

## Параллельная MIMD-модель сетевого объекта

Д.С.Разинков, А.В. Молдованов, В.А. Святный

Кафедра ЭВМ, ДонГТУ, Украина

email: (rasinkov,mold,svjatnyj)@cs.dgtu.donetsk.ua

С.Н.Святный

Институт динамики сложных технических систем им. М.Планка,

Магдебург, Германия

email: svjatnyj@mpi-magdeburg.mpg.de

### Abstract

*Rasinkov D.S., Moldovanov A.V., Svjatnyj V.A., Sviatnyi S.N. An approach to concurrent simulation of coal mine ventilation systems is described. A ventilation system may be represented as a graph and described with differential equations. The approach consists in the use of a Message-Passing Interface (MPI) library for concurrent solution of differential equation systems. The obtained results confirm the efficiency of the equation-oriented approach to the simulation of ventilation systems.*

### Введение

Система проветривания шахтных выработок в качестве объекта моделирования является сложным сетевым динамическим объектом, для формального представления которого используются:

- граф с  $n$  узлами и  $m$  ветвями для описания топологии сети.
- системы алгебро-дифференциальных уравнений (ДАУ) для описания аэродинамической и газодинамической составляющей сети.
- система дифференциальных уравнений для описания системы автоматизированного управления.

В работе [1] дано детальное описание принципов построения модели сетевого объекта, способы представления его топологии, а также вывод систем ДАУ, описывающих аэрогазодинамические процессы, происходящие в шахтных выработках.

### 1. Тестовый сетевой объект

В настоящее время для моделирования сложных сетевых динамических объектов все чаще используются параллельные вычислительные системы. Для иллюстрации возможностей параллельного программирования с использованием коммуникационного стандарта MPI (Message Passing Interface), предложенного для организации взаимодействия процессов (процессоров) в параллельной вычислительной среде [2], рассмотрим модель тестового сетевого объекта (Рис. 1).

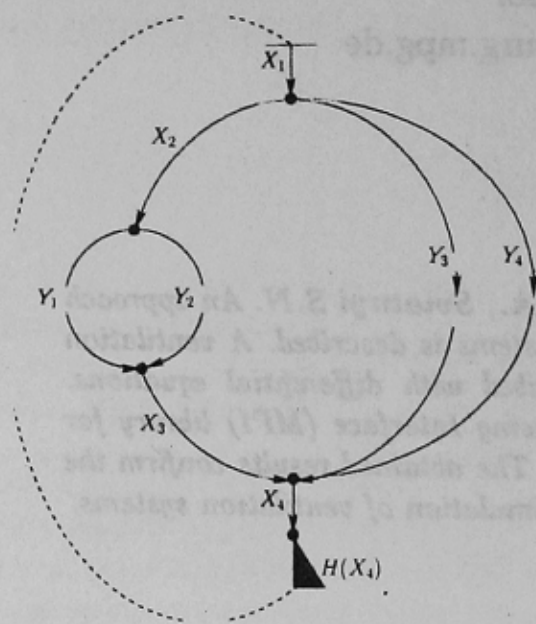


Рис. 1. Тестовый сетевой объект

$$Q = (X, Y)^T \tag{1}$$

$$A = (A_X, A_Y) \tag{2}$$

$$S = (S_X, S_Y) \tag{3}$$

$$Z = (Z_Y, Z_Y) \tag{4}$$

$$H = (H_X, H_X) \tag{5}$$

$$Z_X = X|X| \tag{6}$$

$$Z_Y = Y|Y| \tag{7}$$

$$K = \begin{pmatrix} K_X & 0 \\ 0 & K_Y \end{pmatrix} \tag{8}$$

$$R = \begin{pmatrix} R_X & 0 \\ 0 & R_Y \end{pmatrix} \tag{9}$$

$$\begin{cases} AQ & = 0 \\ SK\dot{Q} + SRZ & = SH \end{cases} \tag{10}$$

$$\begin{cases} A_X X + A_Y Y & = 0 \\ S_X K_X \dot{X} + S_Y K_Y \dot{Y} + SRZ & = SH \end{cases} \tag{11}$$

$$\begin{cases} \dot{Y} & = H_U - R_U Z \\ X & = -WY \end{cases} \tag{12}$$

Топологическая часть тестового сетевого объекта задается графом  $G(U, Q)$  с 4 узлами и 8 ветвями.

Аэродинамическая часть сети для многих практических задач представляется системой с сосредоточенными параметрами и описывается системой векторно-матричных уравнений (10). При этом  $Q$  - вектор расхода воздуха (1),  $A$  - матрица инциденций (2), составные части которой  $A_X$  и  $A_Y$  имеют вид (13)

$$A_X = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix} \quad A_Y = \begin{pmatrix} 0 & 0 & -1 & -1 \\ -1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \tag{13}$$

$S$  - матрица контуров, состоящая из  $S_X$  и  $S_Y$  вида (14)

$$S_X = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \quad S_Y = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (14)$$

Газодинамическую составляющую, а также систему автоматизации мы в данном примере не рассматриваем.

После ряда преобразований модель сетевого объекта можно представить в виде системы алгебро-дифференциальных уравнений (12).

При этом матрицы  $W$ ,  $H_U$  и  $R_U$  выглядят следующим образом :

$$W = A_X^{-1} A_Y \quad (15)$$

$$H_U = (S_Y K_Y - S_X K_X W)^{-1} S_H \quad (16)$$

$$R_U = (S_Y K_Y - S_X K_X W)^{-1} S_R \quad (17)$$

Численные значения параметров модели заданы в таблице (1).

Пар.	$X_1$	$X_2$	$X_3$	$X_4$	$Y_1$	$Y_2$	$Y_3$	$Y_4$
R	0.387	0.072	0.072	0.314	3.0	2.677	1.993	1.275
K	153.0	29.0	29.0	125.0	465.0	273.0	450.0	221.0
H	0	0	0	4100	0	0	0	0

Таблица 1. Параметры модели

Данная модель была реализована на языке высокого уровня "C" с использованием функций стандарта MPI. Тестирование программы было проведено на кластере из 4 персональных компьютеров под управлением ОС Linux и на супер-ЭВМ Paragon вычислительного центра Штуттгартского университета.

## 2. Структура и функции программы

Так как программа основана на решении системы дифференциальных уравнений 1-го порядка, записанных в матричном виде, определим матрично-векторные операции:

- инициализация матрицы  
void InitMatrix(char\* filename, Matrix pMatrix)
- инициализация вектора  
void InitVector(char\* filename, Vector pVector)
- элемент вектора, полученный умножением i-ой строки матрицы на вектор  
float VectorElement(Matrix pMatrix, Vector pVector, int rank)

Функции инициализации считывают из подготовленных заранее файлов данные (в будущем эти файлы должен генерировать топологический анализатор) и присваивают элементам векторов и матриц полученные значения.



Функция `VectorElement()` вычисляет  $i$ -ый элемент вектора, где  $i$  - номер процесса (процессора), в котором проводятся вычисления.

В основной части программы декларируются необходимые для работы переменные и осуществляется инициализация MPI. Для этого необходимы следующие функции:

```
MPI_Init(&argc, &argv);  
MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);  
MPI_Comm_size(MPI_COMM_WORLD, &p);
```

С помощью этих функций выделяется заданное пользователем число процессоров и им присваиваются логические номера от 0 до  $p-1$ . Подробное описание этих функций и их аргументов дано в работе [2].

После этого в 0 процессоре (процессе) происходит инициализация начальных данных. В программе не используются объединенные вектора  $Q$ ,  $H$  размерности  $(8 \times 1)$  и матрицы  $R$ ,  $S$  размерности  $(8 \times 8)$  и  $(4 \times 8)$ . Вместо этого используются вектора  $X$  и  $Y$  размерности  $(4 \times 1)$  и матрицы  $R_X$ ,  $R_Y$ ,  $S_X$ ,  $S_Y$  размерности  $(4 \times 4)$ .

В то время как 0-ой процесс занят инициализацией, остальные процессы находятся в состоянии ожидания. Для того, чтобы гарантировать запуск остальных процессов только после окончания работы 0-го, используется функция:

```
MPI_Barrier(MPI_COMM_WORLD),
```

т.е. все процессы ждут окончания инициализации.

Алгоритм выполнения программы предусматривает, что значения матриц и векторов известны не только 0-му, но и всем остальным процессам. Для передачи сообщений "от одного всем" служит функция:

```
MPI_Bcast(W, M*N, MPI_FLOAT, 0, MPI_COMM_WORLD);
```

В данном случае 0-ой процесс (4-й аргумент) посылает матрицу  $W$  всем остальным процессам ("остальные" - это `MPI_COMM_WORLD`).

`MPI_Barrier(MPI_COMM_WORLD)` как и раньше позволяет гарантировать окончание всех операций Broadcast до начала следующего программного сегмента.

Полученные в результате моделирования данные должны быть сохранены для последующего анализа и визуализации. Для этого каждый процесс открывает файлы с именами  $X_{\text{proc}} - \text{"номер процесса"}$  и  $Y_{\text{proc}} - \text{"номер процесса"}$  и записывает туда свои данные.

После проведения подготовительных операций начинается непосредственно решение уравнений (их интегрирование).

Сначала вычисляются (в каждом процессе отдельно) компоненты вектора  $X$  (в программе они названы  $X_{\text{local}}$ ). Затем вычисляются компоненты векторов  $Z_x$  и  $Z_y$  (соответственно  $Z_{x\text{local}}$  и  $Z_{y\text{local}}$ ).

После этого происходит формирование векторов  $X$ ,  $Z_x$ ,  $Z_y$  в 0-ом процессе. Для этого используется функция:

```
MPI_Gather (&Xlocal, 1, MPI_FLOAT, X, 1, MPI_FLOAT, 0, MPI_COMM_WORLD)
```

Эта функция собирает переменные Xlocal, находящиеся в каждом процессе (в порядке возрастания номеров процессов) в единый вектор X, находящийся в 0-ом процессе. Аналогичная операция проводится для векторов  $Z_x$  и  $Z_y$ . После выполнения этой функции векторы X,  $Z_x$  и  $Z_y$ , находящиеся в 0-ом процессоре, функциями MPI\_Bcast() рассылаются всем процессам.

Затем в каждом процессе вычисляется правая часть ОДУ (переменные Vlocal и VPlocal), которая используется для интегрирования системы уравнений по методу Адамса-Башфорда. Процесс интегрирования ведется с шагом h до момента времени Tmodel или до выхода на стационарный режим. Для проверки условия стационарности ( $\max(\Delta X, \Delta Y) \leq \text{Delta}$ ), где  $\Delta X$  и  $\Delta Y$  максимальное изменение компонент векторов X и Y между i и i+1 шагами интегрирования, используется еще одна функция MPI:

```
MPI_Reduce (&DeltaY, &MaxDeltaY, 1, MPI_FLOAT,
            MPI_MAX, 0, MPI_COMM_WORLD);
```

В данном случае эта функция находит максимальное значение среди переменных DeltaY и сохраняет его в переменной MaxDeltaY в 0-ом процессе.

Формирование и рассылка вектора Y происходит аналогично описанным выше процедурам для векторов X,  $Z_x$  и  $Z_y$ .

После окончания интегрирования закрываются файлы выходных данных и выполняется завершающая все процессы функция MPI\_Finalize().

## Результаты моделирования

Для визуализации полученных результатов были использованы графические возможности системы Matlab. Происходящие в сети процессы показаны на рис.2. Установившиеся значения расхода воздуха представлены в таблице 2.

$X_1$	=	69.536613 м <sup>3</sup> /с
$X_2$	=	28.032593 м <sup>3</sup> /с
$X_3$	=	28.032593 м <sup>3</sup> /с
$X_4$	=	69.536613 м <sup>3</sup> /с
$Y_1$	=	13.390879 м <sup>3</sup> /с
$Y_2$	=	14.641713 м <sup>3</sup> /с
$Y_3$	=	18.185787 м <sup>3</sup> /с
$Y_4$	=	23.318235 м <sup>3</sup> /с

Таблица 2. Установившиеся значения расхода воздуха

При этом выполняются условия :

$$X_1 = X_2 + Y_3 + Y_4 \quad (18)$$

$$\begin{aligned} X_2 &= Y_1 + Y_2 \\ X_3 &= Y_1 + Y_2 \\ X_4 &= X_3 + Y_3 + Y_4 \end{aligned}$$

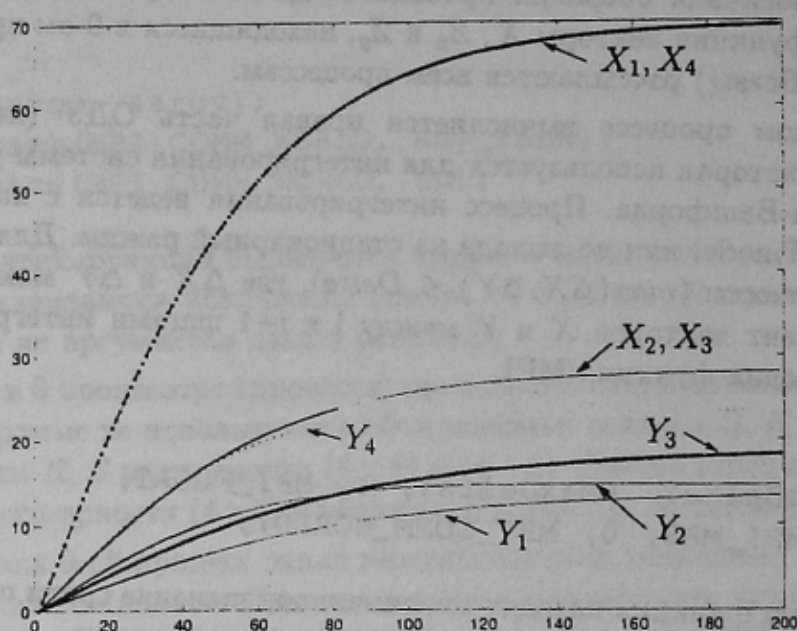


Рис. 2. Результати моделювання

## Заклучение

В данной работе представлена программная реализация на языке "C" модели тестового сетевого объекта. Даны комментарии и пояснения по использованию функций MPI для организации межпроцессного (межпроцессорного) обмена. Данная модель была отлажена и протестирована на параллельном компьютере Paragon и на кластере из 4-х компьютеров типа IBM PC.

Результаты моделирования хорошо согласуются между собой и с теоретическими выкладками.

Результаты, полученные в данной работе, подтверждают возможность создания эффективных параллельных моделирующих сред с помощью уравнение-ориентированного подхода к моделированию на основе стандарта MPI. Дальнейшие работы направлены на отладку и исследование MIMD-моделей сетей реальной размерности.

## Литература

1. Абрамов А.В., Фельдман Л.П., Святный В.А. *Моделирование динамических процессов рудничной аэрологии*, Киев, Наукова думка, 1981
2. Gropp W., Lusk E., Skjellum A., *Using MPI: Portable Parallel Programming with the Message-Passing Interface*, Cambridge, MA, MIT Press, 1994.