УДК 004.421.2

ПАРАЛАБ – СРЕДА ДЛЯ ПРОВЕДЕНИЯ ВЫЧИСЛИТЕЛЬНЫХ ЭКСПЕРИМЕНТОВ ДЛЯ ИЗУЧЕНИЯ И ИССЛЕДОВАНИЯ ПАРАЛЛЕЛЬНЫХ МЕТОДОВ РЕШЕНИЯ СЛОЖНЫХ ВЫЧИСЛИТЕЛЬНЫХ ЗАДАЧ

© 2011 г.

А.А. Лабутина, В.П. Гергель

Нижегородский госуниверситет им. Н.И. Лобачевского

Anna.Labutina@cs.vmk.unn.ru

Поступила в редакцию 05.02.2011

Представлен программный комплекс, который позволяет проводить вычислительные эксперименты для исследования параллельных методов решения трудоемких задач в режиме имитации на одном компьютере с визуализацией процесса решения. Пользователь имеет возможность сформировать вычислительную систему, состоящую из многопроцессорных и многоядерных узлов, соединенных сетью, осуществить постановку задачи, выполнить алгоритм решения, накапливать и анализировать результаты экспериментов.

Ключевые слова: моделирование параллельных вычислений, высокопроизводительные вычисления, параллельные вычисления, кластер, многоядерные архитектуры.

Введение

Развитие компьютерной архитектуры и сетевых технологий, а также появление все более новых, требующих массовых вычислений, научных и прикладных задач показало актуальность параллельных вычислений, выдвинуло их на одно из центральных мест в программировании и вычислительных технологиях.

Но, несмотря на актуальность и необходимость, параллельные вычисления до сих пор не получили столь широкого распространения, которое уже многократно предсказывали многие исследователи. Одна из возможных причин – необходимость построения принципиально новых алгоритмов, решающих поставленные задачи. Известно, что можно достичь ускорения при решении задачи на многопроцессорной вычислительной системе только если разбить алгоритм на совокупность независимых процессов, которые могут выполняться одновременно [1]. Другая причина – сложность отладки параллельных программ, которая делает необходимым глубокое понимание поведения системы параллельно протекающих вычислительных процессов. Тем самым, знание современных тенденций развития ЭВМ и аппаратных средств для достижения параллелизма, умение разрабатывать модели, методы и программы параллельного решения задач обработки данных следует отнести к числу важных квалификационных характеристик современного специалиста по прикладной математике, информатике и вычислительной технике.

«Настоящие» параллельные программы сложны для понимания еще и потому, что не существует простого средства визуализации процесса их выполнения. Это еще более усугубляется тем фактом, что для доступа к многопроцессорным вычислительным системам все чаще используется унифицированный интерфейс, который позволяет пользователю удаленно поставить задачу в очередь на выполнение, а через некоторое время забрать результат выполнения программы, записанный в файл. Для ознакомления с общими принципами работы параллельных алгоритмов наличие параллельной вычислительной системы не является необходимым. Специальная литература может помочь приобрести некоторый базовый уровень теоретических знаний. Для приобретения практических навыков был разработан программный комплекс Параллельная Лаборатория (ПараЛаб), представленный в данной статье.

1. Работа в системе ПараЛаб

При работе в системе ПараЛаб предоставляется широкий спектр возможностей для выбора параметров вычислительного эксперимента. Пользователь может сформировать вычислительную систему, осуществить постановку задачи, выполнить алгоритм решения задачи, накапливать и анализировать результаты вычислительных экспериментов.

1.1. Моделирование вычислительной системы. Система ПараЛаб позволяет эмулировать выполнение параллельных экспериментов на многопроцессорных (*SMP*) и многоядерных (*multicore*) архитектурах [2–4]. Вычислительная система на самом верхнем уровне представляется как состоящая из *компьютеров* (*вычислительных узлов*). Каждый компьютер состоит из одного или нескольких *процессоров*, а каждый процессор – из одного или нескольких *ядер*. Внутреннее представление системы ПараЛаб не ограничивает максимальное количество ядер в процессоре и процессоров в компьютере, но для удобства визуализации введены некоторые ограничения, а именно: число ядер в процессоре может равняться 1, 2 или 4, а число процессоров в компьютере – 1 или 2.

Для формирования модели вычислительной системы необходимо определить топологию сети, количество вычислительных узлов, число процессоров и ядер в одном вычислительном узле, производительность каждого ядра 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 вычислительные узлы содержат одинаковое число процессоров, каждый процессор состоит из одного и того же числа ядер, все ядра обладают одинаковой производительностью, а все каналы связи – одинаковыми характеристиками.

Топология сети передачи данных определяет структуру линий коммутации между вычислительными узлами (компьютерами). В системе ПараЛаб обеспечивается поддержка следующих типовых топологий: линейка, кольцо, звезда, решетка, гиперкуб, полный граф.

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

Под *производительностью ядра* в системе ПараЛаб понимается количество операций с плавающей запятой, которое ядро 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.

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

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

К числу реализованных в системе ПараЛаб методов передачи данных относятся следующие два широко известных способа коммуникации. Первый из них ориентирован на *передачу сообщений* (МПС) как неделимых (атомарных) блоков информации (*store-and-forward routing* or SFR). 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 компьютер может осуществлять пересылку данных по дальнейшему маршруту непосредственно сразу после приема очередного пакета, не дожидаясь завершения приема данных всего сообщения.

1.2. Выбор задачи для решения. В системе ПараЛаб реализованы следующие типы задач: умножение матрицы на вектор, матричное умножение, решение систем линейных уравнений, сортировка, обработка графов, решение дифференциальных уравнений в частных производных и многоэкстремальная оптимизация.

Для каждой из задач предусмотрены, как правило, несколько методов решения [5–8]. Для задачи умножения матрицы на вектор реализованы методы, основанные на блочном, вертикальном и горизонтальном разбиении матрицы. Для задачи матричного умножения реализованы алгоритмы Фокса и Кэннона, а также алгоритм, основанный на ленточном разбиении матрицы. Для задачи решения систем линейных уравнений реализованы параллельные обобщения

методов Гаусса и сопряженных градиентов. Для задачи сортировки данных реализованы параллельные варианты пузырьковой и модифицированной пузырьковой сортировки (когда для сортировки локальных блоков данных используется быстрый алгоритм), а также алгоритмы сортировки Шелла и быстрой сортировки. Для задачи обработки графов реализованы параллельные методы Прима поиска минимального охватывающего дерева, Дейкстры и Флойда поиска кратчайших путей. Для задачи решения дифференциальных уравнений в частных производных реализован параллельный метод Гаусса-Зейделя. Для решения задачи многоэкстремальной оптимизации реализован параллельный индексный метод.

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 некоторых типов задач предоставляется возможность установить некоторые дополнительные параметры. Так, для задач решения дифференциальных уравнений в частных производных предусмотрена возможность выбора типа граничных условий, для задач оптимизации – выбор минимизируемой функции, а для задач обработки графов – возможность формирования графа при помощи встроенного редактора или загрузки графа из файла.

1.3. Выполнение вычислительного эксперимента. Перед проведением вычислительного эксперимента пользователь имеет возможность установить параметры визуализации для выбора желаемого темпа демонстрации, способа отображения пересылаемых между ядрами данных, степени детальности визуализации выполняемых параллельных вычислений.

Эксперимент может быть выполнен в автоматическом и пошаговом режиме, с остановкой после выполнения каждой очередной итерации алгоритма.

В системе ПараЛаб предусмотрена возможность сформировать 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. Для организации анализа полученных данных при проведении длительных вычислений система обеспечивает возможность проведения серии экспериментов в автоматическом режиме с запоминанием результатов в журнале экспериментов.

1.4. Накопление и анализ результатов экспериментов. Накопление итогов экспериментов производится системой ПараЛаб автоматически. При просмотре итогов предоставляется возможность восстановления эксперимента по сохраненной записи и удаления записи.

При сохранении текущего эксперимента в файл происходит запоминание всех записанных результатов.

Для экспериментов, сохраненных в журнале, предусмотрена возможность построения графиков зависимостей времени выполнения и ускорения параллельного алгоритма от параметров задачи и вычислительной системы. Графики строятся согласно теоретическим моделям оценки времени выполнения параллельного алгоритма.

2. Моделирование параллельных вычислений

2.1. Модель выполнения локальных вычислений. Для построения модели время выполнения локальных вычислений можно представить в виде суммы времени вычислений и времени доступа к памяти:

$$T_1 = T_{calc} + T_{mem} \qquad (1)$$

Время вычислений – произведение количества выполненных операций N на время выполнения одной операции τ. Время доступа к памяти – результат деления объема необходимых данных M на пропускную способность памяти β. Для точного построения модели следует учитывать, что данные загружаются в быструю память процессора (кэш) не побайтно, а целыми кэш линейками длиной L байт. В худшем случае каждый элемент, необходимый для проведения вычислений, попадет в отдельную кэш линейку. Следовательно, модель для оценки времени выполнения локальных вычислений может быть представлена следующим образом:

$$T_1 = N \cdot \tau + L \cdot M / \beta \qquad (2)$$

Помимо времени непосредственной загрузки данных, следует также учитывать время задержки оперативной памяти (латентность), так как она может существенно повлиять на время выполнения алгоритма. С учетом латентности

памяти время выполнения вычислений можно подсчитать по следующей формуле:

$$T_1 = N \cdot \tau + M \cdot (\alpha + L/\beta), \quad (3)$$

где α – время латентности оперативной памяти.

Следует отметить, что данная модель рассчитана на худший случай, так как предполагается, что все данные загружаются из оперативной памяти и, кроме того, каждый элемент данных попадает в отдельную кэш линейку. Следовательно, теоретическая оценка времени выполнения алгоритма будет сильно завышена.

Представленная модель не полностью описывает процессы, протекающие при выполнении вычислений современными процессорами. Как уже отмечалось выше, в процессорах существует своя быстрая память – кэш. Для того, чтобы уменьшить затраты на загрузку необходимых данных из оперативной памяти, используются различные алгоритмы предварительной загрузки данных в кэш, эта процедура происходит одновременно с вычислениями. Ситуация, при которой необходимых данных не оказалось в кэше, называется *кэш промахом*. Если произошел промах, то данные загружаются из оперативной памяти. Если же данные находятся в кэше, то промаха не возникает, и не возникает необходимости загружать их из оперативной памяти. Для уточнения модели необходимо измерить количество кэш промахов. Далее, можно скорректировать объем загружаемых из оперативной памяти данных и, следовательно, оценку времени:

$$T_1 = N \cdot \tau + \gamma (L \cdot M/\beta), \quad (4)$$

где γ отношение количества кэш промахов к количеству кэш обращений, которое можно подсчитать теоретически.

Как и для модели (3), в расчетах времени выполнения алгоритма необходимо учесть латентность оперативной памяти:

$$T_1 = N \cdot \tau + \gamma M (L/\beta + \alpha). \quad (5)$$

Итак, для того, чтобы вычислить время выполнения локальных вычислений, необходимо точно определить следующие параметры:

– β – пропускная способность оперативной памяти,

– α – латентность оперативной памяти,

– γ – отношение количества кэш промахов к общему количеству кэш обращений,

– τ – время выполнения одной операции алгоритма.

Для того, чтобы оценить точность построенной математической модели, были проведены вычислительные эксперименты. Для проведения таких экспериментов использовался компьютер на базе процессора Intel Core 2 Quad Q6600. Архитектура процессора включает 4 кэша L1 (по одному на каждое ядро) объемом 64 Кб с пропускной способностью 153 Гбайт/с, латентностью 1.22 нс. Оперативная память экспериментальной вычислительной системы обладает пропускной способностью 12.4 Гбайт/с и латентностью 8.31 – 80 нс. Рассматривалась задача умножения матрицы на вектор. Исходный код программы, реализующей данный алгоритм, представлен ниже:

```
for (i=0; i<Size; i++) {
  pResult[i] = 0;
  for (j=0; j<Size; j++)
    pResult[i] += pMatrix[i*Size+j]*pVector[j];
}
```

Для того чтобы оценить время одной операции τ, необходимо измерить время выполнения последовательного алгоритма умножения матрицы на вектор при малых объемах данных, таких, чтобы матрица и вектор полностью поместились в кэш L1. Поделив полученное время на количество выполненных операций, получим время выполнения одной типовой операции алгоритма $\tau = 3.78$ нс.

В текущей версии системы ПараЛаб реализована более простая модель для оценки времени выполнения локальных вычислений, включающая в себя только количество вычислительных операций и время выполнения одной операции τ. В следующей версии системы планируется использовать уточненные оценки в соответствии с подходом, описанным в данном разделе.

2.2. Модель выполнения операций передачи данных. 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:

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

• *время передачи служебных данных* (t_c) между двумя соседними процессорами (т.е. для процессоров, между которыми имеется физический канал передачи данных); к служебным данным может относиться заголовок сообщения, блок данных для обнаружения ошибок передачи и т.п.;

• *время передачи одного слова данных по одному каналу передачи данных* ($1/\beta$); длительность подобной передачи определяется полосой пропускания коммуникационных каналов в сети.

Рассмотрим метод передачи сообщений (*store-and-forward routing or SFR*). При таком подходе процессор, содержащий сообщение для передачи, готовит весь объем данных для передачи, определяет процессор, которому следует направить данные, 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

$$t_{пд} = \alpha + (t_c + \frac{m}{\beta})l \cdot \qquad (6)$$

При достаточно длинных сообщениях временем передачи служебных данных можно пренебречь и выражение для времени передачи данных может быть записано в более простом виде

$$t_{пд} = \alpha + \frac{m}{\beta} l \cdot \qquad (7)$$

Рассмотрим метод передачи пакетов (*cut-through routing or CTR*). При таком способе коммуникации принимающий процессор может осуществлять пересылку данных по дальнейшему маршруту непосредственно сразу после приема очередного пакета, не дожидаясь завершения приема данных всего сообщения. Время пересылки данных при использовании метода передачи пакетов будет определяться выражением

$$t_{пд} = \alpha + \frac{m}{\beta} + t_c l \cdot \qquad (8)$$

Сравнивая полученные выражения, можно заметить, что в большинстве случаев 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. С другой стороны, реализация пакетного метода требует разработки более сложного аппаратного и программного обеспечения сети, может увечить накладные расходы (время подготовки и время передачи служебных данных); при передаче пакетов возможно возникновения конфликтных ситуаций (дедлоков).

2.3. Особенности передачи данных в многопроцессорных и многоядерных архитектурах. Как уже отмечалось ранее, в рамках системы ПараЛаб вычислительная система представляется как состоящая из компьютеров, связи между которыми определяются формой топологии (линейка, кольцо и т.п.). Каждый компьютер состоит из одного или нескольких *процессоров*, а каждый процессор – из одного или нескольких *ядер*. При этом предполагается, что внутренние линии связи (шины) между ядрами в пределах одного вычислительного узла (в том числе из разных процессоров) образуют топологию полный граф. Возможное число компьютеров в топологии определяется самой топологией.

Для упрощения модели предполагается, что процессы передачи данных и выполнения вычислительных операций ядрами разделены во времени, то есть во время обмена данными вычисления на всех ядрах приостанавливаются и, наоборот, при выполнении вычислительных операций хотя бы на одном ядре во всей топологии не происходит ни одного обмена данными.

Каждая коллективная операция обмена данными между ядрами подразделяется на 3 этапа:

1. Обмен данными между ядрами в рамках одного компьютера (в том числе из разных процессоров), а также отправка ими данных во внешнюю сеть (через сетевой адаптер).

2. Передача данных между отдельными компьютерами по локальной сети. При этом если все данные в пакете были отправлены одним и тем же ядром, то этот пакет визуализируется отдельным конвертом, если же в нем находятся данные сразу от нескольких ядер – то пачкой конвертов.

3. Прием данных отдельными ядрами в рамках одного компьютера (от сетевого адаптера).

Чтобы произвести оценку времени, потраченного на передачу данных, требуется следующая информация:

• Размер одной условной единицы обмена данными, выраженный в байтах.

• Количество единиц передаваемых данных для каждой пары ядер, обменивающихся информацией на текущей итерации.

При определении итогового времени передачи данных учитывается только время второго этапа (передача данных по локальной сети). Время, затрачиваемое на передачу данных по шине в пределах одного компьютера, на 3–4 порядка меньше и не вносит заметного вклада в суммарную оценку, поэтому им можно пренебречь.

Для каждой отдельной пары ядер, обменивающихся данными по локальной сети, время операции определяется по описанным в предыдущих разделах формулам, а за итоговое время операции обмена принимается наибольшее из них.

2.4. Пример оценки времени выполнения эксперимента. Рассмотрим трудоемкость параллельного алгоритма умножения матрицы на вектор, основанного на разбиении матрицы на горизонтальные полосы. В случае, если матрица квадратная размера $n \times n$, последовательный алгоритм умножения матрицы на вектор имеет сложность $O(n^2)$. В случае параллельных вычислений каждое ядро производит умножение только части (полосы) матрицы на вектор, размер этих полос равен n/p строк (p – число ядер). При вычислении скалярного произведения одной строки матрицы и вектора необходимо произвести n операций умножения и $(n-1)$ операций сложения. Предположим, что выполняемые операции умножения и сложения имеют одинаковую длительность τ. Будем предполагать также, что вычислительная система является однородной, т.е. все ядра, составляющие эту систему, обладают одинаковой производительностью. С учетом введенных предположений время выполнения параллельного алгоритма, связанное непосредственно с вычислениями, составляет:

$$T_p(calc) = \lceil n/p \rceil \cdot (2n-1) \cdot \tau . \qquad (10)$$

Для сбора результирующего вектора используется операция обобщенного сбора данных. Данная операция может быть выполнена за $\lceil \log_2 p \rceil$ итераций. На первой итерации взаимодействующие пары процессоров обмениваются сообщениями объемом $w\lceil n/p \rceil$ байт (w есть размер одного элемента вектора в байтах), на второй итерации этот объем увеличивается вдвое и оказывается равным $2w\lceil n/p \rceil$ и т.д. Как результат, длительность выполнения операции сбора данных при использовании модели Хокни [2] может быть определена при помощи следующего выражения:

$$T_p(comm) = \sum_{i=1}^{\lceil \log_2 p \rceil} \left(\alpha + 2^{i-1} w \lceil n/p \rceil / \beta \right) = \\ = \alpha \lceil \log_2 p \rceil + w \lceil n/p \rceil \left(2^{\lceil \log_2 p \rceil} - 1 \right) / \beta . \qquad (11)$$

где α – латентность сети передачи данных, β – пропускная способность сети. Таким образом, общее время выполнения параллельного алгоритма составляет

$$T_p = (n/p)(2n-1)\tau + \alpha \log_2 p + w(n/p)(p-1)/\beta \qquad (12)$$

(для упрощения выражения в (11) предполагалось, что значения (n/p) и $\log_2 p$ являются целыми).

Рассмотрим результаты вычислительных экспериментов для сравнения теоретических оценок и экспериментальных показателей времени вычислений и проверим тем самым точность полученных аналитических соотношений. Эксперименты проводились на вычислительном кластере на базе процессоров Intel XEON 4 EM64T, 3000 МГц и сети Gigabit Ethernet под управлением операционной системы Microsoft Windows Server 2003 Standart x64 Edition.

Определение параметров теоретических зависимостей (величин τ, w, α, β) осуществлялось следующим образом. Для оценки длительности τ базовой скалярной операции проводилось решение задачи умножения матрицы на вектор при помощи последовательного алгоритма и полученное таким образом время вычислений делилось на общее количество выполненных операций – в результате подобных экспериментов для величины τ было получено значение 1.93 нсек. Эксперименты, выполненные для определения параметров сети передачи данных, показали значения латентности α и пропускной способности β соответственно 47 мкс и 53.29 Мбайт/с. Все вычисления производились над числовыми значениями типа double, т. е. величина w равна 8 байт.

Сравнение экспериментального времени T_p^* выполнения параллельного алгоритма и теоретического T_p времени, вычисленного в соответствии с выражением (12), представлено в табл. 2. Эксперименты проводились с использованием двух, четырех и восьми процессоров.

Времена выполнения алгоритмов указаны в секундах.

Заключение

Программная система 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 различными типами вычислительных узлов и 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.

Система ПараЛаб используется в Нижегородском государственном университете и других университетах России для организации лабораторного практикума по курсу «Высокопроизводительные вычисления» для студентов, только начинающих знакомиться с тематикой параллельных вычислений. ПараЛаб является частью разработанного в ННГУ по заказу Microsoft курса «High Performance Computing», который вышел в свет осенью 2007 года и представлен на сайте Microsoft Curricula.

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.

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.

Таблица 1

Сравнение экспериментального и теоретического времени выполнения алгоритма умножения матрицы на вектор

Размер матрицы	Среднее экспериментальное время выполнения алгоритма	Теоретическое время выполнения алгоритма	Относительная погрешность модели
100	0.000076	0.000075531	0.006170109
1000	0.0076	0.007591533	0.001114034
2000	0.030315	0.030377419	0.002059011
3000	0.068784	0.068488307	0.004298866
4000	0.122176	0.121739087	0.003576094
5000	0.190931	0.190254968	0.003540712
6000	0.27478	0.273983619	0.002898249
7000	0.374114	0.372877235	0.003305852
8000	0.489386	0.487209399	0.004447616
9000	0.61864	0.616418456	0.003591012
10000	0.763734	0.76110718	0.003439443

Таблица 2

Результаты вычислительных экспериментов для параллельного алгоритма умножения матрицы на вектор при ленточной схеме разделении данных по строкам

Размер объектов	2 процессора		4 процессора		8 процессоров	
	T_p	T_p^*	T_p	T_p^*	T_p	T_p^*
1000	0.0069	0.0021	0.0108	0.0017	0.0152	0.0175
2000	0.0132	0.0084	0.014	0.0047	0.0169	0.0032
3000	0.0235	0.0185	0.0193	0.0097	0.0196	0.0059
4000	0.0379	0.0381	0.0265	0.0188	0.0233	0.0244
5000	0.0565	0.0574	0.0359	0.0314	0.028	0.015

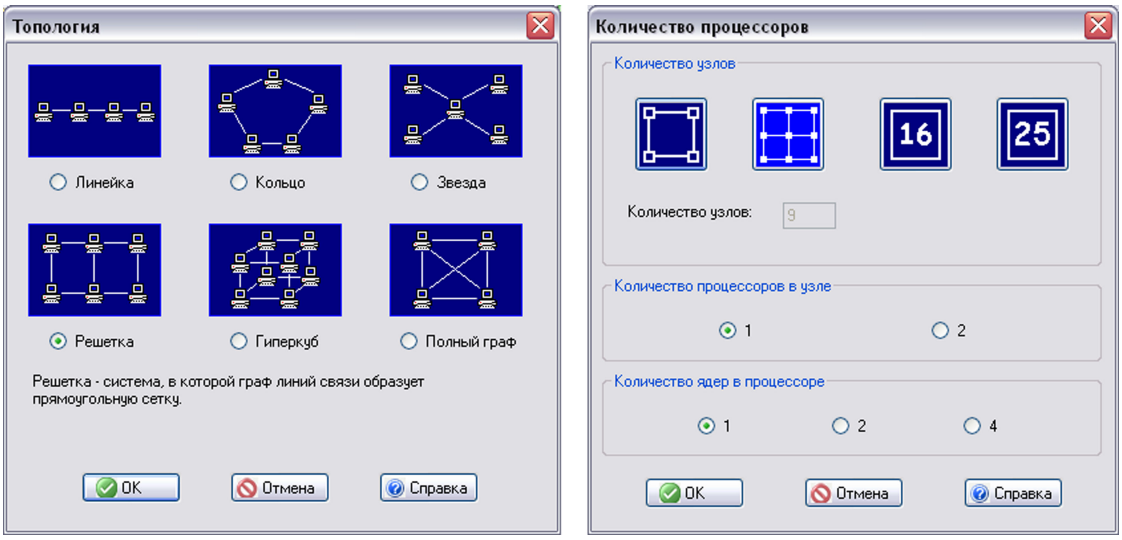

Рис. 1. Диалоговые окна для задания параметров вычислительной системы

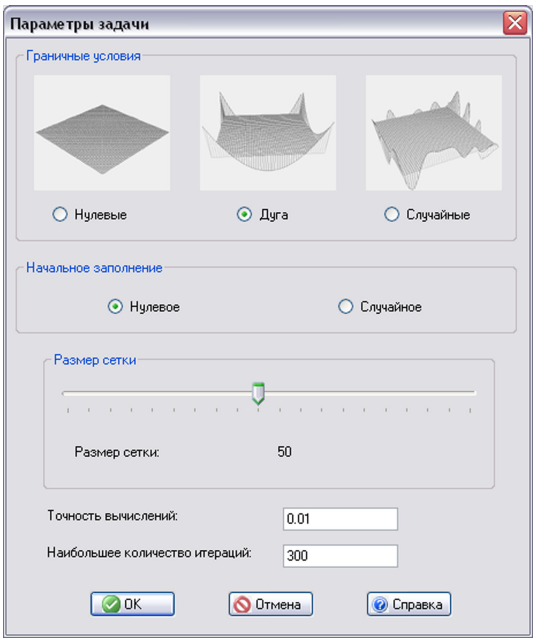

Рис. 2. Диалоговое окно для задания параметров задачи решения дифференциального уравнения в частных производных

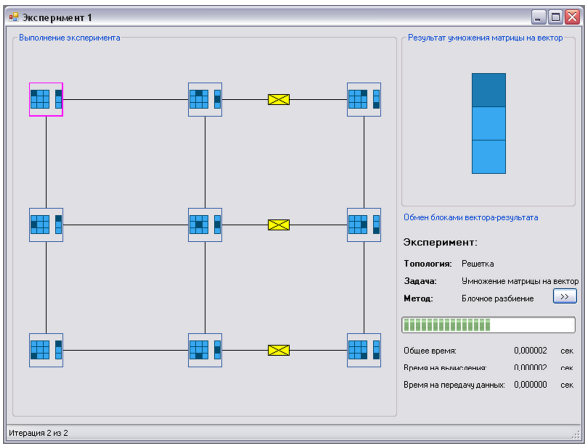

Рис. 3. Окно вычислительного эксперимента при решении задачи умножения матрицы на вектор

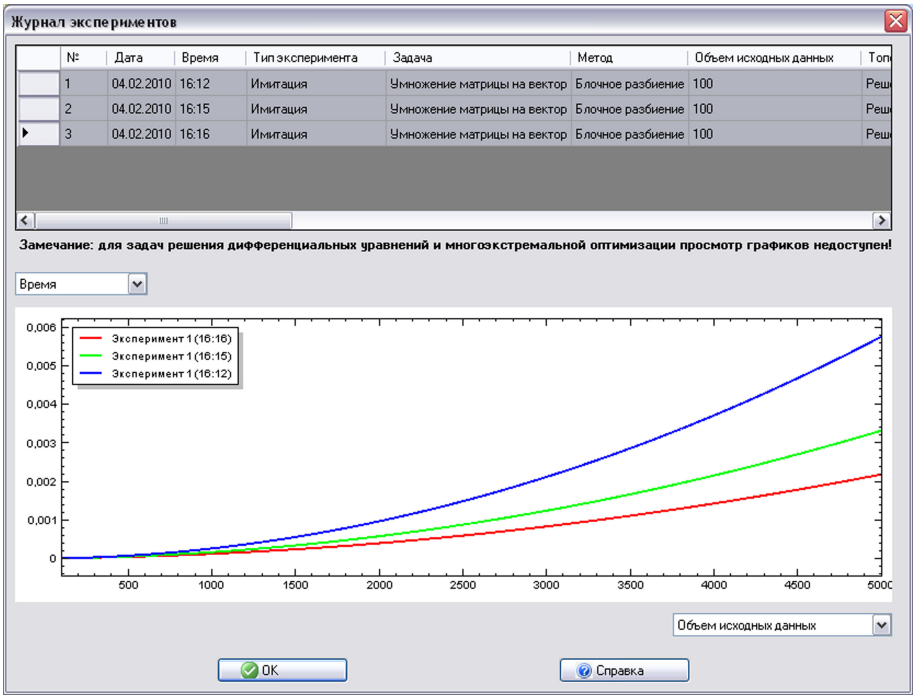

Рис. 4. Окно журнала экспериментов

Список литературы

1. Foster I. Designing and Building Parallel Programs: Concepts and Tools for Software Engineering. Reading, MA: Addison-Wesley, 1995.

2. Hockney R.W., Jesshope C.R. Parallel Computers 2. Architecture, Programming and Algorithms. Adam Hilger, Bristol and Philadelphia, 1988.

3. Rajkumar Buyya. High Performance Cluster Computing. Volume 1: Architectures and Systems. Volume 2: Programming and Applications. Prentice Hall PTR, Prentice-Hall Inc., 1999.

4. Xu Z., Hwang K. Scalable Parallel Computing Technology, Architecture, Programming. Boston: McGraw-Hill, 1998.

5. Kumar V., Grama A., Gupta A., Karypis G. Introduction to Parallel Computing. The Benjamin / Cummings Publishing Company, Inc., 1994.

6. Quinn M.J. Parallel Programming in C with MPI and OpenMP. NY: McGraw-Hill, 2004.

7. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. СПб., БХВ – Петербург, 2002.

8. Гергель В.П., Теория и практика параллельных вычислений. БИНОМ. Лаборатория знаний, Интернет-университет информационных технологий – ИНТУИТ.ру, 2007.

PARALLEL LABORATORY (PARALAB) IS A SYSTEM FOR CARRYING OUT COMPUTING EXPERIMENTS TO STUDY AND INVESTIGATE PARALLEL METHODS OF SOLVING COMPLEX COMPUTATIONAL PROBLEMS

A.A. Labutina, V.P. Gergel

A software package is presented which makes it possible to carry out computing experiments to study and investigate parallel methods of solving complex computational problems in a simulation mode on a single computer with visualization of the solution process. The user has the opportunity to form a computing system consisting of multiprocessor and multicore nodes connected in a network, to build the problem statement, to execute the solution algorithm, to accumulate and analyse the experiment results.

Keywords: parallel computing simulation, high-performance computing (HPC), parallel computing, cluster, multi-core architectures.