СЕКЦІЯ «ІНФОРМАТИКА»

УДК 004.75

PACS 07.05.Mh, 84.35.+i, 05.65.+b, 07.05.Fb, 07.05.Kf

ВОЄВОДІН Є. В.
Черкаський національний університет імені Богдана Хмельницького, аспірант
e-mail: targetjump@gmail.com

ПОРІВНЯННЯ ЕФЕКТИВНОСТІ СТРАТЕГІЙ РОЗПОДІЛЕННЯ З ВИКОРИСТАННЯМ САМООРГАНІЗАЦІЙНИХ КАРТ КОХОНЕНА В СИСТЕМАХ ОРКЕСТРУВАННЯ ВІРТУАЛЬНИХ КОНТЕЙНЕРІВ

Анотація. *У статті описуються основні етапи методу розподілення динамічної послідовності віртуальних контейнерів з використанням самоорганізаційних карт Кохонена та можливі параметри його конфігурації. Описаний метод дає змогу побудувати модель, що базується на неявних зв'язках між віртуальними контейнерами та вузлами в кластері. Також, у статті описані різного роду експерименти, зокрема упаковка кластеру до першого відхилення та упаковка кластеру до повноти. Експерименти проводяться на кластерах різної розмірності, демонструючи особливості поведінки описаного методу в різних варіаціях його конфігурації.*

Описана методологія проведення експериментів може бути використана для дослідження кластерів та пошуку ефективної конфігурації карти Кохонена та топології кластеру. Зроблені висновки можуть бути використанні для розробки більш ефективних методів розподілення з використанням штучних нейронних мереж.

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

Вступ

Системи оркестрування віртуальних контейнерів (СОВК) займають важливу нішу в інформаційних технологіях, зокрема у високонавантажених та розподілених системах. Одиницею віртуалізації в СОВК є віртуальний контейнер (далі просто контейнер), який являє собою виконавче середовище, існуюче в межах однієї СОВК і є обмежене її ресурсами. Контейнер описує ряд вимог якими він має бути забезпечений для повноцінної роботи. У загальному випадку до таких вимог можна віднести кількість ядер центрального процесору (ЦП) та кількість оперативної пам'яті (ОП) потрібної йому для роботи. У залежності від конкретної СОВК, конфігурація контейнера може описувати набір додаткових вимог.

Основною задачею СОВК є забезпечення контролю повного життєвого циклу контейнерів у кластері. Однією з невід'ємних частин життєвого циклу контейнера є його створення. Процес створення включає в себе пошук вузла, тобто вільного місця в кластері, яке б задовольняло вимоги його конфігурації. Для виконання цієї задачі СОВК використовують різні стратегії розподілення.

Використання стратегії розподілення заповненням чи то стратегії розподілення поширенням є стандартним рішенням для СОВК. Перевага першої в тому, що її змогами досягається використання малої кількості вузлів, що зменшує потребу в додаткових апаратних ресурсах, але концентрація багатьох контейнерів на одному вузлі породжує низький рівень відмовостійкості. У свою чергу, більш високий рівень відмовостійкості [1] досягається з використанням стратегії розподілення поширенням, за рахунок розміщення контейнерів на вільних вузлах, але в такому випадку з'являється проблема простою вільних апаратних ресурсів. Обидві стратегії не аналізують

динамічну послідовність та не будують моделі зв'язків, у результаті з'являються випадки, в яких балансування здійснюється неефективно. Це негативно впливає на СОВК, що змушує систему приймати критичне рішення відносно запиту балансування, зокрема: не приймати нових запитів, затримати запит до тих пір поки не з'являться апаратні ресурси, або ж почати створення нового вузла з подальшим запуском контейнера на ньому, що також потребує часу. Використання евристичних методів, включаючи самоорганізаційні карти Кохонена (СКК), дозволяє аналізувати зв'язки елементів динамічної послідовності та діяти більш ефективно.

Аналіз останніх досліджень і публікацій

Про неефективність існуючих методів у своїй роботі описує Є.В. Воєводін. Автор порівнює між собою методи розподілення заповненням і розподілення поширенням, наводячи їх переваги та недоліки. У ході порівняння автор показує, за яких умов існуючі стратегії здійснюють неефективне балансування, та описує можливу поведінку СОВК у критичних випадках. У роботі також описуються можливі способи оптимізації процесу балансування, зокрема з використанням евристичних методів.

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

Алгоритм параметричної оптимізації на основі моделі поведінки рою світлячків описують В. Курейчик та інші [2]. В основі поведінки рою лежить самоорганізація, яка забезпечує досягнення спільних цілей на основі взаємодій низького рівня. Автори розробляють метод, який базується на динамічній області прийняття рішення кожним агентом, що гарантує знаходження всіх локальних оптимумів цільової функції за поліноміальний час. Ефективність методу автори доводять експериментально, порівнюючи його з мурашиним алгоритмом та методом рою часток.

Метою статті є опис, експериментальна перевірка та порівняння різних конфігурацій методу балансування динамічної послідовності віртуальних контейнерів з використанням СКК.

Теоретичні основи методу

У даному дослідженні для розв'язання задачі розподілення динамічної послідовності контейнерів використовується СКК. СКК – це штучна нейронна мережа [4-5], яка навчається без учителя і за своєю суттю добре підходить для розв'язання задач кластеризації. СКК дають змогу відображати простір більшого виміру в простір меншого виміру, зберігаючи топологічні властивості вхідного простору.

Оскільки СКК може приймати на вхід багато елементів, то можна представити вимоги контейнера у вигляді вхідних даних для СКК, відповідно вузли в кластері є вихідним шаром (рис. 1), або ж так званою картою. Тобто карта – є відображенням динамічної послідовності контейнерів в одновимірний простір вузлів. Таке відображення дозволяє формувати групи схожих за вимогами контейнерів на основі неявних залежностей.

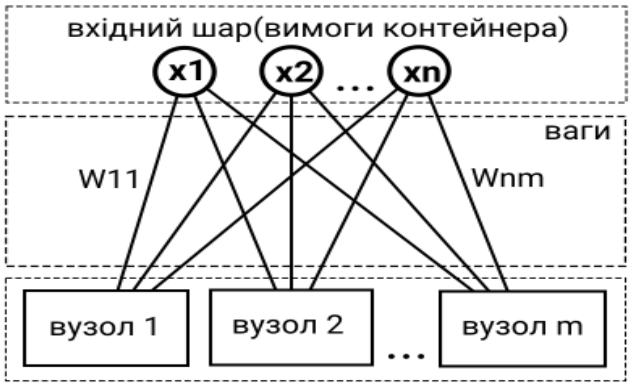

Рис. 1. Візуальна схема СКК

Для початку опишемо основні етапи роботи алгоритму СКК та можливі варіації конфігурацій на різних етапах алгоритму.

Етап перший – ініціалізація вагових коефіцієнтів w_{ij}. Ініціалізувати вагові коефіцієнти СКК можна по-різному. Одним з основних методів ініціалізації є ініціалізація випадковими маленькими значеннями [6], наприклад, з проміжку [0,1]. У такому випадку є сенс у тому, щоб нормалізувати множину вхідних даних, тоді всі значення з множини також будуть з числового проміжку [0,1]. Для нормалізації вхідних значень можна використати формулу:

$$\vec{z}_i = \frac{\vec{x}_i - \min(\vec{x})}{\max(\vec{x}) - \min(\vec{x})},$$

де $\vec{x}$ – вектор вхідних даних; $\vec{z}$ – нормалізований вектор.

Іншим основним методом ініціалізації є встановлення значень вагових коефіцієнтів рівними значенням випадково обраних векторів із 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 усередині якогось із отриманих відрізків.

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

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

де c – 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 $h_{cj}(t)$, тоді функція константного впливу (ФКВ) буде мати вигляд:

$$h_{cj}(t)=\begin{cases}0, & d>\sigma(t)\\ const, & d\le\sigma(t)\end{cases},$$

де c – індекс нейрона-переможця; j – індекс нейрона, що пробуджується; d – відстань від нейрона-переможця до нейрона з індексом j. Опишемо відстань наступною формулою:

$$d=\left\|ne_c-ne_j\right\|,$$

де ne_c – позиція нейрона-переможця, ne_j – позиція нейрона, що пробуджується. У свою чергу, $\sigma(t)$ відповідає за радіус впливу, в якості такої функції оберемо наступну:

$$\sigma(t)=\frac{R}{1+\frac{t}{T}},$$

де $R=const$ – початковий радіус; T – загальна кількість ітерацій.

Основна ідея ФКВ у тому, що нейрони які підлягають адаптації, змінюють значення вагових коефіцієнтів незалежно від того, наскільки далеко вони знаходяться від нейрона-переможця.

У якості функції впливу також може виступати і функція Гауса (ФВГ) [5]:

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

де $\sigma(t)$ – це функція визначення топологічної околиці, в якості якої можна використати функцію експоненціального розпаду [7]:

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

де $\sigma_0=const$ – початкова ширина топологічної околиці; t – номер ітерації; T – загальна кількість ітерацій.

На відмінну від ФКВ, ФВГ дає змогу 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 у напрямку вхідного вектору. Коригування 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, у свою чергу, дозволяє керувати інтенсивністю адаптації вагових коефіцієнтів на відповідній ітерації.

У якості такої функції, як і в ФВГ, можна використати експоненціальну функцію розпаду, але дещо з іншими параметрами:

$$\alpha(t) = \alpha_0 e^{\frac{t}{n}},$$

де $\alpha_0 = const$ – 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.

Основні результати дослідження

Для проведення експерименту була реалізована стратегія балансування у СОВК *Docker Swarm* [8]. Реалізація складається з двох основних частин. Перша – ініціалізація стратегії, у ході ініціалізації здійснюється конфігурація та навчання СКК на основі попередньо вказаної множини конфігурацій. Друга – розподілення контейнерів. Щоб розподілити один контейнер 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 повернути помилку, в іншому ж випадку – повернути відфільтрований список вузлів.

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

Опис вимог для ГПВК

Таблиця 1

Класифікація	**Ймовірність**	**Опис вимог**
Низькі вимоги	50%	[ОП 1], [ОП 2], [ОП 3]
Середні вимоги	33%	[ОП 4], [ОП 6]
Високі вимоги	12%	[ОП 8], [ОП 16]
Надвисокі вимоги	5%	[ОП 32]

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

Перший експеримент полягає у тому, щоб здійснювати паралельну упаковку до першого відхилення запиту. Експеримент проводиться для кількох 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 у динамічному ряді, оскільки у кластері просто немає вузла, який відповідає її вимогам. Провівши такий експеримент

кілька разів, скажімо, 1000, можна визначити, для якої кількості динамічних послідовностей контейнерів одна стратегія була більш ефективна за іншу, тобто, частіше розподіляла більше контейнерів.

Для початку визначимо залежності для коефіцієнтів R для ФКВ та σ_0 для ФВГ. Нехай у нас є кластер, який складається із 10 вузлів 32Гб ОП кожен, тоді, провівши описаний експеримент для стратегії розподілення заповненням попарно з ФВГ та ФКВ, можна визначити, при яких значеннях коефіцієнтів стратегії є максимально ефективними.

З графіку (рис. 2) видно, що ефективність розподілення стратегії, яка базується на ФВГ, є максимальною при $\sigma_0 = 4$, тоді коли з використанням ФКВ максимальної ефективності вдається досягти при $R = 3$.

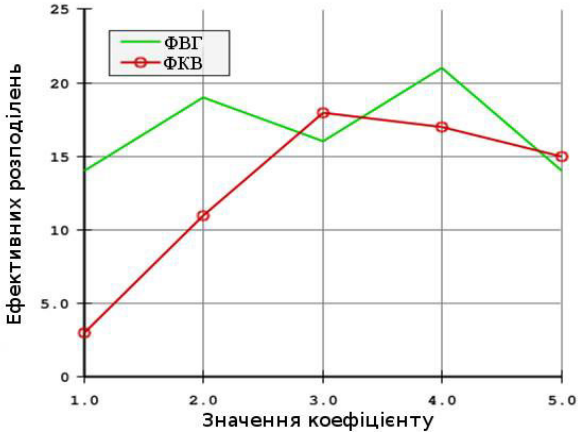

Рис. 2. Ефективність розподілення ФВГ та ФКВ для різних коефіцієнтів

Оскільки у описаному алгоритмі СКК використовується з одновимірною топологією, то має сенс визначення залежності величини коефіцієнта від розміру кластера. З отриманих даних для ФВГ така залежність буде мати вигляд $\sigma_0 = 0.25n$, а для ФКВ $R = 0.33n$, де n – це кількість вузлів у кластері.

Для дослідження поведінки функцій для кластерів інших розмірностей на базі отриманих залежностей проведемо аналогічний експеримент, здійснюючи одночасне розподілення стратегіями з використанням ФВГ та ФКВ. Кожний кластер, що бере участь в експерименті, складається із вузлів розмірністю 32 Гб ОП, кількість вузлів варіюється.

Для роз'яснення значень на графіку (рис. 3) візьмемо, наприклад, значення ефективності розподілень для 50 вузлів. Для 225 динамічних рядів із 1000 використання ФВГ було більш ефективне, ніж ФКВ, у свою чергу для 272 – навпаки, а в 503 випадках використання обох функцій було однаково ефективними. Однак, як можна бачити з графіку, для невеликих розмірів кластеру використання обох функцій є однаково ефективним, але з ростом розміру кластера більш ефективним є використання ФКВ.

Проведемо той же дослід, але застосовуючи в ході навчання карти функцію швидкості росту $\alpha(t) = 1$, тобто модифікація вагових коефіцієнтів базується тільки на значенні, що повертається функцією впливу.

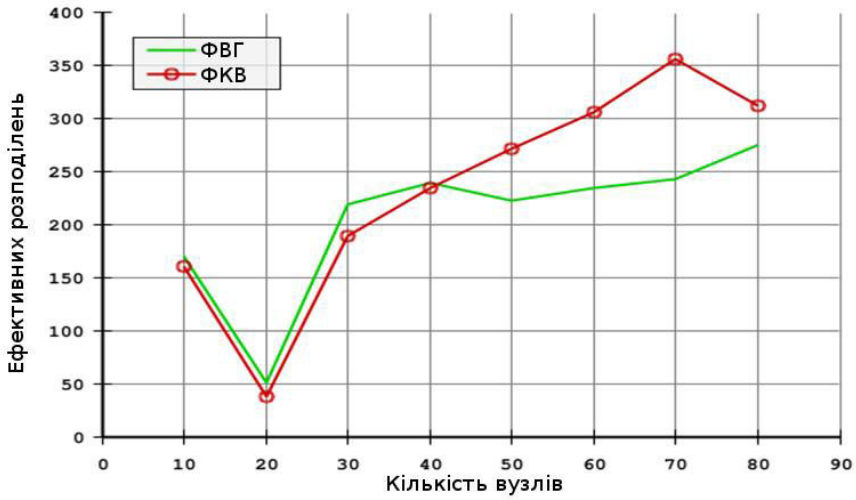

Рис. 3. Ефективність розподілення ФВГ та ФКВ

З графіку (рис. 4) видно, що для малих та середніх розмірів кластеру використання ФВГ дозволяє досягти максимально ефективних результатів, тоді як ФКВ продовжує бути більш ефективною для великої кількості вузлів. Так, наприклад, для 20 вузлів використання ФВГ дає змогу досягти не менш ефективного розподілення, ніж з використанням ФКВ для 956 динамічних рядів із 1000, тоді коли для 70 вузлів використання ФКВ є не менш ефективним у 705 випадках.

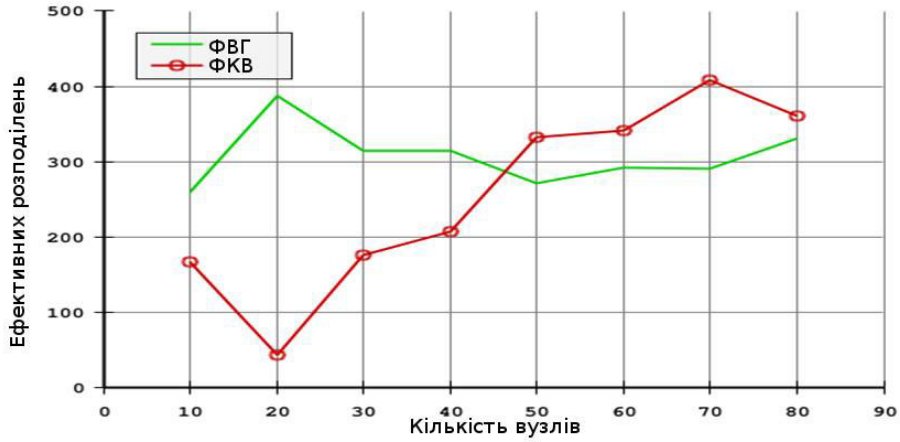

Рис. 4. Ефективність розподілення ФВГ та ФКВ
зі сталою швидкістю росту рівній 1

Провівши описаний експеримент ще раз для однієї і тієї ж функції, але з використанням різних методів ініціалізації, можна бачити (рис. 5), що значення розподілень знаходяться приблизно в тому ж діапазоні, з невеликим відхиленням у бік ініціалізації векторами з вибірки для ФКВ.

Інший експеримент полягає у тому, щоб запаковувати кластери до тих пір, поки у кластері не буде місця хоча б для конфігурації з найменшими потребами, при цьому, як і у попередньому експерименті, використовувати одну й ту ж динамічну послідовність конфігурацій контейнерів. Провівши такий експеримент достатню кількість разів, скажімо, 1000, і взявши середнє значення розподілень, можна проаналізувати, яка із стратегій в середньому може розподілити більше контейнерів, при цьому зробивши меншу кількість помилок.

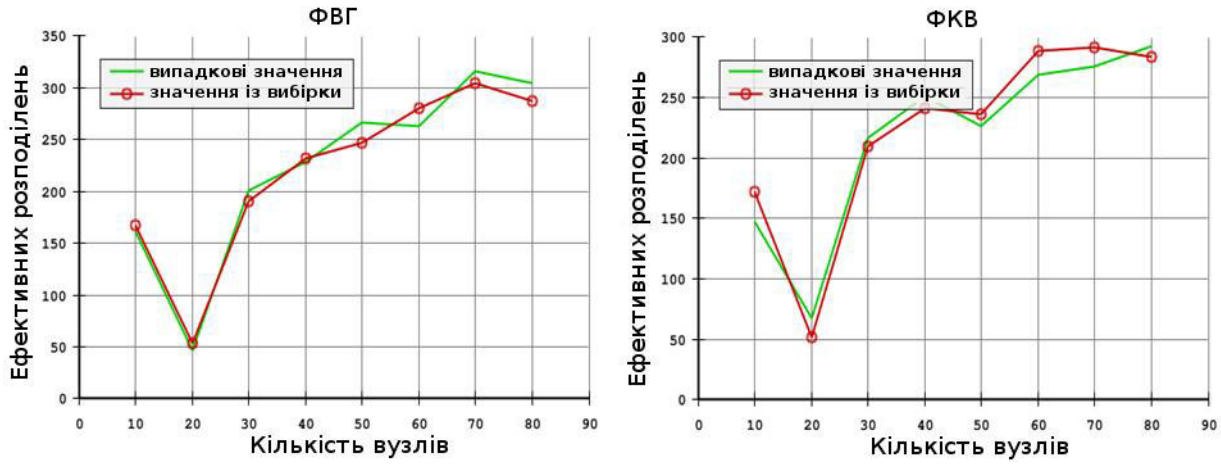

Рис. 5. Ефективність розподілення ФВГ та ФКВ
з використанням різних методів ініціалізації

Почнемо з того, що знайдемо залежність між коефіцієнтом для функцій впливу та кількістю вузлів у кластері, базуючись на середньому значенні запакованих контейнерів та зроблених помилок.

З графіків (рис. 6) видно, що кількість розподілених контейнерів у середньому складає близько 65 контейнерів для кластеру із 10 вузлів по 32Гб кожен. Однак, максимальне середнє значення для ФКВ досягається при $R=1$, відповідно залежність радіусу від кількості вузлів має наступний вигляд $R=0.1n$, у свою чергу для ФВГ $\sigma_0=0.25n$, де n – це кількість вузлів у кластері.

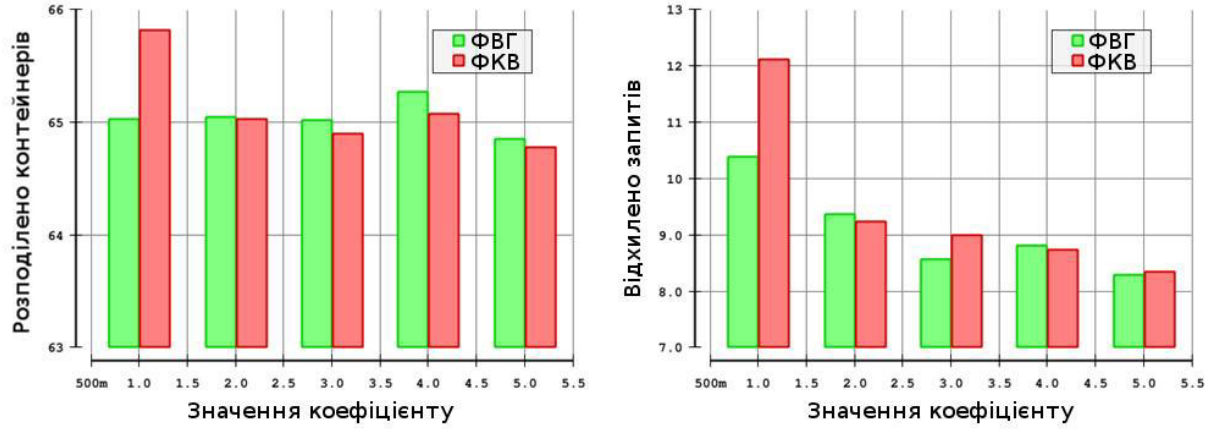

Рис. 6. Ефективність розподілення ФВГ та ФКВ для різних коефіцієнтів

Визначивши залежності для коефіцієнтів функцій впливу, проведемо експеримент для кластерів різної розмірності. Результат такий, що кількість середніх розподілень стратегіями з використанням ФКВ та ФВГ є майже еквівалентними, з мізерною перевагою ФКВ. Але при цьому, у більшості випадків, заповнення кластеру з меншою кількістю помилок вдається здійснити з використанням ФВГ, особливо для кластерів невеликої розмірності (рис. 7).

При використанні $\alpha(t)=1$ середні значення майже не змінюються, окрім того, що кількість відхилених запитів при використанні ФКВ для кластерів великого розміру зменшується.

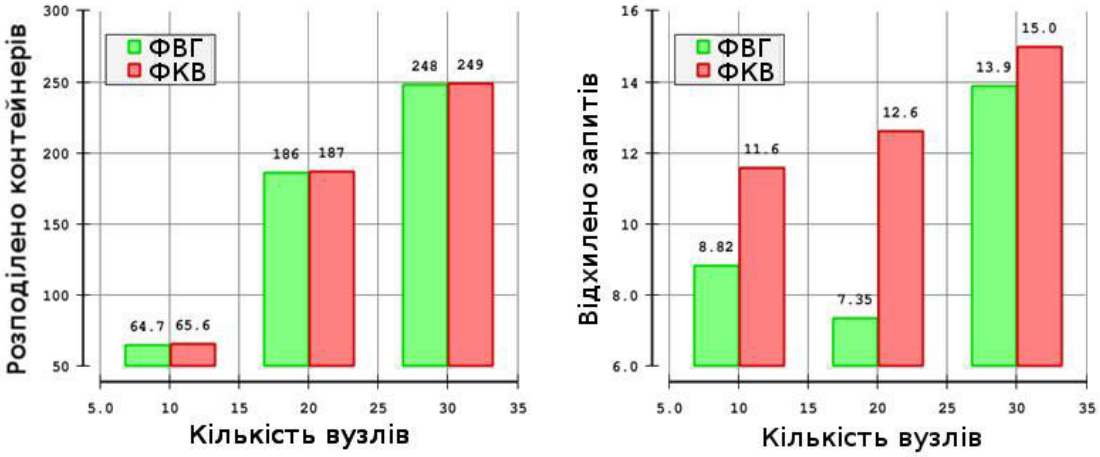

Рис. 7. Графіки розподілень та відхилень для невеликих кластерів

При проведені того ж експерименту та порівнянні функцій самих із собою при використанні різних методів ініціалізації, можна спостерігати значне зменшення кількості відхилень для ФКВ з використанням ініціалізації випадковими значеннями, коли для ФВГ кількість відхилень не значно зростає.

Висновки

У ході дослідження було описано основні етапи роботи СКК та можливі параметри їх конфігурації. Також був описаний алгоритм роботи стратегії розподілення, що базується на СКК. У ході проведених експериментів було виявлено, що для кластерів невеликої розмірності (10-30 вузлів) використання функції впливу Гауса дозволяє досягти максимальної ефективності, особливо коли швидкість навчання ігнорується. Для кластерів більшої розмірності (70-80) максимальної ефективності вдається досягти з використанням ФКВ. Водночас використання ініціалізації випадковими значеннями дає змогу зменшити кількість відхилень запитів на розподілення. У результаті дослідження можна стверджувати, що ефективність стратегії розподілення динамічної послідовності контейнерів з використання СКК залежить від конфігурацій останньої та топологічних властивостей кластеру.

Для покращення ефективності алгоритму роботи СКК є сенс провести дослідження різних топологій карти з подальшим аналізом груп, які утворилися в ході її навчання.

Список використаної літератури:

1. Таненбаум Э. Распределенные системы принципы и парадигмы / Э. Таненбаум, М. ван Стеен. – СПб: Питер, 2003. – 877 с.
2. Курейчик В. В. Алгоритм параметрической оптимизации на основе модели поведения роя светлячков / В. В. Курейчик, Д. В. Заруба, Д. Ю. Запорожец. // Известия ЮФУ. Технические Науки. – 2015. – №6. – С. 263.
3. Ritter H. Neural Computation and Self-Organizing Maps. An Introduction / H. Ritter, T. Martinetz, K. Schulten., 1992. – (Addison-Wesley).
4. Городич О. Самоорганізація нейромереж та класифікація даних / О. Городич, Ю. Щербина. // ВІСНИК ЛЬВІВ. УН–ТУ. Сер. прикл. матем. інформ. – 2003. – №7. – С. 234–247.
5. Хайкин С. Нейронные сети полный курс / Саймон Хайкин. – Москва: Вильямс, 2006. – 1104 с.
6. Sun M. Improving the Self-Organizing Feature Map Algorithm Using an Efficient Initialization Scheme / M. Sun, T. Liu, H. Chang. // Tamkang Journal of Science and Engineering. – 2002. – Vol. 5, №1. – С. 35–48.
7. Hasan S. Multistrategy Self-Organizing Map Learning for Classification Problems [Електронний ресурс] / S. Hasan, S. Shamsuddin. – 2011. – Режим доступу до ресурсу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3157650/.
8. Docker Swarm overview [Електронний ресурс] – Режим доступу до ресурсу:

https://docs.docker.com/swarm/overview/

References:

1. Tanenbaum E. & Steen M. (2003). *Principles and paradigms of distributed systems. St. Petersburg*: Piter. (in Rus)
2. Kurejchyk V. V. & Zaruba D. V.&Zaporozhec D. Ju. (2015). Algorithm of parametric optimization based on the firefly swarm pattern. *Izvestija JuFU. Tehnicheskie Nauki (Tidings of SFedU. Engineering Sciences)*, 6. (in Rus)
3. Ritter H. & Martinetz T. & Schulten K. (1992). *Neural Computation and Self-Organizing Maps. An Introduction*. Boston, MA, USA: Addison-Wesley Longman Publishing.
4. Horodych O. & Scherbyna Yu. (2003). Self-organizing neural network and its application for data classification. *VISNYK L'VIV. Ser. prykl. matem. inform. (VISNYK LVIV UNIV. Ser. Appl. Math. Comp. Sci)*, 7, 234-247. (in Ukr)
5. Hajkin S. (2006). *The complete course of neural networks*. Moscow: Vil'jams. (in Rus)
6. Sun M. Liu T. Chang H. (2002). Improving the Self-Organizing Feature Map Algorithm Using an Efficient Initialization Scheme. *Tamkang Journal of Science and Engineering, 5(*1), 35–48.
7. Hasan S. (2011). *Multistrategy Self-Organizing Map Learning for Classification Problems*. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3157650/.
8. *Docker Swarm overview.* (n.d). Retrieved from https://docs.docker.com/swarm/overview/.

VOIEVODIN Yevhenii,
The Bohdan Khmelnytsky National University of Cherkasy, PhD-student

COMPARISON OF THE EFFICIENCY OF DISTRIBUTION STRATEGIES BASED ON SELF-ORGANIZING KOHONEN MAPS IN VIRTUAL CONTAINERS ORCHESTRATION SYSTEMS

Abstract. *The paper starts from the description of the basic functions of orchestration systems, including distribution of dynamic virtual containers sequence. The positive and negative aspects of existing containers distribution strategies are discussed next. Recent studies and publications mention inefficiency of existing distribution strategies and describe alternative methods that can be used to optimize them. These methods include self-organizing maps.*

The paper covers a detailed overview of the structure of Kohonen maps and the main stages of their work, which are: initialization, competition, cooperation and adaptation. The author describes detailed map configuration options for each of the stages, such as different initialization approaches, different influence functions and their parameters. Then the author defines the algorithm for distributing containers between nodes in a cluster based on self-organized maps.

Also, the article covers two types of experiments, they are: packing cluster until the first packing request is rejected and packing cluster unit it is full. The first one demonstrates in how many cases, the differently configured strategies, will pack more containers for the same dynamic sequence. The second experiment shows the average number of containers that can be packed. The experiment is done for the same dynamic sequences; differently configured strategies try to fill up dedicated clusters until they are full. As a result experiments show that strategies based on self-organizing maps behave differently in case different map configurations are used. So, for instance, using the Gaussian influence function is more efficient for small clusters, while using the constant influence function is more efficient for large clusters. The findings can be used to develop more efficient methods using artificial neural networks.

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

Стаття надійшла 10.04.2017
Прийнято до друку 29.05.2017