, Cactus, COM/DCOM и др.);

– модель OGSA. Представляет собой открытую архитектуру грид-сервисов в Интернете, обеспечивает прозрачность размещения и многопротокольные связывания для экземпляров сервисов и поддерживает интеграцию со средствами платформ (Globus);

– координационные модели. Предоставляют средства интегрирования разнородных компонентов путем их связывания с помощью интерфейсов для формирования единого приложения, которое сможет выполняться в параллельной или распределенной системе.

Выбор той или иной модели грид-системы определяется классом решаемой задачи и имеющимся оборудованием.

2. Многопроцессная обработка биомедицинских изображений

В многопроцессной обработке биомедицинских изображений, как и в других подобных задачах, выделяются следующие фундаментальные части: коммуникация данных, организация управляющей структуры, статическое и динамическое распределения вычислительных ресурсов, синхронизация и оптимизация процессов [2, 3], причем наиболее узким местом в таких вычислительных системах будет коммуникация данных из-за больших объемов передаваемой информации. Поэтому следует использовать различные методы, позволяющие минимизировать объемы передаваемых видеоданных. Одним из способов решения данной проблемы является организация многоуровневых приложений [3], где работой клиентов управляет специальный компьютер или процесс, называемый сервером заданий (рис. 1).

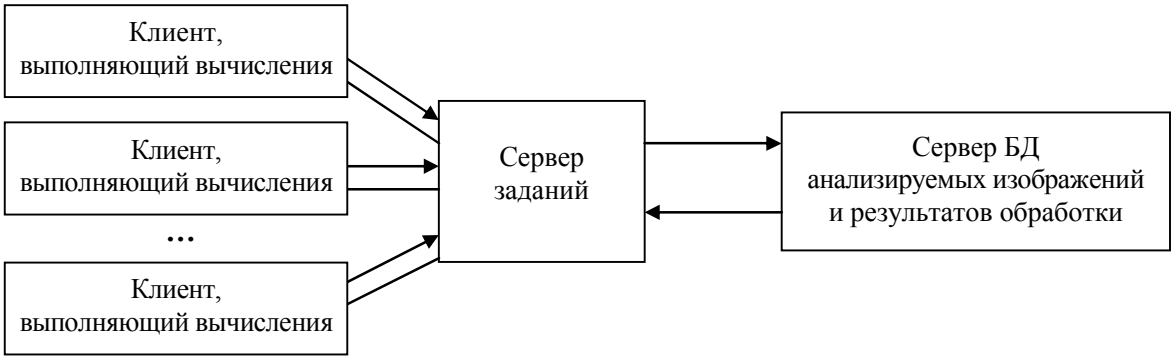

Рис. 1. Трехуровневое приложение распределенной обработки

Такая архитектура применяется при передаче сообщений и в объектно-компонентных системах. Применение программных пакетов, удовлетворяющих спецификации MPI, наряду с

преимуществами простой реализации распараллеливания вне зависимости от архитектуры используемой вычислительной системы имеет и свои недостатки. Разработчику приходится реализовывать ряд уже стандартных алгоритмов и методов преобразований цифровых изображений. В частности, это касается всевозможных алгоритмов попиксельной обработки изображений; всех видов алгоритмов, базирующихся на локальных свертках, включая вычисление градиента и большинство алгоритмов математической морфологии; алгоритмов, основанных на методах линейной алгебры, и т. п. В то же время, обладая хорошей структурированностью и регулярностью, данные классы алгоритмов допускают эффективное распараллеливание. Следует отметить, что значительная часть прикладных программных средств обработки, анализа и распознавания биомедицинских изображений часто основывается на использовании мощных пакетов статистического анализа и распознавания образов, реализованных, например, в среде многоплатформной системы R (http://www.r-project.org/). Поэтому естественным требованием к инструментальным средствам организации распределенной обработки данных является использование соответствующих пакетов системы R. Существует несколько явных способов организации распределенных вычислений средствами R. Можно организовать одноранговые вычисления, при этом передачу информации осуществлять средствами стека протоколов TCP/IP, используя механизм сокет (socket). При этом в качестве сервера заданий (см. рис.1) может выступать экземпляр R, выполняющий функции TCP-сервера. Взаимодействие между элементами распределенной системы осуществляется на сетевом и транспортном уровнях сетевых протоколов (согласно модели OSI). При реализации такого взаимодействия необходимо осуществить:

– распараллеливание решаемой задачи, т. е. выделение блоков задачи, которые могут быть выполнены одновременно;

– реализацию механизмов планирования и синхронизации параллельно выполняемых подзадач, что является достаточно сложной задачей;

– учет и резервирование вычислительных клиентов, т. е. контролирование, какой из удаленных клиентов в настоящий момент может принять участие в вычислениях, а какой из клиентов – нет.

Унифицированное решение данных задач средствами пакета R требует больших затрат, поэтому приходится при адаптации того или иного алгоритма каждый раз решать данные задачи в частном случае, т. е. применительно к какому-либо конкретному алгоритму. Часть проблем можно решить, если воспользоваться средствами MPI. В этом случае необходимо лишь написать унифицированный алгоритм обработки цифровых изображений, который мог бы выполняться на любом из компьютеров кластера MPI. Организацию кластера и определение доступных ресурсов возьмет на себя библиотека MPI, однако все равно необходимо будет писать отдельные распараллеливаемые реализации каждого алгоритма, что потребует значительных временных затрат для организации распределенных вычислений.

Для обработки биомедицинских изображений применяют два основных класса алгоритмов:

– распараллеливаемые ресурсоемкие алгоритмы (количество операций от n^4 до n^6, где n – количество обрабатываемых пикселей), позволяющие обрабатывать исходное изображение (или пакет изображений) одновременно, разбив его на части;

– множественные независимые последовательные (сериализируемые) алгоритмы.

Распараллеливаемые алгоритмы, как правило, применяются для обработки разных наборов данных, сформированных из одного изображения. При этом параллельные вычисления выполняются независимо – без обмена информацией между собой (например, техника скользящего окна). Очевидно, что любой такой распараллеливаемый алгоритм можно легко свести к множеству (серии) последовательных независимых алгоритмов, которые могут быть выполнены параллельно на узлах кластера.

Будем применять метод декомпозиции. Каждый *i*-й алгоритм представим в виде

$$\Phi_i\left(\mathbf{x}, \mathbf{y}, \mathbf{z}, t\right) = 0, \tag{1}$$

где Φ_i – алгоритм обработки; $\mathbf{x}$ – вектор входных переменных, $\mathbf{x} = \{x_1(t), x_2(t), \ldots, x_n(t)\}$; $\mathbf{y}$ – вектор выходных переменных, $\mathbf{y} = \{y_1(t), y_2(t), \ldots, y_m(t)\}$; $\mathbf{z}$ – вектор необходимых ресурсов, $\mathbf{z} = \{z_1(t), z_2(t), \ldots, z_r(t)\}$; t – время.

Пронумеруем каждый вектор входных и выходных переменных, а также вектор необходимых ресурсов. В итоге получим три множества:

$$\mathbf{x}_i \in \mathrm{X}, \ \mathbf{y}_i \in \mathrm{Y}, \ \mathbf{z}_i \in \mathrm{Z},$$

где $i = \overline{1, N}$, N – количество алгоритмов обработки.

Для каждого алгоритма необходимо описать входные и выходные интерфейсы. Под входным интерфейсом будем понимать унифицированный вектор **x**, под выходным – вектор **y**. Будем считать, что интерфейсы совпадают, если в соответствующих векторах совпадают имена переменных (значения переменных могут и не совпадать). Тогда каждый интерфейс будет представлять собой класс объектов, а векторы $\mathbf{x}_i$, $\mathbf{y}_i$, $\mathbf{z}_i$ будут являться экземплярами соответствующего класса интерфейса. Дополним класс интерфейса свойством, содержащим ссылку на другой интерфейс, с которым связывается данный, т. е. интерфейсы будем рассматривать попарно: входной – выходной. Сформируем множество входных интерфейсов $\mathrm{I}_{\text{вх}}$ и множество выходных интерфейсов $\mathrm{I}_{\text{вых}}$. Очевидно, что эти два множества могут отличаться только на интерфейсы векторов необходимых ресурсов и на один выходной набор переменных, представляющий результат обработки всей системы в целом. В противном случае будет иметься тупиковый алгоритм, не возвращающий результат. Всякий тупиковый алгоритм в данном случае является излишним и может быть исключен из рассмотрения. Таким образом, каждый алгоритм будет определяться соотношением (1), поддерживаемыми интерфейсами и связями с другими алгоритмами.

Определим иерархические связи между последовательностью алгоритмов обработки. Введем понятие уровня (слоя) по отношению к возможности параллельного исполнения алгоритма в кластере. Таким образом, алгоритмы могут располагаться на различных уровнях. Уровни между собой взаимодействуют последовательно, т. е., пока не выполнятся все алгоритмы нижнего уровня, переход к алгоритмам следующего уровня невозможен. Элементы одного уровня функциональной модели могут выполняться совместно (параллельно). Следовательно, еще на этапе создания алгоритмов можно осуществить их декомпозицию и определить порядок взаимодействия. При таком подходе любой распараллеливаемый алгоритм может быть преобразован в сериализируемый, что даст возможность проводить распределенные вычисления без модификации самих алгоритмов. Таким образом, предлагаемая технология позволяет использовать ряд имеющихся готовых алгоритмов обработки биомедицинских изображений без дополнительной переработки в кластерных системах.

В силу больших объемов обрабатываемых данных затраты на коммуникацию данных оказались выше затрат на их обработку. Кроме того, сложности в организации сериализируемой обработки привели к тому, что схему организации распределенных вычислений, представленную на рис. 1, применять на практике оказалось нецелесообразно. Чтобы уменьшить затраты на коммуникацию данных, увеличить эффективность распределенных вычислений и позволить в полной мере воспользоваться преимуществами сериализируемой обработки, предлагается использовать несколько видоизмененную архитектуру вычислительной системы (рис. 2).

На рис. 2 в качестве клиентов выступают компьютеры локальной сети, в качестве БД изображений была выбрана СУБД Interbase 7.5, в качестве сервера заданий используется грид Unicore (www.unicore.eu) и разработанный планировщик сериализируемой обработки. При такой структуре вычислительной системы минимизируется внутрисетевой трафик за счет исключения передачи данных между планировщиком задач и вычислительными клиентами. Данные могут передаваться как на сетевом и транспортном уровне протоколов, так и на прикладном (согласно модели OSI). Использование прикладного уровня протоколов позволяет изоли-

ровать вычислительную часть от канальной части, обеспечивающей взаимодействие и передачу информации между клиентами.

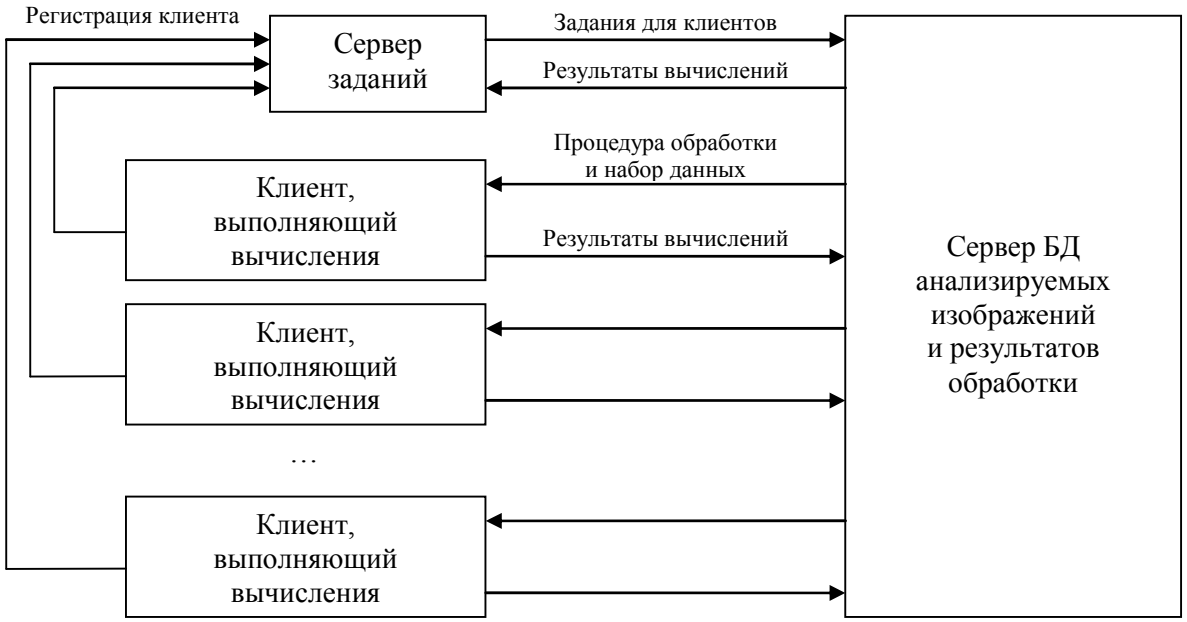

Рис. 2. Модифицированная схема организации распределенной обработки данных при обработке, анализе и распознавании биомедицинских изображений

Суть предлагаемого подхода заключается в том, что каждый клиент обращается к СУБД и, подключаясь, заносит в базу данных свою идентификационную информацию, сообщая тем самым о готовности принять участие в вычислениях. Сервер заданий анализирует количество доступных клиентов и помещает в базу данных информацию о том, какую часть изображения (или изображение целиком) должен обработать данный клиент. Каждый клиент считывает из базы данных обрабатываемую часть изображения и алгоритм обработки, выполняет вычисления и помещает результат назад в базу данных, устанавливая признак, что вычисления выполнены. Таким образом, по прошествии определенного времени планировщик может определить, выполнена ли задача полностью, а если не выполнена, то какой блок должен быть обработан повторно.

3. Вычислительный эксперимент

Для экспериментальной оценки производительности рассматриваемой архитектуры была проведена апробация предлагаемой методики организации распределенных вычислений и программного обеспечения при анализе биомедицинских изображений. В качестве тестовой рассматривалась специальная задача фильтрации растрового изображения повышенной вычислительной емкости с использованием техники скользящего окна достаточно большого размера. Суть задачи заключалась в вычислении градиентного изображения путем сравнения соответствующих выборок пикселей, взятых из двух половин окна с использованием *t*-теста Стьюдента. Указанный тест выполнялся для обоих ортогональных направлений в окрестности каждого пикселя изображения, полученное значение средней величины статистической значимости различий выборок трактовалось как градиент и его значение присваивалось соответствующему пикселю выходного изображения. Таким образом, при размере тестовых изображений 1024×768 пикселей вычисление выходного градиентного изображения предполагало повторение статистического теста Стьюдента около полутора миллионов раз (точнее, 1 572 864 раза за вычетом краев). Для организации распределенной обработки изображение разделялось на части и передавалось на рабочие станции по локальной вычислительной сети.

Каждая рабочая станция обрабатывала свою часть изображения и отправляла результаты на сервер. Для обработки изображений на клиентских компьютерах и передачи информации на

сервер использовалась прикладная программа R и разработанные в среде Delphi расчетные модули. Обрабатываемое растровое изображение размером 1024×768 32-битных значений пикселей группами по нескольку десятков строк передавалось на обрабатывающие станции. Окрестность определялась в форме квадрата размером 55×55 пикселей. В случае использования 10 компьютеров на каждый из них передавалось по 187 строк. Передача данных осуществлялась последовательно каждому компьютеру. Таким же образом было организовано и получение результатов.

Процесс тестирования для каждого размера кластера повторялся 10 раз. Среднее время при однопоточной обработке составило 357 с (рис. 3). На управляющем компьютере расчетов не производилось. Рабочие станции функционировали под управлением ОС Windows XP, CPU Celeron 1000, ОЗУ 256 Мб. Сеть – Fast Ethernet 100 Мбит/с. Время передачи обрабатываемой информации при 10 узлах составило порядка 0,197 с, время обработки на одном узле при 10 станциях – порядка 8 с.

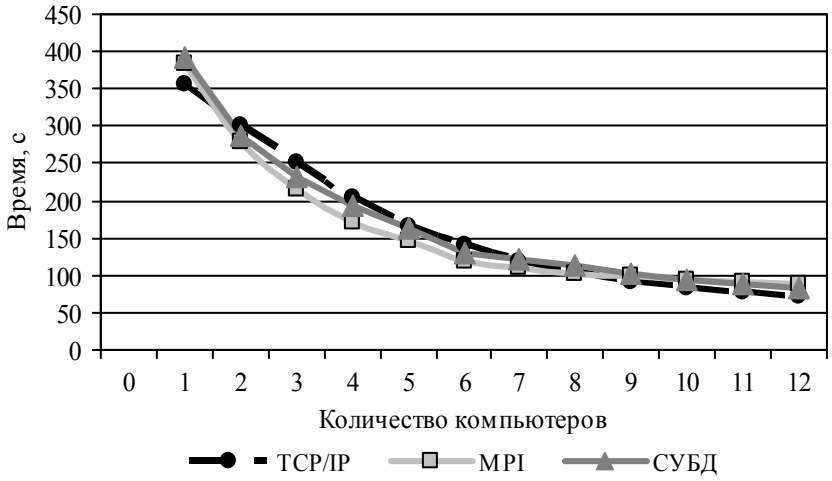

Рис. 3. Зависимость среднего времени распределенной обработки
от количества используемых компьютеров

Аналогичная задача была решена в том же кластере, но для реализации параллельного выполнения вычислений использовался пакет MPICH (http://www-unix.mcs.anl.gov/mpi/mpich2), удовлетворяющий стандарту MPI. Затем та же задача была решена с применением предлагаемого подхода. СУБД Interbase и Unicore были установлены на сервере заданий. Сервер заданий в расчетах не использовался. Результаты вычислительного эксперимента показаны на рис. 3 и 4.

Рис. 4. Зависимость времени распределенной обработки
от количества используемых компьютеров

Заключение

Анализируя результаты проведенных вычислительных экспериментов, можно сделать следующие выводы:

1. Сериализируемая обработка биомедицинских изображений не уступает по скорости распараллеливаемой обработке одними и теми же алгоритмами, но при этом обладает рядом преимуществ: во-первых, не требуется адаптации алгоритмов к распределенной вычислительной среде, во-вторых, минимизируется внутрисетевой трафик, что существенно уменьшает время вычислений при обработке пакета изображений.

2. Обработка биомедицинских изображений требует значительных затрат времени и обладает хорошей распараллеливаемостью. Эффективность использования кластера возрастает с увеличением объема вычислений пропорционально числу компьютеров.

3. Для каждой задачи существует такое предельное количество узлов кластера, после которого время обработки изображения перестает уменьшаться при наращивании количества компьютеров и начинает, наоборот, расти. Это обусловливается дополнительными затратами времени на расщепление изображения на части, передачу данных и другими накладными расходами. Таким образом, скорость передачи видеоданных становится сопоставимой со скоростью обработки. В этом случае эффективность кластера падает. Следовательно, в каждой конкретной архитектуре вычислительно-коммуникационных средств грид-сети возникает задача определения количества узлов, при котором она будет использоваться наиболее эффективно.

Работа выполнена в рамках союзной программы «СКИФ – ГРИД».

Список литературы

1. Гонсалес, Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – М. : Техносфера, 2006. – 1070 с.

2. William, K. Pratt Digital image processing / K. William. – Hoboken, New Jersey, 2007.

3. Foster, I. The anatomy of the grid: enabling scalable virtual organizations / I. Foster, C. Kesselman, S. Tuecke // International J. of Supercomputer Applications. – 2001. – № 15. – P. 200–222.

4. Lee, C. Grid programming models: current tools, issues and directions / C. Lee, D. Talia // Grid Computing – Making the Global Infrastructure a Reality. – Wiley, 2003. – P. 555–578.

5. Модели и средства программирования ГРИД-систем / А.Е. Дорошенко [и др.] // Проблеми програмування. – 2007. – № 3. – С. 16–31

6. MPI: The Complete Reference / M. Snir [et al.]. – N.Y. : MIT Press, 1996.

Поступила 07.09.2009

[1]*Гомельский государственный*
технический университет им. П.О. Сухого,
Гомель, пр. Октября, 48
e-mail: kurochka@gstu.gomel.by

[2]*Объединенный институт проблем*
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: vassili.kovalev@gmail.com

K.S. Kurochka, V.A. Kovalev

A TECHNIQUE FOR ORGANISING DISTRIBUTED COMPUTING IN BIOMEDICAL IMAGE PROCESSING

The paper is dedicated to the implementation of a distributed computing architecture for biomedical image processing. It is capitalized on a natural possibility of splitting the image data for block-wise processing algorithms and makes use suitable packages included in the free multi-platform language and software environment called R. Experimental results for the computationally-expensive problem of calculating generalized gradient images based on a *t*-test being applied to the halves of large pixel neighborhoods are reported.