УДК 004.75

Воєводін Є.В.
Черкаський національний університет імені Богдана Хмельницького

Авраменко В.С.
Черкаський національний університет імені Богдана Хмельницького

ПОРІВНЯННЯ ЕФЕКТИВНОСТІ ТОПОЛОГІЙ САМООРГАНІЗАЦІЙНИХ КАРТ КОХОНЕНА В СИСТЕМАХ ОРКЕСТРУВАННЯ ВІРТУАЛЬНИХ КОНТЕЙНЕРІВ

У статті описується метод розподілення динамічної послідовності віртуальних контейнерів з використанням самоорганізаційних карт Кохонена. Описаний метод дає змогу побудувати модель, яка базується на неявних зв'язках між віртуальними контейнерами та вузлами в кластері, таким чином в деяких випадках дозволяючи проводити балансування більш ефективно. Також у роботі розглядаються одновимірна топологія карти та топологія у вигляді квадратної матриці. Ефективність роботи методу, який базується на різних топологіях, перевіряється в ході проведення експериментів заповнення до першого відхилення та заповнення до повноти кластеру. Аналіз отриманих результатів показує позитивні та негативні сторони використання різних топологій, та виявляє, що ефективність розподілення залежить від структури карти. Описана методологія проведення експериментів може бути використана для дослідження кластерів та пошуку ефективної конфігурації карти Кохонена. Зроблені висновки можуть бути використанні для розробки нових, більш ефективних евристичних методів розподілення.

Ключові слова: *розподілення, балансування, самоорганізаційні карти Кохонена, штучні нейронні мережі, віртуалізація, контейнер, система оркестрування.*

Постановка проблеми. Системи оркестрування віртуальних контейнерів (СОВК) займають важливу нішу в розподілених системах. Одиницею віртуалізації в СОВК є віртуальний контейнер (далі – контейнер). Контейнер – це легковагове, незалежне, виконавче середовище, що містить усе потрібне для виконання в ньому програми, включаючи код, бібліотеки, засоби розробки тощо. Контейнер описує ряд вимог, якими він має бути забезпечений для повноцінної роботи. У загальному випадку до таких вимог можна віднести кількість ядер центрального процесора та кількість оперативної пам'яті (ОП), потрібної йому для роботи. У залежності від конкретної СОВК, конфігурація контейнера може описувати набір додаткових вимог.

Однією із задач СОВК є забезпечення контролю повного життєвого циклу контейнерів у кластері. Створення контейнера є невід'ємною частиною його життєвого циклу. Процес створення включає в себе пошук вільного місця в кластері, так званого вузла, яке б задовольняло вимоги його конфігурації. Для виконання цієї задачі СОВК використовують різні стратегії розподілення.

Стратегія розподілення заповненням чи то стратегія розподілення поширенням є стандартними рішеннями для СОВК. Перевага першої полягає у використання малої кількості вузлів, що зменшує потребу в додаткових апаратних ресурсах, але концентрація багатьох контейнерів на одному вузлі породжує низький рівень відмовостійкості [1]. У свою чергу, більш високий рівень відмовостійкості досягається з використанням стратегії розподілення поширенням, за рахунок розміщення контейнерів на вільних вузлах. Однак такий підхід виливається в проблему простою вільних апаратних ресурсів. У результаті того, що стратегії не аналізують динамічну послідовність, з'являються випадки в яких балансування здійснюється неефективно. Це негативно впливає на СОВК та змушує систему приймати критичне рішення відносно запиту балансування, зокрема: не приймати нових запитів, затримати запит до появи апаратних ресурсів, створити новий вузол з подальшим запуском контейнера на ньому.

Аналіз останніх досліджень і публікацій.

Проблеми використання існуючих методів у своїй роботі описує Є.В. Воєводін [2]. Автор

порівнює між собою методи розподілення заповненням і розподілення поширенням, наводячи їхні переваги та недоліки. У ході порівняння автор демонструє за яких умов стратегії здійснюють неефективне балансування. Окрім цього, в роботі описується можлива поведінка СОВК у критичних випадках. Серед потенційних способів покращення процесу балансування автор виділяє евристичні методи, зокрема самоорганізаційні карти Кохонена (СКК) [3].

У своїй іншій роботі Є.В. Воєводін описує, як можна використати СКК для реалізації евристичної стратегії балансування контейнерів [4]. Також автор проводить ряд експериментів, які показують, для яких динамічних послідовностей можна добитися більш ефективних результатів ніж з використанням стандартних стратегій. Так, наприклад, динамічну послідовність контейнерів з вимогами до оперативної пам'яті 1 Гб, 3 Гб, 4 Гб, 3 Гб, 5 Гб ні стратегія розподілення заповненням, ні стратегія розподілення поширенням не може розподілити повністю на трьох вузлах місткістю 6 Гб, тоді коли стратегія з використанням СКК успішно справляється з цією задачею.

Різні структури СКК та їхній вплив на поведінку карти описують в своїй роботі Ф. Джанг та інші [5]. Проаналізувавши різні топології СКК, автори експериментально доводять, що ефективність роботи карт можна підвищити близько на 10% за рахунок оптимізації їхньої структури.

Метою статті є опис, експериментальна перевірка та порівняння ефективності різних топологій СКК для балансування динамічної послідовності контейнерів.

Основні результати дослідження.

У даному дослідженні для вирішення задачі розподілення динамічної послідовності контейнерів використовується СКК. СКК – це штучна нейронна мережа, яка навчається без учителя. СКК дають змогу відображати простір більшого виміру в простір меншого виміру, зберігаючи топологічні властивості вхідного простору. СКК є відображенням динамічної послідовності контейнерів в одновимірний простір вузлів. Таке відображення дозволяє формувати групи схожих за вимогами контейнерів на основі неявних залежностей. Опишемо основні етапи роботи алгоритму СКК та стратегії розподілення.

Етап перший – ініціалізація вагових коефіцієнтів $^{--\rho c}$. Ініціалізувати вагові коефіцієнти СКК можна по різному. У даному досліджені буде використовуватись ініціалізація значеннями випадково обраними із векторів вхідних даних. Якщо множина вхідних даних більша ніж розмір карти, то має сенс у ході вибору випадкових векторів розбити всю множину на стільки відрізків, скільки нейронів у карті та взяти кожний такий вектор, який знаходиться в середині якогось із отриманих відрізків.

Етап другий – змагання. У ході змагання знаходиться відстань від вхідного вектору до кожного з нейронів. В якості функції пошуку відстані, в межах даної роботи, буде використовуватись Евклідова відстань. Тоді відстань до нейрона переможця можна виразити наступним чином:

$$\vec{x}(t) - \vec{w}^{(c)}(t) = \min_{j\in\{1,\dots,M\}} \left\{ \vec{x}(t) - \vec{w}^{(j)}(t) \right\},$$

де c – це індекс нейрона переможця; t – номер ітерації; $\vec{x}(t)$ – вектор вхідних даних на ітерації t; $\vec{w}^{(j)}(t)$ – вектор вагових коефіцієнтів для нейрона з індексом j на ітерації t; M – кількість нейронів у СКК. Нейрон переможець – це нейрон, який знаходиться найближче до вхідного вектору.

Етап третій – кооперація. У ході третього етапу потрібно знайти топологічну околицю навколо нейрона переможця, тобто знайти нейрони чиї вагові коефіцієнти потрібно адаптувати до вхідних даних. Позначимо функцію сусідства між нейронами, як $h_{cj}(t)$. В якості функції впливу буде використовуватись функція впливу Гауса:

$$h_{cj}(t) = \mathrm{e}^{-\frac{d^2}{2\sigma(t)^2}}, \qquad (1)$$

де $\sigma(t)$ – це функція визначення топологічної околиці, в якості якої можна використати функцію експоненціального розпаду [6]:

$$\sigma(t) = \sigma_0 e^{-\frac{t}{T}}, \qquad (2)$$

де $\sigma_0 = const$ – початкова ширина топологічної околиці; t – номер ітерації; T – загальна кількість ітерацій. У свою чергу d – це відстань між нейронами, $d = ne_c - ne_j$, де ne_c – позиція нейрона переможця; ne_j – позиція нейрона, що пробуджується. Функція Гауса надає можливість змінювати вагові коефіцієнти нейронів по-різному. Чим далі цільовий нейрон від нейрона переможця, тим слабше він адаптується до вхідних даних, окрім цього зі збільшенням ітерації зменшується топологічна околиця.

Етап четвертий – адаптація. У ході адаптації значення вагових коефіцієнтів СКК змінюються в напрямку вхідного вектору. Корегування вагових коефіцієнтів здійснюється за формулою:

$$\vec{w}^{(j)}(t+1) = \vec{w}^{(j)}(t) + \alpha(t) h_{cj}(t) \left[\vec{x}(t) - \vec{w}^{(j)}(t) \right],$$

де $\alpha(t)$ – функція швидкості навчання, яка в також дозволяє керувати інтенсивністю адаптації вагових коефіцієнтів на відповідній ітерації. В якості такої функції, як і у випадку з функцією

Гауса (1), можна використати експоненціальну функцію розпаду, але дещо з іншими параметрами:

$$\alpha(t) = a_0 e^{-\frac{t}{n}},$$

де $a_0 = const$ – початкове значення швидкості; $n = const$ – дозволяє контролювати, як швидко зменшується значення функції; t – це номер ітерації.

Процес навчання СКК здійснюється за T ітерацій, де на кожній наступній ітерації з вибірки обирається випадковий вхідний вектор $\vec{x}$, та проходять вищеописані етапи – змагання, кооперації та адаптації.

Для проведення експерименту була реалізована стратегія балансування в *COBK Docker Swarm*. Реалізація складається з двох основних частин. Перша – ініціалізація стратегії, в ході ініціалізації здійснюється конфігурація та навчання СКК на основі попередньо вказаної множини конфігурацій. Друга – розподілення контейнерів. Для розподілення одного контейнеру потрібно зробити наступне:

1. Перетворити конфігурацію контейнера у вектор вхідних даних.
2. Знайти вектор відстаней, так само як і на етапі змагання.
3. Відсортувати послідовність вузлів за значеннями векторів відстаней.
4. З отриманої послідовності виключити всі вузли, які не задовольняють вимоги конфігурації.
5. Якщо після вилучення не залишається вузлів, то відхилити запит балансування, в іншому ж випадку – помістити контейнер на першому вільному вузлі.

У ході експерименту використовується генератор псевдо-випадкових конфігурацій (ГПВК), який дає змогу отримувати динамічні послідовності конфігурацій контейнерів за попередньо заданих вимог та обмежень. У контексті даного дослідження використовуються експериментальні вимоги, описані в таблиці 1.

Таблиця 1

Класифікація	Ймовірність	Опис вимог
Низькі вимоги	50%	[ОП 1], [ОП 2], [ОП 3]
Середні вимоги	30%	[ОП 4], [ОП 5], [ОП 6]
Високі вимоги	20%	[ОП 8], [ОП 16]

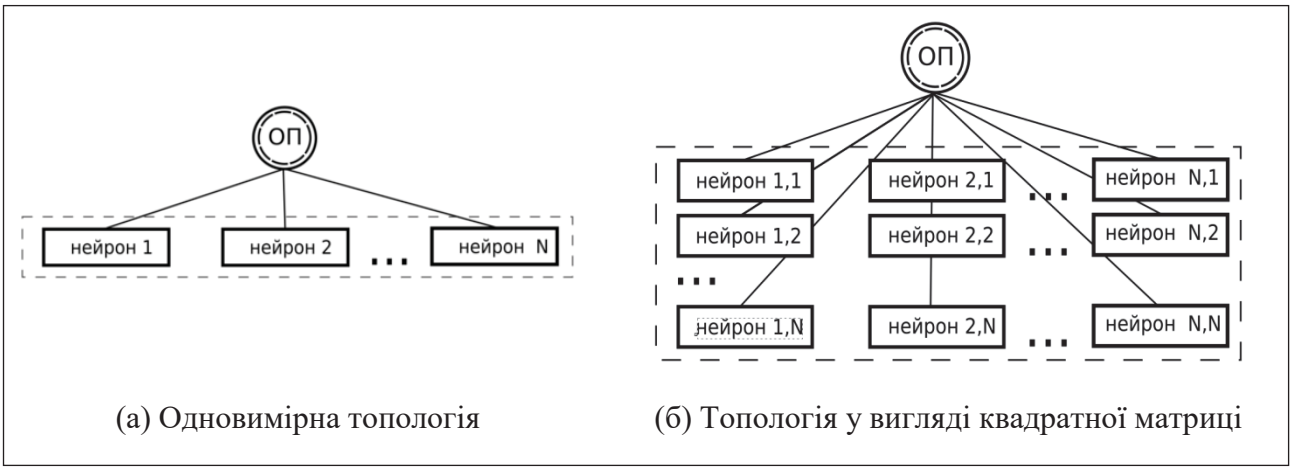
Динамічна послідовність формується за допомогою значень, які генерує ГПВК. Спочатку ГПВК обирає якусь конкретну класифікацію, керуючись ймовірностями описаними в заданій таблиці, далі з описаної множини вимог (конфігурацій) обирається випадкове значення, яке, при потребі, стає наступним елементом динамічного ряду.

Досліджуватись будуть дві різних топології. Перша – одновимірна (рис. 1. а), при використанні якої в СКК окрім шару вхідних даних присутній всього один шар нейронів. Кількість нейронів рівна кількості вузлів в кластері. Друга – квадратна матриця (рис. 1. б). Кількість елементів у матриці становить N^2, де N можна порахувати як:

$$N = g\left(\sqrt{c}\right); g(x) = \begin{cases} x, x = x \\ x + 1, x \neq x \end{cases},$$

де c – розмір кластеру; x – оператор "підлоги", $x = \max\{m \in \mathbb{Z} | m \leq x\}$. Таким чином, матриця є такого мінімального розміру, щоб можна було вмістити всі вузли кластера.

Перший експеримент полягає в тому, щоб здійснювати паралельну упаковку до першого відхилення запиту. Експеримент проводиться для



(а) Одновимірна топологія (б) Топологія у вигляді квадратної матриці

Рис. 1. Топології СКК

декількох стратегій паралельно, при цьому використовується одна й та ж динамічна послідовність конфігурацій контейнерів. Під відхиленням запиту мається на увазі ситуація, коли стратегія не може розподілити наступну конфігурацію в динамічному ряді, оскільки в кластері просто немає вузла який відповідає її вимогам. Експеримент проводиться 10000 разів для кожної розмірності кластеру. Експеримент дозволяє визначити, для якої кількості динамічних послідовностей контейнерів одна стратегія була більш ефективна за іншу, тобто частіше розподіляла більше контейнерів. Місткість одного вузла становить 16 Гб, експериментальна початкова ширина топологічної околиці (2) $\sigma_0 = 4$. З графіку (рис. 2) видно, що з ростом кількості вузлів в кластерів змінюється тенденція ефективності. По-перше, для розмірності кластеру в 10 вузлів, як стратегія, що базу-

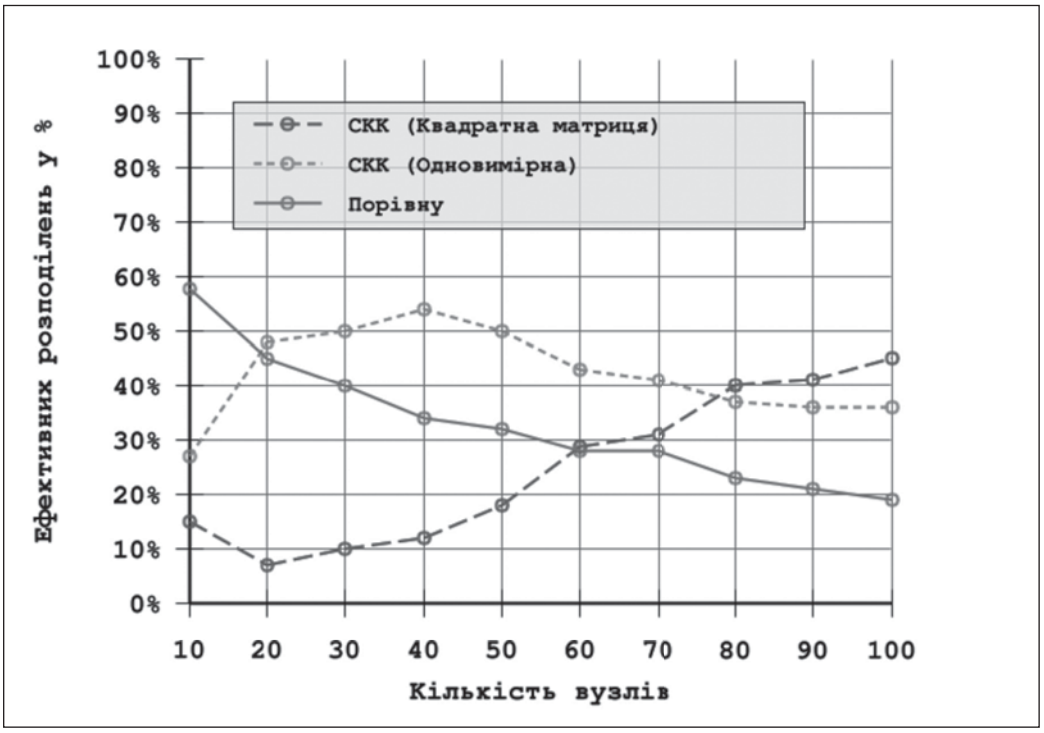


Рис. 2. Градація ефективності різних топологій СКК

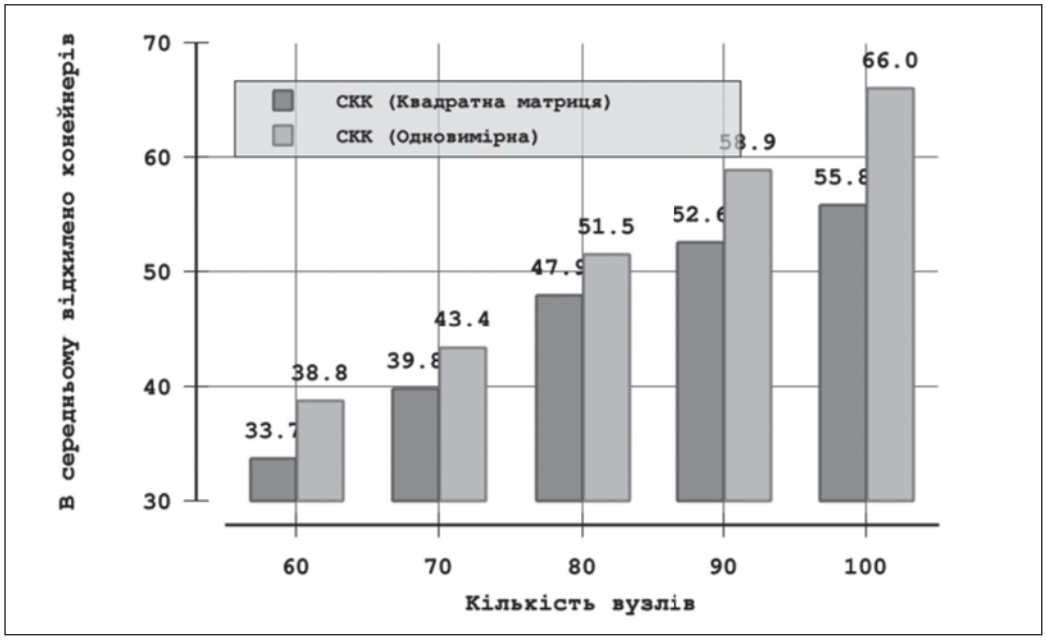


Рис. 3. Діаграма кількості відхилень для різних топологій

ється на одновимірній топології, так і стратегія, що базується на топології квадратної матриці, в більшості випадків (60%) є однаково ефективними. Чим більше в кластері вузлів, тим меншим є значення порівну запакованих спроб.

По-друге, одновимірна структура дозволяє досягти більшої ефективності для кластерів малого та середнього розміру, тоді як СКК з топологією квадратної матриці дозволяє досягти більшої ефективності для кластерів більшої розмірності. При зміні вмісту одного вузла до 32 Гб, результат є ідентичним до попереднього, окрім незначних відхилень.

Для такої ж конфігурації проведемо експеримент упаковки до повноти кластеру. Суть експерименту в тому, щоб продовжувати розподілення

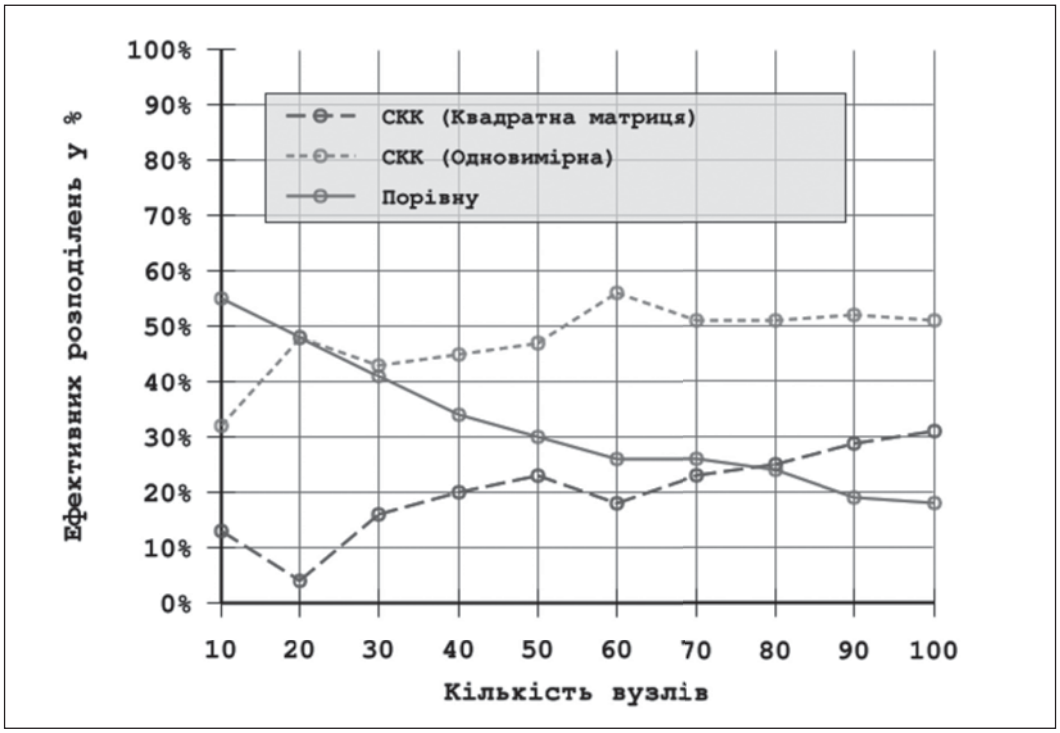


Рис. 4. Градація ефективності різних топологій СКК з динамічним ростом σ_0

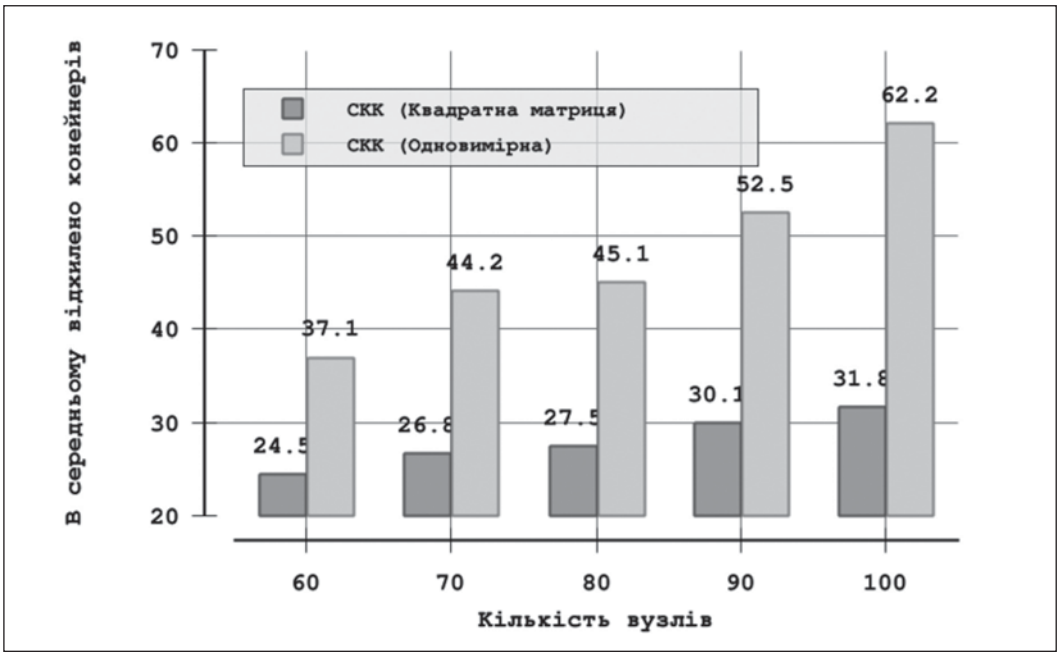


Рис. 5. Діаграма кількості відхилень для різних топологій СКК з динамічним ростом

ресурсів до тих пір, поки всі ресурси кластеру не будуть повністю використані. Якщо з використанням стратегії на основі якоїсь із топологій СКК вдається повністю запакувати кластер, то стратегія перестає брати участь у балансуванні, та очікує, поки ця ж умова буде правдива і для іншої. У результаті експерименту можна перевірити, яке в середньому значення контейнерів вдається розподілити з використанням різних структур СКК, а також яку середню кількість контейнерів було відхилено, до поки кластер не є повністю заповнений. У результаті середня кількість створених контейнерів обома стратегіями суттєво не відрізняється, однак увагу привертає кількість відхилень для кластерів середніх та великих розмірів. З діаграми (рис. 3) видно, що середня кількість відхилень, здійснених при використанні одновимірної топології, значно збільшується.

Спробуємо прив'язати розмірність кластера до початкового значення ширини топологічної околиці функції Гауса (1). Таким чином, візьмемо, наприклад, значення топологічної околиці з попереднього експерименту $\sigma_0 = 4$, та величину кластера в 10 вузлів, оскільки в даній точці стратегії є однаково ефективними. Тоді залежністю для початкового значення топологічної околиці (2) буде $\sigma_0 = 0.4c$, де c – кількість вузлів в кластері. Проведемо експеримент упаковки до першого відхилення для кластеру різної розмірності з вузлами місткістю 16 Гб (рис. 4).

У загальному випадку тенденція росту змінюється лише для одновимірної топології. На всьому проміжку вузлів балансування з використанням одновимірної топології є більш ефективним. Збільшення розмірності одного вузла вдвічі, як і в попередньому випадку не приводить до зміни результатів.

Провівши експеримент заповнення до повноти кластера для такої ж конфігурації, можна підмітити, що середнє значення кількості створених контейнерів суттєво не відрізняється від варіації топології. Однак кількість відхилень здійснених стратегією з використанням топології квадратної матриці значно зменшилась (рис. 5). Так, для 100 вузлів кількість відхилень при використанні одновимірної топології є майже в два рази більшою, ніж при використанні квадратної матриці.

Висновки та пропозиції. У ході дослідження було описано основні етапи роботи СКК, алгоритму балансування динамічної послідовності контейнерів з використанням карт та двох топологій: одновимірної та квадратної матриці. У ході експерименту заповнення до першого відхилення було виявлено, що для кластерів невеликої розмірності використання різних структур є однаково ефективним. Для кластерів середньої розмірності використання одновимірної топології показує кращі результати. У випадку кластерів великої розмірності максимальної ефективності вдається досягти з використанням топології квадратної матриці, коли ширина топологічної околиці є сталим значенням. І навпаки, при використанні залежності початкового значення ширини топологічної околиці від кількості вузлів в кластері – більш ефективною є використання одновимірної топології. Водночас у ході проведення експерименту упаковки до повноти кластеру було виявлено, що кількість створених контейнерів для різних структур є в більшості випадків однаковою з незначним відхиленням. Тоді коли кількість відхилень для кластерів середнього та великого розміру є значно меншою при використанні топології квадратної матриці.

У результаті дослідження можна стверджувати, що топологія карти впливає на ефективність стратегії розподілення динамічної послідовності контейнерів, яка базується на СКК.

Для покращення ефективності алгоритму роботи СКК є сенс провести аналіз груп, які утворилися в ході навчання карти, та дослідити процес явної конфігурації вузлів у місцях групування.

Список літератури:

1. Таненбаум Э., Стеен М. Распределенные системы принципы и парадигмы. Санкт-Петербург, 2003. 877 с.
2. Воєводін Є.В. Проблеми сучасних методів розподілення динамічної послідовності ресурсів у системах оркестрування віртуальних контейнерів. Науковий журнал «Альманах науки». 2017. № 4. С. 54–58.
3. Хайкин С. Нейронные сети полный курс. Москва, 2006. 1104 с.
4. Воєводін Є.В. Використання самоорганізаційних карт Кохонена для балансування динамічної послідовності ресурсів у системах оркестрування віртуальних контейнерів. Науковий журнал «Альманах науки». 2017. №5. С. 45–50.
5. Jiang F., Berry H., Schoenauer M. The Impact of Network Topology on Self-Organizing Maps. URL: http://www.inrialpes.fr/Berry/Images/GEC09_EVVON (дата звернення: 05.02.2018).
6. Hasan S., Shamsuddin S. Multistrategy Self-Organizing Map Learning for Classification Problems. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3157650 (дата звернення: 05.02.2018).

СРАВНЕНИЕ ЭФФЕКТИВНОСТИ ТОПОЛОГИЙ САМООРГАНИЗУЮЩИХСЯ КАРТ КОХОНЕНА В СИСТЕМАХ ОРКЕСТРАЦИИ ВИРТУАЛЬНЫХ КОНТЕЙНЕРОВ

В статье описывается метод распределения динамической последовательности виртуальных контейнеров с использованием самоорганизующихся карт Кохонена. Описанный метод позволяет построить модель, основанную на неявных связях между виртуальными контейнерами и узлами в кластере, таким образом в некоторых случаях позволяя производить балансировку более эффективно. Также в работе рассматриваются одномерная топология карты и топология в виде квадратной матрицы. Эффективность работы метода, основанного на различных топологиях, тестируется в ходе проведения экспериментов заполнения к первому отклонению и заполнения к полноте кластера. Анализ полученных результатов показывает положительные и отрицательные стороны использования различных топологий, а также что эффективность распределения зависит от структуры карты. Описанная методология проведения экспериментов может быть использована для исследования кластеров и поиска эффективной конфигурации карты Кохонена. Сделанные выводы могут быть использованы для разработки новых, более эффективных эвристических методов распределения.

Ключевые слова: *распределение, балансирование, самоорганизующиеся карты Кохонена, искусственные нейронные сети, виртуализация, контейнер, система оркестрации.*

COMPARING EFFICIENCY OF DIFFERENT SELF-ORGANIZING KOHONEN MAP TOPOLOGIES IN VIRTUAL CONTAINERS ORCHESTRATION SYSTEMS

The article describes a method for distributing a dynamic sequence of virtual containers using self-organizing Kohonen maps. The described method makes it possible to construct a model based on implicit connections between virtual containers and nodes in a cluster, thus allowing more efficient balancing. The paper also covers one-dimensional self-organizing map topology and topology in the form of a square matrix. The effectiveness of a method based on different topologies is checked within experiments, which are packing until first rejection and packing until cluster is completely full. The analysis of the results shows the positive and negative aspects of using different topologies, and finds that the distribution efficiency depends on the structure of the map. The described methodology for conducting experiments can be used to find an effective configuration of the Kohonen map and its topology in case of resources distribution. The findings can be used to develop a new, more efficient heuristic distribution methods.

Key words: *distribution, balancing, self-organizing Kohonen map, artificial neural networks, virtualization, container, orchestration system.*