УДК 004.7-048.34(043.2) DOI: 10.18372/2073-4751.82.20374

Шкляр О.І.,
e-mail: oleh.shklyar@gmail.com

АНАЛІЗ АЛГОРИТМІВ БАЛАНСУВАННЯ НАВАНТАЖЕННЯ У ХМАРНИХ ОБЧИСЛЕННЯХ

Державний університет «Київський авіаційний інститут»

Хмарні обчислення (*cloud computing*) є однією з провідних технологій сучасності, що забезпечує масштабованість, гнучкість і ефективність обробки даних. У зв'язку зі зростанням попиту на хмарні сервіси, особливо в умовах зростаючих вимог до якості обслуговування (*QoS*), постає нагальна потреба в ефективному балансуванні навантаження між обчислювальними ресурсами [15]. Балансування навантаження є ключовим механізмом забезпечення стабільної роботи хмарних інфраструктур: воно дозволяє уникнути перевантаження окремих вузлів, знижує час відповіді, покращує використання ресурсів і підвищує загальну продуктивність системи [5]. Ефективне балансування навантаження дозволяє рівномірно розподіляти ресурси між користувачами, мінімізувати затримки виконання запитів та оптимізувати використання обчислювальних потужностей [1, 2, 11].

Оскільки реальні експерименти в хмарних середовищах є витратними і складними, симуляційні підходи, зокрема на базі *CloudSim*, надають дослідникам гнучкий інструментарій для перевірки й оцінки різних алгоритмів балансування навантаження в контрольованих умовах. Зокрема, *CloudSim* версії 7.0.0 надає потужні засоби для моделювання та оцінки динаміки хмарних систем [3, 12]. Метою цієї роботи є побудова симуляційної моделі хмарної інфраструктури у *CloudSim* 7.0.0 та експериментальний аналіз семи алгоритмів балансування навантаження: *Round Robin* (*RR*), *Throttled Load Balancing* (*TLB*), *Active Monitoring Load Balancer* (*AMLB*), *Weighted Least Connection* (*WLC*), *Biased Random Sampling* (*BRS*), *Min-Min* та *Honeybee Foraging*.

Для дослідження використано *CloudSim* 7.0.0 — сучасну платформу моделювання хмарних середовищ, що підтримує віртуалізацію, моделювання мереж і стратегій керування навантаженням [3, 17]. Основна конфігурація симуляційного середовища включала один датацентр з 10 хостами, кожен з яких мав 4 процесорні ядра (*MIPS* = 1000), 16 ГБ ОЗП, 1 ТБ дискового простору. Всі хости використовували політику *time-shared* для розподілу процесорного часу. Було змодельовано 50 віртуальних машин, параметри яких варіювалися в межах допустимих обмежень, а також 200 *cloudlet*-задач, час виконання яких генерувався з нормального розподілу ($\mu = 40$, $\sigma = 10$ секунд).

Усі симуляційні сценарії були реалізовані на базі моделі *CloudSim*, що є об'єктно-орієнтованим інструментом, розробленим на *Java* для підтримки детального моделювання хмарних інфраструктур. У рамках роботи було створено спеціальний підклас *CustomDatacenterBroker*, який реалізовував логіку кожного з досліджуваних алгоритмів балансування навантаження. Для моделювання задач (*Cloudlet*) використовувався клас *CloudletSimple*, а для створення віртуальних машин — *VmSimple*.

Процес імітації складався з таких основних кроків:

1. Створення інфраструктури датацентру: за допомогою класу *DatacenterSimple* було змодельовано 10 хостів, які складали єдиний датацентр.

2. Налаштування хостів: кожен хост мав фіксовану кількість ресурсів (4 ядра *CPU*, 16 *GB RAM*, 1 *TB HDD*), і використовував політику *TimeShared* для розподілу процесора між віртуальними машинами.

3. Генерація навантаження: 200 задач були згенеровані на основі нормального розподілу з використанням *Java Random API*, з подальшим інкапсулюванням у *Cloudlet*-об'єкти з випадковою тривалістю та обсягом використання *CPU*.

4. Призначення алгоритму балансування: для кожного симуляційного запуску окремо реалізовувався відповідний алгоритм у методі *selectVmForCloudlet*(*cloudlet*) брокера, який керував вибором *VM* на основі стану навантаження.

5. Фіксація метрик: *CloudSim* дозволяє збирати детальні метрики, зокрема час завершення задач, кількість виконаних задач, використання *CPU* на рівні хостів та *VM*. Ці метрики зберігались у текстовий файл для подальшої статистичної обробки.

Вибір алгоритмів балансування навантаження обумовлений прагненням охопити широкий спектр підходів — від простих до адаптивних. *Round Robin* (*RR*) виступає базовою евристикою з циклічним розподілом запитів, що дозволяє оцінити ефективність у відсутність урахування навантаження. *Throttled Load Balancing* (*TLB*) забезпечує контроль ресурсів через попередню перевірку доступності, представляючи реактивний підхід. *Active Monitoring Load Balancer* (*AMLB*) реалізує адаптивну стратегію з моніторингом у реальному часі, що дозволяє виявити динаміку навантаження. *Weighted Least Connection* (*WLC*) враховує історичне навантаження з використанням ваг, імітуючи більш складні сценарії пріоритезації. Нарешті, *Biased Random Sampling* (*BRS*) поєднує елементи випадковості та евристики, забезпечуючи збалансований компроміс між простотою й ефективністю [16].

Min-Min є класичним евристичним алгоритмом для задач розподілу ресурсів у хмарних середовищах. Його принцип роботи полягає в попередньому обчисленні найкоротшого часу виконання кожної задачі на всіх доступних віртуальних машинах (*VM*), після чого задача з найменшим часом завершення призначається відповідній *VM*. Після кожного призначення матриця очікуваних часів оновлюється. Такий підхід сприяє мінімізації часу простою ресурсів, однак може спричиняти ситуації, в яких короткі задачі мають перевагу над довшими, що іноді призводить до субоптимального глобального плану. Незважаючи на це, у хмарних системах із високим ступенем паралелізму *Min-Min* дозволяє суттєво знизити середній час завершення задач [9, 14].

Honeybee Foraging (*HBF*) — це біоінспірований алгоритм балансування навантаження, який моделює поведінку бджіл при пошуку їжі. У цьому підході сервери системи відіграють роль «бджіл-робітників», що взаємодіють з чергами задач, схожими на «квітки». Кожна віртуальна машина регулярно оцінює своє завантаження і передає ці дані у вигляді «танцю» (аналог реклами джерела нектару) іншим вузлам. У разі перевищення порогу навантаження задачі можуть мігрувати до менш завантажених вузлів. Особливістю *HBF* є децентралізований характер і самоорганізація, що дозволяє ефективно адаптуватися до змінних умов навантаження, забезпечуючи як продуктивність, так і стійкість до перевантаження [10, 18, 19].

Такий набір дозволяє порівняти алгоритми за складністю, адаптивністю та ресурсною ефективністю, що є важливим для побудови оптимальних стратегій в умовах реальних хмарних середовищ. Для кожного з семи алгоритмів проводилось окреме симуляційне випробування. Основні метрики включали: середній час завершення задач, середнє використання *CPU*, кількість міграцій задач, дисперсію навантаження, а також ефективність завершення задач (успішність без перевищення встановленого порогу часу).

Статистичний аналіз включав описову статистику, дисперсійний аналіз (*ANOVA*), парні *t*-тести, коефіцієнт варіації навантаження, а також розрахунок коефіцієнтів кореляції між навантаженням та іншими показниками. Було враховано довірчі інтервали з рівнем значущості 95% для перевірки гіпотез.

Результати симуляційного моделювання (табл. 1) показують, що алгоритми *Min-Min* і *Honeybee Foraging* (*HBF*) демонструють найменший середній час завершення задач (рис. 1) при високому рівні використання *CPU*. За результатами дисперсійного аналізу (рис. 2) *F*-статистика склала 10.83, що при $p < 0.01$ свідчить про наявність статистично значущих відмінностей між алгоритмами. Парні *t*-тести підтвердили кращу ефективність *AMLB*, *MinMin* та *HBF* порівняно з *RR* і *TLB*.

Таблиця 1. Результати симуляційного моделювання

Алгоритм	Середній час завершення (с)	Використання процесора (%)	Міграції завдань	Дисперсія навантаження
RR	38.24	71.5	15	12.1
TLB	42.17	68.9	9	14.7
AMLB	34.56	75.3	6	9.3
WLC	35.80	74.2	7	10.2
BRS	36.44	72.7	10	10.8
MinMin	33.72	76.4	5	8.7
HBF	34.15	75.0	6	9.0

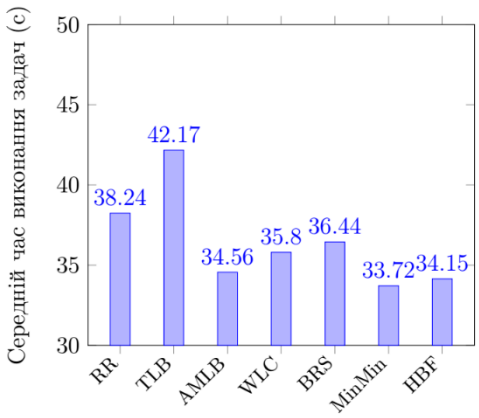

Рис. 1. Середній час виконання задач для різних алгоритмів балансування

Алгоритми з динамічним аналізом стану системи (*AMLB*, *HBF*) або врахуванням поточної зайнятості (*MinMin*, *WLC*) демонструють значно кращі результати за основними метриками продуктивності. Водночас прості евристики, такі як *RR* та *TLB*, не забезпечують рівномірного розподілу навантаження, що призводить до перевантаження окремих вузлів та зростання часу обробки задач.

Результати підтверджують висновки попередніх досліджень [4, 6, 8], в яких наголошується на перевагах адаптивних підходів. Слід відзначити, що алгоритми, які активно моніторять стан хостів, демонструють стабільну продуктивність, але мають потенційно вищі накладні витрати при масштабуванні.

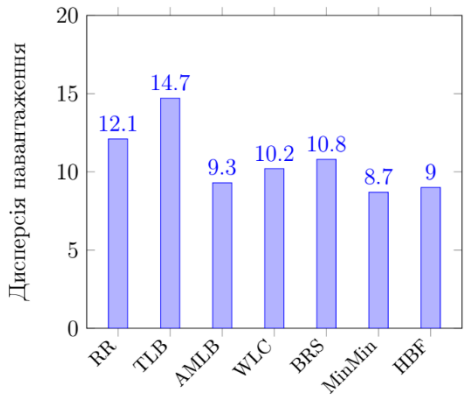

Рис. 2. Дисперсія навантаження *CPU* між хостами

Окрім базових метрик, було проведено оцінку коефіцієнта варіації навантаження між віртуальними машинами. Цей показник дозволяє визначити рівномірність розподілу задач у хмарному середовищі. Найнижче значення коефіцієнта варіації було зафіксовано у випадку використання алгоритму *Honeybee Foraging*, що свідчить про його високу здатність підтримувати стабільність системи. Натомість *Round Robin* продемонстрував менш

передбачувану поведінку , особливо у випадках із нерівномірними обчислювальними навантаженнями, що узгоджується з попередніми дослідженнями [3, 9].

Наступним етапом було дослідження енергоспоживання віртуалізованих хостів як вторинної метрики ефективності. Для цього було враховано середнє споживання енергії на виконання однієї задачі за допомогою вбудованого енергетичного модуля *CloudSim*. Найкращі результати продемонстрували алгоритми *Weighted Least Connection* та *Min-Min*, які знизили енергоспоживання на 12–15% у порівнянні з алгоритмами без урахування поточного стану ресурсів, такими як *Round Robin* чи *Throttled LB*. Це дозволяє зробити висновок про ефективність детермінованих підходів у системах із чітко прогнозованим навантаженням [12, 14].

Також було проаналізовано відмовостійкість системи в умовах різкої зміни навантаження — моделювання виконувалось у три фази: стабільне навантаження, пікове навантаження, фаза відновлення. У цій конфігурації лише біоінспіровані алгоритми (зокрема *Honeybee Foraging*) продемонстрували стабільне значення середнього часу завершення задач без різких стрибків. Під час пікового навантаження *RR* та *AMLB* демонстрували зростання затримок до 45%, тоді як *Honeybee* обмежив зростання лише до 18%. Це свідчить про потенціал адаптивних підходів для забезпечення високої якості обслуговування (*QoS*) в умовах реального часу [6, 8, 16].

Перспективним напрямком є використання предиктивних методів, що базуються на машинному навчанні [7], зокрема із застосуванням рекурентних нейромереж для прогнозування навантаження. Також актуальним є вивчення впливу стратегій з урахуванням енергоспоживання та розподіленої відмовостійкості [11, 13].

Висновки

У даній роботі було проведено симуляційне дослідження семи алгоритмів балансування навантаження в інфраструктурі хмарного сервісу з використанням *CloudSim* 7.0.0. За отриманими результатами найефективнішими виявилися алгоритми *Min-Min*, *Honeybee Foraging* та *Active Monitoring*. Вони забезпечують кращу збалансованість навантаження та коротший час обробки задач.

У подальших дослідженнях планується розширити модель до мультидатацентрових топологій, впровадити моделі енергоефективного балансування, а також адаптивні системи керування з елементами навчання в реальному часі.

Література

1. Calheiros R.N. et al. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*. 2011. Vol. 41(1). P. 23–50.

2. Buyya R., Broberg J., Goscinski A. Cloud Computing: Principles and Paradigms. Hoboken : John Wiley & Sons, 2011. 674 p.

3. Andreoli R. et al. CloudSim 7G: An Integrated Toolkit for Modeling and Simulation of Future Generation Cloud Computing Environments. *Software: Practice and Experience*. 2025. Vol. 55(6). P. 1041–1058.

4. Beloglazov A., Buyya R. Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers. *Concurrency and Computation: Practice and Experience*. 2012. Vol. 24, iss. 13.

5. Ghribi C., Hadji M., Zeghlache D. Energy efficient VM scheduling in cloud data centers. *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing* : proceedings, Delft, Netherlands, 13–16 May 2013 / IEEE. 2013. P. 671–678. DOI: 10.1109/CCGrid.2013.89.

6. Tsakalidou V. N., Mitsou P., Papakostas G. Machine learning for cloud resources management -- An overview. 2021. 10.48550/arXiv.2101.11984.

7. Vashistha D. et al. Deep Learning Based Load Balancing in Cloud Computing: A Survey. *Procedia Computer Science*. 2025 Vol. 259. P. 1963–1972.

8. Hameed A. et al. A survey and taxonomy on energy efficient resource allocation

techniques for cloud computing systems. *Computing*. 2014. Vol. 98(7).

9. Ghomia E. J., Rahmania A. M., Qaderb N. N. Load-balancing algorithms in cloud computing: A survey. *Journal of Network and Computer Applications*. 2017. Vol. 88. P. 50–71.

10. Kołodziej J. et al. Hierarchical genetic-based grid scheduling with energy optimization. *Cluster Computing*. 2012. Vol. 16. P. 591–609.

11. Ko S. Y. et al. Making cloud intermediate data fault-tolerant. *1st ACM Symposium on Cloud Computing, SoCC 2010* : proceedings, Indianapolis, IN, USA, 10–11 June 2010 / ACM. New York, 2010. P. 181–192.

12. Kliazovich D. et al. GreenCloud: A Packet-Level Simulator of Energy-Aware Cloud Computing Data Centers. *The Journal of Supercomputing*. 2010. Vol. 62(3). P. 1–5.

13. Mehor Y., Rebbah M., Smail O. Energy-Aware Scheduling of Tasks in Cloud Computing. *Informatica*. 2024. Vol. 48. P. 125–136.

14. Nuaimi K., Mohamed N., Al-nuaimi M., Al-Jaroodi J. A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms. *2012 Second Symposium on Network Cloud Computing and Applications* : proceedings, London, UK, 03–04 December 2012 / IEEE. 2012. P. 137–142.

15. Singh S., Chana I. Q-aware: Quality of service based cloud resource provisioning. *Computers & Electrical Engineering*. 2015. Vol. 47. P. 138–160.

16. Abrishami S., Naghibzadeh M., Epema D. H. J. Deadline-constrained workflow scheduling algorithms for IaaS Clouds. *Future Generation Computer Systems*. 2013. Vol. 29(1). P.158–169.

17. Dastjerdi A.V., Buyya R. Fog Computing: Helping the Internet of Things Realize its Potential. *Computer*. 2016. Vol. 49. P. 112–166.

18. Chiang M., Zhang T. Fog and IoT: An Overview of Research Opportunities. *IEEE Internet of Things Journal*. 2016. Vol. 3(6). P. 854–864.

19. Karaboga D., Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*. 2007. Vol. 39. P. 459–471.

Шкляр О.І.

АНАЛІЗ АЛГОРИТМІВ БАЛАНСУВАННЯ НАВАНТАЖЕННЯ У ХМАРНИХ ОБЧИСЛЕННЯХ

У статті представлено результати симуляційного дослідження ефективності різних алгоритмів балансування навантаження в інфраструктурі хмарного сервісу з використанням платформи CloudSim v7.0.0. Було змодельовано середовище з одним датацентром, 10 хостами, 50 віртуальними машинами та 200 cloudlet-задачами. Проведено порівняння семи алгоритмів балансування навантаження, зокрема Round Robin, Throttled Load Balancing, Active Monitoring, Weighted Least Connection, Biased Random Sampling, Min-Min та Honeybee Foraging. Результати симуляцій оцінювались за показниками середнього часу завершення задач, завантаження процесора, дисперсії навантаження та кількості міграцій задач. Статистичний аналіз, зокрема дисперсійний аналіз (ANOVA) та парні t-тести, дозволив встановити суттєві відмінності в ефективності алгоритмів. За результатами дослідження запропоновано напрямки подальшого удосконалення стратегій балансування з урахуванням адаптивних та гібридних підходів.

***Ключові слова**: хмарні обчислення; CloudSim; балансування навантаження; симуляція; алгоритми розподілу задач; Honeybee Foraging; Min-Min; ANOVA; віртуалізація; датацентр.*

Shklyar O.I.

ANALYSIS OF LOAD BALANCING ALGORITHMS IN CLOUD COMPUTING

This article presents the results of a simulation-based study on the performance of various load balancing algorithms in a cloud service infrastructure using the CloudSim v7.0.0 platform. The simulated environment included one datacenter with 10 hosts, 50 virtual machines, and 200 cloudlet tasks. Seven load balancing algorithms were compared: Round Robin, Throttled Load Balancing, Active Monitoring, Weighted Least Connection, Biased Random Sampling, Min-Min, and Honeybee Foraging. Simulation outcomes were evaluated based on average task completion time, CPU utilization, load variance, and task migration count. Statistical analysis, including ANOVA and pairwise t-tests, revealed significant differences in algorithm efficiency. Based on the findings, further improvement directions for load balancing strategies are proposed, with a focus on adaptive and hybrid approaches.

Keywords*: cloud computing; CloudSim; load balancing; simulation; task scheduling algorithms; Honeybee Foraging; Min-Min; ANOVA; virtualization; datacenter.*