УДК 004.94

П.Ю. Катін, Є.В. Досенко, Р.Б. Іванів

МОДЕЛЮВАННЯ ПРОГРАМ ІЗ ПАРАЛЕЛЬНИМИ ОБЧИСЛЕННЯМИ ЗА ДОПОМОГОЮ МЕРЕЖ ПЕТРІ

У статті розкрито питання проектування програм з паралельним обчисленням та методи синхронізації в багатопоточних програмах, які значно пришвидшують аналіз економічних процесів. Для наочного подання об'єктів синхронізації був вибраний засіб графічного моделювання мережі Петрі. Розглянуто основні системні об'єкти синхронізації та ряд базових задач, які вирішуються за допомогою цих об'єктів.

The article is about the design of applications with parallel computing and synchronization in multithreaded applications, which considerably speed up the analysis of economic processes. The Petri nets was selected as mathematical tool for illustrative representation of synchronization

objects. We discussed the main system objects and some basic synchronization problems which can be solved using this objects.

Ключові слова: мережі, Петрі, паралельне, обчислення, об'єкти, синхронізація, економіка.

Сучасний «інформаційний» етап розвитку економіки характеризується величезним обсягом інформації, яка доступна для аналізу різними способами та алгоритмами. Також реалізується встановлення причинно-наслідкових зв'язків та інтерпретація отриманих результатів. Паралельні обчислення полегшують задачі аналізу та обробки даних завдяки суттєвому прискоренню розрахунку параметрів.

Використання паралельних обчислень для біржової торгівлі дозволяє комп'ютерним програмам (торговим роботам) самостійно відслідковувати дані з декількох індексів на фондових біржах, оптимізовувати торгові стратегії, які будуються на основі аналізу цінових рядів за допомогою великої кількості індикаторів. Під час цього здійснюються мільйони операцій відповідно до поведінки цінового ряду, торгової стратегії та значення індикаторів за максимально короткий проміжок часу [1].

За проектування багатопоточних програм виникає необхідність синхронізації потоків у зв'язку зі спільним використанням апаратних чи програмних ресурсів [2]. Синхронізація необхідна для виключення можливості виникнення стану гонки чи тупиків при обміні даними між потоками, доступу до спільних даних чи інших спільних ресурсів.

У багатьох операційних системах (ОС) реалізована багатопоточність. Можна надати кожному потоку свою задачу, що дозволить виконувати декілька програм паралельно. Щоб виконувати операції паралельно та незалежно одна від одної, бажано використовувати декілька процесорів. Сьогодні, коли існує тенденція збільшення кількості процесорних ядер в одній обчислювальній системі, задача синхронізації є особливо актуальною. Під час розробки програмного продукту (ПП) розпаралелення задач треба виконувати дуже обережно. Необхідно враховувати той факт, що в економіці, як і в будь-якій іншій сфері, виникають помилки при розробці ПП. Помилки при розробці багатопоточних програм важко виявляються, оскільки вони мають нерегулярний характер. Особливо небезпечні для економічних ПП приховані помилки, що не виявлені під час налагодження. Отже, потрібна правильна синхронізація потоків, яку складно забезпечити без наочної моделі і без математичної обробки.

Вищезгадані задачі зручно вирішувати за допомогою мереж Петрі.

Мережі Петрі (МП)— один із найефективніших засобів графічного і математичного моделювання систем різних класів [3]. Аналіз мереж Петрі дозволяє отримати інформацію про структуру і динамічну поведінку системи, що моделюється.

Використання мереж Петрі при моделюванні багатопоточних програм доцільно, оскільки в мережах Петрі моделювання відбувається на подієвому рівні. Під час моделювання визначаються дії, які відбуваються в системі. Визначаються стани, що передували цим діям, виявляється в який стан перейде система після виконання певних дій. Отже, мережі Петрі описують поведінку систем, а дослідження результатів моделювання дозволяє визначити в яких станах перебувала чи не перебувала система, а також які стани взагалі не досяжні для системи.

Існує багато варіантів опису мереж Петрі. Використаємо визначення мереж Петрі, що реалізовано у Тадао Мурата (Tadao Murata) [4]. Отже, мережею Петрі називається п'ятірка, $PN = (P, T, F, W, M_0)$, де $P = \{p_1, p_2, ..., p_m\}$ – скінченна множина позицій, $T = \{t_1, t_2, ..., t_m\}$ – скінченна множина переходів, $F \subseteq (P \times T) \cup (T \times P)$

– множина дуг, $W : F \to \{1,2,3,...\}$ – функція ваги, а $M_0 : P \to \{0,1,2,3,...\}$ – початкове маркування, причому $P \cap T = \varnothing$ і $P \cup T \neq \varnothing$.

Зважаючи на вивченість МП і широкий спектр їх застосування, введемо певні обмеження на математичний апарат МП. Модель, побудована на базі класичної, нефарбованої, однорівневої МП. Передбачена обов'язкова початкова розмітка МП. Моделі, побудовані на базі таких МП доцільно використовувати для представлення економічних процесів [3; 4; 5].

Як об'єкт моделювання оберемо спеціальні об'єкти ядра ОС. Їх набір залежить від конкретної ОС, яка створює ці об'єкти за запитом процесів. Основними об'єктами синхронізації є м'ютекс, семафор та подія [6].

Семафор – це об'єкт ядра ОС, який можна використовувати для керування доступом потоків до спільних ресурсів. Семафор можна розглядати як лічильник, який містить ціле число в діапазоні від 0 до n, де $n \in N$, n – максимальне значення семафору. При досягненні семафором значення 0 він переходить у несигнальний стан, при будь-яких інших значеннях семафора – його стан сигнальний. Семафор дозволяє одночасний доступ кількох потоків до спільних ресурсів.

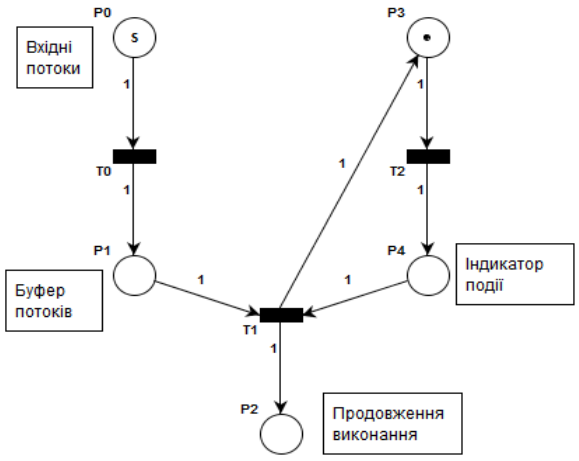
Для доступу до семафора використовуються дві атомарні операції: P (від датського слова proberen – перевіряти) і V (від verhogen – збільшувати). Нехай змінна Semaphore представляє собою семафор. Тоді дія V (Semaphore) буде збільшувати значення Semaphore на 1. Причому операція P є атомарною, а, отже, операція вибірки, інкременту і зберігання значення Semaphore не може бути перервана. Дія P (Semaphore) спочатку перевіряє значення Semaphore і, якщо Semaphore більше нуля, то виконається віднімання одиниці від Semaphore. Якщо ж Semaphore дорівнює або менше нуля, то потік заблокується до того часу, поки Semaphore не стане більше нуля. Оскільки операція V є атомарною, то успішна перевірка і декремент значення Semaphore є неподільною операцією. Під час виконання операцій P і V недоступні жодні переривання [7].

М'ютекс (англ. mutex, від англ. mutual exclusion – об'єкт взаємного виключення) – це об'єкт ядра ОС, який надає доступ до спільних ресурсів не більше, ніж одному потоку в будь-який момент часу. М'ютекс є частковим випадком семафора, а саме двійковим семафором, значення якого можуть бути лише 0 та 1. М'ютекс може знаходитись в одному з двох станів – сигнальному та несигнальному. Коли який-небудь потік хоче отримати доступ до спільних ресурсів і захоплює м'ютекс, м'ютекс переводиться в несигнальний стан (і перебуває в ньому до моменту звільнення м'ютекса, тобто до моменту, коли потік не закінчить роботу з спільними ресурсами), що не дозволяє іншим потокам захопити його і отримати доступ до ресурсів у цей час. Ці потоки очікують звільнення м'ютекса.

Подія – це простий метод синхронізації, сутність якого полягає в передачі сигналу потоку, який очікує виконання деякої події, що можна розпочинати роботу. Є два типи подій: з ручним керуванням та з автоскиданням. Якщо подія з автоскиданням, то за сигналом події найближчий потік, що чекає, вивільняється, а подія автоматично скидається назад у нейтральний стан. Коли ж подія зі ручним керуванням, то як тільки подія встановиться в стан сигналу, вивільняються всі потоки, що чекають, а повернути подію в нейтральний стан можна лише вручну.

Моделювання об'єктів синхронізації за допомогою мереж Петрі. На основі теоретичного матеріалу [2; 3; 4; 8] нами було запропоновано своє бачення моделей подій із автоскиданням та з ручним керуванням.

На рис. 1 зображено модель події з автоскиданням. Спочатку всі потоки знаходяться в початковому стані (P0). Під час виконання програми потоки починають накопичуватися у буфері потоків (P1). У цьому стані вони чекають поки не відбудеться подія. Коли відбувається подія в стані індикатор події (P4) з'являється мітка, яка сигналізує про те, що можна вивільнити з черги один потік. Після вивільнення одного потоку індикатор події стає в нейтральний стан, а потоки в буфері продовжують чекати, поки не відбудеться нова подія. На рис. 1 реалізована наступна формалізація: накопичення буфера подій (T0); виклик події (T2); звільнення одного потоку та скидання індикатора подій (T1).



,

де S – кількість вхідних потоків.

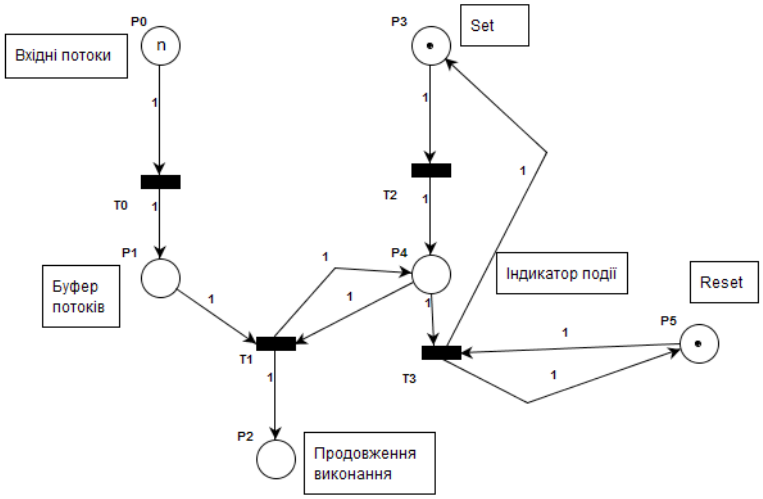
Рис. 1. Подія з автоскиданням (авторська розробка)

На рис. 2 зображено модель події з ручним керуванням. Вхідні потоки починають накопичуватися в буфері потоків (P1) та чекають там, поки не відбудеться подія. Для виникнення події використовується перехід T2 і при цьому стан P3 втрачає мітку, а індикатор події (P4) отримує мітку, яка сигналізує про те, що подія відбулася і всі потоки в буфері можуть продовжувати виконання, тобто перейти в стан P2. Після того, як всі потоки вивільняться, можна виконати дію Reset для того, щоб повернути індикатор події в нейтральний стан.

Зауваження. Виконувати дію Reset (T3) можна лише тоді, коли буфер потоків пустий. На рис. 2 реалізована така формалізація: накопичення буфера подій (T0); виклик події (T2); скидання події (T3); звільнення всіх потоків (T1).

На рис. 3 показано модель м'ютекса. Практика розробки і налагодження програм показала, що для формалізації роботи м'ютекса його модель у вигляді мережі Петрі зручно представити як це показано на рис. 3. У моделі маємо два переходи. Перехід

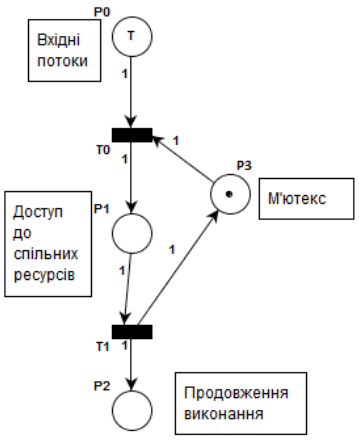
T0 формалізує очікування звільнення м'ютекса з негайним захопленням (атомарна операція). Перехід T1 формалізує звільнення м'ютекса.



,

де n — кількість вхідних потоків.

Рис. 2. Подія з ручним керуванням (авторська розробка)



,

де T — кількість вхідних потоків (ємність обмежується доступною пам'яттю)

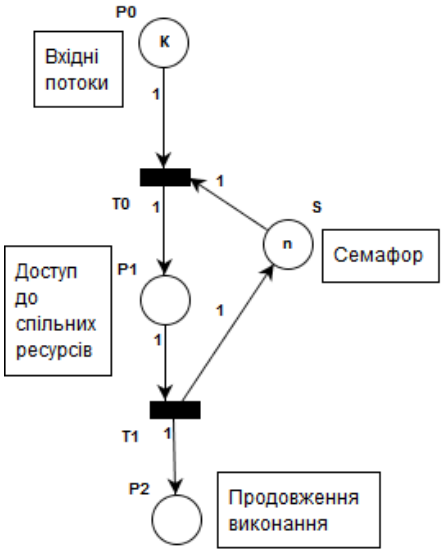
Рис. 3. М'ютекс (авторська розробка)

Наявність мітки в позиції P3 вказує на те, що м'ютекс знаходиться у вільному стані. За умови наявності вхідного потоку (мітка в позиції P0) і вільного стану м'ютекса, стає можливим перехід T0, тобто захоплення м'ютекса. Оскільки після виконання переходу T0 мітки в позиції P3 не буде, то подальші потоки будуть вимушені чекати звільнення м'ютекса для активації переходу T0. Після захоплення м'ютекса потік

виконує операції над спільними ресурсами (P1). Після цього виконується перехід T1, який передає мітку в позицію P3, тобто звільняє м'ютекс.

Практика розробки і налагодження програм показала, що для формалізації роботи семафору його модель у вигляді мережі Петрі зручно представити як це показано на рис. 4.

Семафор є логічним розширенням м'ютекса: м'ютекс може приймати лише значення 1 або 0, а семафор може мати значення більше одиниці. Семафор дає доступ до спільних ресурсів кільком потокам одночасно. Переходи T0 та T1 відповідають операціям над семафором P(S) і V(S) відповідно. Алгоритм роботи семафора аналогічний алгоритму роботи м'ютекса за виключенням того, що доступ до спільних ресурсів (P1) може надаватись кільком потокам одночасно. Їхня кількість обмежується максимальним значенням семафору.



,

де T — кількість вхідних потоків; n — максимальне значення семафора (0..32767).

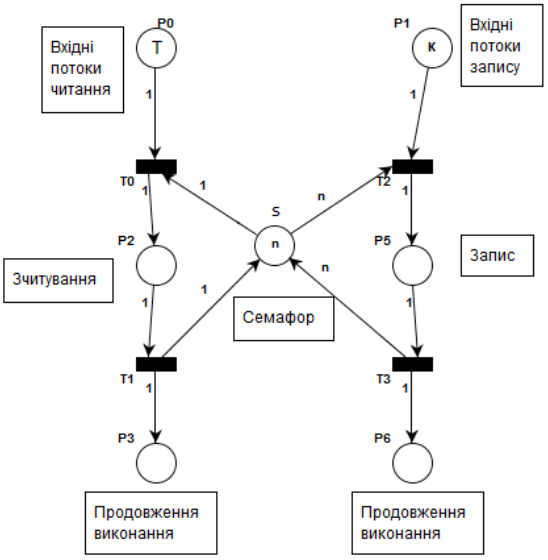
Рис. 4. Семафор (авторська розробка)

Нижче розглянуті типові задачі, синхронізація потоків в яких досягається за допомогою семафорів. Такі задачі часто виникають при моделюванні паралельних економічних процесів.

Насамперед розглянемо поширену задачу взаємодії потоків двох типів: читання та запису. Всі потоки використовують спільний ресурс (файл, змінну і т. д.), причому потоки читання не можуть змінити дані, а потоки запису можуть. Для вирішення даної задачі необхідно синхронізувати потоки таким чином, щоб один потік запису повністю виключав можливість доступу інших потоків до спільного ресурсу під час роботи. При цьому, декілька потоків читання можуть мати доступ до спільних даних одночасно.

На рис. 5 показано модель системи, в якій взаємодіють потоки читання та запису за умови, що кількість потоків читання не більше n. При необмеженій кількості потоків читання одночасно зможуть лише n потоків.

Проаналізувавши модель видно, що при виконанні потоку запису в позиції P4 не залишається жодної мітки, що, своєю чергою, блокує запуск інших потоків до того моменту, поки потік запису не поверне n міток в позицію S. При виконанні хоча б одного потоку читання у позиції S буде недостатня кількість міток для активації переходу T2, тобто для запуску потоку запису. Після завершення роботи всіх потоків читання у позиції S знову буде n міток, що дозволить запуск потоків запису.



де T — кількість потоків читання; K — кількість потоків запису; n — максимальне значення семафора.

Рис. 5. Задача читання — запису (авторська розробка)

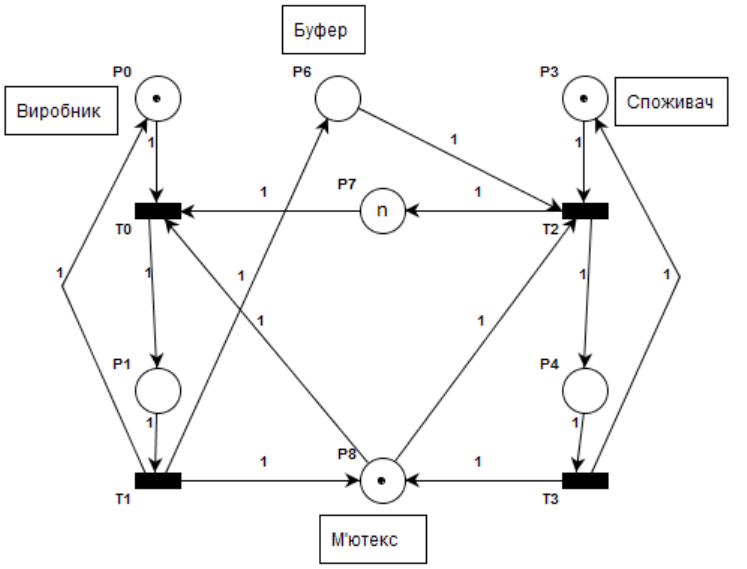
Для уявлення паралелізму, тобто для паралельного виконання потоків, будемо використовувати два оператори, як пропонує Е. Дейкстра [1].

```
program Read/Write;
var T: Semaphore, K: Semaphore, S: Semaphore;
procedure Read();
begin; P(S); P(T); ReadData; V(S); V(T); end;
procedure Write(integer S); integer i = S; integer j = S;
while i>0 do begin: P(S); i--; end;WriteData;
while j>0 do begin: V(S); j--; end;
begin; parbegin; Read();Write(S); parend; end.
```

Іншою типовою задачею, яка вимагає організації взаємодії потоків, є так звана задача виробник — споживач. Сутність задачі полягає у тому, що два потоки взаємодіють між собою через буфер обмеженого розміру. Виробник закладає інформацію в буфер, а споживач забирає її звідти. Якщо буфер заповнений, то виробник повинен

чекати, доки в ньому з'явиться місце, щоб закласти туди нову порцію інформації. Якщо ж буфер порожній, то споживач повинен очікувати надходження інформації в буфер. Розглянувши рішення цієї задачі, яке запропонував Пітерсон Дж. [3], було виявлено деякі недоліки та запропоновано власне рішення задачі виробник — споживач.

На рис. 6 показана мережа Петрі, яка відображає розглянуті дії потоків. Стан буферу описують дві позиції: P6 вказує на кількість блоків інформації, розміщених у буфері, але ще не використаних, а P7 показує кількість незайнятих місць у буфері. Спочатку позиція P6 немає міток взагалі, а позиція P7 має n міток (розмір буфера). Якщо буфер буде заповнений повністю, то в позиції P6 буде n міток, а в позиції P7 буде 0 міток. Проаналізувавши роботу Пітерсон Дж., зроблено висновок, що оскільки операції «закладання» і «вийманння» інформації з буферу не є атомарними, то необхідно передбачити механізм взаємовиключення для них, оскільки їхній перетин може призвести до помилок у роботі програми. Цю важливу деталь не передбачив вищезазначений автор. Для реалізації механізму взаємовиключення потоків застосуємо м'ютекс (P8). З наведеної моделі видно, що потік-виробник не має доступу до заповненого буферу, а потік-споживач не може взаємодіяти з порожнім буфером.



,

де n — розмір буфера.

Рис. 6. Задача виробник — споживач (авторська розробка)

program *Producer — Consumer;*
var *Mutex: Semaphore, ItemsIn: Semaphore, Free_places: Semaphore;*
procedure *Producer()*
begin: *P(Mutex); P(Free_places); Put_Item; V(Mutex); V(ItemsIn);* end.

procedure *Consumer()*
begin: *P(Mutex); P(Free_places); Get_Item; V(Mutex); V(ItemsIn);* end.
begin: *Mutex = 1; ItemsIn = 0; Free_places = N; <Розмір буфера>*
parbegin; *Producer(); Consumer();* parend; end.

Таким чином, у статті розглянуто задачу синхронізації потоків, яка дозволяє значно пришвидшити процес аналізу та обробки економічних даних завдяки використанню паралельних обчислень. Це, своєю чергою, дозволяє створювати ефективні програми, наприклад, для біржової торгівлі. Вказано на головні проблеми в проектуванні багатопоточності, такі як виникнення стану гонки чи тупиків при обміні даними між потоками, доступі спільних ресурсів. Для уникнення вищенаведених проблем щодо проектування багатопоточних програм було запропоновано використовувати мережі Петрі. Розглянуто аспекти в моделюванні мережами Петрі та основні об'єкти синхронізації: подія з ручним керуванням, подія з автоскиданням та семафор. Для наглядної демонстрації роботи об'єктів синхронізації побудовано їхні моделі, створені за допомогою мереж Петрі. У цілому, розглянуто проблеми проектування багатопоточності та запропоновано шляхи їх вирішення.

1. *Артемьев С.С.* Математическое и статистическое моделирование на фондовых рынках / С.С. Артемьев, М.А. Якунин; под ред. Г.А. Михайлова. — Новосибирск: ИВМИМГ, 2003. — 158 с.; 2. Многопоточность и синхронизация / А. Курзенков: [Электронный ресурс]. — Режим доступа: http://www.kurzenkov.com/Articles/multithreading1.html; 3. *Питерсон Дж.* Теория сетей Петри и моделирование систем / Дж. Питерсон; пер. с анг. М. Горбатова, В. Торхова, В. Черверикова. — М.: Мир, 1984. — 264 с.; 4. Petri Nets: Properties, Analysis, and Applications / Tadao M.: [Electronic resourse]. — Mode of access: http://embedded.eecs.berkeley.edu/Research/hsc/class.F03/ee249/discussionpapers/PetriNets.pdf; 5. *Федотов И.Е.* Некоторые приемы параллельного программирования. — М., Изд-во МГИРЭА(ТУ), 2008. — 188 с.; 6. *Рiхтер Дж.* Прогресивний Windows / Рiхтер Дж. — Редмонд: Microsoft видавництво, 1997. — 1048 с.; 7. *Дейкстра Э.* Взаимодействие последовательных процессов / Э. Дейкстра; пер. с анг. В. Кузнецова. — М.: Мир, 1972; 8. *Хьюз К.* Параллельное и распределенное программирование с использованием C++ / К. Хьюз, Т. Хьюз; пер. с анг. Н. Ручко. — М.: Вильямс, 2004. — 672 с.