УДК 004.75:004.453.3
DOI https://doi.org/10.32782/2663-5941/2024.3.1/08

Волк М.О.
Харківський національний університет радіоелектроніки

Саранча С.М.
Харківський національний університет радіоелектроніки

Гора М.В.
Харківський національний університет радіоелектроніки

Ковтун Є.І.
Харківський національний університет радіоелектроніки

Лабазов В.Г.
Харківський національний університет радіоелектроніки

Полозов Д.М.
Харківський національний університет радіоелектроніки

МОДЕЛІ РЕСУРСІВ ТА ПРОГРАМНИХ ЗАВДАНЬ ДЛЯ СИСТЕМ ПІДТРИМКИ ФУНКЦІОНАЛЬНОЇ СТІЙКОСТІ РОЗПОДІЛЕНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

У статті розглядаються питання підвищення ефективності розподілених інформаційних систем. Однією з основних задач, яка з'являється в процесі їх роботи – функціональна стійкість. Вона забезпечує продовження виконання функцій інформаційною системою в умовах відмови апаратних або програмних ресурсів, впливу дестабілізуючих факторів з боку внутрішніх компонентів або зовні. У роботі проводяться дослідження моделей, які використовуються для оцінки ефективності методів самовідновлення для забезпечення функціональної стійкості. Сучасні методи самовідновлення характеризуються значною вартістю, тому використання розроблених моделей спрямовано на її зниження. Метою даної роботи є підвищення ефективності розподілених інформаційних систем шляхом створення моделей ресурсів та програмних завдань з урахуванням процесів забезпечення функціональної стійкості. Запропоновані моделі враховують логічну структуру програмних завдань, що обслуговують інформаційну систему, поділяють їх на код та дані, структурно окреслюють підсистеми моніторингу, контролю, збереження даних та прикладну задачу. Основним цільовим параметром виступає продуктивність використання процесорів. Її урахування в процесах розподілу ресурсів дозволив підвищити коефіцієнт використання ресурсів, зменшити кількість ресурсів та зменшити час виконання програмних завдань. Для експериментальної оцінки моделей використовувалось середовище імітаційного моделювання хмарних систем WorkflowSim. У системі моделювання було використано кілько методів розподілу ресурсів. Для всіх методів була модифікована функція оцінки продуктивності, що була реалізована з урахуванням запропонованих моделей. Проведено порівняльний аналіз використання стандартних та пропонованих моделей. Аналіз результатів показує ефективність запропонованих моделей, а саме зменшення вартості виконання завдань.

Ключові слова: *інформаційна система, комп'ютерні ресурсі, хмарні обчислення, розподілені завдання, програма, продуктивність.*

Постановка проблеми. Одною з основних властивостей сучасних розподілених інформаційних систем є підтримка функціональної стійкості, яка полягає у здатності продовжувати виконання своїх функцій в умовах відмови апаратних або програмних ресурсів, впливу внутрішніх або зовнішніх дестабілізуючих факторів [1]. Одним з ефективних методів підтримки функціональної стійкості є самовідновлення структурних елементів і взаємозв'язків самою інформаційною системою [2; 3].

Реалізація функцій самовідновлення може використовувати спеціальні програмні компоненти,

спеціалізовані бібліотеки функцій, внутрішні сервіси операційних систем, віртуальні та фізичні машини тощо.

Процеси самовідновлення у програмних системах управляються брокерами на основі ряду методів. Прикладами є методи самовідновлення програм з перезапуском програмної системи, використанням дампів пам'яті, журналізації зміни даних, мажоритарного резервування тощо. Ці універсальні методи дозволяють зберігати стан програмної системи і оновлювати стан при перезапуску програм. Загалом, ці методи нескладно реалізувати і вони використовують засоби операційних систем

Але основна проблема цих методів полягає в великій вартості. Це обумовлено об'ємами фізичної пам'яті для зберігання даних і стану програмної системи, часом, який витрачається на моніторинг, збереження стану та відновлення після збоїв. Тому для зменшення вартості обслуговування процесів забезпечення функціональної стійкості необхідно розвивати методи управління розподіленими обчисленнями.

Аналіз останніх досліджень і публікацій. Використання спеціальних програм, які є складовою інформаційної системи, надає можливість відновити виконання функції зсередини самої системи [4; 5]. Такі програмні засоби зустрічаються в сучасних операційних системах та фреймворках, наприклад у платформі .NET. Також вони можуть бути частиною бідь якої програми та оброблятися винятками (an exception), які дозволяють обробити помилку в момент її виникнення та автоматично або автоматизовано дозволити системі управління уникати її у майбутньому.

У тих випадках, коли відбуваються апаратні збої, в процесі самовідновлення виникають задачі пошуку нових ресурсів для перерозподілу програм з відновленням взаємозв'язків. Ефективність роботи тоді залежить від роботи модулів моніторингу програм, обладнання, каналів зв'язку та методів управління розподіленими обчисленнями [6].

В даній статті будемо використовувати розподілену структуру середовища виконання розподіленого програмного забезпечення з підтримкою самовідновлення, яка запропонована в роботах [3; 6]. Структурна схема середовища розподіленого обчислювального процесу наведена рис. 1.

Теоретико-множина модель середовища розподіленого обчислювального процесу містить наступні елементи:

$$PS = \bigcup_i \Pr_i, i = \overline{1, N},$$
$$\Pr = \{\bigcup Cd, Dt\} = \{Cd_u, Cd_m, Cd_s, Dt\}, \quad (1)$$

де Pr – програма (належить інформаційній системі), яка в свою чергу складається з коду (Cd) і даних (Dt). Код програмних компонент поділяється на три групи: обчислювальну прикладну задачу (Cd_u), моніторингу та контролю (Cd_m), збереження даних програми (Cd_s).

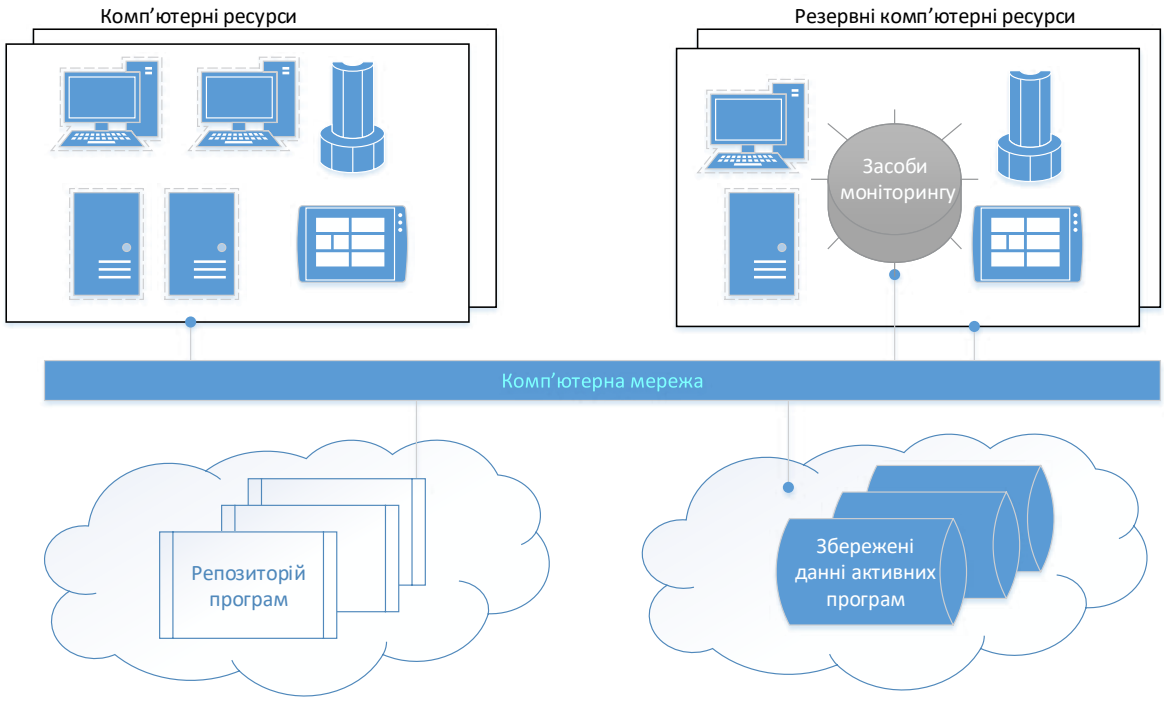

Рис. 1. Структура середовища розподіленого обчислювального процесу

Постановка завдання. Мета даної роботи полягає у підвищенні ефективності розподілених інформаційних систем шляхом створення моделей ресурсів та програмних завдань для процесів забезпечення функціональної стійкості.

Виклад основного матеріалу. В процесі розробки програмного забезпечення створюється програмний код на високорівневій мові програмування. Компілятор транслює цій код в бінарний файл виконання на машинній мові процесора, який є ядром обчислювального ресурсу. Це може відбуватися зазделегіть (з отриманням файлу, що готов до виконання) або у реальному часі, як це відбувається у кластерних системах. Тому, пропонується розширити модель програмних компонент завдання (1) компілятором Cd_C. Крім того, конкретизуємо дані програм вхідними файлами з кодами програм на високорівневих мовах програмування $Dt_{\Pr_i}, i=\overline{1,N}$. В розподіленому обчислювальному середовищі кількість компіляторів загалом залежить від кількості існуючих платформ $Cd_{Cp}, p=\overline{1,P}$, де принципово можлно виконати програму Pr_i. Кількість таких компіляторів може збільшуватися у часі з появою нових процесорів, операційних систем, фреймворків. З іншого боку, деякі ресурси неможливо використовувати з урахуванням гетерогенності інформаційних систем. Тому множину ресурсів, що долучаються до обчислювального процесу, можно поділити за ознакою можливості створення програми Pr_i для конкретного а обчислювального ресурсу R_j.

Розглянемо функцію $\varphi(Dt_{\Pr_i}, R_j)$, що буде відповідати за визначення можливості створення програми Pr_i на базі існуючого входного коду $Dt_{\Pr_i}, i=\overline{1,N}$, компілятору даної обчислювальної платформи $Cd_{Cp}, p=\overline{1,P}$, для конкретного ресурсу R_j:

$$\varphi(Dt_{\Pr_i}, R_j) = \begin{cases} \Pr_i^j, \textit{ якщо } \exists Cd_{Cp}, p=\overline{1,P},\ Dt_{\Pr_i} \xrightarrow{Cd_{Cp}} \Pr_i \\ \varnothing, \textit{ в іншому випадку} \end{cases} \quad (2)$$

Результатом виконання функції (2) на пространстві можливих пар елементів $Dt_{\Pr_i}, R_j$ є матриця Ф, яка містить атрибути програм для ресурсів з можливим виконанням, які вказують на можливість створення коду *i*-ї програми для *j*-го обчислювального ресурсу:

$$\Phi = \begin{vmatrix} \varphi_{11} & \varphi_{12} & \varphi_{13} & \cdots & \varphi_{1N} \\ \varphi_{21} & \varphi_{22} & \varphi_{23} & \cdots & \varphi_{2N} \\ & & \cdots\cdots & & \\ \varphi_{M1} & \varphi_{M2} & \varphi_{M3} & \cdots & \varphi_{MN} \end{vmatrix}. \quad (3)$$

Для спрощення можна використовувати топологічну матрицю Ф′, яка створюється на базі основної матриці (3), при цьому кожний елемент нової матриці Ф′ формується за виразом:

$$\varphi'_{ij} = \begin{cases} 1, \textit{ якщо } & \varphi_{ij} = \Pr_i^j \\ 0, \textit{ якщо } & \varphi_{ij} = \varnothing \end{cases}. \quad (4)$$

В результаті отримаємо матрицю

$$\Phi' = \begin{vmatrix} \varphi'_{11} & \varphi'_{12} & \varphi'_{13} & \cdots & \varphi'_{1N} \\ \varphi'_{21} & \varphi'_{22} & \varphi'_{23} & \cdots & \varphi'_{2N} \\ & & \cdots\cdots & & \\ \varphi'_{M1} & \varphi'_{M2} & \varphi'_{M3} & \cdots & \varphi'_{MN} \end{vmatrix}. \quad (5)$$

Матриця (5) оцінює можливість розподілення *i*-ї програми за *j*-им обчислювальним ресурсом. Матриця (3) містить атрибути (referense), що дозволяють знайти у репозиторії (рис. 1) *i*-у програму та компілятор для відповідного *j*-го ресурсу з резервних комп'ютерів.

В процессах самовідновленя програмн забезпечення інформаційних систем для підвищення ефективності методів управління треба враховувати продуктивність комп'ютерних ресурсів та обчислювальну складність програм. Для цього пропонується два вектора:

$$\overline{\rho_j^R} = <\rho_1^R, \rho_2^R, \ldots, \rho_M^R>,\ j=\overline{1,M} \quad (6)$$

та

$$\overline{\rho_i^P} = <\rho_1^P, \rho_2^P, \ldots, \rho_N^P>,\ i=\overline{1,N}, \quad (7)$$

що оцінюють продуктивність ресурсів *j* та заплановану продуктивність, необхідну для виконання *i*-ї програми.

Значення продуктивності ρ може бути характеристикою процесору або визначатися на основі функції згортання на основі множини атрибутів комп'ютера, таких як обсяги фізичної пам'яті, коефіцієнт використання процесору, часу доступу до зовнішньої пам'яті та комунікаційних каналів тощо. В останньому випадку, задача розподілу програм за комп'ютерними ресурсами ускладняється.

У випадку гетерогенних інформаційних систем це завдання є більш складним. Методи управління в таких випадках повинні враховувати різні характеристики продуктивності наявних ресурсів та програм. Загалом, в такому випадку, для вирішення завдань ефективного використання ресурсів, пропонується разширити модель розподіленого обчислювального середовища з використанням векторів (6) та (7) та отримати наступну матрицю:

$$\Phi^{\rho} = \begin{vmatrix} \rho_1^P/\rho_1^R & \rho_2^P/\rho_1^R & \rho_3^P/\rho_1^R & \dots & \rho_N^P/\rho_1^R \\ \rho_1^P/\rho_2^R & \rho_2^P/\rho_2^R & \rho_3^P/\rho_2^R & \dots & \rho_N^P/\rho_2^R \\ & \dots\dots & & & \\ \rho_1^P/\rho_M^R & \rho_2^P/\rho_M^R & \rho_3^P/\rho_M^R & \dots & \rho_N^P/\rho_M^R \end{vmatrix}, \quad (8)$$

елемент якої ρ_i^P/ρ_j^R є коефіцієнтом використання продуктивності j-го комп'ютерного ресурсу i-ю програмою. Будемо враховувати, що у випадку $M \neq N$ (кількість програм і ресурсів не рівні), можливі ситуації, коли кілька програм розподіллено на один ресурс і, навпаки, деякі ресурси зовсім не завантажені програмами.

Для можливості реалізації таких моделей продуктивність завдань та комп'ютерних ресурсів повинні бути нормованими або дійсними (для програмних завдань це виконує користувач, для ресурсів – постачальник), але приведеними для забезпечення можливості порівняння. Наприклад, у GRID-системах цей обов'язок покладається користувача: задати планову продуктивність або плановий час виконання за допомогою відкритої статистичної інформації про кластер, що використовується. Але, на практиці, часто дійсні параметри в умовах динамічної зміни ресурсів, складності оцінки програм, такі характеристики отримати складно. У такому випадку перспективним бачиться отримання нормованих характеристик на базі бенчмарків, що виконуються як для програм в завданнях, так і для комп'ютерних ресурсів до початку виконання.

Для моделювання процесу розподілених обчислень в хмарних системах було використано проект з відкритим доступом WorkflowSim [9], який дозволяє порівняти ефективність роботи системи управління з використанням запропонованих моделей з моделями, які реалізовані в системі моделювання та представляють найбільш використовувані моделі в сучасних планувальниках. У системі моделювання реалізовано кілько методів розподілу ресурсів. Для проведення експериментів було вибрано наступні методи: first-come-first-served (FCFS), Min–Max (мінімальний час виконання, максимальна продуктивність), і Round Robin (RR). В усіх методах була добавлена функція оцінки продуктивності, побудована на запропонованих моделях. Методи самовідновлення використовувалися стандартні, заложенні в систему керування розподіленими обчисленнями.

В методі FCFS у порядку надходження програм на виконання на основі вектору (6) обирався ресурс з найбільш близькою продуктивністю, що підвищувало коефіцієнт використання ресурсу та зменшувало час простою ресурсу, чим зменшувало загальну вартість виконання обчислювального процесу.

В двох наступних методах аналіз проводився у ситуації, коли рішення приймалося при наявності повної інформації про множину програм та ресурсів. Для формування функції вибору ресурсу для конкретної програми використовувалась матриця (8). Функція містила три етапи. На першому етапі працювала стандартна функція методу. На другому – з допустимих рішень обиралось то, додавання продуктивності якого до одного з ресурсів не перевищувало загальну продуктивність ресурсу. Якщо на другому етапі ресурс було неможливо призначити, обирався найближчій за продуктивністю ресурс з решти допустимих рішень.

Для проведення експериментів в системі моделювання було використано різну кількість ресурсів і програм для перевірки валідності запропонованих моделей. Кількість програм визначалась у діапазоні від 100 до 1000, кількість ресурсів – 16–64, об'єм фізичної пам'яті кожного ресурсу – 32 Gb, об'єм зовнішньої пам'яті – 8Tb, гіпервізор Xen. Потік програм, поточна конфігурація та збої генерувались випадково, але для кожної згенерованої конфігурації застосовувались усі три метода з стандартними та модифікованими моделями. Отримані в результаті характеристики були усереднені на основі нормування умовної вартості використання ресурсів. Результати оцінки вартості використання ресурсів наведено на рис. 2.

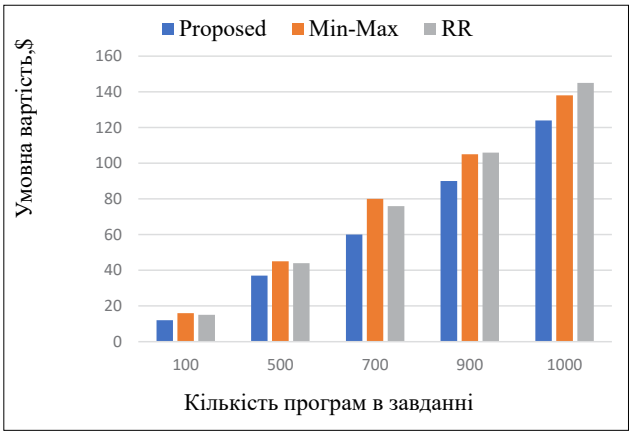

Рис. 2. Оцінка вартості розподілених обчислень

В експериментах використано однорідні ресурсі, тому і вартість використання усіх ресурсів була визначена однаковою. Аналіз результатів показує ефективність запропонованих моделей, а саме зменшення вартості виконання завдань. За рахунок відновлення стану програмних компонент, обчислювальних процес відновлює виконання своїх функцій швидше, ніж з застосуванням стандартних методів.

Висновки. У статті розглянуто задачу підвищення ефективності розподілених інформаційних систем с використанням самовідновлення для підтримки функціональної стабільності. Запропоновані в роботі моделі враховують логічну структуру програмного забезпечення інформаційної системи, структурно виділяють в них код, дані, підсистему моніторингу, контролю, збереження даних та прикладну задачу.

Використання запропонованих моделей в стандартних методах управління розподіленими обчисленнями, дозволило в описаних експериментах знизити вартість виконання програмних завдань.

При проведенні експериментів використовувався випадок однорідних ресурсів і вартість використання усіх ресурсів була визначена однаковою. Однак, перспективним бачиться вивчення питання модифікації моделей з урахуванням гетерогенності сучасних комп'ютерних ресурсів та застосування моделей в інших методах розподілу ресурсів та управлінні обчислювальним процесом хмарних систем.

Список літератури:

1. Собчук В.В., Барабаш О. В., Мусієнко А.П. Основи забезпечення функціональної стійкості інформаційних систем підприємств в умовах впливу дестабілізуючих факторів : монографія. Київ : Міленіум, 2022. 272 с. ISBN 973-966-8063-82-3

2. Hudaib, A., Fakhouri, HN., Al Adwan, FE., & Fakhouri, SN. A Survey about Self-Healing Systems (Desktop and Web Application). *Communications and Network*. 2017. Vol. 09. № 01. P. 71–88. DOI: 10.4236/cn.2017.91004

3. Волк М.О., Гора М.В., Филимончук Т.В., Казмина Д.Р., Ольшанска Т.І. Модифікований метод самовідновлення програмних систем з використанням дампу пам'яті. *Сучасний стан наукових досліджень та технологій в промисловості*. Харків, 2019. № 1 (7). С. 121–129. DOI: 10.26906/SUNZ.2021.3.074

4. Manzoor A., Rajput U, Phulpoto N, Abbas F, Rajput M. Self-healing in Operating Systems. *IJCSNS International Journal of Computer Science and Network Security.* May 2018. Vol. 18. № 5. P. 92–98.

5. Wang Z., Wang J. Self-healing resilient distribution systems based on sectionalization into microgrid. *IEEE Transactions on Power Systems*. 2015. 30 (6). P. 3139–3149. DOI: 10.1109/TPWRS.2015.2389753

6. De Lemos R., Giese H., Muller H.A., Shaw M., Andersson J., Litoiu M., Schmerl B., Tamura G., Villegas N.M., Vogel T. et al.: Software engineering for self-adaptive systems: a second research roadmap. In: *Software Engineering for Self-Adaptive Systems II, Springer.* 2013. P. 1–32. DOI: 10.1007/978-3-642-35813-5_1

7. Рубан І.В., Волк М.О., Рісухін М.В. Метод самовідновлення розподіленого програмного забезпечення в гетерогенних комп'ютерних системах. *Телекомунікаційні та інформаційні технології.* 2019. № 3 (64). С. 17–23. DOI: 10.31673/2412-4338.2019.031723

8. Волк М.О., Гора М.В., Лабазов В. Г., Міщенко А.В., Барсуков А.І., Голець В.В. Журналізація стану програм для самовідновлення паралельних програмних систем. *Системи управління, навігації та зв'язку*. 2023. № 2 (72). С. 76–82. DOI: 10.26906/SUNZ.2023.2.080

9. WorkflowSim https://github.com/WorkflowSim/WorkflowSim-1.0. (date of access: 20.01.2024)

Volk M.O., Sarancha S.M., Hora M.V., Kovtun Ye.I., Labazov V.H., Polozov D.M.

MODELS OF RESOURCES AND SOFTWARE TASKS FOR SYSTEMS IMPROVEMENT OF FUNCTIONAL STABILITY OF DISTRIBUTED INFORMATION SYSTEMS

The article examines the nutritional efficiency of the distributed information systems. One of the main tasks that appear in the process of their work is functional stability. It will ensure the continuation of the established functions of the information system in the minds of hardware and software resources, the influx of destabilizing factors from the internal components or external parts. The article is conducting research on models that are being tested to evaluate the effectiveness of self-healing methods to ensure functional stability. Current methods of self-innovation are characterized by significant variability, which means that the development of fragmented models is directly related to their reduction. The goal of this work is to improve the efficiency of distributing information systems by creating models of resources and software specifications to ensure functional stability. The proposed models provide a logical structure of software tasks that serve the information system, subordinate them to the code and data, structurally integrate subsystems for monitoring, control, data saving and the application task. The main target parameter is the productivity

of the processor using. This regulation in the processes of distribution of resources will allow you to increase the ratio of resource allocation, change the number of resources and reduce the time required to complete program tasks. For the experimental evaluation of the models, the middle part of the simulated WorkflowSim system was used. The modelling system had a number of methods available for different types of resources. For all methods, the productivity assessment function was modified and implemented in accordance with the structure of the models. A comprehensive analysis of the various standard and proposed models was carried out. An analysis of the results shows the effectiveness of the model assignments, and the resulting change in the performance of the task.

Key words: *information system, computer resources, cloud computing, distributed tasks, program, productivity.*