

ГЛАВА 13

ПРОГРАММНАЯ СИСТЕМА ПараЛаб ДЛЯ ИССЛЕДОВАНИЯ МЕТОДОВ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

13.1. Введение

Программная система **Параллельная Лаборатория** (сокращенное наименование **ПараЛаб**) обеспечивает возможность проведения вычислительных экспериментов с целью изучения и исследования параллельных алгоритмов решения сложных вычислительных задач. Система может быть использована для организации лабораторного практикума по различным учебным курсам в области параллельного программирования, в рамках которого обеспечивается возможность

- *моделирования многопроцессорных и многоядерных вычислительных систем с различной топологией сети передачи данных;*
- *получения визуального представления о вычислительных процессах и операциях передачи данных, происходящих при параллельном решении разных вычислительных задач;*
- *построения оценок эффективности изучаемых методов параллельных вычислений.*

Проведение такого практикума может быть организовано на «обычных» однопроцессорных компьютерах, работающих под управлением операционных систем MS Windows (режим многозадачной имитации параллельных вычислений). Кроме режима имитации, в системе ПараЛаб может быть обеспечен удаленный доступ к имеющейся многопроцессорной вычислительной системе для выполнения экспериментов в режиме «настоящих» параллельных вычислений для сопоставления результатов имитации и реальных расчетов.

В целом система ПараЛаб представляет собой *интегрированную среду для изучения и исследования параллельных алгоритмов* решения сложных вычислительных задач. Широкий набор имеющихся средств визуализации процесса выполнения эксперимента и анализа полученных результатов позволяет изучить эффективность использования тех или иных алгоритмов на разных вычислительных системах, сделать выводы о масштабируемости

алгоритмов и определить возможное ускорение процесса параллельных вычислений.

Реализуемые системой ПараЛаб процессы изучения и исследований ориентированы на активное освоение основных теоретических положений и способствуют формированию у пользователей своих собственных представлений о моделях и методах параллельных вычислений путем наблюдения, сравнения и сопоставления широкого набора различных визуальных графических форм, демонстрируемых в ходе выполнения вычислительного эксперимента.

Основной сферой использования системы ПараЛаб является *учебное применение* студентами и преподавателями вузов для исследований и изучения параллельных алгоритмов решения сложных вычислительных задач в рамках лабораторного практикума по различным учебным курсам в области параллельного программирования. Система может использоваться также и при проведении *научных исследований* для оценки эффективности параллельных вычислений.

Пользователи, начинающие знакомиться с проблематикой параллельных вычислений, найдут систему ПараЛаб полезной для освоения методов параллельного программирования, опытные вычислители могут использовать систему для оценки эффективности новых разрабатываемых параллельных алгоритмов.

13.2. Общая характеристика системы

ПараЛаб – программная система, которая позволяет проводить как реальные параллельные вычисления на многопроцессорной вычислительной системе, так и имитировать такие эксперименты на одном последовательном компьютере с визуализацией процесса параллельного решения сложной вычислительной задачи.

При проведении имитационных экспериментов ПараЛаб предоставляет возможность для пользователя:

- *определить топологию* параллельной вычислительной системы для проведения экспериментов, *задать число процессоров и ядер* в этой топологии, *установить производительность* процессоров, *выбрать характеристики коммуникационной среды и способ коммуникации*;
- *осуществить постановку вычислительной задачи*, для которой в составе системы ПараЛаб имеются реализованные параллельные алгоритмы решения, *выполнить задание параметров задачи*;
- *выбрать параллельный метод* для решения выбранной задачи;
- *установить параметры визуализации* для выбора желаемого темпа

демонстрации, способа отображения пересылаемых между процессорами данных, степени детальности визуализации выполняемых параллельных вычислений;

- *выполнить эксперимент* для параллельного решения выбранной задачи; при этом в системе ПараЛаб может быть сформировано несколько различных заданий для проведения экспериментов с отличающимися типами многопроцессорных систем, задач или методов параллельных вычислений, для которых выполнение эксперимента может происходить одновременно (в режиме разделения времени); одновременное выполнение эксперимента для нескольких заданий позволяет наглядно сравнивать динамику решения задачи различными методами, на разных топологиях, с разными параметрами исходной задачи; при выполнении *серии экспериментов*, требующих длительных вычислений, в системе имеется возможность их проведения в автоматическом режиме с запоминанием результатов в *журнале экспериментов* для организации последующего анализа полученных данных;

- *накапливать и анализировать результаты выполненных экспериментов*; по запомненным результатам в системе имеется возможность построения графиков, характеризующих параллельные вычисления, зависимостей (*времени решения, ускорения*) от параметров задачи и вычислительной системы.

Одной из важнейших характеристик системы является возможность выбора способов проведения экспериментов. Эксперимент может быть выполнен в *режиме имитации*, т. е. проведен на одном процессоре без использования каких-либо специальных программных средств типа библиотек передачи сообщений. Кроме того, в рамках системы ПараЛаб обеспечивается возможность следующих способов проведения *реального вычислительного эксперимента*:

- *на одном компьютере*, где имеется библиотека передачи сообщений MPI (выполнение эксперимента в режиме разделения времени); для данной библиотеки имеются общедоступные реализации, которые могут быть получены в сети Интернет и установлены на компьютере под управлением операционных систем MS Windows,

- *на реальной многопроцессорной кластерной вычислительной системе*,

- *в режиме удаленного доступа* к вычислительному кластеру.

При построении зависимостей временных характеристик от параметров задачи и вычислительной системы для экспериментов, выполненных в режиме имитации, используются теоретические оценки в соответствии с имеющимися моделями параллельных вычислений (см. например,

[8,10,67]). Для реальных экспериментов на многопроцессорных вычислительных системах зависимости строятся по набору результатов проведенных вычислительных экспериментов.

Важно отметить, что в системе ПараЛаб обеспечена возможность ведения *журнала экспериментов* для запоминания результатов выполненных экспериментов. Запомненные результаты позволяют выполнить анализ полученных данных; по имеющейся в журнале информации любой из проведенных ранее экспериментов может быть восстановлен для повторного выполнения или продолжения расчетов.

Реализованные таким образом процессы изучения и исследований позволяют освоить теоретические положения и помогут формированию представлений о методах построения параллельных алгоритмов, ориентированных на решение конкретных прикладных задач.

Демонстрационный пример

Для выполнения примера, имеющегося в комплекте поставки системы:

- выберите пункт меню **Начало** и выполните команду **Загрузить**;
- выберите строку **first.prl** в списке имен файлов и нажмите кнопку **Открыть**;
- выберите пункт меню **Выполнение** и выполните команду **В активном окне**.

В результате выполненных действий на экране дисплея будет представлено окно для выполнения вычислительного эксперимента (рис. 13.1). В этом окне демонстрируется решение задачи умножения матриц при помощи ленточного алгоритма.

В области «Выполнение эксперимента» представлены данные, которые обрабатывают процессоры в каждый момент выполнения алгоритма (для ленточного алгоритма умножения матриц – это несколько последовательных строк матрицы A и несколько последовательных столбцов матрицы B), а также структура линий коммутации. При помощи динамически перемещающихся прямоугольников (пакетов) желтого цвета изображается обмен данными, который осуществляют процессоры.

В области «Результат умножения матриц» изображается текущее состояние матрицы – результата умножения. Поскольку результатом перемножения полос исходных матриц A и B является блок матрицы C , получаемая результирующая матрица имеет блочную структуру. Темно-синим цветом обозначены уже вычисленные блоки, голубым цветом выделены блоки, еще подлежащие определению.

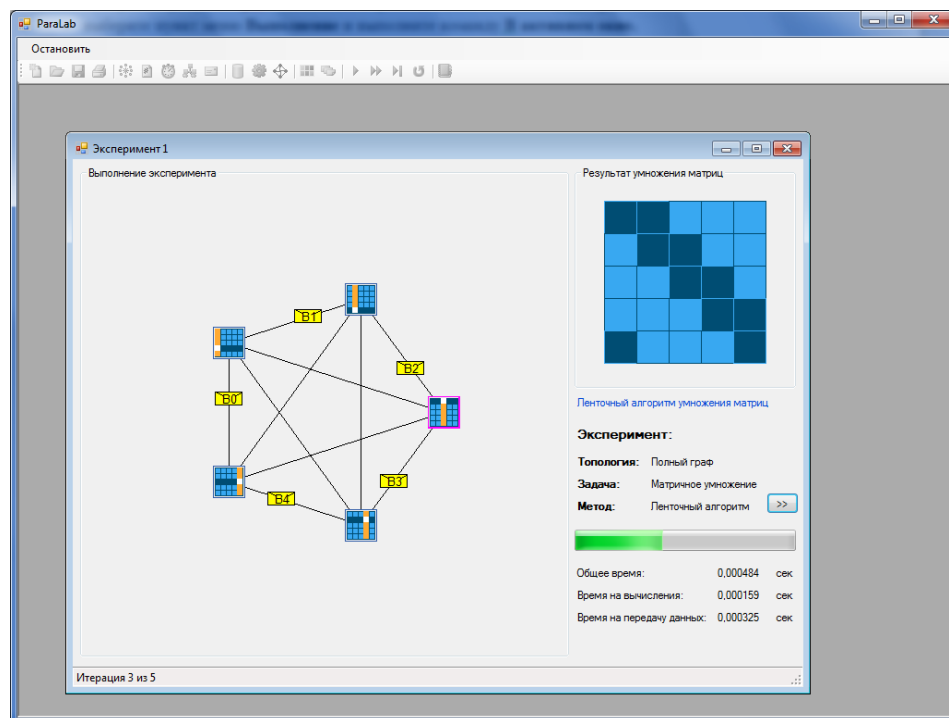


Рис. 13.1. Окно вычислительного эксперимента

В области «Эксперимент» отображены основные параметры выполняемого эксперимента. К таким параметрам относятся топология, решаемая задача, а также метод ее решения. Ленточный индикатор области «Эксперимент» отображает текущую стадию выполнения алгоритма. В строках «Общее время», «Время на вычисления» и «Время на передачу данных» представлены временные характеристики алгоритма.

После выполнения эксперимента (восстанавливается главное меню системы) можно завершить работу системы. Для этого выберите пункт меню **Архив** и выполните команду **Выход**.

13.3. Формирование модели вычислительной системы

Для формирования модели вычислительной системы необходимо определить топологию сети, количество процессоров и ядер на них, производительность каждого процессора и характеристики коммуникационной среды (латентность, пропускную способность и метод передачи данных). Следует отметить, что в рамках системы ПараЛаб вычислительная система

полагается однородной, т. е. все процессоры обладают одинаковой производительностью, а все каналы связи имеют одинаковые характеристики.

13.3.1. Выбор топологии сети

Топология сети передачи данных определяется структурой линий коммутации между процессорами вычислительной системы. В системе ПараЛаб обеспечивается поддержка следующих типовых топологий:

- **полный граф** (*completely-connected graph* или *clique*) – система, в которой между любой парой процессоров существует прямая линия связи; как результат, данная топология обеспечивает минимальные затраты при передаче данных, однако является сложно реализуемой при большом количестве процессоров;
- **линейка** (*linear array* или *farm*) – система, в которой каждый процессор имеет линии связи только с двумя соседними (с предыдущим и последующим) процессорами; такая схема является, с одной стороны, просто реализуемой, а с другой стороны, соответствует структуре передачи данных при решении многих вычислительных задач (например, при организации конвейерных вычислений);
- **кольцо** (*ring*) – данная топология получается из линейки процессоров соединением первого и последнего процессоров линейки;
- **звезда** (*star*) – система, в которой выделенный центральный процессор, у которого с любым процессором существует прямая линия связи; все остальные процессы имеют связь только с центральным; как результат, данная топология обеспечивает малые затраты при реализации, но при большом количестве процессоров вызывает очень большую нагрузку на центральный процессор;
- **решетка** (*mesh*) – система, в которой граф линий связи образует прямоугольную двумерную сетку; подобная топология может быть достаточно просто реализована и, кроме того, может быть эффективно используется при параллельном выполнении многих численных алгоритмов (например, при реализации методов блочного умножения матриц);
- **гиперкуб** (*hypercube*) – данная топология представляет частный случай структуры N -мерной решетки, когда по каждой размерности сетки имеется только два процессора (т. е. гиперкуб содержит 2^N процессоров при размерности N); данный вариант организации сети передачи данных достаточно широко распространен в практике и характеризуется следующим рядом отличительных признаков:
 - два процессора имеют соединение, если двоичное представление их номеров имеет только одну различающуюся позицию;

- в N -мерном гиперкубе каждый процессор связан ровно с N соседями;
- N -мерный гиперкуб может быть разделен на два $(N-1)$ -мерных гиперкуба (всего возможно N различных таких разбиений);
- кратчайший путь между двумя любыми процессорами имеет длину, совпадающую с количеством различающихся битовых значений в номерах процессоров (данная величина известна как *расстояние Хэмминга*).

Правила использования системы ПараЛаб

1. Запуск системы. Для запуска системы ПараЛаб выделите пиктограмму системы и выполните двойной щелчок левой кнопкой мыши (или нажмите клавишу **Enter**). Далее выполните команду **Выполнить новый эксперимент** (пункт меню **Начало**) и нажмите в диалоговом окне **Название эксперимента** кнопку **ОК** (при желании до нажатия кнопки **ОК** может быть изменено название создаваемого окна для проведения экспериментов).

2. Выбор топологии вычислительной системы. Для выбора топологии вычислительной системы следует выполнить команду **Топология** пункта меню **Система**. В появившемся диалоговом окне (рис. 13.2) щелкните левой клавишей мыши на пиктограмме нужной топологии или внизу в области соответствующей круглой кнопки выбора (радиокнопки). При нажатии кнопки **Справка** можно получить справочную информацию о реализованных топологиях. Нажмите кнопку **ОК** для подтверждения выбора и кнопку **Отмена** для возврата в основное меню системы ПараЛаб.

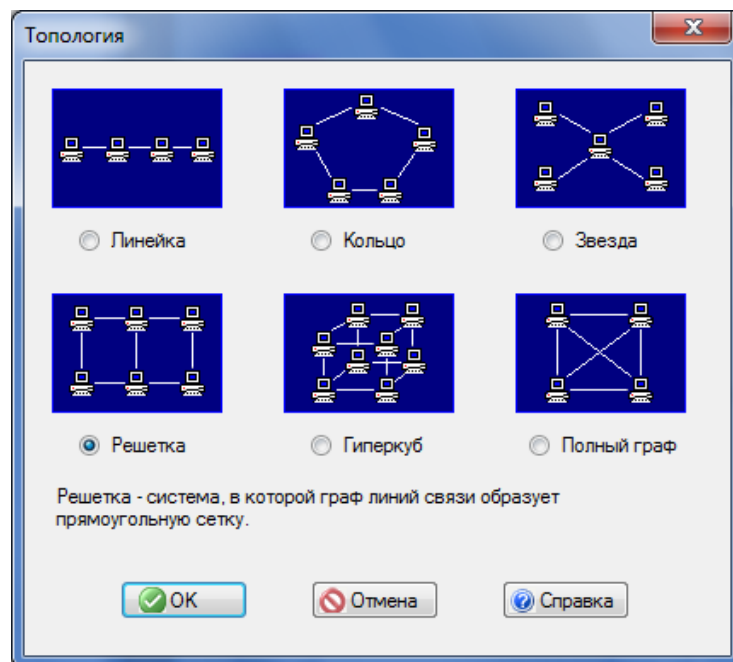


Рис. 13.2. Диалоговое окно для выбора топологии

13.3.2. Задание количества процессоров и ядер

Для выбранной топологии система ПараЛаб позволяет установить необходимое количество процессоров и ядер. Выполняемый при этом выбор конфигурации системы осуществляется в соответствии с типом используемой топологии. Так, число процессоров в двумерной решетке должно являться полным квадратом (размеры решетки по горизонтали и вертикали совпадают), а число процессоров в гиперкубе – степенью числа 2. В данной реализации системы ядер на процессоре может быть одно, два или четыре.

Под *производительностью процессора (каждого ядра)* в системе ПараЛаб понимается количество операций с плавающей запятой, которое процессор может выполнить за секунду (floating point operations per second – flops). Важно отметить, что при построении оценок времени выполнения эксперимента предполагается, что все машинные команды являются одинаковыми и соответствуют одной и той же операции с плавающей точкой.

Правила использования системы ПараЛаб

1. Задание количества процессоров и ядер. Для выбора числа процессоров и ядер на процессоре необходимо выполнить команду **Количество Процессоров** пункта меню **Система**. В появившемся диалоговом окне (рис. 13.3) Вам предоставляется несколько пиктограмм со схематическим изображением числа узлов вычислительной системы. Для каждого узла есть возможность задать количество процессоров и ядер на процессоре. Для выбора щелкните левой клавишей мыши на нужной пиктограмме. Пиктограмма, соответствующая текущему числу узлов, выделена яркосиним цветом.

Нажмите кнопку **ОК** для подтверждения выбора или кнопку **Отмена** для возврата в основное меню системы ПараЛаб без изменения числа процессоров.

2. Определение производительности ядер процессора. Для задания производительности ядер процессоров, составляющих многопроцессорную вычислительную систему, следует выполнить команду **Производительность Процессоров** пункта меню **Система**. Далее в появившемся диалоговом окне (рис. 13.4) при помощи бегунка или поля ввода задать величину производительности. Для подтверждения выбора нажмите кнопку **ОК** (или клавишу **Enter**). Для возврата в основное меню системы ПараЛаб без изменений нажмите кнопку **Отмена** (или клавишу **Escape**).

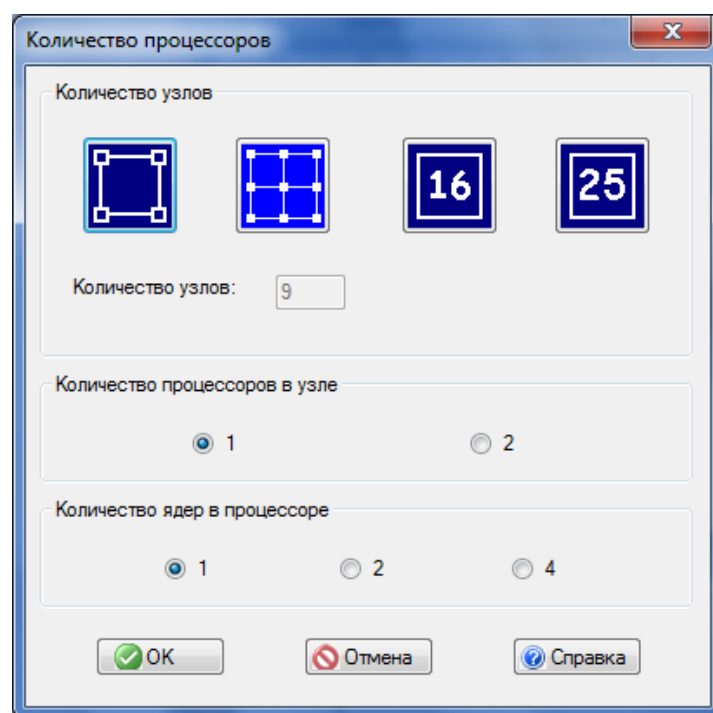


Рис. 13.3. Диалоговое окно для задания числа процессоров

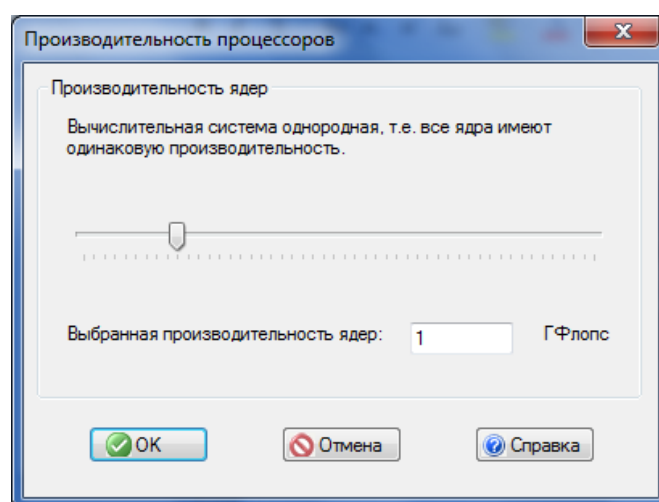


Рис. 13.4. Диалоговое окно для задания производительности процессора

13.3.3. Задание характеристик сети

Время передачи данных между процессорами определяет коммуникационную составляющую (*communication overhead*) длительности выполнения параллельного алгоритма в многопроцессорной вычислительной системе. Основной набор параметров, описывающих время передачи данных, состоит из следующего ряда величин:

- *латентность* (α) – время начальной подготовки, которое характеризует длительность подготовки сообщения для передачи, поиска маршрута в сети и т. п.;

- *пропускная способность сети* (β) – определяется как максимальный объем данных, который может быть передан за некоторую единицу времени по одному каналу передачи данных. Данная характеристика измеряется, например, количеством переданных бит в секунду.

К числу реализованных в системе ПараЛаб методов передачи данных относятся следующие два широко известных способа коммуникации (см. [10,72]). Первый из них ориентирован на *передачу сообщений* (МПС) как неделимых (атомарных) блоков информации (*store-and-forward routing* или *SFR*). При таком подходе процессор, содержащий исходное сообщение, готовит весь объем данных для передачи, определяет транзитный процессор, через который данные могут быть доставлены целевому процессору, и запускает операцию пересылки данных. Процессор, которому направлено сообщение, в первую очередь осуществляет прием полностью всех пересылаемых данных и только затем приступает к пересылке принятого сообщения далее по маршруту. Время пересылки данных T для метода передачи сообщения размером m по маршруту длиной l определяется выражением:

$$T = \alpha + m/\beta \cdot l.$$

Второй способ коммуникации основывается на представлении пересылаемых сообщений в виде блоков информации меньшего размера (*пакетов*), в результате чего передача данных может быть сведена к *передаче пакетов* (МПП). При таком методе коммуникации (*cut-through routing* or *CTR*) транзитный процессор может осуществлять пересылку данных по дальнейшему маршруту непосредственно сразу после приема очередного пакета, не дожидаясь завершения приема данных всего сообщения. Количество передаваемых при этом пакетов равно

$$n = \left\lceil \frac{m}{V - V_0} \right\rceil + 1,$$

где V есть размер пакета, а величина V_0 определяет объем служебных данных, передаваемых в каждом пакете (*заголовок пакета*). Как результат, время передачи сообщения в этом случае составит

$$T = \alpha + \frac{V}{\beta} \left(l + \left\lceil \frac{m}{V - V_0} \right\rceil \right) = \alpha + \frac{V}{\beta} l + n - 1$$

(скобки $\lceil \cdot \rceil$ обозначают операцию приведения к целому с избытком).

Сравнивая полученные выражения, можно заметить, что в случае, когда длина маршрута больше единицы, метод передачи пакетов приводит к более быстрой пересылке данных; кроме того, данный подход снижает потребность в памяти для хранения пересылаемых данных для организации приема-передачи сообщений, а для передачи пакетов могут использоваться одновременно разные коммуникационные каналы.

Правила использования системы ПараЛаб

1. Определение характеристик коммуникационной среды. Для определения характеристик сети выполните команду **Характеристики сети** пункта меню **Система**. В открывшемся диалоговом окне (рис. 13.5) при помощи бегунков можно задать время начальной подготовки данных (латентность) в микросекундах и пропускную способность каналов сети (Мбайт/с). Для подтверждения выбора нажмите кнопку **ОК**. Для возврата в основное меню системы ПараЛаб без изменения этих параметров нажмите кнопку **Отмена**.

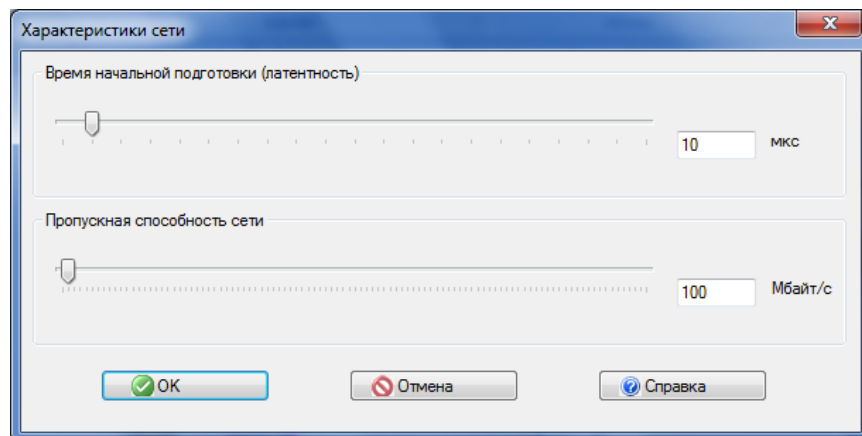


Рис. 13.5. Диалоговое окно для задания характеристик сети

2. Определение метода передачи данных. Для определения метода передачи данных, который будет использоваться при проведении вычисли-

тельного эксперимента и при построении временных характеристик, необходимо выполнить команду **Метод Передачи Данных** пункта меню **Система**. В открывшемся диалоговом окне (рис. 13.6) следует щелкнуть левой клавишей мыши в области радиокнопки, которая соответствует желаемому методу передачи данных. Если выбран метод передачи пакетов, при помощи бегунков возможно задать длину пакета и длину заголовка пакета в байтах. Для подтверждения выбора метода передачи данных и его параметров нажмите кнопку **ОК**.

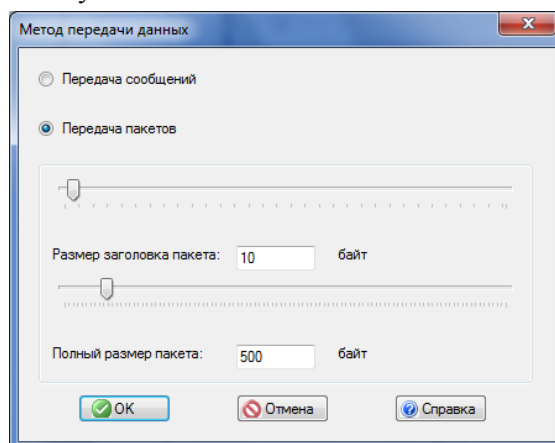


Рис. 13.6. Диалоговое окно для задания метода передачи данных

3. Завершение работы системы. Для завершения работы системы ПараЛаб следует выполнить команду **Завершить** (пункт меню **Архив**).

13.4. Постановка вычислительной задачи и выбор параллельного метода решения

Для параллельного решения тех или иных вычислительных задач процесс вычислений должен быть представлен в виде набора независимых вычислительных процедур, допускающих выполнение на независимых процессорах.

Общая схема организации таких вычислений может быть представлена следующим образом:

- разделение процесса вычислений на части, которые могут быть выполнены одновременно,
- распределение вычислений по процессорам,
- обеспечение взаимодействия параллельно выполняемых вычислений.

Возможные способы получения методов параллельных вычислений:

- разработка новых параллельных алгоритмов,
- распараллеливание последовательных алгоритмов.

Условия эффективности параллельных алгоритмов:

- равномерная загрузка процессоров (отсутствие простоев),
- низкая интенсивность взаимодействия процессоров (независимость).

В системе ПараЛаб реализованы широко применяемые параллельные алгоритмы для решения ряда сложных вычислительных задач из разных областей научно-технических приложений: алгоритмы сортировки данных, матричных операций, решения систем линейных уравнений, решения дифференциальных уравнений, обработки графов и многоэкстремальной оптимизации.

Правила использования системы ПараЛаб

1. Выбор задачи. Для выбора задачи из числа реализованных в системе выберите пункт меню **Задача** и выделите левой клавишей мыши одну из строк: **Умножение матрицы на вектор**, **Матричное умножение**, **Решение систем лин. уравнений**, **Сортировка**, **Обработка графов**, **Решение диффер. уравнений**, **Многоэкстремальная оптимизация**. Выбранная задача станет текущей в активном окне

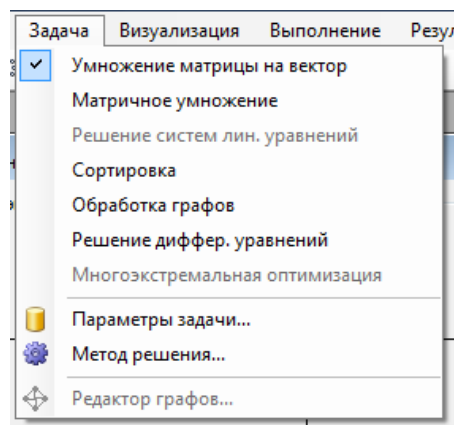


Рис. 13.7. Выбор задачи

Следует отметить, что решение отдельных классов задач может быть выполнено только при использовании вполне определенных топологиях вычислительной системы (так, например, решение задач многоэкстремаль-

ной оптимизации возможно только при топологии полного графа). Сводка соответствия классов задач и топологий приводится в приложении к данной главе; см. также файл *MethodsTopologies* в каталоге *Doc* комплекта поставки системы ПараЛаб.

2. Определение параметров задачи. Основным параметром задачи в системе ПараЛаб является объем исходных данных. Для задачи сортировки – это размер упорядочиваемого массива данных, для матричных операций и задачи решения систем линейных уравнений – размерность исходных матриц, для задачи обработки графов – число вершин в графе. Для выбора параметров задачи необходимо выполнить команду **Параметры задачи** пункта меню **Задача**. В появившемся диалоговом окне (рис. 13.8) следует при помощи бегунка задать необходимый объем исходных данных. Нажмите **ОК** (Enter) для подтверждения задания параметра. Для возврата в основное меню системы ПараЛаб без сохранения изменений нажмите **Отмена** (Escape).

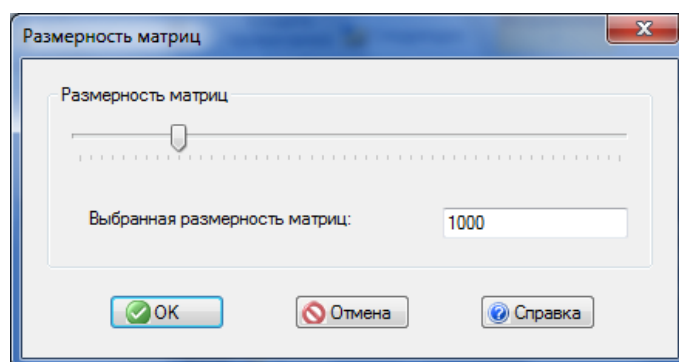


Рис. 13.8. Диалоговое окно задания параметров задачи в случае решения задачи матричного умножения

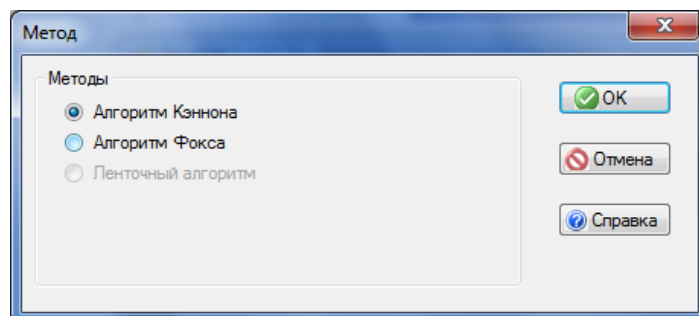


Рис. 13.9. Диалоговое окно выбора метода в случае решения задачи матричного умножения

3. Определение метода решения задачи. Для выбора метода решения задачи выполните команду **Метод решения** пункта меню **Задача**. В появившемся диалоговом окне (рис. 13.9) выделите мышью нужный метод, нажмите **ОК** для подтверждения выбора, нажмите **Отмена** для возврата в основное меню системы ПараЛаб.

13.4.1. Умножение матрицы на вектор

В результате *умножения матрицы A размерности $m \times n$ и вектора b* , состоящего из n элементов, получается вектор c размера m , каждый i -й элемент которого есть результат скалярного умножения i -й строки матрицы A (обозначим эту строчку a_i) и вектора b :

$$c_i = a_i, b = \sum_{j=1}^n a_{ij} b_j, 1 \leq i \leq m.$$

Тем самым получение результирующего вектора c предполагает повторение m однотипных операций по умножению строк матрицы A и вектора b . Каждая такая операция включает умножение элементов строки матрицы и вектора b (n операций) и последующее суммирование полученных произведений ($n-1$ операций). Общее количество необходимых скалярных операций есть величина

$$T_1 = m \cdot 2n - 1.$$

Для операции умножения матрицы на вектор характерным является повторение одних и тех же вычислительных действий для разных элементов матриц. Данный момент свидетельствует о наличии *параллелизма по данным* при выполнении матричных расчетов и, как результат, распараллеливание матричных операций сводится в большинстве случаев к разделению обрабатываемых матриц между процессорами используемой вычислительной системы. Выбор способа разделения матриц приводит к определению конкретного метода параллельных вычислений; существование разных схем распределения данных порождает целый ряд *параллельных алгоритмов матричных вычислений*.

Наиболее общие и широко используемые способы разделения матриц состоят в разбиении данных на *полосы* (по вертикали или горизонтали) или на прямоугольные фрагменты (*блоки*).

Более полная информация об алгоритмах матрично-векторного умножения, реализованных в системе ПараЛаб, содержится в главе 6.

1. Умножение матрицы на вектор при разделении данных по строкам. Данный алгоритм основан на представлении матрицы непрерывными наборами (горизонтальными полосами) строк. Полученные полосы

распределяются по процессорам вычислительной системы. Вектор b копируется на все процессоры. Перемножение полосы матрицы на вектор (а выполнение процессорами этой операции может быть выполнено параллельно) приводит к получению блока элементов результирующего вектора c . Для объединения результатов расчета и получения полного вектора c на каждом из процессоров вычислительной системы необходимо выполнить операцию обобщенного сбора данных.

Задания и упражнения

1. Создайте в системе ПараЛаб новое окно вычислительного эксперимента. Для этого окна выберите задачу умножения матрицы на вектор (щелкните левой кнопкой мыши на строке **Умножение матрицы на вектор** пункта меню **Задача**).
2. Откройте диалоговое окно выбора топологии и выберите топологию полный граф.
3. Откройте диалоговое окно выбора метода и убедитесь в том, что выбран метод, основанный на горизонтальном разбиении матрицы.
4. Проведите несколько вычислительных экспериментов. Изучите зависимость времени выполнения алгоритма от объема исходных данных и от количества узлов, процессоров и ядер.

2. Умножение матрицы на вектор при разделении данных по столбцам. Другой подход к параллельному умножению матрицы на вектор основан на разделении исходной матрицы на непрерывные наборы (вертикальные полосы) столбцов. Вектор b при таком подходе разделен на блоки. Вертикальные полосы исходной матрицы и блоки вектора распределены между процессорами вычислительной системы.

Параллельный алгоритм умножения матрицы на вектор начинается с того, что каждый процессор i выполняет умножение своей вертикальной полосы матрицы A на блок элементов вектора b , в итоге на каждом процессоре получается блок вектора промежуточных результатов $c'(i)$. Далее для получения элементов результирующего вектора c процессоры должны обмениваться своими промежуточными данными между собой.

3. Умножение матрицы на вектор при блочном разделении данных. В ПараЛаб реализован еще один параллельный алгоритм умножения матрицы на вектор, который основан на ином способе разделения данных – на разбиении матрицы на прямоугольные фрагменты (*блоки*). При таком способе разделения данных исходная матрица A представляется в виде набора прямоугольных блоков. Вектор b также должен быть разделен на блоки. Блоки мат-

рицы и блоки вектора распределены между процессорами вычислительной системы. Логическая (виртуальная) топология вычислительной системы в данном случае имеет вид прямоугольной двумерной решетки. Размеры процессорной решетки соответствуют количеству прямоугольных блоков, на которые разбита матрица A . На процессоре p_{ij} , расположенном на пересечении i -й строки и j -го столбца процессорной решетки располагается блок A_{ij} матрицы A и блок b_j вектора b .

После перемножения блоков матрицы A и вектора b каждый процессор p_{ij} будет содержать вектор частичных результатов $c'(i,j)$. Поэлементное суммирование векторов частичных результатов для каждой горизонтальной строки процессорной решетки позволяет получить результирующий вектор c .

Задания и упражнения

1. Запустите систему ПараЛаб. Установите количество процессоров, равное четырем.
2. Выполните три последовательных эксперимента с использованием трех различных алгоритмов умножения матрицы на вектор – алгоритмов, основанных на горизонтальном, вертикальном и блочном разбиении матрицы. Сравните временные характеристики алгоритмов, которые отображаются в правой нижней части окна.
3. Измените объем исходных данных (выполните команду **Параметры задачи** пункта меню **Задача**). Снова проведите эксперименты. Сравните временные характеристики алгоритмов.
4. Измените количество процессоров, установите количество узлов, равное 16 (выполните команду **Количество процессоров** пункта меню **Система**). Проведите вычислительные эксперименты и сравните временные характеристики. Сделайте выводы о сравнительной эффективной методов матрично-векторного умножения.

13.4.2. Матричное умножение

Задача умножения матрицы на матрицу определяется соотношениями:

$$c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}, 1 \leq i, j \leq n$$

(для простоты изложения материала будем предполагать, что перемножаемые матрицы A и B являются квадратными и имеют порядок $n \times n$). Как следует из приведенных соотношений, вычислительная сложность задачи

является достаточно высокой (оценка количества выполняемых операций имеет порядок n^3).

Основу возможности параллельных вычислений для матричного умножения составляет независимость расчетов для получения элементов c_{ij} результирующей матрицы C . Тем самым, все элементы матрицы C могут быть вычислены параллельно при наличии n^2 процессоров, при этом на каждом процессоре будет располагаться по одной строке матрицы A и одному столбцу матрицы B . При меньшем количестве процессоров подобный подход приводит к *ленточной схеме* разбиения данных, когда на процессорах располагаются по несколько строк и столбцов (полос) исходных матриц.

Другой широко используемый подход для построения параллельных способов выполнения матричного умножения состоит в использовании *блочного представления* матриц, при котором исходные матрицы A , B и результирующая матрица C рассматриваются в виде наборов блоков (как правило, квадратного вида некоторого размера $m \times m$). Тогда операцию матричного умножения матриц A и B в блочном виде можно представить следующим образом:

$$\begin{pmatrix} A_{11}A_{12}\dots A_{1k} \\ \dots \\ A_{k1}A_{k2}\dots A_{kk} \end{pmatrix} \times \begin{pmatrix} B_{11}B_{12}\dots B_{1k} \\ \dots \\ B_{k1}B_{k2}\dots B_{kk} \end{pmatrix} = \begin{pmatrix} C_{11}C_{12}\dots C_{1k} \\ \dots \\ C_{k1}C_{k2}\dots C_{kk} \end{pmatrix},$$

где каждый блок C_{ij} матрицы C определяется в соответствии с выражением:

$$C_{ij} = \sum_{l=1}^k A_{il}B_{lj}.$$

Полученные блоки C_{ij} также являются независимыми и, как результат, возможный подход для параллельного выполнения вычислений может состоять в выполнении расчетов, связанных с получением отдельных блоков C_{ij} , на разных процессорах. Применение подобного подхода позволяет получить многие *эффективные параллельные методы умножения блочно-представленных матриц*.

В системе ПараЛаб реализованы *параллельный алгоритм умножения матриц при ленточной схеме разделения данных* и два метода (*алгоритмы Фокса и Кэннона*) для блочно-представленных матриц. Более полная информация об алгоритмах умножения матриц, реализованных в системе ПараЛаб, содержится в главе 7.

1. Ленточный алгоритм. При ленточной схеме разделения данных исходные матрицы разбиваются на горизонтальные (для матрицы A) и верти-

кальные (для матрицы B) полосы. Получаемые полосы распределяются по процессорам, при этом на каждом из имеющегося набора процессоров располагается только по одной полосе матриц A и B . Перемножение полос (а выполнение процессорами этой операции может быть выполнено параллельно) приводит к получению части блоков результирующей матрицы C . Для вычисления оставшихся блоков матрицы C сочетания полос матриц A и B на процессорах должны быть изменены. В наиболее простом виде это может быть обеспечено, например, при кольцевой топологии вычислительной сети (при числе процессоров, равном количеству полос) – в этом случае необходимое для матричного умножения изменение положения данных может быть обеспечено циклическим сдвигом полос матрицы B по кольцу. После многократного выполнения описанных действий (количество необходимых повторений является равным числу процессоров) на каждом процессоре получается набор блоков, образующий горизонтальную полосу матрицы C .

Рассмотренная схема вычислений позволяет определить параллельный алгоритм матричного умножения при ленточной схеме разделения данных как итерационную процедуру, на каждом шаге которой происходит параллельное выполнение операции перемножения полос и последующего циклического сдвига полос одной из матриц по кольцу. Подробное описание ленточного алгоритма приводится в главе 7.

Задания и упражнения

1. Создайте в системе ПараЛаб новое окно вычислительного эксперимента. Для этого окна выберите задачу матричного умножения (щелкните левой кнопкой мыши на строке **Матричное умножение** пункта меню **Задача**).
2. Откройте диалоговое окно выбора топологии и выберите топологию кольцо.
3. Откройте диалоговое окно выбора метода и убедитесь в том, что выбран метод ленточного умножения матриц.
4. Проведите несколько вычислительных экспериментов. Изучите зависимость времени выполнения алгоритма от объема исходных данных и от количества узлов, процессоров и ядер.

2. Блочные алгоритмы Фокса и Кэннона. При блочном представлении данных параллельная вычислительная схема матричного умножения в наиболее простом виде может быть представлена, если топология вычислительной сети имеет вид прямоугольной решетки (если реальная топология сети имеет иной вид, представление сети в виде решетки можно обес-

печить на логическом уровне). Основные положения параллельных методов для блочно представленных матриц состоят в следующем:

- Каждый из процессоров решетки отвечает за вычисление одного блока матрицы C ;
- В ходе вычислений на каждом из процессоров располагается по одному блоку исходных матриц A и B ;
- При выполнении итераций алгоритмов блоки матрицы A последовательно сдвигаются вдоль строк процессорной решетки, а блоки матрицы B – вдоль столбцов решетки;
- В результате вычислений на каждом из процессоров вычисляется блок матрицы C , при этом общее количество итераций алгоритма равно \sqrt{p} (где p – число процессоров),

В главе 7 приводится полное описание параллельных методов Фокса и Кэннона для умножения блочно-представленных матриц.

Задания и упражнения

1. В активном окне вычислительного эксперимента системы ПараЛаб установите топологию **Решетка**. Выберите число узлов, равное девяти. Сделайте текущей задачей этого окна задачу матричного умножения.
2. Выберите метод Фокса умножения матриц и проведите вычислительный эксперимент.
3. Выберите алгоритм Кэннона матричного умножения и выполните вычислительный эксперимент. Пронаблюдайте различные маршруты передачи данных при выполнении алгоритмов. Сравните временные характеристики алгоритмов.
4. Измените число узлов в топологии **Решетка** на шестнадцать. Последовательно выполните вычислительные эксперименты с использованием метода Фокса и метода Кэннона. Сравните временные характеристики этих экспериментов.

13.4.3. Решение систем линейных уравнений

Системы линейных уравнений возникают при решении ряда прикладных задач, описываемых системами нелинейных (трансцендентных), дифференциальных или интегральных уравнений. Они могут появляться также в задачах математического программирования, статистической обработки данных, аппроксимации функций, при дискретизации краевых дифференциальных задач методом конечных разностей или методом конечных элементов и др.

Линейное уравнение с n неизвестными x_0, x_1, \dots, x_{n-1} может быть определено при помощи выражения

$$a_0x_0 + a_1x_1 + \dots + a_{n-1}x_{n-1} = b$$

где величины a_0, a_1, \dots, a_{n-1} и b представляют собой постоянные значения.

Множество n линейных уравнений

$$a_{0,0}x_0 + a_{0,1}x_1 + \dots + a_{0,n-1}x_{n-1} = b_0$$

$$a_{1,0}x_0 + a_{1,1}x_1 + \dots + a_{1,n-1}x_{n-1} = b_1$$

...

$$a_{n-1,0}x_0 + a_{n-1,1}x_1 + \dots + a_{n-1,n-1}x_{n-1} = b_{n-1}$$

называется *системой линейных уравнений* или *линейной системой*. В более кратком (*матричном*) виде система может быть представлена как

$$Ax = b,$$

где $A=(a_{ij})$ есть вещественная матрица размера $n \times n$, а векторы b и x состоят из n элементов.

Под *задачей решения системы линейных уравнений* для заданных матрицы A и вектора b обычно понимается нахождение значения вектора неизвестных x , при котором выполняются все уравнения системы.

В системе ПараЛаб реализованы два алгоритма решения систем линейных уравнений: широко известный *метод Гаусса* (прямой метод решения систем линейных уравнений, находит точное решение системы с невырожденной матрицей за конечное число шагов) и *метод сопряженных градиентов* – один из широкого класса итерационных методов решения систем линейных уравнений с матрицей специального вида. Более полная информация об алгоритмах решения систем линейных уравнений, реализованных в системе ПараЛаб, содержится в главе 8.

1. Алгоритм Гаусса. *Метод Гаусса* включает последовательное выполнение двух этапов. На первом этапе – *прямой ход метода Гаусса* – исходная система линейных уравнений при помощи последовательного исключения неизвестных (при помощи эквивалентных преобразований) приводится к верхнему треугольному виду. На *обратном ходе метода Гаусса* (второй этап алгоритма) осуществляется определение значений неизвестных. Из последнего уравнения преобразованной системы может быть вычислено значение переменной x_{n-1} , после этого из предпоследнего уравнения становится возможным определение переменной x_{n-2} и т. д.

При внимательном рассмотрении метода Гаусса можно заметить, что все вычисления сводятся к однотипным вычислительным операциям над

строками матрицы коэффициентов системы линейных уравнений. Как результат, в основу параллельной реализации алгоритма Гаусса может быть положен принцип распараллеливания по данным. В этом случае матрицу A можно разделить на горизонтальные полосы, а вектор правых частей b – на блоки. Полосы матрицы и блоки вектора правых частей распределяются между процессорами вычислительной системы.

Для выполнения **прямого хода** метода Гаусса необходимо осуществить $(n-1)$ итерацию по исключению неизвестных для преобразования матрицы коэффициентов A к верхнему треугольному виду.

Выполнение итерации i , $0 \leq i < n-1$, прямого хода метода Гаусса включает ряд последовательных действий. Процессор, на котором расположена строка i матрицы, должен разослать ее и соответствующий элемент вектора b всем процессорам, которые содержат строки с номерами k , $k > i$. Получив ведущую строку, процессоры выполняют вычитание строк, обеспечивая тем самым исключение соответствующей неизвестной x_i .

При выполнении **обратного хода** метода Гаусса процессоры выполняют необходимые вычисления для нахождения значения неизвестных. Как только какой-либо процессор определяет значение своей переменной x_i , это значение должно быть разослано всем процессорам, которые содержат строки с номерами k , $k < i$. Далее процессоры подставляют полученное значение новой неизвестной и выполняют корректировку значений для элементов вектора b .

Задания и упражнения

1. В активном окне вычислительного эксперимента системы ПараЛаб установите топологию **Полный Граф**. Выберите число узлов, равное десяти. Сделайте текущей задачей этого окна задачу решения системы линейных уравнений.

2. Выберите метод Гаусса решения задачи и выполните вычислительный эксперимент. Пронаблюдайте маршруты передачи данных при выполнении алгоритма.

2. Метод сопряженных градиентов. *Метод сопряженных градиентов* – один из широкого класса итерационных методов решения систем линейных уравнений, который может быть применен для решения симметричной положительно определенной системы линейных уравнений большой размерности. При выполнении итераций метода сопряженных градиентов к искомому точному решению x^* системы $Ax=b$ строится последовательность приближенных решений $x^0, x^1, \dots, x^k, \dots$. При этом процесс вычислений организуется таким способом, что каждое очередное приближение дает оценку точного решения со все уменьшающейся погрешно-

стью, и при продолжении расчетов оценка точного решения может быть получена с любой требуемой точностью.

При разработке параллельного варианта метода сопряженных градиентов для решения систем линейных уравнений в первую очередь следует учесть, что выполнение итераций метода осуществляется последовательно и, тем самым, наиболее целесообразный подход состоит в распараллеливании вычислений, реализуемых в ходе выполнения отдельных итераций. Основные вычисления, выполняемые в соответствии с методом, представляют собой операции матрично-векторного умножения, и, как результат, при организации параллельных вычислений могут быть использованы параллельные алгоритмы умножения матрицы на вектор.

Задания и упражнения

1. В активном окне вычислительного эксперимента системы ПараЛаб установите топологию **Полный Граф**. Выберите число процессоров, равное десяти. Сделайте текущей задачей этого окна задачу решения системы линейных уравнений.

2. Выберите метод сопряженных градиентов решения задачи и выполните вычислительный эксперимент. Пронаблюдайте маршруты передачи данных при выполнении алгоритма. Сравните эффективность алгоритма с эффективностью метода Гаусса.

13.4.4. Сортировка данных

Сортировка является одной из типовых проблем обработки данных, и обычно понимается как задача размещения элементов неупорядоченного набора значений

$$S = \{a_1, a_2, \dots, a_n\}$$

в порядке монотонного возрастания или убывания

$$S \sim S' = \{(a'_1, a'_2, \dots, a'_n) : a'_1 \leq a'_2 \leq \dots \leq a'_n\}.$$

Вычислительная трудоемкость процедуры упорядочивания является достаточно высокой. Так, для ряда известных простых методов (пузырьковая сортировка, сортировка включением и др.) количество необходимых операций определяется квадратичной зависимостью от числа упорядочиваемых данных

$$T_1 \sim n^2.$$

Для более эффективных алгоритмов (сортировка слиянием, сортировка Шелла, быстрая сортировка) трудоемкость определяется величиной

$$T_1 \sim n \log_2 n.$$

Данное выражение дает также нижнюю оценку необходимого количества операций для упорядочивания набора из n значений; алгоритмы с меньшей трудоемкостью могут быть получены только для частных вариантов задачи.

Ускорение сортировки может быть обеспечено при использовании нескольких ($p, p > 1$) процессоров. Исходный упорядочиваемый набор в этом случае разделяется между процессорами; в ходе сортировки данные пересылаются между процессорами и сравниваются между собой. Результирующий (упорядоченный) набор, как правило, также разделен между процессорами, при этом для систематизации такого разделения для процессоров вводится та или иная система последовательной нумерации и обычно требуется, чтобы при завершении сортировки значения, располагаемые на процессорах с меньшими номерами, не превышали значений процессоров с большими номерами.

В системе ПараЛаб в качестве методов упорядочения данных представлены *пузырьковая сортировка*, *сортировка Шелла*, *быстрая сортировка*. Исходные (последовательные) варианты этих методов изложены во многих изданиях (см., например, [71]), способы их параллельного выполнения излагаются в главе 9.

1. Пузырьковая сортировка. Алгоритм пузырьковой сортировки в прямом виде достаточно сложен для распараллеливания: сравнение пар соседних элементов происходит строго последовательно. Параллельный вариант алгоритма основывается на *методе чет – нечетной перестановки*, для которого на нечетной итерации выполнения сравниваются элементы пар

$$(a_1, a_2), (a_3, a_4), \dots, (a_{n-1}, a_n).$$

Если пара не упорядочена, то ее элементы переставляются. На четной итерации упорядочиваются пары

$$(a_2, a_3), (a_4, a_5), \dots, (a_{n-2}, a_{n-1}).$$

После n -кратного повторения подобных итераций массив оказывается отсортированным. Более подробная информация о параллельном варианте алгоритма приводится в главе 9.

Задания и упражнения

1. Запустите систему ПараЛаб и создайте новый эксперимент. Выберите пункт меню **Задача** и убедитесь, что в активном окне текущей задачей является задача сортировки. Откройте диалоговое окно выбора метода и посмотрите, какие алгоритмы сортировки могут быть выполнены на те-

кущей топологии. Так как при создании эксперимента по умолчанию текущей топологией становится решетка, то единственный возможный алгоритм – алгоритм сортировки пузырьком. Закройте диалоговое окно и вернитесь в основное меню системы ПараЛаб.

2. Выполните несколько экспериментов, изменяя размер исходных данных. Для выполнения эксперимента выполните команду **В активном окне** пункта меню **Выполнить**. Проанализируйте временные характеристики экспериментов, которые отображаются в правой нижней части окна.

3. Проведите несколько вычислительных экспериментов, изменяя количество узлов, процессоров и ядер вычислительной системы. Проанализируйте полученные временные характеристики.

2. Сортировка Шелла. *Параллельный алгоритм сортировки Шелла* может быть получен как обобщение метода параллельной пузырьковой сортировки. Основное различие состоит в том, что на первых итерациях алгоритма Шелла происходит сравнение пар элементов, которые в исходном наборе данных находятся далеко друг от друга (для упорядочивания таких пар в пузырьковой сортировке может потребоваться достаточно большое количество итераций). Подробное описание параллельного обобщения алгоритма сортировки Шелла можно найти в главе 9.

Задания и упражнения

1. Запустите систему ПараЛаб. Выберите топологию кольцо и установите количество узлов, равное восьми. Проведите эксперимент по выполнению алгоритма пузырьковой сортировки.

2. Установите в окне вычислительного эксперимента топологию гиперкуб и число узлов, равное восьми.

3. Откройте диалоговое окно выбора метода и посмотрите, какие алгоритмы сортировки могут быть выполнены на этой топологии. Выберите метод сортировки Шелла. Закройте окно.

4. Проведите вычислительный эксперимент. Сравните количество итераций, выполненных при решении задачи при помощи метода Шелла, с количеством итераций алгоритма пузырьковой сортировки (количество итераций отображается внизу в строке состояния). Убедитесь в том, что при выполнении эксперимента с использованием алгоритма Шелла, для сортировки массива необходимо выполнить меньшее количество итераций.

5. Проведите эксперимент с использованием метода Шелла несколько раз. Убедитесь, что количество итераций не является постоянной величиной и зависит от исходного массива.

3. Быстрая сортировка. *Алгоритм быстрой сортировки* основывается на последовательном разделении сортируемого набора данных на блоки меньшего размера таким образом, что между значениями разных блоков обеспечивается отношение упорядоченности. При параллельном обобщении алгоритма обеспечивается отношение упорядоченности между элементами сортируемого набора, расположенными на процессорах, соседних в структуре гиперкуба. В главе 9 Вы можете найти подробное описание алгоритма быстрой сортировки.

Задания и упражнения

1. Запустите систему ПараЛаб. В появившемся окне вычислительного эксперимента установите топологию гиперкуб и количество узлов, равное восьми.

2. Выполните три последовательных эксперимента с использованием трех различных алгоритмов сортировки: сортировки пузырьком, сортировки Шелла и быстрой сортировки. Сравните временные характеристики алгоритмов, которые отображаются в правой нижней части окна. Убедитесь в том, что у быстрой сортировки наименьшее время выполнения алгоритма и время передачи данных.

3. Измените объем исходных данных (выполните команду **Параметры задачи** пункта меню **Задача**). Снова проведите эксперименты. Сравните временные характеристики алгоритмов.

4. Измените количество процессоров (выполните команду **Количество процессоров** пункта меню **Система**). Проведите вычислительные эксперименты и сравните временные характеристики.

13.4.5. Обработка графов

Математические модели в виде графов широко используются при моделировании самых разнообразных явлений, процессов и систем. Как результат, многие теоретические и реальные прикладные задачи могут быть решены при помощи тех или иных процедур анализа графовых моделей. Среди множества этих процедур может быть выделен некоторый определенный набор типовых алгоритмов обработки графов.

В системе ПараЛаб реализованы параллельные алгоритмы решения двух типовых задач на графах: *алгоритм Прима для поиска минимального охватывающего дерева* и *алгоритм Дейкстры для поиска кратчайших путей*. Более полная информация об алгоритмах обработки графов, реализованных в системе ПараЛаб, содержится в главе 10.

Правила использования системы ПараЛаб

1. Переход в режим редактирования графа. При выборе задачи **Обработка графов** в системе ПараЛаб предусмотрена возможность создания и редактирования графов, запоминания графов в файле и чтения графа из файла. Чтобы перейти в режим редактирования графа, выполните команду **Редактор графов** пункта меню **Задача**. Заметим, что команда доступна только в том случае, когда текущей задачей является задача обработки графов.

После выполнения команды **Редактор графа** на экране дисплея появляется новое окно (рис. 13.10), в рабочей области которого отображается граф активного эксперимента. Если граф в эксперимент не был загружен, то рабочая область окна пуста.

Вам предоставляется возможность создавать новые графы, редактировать уже существующие, сохранять новые графы в файл и загружать граф в активный эксперимент.

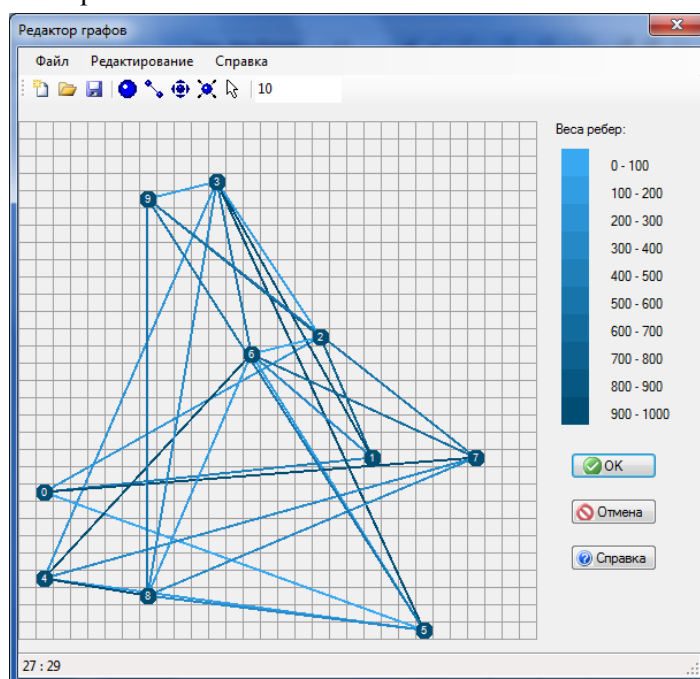





Рис. 13.10. Окно встроенного редактора графов


2. Создание нового графа. Для создания нового пустого графа выберите пункт меню **Файл** и выполните команду **Создать** (или щелкните левой кнопкой мыши на иконке  панели инструментов). Если граф, который


отображается в рабочей области, был изменен, то Вам будет предложено сохранить измененный граф в файл.


3. Открытие существующего графа. Для загрузки графа из файла выберите пункт меню **Файл** и выполните команду **Открыть** (или щелкните левой кнопкой мыши на иконке  панели инструментов). В появившемся диалоговом окне выберите имя файла (файлы графов ПараЛаб имеют расширение .plg) и нажмите кнопку **Открыть**.

4. Сохранение графа. Для сохранения графа в файл выберите пункт меню **Файл** и выполните команду **Сохранить** (или щелкните левой кнопкой мыши на иконке  панели инструментов). В появившемся диалоговом окне введите имя нового файла или выберите какой-либо из существующих файлов для того, чтобы сохранить граф в этом файле. Нажмите кнопку **Сохранить**.


5. Редактирование графа. С графом, расположенным в рабочей области окна, можно производить следующие операции:


- Для того, чтобы добавить к графу одну или несколько новых вершин, выберите пункт меню **Редактирование** и выполните команду **Добавить вершину** (или щелкните левой кнопкой мыши на иконке  панели инструментов). При этом вид курсора изменится, над указателем появится символическое изображение вершины. Рабочая область окна представляет собой сетку. Щелкая левой кнопкой мыши в различных клетках сетки, Вы можете добавлять в граф новые вершины. Если Вы щелкнули в клетке, где уже есть вершина, то добавления вершины не произойдет.

- Чтобы соединить две вершины графа ребром, выберите пункт меню **Редактирование** и выполните команду **Добавить/удалить ребро** (или щелкните левой кнопкой мыши на иконке  панели инструментов). После этого курсор изменит форму, под указателем появится изображение двух вершин, соединенных ребром. Выделите одну из вершин графа, щелкнув на ней левой кнопкой мыши. Ее цвет изменится на темно-красный. Выделите другую вершину графа – в результате между первой и второй указанными вершинами появится ребро. Вес ребра определяется случайным образом. Если между первой и второй вершинами до редактирования существовало ребро, то оно будет удалено.

- Для того, чтобы переместить вершину графа, выберите пункт меню **Редактирование** и выполните команду **Переместить вершину** (или щелкните левой кнопкой мыши на иконке  панели инструментов). После этого курсор изменит форму, примет вид, изображенный на кнопке панели инструментов. Выделите одну из вершин графа, щелкнув на ней левой кнопкой мыши. Цвет вершины изменится на темно-красный. Перемещайте курсор мыши по рабочей области окна – Вы увидите, что вершина пере-

мещается вслед за курсором. Щелкните на любой пустой клетке рабочей области, и выделенная вершина переместится в эту точку.

- Для удаления вершины графа выберите пункт меню **Редактирование** и выполните команду **Удалить вершину** (или щелкните левой кнопкой мыши на иконке  панели инструментов). После этого курсор изменит вид, под указателем появится пиктограмма перечеркнутой вершины. Щелкните левой кнопкой мыши на любой вершине графа, чтобы удалить ее.

Для выхода из любого из режимов (Добавление вершины, Удаление вершины, Перемещение вершины, Добавление ребра) щелкните левой кнопкой мыши на иконке  панели инструментов.

6. Формирование графа при помощи случайного механизма. Граф можно задавать при помощи случайного генератора. Для этого в редакторе, расположенном на панели инструментов, укажите число вершин графа, далее выберите пункт меню **Редактирование** и выполните команду **Создать случайный граф**.

7. Редактирование веса ребра графа. Цвет ребер графа имеет разную интенсивность. Чем темнее цвет, тем больше вес ребра. Чтобы приблизительно определить вес ребра, нужно сравнить его цвет со шкалой, расположенной справа. Для изменения веса ребра щелкните на нем правой кнопкой мыши. Рядом с ребром появится ползунок. Перемещая его вправо, Вы увеличиваете вес ребра, перемещая влево – уменьшаете. Для закрепления изменений щелкните мышкой в любой точке рабочей области или нажмите любую клавишу.

8. Выход из режима редактирования. Для загрузки текущего графа в активный эксперимент, нажмите кнопку **ОК**. Для выхода без сохранения изменений нажмите кнопку **Отмена**.

1. Алгоритм Прима поиска минимального охватывающего дерева. *Охватывающим деревом* (или *остовом*) неориентированного графа G называется подграф T графа G , который является деревом и содержит все вершины из G . Определив вес подграфа для взвешенного графа как сумму весов входящих в подграф дуг, тогда под *минимально охватывающим деревом* (МОД) T будем понимать охватывающее дерево минимального веса. Содержательная интерпретация задачи нахождения МОД может состоять, например, в практическом примере построения локальной сети персональных компьютеров с прокладыванием наименьшего количества соединительных линий связи.

Дадим краткое описание алгоритма решения поставленной задачи, известного под названием *метода Прима* (*Prim*). Алгоритм начинает работу с произвольной вершины графа, выбираемого в качестве корня дерева, и в

ходе последовательно выполняемых итераций расширяет конструируемое дерево до МОД. Распределение данных между процессорами вычислительной системы должно обеспечивать независимость перечисленных операций алгоритма Прима. В частности, это может быть обеспечено, если каждая вершина графа располагается на процессоре вместе со всей связанной с вершиной информацией.

С учетом такого разделения данных итерация параллельного варианта алгоритма Прима состоит в следующем:

- определяется вершина, имеющая наименьшее расстояние до построенного к этому моменту МОД (операции вычисления расстояния для вершин графа, не включенных в МОД, независимы и, следовательно, могут быть выполнены параллельно);
- найденная вершина включается в состав МОД.

2. Алгоритм Дейкстры поиска кратчайших путей. *Задача поиска кратчайших путей* на графе состоит в нахождении путей минимального веса от некоторой заданной вершины s до всех имеющихся вершин графа. Постановка подобной проблемы имеет важное практическое значение в различных приложениях, когда веса дуг означают время, стоимость, расстояние, затраты и т. п.

Возможный способ решения поставленной задачи, известный как алгоритм Дейкстры, практически совпадает с методом Прима. Различие состоит лишь в интерпретации и в правиле оценки расстояний. В алгоритме Дейкстры эти величины означают суммарный вес пути от начальной вершины до всех остальных вершин графа. Как результат, на каждой итерации алгоритма выбирается очередная вершина, расстояние от которой до корня дерева минимально, и происходит включение этой вершины в дерево кратчайших путей.

Задания и упражнения

1. Запустите систему ПараЛаб. В активном окне вычислительного эксперимента установите топологию **Полный граф**. Текущей задачей этого окна сделайте задачу обработки графов.

2. Выполните команду **Редактор графов** пункта меню **Задача**. В появившемся редакторе графов сформируйте случайным образом граф с десятью вершинами.

3. Выполните вычислительный эксперимент по поиску минимального охватывающего дерева с помощью алгоритма Прима (выполните команду **Метод решения** пункта меню **Задача**, в появившемся диалоговом окне выберите **Алгоритм Прима (поиск охватывающего дерева)**).

4. Проведите несколько экспериментов, изменяя количество узлов, процессоров и ядер. Изучите зависимость временных характеристик алгоритма Прима от количества процессоров.

5. Проведите аналогичную последовательность экспериментов для изучения временных характеристик метода Дейкстры.

13.4.6. Решение дифференциальных уравнений

Дифференциальные уравнения в частных производных представляют собой широко применяемый математический аппарат при разработке моделей в самых разных областях науки и техники. К сожалению, явное решение этих уравнений в аналитическом виде оказывается возможным только в частных простых случаях, так что возможность анализа математических моделей, построенных на основе дифференциальных уравнений, обеспечивается при помощи приближенных численных методов решения. Объем выполняемых при этом вычислений обычно является значительным и использование высокопроизводительных вычислительных систем является традиционным для данной области вычислительной математики.

Приведем краткую информацию по решению дифференциальных уравнений в системе ПараЛаб; полная информация может получена в главе 11.

В системе ПараЛаб, в качестве учебного примера рассматривается *проблема численного решения задачи Дирихле для уравнения Пуассона*, определяемую как задачу нахождения функции $u = u(x, y)$, удовлетворяющей в области определения D уравнению

$$\begin{cases} \frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 u}{\delta y^2} = f(x, y), & (x, y) \in D, \\ u(x, y) = g(x, y), & (x, y) \in D^0, \end{cases}$$

и принимающей значения $g(x, y)$ на границе D^0 области D (f и g являются функциями, задаваемыми при постановке задачи). Подобная модель может быть использована для описания установившегося течения жидкости, стационарных тепловых полей, процессов теплопередачи с внутренними источниками тепла и деформации упругих пластин и др.

Для простоты изложения материала в качестве области задания D функции $u(x, y)$ используется единичный квадрат

$$D = \{(x, y) \in D : 0 \leq x, y \leq 1\}.$$

Одним из наиболее распространенных подходов численного решения дифференциальных уравнений является *метод конечных разностей* (*метод сеток*). Следуя этому подходу, область решения D представляется в виде дискретного (как правило, равномерного) набора (*сетки*) точек (*узлов*). Так, например, прямоугольная сетка в области D может быть задана в виде (рис. 13.11)

$$\begin{cases} D_h = \{(x_i, y_j) : x_i = ih, y_j = jh, 0 \leq i, j \leq N+1, \\ h = 1/(N+1), \end{cases}$$

где величина N задает количество узлов по каждой из координат области D .

Обозначим оцениваемую при подобном дискретном представлении аппроксимацию функции $u(x, y)$ в точках (x_i, y_j) через u_{ij} . Тогда, используя *пятиточечный шаблон* (см. рис. 13.11) для вычисления значений производных, мы можем представить уравнение Пуассона в *конечно-разностной форме*

$$\frac{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{ij}}{h^2} = f_{ij}.$$

Данное уравнение может быть разрешено относительно u_{ij}

$$u_{ij} = 0.25(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - h^2 f_{ij}).$$

Разностное уравнение, записанное в подобной форме, позволяет определять значение u_{ij} по известным значениям функции $u(x, y)$ в соседних узлах используемого шаблона. Данный результат служит основой для построения различных *итерационных схем* решения задачи Дирихле, в которых в начале вычислений формируется некоторое приближение для значений u_{ij} , а затем эти значения последовательно уточняются в соответствии с приведенным соотношением.

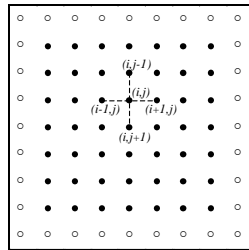


Рис. 13.11. Прямоугольная сетка в области D (темные точки представляют внутренние узлы сетки, нумерация узлов в строках слева направо, а в столбцах – сверху вниз)

Для организации параллельных вычислений на системах с распределенной памятью необходимо выбрать способ разделения обрабатываемых данных между вычислительными серверами. Успешность такого разделения определяется достигнутой степенью локализации вычислений на серверах (в силу больших временных задержек при передаче сообщений интенсивность взаимодействия серверов должна быть минимальной).

В рассматриваемой учебной задаче по решению задачи Дирихле применяется *ленточная* схема разделения данных (см. рис. 13.12) вычислительной сетки.

При ленточном разбиении область расчетов делится на горизонтальные или вертикальные полосы (не уменьшая общности, далее будем рассматривать только горизонтальные полосы). Число полос определяется количеством процессоров, размер полос обычно является одинаковым, узлы горизонтальных границ (первая и последняя строки) включаются в первую и последнюю полосы соответственно. Полосы для обработки распределяются между процессорами.

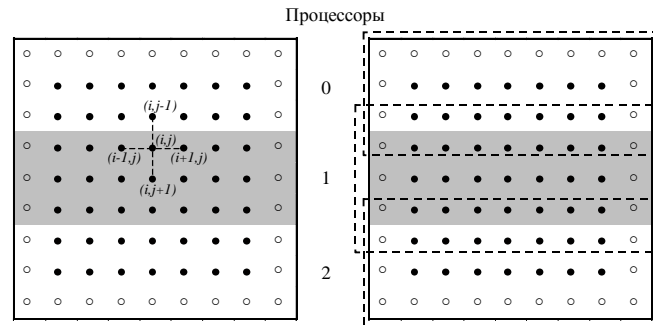


Рис. 13.12. Ленточное разделение области расчетов между процессорами (кружки представляют граничные узлы сетки)

Основной момент при организации вычислений с подобным разделением данных состоит в том, что на процессор, выполняющий обработку какой-либо полосы, должны быть продублированы граничные строки предшествующей и следующей полос вычислительной сетки (получаемые в результате расширенные полосы показаны на рис. 13.12 справа пунктирными рамками). Продублированные граничные строки полос используются только при проведении расчетов, пересчет же этих строк происходит в полосах своего исходного месторасположения. Тем самым дублирование граничных строк должно осуществляться перед началом выполнения каждой очередной итерации метода сеток.

Для решения задачи Дирихле необходимо задать *граничные условия* и *начальные значения* узловых элементов сетки. Для решения уравнения необходимо так же задать *количество узлов в сетке*. Длительность вычислений определяется требуемой *точностью* и *максимально-возможным количеством* выполняемых итераций. Все перечисленные параметры в системе ПараЛаб можно задать в *параметрах задачи* (рис. 13.13).

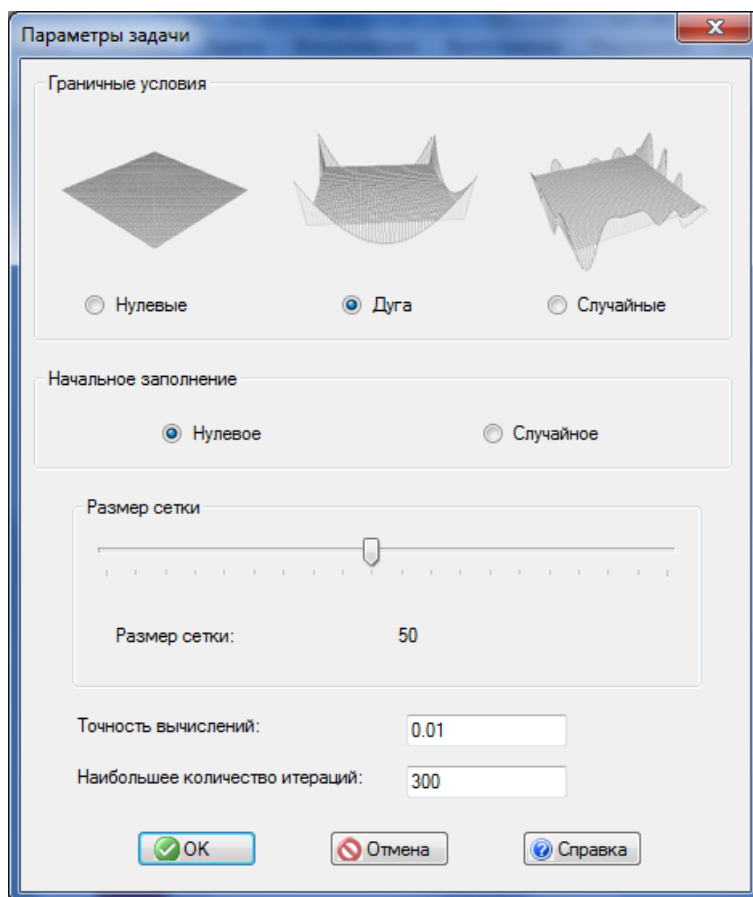


Рис. 13.13. Установка параметров задачи решения дифференциальных уравнений

Задания и упражнения

1. Запустите систему ПараЛаб. В активном окне вычислительного эксперимента установите топологию **Решетка**. Текущей задачей этого окна установите задачу решения системы дифференциальных уравнений.

2. Задайте начальные условия для задачи решения дифференциального уравнения.
3. Выполните вычислительный эксперимент.
4. Проведите несколько экспериментов, изменяя количество узлов, процессоров и ядер. Изучите зависимость временных характеристик от количества процессоров и выбранной топологии узлов.

13.4.7. Решение задач многоэкстремальной оптимизации

Во всех областях своей целенаправленной деятельности перед человеком возникает проблема выбора наилучшего решения из множества всех возможных. Примерами могут служить экономика (управление экономическими объектами), техника (выбор оптимальной конструкции). К числу наиболее распространенных моделей рационального выбора относятся математические задачи оптимизации (максимизации или минимизации) некоторого функционала при ограничениях типа неравенств. При этом выполнение ограничений для некоторого вектора параметров, определяющего решение, интерпретируется как допустимость этого решения, т. е. как возможность его реализации при имеющихся ресурсах. Теорию и методы отыскания минимумов функций многих переменных при наличии дополнительных ограничений на эти переменные обычно рассматривают как отдельную дисциплину – математическое программирование.

Полное описание рассматриваемой темы приводится в главе 12; здесь же дается краткий материал, достаточный для изучения методов многоэкстремальной оптимизации в системе ПараЛаб.

В общем виде задачу математического программирования можно сформулировать следующим образом. Пусть $\varphi(y)$, $g_j(y) \leq 0$, $1 \leq j \leq m$, есть действительные функции, определенные на множестве D N -мерного евклидова пространства R^N , и пусть точка y^* удовлетворяет условию

$$\varphi(y^*) = \min \{ \varphi(y) : y \in D, g_j(y) \leq 0, 1 \leq j \leq m \}.$$

Точка из y^* обычно называется *глобально-оптимальной точкой* или *глобально-оптимальным решением*. При этом функцию $\varphi(y)$ называют *функцией цели*, или *целевой функцией*, а функции $g_j(y) \leq 0$, $1 \leq j \leq m$, – *ограничениями задачи*.

Область D называют *областью поиска* и обычно описывают как некоторый гиперинтервал из N -мерного евклидова пространства

$$D = \{ y \in R^N : a_i \leq y_i \leq b_i, 1 \leq i \leq n \},$$

где $a, b \in R^N$ есть заданные векторы. Точки из области поиска, удовлетворяющие всем ограничениям, называются *допустимыми точками* или *допустимыми решениями*. Множество

$$Q = \{y: y \in D, g_j(y) \leq 0, 1 \leq j \leq m\}$$

всех таких точек называют *допустимой областью*.

Важный в прикладном отношении подкласс задач вида характеризуется тем, что все функционалы, входящие в определение задачи, заданы некоторыми (программно реализуемыми) алгоритмами вычисления значений $\varphi(y)$, $g_j(y)$, $1 \leq j \leq m$, в точках области поиска D . При этом *решение задачи* сводится к построению оценки $y_* \in Q$, отвечающей некоторому понятию близости к точке y^* (например, чтобы $\|y^* - y_*\| \leq \varepsilon$ или $|\varphi(y^*) - \varphi(y_*)| \leq \varepsilon$, где $\varepsilon > 0$ есть заданная точность) на основе некоторого числа k значений функционалов задачи, вычисленных в точках области D .

В задачах многоэкстремальной оптимизации возможность достоверной оценки глобального оптимума принципиально основана на наличии *априорной информации* о функции, позволяющей связать возможные значения минимизируемой функции с известными значениями в точках осуществленных поисковых итераций. Весьма часто такая априорная информация о задаче представляется в виде предположения, что целевая функция φ (в дальнейшем обозначаемая также g_{m+1}) и левые части ограничений $g_j, 1 \leq j \leq m$, удовлетворяют *условию Липшица* с соответствующими константами $L_j, 1 \leq j \leq m+1$, а именно

$$|g_j(y_1) - g_j(y_2)| \leq L_j \|y_1 - y_2\|, 1 \leq j \leq m+1, y_1, y_2 \in D.$$

В общем случае все эти функции могут быть *многоэкстремальными*.

Фундаментальные результаты в этом направлении были получены Нижегородской школой глобальной оптимизации (коллектив исследователей под руководством Р.Г. Стронгина, Нижегородский государственный университет). Один из наиболее важных результатов состоит в разработке информационно-статистического подхода к построению алгоритмов многоэкстремальной оптимизации [28,97]. Минимизируемая функция рассматривается как реализация некоторого случайного процесса, и решающие правила алгоритма конструируются таким образом, что очередная итерация проводится в точке глобального минимума математического ожидания значений функции. Более подробная

информация о информационно-статистическом подходе приводится в главе 12.

Для построения параллельного алгоритма используются множественные отображения (см. главу 12). Использование множественных отображений позволяет решать исходную задачу путем параллельного решения индексным методом $L+1$ задач на наборе отрезков $[0,1]$. Каждая одномерная задача (или группа задач при недостаточном количестве процессоров) решается на отдельном процессоре. Результаты испытания в точке x^k , полученные конкретным процессором для решаемой им задачи, интерпретируются как результаты испытаний во всех остальных задачах (в соответствующих точках $x^{k0}, x^{k1}, \dots, x^{kL}$). При таком подходе испытание в точке $x^k \in [0,1]$, осуществляемое в s -й задаче, состоит в последовательности действий:

1. Определить образ $y^k = y^s(x^k)$ при соответствии $y^s(x)$.
2. Вычислить величину $g_0(y^k)$. Если $g_0(y^k) \leq 0$, то есть $y^k \in D$, то проинформировать остальные узлы о начале проведения испытания в точке y^k (блокирование точки y^k).
3. Вычислить величины $g_1(y^k), \dots, g_\nu(y^k)$, где значения индекса $\nu \leq m$ определяются условиями

$$g_j(y^k) \leq 0, 1 \leq j < \nu, g_\nu(y^k) > 0, \nu \leq m.$$

Выявление первого нарушенного ограничения прерывает испытание в точке y^k . В случае, когда точка y^k допустима, т. е. когда $y^s(x^k) \in Q_{m+1}$, испытание включает вычисление значений всех функционалов задачи. При этом значение индекса принимается равным величине $\nu = m+1$, а тройка

$$y^s(x^k), \nu = \nu(x^k), z^k = g_\nu(y^s(x^k)),$$

является *результатом испытания* в точке x^k .

4. Если $\nu(x^k) > 0$, то есть $y^k \in D$, то определить прообразы $x^{kl} \in [0,1]$, $0 \leq l \leq L$, точки y^k , и интерпретировать испытание, проведенное в точке $y^k \in D$, как проведение испытаний в $L+1$ точке

$$x^{k0}, x^{k1}, \dots, x^{kL},$$

с одинаковыми результатами

$$\begin{aligned} \nu(x^{k0}) &= \nu(x^{k1}) = \dots = \nu(x^{kL}) = \nu(x^k), \\ g_\nu(y^0(x^{k0})) &= g_\nu(y^1(x^{k1})) = \dots = g_\nu(y^L(x^{kL})) = z^k. \end{aligned}$$

Проинформировать остальные узлы о результатах испытания в точке y^k .

В случае если $v(x^k)=0$, то есть $y^k \notin D$, результат испытания относится только к s -й задаче.

Каждый узел имеет свою копию программных средств, реализующих вычисление функционалов задачи, и решающее правило алгоритма. Для организации взаимодействия на каждом узле создается $L+1$ очередь, в которые процессоры помещают информацию о выполненных итерациях в виде троек: точка очередной итерации, индекс и значение, причем индекс заблокированной точки полагается равным -1 , а значение функции в ней не определено.

Предлагаемая схема не содержит какого-либо единого управляющего узла, что увеличивает надежность выполняемых вычислений.

Для решения задачи оптимизации необходимо задать параметры, указанные в диалоговом окне Параметры задачи (см. рис. 13.14).

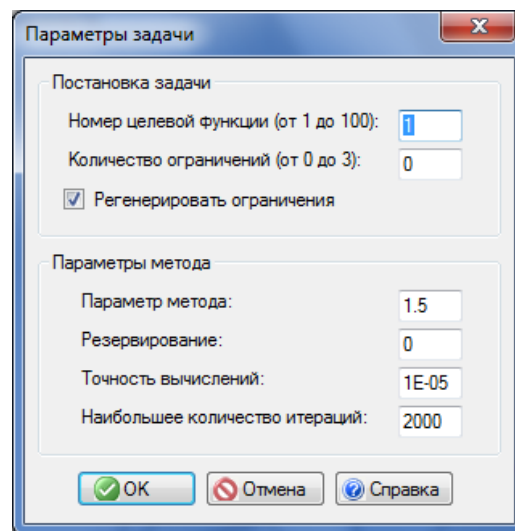


Рис. 13.14. Задание параметров задачи многоэкстремальной оптимизации

Задания и упражнения

1. Запустите систему ПараЛаб. В активном окне вычислительного эксперимента установите топологию **Полный граф**. Текущей задачей этого окна сделайте задачу многоэкстремальной оптимизации.
2. Задайте начальные условия для задачи многоэкстремальной оптимизации.

3. Выполните вычислительный эксперимент по решению задачи многоэкстремальной оптимизации.

4. Проведите несколько экспериментов, изменяя количество узлов, процессоров и ядер. Изучите зависимость временных характеристик от количества процессоров, узлов и ядер.

13.5. Определение графических форм наблюдения за процессом параллельных вычислений

Для наблюдения за процессом выполнения вычислительного эксперимента по параллельному решению сложных вычислительно трудоемких задач в рамках системы ПараЛаб предусмотрены различные формы графического представления результатов выполняемых параллельных вычислений.

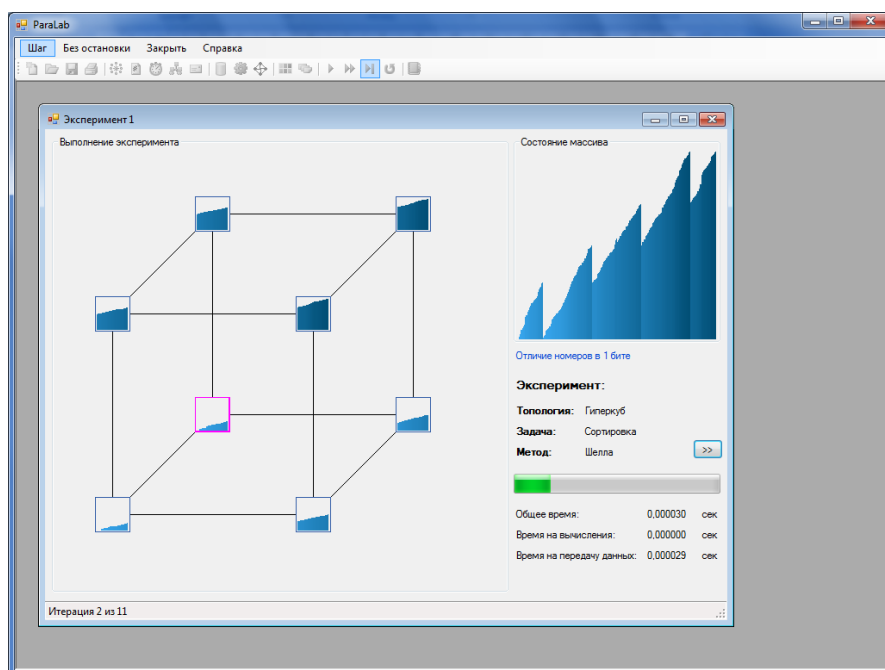


Рис. 13.15. Вид окна вычислительного эксперимента

Для представления сведений о ходе выполнения эксперимента в рабочей области системы ПараЛаб для каждого эксперимента выделяется прямоугольный участок экрана – *окно вычислительного эксперимента*. В левой части окна выделена область «Выполнение эксперимента», где изображаются процессоры многопроцессорной вычислительной системы, объединенные в ту или иную топологию, данные, расположенные на каждом

процессоре, и взаимообмен данными между процессорами системы. В правой верхней части окна отображается область с информацией о текущем состоянии объекта, являющегося результатом выполняемого эксперимента. В зависимости от того, какой эксперимент выполняется, эта область носит название:

- «Результат умножения матрицы на вектор» при выполнении матрично-векторного умножения
- «Результат умножения матриц» при выполнении матричного умножения,
- «Состояние линейной системы» при решении системы линейных уравнений,
- «Текущее состояние массива» при выполнении алгоритма сортировки,
- «Результат обработки графа» при выполнении алгоритмов на графах,
- «Распределение тепла» при решении дифференциальных уравнений,
- «Целевая функция» при решении задачи многоэкстремальной оптимизации.

В средней части правой половины окна эксперимента приводятся сведения о выполняемой задаче.

В правом нижнем углу располагается ленточный индикатор выполнения эксперимента и его текущие временные характеристики.

Дополнительно в отдельном окне могут быть более подробно визуально представлены вычисления, которые производит один из имеющегося набора процессор (задание режима высвечивания этого окна и выбор наблюдаемого процессора осуществляется пользователем системы).

13.5.1. Область «Выполнение эксперимента»

В этой области окна изображены данные узлов многопроцессорной вычислительной системы, соединенные линиями коммутации в ту или иную топологию. Если дважды щелкнуть левой клавишей мыши на изображении данных ядра, то появится окно «Демонстрация работы ядра», где будет детально отображаться деятельность этого ядра.

Для каждого ядра схематически изображаются данные, которые находятся на нем в данный момент выполнения эксперимента. Если эксперимент состоит в изучении какого-либо параллельного алгоритма сортировки, то на каждом ядре изображается часть сортируемого массива. Каждый

элемент массива изображается вертикальной линией. Высота и интенсивность цвета линии характеризуют величину элемента (чем выше и темнее линия, тем больше элемент). Если эксперимент заключается в изучении алгоритмов матричного умножения, то на каждом ядре изображен силуэт матрицы, на котором цветом выделены части исходных данных, располагаемых на ядре (предполагается, что исходные матрицы A и B являются квадратными порядка $n \times n$). Синим цветом помечается часть матрицы A на ядре (блок или горизонтальная полоса), оранжевым – часть матрицы B (блок или вертикальная полоса). Если же эксперимент состоит в изучении алгоритмов обработки графов, то на каждом ядре изображается подграф, состоящий из вершин, расположенных на этом процессоре.

В процессе выполнения эксперимента в области «Выполнение эксперимента» также отображается обмен данными между процессорами многопроцессорной вычислительной системы. Это может происходить в двух режимах:

- режим «**Выделение линий связей**» – выделяется красным та линия коммутации, по которой происходит обмен,
- режим «**Пересылка пакетов**» – визуализация обмена при помощи движущегося от одного ядра к другому прямоугольника (конверта). Если изучаются параллельные алгоритмы матричного умножения, то на конверте изображается номер блока, который пересылается.

При выполнении алгоритмов на графах все итерации параллельного алгоритма однотипны и число их велико (равно числу вершин в графе). Для отображения всех итераций понадобится достаточно много времени. Поэтому в системе ПараЛаб реализована возможность отображать не все итерации, а лишь некоторые из них.

Правила использования системы ПараЛаб

1. Изменение способа отображения пересылки данных. Для задания способа отображения коммуникации процессоров выполните команду **Пересылка данных** пункта меню **Визуализация**. В появившемся списке выберите название желаемого способа отображения.

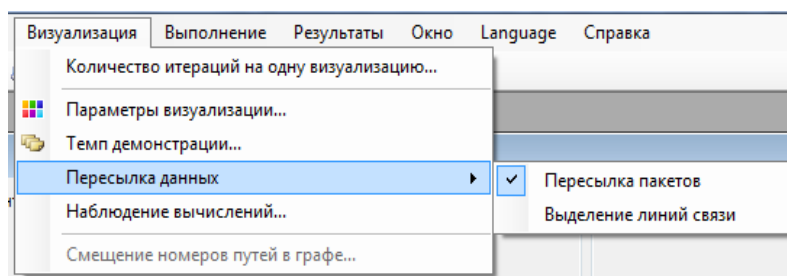
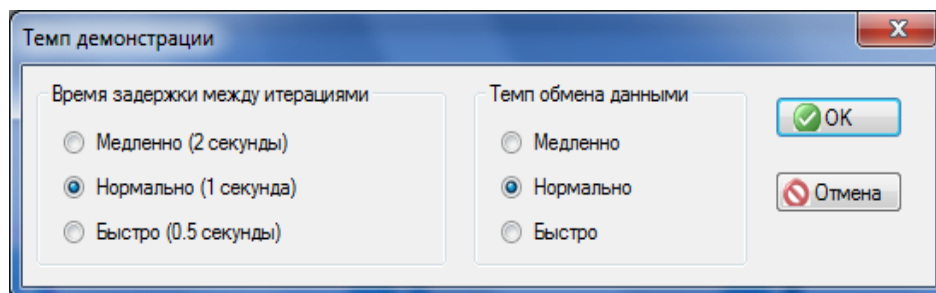


Рис. 13.16. Выбор способа отображения пересылки данных

2. Выбор темпа демонстрации. Для выбора темпа демонстрации необходимо выполнить команду **Темп Демонстрации** пункта меню **Визуализация**. В появившемся диалоговом окне (рис. 13.17) предоставляется возможность выбора величины задержки между итерациями алгоритма и скорости движения пакетов (времени цветового выделения канала) при отображении коммуникации процессоров.

**Рис. 13.17.** Диалоговое окно выбора темпа демонстрации

Нажмите **ОК** (Enter) для подтверждения выбора темпа демонстрации. Для возврата в основное меню системы ПараЛаб без сохранения изменений нажмите **Отмена** (Escape).

3. Изменение шага визуализации. Для того, чтобы в рабочей области системы ПараЛаб отображалась не каждая итерация, а лишь некоторые из них, выполните команду **Количество итераций на одну визуализацию** пункта меню **Визуализация**. В появившемся диалоговом окне установите при помощи бегунка желаемую частоту отображения итераций. Для выбора шага визуализации нажмите **ОК** (Enter), для возврата в основное меню системы ПараЛаб нажмите **Отмена** (Escape)

4. Настройка цветовой палитры. Для изменения цветов, которые используются в системе ПараЛаб для визуализации процесса решения задач, выполните команду **Параметры визуализации** пункта меню **Визуализация**. В появившемся диалоговом окне (рис. 13.18) Для каждого алгоритма можно задать цвета отображения различных состояний данных, а также метод отображения результатов (двух- или трех- мерный). На приведенном рисунке представлен диалог для блочного перемножения матриц. Соответственно, матрицу можно отобразить только на двухмерной области и есть возможность задать цвет для вычисленных блоков и блоков, еще подлежащих вычислению.

Чтобы изменить цвета, щелкните левой клавишей мыши на цветной кнопке соответствующего параметра. В результате появится стандартное диалоговое окно выбора цвета операционной системы Windows. В этом окне выберите цвет и нажмите кнопку **ОК**. Цвет будет изменен. Для того, чтобы изменить цвет выделения, щелкните левой клавишей мыши на отображающем этот цвет квадрате. При помощи стандартного диалогового окна задайте необходимый цвет.

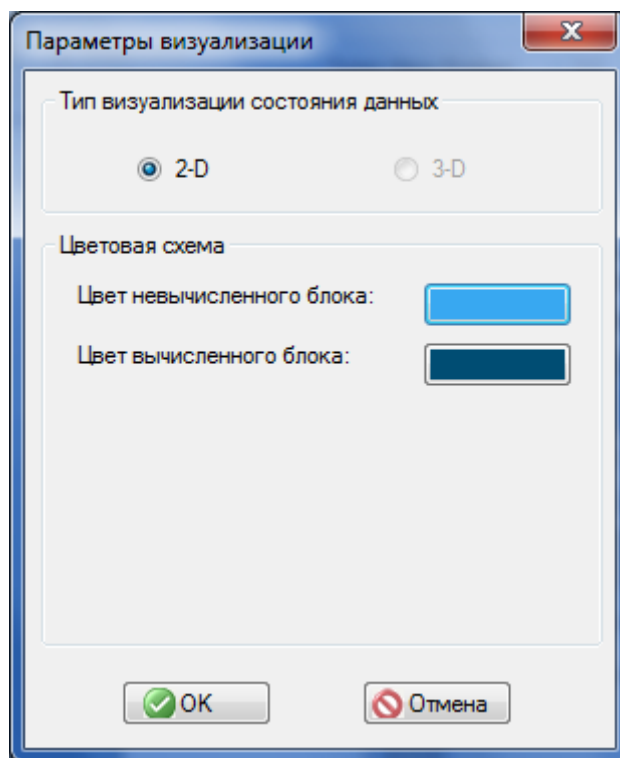


Рис. 13.18. Диалоговое окно настройки визуализации

Для изменения цветовой палитры нажмите кнопку **ОК** (Enter) в диалоговом окне «Параметры визуализации». Для возврата в основной режим работы системы ПараЛаб нажмите **Отмена** (Escape).

13.5.2. Область «Текущее состояние массива»

Эта область расположена в правой верхней части окна и отображает последовательность элементов сортируемого массива. Каждый элемент, как и в области «Выполнение эксперимента», отображается вертикальной

линией. Высота и интенсивность цвета линии дают представление о величине элемента: чем выше и темнее линия, тем больше элемент.

Все параллельные алгоритмы используют идею разделения исходного массива между процессорами. Блоки, выстроенные один за другим в порядке возрастания номеров процессоров, на которых они располагаются, образуют результирующий массив. После выполнения сортировки блоки массива на каждом процессоре должны быть отсортированы и, кроме того, элементы, находящиеся на ядре с меньшим номером, не должны превосходить элементов, находящихся на ядре с большим номером.

Изначально массив – случайный набор элементов. После выполнения сортировки массив (при достаточно большом объеме исходных данных) изображается в виде прямоугольного треугольника с плавным перетеканием цвета из голубого в темно-синий.

13.5.3. Область «Результат умножения матрицы на вектор»

Эта область находится в правой верхней части окна и отображает состояние вектора-результата в ядре выполнения параллельного алгоритма умножения матрицы на вектор.

При выполнении алгоритма, основанного на ленточном горизонтальном разбиении матрицы, каждое ядро вычисляет один блок результирующего вектора путем умножения полосы матрицы A на вектор-аргумент b . Вычисленный на активном ядре (подсвечен синим цветом) блок изображается темно-синим цветом. После выполнения коммуникации, на каждом ядре располагается весь результирующий вектор. Таким образом, все блоки вектора становятся темно-синими.

При выполнении алгоритма, основанного на ленточном вертикальном разбиении матрицы, каждое ядро вычисляет вектор частичных результатов путем умножения полосы матрицы на блок вектора-аргумента b ; все блоки результирующего вектора в области «Результат умножения матрицы на вектор» подсвечиваются светло-синим цветом. После выполнения коммуникационного шага на каждом ядре располагается блок результирующего вектора, блок активного процессора отображается в области «Результат умножения матрицы на вектор» темно-синим цветом.

При выполнении алгоритма, основанного на блочном разбиении матрицы, вектор b распределен между узлами, составляющими столбцы вычислительной системы. После умножения блока матрицы A на блок вектора b ядра процессоров вычисляет блок вектора частичных результатов – он подсвечивается светло-синим цветом. После обмена блоками в рамках одной строки узлов вычислительной системы, каждый узел этой строки содержит блок результирующего вектора, блок активного

ядра отображается в области «Результат умножения матрицы на вектор» темно-синим цветом.

13.5.4. Область «Результат умножения матриц»

Эта область находится в правой верхней части окна и отображает состояние матрицы – результата в ядре выполнения параллельного алгоритма матричного умножения.

Матрица C представляется разбитой на квадратные блоки. Каждое ядро многопроцессорной вычислительной системы отвечает за вычисление одного (алгоритмы Фокса и Кэннона) или нескольких (ленточный алгоритм) блоков результирующей матрицы C .

При выполнении ленточного алгоритма умножения темно-синим цветом закрашиваются те блоки, которые уже вычислены к данному моменту.

Если же выполняется алгоритм Фокса или алгоритм Кэннона, то все блоки матрицы C вычисляются одновременно, ни один из блоков не может быть вычислен раньше, чем будут выполнены все итерации алгоритма. Поэтому в области «Результат умножения матриц» отображается динамика вычисления того блока результирующей матрицы, который расположен на активном ядре (это ядро в области «Выполнение эксперимента» выделен синим цветом). Вычисленные к этому моменту слагаемые написаны темно-синим цветом, вычисляемое на данной итерации слагаемое – цветом выделения.



Рис. 13.19. Область «Результат умножения матриц» при выполнении алгоритма Фокса

13.5.5. Область «Результат решения системы уравнений»

Эта область расположена в правой верхней части окна вычислительного эксперимента и отображает текущее состояние матрицы линейной системы уравнений в ходе выполнения алгоритма Гаусса. Темно-синим цветом изображаются ненулевые элементы, а голубым – нулевые. После выполнения прямого хода алгоритма Гаусса ниже главной диагонали распо-

ложены только нулевые элементы. После выполнения обратного хода все ненулевые элементы расположены на главной диагонали.

13.5.6. Область «Результат обработки графа»

Эта область расположена в правой верхней части окна вычислительного эксперимента и отображает текущее состояние графа. Вершины графа имеют такое же взаимное расположение, как и в режиме редактирования графа. Дуги графа изображаются разными цветами: чем темнее цвет, тем больший вес имеет дуга.

В процессе выполнения алгоритмов на графах цветом выделения помечаются вершины и ребра, включенные к данному моменту в состав минимального охватывающего дерева (алгоритм Прима) или в дерево кратчайших путей (алгоритм Дейкстры).

13.5.7. Область «Распределение тепла»

Эта область расположена в правой верхней части окна вычислительного эксперимента и отображает текущее состояние узлов сетки разностной схемы. Каждый узел отображается точкой. Если точка маленькая, то это означает, что в узле на текущей итерации вычисления еще не выполнялись; если же узел выделен увеличенным маркером, то вычисления в узле уже завершены. Цветами обозначена принадлежность тому или иному узлу сетки.

В процессе выполнения алгоритма динамически отображается последовательность обработки узлов.

13.5.8. Область «Целевая функция»

Эта область расположена в правой верхней части окна вычислительного эксперимента и отображает линии уровней и точки проведенных испытаний.

В процессе выполнения алгоритма поиска оптимального значения на графике линий уровня цветными точками отображаются точки испытаний, в которых выполнялось вычисления значения минимизируемой функции.

13.5.9. Выбор процессора

Для более детального наблюдения за процессом выполнения эксперимента в системе ПараЛаб предусмотрена возможность отображения вычислений одного из процессоров системы в отдельном окне

Один из способов выбора процессора – выполнить команду **Наблюдение Вычислений** пункта меню **Визуализация**. В появившемся диалоговом окне при помощи бегунка укажите номер процессора и нажмите **ОК (Enter)**. Для возврата в основное меню системы и отказа от выбора процессора нажмите **Отмена (Escape)**.

Второй способ выбора процессора – в рабочей области навести на него указатель мыши и щелкнуть дважды левой клавишей.

Далее в появившемся окне «**Демонстрация работы ядра**» будет детально изображаться ход вычислений.

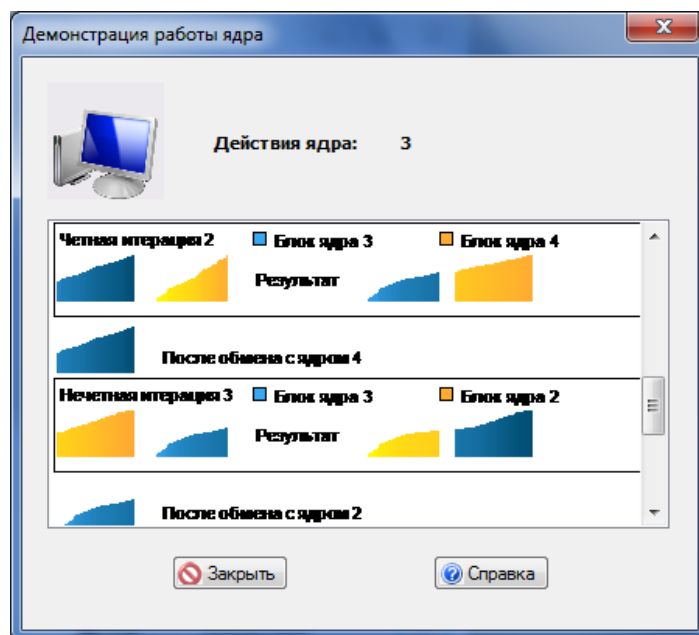


Рис. 13.20. Вид окна демонстрации работы процессора

13.6. Накопление и анализ результатов экспериментов

Выполнение численных экспериментов для изучения различных параллельных алгоритмов решения сложных вычислительных задач во многих случаях может потребовать проведения длительных вычислений. Для обоснования выдвигаемых предположений необходимо выполнить достаточно широкий набор экспериментов. Эти эксперименты могут быть выполнены на различных многопроцессорных вычислительных системах, разными методами, для различных исходных данных. Для возможности

сравнения результатов выполненных численных экспериментов система ПараЛаб обеспечивает ведение **Журнала экспериментов** и обеспечивает разнообразные способы представления данных журнала в виде форм, удобных для проведения анализа.

13.6.1. Общие результаты экспериментов

Накопление итогов экспериментов производится системой ПараЛаб автоматически. О каждом проведенном эксперименте хранится исчерпывающая информация: дата и время проведения, детальное описание вычислительной системы и решаемой задачи, время, потребовавшееся для выполнения эксперимента. Следует отметить, что повторение эксперимента с идентичными исходными установками приведет к повторному учету результатов.

При просмотре итогов предоставляется возможность восстановления эксперимента по сохраненной записи. Можно выполнять операции удаления записи и очистки списка итогов.

13.6.2. Просмотр итогов

Для отображения итогов экспериментов в системе ПараЛаб существует окно визуализации **Журнала экспериментов**. Данное окно содержит **Таблицу итогов** и **Лист графиков**.

Каждая строка **Таблицы итогов** (см. рис. 13.21) представляет один выполненный эксперимент. По умолчанию, в таблице итогов выделена первая строка и по ней в области **Лист графиков** построен график зависимости времени выполнения эксперимента от объема исходных данных на листе графиков. Для того чтобы изменить вид отображаемой зависимости, нужно выбрать соответствующие пункты в списках, расположенных в левом верхнем и нижнем правом углу листа графиков. Можно построить зависимости времени выполнения эксперимента и ускорения от объема исходных данных, количества узлов, количества процессоров в узле, количества ядер в процессоре, производительности процессоров и характеристик сети. Выделяя различные строки таблицы, Вы можете просматривать графики, соответствующие различным экспериментам.

Следует отметить, что при построении графиков для экспериментов, проведенных в режиме имитации, используются необходимые аналитические зависимости (см. главы 6–10). Для экспериментов, проведенных на вычислительном кластере, используется набор полученных к данному моменту результатов реальных экспериментов.

При выделении нескольких строк в таблице результатов на листе графиков отображается несколько зависимостей. Цвет линии графика соот-

ветствует тому цвету, которым выделена левая ячейка строки, по которой построена эта зависимость.

При переходе к выполнению экспериментов, результаты которых не могут быть сопоставлены с итогами ранее проведенных вычислений, в системе ПараЛаб предусмотрена возможность очистить список итогов

Правила использования системы ПараЛаб

1. Общие результаты. Для демонстрации накопленных результатов экспериментов следует выбрать пункт меню **Результаты**, выделить команду **Показать** и выполнить одну из двух команд: **Из активного окна** или **Из всех окон**. При выполнении первой команды будут отображены результаты, накопленные в активном окне вычислительного эксперимента. При выполнении второй команды – результаты из всех открытых окон экспериментов. Вид диалогового окна с результатами экспериментов представлен на рис. 13.21. Это окно содержит таблицу результатов и лист графиков.

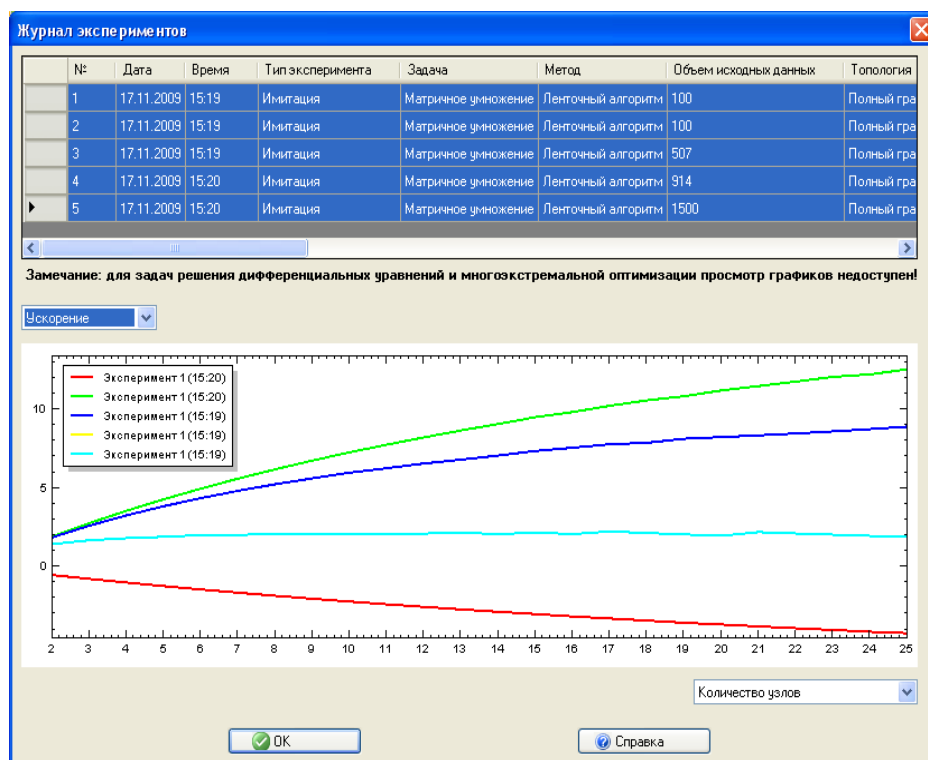


Рис. 13.21. Анализ результатов экспериментов

2. Выделение строки в таблице результатов. Каждая строка таблицы представляет один выполненный эксперимент. Для выделения строки на-

ведите указатель мыши на нужную строчку и нажмите левую кнопку мыши. Также можно воспользоваться курсорными стрелками вверх и вниз (выделенной строкой станет соответственно предыдущая или следующая строка). Если выделенная строчка одна, то на листе графиков отображается только зависимость, соответствующая выделенной строке.

3. Выделение нескольких строк в таблице результатов. Чтобы выделить несколько подряд идущих строк в таблице итогов, нажмите **Shift** и выделите мышью первую и последнюю строчку желаемого диапазона. Для выделения нескольких строк, не образующих непрерывную последовательность, нажмите **Ctrl** и выделяйте строки в произвольном порядке. Для того, чтобы выделить несколько строк при помощи курсорных клавиш, нажмите **Shift** и перемещайтесь по таблице при помощи клавиш вверх и вниз. При выделении нескольких строк в таблице результатов на листе графиков отображается несколько зависимостей. Цвет линии графика соответствует тому цвету, который сопоставлен в легенде проводимого эксперимента.

4. Восстановление эксперимента по записи в таблице итогов. Как уже отмечалось выше, запись в таблице итогов содержит исчерпывающую информацию о вычислительном эксперименте. Для восстановления эксперимента по записи необходимо выделить эту запись одним из перечисленных способов, щелкнуть правой кнопкой мыши в области списка итогов и выполнить команду **Восстановить эксперимент** появившегося контекстного меню. Эксперимент будет восстановлен в активном окне.

5. Печать таблицы итогов. Для печати списка итогов на печатающем устройстве щелкните правой кнопкой мыши в области таблицы и выполните команду **Печать** появившегося контекстного меню.

6. Удаление записи. Для удаления выделенной записи выполните команду **Удалить запись** контекстного меню списка итогов.

7. Удаление результатов. Для удаления накопленных результатов и перехода к построению новых оценок выполните команду **Очистить список** контекстного меню списка итогов.

8. Изменение вида зависимости на листе графиков. Для того, чтобы изменить вид зависимости, изображенной на листе графиков, выберите нужные значения в списках, расположенных слева сверху и справа внизу от листа графиков. Нижний правый список позволяет выбрать аргумент зависимости, а левый верхний – функцию.

9. Копирование листа графиков в буфер обмена. Для копирования графического изображения листа графиков в буфер обмена Windows выполните команду **Копировать в буфер обмена** контекстного меню.

10. Печать листа графиков. Для печати листа графиков на печатающем устройстве выполните команду **Печать** контекстного меню.

Задания и упражнения

Выполните несколько экспериментов с одним и тем же методом умножения матриц, изменяя объем исходных данных и количество процессоров. Используя окно итогов экспериментов, проанализируйте полученные результаты. Постройте одновременно несколько графиков на листе графиков и сравните их.

13.7. Выполнение вычислительных экспериментов

В рамках системы ПараЛаб допускаются разные схемы организации вычислений при проведении экспериментов по изучению и исследованию параллельных алгоритмов решения сложных вычислительных задач. Решение задач может происходить в режиме последовательного исполнения или в режиме разделения времени с возможностью одновременного наблюдения итераций алгоритмов во всех окнах вычислительных экспериментов. Проведение серийных экспериментов, требующих длительных вычислений, может происходить в автоматическом режиме с возможностью запоминания результатов решения для организации последующего анализа полученных данных. Выполнение экспериментов может осуществляться и в пошаговом режиме.

13.7.1. Последовательное выполнение экспериментов

В общем случае цель проведения вычислительных экспериментов состоит в оценке эффективности параллельного метода при решении сложных вычислительных задач в зависимости от параметров многопроцессорной вычислительной системы и от объема исходных данных. Выполнение таких экспериментов может сводиться к многократному повторению этапов постановки и решения задач. При решении задач в рамках системы ПараЛаб процесс может быть приостановлен в любой момент времени (например, для смены графических форм наблюдения за процессом решения) и продолжен далее до получения результата. Результаты решения вычислительных задач могут быть записаны в журнал экспериментов и представлены в виде, удобном для проведения анализа.

Правила использования системы ПараЛаб

1. Проведение вычислительного эксперимента. Для выполнения вычислительного эксперимента выберите пункт меню **Выполнение** и выполните команду **В активном окне**. Решение задачи осуществляется без остановки до получения результата. В ходе выполнения эксперимента ос-

новное меню системы заменяется на меню с командой **Остановить**; после завершения решения задачи основное меню системы восстанавливается.

2. Приостановка решения. Для приостановки процесса выполнения эксперимента следует выполнить в строке меню команду **Остановить** (команда доступна только до момента завершения решения).

3. Продолжение решения. Для продолжения ранее приостановленного процесса выполнения эксперимента следует выполнить команду **Продолжить** пункта меню **Выполнение** (команда может быть выполнена только в случае, если после приостановки процесса поиска не изменялись постановка задачи и параметры вычислительной системы; при невозможности продолжения ранее приостановленного процесса выполнения эксперимента имя данной команды высвечивается серым цветом).

Задания и упражнения

1. В активном окне вычислительного эксперимента установите топологию **Кольцо** и число процессоров, равное десяти. Сделайте текущей задачей задачу сортировки с использованием пузырькового алгоритма.
2. Выполните первые две итерации алгоритма и приостановите процесс вычислений.
3. Измените темп демонстрации и способ отображения пересылки данных.
4. Продолжите выполнение эксперимента до получения результата.

13.7.2. Выполнение экспериментов по шагам

Для более детального анализа итераций параллельного алгоритма в системе ПараЛаб предусмотрена возможность пошагового выполнения вычислительных экспериментов. В данном режиме после выполнения каждой итерации происходит приостановка параллельного алгоритма. Это дает исследователю возможность подробнее изучить результаты проведенной итерации.

Правила использования системы ПараЛаб

1. Пошаговый режим. Для задания режима приостановки вычислительного эксперимента после выполнения каждой итерации следует выполнить команду **Пошаговый режим** пункта меню **Выполнение**. После выполнения этой команды основное меню системы ПараЛаб заменяется на меню пошагового выполнения эксперимента с командами:

- команда **Шаг** – выполнить очередную итерацию поиска,
- команда **Без Остановки** – продолжить выполнение эксперимента без остановки,

- команда **Заккрыть** – приостановить выполнение эксперимента и вернуться к выполнению команд основного меню.

13.7.3. Выполнение нескольких экспериментов

Последовательное выполнение экспериментов затрудняет сравнение результатов итераций параллельных алгоритмов. Для возможности более детального сравнения таких данных система ПараЛаб позволяет демонстрировать на экране дисплея одновременно результаты всех сравниваемых экспериментов. Для этого экран дисплея может разделяться на несколько прямоугольных областей (*окон экспериментов*), в каждой из которых могут высвечиваться результаты отдельно проводимого эксперимента. В любой момент пользователь системы ПараЛаб может создать новое окно для выполнения нового эксперимента. При этом итоги экспериментов и журнал экспериментов формируются отдельно для каждого имеющегося окна. При визуализации окна экспериментов могут разделять экран (в этом случае содержимое всех окон является видимым) или могут перекрываться. Исследователь может выбрать любое окно активным для выполнения очередного эксперимента. Но вычисления могут быть выполнены и во всех окнах одновременно в режиме разделения времени, когда каждая новая итерация выполняется последовательно во всех имеющихся окнах. Используя этот режим, исследователь может наблюдать за динамикой выполнения нескольких экспериментов, результаты вычислений могут быть визуально различимы, и их сравнение может быть выполнено на простой наглядной основе.

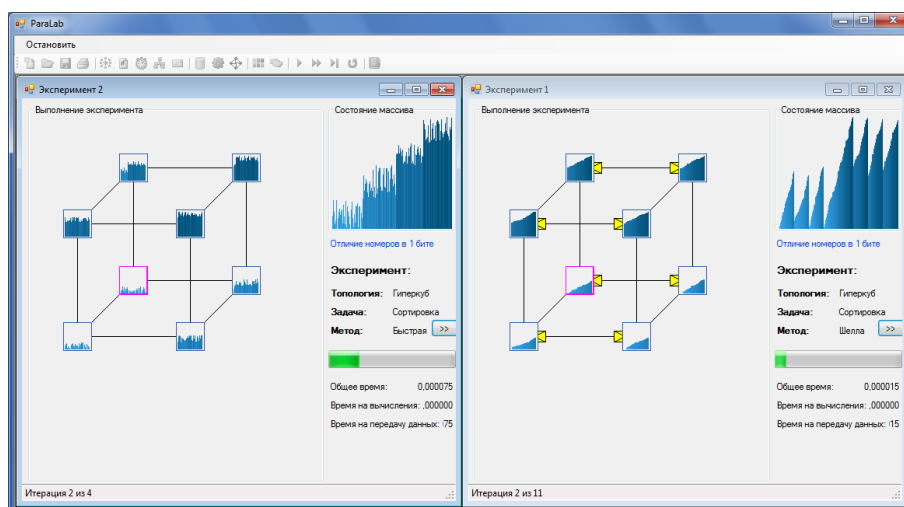


Рис. 13.22. Пример демонстрации нескольких окон экспериментов

Следует отметить, что итоги экспериментов, проведенных в разных окнах, могут высвечиваться совместно в одной и той же таблице итогов (см. п. 13.6.2). Это возможно и для данных из журналов экспериментов, соответствующих различным окнам.

Правила использования системы ПараЛаб

1. Создание окна. Для создания окна для проведения экспериментов следует выполнить команду **Создать новый** пункта меню **Архив**. Закрывание окна эксперимента производится принятыми в операционной системе Windows способами (например, путем нажатия кнопки закрытия окна в правом верхнем углу окна).

2. Управление окнами. Управление размерами окон экспериментов осуществляется принятыми в системе Windows способами (максимизация, минимизация, изменение размеров при помощи мыши). Для одновременного показа всех имеющихся окон без перекрытия можно использовать команду **Показать все** пункта меню **Окно**; для выделения большей части экрана для активного окна (но при сохранении возможности быстрого доступа ко всем имеющимся окнам) следует применить команду **Расположить каскадом** пункта меню **Окно**.

3. Проведение экспериментов во всех окнах. Для выполнения вычислительных экспериментов во всех имеющихся окнах в режиме разделения времени (т.е. при переходе к выполнению следующей итерации только после завершения текущей во всех имеющихся окнах) следует применить команду **Во всех окнах** пункта меню **Выполнение**. Управление процессом вычислений осуществляется так же, как и при использовании единственного окна (приостановка выполнения алгоритмов по команде **Остановить**, продолжение вычислений по команде **Продолжить** пункта меню **Выполнение**).

4. Сравнение итогов экспериментов. Для того, чтобы свести в одну таблицу итогов результаты, полученные во всех окнах экспериментов, выполните последовательность команд **Результаты**→**Показать**→**Из всех окон**.

Задания и упражнения

1. Откройте второе окно вычислительного эксперимента, установите режим показа окон без перекрытия.

2. В первом окне выберите метод пузырьковой сортировки. Во втором окне установите топологию **Гиперкуб** и выберите метод сортировки Шелла. Выполните копирование вычислительной системы в первое окно.

3. В обоих окнах установите режим автозаписи результатов в журнал экспериментов.

4. Выполните вычислительные эксперименты одновременно в обоих окнах; отрегулируйте скорость демонстрации установкой подходящего темпа показа.

5. Получите сводную таблицу итогов. Сравните временные характеристики алгоритмов пузырьковой сортировки и сортировки Шелла.

13.7.4. Выполнение серии экспериментов

ПараЛаб обеспечивает возможность автоматического (без участия пользователя) выполнения серий экспериментов, требующих проведения длительных вычислений. При задании этого режима работы системы пользователь должен выбрать окно, в котором будут выполняться эксперименты, установить количество экспериментов и выбрать тот параметр, который будет изменяться от эксперимента к эксперименту (объем исходных данных или количество процессоров). Результаты экспериментов могут быть запомнены в журнале экспериментов, а в последующем проанализированы.

Правила использования системы ПараЛаб

1. Выполнить серию. Переход в режим выполнения последовательности экспериментов осуществляется при помощи команды **Серия экспериментов** пункта меню **Выполнение**. При выполнении команды может быть задано число экспериментов в серии, а также выбран тип серии: исследуется ли зависимость времени и ускорения решения поставленной задачи от объема исходных данных, от количества используемых процессоров, от количества процессоров в узле или от количества ядер на процессоре.

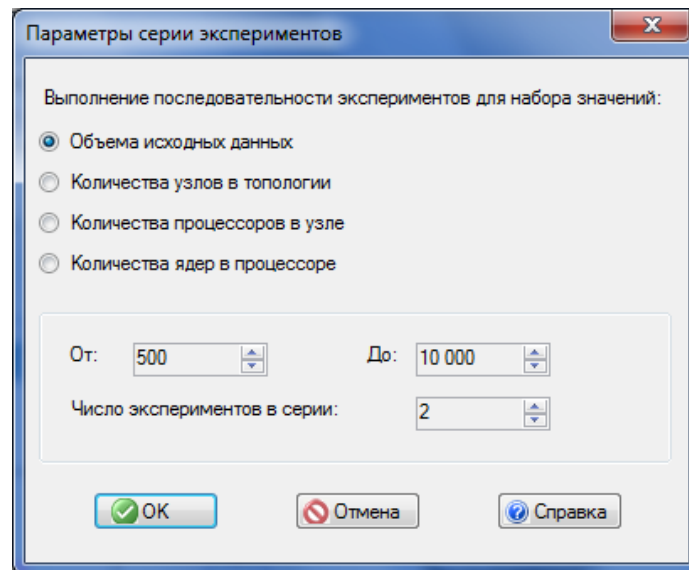


Рис. 13.23. Диалоговое окно задания параметров серии

При выполнении серии экспериментов основное меню системы ПараЛаб заменяется на меню управления данным режимом вычислений, команды которого позволяют:

- команда **Запуск** – выполнить последовательность экспериментов,
- команда **Закрыть** – приостановить выполнение данного режима и вернуться к выполнению команд основного меню,
- команда **Справка** – получение дополнительной справочной информации.

При решении серии поставленных задач (после выполнения команды **ОК**) выполнение эксперимента может быть приостановлено в любой момент времени при помощи команды **Остановить**.

13.7.5. Выполнение реальных вычислительных экспериментов

Помимо выполнения экспериментов в режиме имитации, в системе ПараЛаб предусмотрена возможность проведения реальных экспериментов в режиме удаленного доступа к вычислительному кластеру. При выборе этого режима выполнения эксперимента необходимо поставить задачу и выбрать нужное количество процессоров для ее решения. После выполнения имитационных и реальных экспериментов пользователь ПараЛаб может сравнить результаты и оценить точность используемых

в системе теоретических моделей времени выполнения параллельных алгоритмов. Результаты реальных экспериментов автоматически заносятся в таблицу итогов, кроме того, они могут быть запомнены в журнале экспериментов.

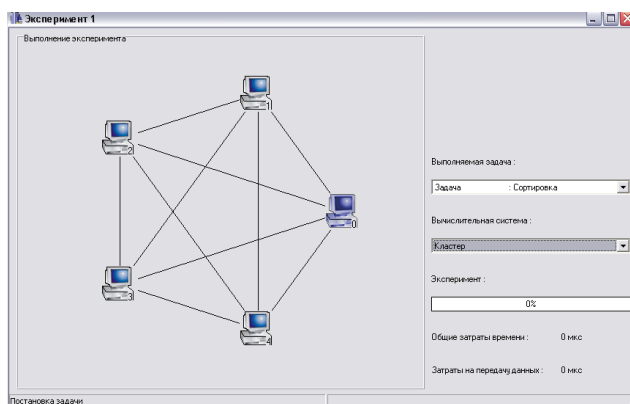


Рис. 13.24. Окно для выполнения реального вычислительного эксперимента

При выполнении реального эксперимента ход вычислений и обмен данными между процессорами не отображаются. В списке параметров вычислительной системы присутствует только строка, указывающая на то, что выполняется эксперимент в режиме удаленного доступа к кластеру, и указывается число процессоров. Режим пошагового выполнения эксперимента недоступен.

Правила использования системы ПараЛаб

1. Переход в режим реального выполнения эксперимента. Для перехода в режим выполнения реальных вычислительных экспериментов в режиме удаленного доступа к вычислительному кластеру выберите пункт меню **Система** и выделите мышью команду **Кластер**. Подтверждением того, что данный режим активен, является значок ☒ слева от надписи. После выбора этого режима топология вычислительной системы автоматически заменяется на топологию **Полный граф**, так как последняя соответствует топологии кластера. Постановка задачи осуществляется так же, как и при выполнении экспериментов в режиме имитации.

2. Задание количества вычислительных узлов (процессоров). Для выбора числа процессоров выполните команду **Количество процессоров** пункта меню **Система**. В появившемся диалоговом окне при помощи бегунка задайте нужное число процессоров. Нажмите **ОК** (Enter) для под-

тверждения выбора или **Отмена** (Escape) для возврата в основное меню системы без изменений.

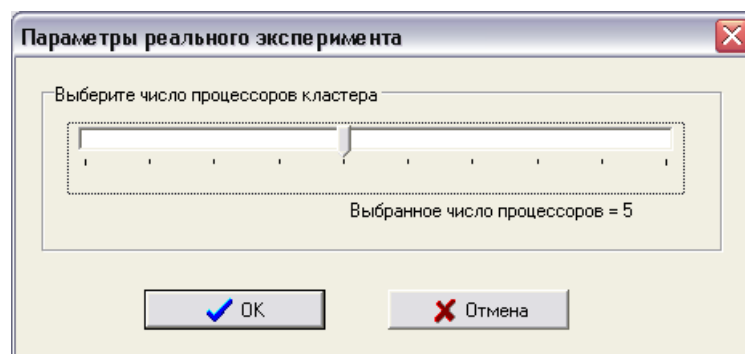


Рис. 13.25. Окно выбора количества вычислительных узлов

3. Проведение реального эксперимента. Для проведения реального вычислительного эксперимента выполните команду **В** активном окне пункта меню **Выполнение**.

13.8. Использование результатов экспериментов: запоминание, печать и перенос в другие программы

13.8.1. Запоминание результатов

В любой момент результаты выполненных в активном окне вычислительных экспериментов могут быть сохранены в архиве системы ПараЛаб. Данные, сохраняемые для окна проведения эксперимента, включают:

- параметры активной вычислительной системы (топология, количество процессоров, производительность процессора, время начальной подготовки данных, пропускная способность сети, метод передачи данных),
- постановку задачи (тип задачи, размер исходных данных, метод решения).

Данные, сохраненные в архиве системы, в любой момент могут быть восстановлены из архива и, тем самым, пользователь может продолжать выполнение своих экспериментов в течение нескольких сеансов работы с системой ПараЛаб.

Кроме того, в рамках системы ПараЛаб исследователю предоставляется возможность сохранения в архиве и чтения из архива сформированных графов специального вида (см. п. 13.4.5).

Правила использования системы ПараЛаб

1. Запись данных. Для сохранения результатов выполненных экспериментов следует выполнить команду **Сохранить** пункта меню **Архив**. При выполнении записи в диалоговом окне следует задать имя файла, в котором будут сохранены данные. Расширение имени файла может не указываться. Файлы с параметрами вычислительных экспериментов имеют расширение **.prl**.

2. Чтение данных. Для чтения параметров экспериментов, записанных ранее в архив системы ПараЛаб, следует выбрать пункт меню **Архив** и указать команду **Загрузить**. После выполнения этой команды в активное окно будут загружены параметры вычислительного эксперимента, сохраненные в выбранном файле.

Задания и упражнения

Выполните вычислительные эксперименты, план проведения которых состоит в следующем:

1. Выполните какой-либо эксперимент и сохраните параметры выполненного эксперимента в архиве системы.
2. Завершите выполнение системы.
3. Выполните повторный запуск системы и загрузите запомненные параметры эксперимента из архива.

13.8.2. Печать результатов экспериментов

При выполнении экспериментов в системе ПараЛаб получаемые результаты могут быть напечатаны в виде разнообразных графических и табличных форм. Пользователь системы может напечатать:

- таблицы итогов, сохраненных в журнале экспериментов,
- графические формы, являющиеся точными копиями содержимого окон проведения экспериментов,
- графические формы листа графиков из формы представления итогов экспериментов,
- графические формы, представляющие окно редактора графов.

Для печати результатов экспериментов могут быть использованы также стандартные возможности печати системы Windows при помощи копирования содержимого экрана.

Правила использования системы ПараЛаб

1. Печать таблицы результатов экспериментов. Для печати таблицы следует открыть окно представления итогов экспериментов (выполнить

последовательность команд: **Результаты→Показать→Из активного окна** или **Результаты→Показать→Из всех окон**), вызвать контекстное меню, связанное с таблицей итогов (щелкнуть правой кнопкой мыши в области таблицы), и выполнить команду **Печать**.

2. Печать графической формы листа графиков. Для печати листа графиков следует открыть окно представления итогов экспериментов, вызвать контекстное меню листа графиков и выполнить команду **Печать**.

3. Печать окон экспериментов. Для печати содержимого активного окна эксперимента следует выполнить команду **Печать** пункта меню **Архив**.

4. Печать окна редактора графов. Для печати содержимого окна редактора графов выполните команду **Печать** пункта меню **Файл** этого окна.

Задания и упражнения

Выполните эксперименты и напечатайте графические формы окон экспериментов и таблицы итогов экспериментов.

13.8.3. Копирование результатов в другие программы

При выполнении вычислительных экспериментов в системе ПараЛаб получаемые результаты могут быть скопированы в буфер обмена системы Windows в графическом формате и могут, тем самым, быть перемещены в любые другие программы системы Windows для последующего анализа и обработки. В буфер обмена могут быть скопированы:

- Графики зависимостей времени и ускорения от других параметров вычислительной системы.

Графическое представление результатов экспериментов может быть далее использовано в графических редакторах типа Paint, Photoshop или Corel Draw.

Приложение: Таблица выполнимости методов на различных топологиях в системе ПараЛаб

Метод	Линейка	Кольцо	Звезда	Решетка	Гиперкуб	Полный граф
Умножение матрицы на вектор						
Горизонтальное разбиение						
Вертикальное разбиение						
Блочное разбиение						
Матричное умножение						
Ленточный алгоритм						
Алгоритм Фокса						
Алгоритм Кэннона						
Решение систем линейных уравнений						
Алгоритм Гаусса						
Метод сопряженных градиентов						
Сортировка						
Пузырьковая						
Модифицированная пузырьковая						
Шелла						

Быстрая						
Обработка графов						
Алгоритм Прима						
Алгоритм Флойда						
Алгоритм Дейкстры						
Решение диффер. уравнений						
Метод Гаусса-Зейделя						
Многоэкстремальная оптимизация						
Индексный						