Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
Национальный исследовательский университет «МЭИ»

Лабораторные работы по дисциплине

«Архитектура вычислительных систем. Часть 2»

Автор
Доцент кафедры ПМ О.Ю.Шамаева

Лабораторные работы №№2 – 5 основаны на пособии:
И.И. Ладыгин, А.А. Крюков, Г.А.Калинина. Архитектура ЭВМ и систем: Лабораторный практикум. – М.:МГТУ "СТАНКИН", 2003. – 35 с.

Москва, 2022

Оглавление

Лабораторная работа №1
Распараллеливание процессов вычислений на примере вычисления выражений

Целью работы является приобретение навыков по распараллеливанию простых и рекурсивных арифметических выражений (АВ) и оценке характеристик параллельности и эффективности решения.

1. Теоретическое введение

1.1. Основные подходы к распараллеливанию

Проблема распараллеливания арифметических и логических выражений представляет большой интерес [1-5]. К настоящему времени разработано достаточно много алгоритмов распараллеливания арифметических выражений, некоторые из которых положены в основу распараллеливающих программ и трансляторов.

Пусть E – простое арифметическое выражение, удовлетворяющее следующему условию:

"*Каждая из переменных входит в E ровно один раз*". (*)

Выполнения этого условия всегда можно добиться переименованием переменных.

Цель любого алгоритма распараллеливания – минимизировать время, необходимое для параллельного вычисления E.

Наиболее известные алгоритмы распараллеливания арифметических выражений Баера-Бовета, Брента и Винограда [2,5] основаны на общем принципе: *ориентированный ациклический граф, описывающий последовательное вычисление выражения E, представляет собой, в силу свойства* (*), *бинарное дерево*.

К этому дереву применяются преобразования в соответствии с законами коммутативности, ассоциативности и дистрибутивности. В результате бинарное дерево трансформируется в ориентированный ациклический граф $\tilde{G}$, соответствующий выражению $\tilde{E}$, эквивалентному E.

Таким образом, ***основная цель*** распараллеливания арифметических выражений – разработать дерево вычислений минимальной высоты. То есть, при условии реализации на каждом ярусе максимального числа независимых операций, выражение должно вычисляться за минимально возможное время или за минимальное число шагов. Распараллеливание АВ осуществляется в рамках модели MIMD (Multiple Instruction Multiple Date – множественный поток команд и множественный поток данных) с неограниченным параллелизмом.

По сравнению с операторными схемами выражения имеют принципиальное отличие – в них не всегда указан порядок применения операций. В том случае, когда в выражении присутствуют скобки, имеется

некая свобода в применении операций. Даже если опускать некоторые скобки, то дерево выражения также строится неоднозначно. Например, $a*b*c$ может быть “восстановлено” как $a*(b*c)$ или $(a*b)*c$.

Порядок вычисления может быть перераспределен или доопределен таким образом, чтобы некоторые операции можно было выполнять параллельно. Поэтому говорят, что выражения обладают *неявным параллелизмом*.

Задача распараллеливания выражений заключается в построении такого алгоритма, который по каждому выражению E дает эквивалентное ему $\tilde{E}$ с минимальной высотой дерева вычислений.

Два выражения E и $\tilde{E}$ называются *эквивалентными* ($E \sim \tilde{E}$) в том и только в том случае, если выражение E преобразуется в $\tilde{E}$ и наоборот, путем использования конечного числа раз законов ассоциативности, коммутативности и дистрибутивности.
Рассмотрим процесс вычисления арифметического выражения

$$E = (x + (a * ((b / c) * d))) - (y - z)$$

и эквивалентного ему:

$$\tilde{E} = (a * b) / (c / d) - ((y - z) - x).$$

На рис.1 и рис.2 приведены деревья вычислений для двух эквивалентных арифметических выражений E и $\tilde{E}$.

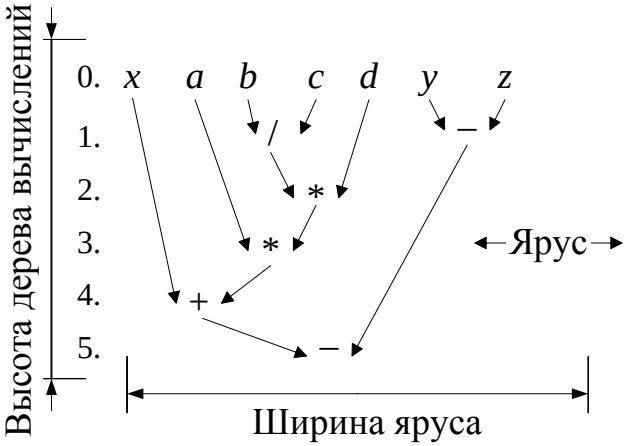

Рис. 1. Дерево вычислений для E

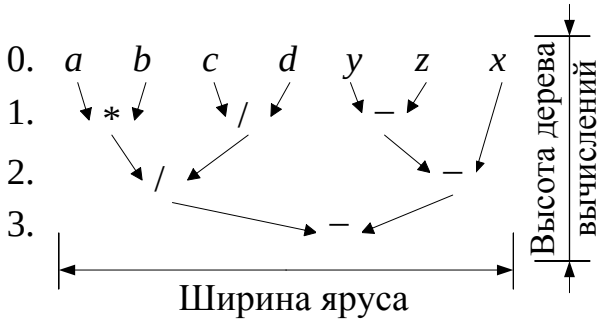

Рис. 2. Дерево вычислений для $\tilde{E}$

1.2. Характеристики сложности и параллельности

Характеристиками сложности вычисления арифметического выражения являются:

- время ***t***, затрачиваемое на вычисление арифметического выражения;
- общее число операций ***w***, необходимое для вычисления выражения;
- число вычислителей (или процессоров) ***p***, требуемое для реализации вычислений.

Если считать, что любая операция занимает одну единицу времени, то время ***t*** вычисления арифметического выражения равно числу ярусов (высоте) дерева вычислений. Операции, лежащие на одном ярусе дерева, могут выполняться параллельно и определяют ширину данного яруса. Поэтому высота дерева соответствует числу шагов параллельного алгоритма для арифметического выражения.

Требуемое число вычислителей (или процессоров) ***p*** определяется как максимальная ширина яруса дерева вычислений.

Для выражений E и $\tilde{E}$ имеем следующие характеристики сложности вычисления:

Для выражения E	Для выражения $\tilde{E}$
$t = 5$	$t = 3$
$p = 2$	$p = 3$
$w = 6$	$w = 6$

Следует отметить, что использование законов дистрибутивности при преобразованиях арифметических выражений в эквивалентные может привести к переполнению разрядной сетки. Например, если в выражении $a*(b–c)$, a, b и c – большие положительные числа, то умножение $a*(b–c)$ проходит без коллизий, так как величины b и c “компенсируют” друг друга, в то время как при порядке вычисления $a*b– a*c$ при умножениях может возникнуть переполнение.

Впрочем, подобное может случиться и при использовании других законов, например, ассоциативности.

Например, $(a*b)*c$ эквивалентно $a*(b*c)$, но при больших a и b и $c<<1$ при этом порядке умножения может возникнуть переполнение, которого бы не было при умножении $a*(b*c)$. Даже при перестановке аргументов ($a*b = b*a$) в некоторых умножителях может возникнуть переполнение или потеря точности. Поэтому во многих случаях процедура распараллеливания распространяется только на бесскобочные фрагменты выражения, т.е. на фрагменты, в которых очередность определяется лишь приоритетностью операций и никак не ограничена программистом.

Для оценки распараллеленного выражения будем использовать такие характеристики параллельности, как ***степень параллелизма***, ***ускорение*** и ***эффективность*** параллельных алгоритмов.

Пусть n – число параметров задачи (для арифметического выражения – число различных переменных, входящих в арифметическое выражение).

$T_p(n)$ – время выполнения параллельного алгоритма на вычислительной системе с числом процессоров p>1.

$T_1(n)$ – время выполнения “наилучшего” последовательного алгоритма.

Степень параллелизма задачи (алгоритма) определяется как максимально возможное число независимых элементарных операций (действий), выполняемых при реализации задачи (алгоритма) на *неограниченных аппаратных ресурсах* (в режиме паракомпьютинга).

Ускорением S_p параллельного алгоритма называют отношение $S_p(n) = \frac{T_1(n)}{T_p(n)}$, а ***эффективность*** E_p параллельного алгоритма определяется по формуле $E_p(n) = \frac{S_p(n)}{p}$.

Очевидно, чем ближе значение $S_p(n)$ к p, а $E_p(n)$ – к 1, тем “лучше” построенный параллельный алгоритм.

Имеем следующие характеристики параллельных вычислений:

Для выражения E	Для выражения $\tilde{E}$
$T_1(n) = n - 1 = 6$	$T_1(n) = n - 1 = 6$
$T_p(n) = t = 5$	$T_p(n) = t = 3$
$S_p(n) = 6/5 = 1.2$	$S_p(n) = 6/3 = 2$
$E_p(n) = 1.2/2 = 0.6$	$E_p(n) = 2/3 = 0.66$

Таким образом, выражение $\tilde{E}$ имеет лучшие характеристики параллельности, чем выражение E.

Проанализируем дерево вычислений на рис.2 с точки зрения оптимизации загрузки процессоров и наилучшей эффективности. Попробуем уменьшить число процессоров, но при этом загрузим их более равномерно, как представлено на рис.3.

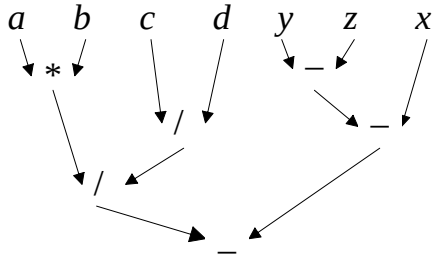

Рис. 3. Дерево вычислений для оптимизации загрузки процессоров

Получим следующие характеристики параллельности:

$T_1(n) = w = 6$ $S_p(n) = 6/4 = 1.5$

$T_p(n) = t = 4$ $E_p(n) = 1.5/2 = 0.75$

В результате оптимизации загрузки процессоров получили по сравнению с деревом на рис.2 несколько меньшее ускорение, но более высокую эффективность.

Введем в рассмотрение еще две полезные характеристики, а именно ***цену*** и ***ценность параллельного решения***.

Цена параллельного решения

$$C_p = p*T_p(n)$$

Ценность параллельного решения

$$F_p = S_p(n)/C_p = T_1(n)/(p*T_p^2(n)).$$

В заключении этого раздела приведем полезное утверждение, сформулированное в виде леммы Брента [6,7].

Лемма Брента. Если при неограниченном числе процессоров для вычисления АВ, состоящего из w операций, требуется время t, то при наличии ограниченного числа процессоров p' вычисление АВ может быть выполнено не более чем за время t', определяемое по формуле:

$$t' = t + (w - t) / p'.$$

Данное утверждение позволяет, зная характеристики АВ при распараллеливании на неограниченном числе процессоров, оценить время реализации выражения на ограниченном числе вычислителей. Проверим лемму для полученного на рис.3 дерева вычисления:

- число операций $w = 6$,
- время выполнения на неограниченном числе процессоров равно $t = 3$ (см. рис.2),
- при наличии процессоров $p' = 2$ время выполнения, в соответствии с леммой, составит не более $t' = 3 + (6 - 3)/2 = 4.5$.

Реально, исходя из рис.3, $t = 4$, то есть $t < t'$ и можно утверждать, что лемма выполняется.

1.3. Распараллеливание рекурсивных схем

Важнейшим классом арифметических выражений являются рекурсивные выражения. Рекурсия встречается в матричных операциях, является основой многих методов для вычисления значений функции и используется в ряде методов численного интегрирования.

При рекурсивных операциях значение каждого очередного члена последовательности или элемента структуры зависит от значений одного или нескольких предшествующих членов.

В последовательных ЭВМ рекурсия реализуется в виде циклических процессов. Рекурсия встречается не только при числовой обработке, но и при обработке сигналов, операциями над символами и т.п. Рекурсивный характер обработки накладывает ограничения на возможности распараллеливания, которое может быть достигнуто лишь реорганизацией последовательности обработки.

Остановимся на простейшем примере линейной рекурсии первого порядка для вычисления суммы $S = \sum_{i=1}^{n} x_i$ всех элементов вектора $X = \{x_i\}$, где i =1, 2, …,n . В общем случае, в качестве операции может выступать любая ассоциативная операции. К числу таких операций относятся сложение, умножение, поиск максимума и минимума среди вещественных чисел, операции сложения и умножения матриц.

Для нахождения этой суммы можно воспользоваться следующим рекурсивным соотношением:

$$S_i = S_{i-1} + x_i,\ i = 1, 2, \ldots, n.$$

Последовательная реализация вычисления этого выражения иллюстрируется информационным графом на рис. 4, и поскольку каждый из операндов используется однократно, такой граф представляет собой бинарное дерево. Если выполнение каждой примитивной операции занимает время t, то суммарные затраты времени (при высоте графа q=n–1) реализации этого графа составят $T_1(n) = t * (n - 1) = t * q$.

Очевидно, что при наличии нескольких вычислителей алгоритм будет неэффективным, и задача состоит в том, чтобы минимизировать высоту дерева. Преобразование выполняется в соответствии с законами коммутативности, ассоциативности и дистрибутивности, в результате чего может быть получен трансформированный граф, показанный на рис. 5.

Высота такого дерева $q'=[\log_2 n]$, а длительность реализации составит $T_p=t*[\log_2 n]$ в предположении, что операции суммирования реализуются за то же время t и затраты времени на обмен между отдельными вычислителями эквивалентны затратам времени на обмен между ячейкой памяти и вычислителем в последовательной структуре.

Такой метод нахождения суммы элементов вектора получил название ***метода рекурсивного сдваивания*** или ***метода нахождения параллельных каскадных сумм***.

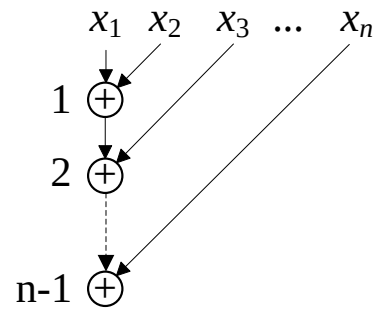

Рис. 4. Прямое суммирование

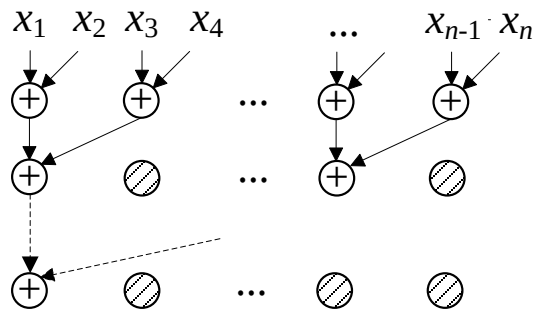

Рис. 5. Попарное суммирование

Максимальное ускорение $S_p=T_1/T_p$ можно оценить, как O(n/log$_2$n). Оно достигается при числе вычислителей не менее n/2, причем полностью все

вычислители загружены только на первом шаге вычислений, на последнем же шаге используется лишь один вычислитель.

Не используемые на каждом шаге вычислители можно отразить на графе в виде «пустых» операторов (заштрихованных кругов на рис. 5). Тогда эффективность построенной схемы алгоритма можно оценить, как отношение числа действительных к общему числу операторов (действительных и пустых). Очевидно, что такое определение понятия эффективности сродни понятию коэффициента полезного действия. Для рассматриваемого алгоритма оно составляет 0.5 и не зависит от n.

Рассмотренный принцип рекурсивного сдваивания справедлив для произвольной ассоциативной бинарной операции на произвольном множестве.

Более общим случаем линейной рекурсии является вычисление значения полинома по схеме Горнера:

$$S = (\ldots(d_0 a_1 + d_1)\cdot a_2 + d_2)\cdot a_3 + d_3)\ldots)\cdot a_n + d_n).$$

Отличительной особенностью схемы является чередование аддитивных (сложение, вычитание) и мультипликативных (умножение, деление, обратное деление) операций. Подобные выражения называют ***альтернированными***.

Рекурсивный алгоритм для реализации этой схемы имеет вид:

$$S_i = S_{i-1}\cdot a_i + d_i, \quad i = 1,2,\ldots,n \text{ и } S_0 = d_0.$$

Последовательная программа для нахождения S по этой схеме подобна приведенной выше для нахождения суммы, лишь команде сложения должна предшествовать команда умножения. Схема Горнера и рекурсивный алгоритм ее реализации весьма удобны для последовательных вычислений, однако их непосредственная реализация в параллельных вычислениях не приводит к ускорению, так как для выполнения очередной операции необходим результат, полученный на предыдущем шаге.

Известно, что альтернированные выражения всегда могут быть представлены путем разложения в виде функции нескольких арифметических выражений [3], например, в виде:

$$S = A + B,$$

где каждое арифметическое выражение A и B содержит не более n арифметических операций, что доказывает возможность построения более быстрого параллельного алгоритма.

Поскольку основным ограничением для параллельной реализации рассматриваемой рекурсии является зависимость текущей операции от результата предыдущей, преобразуем схему вычислений так, чтобы вычисляемое значение S_i на текущем шаге было связано с результатом, полученным не на предыдущем шаге, а на один шаг раньше, т.е. с S_{i-2}:

$$S_i = S_{i-1} \cdot a_i + d_i,$$

$$S_{i-1} = S_{i-2} \cdot a_{i-1} + d_{i-1}.$$

После подстановки второго выражения в первое получим:

$$S_i = (S_{i-2} \cdot a_{i-1} + d_{i-1}) \cdot a_i + d_i = S_{i-2} \cdot a_{i-1} \cdot a_i + d_{i-1} \cdot a_i + d_i.$$

Обозначив в этом выражении через $a_i^{(1)} = a_i \cdot a_{i-1}$ и $d_i^{(1)} = d_{i-1} \cdot a_i + d_i$, где верхний индекс означает номер последовательно применяемого описанного преобразования, получим:

$$S_i = S_{i-2} \cdot a_i^{(1)} + d_i^{(1)}.$$

Последнее выражение также является рекурсией, и к нему вновь можно применить это преобразование:

$$S_{i-2} = S_{i-4} \cdot a_{i-2}^{(1)} + d_{i-2}^{(1)},$$

$S_i = (S_{i-4} \cdot a_{i-2}^{(1)} + d_{i-2}^{(1)}) \cdot a_i^{(1)} + d_i^{(1)} = S_{i-4} \cdot a_i^{(2)} + d_i^{(2)}$, где $a_i^{(2)} = a_i^{(1)} \cdot a_{i-2}^{(1)}$ и $d_i^{(2)} = d_{i-2}^{(1)} \cdot a_i^{(1)} + d_i^{(1)}$.

Последовательное применение указанного преобразования носит название ***циклической редукции***, на каждом шаге которой получаем:

$$a_i^{(k)} = a_i^{(k-1)} \cdot a_{i-2^{k-1}}^{(k-1)} \text{ и } d_i^{(k)} = d_{i-2^{k-1}}^{(k-1)} \cdot a_i^{(k-1)} + d_i^{(k-1)},$$

где k – номер шага редукции.

В результате применения $\log_2 n$ шагов получим

$$S = A + B,$$

где $A = a_n a_{n-1} \ldots a_1 d_0$, $B = a_n a_{n-1} \ldots a_2 d_1 + a_n a_{n-1} \ldots a_3 d_2 + \ldots + d_n$.

Таким образом, нахождение значения полинома сводится к нахождению каскадной суммы произведений. Параллельный алгоритм для вычисления коэффициентов $a_i^{(k)}$ аналогичен рассмотренному выше для нахождения сумм элементов вектора (операция сложения заменяется на операцию умножения).

Рассмотрим пример вычисления схемы Горнера для n=4.

$S = (((d_0 a_1 + d_1) \cdot a_2 + d_2) \cdot a_3 + d_3) \cdot a_4 + d_4$, $S_0 = d_0$.

Используя формулы для циклической редукции, с учетом введенных выше обозначений, имеем:

$$S_4 = S_2 \cdot a_4^{(1)} + d_4^{(1)} = S_2 \cdot a_3 \cdot a_4 + d_3 \cdot a_4 + d_4,$$

$$S_2 = S_0 \cdot a_2^{(1)} + d_2^{(1)} = d_0 \cdot a_1 \cdot a_2 + d_1 \cdot a_2 + d_2.$$

Для данного примера дерево параллельного вычисления может быть представлено в следующем виде (см. рис. 6):

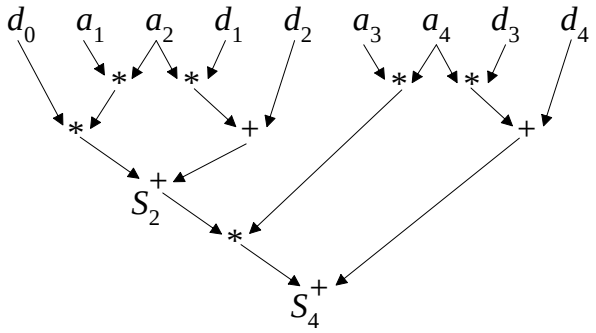

Рис.6. Дерево параллельного вычисления схемы Горнера для S_4

Все арифметические операции сгруппированы по уровням; однотипные операции умножения или сложения, находящиеся в отношении безразличия, могут выполняться параллельно.

Характеристики сложности и параллельности для примера:

w_p=10, p=4, $T_1(n)$=8, $T_p(n)$=5, $S_p(n)$=8/5=1,6, $E_p(n)$=1,6/4=0,4.

Построенное на рис.6 дерево вычисления можно перестроить так, чтобы повысить эффективность вычислений до $E_p(n)$=1,6/3=0,53 (предлагается выполнить самостоятельно).

2. Подготовка к работе и порядок ее выполнения

Изучить материал лекций по теме «Распараллеливание выражений».

Ознакомиться с теоретическим введением к данной работе.

Для своего варианта из перечня арифметических выражений, используя программу **Lab_Work1.exe**:

1. Построить дерево параллельного вычисления арифметического выражения минимальной высоты.

2. Определить характеристики сложности и параллельности (**степень параллелизма выражения, ускорение и эффективность, цена и ценность**) для построенной параллельной схемы.

3. Проанализировать построенную параллельную схему с точки зрения оптимизации загрузки процессоров и наилучшей эффективности, если необходимо, перестроить дерево вычислений.

4. Определить **ускорение и эффективность** параллельной схемы для случая оптимальной загрузки процессоров. Сравнить эти характеристики со значениями, полученными в п.2.

5. Графически исследовать **зависимости характеристик ускорения и эффективности от числа процессоров**. Определить **число процессоров,** соответствующее наилучшему распараллеливанию выражения как с точки зрения ускорения, так и эффективности.

6. Проверить лемму Брента на применимость к данному выражению, исследовав несколько параллельных схем вычислений с различным числом процессоров.

7. Для вычисления значения полинома по схеме Горнера осуществить разложение схемы по методу циклической редукции. Построить вручную дерево параллельного вычисления. Определить характеристики сложности и параллельности. Вариант с четным номером выполняется для n = 8, с нечетным – для n = 9.

8. Проанализировать построенную параллельную схему для вычисления полинома с точки зрения оптимизации загрузки процессоров и, если необходимо, перестроить дерево вычислений. Определить характеристики параллельности полученного дерева вычислений.

9. Построить дерево параллельного вычисления схемы Горнера с помощью программы **Lab1_Work1.exe**, используя полученное в п.7 разложение, переименовав переменные в нем таким образом, чтобы они не повторялись, и избегая, по возможности, в записи выражения подряд идущих скобок.

10. Оформить результаты выполнения всех пунктов в отчет по работе.

11. При подготовке к защите работы ответить на контрольные вопросы

3. Список арифметических выражений для выполнения задания

1. (r/t/y/u + i∗o + p – a + s)+(d – f)∗(g + h)∗(j + k)
2. (a∗(s – d) + f + g + h + j + k)∗(l + z – x/(c + v) + b∗n)
3. (z + x + c + v + b)/(n + m∗q + w/e + r/t∗y) – u + i
4. (s – d)/(f + g)∗(h – j)∗(k + l) + (z + x + c + v)∗(b + n + m)
5. (x∗c∗v∗b + n/m/a/s + d)/(f∗(g + h) + (j – k)/ l)
6. (c + v + b + n/a)∗(s – d – f + r + t∗y + u∗k + p + z)
7. v – b – n + m∗a∗s∗d + (f∗g + h/j) + (k/l + q∗w)
8. (b + n + m + q)/(w + e – r∗t + y)∗(u + a – s – d) + f + g
9. (s – d)∗(f + g)∗(h – j)/(k – l) + (z/x/c/v/b + n∗m)
10. (r∗t∗y∗u + i/o + p – a – s)+(d – f)∗(g – h + j + k)
11. (d – f∗g + h)∗(j + k)/(r + t/y∗u + i∗o + p – a/s)
12. (a/(s – d) + f – g∗h + j∗k)∗(l∗z + b/n – x/(c – v))
13. (m + n)∗(b – v) + c + x + z/(l + k) + j/(h – g) + f∗(d + s)
14. (l – k – j – h∗g + f/d/s)/(a∗p∗o + i + u – y + t)
15. (p – o)∗(i + u) + (y + t – r + e∗(w – q) + l)/(k/j + m∗n)
16. (m + n∗b/v) + c∗x + z/(l + k) + j/(h – g) + f∗(d – s)
17. (l – k – j – h + g∗f∗d/s)/(a∗p∗o + i∗(u – y + t))
18. e + r – t/y∗(u∗i + o/p) + a∗s + d∗(f – g)∗(h + j)
19. (u∗y∗t + r – e – w)∗(q∗l + k/j∗h + g – f + d – s)
20. (h – g – f + d)∗s + (a + q – w/e + r/t + y∗z)∗(x + c)
21. (u∗y/t + r – e∗w)∗(q∗l – k/j/h + g) – (f + d)∗s
22. (f + d)/(s – a) + (e – r)∗(t + y)∗(u + p – m/n) + (b – v)∗c
23. a+b+c+d*q*w*e-t/r+u*i*o*p/h/j+k+l/n/m
24. (l – k –(j – h)∗g + f/d*s)*(a/p/o - i + u + y - t)
25. (s + d)/(f - g)/(h – j)+(k + l) + (z - x + c + v)∗(b + n)* m
26. (b+r/t+c/y+e/u + i∗o + s)+(d – f)∗(g + h)∗(j + k)
27. (a+s – d * f * g + h * j + k)/(l * z – x/(c + v) + b∗n)
28. z * x * c * v * b+(n + m/q + w+e + r/t∗y) – a + i
29. (s + d)*(f – g)∗(h + j)∗k*l + z* x + c + v∗(b * n + m)
30. (x∗c∗v∗b + n+m+a+s + d)/(f∗g + h + j – k/ l)
31. c * v + b * n/a*(s + d + f * r + t∗y + u∗k + p * z)
32. v * b + n + m/a∗s∗d + f∗g + h*j* k/l + q∗w
33. (b ∗n + m ∗ q)/(w ∗ e – r+t∗ y)∗u ∗ a – s – d ∗ f + g
34. (p + o)/ (i – u ∗ y + t ∗ r + e∗(w – q) ∗(k+j + m∗n)
35. (m∗n∗b/v) + c∗x + z/(l + k) + j/(h – g) + f∗(d – s)
36. (l ∗ k – j ∗ h + g∗f∗d/s)/(a∗p+o + i∗(u ∗ y + t))
37. e ∗ r – t/y + (u∗i – o∗p) + a / s + d∗(f – g)∗(h + j)
38. (u∗y∗t + r + e – w∗q) ∗l + k∗j+h ∗ (g – f + d ∗ s)
39. (h – g ∗ f + d)∗s + (a ∗ q – w/e + r∗ (t + y∗z)∗x + c)
40. (u∗y/t + r – e∗w)∗(q∗l – k∗j∗h + g∗ (f + d/s))

4. Краткое описание программы распараллеливания выражений

Для выполнения практического задания необходимо использовать программу для распараллеливания АВ.

Программа позволяет:

- редактировать АВ;
- визуализировать дерево параллельного вычисления АВ;
- масштабировать дерево вычисления АВ;
- осуществлять настройку на число вычислителей;
- оценить характеристики параллельности и численной устойчивости:
- выдавать на принтер результаты вычисления АВ.

На рис.7 приведен интерфейс программы при распараллеливании АВ.

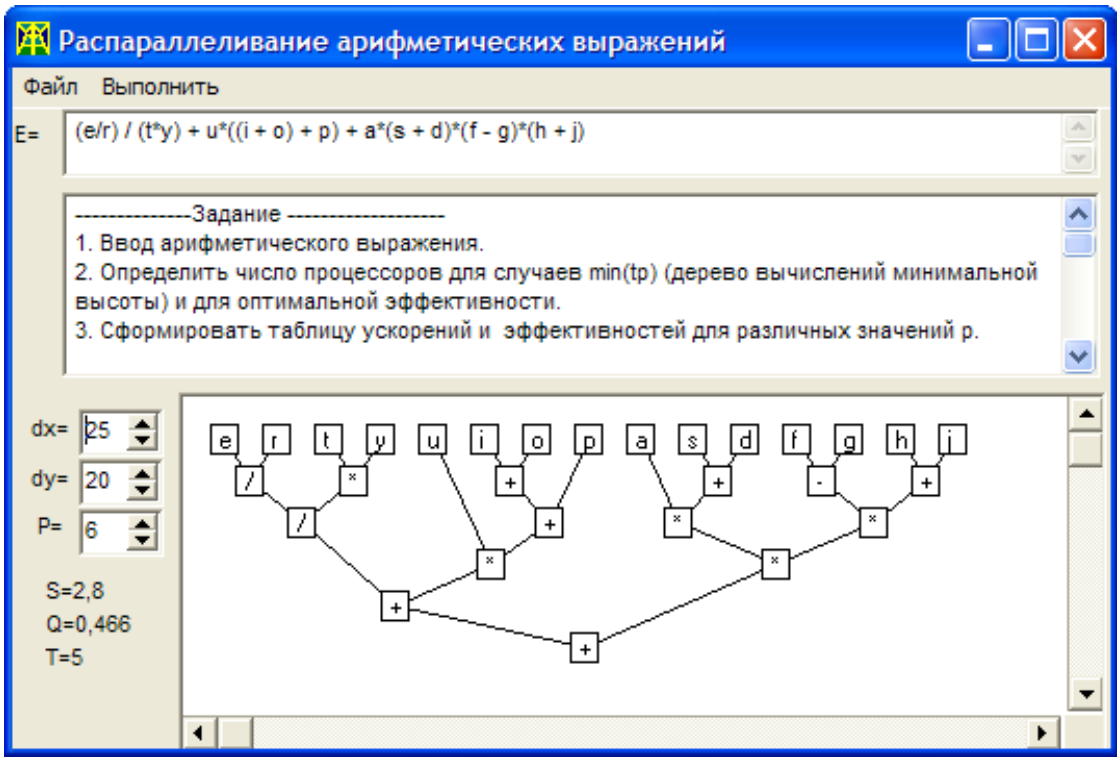

Рис.7. Пример работы программы

5. Контрольные вопросы

1. Какой принцип распараллеливания используется при распараллеливании арифметических выражений?
2. Почему арифметические выражения относят к классу задач с *неявным параллелизмом?*
3. В чем основной смысл леммы Брента?
4. Какие характеристики параллельности Вы знаете?
5. Чем характеризуется эффективность схемы при оптимальной загрузки процессоров?

6. К каким выражениям применяется схема сдваивания?

7. Какие выражения называют альтернированными?

8. На чем основана возможность распараллеливания рекурсивной схемы Горнера?

9. Какие основные свойства арифметических операций используются при преобразовании арифметических выражений в эквивалентные?

6. Список используемых источников

1. Системы параллельной обработки: Пер. с англ./ Под ред. Д. Ивенса. – М.: Мир, 1985. – 416 с.
2. Элементы параллельного программирования / Под ред. В. Е. Котова. – М.: Радио и связь, 1983. – 240 с.
3. Водяхо А.И., Горнец Н.Н., Пузанков Д.В. Высокопроизводительные системы обработки данных: Учеб. пособие для вузов. – М.: Высш. школа, 1997. – 304 с.
4. Математические модели и методы в параллельных процессах. /Под ред. Воеводина В.В., 1986.
5. J.L. Baer and D.P. Bovet. Compilation of arithmetic expressions for parallel computations. In Proc. IFIP Congress, pp 340-346, Amsterdam, 1968. “Компиляция *арифметических выражений для параллельных вычислений”*
6. R. P. Brent. *The parallel evaluation of arithmetic expressions in logarithmic time In Complexity of Sequential and Parallel Numerical Algorithms*, J. F. Traub, Ed, Academic Press, New York, pp. 83-102, 1973.
7. Richard P. Brent. *The Parallel Evaluation of General Arithmetic Expressions*. Journal of the ACM, v.21 n.2, pp. 201-206, 1974.
8. S. Winograd. *On the parallel evaluation of certain arithmetic expressions*. Journal of the ACM, v.22 n.4, pp. 477-492, 1975.

Лабораторная работа №2
Исследование функционирования многопроцессорной системы в однозадачном режиме

Лабораторные работы №№2 – 5 основаны на материалах пособия: И.И. Ладыгин, А.А. Крюков, Г.А.Калинина. Архитектура ЭВМ и систем: Лабораторный практикум. – М.:МГТУ "СТАНКИН", 2003. – 35 с.

Перед началом выполнения работ следует внимательно ознакомиться с информацией, представленной в ***Приложениях 1, 2 и 3.***

Цель работы: изучение принципов распределения узлов вычислительного процесса (ВП) в ***однозадачном режиме*** с использованием различных стратегий назначения готовых к исполнению узлов (подзадач).

1. Теоретическое введение

В этой работе исследуются принципы распределения узлов ВП в однозадачном режиме и анализируются временные диаграмм с целью определения времен простоя процессоров из-за неготовности узлов задач. Интервалы простоя на временных диаграммах, назовем их "пустотами", могут быть использованы для выполнения готовых копий узлов подзадач. Тем самым повышается вероятность обнаружения ошибок при решении задач в МВС, которая рассчитывается по формуле

$$\mathbf{P_{ош} = \Sigma\ T_{дубл} / T_{max}} \quad (*)$$

где $\mathbf{T_{дубл}}$ - суммарное время выполнения копий улов задач;

$\mathbf{T_{max}}$ - суммарное время выполнения всех узлов.

В процессе выполнения лабораторной работы необходимо построить временные диаграммы выполнения вычислительного процесса, проанализировать "пустоты" и определить оптимальное множество копий узлов для достижения максимальной вероятности обнаружения ошибки при заданном времени выполнения ВП.

Рассмотрим пример выполнения некоторых пунктов задания лабораторной работы относительно графа задачи, приведенного на рис.1. Для этого необходимо определить критический путь на графе и минимальное время выполнения графа задачи, которое соответствует значению критического пути. Алгоритм поиска критического пути в графе задачи рассмотрен в Приложении 2.

Для этого графа задачи T_{max}=48, T_{min}=23, следовательно, первое приближение числа процессоров МВС, с помощью которых можно выполнить задачу за минимальное время n=3 (ближайшее большее целое от значения T_{max}/T_{min}).

Примем в качестве стратегии назначения - выбор готового к исполнению узла ВП с максимальным временем выполнения. Временная диаграмма выполнения задачи, соответствующая данной стратегии, представлена на рис. 2. В случае нескольких свободных процессоров назначение готового к исполнению

узла осуществляется на процессор с меньшим номером. По вертикальной оси - 1,2,3 - номера процессоров, по горизонтальной - время, заданное в условных единицах. В интервалах занятости процессоров проставлены номера узлов задачи.

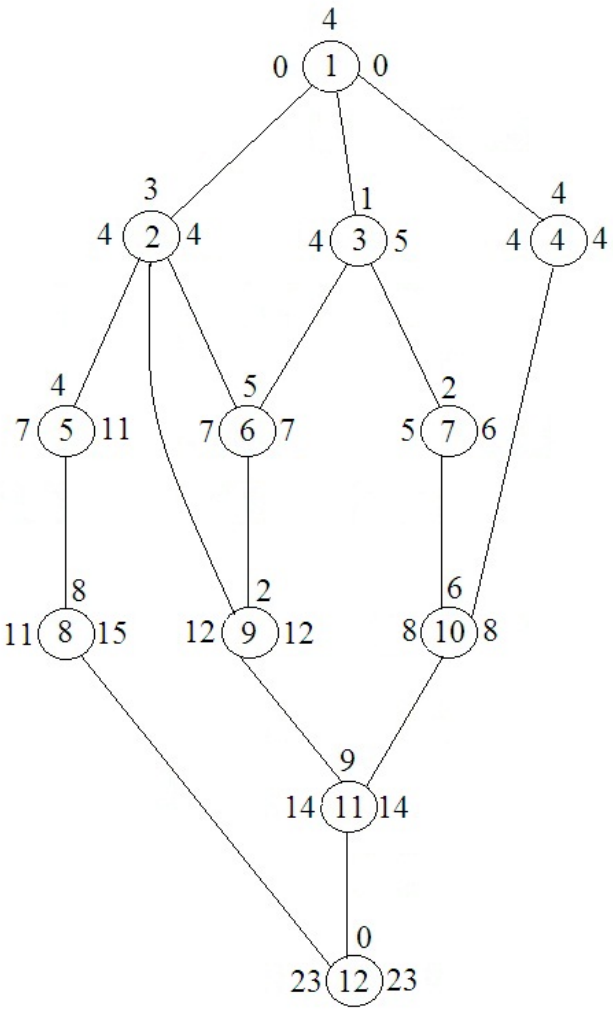

Рис. 1. Граф анализируемой задачи.

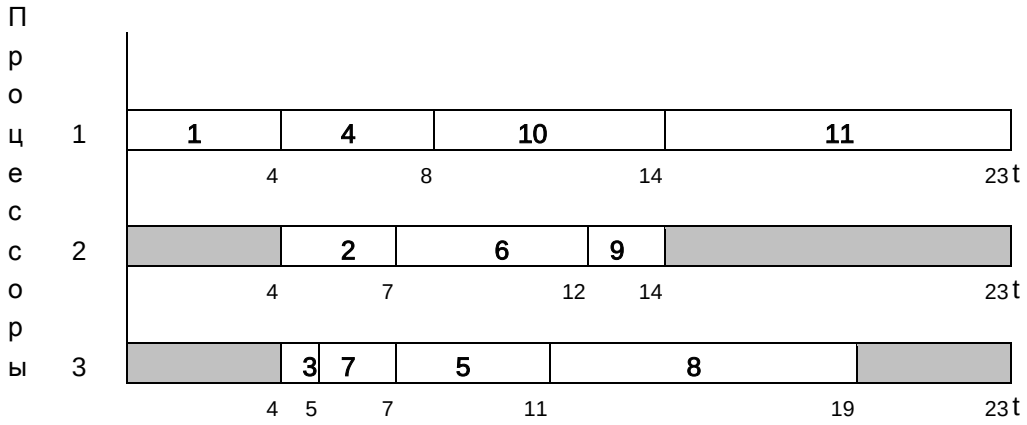

Рис. 2. Диаграмма выполнения задачи без учета времени передач

Анализируя временную диаграмму, можно сделать следующие выводы:

- Минимальное число процессоров, с помощью которых задача выполняется за T_{min}=23, составляет n_{min}=3, один из процессоров (первый) работает без простоев, два других заняты с 5 по 14 такт и с 5 по 19 такт соответственно.
- Коэффициенты загрузки процессоров соответственно равны **$K_{з1}$ = 1, $K_{з2}$ = 10/23 = 0,43, $K_{з3}$ = 15/23 = 0,65.**.
- Коэффициент ускорения (максимально возможный для данной задачи)

$$\mathbf{Ку= T_{max}/T_{min} = 48/23= 2,09}.$$

- Заштрихованные интервалы времени ("пустоты") для второго и третьего процессоров можно использовать для выполнения копий узлов ВП, готовых к исполнению на начало интервала простоя процессора. Следует обратить внимание на то, что при заполнении "пустот", с точки зрения контроля всех процессоров, целесообразно, чтобы копии одинаковых узлов выполнялись на разных процессорах. Соответствующая диаграмма представлена на рис. 3.

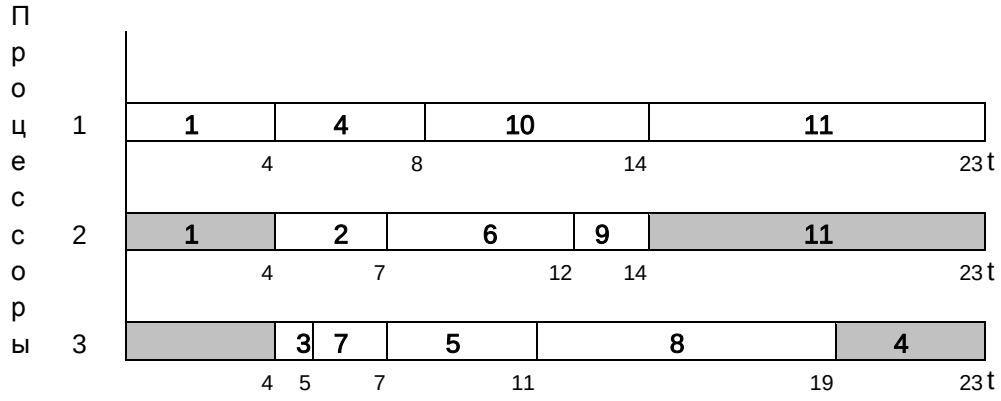

Рис. 3. Пример выполнения копий узлов в "пустотах" временной диаграммы

На основании результатов ее анализа определим вероятность обнаружения ошибки

$$\mathbf{P_{ош} = (4+9+4)\ /48 = 17/48=0,35}$$

2. Домашняя подготовка

1. Изучить соответствующий раздел лекционного курса и настоящее описание лабораторных работ. По номеру варианта из файла ***Графы №1.doc*** взять исходные данные для исследования - **граф задачи №1** для задания 1, а из файла ***Графы №2.doc*** – **граф задачи №2** для задания 2.

2. На заданных графах с использованием алгоритма определения критического пути (см. Приложение 2) рассчитать следующие характеристики:

2.1. Критические пути.

2.2. Временные характеристики – максимальное T_{max} и минимальное T_{min} время выполнения графа задачи.

2.3. Оценить требуемое число процессоров.

3. Лабораторное задание

Часть 1. Исследование графа задачи №1

1. Промоделировать с помощью программы ***Lab_Work2.exe*** исследуемый **граф задачи №1** с использованием различных стратегий выбора готовых узлов и разного числа используемых процессоров для достижения минимально возможного времени выполнения графа задачи.
2. Результаты моделирования для ***различных стратегий выбора готовых*** узлов свести в таблицу (стратегия, количество процессоров, время решения задачи t(p), коэффициент средней загрузки процессоров, ускорение). По результатам рассчитать достигаемое ускорение.
3. По результатам моделирования построить зависимости:
   a. времени решения **t(p)**,
   b. ускорения $\mathbf{T_{max}/t(p)}$,
   c. средней загрузки процессоров (эффективности)

   для различных стратегий выбора готовых узлов, где **t(p)** – время выполнения вычислительного процесса, **p** - количество процессоров, на которых выполняется ВП.
4. Проанализировать полученные графики, **выбрать наилучшую стратегию** для решения данной задачи и объяснить результаты.

Часть 2. Исследование графа задачи №2

5. Повторить **п.1** для **графа задачи №2** для
   - достижения минимального времени решения задачи $\mathbf{T_{min}}$
   - заданного времени выполнения задачи, определенного как

   $\mathbf{T_{зад} = T_{min} + 4.}$
6. При заданном времени выполнения задачи $\mathbf{T_{зад}}$ и определенном количестве процессоров достичь максимальной вероятности обнаружения ошибки $\mathbf{P_{ош}}$, выполняя копии узлов в "пустотах" на временной диаграмме.

Для этого распределить узлы ВП для **графа задачи №2** в режиме максимального заполнения "пустот" временной диаграммы ***выполнением копий узлов ВП вручную.***

7. Проверить правильность выполнения ВП для случая достижения максимальной вероятности обнаружения ошибки с помощью программного моделирования.

Для этого необходимо изменить **граф задачи №2** и внести соответствующие изменения в описание графа с помощью матрицы смежности.

8. Сравнить результаты ручного и программного вариантов повышения вероятности обнаружения ошибок.
9. Определить временные и ресурсные издержки (во сколько раз необходимо увеличить допустимое время решения или число используемых процессоров) для случая достижения $\mathbf{P_{ош} =1.}$ Проанализировать и объяснить полученные результаты.

4. Методические рекомендации

Для выполнения п.6 перерисовать временные диаграммы выполнения ВП, полученные в п.5. для случая выполнения ВП за заданное время $\mathbf{T_{зад}}$ на минимальном количестве процессоров.

Выбор копий узлов для заполнения «пустот» производить исходя из условий: $P_{ош} = \Sigma\ T_{дубл}/T_{max} \rightarrow max$, где $\Sigma T_{дубл}$ – суммарное время выполнения копий узлов; T_{max} – суммарное время выполнения всех узлов ВП.

Для программного моделирования ВП с учетом копий узлов необходимо произвести изменение исходной матрицы смежности, которая служит формальной моделью графа в используемых программных моделях. Для этого необходимо добавить в матрицу смежности число дополнительных строк и столбцов, равное числу копий узлов ВП, правильно расставить связи между узлами.

Для достижения $P_{ош} = 1$ необходимо продублировать все узлы ВП, т.е. выполнить параллельно две одинаковые задачи в многозадачном режиме. Для этой цели необходимо использовать программу ***Lab_Work3.exe,*** моделирующую функционирование МВС в многозадачном режиме.

5. Требования

Отчет по лабораторной работе может быть выполнен в электронной форме и должен содержать:

1. результаты домашней подготовки;
2. таблицы с результатами моделирования и графики по пп.1, 2, 3 и 5 задания;
3. не избыточный и избыточный (с копиями узлов) графы задачи, временные диаграммы с указанием копий узлов и количественные оценки по пп.6 - 9 лабораторного задания;
4. выводы по всем полученным результатам.

6. Контрольные вопросы

1. Что понимается под решением задачи назначения?
2. Какие критерии оптимальности распределения узлов ВП применяются при решении задачи назначения?
3. Как определить критический путь для графа задачи без учета передач? В каком случае узел является критическим?
4. Что влияет на выбор стратегии назначения готовых к выполнению узлов ВП? Какие стратегии выбора чаще всего используются?
5. Как повысить вероятность обнаружения ошибок в ВП? Как можно достичь $P_{ош}=1$?
6. Чему равна максимальная вероятность обнаружения ошибок при выполнении задачи за заданное время на определенном количестве процессоров? Всегда ли можно достичь этого значения?
7. Показать, что коэффициент средней загрузки процессоров определяет эффективность выполнения ВП?

Лабораторная работа № 3

Организация многозадачного режима выполнения вычислительного процесса в МВС

Цель работы: исследование задачи составления расписания выполнения ВП в многозадачном режиме в МВС с использованием различных стратегий назначения готовых узлов, с учетом приоритета выполняемой задачи и без учета приоритета; изучение методики выбора характеристик МВС (количество процессоров, стратегия назначения, приоритетность задачи) для достижения заданного времени выполнения набора задач.

1. Теоретическое введение

Во данной лабораторной работе в отличие от предыдущей изучаются принципы распределения узлов ВП **в многозадачном режиме с помощью** программы ***Lab_Work3.exe***.. Так же, как и в однозадачном режиме готовый к исполнению узел ВП, выбранный с учетом приоритета задачи и стратегии назначения, начинает выполняться на свободном процессоре без временных задержек независимо от того, какой задаче он принадлежит. В многозадачном режиме ***максимальное время выполнения набора задач равно сумме времен выполнения всех узлов всех задач набора. Минимальное время*** выполнения набора задач ***равно максимальному значению из минимальных времен выполнения каждой задачи.*** Следует обратить внимание на то, что в многозадачном режиме с учетом приоритетов среди готовых к исполнению узлов ВП формируются отдельные очереди на выполнение в соответствие с приоритетами. Первыми назначаются на выполнение узлы из очереди с наивысшим приоритетом, в соответствие с выбранной стратегией. Если эта очередь пуста, то анализируется очередь с меньшим приоритетом и т.д.

2. Домашняя подготовка

1. Изучить соответствующий раздел лекционного курса и настоящее описание лабораторных работ. Исходными данными для исследования является набор из двух задач: **граф задачи №1** и **граф задачи №2 из лабораторной работы №2.**

2. Для заданного набора задач рассчитать минимальное и максимальное время выполнения.

3. Лабораторное задание

1. В **однозадачном режиме** промоделировать отдельно выполнение каждой задачи с использованием всех допустимых стратегии выбора готовых узлов на заданном количестве процессоров.

Результаты моделирования для каждой задачи при различных стратегиях свести в таблицу (стратегия, количество процессоров, время решения задачи, ускорение, коэффициент средней загрузки процессоров).

Определить наиболее оптимальную стратегию или стратегии для каждой задачи (графа) с точки зрения минимального времени решения[1].

Определить суммарное время выполнения двух задач в однозадачном режиме.

2. В **многозадачном режиме** промоделировать выполнение набора из двух задач с **равными приоритетами** в МВС на том же количестве процессоров и с теми же стратегиями назначения, что и в п.1**.** Определить время выполнения набора задач, ускорение и коэффициенты загрузки процессоров.

Результаты моделирования для различных стратегиях свести в таблицу (стратегия, количество процессоров, время решения задач, ускорение, коэффициент средней загрузки процессоров). **Определить наиболее оптимальную стратегию или стратегии для набора задач с точки зрения минимального времени решения.**

3. Сравнить ***графически*** **однозадачный и многозадачный режимы** функционирования МВС по результатам выполнения п.1 и п.2. (по *времени выполнения*, по достигаемому *ускорению* и *загрузке процессоров*). Проанализировать и объяснить полученные результаты.

4. На основании результатов п.2. выбрать ***наилучшую стратегию назначения*** и для этой ***одной стратегии*** выполнить набор задач в **многозадачном режиме для различных вариантов задания приоритетов.** Выбрать наилучшее соотношение приоритетов выполняемых задач.

5. Определить характеристики МВС, обеспечивающие достижение заданного времени выполнения набора задач при минимальных аппаратных затратах (числе процессоров ***p***) $\mathbf{T_{зад} = T_{min} + 4.}$

Варьируемыми параметрами при выполнении этого пункта являются количество процессоров и стратегии назначения готовых к выполнению узлов.

Проанализировать различные варианты изменения характеристик МВС при достижении $T_{зад}$ также в зависимости от приоритета задач.

Исследовать зависимости **коэффициента ускорения** выполнения набора задач и **коэффициента загрузки** процессоров от количества процессоров при различных стратегиях назначения готовых узлов и различных приоритетах задач.

Проанализировать полученные результаты, выбрать наилучший способ организации вычислительного процесса для $\mathbf{T_{зад}}$.

---

[1] При выполнении этого пункта можно воспользоваться результатами из лабораторной работы №2, дополнив их результатами моделирования с учетом новых стратегий

4.Требования

Отчет по лабораторной работе может быть выполнен в электронной форме и должен содержать:

1. результаты домашней подготовки;
2. таблицы результатов моделирования ВП;
3. графики сравнения характеристик однозадачного и многозадачного режимов при задании различных приоритетов;
4. выводы по результатам.

5. Контрольные вопросы

1. Как определяются максимальное и минимальное время выполнения набора задач?

2. Объяснить зависимость коэффициента ускорения от количества процессоров в однозадачном и многозадачном режимах при моделировании выполнения набора задач.

3. Как влияет учет приоритета задач в наборе на выбор стратегии назначения готовых к выполнению узлов ВП?

4. Объяснить изменения коэффициентов загрузки процессоров при выполнении набора задач и каждой задачи в отдельности.

5. Как определить минимальные аппаратные затраты, обеспечивающие достижение заданного времени выполнения набора задач?

Лабораторная работа № 4

Исследование принципов организации вычислительного процесса в МВС с общей памятью

Цель работы: изучение способов организации ВП при выполнении набора задач различных типов на МВС с общей памятью и шинной организацией коммутации.

Результатом является определение параметров МВС (количество процессоров, шин, стратегия выбора узлов подзадач на выполнение), позволяющих выполнить набор задач конкретного типа за заданное время.

1. Теоретическое введение

На рис.1а приведена общая структура МВС с общей памятью.

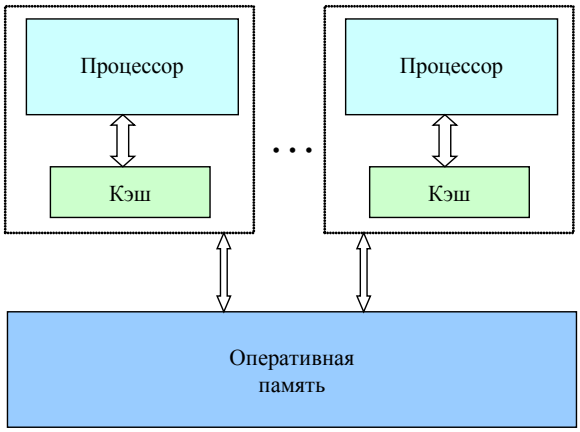

Рис. 1а. Общая структура МВС с общей памятью

Для организации взаимодействия процессорных узлов обычно используется коммутационная сеть в виде набора шин как показано на рис.1б.

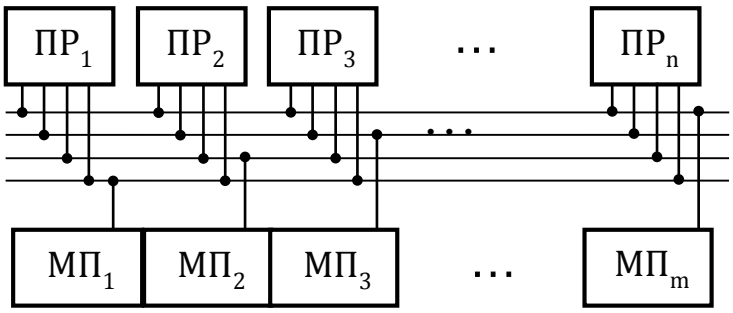

Рис. 1б. МВС с общей памятью и шинной организации коммуникаций

Отметим некоторые важные особенности процесса моделирования, которые необходимо учитывать при выполнении работы.

1. МВС состоит из ***n*** процессорных узлов $ПР_i$, i=1 – n, ***m*** шин, связывающих каждый процессор с каждым из модулей памяти $МП_i$. i=1 – m.

2. Процессор, выполнив очередной узел ВП, может "захватить" любую свободную шину и передать все результаты обработки в любой один

свободный с точки зрения процесса «записи/чтения» модуль памяти (емкость модуля памяти считается неограниченной).

3. Процессор, на который назначен для выполнения узел ВП, может "захватить" любую свободную шину, и если модуль памяти, в котором находятся исходные данные для этого узла, не занят «записью/чтением» данных другим процессором, то происходит их считывание.

4. При реализации стратегии назначения готовых узлов задачи на выполнение учитываются только времена выполнения узлов задачи.

Рассмотрим пример моделирования процессов вычислений в МВС с общей памятью при решении задачи, граф которой представлен на рис. 2.

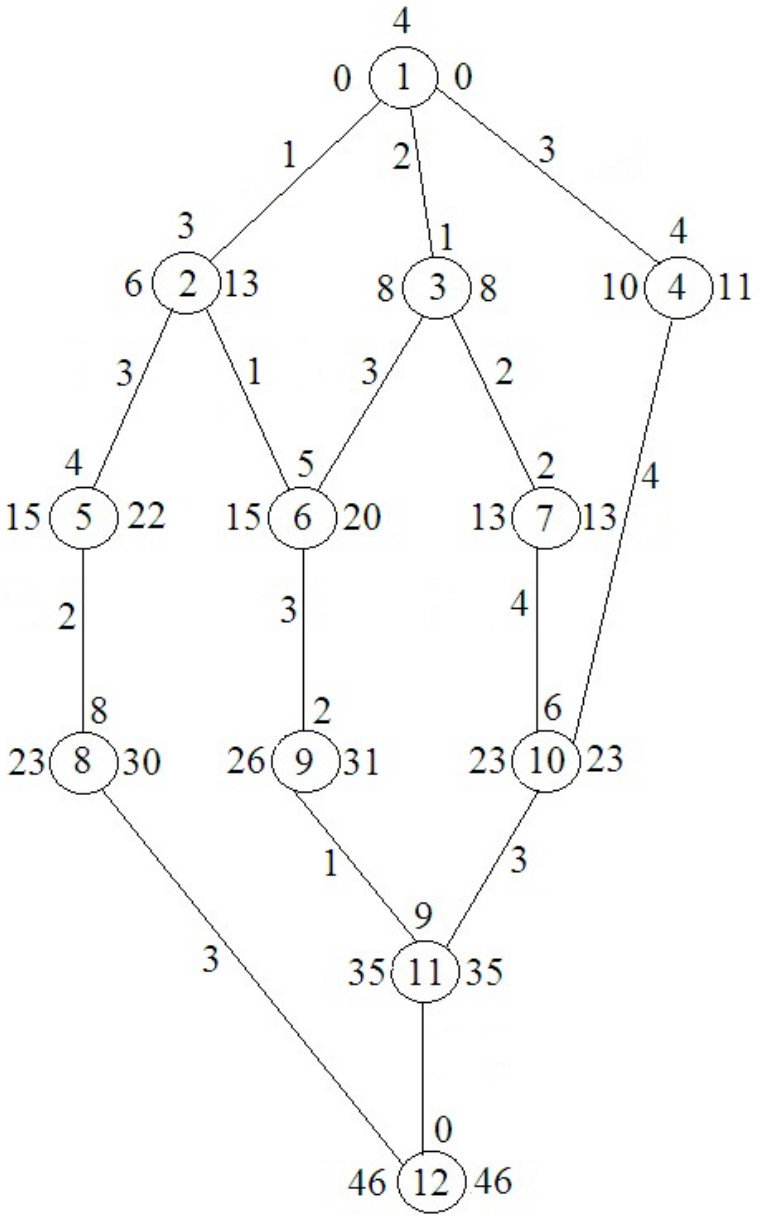

Рис. 2. Определение критического пути для графа задачи с учетом передач на МВС с общей памятью

Во-первых, необходимо определить критический путь на данном графе **среднесвязной задачи** из вершины 1 в вершину 12 **с учетом передачи данных** между подзадачами. Дуги графа задачи взвешены числами, соответствующими временам занятости шины при передачи данных между узлами графа.

Поиск критического пути на графе, а, следовательно, и минимального времени его выполнения с учетом времени передач данных от узла к **узлу,** в общем случае, сводится к задаче полного перебора. Тем не менее, вводя ряд ограничений на условие передачи данных, а они связаны с организацией обменов между процессорами и памятью, можно предложить алгоритмы, сокращающие полный перебор и дающие искомый результат.

Сделаем следующие допущения.

- Граф задачи реализуется на системе с неограниченными ресурсами.
- Каждый узел графа может передавать данные по всем выходящим ветвям параллельно и принимать данные по всем входящим ветвям параллельно.

При этих условиях T_{min} определяется в зависимости от способа организации памяти, поскольку этим, в частности, определяются времена передачи между узлами.

- Для случая МВС с общей памятью ***времена передачи данных удваиваются.***

Для графа, изображенного на рис. 2 критический путь с учетом передач **-1; 3; 7; 10; 11; 12**, а $\mathbf{T_{tin}}$ = **46**.

Проанализируем временную диаграмму, приведенную на рис. 3, построенную как результат моделирования процессов выполнения заданного графа в МВС с общей памятью при **n=3**, **m=2**.

На диаграмме прямоугольники, помеченные одной цифрой, - это интервалы времени обработки соответствующего узла задачи процессором, двумя цифрами - интервалы времени, соответствующие передаче данных от одного узла задачи (верхняя цифра) к другому (нижняя цифра) по тракту «процессор - шина – память». Время выполнения задачи на МВС с общей памятью при **n=3**, **m=2** равно 61 единицам времени.

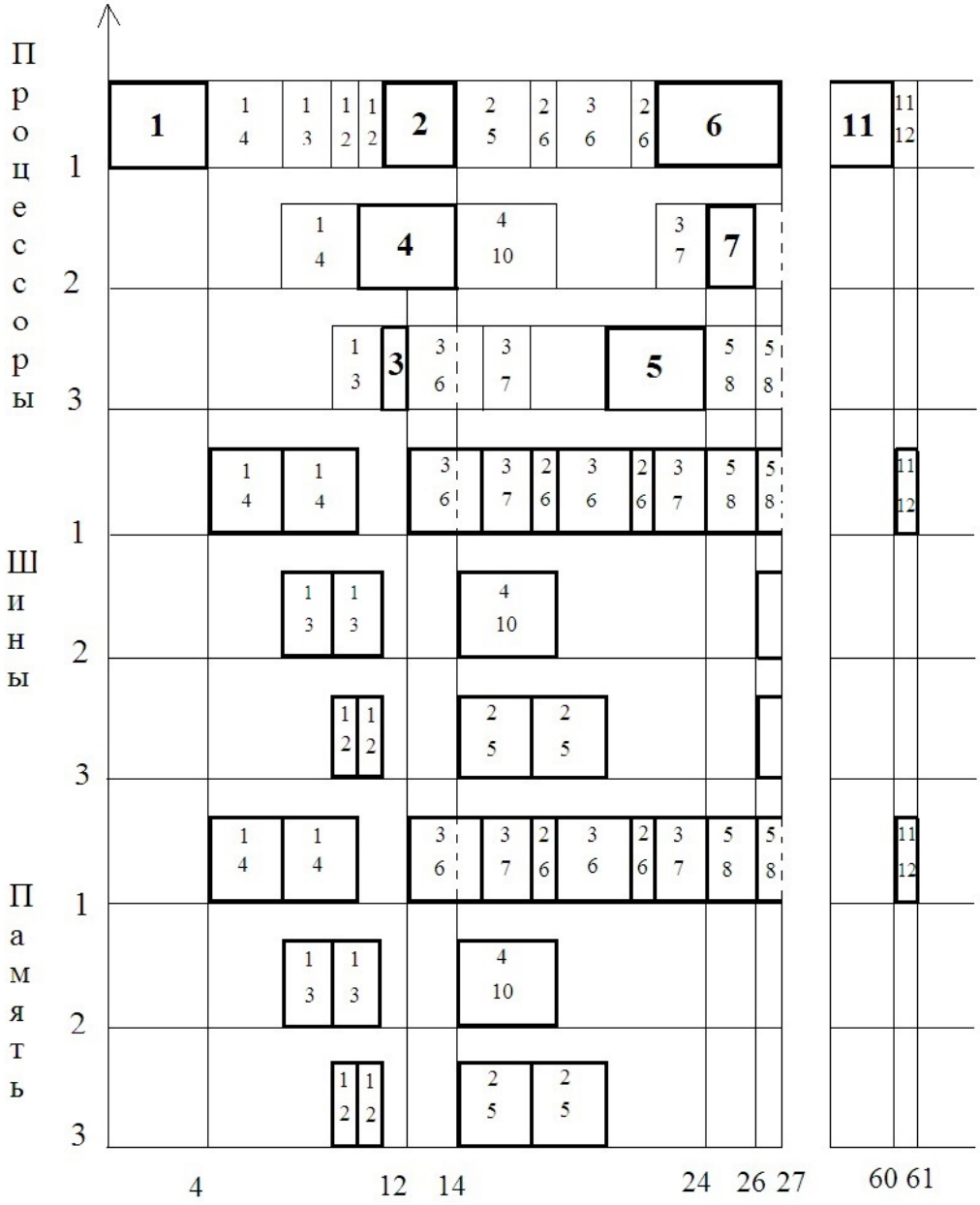

Рис. 3. Диаграмма выполнения задачи в МВС с общей памятью

После завершения выполнения 1-го узла графа задачи на 1-ом процессоре необходимо передать данные для выполнения 2-го, 3-его и 4-го узлов (подзадач), записав их в многомодульную память. Процессор 1-ый последовательно, захватывая 1-ую, 2-ую и 3-ю шины, передает данные сначала для 4-го, затем 3-его и в завершении для 2-го узлов задачи. Данные по шинам передаются в соответствующие модули оперативной памяти, где сохраняются до моментов востребования. После завершения «записи» данных в память 1-ый процессор начинает «чтение» данных из памяти для последующего выполнения 2-го узла. На 2-ой процессор назначен для выполнения 4-ый узел, поэтому, как только завершится «запись» в память данных для этого узла, 2-ой процессор начинает «чтение» этих данных. Аналогично, на 3-ий процессор назначен для выполнения 3-ый узел, поэтому, как только завершится «запись»

в память данных для этого узла, 3-ий процессор начинает «чтение» этих данных и т.д.

Рис. 3 достаточно подробно иллюстрирует процесс выполнения графа задачи в МВС с общей памятью при **n=3, m=2.** Поскольку состояния шин и модулей памяти одинаковы, то в лабораторных работах при моделировании визуализируются только состояния шин.

2. Домашняя подготовка

1. Изучить соответствующий раздел лекционного курса и настоящее описание лабораторных работ. В качестве исходных данных для выполнения домашней подготовки взять граф, представленный в текстовом файле, из папки **Графы для однозадачного режима** в соответствии с **Таблицей номеров графов** (см. Приложение 3). Для выбранного графа указаны времена выполнения узлов и передач по шине между узлами.

2. Для заданного графа самостоятельно рассчитать критический путь и минимальное время выполнения задачи с учетом времени передач по шине, сравнить полученный критический путь с тем же графом, но без учета времени передач данных.

3. Выбрать значения параметров структуры МВС с общей памятью, на которой возможно выполнение задачи за минимальное время.

3.Лабораторное задание

Часть 1. Однозадачный режим.

1. Проверить выбранный в п.3 домашней подготовки вариант структуры МВС на программной модели ***Lab_Work4.exe***. Проанализировать и объяснить полученные результаты.

Часть 2. Многозадачный режим.

Для выполнения дальнейших пунктов лабораторного задания в соответствии с вариантом из **Таблицы 2 номеров графов** (см. Приложение 3) выбрать **три различных** наборы из **двух однотипных задач:**

- слабосвязанные задачи (из папки **Слабосвязные графы**), в которых время выполнения узлов задачи много больше времени передач между узлами $t_p \gg t_n$;
- среднесвязанные задачи (из папки **Среднесвязные графы**), в которых $t_p \approx t_n$;
- сильносвязанные задачи (из папки **Сильносвязные графы**), в которых $t_p \ll t_n$.

1. Промоделировать выбранные наборы задач ***при различных стратегиях назначения*** готовых узлов, ***разном числе*** используемых ***процессоров*** и ***модулей памяти***. Определить время выполнения задач, ускорение и среднюю загрузку процессоров. Результаты свести в таблицы. *Выявить параметры архитектуры МВС, при которых решение является наилучшим.*
2. Проанализировать графически зависимости времени решения задач, ускорение, эффективности от числа процессоров и числа шин (модулей памяти) для набора задач каждого типа для ***одной наилучшей стратегии***. Сравнить и объяснить результаты.
3. Определить и ***изобразить структуру*** МВС, позволяющую выполнить набор задач каждого типа за заданное время

$$\mathbf{T_{зад} = 1{,}33\ T_{min}}$$

4. Проанализировать полученные результаты и объяснить их.

4.Требования

Отчет по лабораторной работе может быть выполнен в электронной форме и должен содержать:

1. результаты домашней подготовки;
2. таблицы результатов моделирования ВП;
3. необходимые графики для анализа полученных данных моделирования и соответствующие выводы;
4. структурные схемы по п. 3 лабораторного задания.

5.Контрольные вопросы

1. Задачи каких типов исследуются в данной лабораторной работе?

2. Как определить критический путь и минимальное время выполнения задачи с учетом времени передач на МВС с общей памятью, какая структура МВС этому соответствует? Всегда ли этот критический путь совпадает с критическим путем задачи без учета передач?

3. Изменение какого параметра (число процессоров, шин или модулей памяти) является наиболее существенным для уменьшения времени выполнения задач различных типов? Как это можно объяснить на основании анализа зависимостей, полученных в п.3 лабораторного задания?

4. Как можно объяснить увеличение в некоторых случаях времени выполнения набора задач при увеличении количества процессоров? На каких типах задач это наиболее наглядно? Какие значения остальных параметров МВС этому соответствуют?

5. Как определить значения параметров МВС (число процессоров, шин или модулей памяти), позволяющих выполнить набор задач за заданное время при наименьшем количестве шагов моделирования?

Лабораторная работа № 5

Исследование принципов организации вычислительного процесса в МВС с распределенной памятью

Цель работы: изучение способов организации вычислительного процесса в МВС с распределенной памятью для определения параметров МВС (количество процессоров и шин), позволяющих выполнить задачу за заданное время; сравнение полученных характеристик с аналогичными характеристиками для МВС с общей памятью.

1.Теоретическое введение

На рис. 1 приведена общая структура МВС с физически распределенной памятью. В качестве коммуникационной сети в лабораторных работах используются ***шины*.**

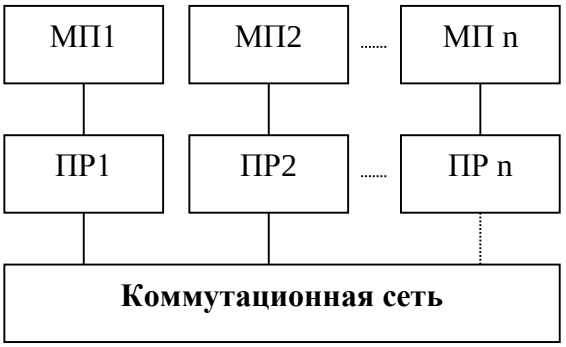

Рис. 1. МВС с распределенной памятью

Рассмотрим МВС, изображенную на рис. 1, со следующими характеристиками. Каждый из **n** процессоров ПРi имеет свою локальную память МПi, емкостью достаточной для хранения промежуточных результатов обработки любого узла задачи. Каждый процессор имеет доступ к памяти любого другого процессора по считыванию из нее данных в свою память по одной из m шин, при этом $\mathbf{m \leq n/2}$. Одновременно запись в память или считывания из нее осуществляется только по одному каналу. Рассмотрим пример моделирования процессов вычислений в МВС с распределенной памятью при решении задачи, граф которой представлен на рис. 2.

Во-первых, необходимо определить критический путь (КП) на данном графе **среднесвязной задачи** из вершины 1 в вершину 12 **с учетом передач.** Дуги графа задачи взвешены числами, соответствующими временам занятости шины при передачи данных между узлами графа.

Рассматриваемая задача аналогична задаче, граф которой приведен на рис.1 из лабораторной работы 4. Следует обратить внимание на то, что целью вычисления КП при реализации задачи на МВС с распределенной памятью является определение среди узлов – последователей узла, который назначается на процессор, выполняющий узел-предшественник. В этом случае данные от узла-предшественника не передаются узлу - последователю и время передачи

данных может быть приравнено к нулю (т.к. время доступа к своей оперативной памяти существенно меньше времени доступа к удаленной памяти).

Следующие допущения являются основными при реализации графа задачи на МВС с распределенной памятью.

- Граф реализуется на системе с неограниченными ресурсами.
- От каждого узла графа данные могут передаваться параллельно по исходящим ветвям.
- Каждый узел графа может принимать данные параллельно по входящим ветвям.
- Каждый узел графа может иметь не больше одной входящей и не больше одной исходящей ветви, время передачи по которой в результате поиска КП может быть равное нулю.

Алгоритм определения КП для графа задачи, выполняемой на МВС с распределенной памятью, состоит в том, что при вычислении минимально возможного времени начала выполнения i-го узла $T_{min\ i}$ определяется одна из входящих в него ветвей, времени передачи по которой присваивается значение равное нулю. При этом это должна быть такая ветвь, которая вносит максимальный эффект в уменьшении $T_{min\ i}$. Очевидно, что не всегда с первой же попытки можно определить $T_{min\ i}$. Например, определив ветвь di, входящую в i-й узел, обуславливающую $T_{min\ i}$ необходимо провести анализ узла d, из которого исходит эта ветвь. Если у этого узла уже есть одна исходящая ветвь, которой присвоено значение $T_{dk}=0$ (т.е. определен k-й узел - последователь d-го узла, и, следовательно, процессор оставил необходимые данные для выполнения к-го узла в своей локальной памяти), то ветвь d_i из рассмотрения исключается. Находится следующая по значению эффективности ветвь, входящая в i-й узел и т.д.

Если у d-го узла нет исходящей ветви, которой присвоено значение равное нулю, то для этого d-го узла определяется присутствие такой ветви, у которой $\tau_{dk} > \tau_{di}$. Если такая ветвь есть, то i-й узел исключается из рассмотрения до тех пор, пока не будет принято решение о присвоении или не присвоении ветви dk значения $T_{dk}=0$. Если такой ветви нет, то T_{di} присваивается значение, равное нулю, и уже с учетом этого определяется $T_{min\ i}$.

Отметим следующее. Если k-й узел расположен на более низком ярусе графа, чем i-й узел, то при нахождении $T_{min\ k}$ анализируются все пути от вершины d к вершине к. Значение 0 может быть присвоено (или не присвоено) в соответствии с правилами, изложенными выше для узла i, одной из ветвей, исходящих из узла d (необязательно для ветви dk).

Максимально возможное время начала выполнения узла $T_{max\ i}$ вычисляется при обратном движении по графу уже с учетом определенных выше нулевых ветвей. Данные ветви помечены на рис. 2 стрелкой (например, $3 \rightarrow 0$). В приведенном примере граф задачи имеет единственный критический путь: **1-4-10-11--12**, длина которого равна $\mathbf{T_{min}=27}$.

Длина критического пути (T_{min}) является основным параметром, характеризующим информационно-логическую структуру вычислительного процесса, реализующего решение заданной прикладной задачи. В случае учета времени передачи данных от одного узла к другому при решении задачи в МВС с ограниченными ресурсами, значение минимального времени увеличивается и определяется характеристиками реальной структуры МВС. Очевидно, что это увеличение будет тем меньше, чем удачнее распределены узлы графа по процессорам МВС.

Таким образом, возникает задача построения оптимального расписания (назначения готовых к исполнению узлов ВП на свободные процессоры). Подробно задача назначения рассматривается в приложении 2.

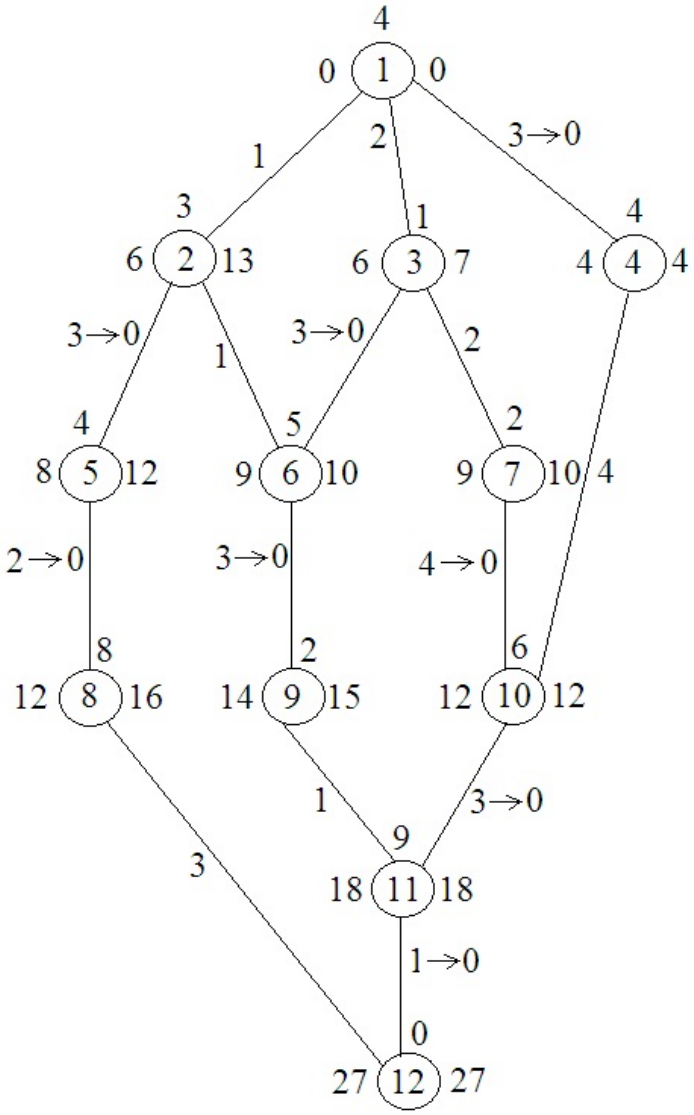

Рис. 2. Определение критического пути для графа задачи на МВС с распределенной памятью

Теперь проанализируем представленную на рис. 3 временную диаграмму выполнения графа задачи на МВС при **n =3, m=1**, полученную в результате моделирования ВП при использовании стратегии назначения готового к исполнению узла с *максимальным временем выполнения*.

После завершения выполнения 1-го узла задачи 1-ый процессор в своей локальной памяти оставляет данные для узла 4, а для узлов 2 и 3 данные передает по единственной шине последовательно из своей локальной памяти в локальную память 2-го и 3-его процессоров соответственно. Как только передача завершается, 1-ый процессор приступает к выполнению 4-го узла.

Однако в данном примере на процессор, выполнивший очередной узел, может быть назначен узел, следующий за ним и не нуждающийся в приеме данных от других узлов. Так, второй процессор за 5-м узлом выполняет 8-й, хотя 1-ый процессор закончил выполнение 4-го узла ранее и в тот момент готовым к исполнению был только узел 7, поэтому он был назначен на 1-й процессор.

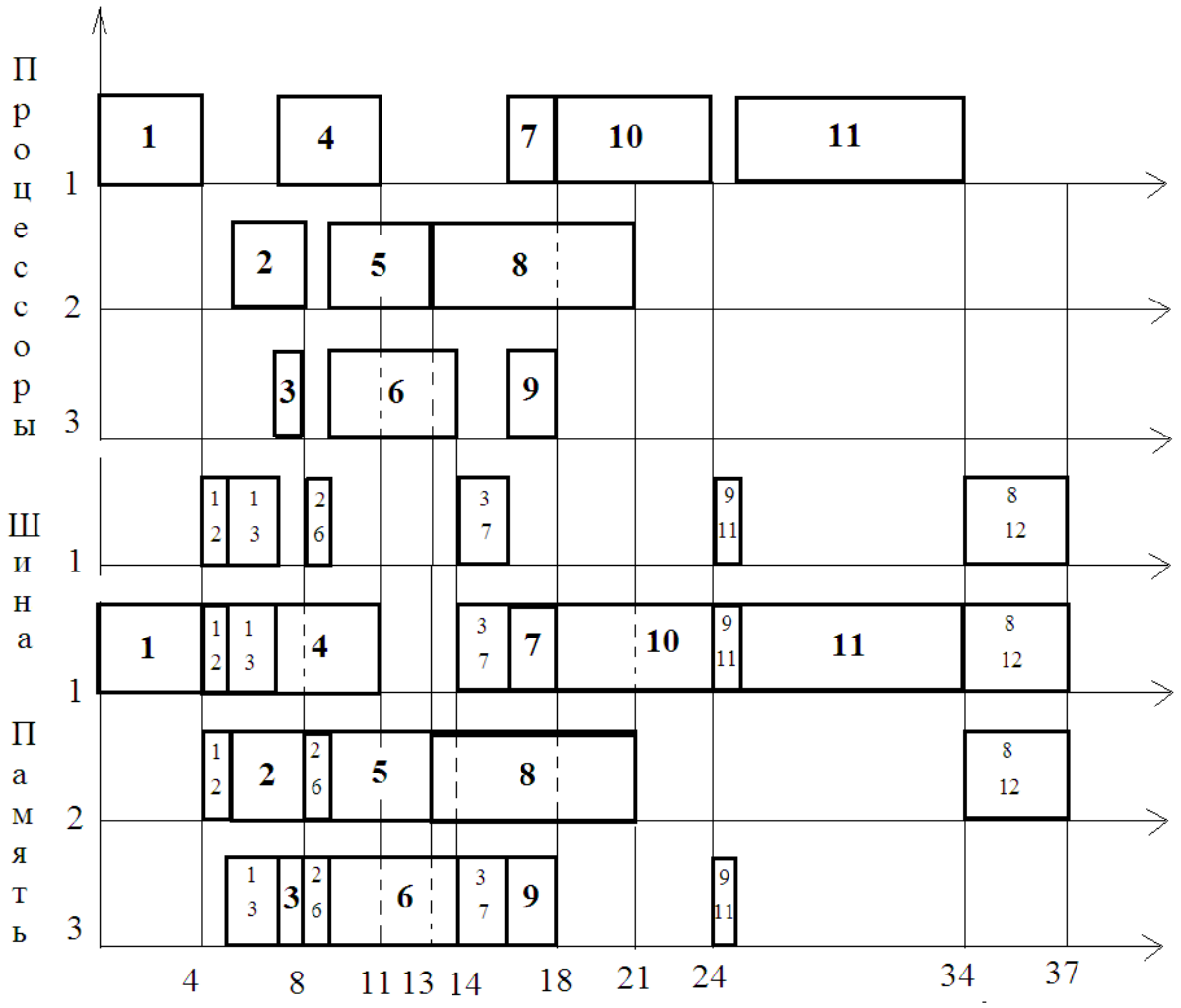

Рис.3. Диаграмма выполнения задачи в МВС с распределенной памятью

2.Домашняя подготовка

1. Изучить соответствующий раздел лекционного курса и настоящее описание лабораторных работ. В качестве исходных данных для выполнения домашней

подготовки взять тот же граф, что исследовался в лабораторной работе №4, подготовленный в текстовом файле, из папки **Графы для однозадачного режима** в соответствии с **Таблицей номеров графов** из Приложения 3. Для выбранного графа указаны времена выполнения узлов и передач по шине между узлами.

2. Для заданного графа с использованием алгоритма, приведенного выше, рассчитать минимальное время выполнения задачи в МВС с распределенной памятью. Сравнить полученное значение с минимальным временем выполнения задачи в МВС с общей памятью.

3. Предложить значения параметров структуры МВС с распределенной памятью, на которой возможно выполнение задачи за минимальное время.

3.Лабораторное задание

Часть 1. Однозадачный режим

1. Проверить выбранный при домашней подготовке вариант на программной модели ***Lab_Work5.exe***, проанализировать и объяснить полученные результаты.

2. Построить зависимости **времени решения задачи** от числа процессоров и числа шин при решении заданной задачи в МВС с распределенной памятью. Выявить параметры, дающие наиболее существенный выигрыш во времени решения.

3. Построить (или взять результаты из лабораторной работы №3) аналогичные зависимости для МВС с общей памятью (для сравнения необходимо в п.2 и 3 рассматривать МВС с одинаковыми параметрами).

4. Определить коэффициенты ускорения времени выполнения задачи в МВС с различной организацией; коэффициенты загрузки процессоров и шин в МВС с распределенной памятью и сравнить их с коэффициентами загрузки процессоров и шин в МВС с общей памятью. Проанализировать и объяснить полученные результаты.

Часть 2. Многозадачный режим.

5. Исследовать функционирование МВС в **многозадачном режиме** на **трех различных наборах однотипных задач** из лабораторной работы № 4:

- определить время решения, ускорение, загрузку процессоров в зависимости от числа процессоров и шин. **Построить графически зависимости времени** решения задач, **ускорения** и **средней загруженности** процессоров от числа процессоров, числа шин (модулей памяти) для набора задач каждого типа;
- сделать **выводы из полученных графиков** (определить изменение каких параметров наиболее существенно влияет на процессы решения задач в МВС с распределенной памятью);

- определить какие наборы наиболее эффективно решаются в данной архитектуре МВС.

6. Подобрать путем моделирования оптимальные характеристики МВС для решения каждого набора и изобразить архитектуры соответствующих МВС.

4.Требования

Отчет по лабораторной работе может быть выполнен в электронной форме и должен содержать:

1. результаты домашней подготовки;
2. таблицы результатов моделирования ВП;
3. необходимые графики для анализа полученных данных моделирования и соответствующие выводы;
4. структурные схемы МВС по п. 6 лабораторного задания.

5. Контрольные вопросы

1. Как определить критический путь и минимальное время выполнения задачи с учетом времени передач на МВС с распределенной памятью? Какая организация МВС (число процессоров, шин, распределение узлов по процессорам) этому соответствует?

2. Как объяснить изменение коэффициентов ускорения выполнения задачи на МВС с различной организацией с изменением числа процессоров?

3. Как объяснить изменение коэффициентов загрузки процессоров и шин при выполнении задач на МВС с различной организацией - с распределенной и общей памятью?

4. Объяснить сравнительные характеристики выполнения задач в МВС с общей и распределенной памятью.

Приложение 1
Общие методические указания по выполнению лабораторных работ № 2-5

Цикл лабораторных работ посвящен одному из основных разделов курса, связанному с организацией вычислений в многопроцессорных вычислительных системах (**МВС**). Рассматриваемые системы относятся к классу "множественный поток команд, множественный поток данных" (**МКМД**).

Основными компонентами современных МВС являются:

- **решающее поле**, состоящее из однородных или разнородных процессоров, со своей локальной памятью (или без нее);
- **коммутационная сеть**, позволяющая вести обмен данными, как между процессорами, так и между процессорами и общей памятью;
- **общая память модульного типа;**
- **иерархическая система управления,** позволяющая реализовать параллелизм обработки данных на уровне независимых задач, независимых ветвей задач, команд и операций.

В качестве параметров компонент системы используются:

- **решающее поле** – число процессоров (в данном цикле лабораторных работ рассматриваются МВС с одинаковыми процессорами), характеризующихся относительным временем выполнения независимых ветвей задачи, определяемым быстродействием процессоров;
- **локальная память (ЛП) процессора** - емкость (предполагается, что быстродействие ЛП таково, что не вносит задержку при обращении к ней процессора за данными, которые в ней находятся);
- **коммутационная сеть (КС)** – определяется типом (общая шина или мультиплексная шина), числом шин, пропускной способностью при передаче данных (характеризуется относительным временем занятия сформированного канала: «процессор – процессор», «процессор - общая память», «общая память – процессор»);
- **общая память** - число модулей памяти, быстродействие, косвенно отображаемое временем занятия канала процессор - общая память, общая память - процессор.

Затраты на управление в моделях не учитываются.

На вход такой МВС в общем случае может поступать:

1. ***одна сложная задача***, декомпозированная на подзадачи (***однозадачный режим*** функционирования МВС) или
2. набор независимых задач, каждая их которых может иметь свой приоритет (***многозадачный режим*** функционирования МВС).

В качестве модели задачи используется направленный граф, узлы которого отображают подзадачи. Каждой подзадаче соответствует часть (ветвь) программы, начав выполнение которой процессор заканчивает ее без прерываний. Дуги графа моделируют связи по данным между соответствующими подзадачами (ветвями программы).

Узлы графа взвешиваются целыми числами, соответствующими временам их выполнения в условных единицах (например, тактах) на процессорах. Дуги графа взвешиваются тоже целыми числами, соответствующими временам занятия канала связи (шины) при передаче данных от одного узла графа к другому.

В настоящем цикле лабораторных работ рассматриваются графы задач без обратных связей, с различным соотношением времен выполнения узлов графа ($\mathbf{t_p}$) и передачи данных ($\mathbf{t_n}$). В соответствие с этим соотношением различают **слабосвязные задачи** ($\mathbf{t_p \gg t_n}$), **среднесвязные** ($\mathbf{t_p \approx t_n}$), **сильносвязные** ($\mathbf{t_p \ll t_n}$).

Каждая задача может характеризоваться максимальным временем выполнения $\mathbf{T_{max}}$ и минимальным временем $\mathbf{T_{min}}$. Например, для задач, у которых для всех узлов $\mathbf{t_n=0}$, $\mathbf{T_{max}}$ вычисляется как сумма времен выполнения всех узлов графа (следует иметь в виду, что это справедливо, если все узлы графа выполняются на одном и том же процессоре), а $\mathbf{T_{min}}$ - как сумма времен выполнения узлов графа, принадлежащих его критическому пути.

Перед началом каждой лабораторной работы необходимо получить у преподавателя исходные данные для ее выполнения. В работах исследуются **графы задач №1** (число вершин 20-23) и **графы задач №2** (число вершин 10-13). Варианты графов задач представлены в соответствующих файлах.

При выполнении лабораторных работ используются программные модели: ***Lab_Work2.exe***, ***Lab_Work3.exe***, ***Lab_Work4.exe***, ***Lab_Work5.exe***. При запуске программ на экране появляется стандартное меню, включающее

- блок ввода исходных данных,
- блок моделирования и
- блок просмотра результатов.

При вводе исходных данных необходимо последовательно ввести следующие данные:

- режим работы МВС (однозадачный или многозадачный);
- граф задачи (или графы задач), который вводится из файла данных;
- характеристики системы: количество процессоров, шин, модулей памяти; стратегию выбора готовых узлов.

В блоке просмотра результатов приводятся временные характеристики вычислительного процесса (ВП), распределение узлов графа по процессорам, временные диаграммы выполнения ВП, коэффициенты загрузки процессоров, шин и т.д.

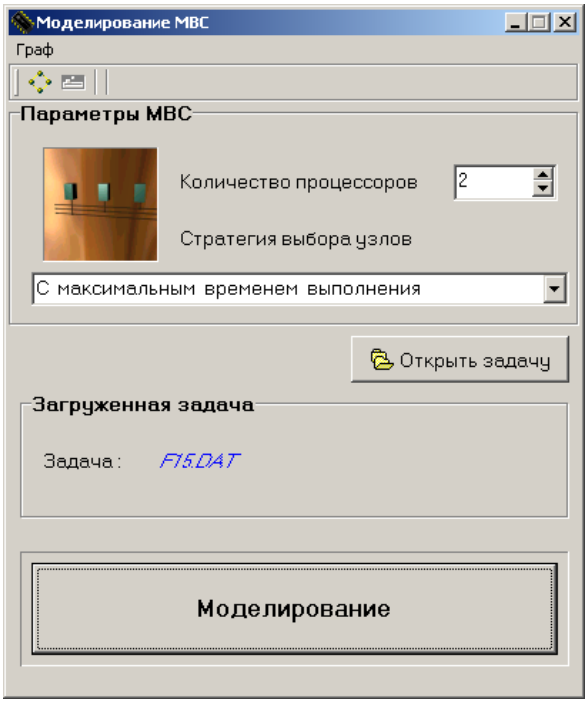

Рис.1. Окно моделирующей программы для лабораторной работы 2

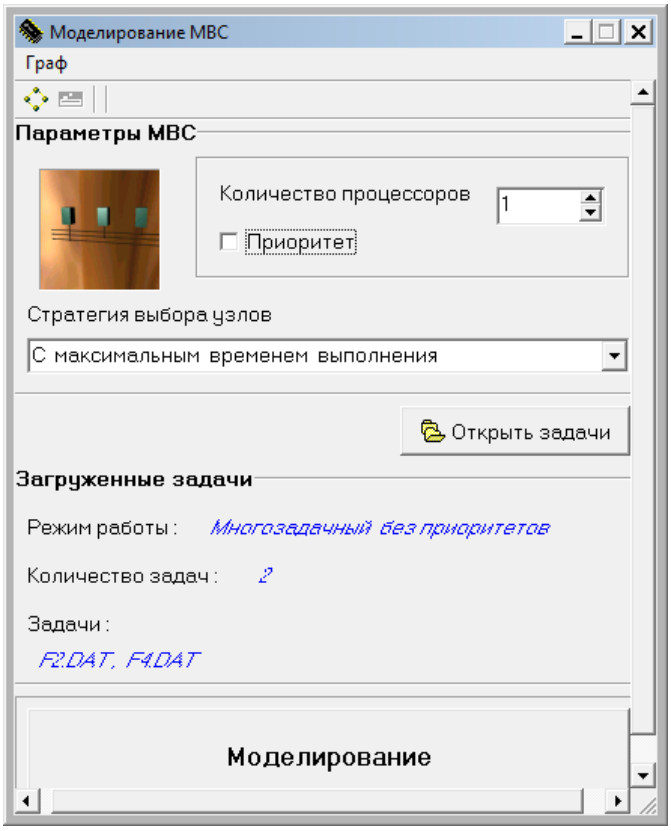

Рис.2. Окно моделирующей программы для лабораторной работы 3

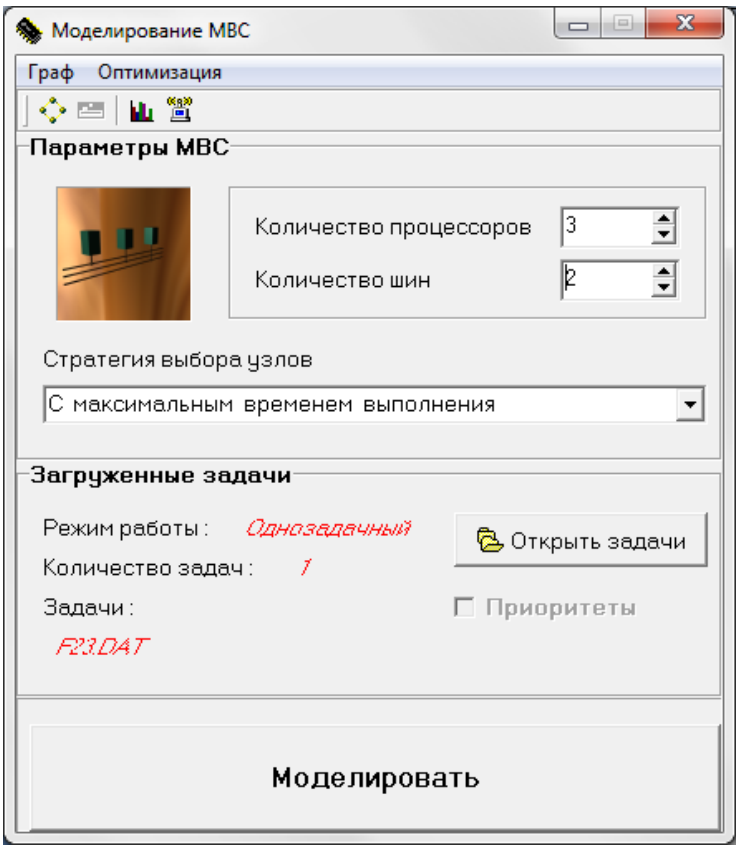

Рис.3. Окно моделирующей программы для лабораторных работ 4 и 5

Полученные результаты моделирования используются для исследований при выполнении лабораторных работ.

После анализа результатов необходимо **изобразить структуру МВС**, в которой наиболее эффективно решается анализируемая задача.

Приложение 2

Определение критического пути на графе задачи

Суть алгоритма заключается в определении минимально возможного и максимально возможного времени начала выполнения узлов графа. В предлагаемом комплексе лабораторных работ используется следующий алгоритм определения критических путей (КП) в графе со взвешенными узлами и дугами, то есть с учетом времен обработки узлов графа задачи и передачи данных между узлами графа. Данный алгоритм, аналогично соответствующему алгоритму для графа без учета времени передачи, основан на последовательном проходе по дереву графа от начальной вершины к конечной и возврата к начальной вершине. При начальном проходе по графу определяется минимально возможное время Tmin начала выполнения каждого узла по формуле:

$$T_{min\,i} = \max_{j=1}^{s} (T_{min\,j} + t_j + T_{ij}) \qquad (1)$$

где s - число узлов-предшественников i-го узла;

Tmin i (j) - минимально возможное время начала выполнения i-го (j-го) узла; t j - время выполнения j -го узла;

Tji - время передачи данных между узлами j и i, которому присваивается одно из значений множества $\{0, \tau_{ji}, 2\tau_{ji}\}$ в зависимости от способа организации памяти, где τ_{ji} - время передачи данных между узлами j и i, задаваемое на исходном графе задачи как вес ребра между j и i вершинами.

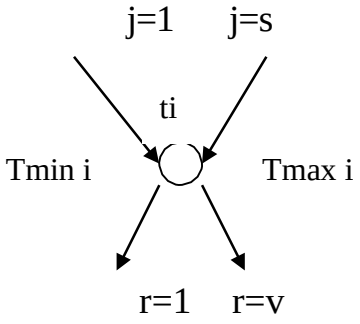

Рис. 1. Иллюстрация к алгоритму поиска критических узлов

При повторном анализе графа при проходе от конечной вершины к начальной определяется максимально возможное время начала выполнения узла по формуле:

$$T_{max\,i} = \min_{r=1}^{v} (T_{max\,r} - t_i - T_{ir}) \qquad (2)$$

где v - число узлов-последователей i-го узла;

$T_{max\ i\ (r)}$ - максимально возможное время начала выполнения i-го (r-го) узла; ti - время выполнения i-го узла;ir - время передачи данных от i-го узла к узлу r, значение которому присваивается аналогично T_{ij}

При этом для начальной вершины Tmin i = 0, а для конечной вершины минимально возможное время начала ее выполнения совпадает с максимально возможным временем начала выполнения - Tmax i = Tmin i.

Узел i -ый является критическим, если выполняется равенство
$T_{max\ i} = T_{min\ i}$

Критическим путем графа является множество последовательных узлов, начинающихся с входной вершиной и заканчивающихся выходной вершиной, удовлетворяющих условию **Tmax i Tmin i**, причем для каждой пары узлов принадлежащих КП соотношения (1) и (2) определяются соседней вершиной в паре.

Длина критического пути определяет минимальное возможное время выполнения графа задачи на МВС.

Граф задачи при выполнении ее на МВС с различной организацией может иметь различные критические пути вследствие изменения времени передач между узлами. Последнее определяется способом организации памяти.

Например, в МВС только с общей памятью (без локальной памяти процессоров) при определении критического пути время передачи данных от j-го узла i-му и от i-го узла r-му необходимо удваивать. Двойной учет этого времени происходит потому, что при передаче данных, во-первых, необходимо время τ_{ji}, чтобы передать данные j-го узла в общую память, и, во-вторых, время τ_{ji}, чтобы забрать данные из общей памяти и передать i -му узлу, даже в случае выполнения i -го и j -го узлов на одном процессоре.

В МВС с распределенной памятью каждый процессор имеет локальную память, в которой могут храниться промежуточные результаты. При этом, если узлы j и i обрабатываются одним процессором, то время обмена между ними считается равным 0 (то есть Tij = 0), если разными, то время обмена между узлами равно τji. Такое допущение объясняется тем фактом, что время передачи данных из процессора в свою оперативную память существенно меньше, нежели время пересылки данных в память удаленного процессора.

Выполнение этих дополнительных условий должно учитываться при решении задачи назначения.

Рассмотрим пример поиска критического пути для графа задачи без учета времен передач, представленного на рис. 2. Цифра внутри каждого кружка (узел графа) - номер узла, цифра над кружком - время выполнения узла tj. Дуги графа не взвешены, следовательно, Tji = Tir = 0.

Основным допущением при поиске КП на графе является следующее: граф задачи реализуется на системе с неограниченными ресурсами, в данном случае на МВС с неограниченным числом процессоров.

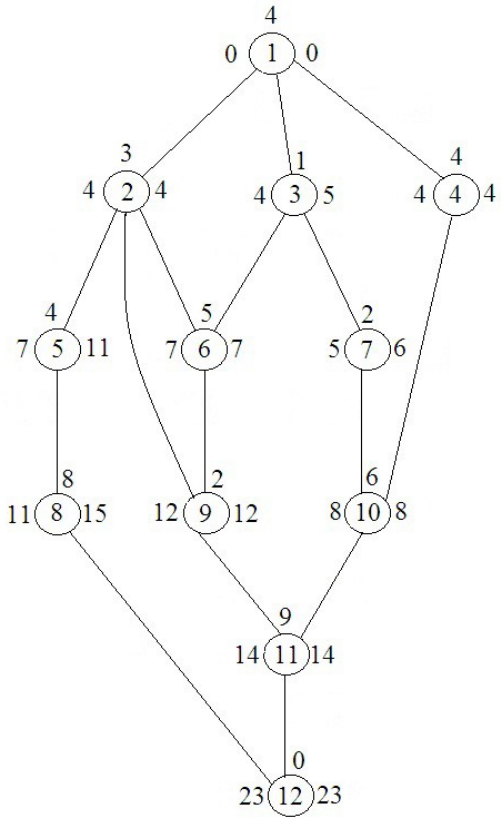

Рис. 2. Определение критического пути для графа задачи без учета передач

При движении по графу назад от конечной вершины к первой, справа от узла записывается максимально возможное время начала выполнения этого узла $T_{max\,i}$. Например, для узла 2 по формуле (2) находим:

$T_{max\,5} - t_2 = 11-3=8;$

$T_{max\,6} - t_2 = 7-3=8.$

Следовательно, $T_{max\,2} = \min (T_{max\,5}, T_{max\,6}) = 4.$

Таким образом, для графа, изображенного на рис. 2, имеются два критических пути **1→2→6→9→11→12** и **1→4→10→11→12**. При этом путь **1→2→6→9→11→12** не является критическим, в частности, потому что для пары узлов 2 и 9 соотношения (1) и (2) не определяются соседним в паре узлом. Например, $T_{min\,9}$ определяется не узлом 2, а узлом 6, аналогично и $T_{max\,2}$ не определяется узлом 9, т.е.:

$T_{min\,9} \neq T_{min\,2} + t_2 = 4+3 = 7;$

$T_{max\,2} \neq T_{max\,9} - t_2 = 12-3 = 9.$

Следовательно, для данной прикладной задачи $T_{min} = 23$, $T_{max} = 48$.

Приложение 3

Постановка задачи назначения

Прежде чем рассматривать существо задачи назначения, отметим, что задача назначения может решаться как в **статическом,** так и в **динамическом режимах.**

В первом случае задача назначения выполняется до начала реализации ВП в МВС (именно этот случай исследуется в данном цикле лабораторных работ), во втором она выполняется непосредственно в процессе реализации ВП.

Теоретической основой решения задачи назначения является теория расписаний.

Под решением задачи назначения понимается процесс распределения узлов графа задачи (набора задач), выполняемой в МВС, между ее процессорами, при котором определяется время начала выполнения узла, его длительность и назначение процессора, который обеспечит это выполнение.

Модель процесса распределения включает средства, описывающие ресурсы, систему узлов и дуг графа задачи (набора задач) и критерий оптимальности распределения. Под ресурсами понимаются: модули обработки (процессоры), модули памяти (она может быть распределенной, общей или смешанной), внутрисистемный интерфейс (общая шина, мультиплексная шина). При построении алгоритмов назначения в отказоустойчивых МВС в модель должны быть введены средства, описывающие систему обеспечения отказоустойчивости МВС. Например, средства, обеспечивающие введение дополнительных копий узлов графа задачи и дополнительных процессоров.

Рассмотрим наиболее простой случай.

Пусть в качестве ресурсов в модели используется только набор однотипных процессоров, имеющих равное быстродействие. На данном наборе процессоров выполняется вычислительный процесс, имеющий сетевую структуру и представляющий собой совокупность отдельных алгоритмов (сегментов задачи) $Z = \{z_i\}$.

Формально модель выполнения задачи Z можно представить совокупностью

$$\mathbf{\{Z, <, T, W, \Theta\},}$$

где $\mathbf{Z} = \{ z_i , z_L \}$ - множество сегментов задачи, выполняемых в системе (узлы графа);

< - означает задание в множестве Z отношения частичного порядка, которое определяет последовательность выполнения сегментов и информационные связи между ними (связность узлов);

$\mathbf{T} = \{t_1, \ldots, t_L\}$ - вектор времен выполнения сегментов на процессоре с заданным быстродействием;

W = $\{w_1, \ldots, w_L\}$ - вектор коэффициентов важности сегментов, соответствующих коэффициентам относительных потерь эффективности из-за невыполнения сегментов задачи вследствие отказа процессора, на котором выполняется данный сегмент;

|| Θ || = || τ_{iq} ||, i = 1 … L, q = 1 ... L-1 - матрица времен занятости шины с заданной пропускной способностью при передаче данных между узлами i и q.

Величина τ_{iq} является характеристикой не только структуры графа задачи и пропускной способности шины, но и способа организации МВС.

Как было отмечено выше, в качестве модели задачи используется направленный граф, узлы которого отображают сегменты задачи. При построении алгоритмов, реализующих задачу назначения, направленный граф преобразуется в таблицу связности, элементы которой

$$A_{ij} = \begin{cases} 1, \text{ если выходная информация узла } z_i \text{ является входной} \\ \text{для узла } z_j \\ 0 \text{ в противном случае.} \end{cases}$$

В качестве критериев оптимальности распределения узлов в МВС применяются обычно либо ***минимум времени выполнения задачи (набора задач) при ограничении на число процессоров***, либо ***минимум числа требуемых процессоров при ограничении на время решения задачи*** (набора задач).

В качестве производных от этих критериев используются следующие:

- максимум загрузки каждого процессора ($\mathbf{K_{загр\ i}}$),
- минимум простоев каждого процессора ($\mathbf{K_{пр\ i}}$).

$$\mathbf{K_{загр\ i} = T_i / T_{вып},}$$

где $\mathbf{T_i}$ – время, в течение которого i-й процессор занят обработкой задачи; $\mathbf{T_{вып}}$ - время выполнения задачи;

$$\mathbf{K_{пр\ i} = 1 - K_{загр\ i}}.$$

Следует отметить при этом, что эффективность как алгоритмов распределения узлов задачи, так и выбранной структуры МВС можно оценить с помощью коэффициента ускорения ($\mathbf{K_{уск}}$), показывающего ускорение времени решения задачи на **n** процессорах в сравнении с временем решения этой же задачи на одном процессоре

$$\mathbf{K_{уск} = T_{max} / T_n,}$$

где $\mathbf{T_{max}}$ – время решения задачи на одном процессоре; $\mathbf{T_n}$ – время решения задачи на n процессорах.

В теоретическом плане при построении алгоритмов оптимальных расписаний могут быть использованы математические методы, например, динамического программирования. Однако это достаточно сложная проблема, так как необходимо решать задачи большой размерности, при этом они относятся к NP -полным. На практике оптимальные расписания с использованием таких методов построены для простых типов прикладных задач сравнительно небольшой размерности, причем в основном для двухпроцессорных систем. Поэтому обычно применяются методы приближенной оптимизации, в частности, эвристические методы. Эвристические методы построения расписаний характеризуются тем, что они приспосабливаются как к информационно-логической структуре ВП, так и к структурной организации МВС, и относятся к классу приоритетного распределения. В таких расписаниях узлам задачи присваивается приоритет по тем или иным правилам (стратегии назначения), после чего узлы упорядочиваются в виде линейного списка по убыванию приоритетов. В процессе составления расписания осуществляется назначение узлов на процессоры в соответствие с их приоритетами для их выполнения. Наиболее исследованы и представлены в литературе различными моделями следующие стратегии назначения готовых к выполнению узлов вычислительного процесса:

1. равновероятный выбор;
2. выбор узла с минимальным временем выполнения;
3. выбор узла с максимальным временем выполнения;
4. выбор узла, принадлежащего критическому пути;
5. выбор узла, имеющего наибольшее число связей с последующими узлами;
6. выбор узла в порядке поступления в очередь на исполнение.

В данном цикле лабораторных работ используются стратегии **2-5**, при этом в ряде работ, где исследуются вопросы обработки набора задач, выбор готового к исполнению узла осуществляется **с учетом приоритета задачи.**

При выполнении лабораторных работ постановка задачи организации параллельных вычислений в МВС сводится к реализации следующей целевой функции:

Определить:

- минимальное число процессоров,
- шин связи коммутационной сети,
- модулей памяти,
- способ организации памяти
- тип стратегии назначения,

обеспечивающих выполнение прикладной задачи (набора задач) за заданное время.

Таким образом, при реализации выбранной стратегии назначения узлы ВП $Z-\{z_i\}$, i=1...L статически распределяются по процессорам МВС так,

что каждому из выбранного числа процессоров M_j сопоставляется некоторое подмножество узлов $z_j \in Z$.

Результатом распределения сегментов задач по процессорам является матрица $||X|| = ||X_{ij}||$, (i=1...L, j=1...n) и временная диаграмма занятости процессоров и шин, число которых тоже выбрано при выполнении поставленной задачи.

Приложение 4

Таблица номеров графов

для выполнения лабораторных работ №4 и №5 по моделированию однозадачного и многозадачного режимов вычислений в МВС с общей и распределенной памятью

№ варианта по журналу	№ графа задачи 1 из папки «Графы для однозад. режима» (Fi)	№ графов задачи 1 из папки «Слабосвязные графы» (F1_i, i=1,10)	№ графов задачи 2 из папки «Среднесвязные графы» (F2_i, i=1,10)	№ графов задачи 3 из папки «Сильносвязные графы» (F3_i, i=1,10)
1	1	8, 1	7, 1	10, 2
2	2	8, 2	7, 2	10, 3
3	3	8, 3	7, 3	10, 4
4	4	8, 4	7, 4	10, 5
5	5	8, 5	7, 5	10, 6
6	6	8, 6	7, 6	10, 7
7	7	8, 7	7, 8	10, 8
8	8	8, 9	7, 9	10, 9
9	9	8, 10	7, 10	10, 1
10	10	7, 1	1, 5	4, 3
11	9	7, 2	1, 6	4, 5
12	8	7, 3	1, 2	4, 7
13	7	7, 4	1, 10	4, 2
14	6	7, 5	1, 8	4, 7
15	5	7, 6	1, 9	4, 8
16	4	7, 9	6, 4	4, 1
17	3	7, 10	6, 2	4, 6
18	2	6, 4	6, 3	4, 9
19	1	6, 5	6, 5	3, 7
20	2	6, 3	6, 1	3, 8
21	3	6, 2	6, 8	3, 9
22	4	6, 1	6, 9	3, 6
23	5	5, 4	5, 2	3, 5
24	6	6, 9	5, 3	3, 4
25	7	6, 10	5, 4	3, 2
26	8	5, 4	5, 1	3, 1
27	9	5, 1	5, 6	7, 8
28	10	5, 2	5, 8	7, 2
29	9	5, 3	5, 9	7, 6
30	8	5, 9	5, 10	7, 5
31	7	5, 10	4, 2	7, 9
32	6	4, 3	4, 3	7, 3
33	5	4, 2	4, 8	7, 4
34	4	4, 1	4, 9	7, 1

35	**3**	**4, 9**	**4, 10**	**6, 8**
36	**2**	**4, 10**	**4, 1**	**6, 2**
37	**1**	**3, 1**	**3, 10**	**6, 5**
38	**8**	**3, 2**	**3, 2**	**6, 9**
39	**6**	**3, 9**	**3, 8**	**5, 9**
40	**9**	**3, 10**	**3, 9**	**5, 8**