суперечливих правил показала, що приблизно 5-15 % правил є суперечливими. Задіювання таких правил у процесі діагностування може давати суперечливі або невірні результати.

Запропонований метод пошуку суперечностей, що базується на порівнянні вхідних та вихідних даних різних правил, забезпечив виявлення прямих та ланцюжкових конфліктів у БЗ. Метод може використовуватись для БЗ невеликого об'єму (100-700) правил.

Подальшим напрямком досліджень є розроблення методів пошуку у БЗ конфліктів комплексного типу.

Література

1. Hamscher W. C. Modeling Digital Circuits for Troubleshooting // Artificial Intelligence. – 1991. – Vol 51, № 1-3. – P. 223 – 271.

2. Поморова О. В. Апріорна діагностична інформація в структурі нейромережних експертів ідентифікації стану компонентів комп'ютерних систем // Радіоелектронні і комп'ютерні системи. – 2007. – № 8. – С.145 – 151.

3. Попов Э. В. Экспертные системы: Решение неформализованных задач в диалоге с ЭВМ. – М.: Наука, 1987. – 288 с.

4. Джексон П. Введение в экспертные системы. – М.: СпБ., Киев: "Вильямс", 2001. – 624 с.

5. Гаврилова Т.А., Хорошевський В.В. Базы знаний интелектуальных систем – Спб: Питер, 2001. – 384с.

6. Герасимов Б.М., Субач І.Ю. Показники якості інформаційного забезпечення та їх вплив на ефективність застосування ІСППР. Вісник Київського національного університету ім. Т. Шевченка; Військово-Спец. науки, 2008. – № 2, С. 18-25.

7. Поморова О.В., Гнатчук Є.Г. Виявлення суперечливості правил нечітких баз знань інтелектуальних систем технічного діагностування.

Надійшла 23.5.2010 р.

УДК 004.272.2

Д.М. МЕДЗАТИЙ, А.О. МАМЧУР, Л.А. МАТВІЙЧУК, В.В. ГОНЧАРУК
Хмельницький національний університет

АНАЛІЗ ТА СИНТЕЗ ТОПОЛОГІЙ РОЗПОДІЛЕНИХ ОБЧИСЛЕНЬ

В статті розглянуто основні принципи та особливості використання топологічного аналізу для оптимізації паралельного обчислювального процесу. Проведено аналіз методів представлення топологій обчислювального процесу. Наведено приклад реалізації запропонованого підходу.

The article reviews the basic principles and specific use of topological analysis for optimizing parallel computation process. The analysis methods of presentation of topology computation process. Implemented the proposed approach.

Ключові слова: паралельні обчислення, розподілені обчислення, топологія обчислювального процесу.

Вступ

На сьогодні розподілені обчислення є одним з ефективних засобів розв'язання трудомістких задач з використанням двох або більше комп'ютерів, об'єднаних в мережу. Доступність та економічна доцільність цього підходу (особливо для систем типу Beowulf) робить його привабливим для застосування у навчальних закладах, наукових лабораторіях та інших установах, де є потреба у розв'язання трудомістких задач. Однак, розроблення програмного забезпечення (ПЗ), яке використовує розподілені обчислення для розв'язання поставлених задач, не є простим завданням. Основними проблемами розробки такого ПЗ є складність контролю виконання та складність оптимального розподілення задач між вузлами мережі, а також синхронізація даних.

З ускладненням обчислювального процесу зростає складність контролю над ним, збільшується ймовірність виникнення помилок та зменшується продуктивність системи. Наприклад, впровадження нового проміжного рівня обчислення задачі може призвести до повної зміни обчислювального процесу, проконтролювати який досить складно, а в деяких випадках і зовсім неможливо.

Аналіз досліджень та публікацій

Відомі підходи до розв'язання задачі проектування паралельних алгоритмів і програмного забезпечення зорієнтовані на розроблення систем шляхом аналізу поставленої задачі та можливих шляхів її розв'язання [1, 2]. Широко використовується масштабування підзадач та їх розподіл між процесорами [1, 3, 4]. Разом з тим, на нашу думку, у дослідженнях недостатньо висвітлені питання автоматизації процесу розроблення програмних засобів орієнтованих на виконання у розподілених системах. Задачі аналізу та синтезу «паралельних» алгоритмів та програмного забезпечення цілком покладаються на програміста. Таке положення ставить під сумнів ефективність та оптимальність отриманих програмних засобів. Отже, задача розроблення підходів щодо аналізу та оптимізації топологій систем розподілених обчислень, які придатні до автоматизації, є актуальною.

Одним з варіантів автоматизації процесу аналізу та синтезу топологій систем розподілених

обчислень може бути запропонований у [5] підхід. Дана робота має на меті довести можливість та доцільність використання загальних принципів аналізу та синтезу топологій, запропонованих у [5] при розв'язанні задачі автоматизації процесу проектування систем розподілених обчислень.

Способи представлення топологій

Перед розробкою програмного забезпечення для розв'язання задачі необхідно, в першу чергу, спланувати етапи обчислення та представити їх в одному із способів представлення топології. Топологія обчислення може бути представлена за допомогою наступних способів [1, 5]: вербально-дедуктивний, графічний, матричний або аналітичний спосіб.

Вербально-дедуктивний спосіб або словесно-логічний спосіб застосовуються, як правило, на початку проектування систем, найчастіше на етапі планування та формування технічного завдання. Це найбільш простий і доступний спосіб для різних спеціалістів, які можуть бути залучені до створення системи, не маючи спеціальних знань [5].

До графічних способів відносяться графи та мережі Петрі. Граф $G = \{B, D\}$ – це сукупність двох множин: $B = \{b_1, b_2, \mathbf{K}, b_n\}$ – скінченно непуста множина вершин (вузлів); $D = \{d_1, d_2, \mathbf{K}, d_n\}$ – скінченна множина пар вершин, що з'єднані між собою і утворюють ребро (дугу) графа G. Даний спосіб є оптимальним для відображення та конструювання етапів обчислення. В даному випадку вершиною графа можна представити під задачу, яка може бути обчислена паралельно, а ребром представити зв'язок однієї задачі з іншою, значенням якого буде час на виконання та передачу даних до інших підзадач.

До матричних способів відносяться: матриці (суміжності, інциденцій, цикломатичні), n-мірні таблиці, n-мірні куби. Одним із основних способів представлення топології є матриця суміжності. Матриця суміжності – це квадратна матриця, що задає топологію системи і має наступний вигляд:

$$A = \begin{vmatrix} a_{11} & a_{12} & \mathbf{L} & a_{1n} \\ a_{21} & a_{22} & \mathbf{L} & a_{2n} \\ \mathbf{M} & \mathbf{M} & a_{ij} & \mathbf{M} \\ a_{n1} & a_{n2} & \mathbf{L} & a_{nn} \end{vmatrix}, \text{ де } \forall a_{ij} \in \{0,1\}, i = \overline{1,n}, j = \overline{1,n} \quad (1)$$

Якщо, $a_{ij} = 1$, то це означає, що елемент *i* та *j* топології системи зв'язані між собою, а якщо $a_{ij} = 0$, то жодного зв'язку між цими елементними немає.

Для представлення топології системи іноді застосовуються матриці інциденції. Якщо топологія задається орієнтованим графом $G = \{B, D\}$, де $B = \{b_1, b_2, \mathbf{K}, b_n\}$, $D = \{d_1, d_2, \mathbf{K}, d_m\}$, то матриця інциденцій для дуг *S* матиме наступний вигляд:

$$S = \begin{vmatrix} s_{11} & s_{12} & \mathbf{L} & s_{1m} \\ s_{21} & s_{22} & \mathbf{L} & s_{2m} \\ \mathbf{M} & \mathbf{M} & s_{ij} & \mathbf{M} \\ s_{n1} & s_{n2} & \mathbf{L} & s_{nm} \end{vmatrix}, \text{ де } s_{ij} = \begin{cases} +1, \textit{ якщо } d_j \textit{ виходить з } b_i \\ -1, \textit{ якщо } d_j \textit{ входить в } b_i \\ 0, \textit{ якщо } d_j \textit{ не інцидентна } b_i \end{cases} \quad (2)$$

Якщо топологія задається неорієнтованим графом $G = \{B, D\}$, де $B = \{b_1, b_2, \mathbf{K}, b_n\}$, $D = \{d_1, d_2, \mathbf{K}, d_m\}$, то матриця інциденцій для ребер *R* матиме вигляд:

$$R = \begin{vmatrix} r_{11} & r_{12} & \mathbf{L} & r_{1m} \\ r_{21} & r_{22} & \mathbf{L} & r_{2m} \\ \mathbf{M} & \mathbf{M} & s_{ij} & \mathbf{M} \\ r_{n1} & r_{n2} & \mathbf{L} & r_{nm} \end{vmatrix}, \text{ де } r_{ij} = \begin{cases} +1, \textit{ якщо } d_j \textit{ інцидентна } b_i \\ 0, \textit{ якщо } d_j \textit{ не інцидентна } b_i \end{cases} \quad (3)$$

Перевага застосування матриць суміжності полягає у зручному представленні та опрацюванні топологій з довільною кількістю елементів. Єдиним, але суттєвим, недоліком матричних способів представлення топологій є низька наочність. Тому, для виведення на екран монітора результатів аналізу топологій у відповідних програмах необхідно передбачити перетворення матриць у графи.

До аналітичних способів визначення топологій систем можна віднести логічні схеми алгоритмів (ЛСА). ЛСА спочатку були розроблені для компактного опису процесу функціонування цифрових пристроїв у вигляді формул, що можуть складатися з двох об'єктів – оператор та логічна умова [5].

Таким чином, для аналізу топологій будемо застосовувати лише два способі її представлення: графічний, за допомогою графів та матричний. Представлення топології в вигляді графів є найзручнішим способом введення структури топології для користувача, і буде застосовуватися як основний засіб комунікації людини з програмним забезпеченням аналізу топології систем. Матричне представлення топологій буде застосовуватись як основний засіб аналізу систем, так як цей спосіб найзручніший для опрацювання даних.

Оптимізація топології розподілених обчислень

Розглянемо процес розв'язання задачі оптимізації топології розподілених обчислень на прикладі

конкретної задачі. Нехай задана трудомістка задача, яка потребує великих обчислювальних ресурсів. Дана задача може бути розділена на N підзадач (де $N \geq 2$), кожна із підзадач потребує M вхідних даних (де $1 \leq M \leq N$) та видає єдиний результат обчислення, який може передаватись K іншими підзадачами (де $1 \leq K < N$). Для зручності аналізу системи припустимо, що кількість елементів обчислення в кластері необмежена. Топологія обчислення даної задачі може бути представлена у вигляді орієнтованого графа (рис. 1).

Метою є визначення правильності побудови топології системи та кількості необхідних обчислювальних елементів кластера для обчислення даної задачі.

Відмітимо, що на графі позначені дві особливі вершини: 0 – підзадача введення даних; 11 – підзадача виведення даних, наявність яких ніяк не впливає на методику обчислення, і зображені вони тільки для повноти сприйняття обчислювального процесу. Надалі підзадачі введення-виведення відображатись не будуть, але необхідно мати на увазі, що вони завжди присутні в реальних системах.

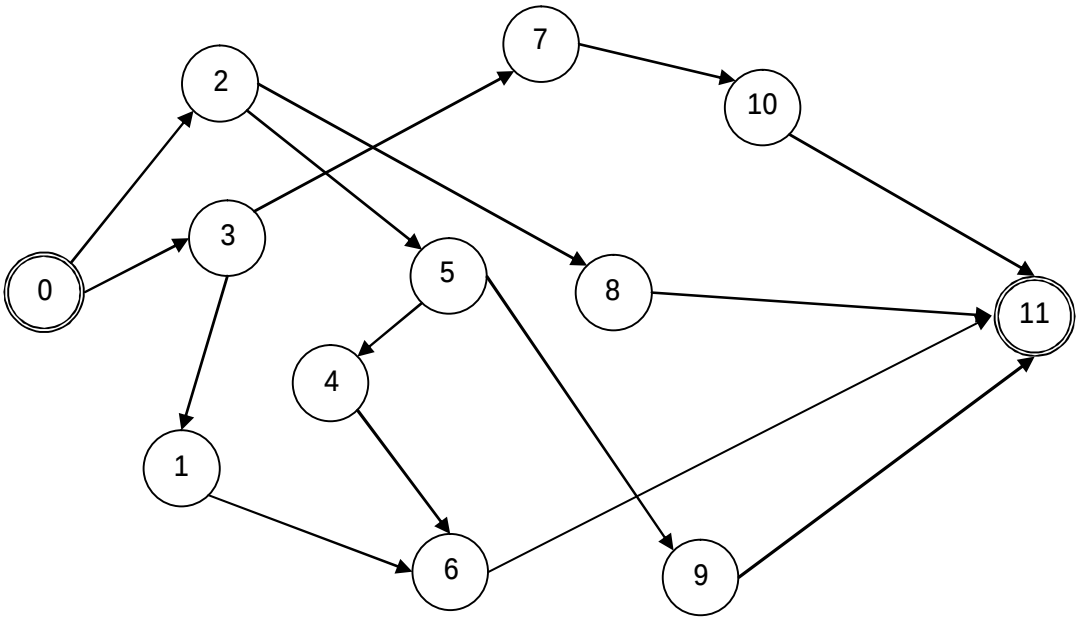

Рис. 1. Представлення топології розподіленого обчислення у вигляді графа

Топологія обчислюваної системи розроблена правильно, якщо вона задовольняє наступним вимогам:

1. Топологія системи повинна буди представлена у вигляді орієнтованого графа.
2. Топологія не повинна мати циклів.
3. Топологія системи не повинна бути послідовною.

Вилучимо підзадачі введення-виведення інформації, оскільки нас цікавлять тільки підзадачі обчислення (рис. 2).

Представлення топології у вигляді графа зручне для користувача, але аналіз її програмними засобами, в такому вигляді, потребує додаткових обчислювальних ресурсів. Для спрощення аналізу топології системи, представимо її у вигляді матриці суміжності з'єднань за входами та виходами. Для обох матриць справедливе наступне твердження $A_I = (A_O)^T$ та $A_O = (A_I)^T$, де $()^T$ – операція транспонування матриць.

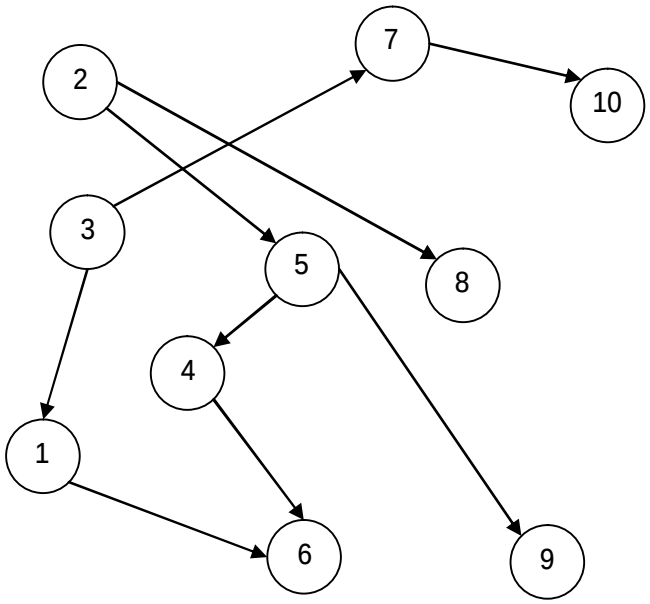

Рис. 2. Представлення топології розподіленого обчислення у вигляді графа, без підзадач введення-виведення

На даному етапі застосовуються прості операції для перетворення та перевірки топології. Для перевірки топології на наявність контурів (циклів) довжиною рівних k, необхідно виконати операцію логічного піднесення до степеня n матриці суміжності, де $1 \leq n \leq k$. При кожному піднесенні до степеня необхідно перевіряти наявність одиниць на головній діагоналі результуючої матриці. Даний метод вважається класичним і є найпростішим для опанування, але потребує значних обчислювальних ресурсів,

так як складність даного алгоритму становить $O(N^3)$.

$$A_I = \begin{vmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{vmatrix}, A_O = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

Для перевірки наявності контурів довжиною 2 використовується операція логічного множення матриць. Якщо задана матриця $A = \left|a_{ij}\right|_{mxn}, \forall a_{ij} \in \{0,1\}$ і матриця $B = \left|b_{ij}\right|_{nxp}, \forall b_{ij} \in \{0,1\}$, то кожен елемент результуючої матриці буде визначатися як $c_{ij} = \vee_{s=1}^{P} a_{is} \,\&\, b_{si}$.

Якщо основна діагональ результуючої матриці, яка була отримана в результаті логічного множення суміжної матриці самої на себе, містить одиниці, то дана топологія містить хоча б один контур довжиною 2 [6].

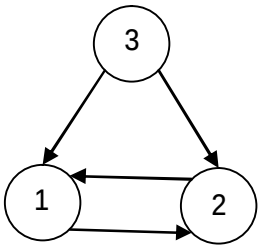

Рис. 3. з одним контуром довжиною 2

Для прикладу розглянемо граф, який містить один контур довжиною 2 (рис. 3), та визначимо його наявність за допомогою логічного множення матриць.

$$A = \begin{vmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{vmatrix}, \; A^2 = \begin{vmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{vmatrix} \& \begin{vmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{vmatrix}$$

Так як $\sum_{i=1}^{n} a_{ii} \neq 0$, то заданий граф містить хоча б один контур довжиною 2.

Перевіримо топологію на наявність контуру довжиною 2. Для цього проведемо аналогічні дії.

$$A^2 = A \,\&\, A = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}, \quad A^3 = A^2 \,\&\, A = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix},$$

$$A^4 = A^2 \,\&\, A^2 = \begin{vmatrix} 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0 \end{vmatrix}, \quad A^5 ... A^{10} = \begin{vmatrix} 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0 \end{vmatrix}$$

Виходячи з умови:

$$\sum_{k=1}^{n}\sum_{i=1}^{n} a_{ii}^{k} = 0\text{, де } a_{ii}^{k} \text{ елемент матриці } A^k \tag{4}$$

дана топологія не містить контурів та петель [6, 9].

Наступним етапом повірки заданої топології на можливість розподіленого обчислення є визначення типу топології. Для цього використаємо метод визначення послідовної топології системи. Топологія є послідовною, якщо існує матриця суміжності з'єднань за входами $A_I = \left|a_{ij}\right|_{n\times n}$ і відповідна їй матриця суміжності з'єднань за виходами $A_O = \left|a_{ij}\right|_{n\times n}$, в якій є тільки один рядок, що містить всі нульові елементи, а решта рядків містять тільки по одному одиничному елементу не на головній діагоналі.

Розглянемо граф, який відображає послідовну топологію (рис. 4).

Рис. 4. Граф послідовної топології

Для даного графа побудуємо матриці суміжності з'єднань за входами (A_I) та виходами (A_O).

$$A_I = \begin{vmatrix} 0&0&0&0&0\\ 1&0&0&0&0\\ 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&0&1&0 \end{vmatrix}, \; A_O = \begin{vmatrix} 0&1&0&0&0\\ 0&0&1&0&0\\ 0&0&0&1&0\\ 0&0&0&0&1\\ 0&0&0&0&0 \end{vmatrix}$$

Для зручності аналізу топології на послідовність впровадимо ряд функцій:

$$F(A) = \sum_{i=1}^{n} a_{ii} \text{ – сума елементів основної діагоналі,} \tag{5}$$

$$G(A,i) = \begin{cases} \textit{істинне, якщо} \sum_{j=1}^{n} a_{ij} \in \{0,1\} \\ \textit{хибне, якщо} \sum_{j=1}^{n} a_{ij} \notin \{0,1\} \end{cases} \tag{6}$$

$$P(A,i) = \begin{cases} \textit{істина, якщо} \sum_{j=1}^{n} a_{ij} = 0 \\ \textit{хибна, якщо} \sum_{j=1}^{n} a_{ij} \neq 0 \end{cases} \tag{7}$$

Отже, якщо матриця описує послідовну топологію, то вона має задовольняти наступним умовам: $F(A_I) = 0 \wedge F(A_O) = 0 \wedge \forall i G(A_I, i) \wedge \forall i G(A_O, i) \wedge \exists! i P(A_I, 1) \wedge \exists! i P(A_O, 1)$, де A_I – матриця суміжності з'єднань за входами, A_O – матриця суміжності з'єднань за виходами.

Застосуємо цей метод для топології, яка зображена на рис. 4.

$F(A_I) = 0$, $F(A_O) = 0$,

$\forall iG(A_I, i)$ – твердження істинне,

$\forall iG(A_O, i)$ – твердження істинне,

$\exists! iP(A_I, 1)$ – твердження істинне для *i* = 1, і тільки для нього одного,

$\exists! iP(A_O, 1)$ – твердження істинне для і = 5, і тільки для нього одного.

Отже топологія є послідовною.

Спробуємо визначити належність графа рис. 2 до послідовної топології.

$F(A_I) = 0$, $F(A_O) = 0$,

$\forall iG(A_I, i)$ – твердження хибне, отже топологія не є послідовною.

Проаналізувавши властивості данаї топології, можна зробити висновок, що вона побудована вірно і придатна для проведення обчислень на розподілених системах, а саме:

- топологія представлена орієнтованим графом;
- топологія не містить жодного циклу чи петель;
- топологія не є послідовною.

Наступним етапом аналізу топології розподіленого обчислення є визначення кількості паралельних гілок, необхідної кількості елементів кластера, кількості етапів обробки інформації та знаходження критичних шляхів топології. Для вирішення цих проблем застосовуються методи перетворення графів в ярусно-паралельну форму.

Ярусно-паралельна форма графа (ЯПФ) – це граф, у якого всі вершини розподілені на яруси таким чином, що в межах одного ярусу між вершинами графу зв'язки відсутні [7].

Нумерацію ярусів проводять залежно від того, яку матрицю суміжності обрали за основу – з'єднань за виходами чи з'єднань за входами. В першому випадку нумерація починається з ярусу, що містить вихідні елементи топології, тобто елементи, виходи яких є виходами системи. В другому випадку нумерація починається з ярусу, що містить вхідні елементи, виходи яких є входами системи.

Якщо є матриця з'єднань за виходами $A = \left| a_{ij} \right|_{n \times n}$ і матриця $B = \left| b_{ij} \right|_{n \times n}$, нульові елементи якої відповідають вершинам k-го ярусу утворюється матриця $F = B \cdot A_O$, нульові елементи якої відповідають тільки вершинам графу k-го і $(k+1)$-го ярусів. Також якщо існує матриця $F = \left| f_i \right|_n$, нульові елементи якої відповідають вершинам графу k-го і $(k+1)$-го ярусів, і якщо існує матриця $B = \left| b_{ij} \right|_n$, одиничні елементи якої відповідають вершинам графу k-го ярусу, то в матриці $C = \overline{F}\ \&\ B$ одиничні елементи відповідають тільки вершинам графа $(k+1)$-го ярусу. Застосуємо даний підхід до нашої топології.

$$A = \begin{vmatrix} 0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0 \\ 1&0&0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0 \end{vmatrix}, B_0 = \begin{vmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{vmatrix},$$

$$\cdot\ B_1 = B_0 * A = \begin{vmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{vmatrix} * \begin{vmatrix} 0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0 \\ 1&0&0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0 \end{vmatrix} = \begin{vmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{vmatrix}, \quad C_1 = \overline{B_1} * B_0 = \begin{vmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix} \& \begin{vmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{vmatrix} = \begin{vmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{vmatrix},$$

$$B_2 = B_1 * A = \begin{vmatrix}1\\0\\0\\1\\1\\1\\1\\1\\1\end{vmatrix} * \begin{vmatrix}0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\0&1&0&0&0&0&0&0&0&0\\1&0&0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&1&0&0&0\end{vmatrix} = \begin{vmatrix}0\\0\\0\\1\\0\\1\\0\\0\\1\\1\end{vmatrix}, \quad C_2 = \overline{B_2} * B_1 = \begin{vmatrix}1\\1\\1\\0\\1\\0\\1\\1\\1\\0\end{vmatrix} \& \begin{vmatrix}1\\0\\0\\1\\1\\1\\1\\1\\0\\1\end{vmatrix} = \begin{vmatrix}1\\0\\0\\0\\1\\0\\1\\1\\0\\0\end{vmatrix},$$

$$B_3 = B_2 * A = \begin{vmatrix}0\\0\\0\\1\\0\\1\\0\\0\\1\\1\end{vmatrix} * \begin{vmatrix}0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\0&1&0&0&0&0&0&0&0&0\\1&0&0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&1&0&0&0\end{vmatrix} = \begin{vmatrix}0\\0\\0\\0\\1\\0\\0\\0\\0\\0\end{vmatrix}, \quad C_3 = \overline{B_3} * B_2 = \begin{vmatrix}1\\1\\1\\1\\1\\0\\1\\1\\1\\1\end{vmatrix} \& \begin{vmatrix}0\\0\\0\\1\\0\\1\\0\\0\\1\\1\end{vmatrix} = \begin{vmatrix}0\\0\\0\\1\\0\\0\\0\\0\\1\\1\end{vmatrix},$$

$$B_4 = B_3 * A = \begin{vmatrix}0\\0\\0\\0\\0\\1\\0\\0\\0\\0\end{vmatrix} * \begin{vmatrix}0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\0&1&0&0&0&0&0&0&0&0\\1&0&0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0&0\\0&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&1&0&0&0\end{vmatrix} = \begin{vmatrix}0\\0\\0\\0\\0\\0\\0\\0\\0\\0\end{vmatrix}, \quad C_4 = \overline{B_4} * B_3 = \begin{vmatrix}1\\1\\1\\1\\1\\1\\1\\1\\1\\1\end{vmatrix} \& \begin{vmatrix}0\\0\\0\\0\\0\\1\\0\\0\\0\\0\end{vmatrix} = \begin{vmatrix}0\\0\\0\\0\\0\\1\\0\\0\\0\\0\end{vmatrix}.$$

Результатом є матриця $C = \left|c_{ij}\right|_{k \times n}$, де k – кількість ярусів, а кожний стовпець цієї матриці говорить про наявність певної вершини на j-му ярусі (рис. 5).

$$C = \begin{vmatrix}0&1&1&0&0&0&0&0&0&0\\1&0&0&0&1&0&1&1&0&0\\0&0&0&1&0&0&0&0&1&1\\0&0&0&0&0&1&0&0&0&0\end{vmatrix}$$

Для визначення максимальної кількості елементів кластера для обчислення даної задачі використаємо формулу:

$$M = \max_{i=1,k} \sum_{j=1}^{n} c_{ij} \tag{8}$$

В нашому випадку максимальної кількості елементів кластера рівна 4 (на 2-му ярусі (рис. 5).

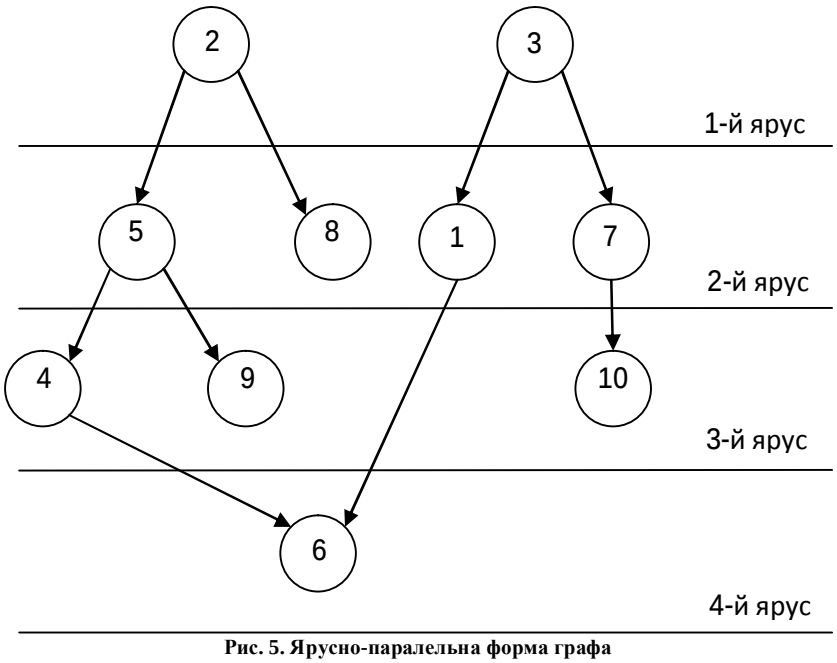

Рис. 5. Ярусно-паралельна форма графа

Одним з найважливіших етапів аналізу топології розподіленого обчислення є визначення критичних шляхів та елементів які входять до нього. Саме результати цього аналізу дають змогу розробнику визначити слабкі місця в системі та вжити заходи для їх усунення. Для цього необхідні наступні дані: результати представлення топології у ярусно-паралельній формі та час виконання задачі. Зауважимо, що при визначенні часу виконання задачі необхідно враховувати час передачі даних мережею. Представимо час виконання у вигляді:

$$T_b = T_O + T_p \tag{9}$$

де T_O – час обчислення, T_P – час передачі даних. При цьому необхідно враховувати, що час передачі даних може бути рівним 0. Наприклад, при виконанні задачі 7 немає сенсу виконувати задачі 10 на іншому кластері, так як відбудеться втрата часу на передачу результатів по мережі.

Для зручності обчислень припустимо, що об'єм даних, які є результатом обчислень максимально малий, а мережа має необмежену пропускну здатність. Таким чином T_P можна вважати рівним 0. Представимо час обчислення кожної задачі у вигляді матриці, де t_i – час виконання i-ї задачі:

$$T = \left| 10 \quad 1 \quad 9 \quad 3 \quad 2 \quad 4 \quad 1 \quad 12 \quad 4 \quad 3 \right|.$$

Визначення критичного шляху зводиться до виконання наступних дій:

1. Обчислення матриці часу роботи i-го ярусу.
2. Обчислення матриці часових сум i-го ярусу.
3. Обчислення сумарної матриці часів.
4. Визначення часу критичного шляху.

Час виконання першого ярусу визначається досить просто, так як час роботи його ніяким чином не залежить від інших ярусів, то $T_1 = T \otimes C_1$, де $\otimes$ – операція прямого множення матриць. Для інших ярусів необхідно спочатку знайти матрицю часових сум:

$$T_j^{Y} = (F^{K(T)})^T + F^{K(T_\Sigma)} \tag{10}$$

$$k \tag{11}$$

$$T_j = (\sup\{B_j\})^T \otimes C_j \tag{12}$$

Тут $F^{K(T)}$ – квадратна матриця, яка утворюється повторенням рядка Т, у нашому випадку:

$$F^{K(T)}=\begin{vmatrix}10&1&9&3&2&4&1&12&4&3\\10&1&9&3&2&4&1&12&4&3\\10&1&9&3&2&4&1&12&4&3\\10&1&9&3&2&4&1&12&4&3\\10&1&9&3&2&4&1&12&4&3\\10&1&9&3&2&4&1&12&4&3\\10&1&9&3&2&4&1&12&4&3\\10&1&9&3&2&4&1&12&4&3\\10&1&9&3&2&4&1&12&4&3\\10&1&9&3&2&4&1&12&4&3\end{vmatrix},$$

тоді:

$$T_1=T\otimes C_1=\begin{vmatrix}10&1&9&3&2&4&1&12&4&3\end{vmatrix}\otimes\begin{vmatrix}0&1&1&0&0&0&0&0&0&0\end{vmatrix}=$$
$$=\begin{vmatrix}0&1&9&0&0&0&0&0&0&0\end{vmatrix},$$

$$T_2^{Ч}=(F^{K(T)})^T+F^{K(T_1)}=\begin{vmatrix}10&11&19&10&10&10&10&10&10&10\\1&2&10&1&1&1&1&1&1&1\\9&10&18&9&9&9&9&9&9&9\\3&4&12&3&3&3&3&3&3&3\\2&3&11&2&2&2&2&2&2&2\\4&5&13&4&4&4&4&4&4&4\\1&2&10&1&1&1&1&1&1&1\\12&13&21&12&12&12&12&12&12&12\\4&5&13&4&4&4&4&4&4&4\\3&3&12&3&3&3&3&3&3&3\end{vmatrix},$$

$$B_2=T_2^{Ч}\otimes A=\begin{vmatrix}0&0&19&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0\\0&0&0&0&3&0&0&0&0&0\\0&3&0&0&0&0&0&0&0&0\\4&0&0&4&0&0&0&0&0&0\\0&0&10&0&0&0&0&0&0&0\\0&13&0&0&0&0&0&0&0&0\\0&0&0&0&4&0&0&0&0&0\\0&0&0&0&0&0&3&0&0&0\end{vmatrix},$$

$$T_2=(\sup\{B_2\})^T\otimes C_2=\begin{vmatrix}19\\0\\0\\3\\3\\4\\10\\13\\4\\3\end{vmatrix}\otimes\begin{vmatrix}1&0&0&0&1&0&1&1&0&0\end{vmatrix}=\begin{vmatrix}19&0&0&0&3&0&10&13&0&0\end{vmatrix},$$

$$T_3=\begin{vmatrix}0&0&0&6&0&0&0&0&0&13\end{vmatrix},$$

$$T_4 = \left|0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 23 \quad 0 \quad 0 \quad 7 \quad 0\right|,$$

$$T_\Sigma = \sum_{i=1}^{k} T_i = \left|19 \quad 10 \quad 9 \quad 6 \quad 3 \quad 23 \quad 10 \quad 13 \quad 7 \quad 13\right|,$$

$$T_L = \sup\{T_\Sigma\} = 23.$$

Критичний шлях даної топології дорівнює 23 часових одиниць.

Висновки

В роботі доведено можливість та доцільність використання принципів та підходів до аналізу і синтезу топологій для розв'язання задачі аналізу та синтезу топологій розподілених обчислень. Використання запропонованого підходу дозволяє автоматизувати даний процес, оскільки виключається необхідність виконання операцій людиною. На користувача може покладатись задача перетворення представлення топології у вигляді графа в матричну форму. Отже, доцільним є продовження досліджень у даному напрямі з метою отримання автоматизованого методу та засобів аналізу і синтезу топологій розподілених обчислень.

Література

1. Воеводин В.В. Параллельные вычисления. / Воеводин В.В., Воеводин Вл. БХВ-Петербург, 2002. – 608 стр.

2. Немнюгин С. Параллельное программирование для многопроцессорных вычислительных систем / Немнюгин С., Стесик О. – СПб.: БХВ-Петербург, 2002. – 400 с.

3. Богачев К.Ю. Основы параллельного программирования / К.Ю Богачев – М.: БИНОМ. Лаборатория знаний, 2003. – 342 с.

4. Эндрюс Г.Р. Основы многопоточного, параллельного и распределенного программирования. – М.: Издательский дом "Вильяме", 2003. – 674 с.

5. Дунець Р.Б. Аналіз та синтез топологій комп'ютерних видавничо-поліграфічних систем / Р.Б. Дунець. – Львів, 2003. – 174 с.

6. Нечепуренко М. И. Алгоритмы и программы решения задач на графах и сетях / Нечепуренко М. И., Попков В.К., Майнагашев С.М. – Новосибирск: Наука, Сиб. отд., 1990. – 217 с.

7. Берж К. Теория графов и ее применение / К. Берж. – М. Иностранная литература, 1962. – 180 с.

8. Вальковский В.А. Распараллеливание алгоритмов и программ. Структурный подход /А.В. Вальковский. – М.: Радио и связь, 1988. – 115 с.

9. Кормен Т. Алгоритмы: построение и анализ / Кормен Т., Лейзерсон Ч., Риверсон Р. – М.: МЦНМО, 2001. – 540 с.

Надійшла 8.5.2010 р.

УДК 519.21

А.Б. ГОРКУНЕНКО, С.А. ЛУПЕНКО, А.М. ЛУЦКІВ
Тернопільський національний технічний університет імені Івана Пулюя

МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ЕКОНОМІЧНИХ ЦИКЛІЧНИХ ПРОЦЕСІВ ДЛЯ ЇХ АВТОМАТИЗОВАНОГО АНАЛІЗУ ТА ПРОГНОЗУ

У роботі обґрунтовано нову математичну модель циклічних економічних явищ у вигляді циклічного випадкового процесу та розглянуто методи їх статистичного аналізу.

The mathematical model of cyclic economic phenomena in the form of the cyclic stochastic process is substantiated and methods of their statistical analysis are described.

Ключові слова: економічні цикли, моделювання, аналіз, прогноз, циклічний випадковий процес.

Вступ

Серед значної множини економічних явищ та процесів, особливе місце посідають економічні процеси, які мають циклічний, коливний характер. Зокрема, циклічну структуру мають такі економічні процеси, як індекси ділової активності усіх галузей економіки, валовий національний продукт країн, сезонні індекси доходів підприємств і т.д. На рис. 1. надано декілька реалізацій типових економічних циклічних процесів.