УДК 004.22

Отвагин А.В., Дудкин А.А.

РЕАЛИЗАЦИЯ ПАРАЛЛЕЛЬНОЙ ОБРАБОТКИ ИЗОБРАЖЕНИЙ В СИСТЕМАХ ТЕХНИЧЕСКОГО ЗРЕНИЯ НА БАЗЕ MPI И МНОГОАГЕНТНОЙ АРХИТЕКТУРЫ

Введение

Современные технологии изготовления интегральных схем (ИС) требуют тщательного контроля всех критических моментов производственного процесса. Важную роль в этом процессе играют системы технического зрения (СТЗ) [1], которые обеспечивают регистрацию изображений топологий слоев ИС, их предварительную обработку и анализ.

СТЗ работает следующим образом. Изображение объекта через оптический прибор передается на преобразователь «свет-сигнал», электрический сигнал усиливается и запоминается. Затем выполняется подготовка изображения – совмещение кадров и улучшение изображения для уменьшения общего времени обработки изображения: изображения фильтруются, чтобы удалить фоновый шум и скорректировать геометрические искажения, производимые системой сбора данных.

Анализ изображения служит для извлечения информативных признаков (центра объекта, важных характерных особенностей: размера, позиции, углов, расстояний от центра до края, размер контура, текстурные меры областей и т.д.), выделения и распознавания объекта по его информационным признакам. Содержание данного этапа существенно зависит от модели распознавания, ориентированной на конкретный класс изображений, а также стратегии распознавания.

Далее результат обработки используется для описания и корректировки объектов топологии или на основе полученной информации контроллер связи СТЗ выбирает управляющие сигналы, приводящие в действие исполнительные механизмы, осуществляющие целенаправленное воздействие на объект анализа (напр., удаляет дефектные изделия из линейки производства).

Когда процесс обработки ограничен во времени и выходит за пределы возможностей обработки основного процессора, то для увеличения скорости обработки используется многопроцессорные вычислительные системы и специальное аппаратное обеспечение: ДСП, ПЛИС, заказные СБИС. Использование параллельной обработки дает возможность существенно ускорить обработку потока изображений в системе оперативного анализа ИС. Поэтому параллельная обработка информации является одной из наиболее востребованных технологий проектирования программного обеспечения (ПО). Соответственно, существует необходимость создания средств, облегчающих процесс разработки, анализа и планирования параллельных приложений.

Существует ряд зарубежных систем обработки данных СТЗ, использующих параллельную и распределенную обработку [2, 3]. В основе архитектуры этих систем используются различные технологии, например CORBA [2] или многоагентный подход [2]. Мы предлагаем использовать для проектирования и организации параллельных вычислений интегрированный набор средств, включающий в себя визуальный редактор, транслятор, систему оптимизации и систему поддержки параллельных вычислений на базе MPI[4]. Использование MPI делает нашу систему широко применимой для различных параллельных компьютеров.

1. Принципы разработки средств организации параллельных приложений

При проектировании средств реализованы следующие возможности:

1. Визуальное представление алгоритмов, основанное на графовом представлении и модели вычислений с параллелизмом задач. При этом вводится концепция вычислительной гранулы – независимой единицы вычислений, которая может разрабатываться на различных языках программирования и должна лишь реализовать конкретный интерфейс для интеграции в параллельный алгоритм.
2. Поддержка переносимости, реализованная в виде библиотек вычислительных гранул, подключаемых к системе организации вычислений во время выполнения. Сами средства разработки, анализа и выполнения программ строятся на платформо-независимых языках и стандартах (C++, Java, MPI).
3. Статическая и динамическая оптимизация разработанных параллельных алгоритмов с использованием алгоритма виртуальной ассоциативной сети. Этот алгоритм является разновидностью гибридного генетического алгоритма и обеспечивает быстрый поиск решения, близкого к оптимальному. Алгоритм имеет две модификации: первая применяется на этапе анализа при проектировании (статическая оптимизация), вторая – на этапе выполнения полученной программы (динамическая оптимизация).
4. Назначение операций алгоритма на процессоры вычислительной системы с учетом сведений, полученных на этапе оптимизации. Информация о назначении помещается в текстовое представление алгоритма и используется системой времени выполнения для реализации расписания.
5. Организация параллельных вычислений на основе многоагентной архитектуры вычислительной системы на базе MPI, что позволяет реализовать сложное поведение для оптимизации параллельных вычислений в условиях нестабильности характеристик аппаратных ресурсов и вычислительной нагрузки.

2. Базовые операции обработки изображений слоев ИС

Центральное место при обработке информации занимает идентификация объектов на изображениях. Объектами на топологическом слое ИС являются контактная площадка, контактное окно, диффузионный или металлизированный проводник, другие элементы топологии. Каждый из объектов характеризуется набором информационных признаков: цветом, формой и размером. Для идентификации объекта, весь набор информационных признаков должен удовлетворять заданным условиям.

В [5-6] нами предложены два алгоритма сегментации изображений топологии ИС. В первом, в качестве критерия разделения на области выступает яркость (интенсивность), а выделение контуров осуществляется на основе дискретного ортогонального преобразования Уолша, что позволяет получить более тонкую контурную линию. Данным алгоритмом хорошо разделяются изображения, имеющие две градации яркости: объект и фон принадлежат разным градациям. Недостатком названного выше алгоритма является его неустойчивость к изменениям условий съемки различных кадров изображения. Второй алгоритм сегментации основан на кластеризации, но учитывает возможное смещение центра цветового

Отвагин А.В., *младший научный сотрудник ОИПИ НАН Беларуси.*
Дудкин А.А., *ведущий научный сотрудник ОИПИ НАН Беларуси.*

кластера объектов, которое приводит к ошибкам сегментации.

Но основе этих двух алгоритмов разработан ряд алгоритмов идентификации, которые отличаются как операциями, так и параметрами обработки в соответствии с типом и свойствами изображений.

Большинство изображений характеризуется наличием мешающего фона, а также неопределенностью положения и ориентации отдельных элементов, приводящим к большой избыточности, что диктует необходимым использование методов предварительной обработки изображений: фильтрации, сглаживания и др. Изображения различных слоев (металлизации, поликремниевых и диффузионных) существенно различаются как по цветовым характеристикам, так и по форме объектов. Поэтому предлагаемые методы существенно учитывают особенности изображений, алгоритмы обработки слоя являются параметрически настраиваемыми, причем шаги алгоритма также могут варьироваться.

Выбор алгоритмов пре- и постобработки, их последовательность полностью определяется потребностями последующих алгоритмов обработки, а окончательная подстройка всех параметров алгоритмов заканчивается только после экспериментального тестирования всей системы. По сути, они уникальны для каждой решаемой задачи.

Предобработка состоит из следующих операторов:

- медианная фильтрация (с параметрами: размер окна, количество выполняемых итераций) выполняется для устранения шумовых составляющих вдоль границ металлических проводников и для размытия изображения;
- коррекция гистограммы по яркости с целью устранения теней вдоль границ проводников;
- гаусовская фильтрация (с параметрами: размер оператора, сигма, количество итераций) для более сильного размытия изображения;
- площадной фильтр (с параметром: площадь) – применяется, если изображение содержит элементы, которые соответствуют шуму, и размер этих элементов меньше размера реальных элементов. Удаляет элементы изображения, площадь которых меньше указанного минимально-допустимого значения (минимально-допустимая площадь задается в пикселях).

Постобработка состоит из операций улучшения сегментированного изображения:

- морфологических операций расширения и эрозии (с параметрами: размер окна, количество итераций);
- семантической фильтрации (с параметрами: площадь сегмента, размеры аппроксимирующего прямоугольника для сегмента, наибольшее расстояние между пикселями сегмента).

Параметры управления обработкой в первую очередь существенно зависят от типа обрабатываемого слоя.

Например, для обработки слоя металлизации имеем следующие операции предобработки и их параметры:

- медианная фильтрация (размер окна – 3×3, количество итераций – 5);
- коррекция гистограммы;
- гауссовская фильтрация (размер оператора – 3, сигма – 2, количество итераций – 2);
- медианная фильтрация (размер окна – 5×5, количество итераций – 80);
- и операции постобработки;
- расширение и эрозия (количество итераций – 4).

Сценарии, построенные оператором на одном примере изображения кадров слоя, должны быть применены ко всему множеству изображений этого слоя. Эти изображения вводятся в систему, которая может обрабатывать много изображений различных типов одновременно. Задача автоматизации состоит в разработке методов и средств для создания, моделирования, анализа и синтеза систем параллельной обработки.

3. Модель представления параллельного алгоритма

Параллельная обработка информации выполняется на кластере [7], в общем случае гетерогенном. Он представляется множеством $P=\{p_1, p_2,..., p_m\}$, где p_i - отдельный компьютер (узел кластера). Каждый компьютер p_i имеет характеристику производительности s_i, которая определяет время выполнения одной условной единицы вычислений. Узлы кластера соединены коммуникационными каналами. Каждый канал между узлами p_i *and* p_j, обозначенный l_{ij}, имеет характеристику b_{ij}, которая определяет ширину канала и скорость передачи данных между p_i *and* p_j.

Мы рассматриваем несколько вариантов параллельной обработки данных. Первый вариант – простой параллелизм операций. В этом случае объект данных поступает на вход параллельной программы и обрабатывается согласно алгоритму. Новый объект данных может быть обработан только после завершения обработки предыдущего. В этом случае распределение операций и составление расписания ориентировано в большей степени на высокую скорость обработки информационного объекта. Такое расписание называется блочным.

Другой вариант обработки – совмещение параллелизма операций и конвейерного параллелизма. В этом случае на вход параллельного приложения поступает сразу несколько объектов, формирующих поток данных. Начало обработки объекта В может начинаться параллельно с обработкой объекта А, как только процессоры, отвечающие за выполнение операций обработки объекта В, станут свободными. В этом случае получается расписание с перекрытием. При оптимизации расписания для конвейера следует учитывать не только характеристики алгоритма обработки, но и количество поступивших объектов.

Сам по себе поток объектов может не быть однородным, т.е. отдельные объекты обрабатываются по различным алгоритмам. Мы называем алгоритм обработки объекта определенного типа сценарием. Сценарии обработки в общем случае содержат множество операций, каждая из которых может избирательно применяться к объектам различных типов, либо применяться с различными параметрами в зависимости от типа обрабатываемого объекта. В этом случае поток может быть детерминированным, когда его структура и характеристики объектов известны заранее, или стохастическим, когда количество и порядок следования типов объектов случайны. Средства организации и оптимизации вычислений позволяют управлять обработкой как детерминированного, так и стохастического потока.

Поток, обрабатываемый параллельным алгоритмом, представляется множеством объектов $J=\{j_1,\ldots,j_n\}$. Каждый объект обрабатывается по сценарию, определяемому типом данного кадра. Система обработки способна выполнять множество операций обработки $O=\{o_1,\ldots,o_k\}, k>m$. Сценарий обработки отдельного типа кадров содержит подмножество операций $O_j=\{o_{j_1},\ldots,o_{j_k}\}, \bigcup_{j=1}^{n} O_j = O$. Операции в каждом сценарии упорядочены отношениями следования $o_{ja} \succ o_{jb}$, т.е. для кадра с типом j операция a выполняется перед операцией b. Каждая операция обработки o_i характеризуется сложностью вычислений $w_{o_i}^{j}$, представленной количеством единиц вычисления данной операции для определенного типа кадров j. Две операции над различными кадрами,

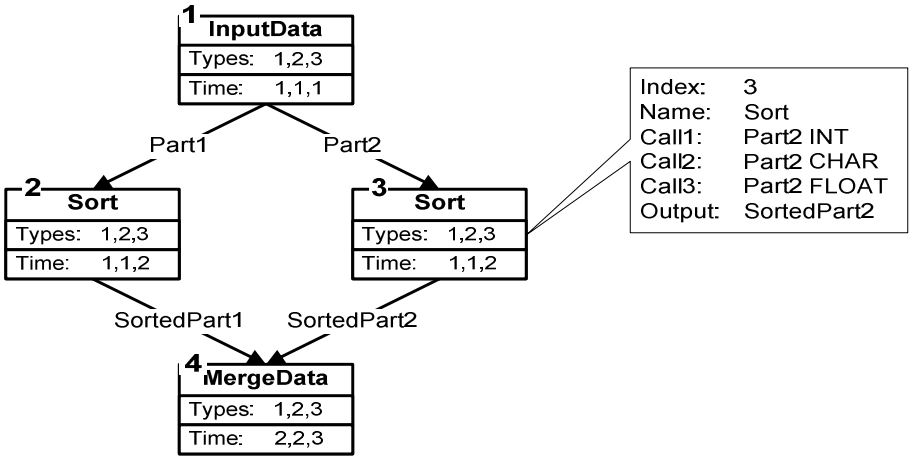

Рис. 1. *Пример графа сценариев операции параллельной сортировки*

назначенные на один и тот же процессор, не могут выполняться одновременно. Два экземпляра одной и той же операции над различными кадрами также не могут выполняться в один и тот же промежуток времени.

Все сценарии обработки представляются в форме направленных ациклических графов (НАГ), где вершины представляют операции обработки данных (гранулы параллелизма), а дуги - отношение предшествования между операциями алгоритма и определяют передачу результатов обработки от источника к приемнику. Вершины помечены перечнем типов данных $d_1, d_2, \ldots, d_k$, которые обрабатываются этой операцией, и соответствующими временами обработки каждого типа $t_1, t_2, \ldots, t_k$, а дуги – значениями объемов передаваемой информации между двумя операциями обработки данных. Связанные дугой операции используют один формат данных, поэтому для всех сценариев и типов кадров дуги имеют равную стоимость.

Построение приложения для параллельной обработки потока данных состоит из нескольких шагов:

1. Создание НАГ сценариев, описывающего логическую структуру программы.

2. Назначение операций обработки вершинам графа и определение параметров вызова для каждого типа данных.

3. Отображение графа сценариев на топологию вычислительного кластера. Параллельная программа представляется декомпозицией $((O),(P)) \longrightarrow \bigcup_{p \in P} (O_p)$, где $\forall O_k, O_p : O_k \neq O_p \supset O_k \bigcap O_p = \varnothing$. Каждое подмножество операций O_p размещается на выбранном узле кластера.

Целью обработки потока является минимизация времени обработки N объектов данных

$$T_{opt} = \min\{\max T_N\}, \quad (1)$$

где T_N означает момент окончания обработки объекта N. Оптимизационный алгоритм использует для оценки расписания имитационную модель, эквивалентную реальному вычислительному процессу.

Пример НАГ сценариев обработки и обозначения элементов графа для параллельной сортировки массивов, содержащих три типа данных, приведен на рис. 1.

Каждая вершина НАГ сценариев обработки данных имеет уникальный индекс, используемый при определении топологии графа. Перечень типов данных, которые обрабатываются этой операцией, одинаков для всех вершин: d_1 =1, d_2 =2, d_3 =3 (поле Types), и соответствующие времена обработки каждого типа t_1 = t_2 = t_3 =1 (поле Time). Эти значения используются при имитационном моделировании и составлении расписания в случае детерминированного потока. В случае обработки стохастического потока временные характеристики определяются в ходе обработки. Каждой вершине графа назначена операция (в поле Name указано ее название), определяющая вычислительную гранулу. Поля вида CallN содержат список параметров, используемых для вызова операции при обработке соответствующего типа данных. В приведенном примере операция сортировки 3 обрабатывает три типа данных (INT, CHAR и FLOAT), содержащиеся в массиве Part2. Поле Output определяет имя переменной, которая содержит результат работы вычислительной гранулы. Это имя также указано на соответствующей дуге, которая соединяет источник и приемник данных.

Созданный НАГ сценариев обработки представляется в форме документа XML, который используется как для визуального представления, так и для выполнения программы средствами организации вычислительного процесса.

Пример соответствующего сценарию рис. 1 файла сценария на XML приведен на рис. 2.

```
<scenario>
  <operation id=”1” name=“InputData”>
        <call id=”1” params=”InputArray INT” />
        <call id=”2” params=” InputArray CHAR” />
        <call id=”3” params=” InputArray FLOAT” />
        <successor id="2" value="1.0" name=”Part1”/>
        <successor id="3" value="1.0" name=”Part2”/>
  </operation>
  <operation id=”2” name=“Sort”>
        <call id=”1” params=”Part1 INT” />
        <call id=”2” params=” Part1 CHAR” />
        <call id=”3” params=” Part1 FLOAT” />
        <successor id="4" value="1.0" name=”SortedPart1”/>
  </operation>
  <operation id=”3” name=“Sort”>
        <call id=”1” params=”Part2 INT” />
        <call id=”2” params=” Part2 CHAR” />
        <call id=”3” params=” Part2 FLOAT” />
        <successor id="4" value="1.0" name=”SortedPart2”/>
  </operation>
  <operation id=”2” name=“MergeData”>
        <call id=”1” params=”Part1 Part2 INT” />
        <call id=”2” params=” Part1 Part2 CHAR” />
        <call id=”3” params=” Part1 Part2 FLOAT” />
  </operation>
</scenario>
```

Рис. 2. *Пример описания сценария на XML*

Каждая вычислительная операция графа реализуется в виде гранулы, использующей интерфейс взаимодействия с платформой для получения исходных данных и сохранения результатов вычислений. При разработке вычислительных гранул преследовалась цель обеспечить быструю интеграцию существующих последовательных алгоритмов обработки информации в структуру параллельного приложения. Для этого алгоритмы на языке C или C++ преобразуются в объекты, формирующие контекст вызова операции на основе переданных ей параметров. Каждая операция интерпретирует строку параметров и преобразует каждый параметр либо в имя переменной, либо в константную величину (строку или число). Порядок следования параметров и их интерпретация определяются спецификацией операции.

Вычислительные операции взаимодействуют со специальным механизмом хранения данных, входящим в архитектуру платформы организации параллельных вычислений. Этот механизм реализует интерфейс доступа к разделяемой памяти, в которой находятся данные и результаты операций. Вариант разделяемой памяти реализован на базе разделяемой файловой системы, которая присутствует в большинстве современных кластеров. В этом случае все переменные, участвующие в обработке, хранятся на общей файловой системе. Механизм доступа обеспечивает запись или чтение переменной с определенным именем, однозначно определяющим ее. Реализован также механизм промежуточного хранения результатов вычислений в локальной памяти каждого вычислительного узла, на котором они были записаны или прочитаны. Это позволяет сократить затраты на получение переменной при повторном обращении к ней. Структура вычислительной гранулы и ее взаимодействие с механизмом хранения данных показано на рис. 3.

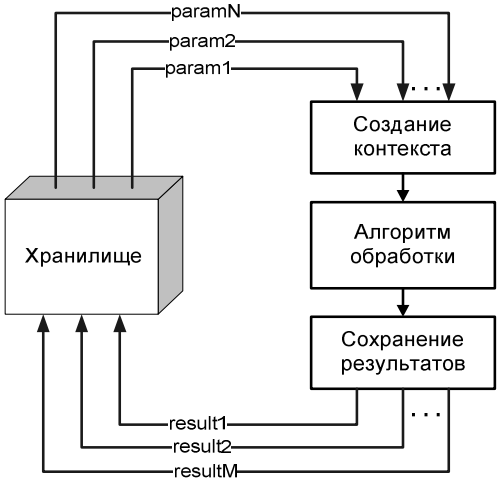

Рис. 3. *Структура вычислительной гранулы и взаимодействие с хранилищем данных*

Указанные при вызове операции параметры (на рис. 3 - param1, param2, paramN) извлекаются из хранилища через операцию Read. Параметр определяется идентификатором объекта данных, представленным в виде строки. Каждый параметр помещается в соответствующую его типу переменную, формируя контекст вызова операции обработки данных. После завершения формирования контекста происходит непосредственное выполнение операции, а затем результаты сохраняются через операцию Write в хранилище данных (на рис. 3 - result1, result2, resultM). С этого момента они становятся доступны другим вычислительным гранулам в составе параллельного приложения.

Вычислительные гранулы объединяются в специализированные библиотеки функций, которые динамически подключаются при выполнении параллельного приложения. Гранула загружается из библиотеки при необходимости ее выполнения и идентифицируется по имени операции. Реализация различных классов алгоритмов обработки информации в виде библиотек вычислительных гранул позволяет расширить сферу применения системы организации параллельных вычислений и дает возможность создавать новые классы приложений.

4. Архитектура платформы организации параллельных вычислений

Общая архитектура платформы показана на рис. 4.

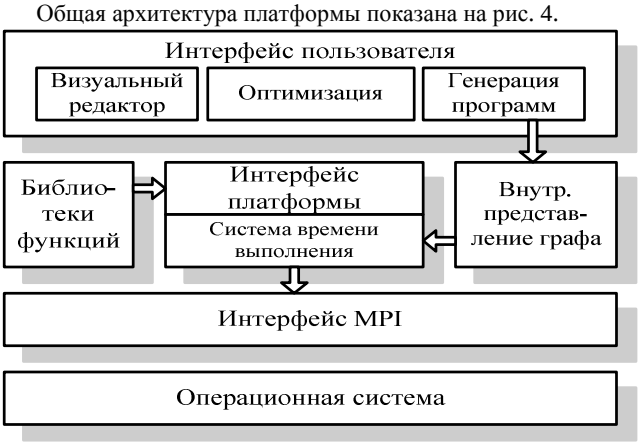

Рис. 4. *Архитектура платформы организации параллельных вычислений*

Интерфейс пользователя представляет собой средство визуального проектирования и анализа параллельной программы обработки потока данных. Визуальный редактор позволяет создать графовую модель программы, а также задать топологию и характеристики вычислительной системы, на которой будет выполняться разработанное приложение. Для анализа детерминированных потоков в редакторе задается шаблон потока в виде перечня типов данных, поступающих на вход приложения. Средства анализа состоят из алгоритма оптимизации расписания и имитационной модели, которая используется для оценки и визуализации расписаний. Интерфейс пользователя позволяет также визуально сравнить две модели расписания, а также корректировать модель вручную. Для оптимизации расписания используется разработанный авторами оригинальный алгоритм виртуальной ассоциативной сети (ВАС) [8], относящийся к классу гибридных генетических алгоритмов (ГА) [9]. Алгоритм использует для оптимизации определенную структуру ассоциативной памяти, содержащую ассоциативные связи. Эта память обучается на основе опыта, накопленного в ходе поиска решения.

Накопление опыта позволяет реализовать направленный поиск в пространстве возможных решений. Этот поиск выполняется значительно быстрее и способен привести к лучшим решениям на ранних стадиях поиска. Размер популяции в алгоритме виртуальной ассоциативной сети меньше (5-10 хромосом), чем в классическом ГА (25-30 хромосом), и требует меньшего времени для оценки.

Созданная в визуальном редакторе программа преобразуется в текстовое представление на языке XML. Это представление содержит информацию для системы времени выполнения, а также отображение графа на топологию вычислительной системы. Файл XML при выполнении программы транслируется во внутреннее представление, позволяющее каждому элементу системы времени выполнения получать необходимую информацию во время работы.

Интерфейс платформы реализует возможности по доступу к данным, хранящимся в хранилище (разделяемой памяти), а также механизмы загрузки гранул из библиотек и определения параметров гранул при вызове. При реализации гранулы

используют этот интерфейс для осуществления взаимодействия в рамках платформы.

Система времени выполнения является многоагентной системой, использующей средства MPI для осуществления координации взаимодействия элементов. Ее архитектура изолирует логику поведения, определяемую сценарием обработки, от базовых средств организации и оптимизации параллельного процесса. Многоагентная система является абстрактным механизмом интерпретации и исполнения графового представления параллельного алгоритма.

Количество агентов-исполнителей при выполнении параллельного приложения равно количеству физических процессоров целевой вычислительной системы плюс один агент-координатор. Взаимодействие между агентами осуществляется с помощью передачи сообщений, реализующих простой протокол взаимодействия. Основные части системы времени выполнения показаны на рис. 5.

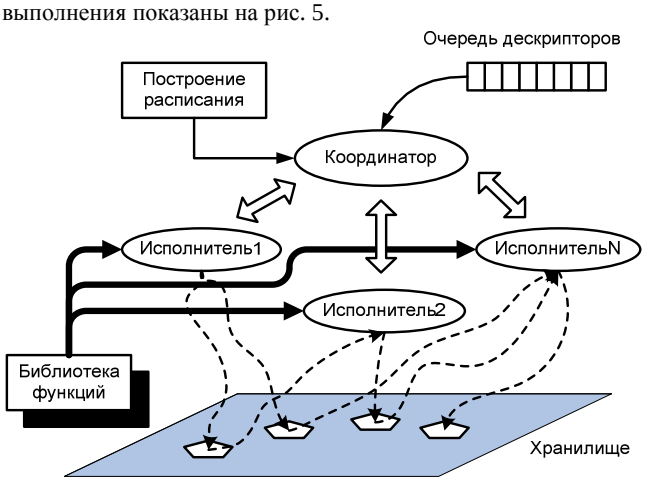

Рис. 5. *Архитектура системы времени выполнения*

Координатор является главным процессом, управляющим реализацией логической структуры параллельного алгоритма. Он содержит очередь дескрипторов обрабатываемых объектов. Каждый дескриптор определяет тип объекта и текущую операцию, которая должна быть выполнена. Для каждой операции создается собственный дескриптор. Очередь дескрипторов представляет собой множество операций, которые готовы к выполнению. Основная задача координатора состоит в передаче дескрипторов операций свободным исполнителям в соответствии с отображением операций графа на топологию системы. Кроме того, координатор следит за завершением операций и помещает в очередь новые дескрипторы в соответствии с топологией графа приложения. Задачей координатора является также реконфигурирование приложения (перемещение некоторых гранул на другие процессоры) для оптимизации процесса выполнения согласно (1). Для ее решения используется модифицированный алгоритм виртуальной ассоциативной сети.

5. Пример приложения и результаты экспериментов.

Предложенные средства организации параллельных вычислений использовались для создания приложений для параллельной обработки изображений слоев интегральных схем. Был реализован ряд вычислительных операций по предобработке изображений, а также операции и сценарии выделения контуров объектов и однородных областей на изображении.

Пример фрагмента графа сценария для операции выделения контуров на основе преобразования Хафа приведен на рис. 6.

Здесь явно указаны четыре параллельных потока для поиска прямых линий, расположенных под определенным углом и выделения точек пересечения для последующего поиска контуров. Операции `Hough` и `Max_Extract` выполняются для всех типов обрабатываемых изображений с одинаковыми временными характеристиками. Время работы остальных операций зависит от типа обрабатываемого кадра.

Для оценки разработанной платформы организации параллельных вычислений были проведены две серии экспериментов. Первая серия проведена для детерминированных потоков, имеющих фиксированное количество объектов с определенной последовательностью типов. Вторая серия экспериментов выполнена для стохастических потоков, когда исходные объекты и их типы генерировались случайным образом. Эксперименты проведены на суперкомпьютере СКИФ К-500, находящемся в ОИПИ НАН Беларуси.

Экспериментальные потоки данных имели нерегулярную структуру и генерировались случайным образом (рис. 7) содержит результаты статической оптимизации для детерминированных потоков, представленные в виде сравнения продолжительности обработки потоков расписаниями, полученными алгоритмами классического ГА и ВАС. На диаграмме указаны результаты для различного количества процессоров (сокращение пр.), участвовавших в обработке.

Результаты свидетельствуют о том, что алгоритм виртуальной сети находит лучшие расписания, при этом производительность алгоритма возрастает с увеличением пространства поиска. Алгоритм виртуальной сети имеет меньшую вычислительную сложность и находит решения быстрее, чем классический ГА.

Во второй серии экспериментов для стохастических потоков использовались полученные в первой серии оптимальные статические расписания, которые сравнивались с расписаниями, реализованными динамически изменяемой конфигурацией приложения. Динамическая оптимизация выполнялась

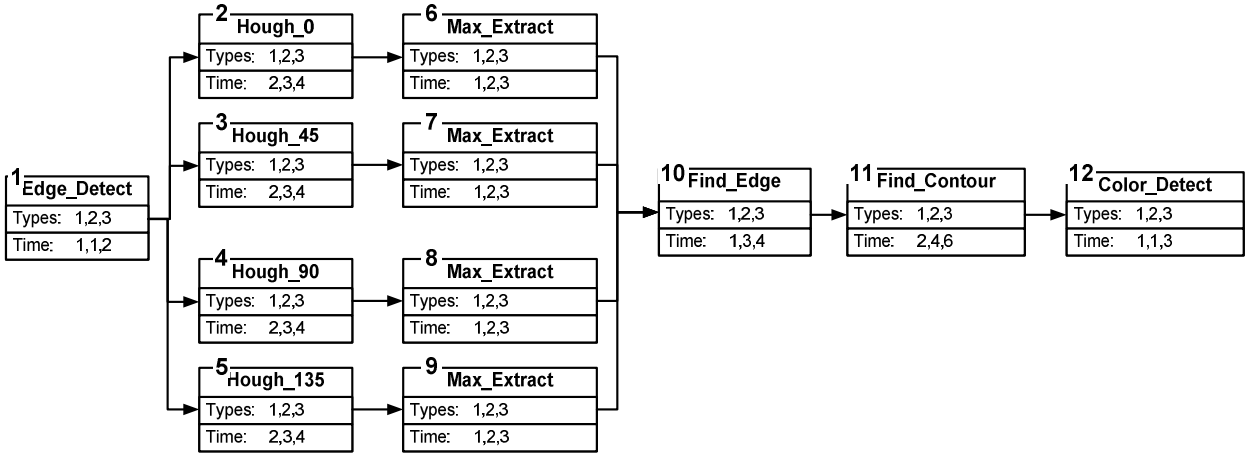

Рис. 6. *Пример сценария для выделения контуров на изображении*

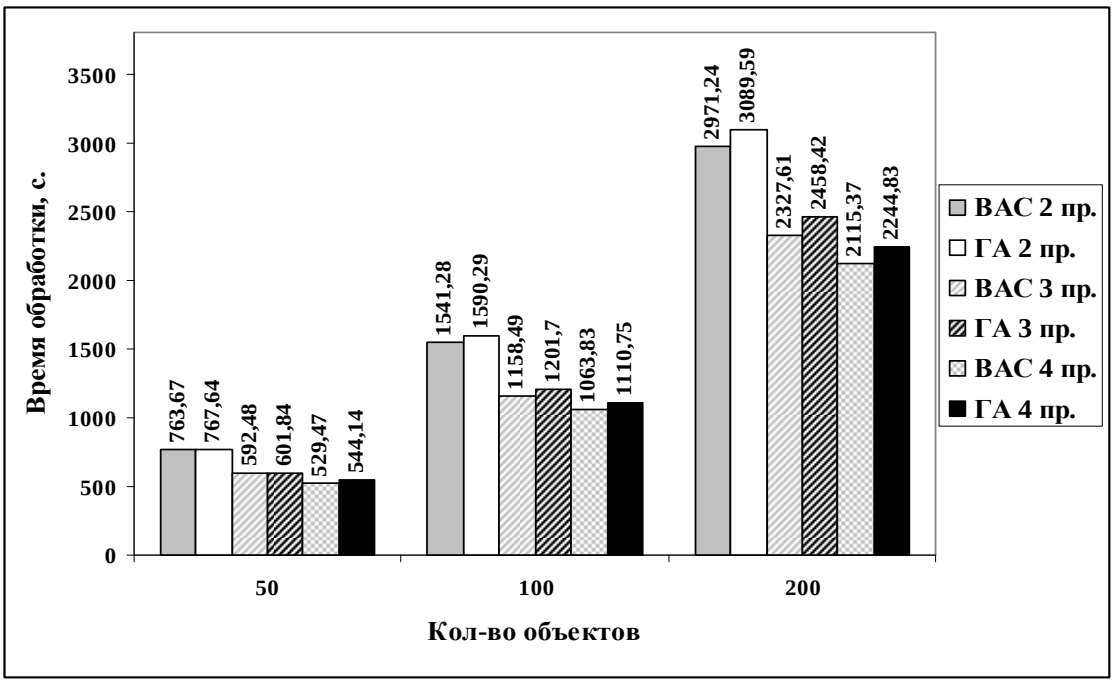

Рис. 7. *Улучшение показателей для статической оптимизации*

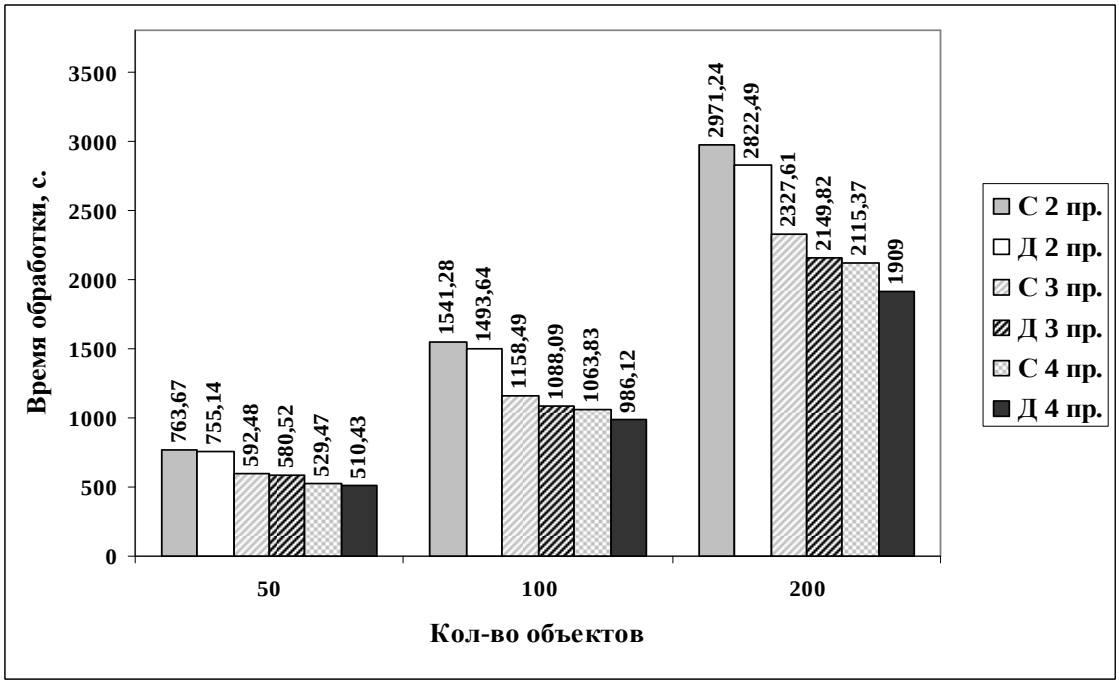

Рис. 8. *Улучшение показателей для динамической оптимизации*

с помощью алгоритма виртуальной ассоциативной сети, (рис. 8) содержит результаты оптимизации, показывающие изменение времени обработки для статического (С) и динамического (Д) алгоритма ВАС.

Результаты показывают, что алгоритм виртуальной сети, управляя системой динамической оптимизации, существенно улучшает показатели производительности в случае обработки стохастических потоков изображений.

Заключение

Реализация параллельных приложений в рамках специализированных компонентных архитектур дает возможность пользователям существенно ускорить процессы создания, анализа и оптимизации программ.

Архитектура средств обработки потоков данных, основанная на использовании многоагентных систем, может быть легко адаптирована для многих приложений, предполагающих параллельную обработку. Использование механизма интерпретации для графового представления алгоритма обработки и динамическое формирование контекста операции дает возможность реализовать гибкую структуру параллельного приложения, позволяющую адаптировать вычислительный процесс к условиям функционирования. Средства проектирования легко расширяются новыми операциями, алгоритмами и типами данных для реализации приложении обработки информационных потоков из различных предметных областей.

СПИСОК ЦИТИРОВАННЫХ ИСТОЧНИКОВ

1. M. Voganti, F. Ercal, C. Dagli, S. Tsunekawa. Automatic PCI Inspection Algorithms: A Survey // Computer Vision and Image Understanding. - 1996. -Vol. 63. - p. 287-313.
2. J. Wickel, P. Alvarado, P. Dörfler et al. Axiom — a modular visual object retrieval system // M. Jarke, J. Koehler, and G. Lakemeyer, editors, KI 2002: Advances in Artificial Intelligence. - LNAI 2479. - Springer, 2002. - p. 253–267.
3. D. Argiro, S. Kubica, M. Young, and S. Jorgensen. Khoros: An integrated development environment for scientific computing and visualization. Whitepaper, Khoral Research, Inc., 1999.
4. W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message Passing Interface*. - MIT Press. - 1995.
5. A. Doudkin, A. V. Inyutin, M. E. Vatkin. Objects identification on the color layout images of the integrated circuit layers // Proceedings of 3rd IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 5-7 September 2005, Sofia, Bulgaria. - IEEE, 2005. - p. 610-614.
6. A. Doudkin, D. A. Vershok. Integrated circuit and photomask images processing technology // J. AMSE. – 2005. - p. 81-88.
7. K. Hwang, Z. Xu. Scalable Parallel Computing – Technology, Architecture, Programming. - McGraw-Hill, USA, 1998.
8. Р. Х. Садыхов, А. В. Отвагин. Алгоритм поиска решений на основе модели виртуальной сети в системах параллельной обработки //Автоматика и вычислительная техника. – №1. – Рига, Латвия. – 2001. – С. 25-33.
9. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы.: Учебное пособие / Под ред. В.М.Курейчика. - М.: Физматлит, 2004.

Статья поступила в редакцию 28.01.2007

УДК [004.5+681.3]:691-419

Разумейчик В.С., Дереченник А.С., Дереченник С.С.

АНАЛИЗ ПОРИСТОСТИ КОМПОЗИЦИОННЫХ МАТЕРИАЛОВ НА ОСНОВЕ ПРОЦЕДУРЫ ИЗОМЕТРИЧЕСКОГО ПОКРЫТИЯ ПОРОВЫХ СЕГМЕНТОВ ЦИФРОВОГО ИЗОБРАЖЕНИЯ

Введение

Пористость является одной из важнейших характеристик композиционных материалов (композитов) и определяет их основные эксплуатационные свойства – влаго- и газопроницаемость, жесткость, прочность и т.д. В материаловедении пористый материал рассматривается как дисперсная структура, в которой поры (пустоты различного размера и формы – сферы, капилляры, микротрещины и т.п.) составляют, в совокупности, дисперсную фазу, неупорядоченно расположенную в дисперсионной среде – твердой фазе. Отдельные поровые пустоты могут сливаться, порождая широкое многообразие конфигураций пространственных кластеров – как прерывистых, изолированных друг от друга, так и перколяционных, т.е. допускающих протекание газа или жидкости из одного кластера в другой. Сложность и неупорядоченность структуры порового пространства, равно как и тесная связь характеристик пористости с важными для практики свойствами материала, обусловливают актуальность экспериментальных и теоретических исследований пористости в широком спектре прикладных задач материаловедения.

Типичными по своей природе капиллярно-пористыми композитами являются материалы на основе цемента – цементный камень, раствор и бетон. Они характеризуются, в общем случае, весьма сложной многоуровневой структурой составляющих материал фаз, в том числе и порового пространства. Оказывая огромное влияние на важнейшие физико-механические свойства формируемого материала, пористость цементных композитов в то же время очень чувствительна к технологическим факторам. В связи с этим изучение пористости цементных композитов необходимо как для обеспечения качества строительных изделий, так и при разработке новых технологий создания строительных материалов с заданными свойствами.

Характеристики и методы исследования пористости цементных композитов

Согласно одной из классификаций, по степени дисперсности (т.е. размерам, или эффективным радиусам), поры цементного камня и бетона делят на группы [1]:

- микрокапилляры (радиус менее 0,1 мкм), которые могут заполняться за счет сорбции паров из окружающей среды и образования пленок на стенках;
- макрокапилляры (мезопоры между частицами геля радиусом от 0,1 до 1…10 мкм), заполняющиеся жидкостью только при непосредственном контакте с ней;
- некапиллярные (технологические) поры – дефекты структуры, которые могут быть связаны с вовлеченным или защемленным воздухом, раковинами и т.п.

В зависимости от масштабного уровня исследования, иногда дополнительно различают также ультра-микропоры (так называемые гелевые поры с надмолекулярным уровнем дисперсности 0,5…5 нм) и переходные микропоры (5…10 нм).

Структура порового пространства, в первом приближении, характеризуется объемом пор (интегральные параметры), а также их размерами и удельной поверхностью (дифференциальные параметры).

Основной интегральный параметр – истинная (полная) пористость, определяемая как доля суммарного объема порового пространства материала. Эмпирические методы исследования позволяют, как правило, найти лишь открытую (кажущуюся, или доступную) пористость, т.е. суммарный объем всех пор, соединяющихся между собой и с поверхностью материала, и потому доступных для определения данным экспериментальным методом. При этом не учитывается т.н. условно замкнутая пористость – объем изолированных от поверхности образца пор и поровых кластеров.

Наиболее употребительные дифференциальные параметры:

- функция распределения объема, поверхности или количества пор по размерам;
- различного рода условные размеры пор: средний, эффективный, гидравлический, максимальный и т.п. радиусы;
- геометрические характеристики пор: форма, взаимное

***Разумейчик Вита Станиславовна**, ассистент кафедры «ЭВМ и системы» БрГТУ, vita_r@tut.by.*
***Дереченник Анна Станиславовна,** аспирант кафедры «ЭВМ и системы» БрГТУ, ann_derechennik@tut.by.*
***Дереченник Станислав Станиславович,** к.т.н., доцент, заведующий кафедрой «ЭВМ и системы» БрГТУ, chief.cm@bstu.by.*
Беларусь, Брестский государственный технический университет, 224017, г. Брест, ул. Московская, 267.