doi:https://doi.org/10.15407/emodel.41.02.023

УДК 004.75

Я.Ю. Дорогий, канд. техн. наук,
О.О. Дорога-Іванюк, аспірантка, **Д.А. Ференс**
Національний технічний університет України
«Київський політехнічний ін-т ім. Ігоря Сікорського»
(Україна, 03056, Київ, пр. Перемоги, 37,
тел. +380970060025; e-mail: cisco.rna@gmail.com;
dorogaya.helen@gmail.com)

Модель розподілу ресурсів критичної IT-інфраструктури з чіткими параметрами на основі методу рою часток

Подано детальний аналіз методів та алгоритмів розміщення ресурсів віртуалізованих IT-інфраструктур. Наведено опис математичної моделі розподілу ресурсів критичної IT-інфраструктури з чіткими параметрами та її використання у поєднанні з методом рою часток. Розкрито суть методу рою часток, розглянуто принцип пошуку найкращого рішення та його основні операції для розв'язку поставленої задачі. Наведено результати експериментальних досліджень запропонованої моделі розподілу ресурсів критичної IT-інфраструктури з чіткими параметрами на основі методу рою часток.

К л ю ч о в і с л о в а: архітектура, розподіл ресурсів, метод рою часток, критична IT-інфраструктура.

Постановка проблеми. Віртуалізація ресурсів стала широко поширеною технологією надання доступу до обчислювальних ресурсів. Здебільшого це обумовлено тим, що віртуалізовані системи можуть забезпечити практично будь-який вид обслуговування і позбавити клієнтів необхідності створювати фізичні інфраструктури. У зв'язку зі збільшенням попиту на віртуалізовані сервіси, оптимізація використання ресурсів віртуалізованих IT-інфраструктур стає ще більш важливою, оскільки вирішення цієї проблеми дасть змогу підвищити продуктивність, доступність та надійність таких систем. Технології віртуалізації виявились досить зручними для вирішення таких проблем. Віртуалізація серверів дозволяє ефективно розподіляти фізичні обчислювальні ресурси між користувачами: декілька віртуальних машин (ВМ) можуть працювати ефективно та ізольовано, незалежно від їх фізичного розміщення. Зважаючи на це, провайдери хмарних сервісів мають можливість обслуговувати більшу кількість клієнтів гнучким та ефективним способом.

Різні користувачі можуть мати різні вимоги щодо об'єму обчислень, пам'яті або розміру сховища даних. Фізичні сервери також можуть мати різні можливості. Це зумовлює проблему оптимізації, відому також як проблема розміщення ВМ з врахуванням критичності ресурсів, які потрібно на них розмістити. Вирішення цієї проблеми дасть змогу збільшити об'єм використання фізичних ресурсів, зменшити споживання електроенергії, підвищити надійність ІТ-інфраструктур. Надзвичайно актуальною дана проблема є для критичних ІТ-інфраструктур, де вимоги до надійності функціонування є критичними.

Аналіз останніх досліджень. Програмування з обмеженнями — техніка, що часто використовується для розміщення ВМ у віртуалізованих середовищах. Вона найвигідніша для комбінаторних задач, розв'язки яких повинні задовольняти декільком обмеженням щодо змінних. Зазвичай задані обмеження можна легко доповнити для забезпечення додаткових потреб. Провайдери мережевих інфраструктур намагаються автоматизувати менеджмент ВМ, враховуючи кінцеву якість надання послуг. Таку проблему можна подати як задачу оптимізації з обмеженнями.

Функція, що оптимізується, обирається так, щоб досягти повного забезпечення вимог до якості сервісу з мінімальними витратами на обслуговування. Значення цієї функції являє собою числове подання ступеня забезпечення вимог щодо якості обслуговування. Алгоритми розміщення ВМ при цьому складаються з двох модулів:

1) модулю локальних рішень — кожен сервіс віртуалізованої ІТ-інфраструктури асоціюється з функцією, значення якої дорівнює ступеню забезпечення якості надання ресурсів цьому сервісу;

2) модулю глобальних рішень — за допомогою локальних рішень усіх сервісів у віртуалізованій ІТ-інфраструктурі виконується пошук оптимального значення глобальної функції.

При локальній оптимізації необхідно врахувати деякі обмеження, а саме: кількість ВМ кожного класу повинна відповідати вимогам фізичної машини (ФМ); кожен сервіс критичної ІТ-інфраструктури встановлює верхню межу на кількість ВМ кожного класу та загальну кількість ВМ.

Під час глобальної оптимізації необхідний об'єм кожного ресурсу виділених ВМ не повинен перевищувати максимальний об'єм ресурсу ФМ, оскільки, зрештою, основною ціллю є зменшення кількості активних ФМ та повне забезпечення ресурсами всіх критичних процесів.

Оскільки задача є *NP*-складною [1], застосування повного перебору можливе лише для невеликих розмірностей. Для інших випадків відома велика кількість евристичних алгоритмів [2—8], що дають хороші результати:

Best Fit (найліпший підходящий);

First Fit (перший підходящий);

Best Fit Decreasing (найліпший підходящий за спаданням);

First Fit Decreasing (перший підходящий за спаданням);

Heaviest First (перший найбільший);

Worst Fit (найгірший підходящий).

Залежно від мети, алгоритми розміщення можуть бути поділені на дві категорії:

1) алгоритми оптимізації споживання енергії — пошук розміщення, яке дозволить системі споживати найменшу кількість енергії;

2) алгоритми оптимізації якості обслуговування ((QoS) Quality of Service)) — пошук розміщення, яке дозволить системі задовільнити вимоги користувачів щодо якості обслуговування.

Залежно від необхідності виконувати міграції (переміщення ВМ від однієї ФМ до іншої) алгоритми розміщення поділяються на такі категорії:

статичне розміщення — не враховуються стани ВМ та ФМ;

динамічне розміщення — виконується пошук оптимального рішення з урахуванням поточного розміщення ВМ.

В свою чергу, динамічні алгоритми розміщення бувають двох типів:

1) проактивне — розміщення та міграції ВМ виконуються до того моменту, коли система набуде певного стану або до виконання певного критерію;

2) реактивне — розміщення та міграції ВМ виконуються відповідно до виконання певного критерію або до досягнення певного небажаного стану, наприклад зміна розміщення або виконання міграцій можуть бути необхідні під час технічного обслуговування ФМ або після реєстрування погіршення якості надання послуг.

Система pMapper [3] досягає оптимальних витрат енергії через зменшення кількості міграцій під час пакування ВМ у фіксовану кількість ФМ. У роботі [6] запропоновано одноцільовий алгоритм, базований на MMAS (Max-min Ant System) метаевристиці, що дозволяє використовувати мінімальну кількість ФМ.

У роботі [9] наведено алгоритми розміщення, які максимізують значення метрики під назвою «задоволення» (satisfaction), що відображає відносну придатність ФМ до розміщення ВМ на ній. У роботі [10] запропоновано евристичний підхід до пошуку перевантажених вузлів в датацентрах.

У роботі [11] запропоновано алгоритм, за допомогою якого можна знаходити оптимальні переміщення ВМ у випадках відмови ФМ. У [12] надано рішення під назвою «Backward Speculation Placement», тобто при

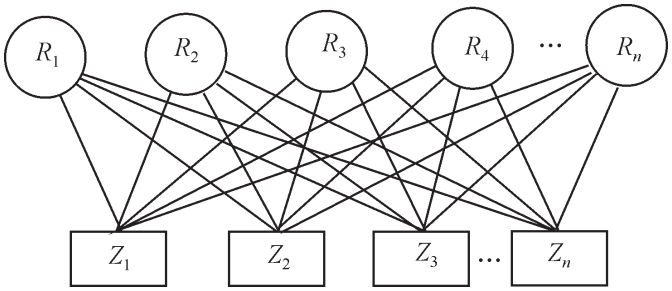

Рис. 1. Модель розподілу обмежених ресурсів IT-інфраструктури

виборі переміщень використовуються статистичні дані навантаження та метрика під назвою «ризик попиту».

Усі розглянуті алгоритми та методи спроектовано для оптимізації фіксованого набору ресурсів, таких як об'єм оперативної пам'яті, використання жорсткого диску або час CPU (Central Processing Unit). Неможливість гнучко налаштовувати набір ресурсів, а також відсутність поділу ресурсів за їх критичністю робить більшість розглянутих методів непрактичними для використання в критичних IT-інфраструктурах. Окрім того, наведені алгоритми розміщення не допускають введення додаткових специфічних умов оптимізації, що унеможливлює використання їх на практиці, а тому є вкрай необхідною розробка нової моделі розподілу ресурсів для критичної IT-інфраструктури.

Вибір моделі розподілу ресурсів. Для впровадження моделі розподілу ресурсів необхідно розглянути декілька ознак, які можуть вплинути на тип моделі. В залежності від методів ведення бізнесу (для власних бізнес-процесів чи для зовнішніх клієнтів) моделі можуть бути різними. На вибір моделі впливає також тип архітектури критичної IT-інфраструктури. Залежно від рівня абстракції ресурсів варіюється складність математичної моделі задачі, що розв'язується. Задачі великої розмірності розв'язуються у два етапи: на першому етапі здійснюється розподіл абстрактних ресурсів кожного типу без прив'язки до їх конкретного місцезнаходження, на другому — здійснюється уточнення отриманих результатів.

Дуже істотно впливає на вибір моделі ознака забезпеченості ресурсами, а саме, чи дозволена часткова підтримка сервісів, чи вони мають підтримуватися у повному обсязі, або не підтримуватися взагалі. У найпростішому випадку дворівневої клієнт-серверної архітектури модель розподілу обмежених ресурсів може подаватись як декілька бізнес-процесів, що безпосередньо використовують частину одного чи декількох незалежних один від одного ресурсів (рис. 1) [13].

Введемо необхідні для побудови моделей позначення:

$Z_1 \dots Z_n$ — бізнес-процеси, підтримку яких забезпечує функціонування інформаційно-телекомунікаційної системи (ІТС);

$W = (w_1, ..., w_n)$ — коефіцієнти важливості бізнес-процесів $Z_1 \dots Z_n$;

$R_1 \dots R_m$ — інтегровані ресурси ІТС, необхідні для підтримки функціонування бізнес-процесів;

$P = \|p_{ij}\|$ — матриця потреб бізнес-процесів у ресурсах ІТС, де p_{ij} відповідає кількості потрібного для процесу Z_i ресурсу R_j чи 0, якщо ресурс не потрібен;

$D = \|d_{ij}\|$ — матриця наявності потреб бізнес-процесів у ресурсах ІТС, де $d_{ij} = 1$, якщо $p_{ij} > 0$ і $d_{ij} = 0$, якщо $p_{ij} = 0$;

$R = (r_1, ..., r_m)$ — вектор встановлених обмежень на ресурси.

Розглянемо дискретний випадок, коли бізнес-процес або підтримується в повному обсязі, або повністю блокується. Введемо вектор $X = (x_1, ..., x_n)$, де $x_i = 1$, якщо процес Z_i обслуговується, $x_i = 0$ в протилежному випадку. Тоді критерій оптимального розподілу ресурсів ІТС можна подати у вигляді $\max \sum_{i=1}^{n} x_i w_i$. Будемо також використовувати обмеження по ресурсам

$$\sum_{i=1}^{n} x_i p_{ij} \le r_j,\ j = 1, ..., m.$$

Варто зазначити, що наведена вище модель, а також інші моделі, що розглядаються, належать до лінійних або нелінійних, неперервних, булевих та змішаних задач математичного програмування, які мають стохастичні аналоги. Для детермінованих задач можна використовувати евристичні алгоритми, методи м'яких обчислень та точні методи, використання яких обмежено з огляду на розміри задач. Так, для задач лінійного програмування можна використовувати відомі методи. Задачі булевого програмування можна розв'язувати методами часткового перебору. Точні методи дають змогу знайти найкращий розв'язок, але їх використання можливе лише для задач обмеженої розмірності, оскільки час пошуку рішень суттєво зростає зі збільшенням складності задачі. На відміну від них евристичні методи та методи штучного інтелекту, насамперед генетичні алгоритми (ГА) та метод рою часток (МРЧ), дають змогу знайти задовільний розв'язок впродовж короткого часу навіть задач великої розмірності, крім того, їхню ефективність можна поліпшити, враховуючи особливості задач.

Для управління віртуальними машинами при серверній віртуалізації використаємо модель, описану в [14].

Застосування технологій штучного інтелекту для розв'язку задачі багатомірного пакування. Задачі управління ресурсами можна сформулювати як задачі лінійного програмування та розв'язувати їх оптимально, знаходження точного розв'язку зазвичай є марною тратою ресурсів. Оскільки такі задачі належать до класу *NP*-повних, простір пошуку розв'язків є надзвичайно великим, містить значну кількість потенційних розв'язків та потребує значної кількості обчислювальних ресурсів. На практиці достатньо знайти розв'язок, близький до оптимального, за короткий проміжок часу.

Популяційний метод МРЧ просто реалізується та застосовується для розв'язання різноманітних задач оптимізації. Кожен індивід популяції, який називається «часткою», являє собою рішення у просторі пошуку. Популяція, що називається «роєм», складається з множини потенційних розв'язків задачі оптимізації. Основна ідея МРЧ — взаємодія та комунікація часток між собою. МРЧ дозволяє знаходити близьке до оптимального рішення за прийнятний час.

Кожна частка описується наступною групою векторів:

$$(\mathbf{x}_i, \mathbf{v}_i, \mathbf{p}_i), \tag{1}$$

$$\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{iD}), \tag{2}$$

$$\mathbf{v}_i = (v_{i1}, v_{i2}, ..., v_{iD}), \tag{3}$$

$$\mathbf{p}_i = (p_{i1}, p_{i2}, ..., p_{iD}). \tag{4}$$

Кожна частка має позицію (2), швидкість руху (3) та найкраще рішення, що було досягнуто часткою в просторі рішень (4). На початку роботи алгоритму позиції часток генеруються випадково. Розмірність векторів визначається розмірністю задачі оптимізації. Серед усіх часток найкраще рішення популяції має вигляд

$$\mathbf{g}_i = (g_{i1}, g_{i2}, ..., g_{iD}).$$

На кожній ітерації вектори позиції та швидкості кожної частки змінюються так:

$$v_{id}[t+1] = wv_{id}[t] + c_1 r_1 (p_{id} - x_{id}[t]) + c_2 r_2 (p_{gd} - x_{id}[t]),$$

$$x_{id}[t+1] = x_{id}[t] + v_{id}[t+1],$$

де $i = 1, 2, ..., m$ — кількість часток; p_i — позиція i-ї частки; p_g — найкраще рішення популяції; $d = 1, 2, ..., D$ — розмірність часток; r_1, r_2 — випадкові числа; c_1 — індивідуальна швидкість навчання; c_2 — соціальна швидкість навчання; w — коефіцієнт інерції.

Параметр c_1 визначає власну здатність індивіда до пошуку рішення, тоді як параметр c_2 визначає соціальну комунікацію, тобто вплив соціального середовища на рух частки. Визначення оптимальних значень обох параметрів є предметом багатьох досліджень. За допомогою МРЧ виконуємо пошук оптимального рішення, оновлюючи популяцію часток у кожній ітерації. Критерієм зупинки може бути досягнення максимальної кількості ітерацій або бажаного мінімального значення помилки.

Недоліком МРЧ є тенденція часток до досягнення локального оптимуму. Рух рою зупиняється, коли кожна частка досягає свого локального оптимуму. Для вирішення цієї проблеми часто використовують метод Local Improvement Procedure [15], що дає змогу кожній частці продовжити рух після досягнення локального оптимуму. Зрештою, це можна вирішити, підібравши значення параметру інерції: велике значення інерції сприяє пошуку глобального рішення, а мале значення — локальних. У роботі [16] зазначено, що МРЧ дає кращі результати при використанні не фіксованого значення, а такого, яке змінюється лінійно від одиниці до 0,4 з плином ітерацій. У роботі [17] показано, що МРЧ має здатність швидко сходитись та якість отриманих рішень не суттєво залежить від розміру популяції.

Для роботи МРЧ у динамічному середовищі часто використовують наступні методи:

1. Випадковий коефіцієнт інерції — найпростіший спосіб модифікації алгоритму для роботи в динамічному середовищі. У роботі [18] використано формулу $w = 0{,}5 + rnd/2{,}0$ для зміни коефіцієнта інерції. Це дозволяє відслідковувати оптимум, що змінюється в динамічній системі.

2. Рерандомізація — метод описує ефект рерандомізації для знаходження оптимуму в динамічному середовищі. У роботах [17, 18] запропоновано два методи: Changed-gBest-Value та Fixed-gBest-Value. Метод Changed-gBest-Value базується на припущенні, що оптимальне значення кожної частки змінюється тоді, коли змінюється оптимальне значення системи. При виявленні такої ситуації відбувається скидання пам'яті часток. Незважаючи на те, що метод швидкий, він потребує витрат на переобчислення функції пристосування. Метод Fixed-gBest-Value полягає у відслідковуванні найкращого глобального значення та найкращого глобального значення для фіксованої кількості ітерацій. Якщо протягом ітерацій ці значення не змінюються, може відбутись зміна оптимуму системи. В такій ситуації метод рерандомізує набір часток для відслідковування зміни оптимуму. Рерандомізацію також можна розглядати як заміну старих часток новими. У більшості випадків радять використовувати значення параметру, що дорівнює 1%.

3. Динамічний МРЧ з використанням опорних часток. Цей метод [19] базується на використанні фіксованої кількості стаціонарних опорних часток, які використовуються для виявлення змін стану динамічного середовища. При виявленні зміни середовища виконується очистка пам'яті, а саме кожна частка замінює значення своєї найкращої позиції значенням поточної позиції. Недоліком методу є те, що опорні частки можуть виявляти лише локальні зміни, тобто зміни в місцях, де вони розташовані. У такому випадку класична децентралізована модель обчислень МРЧ перетворюється у централізовану, що створює деякі труднощі при використанні у децентралізованому середовищі. У роботі [20] запропоновано метод Multi Ensemble Particle Swarm Optimization (мультіансамблевий МРЧ) для роботи у динамічному середовищі. Популяція часток поділяється на дві групи. Частки першої групи виконують пошук глобального оптимуму у просторі рішень в поточному середовищі. Це гарантує хороше сходження алгоритму. Для підвищення ефективності пошуку часток першої групи використано гаусів локальний пошук. Частки другої групи рухаються на межі простору пошуку часток першої групи, таким чином виявляючи зміни в глобальному оптимумі.

4. Децентралізований МРЧ без опорних часток. У роботі [19] описано динамічний адаптивний МРЧ (ДАМРЧ) для моніторингу та реагування на зміни в середовищі. Цей метод розроблено для пошуку рухомого глобального оптимуму. Головною характеристикою алгоритму є можливість оновлення пам'яті часток без централізованого управління. У ДАМРЧ немає спеціальних часток, які відслідковують зміни, або додаткової функції пристосованості. Натомість, кожна частка порівнює значення функції пристосованості поточної позиції до значення її найкращої позиції. Якщо

Приклад кодування розміщення десяти ВМ на чотирьох ФМ

Номер ВМ	x_{ij}^k	s_{ij}^k	r_{ij}^k
0	1,13	3	4
1	3,91	9	2
2	–0,21	1	2
3	–0,39	0	1
4	3,25	7	4
5	3,51	8	1
6	2,74	5	2
7	3,1	6	3
8	0,99	2	3
9	1,62	4	1

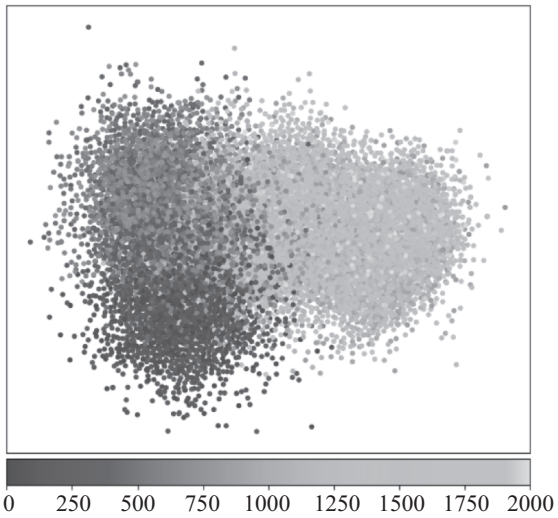

Рис. 2. Візуалізація руху часток МРЧ за допомогою алгоритму *t*-SNE

значення поточної позиції не поліпшується, частка почне змінювати значення пристосованості своєї найкращої позиції за таким законом:

$$P_i(t+1)=\begin{cases}P_i(t)*T,\ f(X_i(t+1))\le P_i(t)\,T,\\ f(X_i(t+1)),\ f(X_i(t+1))>P_i(t)\,T,\end{cases}$$

де T — фактор випаровування у межах [0, 1].

Найважливішим кроком проектування успішного МРЧ є вибір правильного подання рішення, яке точно описує відношення між частками МРЧ та задачею, що вирішується [21]. При розміщенні ВМ кожну частку буде подано у вигляді N-мірного вектора. Вектори позиції та швидкості визначаємо так:

$$x_i^k=(x_{i1}^k, x_{i2}^k, ..., x_{in}^k),\ v_i^k=(v_{i1}^k, v_{i2}^k, ..., v_{in}^k),$$

де x_i^k — позиція j-ї ВМ i-ї частки у k-й ітерації; v_i^k — швидкість j-ї ВМ i-ї частки у k-й ітерації.

Оригінальний МРЧ використовується для задач оптимізації, в яких елементами простору рішень є неперервні дійсні числа. Саме неперервна природа алгоритму є складною перешкодою для використання алгоритму у комбінаторних задачах. Існує багато методів виправлення цього недоліку, одним із яких є метод найменшого значення позиції (НЗП) (Smallest Position Value) [22]. За допомогою цього методу неперервні значення позиції перетворюються у таку послідовність розміщень:

$$s_i^k=(s_{i1}^k, s_{i2}^k, ..., s_{in}^k), \tag{5}$$

$$r_i^k = (r_{i1}^k, r_{i2}^k, ..., r_{in}^k), \quad (6)$$

$$r_{ij}^k = s_{ij}^k \bmod M. \quad (7)$$

Остаточний вектор ідентифікаторів ФМ (6) вираховується за формулою (7). Наприклад, нехай необхідно розмістити 10 ВМ на чотирьох ФМ. Приклад кодування вектора позиції i-ї частки у k-й ітерації наведено у таблиці. За правилом НЗП найменше значення позиції i-ї частки дорівнює $x_{i3}^k = -0,39$, отже, ВМ під номером три призначається першій ФМ. Друге найменше значення позиції дорівнює $x_{i2}^k = -0,21$, отже, ВМ під номером два розміщується на наступній ФМ.

Результати експериментальних досліджень. На початку роботи алгоритму вектори позиції та швидкостей для усієї популяції обирають випадково. Під час першої ітерації алгоритму найкращі значення позиції кожної частки встановлюють відповідно до її початкової позиції.

Значення параметрів компонентів алгоритму МРЧ:
кількість часток — 100;
кількість поколінь — 50;
індивідуальна швидкість навчання, $c_1 = 2,0$;
соціальна швидкість навчання, $c_2 = 2,0$;
коефіцієнт інерції — від 0,9 до 0,4 змінний протягом ітерацій;
випадкові числа r_1, r_2 — рівномірний розподіл у межах [0, 1];
вектор позиції — дійсні числа [0, 100];
вектор швидкості — дійсні числа [–100, 100].

В усіх тестових сценаріях відбувалась оптимізація розміщення за трьома ресурсами: обчислювальна здатність у кількості інструкцій на секунду, об'єм оперативної пам'яті та пропускна здатність мережі (Network). В усіх тестових сценаріях кількість ФМ фіксована та дорівнює дев'яти, а кількість сервісів змінюється за значеннями x_1, x_5, x_{10} та x_{20} залежно від кількості ФМ.

Для візуалізації руху часток МРЧ використано алгоритм зниження розмірності t-SNE [23], результат роботи якого наведено на рис. 2. За результатами моделювання можна зробити висновок, що частки у МРЧ справді демонструють просторову організацію рою та зберігають її протягом певного часу. Об'єми використаних ресурсів ФМ для кожного значення кількості сервісів наведено на рис. 3.

Результати моделювання свідчать про ефективність алгоритму. Для розміщення малої кількості сервісів алгоритм дозволяє використати якомога меншу кількість ФМ. Сервери, у яких відсутнє навантаження, можна вимкнути і таким чином зменшити споживання електроенергії у центрі обробки даних (ЦОД). При збільшенні кількості сервісів алгоритм дає

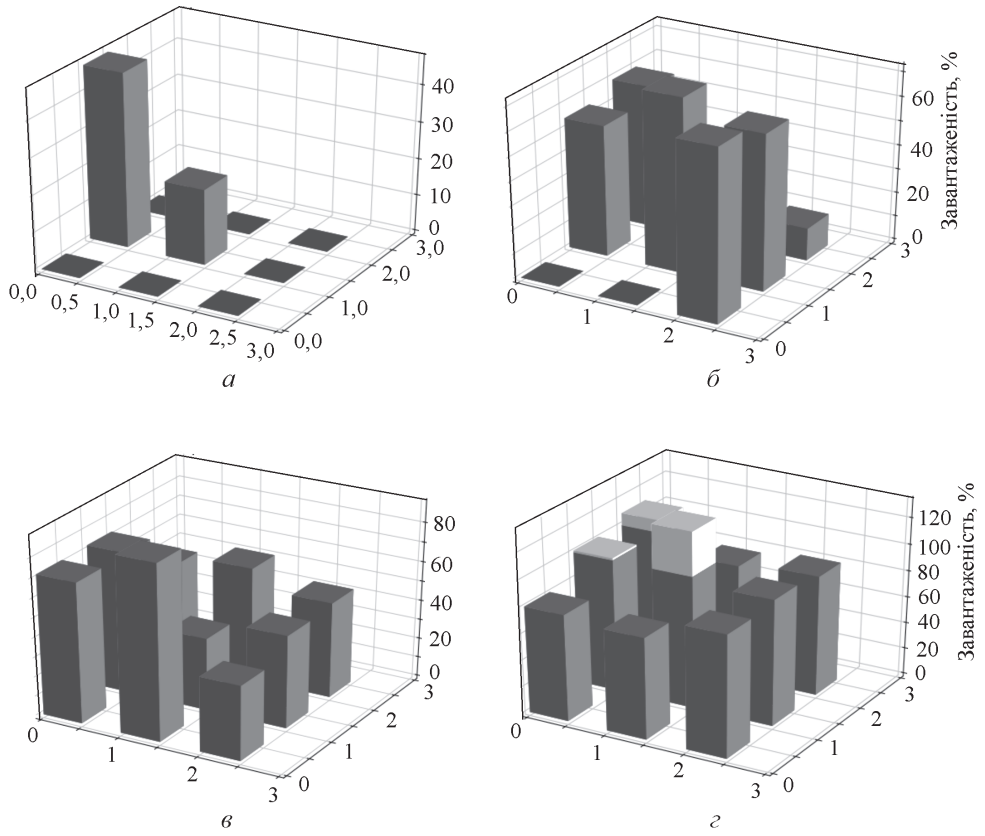

Рис. 3. Середнє споживання ресурсів ФМ для x_1 (*а*), x_5 (*б*), x_{10} (*в*) и x_{20} (*г*): ■ — ресурси, що використовуються; ■ — надмірне споживання ресурсу

змогу розмістити більшу кількість серверів. У випадку, коли споживання ресурсів значно перевищує можливості ЦОД (для x_{20}), розміщення обирається так, щоб мінімізувати сумарне перевантаження на кожній ФМ.

Висновки

Детальний аналіз методів та алгоритмів розміщення ресурсів віртуалізованих IT-інфраструктур дозволяє зробити висновок про можливість їх використання для критичних IT-інфраструктур. Запропонована та досліджена модель розподілу ресурсів критичної IT-інфраструктури з чіткими параметрами на базі методу рою часток свідчить про ефективність даного алгоритму та дозволяє так розмістити наявну множину серверів, щоб максимально зменшити споживання електроенергії у ЦОД та мінімізувати перевантаження серверного обладнання.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. *Глоба Л.С., Скуліш М.А., Дяденко О.М.* Математичні основи побудови інформаційно-телекомунікаційних систем. Київ: Норіта-плюс, 2007.
2. *Горин М.* Корпоративный ЦОД: за рамками технологий // Connect! Мир Связи. 2007, № 8, с. 19—20.
3. *Кириллов И.* Коммерческие ЦОД в Украине: новый этап развития // Сети и бизнес, 2010, № 3 (52). [Электронный ресурс] Режим доступа: http://www.sib.com.ua/arhiv 2010/2010_3/statia_3_1_2010/statia_3_1_2010.htm. Дата звернення: 10.12.2018. Назва з екрану.
4. *Badger L., Grance T. et al.* Computing Synopsis and Recommendations. Recommendations of the National Institute of Standards and Technology// Special Publication 800-146. NIST, 2012.
5. *Бибершmейн Н., Боуз С.* Компас в мире сервис-ориентированной архитектуры (SOA). М.: КУДИЦ-Пресс, 2007.
6. *Krafzik D., Banke K., Slama D.* Enterprise SOA: Service-Oriented Architecture Best Practices. Prentice Hall Professional, 2005, 383 p.
7. *Kaur R., Kaur A.* A Review Paper on Evolution of Cloud Computing, its Approaches and Comparison with Grid Computing // International Journal of Computer Science and Information Technologies, 2014, Vol. 5, p. 6060—6063.
8. *Gupta A., Sarood O., Laxmikant V.K.* Optimizing VM Placement for HPC in the Cloud // International Letters of Social and Humanistic Sciences, 2014, Vol. 16, p. 1—6.
9. *Joseph, J., Fellenstein C.* Grid Computing. Prentice Hall Professional, 2004, 378 p.
10. *Nabrzyski J., Schopf J.M.* Grid Resource Management: State of the Art and Future Trends Weglarz — Springer, 2004.
11. *Goldworm B., Skamarock A.* Blade servers and virtualization: transforming enterprise computing while cutting costs. Wiley Publishing, Inc., 2007.
12. *Ruest N., Ruest D.* Virtualization. A Beginner's Guide. McGraw Hill Professional, 2009.
13. *Теленик С.Ф., Ролік О.І., Букасов М.М., Лабунський А.Ю.* Моделі управління віртуальними машинами при серверній віртуалізації// Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка, 2009, № 51, с. 147—152.
14. *Дорогий Я.Ю., Дорога-Іванюк О.О., Ференс Д.*А. Модель розподілу ресурсів критичної IT-інфраструктури з чіткими параметрами на основі генетичного алгоритму // Information technology and security, 2018, Vol. 6, Iss. 2(11), p. 124—144.
15. *Buyya R., Broberg J., Goscinski A.M.* Cloud Computing: Principles and Paradigms. John Wiley & Sons, 2010, 664 p.
16. *Hu X., Eberhart R.C.* Tracking dynamic systems with PSO: Where's the cheese // Proc. of the Workshop on Particle Swarm Optimization. IN, 2001, p. 80—83.
17. *Shi Y.H., Eberhart R.* Empirical study of particle swarm optimization// Proc. of the Congress on Evolutionary Computation, 1999, Vol. 3, p. 1945—1950.
18. *Eberhart R.C., Shi Y.* Tracking and optimizing dynamic systems with particle swarms // Proc. of the 2001 Congress on Evolutionary Computation, 2001, Vol. 1, p. 94—100. DOI: 10.1109/CEC.2001.934376.
19. *Carlisle A., Dozler G.* Tracking changing extrema with adaptive particle swarm optimizer // Proc. of the 2002 Soft Computing, Multimedia Biomedicine, Image Processing and Financial Engineering. Orlando, FL, USA, 2002, p. 265—270.
20. *Du W., Li B.* Multi-strategy ensemble particle swarm optimization for dynamic optimization // International Journal of Information Sciences, 2008, Vol. 178, Iss. 15, p. 3096—3109. DOI: 10.1016/j.ins.2008.01.020.

21. *Tasgetiren M.F., Sevkli M., Liang Y.-C., Gencyilmaz G.* Particle swarm optimization algorithm for permutation flow shop sequencing problem // Proc. of the 4th International Workshop on Ant Colony. Optimization and Swarm Intelligence (ANTS2004), LCNS 3172. Brussels, Belgium, 2004, p. 382—390.
22. *Tasgetiren M.F., Sevkli M., Liang Y.-C., Gencyilmaz G.* Particle swarm optimization algorithm for single-machine total weighted tardiness problem // Proc. of the Congress on Evolutionary Computation, 2006, Vol. 2, p. 1412—1419.
23. *Van der Maaten, L.J.P., Hinton G.E.* Visualizing High-Dimensional Data Using t-SNE // Journal of Machine Learning Research, 2008, Vol. 9 (Nov), p. 2579—2605.

Отримано 04.01.19

REFERENCES

1. Globa, L.S., Skulish, M.A. and Dyadenko, O.M. (2007), *Matematicheskie osnovy postroeniya informatsionno-telekommunikatsionnykh sistem* [Mathematical Foundations of Construction of Information and Telecommunication Systems], Norita-plus, Kyiv, Ukraine.
2. Gorin, M. (2007), "Corporate Data Center: Beyond Technology", *Connect! Mir Svjazi*, no. 8, pp. 19-20.
3. Kirillov, I. (2010), "Commercial data centers in Ukraine: a new stage of development", *Seti i biznes*, no. 3, available at: http://www.sib.com.ua/arhiv 2010/2010_3/statia_3_1_2010/statia_3_1_2010.htm (accessed December 10, 2018).
4. Badger, L., Grance, T., Patt-Corner, R. and VoasCloud, J. (2012), "Computing Synopsis and Recommendations. Recommendations of the National Institute of Standards and Technology", *Special Publication.*
5. Bibershtein, N. and Bouz, S. (2007), *Kompas v mire servis-orientirovannoy arhitektury* [Compass in the world of service-oriented architecture], KUDITs-Press, Moscow, Russia.
6. Krafzik, D., Banke, K. and Slama, D. (2005), Enterprise SOA: Service-Oriented Architecture Best Practices, Prentice Hall Professional.
7. Kaur, R. and Kaur, A. (2014), "A Review Paper on Evolution of Cloud Computing, its Approaches and Comparison with Grid Computing", *International Journal of Computer Science and Information Technologies*, Vol. 5, pp. 6060-6063.
8. Gupta, A., Sarood, O. and Kale, L.V. (2014), "Optimizing VM Placement for HPC in the Cloud", *International Letters of Social and Humanistic Sciences*, Vol. 16, pp. 1-6.
9. Joseph, J. and Fellenstein, C. (2004), Grid Computing, Prentice Hall Professional.
10. Nabrzyski, J., Schopf, J.M. and Weglarz, J. (2004), Grid Resource Management: State of the Art and Future Trends, Springer.
11. Goldworm, B. and Skamarock, A. (2007), Blade servers and virtualization: transforming enterprise computing while cutting costs, Wiley Publishing, Inc.
12. Ruest, N. and Ruest, D. (2009), Virtualization, A Beginner's Guide, McGraw Hill Professional.
13. Telenik, S.F., Rolik, O.I., Bukasov, M.M. and Labunskiy, A.Yu. (2009), "Virtual Machine Management Models for Server Virtualization", *Visnik NTUU «KPI»: Informatika, upravlinnya ta obchislyuvalna tekhnika*, no. 51, pp. 147-152.
14. Dorogyy, Ya.Yu., Doroga-Ivanyuk, O.O. and Ferens, D.A. (2018), "Critical IT infrastructure resource allocation model with clear parameters based on genetic algorithm", *Information technology and security,* Vol. 6, no. 2(11), pp. 124-144.
15. Buyya, R., Broberg, J. and Goscinski, A.M. (2010), Cloud Computing: Principles and Paradigms, John Wiley & Sons.
16. Hu, X. and Eberhart, R.C. (2001), "Tracking dynamic systems with PSO: Where's the cheese", *Proceedings of the Workshop on Particle Swarm Optimization,* IN, pp. 80-83.

17. Shi, Y.H. and Eberhart, R. C. (1999), "Empirical study of particle swarm optimization", *Proceedings of the Congress on Evolutionary Computation*, Vol. 3, pp. 1945-1950.
18. Eberhart, R.C. and Shi, Y. (2001), "Tracking and optimizing dynamic systems with particle swarms", *Proceedings of the 2001 Congress on Evolutionary Computation*, Vol. 1, pp. 94-100. DOI: 10.1109/CEC.2001.934376.
19. Carlisle, A. and Dozler, G. (2002), "Tracking changing extrema with adaptive particle swarm optimizer", *Proceedings of the 2002 Soft Computing, Multimedia Biomedicine, Image Processing and Financial Engineering*, Orlando, FL, USA, pp. 265-270.
20. Du, W., and Li, B. (2008). "Multi-strategy ensemble particle swarm optimization for dynamic optimization", *International Journal of Information Sciences*, Vol. 178, no. 15, pp. 3096-3109. DOI: 10.1016/j.ins.2008.01.020.
21. Tasgetiren, M.F., Sevkli, M., Liang, Y.-C. and Gencyilmaz, G. (2004), "Particle swarm optimization algorithm for permutation flow shop sequencing problem", *Proceedings of the 4th International Workshop on Ant Colony, Optimization, and Swarm Intelligence* (ANTS2004), Brussels, Belgium, pp. 382-390.
22. Tasgetiren, M.F., Sevkli, M., Liang, Y.-C. and Gencyilmaz, G. (2006), "Particle swarm optimization algorithm for single-machine total weighted tardiness problem", *Proceedings of the Congress on Evolutionary Computation*, Vol. 2, pp. 1412-1419.
23. Van der Maaten, L.J.P. and Hinton, G.E. (2008), "Visualizing High-Dimensional Data Using *t*-SNE", *Journal of Machine Learning Research,* Vol. 9, pp. 2579-2605.

Received 04.01.19

Я.Ю. Дорогий, Е.А. Дорога-Иванюк, Д.А. Ференс

МОДЕЛЬ РАСПРЕДЕЛЕНИЯ РЕСУРСОВ
КРИТИЧЕСКОЙ ИТ-ИНФРАСТРУКТУРЫ С ЧЕТКИМИ
ПАРАМЕТРАМИ НА ОСНОВЕ МЕТОДА РОЯ ЧАСТИЦ

Проведен детальный анализ методов и алгоритмов размещения ресурсов виртуализированных ИТ-инфраструктур. Приведено описание математической модели распределения ресурсов критической ИТ-инфраструктуры с четкими параметрами и ее использование в сочетании с методом роя частиц. Раскрыта суть метода роя частиц, рассмотрен принцип поиска наилучшего решения и его основные операции для решения поставленной задачи. Приведены результаты экспериментальных исследований предложенной модели распределения ресурсов критической ИТ-инфраструктуры с четкими параметрами на основе метода роя частиц.

К л ю ч е в ы е с л о в а: архитектура, распределение ресурсов, метод роя частиц, критическая ИТ-инфраструктура.

Y.Y. Dorogyy, O.O. Doroha-Ivaniuk, D.A. Ferens

RESOURCES DISTRIBUTION MODEL OF CRITICAL
IT INFRASTRUCTURE WITH CLEAR PARAMETERS BASED
ON THE PARTICLE SWARM ALGORITHM

A detailed analysis of the methods and algorithms for allocating resources for virtualized IT infrastructures has been carried out. A detailed description of the mathematical model of resource allocation of a critical IT infrastructure with clear parameters and its use in combination with the particle swarm method is given. The method of particle swarm, the principle of finding the best

solution and its basic operations for solving a given problem are disclosed. The last part of the article presents experimental researches of the proposed model of distribution of critical IT infrastructure resources with clear parameters based on the particle swarm method.

K e y w o r d s: architecture, resource allocation, particle swarm method, critical IT infrastructure.

ДОРОГИЙ Ярослав Юрійович, канд. техн. наук, доцент, доцент кафедри автоматики і управління в технічних системах Національного технічного університету України «Київський політехнічний ін-т ім. Ігоря Сікорського», котрий закінчив в 2002 р. Область наукових досліджень — штучний інтелект, критичні IT-інфраструктури, теорія розпізнавання.

ДОРОГА-ІВАНЮК Олена Олександрівна, аспірантка, асистент кафедри автоматики і управління в технічних системах Національного технічного університету України «Київський політехнічний ін-т ім. Ігоря Сікорського», котрий закінчила в 2007 р. Область наукових досліджень — штучний інтелект, критичні IT-інфраструктури.

ФЕРЕНС Дмитро Андрійович, магістрант кафедри автоматики і управління в технічних системах Національного технічного університету України «Київський політехнічний ін-т ім. Ігоря Сікорського». Область наукових досліджень — штучний інтелект.