Министерство образования Российской Федерации
Нижегородский государственный университет
им. Н.И. Лобачевского

В.П. Гергель, А.А. Лабутина

П А Р А Л А Б
ПРОГРАММНАЯ СИСТЕМА ДЛЯ ИЗУЧЕНИЯ И ИССЛЕДОВАНИЯ МЕТОДОВ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

Учебное пособие

Нижний Новгород
Издательство Нижегородского госуниверситета
2004

УДК 004.421.2
ББК 32.973.26-018.2
Г 37

Г 37 Гергель В.П., Лабутина А.А. ПараЛаб. Программная система для изучения и исследования методов параллельных вычислений. Учебное пособие – Нижний Новгород: Изд-во ННГУ им. Н.И. Лобачевского, 2003. 125 с.

В учебном пособии описывается учебно-исследовательская система ПараЛаб для проведения вычислительных экспериментов с целью изучения и исследования методов параллельных вычислений.

Использование системы ПараЛаб в лабораторном практикуме обеспечивает возможность моделирования многопроцессорных 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.

Учебное пособие предназначено для широкого круга студентов, аспирантов и специалистов, желающих изучить и практически использовать параллельные компьютерные системы для решения вычислительно трудоемких задач.

. ББК 32.973.26-018.2

ISBN 5-85746-738-1

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 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 2000 или MS Windows XP (режим многозадачной имитации параллельных вычислений). Кроме режима имитации, в системе ПараЛаб обеспечивается 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 ПараЛаб 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.

1. Общая характеристика системы

Возможности системы. ПараЛаб - программный комплекс, который позволяет проводить как реальные параллельные вычисления на многопроцессорной вычислительной системе, так и имитировать такие эксперименты на одном последовательном компьютере с визуализацией процесса решения сложной вычислительной задачи.

При проведении имитационных экспериментов ПараЛаб предоставляет возможность для пользователя:

- *определить топологию* параллельной вычислительной системы для проведения экспериментов, *задать число процессоров* 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*, *эффективности*) 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 (многопоточное выполнение эксперимента); для данной библиотеки имеются общедоступные реализации, которые могут быть получены в сети Интернет и установлены на компьютере под управлением операционных систем MS Windows 2000 или MS Windows XP (требование данного типа операционных систем определяется условиями разработки системы ПараЛаб),

- *на реальной многопроцессорной кластерной вычислительной системе*,

- *в режиме удаленного доступа* к вычислительному кластеру.

Если проводится реальный эксперимент на многопроцессорной вычислительной системе или в режиме удаленного доступа, система ПараЛаб предоставляет возможность выбора типов вычислительных узлов (например, кластер Нижегородского университета состоит из однопроцессорных рабочих станций и многопроцессорных серверов с общей памятью).

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 [1-3]. Для реальных экспериментов на многопроцессорных вычислительных системах зависимости строятся по набору результатов проведенных вычислительных экспериментов. Любой из проведенных ранее экспериментов может быть восстановлен для повторного проведения. Кроме того, обеспечена возможность ведения журнала экспериментов с записью туда постановки задачи, параметров вычислительной системы и полученных результатов.

Реализованные таким образом процессы изучения и исследований позволят 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. 1). В этом окне демонстрируется решение задачи умножения матриц при помощи ленточного алгоритма.

В области "Выполнение эксперимента" представлены процессоры вычислительной системы и структура линий коммутации. Рядом с каждым процессором изображены те данные, которые он обрабатывает в каждый момент выполнения алгоритма (для ленточного алгоритма умножения матриц – это несколько последовательных строк матрицы A и несколько последовательных столбцов матрицы B). 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.

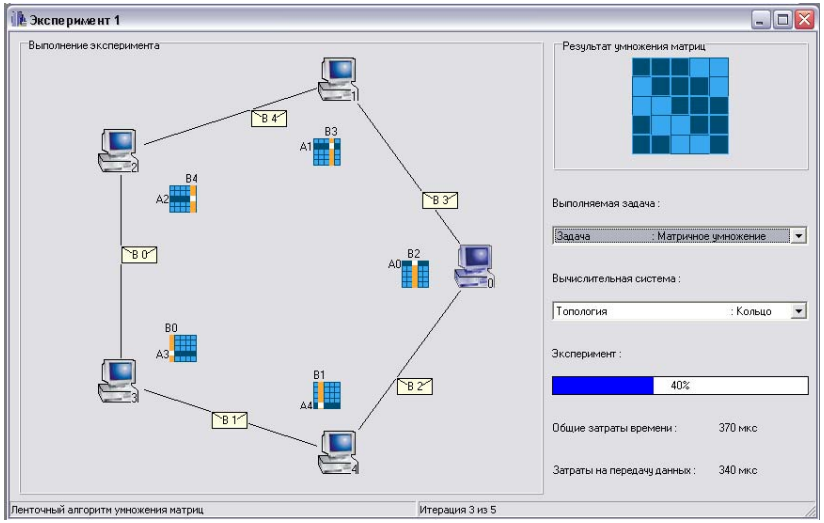


Рис. 1. Окно вычислительного эксперимента

В списке "Выполняемая задача" представлены параметры решаемой задачи: название, метод решения, объем исходных данных. В списке "Вычислительная система" приводятся атрибуты выбранной вычислительной системы: топология, количество и производительность процессоров, характеристики сети.

Ленточный индикатор "Эксперимент" отображает текущую стадию выполнения алгоритма. В строках "Общие затраты времени" и "Затраты 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 **Завершить**.

2. Формирование модели вычислительной системы

Для формирования модели вычислительной системы необходимо определить топологию сети, количество процессоров, 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 – одинаковыми характеристиками.

2.1. Выбор топологии сети

Топология сети передачи данных определяет структуру линий коммутации между процессорами вычислительной системы. В системе ПараЛаб обеспечивается поддержка следующих типовых топологий [3]:

- **полный граф** (*completely-connected graph* or *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* or *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;
- **решетка** (*mesh*) – система, в которой граф линий связи образует прямоугольную двухмерную сетку; подобная топология может быть достаточно просто реализована и, кроме того, может быть эффективно используема 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. 2) щелкните левой клавишей мыши на пиктограмме нужной топологии или внизу в области соответствующей круглой кнопки выбора (радиокнопки). При нажатии кнопки **Помощь** можно получить справочную информацию о реализованных топологиях. Нажмите кнопку **ОК** для подтверждения выбора и кнопку **Отмена** для возврата в основное меню системы ПараЛаб.

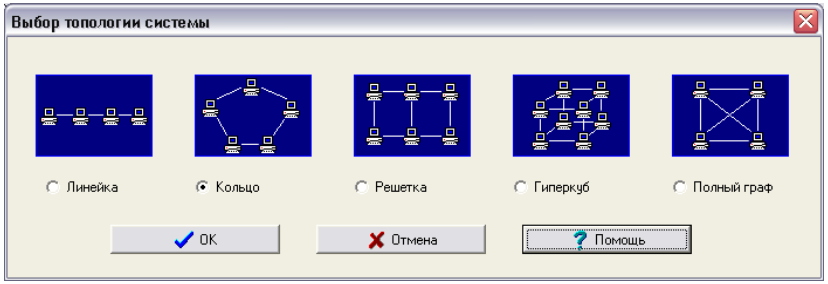


Рис. 2. Диалоговое окно для выбора топологии

2.2. Задание количества процессоров

Для выбранной топологии система ПараЛаб позволяет установить необходимое количество процессоров. Выполняемый при этом выбор конфигурации системы осуществляется в соответствии с типом используемой топологии. Так, например, число процессоров в двухмерной решетке должно являться произведением целых чисел (размеров решетки по диагонали и вертикали), а число процессоров в гиперкубе – степенью числа 2.

Под *производительностью процессора* 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.

Задание количества процессоров.

Для выбора числа процессоров необходимо выполнить команду **Количество_Процессоров** пункта меню **Система**. В появившемся диалоговом окне (рис. 3) Вам предоставляется несколько пиктограмм со схематическим изображением числа процессоров в активной топологии. Для выбора щелкните левой клавишей мыши на нужной пиктограмме. Пиктограмма, соответствующая текущему числу процессоров, выделена ярко-синим цветом.

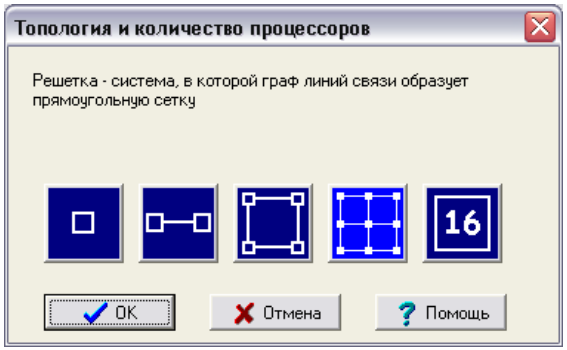


Рис. 3. Диалоговое окно для задания числа процессоров

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

2. Определение производительности процессора.

Для задания производительности процессоров, составляющих многопроцессорную вычислительную систему, следует выполнить команду **Производительность Процессора** пункта меню **Система**. Далее в появившемся диалоговом окне (рис. 4) при помощи бегунка 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**).

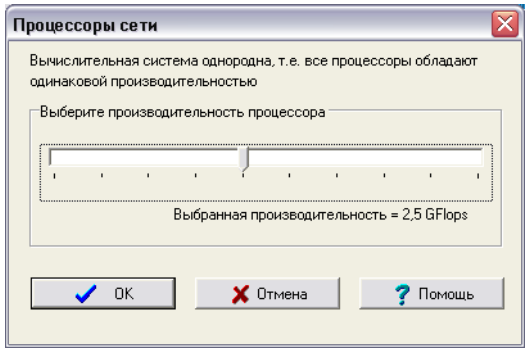


Рис. 4. Диалоговое окно для задания производительности процессора

2.3. Задание характеристик сети

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

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

– *пропускная способность сети* (R) – 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 [3]. Первый из них ориентирован на *передачу сообщений* (МПС) как неделимых (атомарных) блоков информации (*store-and-forward routing* or *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 $t_{пд}$ для метода передачи сообщения размером m по маршруту длиной l определяется выражением:

$$t_{пд} = t_н + \left(m/R\right) l .$$

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

$$t_{пд} = t_н + \frac{V}{R}\left(l + \left\lceil \frac{m}{V - V_0} \right\rceil \right) = t_н + \frac{V}{R}\left(l + n - 1\right)$$

(скобки $\lceil\ \rceil$ 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.

Определение характеристик коммуникационной среды.

Для определения характеристик сети выполните команду **Характеристики сети** пункта меню **Система**. В открывшемся диалоговом окне (рис. 5) при помощи бегунков можно задать время начальной подготовки данных (латентность) в микросекундах и пропускную способность каналов сети (Мбит/с). Для подтверждения выбора нажмите кнопку **ОК**. Для возврата в основное меню системы ПараЛаб без изменения этих параметров нажмите кнопку **Отмена**.

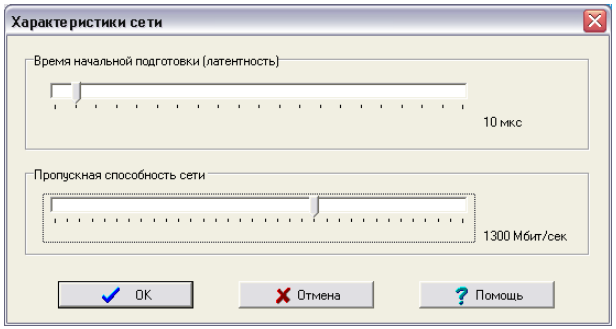


Рис. 5. Диалоговое окно для задания характеристик сети

2 Определение метода передачи данных.

Для определения метода передачи данных, который будет использоваться при проведении вычислительного эксперимента и при построении временных характеристик, необходимо выполнить команду **Метод Передачи Данных** пункта меню **Система**. В открывшемся диалоговом окне (рис. 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**.

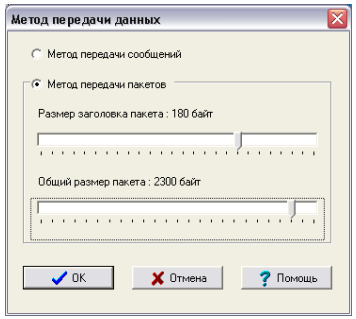


Рис. 6. Диалоговое окно для задания метода передачи данных

3. Завершение работы системы.

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

3. 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, матричного умножения и обработки графов.

Выбор задачи.

Для выбора задачи из числа реализованных в системе выберите пункт меню **Задача** и выделите левой клавишей мыши одну из строк: **Сортировка**, **Матричное умножение**, **Обработка графов**. Выбранная задача станет текущей в активном окне.

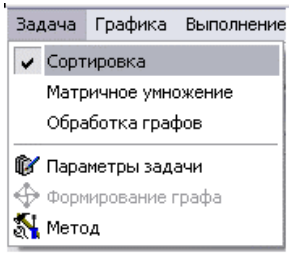


Рис. 7. Выбор задачи

2. Определение параметров задачи.

Основным параметром задачи в системе ПараЛаб является объем исходных данных. Для задачи сортировки – это размер массива, для задачи матричного умножения – 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. 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).

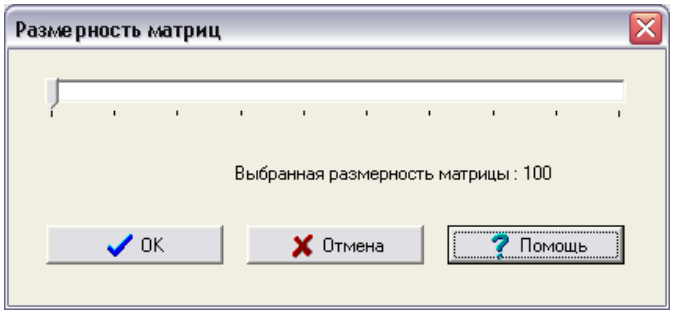


Рис. 8. Диалоговое окно задания параметров задачи в случае решения задачи матричного умножения

3. Определение метода решения задачи.

Для выбора метода решения задачи выполните команду **Метод** пункта меню **Задача**. В появившемся диалоговом окне (рис. 9) выделите мышью нужный метод, нажмите **ОК** для подтверждения выбора, нажмите **Отмена** для возврата в основное меню системы ПараЛаб.

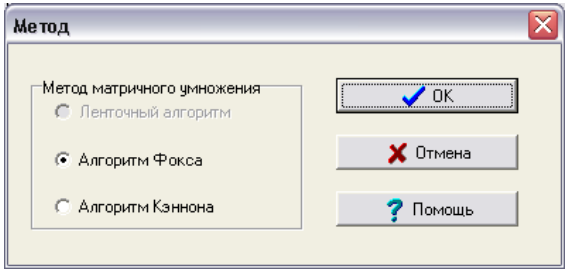


Рис. 9. Диалоговое окно выбора метода в случае решения задачи матричного умножения.

3.1. Сортировка данных

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

неупорядоченного набора значений

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

в порядке монотонного возрастания или убывания

$$S \sim S' = \{(a_1', a_2', ..., a_n') : a_1' \le a_2' \le ... \le a_n'\}$$

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, [7]), способы их параллельного выполнения излагаются в разделе 8 пособия.

3.1.1. 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* [23], для которого на нечетной итерации выполнения сравниваются элементы пар

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

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

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

После n- кратного повторения подобных итераций массив оказывается отсортированным. Более подробная информация о параллельном варианте алгоритма приводится в разделе 8.

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 текущей топологии. Так как при создании эксперимента по умолчанию текущей топологией становится кольцо, 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 процессоров вычислительной системы. Проанализируйте полученные временные характеристики.

3.1.2. 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 понадобиться достаточно большое количество итераций). Подробное описание параллельного обобщения алгоритма сортировки Шелла можно найти в разделе 8.

Задания и упражнения

1. Запустите систему ПараЛаб. Выберите топологию кольцо и установите количество процессоров, равное восьми. Проведите эксперимент по выполнению алгоритма пузырьковой сортировки.
2. Установите в окне вычислительного эксперимента топологию гиперкуб и число процессоров, 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 справа 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.

3.1.3. 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 разделе 8 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 процессоров, 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.

3.2. Матричное умножение

Задача умножения матрицы на матрицу определяется соотношениями:

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

(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:

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

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 выделении для 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.

3.2.1. Ленточный алгоритм.

При ленточной схеме разделения данных исходные матрицы разбиваются на горизонтальные (для матрицы *A*) и вертикальные (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 разделе 8 пособия.

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 метода и убедитесь в том, что выбран метод ленточного умножения матриц.
3. Проведите несколько вычислительных экспериментов. Изучите зависимость времени выполнения алгоритма от объема исходных данных и от количества процессоров.

3.2.2. 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 обеспечить 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 разделе 8 пособия 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 процессоров, 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 процессоров в топологии Решетка на шестнадцать. Последовательно выполните вычислительные

эксперименты с использованием метода Фокса и метода Кэннона. Сравните временные характеристики этих экспериментов.

3.3. 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 [8].

В системе ПараЛаб реализованы параллельные алгоритмы решения двух типовых задач на графах: алгоритм Прима поиска минимального охватывающего дерева, алгоритм Дейкстры поиска кратчайших путей.

Правила использования системы ПараЛаб

1. Переход в режим редактирования графа.

При выборе задачи **Обработка графов** в системе ПараЛаб предусмотрена возможность создания, загрузки и редактирования графа. Для того чтобы перейти в режим редактирования графа, выполните команду **Формирование графа** пункта меню **Задача**. Заметим, что команда доступна только в том случае, когда текущей задачей является задача обработки графов.

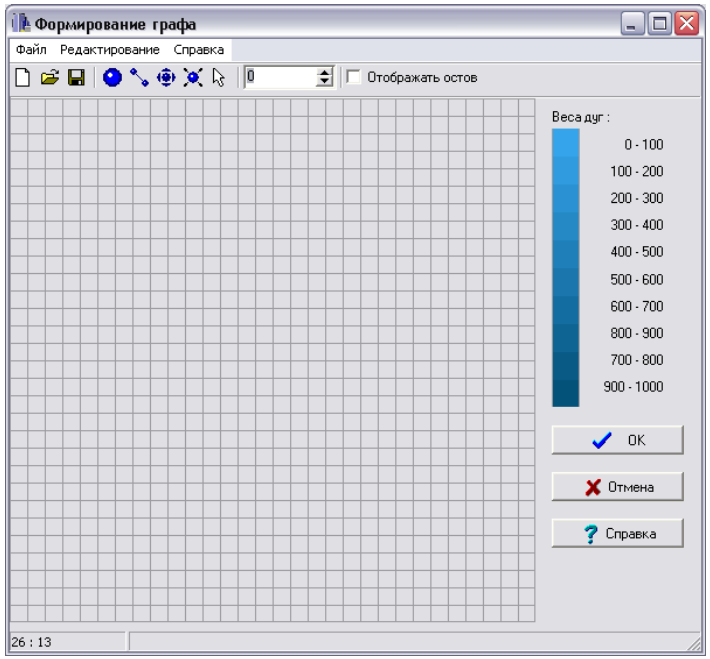


Рис. 10. Окно встроенного редактора графов

После выполнения команды **Формирование графа** на экране дисплея появляется новое окно (рис. 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.

2. Создание нового графа.

Для создания нового пустого графа выберите пункт меню **Файл** и выполните команду **Новый** (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 **Загрузить** (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.

- Для того, чтобы соединить две вершины графа ребром, выберите пункт меню **Редактирование** и выполните команду **Добавить 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. Между первой и второй 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 перемещается вслед за курсором. Щелкните на любой пустой клетке рабочей области, и выделенная вершина переместится в эту точку.

- Для того, чтобы удалить вершину 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 случайным образом. Для этого в редакторе, расположенном на панели инструментов, укажите число вершин графа, далее выберите пункт меню **Редактирование** и выполните команду **Случайное формирование**.

7. Редактирование веса ребра графа.

Цвет ребер графа имеет разную интенсивность. Чем темнее цвет, тем больше вес ребра. Для того, чтобы приблизительно определить вес ребра, нужно сравнить его цвет со шкалой, расположенной справа. Для того, чтобы изменить вес 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**.

3.3.1. 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);

- эта вершина включается в состав МОД.

3.3.2. 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 **Формирование графа** 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** пункта меню **Задача**, в появившемся диалоговом окне выберите **метод Прима**).

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

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

4. 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.

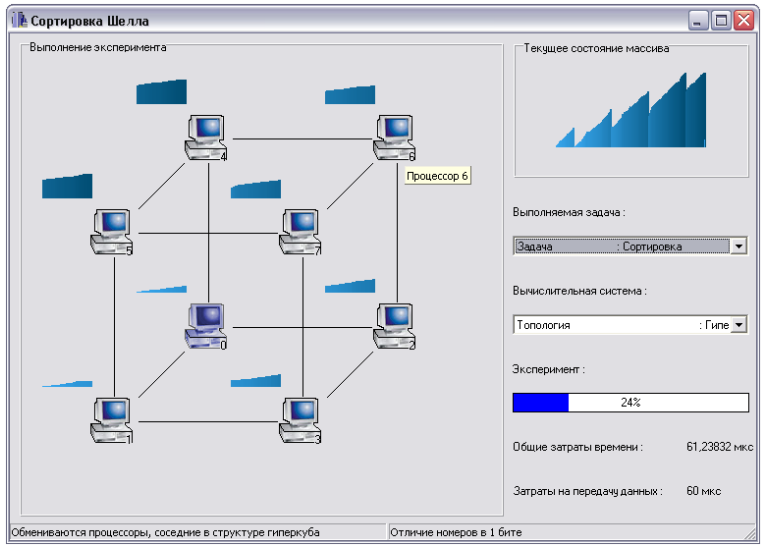


Рис. 11. Вид окна вычислительного эксперимента

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

- “Текущее состояние массива” при выполнении алгоритма сортировки;
- “Результат умножения матриц” при выполнении матричного умножения;
- “Результат обработки графа” при выполнении алгоритмов на графах.

В средней части правой половины окна эксперимента приводятся **сведения о выполняемой задаче**. Здесь же в **списке “Вычислительная система”** указаны характеристики вычислительной системы, такие как топология, количество процессоров, производительность процессоров и характеристики коммуникационной среды.

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 высветки этого окна и выбор наблюдаемого процессора осуществляется пользователем системы).

4.1. Область "Выполнение эксперимента"

В этой области окна изображены процессоры многопроцессорной вычислительной системы, соединенные линиями коммутации в ту или иную топологию. Процессоры в топологии пронумерованы. Для того чтобы узнать номер процессора, достаточно навести на него указатель мыши. Вид указателя изменится и появится подсказка с номером процессора. Если при этом дважды щелкнуть левой клавишей мыши на изображении процессора, то появится окно **“Демонстрация работы процессора”**, где будет детально отображаться деятельность этого процессора.

Около каждого процессора схематически изображаются данные,

которые находятся на нем в данный момент выполнения эксперимента. Если эксперимент состоит в изучении какого-либо параллельного алгоритма сортировки, то рядом с процессором 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 рядом с каждым процессором изображен силуэт матрицы, на котором цветом выделены части исходных данных, располагаемых на процессоре (предполагаем, что исходные матрицы A и B квадратные размерности $n \times n$). Синим цветом помечается часть матрицы A на процессоре (блок или горизонтальная полоса), оранжевым – часть матрицы B (блок или вертикальная полоса). Если же эксперимент состоит в изучении алгоритмов обработки графов, то рядом с каждым процессором 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 каналов**” - выделяется красным та линия коммутации, по которой происходит обмен;

- режим “**Движение пакетов**” - визуализация обмена при помощи движущегося от одного процессора к другому 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.

Изменение способа отображения пересылки данных.

1. Для задания способа отображения коммуникации процессоров выполните команду **Пересылка данных** пункта меню **Графика**. В появившемся списке выделите название желаемого способа отображения.

Рис. 12. Выбор способа отображения пересылки данных

2. Выбор темпа демонстрации.

Для выбора темпа демонстрации необходимо выполнить команду **Темп Демонстрации** пункта меню **Графика**. В появившемся диалоговом окне (рис. 13) предоставляется возможность выбора величины задержки между итерациями алгоритма и скорости движения пакетов (времени выделения канала) при отображении коммуникации процессоров.

Рис. 13. 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 **Шаг визуализации** пункта меню **Графика** (команда доступна только в случае, когда текущей задачей активного окна вычислительного эксперимента является задача обработки графов). В появившемся диалоговом окне установите при помощи ползунка 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 **Настройка цвета** пункта меню **Графика**. В появившемся диалоговом окне (рис. 14) Вы увидите прямоугольник с линейным перетеканием более светлого цвета в более темный и квадрат, залитый цветом выделения. При выполнении алгоритмов сортировки более светлый (левый) цвет используется для отображения минимальных элементов сортируемого массива, а более темный – для отображения максимальных элементов. При выполнении алгоритмов на графах более светлый цвет используется для изображения дуг графа, имеющих минимальный вес, а темный – для дуг максимального веса. Цвет выделения используется при выполнении алгоритмов умножения матриц для отображения в области “Выполнение эксперимента” блоков матриц, расположенных на процессорах, и в алгоритмах на графах для отображения минимального охватывающего дерева (*алгоритм Прима*) и дерева кратчайших путей (*алгоритм Дейкстры*).

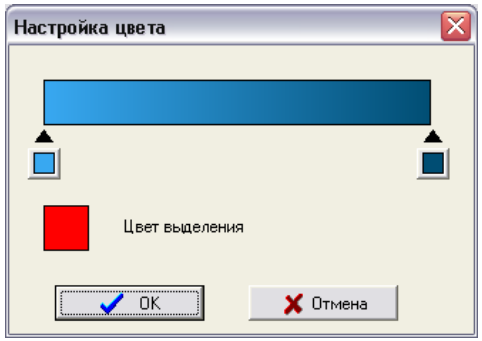


Рис. 14. Диалоговое окно настройки цветовой палитры

Чтобы изменить эти цвета, щелкните левой клавишей мыши на кнопке, расположенной слева или справа под шкалой перетекания. В результате появится стандартное диалоговое окно выбора цвета операционной системы Windows. В этом окне выберите цвет и нажмите кнопку **OK**. 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.

Для изменения цветовой палитры нажмите кнопку **OK** (Enter) в диалоговом окне “Настройка цвета”. Для возврата в основной режим работы системы ПараЛаб нажмите **Отмена** (Escape).

4.2. 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 процессоре с меньшим номером, не должны превосходить элементов, находящихся на процессоре 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.

4.3. Область «Результат умножения матриц»

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

Матрица *C* представляется разбитой на квадратные блоки. Каждый процессор 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 процессоре (этот процессор в области “Выполнение эксперимента” выделен синим цветом). Вычисленные к этому моменту слагаемые написаны темно-синим цветом, вычисляемое на данной итерации – цветом выделения.

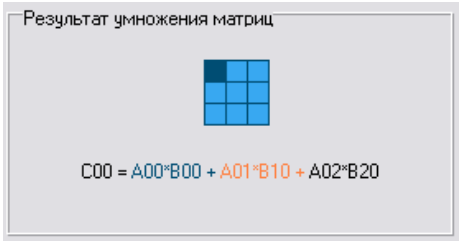


Рис. 15. Область “Результат умножения матриц” при выполнении алгоритма Фокса

4.4. 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*).

4.5. Выбор процессора

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

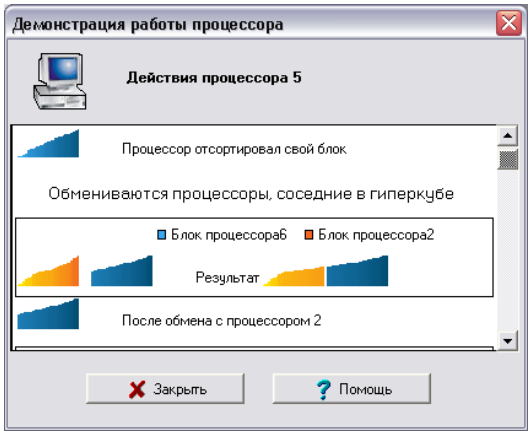


Рис. 16. Вид окна демонстрации работы процессора

Один из способов выбора процессора – выполнить команду **Наблюдение Вычислений** пункта меню **Графика**. 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 (при этом его форма изменится и появится подсказка с номером выбранного процессора) и щелкнуть левой клавишей.

Далее в появившемся окне “Демонстрация работы процессора” будет детально изображаться ход вычислений.

4.6. Визуализация алгоритмов пузырьковой сортировки и сортировки Шелла

Согласно параллельным алгоритмам пузырьковой сортировки и сортировки Шелла, первоначально на каждом процессоре располагается неотсортированный блок исходного массива. Он отображается в первой строке окна “Демонстрация работы процессора”.

Далее на каждом процессоре происходит внутренняя сортировка полученного блока. Она может выполняться при помощи любого

алгоритма сортировки. В системе ПараЛаб при проведении экспериментов используется алгоритм быстрой сортировки. Во второй строке изображается блок массива после сортировки.

Далее выполняется алгоритм чет – нечетной перестановки блоков. На каждой итерации этого алгоритма выбранный процессор осуществляет операцию слияния двух упорядоченных массивов – своего и полученного от следующего или предыдущего (в зависимости от четности номера итерации). Соответственно в окне “Демонстрация работы процессора” отображается:

- блок массива, который находился на процессоре перед началом итерации (изображается синим цветом);
- блок, полученный от соседа (изображается желтым цветом);
- результат слияния этих двух блоков, составленный из двух частей – синей и желтой. Синий блок становится текущим на выбранном процессоре, а желтый отбрасывается.

В последней строке изображен блок массива, который располагается на выбранном процессоре после выполнения необходимого числа итераций. Массив, составленный из блоков в порядке, соответствующем нумерации процессоров, отсортирован.

Задания и упражнения

1. В активном окне вычислительного эксперимента установите топологию гиперкуб, число процессоров – восемь. Сделайте текущей задачей этого окна задачу сортировки. Выберите третий процессор вычислительной системы для наблюдения за его действиями в отдельном окне.

2. Проведите вычислительный эксперимент с использованием алгоритма сортировки пузырьком. Затем – с использованием сортировки Шелла. Сравните порядок обменов данными между процессорами.

4.7. Визуализация быстрой сортировки

Согласно параллельному алгоритму быстрой сортировки, первоначально на процессоре располагается неотсортированный блок исходного массива. Он отображается в первой строке окна “Демонстрация работы процессора”.

На следующих итерациях обмениваются данными процессоры, соседние в структуре гиперкуба. В таком обмене принимают участие два процессора:

- выбирается ведущий элемент;
- блоки процессоров, участвующих в обмене, разбиваются на две части: в первой части находятся элементы, меньшие чем ведущий, во второй – большие значения;
- процессор с меньшим рангом отсылает вторую часть процессору с большим рангом, а процессор с большим рангом, наоборот, отсылает первую часть своего блока процессору с меньшим номером.

В результате на процессоре с большим номером располагаются элементы, большие чем ведущий, а на процессоре с меньшим рангом – меньшие.

В окне “Демонстрация работы процессора” в строке слева изображается блок процессора после обмена с соседом, а затем ведущий элемент, по которому происходил обмен. Соответственно, если изображен блок процессора, ранг которого в текущей итерации больше, чем ранг соседа, то все элементы блока после итерации будут больше (выше) ведущего и, наоборот, на процессоре с меньшим рангом − элементы меньше (ниже) ведущего.

В последней строке изображается блок процессора после выполнения последней итерации алгоритма – локальной сортировки блока.

4.8. Визуализация ленточного алгоритма умножения матриц

Согласно ленточному алгоритму умножения матриц, в каждый момент вычислений на процессоре располагается одна горизонтальная

полоса матрицы *A* и одна вертикальная полоса матрицы *B*. В окне “Демонстрация работы процессора” на каждой итерации изображаются силуэты матриц *A* и *B*. Темно-синим цветом выделены те полосы матриц, которые в данный момент находятся на выбранном процессоре (полоса матрицы *A* всегда одна и та же, полоса матрицы *B* получена на предыдущем шаге от процессора, следующего в структуре кольца).

При перемножении полос матриц получается один блок матрицы *C*. Он отображен на сетке матрицы *C* (правая часть равенства) темно-синим квадратом.

Справа от матричных равенств отображается номер итерации.

Умножение матриц полностью выполнено, если на каждом процессоре получена составленная из блоков полоса матрицы *C*. Для этого необходимо *p* итераций.

4.9. Визуализация алгоритмов Фокса и Кэннона умножения матриц

Согласно алгоритмам Фокса и Кэннона, в каждый момент выполнения вычислений на процессоре находится один блок матрицы *A* и один блок матрицы *B*. Каждый процессор вычисляет один блок матрицы *C*.

На каждой итерации алгоритма происходит перемножение блоков матриц *A* и *B* и прибавление результата такого умножения к результирующему блоку матрицы *C*.

В окне “Демонстрация работы процессора” изображены блоки матриц, расположенные на этом процессоре (рис 17, блоки (2)) и состояние результирующего блока (рис 17, блок (1)). Справа отображается номер итерации.

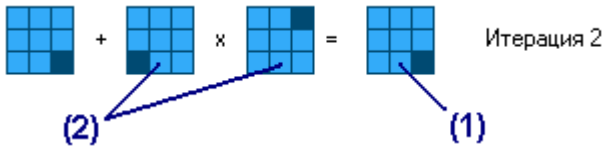


Рис. 17. Отображение блочных алгоритмов матричного умножения в отдельном окне

Алгоритм выполнен, если выполнено $\sqrt{p}$ операций умножения и сложения матричных блоков.

- *Задания и упражнения*

1. В активном окне эксперимента выберите топологию Решетка, число процессоров – девять. Установите задачу матричного умножения при помощи алгоритма Кэннона.

2. Сделайте активным второй процессор вычислительной системы. Выберите движение пакетов в качестве способа отображения пересылки данных, задайте медленный темп показа и проведите вычислительный эксперимент. Обратите внимание на изменение цвета слагаемых, которые отображаются в области “Результат умножения матриц”.

3. Сделайте активным седьмой процессор системы. Включите режим отображения вычислений этого процессора в отдельном окне. Выполните эксперимент. Сравните, как отображаются итерации параллельного алгоритма в области “Результат умножения матриц” и в окне “Демонстрация работы процессора”.

5. 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 содержит различные средства для их накапливания и обеспечивает разнообразные способы представления этих данных в виде форм, удобных для проведения анализа.

5.1. 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.

При сохранении текущего эксперимента в файле происходит сохранение всех записанных результатов.

5.2. Просмотр итогов

Для отображения итогов экспериментов в системе ПараЛаб существует окно, содержащее **Таблицу итогов** и **Лист графиков**.

Каждая строка таблицы итогов (см. рис. 18) представляет один выполненный эксперимент. По умолчанию, в таблице итогов выделена первая строка и по ней 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 процессоров, производительности процессора 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. раздел 8). 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 соответствует тому цвету, которым выделена левая ячейка строки, по которой построена эта зависимость.

Для более детального изучения зависимостей временных характеристик выполнения алгоритма от различных параметров и для сравнения нескольких графиков на листе графиков предусмотрена возможность изменения масштаба.

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

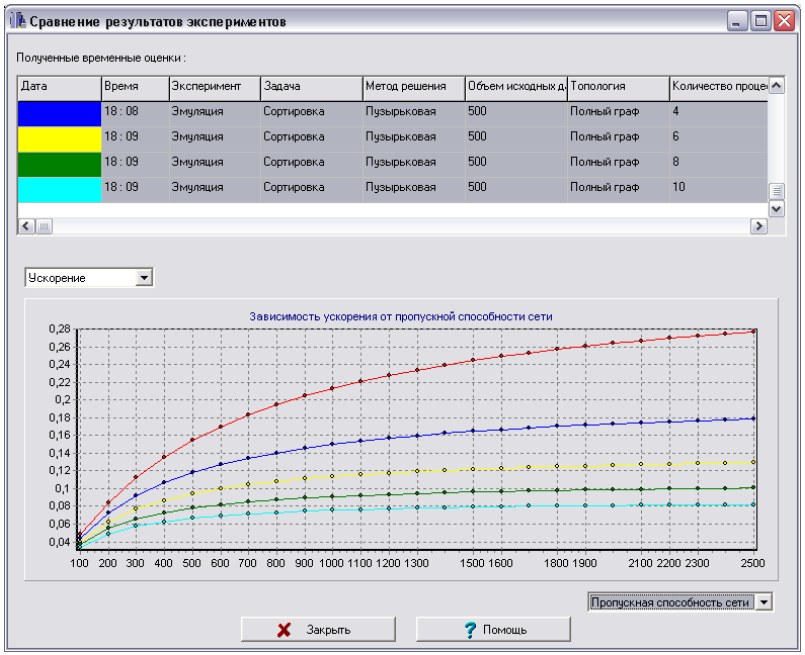


Рис. 18. Анализ результатов экспериментов

Правила использования системы ПараЛаб

1. Общие результаты.

Для демонстрации накопленных результатов экспериментов следует выбрать пункт меню **Результаты**, выделить команду **Итоги** 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. 18. Это окно содержит таблицу результатов и лист графиков.

2. Выделение строки в таблице результатов.

Каждая строка таблицы представляет один выполненный эксперимент. Для выделения строки наведите указатель мыши на нужную строчку и нажмите левую клавишу. 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, которым выделена левая ячейка строки, по которой построена эта зависимость.

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** 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 вверху и справа внизу от листа графиков. Нижний правый список позволяет выбрать аргумент зависимости, а левый верхний – функцию.

9. Изменение масштаба на листе графиков.

Для того чтобы увеличить область листа графиков, выполните команду **Увеличить масштаб** контекстного меню листа графиков (при этом масштаб будет увеличен в 2 раза, отображаться будет геометрический центр области) или выделите эту область: нажмите левую клавишу мыши в левом верхнем углу интересующей области и, не отпуская левую клавишу, переместите указатель в правый нижний угол области. Для возвращения к исходному масштабу щелкните правой кнопкой мыши в области листа графиков и выполните команду **Вернуться к исходному масштабу** или выделите любую область листа графиков, передвигая мышь из правого нижнего угла области в левый верхний.

10. Копирование листа графиков в буфер обмена.

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

11. 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.

5.3. Журнал экспериментов

Для сохранения результатов решения конкретных сложных вычислительных задач система ПараЛаб содержит специальную область памяти, называемую далее *журналом экспериментов*. Данные, записываемые в журнал экспериментов, включают:

- характеристики вычислительной системы (топология, количество процессоров, производительность процессора, время начальной подготовки данных, пропускная способность сети, метод передачи данных);
- постановку задачи (размер исходных данных, метод решения);
- время выполнения эксперимента.

Результаты записываются в журнал либо под управлением пользователя по команде системы, либо же системой автоматически (при установке режима **Автозаписи**). Данные журнала могут демонстрироваться в табличной и графической форме.

Правила использования системы ПараЛаб

1. Записать в журнал.

Для записи результатов последнего выполненного эксперимента в журнал экспериментов выполните последовательно команды: **Результаты→Журнал экспериментов→Записать**. Следует отметить, что повторная запись результатов одного и того же эксперимента не выполняется, однако эксперименты, выполненные с идентичными исходными установками, считаются различными, и их результаты могут быть записаны в журнал экспериментов раздельно.

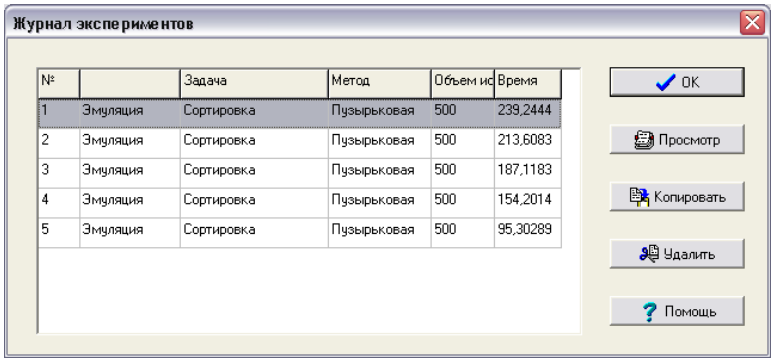


Рис. 19. Окно с табличной формой представления данных из журнала экспериментов

2. Демонстрация журнала.

Для демонстрации журнала экспериментов выполните последовательно команды: **Результат→Журнал экспериментов→Показать→Из активного окна** (или **Из всех окон**). В окне показа журнала (см. рис. 19) предоставляется возможность выполнения следующих действий:

- кнопка **ОК** – для завершения просмотра журнала экспериментов;
- кнопка **Просмотр** – для просмотра параметров эксперимента, соответствующего выделенной строке таблицы;
- кнопка **Копировать** – для записи данных (в текстовом формате) из журнала экспериментов в буфер обмена системы Windows;

- кнопка **Печать** – для печати таблицы на печатающем устройстве;
- кнопка **Помощь** – для получения дополнительной справочной информации.

3. Удаление данных.

Для удаления данных, записанных в журнал экспериментов, следует выполнить последовательно команды: **Результаты→Журнал экспериментов→Обнулить**.

4. Режим Автозаписи.

Для автоматической записи результатов выполняемых экспериментов в журнал экспериментов следует выполнить последовательно команды: **Результаты→Журнал экспериментов→Автозапись** (признаком фиксации данного режима является высветка метки ✔ перед командой **Автозапись**). Результаты экспериментов заносятся в журнал экспериментов в момент окончания решения задачи. Для отмены режима необходимо повторно выполнить команду **Автозапись**.

Задания и упражнения

Выполните следующие задания для освоения правил работы с журналом экспериментов:

- установите режим **Автозаписи** и выполните несколько вычислительных экспериментов по изучению зависимости времени решения задачи сортировки при помощи метода Шелла от числа процессоров;
- проанализируйте результаты экспериментов, записанных в журнал; выделите эксперимент, в котором были получены наилучшие (наихудшие) временные оценки, и рассмотрите топологию вычислительной системы в таких экспериментах;
- выполните печать журнала экспериментов.

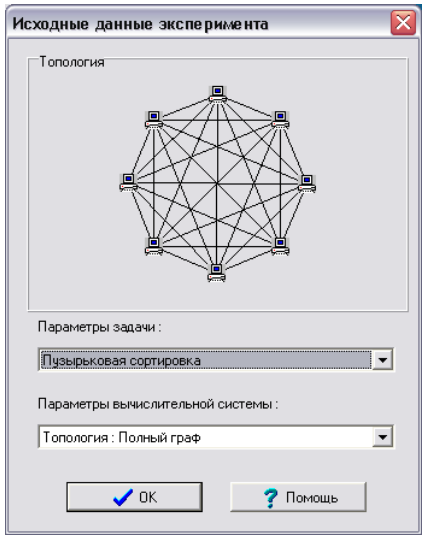


Рис. 20. Просмотр записи в журнале экспериментов

6. 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.

6.1. 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

Проведение вычислительного эксперимента.

1. Для выполнения вычислительного эксперимента выберите пункт меню **Выполнение** и выполните команду **В активном окне**. Решение задачи осуществляется без останова до получения результата. В ходе выполнения эксперимента основное 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, 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.

6.2. 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 **Закрыть -** приостановить выполнение эксперимента и вернуться к выполнению команд основного меню.
-

6.3. 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 раздельно 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.

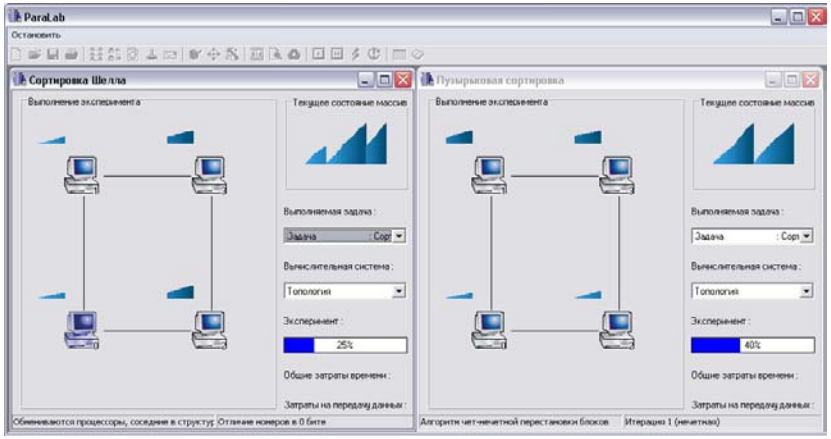


Рис. 21. Пример демонстрации нескольких окон экспериментов

Следует отметить, что итоги экспериментов, проведенных в разных окнах, могут высвечиваться совместно в одной и той же таблице итогов (см. раздел 5). 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 **Эксперимент**. Закрытие окна эксперимента производится принятыми в операционной системе Windows способами (например, путем нажатия кнопки закрытия окна в правом верхнем углу окна). Для одновременного закрытия всех имеющихся окон следует выполнить команду **Закрыть все** пункта меню **Эксперимент**.

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 **Эксперимент**; 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 **Эксперимент**.

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. Копирование параметров задачи и вычислительной системы между окнами.

Для копирования постановки задачи или характеристик вычислительной системы из окна в окно нужно выполнить следующую последовательность действий:

- сделать активным окно, задачу (систему) которого предполагается скопировать; для этого необходимо указать курсором мыши любую точку окна и нажать левую кнопку мыши;
- выполнить команду **Запомнить как образец** пункта меню **Эксперимент**, в появившемся диалоговом окне следует выбрать, производится ли запоминание задачи или вычислительной системы;
- сделать активным окно, в которое необходимо скопировать задачу (систему);
- выполнить команду **Взять образец** пункта меню **Эксперимент**.

Если же одна задача (система) должна быть установлена во всех имеющихся окнах, возможен иной – более быстрый – способ выполнения этой операции. Для этого следует сделать активным окно, задачу (систему) которого предполагается скопировать, и выполнить команду **Копировать во все** окна пункта меню **Эксперимент**.

5. Сравнение итогов экспериментов.

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

6. Сравнение журналов экспериментов.

Для того, чтобы одновременно просмотреть все данные, записанные в журналах экспериментов всех окон, выполните последовательность команд: **Результаты→Журнал экспериментов→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 экспериментов и объединенный журнал экспериментов. Сравните временные характеристики алгоритмов пузырьковой сортировки и сортировки Шелла.

6.4. Выполнение серии экспериментов

ПараЛаб обеспечивает возможность автоматического (без участия пользователя) выполнения длительных 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 списке итогов и журнале экспериментов, а в последующем проанализированы.

Правила использования системы ПараЛаб

1. Выполнить серию.

Переход в режим выполнения последовательности экспериментов осуществляется при помощи команды **Выполнить Серию** 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 или от количества используемых процессоров. Перед заданием режима может оказаться полезной установка автозаписи результатов экспериментов в журнал

экспериментов (Команда **Автозапись** пункта меню **Журнал Экспериментов**).

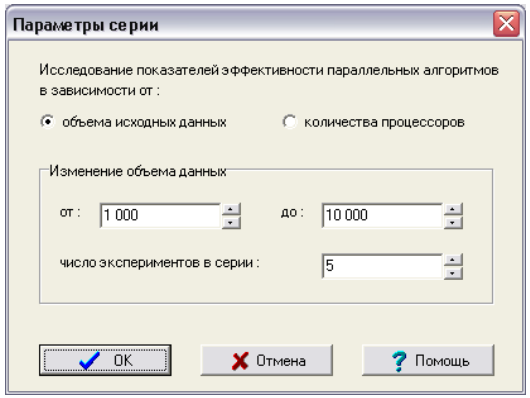


Рис. 22. Диалоговое окно задания параметров серии

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

- команда **Пуск** – 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 **Пуск**) выполнение эксперимента может быть приостановлено в любой момент времени при помощи команды **Остановить**.

6.5. 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.

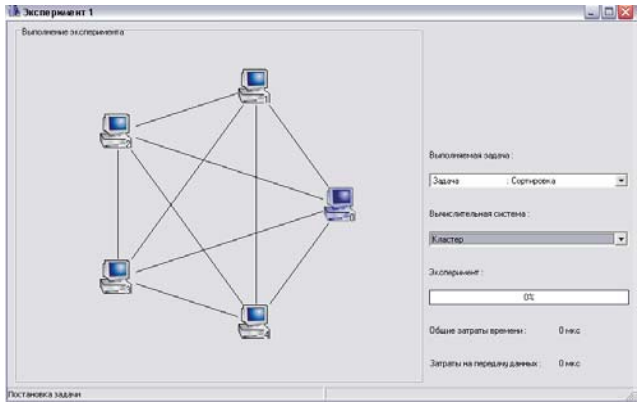


Рис. 23. Окно для выполнения реального вычислительного эксперимента

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

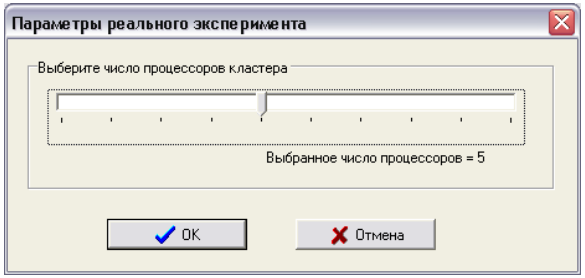


Рис. 24. Окно выбора количества вычислительных узлов

3. Проведение реального эксперимента.

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

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

7.1. 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),
- таблицу результатов, ранее полученных в этом окне,
- данные, записанные в журнал экспериментов.

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. раздел 3).

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 **Сохранить файл как** 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, таблица итогов и журнал экспериментов, 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.

7.2. 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 результатов, сформированные по набору накопленных итогов экспериментов;
- таблицы данных, 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→Итоги→Из активного окна** или **Результаты→Итоги→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 журнала экспериментов.

Для печати результатов, запомненных в журнале экспериментов активного окна, следует последовательно выполнить команды: **Результаты→Журнал экспериментов→Показать→Из активного окна** и в появившемся диалоговом окне Журнал экспериментов нажать кнопку **Печать**.

4. Печать всех журналов экспериментов.

Для печати результатов из всех имеющихся окон, запомненных в журналах экспериментов, следует последовательно выполнить команды: **Результаты→Журнал экспериментов→Показать→Из всех окон** и в появившемся диалоговом окне Журнал экспериментов нажать кнопку **Печать**.

5. Печать окон экспериментов.

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

6. 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.

7.3. Копирование результатов в другие программы

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

- таблицы результатов, сформированные по итогам всех проведенных экспериментов (в текстовом формате);
- таблицы результатов, сформированные по данным из журнала экспериментов (в текстовом формате);
- графическое представление окна вычислительного эксперимента системы ПараЛаб.

Результаты экспериментов, скопированные в буфер обмена в текстовом формате, могут быть далее перенесены в текстовый редактор Word или систему обработки электронных таблиц Excel. Графическое представление окон экспериментов может быть далее использовано в графических редакторах типа Paint, Photoshop или Corel Draw.

Правила использования системы ПараЛаб

1. Копирование таблицы результатов.

Для копирования таблицы результатов экспериментов в буфер обмена системы Windows следует открыть окно представления итогов, вызвать контекстное меню, связанное с таблицей результатов, и выполнить команду **Копировать в буфер обмена**.

2. Копирование журнала экспериментов.

Для копирования данных, сохраненных в журнале экспериментов, в буфер обмена системы Windows откройте диалоговое окно, представляющее журнал экспериментов в табличной форме, и нажмите кнопку **Копировать**.

3. Копирование окна системы ПараЛаб.

Для копирования графического представления окна системы ПараЛаб в буфер обмена системы Windows следует последовательно выполнить команды: **Архив→Копировать в буфер обмена**.

Задания и упражнения

Выполните вычислительные эксперименты, план проведения которых состоит в следующем:

1. Выполните несколько вычислительных экспериментов.
2. Скопируйте данные журнала экспериментов в буфер обмена.
3. Запустите текстовый редактор Word и вставьте в текстовый документ результаты экспериментов из буфера.
4. Запустите систему обработки электронных таблиц Excel и вставьте в таблицу результаты экспериментов из буфера. Преобразуйте средствами системы Excel скопированные данные из текстового формата в числовую форму.
5. Скопируйте в системе ПараЛаб графическое представление окна эксперимента в буфер обмена.

6. Запустите графический редактор Paint и вставьте в рабочую область редактора содержимое буфера обмена.

8. Описание параллельных методов решения сложных вычислительных задач

8.1. Сортировка данных

Общая схема параллельных вычислений при сортировке данных (см. раздел 3 пособия) состоит в разделении исходного упорядочиваемого набора на блоки и их распределения между процессорами, в ходе сортировки блоки пересылаются между процессорами и содержащиеся в них данные сравниваются между собой для упорядочения. Результирующий (отсортированный) 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.

8.1.1. Алгоритм пузырьковой сортировки

Напомним кратко общую схему данного метода упорядочения данных [1]. Алгоритм основан на применении базовой операции "*сравнить и переставить*" (*compare-exchange*), состоящей в сравнении той или иной пары значений из сортируемого набора данных и перестановки этих значений, если их порядок не соответствует условиям сортировки:

```
// операция "сравнить и переставить"
if ( a[i] > a[j] ) {
  temp = a[i];
  a[i] = a[j];
  a[j] = temp;
```

```
}
```

На первой итерации алгоритма осуществляется последовательное сравнение всех соседних элементов; в результате прохода по упорядочиваемому набору данных в последнем (верхнем) элементе оказывается максимальное значение ("всплывание пузырька"); далее для продолжения сортировки этот уже упорядоченный элемент не рассматривается и действия алгоритма повторяются:

```
// пузырьковая сортировка
for ( i=1; i<n; i++ ){
  for ( j=0; j<n-i; j++ )
    <сравнить и переставить элементы (a[j],a[j+1])>
}
```

Алгоритм пузырьковой сортировки в прямом виде достаточно сложен для распараллеливания: сравнение пар соседних элементов происходит строго последовательно. Для организации параллельных вычислений обычно используется модификация алгоритма пузырьковой сортировки – *метод чет-нечетной перестановки* [23]. Суть модификации состоит в том, что в алгоритм сортировки вводятся два разных правила выполнения итераций метода – в зависимости от четности или нечетности номера итерации сортировки для обработки выбираются элементы с четными или нечетными индексами соответственно, сравнение выделяемых значений всегда осуществляется с их правыми соседними элементами, т.е. на всех нечетных итерациях сравниваются пары:

(a_1, a_2), (a_3, a_4),..., (a_{n-1}, a_n) (при четном n),

на четных итерациях обрабатываются элементы

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

После n-кратного повторения подобных итераций сортировки исходный набор данных оказывается упорядоченным.

Параллельное обобщение этого алгоритма не вызывает затруднений, так как сравнение элементов в парах происходит независимо и может выполняться одновременно. Сначала рассмотрим схему вычислений, когда на каждый процессор приходится один элемент исходного массива. Предположим, что процессоры соединены

в кольцо и элементы a_i расположены на процессорах p_i (i=1, 2,..., n). Тогда сравнение пары значений a_i и a_{i+1}, $1 \le i < n$, располагаемых на процессорах P_i и P_{i+1} соответственно, можно организовать следующим образом:

- выполнить взаимообмен имеющихся на процессорах P_i и P_{i+1} значений (с сохранением на этих процессорах исходных элементов);
- сравнить на каждом процессоре P_i и P_{i+1} получившиеся одинаковые пары значений (a_i, a_{i+1}); результаты сравнения используются для разделения данных между процессорами – на одном процессоре (например, P_i) остается меньший элемент, другой процессор (т.е. P_{i+1}) запоминает для дальней обработки большее значение пары

$a_i' = \min(a_i, a_{i+1})$, $a_{i+1}' = \max(a_i, a_{i+1})$.

Рассмотренная параллельная схема может быть надлежащим образом адаптирована и для случая $p < n$, когда количество процессоров является меньшим числа упорядочиваемых значений. В данной ситуации каждый процессор будет содержать уже не единственное значение, а часть (*блок* размера n/p) сортируемого набора данных. Эти блоки обычно упорядочиваются в самом начале сортировки на каждом процессоре в отдельности при помощи какого-либо быстрого алгоритма (предварительная стадия параллельной сортировки). Далее, следуя схеме одноэлементного сравнения, взаимодействие пары процессоров P_i и P_{i+1} для совместного упорядочения содержимого блоков A_i и A_{i+1} может быть осуществлено следующим образом:

- выполнить взаимообмен блоков между процессорами P_i и P_{i+1};
- объединить блоки A_i и A_{i+1} на каждом процессоре в один отсортированный блок двойного размера (при исходной

упорядоченности блоков A_i и A_{i+1} процедура их объединения сводится к быстрой операции слияния упорядоченных наборов данных);

- разделить полученный двойной блок на две равные части и оставить одну из этих частей (например, с меньшими значениями данных) на процессоре P_i, а другую часть (с большими значениями соответственно) – на процессоре P_{i+1}

$$[A_i \cup A_{i+1}]_{сорт} = A_i^{'} \cup A_{i+1}^{'} : \forall a_i^{'} \in A_i^{'}, \forall a_{i+1}^{'} \in A_{i+1}^{'} \Rightarrow a_i^{'} \le a_{i+1}^{'}$$

.

Следует отметить, что сформированные в результате такой процедуры блоки на процессорах P_i и P_{i+1} совпадают по размеру с исходными блоками A_i и A_{i+1} и все значения, расположенные на процессоре P_i, являются меньшими значений на процессоре P_{i+1}.

Рассмотренная процедура обычно именуется в литературе как операция "*сравнить и разделить*" (*compare-split*). Для пояснения такого параллельного способа сортировки на рис. 25 приведен пример упорядочения данных при $n = 8$, $p = 4$ (т.е. блок значений на каждом процессоре содержит $n / p = 2$ элементов). В первом столбце таблицы приводится номер и тип итерации метода, перечисляются пары процессоров, для которых параллельно выполняется операция "сравнить и разделить"; взаимодействующие пары процессоров выделены в таблице двойной рамкой. Для каждого шага сортировки показано состояние упорядочиваемого набора данных до и после выполнения итерации.

Вычислительная трудоемкость алгоритма определяется выражением:

$$T_p = 6 \cdot (n/p)^2 + 2n .$$

Первая часть выражения определяет сложность начальной сортировки блоков с использованием алгоритма пузырьковой сортировки. Вторая часть отражает суммарную сложность всех итераций алгоритма чет-нечетной перестановки блоков (для слияния двух упорядоченных блоков размера *n/p* необходимо *2(n/p)* операций).

№ и тип итерации	Процессоры			
	1	2	3	4
Исходные данные	2 3	3 8	5 6	1 4
1 нечет (1,2),(3.4)	2 3	3 8	5 6	1 4
	2 3	3 8	1 4	5 6
2 чет (2,3)	2 3	3 8	1 4	5 6
	2 3	1 3	4 8	5 6
3 нечет (1,2),(3.4)	2 3	1 3	4 8	5 6
	1 2	3 3	4 5	6 8
4 чет (2,3)	1 2	3 3	4 5	6 8
	1 2	3 3	4 5	6 8

Рис. 25. Пример сортировки данных параллельным методом чет-нечетной перестановки

Коммуникационная трудоемкость алгоритма определяется в зависимости от метода передачи данных:

- Метод передачи сообщений:
  - время передачи данных

$$T_{пд} = p \cdot \left(t_н + \frac{n/p}{R} \right),$$

  - общее время выполнения алгоритма

$$T = \frac{1}{F}\left(6 \cdot (n/p)^2 + 2n\right) + p\left(t_н + \frac{n/p}{R} \right);$$

- Метод передачи пакетов:
  - время передачи данных

$$T_{пд} = p \cdot \left(t_н + \frac{V}{R} \left\lceil \frac{(n/p)}{(V - V_0)} \right\rceil \right),$$

– общее время выполнения алгоритма

$$T = \frac{1}{F}\left(6 \cdot (n/p)^2 + 2n\right) + p \cdot \left(t_н + \frac{V}{R} \left\lceil \frac{(n/p)}{(V - V_0)} \right\rceil \right)$$

(здесь и далее F – производительность процессоров, составляющих многопроцессорную вычислительную систему).

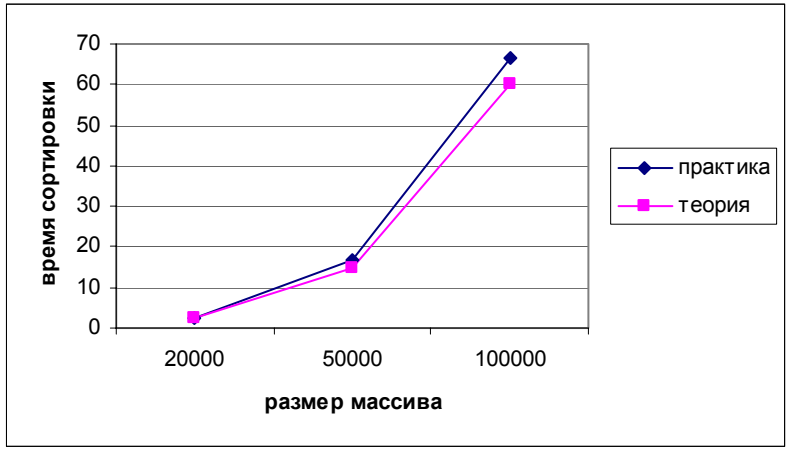


Рис. 26. Зависимость времени сортировки от размера массива

Проведенные на реальной многопроцессорной вычислительной системе эксперименты доказывают справедливость теоретических временных оценок, приведенных выше. На графике представлены теоретическая и практическая зависимости времени сортировки массива от объема исходных данных при выполнении параллельного обобщения алгоритма пузырьковой сортировки.

- *Задания и упражнения*

Проведите вычислительные эксперименты с методом пузырьковой сортировки. Исследуйте зависимость временных характеристик

алгоритма от размера сортируемого массива и от количества процессоров. Для одной и той же вычислительной системы проследите последовательно за вычислениями, которые производят процессоры с четным и нечетным номером.

8.1.2. Алгоритм сортировки 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 понадобиться достаточно большое количество итераций).

Для алгоритма Шелла может быть предложен параллельный аналог метода, если топология коммуникационный сети имеет структуру N-мерного гиперкуба (т.е. количество процессоров равно $p=2^N$). Выполнение сортировки в таком случае может быть разделено на два последовательных этапа. На первом этапе (N итераций) осуществляется взаимодействие процессоров, являющихся соседними в структуре гиперкуба (но эти процессоры могут оказаться далекими при линейной нумерации; для установления соответствия двух систем нумерации процессоров обычно используется код Грея). Второй этап состоит в реализации обычных итераций параллельного алгоритма чет-нечетной перестановки. Итерации данного этапа выполняются до прекращения фактического изменения сортируемого набора и, тем самым, общее количество L таких итераций может быть различным - от 2 до p. Трудоемкость параллельного варианта алгоритма Шелла определяется выражением:

$$T_p = (n/p)\log(n/p) + (2n/p)\log p + L(2n/p),$$

где вторая и третья части соотношения фиксируют вычислительную сложность первого и второго этапов сортировки соответственно. Как можно заметить, эффективность данного параллельного способа сортировки оказывается лучше показателей обычного алгоритма чет-нечетной перестановки при $L<p$.

Коммуникационная трудоемкость алгоритма:

- Метод передачи сообщений:

– время передачи данных

$$T_{пд} = (N+L)\cdot\left(t_н + \frac{n/p}{R}\right),$$

– общее время выполнения алгоритма

$$T = \frac{1}{F}\left((n/p)\log(n/p) + (N+L)2n/p\right) + (N+L)\cdot\left(t_н + \frac{n/p}{R}\right)$$

;

- Метод передачи пакетов:

  – время передачи данных

$$T_{пд} = (N+L)\cdot\left(t_н + \frac{V}{R}\left\lceil\frac{n/p}{V-V_0}\right\rceil\right),$$

  – общее время выполнения алгоритма

$$T = \frac{1}{F}\left((n/p)\log(n/p) + (N+L)2n/p\right) + (N+L)\cdot\left(t_н + \frac{V}{R}\left\lceil\frac{n/p}{V-V_0}\right\rceil\right)$$

.

- *Задания и упражнения*

Убедитесь в том, что алгоритм сортировки Шелла может быть выполнен только в том случае, когда топология вычислительной системы – гиперкуб. Пронаблюдайте за последовательностью обменов данными между процессорами. Проверьте на практике, что для выполнения сортировки методом Шелла в общем случае требуется меньшее число итераций, чем при сортировке пузырьком.

8.1.3. Алгоритм быстрой сортировки

Алгоритм быстрой сортировки [7] основывается на последовательном разделении сортируемого набора данных на блоки меньшего

размера таким образом, что между значениями разных блоков обеспечивается отношение упорядоченности. На первой итерации метода осуществляется деление исходного набора данных на первые две части – для организации такого деления выбирается некоторый ведущий элемент и все значения набора, меньшие ведущего элемента, переносятся в первый формируемый блок, все остальные значения образуют второй блок набора. На второй итерации сортировки описанные правила применяются последовательно для обоих сформированных блоков и т.д. После выполнения *log(n)* итераций исходный массив данных оказывается упорядоченным (при оптимальном выборе ведущих элементов).

Параллельное обобщение алгоритма быстрой сортировки наиболее простым способом может быть получено для вычислительной системы с топологией в виде *N*-мерного гиперкуба (т.е. $p=2^N$). Пусть, как и ранее, исходный набор данных распределен между процессорами блоками одинакового размера *n/p*; результирующее расположение блоков должно соответствовать нумерации процессоров гиперкуба. Возможный способ выполнения первой итерации параллельного метода при таких условиях может состоять в следующем:

- выбрать каким-либо образом ведущий элемент и разослать его по всем процессорам системы;
- разделить на каждом процессоре имеющийся блок данных на две части с использованием полученного ведущего элемента;
- образовать пары процессоров, для которых битовое представление номеров отличается только в позиции *N*, и осуществить взаимообмен данными между этими процессорами; в результате таких пересылок данных на процессорах, для которых в битовом представлении номера бит позиции *N* равен 0, должны оказаться части блоков со значениями, меньшими ведущего элемента; процессоры с номерами, в которых бит *N* равен 1, должны собрать, соответственно, все значения данных, превышающие значение ведущего элемента.

В результате выполнения такой итерации сортировки исходный набор оказывается разделенным на две части, одна из которых (со значениями меньшими, чем значение ведущего элемента) располагается на процессорах, в битовом представлении номеров которых бит *N* равен 0. Таких процессоров всего *p/2* и, таким образом, исходный *N*-мерный гиперкуб оказывается разделенным на два

гиперкуба размерности *(N-1)*. К этим подкубам, в свою очередь, может быть параллельно применена описанная выше процедура. После *N*-кратного повторения подобных итераций для завершения сортировки достаточно упорядочить блоки данных, получившиеся на каждом отдельном процессоре вычислительной системы. Эффективность параллельного метода быстрой сортировки, как и в последовательном варианте, во многом зависит от успешности выбора значений ведущих элементов. Определение общего правила для выбора этих значений является достаточно трудной задачей; сложность такого выбора может быть снижена, если выполнить упорядочение локальных блоков процессоров перед началом сортировки и обеспечить однородное распределение сортируемых данных между процессорами вычислительной системы.

Вычислительная сложность параллельного алгоритма быстрой сортировки определяется выражением:

$$T_p = N \cdot \left(\frac{2n}{p}\right) + \frac{n}{p}\log\left(\frac{n}{p}\right),$$

где первая часть определяет общую трудоемкость всех операций выбора ведущего элемента и разделения блоков согласно этому элементу, а вторая – трудоемкость локальной сортировки блоков на последней итерации алгоритма.

Коммуникационная трудоемкость, как и в предыдущих случаях, зависит от выбранного метода передачи информации:

- Метод передачи сообщений:
  - время передачи данных

$$T_{пд} = N \cdot \left(2t_н + \frac{(n/2p)+1}{R}\right),$$

  - общее время выполнения алгоритма

$$T = \frac{1}{F}\left(N \cdot (2n/p) + (n/p)\log(n/p)\right) + N \cdot \left(2t_н + \frac{(n/2p)+1}{R}\right)$$

;

- Метод передачи пакетов:

  - время передачи данных

$$T_{пд} = N \cdot \left(2t_н + \frac{V}{R}\left(1 + \left\lceil \frac{n/2p}{V - V_0} \right\rceil\right)\right),$$

  - общее время выполнения алгоритма

$$T = \frac{1}{F}\left(N \cdot (2n/p) + (n/p)\log(n/p)\right) + N \cdot \left(2t_н + \frac{V}{R}\left(1 + \left\lceil \frac{n/2p}{V - V_0} \right\rceil\right)\right)$$

.

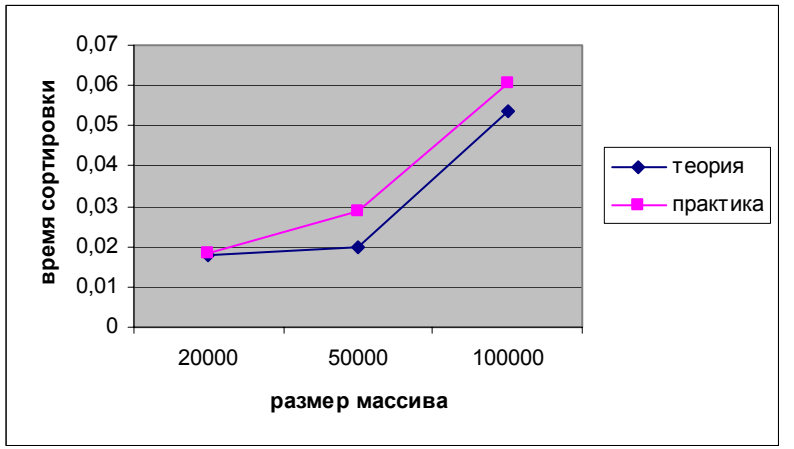


Рис. 27. Зависимость времени выполнения параллельного алгоритма быстрой сортировки от размера массива

Проведенные на реальной многопроцессорной вычислительной системе эксперименты доказывают справедливость теоретических временных оценок, приведенных выше. На графике представлены теоретическая и практическая зависимости времени сортировки массива от объема исходных данных при выполнении параллельного алгоритма быстрой сортировки на четырех процессорах.

Задания и упражнения

Проведите вычислительные эксперименты с алгоритмом быстрой сортировки. Изучите зависимость времени и ускорения от объема исходных данных и от количества процессоров. Пронаблюдайте за вычислениями отдельно взятого процессора. Сопоставьте временные оценки алгоритма быстрой сортировки с оценками алгоритмов сортировки пузырьком и сортировки Шелла.

8.2. Матричное умножение

Как уже отмечалось ранее в разделе 3 пособия, возможность организации параллельных вычислений для матричного умножения обеспечивается независимостью расчетов для получения элементов результирующей матрицы. В предельном случае (при наличии n^2 процессоров) все элементы матрицы C могут быть вычислены параллельно, при меньшем количестве процессоров объем обрабатываемых данных на каждом из процессоров увеличивается и применяемые при этом способы разделения данных, в большинстве случаев, сводятся либо к *ленточной,* либо к *блочной* схемам представления матриц. В системе ПараЛаб представлены оба способа разделения данных и обеспечивается возможность проведения вычислительных экспериментов с параллельным алгоритмом умножения матриц при ленточной схеме разделения данных и двумя методами (алгоритмы Фокса и Кэннона) для блочно-представленных матриц.

8.2.1. Ленточный алгоритм умножения матриц

При ленточной схеме разделения данных исходные матрицы разбиваются на горизонтальные (для матрицы *A*) и вертикальные (для матрицы *B*) полосы (см. рис. 27). 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.

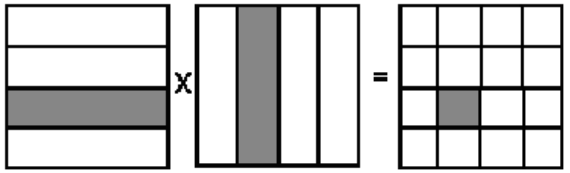


Рис. 28. Разбиение данных при выполнении ленточного алгоритма

Временные характеристики этого алгоритма:

- Метод передачи сообщений:
  - время передачи данных

$$T_{пд} = (p-1)\cdot\left(t_н + \frac{32\cdot n\cdot p}{R}\right),$$

  - общее время выполнения алгоритма

$$T = p\cdot\frac{n^3/p^2}{F} + (p-1)\cdot\left(t_н + \frac{32\cdot n\cdot p}{R}\right);$$

- Метод передачи пакетов:
  - время передачи данных

$$T_{пд} = (p-1)\cdot\left(t_н + \frac{V}{R}\left\lceil\frac{n\cdot p}{V-V_0}\right\rceil\right),$$

  - общее время выполнения алгоритма

$$T = p\cdot\frac{(n^3/p^2)}{F} + (p-1)\cdot\left(t_н + \frac{V}{R}\left\lceil\frac{n\cdot p}{V-V_0}\right\rceil\right).$$

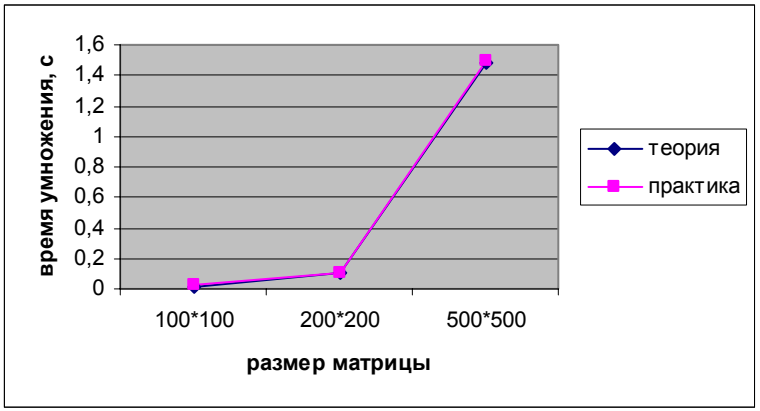


Рис. 29. Зависимости времени выполнения ленточного алгоритма умножения матриц от их размерности

Проведенные на вычислительном кластере эксперименты доказывают справедливость временных оценок, полученных теоретически. На графике представлены теоретическая и практическая зависимости времени умножения матриц от объема их размерности при выполнении ленточного алгоритма умножения на четырех процессорах.

Задания и упражнения

Проведите вычислительные эксперименты с ленточным алгоритмом умножения матриц. Наблюдайте за последовательностью обмена данными между процессорами. Определите, на каких топологиях вычислительной системы, кроме кольца, возможна реализация ленточного алгоритма матричного умножения. Исследуйте зависимость времени выполнения алгоритма от количества процессоров и от объема исходных данных. Оцените, какое максимальное ускорение можно получить при использовании этого алгоритма. Определите, при каких параметрах вычислительной системы оно достигается.

8.2.2. Алгоритм Фокса

Для организации параллельных вычислений при блочном представлении матриц предположим, что процессоры образуют логическую прямоугольную решетку размером $k\times k$ (обозначим через p_{ij} процессор, располагаемый на пересечении i строки и j столбца решетки). Основные положения параллельного метода, известного как алгоритм Фокса (*Fox*) [23], состоят в следующем:

- каждый из процессоров решетки отвечает за вычисление одного блока матрицы C;
- в ходе вычислений на каждом из процессоров p_{ij} располагается четыре матричных блока:
  - блок C_{ij} матрицы C, вычисляемый процессором;
  - блок A_{ij} матрицы A, размещенный в процессоре перед началом вычислений;
  - блоки $A_{ij}^{'}$, $B_{ij}^{'}$ матриц A и B, получаемые процессором в ходе выполнения вычислений.

Выполнение параллельного метода включает:

- этап инициализации, на котором на каждый процессор p_{ij} передаются блоки A_{ij}, B_{ij} и обнуляются блоки C_{ij} на всех процессорах;

- этап вычислений, на каждой итерации l, $1 \le l \le k$, которого выполняется:

– для каждой строки i, $1 \le i \le k$, процессорной решетки блок A_{ij} процессора p_{ij} пересылается на все процессоры той же строки i; индекс j, определяющий положение процессора p_{ij} в строке, вычисляется по соотношению

$$j = (i + l - 1) \bmod k + 1,$$

(mod есть операция получения остатка от целого деления);

– полученные в результате пересылок блоки A'_{ij}, B'_{ij} каждого процессора p_{ij} перемножаются и прибавляются к блоку C_{ij}:

Рис. 30. Состояние блоков на каждом процессоре в ходе выполнения итераций этапа вычислений

$$C_{ij} = C_{ij} + A'_{ij} \times B'_{ij};$$

– блоки B'_{ij} каждого процессора p_{ij} пересылаются процессорам p_{ij}, являющимися соседями сверху в столбцах процессорной решетки (блоки процессоров из первой строки решетки пересылаются процессорам последней строки решетки).

Для пояснения приведенных правил параллельного метода на рис. 30 приведено состояние блоков на каждом процессоре в ходе выполнения итераций этапа вычислений (для процессорной решетки 2×2).

Время выполнения этого алгоритма:

- Метод передачи сообщений:

$$T = k \cdot \frac{(n/k)^3}{F} + 2 \cdot \left(t_{и} + \frac{8 \cdot (k-1) \cdot (n/k)^2}{R} \right),$$

- Метод передачи пакетов:

1 итерация

1)

		1	2
1	A'_{ij}	A_{11}	A_{11}
	B'_{ij}	B_{11}	B_{12}
	C_{ij}	0	0
2	A'_{ij}	A_{22}	A_{22}
	B'_{ij}	B_{21}	B_{22}
	C_{ij}	0	0

2)

		1	2
1	A'_{ij}	A_{11}	A_{11}
	B'_{ij}	B_{11}	B_{12}
	C_{ij}	$A_{11}B_{11}$	$A_{11}B_{12}$
2	A'_{ij}	A_{22}	A_{22}
	B'_{ij}	B_{21}	B_{22}
	C_{ij}	$A_{22}B_{21}$	$A_{22}B_{22}$

3)

		1	2
1	A'_{ij}	A_{11}	A_{11}
	B'_{ij}	B_{21}	B_{22}
	C_{ij}	$A_{11}B_{11}$	$A_{11}B_{12}$
2	A'_{ij}	A_{22}	A_{22}
	B'_{ij}	B_{11}	B_{12}
	C_{ij}	$A_{22}B_{21}$	$A_{22}B_{22}$

2 итерация

1)

		1	2
1	A'_{ij}	A_{12}	A_{12}
	B'_{ij}	B_{21}	B_{22}
	C_{ij}	$A_{11}B_{11}$	$A_{11}B_{12}$
2	A'_{ij}	A_{21}	A_{21}
	B'_{ij}	B_{11}	B_{12}
	C_{ij}	$A_{22}B_{21}$	$A_{22}B_{22}$

2)

		1	2
1	A'_{ij}	A_{12}	A_{12}
	B'_{ij}	B_{21}	B_{22}
	C_{ij}	$A_{11}B_{11} + A_{12}B_{21}$	$A_{11}B_{12} + A_{12}B_{22}$
2	A'_{ij}	A_{21}	A_{21}
	B'_{ij}	B_{11}	B_{12}
	C_{ij}	$A_{22}B_{21} + A_{21}B_{11}$	$A_{22}B_{22} + A_{21}B_{12}$

3)

		1	2
1	A'_{ij}	A_{12}	A_{12}
	B'_{ij}	B_{21}	B_{22}
	C_{ij}	$A_{11}B_{11} + A_{12}B_{21}$	$A_{11}B_{12} + A_{12}B_{22}$
2	A'_{ij}	A_{21}	A_{21}
	B'_{ij}	B_{11}	B_{12}
	C_{ij}	$A_{22}B_{21} + A_{21}B_{11}$	$A_{22}B_{22} + A_{21}B_{12}$

$$T = k \cdot \frac{(n/k)^3}{F} + 2 \cdot \left(\frac{V}{R} \left((k-1) + \left\lceil \frac{(n/k)^2}{V - V_0} \right\rceil \right) \right).$$

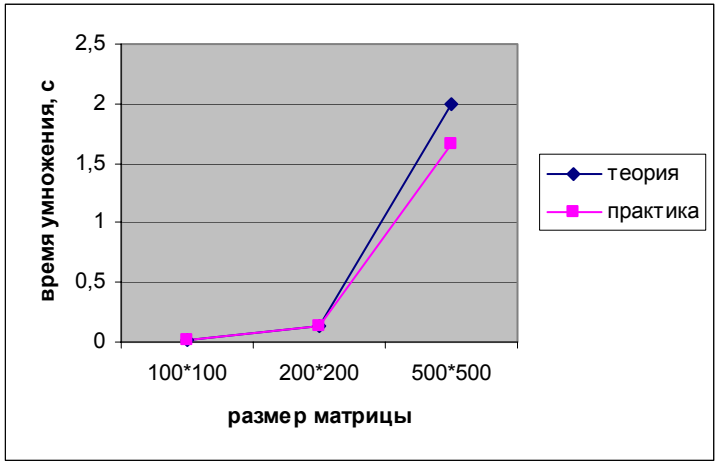


Рис. 31. Зависимость времени перемножения квадратных матриц от их размерности при выполнении алгоритма Фокса

Эксперименты, проведенные на реальной многопроцессорной вычислительной системе, доказывают справедливость приведенных выше теоретических временных оценок. На графике (рис. 31) представлены теоретическая и практическая зависимости времени перемножения квадратных матриц $n \times n$ от размерности этих матриц n при выполнении Алгоритма Фокса на четырех процессорах.

Задания и упражнения

Проведите несколько вычислительных экспериментов с использованием алгоритма Фокса умножения матриц. Убедитесь в том, что полученные на практике зависимости времени и ускорения от параметров вычислительной системы и задачи удовлетворяют теоретическим зависимостям, приведенным выше.

8.2.3. Алгоритм Кэннона

Отличие алгоритма Кэннона от представленного в предыдущем разделе метода Фокса состоит в изменении схемы начального распределения блоков перемножаемых матриц между процессорами

вычислительной системы. Начальное расположение блоков в алгоритме Кэннона подбирается таким образом, чтобы располагаемые блоки на процессорах могли бы быть перемножены без каких-либо дополнительных передач данных между процессорами. При этом подобное распределение блоков может быть организовано таким образом, что перемещение блоков между процессорами в ходе вычислений может осуществляться с использованием более простых коммуникационных операций.

С учетом высказанных замечаний этап инициализации алгоритма Кэннона включает выполнение следующих операций передач данных:

- на каждый процессор p_{ij} передаются блоки A_{ij}, B_{ij};
- для каждой строки i процессорной решетки блоки матрицы A сдвигаются на $(i-1)$ позиций влево;
- для каждого столбца j процессорной решетки блоки матрицы B сдвигаются на $(j-1)$ позиций вверх.

Для демонстрации получаемого распределения данных на рис. 32 показан пример расположения блоков для процессорной решетки размера 4x4.

A_{11}, B_{11}	A_{12}, B_{22}	A_{13}, B_{33}	A_{14}, B_{44}
A_{22}, B_{21}	A_{23}, B_{32}	A_{24}, B_{43}	A_{21}, B_{14}
A_{33}, B_{31}	A_{34}, B_{42}	A_{31}, B_{13}	A_{32}, B_{24}
A_{44}, B_{41}	A_{41}, B_{12}	A_{42}, B_{23}	A_{43}, B_{34}

Рис. 32. Начальное распределение блоков в алгоритме Кэннона для процессорной решетки 4x4

В ходе вычислений на каждой итерации алгоритма Кэннона каждый блок матрицы A сдвигается на один процессор влево по решетке, а каждый блок матрицы B - на один процессор вверх.

Временные характеристики алгоритма:

- Метод передачи сообщений:

$$T = \sqrt{p}\,\frac{\left(n/\sqrt{p}\right)^3}{F} + 2\cdot\left(t_н + \left(\sqrt{p}-1\right)\frac{n^2}{p\cdot R}\right) + \left(t_н + \frac{n^2}{p\cdot R}\right)\left(2\sqrt{p}-1\right),$$

- Метод передачи пакетов:

$$T = \sqrt{p}\,\frac{\left(n/\sqrt{p}\right)^3}{F} + \left(2\sqrt{p}+1\right)\cdot\left(t_н + \frac{V}{R}\left(\sqrt{p}-1+\left\lceil\frac{\left(n/\sqrt{p}\right)^2}{V-V_0}\right\rceil\right)\right).$$

Задания и упражнения

Проведите несколько вычислительных экспериментов с использованием алгоритма Кэннона для умножения матриц. Сравните порядок обмена данными между процессорами при выполнении алгоритма Фокса и алгоритма Кэннона. Исследуйте, какой из алгоритмов матричного умножения обладает наилучшими временными характеристиками.

8.3. Обработка графов

Для описания графов известны различные способы задания. Пусть G есть граф

$$G = (V, R),$$

для которого набор вершин v_i, $1 \le i \le n$, задается множеством V, а список дуг графа

$$r_j = (v_{s_j}, v_{t_j}),\ 1 \le j \le k,$$

определяется множеством R. В общем случае дугам графа могут приписываться некоторые числовые характеристики w_j, $1 \le j \le k$ (взвешенный граф).

При малом количестве дуг в графе (т.е. $k<<n^2$) целесообразно использовать для определения графов списки, перечисляющие имеющиеся в графах дуги. Представление достаточно плотных графов, для которых почти все вершины соединены между собой дугами (т.е. $k \sim n^2$), может быть эффективно обеспечено при помощи матрицы

инцидентности $A=(a_{ij})$, $1 \le i, j \le n$, ненулевые значения элементов которой соответствуют дугам графа

$$a_{ij} = \begin{cases} w(v_i, v_j), & \text{если } (v_i, v_j) \in R, \\ 0, & \text{если } i = j, \\ \infty, & \text{иначе} \end{cases}$$

Использование матрицы инцидентности позволяет применять также при реализации вычислительных процедур для графов матричные алгоритмы обработки данных. Более широко способы представления графов рассмотрены, например, в [8].

В системе ПараЛаб в качестве примеров задач обработки графов рассматриваются проблемы нахождения минимального охватывающего дерева и поиска минимальных путей, решаемые при помощи алгоритмов Прима и Дейкстры.

8.3.1. Алгоритм Прима нахождения 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.

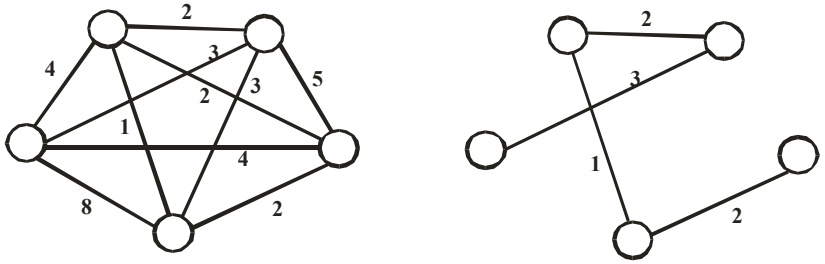


Рис. 33. Ненаправленный взвешенный граф и его минимальное охватывающее дерево

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. Пусть V_T есть множество вершин, уже включенных алгоритмом в МОД, а величины d_i, $1 \le i \le n$, характеризуют дуги минимальной длины от вершин, еще не включенных в дерево, до множества V_T, т.е.

$$\forall i \notin V_T \Rightarrow d_i = \min\{w(i,u) : u \in V_T, (i,u) \in R\}$$

(если для какой либо вершины $i \notin V_T$ не существует ни одной дуги в V_T, значение d_i устанавливается в ∞). При начале работы алгоритма выбирается корневая вершина МОД s и полагается

$$V_T = \{s\},\ d_s = 0.$$

Действия, выполняемые на каждой итерации алгоритма Прима, состоят в следующем:

- определяются значения величин d_i для всех вершин, еще не включенные в состав МОД;
- выбирается вершина t графа G, имеющая дугу минимального веса до множества V_T

$$t : d_t = \min d_i, i \notin V_T;$$

- включение выбранной вершины t в V_T.

После выполнения $n-1$ итераций метода МОД будет сформировано; вес этого дерева может быть получен при помощи выражения

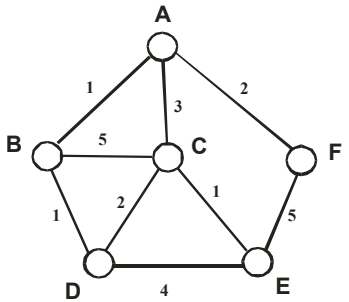
$$W_T = \sum_{i=1}^{n} d_i .$$

Трудоемкость нахождения МОД характеризуется квадратичной зависимостью от числа вершин графа G

$$T_1 \sim n^2 .$$

На рис. 34 представлен процесс нахождения МОД при помощи алгоритма Прима. В левом столбце изображается текущее состояние графа; жирными линиями выделены те ребра графа, которые уже включены в состав МОД. В правом столбце отображается массив величин *d[i]* (эти величины вычисляются заново на каждой итерации) и матрица инцидентности графа *G* (не изменяется в ходе выполнения вычислений).

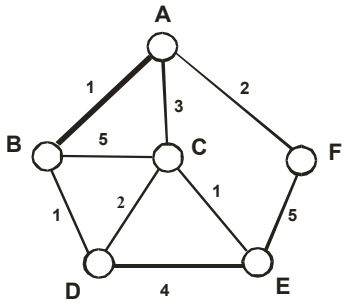
Исходный граф. В качестве корня МОД выбираем вершину B



d[]	a	b	c	d	e	f
	1	0	5	1	∞	∞

a	0	1	3	∞	∞	2
b	1	0	5	1	∞	∞
c	3	5	0	2	1	∞
d	∞	1	2	0	4	∞
e	∞	∞	1	4	0	5
f	2	∞	∞	∞	5	0

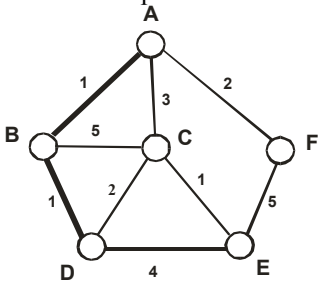
Значение переменных после того, как выбрано первое ребро



d[]	a	b	c	d	e	f
	1	0	2	1	4	∞

a	0	1	3	∞	∞	2
b	1	0	5	1	∞	∞
c	3	5	0	2	1	∞
d	∞	1	2	0	4	∞
e	∞	∞	1	4	0	5
f	2	∞	∞	∞	5	0

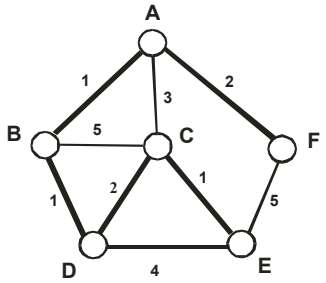
Значение переменных после того, как выбрано второе ребро



d[]	a	b	c	d	e	f
	1	0	2	1	4	2

a	0	1	3	∞	∞	2
b	1	0	5	1	∞	∞
c	3	5	0	2	1	∞
d	∞	1	2	0	4	∞
e	∞	∞	1	4	0	5
f	2	∞	∞	∞	5	0

Окончательное минимальное охватывающее дерево



d[]	a	b	c	d	e	f
	1	0	2	1	1	2

a	0	1	3	∞	∞	2
b	1	0	5	1	∞	∞
c	3	5	0	2	1	∞
d	∞	1	2	0	4	∞
e	∞	∞	1	4	0	5
f	2	∞	∞	∞	5	0

Рис. 34. Алгоритм Прима поиска минимального охватывающего дерева. Минимальное охватывающее дерево имеет корень в вершине B.

Оценим возможности для параллельного выполнения рассмотренного алгоритма нахождения минимально охватывающего дерева. Итерации метода должны выполняться последовательно и, тем самым, не могут быть распараллелены. С другой стороны, выполняемые на каждой итерации алгоритма действия являются независимыми и могут реализовываться одновременно. Так, например, определение величин d_i может осуществляться для каждой вершины графа в отдельности, нахождение дуги минимального веса может быть реализовано по каскадной схеме и т.д.

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. Соблюдение данного принципа приводит к тому, что при равномерной загрузке каждый процессор P_j, $1 \le j \le p$, будет содержать набор вершин

$$V_j = \{v_{i_j+1}, v_{i_j+2}, ..., v_{i_j+k}\},\ i_j = k(j-1),\ k = n/p,$$

соответствующий этому набору блок из k величин d_i, $1 \le i \le n$, и

вертикальную полосу матрицы инцидентности графа G из k соседних столбцов, а также общую часть набора V_j и формируемого в процессе вычислений множества вершин V_T.

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

- определяются значения величин d_i для всех вершин, еще не включенных в состав МОД; данные вычисления выполняются независимо на каждом процессоре в отдельности; трудоемкость такой операции ограничивается сверху величиной
  - $T = \frac{1}{F} \cdot \frac{n}{p}$;

(на первой итерации алгоритма необходим перебор всех вершин, что требует вычислений порядка $T = \frac{1}{F} \cdot \frac{n^2}{p}$);

- выбирается вершина t графа G, имеющая дугу минимального веса до множества V_T; для выбора такой вершины необходимо осуществить поиск минимума в наборах величин d_i, имеющихся на каждом из процессоров (количество параллельных операций n/p), и выполнить сборку полученных значений. Длительность такой операции передачи данных на полном графе равна:
  - при пересылке сообщений:

$$T = t_н + \frac{1}{R};$$

  - при пересылке пакетов:

$$T = t_н + \frac{V}{R};$$

- рассылка всем процессорам номера выбранной вершины для включения в охватывающее дерево.

Получение МОД обеспечивается при выполнении $(n-1)$ итерации алгоритма Прима; как результат, общая трудоемкость метода определяется соотношением:

- Метод передачи сообщений:

$$T = \frac{1}{F}\left(\frac{n^2}{p} + \frac{n(n-2)}{p}\right) + (n-1)\cdot\left(t_н + \frac{1}{R}\right),$$

- Метод передачи пакетов:

$$T = \frac{1}{F}\left(\frac{n^2}{p} + \frac{n(n-2)}{p}\right) + (n-1)\cdot\left(t_н + \frac{V}{R}\right).$$

Задания и упражнения

Проведите последовательность вычислительных экспериментов с алгоритмом Прима поиска минимального охватывающего дерева. При помощи встроенного редактора графов задавайте различные виды графов (дерево, цикл, граф, сформированный при помощи случайного механизма) и наблюдайте за процессом построения МОД. Изучите зависимости временных характеристик выполнения алгоритма от параметров задачи и вычислительной системы.

8.3.2. 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 вспомогательных величин d_i, $1 \le i \le n$. В алгоритме Дейкстры эти величины означают суммарный вес пути от начальной вершины до всех остальных вершин графа. Как результат, после выбора очередной вершины t графа для включения в множество выбранных вершин V_T, значения величин d_i, $1 \le i \le n$, пересчитываются в соответствии с новым правилом:

$$\forall i \notin V_T \Rightarrow d_i = \min\{d_i, d_t + w(t,i)\}.$$

С учетом измененного правила пересчета величин d_i, $1 \le i \le n$, схема распределения данных по процессорам при выполнении алгоритма Дейкстры может быть сформирована по аналогии с параллельным вариантом метода Прима.

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

- определяются значения величин d_i для всех вершин, еще не включенных в состав дерева кратчайших путей; данные вычисления выполняются независимо на каждом процессоре в отдельности; трудоемкость такой операции ограничивается сверху величиной

$$T = \frac{1}{F} \cdot \frac{n}{p};$$

(на первой итерации алгоритма необходим перебор всех вершин, что требует вычислений порядка

$$T = \frac{1}{F} \cdot \frac{n^2}{p};$$

- выбирается вершина t графа G, имеющая дугу минимального веса до множества V_T; для выбора такой вершины необходимо осуществить поиск минимума в наборах величин d_i, имеющихся на каждом из процессоров (количество параллельных операций n/p), и выполнить сборку полученных значений. Длительность такой операции передачи данных на полном графе равна:
  - при пересылке сообщений:

$$T = t_н + \frac{1}{R},$$

  - при пересылке пакетов:

$$T = t_н + \frac{V}{R};$$

- рассылка номера выбранной вершины для включения в дерево кратчайших путей всем процессорам.

Получение дерева кратчайших путей обеспечивается при выполнении $(n-1)$ итерации алгоритма Дейкстры; как результат, общая трудоемкость метода определяется соотношением:

- Метод передачи сообщений:

$$T = \frac{1}{F}\left(\frac{n^2}{p} + \frac{n(n-2)}{p}\right) + (n-1)\cdot\left(t_н + \frac{1}{R}\right),$$

- Метод передачи пакетов:

$$T = \frac{1}{F}\left(\frac{n^2}{p} + \frac{n(n-2)}{p}\right) + (n-1)\cdot\left(t_н + \frac{V}{R}\right).$$

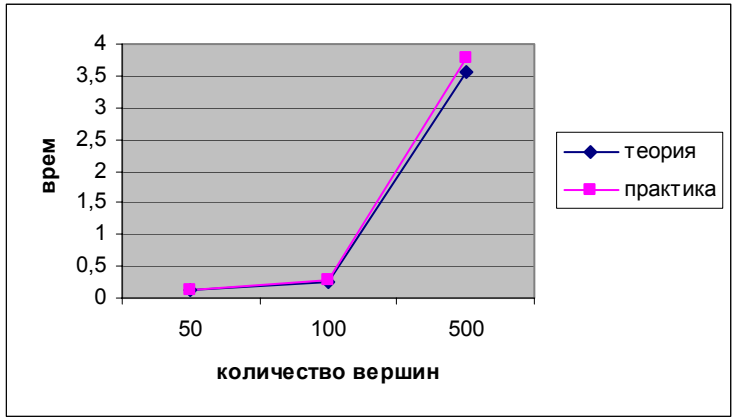


Рис. 35. Зависимости времени выполнения алгоритма Дейкстры поиска кратчайших путей от количества вершин в графе

Эксперименты, проведенные на реальной многопроцессорной вычислительной системе, доказывают справедливость приведенных выше теоретических временных оценок. На графике представлены теоретическая и практическая зависимости времени выполнения алгоритма Дейкстры поиска кратчайших путей на четырех процессорах от количества вершин в графе.

Задания и упражнения

Проведите несколько вычислительных экспериментов с использованием алгоритма Дейкстры поиска кратчайших путей. Пронаблюдайте за процессом построения дерева кратчайших путей в случае, когда исходный граф имеет специальный вид (дерево, полный граф, двудольный граф). Изучите зависимость времени построения дерева кратчайших путей от параметров задачи и вычислительной системы. Исследуйте, при каких значениях параметров вычислительной системы достигается максимальное ускорение параллельных вычислений.

9. Использование системы ПараЛаб в учебном процессе

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 ПараЛаб может использоваться также и при проведении научных исследований для оценки эффективности параллельных вычислений.

Проведение лабораторных занятий с использованием комплекса ПараЛаб может быть организовано как выполнение следующей последовательности работ:

- Моделирование многопроцессорных вычислительных систем (выбор топологии, задание количества и производительности процессоров, выбор метода передачи данных и задание коммуникационных характеристик сети).
- Определение класса решаемых задач и задание параметров задачи.
- Выбор параллельного метода решения задачи и настройка значений его параметров.
- Установка графических индикаторов для наблюдения за процессом параллельных вычислений (состояние данных на процессорах системы, передача информации по сети, текущая оценка решения исходной вычислительной задачи).
- Проведение экспериментов в режиме имитации вычислений; пошаговый, последовательный (непрерывный) и циклический (серийный) способы проведения экспериментов; одновременное выполнение нескольких экспериментов в режиме разделения времени для разных вариантов топологии вычислительной системы, параметров задачи, количества процессоров и т.п.
- Анализ результатов с использованием сведений из журнала экспериментов; оценка времени решения задач в зависимости от размерности задачи и количества процессоров; построение зависимостей ускорения и эффективности параллельных вычислений.

- Проведение экспериментов в режиме реальных параллельных вычислений; выполнение параллельных программ в виде множества независимых процессов на одном процессоре; удаленный доступ к многопроцессорной вычислительной системе (кластеру); сравнение теоретических оценок и результатов реальных вычислительных экспериментов.

Наряду с теоретическими и практическими аспектами изучения параллельных методов решения сложных вычислительных задач система ПараЛаб обеспечивает:

- овладение технологией и методикой построения параллельных алгоритмов;
- навыки работы в среде интеллектуальной программной системы, снабженной дружественным интерфейсом;
- наглядную демонстрацию средствами цветной дисплейной графики сложных понятий и процессов;
- обучение выполнению операций (выбор, создание, редактирование) со сложными информационными структурами (аналитическими, графическими и др.).

Лабораторные занятия, проводимые при использовании комплекса ПараЛаб, могут проводиться, например, по следующему плану:

- обучаемый решает определенную сложную вычислительную задачу при помощи нескольких параллельных методов на заданной им вычислительной системе, сопоставляет результаты и дает их интерпретацию в рамках теории построения параллельных алгоритмов;
- обучаемый самостоятельно конструирует средствами комплекса несколько вычислительных систем таким образом, чтобы при решении задачи продемонстрировать основные теоретические понятия;
- обучаемый самостоятельно формирует одну или несколько вычислительных систем и решает задачи при различных значениях параметров вычислительной системы, изучая тем самым влияние параметров на временные характеристики алгоритма;
- обучаемый проводит реальные вычислительные эксперименты в режиме удаленного доступа к кластеру и сравнивает результаты реальных и имитационных экспериментов.

Более подробно рекомендуемая тематика занятий описана в следующем разделе.

10. Тематика предлагаемых учебных занятий

При практическом использовании комплекса ПараЛаб в обучении можно рекомендовать следующую схему проведения лабораторных занятий.

10.1 Общее знакомство с возможностями системы

Задание 1. *Общие принципы использования системы ПараЛаб.*

Основные правила взаимодействия с системой. Организация управления при помощи манипулятора мышь. Способы получения справочной информации.

Задание 2. *Освоение способов формирования вычислительной системы*.

Выбор топологии вычислительной системы. Задание количества процессоров и их производительности. Задание характеристик коммуникационной среды: латентности и пропускной способности. Выбор метода передачи данных. Просмотр справочной информации по моделированию многопроцессорной вычислительной системы.

10.2. Изучение параллельных методов решения сложных вычислительных задач

Задание 3. *Изучение параллельных методов решения задачи сортировки.*

Проведение вычислительных экспериментов с методом пузырьковой сортировки на разных топологиях вычислительной системы. Изучение временных характеристик алгоритма. Сравнение алгоритма пузырьковой сортировки с алгоритмом сортировки Шелла. Изучение алгоритма быстрой сортировки. Сравнение временных характеристик и количества итераций с пузырьковым методом и методом Шелла.

Изучение графических форм комплекса ПараЛаб для наблюдения за процессом выполнения эксперимента:

- изменение темпа демонстрации;
- изменение способа отображения обмена данными;
- пошаговый режим исполнения итераций алгоритма;
- наблюдение за вычислениями одного из процессоров в отдельном окне.

Задание 4. *Изучение параллельных методов решения задачи матричного умножения.*

Проведение вычислительных экспериментов с ленточным методом умножения матриц, с методами Фокса и Кэннона. Знакомство с понятием блочных матричных операций. Сравнение результатов экспериментов при помощи таблицы итогов и журнала экспериментов.

Задание 5. *Изучение параллельных методов решения задач обработки графов.*

Знакомство со способами формирования графов в системе ПараЛаб. Ручное и случайное формирование графа. Сохранение графа в архиве системы и загрузка графа. Проведение вычислительных экспериментов с алгоритмом Прима поиска минимального охватывающего дерева и алгоритмом Дейкстры поиска кратчайших путей.

10.3. Сравнение результатов имитационных экспериментов с результатами реальных экспериментов

Задание 6. *Выполнение реальных экспериментов.*

Выполнение нескольких экспериментов с одной из задач в режиме удаленного доступа к вычислительному кластеру. Построение зависимостей временных характеристик выполнения эксперимента от параметров задачи (объем исходных данных) и вычислительной системы (количество процессоров) при помощи таблицы итогов.

Задание 7. *Сравнение результатов реальных и имитационных экспериментов.*

Проведение вычислительного эксперимента в режиме имитации и в режиме реальных вычислений. Сравнение результатов этих экспериментов. Нахождение для имитационного эксперимента таких параметров вычислительной системы, при которых результаты имитационного и реального эксперимента являются близкими.

Приложение. Описание команд системы ПараЛаб

В зависимости от действий, выполняемых пользователем, система может находиться в одном из следующих режимов функционирования:

- Начало работы,
- Основной режим работы,
- Режим выполнения эксперимента,
- Режим пошагового выполнения эксперимента,
- Режим выполнения серии экспериментов.

Схема переходов между указанными режимами представлена на рис. 36. Обозначенные стрелками переходы между режимами помечены наименованием команд, применение которых приводит к выполнению данных переходов. Завершение работы с системой происходит при выполнении команды **Завершить** начального и основного меню системы.

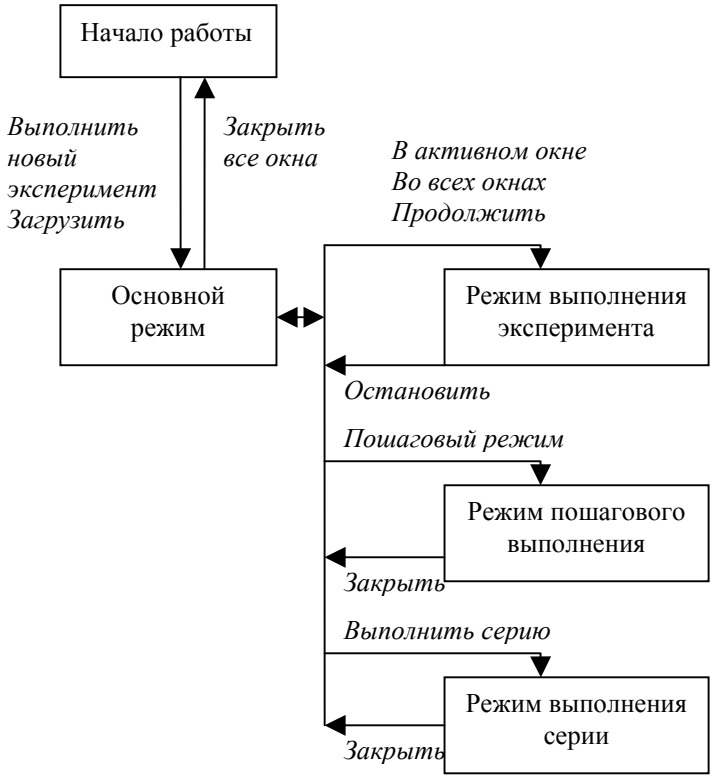


Рис. 36. Схема переходов между режимами работы системы ПараЛаб

Ниже следует описание наборов (меню) команд для каждого режима работы системы.

1. Начальное меню системы

Начало

- *Выполнить новый эксперимент*
  Создание окна эксперимента и переход к основному меню. При выполнении команды запрашивается название создаваемого окна.

- *Загрузить*
  Чтение из архива системы запомненного ранее пакета заданий по решению задач при помощи параллельных алгоритмов. Переход к основному меню.
- *Завершить*
  Завершение работы системы.

Справка

- *Содержание*
  Получение списка разделов справочной информации, имеющейся в системе.
- *О программе*
  Получение общей информации о системе.

2. Основное меню

Архив

Пункт меню содержит команды работы с архивом, печати и завершения.

- *Создать новый*
  Создание нового окна эксперимента.
- *Загрузить*
  Команда чтения данных из архива системы.
- *Сохранить*
  Команды записи данных в архив системы.
- *Печать*
  Пункт меню содержит команды печати графического представления окон с результатами выполненных экспериментов.
- *Настройка принтера*
  Вызывает диалоговое окно установки принтера. Диалог выполняет операции по установке принтера, на котором будет производиться печать, и задание его свойств.
- *Копировать в Буфер Обмена*
  Копирование графического изображения активного окна в буфер обмена системы Windows.
- *Завершить*
  Завершение работы системы.

Система

Пункт меню содержит команды для выбора параметров вычислительной системы, на которой будет проведен эксперимент.

- *Эмуляция*

  Команда, которая указывает, что эксперимент будет проводиться в режиме эмуляции.
- *Кластер*

  Команда, которая указывает, что эксперимент будет проводиться на реальной многопроцессорной вычислительной системе.

Эти два пункта меню сгруппированы, т.е. в любой момент времени эксперимент может быть выполнен только в одном из режимов. Активный режим помечен черной точкой слева.

- *Топология*

  Выбор топологии вычислительной системы из числа реализованных.
- *Количество процессоров*

  Выбор количества процессоров в рамках выбранной топологии.
- *Производительность процессоров*

  Определение производительности процессоров, составляющих параллельный вычислительный комплекс (предполагается, что все процессоры, составляющие многопроцессорную вычислительную систему, обладают одинаковым быстродействием).
- *Характеристики сети*

  Определение характеристик коммуникационной среды, связывающей процессоры, задание латентности и пропускной способности сети.
- *Метод передачи данных*

  Выбор метода передачи данных и задание его параметров.

Задача

Пункт меню содержит команды для постановки задачи.

- *Сортировка*

  По умолчанию этот пункт меню активен (помечен черной точкой слева) и указывает на то, что задачей является упорядочивание линейного массива данных.
- *Матричное умножение*

  Если этот пункт меню активен, текущей задачей в активном окне является задача матричного умножения.
- *Обработка графов*

  Если этот пункт меню активен, текущей задачей в активном окне является задача обработки графов.

Эти три пункта меню сгруппированы. В каждый момент времени активным может быть только один. Текущая задача помечена черной точкой слева.

- *Параметры задачи*

  Выбор параметров задачи (для сортировки – выбор объема исходных данных).
- *Формирование графа*

  Вызывает окно редактора графов. Если в активном окне граф не был сформирован, пользователю предоставляется возможность создать или загрузить граф. Если же в активном окне уже содержится непустой граф, то выполнение этой команды даст возможность его отредактировать.
- *Метод*

  Выбор метода решения задачи из числа реализованных в системе.

Графика

Пункт меню содержит команды для задания графических форм наблюдения за процессом выполнения эксперимента-эмуляции.

- *Настройка цвета*

  Выбор цветов для отображения данных и цвета выделения процессора.

- *Темп демонстрации*

  Регулирование темпа демонстрации результатов выполняемых итераций параллельного алгоритма.
- *Шаг визуализации*

  При выполнении алгоритмов обработки графов все итерации однотипны, и число их велико. Если отображать каждую итерацию, для выполнения алгоритма потребуется достаточно много времени. Выполнение команды позволяет отображать лишь некоторые итерации.
- *Пересылка данных*

  Выбор одного из двух режимов отображения коммуникации процессоров: движение пакетов или выделение каналов.
- *Наблюдение вычислений*

  Выбор процессора для более детального отображения его действий в отдельном окне.

Выполнение

Пункт меню содержит команды для выполнения экспериментов по исследованию принципов работы и эффективности параллельных алгоритмов.

- *В активном окне*

  Выполнение поставленной задачи в активном окне. Если активен эксперимент-эмуляция, происходит переход к режиму выполнения эксперимента.
- *Во всех окнах*

  Одновременное решение задач во всех имеющихся окнах. Если среди них есть хотя бы одно окно с экспериментом-эмуляцией, происходит переход к режиму выполнения эксперимента.
- *Выполнить серию*

  Переход к режиму выполнения серии.
- *Пошаговый режим*

  Переход к режиму пошагового выполнения

эксперимента.

- *Продолжить*

  Продолжение ранее приостановленного выполнения эксперимента-эмуляции. Продолжение вычислений возможно, только если после остановки не изменялись постановка задачи и вычислительная система.
- *Остановить*

  Остановка выполнения реального эксперимента. Пункт меню доступен, только если в активном окне выполняется реальный эксперимент.

Результаты

Пункт меню содержит команды для просмотра и анализа общих результатов выполненных экспериментов.

- *Итоги*

  Получение списка выполненных экспериментов с указанием их установок и возможностью графического построения зависимости временных характеристик от различных параметров задачи и системы.
- *Журнал экспериментов*

  Пункт меню содержит команды для записи, просмотра и анализа результатов в журнале экспериментов. В журнале запоминается тип выполняемого эксперимента, поставленная задача, параметры вычислительной системы и полученные временные характеристики.

  Записать

  Запись результатов последнего выполненного эксперимента в журнал экспериментов.

  Обнулить

  Удаление всех записанных в журнале экспериментов результатов.

  Автозапись

  Включение режима автоматической записи результатов выполняемых экспериментов в журнал. Отмена режима происходит при повторном

выполнении команды.

Показать

Получение для просмотра результатов, запомненных в журнале экспериментов. Содержимое журнала представляется в виде таблицы.

Эксперимент

Пункт меню содержит команды для создания и управления окнами в рабочей области системы.

- *Создать новый*

  Создание нового окна в рабочей области системы для проведения экспериментов по исследованию принципов функционирования и эффективности параллельных алгоритмов.
- *Расположить каскадом*

  Расположение окон в порядке, при котором они перекрываются (активное окно занимает большую часть рабочей области), но сохраняется видимость строк заголовков всех окон.
- *Показать все*

  Распределение рабочей области поровну между всеми имеющимися окнами.
- *Упорядочить пиктограммы*

  Упорядочение расположения пиктограмм всех минимизированных окон рабочей области системы.
- *Закрыть все*

  Закрытие всех окон рабочей области системы. Переход к начальному меню.
- *Запомнить как образец*

  Запоминание задачи или вычислительной системы (по выбору пользователя) активного окна как образца для копирования в другие окна.
- *Взять образец*

  Взятие ранее запомненного образца задачи или системы в качестве установок эксперимента в активном окне.
- *Копировать во все окна*

Копирование задачи или системы из активного окна во все имеющиеся окна экспериментов.

Справка

- *Содержание*

  Получение списка разделов справочной информации, имеющейся в системе.
- *О программе*

  Получение общей информации о системе.

3. Меню выполнения эксперимента

Остановить

Приостановка процесса выполнения эксперимента-эмуляции. Переход к основному режиму работы.

4. Меню пошагового выполнения эксперимента

Шаг

Выполнение очередной итерации параллельного алгоритма.

Без остановки

Выполнение эксперимента без остановки до завершения. Переход в режим выполнения эксперимента

Закрыть

Приостановка процесса выполнения эксперимента-эмуляции. Переход к основному режиму работы

Справка

Получение справочной информации о правилах пошагового выполнения экспериментов.

5. Меню выполнения серии экспериментов

Пуск

Переход к выполнению последовательности экспериментов для исследования зависимостей временных характеристик выполнения эксперимента от выбранного параметра.

Закрыть

Завершение выполнения серии и переход к основному режиму работы.

Справка

Получение справочной информации о правилах выполнения серии экспериментов.

ЛИТЕРАТУРА

1. Гергель В.П., Стронгин Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем. - Н.Новгород: ННГУ им. Н.И. Лобачевского, 2001.
2. Богачев К.Ю. Основы параллельного программирования. - М.: БИНОМ. Лаборатория знаний, 2003.
3. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. - СПб.: БХВ-Петербург, 2002.
4. Немнюгин С., Стесик О. Параллельное программирование для многопроцессорных вычислительных систем — СПб.: БХВ-Петербург, 2002.
5. Березин И.С., Жидков И.П. Методы вычислений. - М.: Наука, 1966.
6. Дейтел Г. Введение в операционные системы. Т.1.- М.: Мир, 1987.
7. Кнут Д. Искусство программирования для ЭВМ. Т. 3. Сортировка и поиск. - М.: Мир, 1981.
8. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. - М.: МЦНТО, 1999.
9. Корнеев В.В.. Параллельные вычислительные системы. - М.: Нолидж, 1999.
10. Корнеев В.В. Параллельное программирование в MPI. Москва-Ижевск: Институт компьютерных исследований, 2003.
11. П.Тихонов А.Н., Самарский А.А. Уравнения математической физики. -М.:Наука, 1977.
12. Хамахер К., Вранешич З., Заки С. Организация ЭВМ. - СПб: Питер, 2003.
13. Шоу А. Логическое проектирование операционных систем. - М.: Мир, 1981.
14. Andrews G.R. Foundations of Multithreading, Parallel and Distributed Programming. Addison-Wesley, 2000 (русский перевод Эндрюс Г.Р. Основы многопоточного, параллельного и распределенного программирования. - М.: Издательский дом "Вильяме", 2003)
15. Barker, M. (Ed.) (2000). Cluster Computing Whitepaper http://www.dcs.port.ac.uk/~mab/tfcc/WhitePaper/.
16. Braeunnl T. Parallel Programming. An Introduction.- Prentice Hall, 1996.

17. Chandra, R., Menon, R., Dagum, L., Kohr, D., Maydan, D., McDonald J. Parallel Programming in OpenMP. - Morgan Kaufinann Publishers, 2000
18. Dimitri P. Bertsekas, John N. Tsitsiklis. Parallel and Distributed Computation. Numerical Methods. - Prentice Hall, Englewood Cliffs, New Jersey, 1989.
19. Fox G.C. et al. Solving Problems on Concurrent Processors. - Prentice Hall, Englewood Cliffs, NJ, 1988.
20. Geist G.A., Beguelin A., Dongarra J., Jiang W., Manchek B., Sunderam V. PVM: Parallel Virtual Machine - A User's Guide and Tutorial for Network Parallel Computing. MIT Press, 1994.
21. Group W, Lusk E, Skjellum A. Using MPI. Portable Parallel Programming with the Message-Passing Interface. - MIT Press, 1994.(htp://www.mcs.anl.gov/mpi/index.html)
22. Hockney R. W., Jesshope C.R. Parallel Computers 2. Architecture, Programming and Algorithms. - Adam Hilger, Bristol and Philadelphia, 1988. (Русский перевод 1 издания: Р.Хокни, К.Джесссхоуп. Параллельные ЭВМ. Архитектура, программирование и алгоритмы. - М.: Радио и связь, 1986)
23. Kumar V., Grama A., Gupta A., Karypis G. Introduction to Parallel Computing. - The Benjamin/Cummings Publishing Company, Inc., 1994
24. Miller R., Boxer L. A Unified Approach to Sequential and Parallel Algorithms. Prentice Hall, Upper Saddle River, NJ. 2000.
25. Pacheco, S. P. Parallel programming with MPI. Morgan Kaufmann Publishers, San Francisco. 1997.
26. Parallel and Distributed Computing Handbook. / Ed. A.Y. Zomaya. - McGraw-Hill, 1996.
27. Pfister, G. P. In Search of Clusters. Prentice Hall PTR, Upper Saddle River, NJ 1995. (2nd edn., 1998).
28. Quinn M. J. Designing Efficient Algorithms for Parallel Computers. - McGraw-Hill, 1987. 29.Rajkumar Buyya. High Performance Cluster Computing. Volume l: Architectures and Systems. Volume 2: Programming and Applications. Prentice Hall PTR, Prentice-Hall Inc., 1999. 30.Roosta, S.H. Parallel Processing and Parallel Algorithms: Theory and Computation. Springer-Verlag, NY. 2000.
31. Xu, Z., Hwang, K. Scalable Parallel Computing Technology, Architecture, Programming. McGraw-Hill, Boston. 1998.
32. Wilkinson B., Allen M. Parallel programming. - Prentice Hall, 1999.

Информационные ресурсы сети Интернет

33. Информационно-аналитические материалы по параллельным вычислениям (http://www.parallel.ru)
34. Информационные материалы Центра компьютерного моделирования Нижегородского университета им. Н.И.Лобачевского (http://www.software.unn.ac.ru/ccam)
35. Информационные материалы рабочей группы IEEE по кластерным вычислениям (http://www.ieeetfcc.org)
36. Introduction to Parallel Computing (Teaching Course) (http://www.ece.nwu.edu/~choudhar/C58/)
37. Foster I. Designing and Building Parallel Programs. — Addison Wesley, 1994.(http://www.mcs.anl.gov/dbpp)

Предметный указатель

Содержание

Виктор Павлович Гергель,
Анна Андреевна Лабутина

ПараЛаб
Программная система для изучения и исследования методов параллельных вычислений

Учебное пособие

Редактор Е.В. Тамберг

Формат 60x84 1/16. Бумага офсетная № 1. Печать офсетная.
Гарнитура Таймс. Усл. печ. л. 7,3. Уч. изд. л. 7,5
Тираж 300 экз. Заказ .

Издательство Нижегородского государственного
университета им. Н.И. Лобачевского
603590, Н. Новгород, пр. Гагарина, 23

Типография ННГУ: 603000, Н. Новгород, ул. Б. Покровская, 37