УДК 004.7

ТЕОРЕТИКО-ІГРОВИЙ АНАЛІЗ ПЛАНУВАЛЬНИКІВ У БАГАТОПРОЦЕСОРНИХ СИСТЕМАХ. ІМІТАЦІЙНА МОДЕЛЬ

О.П. Ігнатенко, В.Я. Одобеску

В даній роботі досліджується ігрова модель взаємодії користувачів, що виконують паралельні обчислення у гетерогенній багатопроцесорній системі. На прикладі задачі множення матриць побудований підхід до потокового моделювання процесів планування. Пропонується ігрова модель взаємодії, де стратегіями є вибір блоку розрізання матриці. Знайдені оцінки стану рівноваги та проведені експерименти, що підтверджують теоретично отримані результати. Побудована імітаційна модель, яка демонструє точки рівноваги Неша у грі взаємодії користувачів.
Ключові слова: паралельні обчислення, теорія ігор, рівновага Неша, імітаційна модель.

В работе исследуется игровая модель взаимодействия пользователей, выполняющих параллельные вычисления в гетерогенной многопроцессорной системе. Предложенный поход применяется к задаче умножения матриц с использованием планировщика мин-мин. Действием пользователей в этом случае является размер блоков, на которые разрезается матрица. Экспериментально полученные характеристики системы были использованы для настройки имитационной модели, что позволило измерить оценку времени завершения работы для всех возможных комбинаций разбиения задач по процессорам и построить поверхность времени окончания работы для каждого пользователя. Полученные результаты были обоснованы и обобщены на основе игрового подхода, в частности показано существования точки равновесия Неша в игре взаимодействия двух пользователей и найдены условия ее Парето неэффективности.
Ключевые слова: параллельные вычисления, теория игр, потоковая модель, равновесие Неша, имитационная модель.

This paper deals with a game model of users performing parallel computing in a heterogeneous multiprocessor system. The proposed approach is applied to the problem of matrix multiplication on the system with the scheduler of min-min type. The user's action is to choose the size of the blocks into which the matrix is cut. Each user tries to optimize own finish time, which leads to conflict. Using the game theoretic approach, we build game model and found the conditions of Nash equilibrium existence in the scheduling game of two users. Simulation program was built to provide experimental data.
Key words: parallel scheduling, game theory, fluid model, Nash equilibrium, simulation.

Вступ

Сучасні прикладні наукові задачі вимагають значних обчислювальних ресурсів, тому задача оптимізації прикладних задач у багатопроцесорних середовищах займає чи не найперше місце в процесі розробки таких високопродуктивних програм. Ефективне виконання обчислень вимагає використання ефективних паралельних алгоритмів, швидкодія яких у свою чергу залежить від конкретного середовища виконання. Необхідність оптимізації виникає тому, що на етапі проектування застосування частіше за все інформація про середовище виконання відсутня. І навіть якщо конфігурація відома, то майже неможливо визначити таку конфігурацію програми, за якої обчислення будуть виконуватися з максимально ефективним використанням ресурсів системи. Аналітичне моделювання дозволяє формалізувати роботу системи, поведінку користувачів і залучити їх у побудовану модель [1, 2]. Кожен гравець тут є автономним агентом, що прагне отримати певну частину обчислювального ресурсу. Припускаючи, що гравці діють раціонально – тобто максимізують свою функцію корисності можна здійснити аналіз отриманої гри. Ключовим елементом аналізу є пошук точки рівноваги та її характеристика. Рівновага – це одна з головних ідей теорії ігор. Буквально – рівноважний стан відповідає ситуації, коли гравці досягли максимуму своїх функцій корисності. В залежності від зовнішніх факторів можливі стійкі і нестійкі рівноваги. Найбільш відомою є рівновага за Нешем: множина стратегій гравців утворюють рівновагу за Нешем, якщо ніхто з них не може покращити свою функцію корисності змінюючи окремо свою стратегію. Іншими словами кожен гравець використовує найкращу можливу стратегію у відповідь на стратегії своїх опонентів.

1. Теоретико-ігрове моделювання процесів планування

В роботах [4, 6] була побудована аналітична модель алгоритму множення матриць, яка дозволяє оцінити оптимальний за часом розв'язок.

Коротко нагадаємо основні визначення. Нехай система складається з m обчислювальних елементів та кожен з них характеризується швидкістю роботи p_i, $i = 1,...,m$ – тобто кількістю операцій з плаваючою точкою за секунду, які він може здійснити. Процесори з'єднані лініями зв'язку з планувальником, який передає задачі та приймає від них результат. Будемо вважати, що лінії зв'язку ідентичні та мають швидкість передачі даних q та затримку l.

Будемо вважати, що виконуються наступні припущення

1. Всі процесори починають роботу одночасно.

2. Планувальник здійснює призначення миттєво.

Нехай задані дві квадратні матриці розмірності $N \times N$, результат множення яких необхідно обчислити. При використанні блочного алгоритму користувач задає розмір блоку n, в результаті чого формуються $k = \frac{N^2}{n^2}$ задач, кожна з яких буде мати складність $O(n^2)$. Припустимо, що планувальник забезпечує пересилку повідомлень на вузли за певним фіксованим алгоритмом, який завершує обчислення за час $T(N,n)$. Тоді задача користувача полягає у пошуку мінімуму функції:

$$T(N,n) \to \min .$$

Функція $T(N,n)$, взагалі кажучи, може мати багато локальних мінімумів (в залежності від обчислювальної системи та планувальника) оскільки існує мінімальна фіксована величина задачі.

Одним з розповсюджених підходів до аналізу таких задач полягає у дослідженні *потокової* моделі даного процесу [3].

Припустимо, що користувач вибрав вектор $x \in R^m$ з компонентами x_i, де $x_i \ge 0$, $x_i \le k$, $i = 1,...,m$, $\sum_{i=1}^{k} x_i = k$. Всі таки вектори утворюють множину $X(n)$. Кожен компонент вектора x описує відсоток задач, призначених для виконання на i-тому процесорі. Будемо брати до уваги тільки операції, множення. Таке спрощення дозволяє у явному вигляді виписати функції часу. Загальний час закінчення залежить від x, та дорівнює

$$T(x, X(n)) = \max_{i=1,...,m} \left\{ \frac{x_i N n^2}{p_i} \right\}.$$

Потокова модель передбачає можливість розділення задачі на «шматочки» розміру $\varepsilon = Nn^2$, компонування з них підходящих підзадач та визначення загального часу при $\varepsilon \to 0$.

Твердження 1 [6]. Мінімальний час закінчення обчислень (без пересилок) для потокової моделі з одним користувачем дорівнює $T = N^3 \left(\sum_{i=1}^{m} p_i \right)^{-1}$. Відповідно, користувач, для досягнення оптимального часу, має розділити свою задачу таким чином, щоб вузол i отримав $p_i T$ обчислень.

Функція Мінковського для множини X та вектора $p \in R^m$ визначається наступним чином:

$$\mu_X(p) = \inf\{\lambda > 0 : p \in \lambda X\} .$$

Відомо, що ця функція опукла для опуклої X.

Визначимо множину потужностей системи $R = \left\{ r \in R^m : r_i \in [0, p_i] \right\}$ та масштабуємо її наступним чином:

$$R(n) = \frac{1}{Nn^2} R .$$

Тоді

$$T(x, X(n)) = \mu_{R(n)}(x) .$$

Доведення. Розглянемо праву частину: $\mu_{R(n)}(x) = \inf\{\lambda > 0 : x \in \lambda R(n)\}$. Умова належності вектора x множині R записується у вигляді $\max_{i=1,...,m} \left\{ \frac{x_i}{p_i} \right\} = 1$, отже

$$\mu_{R(n)}(x) = \inf \left\{ \lambda > 0 : \max_{i=1,...,m} \left\{ \frac{x_i}{p_i} \right\} = \frac{\lambda}{Nn^2} \right\}.$$

В іншому вигляді $\lambda = \max\limits_{i=1,\dots,m}\left\{\dfrac{x_i N n^2}{p_i}\right\}$. З властивостей функції $\mu_{R(n)}(x)$ випливає, що мінімальний час $T_{\min} = \min\limits_{x\in X(n)} \mu_{R(n)}(x)$ існує і єдиний. Для врахування пересилок потрібно зазначити, що алгоритм надсилає $2x_i Nn$ елементів на відповідний вузол та приймає $x_i n^2$ елементів. Отже, сумарний час закінчення з урахуванням пересилок дорівнює

$$T_s(x, X(n)) = \max_{i=1,\dots,m}\left\{\frac{x_i N n^2}{p_i} + \frac{x_i\left(n^2 + 2Nn\right)}{q}\right\}.$$

Твердження 2 [6]. Існує мінімум часу по x –

$$\min_{x\in X(n)} T_s(x, X(n)).$$

Дискретна модель обчислення множення матриць.

Нехай розмір блоку n може бути тільки цілим числом, причому таким, щоб $k = \dfrac{N^2}{n^2}$ теж було цілим. Тоді мінімізація часу буде проводитись по скінченній множині точок

$$Y(n) = \left\{\begin{array}{l} y \in R^m : y_i \in \{0,1,..,k\}, \\ \sum\limits_i y_i = k, i = 1,..,m \end{array}\right\}.$$

Зрозуміло, що

$$Y(n) \subset X(n).$$

Введемо поняття планувальника. Спочатку користувач вибирає розмір блоку n, в результаті чого формується множина $Y(n)$. Ця множина описує всі можливі розташування задач на процесорах.

Планувальник відповідає за вибір конкретного $y^* \in Y(n)$. В даній роботі ми розглянемо прості планувальники типу $extr\,extr$. Один з широко вживаних планувальників такого типу називається $\min\min$ і він вибирає розподіл за наступним алгоритмом:

1) формується черга з k задач, кожна обсягом обчислень Nn^2;

2) вибирається задача з найменшим обсягом обчислень. (в даній роботі розглядається випадок однакових задач, тому вибирається довільна);

3) задача надсилається на вільний процесор з найбільшою потужністю (тобто мінімізується час виконання), якщо вільних процесорів немає, чекаємо поки з'явиться;

4) якщо залишились задачі у черзі, то повертаємось до пункту 2.

Твердження 3. [6] Для будь-якого n виконуються нерівності:

$$T_d(n) = \min_{y\in Y(n)} T(y, X(n)) \geq \min_{x\in X(n)} T(x, X(n)).$$

2. Некооперативна ігрова модель планування множення матриць

Некооперативна гра описує ситуацію прийняття рішень гравцями за умов конфлікту інтересів. Під терміном гравець тут ми розуміємо користувача, який впливає на систему шляхом вибору стратегій – дій, які він може застосовувати. В залежності від дій його та інших учасників відбувається визначення виграшів гравців. Говорять, що гравець раціональний, якщо його дії спрямовані на максимізацію власного виграшу.

Якщо у грі приймають участь хоча б двоє учасників, можлива ситуація, коли перший гравець може покращити свій виграш за рахунок зменшення виграшу інших. В такому випадку кажуть про конфліктну взаємодію. Частинним випадком конфлікту є ситуація повністю протилежних інтересів - коли виграш одного є програшем іншого. Такі ігри називають антагоністичними або іграми з нульовою сумою.

Зауважимо, що некооперативність не означає, що гравці взагалі не кооперуються, а тільки те, що немає зовнішніх причин, які б спричиняли координацію або узгодження їх стратегії. Будь-яка кооперація що може виникнути має виходити зі структури гри і визначатися функціями корисності учасників.

Гру називають статичною, якщо гравці приймають свої рішення одноразово, незалежно один від одного. В певному розумінні, статична гра не залежить від часу. Навіть якщо гравці приймають рішення протягом певного часового інтервалу, якщо вони не володіють інформацією щодо дій інших гравців, така гра є статичною.

У динамічних іграх гравці отримують певну інформацію щодо рішень інших учасників і можуть змінювати свою стратегію у часі, тобто приймають рішення більше одного разу. Динамічні ігри є найбільш складним для аналізу і відіграють важливу роль у дослідженні процесів у мережах [5, 6].

Опишемо задачу множення матриць у вигляді статичної некооперативної гри.

Основні визначення.

При визначенні статичних ігор загальновживаною є стратегічна або нормальна форма, яка включає опис трьох компонентів: множини гравців, їх стратегій і виграшів.

Гравцями в даному випадку являються користувачі $\{u_i\}_{i\in L}$, L – множина індексів, які мають доступ до спільного ресурсу з m обчислювальних елементів з швидкістю роботи p_i, $i=1,...,m$. Будемо вважати, що для кожного користувача задані квадратні матриці розмірності n_l, $l\in L$. Тоді стратегіями користувачів є допустиме розбиття матриць на блоки $k_l\in K_l$, $l\in L$. Після розбиття матриць, блоки потрапляють у планувальник, який надсилає їх на процесори згідно з певним визначеним алгоритмом роботи. Часом закінчення обчислень T_l, $l\in L$ будемо називати час закінчення останньої задачі користувача u_l. Кожен користувач намагається зменшити свій час закінчення і через обмеженість спільних ресурсів виграш одного користувача спричинить програш інших.

Будемо вважати, що виконуються наступні припущення:

1. Вибрана множина можливих розмірів $\{n_j\}_{j=1,...,s}$ – стратегій користувачів, впорядкована за збільшенням.

2. Якщо користувачі вибрали однакові розміри блоків, то їх час завершення однаковий і дорівнює подвоєному індивідуальному часу для даного розміру блоку.

3. Існує єдиний мінімум $T_d(n_j)$, $j=1,...,s$. Позначимо індекс стратегії, на якій він досягається j^*.

Планувальники мін-мін та макс-мін. Побудуємо платіжну матрицю гри за наступними правилами. Нехай гравці вибрали стратегії n_1, n_2 відповідно. Позначимо їх виграші $T_1(n_1,n_2)$, $T_2(n_1,n_2)$

1. Якщо $n_1 < n_2$, то виграш першого користувача дорівнює $T_d(n_1)$, другого $T_d(n_1)+T_d(n_2)$.

2. Якщо $n_1 > n_2$, то виграш першого користувача дорівнює $T_d(n_1)+T_d(n_2)$, другого $T_d(n_2)$.

3. Якщо $n_1 = n_2$, то виграш першого і другого користувача дорівнює $2T_d(n_1)$.

Твердження 4. Нехай виконуються нерівність $2T_d(n_{j^*})\le T_d(n_{j^*-1})$, тоді пара стратегій (n_{j^*},n_{j^*}) – рівновага Неша. В іншому разі рівновагою Неша будуть пара стратегій (n_{j^*-1},n_{j^*-1}).

Наслідок. Отримана рівновага Парето неефективна.

$$T_1(n_{j^*},n_{j^*}) < T_1(n_{j^*-1},n_{j^*-1}),\ T_2(n_{j^*},n_{j^*}) < T_2(n_{j^*-1},n_{j^*-1}).$$

Це досить характерна ситуація для ігор такого типу.

Планувальники макс-макс та мін-макс.

Твердження 5. Нехай виконуються нерівність $2T_d(n_{j^*})\le T_d(n_{j^*-1})$, тоді пара стратегій (n_{j^*-1},n_{j^*-1}) – рівновага Неша. В іншому разі рівновагою Неша будуть пара стратегій (n_{j^*},n_{j^*})

3. Побудова імітаційної моделі

Імітаційна модель дозволяє симулювати процес множення матриць у розподіленому середовищі враховуючи як час на власне обчислення так і передачу даних.

Розглянемо задачу множення двох $N*N$ матриць M1 та M2. Позначимо за n кількість рядків матриці M1 та відповідно стовбців матриці M2. Таким чином отримуємо 2 підматриці $n*N$ та $N*n$ які і формують одну задачу, що буде відіслана планувальнику. Таких задач буде $\left[\frac{N}{n}\right]*\left[\frac{N}{n}\right]$. Проте в обох матрицях M1 та M2 у випадку якщо n не дільник N буде залишок рядків M1 та стовбців M2 відповідно. Позначимо розмір залишка за $m = N - n*\left[\frac{N}{n}\right]$. Тому до основних задач ще потрібно додати задачі множення матриць $m*N$ та $N*n$, матриць $n*N$ та $N*m$ і задачу множення $m*N$ та $N*m$.

Тобто задача множення матриць (N,N) та (N,N) розбивається на такі підзадачі:

1. $\left[\frac{N}{n}\right]*\left[\frac{N}{n}\right]$ множень матриць $n*N$ та $N*n$.

2. $\left[\frac{N}{n}\right]$ множень матриць $m*N$ та $N*n$.

3. $\left[\frac{N}{n}\right]$ множень матриць $n*N$ та $N*m$.

4. 1 множення матриць $m*N$ та $N*m$.

Множення матриць $N1*N2$ та $N2*N3$ потребує $N1*N3*N2$ операцій множення та $N1*N3*(N2-1)$ операцій додавання. Позначимо за AM коефіцієнт складності операції множення по відношенню до операції додавання. Тоді складність множення матриць $N1*N2$ та $N2*N3$ можна виразити у одиницях операцій додавання як

$$C(N1,N2,N3) = N1*N3*(N2*AM + N2 - 1).$$

Позначивши потужність обчислювального модуля за P (кількість операцій додавання / секунду) та маючи складність задачі C (кількість операцій додавання) отримаємо $t_p = \frac{C}{P}$.

Також для множення матриць $N1*N2$ та $N2*N3$ потрібно переслати $N1*N2 + N2*N3$ елементів до обчислювального вузла, та отримати результат у розмірі $N1*N3$ елементів. Час на передачу даних обчислюється як

$$t_t = latency + \frac{N1*N2 + N2*N3 + N1*N3}{bandwidth}.$$

Загальний час на обробку задачі $t = t_p + t_t$.

Для двох гравців вибираються n_1 та n_2, формуються задачі, додаються в загальний список та випадково перемішуються і подаються на планувальник. Час для кожного з гравців визначається як час прибуття від планувальника до гравця останньої його задачі. Симуляція реалізована у вигляді утиліти з параметрами консолі та дозволяє вибирати значення N, bandwidth, latency, потужності обчислювальних вузлів як множники номінальної потужності та межі перебору параметрів n_1 і n_2. Програма написана на мові C++ та умовно поділяється на дві логічні частини: модуль обробки параметрів та модуль симуляції.

Модуль обробки параметрів дозволяє задавати розмір матриць, режим одного гравця чи двох, межі перебору стратегій розрізання, параметри планувальника, характеристики обчислювальних модулів, параметрів мережі – bandwidth та latency без перекомпіляції програми через параметри командного рядка. Модуль симуляції генерує задачі для кожного з користувачів, зливає їх в один список та власне подає їх на симулятор, який повертає оброблені задачі з проставленими змінними часу початку та завершення роботи над задачею і номером обчислювального вузла, який обробляю цю задачу.

Етапи роботи симулятора:

1) перемішування списку очікуючих задач з метою емуляції отримання задач у випадковому порядку.

2) сортування списку очікуючих задач по складності у відповідності до пріорітетності задач та списку обчислювальних вузлів по потужності у відповідності до пріорітетності обчислювальних вузлів. Наприклад для minmax планувальника список очікуючих задач буде відсортованим від простої до складної, а список обчислювальних вузлів від потужного до повільного.

3) ініціалізація початкових задач для обчислювальних вузлів вилучаючи перші елементи з відсортованих списків очікуючих задач та вузлів, проставлення початкового часу, який на даний момент рівний 0, та обчислення часу очікуваного завершення виконання задачі. Структури з посиланням на задачу, обчислювальний вузол та даними про початок та завершення виконання задачі поміщаються у чергу з пріорітетом по найменшому часу завершення.

4) взяти з пріорітетної черги задачу, яка буде найближчою по часу наступною виконаною задачею. Прийняти час симуляції за час завершення взятої з черги задачі, додати задачу до списку виконаних задач разом із даними про час її завершення.

5) якщо список очікуючих задач не пустий, то вилучити перший елемент, поставити час початку як час симуляції, обчислити час завершення та додати у чергу з прірітетом, поставивши обчислювальний вузол як перший вільний із відсортованого списку вузлів.

6) якщо пріорітетна черга не пуста, то повернутися на крок 4.

7) знайти у списку виконаних задач найпізніші повернені задачі для кожного з користувачів та повернути їх час завершення.

Результати експериментів

На рис. 1 та 2 показано залежність часу симуляції від розбиття при фіксованих N, latency, bandwidth для 3, 4 та 5 обчислювальних вузлів. Чим більше обчислювальних вузлів, тим швидше множення матриць, проте для деяких розрізань можна побачити майже однаковий час при різній кількості обчислювальних вузлів. Особливо це помітно для 4 та 5, починаючи з розміру розрізання 2500 час для них однаковий.

На рис. 3 показано залежність часу виконання усіх задач користувача 2 від розбиття n при фіксованому розбитті користувача 1 для 5 обчислювальних вузлів. На графіку чітко спостерігається стрибки при переході розбиття користувача 2 за фіксоване значення розбиття 1. Це особливість minmin та minmax оскільки вони в першу чергу виконують найлегші задачі, тому користувач, що вибрав менше розбиття, має менший час виконання усіх його задач.

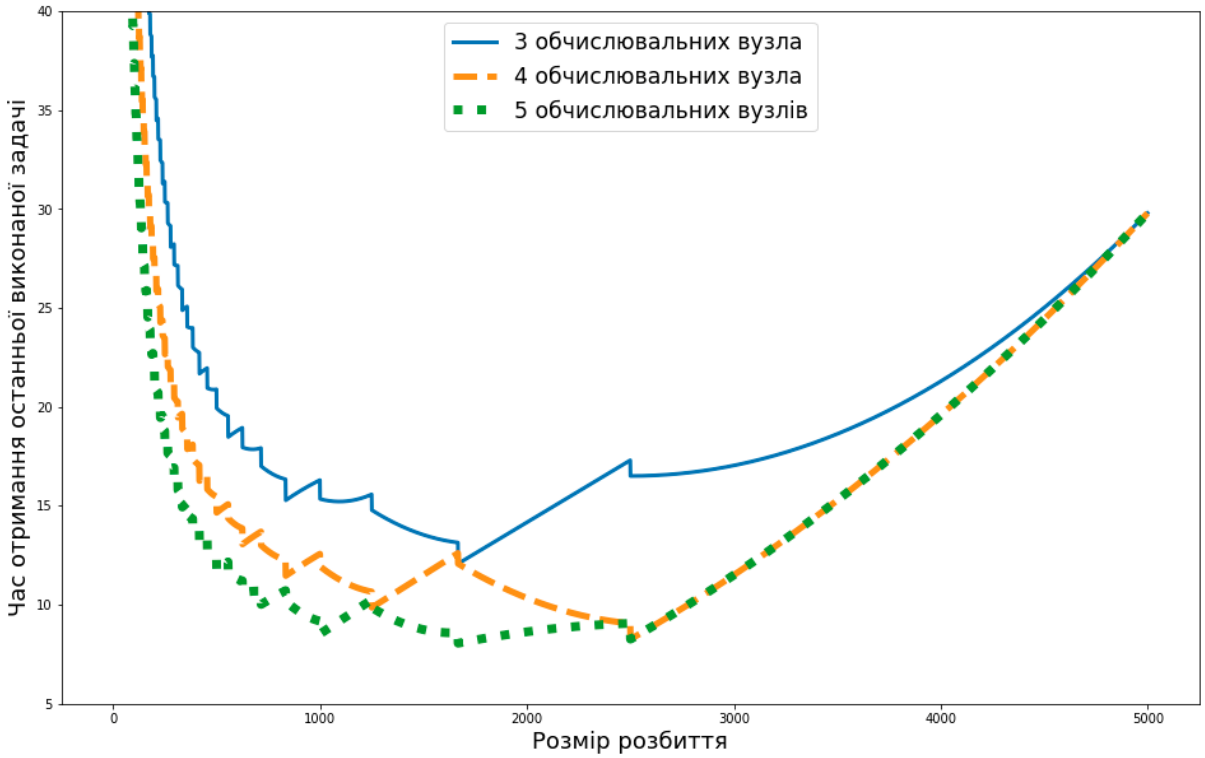

Рис. 1. Графік t від n для одного користувача при N=5000, latency = 0.0, bandwidth = 1e9 для 3, 4 та 5 обчислювальних вузлів

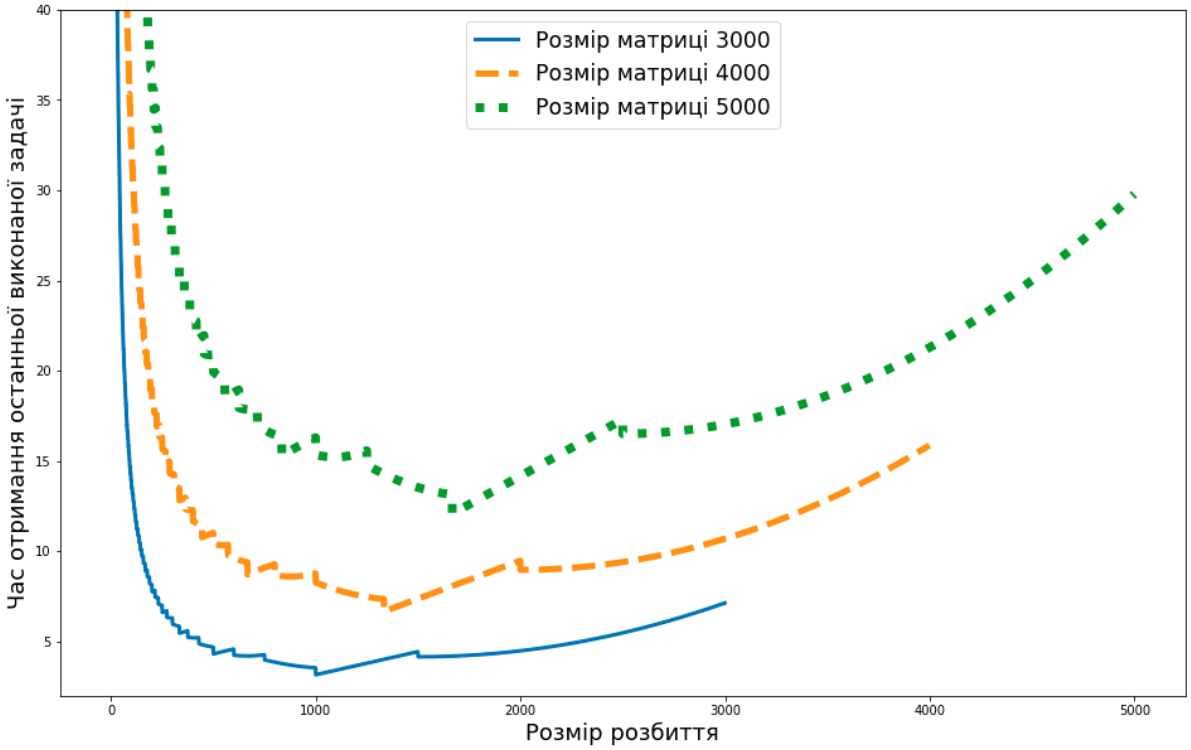

Рис. 2. Графік t від n для одного користувача при latency = 0.0, bandwidth = 1e9 та 3 обчислювальних вузлах

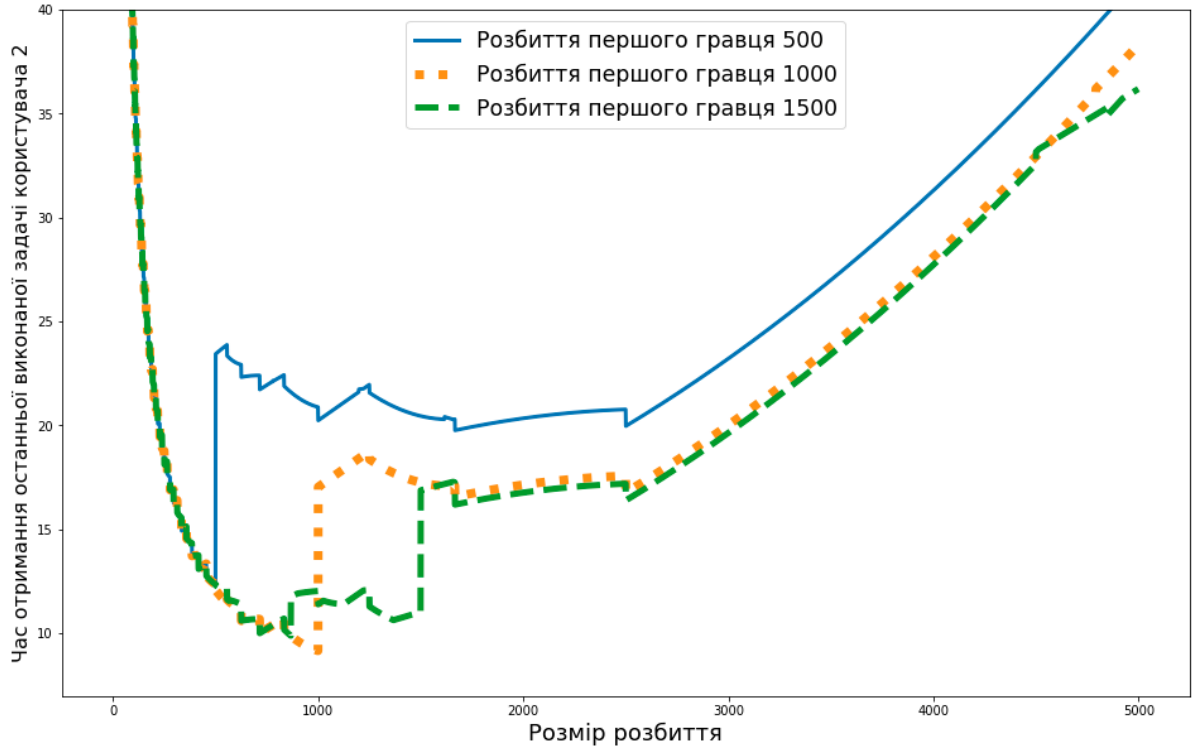

Рис. 3. Графік t від n для двох користувачів при фіксованому розрізанні першого користувача

Висновки

В роботі було розглянуто задачу ігрового моделювання процесів планування багатопроцесорними обчисленнями на прикладі задачі множення матриць. Отримані теоретичні оцінки існування стану рівноваги у грі планування двох користувачів для різних типів планувальників. Показано, що планувальники типу extr-extr розділяються на дві групи, як у певному сенсі є оберненими одна до другою (твердження 4 і 5). Показано, що отриманий стан рівноваги Неша є Парето-неефективним.

Побудована імітаційна модель на мові C++ та підтверджено теоретичні результати.

Література

1. Srinivasa Prasanna G.N., Musicus B. Generalized Multiprocessor Scheduling Using Optimal Control. Proc. SPAA. 1991. P. 216–228.
2. Grosu D., Chronopoulos A.T. Noncooperative load balancing in distributed systems. *Journal of Parallel and Distributed Computing.* 2005. **65** (9). P. 1022–1034.
3. Nazarathy Y., Weiss G. A Fluid Approach to Large Volume Job Shop Scheduling. Journal of Scheduling. 2010. 13(5), P. 509-529.
4. А.Ю. Дорошенко, О.П. Ігнатенко, П.А. Іваненко Про одну модель оптимального розподілу ресурсів у багатопроцесорних середовищах. *Проблеми програмування*. № 1. 2011. С. 21–28.
5. Андон Ф.И., Игнатенко А.П. Моделирование конфликтных процессов в сети Интернет. *Кибернетика и системный анализ*. 2013. № 4. С. 153–162.
6. Ignatenko O. Game theoretic analysis of multi-processor schedulers: matrix multiplication example. Proc. 3th Int. Conf. “ICT in Education, Research and Industrial Applications. Integration, Harmonization and Knowledge Transfer” (ICTERI 2017), Kyiv, Ukraine (15–18 May, 2017). 2017. P. 88–95.

References

1. Srinivasa Prasanna G.N., Musicus B. Generalized Multiprocessor Scheduling Using Optimal Control. Proc. SPAA. 1991. P. 216–228.
2. Grosu D., Chronopoulos A.T. Noncooperative load balancing in distributed systems. *Journal of Parallel and Distributed Computing.* 2005. **65** (9). P. 1022–1034.
3. Nazarathy Y., Weiss G. A Fluid Approach to Large Volume Job Shop Scheduling. *Journal of Scheduling.* 2010. 13(5), P. 509-529.
4. Anatoly, Doroshenko, Ignatenko Oleksii, and Ivanenko Pavlo. One model of optimal resource allocation in homogeneous multiprocessor
5. system. Problems in programming. 1 (2011): 29–39.
6. Andon, F.I., and O.P. Ignatenko. "Modeling conflict processes on the internet." Cybernetics and Systems Analysis 49.4 (2013): 616-623.
7. Ignatenko O. Game theoretic analysis of multi-processor schedulers: matrix multiplication example. Proc. 3th Int. Conf. “ICT in Education, Research and Industrial Applications. Integration, Harmonization and Knowledge Transfer” (ICTERI 2017), Kyiv, Ukraine (15–18 May, 2017). 2017. P. 88–95.

***Про авторів*:**

Ігнатенко Олексій Петрович,
кандидат фізико-математичних наук,
старший науковий співробітник відділу теорії комп'ютерних обчислень
Інституту програмних систем НАН України,
доцент кафедри ММСА Інституту прикладного системного аналізу
НТУ України “КПІ імені Ігоря Сікорського”.
Кількість наукових публікацій в українських виданнях – понад 40.
Кількість наукових публікацій в зарубіжних виданнях – понад 10.
Індекс Хірша – 5.
http://orcid.org/0000-0001-8692-2062,

Одобеску Владислав Якович,
студент Інституту прикладного системного аналізу
НТУ України “КПІ імені Ігоря Сікорського”.
http://orcid.org/0000-0002-8166-0187

Місце роботи авторів:
Інститут програмних систем Національної академії наук України.
03187, м. Київ-187, проспект Академіка Глушкова, 40, корпус 5.
E-mail: o.ignatenko@gmail.com