ГРАФИЧЕСКАЯ КРУГОВАЯ ДИАГРАММНАЯ МОДЕЛЬ ПРЕДСТАВЛЕНИЯ ПРОГРАММ С ЦИКЛАМИ

А.В. Филатов

Чтобы представить, как выполняется или может быть выполнена интересующая последовательная или параллельная программа уже много лет используют графические описательные модели. Например, популярными на сегодняшний день моделями являются: граф ярусно-параллельной формы и диаграмма Ганта. Некоторые графические модели, например граф-схемы [2] даже используются в качестве языка программирования.

Однако, представлять циклы, особенно вложенные, с помощью графа ярусно-параллельной формы или диаграммы Ганта не просто, и по мнению автора такое представление является не всегда наглядным. поэтому желательно иметь специальную, ориентированную на решение этого вопроса модель. В данной статье автором предлагается графическая описательная модель, которую назовём круговой диаграммной моделью представления программ с циклами или просто *круговой диаграммной моделью* (КДМ).

Для начала возьмём единичный цикл. Выполнение каждого из *N* шагов этого цикла можно представить в виде круга разделённого на *M* секторов (см. рис. 1). Выполнение шага цикла можно представить как полный обход (на угол 360°) круга против часовой стрелки от некоторой точки (точки входа-выхода).

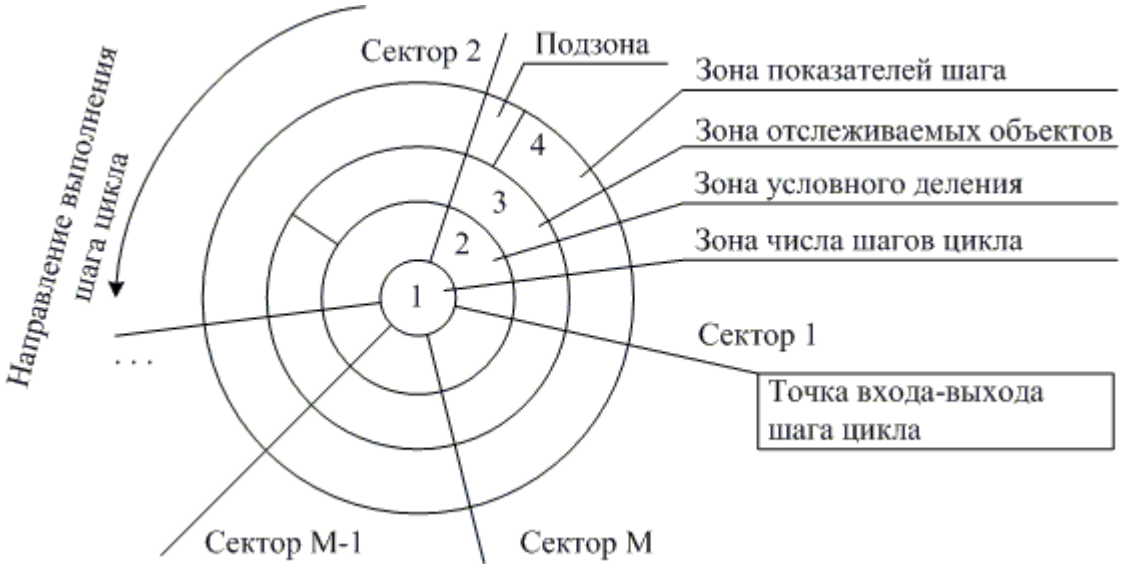

Рис. 1. Схема строения диаграммы КДМ

Круг разбивается на секторы. Каждый из секторов описывает определённый этап выполнения шага цикла. Чтобы описать каждый из этапов, вписанные концентрические окружности разбивают весь круг и его отдельные секторы на зоны.

В зону 1 (общую для всего круга) заносится число шагов цикла, для которых действует описываемая схема. Теоретически, одной диаграммой можно описать все *N* шагов цикла. В зоне 2 описываются условия выполнения этапов шага цикла (например, зависимость от условных операторов *if*). В зоне 3 описываются отслеживаемые объекты. Отслеживаемыми объектами могут быть действия и инструкции, выполняемые в рамках шага (этапа шага) цикла, а также обрабатываемые структуры данных и используемые устройства. В зоне показателей шага (зона 4) указываются идентификаторы этапов и подэтапов шага цикла. Например, в зоне 4, для каждого этапа (сектора) могут быть указаны номера (диапазоны номеров) выполняемых в его рамках инструкций программы. Некоторых зоны могут быть разделены на подзоны для обозначения подэтапов этапов шага цикла.

Порядок расположения зон 2-4 не фиксирован и зависит от удобства и наглядности описания. Следует отметить, что количество зон секторов диаграммы и их назначение могут иными.

Можно выделить КДМ двух видов: пропорциональную и непропорциональную. В пропорциональной КДИ полный угол (360°) диаграммы делится на секторы, угловые размеры которых пропорциональны некоторому показателю этапов. Например, пропорционально времени выполнения каждого из этапов шага цикла. В непропорциональной КДМ угловые размеры секторов диаграммы устанавливаются произвольно.

На рис. 2 показан пример пропорциональной КДМ фрагмента программы с циклом.

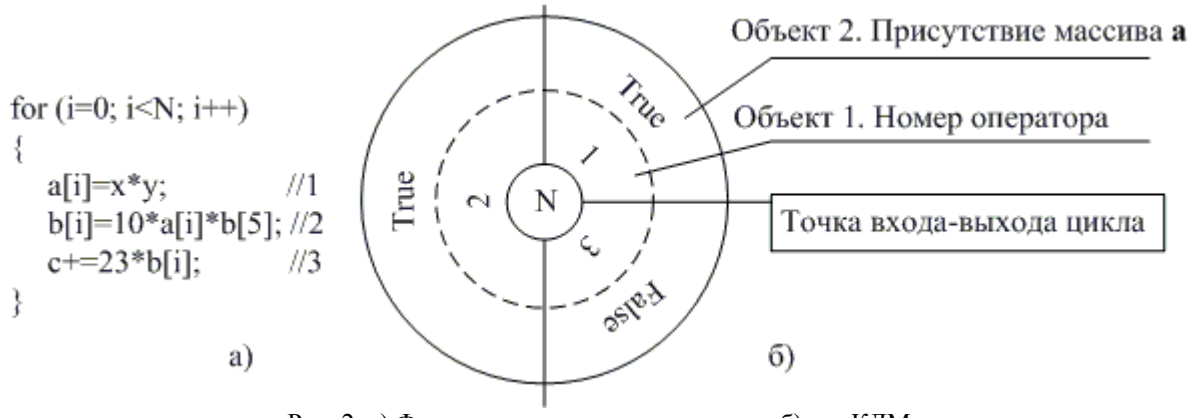

Рис. 2. а) Фрагмент программы с циклом и б) его КДМ

На рис. 2а показан фрагмент программы, а на рис. 2б его КДМ. В модели отражены только зоны 1 (число шагов цикла) и 3 (отслеживаемые объекты). В зоне 3 представлены объекты двух типов. Объект 1 – оператор представленный своим условным номером. Объект 2 – массив **a** представленный признаком использования в операторе (*True* или *False*). Модель пропорциональная, угловой размер секторов пропорционален числу операций умножения в операторе.

В КДМ, параллельной программы, в отличие от последовательной зоны 2, 3, 4 и при необходимости 1 могут быть продублированы по числу параллельных ветвей (потоков) или по числу групп подобных ветвей.

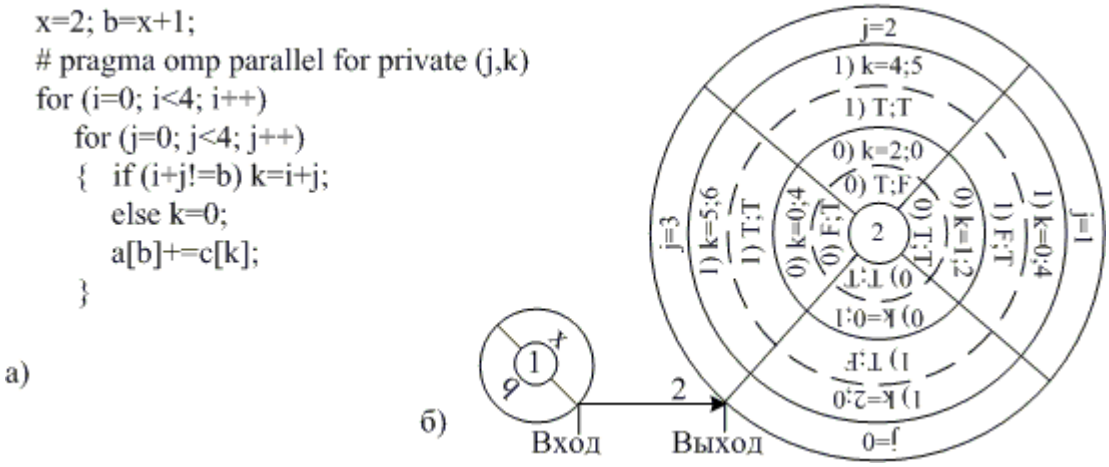

Рис. 3. Пример моделирования программы с распараллеленным на два потока циклом: а) текст фрагмента программы, б) вариант КДМ фрагмента программы.

На рис. 3. представлен пример моделирования фрагмента программы (см. рис. 3а) с двумя вложенным циклами и условным оператором. Директива *#pragma omp parallel for* стандарта *OpenMP* означает, что выполнение шагов внешнего цикла распределяется между запущенными параллельными потоками (пусть в нашем примере их два). На рис. 3б показан один и вариантов КДМ фрагмента программы. Выполнение первых двух операторов представлено отдельной диаграммой как цикл с единичным шагом. Выполнение шагов внешнего (*for i*) распараллеленного цикла представлено второй диаграммой. Число 2 у точки входа-выхода показывает количество параллельных потоков, которыми выполняются шаги цикла. Каждый из потоков выполнит по два шага из четырёх. Первый поток выполнит шаги с индексами *i=0* и *i=1*, а второй выполнит шаги с индексами *i=2* и *i=3*. Диаграмма разбита на четыре сектора по числу этапов. Отдельными этапами обозначены шаги внутреннего (*for j*) цикла которых четыре. Секторы диаграммы разбиты на пять зон (без учёта зоны числа шагов). Первые четыре зоны это продублированные для обоих потоков зоны условий и отслеживаемых объектов. Первая зона – это зона условий нулевого потока, а третья зона – это зона условий первого потока. Вторая же и четвёртая зоны – это зоны отслеживаемых объектов нулевого и первого потоков соответственно. Пятая (наружная) зона – зона показателей шага, которая идентифицирует соответствующий этап. В нашем случае это индекс (*j*) шага внутреннего цикла.

В зонах отслеживаемых объектов отображается значение переменной *k*, которое будет присвоено её на данном этапе. Поскольку переменная *k* заявлена в директиве *parallel for* как частная (клауза *private(j,k)*), то для каждого потока системой будет создан свой экземпляр этой переменной (как и переменной *j*). В каждой зоне, переменной *k* подставлено два значения разделённые точкой с запятой для первого и второго шагов внешнего цикла соответственно. В зонах условий, также для обоих шагов (через точку с запятой), буквами *T* (*True*) и *F* (*False*) показаны результаты выполнения условия *if (i+j!=b)*.

Есть несколько вариантов, как можно смоделировать данный фрагмент программы с помощью диаграммной модели. К сожалению, формат данной публикации не позволяет здесь представить все варианты.

```
for (zo=0; zo<16; zo++)
 for (yo=0; yo<16; yo++)
  for (xo=0; xo<16; xo++)
   for (zi=0; zi<16; zi++)
    for (yi=0; yi<16; yi++)
     for (xi=0; xi<16; xi++)
         Points[zo][yo][xo]+=Poitns[zi][yi][xi]+b;
```

Рис. 4. фрагмент программы с шестью вложенными циклами.

Теперь отдельно рассмотрим три способа описания программ и их фрагментов с вложенными циклами с помощью диаграмм. Первый способ это линеаризация вложенных циклов. В этом случае шаги вложенных циклов рассматриваются как этапы шага внешнего цикла и представляются как секторы диаграммы. Данный способ уже был продемонстрирован в рассмотренном примере. Второй способ, это когда в качестве шага выбирается шаг вложенного цикла и его выполнению отводится полный угол (360°) диаграммы, а число шагов вложенного цикла умножается на количество шагов циклов внешних ему и это значение заносится в центральную (первую) зону диаграммы. Третий способ, это когда шаги внешних и внутренних циклов представляются отдельными, но смежными диаграммами. Рассмотрим третий способ на конкретном примере.

Пусть имеется фрагмент программы с шестью вложенными циклами (см. рис. 4). В представленном фрагменте тремя внешними циклами варьируются элементы массива *Points,* куда будут сохранены (со сложением с имеющимся содержимым) результаты оператора сложения, а тремя внутренними циклами варьируются элементы массива *Points,* откуда берутся исходные данные (одно из слагаемых). Предположим, что размер одного структурированного элемента массива *Points* равен 5 байт, что при наличии 4096 элементов в массиве *Points* делает его размер равным 20480 байт. Предположим, что размер страницы памяти равен 4096 байт, а размер свободного пространства оперативной памяти даёт возможность разместить в ней только четыре страницы из пяти занятых массивом *Points* (для простоты, будем считать, что выравнивание элементов по границам страниц отсутствует). Предположим, что перед нами стоит задача смоделировать обработку данных по страницам, выявить операции подкачки с диска необходимой страницы, взамен откачиваемой на диск старейшей (по времени последнего обращения) страницы. Полученная КДМ представлена на рис. 5.

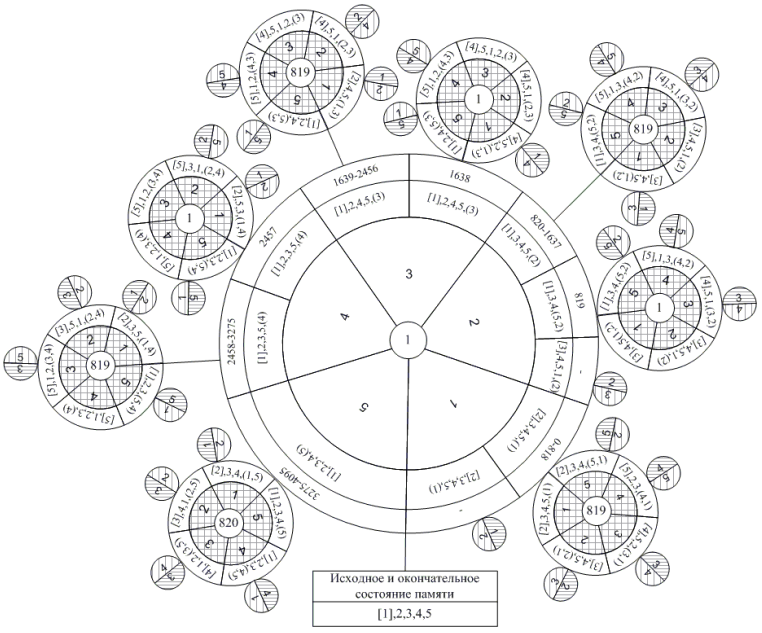

Рис. 5

Центральная диаграмма представляет линеаризацию трёх внешних циклов. Пять этапов, это использование пяти страниц памяти (обозначены во второй зоне) для сохранения результатов. В третьей зоне показано текущее состояние страниц памяти. В круглых скобках активные страницы, в квадратных скобках страница, выгруженная на диск, остальные страницы упорядочены по степени старения справа налево. В четвёртой зоне показаны диапазоны обрабатываемых элементов. Три внутренних цикла линеаризованы и представлены смежными диаграммами, построенными по схожему принципу, только во второй зоне указываются страницы, из которых извлекаются исходные данные, а четвёртая зона пропущена. Отдельными диаграммами показаны одношаговые циклы замены страниц в памяти. По построенной модели легко определить число циклов замены страниц в памяти (их 16393(!)) (отнимающими много времени) и причины, по которым системе их необходимо выполнить. Также можно наметить возможности (в данной работе не рассматриваются) сокращения числа таких циклов путём оптимизации программы.

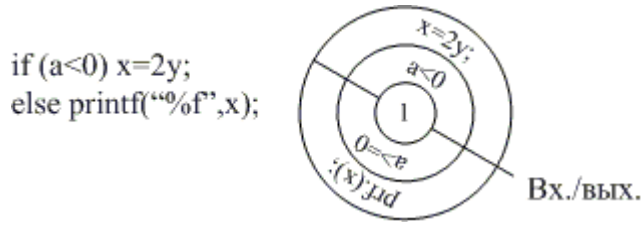

Рис. 6. Пример моделирования конструкции if-else

В дополнение к изложенному отдельно отметим простоту представления альтернативных операций. На рис. 6. приведён пример, как можно смоделировать конструкцию if-else.

Опыт использования автором круговой диаграммной модели показал наглядность и достаточное удобство этого инструмента. В настоящее время ведутся исследования по разработке правил трансформации диаграмм для выявления существенных зависимостей в целях оптимизации программ, также ведутся работы по усовершенствованию КДМ, в том числе в направлении использования их в автоматизированных системах подготовки и исследования параллельных программ, в частности для решения задач адаптации программ к вычислительным ресурсам [3].

ЛИТЕРАТУРА:

1. В.В. Воеводин, Вл.В. Воеводин. Параллельные вычисления. – Спб.: БХВ-Петербург, 2002. – 608 с.: ил. стр. 566 и сл.
2. Д.В. Котляров, В.П. Кутепов, М.А. Осипов. Граф-схемное потоковое параллельное программирование и его реализация на кластерных системах. М: Известия РАН, Теория и системы управления, 2005, №1, стр. 75-96.
3. А.В. Филатов. Схема организации сервисного программного обеспечения, содействующего удалённым пользователям в адаптации их программ к высокопроизводительным вычислительным ресурсам. Научный сервис в сети Интернет: решение больших задач: Труды Всероссийской научной конференции (22-27 сентября 2008 г., г. Новороссийск). - М.: Изд-во МГУ, 2008. - 468 с. стр. 257-260.