УДК 004.75:004.051
DOI https://doi.org/10.32838/2663-5941/2022.4/15

Зіноватна С.Л.
Національний університет «Одеська політехніка»

Тішакін М.В.
Національний університет «Одеська політехніка»

Буджеліда А.Н.
Національний університет «Одеська політехніка»

ЗАСТОСУВАННЯ ТЕХНОЛОГІЙ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ ДЛЯ ОРГАНІЗАЦІЇ ВЗАЄМОДІЇ ПРОЦЕСІВ

Процеси обробки даних у складних інформаційних системах потребують одночасного здійснення декількох операцій – паралельних обчислень. Реалізація паралельних обчислень може бути виконана двома способами: шляхом введення надлишковості незалежно працюючих елементів обчислювальної системи та з використанням обчислювальних технологій паралельного виконання, коли декілька різних команд обробки даних виконуються одночасно. Для виконання паралельних обчислень при наявності розподіленої пам'яті та декількох незалежно працюючих процесорів взаємодія між процесорами здійснюється з використанням інтерфейс передачі даних MPI – Message Passing Interface. Відповідно до традиційних паралельних обчислень, для розподілу незалежних фрагментів обчислень між різними процесорами необхідно створити програмну реалізацію цих фрагментів та розташувати їх на відповідних процесорах. Інтерфейс передачі даних MPI дозволяє спростити вирішення задачі паралелізму обчислень: створюється всього одна програма, яка запускається на виконання на всіх існуючих в обчислювальній системі процесорах. Це відповідає моделі «одна програма множина процесів» – Single Program Multiple Processes (SPMP). У роботі розглянуто ефективність паралельних обчислень з використанням функцій прийому та передачі повідомлень бібліотеки MPI. Систематизовано структуру команд MPI, виділено загальні процедури, функції для організації прийому-передачі даних між обчислювальними процесами, колективні функції та набір функцій, що працює з комунікаторами, групами та областями зв'язку.

Формалізовано процеси для визначення реальної кількості завдань у запущеному застосунку та створення комунікатора-дуплікату для надійного поділу потоків повідомлень. Для обох процесів створено схеми алгоритмів, які прокоментовані відповідно до використовуваних методів бібліотеки MPI. Для початку та завершення процесу використано MPI_Init та MPI_Finalize. Визначення кількості завдань у застосунку та порядкового номера завдання виконано з використанням MPI_Comm_size та MPI_Comm_rank. При створенні комунікатора-дуплікату для надійного поділу потоків повідомлень використано змінений підхід до обробки помилок через MPI_Abort та MPI_Barrier.

Ключові слова: *розподілені обчислення, паралельні процеси, процесор, інтерфейс передачі даних, бібліотека MPI, структура команд, схема алгоритму.*

Постановка проблеми. На теперішній час бурхливий розвиток інформаційних технологій знаходиться на тій стадії, коли зростання складності інформаційних систем вимагає пошуку нових технологій обчислень. Використання складних комп'ютерних вузлів та мереж різного масштабу обумовлює потрібність використання високопродуктивних розподілених обчислень. Це, в свою чергу, вимагає значний об'єм обчислювальних ресурсів, задіяних для виконання обчислювальних процесів. Стратегічним напрямком вирішення цих проблем є перехід до гнучкої структури інформаційної системи з використанням паралельних та розподілених обчислень.

Аналіз останніх досліджень і публікацій. Один із основних факторів, що визначають розвиток обчислювальної системи, – це висока продуктивність. Загальний метод збільшення продуктивності полягає у паралельній обробці інформації, тобто в одночасному розв'язанні кількох завдань або у поєднанні у часі етапів розв'язання одного завдання [1, с. 12].

Способи організації паралельної обробки інформації. З усього різноманіття способів такої організації можна виділити три основні напрямки [2, с. 4; 3, с. 836]:

1) поєднання у часі різних етапів різних завдань;

2) одночасне вирішення частин одного завдання;

3) конвеєрна обробка інформації.

Перший напрямок – це мультипрограмна обробка інформації. Мультипрограмна обробка можлива навіть у однопроцесорному комп'ютерному пристрої і широко використовується у сучасних системах обробки даних.

Другий напрямок можливий лише за наявності кількох обробних пристроїв. У цьому випадку використовуються ті чи інші особливості завдань чи потоків завдань, що дозволяє здійснити той чи інший вид паралелізму [4, с. 1584; 5, с. 210]. Можна виділити кілька типів паралелізму, що відбивають ці особливості.

Природний паралелізм незалежних завдань полягає в тому, що в систему надходить безперервний потік не пов'язаних між собою задач, тобто розв'язання будь-якої цієї задачі не залежить від результатів вирішення інших завдань.

Паралелізм незалежних гілок – один із найпоширеніших типів паралелізму в обробці інформації. Суть його полягає в тому, що при вирішенні завдання можуть бути виділені окремі незалежні частини – гілки програми, які за наявності кількох обробних пристроїв можуть виконуватись паралельно та незалежно один від одного.

Паралелізм об'єктів чи даних має місце тоді, коли за однією і тією ж (або майже за однією і тією ж) програмою повинна оброблятися деяка сукупність даних, що надходять до системи одночасно [6, с. 8].

Розподілені системи здійснюють різноманітну обробку даних. Можна виділити наступні напрямки розподіленої обробки:

− застосування технологій організації обробки: COM, DCOM, CORBA;

− взаємодія з розподіленими сховищами даних;

− застосування технології обміну даними (MPI), агентів та мультиагентних систем.

В даний час існує багато технологій, які дозволяють найбільш легко організувати паралельні обчислення. Одна з найпопулярніших на даний момент технологій – це інтерфейс передачі повідомлень MPI (Message Passing Interface), який забезпечує взаємодію потоків при обробці даних [7, с. 1].

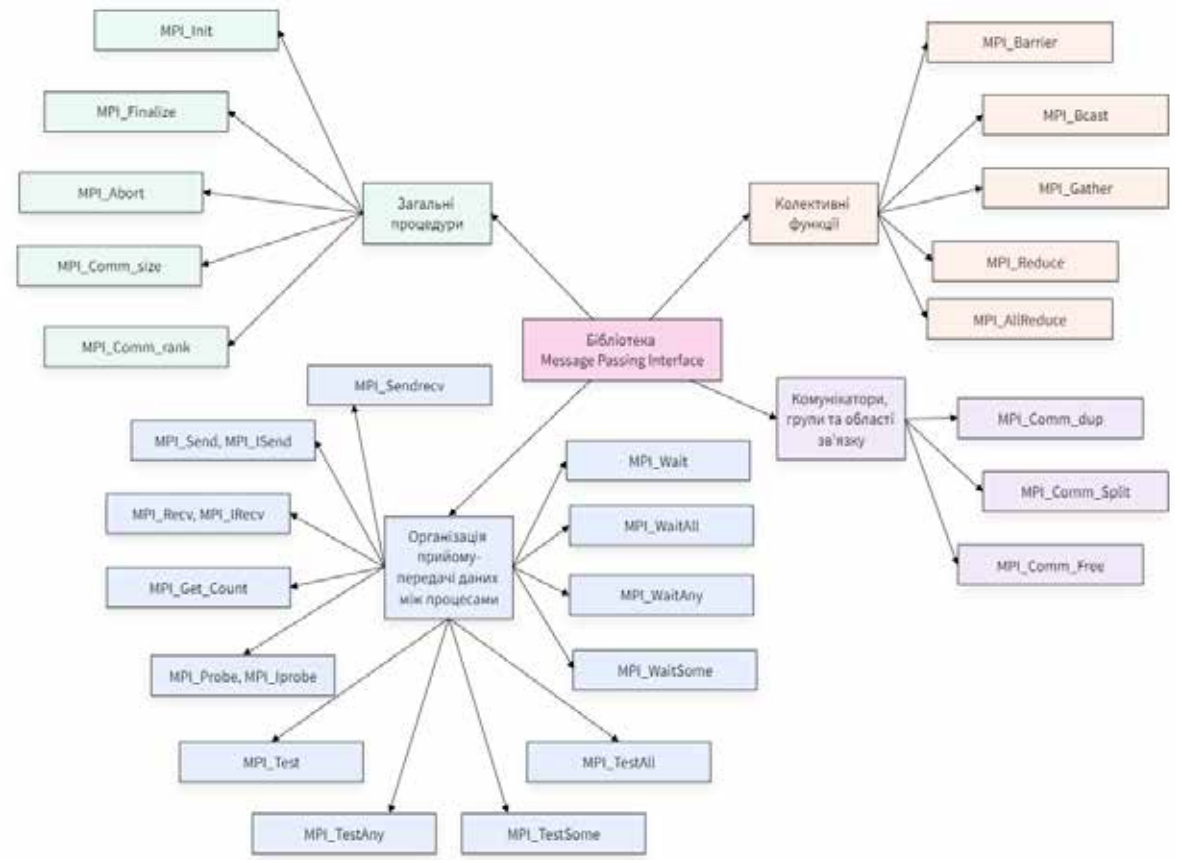

Рис. 1. Структура бібліотеки MPI

Рис. 2. Використання MPI для визначення кількості завдань у застосунку

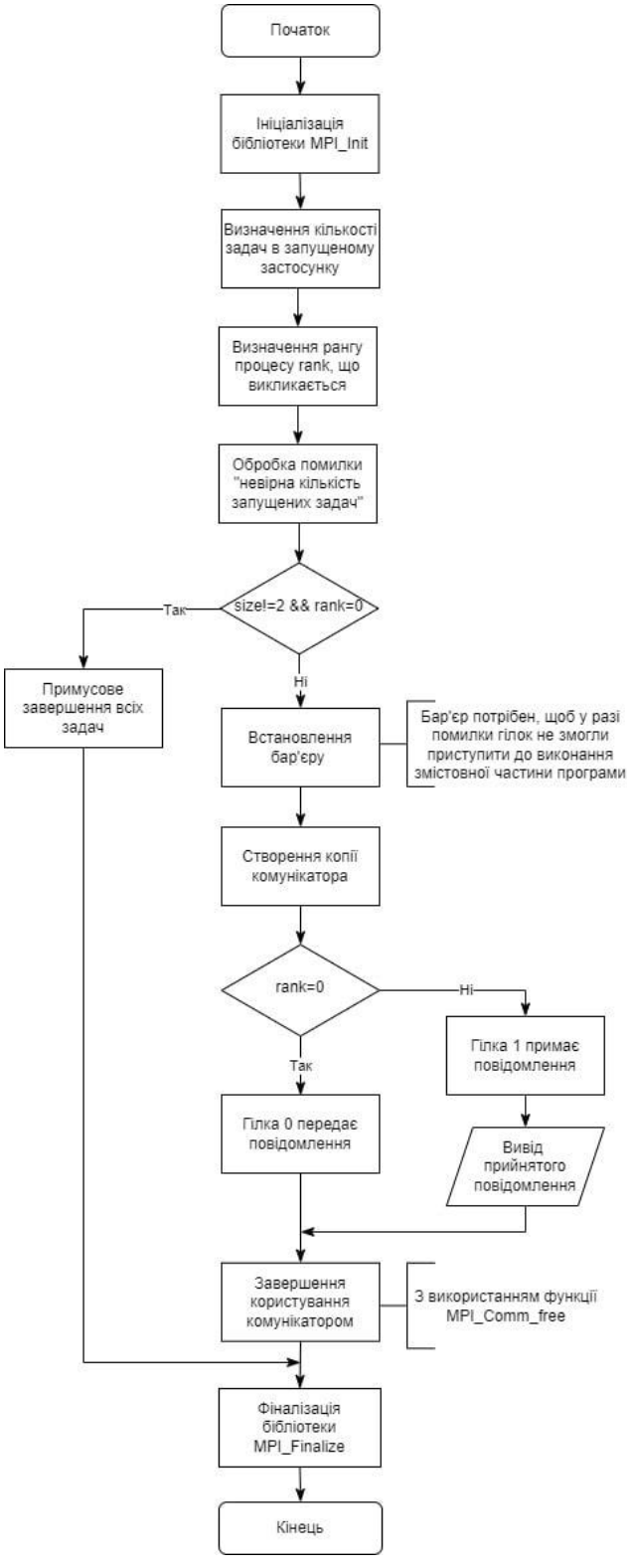

Рис. 3. Використання MPI для надійного поділу потоків повідомлень

Дана робота присвячена аналізу принципів роботи технології MPI. У рамках роботи цю технологію застосовано для організації взаємодії між пов'язаними процесами, які об'єднані у групу та протікають одночасно. Ця взаємодія включає як обмін даними (прийом-передача), так і синхронізацію взаємодіючих процесів у часі, а також моніторинг поточного стану і тривалості процесів.

Мета роботи. Метою роботи є формалізація процесу застосування технології паралельних обчислень MPI для організації взаємодії між пов'язаними процесами, з використанням синхронізації та моніторингу процесів обчислень.

Основна частина роботи

Для організації обміну даними між окремими процесами з використанням бібліотеки MPI використовуються функції прийому та передачі повідомлень; ці функції бувають двох типів: з блокуванням та без блокування. Можливе суміщення функцій передачі та прийому.

Використовуються також функції колективної взаємодії процесів.

Передача повідомлення від точки до точки здійснюється за певним протоколом через буфер. При цьому важливо коректно встановлювати параметри функцій передачі MPI_Send, MPI_ISend і прийому MPI_Resv, MPI_IResv і режиму роботи процедури, що реалізує протокол. Можливі різні режими на передавальній та приймальній стороні.

Великий інтерес представляє суміщена функція передачі/прийому MPI_Sendresv, яка використовує двоступінчасту пам'ять буфера.

При організації процесів, що протікають одночасно, важливо забезпечити різні види їхньої колективної взаємодії: MPI_BCast (один-багатьом); MPI_Gather (багато-одному).

Зазначені особливості пересилання даних спрямовані на мінімізацію часу взаємодії джерела та приймача.

На рисунку 1 показана структура команд MPI.

До загальних процедур відносяться функції, які використовуються в будь-якому, навіть найкоротшому застосунку MPI. Займаються вони не так власне передачею даних, як її забезпеченням [8, с. 146; 9, p. 2].

Найпростіший тип зв'язку між завданнями: одна гілка викликає функцію передачі, а інша – функцію прийому. Набір функцій для організації прийому-передачі даних між окремими процесами є найчисленнішим.

Під терміном "колективні" у MPI маються на увазі три групи функцій: функції колективного обміну даними, точки синхронізації (бар'єри), функції підтримки розподілених операцій.

Область зв'язку – абстрактне поняття, що дозволяє вказати, чи є певне завдання абонентом області зв'язку. Комунікатор, або описувач галузі зв'язку – це верхівка тришарового пирога (групи, області зв'язку, описники областей зв'язку), з яким працюють завдання.

Розглянемо деякі базові паралельні обчислювальні процеси та формалізуємо для них використання засобів бібліотеки MPI [10, с. 1; 11, с. 3].

На рисунку 2 показана схема алгоритму для процесу «Визначення кількості завдань у запущеному застосунку». Для початку та завершення процесу використано MPI_Init та MPI_Finalize. Визначення кількості завдань у застосунку та порядкового номера завдання виконано з використанням MPI_Comm_size та MPI_Comm_rank.

На рисунку 3 наведена схема створення комунікатора-дуплікату для надійного поділу потоків повідомлень. Алгоритм містить загальну та змістовну частини. Використано змінений підхід до обробки помилок через MPI_Abort та MPI_Barrier. Хоча для повідомлень обрано той самий ідентифікатор, описувачі області зв'язку є різними. Після завершення роботи повідомляємо MPI, що більше комунікатором не користуємося. Після цього myComm буде скинуто в MPI_COMM_NULL (тобто в 0), а відповідні дані будуть помічені на видалення.

Висновки. У роботі проведено формалізацію технологій паралельних обчислень для організації взаємодії обчислювальних процесів. Визначено ефективність паралельних обчислень з використанням функцій прийому та передачі повідомлень бібліотеки MPI. Систематизовано структуру команд MPI, виділено загальні процедури, функції для організації прийому-передачі даних між обчислювальними процесами, колективні функції та набір функцій, що працює з комунікаторами, групами та областями зв'язку.

Список літератури:

1. Технології паралельних обчислень : навчальний посібник / Семеренко В. П. Вінниця : ВНТУ, 2018. 104 с.

2. Hemanth D., Elhosney M., Nguyen Tu, Lakshmanan S. *Advances in Parallel Computing Technologies and Applications*. 2021. 52 p.

3. Rodriguez C., Rivera M., Diaz-Cacho M., Portela J. Parallel computing technologies in video stabilization for teaching purposes. *Conference: XL Jornadas de Automatica*. 2019. Pp. 836–841.

4. Jin B., Wang Y., Liu X., Bai Q., Zhang H., Gao Y. Parallel Computation Technology for Distributed Optical Fiber Sensing System. *Conference: 2019 Photonics & Electromagnetics Research Symposium*. 2019. Pp. 1584–1587.

5. Zigunovs Maksims. Implementation of the difference scheme for absorption equation type problems applying parallel computing technologies. *Proceedings of the International Scientific and Practical Conference: Environment. Technologies. Resources*. Vol. 2. 2021. Pp. 210–213.

6. Розподілені обчислення : навчальний посібник / Паулін О. М. Одеса : Одеська політехніка, 2022. 62 с.

7. Atayan A. Solving the diffusion-convection problem using MPI parallel computing technology. *Journal of Physics: Conference Series*. 1902(1):012098. 2021. Pp. 1–12.

8. Tuama, Bilal. Empirical Performance Evaluation of Gaussian Elimination and Parallel Implicit Elimination with Parallel Computing Technologies. *Conference: 2018 1st Annual International Conference on Information and Sciences (AiCIS)*. 2018. Pp. 146–151.

9. Zou Y., Zhu Y., Li Y., Wu F., Wang J. Parallel computing for genome sequence processing. *Briefings in bioinformatics*. 2021. Pp. 26.

10. Yelick K., Buluc A., Awan M., Azad A., Brock B., Egan R., Ekanayake S., Ellis M., Georganas E., Guidi G., Hofmeyr S., Selvitopi R., Teodoropol C., Oliker L. The parallelism motifs of genomic data analysis. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*. 2020. 378. 20190394. Pp. 1–11.

11. V. Gligorijevic, P. D. Renfrew, T. Kosciolek, J. K. Leman, K. Cho, T. Vatanen, D. Berenberg, B. C. Taylor, I. M. Fisk, R. J. Xavier et al. "Structure-based function prediction using graph convolutional networks," *Preprint, p. BioRxiv:786236*, October 2021. Pp. 1–9.

Zinovatna S.L., Tishakin N.V., Budzhelida A.N. APPLICATION OF PARALLEL COMPUTING TECHNOLOGIES FOR ORGANIZATION OF PROCESS INTERACTION

Data processing processes in complex information systems require the simultaneous implementation of several operations – parallel computing. The implementation of parallel computing can be performed in two ways: by introducing redundancy of independently operating elements of the computer system and using computational technologies of parallel execution, when several different data processing commands are executed simultaneously. To perform parallel calculations in the presence of allocated memory and several independently running processors, the interaction between the processors is carried out using the data transmission interface MPI – Message Passing Interface. According to traditional parallel computing, to distribute independent computational fragments between different processors, it is necessary to create a software implementation of these fragments and place them on the appropriate processors. The MPI data transmission interface simplifies the solution of the problem of computational parallelism: only one program is created, which is run on all existing processors in the computer system. This corresponds to the Single Program Multiple Processes (SPMP) model. The paper considers the efficiency of parallel computing using the functions of receiving and transmitting messages of the MPI library. The structure of MPI commands is systematized, the general procedures, functions for the organization of data reception and transfer between computing processes, collective functions and a set of functions working with communicators, groups and communication areas are allocated.

Processes have been formalized to determine the actual number of tasks in a running application and to create a duplicate communicator for reliable separation of message flows. Schemes of algorithms are created for both processes, which are commented according to the used methods of the MPI library. MPI_Init and MPI_Finalize were used to start and end the process. The number of tasks in the application and the sequence number of the task were determined using MPI_Comm_size and MPI_Comm_rank. When creating a duplicate communicator for reliable separation of message flows, a changed approach to error handling via MPI_Abort and MPI_Barrier was used.

Key words: *distributed computing, parallel processes, processor, data interface, MPI library, command structure, algorithm scheme.*