Діагностування Обчислювальних Системах із Загальною Пам'яттю

Олександр Малько
кафедра інформаційно-телекомунікаційних технологій та систем
Івано-Франківський національний технічний університет нафти і газу
Івано-Франківськ, Україна
malko.pochta @.gmail.com

Diagnostics of Computer Systems with Shared Memory

Oleksandr Malko
department of information and telecommunication technologies and systems
Ivano-Frankivsk National Technical University of Oil and Gas
Ivano-Frankivsk, Ukraine
malko.pochta @.gmail.com

Анотація **–На теперішній час ведеться велика кількість як теоретичних досліджень у цій галузі, так і реалізацій практичних інструментаріїв для розробки розподілених алгоритмів, заснованих на мережах Петрі. Розробляється міжнародний стандарт мереж Петрі. Разом з тим, відчувається необхідність розвитку формалізму з метою більш адекватного та зручного представлення систем зі складною структурою. Сучасні системи часто є мультиагентними і мають ієрархічну, багаторівневу структуру. У зв'язку з цим останнім часом ведуться дослідження щодо розширення формалізму мереж Петрі за рахунок ідей об'єктно-орієнтованого підходу з метою отримання моделей, що явно відображають ієрархічну та мультиагентну структуру системи.**

Abstract **– Currently, there is a large amount of both theoretical research in this field and the implementation of practical tools for the development of distributed algorithms based on Petri nets. An international standard for Petri nets is being developed. At the same time, there is a need to develop formalism in order to more adequately and conveniently present systems with a complex structure. Modern systems are often multi-agent and have a hierarchical, multi-level structure. In this regard, research has recently been conducted on extending the formalism of Petri nets due to the ideas of an object-oriented approach in order to obtain models that clearly reflect the hierarchical and multi-agent structure of the system.**

Ключові слова: паралельне програмування, мережі Петрі, примітиви синхронізації, відсутність глухих кутів, математичне програмування

Keywords: parallel programming, Petri nets, synchronization primitives, absence of dead ends, mathematical programming.

I. Вступ

Обчислювальні системи дуже складні, часто великі, включають безліч взаємодіючих компонент. Кожна компонента також може бути дуже складною, оскільки взаємодіє з іншими компонентами системи. Це справедливо і для багатьох інших систем. Економічні системи, юридичні системи, системи управління дорожнім рухом, хімічні системи складаються з багатьох окремих компонент, що взаємодіють один з одним складним чином.

Отже, незважаючи на різноманітність модельованих систем, виділяється декілька загальних рис, які мають бути відбиті в особливостях використовуваної моделі цих систем. Основна ідея полягає в тому, що системи складаються з окремих взаємодіючих компонент. Кожна компонента сама може бути системою, але її поведінку можна описати незалежно від інших компонент системи, за винятком точно певних взаємодій з іншими компонентами. Кожна компонента має свій стан. Стан компоненти - це абстракція відповідної інформації, необхідної для опису її (майбутніх) дій. Поєднана природа дій в системі створює деякі труднощі при моделюванні. Оскільки компоненти системи взаємодіють, потрібне встановлення синхронізації. При недостатньому використанні примітивів синхронізації є ризик отримати програму, яка має у якомусь вигляді проблеми із синхронізацією. Засоби динамічного аналізу паралельних програм не завжди можуть вирішити ці проблеми. Для низки програм потрібен статичний аналіз паралельних алгоритмів. Найбільш складною проблемою є діагностування можливості глухих кутів.

У даній роботі для діагностування глухих кутів в алгоритмах застосовується метод перетворення алгоритму в еквівалентну мережу Петрі з подальшим її аналізом. Як вступ у мережі Петрі можуть розглядатися класичні роботи [1] та [2].

II. Мережі Петрі як інструмент аналізу обчислювальних систем

Основним напрямком застосування мереж Петрі моделювання функціонування причинно-наслідкових зв'язків і системах.

Значна кількість робіт присвячена дослідженню динамічних властивостей обчислювальних систем за допомогою апарату мереж Петрі.

У роботі [3] досліджується застосування мереж Петрі до моделювання обчислювальних систем з пам'яттю, що розділяється. Досить багато уваги приділяється розширеним мережам Петрі з додаваннями у вигляді дуг, що забороняють, переходам з ненульовим часом, проміжних дуг. Пропонується система, звана SOM, заснована мовою ADA для моделювання та верифікації паралельних процесів.

У роботі [4] пропонується оригінальний математичний апарат для ефективного виявлення відсутності глухих кутів у мережах Петрі. Використовується механізм сифонів та пасток та методи математичного програмування.

Ряд робіт присвячено моделюванню багатопотокових додатків з використанням мереж Петрі. Це, наприклад, робота [5], де розглядається оптимізація багатопоточних додатків на багатопроцесорних обчислювальних системах (використовуються мережі Петрі з ненульовим часом переходу).

У роботі [6] розглядається верифікація паралельних програм із використанням мереж Петрі як графа доказів.

Робота [7] розглядає підмножину мереж Петрі — CT-мережі (Casual–Time nets) — для моделювання автоматів паралельного виконання та пропонує засоби для побудови паралельних додатків та контрольованим потоком виконання.

Робота [8] використовує механізм інваріантів моделі мережі Петрі для визначення наступних помилок синхронізації в програмах: глухих кутів, нескінченних внутрішніх циклів і втрати повідомлень при передчасному завершенні. У [9] також пропонується використати механізм інваріантів, але дослідження не просувається досить далеко.

Роботи [10] і [11] пропонують моделі мереж Петрі для основних примітивів синхронізації, але не пропонують методів доказу відсутності помилок синхронізації в програмах, що моделюються.

Аналіз моделей систем на базі мереж Петрі зводиться до аналізу їх властивостей.

Безпека. Одна з найважливіших властивостей мереж Петрі, яка повинна моделювати реальний пристрій, — безпека. Позиція мережі Петрі є безпечною, якщо число фішок в ній ніколи не перевищує 1. Мережа Петрі безпечна, якщо безпечні всі позиції мережі.

Позиція $p_i \in P$ мережі Петрі $C = (P, T, I, O)$ з початковим маркуванням μ є безпечною, якщо $\mu'(p_i) \le 1$ для будь-якої $\mu' \in R(C, \mu)$. Мережа Петрі безпечна, якщо безпечна кожна її позиція. Якщо інтерпретувати мережі як умови і події, маркування кожної позиції повинна бути безпечною.

Структура мережі Петрі визначається її позиціями, переходами, вхідними та вихідними функціями (відображеннями) тобто кортежем множин $C = (P, T, F, I, O, \mu_0)$. Для структурного опису математичної моделі мережею Петрі застосовуються такі позначення і відповідності.

Початковий стан мережі Петрі задається за допомогою маркування її позицій $\mu_0 : P \to N$, яке полягає у наданню кожній позиції мережі деяке число маркерів. Присутність маркера у позиції трактується як виконання деякої умови. Спрацювання переходів міняє розмітку сітки у відповідності з правилами спрацювання переходів. Отже, розмітку μ_0 можна представити вектором з $|P|$ занумерованих елементів, у якому *i*-тий елемент дорівнює кількісті маркерів у і-тій позиції. У загальному випадку кількість маркерів у вузлі може бути більшою за одиницю.

Перехід дозволений якщо кількість маркерів у його вхідних позиціях не менше кількості дуг які йдуть від кожної вхідної позиції до переходу.

Нове маркування μ' здійснюється шляхом видалення маркерів з вхідних позицій переходу з наступним розміщення маркерів у кожній вихідній позиції по одному маркеру на кожну дугу. Це можна представити як функцію наступного стану $\mu' = \delta(\mu, t_j)$ з маркуванням μ і переходом $t_j \in T$, який можливий тоді і тільки тоді, коли $\mu(p_i) \ge \#(p_i, I(t_j))$ для всіх $p_i \in I(t_j)$. Тоді якщо $\delta(\mu, t_j)$ визначена, то $\mu' = \delta(\mu, t_j)$ де $\mu'(p_i) = \mu(p_i) - \#(p_i, I(t_j)) + \#(p_i, O(t_j))$ для всіх $p_i \in I(t_j)$.

Розширення запуску позначається як $\mu'(\sigma)$, де σ - послідовність переходів, що переводять μ у μ'. Множина усіх маркувань, досяжних з μ_0, позначається як $R(\mu_0)$.

Мережа Петрі також може задаватися за допомогою матриці інцидентиості $C = [c_{ij}]$ для якої

$$c_{ij} = \begin{cases} 1, \text{ якщо } t_j \in I(p_i) \cap O(p_i); \\ -1, \text{ якщо } t_j \in I(p_j) \cap O(p_i) \wedge i \ne j; \\ 0, \text{ у іншому випадку.} \end{cases}$$

Обмеженість. Позиція є k- безпечною або k-обмеженою, якщо кількість маркерів в ній не може перевищувати ціле число k .Слід Відмітимо, що межа k' по числу маркерів, які можуть знаходитися в позиції, може бути функцією від позиції (наприклад, позиція p_i може бути 3-безпечною, тоді як позиція p_2 — 8-безпечною).

Збереженість. Мережа Петрі $C=(P,T,I,O)$ з початковою маркуванням μ називається такою, що суворо зберігає, якщо для всіх $\mu' \in R(C,\mu)$:

$$\sum_{p_i \in P} \mu'(p_i) = \sum_{p_i \in P} \mu(p_i).$$

Мережі Петрі можуть застосовуватись у багатьох додатках — для моделювання поведінки складних недетермінованих комплексів, для генерації машинного коду в компіляторах та для колосальної кількості різноманітних завдань. У цьому випадку мережі Петрі будуть застосовуватися для вирішення досить часткового завдання — виявлення наявності тупиків у багатопотоковому обчислювальному середовищі із спільними ресурсами. Отже, потрібно визначення однієї характеристики — активності (liveness).

Якщо перехід у мережі Петрі ніколи не може бути запущений, такий стан називається глухим кутом. Виявлення можливості настання стану глухого кута в мережі Петрі показує наявність можливості стану глухого кута і в модельованому обчислювальному середовищі.

Для аналізу глухих кутів у мережі з маркуванням застосовується наступна класифікація:

Для аналізу глухих кутів у мережі з маркуванням застосовується наступна класифікація:

– активність рівня 0 має перехід t_j, який ніколи не може бути запущений;

– активність рівня 1 має перехід t_j, який потенційно запустимий, тобто існує таке допустиме і досяжне маркування $\mu' \subset R(C,\mu)$, у якому цей перехід дозволено;

– активність рівня 2 має перехід t_j якщо для всякого цілого n існує послідовність запусків t_j, у якій t_j присутнє принаймні n разів.

– активність рівня 3 має перехід t_j, якщо існує нескінченна послідовність запусків, у якій t_j присуне необмежену кількість разів;

– активність рівня 4 має перехід t_j, якщо для будь-якого $\mu' \subset R(C,\mu)$, існує така послідовність σ запусків що t_j дозволений у $\delta(\mu',\sigma)$.

Наявність у мережі переходів з активністю 0 з певністю підтверджує наявність глухих кутів в моделюється системі.

Мережа Петрі не містить глухих кутів, якщо для будь-якого досяжного маркування є принаймні один дозволений перехід. Позиція p обмежена, якщо існує така константа k, що $\mu(p) \le k$ для всіх $\mu \in R(\mu_0)$. Мережа Петрі називається обмеженою, якщо всі позиції мережі обмежені.

Підмножина позицій $S \subseteq P$ називається сифоном (syphon), якщо будь-який вхідний перехід в S є також вихідним переходом у S, тобто $O(S) \subseteq I(S)$. Відповідно є пастка S (trap), якщо $I(S) \subseteq O(S)$. Сифон (пастка) називається мінімальним, якщо він не містить інших сифонів (пасток).

Існує кілька основних методів підходу до аналізу активності мереж Петрі. У цій роботі використовується перетворення рівнянь мережі Петрі на рівняння для методу математичного моделювання.

Процес формування коректного алгоритму паралельної обробки та верифікації складається з низки кроків, які можуть бути описані таким чином:

1. Розробка моделі алгоритму.
2. Представлення моделі алгоритму мовою мереж Петрі.
3. Верифікація побудованої моделі Петрі та повернення до першого пункту для уточнення моделі алгоритму -

Для представлення моделі на мову мереж Петрі потрібно дати відображення основних конструкцій вхідної моделі на вихідну мову. У цій роботі застосовуються примітиви синхронізації двох типів:Mutex та Event.

Примітив типу Mutex є засобом взаємовиключення обчислювальних потоків і включає наступні методи:

– WaitAndLock — очікувати на звільнення з негайним захопленням

– ˆRelease – звільнення.

Mutex має атрибут власника, тобто, властивість приналежності обчислювальному потоку, що його захопив. Цей примітив потребує підтримки операційною системою, може призвести до контекстного перемикання потоків і не переводить обчислювальне ядро у стан активного очікування. Примітив Mutex можна подати у вигляді моделі мережі Петрі (рис.1).

Наявність маркера у позиції P_1 показує, що Mutex знаходиться у вільному стані. Ця позиція повинна використовуватися кількома обчислювальними потоками. З появою маркера в позиції P_0 та наявності маркера в позиції P_1 стає можливим перехід t_0, маркер у позиції P_1 тепер відсутній, що унеможливлює здійснення переходів, які залежать від наявності маркера в позиції P_1.

Після захоплення маркера обчислювальний потік виконує код критичної секції (позиція P_2), виконується перехід t_1, який передає маркер у позицію P_1, роблячи цим можливим здійснення пов'язані з наявністю маркера у цій позиції переходів.

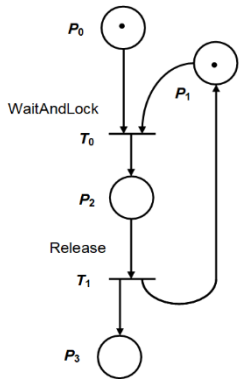

Рис. 1. Схема мережі Петрі примітиву Mutex.

Примітив типу Event використовується для встановлення факту виникнення будь-якої події, може перебувати у двох станах — сигнальному та несигнальному і включає наступні методи:

- Set - встановити об'єкт у сигнальний стан; ˆ
- Reset – встановити об'єкт у несигнальний стан; ˆ
- Wait – очікувати настання сигнального стану.

На відміну від об'єктів типу Mutex об'єкти типу Event не мають атрибуту власника. Тут розглядається варіант об'єктів типу Event, що потребує ручного переведення стану в несигнальний після того, як рівно один з числових потоків отримав доступ до нього. Примітив Event поряд з примітивом Mutex вимагає підтримки операційною системою, що може призвести до перемикання контексту обчислювальних потоків і не переводить обчислювальне ядро в стан активного очікування.

Тут враховується наступний факт: якщо примітив Event знаходиться у сигнальному стані, то наступні виклики методів Set не роблять на це стан впливу. Якщо примітив Event знаходиться в несигнальному стані, то наступні виклики методу Reset не на цей стан. Дані умови призводять до більш точної та більш складної моделі, ніж у роботах [10] та [11].

Мережа Петрі для примітива Event представлена на рис. 2. Позиція P_0 є вхід методу Set. Якщо примітив Event вже знаходиться у стані «сигнал» (що визначається наявністю маркера в позиції P_1), то операція розміщення маркера на позицію P_0 буде еквівалентна порожній операції. Якщо ж цей примітив не перебуває в стані сигналу, то операція розміщення маркера в позицію P_0 наводить примітив у стан сигнал.

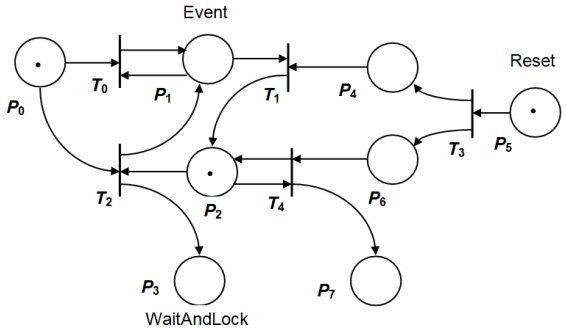

Рис. 2. Схема мережі Петрі примітиву Event

Позиція P_5 являє собою вхід методу Reset. Розміщення маркера в даній позиції перекладає примітив Event в стан «сигнал не встановлений» незалежно від первісного стану примітиву.

Виходом методу Reset є позиція P_7 - поява маркера в даній позиції сигналізує про завершення роботи методу.

Якщо у позиції P_3 вже є маркер, то метод Wait завершується успіхом.

Висновки

У роботі вирішувалося завдання визначення застосовності алгоритму з реалізації пулу обчислювальних потоків з використанням механізму мереж Петрі. Було вперше застосовано уточнену модель мережі Петрі примітив синхронізації Event, що враховує сучасну семантику даного примітиву. За мовою опису моделі була синтезована еквівалентна мережа Петрі. Аналіз еквівалентної мережі Петрі методом математичного програмування дозволив встановити, що ця мережа, отже й оригінальний алгоритм немає тупиків.

Література References

[1] James l. Peterson Petri net theory and the modeling of systems, Prentice Hall; 1St Edition (January 1, 1981), 290 pages.

[2] Murata TadaoPetri Nets: Properties, Analysis and Applications // Proceedings of the IEEE. - 1989. - V. 77, N 4.

[3] Vallejo F., Gregorio JA, Gonzalez Harbour M., Drake JMShared Memory Multiprocessor operating System with Extended Petri Net Model // IEEE transactions on parallel and distributing systems. - 1994. - V. 5, N 7, July.

[4] Feng Chu and Xiao-lan XieDeadlock Analysis of Petri Nets За допомогою Siphons і Mathematical Programming, // IEEE Transactions of Robotics and Automation. 1997. - V. 13, N. 6, December.

[5] Govindarajan F., Suciu WM, ZuberekTimed Petri NetModels з Multithreaded Multiprocessor Architectures // IEEE Preceedings if the 7th International Workshop на Petri Nets and Performance Models. - Saint Malo, June, 1997.

[6] Takaoka TadaoA Systematic Approach до Parallel Verification// Department of Computer Science of University of Ibaraki, August, 1995.

[7] Pommereua F.Petri Nets як Executable Specifications of High-Level Timed Parallel Systems. - 2005.

[8] 8.Bruce P. LesterDetection of Control Flow Errors in parallel Programs at Compile Time // International Journal of Distributed and parallel Systems (IJDPS). - 2010. - V. 1, N 2, November.

[9] 9.Padidar S.Parallel Program verification: A Brief Introduction, January, 2010.

[10] Kavi KM, Moshtaghi A., Deng-Jyi ChenModeling Multithreaded application using Petri nets, // International Journal of Parallel Programming. - 2002. - V. 30, Iss 5. - P. 1-23, October.

[11] Kavi KM, Bukhles PB, Bhat UNIsomorphism Between Petri net and Dataflow Graphs // IEEE Transactions on Software Engineering. - 1987. - V. SE-13, N 10.