КОНЦЕПЦИЯ РАСПРЕДЕЛЕННОГО ХРАНЕНИЯ И ПАРАЛЛЕЛЬНОЙ ОБРАБОТКИ КРУПНОФОРМАТНЫХ ИЗОБРАЖЕНИЙ

С.Б. Попов[1]
[1]*Институт систем обработки изображений РАН, Самара, Россия*

Аннотация

Предлагается новый подход к организации хранения данных на многопроцессорных системах различной архитектуры, базирующийся на концепции распределенных изображений. Рассматриваются основанные на предлагаемой концепции децентрализованный метод динамической балансировки и управления параллельным вычислительным процессом, а также организация визуализации крупноформатных распределенных изображений.

Введение

В настоящий момент растет интерес к системам параллельной или распределенной обработки изображений. В первую очередь это связано с тем, что появилась насущная потребность в обработке крупноформатных изображений. Наблюдается устойчивая тенденция к увеличению размеров формируемых изображений. Изображения земной поверхности большого размера становятся все более доступными, существует несколько спутников, которые предоставляют информацию в режиме реального времени в свободном доступе. Современные электронные микроскопы обеспечивают формирование наборов данных высокого разрешения объемом в несколько гигабайт.

Увеличение размера изображения порождает проблемы при их обработке, хранении и передаче данных. В первую очередь это связано с тем, что постоянное наращивание вычислительной мощности персональных компьютеров сформировало устойчивый выбор интерактивного способа проведения исследований в области обработки изображений: подбор параметров обработки, момент завершения итерационных процедур осуществляется пользователем визуально. Вместе с тем объем обрабатываемых данных сопоставим с объемом оперативной памяти современных высокопроизводительных рабочих станций или превышает его, тогда как широко распространенные универсальные программные системы обработки изображений рекомендуют, чтобы удвоенный объем наибольшего обрабатываемого изображения занимал не более 75% наличной оперативной памяти. При несоблюдении этого условия время выполнения даже простейших операций над крупноформатным изображением становится неприемлемо большим для пользователей интерактивных систем.

При проведении реальных исследований хранение достаточно представительного набора крупноформатных изображений может занимать значительную часть объема данных постоянного хранения, что может существенно снизить эффективность работы всего компьютера. Если работа выполняется группой исследователей, то неизбежно дублирование всего набора или части изображений. При обработке создается некоторое количество промежуточных изображений, объем которых может превышать исходный в несколько раз, обостряя проблему со свободным местом на диске компьютера.

Централизованное размещение крупноформатных изображений на специализированных хранилищах данных предъявляет повышенные требования к пропускной способности сети, причем, как правило, один запрос на визуализацию изображения порождает существенный трафик, поскольку запрашивается и передается весь объем данных, несмотря на то, что реально отображаться при этом может существенно меньшая часть. Обработка изображения на компьютере пользователя увеличивает этот трафик как минимум вдвое. Обработка изображения непосредственно на сервере хранения данных предполагает существенное увеличение вычислительной мощности сервера, а, значит, и стоимости хранения данных на нем.

Обработка изображений с использованием параллельных или распределенных систем решает многие из рассмотренных выше проблем. Однако большинство исследователей решают задачи организации параллельных систем обработки изображений в отрыве от того, как и где хранятся данные изображений [6,7,10,11]. Вместе с тем эффективность использования распределенной системы при обработке крупноформатных изображений во многом определяется способом организации доступа программ обработки к данным изображений и удобством визуализации результатов обработки пользователями.

В данной статье на основе анализа наиболее распространенных вариантов организации параллельной обработки изображений и их узких мест предлагается новый подход к организации хранения данных на многопроцессорных системах различной архитектуры, базирующийся на концепции распределенных изображений. Рассматриваются основанные на предлагаемой концепции децентрализованный метод динамической балансировки и управления параллельным вычислительным процессом, а также организация визуализации крупноформатных распределенных изображений.

1. Проблемы реализации концепции распределенного изображения

Основным показателем эффективности системы обработки изображений является время реакции на запрос пользователя. Это время складывается из

времени выборки данных из устройства хранения, времени обработки изображения (если это необходимо для выполнения запроса), времени передачи данных результирующего изображения на компьютер пользователя для его визуализации или последующей обработки.

Узким местом параллельных систем обработки изображений являются предварительный этап рассылки данных исходных изображений по компьютерам распределенной системы и завершающий этап сбора обработанных фрагментов в единое изображение [9].

Попытки распараллелить или существенно сократить эти необходимые, но непроизводительные этапы распределенной обработки изображений приводят к идее распределенного изображения.

В этом случае данные обрабатываемых изображений хранятся непосредственно на компьютерах, выполняющих параллельную обработку. Каждая задача параллельной программы обрабатывает тот фрагмент изображения, который расположен на компьютере, где выполняется данная задача, результат обработки сохраняется здесь же, как часть нового распределенного изображения, полученного в результате работы всех задач, участвующих в распределенной обработке.

Несмотря на очевидность данной идеи она не получила распространения, потому что исследователи не смогли предложить удовлетворительных решений возникающих при этом проблем:

- Как выполнить декомпозицию (разбиение), близкую к оптимальной для априори неизвестной последующей задачи обработки?
- Как решить проблемы сбалансированности загрузки компьютеров, участвующих в обработке при заранее выполненной декомпозиции данных?
- Как обеспечить достаточный уровень отказоустойчивости распределенного хранения фрагментов изображений?
- Как обеспечить виртуальную целостность распределенного изображения?
- Как обеспечить приемлемый уровень интерактивности системы при визуализации распределенных изображений?

С одной стороны при создании распределенного изображения заранее выполняется декомпозиция данных. С другой стороны при этом априори неизвестно как будет в дальнейшем обрабатываться изображение, на какие фрагменты оптимально поделить изображение с точки зрения сбалансированности загрузки компьютеров, участвующих в обработке. Причем, если программы работают не в режиме монопольного использования компьютера, то предсказать реальную загрузку конкретного компьютера не представляется возможным, и необходимо осуществлять динамическую балансировку в условиях уже существующего разбиения данных (заранее выполненной декомпозиции данных) или выполнять перераспределение данных по компьютерам в процессе обработки, снижая эффективность обработки в целом. В случае динамической балансировки при частых существенных изменениях нагрузки в процессе обработки такое перераспределение может свести на нет весь выигрыш от предварительного размещения данных по компьютерам обработки.

В работе [11] наличие предложенного авторами алгоритма сбалансированной загрузки компьютеров кластера для варианта централизованного менеджера распределения данных перевесило возможный выигрыш от распараллеливания ввода/вывода данных в случае заранее распределенных по компьютерам изображений, т.к. при таком варианте распределения данных указанный алгоритм балансировки загрузки не работает.

Далее рассмотрим выбор способа декомпозиции, который может быть использован при распараллеливании большинства операций обработки изображений.

2. Формализация описания процесса обработки изображений

Выделяют три вида операций [1,2], актуальных для большинства приложений в области обработки изображений, включая создание систем технического зрения.

1. **Низкоуровневые операции.** Эти операции выполняют преобразования над всеми точками изображения и формируют либо измененное изображение, либо некую векторную структуру или даже одно значение. Обычно вычисления имеют локальную природу и определяются как набор операций, необходимых для формирования соответственно пикселя изображения, элемента вектора или результирующего значения. Примерами таких операций являются операции фильтрации, повышения контраста, подчеркивания контуров, геометрические преобразования изображений, формирование гистограмм, получение различного рода статистических параметров и т.п.

2. **Операции промежуточного уровня.** Сюда относят разнообразные операции сегментации изображений, которые формируют некие структурные описания изображений. Например, списки описаний границ объектов на изображении.

3. **Высокоуровневые операции.** На этом уровне выполняется семантическая обработка различного рода структурных описаний, полученных на промежуточном уровне, формируется некое решение. В качестве примеров можно указать различного рода операции распознавания образов, семантической интерпретации сцен и т.п.

Следует отметить, что именно выполнение некой последовательности операций низкого уровня над набором исходных изображений является отправной точкой для многих типичных приложений в области обработки изображений и систем технического зрения. Основные затраты в рассматриваемых приложениях связаны с низкоуровневыми операциями. Эти операции требуют значительных вычислительных

ресурсов, ресурсов хранения. Время реакции всего приложения в целом определяется в большинстве случаев именно временем, затрачиваемым на выполнение низкоуровневых операций.

В общем виде процесс решения задачи обработки изображений предполагает либо изначальное указание последовательности некоторого числа ***шагов решения***, либо возможность декомпозиции метода решения на составляющие части, реализуемые в виде низкоуровневых операций. На каждом шаге решения обрабатываются и/или вырабатываются ***элементы решения***.

Формально этот процесс можно представить следующим образом.

На *I*-ом шаге формируются с помощью преобразования F^I необходимые элементы решения, представляемые, в общем случае, некоторым множеством параметров S^I и множеством изображений $\{X\}^I$

$$\left\{\{X\}^I, S^I\right\} =$$

$$= F^I\left(\{X\}, S, P\right), \{X\} \subset \bigcup_{i=0}^{I-1}\{X\}^i, S \subset \bigcup_{i=0}^{I-1} S^i,$$

где $\{X\}$, S, P – множества входных изображений, входных параметров и собственных параметров преобразования F^I соответственно. Любое из этих множеств на некотором конкретном шаге может быть пустым. $\{X\}^0$, S^0 – исходные множества изображений и параметров для рассматриваемой задачи.

Шаги решения с соответствующими элементами решения удобно представлять в виде ***графа решения*** данной задачи. Множеству шагов решения (операциям алгоритма F^I) ставится во взаимнооднозначное соответствие некоторое множество точек – вершин. Если элемент решения одной операции используется при выполнении другой операции, то соответствующие вершины соединяются дугой, направленной из той точки, где формируется данный элемент решения. Из каждой вершины выходит столько дуг, сколько формируется на данном шаге изображений и параметров. В случае, когда элемент решения является аргументом для нескольких операций, выходящих дуг может быть больше.

При обсуждении проблем организации процесса вычислений в системах обработки изображений нет необходимости в детальной конкретизации отдельных шагов технологии обработки, это задача разработчика прикладной программы, реализующей определенный этап решения. В данном случае существенен только информационный тип программы обработки изображения, то есть способ ее работы с данными изображений.

Большинство исследователей выделяют следующие виды низкоуровневых операций над изображениями [3, 4, 5]:

- поэлементная обработка (*PO – point operators*),
- обработка локальной окрестности (*LNO – local neighborhood operators*) или локальная обработка скользящим окном,
- глобальные операции обработки (*GO – global operators*), как правило, основанные на двумерном преобразовании Фурье или другом подобном преобразовании,
- глобальные операции редукции или операции вычисления параметров изображения, в.т.ч. статистических,
- геометрические преобразования.

Более общая классификация подразделяет операции на глобальные и локальные.

Методы глобальной обработки изображений характеризуются тем, что для вычисления значения каждой точки результирующего изображения $Y = \{y_{ij}\}$, в принципе, используются значения всех точек исходного изображения *X*:

$$y_{ij} = F_{ij}(X),\ y_{ij} \in Y,$$

где F_{ij} – оператор преобразования для точки y_{ij}; *i, j* – здесь и далее целочисленные координаты точек цифрового изображения.

Задачей глобальных операций редукции является получение некоторой оценки (или описания) обрабатываемого изображения. Под оценкой можно понимать, например, некоторые интегральные характеристики исходного изображения, геометрические характеристики различных объектов на изображении, отнесение исходного изображение или его частей к заданным классам. Преобразование редукции *R* по множеству изображений $\{X\}$ определяет необходимые параметры из множества *S*:

$$S = R(\{X\}).$$

Методы локальной обработки изображений формируют каждую выходную точку по результатам анализа ее окрестности на входном изображении. В свою очередь, они включают в себя поэлементные операции и обработку скользящим окном.

Поэлементная обработка включает в себя как унарные, так и бинарные операции. Унарные операции поэлементной обработки каждую точку исходного изображения *X* преобразуют в соответствующую ей точку выходного изображения *Y*:

$$y_{ij} = F(x_{ij}),$$

где $x_{ij} \in X, y_{ij} \in Y$.

Бинарные операции поэлементной обработки формируют каждую точку выходного изображения *Y* как результат некой операции над соответствующими точками двух исходных изображений X' и X'':

$$y_{ij} = F(x'_{ij}, x''_{ij}),$$

где $x'_{ij} \in X'$, $x''_{ij} \in X''$, $y_{ij} \in Y$.

Операции локальной обработки скользящим окном формируют изображение *Y*, каждая точка которого является результатом преобразования над некоторой соответствующей ей окрестностью *Q* точек исходного изображения

$$y_{ij} = F\left(\left\{x_{i+k,\,j+l} \,\middle|\, (k,l) \in Q\right\}\right).$$

Наиболее популярным видом обработки в данном случае является взвешенное суммирование окрестных отсчетов, а в качестве окна обработки – квадратная окрестность, симметрично расположенная относительно текущего отсчета.

$$y_{i,j} = \sum_{k=-K}^{K} \sum_{l=-L}^{L} a_{k,l} \cdot x_{i+k,\,j+l}\,, \quad i = \overline{0, M-1},\ j = \overline{0, N-1}.$$

При $K = L = 1$ это соответствует локальной обработке скользящим окном 3×3, при $K = L = 2$ – окном 5×5 и т.д.

Специальным классом операций над локальной окрестностью является рекурсивная обработка локальной окрестности (*RNO – recursive neighborhood operator*).

В данном случае каждая точка результирующего изображения *Y* зависит не только от точек некоторой окрестности *Q* исходного изображения *X*, но и от точек окрестности Q' самого результирующего изображения Y.

$$y_{ij} = F\left(\left\{x_{i+k,\,j+l},\, y_{i+m,\,j+n} \,\middle|\, (k,l) \in Q,\, (m,n) \in Q'\right\}\right)$$

Обычно такие операции реализуются с помощью следующей итерационной процедуры:

$$x_{i,j}^{p} = \sum_{k=-K}^{K} \sum_{l=-L}^{L} a_{k,l}^{p}\, x_{i+k,\,j+l}^{p-1}, \quad i = \overline{0, M-1},\ j = \overline{0, N-1},\ p = \overline{1, P},$$

т.е. к исходному изображению x_0 последовательно применяется P операций обработки.

Геометрические преобразования сложно отнести как к глобальным, так и к локальным операциям. При геометрических преобразованиях задается некоторая функция преобразования координат *G*

$$(k,l) = G(i,j) : \begin{cases} k = G_1(i) \\ l = G_2(j) \end{cases},$$

которая ставит в соответствие некоторой точке исходного изображения с координатами (i, j) точку результирующего изображения с координатами (k, l). В общем случае, каждая точка результирующего изображения $Y = \{y_{kl}\}$ формируется как результат некоторого комплексирования тех точек исходного изображения *X*, которые пространственно близки к координате $(i, j) = G^{-1}(k,l)$:

$$y_{kl} = F\left(\left\{x_{ij}, x_{i+1\,j}, x_{i\,j+1}, x_{i+1\,j+1} \,\middle|\, \left[i, j\right] = G^{-1}(k,l)\right\}\right),$$

$y_{ij} \in Y$,

где $[\cdot]$ означает формирование целочисленных координат путем отбрасывания дробной части.

Данная операция не относится к глобальным, поскольку для вычисления некоторой точки результирующего преобразования необходимо не более четырех точек исходного изображения. Но и локальной операцией ее нельзя назвать, так как расположение этих четырех точек в общем случае не находится в локальной окрестности формируемой точки.

Таким образом, несмотря на то, что основной особенностью изображений как данных является их двумерность, большинство алгоритмов и методов обработки изображений носят последовательный характер. Это проявляется, с одной стороны, в структуре вычислений «внутри» алгоритма. Очень популярно использование построчной, последовательной развертки отсчетов изображений и преобразований, имеющих локальный характер (поэлементных или на основе скользящего окна) [3,4]. С другой стороны, значительное число методов сложной обработки изображений могут быть реализованы как последовательное применение некоторых законченных типовых операций над изображениями. Именно это обстоятельство определяет эффективность применения разнообразных программных систем обработки изображений, разработанных для использования на универсальных компьютерах, к решению широкого спектра исследовательских и прикладных задач обработки и анализа видеоинформации.

К счастью, структура большинства алгоритмов низкоуровневых операций такова, что позволяет выполнить естественное распараллеливание алгоритмов на основе заложенной в них регулярности.

3. Распараллеливание обработки изображений

Рассмотрим основные варианты распараллеливания низкоуровневых операций обработки изображений.

Наиболее естественным вариантом распараллеливания является декомпозиция по данным на базе выходного изображения [5,6]: отсчеты результирующего изображения разбиваются на непересекающиеся фрагменты, и каждый фрагмент формируется параллельно на отдельном узле многопроцессорного кластера или компьютере распределенной системы, затем фрагменты, при необходимости, собираются в единое изображение. Декомпозиция изображения может быть выполнена следующими способами: одномерная декомпозиция по одной из координат, двумерная декомпозиция [7,8,9,11]. Соответствующая декомпозиция исходного изображения зависит от типа операции обработки [10].

Поэлементная обработка. При параллельном выполнении алгоритмов, осуществляющих поэлементную обработку, разбиение исходного изображения (или нескольких исходных изображений) полностью соответствует разбиению выходного изображения. Эффективность поэлементной обра-

ботки как одиночной операции, так и последовательности таких операций не зависит от способа разбиения изображений на фрагменты.

Локальная обработка скользящим окном. Данные алгоритмы очень хорошо распараллеливаются, но при этом необходимо исходное изображение разбивать на перекрывающиеся фрагменты. Размер перекрывающихся областей зависит от размеров окна обработки. Здесь так же возможен вариант как одномерной, так и двумерной декомпозиции. Поскольку в результате обработки рассматриваемого типа операций формируется распределенное изображение, содержащее непересекающиеся фрагменты, то при последовательном применении данных операций (вариант *RNO*) необходимо производить обмен перекрывающимися частями перед очередным этапом. Временные затраты на такое согласование данных распределенного изображения при различных вариантах декомпозиции различны и зависят от количества фрагментов и параметров коммуникационной среды. Например, при небольшом числе фрагментов и значительной величине латентности одномерный вариант декомпозиции более предпочтителен. Однако при увеличении числа фрагментов размер передаваемой при синхронизации информации значительно меньше для варианта двумерной декомпозиции.

Глобальные операции редукции. Операторы редукции, как правило, являются аддитивными, поэтому данный вид операций практически нечувствителен к способам разбиения обрабатываемых изображений на фрагменты. Но для определения некоторых параметров полезно наличие перекрывающихся областей у фрагментов изображения.

Глобальные операции обработки. Наиболее простым вариантом распараллеливания для этих операций является любой вариант декомпозиции выходного изображения и репликация полного исходного изображения по всем компьютерам, участвующим в обработке. Если полная репликация обрабатываемого изображения по каким-либо причинам невозможна, то при параллельном выполнении операций данного типа будет осуществляться интенсивный обмен данными между параллельно выполняющимися задачами обработки. В этом случае оптимальный выбор распределения данных очень часто зависит от используемого варианта алгоритма выполнения операции, и заранее сделать этот выбор очень сложно.

Геометрические преобразования. Данный вид операций включает в себя операции масштабирования и поворота, а так же нелинейного преобразования координат общего вида. С точки зрения минимизации затрат на передачу данных при осуществлении геометрических преобразований общего вида над распределенными изображениями наиболее оптимальным является разбиение на квадратные блоки. Однако при выполнении поворота изображения на небольшой угол более эффективным является одномерная декомпозиция. Причем как в том, так другом случае исходное изображение полезно разбивать на соответствующие фрагменты с перекрытием.

Наиболее популярным вариантом организации вычислений при распараллеливании низкоуровневых операций на основе декомпозиции по данным является использование программы-менеджера в сочетании с централизованным хранением обрабатываемых изображений. Специальная программа-менеджер осуществляет декомпозицию исходных изображений и распределение их фрагментов по задачам обработки, а также последующую сборку результирующего изображения. В зависимости от конкретного набора операций графа решения задачи обработки программа-менеджер для изображений, являющихся промежуточными элементами решения, либо выполняет сборку полного изображения и последующую рассылку в соответствии с новой схемой декомпозиции, либо производит обмен перекрывающимися частями фрагментов изображения на узлах обработки, если разбиение не изменяется при переходе к следующему шагу обработки.

Эффективность такого варианта параллельной или распределенной обработки изображений существенно снижается из-за наличия следующих узких мест: канал доступа к подсистеме хранения обрабатываемых изображений, когда задачи, обрабатывающие фрагменты изображения, практически одновременно запрашивают необходимые данные, и собственно программа-менеджер, осуществляющая декомпозицию и распределение фрагментов по задачам обработки, а также последующую сборку результирующего изображения [7,11,12].

Крайне желательно устранить или существенно сократить необходимый, но непроизводительный этап распределенной обработки изображений в виде рассылки данных исходных изображений по компьютерам распределенной системы и последующего сбора обработанных фрагментов в единое изображение.

Это возможно, если данные обрабатываемых изображений распределить по компьютерам заранее, то есть хранить их там, где выполняется обработка. Каждая программа обрабатывает тот фрагмент, который расположен на данном узле компьютерной сети, результат обработки сохраняется здесь же, как часть нового распределенного изображения, полученного в результате работы всех узлов, участвующих в распределенной обработке. Таким образом, основной единицей хранения в такой распределенной системе является распределенное изображение.

4. Структура распределенного изображения

Распределенное изображение – это структура данных, определяющая способ и параметры разбиения изображения на фрагменты, список компьютеров, где находятся эти фрагменты, место их размещения и формат хранения. В сущности, это описание того, как собрать из указанных в этой структуре фрагментов целое изображение.

Суммируя сделанный выше анализ вариантов декомпозиции изображений при выполнении различных операций обработки, следует отметить, что наиболее целесообразным представляется декомпозиция распределенного изображения в виде перекрывающихся фрагментов.

Суть предлагаемого подхода заключается в том, что размер необходимого перекрытия фрагментов определяется не параметрами последующей задачи обработки, поскольку она априори неизвестна, а необходимостью решения проблем сбалансированности загрузки компьютеров и обеспечения достаточного уровня отказоустойчивости распределенного хранения фрагментов изображений.

Распределенное изображение определяется в виде набора *перекрывающихся* фрагментов изображения. Для каждого из M компьютеров фрагмент, который он хранит, формируется следующим образом.

Все строки изображения делятся на $2M$ блоков одинакового размера. Фрагмент распределенного изображения на m-ом компьютере содержит два основных блока с номерами $2m$-1 и $2m$. Эти два основных блока соответствуют варианту разбиения изображения на непересекающиеся фрагменты.

Чтобы получить декомпозицию в виде перекрывающихся фрагментов, которая одновременно могла бы обеспечить минимальную степень отказоустойчивости, на данном компьютере дополнительно хранятся два так называемых теневых блока, т.е. копии («тени») примыкающих к нему основных блоков соседних компьютеров, блоков с номерами $2m$-2 и $2m$+1. В свою очередь, младший основной блок хранится в качестве одного из теневых блоков на компьютере с меньшим номером, а старший – на компьютере с большим номером.

Именно данные основных блоков формируются на узле в процессе распределенной обработки изображения. По окончании процесса обработки выполняется обмен теневыми данными.

Такая структура данных распределенного изображения позволяет восстановить его при отказе одного из узлов хранения, а также для большинства операций обеспечивает возможность формировать отсчеты основных блоков без передачи данных между соседними компьютерами в процессе распределенной обработки изображения.

Таким образом, распределенное изображение представляет собой набор *перекрывающихся* фрагментов изображения, который хранится распределено. На каждом узле хранения фрагмент содержит основные и теневые данные. С точки зрения чтения данных они равноправны, но в процессе распределенной обработки узел формирует только отсчеты основных блоков, теневые он получает после завершения процесса обработки от узлов, ответственных за формирование соответствующих основных блоков.

5. Алгоритм динамического распределения вычислительной нагрузки

Предложенный принцип декомпозиции распределенного изображения позволяет реализовать оригинальный алгоритм динамического распределения загрузки процессоров при выполнении операций поэлементной обработки (PO) или локальной обработки скользящим окном, (LNO и RNO).

Суть его рассмотрим на примере взаимодействия двух соседних узлов, m-ого и (m+1)-го, при формировании той части результирующего изображения, которая содержится в блоках данных с номерами $2m$ и $2m$+1. Заметим, что каждый из узлов имеет всю информацию, чтобы сделать эту работу самостоятельно (или почти всю, в случае выполнения операции локальной обработки скользящим окном).

m-й узел начинает формировать строки $2m$-го блока, начиная с первой строки в порядке возрастания номеров строк, (m+1)-й узел, в свою очередь, формирует строки ($2m$+1)-го блока, но начиная с последней строки блока и в порядке убывания номеров строк. После формирования определенного количества строк каждый узел информирует своего соседа о времени, за которое он выполнил эту работу. На основании этой информации вычисляется планируемый номер строки изображения, на которой узлы завершат такую совместную обработку.

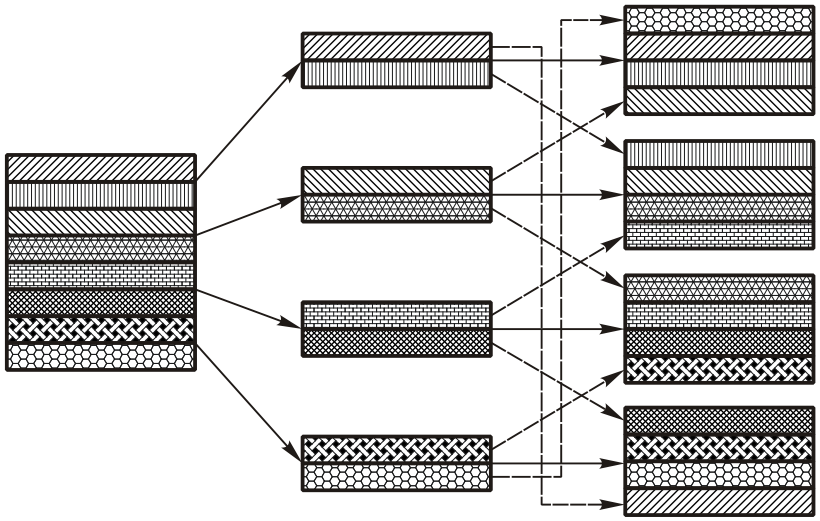

Рис. 1. Формирование структуры данных распределенного изображения

Таким образом, в процессе работы смежные узлы двигаются навстречу друг другу, сообщая о скорости своего процесса вычислений при достижении заранее определенных моментов. При этом прогнозируемый номер строки изображения, на которой эти процессы встретятся, постоянно корректируется в зависимости от текущей загрузки вычислительных узлов. Таким образом, все процессы завершат свою работу практически одновременно. Разница во времени при этом составит не больше, чем время обработки одной строки.

Одновременно каждый узел участвует в двух таких процессах, попеременно формируя строки старшего блока в порядке возрастания номера и строки своего младшего блока в порядке убывания.

В результате будет сформировано результирующее изображение в полном объеме, но размещение его данных по компьютерам, содержащим новое распределенное изображение, будет неравномерным. Далее в фоновом режиме узлы хранения только что сформированного распределенного изображения обменяются своими данными с тем, чтобы привести структуру распределенного изображения к необходимому виду. Однако пользователь может получить результат своего запроса сразу по завершении процесса обработки.

Начиная с какого времени и насколько часто необходимо узлу информировать своего соседа о выполненном текущем объеме работы?

Наиболее простой вариант, заключается в следующем: после того, как сформирована первая строка, отправляется информационное сообщение, содержащее время ее обработки. На основании полученной от смежного узла информации уже может быть спрогнозирован номер строки, на которой процессы встретятся. Точность такого прогноза будет тем выше, чем большее количество строк будет обработано перед отправкой такого сообщения. Однако при существенной разнице в производительности смежных узлов это количество не может быть слишком большим.

Рассмотрим, несколько идеализированный случай несбалансированной вычислительной системы, у которой производительность одного из узлов выше остальных в $\mu > 1$ раз. В случае одинаковой производительности всех M узлов время обработки изображения размером N строк составляет $T_0 = (N/M)\tau$, где τ – время обработки одной строки типовым узлом, каждый узел при этом обрабатывает по N/M строк. При возрастании производительности одного из узлов в μ раз время обработки изображения составит $T_1 = \dfrac{N}{M-1+\mu}\tau$, при этом более производительный узел обработает $\mu N/(M-1+\mu)$ строк, а остальные – по $N/(M-1+\mu)$ строк. Поскольку каждый узел содержит только $2N/M$ строк, то в случае, если для более производительного узла коэффициент μ превысит $\mu_{opt} = \dfrac{2(M-1)}{M-2}$, время обработки будет определяться менее производительными узлами, которые, обрабатывая по $N(M-2)/M(M-1)$ строк каждый, затратят $T_2 = \dfrac{N(M-2)}{M(M-1)}\tau$. Таким образом, если распределенная система состоит из четырех узлов $(M=4)$, то предлагаемый алгоритм может равномерно распределить нагрузку даже при трехкратном превышении производительности одного из узлов над остальными. Более производительный узел обеспечит обработку половины строк изображения, а на долю остальных узлов достанется по 1/6 части. Распределение обрабатываемых строк изображения по вычислительным узлам в данном случае представлено на рис. 2.

При увеличении числа узлов обработки/хранения

разброс производительности, который может быть скомпенсирован без временных потерь, постепенно уменьшается до двухкратного. Относительное сокращение времени обработки изображения от повышения производительности одного узла при различном количестве вычислительных узлов в распределенной системе приведено на рис. 3.

Достоинство предлагаемого алгоритма заключается в том, что он является полностью децентрализованным, и обеспечивает равномерное распределение нагрузки, если производительность соседних узлов не отличается больше чем в два раза.

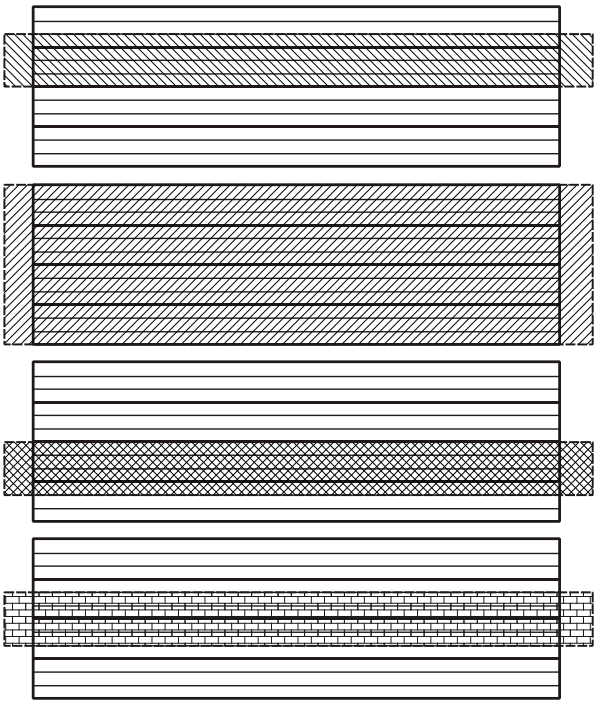

Рис.2. Распределение обрабатываемых строк изображения по четырем узлам

6. Отказоустойчивость распределенных изображений

Важным аспектом распределенных систем хранения является их отказоустойчивость, т.е. способность обеспечивать восстановление данных при отказе какого-либо узла распределенной системы. Это достигается введением необходимой избыточности данных. Перекрывающиеся области фрагментов распределенных изображений и могут выступать в качестве такой избыточности.

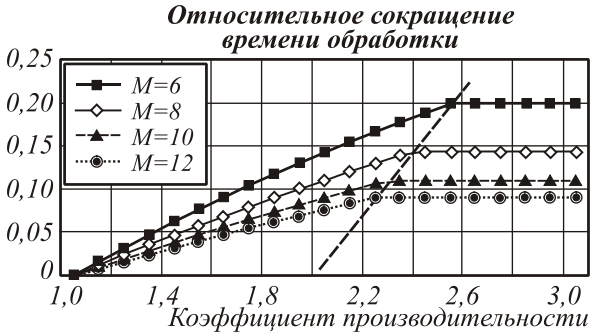

Рис. 3. Относительное сокращение времени обработки изображения при повышении производительности одного из узлов

Минимизацию избыточности, необходимой для восстановления информации при отказе одного из узлов хранения, обеспечивает одномерная декомпозиция в виде перекрывающихся на половину своей высоты горизонтальных полос. В данном случае суммарный объем данных, занимаемый изображением в распределенной системе хранения, возрастает в два раза. Использование двумерной декомпозиции с аналогичными размерами перекрытия увеличивает объем хранимой информации в четыре раза, не обеспечивая при этом повышения уровня отказоустойчивости системы в целом.

Дополнительную избыточность вносит необходимость хранения совместно с фрагментом изображения его эскиза, который обеспечивает приемлемый уровень интерактивности визуализации.

Существенно снизить степень избыточности распределенной системы хранения можно используя иерархические методы компрессии изображений, имеющие поддержку мультиразрешения [3]. Существующие варианты этих методов обеспечивают компрессию данных без внесения погрешности, что необходимо для использования в составе распределенной системы обработки изображений.

7. Визуализация изображений

Для изображений большого размера реализация этой функции должна обеспечивать приемлемый уровень интерактивности, т.е. пользователю необходимо за одну-две секунды предоставить для предварительного просмотра уменьшенную версию полного изображения, то есть выполнить масштабирование полного изображения до размеров окна визуализирующего приложения с типичными значениями в районе 1200×1000 отсчетов.

Это означает, что система хранения должна быстро сформировать необходимое для этого прореженное в несколько раз изображение (для изображений земной поверхности прореживание может составить до 25 раз по каждой координате). Для крупноформатных распределенных изображений возможным решением может быть наличие так называемого эскиза изображения на каждом или некоторых узлах распределенной системы хранения. В

этом случае визуализирующее приложение, обращаясь к узлу, обеспечивающему наиболее быстрый доступ для используемого пользователем компьютера, получает данные, необходимые для предварительного просмотра изображения. Количество узлов хранения, которые содержат эскизы, может выбираться исходя из конфигурации локальной сети рабочей группы пользователей, совместно использующих распределенную систему хранения, или допустимого уровня избыточной информации, которая может храниться в такой системе.

При просмотре изображения в полноразмерном режиме пользователю отображается только выбранный им фрагмент. При прокрутке изображения в окне происходит подкачка необходимых данных. Для обеспечения комфортного уровня интерактивности крайне желательно выполнять подкачку в режиме чтения с упреждением (pre paging). Наиболее удобным для данного режима является разбиение изображения на прямоугольные фрагменты [5]. Наличие перекрытий при этом делает скроллинг более плавным.

Заключение

Предлагаемая концепция организации данных в распределенных изображениях снимает большинство из существующих на сегодня проблем. Она позволяет осуществлять динамическое распределение нагрузки при распределенной обработке изображений с помощью децентрализованного алгоритма балансировки. Обеспечивается необходимый уровень отказоустойчивости распределенного хранения фрагментов изображений. Используемая декомпозиция для большинства операций дает возможность формировать новое изображение без передачи данных между соседними компьютерами в процессе распределенной обработки изображений. Рассмотренная организация данных позволяет организовать визуализацию распределенных изображений с достаточной степенью интерактивности.

В последующих статьях будут более подробно рассмотрены проблемы реализации динамического распределения вычислительной нагрузки узлов распределенной системы обработки крупноформатных изображений, отказоустойчивого хранения распределенных изображений и их визуализации.

Данная работа выполнена при поддержке российско-американской программы «Фундаментальное исследование и высшее образование» (“BRHE”), а также грантом РФФИ № 07-07-00210.

Литература

1. **Ballard, D.** Computer Vision / D. Ballard and C. Brown - Prentice Hall, 1982.
2. **Chaudhary, Vipin** Parallelism in Computer Vision: a review / Vipin Chaudhary and J.K. Aggarwal // In Book: Parallel Algorithms for Machine Intelligence and Vision / Vipin Kumar, P.S. Gopalakrishnan, and Laveen N. Kanal, editors, - Springer-Verlag, 1990. – pp. 271–309
3. Методы компьютерной обработки изображений (Издание второе, исправленное) / Под ред. В.А. Сойфера.. – М.: Физматлит, 2003, 784 с.
4. **Gonzales, R.G.** Digital Image Processing. / R.G. Gonzales and R.E. Woods - Addison-Wesley, 1992.
5. **Pitas.** Parallel Algorithms for Digital Image Processing, Computer Vision and Neural Networks / Pitas - John Wiley&Sons, 1993.
6. **Jonker, P.P.** Parallel image processing on heterogenous SIMD-MIMD machines / P.P. Jonker, J.G.E. Olk, and K. Volker // Proceedings of the IAPR Workshop on Machine Vision Applications, November 1996. – pp. 34-38.
7. **Koelma, D.** A software architecture for application driven high performance image processing / D. Koelma, P.P. Jonker, and H.J. Sips. // Proceedings of SPIE Conference, 1997. – V. 3166. – pp. 340-351.
8. **Olk, J.G.E.** Parallel image processing using distributed arrays of buckets / J.G.E. Olk and P.P. Jonker // Pattern Recognition and Image Analysis, 1997 – V.7(1).
9. **Seinstra, F.J.** Efficient applications in user transparent parallel image processing / F.J. Seinstra [and others] // The 16th International Parallel and Distributed Processing Symposium (IPDPS 2002) - Workshop on Parallel and Distributed Computing in Image Processing, Video Processing, and Multimedia (PDIVM 2002), Ford Lauderdale, Florida, U.S.A., April 2002.
10. **Serot, F.** SKIPPER: A skeleton-based programming environment for image processing applications / F. Serot, D. Ginhac, and J.P. Derutin // Proceedings of the Fifth International Conference on Parallel Computing Technologies, 1999.
11. **Squyres, J.M.** A toolkit for parallel image processing / J.M. Squyres, A. Lumsdaine, and R. Stevenson // Parallel and distributed methods for image processing II, Proc. SRIE 3452, 1998. – pp. 69-80.
12. **Nicolescu, C.** A data and task parallel image processing environment / C. Nicolescu and P. Jonker // Parallel Computing Journal - Special Issue on Parallel Computing in Image and Video Processing, August, 2002. – V.28(7-8). – pp. 945-965.

CONCEPT OF DISTRIBUTED STORAGE AND PARALLEL PROCESSING OF LARGE IMAGES

S.B. Popov[1]
[1]*Image Processing Systems Institute of the RAS, Samara, Russia*

Abstract

The paper proposes a new approach to data storing in multiprocessor systems with different architectures, being based on the distributed image concept. A decentralized technique for dynamic balancing and control of the parallel calculating process, as well as techniques for visualizing large distributed images, being based on the proposed concept, are considered.

Keywords: data storing, large distributed images, dynamic balancing and control, parallel calculating process

Citation: Popov SB. Concept of distributed storage and parallel processing of large images [In Russian]. Computer Optics 2007; 31(4): 77-85.

Acknowledgement: The work was supported by the Russian-American Basic Research and Higher Education Program (BRHE) and the RFBR grant No. 07-07-00210.

References:

[1] Ballard D, Brown C. Computer Vision. Prentice Hall, 1982.
[2] Chaudhary V, Aggarwal J.K. Parallelism in Computer Vision: a review. Parallel Algorithms for Machine Intelligence and Vision. Springer-Verlag, 1990; 271–309.
[3] Soifer VA, ed. Methods of Computer Image Processing. 2nd ed. rev [In Russian]. Moscow: "Fizmatlit" Publisher, 2003; 784 p.
[4] Gonzales RG, Woods RE. Digital Image Processing. Addison-Wesley, 1992.
[5] Pitas I. Parallel algorithms: for digital image processing, computer vision and neural networks. John Wiley & Sons, 1993.
[6] Jonker PP, Olk JGE, Volker K. Parallel image processing on heterogenous SIMD-MIMD machines. Proceedings of the IAPR Workshop on Machine Vision Applications, 1996; 34-38.
[7] Koelma D, Jonker PP, Sips HJ. A software architecture for application driven high performance image processing. Proceedings of SPIE Conference 1997; 3166: 346-357.
[8] Olk JGE, Jonker PP. Parallel image processing using distributed arrays of buckets. Pattern Recognition and Image Analysis 1997; 7(1): 114-121.
[9] Seinstra FJ. Efficient applications in user transparent parallel image processing. The 16th International Parallel and Distributed Processing Symposium (IPDPS 2002). Workshop on Parallel and Distributed Computing in Image Processing, Video Processing, and Multimedia (PDIVM 2002), Ford Lauderdale, Florida, U.S.A., 2002.
[10] Serot F, Ginhac D, Derutin JP. SKIPPER: A skeleton-based programming environment for image processing applications. Proceedings of the Fifth International Conference on Parallel Computing Technologies, 1999.
[11] Squyres JM, Lumsdaine A, Stevenson R. A toolkit for parallel image processing. Parallel and distributed methods for image processing II. Proc. SRIE 1998; 3452: 69-80.
[12] Nicolescu C, Jonker P. A data and task parallel image processing environment. Parallel Computing Journal. Special Issue on Parallel Computing in Image and Video Processing 2002; 28(7-8): 945-965.