УДК 004.94; 004.4; 004.62

МЕТОД КОНСОЛИДАЦИИ ВИРТУАЛЬНЫХ МАШИН НА ОСНОВЕ ЛУЧЕВОГО ПОИСКА

Э.В. Жариков, докторант, доцент, Е.А. Сердюк, студент,
НТУ Украины "КПИ им. Игоря Сикорского"

Аннотация.. *Для решения задачи консолидации виртуальных машин в статье предлагается двухстадийный метод на основе использования локального лучевого поиска. В статье разработаны эвристики первой и второй стадий оптимизации, разработан алгоритм лучевого поиска, разработаны функция оценки и условия окончания работы алгоритма. Для проверки работы метода использованы данные о поступлении задач в кластер Google.*

Ключевые слова*: облачные вычисления, виртуализация, управление ресурсами, центр обработки данных.*

МЕТОД КОНСОЛІДАЦІЇ ВІРТУАЛЬНИХ МАШИН НА ОСНОВІ ПРОМЕНЕВОГО ПОШУКУ

Е.В. Жаріков, докторант, доцент, Є.О. Сердюк, студент,
НТУ України "КПІ ім. Ігоря Сікорського"

Анотація. *Для вирішення задачі консолідації віртуальних машин у статті пропонується двостадійний метод на базі використання локального променевого пошуку. В статті розроблені евристики першої та другої стадій оптимізації, розроблений алгоритм променевого пошуку, функція оцінювання та умови закінчення роботи алгоритму. Для перевірки роботи методу використані дані про надходження завдань в кластер Google.*

Ключові слова*: хмарні обчислення, віртуалізація, управління ресурсами, центр обробки даних.*

METHOD FOR VIRTUAL MACHINE CONSOLIDATION BASED ON BEAM SEARCH

E.V. Zharikov, doctoral student, associate professor, Y.O. Serdiuk, student,
NTU of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”

Abstract. *In this paper, the two-stage method is proposed to solve virtual machine consolidation problem. The proposed method is based on a local beam search algorithm. The heuristics of the first and the second stages of the optimization algorithm, the algorithm of local beam search, the assessment function, and the stop functions of the algorithm are developed. Google cluster-usage traces are used to evaluate the method.*

Key words: *cloud computing, virtualization, resource management, data center*

Вступ

Надання широкого спектру інформаційних послуг в сучасних умовах базується на таких сервісних моделях хмарних обчислень, як інфраструктура як сервіс (англ. IaaS), платформа як сервіс (англ. PaaS) та застосунки як сервіс (англ. SaaS). З метою більш ефективного використання апаратного забезпечення в центрах обробки даних (ЦОД) застосовуються технології віртуалізації апаратного забезпечення, програмних засобів, мереж, сховищ даних, робочих місць та інші.

Сервісна модель IaaS дозволяє більш ефективно використовувати апаратне забезпечення фізичного сервера (ФС) за рахунок віртуалізації його локальних ресурсів. При цьому,

клієнтові надається частина ресурсів ФС у вигляді віртуальної машини (ВМ). Для реалізації сучасних інформаційних послуг клієнт розгортає одну або декілька ВМ в певній конфігурації, яка визначена провайдером хмарних послуг. Таким чином, виникає комплекс задач, пов'язаних з управлінням віртуальними машинами, фізичними серверами, мережевою взаємодією, сховищами, застосуваннями та іншими системами.

Для роботи кожного екземпляру застосування створюється окрема ВМ або контейнер в середині ВМ. В статті розглядається робота застосувань на базі ВМ. Виходячи з певних бізнес-потреб клієнт може динамічно змінювати конфігурацію ВМ або налаштовує механізми балансування навантаження, відмовостійкості та резервного копіювання. В процесі роботи хмарного сервісу клієнти створюють множину ВМ, яка постійно змінюється. Кількість ВМ, розміщених на окремому ФС може постійно змінюватися, тому деякі ФС виявляються незавантаженими до максимально можливого порогу і витрачають зайву енергію.

З метою більш ефективного використання апаратних ресурсів ФС засоби віртуалізації надають можливість міграції ВМ з одного ФС на інший. При цьому, вплив на роботу ВМ є мінімальним, і для клієнта міграція ВМ виявляється непомітною. Але навантаження на фізичні сервери, які обмінюються цією ВМ, зростає.

Таким чином, однією з головних задач при управлінні ресурсами хмарного ЦОД є розміщення віртуальних машин таким чином, щоб задіяти меншу кількість фізичних серверів та зменшити кількість міграцій віртуальних машин. Процес перерозподілу віртуальних машин серед фізичних серверів називають також консолідацією ВМ. Вирішення проблеми консолідації віртуальних машин представлено в багатьох публікаціях [1]. Але дослідження виконувалися для наявних на той час технологій віртуалізації та хмарних технологій. В сучасних ЦОД впроваджуються все нові і нові апаратні та системні засоби, які співіснують з технологіями попередніх поколінь. Таким чином, актуальною є розробка нових алгоритмів і методів для управління обчислювальними ресурсами ЦОД в цілому, та розміщенням віртуальних машин зокрема.

Для вирішення задачі консолідації віртуальних машин у статті пропонується двостадійний метод на базі використання локального променевого пошуку (ПП). Розроблені евристики першої та другої стадій оптимізації, розроблений алгоритм променевого пошуку, функція оцінювання та умови закінчення роботи алгоритму. Для перевірки роботи методу використані дані про надходження завдань в кластер Google.

Аналіз публікацій

Останнім часом запропоновано багато підходів та алгоритмів для вирішення проблеми консолідації віртуальних машин [1, 2, 3]. Проблема консолідації віртуальних машин розглядається як оптимізаційна проблема з різними цільовими функціями. Особливість цієї проблеми полягає в наявності великої кількості станів середовища та обмежень. Крім того, складність виявляється при формулюванні цільової функції, до якої входять декілька показ-ників, які треба оптимізувати. Для промислових ЦОД з хмарними інфраструктурами використовуються прості алгоритми управління, такі як first-fit, best-fit та їх модифікації (Eucalyptus [4], Microsoft [5], Google [6]). Це обумовлено вимогами робастності застосувань клієнтів та участю адміністраторів в автоматизованому процесі управління ресурсами ЦОД при постійному моніторингу віртуальних машин.

В дослідженнях використовуються такі цільові функції, як мінімізація споживання електроенергії, мінімізація порушень угод про якість сервісу (SLA), мінімізація мережевого трафіку, максимізація продуктивності та використання ресурсів. Однією з основних умов для сучасних кластерів ФС є можливість роботи алгоритму в режимі онлайн [3]. Разом з цим допускається застосування методів оптимізації, які спрацьовують при появі певних умов робо-ти кластеру. Для такого випадку нові завдання розміщуються на ФС, які не задіяні в процесі консолідації.

Крім традиційних евристик та детермінованих алгоритмів для вирішення проблеми консолідації ВМ використовуються і алгоритми локального пошуку, такі як еволюційні алгоритми, алгоритми оптимізації мурашиних коло-ній, табу пошук, пошук з емуляцією відпалу. Більш ефективним на нашу дум-

ку є використання локального пошуку на другій стадії оптимізації, після підготовки відповідного набору станів детермінованими алгоритмами з метою покращити рішення, знайдене на першій стадії.

Таким чином, в цій статті пропонується застосувати локальний променевий пошук на підготовленому наборі даних з метою зменшити кількість міграцій ВМ та збільшити кіль-кість вивільнених ФС, отриманих на попередній стадії оптимізації.

Модель системи

На теперішній час більшість послуг клієнти отримують на базі хмарних центрів обробки даних. Хмарні ЦОД являють собою складні системи, що складаються з серверних підсистем, підсистем зберігання даних, мережевих підсистем та підсистем інженерного забезпечення.

В статті розглянуто задачу управління окремою серверною підсистемою ЦОД у вигляді кластера в умовах віртуалізації. Для побудови кластеру в сучасних ЦОД використовують два підходи компоновки: з використанням гетерогенної конфігурації або гомогенної конфігурації. Обидва підходи мають свої недоліки та переваги, однак уникнути розміщення гетерогенних конфігурацій в масштабі ЦОД, коли конфігурації кожного кластеру відрізняються, в більшості випадків не вдається з причин еволюції елементної бази серверів та нових вимог користувачів до IT-інфраструктури.

Розробка і дослідження роботи алгоритму променевого пошуку виконані для кластеру, який складається з фізичних серверів різної конфігурації. Кожен ФС надає для декількох ВМ декілька ресурсів: процесор (CPU), об'єм пам'яті (RAM), доступ до підсистеми зберігання даних (IOPS), доступ до мережевої підсистеми (NET) та інші. В залежності від наявної ємності ресурсів ФС, вимог до ресурсів з боку ВМ, часу виконання завдань всередині ВМ та інтенсивності надходження завдань кількість ВМ, що виконуються на ФС, постійно змінюється. Зміна кількості ВМ відбувається в таких станах: створення нової ВМ, видалення ВМ, міграція ВМ.

В цій статті розглядаються ресурси CPU та RAM, що надаються для ВМ. Однак алгоритм може бути доповнений іншими ресурсами, які потребує ВМ. Вибір саме двох ресурсів обумовлено використанням вхідних даних з набору Google cluster-usage traces (GCT) [7]. Для дослідження роботи алгоритму ПП з набору GCT використано таблиці "Machine events" та "Task events table". Випадковим чином з першої таблиці обрано 6000 ФС, з другої таблиці обрано 70000 завдань. З таблиці "Machine events" для кожного ФС використано такі атрибути: machine ID, capacity: CPU, capacity: memory. З таблиці "Task events table" для кожного ФС використано такі атрибути: task index within the job, machine ID, resource request for CPU cores, resource request for RAM. Дані в таблицях нормовані відносно ФС з найбільшим значенням ємності відповідного ресурсу.

Постановка задачі

Кластер управління складається з множини M фізичних серверів та N віртуальних машин, $N, M \in \mathbb{N}$. В процесі дослідження роботи алгоритму ПП кількість ФС та ВМ не змінюється. Кожне завдання з таблиці "Task events table" виконується в окремій ВМ. В загальному випадку, між запусками алгоритму ПП кількість ФС та ВМ може змінюватись.

Задана ємність j-ї ВМ для ресурсу k, $c_j^k \in (0,1]$, $k \in \{CPU, RAM\}$ визначається вимогами завдання і нормовано відносно ФС з найбільшою ємністю ресурсу k. Ємність фізичного сервера і для ресурсу k, $C_i^k \in (0,1]$ визначається типом ФС і нормовано відносно ФС з найбільшою ємністю ресурсу k.

Множина M складається з множини A фізичних серверів, які визначені для вимикання, та множини B фізичних серверів, які надають ресурси для ВМ, що будуть мігрувати з ФС, які належать до множини A, $A \cup B = M$, $A \cap B = \varnothing$.

Міграцію віртуальної машини j на фізичний сервер і позначимо як $U_{ij} \in \{0,1\}$. Міграція відбувається якщо $U_{ij} = 1$. Кожна ВМ має свій ID, який в процесі роботи алгоритму пов'язаний з номером ***j***. Кожний ФС теж має свій ID, який в процесі роботи алгоритму пов'язаний з номером ***i***.

Метод консолідації ВМ на основі променевого пошуку

Опис основного алгоритму. Вхідними даними алгоритму ПП є список ФС з множини *A* та список ФС з множини *B*, в яких є вільні ресурси і є можливість розмістити додаткові завдання у вигляді ВМ.

Ідея алгоритму: перегляд *i*-го ФС зі списку *A* та пошук таких або такого ФС зі списку *B*, куди можливо мігрувати *j*-ту ВМ з *i*-го ФС. Якщо вдалося звільнити всі або частку ФС зі списку *A*, видаємо результат у вигляді матриці U_{ij}, яка є планом міграцій.

Опис алгоритму променевого пошуку:
Перша стадія: підготовка вхідних даних для другої стадії. Формування списку *A* фізичних серверів, які треба звільнити від ВМ, та списку *B* фізичних серверів для визначення плану міграцій.
Друга стадія виконується для кожного ФС зі списку *A*:
1. На кожному кроці обираємо одну ВМ, назначену на *i*-й ФС і розглядаємо наступні варіанти обміну (міграцій):
a) мігрувати ВМ на інший ФС, в якого залишилось достатньо вільних ресурсів CPU та RAM;
b) перевірити можливість обміну з іншим ФС віртуальною машиною, яка вимагає менше ресурсів (так ми розглядаємо лише стани з кращою оцінкою та уникаємо можливих зациклювань) і, в результаті, ФС не буде перенавантаженим після обміну.
2. З усіх можливих обмінів обирається *n* (ширина променю) обмінів з найвищою оцінкою.
3. Завершуємо пошук якщо *i*-й ФС вдалося звільнити від віртуальних машин або якщо не можна побудувати нові стани (тобто не залишилось жодного варіанту для реалізації допустимого обміну a) або b)).

Порівняння станів виконується за допомогою критерію (1):

$$J = \sum_{i=1}^{m} u_i^2 + \sum_{i=1}^{n} f_i^2 , \qquad (1)$$

де u_i – кількість використовуваних ресурсів на *i*-му ФС, f_i – кількість вільних ресурсів на *i*-му ФС, *m* – кількість ФС в списку *B*, *n* – кількість ФС в списку *A*.

Таким чином, алгоритм поступово зменшує використовувані ресурси на ФС, які належать до списку *A*, та завантажує ФС зі списку *B*.

Умови завершення алгоритму. Для завершення роботи алгоритму пропонуються наступні умови:
1. Вичерпання списку *A*.
2. Неможливість мігрувати всі ВМ з певної визначеної кількості ФС Th_A поспіль.

Якщо другу умову не застосовувати, цикли пошуку виконуються занадто довго, якщо співставити їх з часом на створення нової ВМ та часом на міграцію для ВМ з середніми вимогами до ресурсів пам'яті. Для визначення Th_A пропонується застосувати евристику, яка полягає у розгляді певного відсотку фізичних серверів зі списку *A*, але не менше, ніж α фізичних серверів. В реалізації досліджуваного алгоритму прийнято $Th_A = 0.05$, $\alpha = 10$. З меншими значеннями α алгоритм пропускає значну кількість ФС, що могли бути вимкнуті, але в результаті завершення алгоритму були не звільнені від працюючих на них ВМ. Даний критерій завершення вводиться для зменшення часу виконання алгоритму.

Визначення вхідних даних (опис першої стадії роботи методу). Отримання списку фізичних серверів, з яких треба мігрувати всі ВМ з метою подальшого вимкнення ФС пропонується здійснювати за допомогою двох методик: *нижньої границі* та *порогу вільних ресурсів*. Розглянемо кожну з методик більш детально.

Методика нижньої границі полягає у визначенні такої кількості фізичних серверів, які вимкнути в результаті міграції усіх ВМ, що на них працює, не уявляється можливим. Таке уявлення виникає з гіпотез, що використовуються в роботі алгоритму визначення нижньої границі.

Пошук нижньої границі виконується наступним чином:
1. Знаходимо середній об'єм ресурсів по всім ФС, на яких працюють ВМ, $\sum_{i=1}^{M} C_i^k$. Значення для кожного ресурсу *k* розраховуються окремо.
2. По кожному ресурсу окремо рахуємо суму

необхідних ресурсів для виконання всіх наявних ВМ, $\sum_{j=1}^{N} c_j^k$.

3. Окремо по кожному ресурсу рахуємо відношення необхідних ресурсів до середнього об'єму ресурсів та округляємо значення до більшого цілого.
4. Нижньою границею буде найбільше число з отриманих на кроці 3.
5. Сортуємо фізичні сервери одним з наступних способів:
а) за об'ємом ресурсів ФС, потім за відношенням використаних ресурсів до кількості працюючих ВМ;
б) за об'ємом ресурсів ФС, потім за кількістю назначених завдань, потім відношенням використаних ресурсів до кількості працюючих ВМ.

В результаті досліджень роботи алгоритму з'ясувалося, що варіант b) виявився значно ефективнішим. При його використанні на одному й тому самому наборі даних вдалося вимкнути 82 ФС, тоді як при використанні варіанту а) вдалося вимкнути лише 33 ФС.

В результаті, знаходимо різницю між кількістю ФС, що використовуються, та отриманою нижньою границею. Знайдене число – це кількість ФС списку *A* на вимкнення, які є першими у відсортованому списку. Решта ФС потрапляє у список *B*.

Ідея методики порогу вільних ресурсів полягає в такому формуванні списку *A*, при якому до цього списку потрапляють ФС, загальна кількість невикористаних ресурсів яких перевищує об'єм ресурсів одного з ФС кластеру.

Побудова списку *A* з використанням порогу вільних ресурсів β виконується наступним чином:
1. Встановлюємо значення β .
2. Відбираємо ФС, у яких кожного вільного ресурсу більше за значення β .
3. По кожному ресурсу окремо рахуємо суму вільних ресурсів.
4. Сортуємо обрані ФС аналогічно до варіанту b) з методики нижньої границі.
5. Проходимо по відсортованому списку. Поки сума ресурсів більша за об'єм поточного ФС віднімаємо від суми цей об'єм, додаємо поточний ФС в список *A* та переходимо до наступного ФС, інакше завершуємо проходження списку. Таким чином, отримуємо список *A* з ФС, які треба вимкнути, та список *B* з ФС для обміну віртуальними машинами.

5. Оцінка результатів моделювання

Дослідження роботи алгоритму виконане на фрагментах набору вхідних даних GCT [7]. Дані для тестування і дослідження алгоритму обрані з GCT для певного діапазону часу, за який збиралися дані на реальному кластері. Це необхідно, щоб врахувати всі дані за один і той самий період часу в журналі GCT. З набору випадковим чином відібрано 70000 завдань з певними вимогами до ресурсів k. Кожному завданню відповідатиме окрема ВМ. Обираємо 6000 фізичних серверів, з яких на 5832 серверах розміщено ВМ, та на 168 серверах завдань немає, але вони доступні для розміщення ВМ. Ширина променю для алгоритму дорівнює 5. В даній роботі дослідження впливу ширини променю на роботу алгоритму не досліджувалося.

В таблиці 1 представлені результати роботи алгоритму променевого пошуку з консолідації ВМ для різних значень β . В таблиці 2 представлені результати роботи алгоритму променевого пошуку при консолідації ВМ з використанням методики нижньої границі. Якісними показниками роботи алгоритму є кількість вимкнених ФС та кількість міграцій ВМ, за допомогою яких ФС вивільнюються від ВМ.

Таблиця 1 – Результати моделювання консолідації ВМ алгоритмом променевого пошуку з використанням методики порогу вільних ресурсів

β	*B*	*A*	Вимкнено ФС	Міграцій	Час, с
0.005	199	3	0	0	0.53
0.004	299	3	0	0	0.69
0.003	1098	9	0	0	4.05
0.002	1393	10	0	0	6.55
0.001	1868	28	13	219	38.08
0.0009	1943	31	15	263	42.29
0.0008	2002	39	22	349	67.93
0.0007	2081	43	24	399	81.02
0.0006	2204	48	28	432	94.14
0.0005	2302	55	32	504	105.48
0.0004	2407	63	36	554	141.43
0.0003	2565	73	43	683	143.16
0.0002	2764	82	48	782	166.54
0.0001	3706	95	59	992	238.06
0.00005	4323	106	68	1199	330.18
0.00001	5095	116	75	1336	463.36
0	5328	125	84	1459	518.34

Таблиця 2 – Результати моделювання консолідації ВМ алгоритмом променевого пошуку з використанням методики нижньої границі

B	A	Вимкнено ФС	Міграцій	Час, с
5707	125	82	1460	508.09

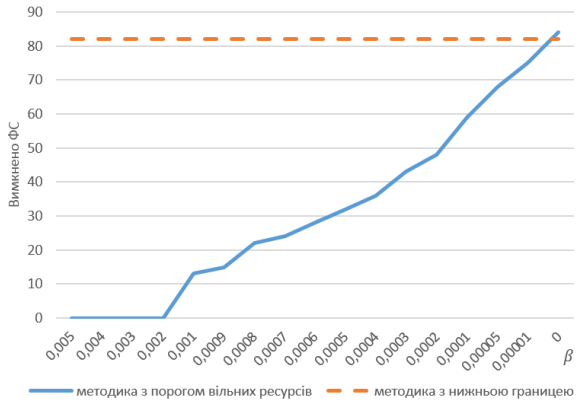

Рис. 1 – Кількість вимкнених ФС

Вплив значення порогу β на якісні показники роботи алгоритму є суттєвим (Рис. 1 та Рис. 2), залежність виявилася майже лінійною. Чим менший поріг β тим більше ФС потрапляють на вхід алгоритму.

При $\beta = 0$ деякі ФС (а саме ті, у яких хоча б один ресурс повністю зайнятий) не потрапляють на вхід алгоритму, тому що при перевірці наявності ресурсів використовується строга нерівність. При $\beta < 0.0002$ алгоритм з методикою порогу вільних ресурсів починає працювати більш ефективно і на граничних значеннях працює краще чим з методикою нижньої границі. Для всебічної перевірки роботи алгоритму з різними методиками та їх порівняння планується створити декілька наборів даних з різною кількістю ФС та ВМ.

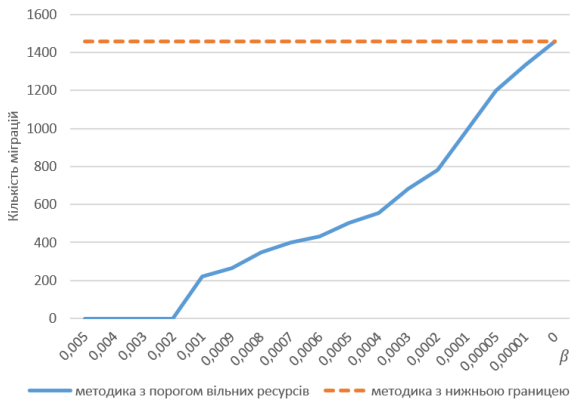

Рис. 2 – Кількість міграцій

Перевагою використання методики з порогом вільних ресурсів є можливість адаптуватися до стану кластера, враховуючи інші ресурси, час міграції ВМ та інтенсивність надходження заявок на створення нових ВМ.

Висновки

Для управління процесом консолідації віртуальних машин в статті запропоновано використовувати двостадійний метод на базі алгоритму променевого пошуку. Розроблені евристики першої та другої стадій роботи алгоритму, розроблений алгоритм променевого пошуку для консолідації ВМ, розроблені функція оцінювання та умови закінчення роботи алгоритму. Для перевірки роботи методу використані дані про надходження завдань в кластер Google. методики нижньої границі та порогу вільних ресурсів, виконано їх порівняльний аналіз. В результаті пропонується використовувати методику з порогом вільних ресурсів, що показала більш якісні результати консолідації ВМ.

Новизна роботи полягає у розробці двостадійного методу консолідації віртуальних машин хмарного центру обробки даних на основі алгоритму локального променевого пошуку. Запропонований метод дозволяє вимкнути майже 50 відсотків фізичних серверів, потенційно визначених для вимикання, за рахунок використання допустимої кількості міграцій віртуальних машин.

Література

1. F. Lopez Pires and B. Baran, “A virtual machine placement taxonomy,” in Proc. of the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2015, pp. 159–168.
2. R. W. Ahmad, A. Gani, S. H. A. Hamid, M. Shiraz, A. Yousafzai, and F. Xia, “A survey on virtual machine migration and server consolidation frameworks for cloud data centers,” Journal of Network and Computer Applications, vol. 52, pp. 11–25, 2015.
3. S. Telenyk, E. Zharikov, O. Rolik, “An approach to virtual machine placement in cloud data centers,” In proc. of the 2016 International Conference Radio Electronics & Info Communications (UkrMiCo) 11–16 September, Kyiv, Ukraine. – 2016 pp. 1–6.
4. Eucalyptus community [Online] Available from: http://open.eucalyptus.com/ [Ac-

cessed Aug 10, 2017]
5. S. Lee, R. Panigrahy, V. Prabhakaran, V. Ramasubrahmanian, K. Talwar, L. Uyeda, and U. Wieder, “Validating heuristics for virtual machines consolidation,” Microsoft Research, MSR-TR-2011- 9, 2011.
6. B. Sharma, V. Chudnovsky, J. L. Hellerstein, R. Rifaat, and C. R. Das, “Modeling and synthesizing task placement constraints in google compute clusters,” In Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC), 2011.
7. C. Reiss, J. Wilkes, and J. L. Hellerstein, “Google cluster-usage traces: format+ schema,” Google Inc., Mountain View, CA, USA, Technical Report, 2011.

Рецензент: С.Ф. Теленик, професор, д.т.н., Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського".

Стаття поступила в редакцію 10 серпня 2017 р.