Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Кваліфікаційна наукова
праця на правах рукопису

ІВАНІСЕНКО ІГОР МИКОЛАЙОВИЧ

УДК 004.7:519.2

ДИСЕРТАЦІЯ

МЕТОДИ БАЛАНСУВАННЯ НАВАНТАЖЕННЯ У РОЗПОДІЛЕНИХ СИСТЕМАХ З УРАХУВАННЯМ САМОПОДІБНИХ ВЛАСТИВОСТЕЙ ВХІДНИХ ПОТОКІВ

05.13.05 – комп’ютерні системи та компоненти
технічні науки

Подається на здобуття наукового ступеня кандидата технічних наук

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело _____ І.М. Іванісенко

Науковий керівник Кіріченко Людмила Олегівна,
доктор технічних наук, професор

Цей примірник дисертації ідентичний
за змістом з іншими, що подані до
спеціалізованої вченої ради Д.64.052.01.

Учений секретар спеціалізованої
вченої ради Д 64.052.01 Є.І. Литвинова

Харків – 2017

АНОТАЦІЯ

Іванісенко І.М. Методи балансування навантаження у розподілених системах з урахуванням самоподібних властивостей вхідних потоків. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.13.05 – комп'ютерні системи та компоненти. – Харківський національний університет радіоелектроніки, Міністерство освіти і науки України, Харків, 2017.

У дисертаційній роботі запропоновано нове вирішення актуальної науково-практичної задачі розробки методів балансування навантаження, які враховують властивості самоподібності трафіка і використання ресурсів кожного вузла та всієї розподіленої системи. Об'єктом дослідження виступають процеси управління трафіком у розподілених інфокомунікаційних системах. Предметом дослідження є моделі та методи балансування самоподібного навантаження в розподілених системах.

Для вирішення поставлених завдань у роботі використовувалися методи фрактального і статистичного аналізів даних (для дослідження властивостей самоподібності та розрахунку характеристик мережного трафіка); методи теорії ймовірностей і випадкових процесів (для побудови математичних моделей фрактального трафіка); методи лінійного програмування, теорії графів і оптимізації (для побудови математичних моделей мультисервісних мереж; імітаційне моделювання (для розробки методів управління потоками даних, перевірки запропонованих моделей і вироблення практичних рекомендацій).

У роботі удосконалено математичну модель системи балансування навантаження, в якій балансувальник навантаження описується за допомогою системи масового обслуговування. Стани серверів описуються обсягом вільних ресурсів ЦПУ і обсягом вільної оперативної пам'яті. Всі значення параметрів моделі мають залежність від часу. Така модель дозволяє описувати поведінку розподіленої мережі в часі для різних класів обслуговування вхідного трафіка,

при заданих обмеженнях на час очікування пакета в черзі і кількість втрачених пакетів.

Для удосконалення моделі балансування навантаження з урахуванням мультифрактальних характеристик вхідних потоків були проведени дослідження фрактальних властивостей сумарних потоків різного типу. Оскільки сучасні інформаційні мережі побудовані на основі мультиплексування потоків даних, одним із важливих завдань для підвищення якості обслуговування мереж є дослідження властивостей адитивних трафіків. Проведено дослідження зміни фрактальних властивостей для адитивних інформаційних потоків у випадках, коли хоча б один з сумованих потоків має самоподібні або мультифрактальні властивості. Результати дослідження дозволяють визначати характеристики фрактального трафіку при мультиплексуванні потоків даних в комп'ютерній мережі.

Набув подальшого розвитку метод розрахунку дисбалансу системи на основі оцінки завантаження вузлів розподіленої системи. Як оцінка завантаження ресурсів вузлів запропоновано характеристики завантаження процесора, пам'яті і пропускної здатності каналу. Розраховується середнє завантаження процесора, пам'яті і пропускної здатності каналу на основі завантаження, яка виміряна системою обліку або моніторингом операційної системи. Запропонований метод дозволяє проводити розрахунок завантаження для потоків різних класів обслуговування, як для кожного сервера окремо, так і для всієї системи. Введено комплексне значення дисбалансу навантаження сервера, що враховує вагові коефіцієнти для процесора, пам'яті і пропускної здатності мережі. Введені вагові коефіцієнти дозволяють визначити значущість кожної характеристики сервера по відношенню одна до одної. Таким чином, цей метод дозволяє обчислити дисбаланс всіх серверів системи й ефективність використання ресурсів системи.

На основі розробленої математичної моделі системи балансування навантаження запропоновано динамічний метод розподілу навантаження, який враховує мультифрактальні властивості трафіка і задані обмеження на час очіку-

вання і кількість втрачених пакетів. Для цього було проведено порівняльний аналіз алгоритмів балансування навантаження за такими показниками продуктивності: пропускна здатність, витрати, відмовостійкість, час міграції, час відгуку, використання ресурсів, масштабованість, продуктивність. На основі проведеного аналізу було обрано найпродуктивніші алгоритми. Також у ході роботи було проведено порівняння різноманітних типів сценаріїв розподілу навантаження у розподіленому середовищі.

Розроблено програмне забезпечення для виконання імітаційного моделювання роботи балансувальника розподіленої системи. Методами імітаційного моделювання виконано порівняння показників якості обслуговування мережі для найбільш затребуваних алгоритмів балансування навантаження. На підставі результатів досліджень запропоновано рекомендації щодо вибору алгоритмів балансування навантаження в комп'ютерних мережах із фрактальним трафіком залежно від мережних параметрів і характеристик.

Дослідження показали, що дисбаланс системи суттєво залежить від мультифрактальних характеристик трафіка. При невеликих значеннях показника Херста та невеликій неоднорідності система балансування приходить у рівноважний стан, а значення дисбалансу прагне до нуля. Зі збільшенням показника Херста з часом дисбаланс системи не затухає, а система балансування не приходить до рівноважного стану. З більшими значеннями показника Херста та великою неоднорідністю система балансування знаходиться у нестійкому стані та значення дисбалансу змінюється у декілька разів, що призводить до максимального завантаження ресурсів.

Запропонований метод балансування навантаження завдяки аналізу та обліку мультифрактальних властивостей вхідного потоку забезпечує статистично рівномірний розподіл навантаження на серверах, високі показники продуктивності і пропускної здатності, а також зниження часу відгуку і кількості втрачених даних.

Розроблено програмні засоби для реалізації моделей і методів балансування навантаження розподілених систем, які враховують фрактальні власти-

вості трафіка та дозволяють забезпечити задану якість послуг і підвищити ефективність використання обладнання та каналів передачі інфокомунікаційної мережі. Проведено тестування і верифікацію програмного продукту для балансування навантаження використовуючи запропоновані моделі і методи балансування навантаження шляхом багаторазового імітаційного моделювання у розподілених інфокомунікаційних мережах.

На підставі проведених досліджень і практичної реалізації представлених методів розроблено методичне та програмне забезпечення, що використовується в навчальному процесі Харківського національного університету радіоелектроніки при вивченні дисциплін «Комп'ютерні мережі» та «Корпоративні комп'ютерні мережі».

Матеріали дисертації достатньо повно викладені в 29 роботах: з них 6 статей у наукових фахових виданнях України з технічних наук, 3 статті - в міжнародних наукових журналах за кордоном та 20 матеріалів і тез доповідей конференцій (з них 6 входить до наукометричної бази Scopus).

Ключові слова: розподілені комп'ютерні системи, балансування навантаження, параметри якості обслуговування, дисбаланс обчислювальних ресурсів системи, самоподібність завантаження, мультифрактальний трафік.

Список публікацій здобувача

1. Иванисенко И.Н. Проблемы моделирования дискретно-событийных систем // Вісник Харківського Університету. – Серія: Актуальні проблеми сучасної науки в дослідженнях молодих вчених м. Харкова. – 2002. – № 551. – С. 195–199.

2. Горбачов В.О. Суб'єктно-об'єктна модель доступу з використанням апаратних закладок до комп'ютерної інформації / В.О. Горбачов, В.В. Степаненко, І.М. Іванісенко // Радиоэлектроника и інформатика. – 2006. – Том 6, №3. – С. 47–50.

3. Горбачев В.А. Классификация и формальные модели аппаратных закладных устройств // В.А. Горбачов, И.Н. Иванисенко // Прикладна

радіоелектроніка та інформатика. – 2007. – Том 6, № 2. – С. 306–310.

4. Иванисенко И.Н. Имитационное моделирование облачных сервисов с учетом самоподобных свойств входящих потоков. Информатика, математическое моделирование, экономика / И.Н. Иванисенко, Ю.А. Кобицкая // Сб. научных статей по итогам Четвертой Междунар. науч.-практ. конф. – 23–25 апр. 2014. – Россия, Смоленск, – Том 1. – С. 79–83.

5. Иванисенко И.Н. Методы балансировки с учетом мультифрактальных свойств загрузки / И.Н. Иванисенко, Л.О. Кириченко, Т.А. Радивилова // Information content and processing. – 2015. – Vol. 2 (4). – P. 345–368.

6. Ivanisenko Igor. Methods and Algorithms of load balancing / Igor Ivanisenko // Information technologies and knowledge. – 2015. – Vol. 9, N. 4. – P. 340–375.

7. Иванисенко И.Н. Об одном методе распределения нагрузки с учетом мультифрактальных свойств трафика / И.Н. Иванисенко, Л.О. Кириченко, Т.А. Радивилова // Системні технології. – 2016. – Вип. 3(104). – С. 125–135. (Входить до міжнар. наукометричних баз Index Copernicus).

8. Иванисенко И.Н. Анализ дисбаланса распределенной системы при самоподобной нагрузке / И.Н. Иванисенко, Л.О. Кириченко, Т.А. Радивилова // Вісник Херсонського національного технічного університету. – 2016. – Вип. 3(58). – С. 224–231. (Входить до міжнар. наукометричних баз: РИНЦ (eLibrary), Google Scholar).

9. Радивилова Т.А. Динамический метод оценки загрузки узлов распределенной системы [Електронний ресурс] / И.Н. Иванисенко, Т.А. Радивилова // Проблеми телекомунікацій. – 2016. – № 1(18). – С. 42–51. – Режим доступу до журн.:
http://pt.journal.kh.ua/2016/1/1/161_radivilova_utilization.pdf .

10. Иванисенко И.Н. О проблеме решения задач оптимального распределения ресурсов в вычислительной системе / И.Н. Иванисенко // Проблемы информатики и моделирования: Восьмая междунар. науч.-техн. конф., 26-28 нояб.2008: матер. конф. – Харьков, Украина, 2008. – С. 53.

11. Иванисенко И.Н. Модель оптимального использования ресурсов IP-сети путем оптимизации трафика / И.Н. Иванисенко // Проблемы информатики и моделирования: Девятая междунар. науч.-техн. конф., 26-28 нояб. 2009: матер. конф. – Харьков, Украина, 2009. – С. 44.

12. Иванисенко И.Н. Методы решения задачи балансировки вычислительной нагрузки в сети распределенных вычислений / И.Н. Иванисенко // Информационные технологии в навигации и управлении: состояние и перспективы развития: Первая научно-техн. конф., 5–6 июля 2010: матер. конф. – Киев, Украина, 2010. – С. 26.

13. Иванисенко И.Н. Обзор и анализ программного обеспечения для решения задачи балансировки нагрузки компьютерной сети / И.Н. Иванисенко // Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління: Перша наук.-техн. конф., 13–14 груд. 2010: матер. конф. – Київ, Україна, 2010. – С. 79.

14. Ivanisenko Igor. Load Balancing in Cloud Services Considering the Self-Similar Properties of Incoming Flows / Igor Ivanisenko, Ludmila Kirichenko, Tamara Radivilova // Modern Problems of Radio Engineering, Telecommunications and Computer Science: XII th International Conference, February 25 – March 1, 2014: Abstract Book. – Lviv-Slavske, Ukraine. – P. 571–572.

15. Иванисенко И.Н. Балансировка нагрузки в облачных сервисах с учетом самоподобных свойств входящих потоков / И.Н. Иванисенко // Радиоэлектроника и молодежь в XXI веке: Междунар. молодеж. форум, 14–16 апр. 2014: матер. форума. – Харків, Україна. – Том 5. – С. 74.

16. Ivanisenko Igor. Investigation of Self-Similar Properties of Additive Data Traffic / Igor Ivanisenko, Ludmila Kirichenko, Tamara Radivilova // Computer Science and Information Technologies (CSIT 2015): X th International Scientific and Technical Conference, 14–17 September 2015: Abstract Book. – Lviv, Ukraine. – P. 169–172. (Входить до міжнар. наукометричної бази Scopus).

17. Иванисенко И.Н. Анализ методов балансировки нагрузки в распределённых системах / И.Н. Иванисенко, Т.А. Радивилова // Сучасні напрямки роз-

витку інформаційно-комунікаційних технологій та засобів управління: П’ята міжнар. наук.-техн. конф., 23–25 квіт. 2015: матер. конф. – Полтава-Баку-Кіровоград-Харків, Україна. – С. 20–21.

18. Ivanisenko Igor. Survey of Major Load Balancing Algorithms in Distributed System / Igor Ivanisenko, Tamara Radivilova // Інформаційні технології у інноваційному бізнесі (ITIB 2015): II Міжнар. наук.-практ. конф., 7–9 жовтня 2015: матер. конф. – Харків, Україна. – С. 89–92. (Входить до міжнар. наукометричної бази Scopus).

19. Ivanisenko Igor. The multifractal load balancing method / Igor Ivanisenko, Tamara Radivilova // Problems of Infocommunications Science and Technology: Second Intern. Scien. Pract. Conf., October 13–15 2015: Abstract Book. – Kharkiv, Ukraine. – P. 122–123. (Входить до міжнар. наукометричної бази Scopus).

20. Иванисенко И.Н. Исследование зашумленных мультифрактальных рядов / И.Н. Иванисенко, Л.О. Кириченко, А.Ю. Хабачева // Информационные системы и технологии: Междунар. науч.-техн. конф., 21–27 сент. 2015: матер. конф. – Харьков, Украина. – С. 62–63.

21. Ivanisenko Igor. Using cloud storage services in decision support system / Igor Ivanisenko, Yu Kobytska // Проблеми автоматизації: Третя міжнар. наук.-техн. конф., 12–13 лист. 2015: тези доп. – Черкаси-Баку-Бельсько-Бяла-Полтава, Украина. – С. 29.

22. Іванісенко І.М. OPEN SOURCE продукти балансування навантаження / І.М. Іванісенко // Free and Open Source Software: VII Всеукр. наук.-практ. конф., 24–27 лист. 2015: матер. конф. – Харків, Україна. – С 85.

23. Иванисенко И.Н. Динамическая балансировка фрактального трафика / И.Н. Иванисенко, Л.О. Кириченко, Т.А. Радивилова // Інформаційні технології в металургії та машинобудуванні (ITMM - 2016): Міжнар. наук.-техн. конф., 29–31 бер. 2016: матер. конф. – Дніпропетровськ, Україна. – С. 58.

24. Kirichenko Lyudmila. Dynamic load balancing algorithm of distributed systems / Lyudmila Kirichenko, Igor Ivanisenko, Tamara Radivilova // Modern Problems of Rradio Engineering, Telecommunications and Computer Science: XIIIth In-

tern. Conf. TCSET'2016, February 23–26, 2016: Abstract Book. – Lviv-Slavsko, Ukraine. – P. 515–518. (Входить до міжнар. наукометричної бази Scopus).

25. Іванісенко І.М. Балансування навантаження з урахуванням рівня дисбалансу системи / І.М. Іванісенко // Сучасні напрямки розвитку інформаційно-комунікаційних технологій та засобів управління: П'ята міжнар. наук.-техн. конф., 21–22 квіт. 2016: матер. конф. – Полтава-Баку-Кіровоград-Харків, Україна. – С. 32.

26. Радивилова Т.А. Алгоритм балансировки самоподобной нагрузки / Т.А. Радивилова, И.Н. Иванисенко, Л.О. Кириченко // Современные информационные и электронные технологии: XVII междунар. науч.-практ. конф., 23–27 мая 2016: труды конф. – Одесса, Украина – С. 115–117.

27. Kirichenko Lyudmila. Investigation of multifractal properties of additive data stream / Lyudmila Kirichenko, Igor Ivanisenko, Tamara Radivilova // 2016 IEEE First International Conference on Data Mining and Processing: First Intern. Conf., 23–27 August 2016: Abstract Book. – Lviv, Ukraine. – P. 305–308. (Входить до міжнар. наукометричної бази Scopus).

28. Кириченко Л.О. Аналіз стану розподіленої системи при самоподібному навантаженні / Л.О. Кириченко, И.Н. Иванисенко, Т.А. Радивилова // Інформаційні управляючі системи та технології: Міжнар. наук.-практ. конф., 20-22 вер. 2016: матер. конф. – Одеса, Україна. – С. 115–117.

29. Kirichenko Lyudmila. Calculation of distributed system imbalance / Lyudmila Kirichenko, Igor Ivanisenko, Tamara Radivilova // Problems of Infocommunications. Science and Technology (PICS&T-2016): Third Intern. Scien.-Pract. Conf., October 4–6 2016: Abstract Book. – Kharkiv, Ukraine. – P. 156–159 (Входить до міжнар. наукометричної бази Scopus).

ABSTRACT

Ivanisenko I.N. Methods of load balancing in distributed systems, taking into account the self-similar properties of input streams. – Qualifying scientific work on the manuscript rights.

A thesis for the candidate degree in technical sciences in the specialty 05.13.05 - computer systems and components. – Kharkiv National University of Radio electronics, Ministry of Education and Science of Ukraine, Kharkiv, 2017.

The thesis proposes a new solution to the actual scientific and practical problem of developing load balancing methods that take into account the self-similarity of traffic and the use of resources of each node and the entire distributed system. The object of research is the processes of traffic management in distributed infocommunication systems. The subject of the study is the models and methods of balancing self-similar load in distributed systems.

The next methods were used to solve the problems: the methods of fractal and statistical analysis (to study the properties of self-similarity and the calculating characteristics of network traffic); methods of probability theory and stochastic processes (to construct mathematical models of fractal traffic); methods of linear programming, graphs theory and optimization (to construct mathematical models of multiservice networks); simulation (to develop methods of flow control, verification of proposed models and develop practical recommendations).

The mathematical model of the load balancing system where the load balancer is described using a mass maintenance system is improved in the work. The states of servers are described by the amount of available CPU and RAM resources. All values of the model parameters depending on time. This model allows to describe the behavior of the distributed network in time for different service classes of incoming traffic, with preset limits on the waiting time of the packet in a queue and the quantity of lost packets.

To improve load balancing model taking into account the multifractal characteristics of the incoming flow, the fractal properties of different types of summaried

flows were studied. Because of modern information networks are based on multiplexing data streams, one of the important tasks for improving the quality of service network is the study of additive traffic properties. The research of the change of fractal properties for additive information flows was carried out in cases where at least one of the summurzed flows has self-similar or multifractal properties. The research results allow determining the characteristics of fractal traffic at multiplexing data streams in a computer network.

The method of calculating the system imbalance has been further developed based on the evaluation of the load distributed system nodes. As an estimate of the nodes' resources load, the characteristics of loading the processor, memory and bandwidth of the channel are proposed.

The average CPU, memory, and bandwidth load are calculated based on the load measured by the accounting or monitoring system of the operating system. The proposed method allows to calculate the load for flows of various clases of service for each server separately and for entire system. The complex value of server load imbalance is introduced, taking into account the weight coefficients for the CPU, memory and bandwidth of network. The introduced weight coefficients allow to determine the significance of each server characteristic in relation to each other. Thus, this method allows to calculate the imbalance of all system servers and the efficiency of using the system resources.

Based on the developed mathematical model of the load balancing system, a dynamic load distribution method is proposed which takes into account the multifractal properties of the traffic and specifies the constraints on the waiting time and the number of lost packets.

Dynamic method of load distribution proposed on the basis of a mathematical model of load balancing that takes into account the properties of multifractal traffic and set limits on the waiting time and the quantity of lost packets. For this purpose, a comparative analysis of load balancing algorithms was performed on such productivity indicators: bandwidth, costs, fault tolerance, migration time, response time, resource utilization, scalability, and productivity. On the basis of the analysis, the most

productive algorithms were chosen. Also, during the performance of work, a comparison of different types of load distribution scenarios in a distributed environment was performed.

The software for simulation of balancer activity of distributed system is developed. Using simulation methods, a comparison of networks' quality of service parameters was performed for the most demanded load balancing algorithms. Based on the results of the research, recommendations were made for choosing load balancing algorithms in computer networks with fractal traffic depending on network parameters and characteristics.

Studies have shown that the imbalance of the system is significantly dependent on the multifractal characteristics of the traffic. With small values of the Hurst index and a small inhomogeneity, the balancing system comes to equilibrium, and the value of the imbalance tends to zero. With the increase of the Hurst index over time, the system imbalance does not fade, and the balancing system does not come to equilibrium. With higher values of the Hurst index and high heterogeneity, the balancing system is in an unstable state and the imbalance value changes several times, that leads to maximum resources load.

The proposed load balancing method through analysis and accounting multifractal properties of the input flow provides a statistically uniform load distribution on servers, high performance and capacity, and reduced response times and the quantity of lost data.

Software tools for implementation of load balancing models and methods of distributed systems that take into account fractal properties of traffic and allow to provide the upset quality of services and increase utilization of equipment and communication channels of an infocommunication network are developed.

The testing and verification of the software for load balancing has been carried out using the proposed models and load balancing methods by multiple simulation of distributed infocommunication networks.

Based on the researches and practical implementation of the presented methods, the methodical and software tools used in the educational process of the Kharkiv

National University of Radio electronics in the study of the disciplines "Computer Networks" and "Corporate Computer Networks" have been developed.

The materials of the thesis are sufficiently detailed in 29 papers: 6 of them in the scientific specialized editions of Ukraine on technical sciences, 3 articles in international scientific journals abroad, and 20 materials and abstracts of conference (6 of which is part of the Scopus science base).

Keywords: distributed computer systems, load balancing, quality of service parameters, imbalance of system computing resources, self-similar loading, multifractal traffic.

List of publications of the applicant

1. Иванисенко И.Н. Проблемы моделирования дискретно-событийных систем // Вісник Харківського Університету. – Серія: Актуальні проблеми сучасної науки в дослідженнях молодих вчених м. Харкова. – 2002. – № 551. – С. 195–199.

2. Горбачов В.О. Суб'єктно-об'єктна модель доступу з використанням апаратних закладок до комп'ютерної інформації / В.О. Горбачов, В.В. Степаненко, І.М. Іванісенко // Радиоэлектроника и інформатика. – 2006. – Том 6, №3. – С. 47–50.

3. Горбачев В.А. Классификация и формальные модели аппаратных закладных устройств // В.А. Горбачов, И.Н. Иванисенко // Прикладна радіоелектроніка та інформатика. – 2007. – Том 6, № 2. – С. 306–310.

4. Иванисенко И.Н. Имитационное моделирование облачных сервисов с учетом самоподобных свойств входящих потоков. Информатика, математическое моделирование, экономика / И.Н. Иванисенко, Ю.А. Кобицкая // Сб. научных статей по итогам Четвертой Междунар. науч.-практ. конф. – 23–25 апр. 2014. – Россия, Смоленск, – Том 1. – С. 79–83.

5. Иванисенко И.Н. Методы балансировки с учетом мультифрактальных свойств загрузки / И.Н. Иванисенко, Л.О. Кириченко, Т.А. Радивилова // Information content and processing. – 2015. – Vol. 2 (4). – P. 345–368.

6. Ivanisenko Igor. Methods and Algorithms of load balancing / Igor Ivanisenko // Information technologies and knowledge. – 2015. – Vol. 9, N. 4. – P. 340–375.

7. Иванисенко И.Н. Об одном методе распределения нагрузки с учетом мультифрактальных свойств трафика / И.Н. Иванисенко, Л.О. Кириченко, Т.А. Радивилова // Системні технології. – 2016. – Вип. 3(104). – С. 125–135. (Входить до міжнар. наукометричних баз Index Copernicus).

8. Иванисенко И.Н. Анализ дисбаланса распределенной системы при самоподобной нагрузке / И.Н. Иванисенко, Л.О. Кириченко, Т.А. Радивилова // Вісник Херсонського національного технічного університету. – 2016. – Вип. 3(58). – С. 224–231. (Входить до міжнар. наукометричних баз: РИНЦ (eLibrary), Google Scholar).

9. Радивилова Т.А. Динамический метод оценки загрузки узлов распределенной системы [Електронний ресурс] / И.Н. Иванисенко, Т.А. Радивилова // Проблеми телекомунікацій. – 2016. – № 1(18). – С. 42–51. – Режим доступу до журн.:

http://pt.journal.kh.ua/2016/1/1/161_radivilova_utilization.pdf .

10. Иванисенко И.Н. О проблеме решения задач оптимального распределения ресурсов в вычислительной системе / И.Н. Иванисенко // Проблемы информатики и моделирования: Восьмая междунар. науч.-техн. конф., 26-28 нояб.2008: матер. конф. – Харьков, Украина, 2008. – С. 53.

11. Иванисенко И.Н. Модель оптимального использования ресурсов IP-сети путем оптимизации трафика / И.Н. Иванисенко // Проблемы информатики и моделирования: Девятая междунар. науч.-техн. конф., 26-28 нояб. 2009: матер. конф. – Харьков, Украина, 2009. – С. 44.

12. Иванисенко И.Н. Методы решения задачи балансировки вычислительной нагрузки в сети распределенных вычислений / И.Н. Иванисенко // Информационные технологии в навигации и управлении: состояние и перспективы развития: Первая научно-техн. конф., 5–6 июля 2010: матер. конф. – Киев, Украина, 2010. – С. 26.

13. Иванисенко И.Н. Обзор и анализ программного обеспечения для решения задачи балансировки нагрузки компьютерной сети / И.Н. Иванисенко // Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління: Перша наук.-техн. конф., 13–14 груд. 2010: матер. конф. – Київ, Україна, 2010. – С. 79.

14. Ivanisenko Igor. Load Balancing in Cloud Services Considering the Self-Similar Properties of Incoming Flows / Igor Ivanisenko, Ludmila Kirichenko, Tamara Radivilova // Modern Problems of Radio Engineering, Telecommunications and Computer Science: XII th International Conference, February 25 – March 1, 2014: Abstract Book. – Lviv-Slavske, Ukraine. – P. 571–572.

15. Иванисенко И.Н. Балансировка нагрузки в облачных сервисах с учетом самоподобных свойств входящих потоков / И.Н. Иванисенко // Радиоэлектроника и молодежь в XXI веке: Междунар. молодеж. форум, 14–16 апр. 2014: матер. форума. – Харків, Україна. – Том 5. – С. 74.

16. Ivanisenko Igor. Investigation of Self-Similar Properties of Additive Data Traffic / Igor Ivanisenko, Ludmila Kirichenko, Tamara Radivilova // Computer Science and Information Technologies (CSIT 2015): X th International Scientific and Technical Conference, 14–17 September 2015: Abstract Book. – Lviv, Ukraine. – P. 169–172. (Входить до міжнар. наукометричної бази Scopus).

17. Иванисенко И.Н. Анализ методов балансировки нагрузки в распределённых системах / И.Н. Иванисенко, Т.А. Радивилова // Сучасні напрямки розвитку інформаційно-комунікаційних технологій та засобів управління: П’ята міжнар. наук.-техн. конф., 23–25 квіт. 2015: матер. конф. – Полтава-Баку-Кіровоград-Харків, Україна. – С. 20–21.

18. Ivanisenko Igor. Survey of Major Load Balancing Algorithms in Distributed System / Igor Ivanisenko, Tamara Radivilova // Інформаційні технології у інноваційному бізнесі (ITIB 2015): II Міжнар. наук.-практ. конф., 7–9 жовтня 2015: матер. конф. – Харків, Україна. – С. 89–92. (Входить до міжнар. наукометричної бази Scopus).

19. Ivanisenko Igor. The multifractal load balancing method / Igor Ivanisenko,

Tamara Radivilova // Problems of Infocommunications Science and Technology: Second Intern. Scien. Pract. Conf., October 13–15 2015: Abstract Book. – Kharkiv, Ukraine. – P. 122–123. (Входить до міжнар. наукометричної бази Scopus).

20. Иванисенко И.Н. Исследование зашумленных мультифрактальных рядов / И.Н. Иванисенко, Л.О. Кириченко, А.Ю. Хабачева // Информационные системы и технологии: Междунар. науч.-техн. конф., 21–27 сент. 2015: матер. конф. – Харьков, Украина. – С. 62–63.

21. Ivanisenko Igor. Using cloud storage services in decision support system / Igor Ivanisenko, Yu Kobytska // Проблеми автоматизації: Третя міжнар. наук.-техн. конф., 12–13 лист. 2015: тези доп. – Черкаси-Баку-Бельсько-Бяла-Полтава, Украина. – С. 29.

22. Іванісенко І.М. OPEN SOURCE продукти балансування навантаження / І.М. Іванісенко // Free and Open Source Software: VII Всеукр. наук.-практ. конф., 24–27 лист. 2015: матер. конф. – Харків, Україна. – С 85.

23. Иванисенко И.Н. Динамическая балансировка фрактального трафика / И.Н. Иванисенко, Л.О. Кириченко, Т.А. Радивилова // Інформаційні технології в металургії та машинобудуванні (ІТММ - 2016): Міжнар. наук.-техн. конф., 29–31 бер. 2016: матер. конф. – Дніпропетровськ, Україна. – С. 58.

24. Kirichenko Lyudmila. Dynamic load balancing algorithm of distributed systems / Lyudmila Kirichenko, Igor Ivanisenko, Tamara Radivilova // Modern Problems of Rradio Engineering, Telecommunications and Computer Science: XIIIth Intern. Conf. TCSET'2016, February 23–26, 2016: Abstract Book. – Lviv-Slavsko, Ukraine. – P. 515–518. (Входить до міжнар. наукометричної бази Scopus).

25. Іванісенко І.М. Балансування навантаження з урахуванням рівня дисбалансу системи / І.М. Іванісенко // Сучасні напрямки розвитку інформаційно-комунікаційних технологій та засобів управління: П'ята міжнар. наук.-техн. конф., 21–22 квіт. 2016: матер. конф. – Полтава-Баку-Кіровоград-Харків, Україна. – С. 32.

26. Радивилова Т.А. Алгоритм балансировки самоподобной нагрузки / Т.А. Радивилова, И.Н. Иванисенко, Л.О. Кириченко // Современные информа-

ционные и электронные технологии: XVII междунар. науч.-практ. конф., 23–27 мая 2016: труды конф. – Одесса, Украина – С. 115–117.

27. Kirichenko Lyudmila. Investigation of multifractal properties of additive data stream / Lyudmila Kirichenko, Igor Ivanisenko, Tamara Radivilova // 2016 IEEE First International Conference on Data Mining and Processing: First Intern. Conf., 23–27 August 2016: Abstract Book. – Lviv, Ukraine. – P. 305–308. (Входить до міжнар. наукометричної бази Scopus).

28. Кириченко Л.О. Аналіз стану розподіленої системи при самоподібному навантаженні / Л.О. Кириченко, И.Н. Иванисенко, Т.А. Радивилова // Інформаційні управляючі системи та технології: Міжнар. наук.-практ. конф., 20-22 вер. 2016: матер. конф. – Одеса, Україна. – С. 115–117.

29. Kirichenko Lyudmila. Calculation of distributed system imbalance / Lyudmila Kirichenko, Igor Ivanisenko, Tamara Radivilova // Problems of Infocommunications. Science and Technology (PICS&T-2016): Third Intern. Scien.-Pract. Conf., October 4–6 2016: Abstract Book. – Kharkiv, Ukraine. – P. 156–159 (Входить до міжнар. наукометричної бази Scopus).

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ADC (Application Delivery Controllers) – контролери доставки сервісів

ARP (Address Resolution Protocol) – протокол визначення адреси

CBQ (Class-Based Queuing) – черга, яка заснована на класифікації потоків по класу обслуговування

CoS (Class of Service) – клас обслуговування

CWND (Congestion Window) – управління вікном перевантаження

DiffServ (Differentiated Service) – диференційований сервіс обслуговування

DNS (Domain Name System) – система доменних імен

DS (Differentiated Services) – диференційована послуга

DSCP (Differentiated Services Code Point) – точка коду диференційованих послуг

HTTP (HyperText Transfer Protocol) – протокол передачі гіпертекста

ICMP (Internet Control Message Protocol) – протокол управління повідомленнями

IntServ (Integrated Service) – інтегрований сервіс обслуговування

IP (Internet Protocol) – міжмережний протокол

LX (level of model OSI) – рівень моделі OSI

MAC-adress (Media Access Control) – управління доступом до середовища

NAT (Translation of Network Address) – перетворення мережних адрес

NFS (Network File System) – мережна файлова система

NPB (Network Packet Broker) – мережний пакетний брокер

QoS (Quality of Service) – якість обслуговування сервісів

PHB (per-hop behavior) – міжвузловий перехід

RED (Random Early Discard) – довільне раннє виявлення

RSVP (Resource ReSerVation Protocol) – протокол резервування мережних ресурсів

RTT (Round Trip Time) – час між відправленням запиту й одержанням відповіді

SLA (Service Level Agreement) – угода про рівень обслуговування

SSTHRETH (slow start threshold) – процедура повільного старту

TE (Traffic Engineering) – управління трафіком

TCP (Transmission Control Protocol) – протокол керування передачею

ToS (Type of Service) – тип обслуговування.

VPN (Virtual Private Network) – віртуальна приватна мережа.

WFQ (Weighted Fair Queuing) – зважена справедлива черга

WRED (Weighted random early detection) – зважене довільне раннє виявлення

МФДФА – мультифрактальний детрендований флуктуаційний аналіз

ОС – операційна система

СМО – система масового обслуговування

ФБР – фрактальний броунівський рух

ФГШ – фрактальний гауссівський шум

ЦПП – центральний процесорний пристрій

ВСТУП

Актуальність теми. У сьогоднішній час разом зі збільшенням швидкостей передачі даних в інфокомунікаціях збільшується частка інтерактивного трафіку, вкрай чутливого до параметрів середовища транспортування. Для надання необхідної кількості ресурсів при передачі різних видів трафіку, що пред'являють різні вимоги до характеристик телекомунікаційної мережі, використовуються різні механізми забезпечення якості обслуговування QoS (Quality of service). Одним з таких механізмів є балансування навантаження. Система балансування навантаження вирішує завдання забезпечення якості обслуговування і підвищення продуктивності розподілених систем за рахунок оптимального розподілу завдань між вузлами обчислювальної системи.

Завдання балансування завантаження в розподілений інфокомунікаційній системі полягає в тому, щоб виходячи з комплекту завдань, що включають обчислення і передачу даних, і системи серверів різної ресурсоємності, знайти такий розподіл завдань по серверам, який забезпечує приблизно рівне обчислювальне завантаження кожного сервера і мінімальні витрати на передачу даних. Для виконання цього завдання можуть використовуватися різні методи та алгоритми балансування навантаження, які враховують оцінки завантаження обчислювального вузла.

Найбільш відомими дослідженнями в галузі управління навантаженням в розподілених системах, балансування, теоретичних досліджень і розробки фундаментальних основ розподілу навантаження, в створенні математичного апарату, моделей і методів управління для розподілу навантаження займалися такі вчені як Є.І. Ігнатенко, В.Н. Тарасов, F. Wang, V. Cardellini, Xing-Guo Luo, Hisao Kameda, H. Mehta, P. Kanungo, M. Casalicchio, Y.S. Hong, а також інші дослідники, що працюють над проблемами розподілу навантаження. Розробляли і вдосконалили алгоритми балансування навантаження такі науковці як S. Keshav, O. Elzeki, M. Reshad, H. Chen, Y. Hu, Shamsollah

Ghanbari, Ratan Mishra, Dhinesh Babu L.D., а також багато інших.

Експериментальні дослідження, проведені в останні десятиліття, свідчать, що трафік у багатьох мультисервісних комп'ютерних мережах має самоподібні (фрактальні) властивості. Причина такого ефекту полягає в особливостях розподілу файлів по серверам, їх розмірах, типової поведінці користувачів, і в значній мірі пов'язана зі змінами мережевих ресурсів і топології мережі. Самоподібний трафік викликає значні затримки і втрати пакетів, навіть якщо сумарна інтенсивність всіх потоків далека від максимально допустимих значень.

Існує велика кількість публікацій, присвячених аналізу фрактальних властивостей трафіку. Самоподібні властивості інформаційних потоків виявлені в локальних і глобальних мережах, зокрема в трафіку Ethernet, ATM, додатках TCP, IP, VoIP. Значний внесок у розвиток теорії самоподібних процесів, дослідження фрактальних властивостей телетрафіка і побудові моделей фрактального трафіку внесли K. Park, W. Willinger, P. Abry, MS Taqqu, I. Norros, Потапов А.А., Цибаков Б.С., Шелухін А.І. та інші вчені.

Наявність у переданих клієнтами інформаційних потоків властивостей самоподібності дуже впливає на ефективність роботи розподілених систем. Особливо важливу роль це відіграє для роботи сервісів, що забезпечують передачу мультимедійного трафіку і трафіку реального часу. Таким чином, актуальною є задача розробки та аналізу методів балансування навантаження, які враховують самоподібність трафіку і завантаження кожного вузла та всієї розподіленої системи.

Мета і завдання досліджень. Метою роботи є розробка модифікованих методів балансування навантаження в розподілених системах, які враховують фрактальні властивості трафіку і дозволяють забезпечити високий рівень якості обслуговування.

Відповідно до поставленої мети необхідно вирішити такі наукові завдання:

– провести огляд і аналіз методів балансування навантаження в

розподілених системах і досліджень впливу фрактальних властивостей трафіку на якість обслуговування в мережах;

– удосконалити метод розрахунку дисбалансу завантаження ресурсів розподіленої системи для різних класів обслуговування вхідного трафіку, який включає в себе комплексний вимір загального рівня дисбалансу системи;

– за допомогою імітаційного моделювання дослідити вплив самоподібних і мультифрактальних властивостей трафіку на показники якості обслуговування в мережі для найбільш затребуваних алгоритмів балансування навантаження;

– удосконалити модель балансування навантаження розподіленої системи, з урахуванням мультифрактальних характеристик вхідних потоків;

– на основі удосконаленої моделі розробити метод балансування, який враховує мультифрактальні властивості трафіку і метод розрахунку дисбалансу ресурсів розподіленої системи;

– провести імітаційне моделювання запропонованих методів балансування навантаження і вирішити практичні завдання.

Об'єкт дослідження – процеси управління трафіком в розподілених інфокоммукаційних системах.

Предмет дослідження – моделі і методи балансування самоподібного навантаження в розподілених системах.

Методи дослідження. Для вирішення поставлених завдань у роботі використовувалися методи фрактального і статистичного аналізів даних (для дослідження властивостей самоподібності і розрахунку характеристик мережного трафіку); методи теорії ймовірностей і випадкових процесів (для побудови математичних моделей фрактального трафіку); методи лінійного програмування, теорії графів і оптимізації (для побудови математичних моделей мультисервісних мереж; імітаційне моделювання (для розробки методів управління потоками даних, перевірки запропонованих моделей і вироблення практичних рекомендацій).

Наукова новизна отриманих результатів. У дисертації отримані такі нові

наукові і практичні результати.

1. Вперше запропоновано метод балансування навантаження, який, на відміну від існуючих, враховує мультифрактальні властивості адитивного трафіку і розрахунок дисбалансу ресурсів розподіленої системи, що дозволяє підвищити ступінь використання ресурсів системи за рахунок спрямування неоднорідних інформаційних потоків на менш завантажені ресурси.

2. Удосконалено модель балансування навантаження розподіленої системи, яка, на відміну від існуючих, враховує обмеження на заданий набір характеристик мережі для різних класів обслуговування та зміну в часі фрактальних параметрів трафіку, що дозволяє розробити методи балансування з урахуванням певних вимог якості обслуговування.

3. Одержав подальший розвиток метод розрахунку дисбалансу ресурсів розподіленої системи, який, на відміну від існуючих, включає в себе комплексний вимір загального рівня дисбалансу системи та враховує вагові коефіцієнти ресурсів серверів, що дозволяє рівномірно розподіляти навантаження в неоднорідній системі.

Особистий внесок здобувача. Всі наукові результати дисертаційної роботи отримані автором самостійно і опубліковані в роботах [36-41, 55-57,117,118,123-137, 140, 146, 147]. У роботах, опублікованих у співавторстві, автору належать наступні результати: у роботі [117] досліджена модель доступу; у роботі [118] проведена класифікація і формальні моделі апаратних закладних пристроїв; у роботі [128] проведено імітаційне моделювання хмарних сервісів з урахуванням самоподібних властивостей вхідних потоків; у роботі [130] запропоновані модель і методи балансування самоподібного трафіку; у роботі [133] запропоновано метод розподілу навантаження з урахуванням мультифрактального властивостей трафіку; у роботі [134] виконано аналіз дисбалансу розподіленої системи при самоподобной навантаженні; у роботі [147] запропоновані методи розрахунку дисбалансу розподіленої системи; у роботі [36] проведено моделювання хмарних сервісів з урахуванням самоподібних властивостей вхідних потоків; у роботі [38]

досліджені самоподібні властивості адитивних потоків; у роботі [131] проведено аналіз методів балансування навантаження в розподілених системах; у роботі [39] виконано огляд алгоритмів балансування в розподілених системах; у роботі [40] запропоновано алгоритм балансування для мультифрактального трафіку; у роботі [132] проведено дослідження зашумлених мультифрактальних рядів; у роботі [41] проведено моделювання хмарних сервісів для самоподібних потоків; у роботі [135] досліджена динамічна балансування фрактального трафіку; у роботі [55] розглянуті алгоритми балансування самоподібним трафіком; у роботі [146] запропоновано метод балансування самоподібним трафіком; у роботі [56] досліджені мультифрактальні властивості адитивних потоків; у роботі [140] досліджено стан розподілених систем при самоподібному навантаженні; у роботі [57] запропоновано метод розрахунку дисбалансу розподіленої системи.

Апробація результатів дисертації. Основні результати проведених досліджень доповідалися і обговорювалися на наступних конференціях:

– 8-й і 9-й міжнародній науково-технічній конференції «Проблеми інформатики і моделювання» (Харків, 2008, 2009);

– 1-й, 2-й науково-технічній конференції «Інформаційні технології в навігації і управлінні: стан та перспективи розвитку» (Київ, 2010, 2011);

– 1-й, 6-й та 7-й науково-технічній конференції «Сучасні напрями розвитку інформаційно-комунікаційніх технологій та засобів управління» (Київ, 2010, Полтава, 2015, Харків, 2016);

– 18-му міжнародному молодіжному форумі «Радіоелектроніка і молодь в XXI столітті» (Харків, 2014);

– Xth International Scientific and Technical Conference «Computer science and information technologies» (Lviv, 2015);

– II Міжнародній науково-практичній конференції «Інформаційні технології у інноваційному бізнесі (ITIB 2015)» (Харків, 2015);

– Second and third International Scientific Practical Conference «Problems of Infocommunications Science and Technology» (Kharkiv, 2015, 2016);

– Міжнародній науково-технічній конференції «Інформаційні системи і технології» (Харків, 2015);

– 3-й міжнародній науково-технічній конференції «Проблеми автоматизації» (Черкаси, 2015);

– 7-й всеукраїнській науково-практичній конференції «FREE AND OPEN SOURCE SOFTWARE» (Харків, 2015);

– Міжнародній науково-технічній конференції «Інформаційні технології в металургії та машинобудуванні (ІТММ - 2016)» (Дніпропетровськ, 2016);

– XIIIth International Conference TCSET'2016 «Modern problems of radio engineering, telecommunications, and computer science» (Lviv-Slavsko, 2016);

– 17-й міжнародній науково-практичній конференції «Сучасні інформаційні та електронні технології» (Одеса, 2016);

– 2016 IEEE First International Conference on data mining and processing. (Львів, 2016);

– Міжнародній науково-практичній конференції «Інформаційні управляючі системи та технології» (Одеса, 2016).

Структура дисертації. Дисертаційна робота складається зі вступу, чотирьох розділів основної частини, висновків, списку використаних джерел, додатків.

У ***першому розділі*** на базі вивчення літературних джерел проведено аналіз предметної області. розглянуто основні поняття балансування навантаження в розподіленій комп'ютерній мережі. Визначено основні властивості основних алгоритмів балансування навантаження. Наведено їхню класифікацію за кількома типами. Виділено основні показники продуктивності: пропускна здатність, витрати, відмовостійкість, час міграції, час відгуку, використання ресурсів, масштабованість, продуктивність, а також наведена постановка задачі на дослідження.

У ***другому розділі*** розглянуто основні механізми забезпечення якості обслуговування мережі. Проведено аналіз механізмів збору інформації щодо завантаженості системи. Під дисбалансом розуміють ступінь рівномірності

розподілу навантаження між серверами. Однією з основних задач балансування навантаження є зменшення дисбалансу.

Запропонований метод розрахунку дисбалансу ресурсів розподіленої системи, на відміну від існуючих, передбачає комплексне вимірювання загального рівня дисбалансу системи і враховує вагові коефіцієнти ресурсів серверів, що дозволяє рівномірно розподіляти навантаження в неоднорідній системі. Також метод враховує розподіл потоків за класами обслуговування.

У ***третьому розділі*** запропоновано удосконалену модель балансування самоподібного навантаження розподіленої системи з урахуванням мультифрактальних характеристик вхідних потоків. Для цього було проведено дослідження зміни фрактальних властивостей для адитивних інформаційних потоків у випадках, коли хоча б один з сумованих потоків має самоподібні або мультифрактальні властивості. Результати дослідження дозволяють визначати характеристики фрактального трафіка при мультиплексуванні потоків даних в комп’ютерній мережі.

Балансувальник навантаження описується за допомогою системи масового обслуговування. Стани серверів описуються обсягом вільних ресурсів ЦПУ і обсягом вільної оперативної пам’яті. Всі значення параметрів моделі мають залежність від часу. Така модель дозволяє описувати поведінку розподіленої мережі в часі для різних класів обслуговування вхідного трафіка, при заданих обмеженнях на час очікування пакета в черзі і кількість втрачених пакетів.

У ***четвертому розділі*** запропоновано метод балансування навантаження, який враховує мультифрактальні властивості адитивного трафіка, і розрахунок дисбалансу ресурсів розподіленої системи. Розроблено програмне забезпечення для виконання імітаційного моделювання роботи балансувальника розподіленої системи. Показано, що запропонований метод балансування навантаження завдяки аналізу та обліку мультифрактальних властивостей вхідного потоку забезпечує статистично рівномірний розподіл навантаження на серверах.

Зв'язок роботи з науковими програмами, планами, темами.

Дисертаційна робота виконувалася відповідно до плану науково-технічних робіт Харківського національного університету радіоелектроніки в рамках держбюджетної теми «Розробка та дослідження застосування GRID-порталу харківського ресурсно-операційного GRID-центру», договору №08-22 (08.04.07-27.06.08) і №08-22/9 (01.07.08-30.09.08) між ХНУРЕ і «ІПСА» НТУУ «КПІ», що виконувалася на підставі Договору «ІПСА» НТУУ «КПІ» з Міністерством освіти і науки України №IT/506-2013, Державної програми «Інформаційні та телекомунікаційні технології в освіті і науці» на 2006-2013 роки (№ДР 0108U008261).

В рамках теми здобувачем були досліджені властивості інформаційних потоків даних у розподілених системах. Результати наукових досліджень використані в науково-технічних звітах про НДР №08-22/9.

Практичне значення результатів дисертації. Результати роботи можуть бути використані при розробці алгоритмів функціонування вузлів інфокомунікаційного обладнання з метою підвищення якості обслуговування і ефективності обробки трафіку, що має властивістю самоподібності. Розроблений метод балансування дозволяє забезпечити задану якість послуг і підвищити ефективність використання обладнання та каналів передачі в умовах навантаження, що має самоподібні властивості. Застосування розробленого методу розрахунку дисбалансу ресурсів розподіленої системи дозволяє проаналізувати можливість максимально ефективного використання ресурсів інфокомунікаційної мережі. Результати дисертаційної роботи впроваджені в навчальний процес кафедри електронних обчислювальних машин ХНУРЕ, м. Харків (акт від 16.01.2017 р.).

Основні результати дисертаційної роботи використовуються в навчальному процесі на кафедрі електронних обчислювальних машин Харківського національного університету радіоелектроніки при проведенні лекційних та лабораторних занять з дисциплін «Комп’ютерні мережі» та «Корпоративні комп’ютерні мережі».

Публікації. Основні результати дисертаційної роботи опубліковано в 29 роботах: з них 6 статей у наукових фахових виданнях України з технічних

наук, 3 статті - в міжнародних наукових журналах за кордоном та 20 матеріалів і тез доповідей конференцій (з них 6 входить до наукометричної бази Scopus).

Список джерел, використаних у даному розділі, наведений у повному списку використаних джерел під номерами: [1, 2, 7, 8, 11, 13, 20, 22, 26, 30, 31, 33, 36-41, 44, 48, 52, 55-57, 66, 68, 73,74, 101, 117,118,123-137, 138, 140, 146, 147, 150,155-158].

РОЗДІЛ 1
ОГЛЯД СТАНУ ПРОБЛЕМИ. ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

Сучасні мережі зв’язку характеризуються значними слабо передбачуваними коливаннями навантаження. Трафік сучасної глобальної мережі виявляє експоненціальне зростання, відбуваються значні структурні зміни, все більш відчутними стають низько передбачувані коливання навантажень. Процеси конвергенції мереж зв’язку привели до домінування протоколу IP у якості універсального для всіх видів даних, що передаються. Однак відсутність вбудованих механізмів інжинірингу трафіку ставить питання про необхідність розробки методів, що дозволяють більш ефективно використовувати можливості наявної мережевої інфраструктури, але не потребують зміни основ функціонування глобальної мережі. Одним з найбільш перспективних варіантів вирішення зазначених проблем на сьогоднішній день є динамічні механізми балансування трафіку, що викликають пильний інтерес наукової спільноти. Застосування даних методів, що реагують на зміни мережевих навантажень в режимі, близькому до реального часу, для локального пом’якшення тимчасових перевантажень в глобальній мережі дозволить позбутися недоліків, властивих існуючим мережам [44,60,82,100].

Експериментальні та чисельні дослідження, проведені в останні десятиліття, свідчать, що трафік у багатьох комп’ютерних мережах має самоподібні властивості. Самоподібний трафік викликає значні затримки і втрати пакетів, навіть у випадку, якщо сумарна інтенсивність усіх потоків далека від максимально допустимих значень. Самоподібні властивості інформаційних потоків виявлені у багатьох локальних і глобальних телекомунікаційних мережах [47,84,87]. У зв’язку з вищевикладеним почали активно досліджуватися механізми підвищення якості обслуговування і методи управління трафіком в комп’ютерних мережах, що функціонують в умовах самоподібного та мультифрактального трафіку [158].

У зв’язку з масовим поширенням розподілених обчислювальних систем стала актуальною проблема їх ефективного використання. Одним з аспектів даної проблеми є ефективне планування і розподіл завдань усередині розподілених обчислювальних систем з метою оптимізації використання ресурсів та скорочення часу обчислення. Вельми часто виникає ситуація, за якої частина обчислювальних ресурсів простоює, у той час, як інша частина ресурсів перевантажена і в черзі мається велика кількість задач, що очікують свого виконання.

Для оптимізації використання ресурсів, скорочення часу обслуговування запитів, горизонтального масштабування (динамічне додавання/видалення пристроїв), а також забезпечення відмовостійкості (резервування) застосовується метод рівномірного розподілу завдань між декількома мережевими пристроями (наприклад, серверами), який називається балансування навантаження, або вирівнювання навантаження (англ.: Load Balancing).

При появі нових завдання програмне забезпечення, що реалізує балансування, має прийняти рішення про те, на якому обчислювальному вузлі слід виконувати обчислення, пов’язані з цим новим завданням. Окрім того, балансування передбачає перенесення частини обчислень з найбільш завантажених обчислювальних вузлів на менш завантажені вузли. При віконанні завдань процесори обмінюються між собою комунікаційніми повідомленнями. У разі низьких витрат на комунікацію, деякі процесори можуть простоювати, у той час як інші будуть перевантажені. Також будуть недоцільними великі витрати на комунікацію. Отже, стратегія балансування повинна бути такою, щоб обчислювальні вузли були завантажені досить рівномірно, але й комунікаційне середовище не повинно бути перевантаженим.

1.1 Задачі балансування обчислювального навантаження у розподілених системах

Проблема балансування обчислювального навантаження виникає за кількома основними причинами [28,44, 60,76]: структура розподіленого додатка є

неоднорідною, різні логічні процеси вимагають різних обчислювальних потужностей; структура розподіленої системи також є неоднорідною, тобто різні обчислювальні вузли мають різну продуктивність; структура міжвузлової взаємодії є неоднорідною, тому що лінії зв'язку, що поєднують вузли, можуть мати різні характеристики пропускної спроможності.

Залежно від поставленого завдання можна використовувати статичне або динамічне балансування [48, 02]. Статичне балансування виконується до початку виконання завдань. Однак попереднє розміщення логічних процесів за процесорами (серверами) не дає потрібного ефекту. Це пояснюється мінливістю обчислювального середовища (вузол може вийти з ладу), зайнятістю вузла іншими обчисленнями. Так чи інакше, виграш від розподілу логічних процесів за серверами з метою виконання паралельної обробки стає неефективним. Динамічне балансування передбачає розподіл обчислювального навантаження на вузли під час виконання завдань. Програмне забезпечення, що реалізує динамічне балансування, визначає: завантаження обчислювальних вузлів; пропускну спроможність ліній зв'язку; кількість вільної пам'яті; частоту обмінів повідомленнями між логічними процесами завдань та інше. На підставі зібраних даних про завдання та обчислювальне середовище приймається рішення про розподіл завдань між вузлами мережі.

Мета балансування завантаження може бути сформульована таким чином: виходячи з набору завдань, що включають обчислення і передачу даних, та мережі серверів певної топології, знайти такий розподіл завдань за серверами, який забезпечить приблизно рівне обчислювальне завантаження серверів і мінімальні витрати на передачу даних.

Зазвичай практичне і повне рішення задачі балансування завантаження складається з чотирьох етапів [14,48,114]: 1) оцінка завантаження обчислювальних вузлів; 2) ініціація балансування завантаження; 3) прийняття рішень про балансування; 4) розподіл завдань.

Розглянемо більш детально кожен етап балансування.

1. Оцінка завантаження системи.

На цьому етапі здійснюється розрахунок завантаження кожного сервера шляхом розрахунку середнього коефіцієнта використання процесорів, пам’яті, пропускної здатності мережі і-го сервера. Отримана інформація про завантаження використовується для процесу балансування, по-перше, для визначення виникнення дисбалансу, по-друге, для визначення нового розподілу завдань шляхом обчислення обсягу робіт, необхідного для переміщення завдань. Таким чином, якість роботи балансування завантаження безпосередньо залежить від точності і повноти інформації.

2. Ініціація.

Занадто часте виконання балансування навантаження може привести до того, що виконання завдань тільки сповільниться. Витрати на саме балансування можуть перевершити можливу вигоду від його проведення. Отже, для продуктивності балансування необхідно якимось чином визначати момент його ініціалізації. Для цього слід: визначити момент виникнення дисбалансу навантаження; визначити ступінь необхідності балансування шляхом порівняння можливої користі від його проведення та витрат на нього.

Дисбаланс навантаження може визначатися синхронно і асинхронно. При синхронному визначенні дисбалансу всі процесори (сервери мережі) переривають роботу у певні моменти синхронізації і визначають дисбаланс навантаження шляхом порівняння завантаження окремого процесора із загальним середнім завантаженням. При асинхронному визначенні дисбалансу кожен сервер зберігає історію свого завантаження. У цьому випадку момент синхронізації для визначення ступеня дисбалансу відсутній. Обчисленням обсягу дисбалансу займається фоновий процес, що працює паралельно з завданнями.

3. Прийняття рішень у процесі балансування.

Більшість стратегій динамічного балансування завантаження можна віднести до класу централізованих або до класу повністю розподілених.

При централізованій стратегії балансувальник збирає глобальну інформацію про стан всієї обчислювальної системи і приймає рішення про переміщення задач для кожного з серверів.

При повністю розподіленій стратегії на кожному сервері виконується алгоритм балансування навантаження, сервери обмінюються між собою інформацією про їх стан. Переміщення завдань відбувається тільки між сусідніми процесорами.

4. Розподіл завдань.

Існує безліч алгоритмів балансування (розподілу завдань). Однак кожен з алгоритмів має переваги і недоліки [29,49]. Найчастіше використовуваними алгоритмами балансування навантаження є наступні: Round Robin Weight [27,52,79], Connection mechanism [31,49,50,88], Least Connection [49,71] та Compare and Balance [29,31,79] та багато інших. Можна обрати найбільш відповідний у залежності від поставлених завдань та структури комп'ютерної мережі.

При появі нових завдань програмне забезпечення, що реалізує балансування, має прийняти рішення про те, на якому обчислювальному вузлі слід виконувати обчислення, пов'язані з цим новим завданням. Крім того, балансування передбачає перенесення частини обчислень з найбільш завантажених обчислювальних вузлів на менш завантажені вузли. При виконанні завдань процесори обмінюються між собою комунікаційними повідомленнями. У разі низьких витрат на комунікацію, деякі процесори можуть простоювати, у той час, як інші будуть перевантажені. Також будуть недоцільні великі витрати на комунікацію. Отже, стратегія балансування повинна бути такою, щоб обчислювальні вузли були завантажені досить рівномірно, але й комунікаційне середовище не повинно бути перевантаженим [8,27,82,120].

Велика увага у науковій літературі приділяється питанням управління навантаженням в розподілених системах. Теоретичні дослідження і розробка фундаментальних основ розподілу навантаження, створення математичного апарату, моделей та методів управління для розподілу навантаження у розподілених системах розглядалися у роботах вчених [8,11,30,44,68,80,91]. Розроблялися і вдосконалювалися алгоритми балансування навантаження залежні від часу завершення завдання на машині в [22,52], алгоритми грубої сили та алгоритми,

засновані на статистичних даних, розглядалися в роботах [13,26,31, 66], алгоритми, засновані на біологічних феноменах, розглядалися у [20,71], а також багатьма іншими вченими, які працюють над проблемами розподілу навантаження.

1.2 Основні властивості та класифікація алгоритмів балансування навантаження

Алгоритми балансування навантаження можуть бути класифіковані за кількома типами [29, 49]:

1) На основі поточного стану системи.

Статичний алгоритм.

Поточний стан вузла не враховується, потрібна попередня база знань про статистику кожного вузла і призначених для користувача вимогах, не гнучкий, не масштабується, не сумісний з мінливими вимогами користувачів і завантаження. Використовується в однорідному середовищі.

Динамічний алгоритм.

Цей тип алгоритму працює відповідно до динамічних змін стану вузлів, тобто він збирає, зберігає і аналізує інформацію про стан системи. Через це таким алгоритмам властиві великі накладні витрати на балансування. Необхідно враховувати розташування процесора, якому передається навантаження від перевантаженого процесора, оцінку навантаження, обмеження числа міграцій. Якщо який-небудь вузол дав збій, це не зупинить роботу всієї мережі, але вплине на продуктивність системи. Використовується у гетерогенному середовищі.

б) На основі ініціатора алгоритму.

Ініційований відправником: відправник визначає, що кількість вузлів велика, і ініціює виконання алгоритму балансування навантаження.

Ініційований одержувачем: вимоги про балансування навантаження можуть бути визначені одержувачем/сервером у хмарі, і тоді сервер ініціює виконання алгоритму балансування навантаження.

Симетричний: являє собою поєднання типів ініційованих відправником і отримувачем алгоритмів.

Динамічний підхід балансування навантаження поділяється на два типи: розподілений і нерозподілений (централізований) підходи. Вони визначаються наступним чином:

– у нерозподіленому підході один вузол або група вузлів відповідають за управління та розподіл всією системою. Інші вузли не розподіляють завдання і не виконують керуючі функції, отже цей тип алгоритмів не є відмовостійким, у ньому може статися перевантаження центрального процесора прийняття рішень. Корисні в невеликих мережах з низьким навантаженням.

– у розподіленому підході кожен вузол незалежно від інших будує свій вектор навантаження. Всі процесори у мережі відповідальні за розподіл навантаження і наповнення власної локальної бази даних для прийняття ефективних рішень балансування. Це призводить до великих комунікаційних витрат і складності алгоритмів. Корисні у великих і гетерогенних мережах.

При розподіленому підході балансування навантаження може мати дві форми: кооперативну і некооперативну. У кооперативній формі вузли працюють пліч-о-пліч для досягнення спільної мети, наприклад, поліпшення загального часу відгуку. При некооперативному підході балансування вузол працює незалежно від мети, наприклад, поліпшення часу виконання місцевого завдання.

Вузли постійно взаємодіють один з одним і при розподіленому підході генерують більше повідомлень, ніж при нерозподіленому. Передача повідомлень між вузлами для обміну інформацією про оновлення системи може привести до зниження продуктивності системи [44]. Один вузол або група вузлів вирішує завдання балансування навантаження при нерозподіленому підході, він може приймати дві форми: централізованої та напіврозподіленої.

У централізованих алгоритмах один вузол одноосібно відповідає за балансування навантаження всієї системи і називається центральним вузлом. Використовується у невеликих мережах. У напіврозподілених алгоритмах кластер

формується групою вузлів системи, балансування навантаження відбувається у кожному кластері з централізованого типу. Серед вузлів у кластері центральний вузол ініціалізує балансування навантаження у цій групі. Напіврозподілені алгоритми обмінюються великою кількістю повідомлень у порівнянні з централізованими.

Класифікація алгоритмів балансування навантаження надана на рисунку 1.1.

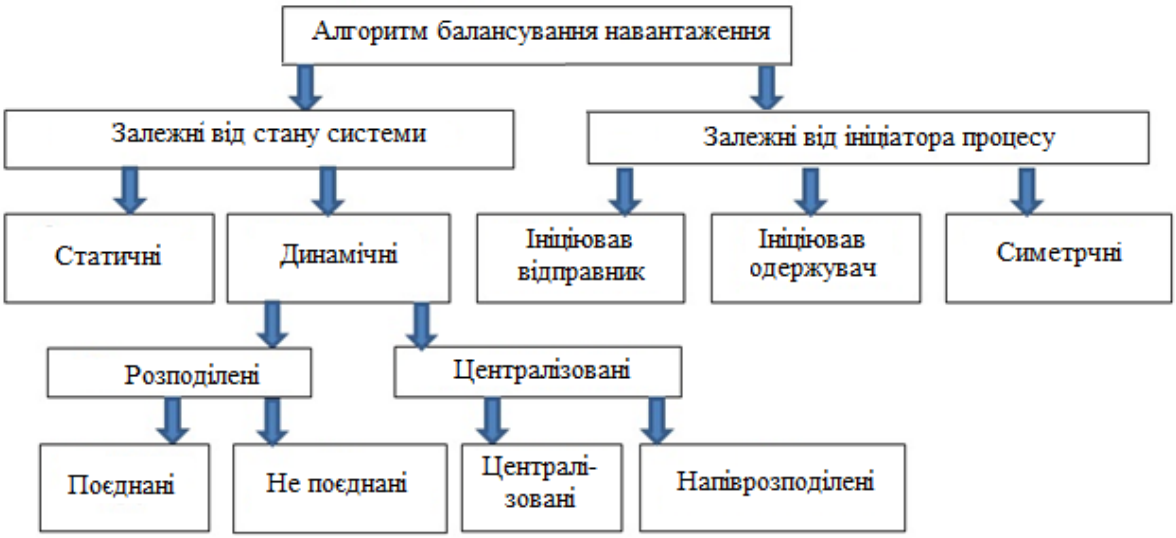


Рисунок 1.1 – Класифікація алгоритмів балансування навантаження

1.3 Показники ефективності алгоритмів балансування

Ефективність алгоритмів балансування навантаження визначається декількома показниками, що представлені нижче [27,78].

1. Пропускна здатність.

Цей показник використовується для оцінки загальної кількості завдань, що успішно завершені. Висока пропускна здатність необхідна для загальної продуктивності системи.

2. Витрати.

Витрати пов’язані з роботою будь-якого алгоритму балансування навантаження і вказують на вартість процесів, що беруть участь у вирішенні задачі, перерозподілі процесів. Вони повинні бути якомога нижче.

3. Відмовостійкість.

Цей показник вимірює здатність алгоритму рівномірно виконувати балансування навантаження у разі будь-якого збою в роботі. Гарний алгоритм балансування навантаження повинен бути дуже нечутливим до несправностей.

4. Час міграції.

Визначається як загальний час переходу завдання від одного вузла або ресурсу до іншого. Він повинен бути зведеним до мінімуму.

5. Час відгуку.

Вимірюється як інтервал часу між відправленням запиту й одержанням відповіді. Він повинен бути зведеним до мінімуму, щоб підвищити загальну продуктивність.

6. Використання ресурсів.

Показник використовується для забезпечення належного використання всіх ресурсів, які включені в систему. Даний показник повинен бути оптимізованим для ефективності алгоритму балансування навантаження.

7. Масштабованість.

Це здатність алгоритму виконувати рівномірне балансування навантаження у системі відповідно до вимог при збільшенні числа вузлів. Кращим є алгоритм з високою масштабованістю.

8. Продуктивність.

Може бути визначена як ефективність системи. Даний показник повинен бути поліпшений при розумних витратах, наприклад, зменшуючи час відгуку та зберігаючи допустимі затримки.

1.4 Основні поняття і властивості інформаційних потоків даних, що мають фрактальні властивості

Численні дослідження процесів в інформаційних мережах показали, що реалізації мережного трафіку має властивість масштабної інваріантності (само-

подібності). Самоподібний трафік має особливу структуру, що зберігається на багатьох масштабах - в реалізації завжди присутня деяка кількість дуже великих викидів при відносно невеликому середньому рівні трафіку. Ці викиди викликають значні затримки та втрати пакетів, навіть коли сумарна потреба всіх потоків далека від максимально допустимих значень.

Самоподібні властивості інформаційних потоків виявлені в багатьох локальних і глобальних телекомунікаційних мережах [1,2,64,86,101,149,151,155, 158]. У зв'язку з вищевикладеним почали активно досліджуватися механізми підвищення якості обслуговування і методів управління трафіком в мультисервісних мережах, що функціонують в умовах самоподібного і мультифрактального трафіку [9,10,96,112,139,142,145,152].

Основні поняття і визначення самоподібних випадкових процесів, стосовно до інформаційних потоків даних, були приведені в роботах [1,17,84,119,157].

Стохастичний процес $X(t),\ t \in \mathbb{R}$ с неперервною змінною часу називається самоподібним з параметром $H,\ 0 < H < 1$, якщо для будь-якого дійсного значення $a > 0$ скінченновимірні розподіли для $X(at),\ t \in \mathbb{R}$ ідентичні скінченновимірним розподілам $a^{-H} X(at),\ t \in \mathbb{R}$, тобто якщо для будь-яких $k \geq 1,\ t_1,\ t_2,\ ...,\ t_k \in \mathbb{R}$ і будь-яких $a > 0$ виконується (1.1):

$$\text{Low}\left\{X(t_2), ...,\ X(t_k)\right\} = \text{Low}\left\{a^{-H}\left(X(at_1),\ a^{-H}X(at_2),\ ...,\ a^{-H}X(at_k)\right)\right\} \quad (1.1)$$

Коротко рівняння (1.1) можна записати у вигляді (1.2):

$$\text{Law}\left\{X(t),\ t \in \mathbb{R}\right\} = \text{Law}\left\{a^{-H}X(at),\ t \in \mathbb{R}\right\}. \quad (1.2)$$

Позначення $\text{Law}\{\cdot\}$ означає скінченномірні закони розподілу випадкового процесу. Параметр H, званий параметром Херста, являє собою міру самоподібності стохастичного процесу.

Початкові моменти самоподібного випадкового процесу можна виразити як (1.3):

$$\mathrm{M}\left[\left|X(t)\right|^q\right] = \mathrm{M}\left[\left|t^H X(1)\right|^q\right] = t^{qH}\,\mathrm{M}\left[\left|X(1)\right|^q\right] = C(q)\cdot t^{qH}, \qquad (1.3)$$

де величина $C(q) = \mathrm{M}\left[\left|X(1)\right|^q\right]$.

Розглянемо поняття самоподібності для процесів з дискретним часом. Пусть $X = (X_1, X_2, ...)$ – відрізок стаціонарного в широкому сенсі випадкового процесу з дискретним часом $t \in N = \{1, 2, ...\}$. Припустимо, процес X має автокореляційну функцію такого вигляду (1.4):

$$r(k) \sim k^{-\beta} L_1(k),\ k \to \infty, \qquad (1.4)$$

де $0 < \beta < 1$ і L_1 – функція, що повільно змінюється на нескінченності, тобто $\lim\limits_{t\to\infty} \dfrac{L_1(tx)}{L_1(t)} = 1$ для усіх $x > 0$.

Позначимо через $X^{(m)} = \{X_1^{(m)}, X_2^{(m)}, ...\}$ усереднений по блокам довжини m процес X, компоненти якого визначаються рівністю (1.5):

$$X_t^{(m)} = \frac{1}{m}(X_{tm-m+1} + ... + X_{tm}),\ m, t \in N. \qquad (1.5)$$

Такий ряд називається агрегованим. Позначимо через $r_m(k)$ і D_m коефіцієнт кореляції і дисперсію процесу $X^{(m)}$ відповідно. Процес X називається строго самоподібним в широкому сенсі з параметром $H = 1 - (\beta / 2), 0 < \beta < 1$, якщо

$r_m(k) = r(k), k \in Z_+,\ m \in \{2, 3, ...\}$, тобто, процес не змінює свій коефіцієнт кореляції після усереднення по блокам довжини m. Іншими словами, X є самоподібним в широкому сенсі, якщо агрегований процес $X^{(m)}$ не відрізняється від

вихідного процесу X, щодо статистичних характеристик другого порядку.

Поняття повільно спадної залежності має ключове значення в теорії самоподібних процесів і описує довгострокову пам'ять процесу. Процес X має повільно спадну залежність, якщо виконується відношення (1.4). Таким чином, процеси з повільно спадною залежністю характеризуються автокореляційною функцією, яка убуває гіперболічно (за степеневим законом) при збільшенні часової затримки. Можна показати, що з (1.4) слідує несумовність автокореляційної функції, тобто $\sum_k r(k) = \infty$. На відміну від процесів з повільно спадною залежністю, процеси з швидко спадною залежністю мають автокореляційну функцією виду

$$r(k) \sim \rho^k, \ k \to \infty, \ 0 < \rho < 1,$$

і, як наслідок, сумовністю автокореляційної функції: $0 < \sum_k r(k) < \infty$.

Довгострокова залежність має на увазі властивість самоподібності в широкому сенсі зі значенням показника $H > 0.5$ і навпаки. Більшість процесів, що мають довгострокову залежність, мають важкі хвости одновимірної функції розподілу ймовірностей. Випадкова величина X має розподіл з важким хвостом, якщо(1.6):

$$P[X > x] \sim c \cdot x^{-\alpha}, \quad x \to \infty, \tag{1.6}$$

де величина α, $0 < a < 2$, називається параметром форми;

c – деяка позитивна константа.

На відміну від розподілів з легкими хвостами, такими як експоненціальне або гауссівське, які мають експоненціальне убування хвоста, випадкові величини з важким хвостом мають хвости, що спадають за степеневим (гіперболічним) законом. При $0 < a < 2$ випадкові величини мають нескінченну дисперсію, а при $0 < a \le 1$ ще і володіють нескінченним середнім. Основна властивість ви-

падкової величини, яка розподілена з важким хвостом, полягає в тому, що вона проявляє високу мінливість. Іншими словами, її вибірка представляє собою в основному відносно невеликі значення, проте також містить і достатню кількість дуже великих значень.

Мультифрактальність – це концепція, яка, з деякими незначними змінами, може бути добре застосована до функцій і мір, детермінованим або стохастичним [97, 143, 158]. На відміну від самоподібних процесів (1.2), мультифрактальні процеси мають більш різноманітну скейлінгову поведінку (1.7) та (1.8):

$$\text{Law}\{X(at)\} = \text{Law}\{M(a)\cdot X(t)\}, \quad a>0, \tag{1.7}$$

$$\text{Law}\{X(at)\} = \text{Law}\{a^{H(a)}\cdot X(t)\}, \tag{1.8}$$

де $M(a)$ – незалежна від $X(t)$ випадкова функція. У разі самоподібного процесу $M(a)=a^H$. З цих властивостей випливає визначальна властивість мультифрактальних процесів [1]: процес $X(t)$ є мультифрактальним, якщо виконується таке відношення(1.9):

$$\text{M}\left[|X(t)|^q\right] = c(q)\cdot t^{qh(q)}, \forall t\in \text{T}, \ \forall q\in\mathbb{Q}, \tag{1.9}$$

де $c(q)$ – деяка детермінована функція.

1.5 Методи оцінювання фрактальних характеристик

Існує безліч методів оцінювання параметрів самоподібних і мультифрактальних процесів. При оцінюванні показника Херста на практиці найбільш часто використовуються методи нормованого розмаху, зміни дисперсії ряду, флуктуаційного аналізу [15,141,156]. При оцінюванні мультифрактальних характеристик найбільш затребуваними є методи мультифрактального детрендірованного флуктуаційного аналізу і метод максимумів модулів безперервного вейв-

лет-перетворення [46, 141,158]. Метод детрендірованного флуктуаційного аналізу (ДФА) є одним з найбільш затребуваних на практиці, оскільки досить точний, та його можливо використовувати як для стаціонарних, так і для нестаціонарних часових рядів [45,46,141].

У методі ДФА для вихідного часового ряду $x(t)$ будується кумулятивний ряд $y(t) = \sum_{i=1}^{t} x(i)$, який розбивається на N сегментів довжиною τ, і для кожного сегмента $y(t)$ обчислюється флуктуаційна функція (1.10):

$$F^2(\tau) = \frac{1}{\tau} \sum_{t=1}^{\tau} (y(t) - Y_m(t))^2 , \qquad (1.10)$$

де $Y_m(t)$ – локальний m-поліноміальний тренд у межах даного сегмента.

Функція $F(\tau)$, усереднена по всьому ряду $y(t)$, має скейлінгову залежність від довжини сегмента ряду: $F(\tau) \propto \tau^H$. Графік залежності $\log F(\tau)$ від $\log \tau$ в певному діапазоні значень буде являти собою пряму лінію, що апроксимовано методом найменших квадратів. Оцінка показника H обчислюється як тангенс кута нахилу прямої залежності $\log F(\tau)$ от $\log(\tau)$.

При дослідженні властивостей мультифрактальних процесів застосовується мультифрактальний флуктуаційний аналіз (МФДФА) [45,144]. При проведенні МФДФА досліджується залежність флуктуаційної функції $F_q(s)$ від параметра q (1.11):

$$F_q(s) = \left\{ \frac{1}{N} \sum_{i=1}^{N} [F^2(s)]^{\frac{q}{2}} \right\}^{\frac{1}{q}} , \qquad (1.11)$$

отриманої зведенням виразу (1.10) в ступінь q і подальшим усередненням за всіма сегментами.

Змінюючи часову шкалу s при фіксованому показнику q, знаходимо за-

лежність $F_q(s)$, представляючи її в подвійних логарифмічних координатах. Якщо досліджуваний ряд зводиться до мультифрактальної множині, яка виявляє довгострокові залежності, то флуктуаційна функція $F_q(s)$ представляється степеневою залежністю $F_q(s) \propto s^{h(q)}$ з функцією узагальненого показника Херста $h(q)$. При $q = 2$ цей показник зводиться до звичайного значення H. Для часових рядів, які відповідають монофрактальній множині, флуктуаційна функція $F_q(s)$ однакова для всіх сегментів, і узагальнений показник Херста $h(q) = H$ не залежить від параметра q.

1.6 Деякі класи моделей фрактального трафіка

Моделі, засновані на фрактальному броунівському русі. Цей клас моделей [42,73, 155] розглядає агрегований трафік як випадковий процес дискретного часу $\{X_t\}$, значеннями X_t якого є число або сумарний обсяг пакетів, що надійшли від джерела на t-м одиничному інтервалі часу. Основою таких моделей є фрактальний гауссівський шум.

Гауссівський процес $X(t)$ називається фрактальним броунівським рухом (ФБР) з параметром $H,\ 0 < H < 1$, якщо прирости випадкового процесу $\Delta X(\tau) = X(t+\tau) - X(t)$ мають розподіл виду

$$P(\Delta X < x) = \frac{1}{\sqrt{2\pi}\sigma_0 \tau^H} \cdot \int_{-\infty}^{x} \mathrm{Exp}\left[-\frac{z^2}{2\sigma_0{}^2 \tau^{2H}} \right] dz,$$

де σ_0 – коефіцієнт дифузії.

ФБР з параметром $H = 0{,}5$ збігається з класичним броунівським рухом. Прирости ФБР називаються фрактальним гауссовским шумом

(ФГШ), дисперсія якого підпорядковується співвідношенню $D[X(t+\tau)-X(t)]=\sigma_0{}^2\tau^{2H}$.

Основною моделлю, яка використовується більшістю дослідників з невеликими модифікаціями, є фрактальний броунівський трафік, запропонований в [73]. Фрактальний броунівський трафік визначається як процес виду

$$A_t = mt + \sqrt{am}Z_{t/t_u},\ t\in(-\infty,\infty),$$

де t являє собою фізичний час;

t_u – одиниця часу;

m и a – позитивні параметри;

Z_t є фрактальним броунівським рухом.

Фрактальный броунівський трафік A_t має три параметри: m (кбіт/сек) – середня інтенсивність процесу, a – коефіцієнт дисперсії з розмірністю біт/сек, H – показник Херста.

Модель, заснована на експоненціальному перетворенні ФГШ. У роботі [122] запропонований підхід, який базується на функціональному перетворенні ФГШ:

$$Y(\tau)=b\cdot\mathrm{Exp}[k\cdot X(\tau)],$$

де $X(\tau)$ – ряд приростів ФБР із заданим параметром H на інтервалі часу τ;

b, k – параметри, що регулюють частоту і величину сплесків даних.

Дане перетворення зберігає довгострокову залежність випадкового процесу $X(\tau)$ і переводить його в самоподібний процес з важкими хвостами. Так як прирости ФБР мають $X(\tau)$ нормальний розподіл, то $Y(\tau)$ має логарифмічно нормальний розподіл.

Визначення параметрів $b,\ k$ модельного процесу за реалізацією реального трафіка здійснюється таким способом:

$$b = \bar{Y} / \sqrt{(F + \bar{Y}) / \bar{Y}}, \qquad k = \sqrt{\ln\left((F + \bar{Y}) / \bar{Y}\right)} / \sigma(\tau),$$

де $\bar{Y} = \frac{1}{\tau}\sum_{t=1}^{\tau} Y_t$ – середнє значення реалізації;

$\tilde{F} = \frac{S^2}{\bar{Y}}$ – відношення середнього квадратичного відхилення до середнього значення;

$\sigma(\tau)$ – середнє відхилення ФГШ.

Мультифрактальні моделі. Використання мультифракталів для моделювання телекомунікаційного трафіку досить ново, і в даний час на практиці використовуються лише кілька результатів. У роботах [16, 83, 158] розглянуті алгоритми моделювання мультифрактального трафіку за допомогою вейвлетів на основі методу зворотного дискретного вейвлет-перетворення. Він полягає у формуванні за допомогою масштабних вейвлет-коефіцієнтів дискретного часового ряду, використовуючи функції деталізації різного масштабу.

До числа найбільш простих способів опису і моделювання мультифрактального випадкового процесу можна віднести моделі, засновані на властивостях бінарних мультиплікативних мультифрактальних каскадів, які були вперше запропоновано як мультифрактальні моделі для трафіку [84]. Цей клас моделей до сих пір є найпопулярнішим для мультифрактального трафіків. [53, 97, 158]

1.7 Огляд досліджень, присвячених вивченню впливу самоподібних властивостей трафіку на якість обслуговування мережі

Фрактальний трафік має особливу структуру, що зберігається на багатьох масштабах – в реалізації завжди присутня деяка кількість дуже великих викидів

при відносно невеликому середньому рівні трафіку. Ці викиди викликають значні затримки і втрати пакетів, навіть коли сумарна потреба всіх потоків далека від максимально допустимих значень.

Більшість використовуваних мереж застосовує пакетну передачу даних, при якій дані передаються за допомогою зберігання і відсилання інформації від вузла до вузла, від буфера до буферу. Для кожного буфера існує окремий алгоритм обробки черги, який вирішує, чи повинен буде пакет зберігатися в пам'яті, бути відправленим або втраченим. У класичному випадку для пуассонівського вхідного потоку буде достатньо буферів помірного розміру: черга може утворитися в короткостроковій перспективі, але за довгий період часу буфери очистяться. Однак при самоподібном навантаженні утворюються черги набагато більшої довжини.

Загальних аналітичних результатів вивчення впливу самоподібності і довготривалої залежності трафіку на якість обслуговування сервісів (QoS) в даний час не існує [158]. Відомі лише окремі аналітичні результати для окремих випадків, наприклад [153].

Найбільш ефективним способом оцінки роботи телекомунікаційних мереж є методи імітаційного моделювання. Саме з цих позицій були розглянуті питання впливу ступеня самоподібності трафіку на ефективність телекомунікаційних систем в роботах [145,149,156,157]. У цих роботах основна увага приділена оцінці впливу самоподібності на побудову черг і ймовірності втрати пакетів. Вплив мультифрактальних властивостей трафіку на побудову черг і втрату даних вивчено в набагато меншому ступені [16,83, 152,158].

Одночасно з вище перерахованим в останні роки почали активно досліджуватися методи управління фрактальним трафіком для поліпшення якості обслуговування мережі. В роботі [152] проведено дослідження фрактальних властивостей трафіка реального часу і оцінка впливу моно- і мультифрактального трафіку на характеристики телекомунікаційної системи з метою забезпечення заданої якості обслуговування QoS. У дисертаційній роботі [112] досліджено вплив методів маршрутизації на якість обслуговування в мультисервіс-

них мережах зв'язку, що функціонують в екстремальних умовах при самоподібному характеру трафіку. В роботі [116] проведено аналіз характеристик трафіку в мережах Metro Ethernet і дослідження протоколів побудови маршрутів для самоподібного трафіку. В роботі [115] досліджені алгоритми управління ресурсами високошвидкісних бездротових мереж в умовах самоподібного трафіку. В роботі [142] запропонований метод адаптивної маршрутизації, що дозволяє зменшити затримку пакетів в мережі на основі прогнозу інтенсивності самоподібного трафіка.

У роботах [12,18,19,61,67] проведені дослідження впливу ступеня самоподібності трафіку на збіжність мережі при управлінні трафіком з різним QoS. У роботах [3,5,9,10,19] розглянуті залежності характеристик QoS, такі як затримки, джиттер і втрати пакетів від параметрів самоподібності вхідного трафіка. Показано, що основний вплив на погіршення характеристик QoS надають великі значення показника Херста, як ступеня самоподібності і коефіцієнта дисперсії, що визначає рівень сплесків трафіка.

На сьогоднішній час розглядається кілька підходів, спрямованих на зменшення впливу самоподібності потоків в інфокомунікаційних мережах. У роботах [21,62,99] пропонуються підходи до скорочення кількості запасної пропускної здатності і загальної вартості мережі. При визначенні шляху для передачі трафіка і необхідної пропускної здатності враховуються параметри самоподібного трафіка. Необхідна пропускна здатність каналу змінюється в залежності від значень показника Херста і коефіцієнта дисперсії, і кількість зарезервованої пропускної здатності каналів зменшується

1.8 Постановка задач дослідження

Аналіз останніх публікацій, представлений вище, показав, що фрактальні властивості мережевого трафіка істотно впливають на характеристики якості

обслуговування сервісів інфокомунікаційних систем. Оскільки загальних аналітичних результатів вивчення впливу самоподібності на якість обслуговування на сьогодні не розроблено, для розрахунку пропускної здатності мережі необхідно використовувати імітаційне моделювання. У переважній більшості робіт для поліпшення якості обслуговування мережі запропоновані методи управління трафіком, що має монофрактальні властивості, в той час як вплив трафіка, що має мультифрактальні властивості, залишається слабо вивченим.

Метою роботи є розробка модифікованих методів балансування навантаження в розподілених системах, які враховують фрактальні властивості трафіку і дозволяють забезпечити високий рівень якості обслуговування.

Відповідно до поставленої мети необхідно вирішити такі наукові завдання:

– провести огляд і аналіз методів балансування навантаження в розподілених системах і досліджень впливу фрактальних властивостей трафіка на якість обслуговування в мережах;

– удосконалити метод розрахунку дисбалансу завантаження ресурсів розподіленої системи для різних класів обслуговування вхідного трафіка, який включає в себе комплексне вимірювання загального рівня дисбалансу системи;

– за допомогою імітаційного моделювання дослідити вплив мультифрактальних властивостей трафіка на показники якості обслуговування в мережі для найбільш затребуваних алгоритмів балансування навантаження;

– удосконалити модель балансування самоподібного навантаження розподіленої системи, з урахуванням мультифрактального характеристик трафіка в постійно мінливих умовах;

– на основі удосконаленої моделі балансування розробити метод балансування, який враховує мультифрактального властивості трафіка і вдосконалений метод розрахунку дисбалансу ресурсів розподіленої системи.

– провести імітаційне моделювання запропонованих методів балансування навантаження і вирішити практичні завдання.

1.9 Висновки з розділу 1

1. Численні дослідження комп’ютерних мереж свідчать, що більшість потоків даних мають самоподібні та мультифрактальні властивості. Ці властивості призвели до появи нових методів управління потоками даних і ресурсами комп’ютерних мереж, що враховують вплив фрактальних властивостей трафіку на якість обслуговування у мережі.

2. Показано, що на даний час актуальним є вивчення впливу методів балансування навантаження на якість обслуговування у комп’ютерних мережах зв’язку, що функціонують в умовах самоподібного трафіку. Особливу увагу слід приділити аналізу стану вузлів у комп’ютерній мережі та розробці методів управління потоками даних і ресурсами мультисервісних комп’ютерних мереж, що враховують самоподібні і мультифрактальні властивості трафіку

3. Показано, що оцінка завантаження обчислювальних вузлів використовується в процесі балансування для визначення виникнення дисбалансу і нового розподілу завдань шляхом обчислення обсягу робіт, необхідного для переміщення завдань. Для розподілу завдань використовуються алгоритми балансування, вибір яких здійснюється за показниками їх ефективності. Розробка методів балансування фрактальних інформаційних потоків, які дозволяють забезпечити високий рівень якості обслуговування в комп’ютерних мережах, є актуальним завданням.

4. Основні результати цього розділу опубліковані в роботах [36, 38, 41, 118, 123].

Список використаних джерел у даному розділі наведено у повному списку використаних джерел під номерами: [1- 3, 5, 8-22, 26-31, 42, 44-50, 52, 53, 60-62, 64, 66-68, 71, 73, 76, 78-80, 82-84, 86-88, 91, 96, 97, 99-101, 112, 114-116, 119, 120, 122, 139, 141-145, 149-153, 155-158].

РОЗДІЛ 2
РОЗРОБКА МЕТОДУ РОЗРАХУНКУ ДИСБАЛАНСУ РЕСУРСІВ РОЗПОДІЛЕНОЇ СИСТЕМИ ДЛЯ ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ОБСЛУГОВУВАННЯ МЕРЕЖІ

Еволюційні процеси, що відбуваються в мережах зв’язку, неминуче впливають на обсяг, а також на внутрішню структуру трафіку. Згідно з численними дослідженнями сумарний обсяг даних, які передаються в глобальній мережі, показує стійке експоненціальне зростання, при тому, що дана тенденція збережеться в найближчі роки.

Із збільшенням обсягів даних, поведінка трафіку сучасної глобальної мережі виявляє таку негативну особливість, як нестабільність навантаження, яка характеризується можливістю появи непередбачуваних стрибків інтенсивності передачі. Існує безліч причин, що викликають подібну нестабільність. Яскравим прикладом може бути поведінка в мережі безлічі користувачів, викликана вірусним характером поширення популярної інформації. Лавиноподібний процес залучення нових користувачів, нові способи колективної комунікації, що базуються на широкому застосуванні соціальних сервісів, масові онлайн-трансляції, - все це змушує говорити про новий характер виникаючих навантажень.

Дослідники відзначають, що сучасні мережі страждають від нестачі пропускної здатності [58]. Відповідно до проведених досліджень близько 20-30% з’єднань глобальної мережі проходять в перевантажених ділянках. При цьому відзначається значна нерівномірність розподілу завантаження канальних ресурсів, що свідчить про неефективність застосовуваних механізмів управління трафіком в нинішніх умовах [23,24]. Виходом зі сформованої ситуації є застосування спеціальних методів балансування трафіку, що дозволяють ефективно розподіляти навантаження відповідно до наявних незадіяних ресурсів.

У сьогоднішній час разом із планомірним збільшенням швидкостей

передачі даних у телекомунікаціях збільшується частка інтерактивного трафіку, вкрай чутливого до параметрів середовища транспортування. Завдання забезпечення QoS стає все більш актуальним і для його вирішення у мережі повинні бути реалізовані механізми, що гарантують потрібну якість обслуговування.

2.1 Основні механізми забезпечення якості обслуговування

У методах забезпечення якості обслуговування використовуються різні механізми, спрямовані на зниження негативних наслідків перебування пакетів в чергах із збереженням у той же час позитивної ролі черг. Набір механізмів досить широкий [74,103,113]. Більшість з них враховує і використовує у своїй роботі факт існування в мережі трафіку різного типу у тому відношенні, що кожен тип трафіку пред’являє різні вимоги до характеристик продуктивності і надійності мережі. Наприклад, трафік перегляду веб-сторінок мало чутливий до затримок пакетів і не вимагає гарантованої пропускної здатності мережі, зате чутливий до втрат пакетів. У той же час голосовий трафік дуже чутливий до затримок пакетів, вимагає гарантованої пропускної здатності мережі, але може «терпіти» втрату невеликого відсотка пакетів без значної шкоди для якості (залежить від використовуваного методу кодування голосового сигналу).

Досягнути одночасного дотримання всіх характеристик QoS для всіх видів трафіку досить складно. Одним з найбільш значущих чинників, які впливають на характеристики якості обслуговування, є рівень завантаження мережі трафіком, тобто, рівень використання пропускної здатності ліній зв’язку мережі [7,27,82, 120].

Якщо цей рівень постійно є досить низьким, то трафік усіх додатків обслуговується з високою якістю протягом більшої частини часу (хоча короткочасні перевантаження мережі, що призводять до затримок і втрат пакетів, все одно можливі, але вони трапляються дуже рідко). Такий стан мережі називається «недовантаженим» або ж використовується термін мережі з

надлишковою пропускною здатністю (англ. – overprovisioning).

Постійно підтримувати усі частини мережі у недовантаженому стані досить дорого і складно, але для найбільш відповідальної частини мережі, такої як магістраль, цей підхід застосовується і пов’язаний він з постійним спостереженням за рівнем завантаження каналів магістралі та періодичним збільшенням їх пропускної здатності в міру наближення завантаження до критичного рівня.

Методи QoS засновані на іншому підході, а саме тонкому перерозподілі наявної пропускної спроможності між трафіком різного типу відповідно до вимог додатків. Очевидно, що ці методи ускладнюють мережеві пристрої, тому що означають необхідність знання вимог усіх класів трафіку, вміння їх класифікувати і розподіляти пропускну здатність мережі між ними. Остання властивість зазвичай досягається за рахунок використання декількох черг пакетів для кожного вихідного інтерфейсу комунікаційного пристрою замість однієї черги. При цьому в чергах застосовують різні алгоритми обслуговування пакетів, чим і досягається диференційоване обслуговування трафіку різних класів. Саме тому методи QoS часто асоціюються з технікою управління чергами.

Крім власне техніки організації черг, до методів QoS відносять методи контролю параметрів потоку трафіку, тому що для гарантовано якісного обслуговування потрібно бути впевненими, що обслуговувані потоки відповідають певному профілю. Ця група методів QoS отримала назву методів кондиціонування трафіку.

Особливе місце займають методи зворотного зв’язку, що призначені для повідомлення джерела трафіку про перевантаження мережі. Ці методи розраховані на те, що при отриманні повідомлення джерело знизить швидкість видачі пакетів в мережу і тим самим ліквідує причину перевантаження.

Механізми QoS можна застосовувати по-різному. У тому випадку, коли вони застосовуються до окремих вузлів без урахування реальних маршрутів слідування потоків трафіку через мережу, умови обслуговування трафіку цими

вузлами поліпшуються, але гарантій того, що потік буде обслуговано з заданим рівнем якості, такий підхід не дає. Гарантії можна забезпечити, якщо застосовувати методи QoS системно, резервуючи ресурси мережі для потоку протягом усього його маршруту, іншими словами, «від краю до краю».

Методи інжинірингу трафіка стоять поряд із методами QoS. Відповідно до методів інжинірингу трафіку, маршрути передачі даних управляються таким чином, щоб забезпечити збалансоване навантаження всіх ресурсів мережі і виключити за рахунок цього перевантаження комунікаційних пристроїв і утворення довгих черг. На відміну від методів QoS, методи інжинірингу трафіку використовують організацію черг з різними алгоритмами обслуговування на мережевих пристроях. У той же час в методах QoS в їх традиційному розумінні не використовують такий потужний важіль впливу на раціональний розподіл пропускної здатності за зміну маршрутів трафіка в залежності від фактичного завантаження ліній зв’язку, що дозволяє легко відокремити методи QoS від методів інжинірингу трафіку.

У наступній групі методів боротьба з перевантаженнями ведеться шляхом зниження постійного навантаження на мережу. Тобто, в цих методах проблема розглядається з іншого боку: якщо пропускної здатності мережі недостатньо для якісної передачі трафіку додатків, то вирішується питання можливості зменшення обсягу самого трафіку. Найбільш очевидним способом зниження обсягу трафіку є його компресія; існують й інші способи, що призводять до того ж результату, наприклад, розміщення джерела даних ближче до його споживача (кешування даних).

Існують такі основні моделі забезпечення QoS:

– інтегрований сервіс – Integrated Service (IntServ);

– диференційоване обслуговування – Differentiated Service (DiffServ);

Інтегрований сервіс.

Модель інтегрованого обслуговування забезпечує скрізну якість обслуговування, гарантуючи необхідну пропускну здатність. IntServ використовує для своїх цілей протокол резервування мережевих ресурсів RSVP, який забезпечує

виконання вимог до усіх проміжних вузлів. По відношенню до IntServ часто використовується термін «резервування ресурсів» (Resource reservation).

Протокол RSVP. Даний протокол дозволяє додаткам відправляти сигнали у мережу про свої QoS- вимоги для кожного потоку. Для визначення кількісних характеристик цих вимог з метою управління доступом використовуються службові параметри.

Протокол RSVP застосовується у додатках з груповою розсилкою, таких як додатки аудіо- та відео конференцій. Незаважаючи на те, що початково протокол RSVP був орієнтованим на мультимедійний трафік, за його допомогою легко можна резервувати смугу пропускання для односпрямованого трафіку, наприклад, для трафіку мережевої файлової системи (Network File System – NFS) та керуючого трафіку віртуальних приватних мереж (Virtual Private Networks – VPN).

Протокол RSVP сигналізує про запити резервування ресурсів за доступним маршрутизованим шляхом у мережі. При цьому RSVP не виробляє власної маршрутизації, навпаки, цей протокол був розроблений для використання інших, більше потужних протоколів маршрутизації. Як і будь-який інший IP-трафік, при визначення шляху для даних та керуючого трафіку, RSVP засновується на застосовуваному у мережі протоколі маршрутизації.

Диференційоване обслуговування.

Забезпечує QoS на основі розподілу ресурсів у ядрі мережі та окремих класифікаторів і обмежень на кордоні мережі, що комбінуються з метою надання необхідних послуг. У цій моделі вводиться розподіл трафіка за класами, для кожного з яких визначається свій рівень QoS. DiffServ складається з управління формуванням трафіку (класифікація пакетів, маркування, управління інтенсивністю) та керування політикою (розподіл ресурсів, політика відкидання пакетів). DiffServ є найбільш відповідним прикладом «розумного» управління пріоритетом трафіку.

Головним завданням підходу diffserv є визначення стандартизованого байта диференційованої послуги (DS) – байта типа обслуговування (Type

of Service – ToS) із заголовка пакета IPv4 та байта класу трафіка (Traffic Class) пакета IPv6. Від даного маркування залежить прийняття рішення про просування пакету даних на кожному переході (per-hop behavior – PHB), тобто у кожному проміжному вузлі.

Архітектура диференційованих послуг забезпечує базову основу, що може бути використана постачальниками послуг для подання своїм клієнтам великого діапазону різноманітних додатків у залежності від пред'явлених вимог до якості обслуговування. Клієнт може обрати необхідний рівень послуг шляхом встановлення відповідного значення поля кода диференційованої послуги (Differentiated Services Code Point – DSCP) для пакетів визначеного додатку. Код диференційованої послуги визначає ланцюжок рішень про просування пакету у кожному проміжному вузлі мережі постачальника послуг (PHB-политика).

PHB-політика – політика покрокового обслуговування – визначає поведінку мережевого вузла по відношенню пакетів з чітким визначенням поля кода диференційованої послуги (DSCP). Усі пакети потоку трафіку зі специфічною вимогою до обслуговування несуть у собі одне й те ж саме значення поля DSCP.

Усі вузли всередині diffserv-домену визначають PHB-політику, що повинна бути застосована до пакету на базі збереженого у ньому значення поля коду диференційованої послуги. Окрім того, прикордонні вузли diffserv-домену виконують важливу функцію формування трафіка, що надходить до diffserv-домену. Формування трафіку включає в себе виконання таких функцій, як:

– класифікація пакетів (встановлення значення поля DSCP);

– обмеження трафіка.

Формування трафіку зазвичай виноситься на вхідний інтерфейс пакетів, що потрапляють у diffserv-домен. Формування грає вирішальну роль у клерувальні вхідним трафіком у diffserv-домені, оскільки у цьому випадку для кожного пакету мережа може визначити відповідну йому PHB-політику.

QoS являє собою низку технологій, що дозволяють додаткам запитувати

та отримувати прогнозований рівень послуг з точки зору пропускної здатності, часового розкиду затримки відгуку, а також загальної затримки доставки даних [74, 82,103,113,114,139]. Зокрема, QoS передбачає покращення параметрів або досягнення більшої передбачливості послуг, що надаються. Це досягається наступними методами: підтримкою певної смуги пропускання; скороченням вірогідності втрати кадрів; виключенням мережевих перевантажень або контролем над ними; можливістю конфігурування мережевого трафіку; встановленням кількісних характеристик трафіку на його шляху через мережу.

Управління перевантаженням може здійснюватися шляхом зміни порядку, в якому відправляються пакети відповідно до приписаного їм пріоритету. QoS-управління перевантаженням має чотири модифікації протоколів управління чергами, кожен з яких дозволяє організовувати різну кількість черг. На другому рівні моделі OSI (L2) QoS припускає таке.

1. Управління вхідними чергами. Коли кадр приходить на вхід порта, він може бути віднесеним до однієї з декількох черг, що асоціюються з портом, перед тим, ніж він буде направленим на один з вихідних портів. Як правило, декілька черг застосовуються тоді, коли різноманітні інформаційні потоки потребують різних рівней послуг або мінімізації затримки. Наприклад, IP-мультімедіа потребує мінімізації затримки, на відміну від передачі даних у FTP, WWW, email, Telnet тощо.

2. Класифікация. Процес класифікації включає перегляд різних полів у заголовку Ethernet L2, а також полів IP-заголовка (L3) та заголовків TCP/UDP (L4), щоб забезпечити певний рівень послуг при комутації пакетів.

3. Політика. Здійснення політики є процесом аналізу кадра Ethernet, щоб визначити, чи не буде перевищеним заданий рівень трафіку за певний інтервал часу (зазвичай цей час є внутрішнім параметром перемикача). Якщо кадр створює ситуацію, за якої трафік перевищить заданий рівень, він буде відкинутий або значення CoS (Class of Service) може бути зниженим.

4. Перезапис. Процес перезапису надає можливість перемикачу модифікувати CoS або ToS (Type of Service) у IPv4-заголовку. Слід

враховувати, що заголовок Ethernet 802.3 поля CoS не має (саме ца версія стандарту є найбільш поширеною в Україні).

5. Управління вихідними чергами. Після процесу перезапису перемикач помістить кадр Ethernet у вихідну чергу для подальшої комутації. Перемикач виконає управління буфером так, щоб не виникло переповнення. Це зазвичай відбувається за допомогою алгоритма RED (Random Early Discard), коли деякі кадри випадковим чином видаляються з черги. Weighted RED (WRED) є директивою RED, де значення CoS аналізуються, щоб визначити, які кадри слід відкинути. Коли буфери стануть заповненими до відповідного рівня, кадри з низьким рівнем пріоритету відкидаються, а у черзі зберігаються тільки високо пріоритетні кадри.

2.2 Балансування навантаження як метод забезпечення QoS

Для оптимізації використання ресурсів, скорочення часу обслуговування запитів, горизонтального масштабування (динамічне додавання/видалення пристроїв), а також забезпечення відмовостійкості (резервування) застосовується метод рівномірного розподілу завдань між декількома мережевими пристроями (наприклад, серверами), що називається балансуванням навантажень або вирівнюванням навантажень (англ.: Load Balancing) [43,76,95,98].

При появі нових завдань програмне забезпечення, що реалізує балансування, повинно прийняти рішення про те, на якому обчислювальному вузлі слід виконувати обчислення, пов’язані з цим новим завданням. Окрім того, балансування припускає перенесення (migration – міграція) частини обчислень з найбільш завантажених обчислювальних вузлів на менш завантажені вузли. При виконанні задач процесори обмінюються між собою комунікаційними повідомленнями. У випадку низьких витрат на комунікацію, деякі процесори (комп’ютери) можуть простоювати, у той час як інші будуть перевантажені. Також недоцільними будуть великі витрати на комунікацію. Таким чином,

стратегія балансування повинна бути такою, щоб обчислювальні вузли були навантажені достатньо рівномірно, але й комунікаційне середовище не повинно бути перевантаженим.

Проблема балансування обчислювального навантаження розподіленого додатку виникає через наступні причини:

– структура розподіленого додатку неоднорідна, різноманітні логічні процеси потребують різноманітних обчислювальних потужностей;

– структура обчислювального комплексу (наприклад, кластеру), також неоднорідна, тобто різні обчислювальні вузли володіють різною продуктивністю;

– структура міжвузлової взаємодії неоднорідна, тому що лінії зв’язку, що поєднують вузли, можуть мати різноманітні характеристики пропускної здатності.

2.2.1 Класифікація стратегій балансування навантаження

Існує базова класифікація стратегій балансування навантаження, що враховує автоматичні стратегії, реалізовані у складі спеціалізованого програмного забезпечення системного або проміжного (*middleware*) рівня, включені у склад прикладних розподілених програмних комплексів. За рядом істотних ознак можна умовно виділити наступні основні класи стратегій [29,49,111]:

1. За принципом обліку динаміки розділяють *статичні*, *напівдинамічні* та *динамічні стратегії*. При статичній стратегії план розподілу навантаження фіксований та відомий завчасно. Напівдинамічна стратегія передбачає, що план визначається на етапі ініціалізації, до початку виконання головних розрахунків. У цьому випадку попередня оцінка існуючих ресурсів покращує показник ефективності обчислень у порівнянні зі статичною стратегією. Динамічною стратегія вважається у тих випадках, коли план розподілу періодично під дією якихось факторів або умов стану середовища, а також завчасно складеному

графіку, перераховується протягом усього часу життя розподілених додатків, а обчислювальні задачі перерозподіляються за вузлами мережі у відповідності до відкоректованого, оптимального на даний момент часу плану.

Динамічне балансування є достатньо складним завданням, у більшості ситуацій його застосування є оправданим і дає значне збільшення продуктивності, але у деяких випадках неправильного використання може привести до зменшення корисного навантаження.

На відміну від статичних та напівдинамічних, динамічні засоби балансування від початку орієнтовані на зміну умови функціонування, тому переваги динамічних стратегій проявляються у повній мірі в системах, де завчасно невідомі деякі параметри функціонування обчислювальних процесів, що пов’язано із структурою і алгоритмом міжпроцесної взаємодії, характером виділення ресурсів в системі тощо. Слід визнати, що статичні стратегії притягують увагу розробників своєю простотою та високою ефективністю на додатках з передбачуваним ходом обчислень, а також за наявності деякої апріорної інформації про обчислення, що планується виконати. Однак, далеко не завжди їх використання є доцільним в умовах динамічно змінюваних середовищ – відгукуються характерні їм обмеження.

З динамічними стратегіями пов’язані поняття міграції ресурсів, а саме міграція процесів та міграція даних. Специфіка переносу навантаження з вузла на інші вузли у гетерогенних системах торкається питань сумісності, портування, інтероперабельності, масштабування.

2. За ступенем пристосуваності до зміни навантажень стратегії поділяються на *адаптивні* та *неадавтивні*. Адаптивні – забезпечують вирівнювання навантаження та перерозподіл ресурсів при зміні ресурсної конфігурації розподіленої системи. Наприклад, при включенні нових вузлів у склад системи або виключенні/відмовах уже працюючих вузлів відбувається перемасштабування обчислень, і, таким чином, автоматична адаптація. Інколи під адаптивним балансуванням навантаження розуміють динамічне балансування, тільки у більш вузькому сенсі цього слова.

3. За способом зміни плану розподілу виділяють стратегії *з залежним* і *незалежним розподілом.* Залежним вважається такий розподіл, за яким відстежуються деякі події, що стосуються зміни характеристик навантаження, або таймерні події, та за їх проявою будується новий план. Під незалежним (за суттю, розподіл, що відноситься одночасно і до напівдинамічного, і до динамічного типу стратегій) розуміється апріорне обчислення послідовності планів до початку роботи розподіленого додатку. За часом його виконання відбувається зміна статичних планів за завчасно складеним сценарієм, що має прив’язку до певних подій системи або цільового обчислювального процесу, таймерних відліків, вже без обчислення саме планів.

4. За принципом управління (характером відповідальності за розподіл) алгоритми балансування можна розподілити на *централізовані* та *децентралізовані*. Тут стратегія визначає, яким саме способом відбувається розподіл ресурсів. Централізовані алгоритми визначаються як глобальні, тобто наявний центральний елемент – планувальник (диспетчер), що для прийняття рішень збирає інформацію з усіх вузлів системи. Зазвичай цей планувальник та необхідність постійної синхронізації відомостей про стан ресурсів у всій системі є вузьким, вразливим місцем стратегії даного типу. Децентралізовані алгоритми не мають потреби у відомостях про навантаження кожного вузла, тому їх визначають як локальні. Розподіл ресурсів планується кожним вузлом окремо, інколи з урахуванням взаємодії з сусідніми вузлами. Децентралізовані схеми є найтиповішими для мережних обчислень у тих випадках, коли структура комунікацій, що пов’язує вузли, добре підходить для конкретних прикладних завдань. Централізовані стратегії балансування найчастіше використовуються через причину універсальності підходу та простоти алгоритмізації, хоча і не завжди дають гарну масштабованість, однак, як було вказано, для деяких прикладних завдань децентралізований підхід стає більш ефективним, незважаючи на його обмежену пристосованість та певні складності при сумісній синхронізації і забезпеченні цілісності усього обчислювального процесу.

5. За ознакою універсальності стратегії поділяють на *універсальні* або *спеціалізовані*. До останніх можна віднести, наприклад, стратегії, розраховані на певну архітектуру розподіленої системи, конкретну топологію мережевого середовища, розроблені під конкретний алгоритм або з урахуванням специфічних властивостей конкретної мережевої інфраструктури. Універсальні стратегії орієнтовані на обслуговування широкого класу алгоритмів, інваріантні до області застосування, і, як правило, володіють легкою вбудованістю у додатки та уніфікованістю інтерфейсу.

6. Запобігання майбутнім змінам стратегії поділяють на *прогностичні* та *без здатності передбачати майбутню зміну станів* (у тому числі навантаження). Очевидним є те, що архітектури, які володіють адекватними засобами короткострокового та довгострокового прогнозування розвитку обчислювального процесу, істотно переважають у плані адаптації серед звичайних систем планування обчислень. Основною метою при розробці прогностичних стратегій є збільшення точності прогнозів, оскільки саме від них залежить отримуваний виграш за продуктивністю за рахунок скорочення передбачених невиробничих витрат. Надлишково складні та обчислювально ємкісні алгоритми розрахунку прогнозів, хоча і є більш точними, можуть не виправдати очікувань стосовно можливого збільшення продуктивності, і, навіть навпаки, можуть сприяти деградації вже досягнутої продуктивності внаслідок занадто великих нецільових витрат доступних ресурсів. Незадовільна точність прогнозів також може негативно впливати на загальну продуктивність системи.

Додамо до класифікації, приведеної вище, типізування стратегій балансування навантаження за деякими важливими другорядними показниками. У розгляд не включено багато ознак, що є несуттєвими для більшості областей застосування (наприклад, виключені класифікації за ступенем гранулярності ресурсів, приорітезації обчислень, прозорості тощо).

7. За обліком причин розбалансування стратегії бувають *з* обліком та *без обліку причин*, що приводять до зміни завантаження, як внутрішніх (відносно розподіленого додатку), так і зовнішніх (не залежних від діяльності

розподіленого додатку). Облік внутрішніх та зовнішніх факторів підвищує прогностичні властивості стратегій.

8. За вибором оцінюваних характеристик можна виділити стратегії, в яких при розподілі *враховується тільки продуктивність вузлів* розподіленої системи, або також *ще продуктивність мережевої (комунікаційної) підсистеми*. В останньому варіанті підлягає аналізу характер інформаційного та керуючого трафіку в мережі.

9. За ступенем точності оцінки кількості вільних ресурсів можна виділити *наближені методи* та достатньо *реалістичні* методи розподілу, що оперують комплексними агрегатними показниками ресурсоспоживання (наприклад, ті, що включають такі характеристики, як число активних потоків виконання у вузлі, кількість та швидкість процесорів вузла, об’єм доступної оперативної та віртуальної пам’яті вузла, міжплатформенні накладні витрати тощо).

10. За ініціатором розподілу розрізняють алгоритми балансування, у яких стороною, що ініцією, виступає приймач навантаження (тобто недовантажені вузли) та алгоритми, в яких такою стороною є джерело навантаження (перевантажені вузли).

2.2.2 Класифікація алгоритмів та методів балансування

На основі поточного стану системи, алгоритми балансування навантаження можуть буть класифіковані за двома типами:

– статичний алгоритм: поточний стан вузла не враховується. Усі вузли та їх властивості відомі від початку. Алгоритм працює на основі цих попередніх даних. Так він не використовує інформацію про поточний стан системи і є легко реалізовуваним;

– динамічний алгоритм. Цей тип алгоритму на основі поточного стану системи. Алгоритм працює у відповідності до динамічних змін стану вузлів, тобто він збирає, зберігає, аналізує інформацію про стан системи. Через це таким алгоритмам властиві великі накладні витрати на балансування у порівнянні

зі статичними алгоритмами. Динамічні алгоритми є складними у реалізації, але вони ефективно балансують навантаження.

На основі ініціатора алгоритму, алгоритми балансування навантаження можна поділити на три типи:

– ініційований відправником: відправник визначає, що кількість вузлів велика та ініціює виконання алгоритму балансування навантаження;

– ініційований отримувачем: вимоги до балансування навантаження можуть бути визначені отримувачем/сервером у хмарі, і тоді сервер ініціює виконання алгоритму балансування навантаження;

– симетричний: полягає у поєднанні типів, ініційованих отримувачем та відправником.

Динамічний підхід балансування навантаження підрозділяється на два типи: розподілений та нерозподілений (централізований) підходи. Вони визначаються наступним чином:

а) у централізованому підході лише один вузол відповідає за управління та розподіл в усій системі. Інші вузли не розподіляють задачі та не виконують керуючих функцій;

б) у розподіленому підході кожен вузол незалежно будує свій вектор навантаження (вектор збору інформації інших вузлів). Усі рішення приймаються на місцевому рівні, використовуючи місцеві вектори навантаження. Розподілений підхід більше підходить для широко розподілених систем, таких як хмарні обчислення.

Зазвичай динамічний алгоритм балансування складається з чотирьох елементів:

– політики балансування (transfer policy);

– алгоритм вибору партнера (location policy);

– алгоритм вибору задачі для передачі (selection policy);

– механізм збору необхідної інформації про стан системи (information policy).

Політика балансування визначає, чи є вузол об’єктом балансуван-

ня [44,51]. Різноманітні види політик балансування використовують порогові значення завантаженості вузлів, відхилення величини навантаження вузла від середнього значення за системою та інші методи. Алгоритм вибору задачі визначає, яку задачу необхідно передати. При виборі завдання для відправки алгоритм може враховувати, що накладні витрати, пов'язані з пересиланням задачі, повинні бути мінімальними, складність виконання задачі повинна бути великою, число зв'язків у задачі з локальними ресурсами повинно бути мінімальним. Алгоритм вибору партнета відповідає за вибір найдоцільнішого вузла для операції балансування. Поширеною технікою у розподілених алгоритмах є опитування (polling) вузлів. Опитування може бути послідовним або паралельним, використовувати результати попередніх опитувань. У централізованих алгоритмах вузол звертається до спеціалізованого координатора для визначення оптимального партнера для балансування. Координатор збирає та підтримує в актуальному стані інформацію про завантаженість вузлів системи, а алгоритм пошуку партнера використовує ці дані.

Механізм збору інформації про завантаженість системи визначає джерело інформації, час збору даних про завантаженість, місце зберігання інформації. Існує декілька класів механізмів збору інформації.

1. Збір даних за необхідністю. У даному класі використовуються розподілені алгоритми, що збирають інформацію про завантаженість, коли вузол потребує балансування навантаження. Вони підрозділяються на ініційовані відправником, отримувачем та симетрично ініційовані. У алгоритмах, що ініціюються відправником, вузол, що передає задачі, шукає отримувачів, яким він може передати частину задач. У алгоритмах, що ініціюються отримувачем, вузол, що отримує задачі, запозичує завдання у відправника. У симетричних алгоритмах використовується комбінація вищевказаних підходів.

2. Періодичний збір даних. Алгоритми даного класу можуть бути як централізованими, так і розподіленими. У залежності від зібраних даних алгоритм ініціює балансування навантаження.

3. Збір даних у випадках зміни стану. У системах, що реалізують алгоритми даного класу, вузли самостійно поширюють інформацію про зміну завантаженості у випадках зміни внутрішнього стану. У ситуації розподілених алгоритмів дані направляються сусіднім вузлам, у випадку централізованих алгоритмів дані направляються координатору. Під стабільністю алгоритма балансування навантаження або системи, що його використовує, зазвичай розуміється одна з двох характеристик:

– системна стійкість, тобто недопустимість ситуації, коли окремі вузли системи перевантажені, у той час як інші простоюють або недовантажені;

– алгоритмічна стабільність, виражена у тому, що алгоритм не здійснює некорисних дій з ненульовою вірогідністю, наприклад, не передає задачу по кругу між вузлами так, що задача не виконається ніколи.

Розглянемо основні методи апаратного та програмного балансування навантаження на різноманітних рівнях мережевої моделі OSI, здійснимо аналіз їх переваг та недоліків.

2.3 Методи балансування навантаження

Балансування навантаження здійснюється за допомогою апаратних, програмних інструментів або комбінації їх обох. Раніше було чітким розмежування на апаратне та програмне балансування навантаження. Зараз, у зв’язку з розвитком та вдосконаленням як апаратних, так і програмних засобів балансування навантаження, межі між ними затираються. [72,85]. Будемо вважати, що якщо використовується комутатор, switch, ADC – Application Delivery Controllers – то це апаратне балансування. При використанні сервера (комп’ютера) будемо вважати, що здійснюється програмне балансування навантаження. Апаратне балансування навантаження часто використовується на канальному, мережевому та транспортному рівнях. Загалом, апаратне балансування навантаження є більш швидким, ніж програмні рішення, але його недоліком є вартість. Балансування навантаження на основі програмного

забезпечення, на відміну від апаратних засобів балансування навантаження, працює на стандартних операційних системах та стандартних апаратних компонентах, таких як ПК. Програмні рішення працюють або у виділеному апаратному вузлі балансування навантаження, або безпосередньо у додатку. У апаратному балансуванні навантаження використовуються пристрої, що називуються Network Packet Broker (або Network Monitoring Switch), що мають 100GbE interfaces та працюють на 2 та 3 рівнях мережевої моделі OSI [63]. NPB – мережевий пристрій, що вбудовується у стойку, отримує та агрегує мережевий трафік з портів, яким у подальшому він маніпулює. Основною і найголовнішою функцією NPB є балансування навантаження. Для балансування навантаження на 3-7 рівнях використовуються контролери доставки додатків (ADC – Application Delivery Controllers) [120].

Процедура балансування навантаження здійснюється за допомогою комплексу алгоритмів та методів, що відповідають наступним рівням мережевої моделі стека мережевих протоколів OSI: канальному, мережевому, транспортному та прикладному [110].

Схематично балансування навантаження у розподілених системах можна зобразити у вигляді структури, що представлена на рисунку 2.1.

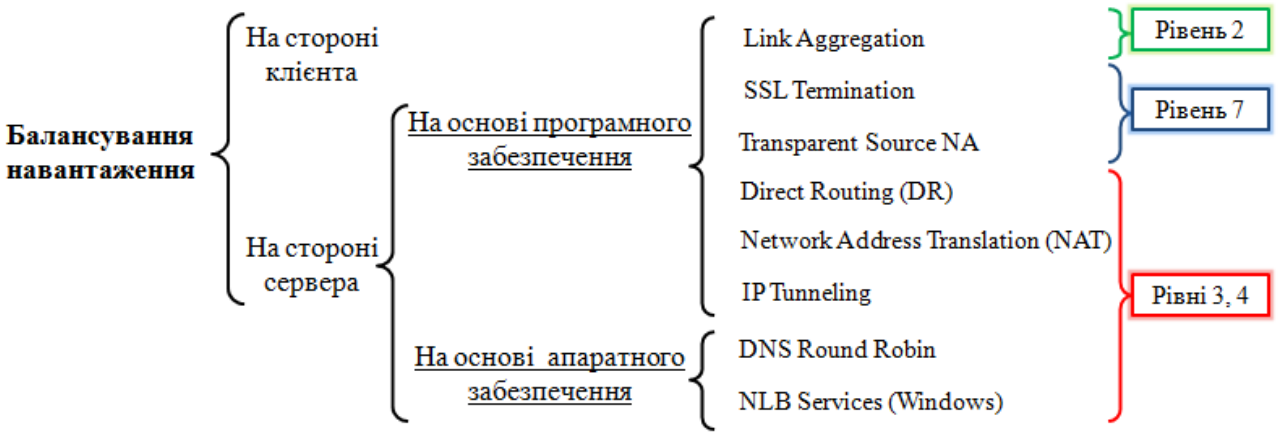


Рисунок 2.1 – Методи балансування навантаження

2.4 Балансування навантаження на стороні клієнта

Балансування навантаження на стороні клієнта, як правило, менш якісне, ніж балансування на сервері [7,59,75]. Причина полягає у тому, що клієнти часто не можуть відстежувати доступність серверу або рівень навантаження. Якщо сервер є перевантаженим або недоступним, клієнт чекає деякий час, перш, ніж зробить спробу підключитися до ще одного сервера. Поширення інформації про навантаження серверу до клієнта створює додаткове навантаження на мережу і затримка поширення інформації додається до загального часу обслуговування.

Доступність серверу також є дуже динамічною, тому клієнт не може використовувати цю інформацію протягом достатньо довгого періоду часу. Окрім цього, балансувальник може вмикати вартість запиту доступності серверу для багатьох запитів. Таким чином, балансування навантаження на стороні серверу не викликає штрафу за затримку у клієнта. Балансування завантаження на стороні клієнта, як правило, використовує більше пропускної здатності, ніж балансування серверного навантаження. Це відбувається тому, що мережевий шлях для кожного шляху клієнт-сервер потенційно може приймати різноманітні маршрути.

Користувач безпосередньо підключається до серверу без будь-якого балансування. Якщо по якимось причинам сервер не відповість, то користувач не зможе отримати доступу до необхідних ресурсів. Також, якщо дуже велика кількість користувачів звернулось до сервера, то багато з них отримають відповідь з великою часовою затримкою, а деякі так і не зможуть її дочекатися. Інший варіант балансування навантаження на стороні клієнта: можна вставляти список серверів додатків у код клієнта. Тобто на стороні клієнта існує список доступних серверів, до яких клієнт намагається звертатися до тих пір, поки не знайде той, який відповідає.

2.5 Балансування навантаження на основі серверу

При балансуванні навантаження на основі сервера можна зробити так, що декілька серверів будуть відображатися як один сервер – один віртуальний сервіс – прозорим розподілом запитів між серверами. Розподіл навантаження на сервери запобігає вимкненню програмного або апаратного забезпечення служби для кінцевих користувачів, а також може надавати послуги аварійного відновлення шляхом перенаправлення запитів на обслуговування до резервної копії, коли відбувається відключення основного ресурсу [8,43,85,94,108]. Існує дві категорії реалізацій балансування навантаження на основі серверу:

1. Балансування навантаження на основі программного забезпечення: складається зі спеціального программного забезпечення, що встановлене на серверах у кластері балансування навантаження. Програмне забезпечення відправляє або приймає запити від клієнта до серверів на основі різноманітних алгоритмів. Наприклад, Microsoft Network Load Balancing є балансуванням навантаження програмного забезпечення для веб-ферм, а Microsoft Component Load Balancing є балансуванням навантаження програмного забезпечення для додатків ферми.

2. Апаратне балансування навантаження складається за спеціального комутатора або маршрутизатора з програмним забезпеченням для забезпечення функції балансування навантаження. Це рішення поєднує комутацію та балансування навантаження у одному пристрої, що приводить до зменшення кількості додаткового устаткування, необхідного для реалізації балансування навантаження.

2.5.1 Балансування навантаження на рівнях мережевої моделі OSI

Балансування на другому (канальному) рівні (рисунок 2.2). При балансуванні на другому рівні стека протоколів можна виділити два варіанта: балансування з використанням окремого виділеного балансувальника та без

нього. [110]. В обох випадках деяка IP-адреса сервіса встановлюється для усіх серверів, або на інший спеціалізований інтерфейс. Робиться це для того, щоб дані сервери могли приймати з'єднання на цю IP-адресу та відповідати з неї, але не відповідали б на ARP-запити (Address Resolution Protocol – протокол визначення адреси), що належать до цієї адреси.

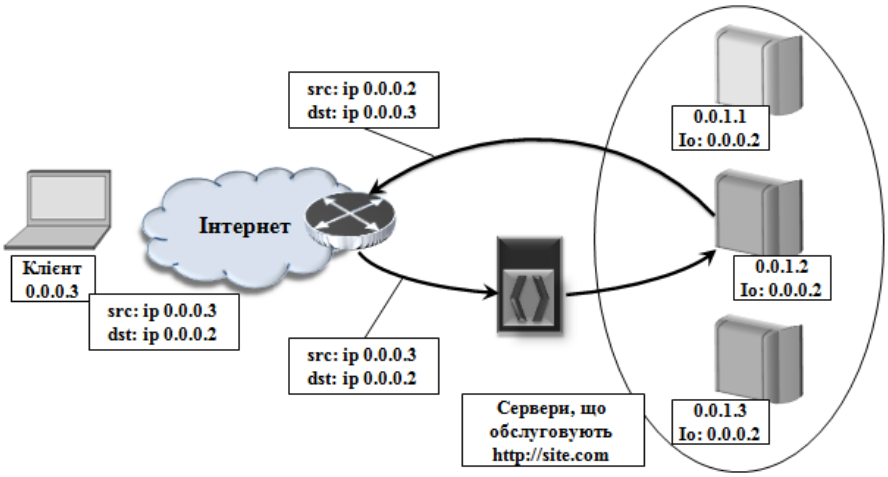


Рисунок 2.2 – Балансування на другому канальному рівні

Таке балансування працює наступним чином: на балансувальник, що має цю IP-адресу і відповідає на ARP, приходить, припустимо, перший пакет з'єднання. Визначається, що він є першим. Необхідним алгоритмом відправляють його на сервер, що нас цікавить, змінюючи MAC-адресу на місце призначення (англ. destination), записуємо його у деяку таблицю з'єднань.

Якщо цей пакет не є першим, то за таблицею з'єднань досліджується, на якому сервері здійснюється обробка цього з'єднання, після чого пакет відправляється туди.

Враховуючи, що заголовки третього рівня та вище не модифікуються, відповіді від серверів можна відправляти повз балансувальника безпосередньо прямо через Інтернет клієнту (до необхідного шлюза).

Найпоширенішим зараз рішенням серед програмних реалізацій даного метода є Linux Virtual Server. У URL-термінології даний метод балансування називається прямою маршрутизацією (англ. Direct Routing).

Балансування без виділеного балансувальника. У даному випадку IP-адреса сервіса прописується як статичний ARP-запис на шлюзі з деякою мультикастною MAC-адресою. Комутатори налаштовуються таким чином, аби фрейми, що приходять на цю MAC-адресу, доставлялись усім необхідним серверам.

На серверах обчислюється деякий HASH, наприклад, від IP-адреси клієнта. За його значенням сервер визначає, чи повинен він відповідати на ці запити. Якщо HASH=0, то повинен відповісти перший сервер. Він і відповідає. Інші сервери «знають», що вони не повинні відповідати.

Плюси методу такі:

– незалежність від протокола високого рівня. Можна балансувати HTTP, FTP або SMTP – різниці не буде;

– є метод балансування без виділеного балансувальника. При невеликій кількості серверів це може бути актуальним;

– є можливість відправляти відповіді повз балансувальника. Враховуючи те, що, наприклад, у протоколі HTTP розмір відповіді зазвичай значно більше, ніж розмір запиту, на ресурсах відбувається досить велика економія;

– відносно мале використання ресурсів.

Мінус методу – усі сервери повинні знаходитись у одному і тому самому сегменті мережі. Необхідним є специфічне налаштування серверів та мережевого обладнання. Тому цей метод не завжди є застосовуваним та зручним.

У якості реалізації цієї схеми можна використовувати кластер IP у файєрволі Iptables для Linux.

Балансування на третьому мережевому рівні (рисунок 2.3*)*. Балансування на мережевому рівні (рівні протоколу IP) передбачає рішення наступного завдання: потрібно зробити так, щоб за одну конкретну IP-адресу сервера відповідали різні фізичні машини [30,85,110]. Балансувальнику призначається та ж IP-адреса сервіса. Коли відбувається звертання до нього, застосовується так званий

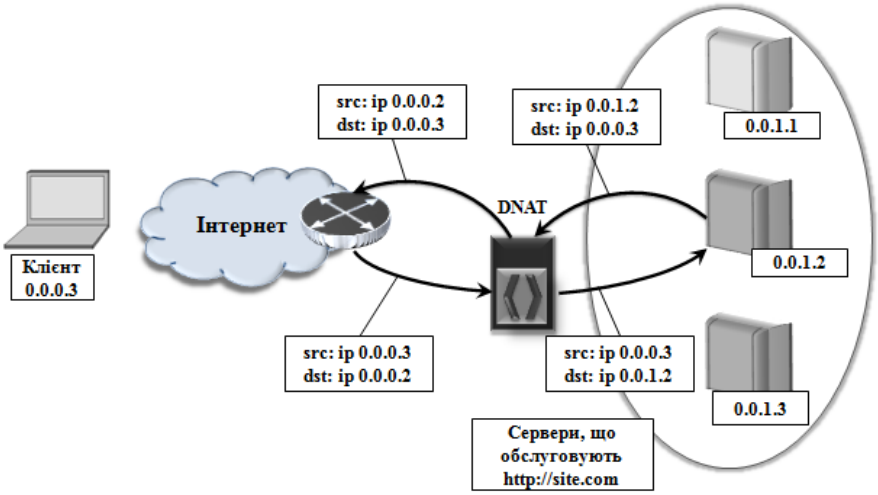


Рисунок 2.3 – Балансування на третьому мережевому рівні

Destination NAT, тобто підміняється IP-адреса призначення у пакеті: IP-адреса поточного сервера змінюється на обрану за необхідним алгоритмом IP-адресу сервера, що буде здійснювати обробку запиту.

Відмінність цього методу від попереднього полягає у тому, що модифікуються заголовки третього рівня. IP-адреса відправника повинна бути зміненою з IP-адреси сервера, що здійснює обробку запиту, на IP-адресу сервіса, який використовується балансувальником.

Реалізацій цього методу достатньо багато.

– DNS-балансування. На одне доменне ім’я виділяється декілька IP-адрес. Сервер, на який буде відправлено клієнтський запит, зазвичай визначається за допомогою алгоритма балансування навантаження, наприклад, Round Robin;

– побудова NLB-кластера. При використанні цього засобу сервери поєднуються у кластер, що складається із вхідних та обчислювальних вузлів. Розподіл навантаження здійснюється за допомогою спеціального алгоритма. Використовується у рішеннях від компанії Microsoft;

– балансування по IP з використанням додаткового маршрутизатора;

– балансування за територіальною ознакою здійснюється шляхом розміщення однакових сервісів з однаковими адресами у територіально різних регіонах Інтернету.

Плюси балансування на третьому рівні: незалежність від протокола високого рівня; абсолютна прозорість для серверів**.** Мінус – зворотний трафік серверів повинен проходити через балансувальник. Відповідно, навантаження на нього буде декілька вищим, ніж при використанні балансування на другому рівні.

Балансування на четвертому транспортному рівні (рисунок 2.4). Цей вид балансування є найпростішим: клієнт звертається до балансувальника, той перенаправляє запит до одного з серверів, що і буде здійснювати його обробку [30, 85,110].

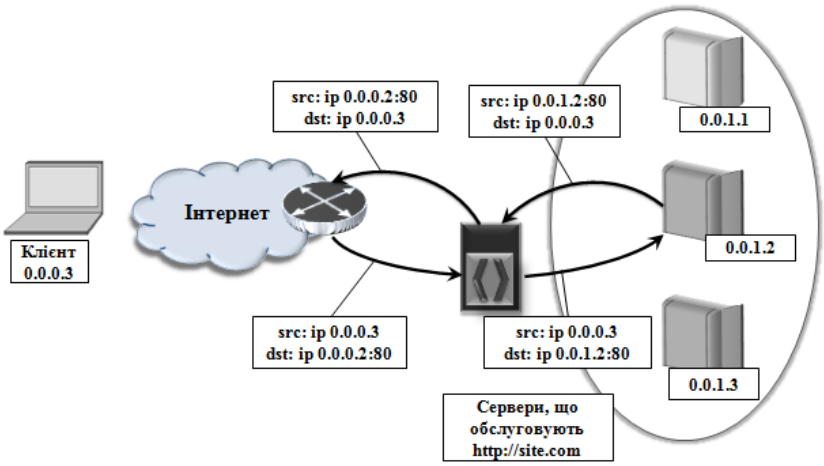


Рисунок 2.4 – Балансування на четвертому транспортному рівні

Вибір серверу, на якому буде здійснюватися обробка запиту, може проводитися у відповідності до різних алгоритмів: шляхом простого кругового перебора, шляхом вибору найменш завантаженого сервера з пулу тощо.

Інколи балансування на транспортному рівні важко відрізнити від балансування на мережевому рівні. При балансуванні вихідного трафіку на мережевому рівні не вказується ані конкретний порт, ані конкретний протокол передачі даних. Відмінність між рівнями балансування можна пояснити наступним чином. До мережевого рівня відносять рішення, що не термінують на собі користувальницькі сесії. Вони просто перенаправляють трафік і не працюють у проксуючому режимі. На мережевому рівні балансувальник просто вирішує, на який сервер передавати пакети. Сесію з клієнтом здійснює сервер.

На транспортному рівні спілкування з клієнтом замикається на балансувальнику, що працює як проксі. Він взаємодіє з серверами від свого імені, передаючи інформацію про клієнта у додаткових даних та заголовках. Таким чином працює, наприклад, популярний програмний балансувальник HAProxy.

На транспортному рівні розподіл відбувається виходячи з діапазону IP-адрес і порта (наприклад, якщо запит поступив на http://site.com, то трафік буде обраблено бекендом, що відповідає за роботу з 80 портом на цьому домені).

Балансування на сьомому прикладному рівні. При балансуванні на прикладному рівні балансувальник працює у режимі «розумного проксі». Він аналізує клієнтські запити та перенаправляє їх на різні сервери у залежності від характеру запитуваного контанту. [30,69,85,98]. Так працює, наприклад, веб-сервер Nginx, розподіляючи запити між фронтендом та бекендом.

Як приклад інструмента балансування на прикладному рівні можна навести pgpool – проміжний шар між клієнтом та сервером СУБД PostgreSQL. З його допомогою можна розподіляти запити за серверами баз даних у залежності від їх складу, наприклад, запити на читання будуть передаватися на один сервер, а запити на запис – на інший.

У прикладі (рисунок 2.5) користувач відвідує високо навантажений веб-сайт. Протягом сесії користувач може запитати статичний контент (зображення або відео), динамічний контент (канал новин, тракзакційна інформація – статус заказу). Балансування навантаження рівня 7 дозволяє маршрутизувати запити на основі інформації, що знаходиться безпосередньо у запиті, наприклад, за запитуваним змістом. Таким чином тепер запит зображення або відео може бути направленим до серверів, які зберігають його, та можна оптимізувати обслуговування для мультимедійного контенту. Запити транзакціонної інформації, такої як сплата, можуть бути направленими на сервер додатків, відповідальний за управління ціноутворенням.

Балансування навантаження рівня 7 також дозволяє збільшити ефективність інфраструктури додатків, тому що різні типи контенту мають

різноманітні вимоги у плані використання центрального процесора, пропускної здатності тощо.

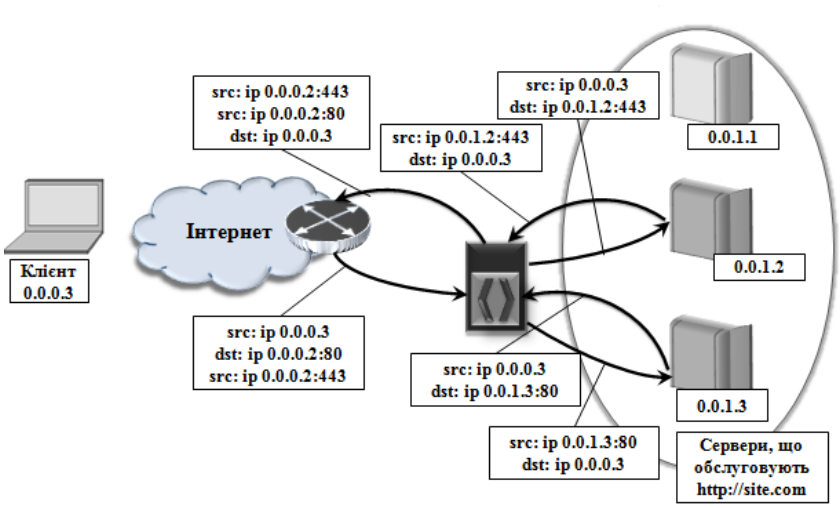


Рисунок 2.5 – Балансування на сьомому прикладному рівні

Таким чином, можливо для отримання найвищої ефективності серверів групувати їх так, щоб деякі з них здійснювали обробку транзакцій, у той час, як інши просто виступають у якості масивних систем зберігання для обслуговування статичних сторінок, або оптимізовані для закачування потокового відео, наприклад.

Комп’ютерна техніка, здатна виконувати балансування навантаження 7 рівня, називається контролерами доставки додатків (ADC). Вони комбінують традиційні можливості балансування навантаження з розширеними додатками комутації 7 рівня, щоб забезпечити проектування масштабованої, оптимізованої мережі доставки додатків. Проаналізуємо різницю між традиційним балансуванням та розширеними додатками комутації 7 рівня, переваги їх об’єднання в одному пристрої.

Балансування навантаження – це процес балансування запитів додатків, у яких балансувальник надає зовнішньому світу «віртуальний сервер», що приймає запити від імені пула (ферми) серверів, що містять такий же контент, і розподіляє ці запити за усіма серверами на основі алгоритму балансування навантаження.

Плюси балансування на сьомому рівні такі:

– рідко викликає знижену продуктивність на сучасному сервері;

– приймаються більш зважені рішення балансування навантаження;

– є можливість застосовувати оптимізацію і зміни змісту (наприклад, сжимання та шифрування);

– використовується буферизація, щоб розвантажити повільні з'єднання для підвищення продуктивності від серверів, що стоять вище.

Мінуси:

– накладні витрати на аналіз запитів є високими;

– обмежена масштабованість у порівнянні з балансуванням навантаження на інших рівнях.

Розглянемо відмінності між 4 та 7 рівнями балансування. Балансування навантаження 4 рівня працює з доставкою повідомлень, без відношення до змісту повідомлення. Протокол управління передачею (TCP) є протоколом 4 рівня для протокола передачі гіпертекста (HTTP) трафіка в мережі Інтернет. Транспортний рівень балансування навантаження просто передає мережеві пакети у напрямах до та від сервера, що стоїть вище, без перевірки змісту пакетів. На даному рівні можна зробити обмежені рішення про маршрутизацію, перевіряючи перші декілька пакетів у потоці TCP.

Балансування навантаження рівня 7 працює на рівні додатків, що має відношення до фактичного змісту кожного повідомлення. HTTP є найрозповсюдженішим протоколом 7 рівня. Балансування навантаження рівня додатків направляє мережевий трафік за допомогою набагато складніших механізмів, ніж вибір шляху на транспортному рівні балансування навантаження. Балансування навантаження рівня 7 призупиняє мережевий трафік протягом читання повідомлень. Таким чином, приймається рішення про балансування навантаження на основі змісту повідомлення (URL або кукі, наприклад). Після цього створюється нове з'єднання TCP для обраного сервера, що стоїть вище (або таке, що існує повторно, за допомогою HTTP-підтримки активності) і відсилається запит до сервера.

2.6 Апаратне балансування навантаження

Пристрої апаратного балансування навантаження працюють на 2-7 рівнях моделі OSI та використовуються для розділення мережевого навантаження поміж декількома серверами на основі таких факторів, як використання процесора CPU, кількість з'єднань, загальна продуктивність сервера [63,72].

Використання цього виду техніки мінімізує ймовірність того, що будь-який конкретний сервер буде перевантаженим, та оптимізує пропускну здатність для кожного комп'ютера або термінала. Окрім того, при використанні балансувальника можна звести до мінімуму час простою мережі, зробити легшою приорітезацію трафіка, забезпечити моніторинг додатків з кінця в кінець, забезпечити автентифікацію користувача та допомогти захиститися проти шкідливої активності, такої, як атака відмови в обслуговуванні (DoS).

Низькорівневе фільтрування, що використовується маршрутизатором LVS, має свої переваги у порівнянні з перенаправленням запитів на рівні додатків, тому що розподіл на рівні пакетів не викликає суттєвих обчислювальних витрат та допускає масштабування [82].

Основний принцип полягає у тому, що мережевий трафік направляється завдяки загальному IP, що називають віртуальним (VIP), або listening IP і ця адреса закріплена за балансувальником. Після цього балансувальник отримує запит на цей VIP і для нього виникає необхідність у прийнятті рішення про те, куди відправити цей трафік. Це рішення, як правило, контролюється алгоритмом балансування навантаження та набором правил.

Після цього запит відправляється на існуючий сервер, що буде здійснювати відповідь, яка у залежності від використовуваного типу балансування навантаження буде відправлена назад або до балансувальника у випадку пристрою 7 рівня, або, як правило, з пристрою 4 рівня, безпосередньо назад до кінцевого користувача (зазвичай за допомогою шлюза за

замовчуванням).

У випадку балансування на основі proxy, запит від веб-сервера може бути поверненим у балансувальник і обробленим перед відправкою назад користувачеві. Ця обробка може включати заміну змісту, сжимання, або інші сценарії. Більш докладно розглянемо, що ж відбувається з пакетами всередині NPB на рівні програмних рішень та що ж у результаті відправляється на балансування, а також – основні методи та алгоритми, що використовуються при балансуванні навантаження.

Балансування навантаження в NPB – процес розділення вхідного потока з одного або декількох інтерфейсів на декілька вихідних інтерфейсів за визначеними правилами або критеріями. Майже завжди разом з балансуванням використовуються такі функції:

– фільтрація – правила, що дозволяють виділити потоки з метою їх подальшого балансування або зменшення кількості даних у цих потоках;

– агрегація – поєднання потоків з декількох вхідних інтерфейсів у один перед виконанням операції балансування;

– комутація – більшість NPB може виконувати роль комутатора.

Основне застосування NPB – виділення з великих потоків даних необхідних та поділ їх на більш малі. Зустрічається це завдання досить часто.

Трафік, що приходить з декількох портів, агрегується та поступає на NPB. Відповідно до першої умови, тут до нього висувається одна із головних вимог – якщо обладнання клієнта працює з потоками на рівні сесій (а це майже завжди так), то NPB не повинен ці сесії порушувати (сесія – група пакетів, що передаються поміж конкретними вузлами), тобто пакети з однієї сесії завжди повинні надходити на один і той самих вихідний інтерфейс. Ця властивість називається Flow Coherency. Як розглянуто у [63], у NPB відбувається фільтрація трафіка, що дозволяє фільтрувати вхідні пакети за допомогою заданих правил за різноманітними мережевими протоколами, вирізати та аналізувати частину пакету, вставляти мітки порту , VLAN, MPLS у пакети, помічати або видаляти дубльовані пакети.

Зазвичай потоки, що поступають на NPB, визначаються за допомогою заголовків (src/dst IP, src/dst порт, протокол). Але структура потоків може змінюватися, а NPB повинен вміти підстроюватися під них. Наприклад, якщо IP-адреси фіксовані, а порти змінюються у залежності від шляху пакетів, або навпаки, порти фіксовані, а IP-адреси можуть змінюватися.

Також є деякі функції, що слідкують за станом каналів, з якими працює NPB:

– Link state awaraness – функція, що дозволяє відстежувати стан вихідних каналів. Якщо один з них (або клієнтське обладнання) ламається, то трафік автоматично перерозподіляється між іншими вихідними портами даної групи. Коли канал повертається у робочий стан, тоді знову відбувається перерозподіл трафіка. У моменти перерозподілу можливими є короткострокові порушення когерентності потоку. Працездатність канала відстежується за наявністю лінка та/або за допомогою keep-alive пакетів;

– N+M redundancy – функція резервування каналів, що працює наступним чином: у балансувальній групі обирається N використовуваних (активних) та M резервних каналів і якщо один з активних каналів групи ламається, його трафік переводиться на один із резервних каналів. При відновлення каналу перерозподіл не відбувається і відновлений канал залишається резервним. Ця функція використовується у тому випадку, коли навіть короткострокові порушення когерентності потока є недопустимими (замість link state awaraness);

– Overflow mode – ця функція дозволяє задіяти канали за мірою збільшення кількості трафіка. Користувач обирає групу каналів та вказує ті, які будуть активними від початку. Інші канали автоматично будуть задіяними після того, як навантаження на основні перевищить заданий користувачем поріг.

На основі властивості Flow Coherency можна привести наступні приклади типів балансування навантаження, що можуть бути використаними у NPB:

– рівномірне (per-packet або round-robin). На усі вихідні порти йде

приблизно однакова кількість трафіка, пакети призначаються на вихідні інтерфейсі по кругу. Даний тип балансування не забезпечує Flow Coherency, тому що направлення пакета ніяк не пов’язане з направленням інших пакетів;

– статичне. Балансування відбувається за фіксованим набором заданих правил, наприклад, за IP-source або типу протокола. При цьому кількість даних, що прийшли на конкретний вихідний інтерфейс, ніяк не враховується, тобто немає ніякої зворотної інформації про те, яких із вихідних інтерфейсів отримав більше або менше трафіка. У залежності від того, за якими саме полями відбувається фільтрація, даний тип балансування навантаження може або забезпечувати, або не забезпечувати Flow Coherency;

– динамичне. Відбувається облік трафіка, що відправляється на кожен вихідний порт. Найбільш оптимальне, якщо є жорсткі вимоги до рівномірного навантаження на вихідних інтерфейсах. За якими полями буде відбуватися даний тип балансування, залежить від алгоритма. Підтримує Flow Coherency;

– на основі хешів. Для вибору порта використовуються значення хеш-функцій, розрахованих за полями пакета (які задаються користувачем). З огляду на те, що для одних і тіх самих полів завжди буде розраховано один і той же хеш, при виборі необхідних полів балансування буде забезпечувати Flow Coherency. Якщо станеться так, що у розподілі хешів буде перекіс, балансування може стати дуже нерівномірним, тому що статистика за реальним вихідним навантаженням не використовується в алгоритмі балансування, проте ймовірність цього є невеликою.

Як уже згадувалося, основні функції пристроїв балансування навантаження полягають в забезпеченні постійної доступності серверів та мінімізації часу відповіді. Зокрема, балансувальники визначають помилки та перенаправляють у випадках необхідності запити на сервери, що працюють, виконують моніторинг стану серверів. У рамках даної технології також реалізуються деякі супутні основній функції: моніторинг та відстежування усіх з’єднань із серверами, що дозволяє зробити навантаження більш рівномірним та визначити сервери, що відмовили. Таким чином, оцінка стану серверів та

з’єднань є основною частиною процеса балансування.

2.7 Оцінка стану вузлів системи балансування навантаження

Для отримання інформації про поточний стан обчислювального вузла виділяють два підходи: зовнішній моніторинг стану обчислювальних вузлів розподіленої системи та внутрішній моніторинг [44,92,122,150].

Зовнішній моніторинг системи балансування навантаження.

Система балансування централізовано збирає дані про стан усіх обчислювальних вузлів розподіленої системи. Існує декілька методів зовнішнього моніторингу.

1. Метод розрахунку системой балансування навантаження часу відповіді обчислювального вузла, для чого він направляє на вузол службовий запит та заміряє час відповіді (наприклад, використовуючи протокол управління повідомленнями Internet Control Message Protocol – ICMP). Даний метод дозволяє системі переконатися у готовності обчислювального вузла та визначити кількість часу, необхідного для передачі даних на обчислювальний вузол від системи балансування навантаження та назад. Однак, затримка в отриманні відповіді може не залежати від завантаження обчислювального вузла. Тут затримки можуть бути внесені за рахунок середовища передачі даних. Якщо навантаження на мережу у розподіленій системі нерівномірне, то дані про завантаження вузлів, основані на часі відповіді вузла, можуть бути сильно викривлені.

2. Метод, заснований на відправці службових пакетів підконтрольним вузлам з метою отримання детальнішої інформації про завантаження обчислювального вузла (встановлення з’єднання з обчислювальним вузлом за протоколом TCP).

3. Метод, що дозволяє забезпечувати моніторинг часу відповіді та готовності обчислювального вузла і додатків, що на ньому працюють [154]. Час відклику додатку визначається як інтервал часу між відправкою запиту на

надання даних і до моменту заявлення про готовність до передачі.

Основною перевагою даного підходу є можливість урахування продуктивності обчислювального вузла та вводу вагових коефіцієнтів, які можна динамічно змінювати у процесі роботи системи балансування, що дозволяє гнучкіше змінювати навантаження на обчислювальний вузол.

Недоліки:

а) дані про завантаження вузла виражаються тільки часом відповіді на запит системи балансування, який може бути більшим через перевантаження у мережі, а не завантаження самого вузла;

б) відсутність даних про стан процесора, пам’яті, системи вводу/виводу і т.д.

Внутрішній моніторинг системи балансування навантаження.

Використовується для отримання детальнішої інформації про стан обчислювального вузла. При такому підході на кожен вузол розподіленої системи розміщується програма-агент, що збирає дані про вузол, на якому вона знаходиться. Дані від програм-агентів можуть передаватися на балансувальник або після запиту центральної (диспетчерської) частини системи балансування навантаження, або програми-агенти можуть самостійно передавати відомості про завантаження підконтрольного вузла на центральний вузол через певний інтервал часу. При даному підході існує проблема визначення частоти передавання обробленої інформації на балансувальник. Перший метод дозволяє за запросом центрального вузла отримувати дані від усіх вузлів у той момент, коли необхідно поставити новий запит клієнта у чергу на виконання. Другий метод дозволяє постійно мати нові та актуальні дані про завантаження обчислювальних вузлів, але при цьому збільшується навантаження на мережу через постійну передачу службових даних.

Основним недоліком даного підходу є використання службовими програмами-агентами певної кількості ресурсів вузла. Об’єм ресурсів вузла, що потребують програми-агенти, залежить від того, за скількома параметрами відбувається оцінка завантаженості обчислювального вузла [122,154], чи

відбувається обробка цих даних безпосередньо на вузлі, або ж вони передаються на центральну машину системи балансування навантаження. Тому при розробці системи балансування навантаження важливим питанням є визначення критеріїв завантаження обчислювального вузла. Потребується більша кількість ресурсів вузла для обробки інформації про завантаження компонентів (процесор, пам'ять) вузла. Однак, оцінка завантаження тількі одного компонента на дає об'єктивних даних про стан усього вузла. Тому у роботі запропоновано критерії, що враховують декілька ресурсів вузла у сукупності.

2.8 Аналіз стану розподіленої системи

Моніторинг стану серверів та вільної пропускної здатності [122,138] можна здійснити трьома способами:

– після кожного вхідного запиту;

– у фіксовані проміжки часу, що визначаються статичним алгоритмом;

– у нефіксовані проміжки часу, що визначаються динамічним алгоритмом.

Інформація, отримана першим способом, є найоб'ємнішою, тому що виміри проводяться після кожного отриманого запиту. При другому способі кількість інформації постійна, проте необхідно визначити інтервал зняття інформації, щоб об'єм інформації не був надлишковим або недостатнім. При третьому способі кількість інформації залежить від частоти інтервалів контроля, що повинен змінюватися залежно від структури отримуваного трафіка.

Перед тим як описати стратегію балансування навантаження, яка включає в себе комплексне вимірювання загального рівня дисбалансу системи (ступіня рівномірності розподілу навантаження між серверами), необхідно ввести деякі понятті та визначення.

Під терміном «доля» будемо розуміти частину ресурсів процесора,

виділених для задачі. Якщо завданню виділяється більше долей процесора, ніж іншим задачам, така задача отримує більше процесорних ресурсів від планувальника долевого розподілу. Долі процесора еквівалентні процентам ресурсів процесора. Долі дозволяють визначити важливість робочих навантажень у відношенні до інших робочих навантажень. У випадку призначення задачі долі процесора найважливішим є не кількість доль, виділених для завдання, а кількість доль, виділених для задачі, у порівнянні з іншими задачами. Слід також враховувати, що багато з цих задач будуть конкурувати із даним завданням за ресурси процесора.

Завантаження процесора, пам’яті і канала будемо розглядати як безрозмірні величини, значення яких нормовані та лежать у діапазоні [0, 1]. Загальновизнано, що якщо середнє значення завантаження постійно перевищує 0.70, слід зрозуміти причину такої поведінки системи, щоб уникнути проблем у майбутньому. Якщо ж середнє завантаження системи близьке до одиниці, то необхідно одразу знайти причину та виправити її.

2.8.1 Опис загального рівня дисбалансу системи кожного сервера

Необхідно ввести інтегровані значення загального рівня дисбалансу системи, а також середній рівень дисбаланса кожного сервера [92]. Одна з інтегрованих метрик балансу навантаження описується наступним чином (2.1)::

$$V = \frac{1}{(1 - CPU_i)(1 - RAM_k)(1 - Net_i)}, \qquad (2.1)$$

де CPU_i, RAM_k, Net_i – середнє завантаження процесора, пам’яті та пропускної здатності мережі, відповідно, протягом кожного періода, що спостерігається. Велике значення V означає високий ступінь комплексного використання. Тому алгоритми міграції можуть бути основані на даних вимірах. Насправді це є стратегією мінімізації комплексного використання

ресурсів шляхом перетворення тривимірної інформації (3D) ресурсів в одновимірну величину (1D). Це перетворення може привести до багатовимірної втрати інформації У [106] було запропоновано іншу інтегровану метрику балансування навантаження (2.2):

$$B = \frac{aN1_i C_i}{N1_m C_m} + \frac{bN2_i M_i}{N2_m M_m} + \frac{cNet_i}{Net_m}. \quad (2.2)$$

Першим обирається фізичний сервер m. Після цього – інші фізичні сервери i порівнюються з сервером m. $N1_i$ – це обсяг ЦПУ, $N2_i$ – це обсяг пам'яті. Тут C_i та M_i означають середні значення використання процесора та пам'яті відповідно. Net_i представляє собою мережеву пропускну здатність. Параметри a, b, c означають вагові коефіцієнти для процесора, пам'яті та пропускної здатності мережі відповідно. Основна ідея цього алгоритма полягає у виборі найменшого значення серед усіх фізичних серверів. Цей метод також перетворює 3D інформацію ресурсів у значення 1D.

2.8.2 Комплексне вимірювання дисбалансу системи

Враховуючи усі переваги та недоліки існуючих метрик для планування ресурсів [100], було розроблено методику комплексного вимірювання загального рівня дисбалансу системи, а також середнього рівня дисбаланса кожного сервера. Розглянуто наступні критерії:

1. Середнє завантаження кожного u-го процесора $CPU_i^u(T)$ i-го сервера визначається як середнє завантаження процесора протягом періода, що спостерігається. Наприклад, якщо період спостерігання складає 1 хв., а завантаження процесора записується через кожні 10 секунд, тоді CPU_i^u – це середнє значення з шести записаних значень i-го сервера.

Аналогічно визначається середнє завантаження кожної r-ї пам'яті $RAM_i^r(T)$ i-го сервера та середнє завантаження k-го канала $Net_i^k(T)$ i-го сервера протягом періода, що спостерігається.

2. Оскільки виміряне завантаження процесора, використання пам'яті та канала потоком класу qs відрізняється від середнього завантаження процесора CPU_i^{qs}, пам'яті RAM_i^{qs} та канала Net_i^{qs}, потоком класу qs відсутністю часу роботи операційної системи та переключенням поміж задачами, то можна ввести величини CPU_i^{qsv}, RAM_i^{qsv} і канала Net_i^{qsv}, які визначатимуть завантаження процесора, пам'яті та канала потоками класу qs, виміряне системою обліку або монітором операційної системи.

Завантаження процесора потоком класу qs обчислюється за формулою (2.3):

$$CPU_i^{qs} = CPU_i^u \times f_{CPU}^{qs}, \tag{2.3}$$

де f_{CPU}^{qs} – доля сумарного використання u-го процесора, яку можна віднести до класу qs. Параметр f_{CPU}^{qs} обчислюється наступним чином:

$$f_{CPU}^{qs} = CPU_i^{qsv} \Big/ \sum_{\forall qs} CPU_i^{qsv} .$$

Аналогічно розраховуються значення завантаження пам'яті та пропускної здатності мережі з урахуванням класів потоків. Завантаження пам'яті потоком класу qs розраховується за формулою (2.4):

$$RAM_i^{qs} = RAM_i^r \times f_{RAM}^{qs}, \tag{2.4}$$

де f_{RAM}^{qs} – доля сумарного використання r-й пам'яті, яку можна віднести до класу qs. Параметр f_{RAM}^{qs} обчислюється наступним чином:

$$f_{RAM}^{qs} = RAM_i^{qsv} \Big/ \sum_{\forall qs} RAM_i^{qsv} \ .$$

Завантаження каналу потоком класу qs розраховується за формулою (2.5):

$$Net_i^{qs} = Net_i^k \times f_{Net}^{qs} \quad , \tag{2.5}$$

де f_{Net}^{qs} – доля сумарного використання k-го каналу, яку можна віднести до qs. Параметр f_{Net}^{qs} обчислюється наступним чином: $f_{Net}^{qs} = Net_i^{qsv} \Big/ \sum_{\forall qs} Net_i^{qsv}$.

2. Введемо середній коефіцієнт використання усіх процесорів у системі. Нехай $CPU_i^{n_i}$ середнє завантаження ЦПУ i-го сервера (формула 2.6),

$$CPU_u^{All} = \frac{\sum_i^N CPU_i^u CPU_i^{n_i}}{\sum_i^N CPU_i^{n_i}}, \tag{2.6}$$

де N – загальна кількість фізичних серверів у системі, n_i – кількість ЦПУ на i-му сервері. Аналогічним чином, середній коефіцієнт використання пам’яті $RAM_i^{m_i}$, пропускна здатність лінії зв’язку $Net_i^{k_i}$ i-го сервера, уся пам’ять RAM_r^{All} і вся пропускна здатність мережі у системі Net_k^{All} може бути визначена за формулами (2.7) та (2.8):

$$RAM_r^{All} = \frac{\sum_i^N RAM_i^r RAM_i^{m_i}}{\sum_i^N RAM_i^{m_i}}, \tag{2.7}$$

$$Net_k^{All} = \frac{\sum_i^N Net_i^k Net_i^{k_i}}{\sum_i^N Net_i^{k_i}}, \tag{2.8}$$

4. Значення дисбалансу усіх процесорів. Використовуючи формулу дисперсії, значення дисбалансу усіх процесорів у системі визначається як (формула 2.9):

$$IMB_{CPU} = 1/N \sum_i^N (CPU_i^u - CPU_u^{All})^2, \quad (2.9)$$

Аналогічно можуть бути розраховані значення дисбалансу пам’яті та пропускної здатності мережі (формули 2.10 та 2.11):

$$IMB_{RAM} = 1/N \sum_i^N (RAM_i^r - RAM_r^{All})^2, \quad (2.10)$$

$$IMB_{Net} = 1/N \sum_i^N (Net_i^k - Net_k^{All})^2, \quad (2.11)$$

5. Введемо комплексне значення дисбалансу навантаження IMB_i i-го сервера, що враховує усі три ресурси сервера. Використовуючи формулу розрахунку дисперсії як міри нерівномірності, інтегроване значення дисбалансу навантаження i-го сервера можемо визначити як (2.12):

$$IMB_i = a(CPU_i^u - CPU_u^{All})^2 + b(RAM_i^r - RAM_r^{All})^2 + c(Net_i^k - Net_k^{All})^2$$

(2.12)

Параметри a, b, c означають вагові коефіцієнти для процесора, пам’яті та пропускної здатності мережі відповідно, які обираються експериментальним шляхом таким чином, що $a + b + c = 1$, у залежності від вирішуваних задач та структури системи.

IMB_i застосовується для позначення рівня дисбалансу навантаження шляхом порівняння коефіцієнтів використання процесора, пам’яті та пропускної здатності мережі.

Мінімізації підлягає значення $IMB_i \to \min$.

Тоді сумарні значення дисбалансу усіх серверів у системі записують як (2.13):

$$IMB_{tot} = \frac{1}{N}\sum_{i}^{N} IMB_i \,, \qquad (2.13)$$

6. Середня тривалість роботи при однаковій кількості задач дозволяє порівнювати різноманітні алгоритми планування.

7. Період обробки на i-му сервері визначається як максимальне навантаження на i-му сервері. Період обробки у системі визначається як середнє завантаження на усіх серверах.

8. Ефективність використання визначається як середнє навантаження на будь-якому сервері.

Таким чином, для планування ресурсів було розроблено методику комплексного вимірювання загального рівня дисбалансу системи, а також середнього рівня дисбалансу кожного сервера.

2.9 Висновки з розділу 2

1. У розділі розглянуто основні методи та механізми забезпечення якості обслуговування в мультисервісних комп'ютерних мережах з точки зору застосування їх при передачі в мережі фрактального трафіку. Проведено аналіз методів балансування навантаження на різних рівнях мережевої моделі, вказані переваги та недоліки кожного методу. Показано, що дослідження впливу методів балансування навантаження на якість обслуговування в мережі при передачі самоподібного трафіку можливо проводити в основному за допомогою імітаційного моделювання.

2. Приведена класифікація стратегій балансування навантаження, яка враховує автоматичні стратегії, реалізовані у складі спеціалізованого програмного забезпечення або включені у склад прикладних розподілених програмних комплексів. Виділено основні класи стратегій за рядом ознак. Описано види політик балансування, які використовують порогові значення завантаженості вузлів, відхилення величини навантаження вузла від середнього значення у системі та інші методи. Також проведено аналіз механізмів збору інформації

про завантаженість системи.

3. Запропонований розрахунок загального рівня дисбалансу системи і кожного сервера. У якості оцінки завантаження ресурсів вузлів запропоновано характеристики завантаження процесора, пам’яті та пропускної здатності каналу. У запропонованому методі розраховується середнє завантаження процесора, пам’яті та пропускної здатності каналу на основі завантаження, що виміряне системою обліку або монітором операційної системи. Запропонований метод дозволяє проводити розрахунок завантаження процесора, пам’яті та пропускної здатності каналу для потоків різних класів обслуговування як для кожного сервера окремо, так і для усієї розподіленої системи. Даний метод дозволяє розраховувати дисбаланс як всіх процесорів розподіленої системи, так і пам’яті та пропускної здатності каналів. Також введено комплексне значення дисбалансу навантаження.

4. Загальні результати цього розділу опубліковані у роботах [37,55,57,117, 124,125,130 131 134,136,147].

Список використаних джерел у даному розділі наведено у повному списку використаних джерел під номерами: [7,8,23,24, 27, 29, 30, 43, 44, 49, 51, 58, 59, 63, 69,72, 74-76, 82, 85, 92, 94, 95, 98, 100, 103, 106, 108, 110, 111, 113, 114, 120, 122, 138, 139, 150, 154].

РОЗДІЛ 3
ДОСЛІДЖЕННЯ МЕХАНІЗМУ ТА РОЗРОБКА МОДЕЛІ СИСТЕМИ БАЛАНСУВАННЯ З УРАХУВАННЯМ САМОПОДІБНИХ ВЛАСТИВОСТЕЙ НАВАНТАЖЕННЯ

3.1 Дослідження властивостей адитивного мультифрактального трафіка

3.1.1 Статистичне мультиплексування самоподібних потоків

Сучасні інформаційні мережі побудовані на основі мультиплексування потоків даних. Відповідно до класичної теорії масового обслуговування, множина потоків даних з випадковими варіаціями розподілів ймовірностей дадуть в результаті деякий усереднений згладжений трафік. Однак цей підхід не застосовний до потоків даних, що має властивості самоподібності. Особливе значення це набуває при побудові хмарних інфраструктур, де важливим завданням є оптимальний розподіл навантаження між компонентами.

Таким чином, одним із важливих завдань для підвищення якості обслуговування мереж є дослідження властивостей адитивних самоподібних трафіків. У роботах [90,153] теоретично і чисельно вивчені властивості самоподібних процесів і показано, що сума декількох самоподібних процесів з різними значеннями показника Херста, має максимальний показник. У роботах [77,107,109] представлені результати експериментальних досліджень властивостей адитивних інформаційних трафіків, які підтверджують теоретичні результати. Однак у цих дослідженнях не враховується берстность (наявність сильних викидів) трафіків, кількісною характеристикою якої є коефіцієнт варіації.

Розглянемо механізм статистичного мультиплексування інформаційних потоків, який широко використовується в телекомунікаціях, оскільки дозволяє економно використовувати пропускну здатність магістральних каналів. Він полягає в тому, що потоки окремих джерел складаються в магістральному каналі з економією пропускної здатності. За умови незалежності і відсутності

довгострокової залежності коефіцієнт варіації, результуючого процесу в магістральному каналі буде зменшуватися і результуючий процес буде значно згладженим.

Однак, якщо хоча б один з потоків є самоподібним, то сумарний потік набуває властивостей самоподібності [77,109]. Якщо підсумовуються кілька самоподібних потоків з різними значеннями показника Херста, то результуючий потік має максимальний показник. У цих випадках сумарний потік не згладжується і алгоритм статистичного мультиплексування виявляється малоефективним.

Метою даного розділу є дослідження властивостей адитивних модельних мультифрактального трафіків, які мають різну ступінь берстності, а також чисельний аналіз зміни фрактальних характеристик мультифрактального потоку при додаванні потоку, який не має мультифрактального властивостей.

3.1.2 Моделювання мультифрактальних потоків

Однією з найважливіших властивостей трафіка як випадкового процесу є наявність важких хвостів його функції розподілу. Тяжкість розподілу хвостів відповідає ступеню берстності. Коефіцієнт варіації можна розглядати як найпростіший кількісною характеристикою розподілу хвоста (3.1):

$$\sigma_{\text{var}}(T) = \frac{\sigma(T)}{M(T)}\ , \tag{3.1}$$

де T – це випадкова величина, значеннями якої є кількість подій в заданому інтервалі часу.

Основним інструментом для вивчення і прогнозування поведінки самоподібних потоків даних є моделювання, для якого необхідна модель самоподібного вхідного навантаження. В роботі [54] була запропонована модель агрегованого самоподібного трафіку, яка враховує як ступінь самоподібності, так і

«важкі хвости» функції розподілу. Параметрами моделі є інтенсивність трафіку, показник Херста і коефіцієнт варіації, який відповідає берстності реалізацій.

Модельна реалізація трафіку визначається як експоненціальне перетворення фрактального гауссівського шуму (3.2):

$$Y(t) = b \cdot \mathrm{Exp}[k \cdot X(t)], \tag{3.2}$$

де $X(t), t = 1,..., N$ – реалізація фрактального гауссівського шуму з показником Херста H; N – довжина реалізації; b і k є параметрами, що регулюють інтенсивність і берстність трафіку.

Стохастичний процес $Y(t)$ є самоподібним стохастичним процесом з тим же показником Херста H, що і початковий фрактальний гауссівський шум. Величина $Y(t)$ має логарифмічно нормальний розподіл. Мультифрактальні властивості $Y(t)$ змінюються в залежності від параметра k. На рисунках 3.1–3.3 представлені графіки узагальненого показника Херста, отримані по реалізаціям довжиною $N = 5000$, при значенні показника Херста фрактального гауссівського шуму $H = 0.8$, значення параметра $b = 1$.

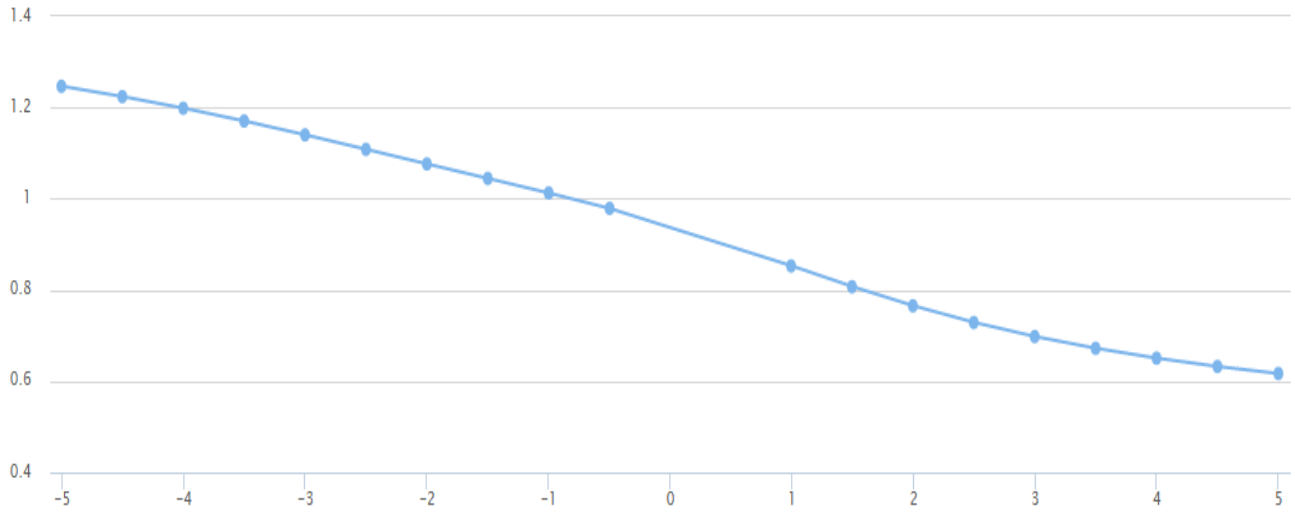


Рисунок 3.1 – Графік $h(q)$ для $k = 1$

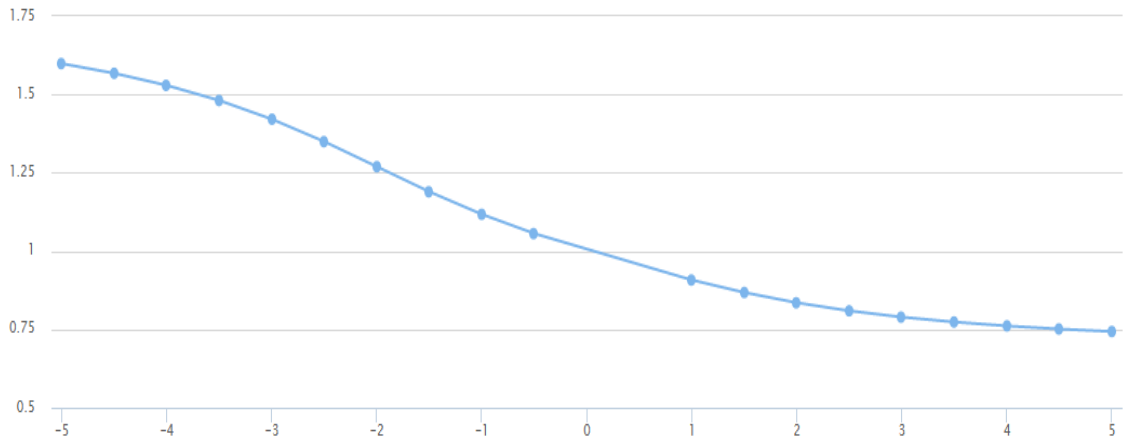


Рисунок 3.2 – Графік $h(q)$ для $k=1,4$

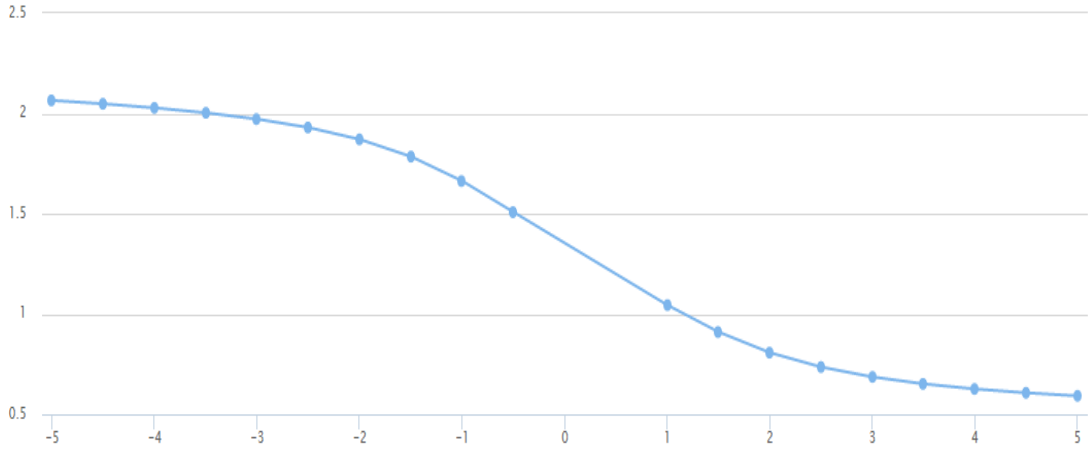


Рисунок 3.3 – Графік $h(q)$ для $k=2$

У табл. 3.1 представлені оцінки значень величин математичного очікування $\bar{X}$, дисперсії S^2, відповідного (3.1) коефіцієнта варіації $\hat{\sigma}_{var}$ і оцінок діапазону узагальненого показника Херста $\Delta h = h(q1) - h(q2)$, $q1 = -5$, $q2 = 5$, розраховані за усередненими за 100 значеннями, отриманих за реалізаціями довжиною $N = 5000$, при значенні показника Херста фрактального гауссівського шума $H = 0.8$. Значення параметра b було обрано $b = 1$, параметр k змінювався від 1 до 2. Із табл. 3.1 очевидно, що, змінюючи параметр k, можна добитися необхідного діапазону мультифрактальних властивостей трафіка, що генерується.

Таблиця 3.1 – Зміна Δh в залежності від параметра k

k	$\bar{X}$	S^2	$\hat{\sigma}_{\text{var}}$	Δh
1	5,51	50,59	1.31	0,44
1,2	6,68	107,36	1.52	0,52
1,4	8,61	294,90	1.99	0,79
1,6	22,27	2499,59	2.27	0,96
1,8	26,62	10598,69	3.86	1,06
2,0	33,02	16749,51	4.31	1,28

Придатними моделями трафіка із заданими мультифрактальними властивостями є стохастичні каскадні процеси [84]. При побудові стохастичних каскадів вагові коефіцієнти є незалежними значеннями деякої випадкової величини. В роботі [53] для вагових коефіцієнтів була використана випадкова величина з бета-розподілом. Це дозволяє генерувати реалізації трафіка з різним ступенем неоднорідності, тобто з великим діапазоном мультифрактальних властивостей.

У табл. 3.2 представлені оцінки значень коефіцієнта варіації $\hat{\sigma}_{\text{var}}$ і оцінок діапазону узагальненого показника Херста $\Delta h = h(q1) - h(q2),\ q1 = -5,\ q2 = 5$, розраховані за усередненими по 100 значенням реалізацій довжиною $N = 4096$ для симетричного бета-розподілу при різних значеннях параметра a.

Таблиця 3.2 – Зміна $\hat{\sigma}_{\text{var}}$ і Δh в залежності від параметра a

a	$\hat{\sigma}_{\text{var}}$	Δh
3	2.6	1.2
1.5	4.1	2
1	5.3	2.5
0.5	10	4.5

3.1.3 Дослідження властивостей адитивних мультифрактальних потоків

У роботі були проведені дослідження сумарних потоків різного типу. Кожна з модельних реалізацій була побудована на основі перетворення (3.2).

Розглянемо суму двох потоків: самоподібного $Y_1(t)$ і потоку з незалежними значеннями $Y_2(t)$. На рис. 3.4 представлені графіки типових реалізацій таких потоків. Реалізація, яка показана вгорі рисунка, є самоподібним потоком з теоретичним значенням показника Херста $H = 0.8$ і значенням коефіцієнта варіації $\sigma 1_{var} = 1.2$. На середньому рисунку показана реалізація трафіку, прирости якого є незалежними випадковими величинами з логнормальним розподілом. Теоретичні значення показника Херста $H = 0.5$ і $\sigma 2_{var} = 1.2$. Нижній графік показує реалізацію сумарного потоку $Y_{\sum}(t)$.

Рисунок 3.4 – Модельні реалізації: самоподібний потік, потік з незалежними значеннями і сумарний

Оцінювання показника Херста сумарного потоку показало, що при рівних відносинах значень коефіцієнтів $\sigma 1_{var}$ і $\sigma 2_{var}$ самоподібного $Y_1(t)$ і не самоподібного $Y_2(t)$ потоків значення показника Херста $H_{\sum}$ для сумарного потоку $Y_{\sum}(t)$ в середньому дорівнює показнику Херста H_1 самоподібного потоку.

Аналогічні дослідження були проведені для випадку, коли потік $Y_2(t)$ був процесом з короткостроковою залежністю. У цьому випадку як утворюючий процес $X(t)$ в перетворенні (3.2) виступав процес авторегресії

$$X(t) = \varphi X(t-1) + \varepsilon(t),$$

де φ – коефіцієнт авторегресії, $\varepsilon(t)$ – білий шум. Потоки даних з авторегресією мають короткострокову залежність, тому що в цьому випадку кореляційна функція спадає експоненціально.

Для рівних значень коефіцієнтів $\sigma 1_{var}$ і $\sigma 2_{var}$ потоку $Y_1(t)$ і потоку з короткостроковою залежністю $Y_2(t)$ значення показника Херста сумарного потоку $H_{\sum}$ в середньому дорівнювало показнику Херста H_1 самоподібного потоку.

Таким чином, адитивний процес самоподібного і не самоподібного потоків набуває властивостей самоподібності, в тому разі, коли коефіцієнти варіації потоків рівні або досить близькі один одному. Аналогічні результати наведені в роботах [77, 107, 109].

Тепер змінимо коефіцієнт варіації $\sigma 2_{var}$ не самоподібного потоку $Y_2(t)$. У разі, коли відношення:

$$R_1 = \frac{\sigma 1_{var}}{\sigma 2_{var}}$$

стає набагато більше одиниці, показник Херста $H_{\sum}$ сумарного процесу $Y_{\sum}(t)$ поступово зменшується, досягаючи значення 0.5.

У табл. 3.3 наведені оцінки показника Херста $\hat{H}_1$ для самоподібного потоку $Y_\sum(t)$ (теоретичне значення показника Херста $H=0.8$, довжина реалізації 1000 значень) і сумарного потоку $\hat{H}_\sum$ в залежності від відношення R_1. У цьому випадку потік $Y_2(t)$ був потоком з незалежними значеннями.

Таблиця 3.3 – Параметри самоподібного і сумарного потоків

$\hat{H}_1$	0.802	0.788	0.812	0.801	0.795
R_1	1	0.85	0.65	0.5	0.35
$\hat{H}_\sum$	0.792	0.754	0.632	0.578	0.497

Тепер розглянемо адитивний потік у разі, коли підсумовуються два самоподібних процеси. У роботах [77,109] показано, що сумарний процес $Y_\sum(t)$ двох самоподібних процесів $Y_1(t)$ и $Y_2(t)$ з показниками Херста H_1 и H_2 є самоподібним з показником Херста, рівним максимальному показнику $H_\sum = \max(H_1, H_2)$.

Проведені дослідження показали, що цей вираз виконується, коли процеси мають відношення коефіцієнтів варіації $R_1 \approx 1$. Якщо змінювати значення $\sigma 2_{\text{var}}$ потоку з меншим показником Херста H_2, то показник Херста сумарного потоку $H_\sum$ буде прагнути до значення H_2.

На рис. 3.5 показана реалізація самоподібного процесу з $H_1 = 0.8$ (точками) і реалізація процесу з $H_2 = 0.6$ (суцільна лінія). Відношення $R_1 = 0.65$. Адитивна реалізація $Y_\sum(t)$ має оцінку Херста $\hat{H}_\sum = 0.714$.

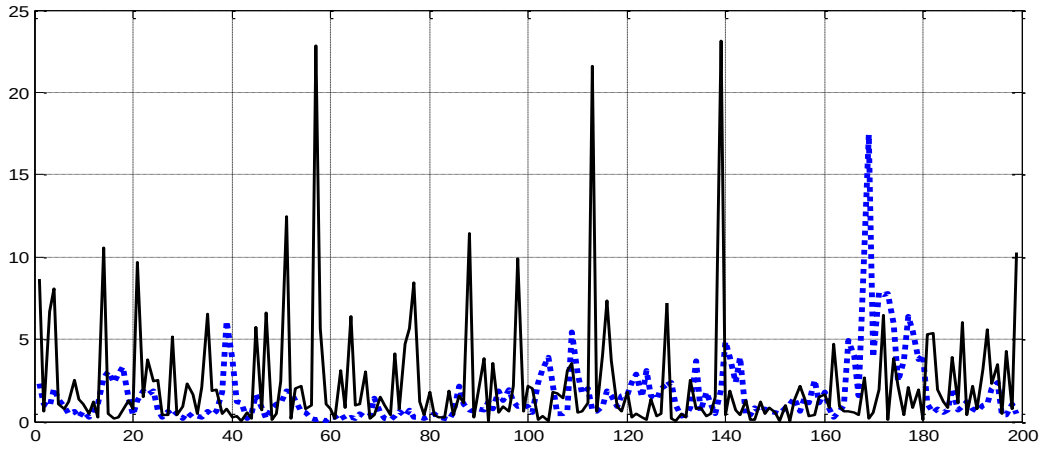


Рисунок 3.5 – Реалізації підсумованих потоків

В табл. 3.4 наведені оцінки показника Херста самоподібних потоків $Y_1(t)$ и $Y_2(t)$ і сумарного $Y_{\sum}(t)$ потоку в залежності від відношення R_1.

Таблиця 3.4 – Параметри підсумованих і сумарного самоподібних потоків

$\hat{H}_1$	0.789	0.791	0.809	0.781	0.805
$\hat{H}_2$	0.612	0.594	0.621	0.609	0.587
R_1	1	0.85	0.65	0.5	0.35
$\hat{H}_{\sum}$	0.805	0.732	0.714	0.634	0.612

У роботі [38] показано, що сумарний процес декількох самоподібних процесів з рівними дисперсіями і показниками Херста H_i є самоподібним з показником Херста, рівним максимальному показнику підсумованих потоків $H_{\sum} = \max(H_i, i = 1,..., N)$. При підсумовуванні декількох самоподібних потоків, з різними коефіцієнтами варіації, доцільно ввести коефіцієнт:

$$R_2 = \frac{\sigma_{\text{var}}(H\max)}{\frac{1}{N-1}\sum_{i=1}^{N-1}\sigma_{\text{var}}(i)},$$

де N – число підсумованих потоків,

$\sigma_{var}(H\max)$ – коефіцієнт варіації потоку з найбільшим показником ступеня самоподібності,

$\sigma_{var}(i)$ – коефіцієнт варіації i-го потоку.

Дослідження показали, що в цьому випадку показник Херста $H_{\sum}$ сумарного процесу $Y_{\sum}(t)$ залежить від того, наскільки значення R_2 менше одиниці. При значеннях $R_2 \approx 1$ показник Херста $H_{\sum}$ збігається з максимальним значенням H_i

3.1.4 Дослідження зміни характеристик мультифрактального потоку при додаванні потоку, який не має мультифрактальних властивостей

Як модельні мультифрактальні часові ряди було використані реалізації стохастичного біноміального мультиплікативного каскаду. При побудові стохастичних каскадів ваговими коефіцієнтами були незалежні значення випадкової величина, що має симетричний бета-розподіл. Це дозволило отримувати реалізації з різним ступенем неоднорідності, тобто з великим діапазоном мультифрактальних властивостей (див. табл. 3.2).

Досліджуваний модельний адитивний сигнал був представлений як (3.3):

$$X_{SUM}(t) = X_{MULTI}(t) + X_{NOISE}(t), \qquad (3.3)$$

де $X_{MULTI}(t)$ – мультифрактальний каскадний часовий ряд,

$X_{NOISE}(t)$ – адитивний шум.

Як величина, що характеризує співвідношення мультифрактального сигналу і шуму, використовувався коефіцієнт (3.4):

$$SNR = Var[X_{MULTI}] / Var[X_{NOISE}]. \qquad (3.4)$$

Розглянемо, як змінюється узагальнений показник Херста у випадку, коли

адитивним сигналом $X_{NOISE}(t)$ є білий шум, який в свою чергу є монофрактальним стохастичним сигналом. На рис. 3.6 представлені модельні реалізації мультифрактального каскаду, експоненціального білого шуму, отриманого відповідно до (3.2) і сумарний потік.

На рис. 3.7 показаний узагальнений показник Херста $h(q)$ в діапазоні значень параметра $-10 \le q \le 10$. Суцільна верхня лінія відповідає мультифрактальному потоку, лінія 1 відповідає адитивному потоку при значенні $SNR = 5$ і лінія 2 відповідає адитивному потоку при $SNR = 1$.

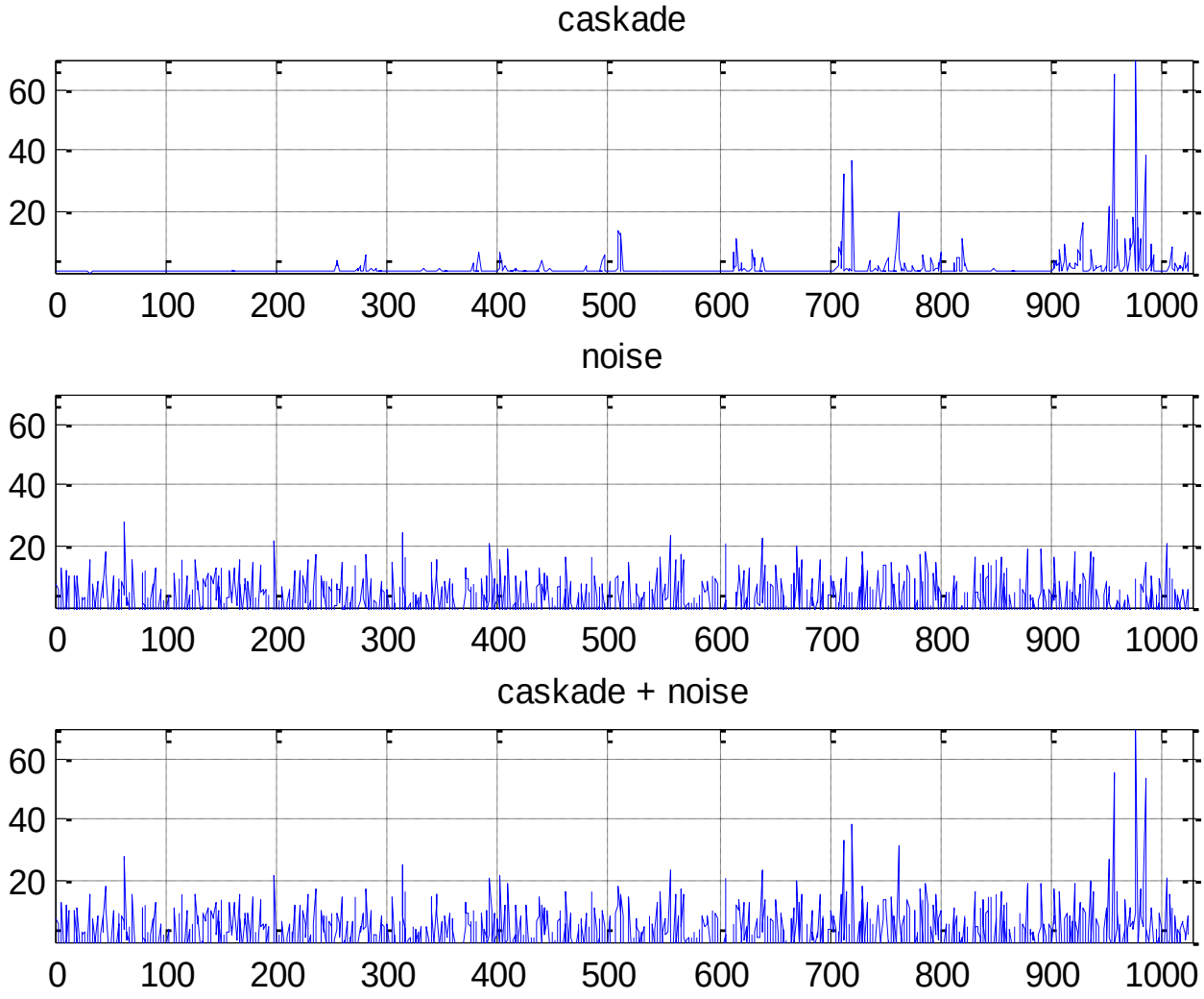


Рисунок 3.6 – Модельні реалізації: мультифрактальний потік, експоненціальний білий шум і сумарний потік

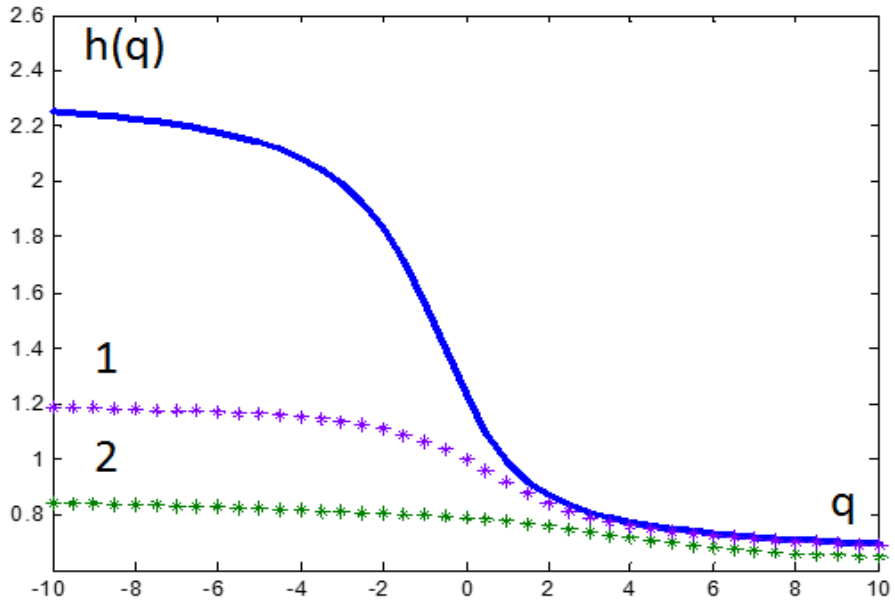


Рисунок 3.7 – Узагальнений показник Херста $h(q)$ для мультифрактального потоку (суцільна лінія), адитивного потоку при $SNR = 5$ (лінія 1) і $SNR = 1$ (лінія 2).

Очевидно, що узагальнений показник Херста адитивного ряду і показник вихідного мультифрактального ряду в випадку 1 при позитивних значеннях параметра q дуже близькі, тобто мультифрактальні властивості ряду не змінюються і можуть бути легко ідентифіковані.

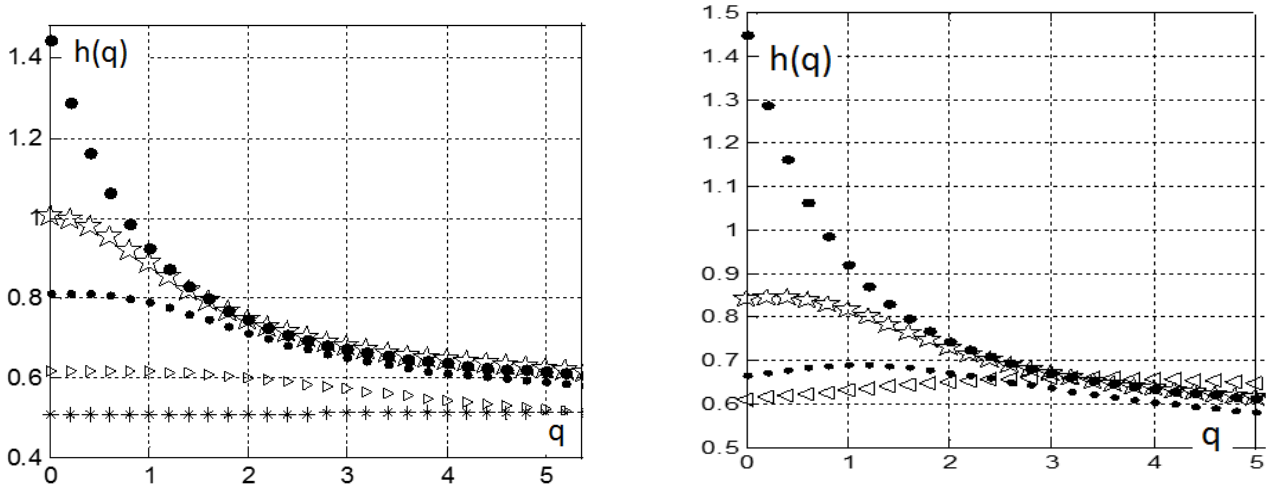
З огляду на вищенаведені результати, в подальшому при аналізі результатів розглядалися значення показника $h(q)$ тільки для позитивних значень параметра q. Чисельні дослідження показали, що при невеликому співвідношенні сигнал/шум ($SNR \le 5$), узагальнений показник Херста $h_{SUM}(q)$ адитивного ряду і показник $h_{MULTI}(q)$ вхідного мультифрактального ряду практично збігаються при $q \ge 0$.

Було чисельно досліджено узагальнений показник Херста $h_{SUM}(q)$ при зміні співвідношення сигнал / шум SNR, і показано, що при зменшенні значення SNR від 5 до 1 показник $h_{SUM}(q)$ зашумленого ряду прагне до $h_{MULTI}(q)$ початкового ряду. На рис. 3.8 (а) приведено функції $h_{MULTI}(q)$ для початкового

мультифрактального ряду (великі точки) і зашумленого ряду при значеннях коефіцієнта $SNR = 2,4,10,20$ (знизу вгору).

Аналогічні дослідження були проведені для випадку, коли шумовий потік не мав властивості самоподібності, а мав незалежні випадкові значення. Показано, що при зменшенні значення SNR від 5 до 1 показник $h_{SUM}(q)$ зашумленого ряду прагне до $h_{MULTI}(q)$ початкового ряду. На рис. 3.8 б наведено функції $h_{MULTI}(q)$ для початкового мультифрактального ряду (великі точки) і $h_{SUM}(q)$ зашумленого ряду при значеннях коефіцієнта $SNR = 2,4,10$ (знизу вгору).

На реалізації мультифрактального каскаду адитивно накладалися такі види шуму: білий шум, рівномірний некорельований шум, корельований шум (авторегресія першого порядку), самоподібний шум (фрактальний гауссівський шум).



а) б)

Рисунок 3.8 – Функції $h_{MULTI}(q)$ для мультифрактального ряду і $h_{SUM}(q)$ при самоподібному шумі для різних SNR (а) и при не самоподібному шумі для різних SNR (б)

Показано, що ця залежність (при зменшенні значення SNR показник $h_{SUM}(q)$ зашумленого ряду прагне до $h_{MULTI}(q)$ початкового ряду) має місце для будь-яких видів шумового сигналу з короткостроковою залежністю [56, 132].

3.2 Порівняльний аналіз алгоритмів балансування навантаження

У процесі балансування навантаження використовуються різноманітні алгоритми для управління трафіком з метою розподілення навантаження та/або максимального використання усіх серверів у кластері. Завдяки алгоритмам балансування досягається більш висока пропускна здатність та покращується час відповіді у розподілених системах. Однак кожен з алгоритмів має як переваги, так і недоліки [29,49].

1. Task Scheduling based on LB [26,27,79,88]: динамічний алгоритм, заснований на балансуванні навантаження, складається з дворівневого механізму планування завдань. Він забезпечує високу ефективність використання ресурсів. Цей алгоритм виконує балансування навантаження шляхом попереднього розподілу задач у віртуальних машинах, а після цього усіх віртуальних машин на ресурси хостів, таким чином, покращуючи час відповіді завдань, використання ресурсів та ефективність роботи середовища хмарних обчислень. Цей алгоритм задовольняє динамічні потреби користувачів та високий коефіцієнт використання ресурсів.

2. Opportunistic Load Balancing (OLB) [27,79,88]: статичний алгоритм, намагається зайняти кожний вузол, тому не рахує поточне навантаження кожного вузла або його придатність для виконання завдання. Іншими словами, OLB відправляє невиконані завдання на вільні у даний час вузли у випадковому порядку, незалежно від поточного навантаження вузлів. Перевагою є простота, досягнення балансу навантаження, проте недолік алгоритму у тому, що в ньому не розглядається очікуваний час виконання кожного завдання, що приводить до підвищення середнього часу завершення (загальний час циклу обробки).

3. Round Robin [27,52,79]: алгоритм кругового обслуговування, являє собою перебір за круговим циклом: перша задача передається одному вузлу, потім наступна задача передається наступному вузлу і так до досягнення останнього вузла, після чого усе починається з початку. У цьому алгоритмі усі задачі

порівну діляться поміж усіма процесорами, однак різні задачі мають різний час виконання, тобто зовсім не враховується завантаженість того чи іншого вузла у складі кластера. У Round Robin Scheduling (управління задачами в системах з розподілом часу) алгоритм визначає кільце як чергу та квант фіксованого часу. Кожне завдання може бути виконаним тільки у цей квант часу та в свою чергу. Якщо задача не може бути завершеною протягом одного кванту, вона повернеться до черги очікувати наступного кола. Однак, важко визначити оптимальний квант часу. Коли квант часу занадто великий, RR-алгоритм планування працює так же, як і FCFS Scheduling. А коли квант часу занадто малий, Round Robin Scheduling відомий як Processor Sharing алгоритм. Метод балансування Round Robin DNS не потребує зв'язку між серверами, тому він може використовуватися як для локального, так і для глобального балансування, також рішення на базі алгоритму Round Robin відрізняються низькою вартістю.

4. Weighted Round Robin [29,52,79]: вдосконалена версія алгоритму Round Robin: кожному вузлу присвоюється ваговий коефіцієнт у відповідності до його продуктивності та потужності. Це допомагає розподіляти навантаження гнучкіше: вузли з великою вагою обробляють більше запитів.

5. Randomized [79,81]: статичний алгоритм, випадково розподіляє навантаження поміж доступними вузлами, обираючи один з них за допомогою генерування випадкових чисел та відправки поточної задачі до нього. Цей алгоритм працює добре, коли усі процеси мають рівне навантаження, однак, коли навантаження різноманітних обчислювальних складностей, виникають проблеми. Цей алгоритм не підтримує детермінований підхід.

6. Min-Min Algorithm [13,22,27,49,79]: це статичний алгоритм балансування навантаження, тому параметри, що відносяться до роботи, відомі завчасно. Алгоритм якнайскоріше виділяє ресурси для задач, що можуть бути виконаними у найкоротші строки. Знаходиться мінімальний час виконання кожної із задач. Серед цих мінімальних часів шукається мінімальне значення і задача з цим значенням відправляється на виконання. До тих пір, поки усі задачі не будуть назначені на виконання, поставлені в чергу завдання будуть

відновлюватися, а виконані задачі видаляються з черги очікування. Завдання, що мають максимальний час виконання повинні ждати неспецифічний період часу. Основною проблемою цього алгоритму є те, що він може привести до простоювання. Найкраще він працює, коли більшість задач мають мінімальний час виконання.

7. Max-Min Algorithm [22,27,49,50,79]: алгоритм роботи майже такий же, як у алгоритмі Min-Min. Основна відмінність полягає у наступному: в цьому алгоритмі визначивши спочатку мінімальний час виконання задач, обирається максимальне значення, яке є максимальним часом серед усіх задач на усіх ресурсах. Далі задача зі знайденим максимальним часом призначається на виконання на конкретно обраний вузол. Після цього здійснюється перерахунок часу виконання усіх завдань на цьому вузлі шляхом додавання часу виконання поставленої задачі до часу виконання інших задач на цьому вузлі. Після цього поставлена задача видаляється зі списку системи.

8. Honeybee Foraging Behavior [27,51]: децентралізований алгоритм, що допомагає досягти збільшення пропускної здатності та глобального розподілу навантаження за допомогою локальних дій сервера. Розраховується поточне навантаження VM, потім вирішується стан VM: або вона перевантажена, або недовантажена, або збалансована. У відповідності до поточного навантаження VM групуються. Пріоритет задачі, що очікує в VM, враховується після видалення її з перевантаженої VM. Після цього ця задача призначається недовантаженій VM. Раніше зняті задачі корисні для пошуку недовантажених VM. Ці задачі відомі як бджоли-розвідники на наступному етапі. Алгоритм зменшує час відповіді VM, а також зменшує час очікування задачі. Продуктивність системи підвищується зі збільшенням часу очікування задачі. Основна проблема полягає у тому, що пропускна здатність не збільшується зі збільшенням розміру системи. Коли необхідним є різноманіття видів послуг, цей алгоритм є найбільш доцільним.

9. Active Clustering [29,79]: у цьому алгоритмі однакові вузли системи згруповані разом, вони працюють у групах. Це працює як у техніці самоагрего-

ваного балансування навантаження, де є перекомутоване для балансування навантаження системи. Система оптимізується з використанням аналогічних завдань роботи, підключивши подібні послуги. Продуктивність системи покращується з покращенням ресурсів. Пропускна здатність покращується шляхом ефективного використання усіх цих ресурсів.

10. Compare and Balance [29,79]: використовується для досягнення рівноважного стану та управління незбалансованим навантаженням системи. У цьому алгоритмі на основі ймовірності (номер віртуальної машини, запущеної на поточному хості та усієї хмарної системи), поточний хост випадковим чином обирає хост та порівнює їх навантаження. Якщо навантаження поточного хоста більше вибраного хоста, він передає додаткове навантаження на цей конкретний вузол. Після цього кожний вузол системи виконує ту ж саму процедуру. Цей алгоритм балансування навантаження також розроблений і реалізований для зменшення часу міграції віртуальних машин. Загальна пам’ять використовується для зменшення часу міграції віртуальних машин.

11. Lock-free multiprocessing solution for LB [27,66,79]: пропонується безблокувальне багатопроцесорне рішення для балансування навантаження, що виключає використання поділюваної пам’яті на відміну від інших рішень багатопроцесорного балансування навантаження, які використовують загальну пам’ять та блокування для підтримки сеансу користувача. Це досягається шляхом модифікації ядра. Дане рішення допомагає у покращенні загальної продуктивності балансування навантаження у багатоядерних середовищах шляхом запуску декількох процесів балансування навантаження в одному балансувальнику навантаження.

12. Ant Colony Optimization [20,50,71,79]: це розподілений алгоритм. У цьому алгоритмі інформація про ресурси динамічно відновлюється при кожному русі мурах. Множинні колонії мурах описуються таким чином, що вузол посилає кольорові колонії по усій мережі. Розмальовані колонії мурах використовуються для уникнення виходу мурах, що йдуть за одним маршрутом, з одного гнізда, а також забезпечення їх розподілу за усіма вузлами системи, де

кожна мураха діє як мобільний агент, який несе оновлену інформацію балансування навантаження у наступний вузол.

13. Shortest Response Time First [49,66,88]: ідеєю алгоритму є пряма переадресація. У ньому кожному процесу призначається пріоритет його запуску. Усі процесі з рівними пріоритетами планується FCFS порядку. SJF алгоритм є окремим випадком загального алгоритму пріоритетного планування. В алгоритмі SJF пріоритет є зворотнім по відношенню до наступного різкого сплеску процесора (CPU). Це означає, що якщо такий сплеск процесора збільшується, то пріоритет знижується. Політика SJF обирає задачу за найкоротшим часом обробки. У цьому алгоритмі короткі задачі виконуються перед довгими задачами. У SJF дуже важливо знати або оцінити час обробки кожного завдання, що є головною проблемою SJF.

14. Based Random Sampling: [78,88]: підхід розподіленого та масштабованого балансування навантаження, який використовує випадкову вибірку з системного домену, щоб домогтися самоорганізації, таким чином балансуючи між усіма вузлами системи. Продуктивність системи покращується при збільшенні кількості подібності ресурсів, що приводить до збільшення пропускної здатності шляхом ефективного використання великого об’єму ресурсів системи. Однак, алгоритм погіршується зі збільшенням різноманіття ресурсів.

15. The two phase scheduling load balancing algorithm [26,50,88]: поєднання OLB (опортуністичне балансування навантаження) та LBMM (Load Balance Min Min) алгоритмів планування, що використовує більш високу ефективність виконання та підтримку системи балансування навантаження. OLB тримає кожний вузол у робочому стані, щоб досягти мети балансування навантаження, а LBMM-алгоритм планування використовується, щоб мінімізувати виконання кожного завдання на вузлі, тим самим мінімізуючи загальний час завершення. Цей алгоритм використовується для підвищення ефективності використання ресурсів та підвищує ефективність роботи.

16. Active Clustering load balancing Algorithm [31,50,88] – алгоритм самоагрегації, оптимізує задачі, підключивши схожі послуги з використанням локального переприсвоювання. Працює на основі групування схожих вузлів. Процес групування заснований на концепції рефері-вузла. Рефері-вузол утворює з'єднання між сусідами, яке є подібним до ініціюючого вузла. Після цього рефері-вузол розриває з'єднання між собою та початковим вузлом. Далі сукупність процесів знову і знову повторюється. Продуктивність системи зростає на базі високої доступності ресурсів, через це пропускна здатність також збільшується.

17. ACCLB [88]*:* методика балансування завантаження, заснована на мурашиній колонії та теорії складної мережі (ACCLB) у відкритих хмарних обчисленнях. Вона використовує низько рівневі та безмасштабні характеристики складної мережі, щоб домогтися кращого розподілу навантаження. Ця методика дозволяє перебороти неоднорідність, є адаптивною до динамічного середовища та має гарну масштабованість, відповідно, допомагає у покращенні продуктивності системи.

18. Decentralized content aware: [80,88]: метод балансування навантаження, що називається робочим навантаженням та політикою клієнтського сповіщення. (WCAP). Даний метод використовує параметр під назвою USP для вказування унікальної особливої властивості запитів, а також обчислювальних вузлів. USP допомагає планувальнику прийняти рішення про найбільш доцільний вузол для обробки завдання. Ця стратегія реалізується на основі децентралізації з низькими витратами. Використовуючи інформацію контенту, щоб звузити пошук, покращується продуктивність пошуку усієї системи, а також знижується час простою обчислювальних вузлів, відповідно, покращується їх використання.

19. Server-based LB for Internet distributed services [80,88]: метод балансування навантаження для веб-серверів, розподілених по усьому світу. Він допомагає у скороченні часу відповіді сервісу за допомогою протоколу, що обмежує перенаправлення запитів до найближчих віддалених серверів без

їхнього перевантаження. Для реалізації цього протоколу характерним є проміжне програмне забезпечення. Він також використовує евристику, щоб допомогти веб-серверам витримати перевантаження.

20. Join-Idle-Queue [29,88]: алгоритм балансування навантаження для динамічно масштабованих веб-сервісів, забезпечує масштабне балансування навантаження за розподіленим відправником. Спочатку обчислює доступність процесорів, що знаходяться у стані простою, у кожного відправника, після цього призначає завдання процесорам для зменшення середньої довжини черги у кожному процесорі. При видаленні задач балансування навантаження з критичного шляху обробки запиту, він ефективно знижує навантаження системи, не несе ніякого комунікаційного навантаження на задачі, що знову прийшли, та не збільшує фактичний час відгуку.

21. Token Routing [81]: основною метою алгоритму є зведення до мінімуму вартості системи шляхом переміщення маркерів всередині системи. Але у масштабованій хмарній системі агенти не можуть мати достатньо інформації про поширення робочого навантаження у зв’язку з комунікаційними вузькими місцями. Так розподіл між агентами не зафіксовано. Недолік даного алгоритму може бути видалено за допомогою евристичного підходу балансування навантаження на основі маркера. Цей алгоритм забезпечує швидке та ефективне рішення маршрутизації. У цьому алгоритмі агенти не повинні мати знання про глобальний стан та робоче навантаження сусіда. Для того щоб приймати рішення про передання маркера агенти дійсно будують власну базу знань. Ця база знань походить від раніше отриманих лексем. Таким чином, у цьому підході не виникає ніяких накладних витрат комунікації.

22. Central queuing [79]: цей алгоритм працює за принципом динамічного розподілу. Кожна нова задача прибуває до менеджера черг та стає в чергу. Коли запит для виконання задачі приймається менеджером черг, він видаляє першу задачу з черги та передає її стороні, що її запитала. Якщо у черзі немає готових задач, то запит буферізується, поки нова задача не буде доступною. Але у випадку запису нової задачі у чергу, поки у черзі існують запити, на які не

відповіли, перший такий запит видаляється з черги, а нова задача ставиться перед нею. Коли завантаження процесора падає нижче порогового значення, локальний менеджер завантаження посилає запит на нову задачу центральному менеджеру завантаження. Після цього центральний менеджер відповідає на запит, якщо знайдена готова задача, у протилежному випадку видержується черговість запитів до надходження нової задачі.

23. Connection mechanism [27,66,81]: алгоритм балансування навантаження також може базуватися на механізмі найменшої кількості з'єднань, який є частиною динамічного алгоритму планування. Він є необхідним для підрахування кількості з'єднань для кожного сервера динамічної оцінки навантаження. Балансувальник навантаження записує кількість з'єднань кожного сервера. Кількість з'єднань збільшується, коли нове з'єднання відправляється до сервера і зменшується, коли з'єднання завершується або виникає переривання з'єднання.

24. Compare and Balance [27,31,79] алгоритм використовується для досягнення рівноважного стану та управління системою балансування навантаження. У цьому алгоритмі, заснованому на ймовірності (кількість віртуальних машин, працюючих на поточному хості та в усій розподіленій системі), поточний хост випадковим чином обирає хост і порівнює їх навантаження. Якщо навантаження поточного хоста більше одного із обраних хостів, він відправляє додаткова навантаження на цей конкретний вузол. Після цього кожний вузол системи виконує ту ж саму процедуру. Цей алгоритм балансування навантаження також розроблений та впроваджений для зменшення часової міграції віртуальної машини. Загальна пам'ять використовується для скорочення часу міграції віртуальних машин.

25. Least connections [71,49] алгоритм відправляє запити на сервер, який у даний час обслуговується найменшою кількістю підключень. Балансувальник навантаження буде контролювати кількість підключень сервера та відправляти на сервер запит з мінімальною кількістю підключень.

У табл. 3.5 наданий порівняльний аналіз алгоритмів балансування навантаження за різними показниками продуктивності, описаними у розділі 1.

Таблиця 3.5 – Порівняння алгоритмів балансування навантаження за показниками продуктивності

Algorithms/ Metrics	Пропускна здатність	Витрати	Відмовостійкість	Час міграції	Час відгуку	Використання ресурсів	Масштабованість	Продуктивність
Task Scheduling based on LB	y	Y	no	y	Y	y	y	y
Opportunistic LB (OLB)	Y	Y	Y	no	no	Y	y	Y
Round Robin	Y	Y	no	no	Y	y	no	Y
Weighted Round Robin	Y	Y	Y	Y	no	y	no	no
Randomized	Y	Y	Y	no	no	y	no	Y
Min-Min	Y	Y	no	no	Y	y	no	Y
Max-Min	Y	Y	no	no	Y	y	no	Y
Honeybee Foraging Behavior	Y	no	no	no	no	y	y	Y
Active Clustering	no	Y	no	Y	no	y	no	no
Compare and Balance	Y	Y	Y	Y	Y	no	y	Y
Lock-free multiprocessing solution for LB	no	Y	Y	no	no	y	no	no
Ant Colony Optimization	Y	Y	no	no	no	y	y	Y
Shortest Response Time First	no	Y	no	Y	Y	y	No	Y
Based Random Sampling	Y	no	no	no	no	no	No	Y
The two phase scheduling LB	Y	Y	no	no	Y	y	No	Y
Active Clustering LB	no	Y	no	Y	no	y	No	no
ACCLB	Y	Y	no	no	no	y	y	Y
Decentralized content aware	Y	Y	no	Y	Y	no	y	Y
Server-based LB	no	no	Y	Y	Y	y	y	Y

Продовження табл. 3.5

Join-Idle-Queue	Y	Y	no	Y	Y	no	y	Y
Token Routing	no	Y	no	Y	no	no	y	no
Central queuing	no	Y	no	Y	Y	y	no	Y
Connection mechanism	Y	Y	y	no	no	y	y	no
Least connections	Y	Y	no	no	Y	Y	Y	Y

У табл. 3.6 наводиться порівняння різноманітних типів сценаріїв розподілу навантаження у розподіленому середовищі. У ній визначається база знань, використання та недоліки кожного типу алгоритму.

Таблиця 3.6 – Порівняння різних типів сценаріїв розподілу навантаження

Тип алгоритму	База знань	Питання, що потребують вирішення	Застосування	Недоліки
Статичні	Потребується попередня база знань про статистику кожного вузла та вимоги користувачів	Час відгуку; використання ресурсів; масштабованість; необхідні потужність та енергія, використання; Makespan; пропускна здатність/продуктивність	Використовуються в однорідному середовищі.	Не гнучкий; не масштабований; не сумісний зі змінними вимогами користувачів та завантаження
Динамічні	Контролюється статистика часу запуску кожного вузла, щоб адаптуватися до змінних потреб навантаження.	Розміщення процесора, якому передається навантаження від перевантаженого процесора; передача задачі на віддалену машину; збір інформації; оцінка навантаження; обмеження числа міграцій; пропускна здатність	Використовується у гетерогенному середовищі.	Складність; великі витрати часу

Продовження табл. 3.6

Централізовані	Один вузол або сервер відповідає за підтримку статистики усієї мережі та періодичне оновлення.	Політика порогових значень; пропускна здатність; невиконання інтенсивності; комунікація між центральним сервером і процесорами в мережі; взаємодія службових сигналів	Корисний у невеликих мережах з низькою навантаженістю.	Не відмовостійкий; перевантаження центрального процесора прийняття рішень
Розподілені	Усі процесори в мережі відповідальні за розподіл навантаження і наповнення власної локальної бази даних для прийняття ефективних рішень балансування. Вибір процесорів, які приймають участь у балансуванні навантаження.	Час міграції; міжпроцесорні комунікації; критерії обміну інформацією; пропускна здатність; відмовостійкість	Корисний у великих та гетерогенних мережах	Складність алгоритмів; комунікаційні витрати
Ієрархічні	Вузли на різних рівнях ієрархії взаємодіють з вузлами, що їм належать, щоб отримати інформацію про продуктивність мережі.	Політика порогових значень; критерії обміну інформацією; вибір вузлів на різних рівнях мережі; інтенсивність відмов; продуктивність; час міграції	Корисні у середній або великій мережі з гетерогенним середовищем.	Чуттєві до невиправностей; складність алгоритмів
Залежність робочого процесу	Орієнтований ациклічний граф використовується для моделювання залежностей задач та може бути використаним для прийняття рішень планування.	Типи робочих процесів; один робочий процес; множина робочих процесів; робочі процесі стимуляції транзакції; робочі процеси стимулювання даних; відмовостійкість; час виконання Makespan; час міграції	Використовується при моделюванні залежностей задач у будь-яких умовах (однорідної або неоднорідної)	Складність моделі; обслуговування бази знань є складним; висока складність

3.3 Розробка математичної моделі балансування з урахуванням самоподібності вхідного навантаження

Система балансування навантаження, що розглядається, складається з групи серверів та балансувальника навантаження [32,33,105]. Розглянемо систему балансування навантаження, схематично надану на рис. 3.9. Балансувальники навантаження використовують різноманітні алгоритмі управління трафіком з метою розподілу навантаження та/або максимального використання усіх серверів у кластері.

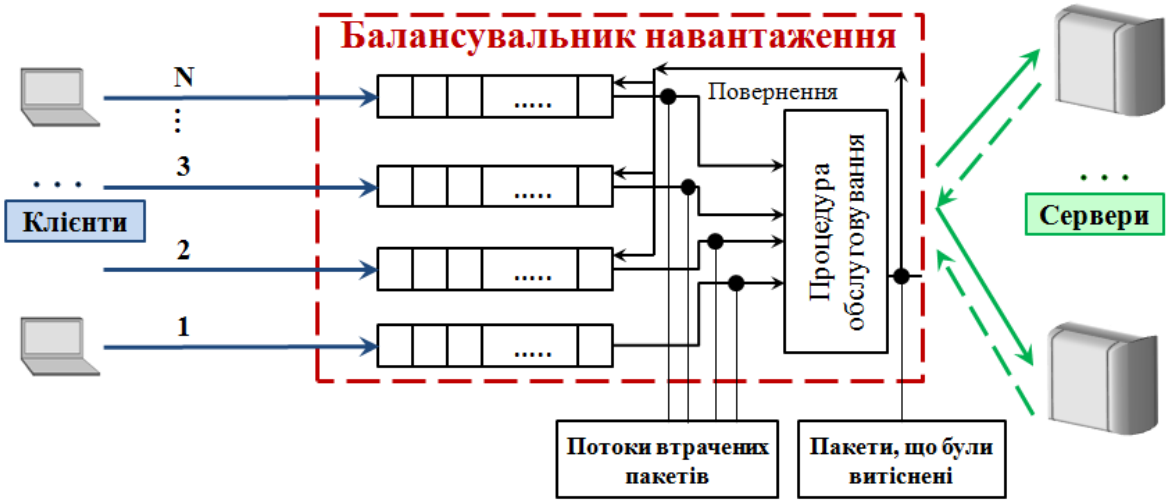


Рисунок 3.9 – Модель балансування навантаження

У кожний момент $t \in T$ на балансувальник LB надходить трафік інтенсивністю $\lambda = [\lambda_1, \lambda_2, .., \lambda_\sigma]$, що відноситься до qs-го класу обслуговування, який необхідно доставити на i-й сервер $Serv_i$ для обробки, без перевищення заданих максимально допустимих значень затримки τ_{qs} і максимально допустимого процента втрат l_{qs} в залежності від поточного завантаження серверів та реальної пропускної здатності у конкретний момент часу. У комп'ютерних мережах на маркування класів обслуговування виділяється 8 бітів, таким чином можливі 24 класи, але зазвичай використовуються 8 класів. Кожному класу об-

слуговування відповідають значення максимально допустимої затримки τ_q і максимально допустимого процента втрат l_q.

Трафік володіє множиною характеристик $V=\{\lambda,h,\mu\}$, де $\lambda=[\lambda_1,\lambda_2,..,\lambda_\sigma]$ – потоки заявок (пакетів) різноманітної інтенсивності; $h=[H,h(q),\Delta h]$, де $h(q)$ – вибіркове значення функції узагальненого показника Херста, $H=h(2)$ – значення параметра Херста, $\Delta h=h(q_{\min})-h(q_{\max})$ – діапазон значень узагальненого показника Херста для ділянки трафіка, μ – трудомісткість запиту. Трудомісткість запиту визначається як вектор ресурсів, що потребуються $\mu=(CPU,Net,RAM)$ для виконання запиту. Кожному qs-му класу обслуговування відповідає набір векторів необхідних ресурсів $\mu_{\text{qs}}=(CPU,Net,RAM)$.

Балансувальник навантаження і сервера поєднані між собою двобічними мережевими зв'язками $Link_{\text{lk}}=\{L_{\text{lk}}\}$, $lk=1,2,..,$ що поділяються на k каналів з пропускною здатністю $Net_{lk}^{k}(t)=\{Net_{lk}^{k}\}$ у момент часу t [44, 92]. Кожен i-й сервер характеризується наступними параметрами: $CPU_i^{n_i}$ середнє завантаження ЦПУ i-го сервера за період часу T, де n_i – кількість ЦПУ (ядер) на i-му сервері (те, як ядра розподілені за процесорами, будемо вважати несуттєвим: два чотириядерних процесори відповідають чотирьом двоядерним та відповідають восьми одноядерним процесорам, має значення лише загальна кількість ядер); середній коефіцієнт використання пам'яті i-го сервера $RAM_i^{m_i}$ за період часу T, де m_i – кількість пам'яті на i-му сервері; середній коефіцієнт використання пропускної здатності каналу $Net_i^{k_i}$ за період часу T, де k_i – кількість каналів у лінії зв'язку між балансувальником та на i-му сервері.

На вхід балансувальника навантаження LB надходить декілька незалежних мультифрактальних потоків пакетів з різною інтенсивністю $\lambda_1,\lambda_2,..,\lambda_\sigma$, кожен з яких відправляється у чергу Q_{w} обмеженої ємності. Час обслуговування заявок залежить від класу обслуговування qs, тобто враховується пріоритетність заявок (найвищий пріоритет – першим). Поки усі пріоритетні

запити на обслуговування не будуть оброблені, пакети інших типів залишаються у черзі до спливання їх часу життя. Пріоритетні запити, що надійшли знову, переривають обробку непріоритетних та з ймовірністю, що дорівнює одиниці, витісняють їх у накопичувач (якщо є вільні місця очікування), або за межі системи (якщо вільних місць немає). Пакети, що витіснили з обслуговування, приєднуються до черги непріоритетних вимог та можуть бути обслужені після усіх пріоритетних. Накопичувачі є розділеними для кожного вхідного потоку, вільні місця очікування є доступними для будь-якого запиту, що знову надійшов.

На відміну від типових пріоритетних СМО, система, що розглядається, має механізм, що працює на основі ймовірності та виконує функцію виштовхування. Пріоритетний пакет, що застав усі місця очікування зайнятими у момент обробки іншого пріоритетного пакету, з заданою ймовірністю витісняє з накопичувача один з найменш пріоритетних пакетів та займає його місце. Пакет, що витіснили, втрачається або відправляється назад у чергу. Підсистема балансування завантаження LB, у відповідності до закладених у неї алгоритмів, вилучає задачі із черг Q_w та призначає їх на вільні обчислювальні ядра доцільних серверів.

Для опису механізму звільнення зайнятих трафіком мережевих ресурсів при закінченні передачі трафіка qs-го класу обслуговування, (це відбувається на основі даних, що поступають від протоколу маршрутизації, який підтримує повідомлення про доступну смугу пропуску та доступні ресурси на сервері, наприклад, CSPF, SNMP), введемо змінну LB $\varepsilon_{Net_{lk}}^{qs,t_0}(t)=\{0,1\}$, що вказує на те, що на момент t на сервер перестав потрапляти трафік класу $qs(\varepsilon=1)$, який було прийнято на обслуговування у момент t_0 та повинен був передаватися шляхом $Net_i^{k_i}(t)$ на сервер $Serv_{\text{i}}$. Дана змінна містить усі необхідні дані для визначення мережевих ресурсів, що повинні бути звільнені.

Балансувальник LB в t-й момент характеризується коефіцієнтом втрат $X_{LB}^{qs}(t) \in X$, середнім часом очікування пакета у черзі $T_{LB}^{qs}(t) \in T$. Змінна $X_{LB}^{qs}(t) \in X$ дорівнює проценту втрат на балансувальнику трафіка з класом обслуговування qs, що передається шляхом $Net_{lk}^{k}(t)$ на сервер $Serv_i$ у момент t. Передбачається, що ймовірністю викривлення пакета у тракті можна знехтувати, а втрати відбуваються виключно у балансувальнику через переповнення буферу.

На розмір втрат накладається таке обмеження (3.5):

$$0 \le X_{LB}^{qs}(t),\ \sum_{i=1}^{N} X_{LB}^{qs}(t) \le l_{qs}, \tag{3.5}$$

для усіх вузлів мережі.

Таким чином, з обмеження (1) виходить, що сумарні втрати для трафіка $\lambda_{Net_i}^{qs}(t)$, що маршрутизується у момент t, не повинні перевищувати максимально допустимого значення для даного класу обслуговування l_{qs}. Втрати визначаються як відношення кількості відкинутих даних до кількості отриманих на обслуговування. Мінімізації підлягає значення $X_{LB}^{qs}(t) \to \min$.

Обмеження, що накладаються на час затримки, аналогічні (3.6):

$$0 \le T_{LB}^{qs}(t),\ \sum_{i=1}^{N} T_{LB}^{qs}(t) \le \tau_{qs}, \tag{3.6}$$

де $T_{LB}^{qs}(t)$ – середній час очікування пакета класу обслуговування qs у черзі на вузлі i. Виконання даного обмеження сприяє тому, що час доставки пакетів не перевищить максимально допустимого значення для заданого класу обслуговування τ_{qs}.

3.4 Висновки з розділу 3

1. Представлені результати численного дослідження самоподібних властивостей потоків даних. Для дослідження було використано модельні реалізації трафіка на основі експоненціального перетворення фрактального гауссівського шуму. Було показано, що при розгляді самоподібних властивостей сумарного потоку необхідно приймати до уваги відношення коефіцієнтів варіації сумованих потоків. Значення показника Херста сумарного загального потоку визначається максимальним значенням показника Херста потоків, що підсумовуються, та відношенням коефіцієнта варіації потоку з максимальним показником Херста до інших потоків.

2. Представлені результати численного дослідження зміни фрактальних характеристик мультифрактального потоку при сумуванні потоку, який не має мультифрактальних властивостей. Результати дослідження показали, що фрактальні характеристики мультифрактального потоку зберігаються в залежності від величини відношення сигнал/шум. Зі збільшенням відношення сигнал/шум узагальнений показник Херста сумарного потоку прагне до показника початкового мультифрактального потоку у діапазоні позитивних значень параметра. Якщо адитивний потік не має самоподібних властивостей, мультифрактальні характеристики зберігаються при меншому співвідношенні сигнал/шум.

3. Представлено опис основних особливостей алгоритмів балансування навантаження, аналіз їх переваг та недоліків. Проведено порівняльний аналіз різноманітних алгоритмів балансування навантаження за різними показниками продуктивності, тобто для кожного алгоритму вказані показники ефективності, які в ньому використовуються.

4. Проведена класифікація найчастіше використовуваних алгоритмів балансування навантаження розподілених систем за різними типами. На основі проведеного аналізу класифікаційних типів алгоритмів балансування наванта-

ження, вказані сфера застосування кожного типа алгоритмів, необхідні вимоги роботи типів алгоритмів, визначені недоліки та питання, що потребують вирішення для кожного типу алгоритмів. Таким чином, можна обрати конкретний тип алгоритму балансування навантаження, виходячи із специфіки конкретної виконуваної задачі або проекту та мети, якої планується досягти.

5. У роботі запропоновано математичну модель системи балансування навантаження, у якій балансувальник навантаження описується за допомогою системи масового обслуговування. Стани серверів описуються об’ємом вільного ЦПУ та об’ємом вільної оперативної пам’яті. Усі значення параметрів моделі мають залежність від часу. Така модель дозволяє описувати поведінку розподіленої мережі протягом часу для різноманітних класів обслуговування вхідного трафіку, при заданих обмеженнях на час очікування пакета у черзі та кількість втрачених пакетів.

6. Основні результати описані у роботах [37-39,55,56,129,130,132,133,140, 147].

Список використаних джерел у даному розділі наведено у повному списку використаних джерел під номерами: [13,20,22,26,27,29,31-33,44,49-54,66,71,77-81,84,88,92,105,107,109,153].

РОЗДІЛ 4
РОЗРОБКА ДИНАМІЧНОГО МЕТОДУ БАЛАНСУВАННЯ НАВАНТАЖЕННЯ, ЯКОМУ ПРИТАМАННІ МУЛЬТИФРАКТАЛЬНІ ВЛАСТИВОСТІ

Найбільший інтерес для великомасштабних гетерогенних розподілених мереж являє динамічна адаптивна стратегія балансування з можливістю аналізу поточних змін вхідного трафіка, навантаження у окремих вузлах, на окремих ділянках мережі та у всій системі в цілому, як це видно з класифікації, що наведена у розділі 2.

Широке поширення ідей розподіленої гетерогенної обробки, підкріплене прогресом у мережевих технологіях, дозволяє зробити висновок про безсумнівну актуальність досліджень в області динамічних адаптивних методів балансування навантаження з можливістю прогнозування.

У залежності від кінцевих цілей використовують довгострокові прогнози (макропрогнозування) та короткострокові (мікропрогнозування) [122,150]. Перший тип прогнозів є корисним, коли планується навантаження на достатньо великі періоди часу та є важливою оцінка загальних об'ємів ресурсів з урахуванням довгочасно діючих факторів (особливо помітним це є через вплив так званих «сезонних» коливань). Мікропрогнозування зазвичай виконується на рівні квантифікованих порцій обчислень, критерієм ефективності короткострокових прогнозів можна вважати виграш, що отримується від прогнозу, з урахуванням обчислювальних витрат на виконання власне прогнозування.

Стратегії прогнозування можна поділити на централізовані та децентралізовані. У першому випадку прогноз складається в одній активній точці на основі даних, зібраних з усіх вузлів мережі. У другому випадку кожний вузол окремо, незалежно від інших, займається прогнозуванням змін власного навантаження, і, таким чином, існує множина активних точок.

Можна ввести прогностичну модель розподілу обчислювального навантаження з метою передбачення майбутньої продуктивності вузла мережі для

динамічної адаптації в умовах постійної зміни показника продуктивності. Функція зміни робочого навантаження вузла розподіленої мережі може бути надана у вигляді послідовності значень вибірок, отриманих на основі проміжних вимірювань. Інтегральною характеристикою для усієї системи стане генеральна сукупність таких послідовностей для усіх вузлів мережі.

Альтернативний підхід до створення прогностичної стратегії базується на фрактальному принципі та пов'язаний із синергетичним представленням про інформаційні процеси у складних системах. Тут гіпотеза про нелінійний характер зміні робочого навантаження спирається на фундаментальні положення теорії детермінованого хаосу та теорії динамічних систем.

Припускається, що розподілене середовище відноситься до класу нелінійних динамічних систем, поведінка яких співвідноситься з хаотичним квазівипадковим рухом, що виражається у виникненні в інформаційному обчислювальному середовищі стохастичних флуктуацій під дією дестабілізуючих факторів. Цей феномен визначається Ч.Л. Сайєрсом наступним чином: «Процес характеризується детермінованим хаосом, якщо він генерований повністю детермінованою системою, що виникає як результат хаотично функціонуючих рядів у стандартних часових діапазонах».

На заміну ортодоксальної класичної моделі пропонується фрактальна модель [158], якій притаманні властивості, що більш характерні для адекватної реальності, оскільки лінійні методи недостатньо добре описують складні нелінійні процеси. Фрактальний аналіз динаміки процесів зміні навантаження зводиться до виявлення у часових траєкторіях фрактальних структур з масштабною інваріантністю. Знайдена фрактальна самоподібність ймовірнісних траекторій може бути використана для прогнозування динаміки навантаження та підбору оптимальних параметрів балансування інформаційного трафіку у розподіленому середовищі.

У даній роботі пропонується динамічний метод балансування, що враховує мультифрактальні властивості вхідного навантаження.

4.1 Розробка динамічного методу балансування мультифрактального трафіка

На основі мультифрактальних властивостей вхідного трафіка пропонується динамічний метод балансування трафіку. В залежності від змін параметрів вхідного потоку використовуються різноманітні методи управління трафіком. Приведемо поетапний опис динамічного методу балансування навантаження:

Потоки, що надходять, підсумовуються та обробляються у відповідності до політики обробки черг, створюють єдиний потік, що має характеристики $V = \{\lambda, h, \mu_{qs}\}$ (див. п. 3.3).

1. У трафіку, що поступає на балансувальник, виділяється вікно X, фіксованої довжини T.

2. Знаходимо за трафіком інтенсивність, вибіркове значення функції узагальненого показника Херста $h(q)$, значення параметра Херста $H = h(2)$ та діапазон значень узагальненого показника Херста $\Delta h = h(q_{\min}) - h(q_{\max})$ для ділянки трафіка у виділеному вікні;

3. Проводимо збір та аналіз статистичної інформації за серверами та каналами: середній коефіцієнт використання пропускної здатності каналу $Net_i^{k_i}$ за період часу T, стани серверів $CPU_i^{n_i}$, $RAM_i^{m_i}$, середнє завантаження ЦПУ та середній коефіцієнт використання пам'яті i-го сервера за період часу T.

4. На основі мультифрактальних властивостей трафіка та значень трудомісткості запитів обчислюємо набір векторів необхідних ресурсів $\mu_{qs}^{new} = (CPU, Net, RAM)$ для кожного qs-го класу трафіка.

$$\mu_{qs}^{new} = \begin{cases} \mu_{qs}, & H \le 0,5; \\ \mu_{qs} + (H - 0.5)\mu_0, & 0.5 < H < 0.9,\ \Delta h \le 0.4; \\ \mu_{qs} + (H - 0.5)(\Delta h - 0.4)\mu_0, & 0.5 < H < 0.9,\ 0.4 < \Delta h < 1; \\ \mu_{qs} + \mu_0, & H \ge 0.9 \text{ or } H > 0.5,\ \Delta h \ge 1, \end{cases}$$

де μ_{qs} визначається у відповідності до класу обслуговування та необхідними ресурсами, значення μ_0 обирається адміністратором мережі з урахуванням стану мережі. Щоб відобразити зміну мультифрактальних властивостей потоків, вектори потрібних ресурсів μ_{qs}^{new} оновлюються у регулярні проміжки часу та перераховуються за формулою.

Кількість потрібних ресурсів не змінюється ($\mu_{qs}^{new} = \mu_{qs}$), якщо трафік є звичайним пуассонівським потоком ($H = 0.5$) або має антиперсистентні властивості ($H < 0.5$). При значенні $0.5 < H < 0.9$ та малому розкиді даних ($\Delta h \le 0.4$) значення μ_{qs} збільшується пропорційно значенню показника Херста. При значенні показника Херста $0.5 < H < 0.9$ та великому розкиді даних ($0.4 < \Delta h < 1$) значення μ_{qs} збільшується пропорційно обом характеристикам. Кількість потрібних ресурсів з максимальним значенням $\mu_{qs} + \mu_0$ отримується при значенні $H \ge 0.9$ або при персистентному трафіку ($H > 0.5$) з діапазоном значень узагальненого показника Херста $\Delta h \ge 1$. Після перерахування вартості усіх шляхів повідомлення про стан шляхів розсилається поміж маршрутизаторами.

5. Проводимо розрахунок розподілу потоків за серверами на основі перерахованих потоків за серверами на основі перерахованих значень трудомісткості, інтенсивності трафіка, стану завантаженості серверів та каналів зв’язку.

6. На основі отриманих даних визначається завантаженість серверів на наступному етапі.

7. Розподіляємо трафік за серверами, згідно алгоритму балансування у межах кожного класу потоку.

8. Якщо увесь трафік не вдалося розподілити, то проводимо розподіл трафіку, що залишився, серед кількості ресурсів, що маються в наявності $CPU_i^u(T)$, $RAM_i^r(T)$, $Net_i^k(T)$. Переоцінка не враховується алгоритмом, тому що не вносить суттєвих змін.

9. Проводимо збір даних про завантаженість серверів $CPU_i^u(T)$, $RAM_i^r(T)$, $Net_i^k(T)$ та передачу їх у систему балансування навантаження для розрахунку нового розподілу потоків.

10. Зсуваємо вікно X довжини T вперед на задану величину зсуву ΔT; здійснюємо аналіз трафіка та розрахунок наступного значення завантаженості серверів.

Метод балансування навантаження, що розробляється, повинен забезпечувати статично рівномірний розподіл навантаження на серверах, високі показники продуктивності, пропускної здатності, відмовостійкості (автоматично знаходячи збої вузлів та перерозподіляючи потік даних серед тих, що залишилися) та низький час відгуку, кількість службової інформації, кількість загублених даних.

Алгоритм балансування повинен розподіляти запити за серверами таким чином, щоб відхилення завантаженості серверів від середнього значення було мінімальним (мінімальне значення величини комплексного дисбалансу).

4.2 Опис моделі системи балансування

Система балансування навантаження, що розглядається, складається з групи серверів та балансувальника навантаження [32, 33, 105]. Надана на рисунку 4.1 схема системи балансування навантаження, побудована на основі підсистеми балансування навантаження та підсистеми управління і моніторингу, що тісно взаємодіють одна з одною:

– підсистема балансування навантаження: алгоритм балансування навантаження, інформація про поточний стан системи, гнучкі налаштування QoS, динамічний розподіл трафіка за різними каналами зв’язку та вузлами у залежності від їх поточного стану, ступеню завантаження, адміністративних політик балансування навантаження;

– підсистема управління та моніторингу: збір та аналіз статистики про поточний стан системи, знаходження мультифрактальних властивостей вхідного потоку даних, розрахунок розподілу потоків за вузлами мережі з урахуванням класифікації трафіку та завантаженості серверів і каналів зв’язку.

–

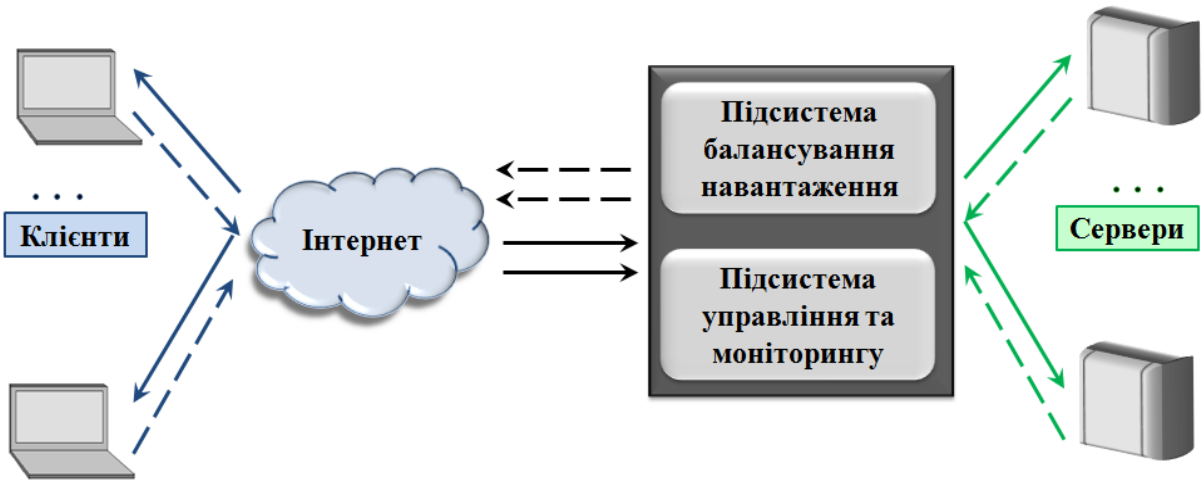


Рисунок 4.1 – Схема системи балансування навантаження

4.3 Взаємодія програмних компонентів системи балансування навантаження

Система балансування складається з набору взаємопов’язаних компонентів, представлених на рис. 4.2: активного балансувальника, резервного балансувальника та групи серверів [82,92,108,138].

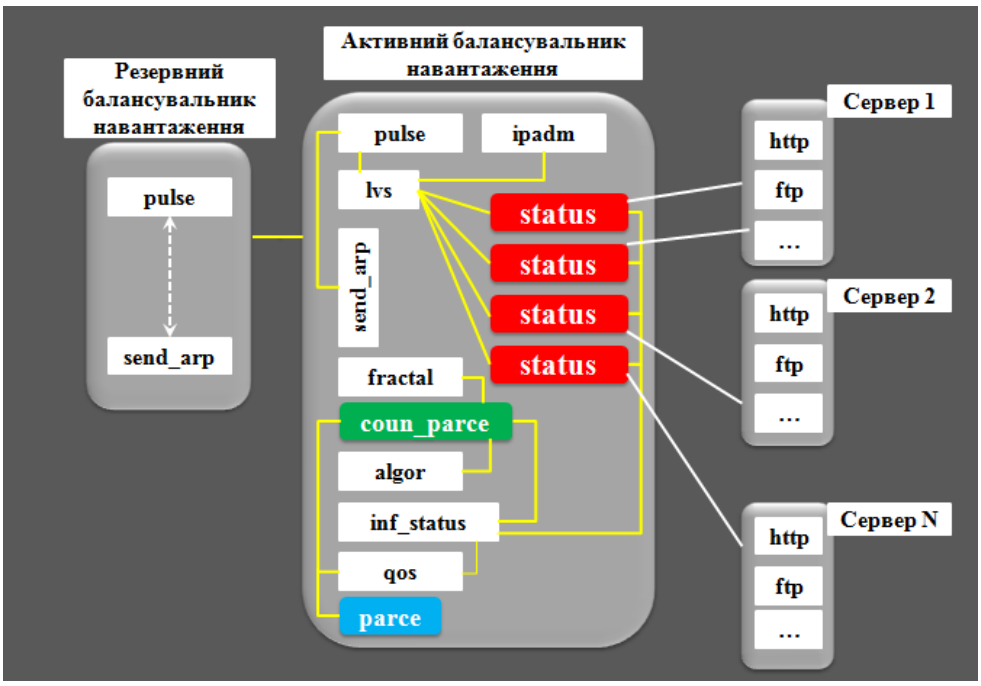


Рисунок 4.2 – Компоненти системи балансування навантаження

Балансувальник навантаження має веб-інтерфейс для відстеження, налаштування та адміністрування розподілу навантаження.

Пропонований метод балансування навантаження заснований на використанні комплексу методів внутрішнього та зовнішнього моніторингу. Використання внутрішнього моніторингу системи дозволяє періодично тестувати мережу для визначення найнавантаженіших сегментів.

Використання внутрішнього моніторингу стану обчислювального вузла дозволяє отримати об’єктивну картину завантаження вузла та дані про завантаження окремих компонентів вузла. Критеріями завантаження обчислювального вузла передбачається вважати завантаження процесора, пам’яті та пропускної здатності каналу для потоків різних класів обслуговування. Щоб враховувати потенційну обчислювальну потужність кожного ресурсу вузла, пропонується використовувати вагові коефіцієнти. Даний метод враховує мультифрактальні властивості вхідного інформаційного потоку. Використання комплексного підходу при балансуванні навантаження дозволить домогтися збільшення продуктивності розподіленої обчислювальної системи.

Процес *pulse* – це керуючий процес, що на активному балансувальнику запускає процес *lvs*, а на резервному балансувальнику буде відстежувати стан активного, періодично опитуючи його. Якщо активний балансувальник не відповідає протягом заданого періоду часу, буде ініційовано процес передачі його функцій резервному балансувальнику. При цьому *pulse* на резервному балансувальнику відправляє процесу *pulse* на активному балансувальнику команду зупинки усіх служб *lvs*, запускає *send_arp* для присвоєння віртуальних IP-адрес MAC-адресу резервного балансувальника та запускає процес *lvs*. Процес *send_arp* розсилає широкосповіщувальні пакети ARP при переході віртуальної IP-адреси від одного вузла іншому [24].

Процес *lvs* запускається на активному балансувальнику за викликом *pulse*. Він визиває службу *ipadm* для створення, додавання, зміни та видалення записів у таблиці маршрутизації IP. Процес *lvs* запускає процес *status* для кожної налаштованої служби розподілу навантаження. Якщо *status* повідомляє про те, що реальний сервер вимкнений, *lvs* змусить утиліту *ipadm* видалити цей сервер з таблиці IP. Основним призначенням процесу *status*, працюючого на активному балансувальнику, є слідкування за навантаженням серверів, збір та аналіз статистики про поточний стан системи та інтенсивність трафіку. Процес *status* передає оброблену та проаналізовану інформацію у підсистему балансування навантаження процесу *inf_status*. Процес *inf_status* у свою чергу передає інформацію про поточний стан системи процесу *coun_parce.*

Процес *fractal* проводить розрахунок мультифрактальних властивостей кожного вхідного потоку даних та передає цю інформацію процесу *coun_parce*. Процес *algor* обирає алгоритм балансування навантаження і також передає цю інформацію процесу *coun_parce*. Процес *coun_parce* проводить розрахунок розподілу потоків за вузлами мережі з урахуванням класифікації трафіку, завантаженості серверів і каналів зв’язку, дисбалансу серверів та усієї системи. Результати розрахунку передаються процесу *parce*, що здійснює динамічний розподіл трафіку за різноманітними каналами зв’язку в залежності від їх поточного стану. Також результати розрахунку процесу *coun_parce* передаються

процесу *qos* для гнучкого налаштування забезпечення Якості обслуговування у відповідності до методів управління трафіком (управління пропускною здатністю каналів та пам'яті, продуктивністю процесів, кеш-пам'яті), якщо це є необхідним. У випадку застосування методів управління трафіком процесом *qos*, інформація про зміни передається процесу *inf_status*.

Для проведення імітаційного моделювання було прийнято рішення розробити програмний продукт, який задовольнив би поставленим потребам. Для цього було проведено порівняльний аналіз систем моделювання розподілених систем.

4.4 Порівняння систем моделювання розподілених систем

Проведемо порівняння наступних інтерпретованих мов програмування: Python, Java, JavaScript, Perl, Tcl, Smalltalk, C++, Common Lisp и Scheme [89].

Java.

Зазвичай очікується, що Python-програми виконуються повільніше, ніж програми Java, проте вони у той же час потребують набагато менше часу для розробки. Python-програми типово повільніші у 3-5 разів, ніж еквівалентні Java-програми. Ця різниця може бути пояснена за рахунок вбудованих високорівневих типів даних Python, та його динамічної типізації. Наприклад, Python-програміст не витрачає часу, описуючи типи аргументів або змінних, а потужні типи поліморфних списків та словників Python, для яких багата синтаксична підтримка вбудована безпосередньо у саму мову, можуть знайти застосування майже у кожній Python-програмі. Через типізування у час виконання, Python повинен виконувати більше роботи, ніж Java.

Наприклад, при обробці виразу a+b, він повинен спочатку дослідити об'єкти a та b, щоб зрозуміти їх типи, які не відомі під час компіляції. Після цього визивається відповідна операція додавання, яка може бути перевантаженим користувальницьким методом. Java, з іншого боку, може виконувати ефективне додавання цілих або чисел із плаваючою крапкою, але потребує опису

змінних a и b, і не дозволяє перевантажити оператор + для екземплярів класів, визначених користувачем.

Через ці причини Python набагато більше підходить як мова, що «клеїть», у той час як Java краще характеризується як низькорівнева мова для реалізації. Фактично, вони разом можуть створити відмінну пару. Компоненти можна реалізувати на Java, а після цього використовувати у додатках Python. Python також корисно використовувати для прототипів компонентів, поки їх розробка не «затвердішає» у Java-реалізації. Для підтримки такого типу розробки створюється реалізація Python, що написана на Java. Вона дозволяє викликати Python-код із Java та навпаки. У цій реалізації вихідний код Python транслюється у байт-код Java (за допомогою бібліотеки часу виконання, для підтримки динамічної семантики Python).

Javascript.

«Об’єктно-заснована» частина Python приблизно еквівалентна JavaScript. Подібно до JavaScript (та на відміну від Java), Python підтримує стиль програмування, що використовує прості функції та змінні без включення у визначення класу. Але для JavaScript це все, що наявне. Python, з іншого боку, підтримує написання набагато більш об’ємних програм, і краще повторне використання коду через дійсно об’єктно-орієнтований стиль програмування, у якому класи та наслідування грають важливу роль.

Perl.

Python та Perl родом зі схожих оточень (сценарії Unix, які обидва значно переросли) та несуть багато схожих особливостей, проте мають різну філософію. Perl націлений на підтримку загальних програмно-орієнтованих задач, наприклад, має вбудовану обробку регулярних виразів, сканування файлів та генерування звітів. Python концентрується на загальних методологіях програмування, таких як розробка структур даних та об’єктно-орієнтоване програмування, сприяє написанню легко підтримуваного коду, шляхом надання елегантної, але не занадто зашифрованої нотації. Як наслідок, Python близько

підходить до Perl, але рідко перемагає у його оригінальній ніші додатків; однак, Python має гарну застосовуваність за кордонами ніші Perl.

Tcl.

Подібно до Python, Tcl є корисним як мова розширення додатків, так і в якості незалежної мови програмування. Однак Tcl, що традиційно зберігає усі дані як строки, володіє скудними структурами даних, а виконує типічний код набагато повільніше, ніж Python. Tcl також не вистачає особливостей, необхідних при написанні великих програм, таких як модульовані простори імен. У той час як «типові» великі додатки, що використовують Tcl, зазвичай складаються з розширень, написаних на C або C++, еквівалентні Python-додатки часто можуть бути написані на «чистому» Python. Безумовно, розробка на чистому Python здійснюється набагато швидше, ніж при написанні та адмініструванні C або C++ компонентів. Було сказано, що одним з якісних розробок на Tcl є пакет Tk. Python пристосував інтерфейс Tk у якості своєї бібліотеки стандартних компонентів GUI. Tcl 8.0 зачіпає питання швидкості, надаючи байт-код компілятору з обмеженою підтримкою типів даних, та додає простори імен. Однак він все ще залишається занадто громіздкою мовою програмування.

Smalltalk.

Можливо найбільша відмінність між Python та Smalltalk складається у Python-синтаксисі «основного потоку», який дає гілку підготовки програмістів. Як і Smalltalk, Python має динамічну типізацію та динамічне зв’язування, усе у Python являє собою об’єкт. Однак, Python відрізняє вбудовані об’єктні типи від класів, визначених користувачем, та до даного часу не допускає наслідування із вбудованих типів. Стандартний набір бібліотеки типів даних Smalltalk більш очищений, тоді як бібліотека Python має більше засобів для роботи з Internet та WWW, наприклад, з e-mail, HTML та FTP. Python має відмінну від інших філософію стосовно середовища розробки та розподілу коду. Там, де Smalltalk за традицією має монолітний «системний образ», що включає в себе як середо-

вище, так і програму користувача, Python зберігає стандартні модулі та модулі користувача в індивідуальних файлах, які можуть легко бути налаштованими знову або поширені за межами системи. Як наслідок, існує більше одного вибору при використанні графічного інтерфейсу користувача (GUI) у Python-програмі, оскільки GUI не вбудований у систему.

C++.

Майже все сказане для Java, також застосовне до мови C++, тим більш, що у випадках, коли код Python у 3–5 разів коротший, ніж еквівалентний код Java, часто він у 5-10 разів коротший еквівалентного коду C++. Анекдотичне підтвердження каже: те, що один програміст Python може завершити за два місяці, два програмісти C++ не зможуть зробити і за рік. Python блискуче використовується як клей, що поєднує компоненти, написані на C++.

Common Lisp і Scheme.

Ці мови близькі до Python у їхній динамічній семантиці, проте настільки відрізняються у їх підході до синтаксису, що їх порівняння стає майже релігійним аргументом: недоліки синтаксису Lisp – це перевага чи недолік? Слід зауважити, що Python має інтроспективні можливості, подібні до Lisp, програми Python можуть створювати та виконувати програмні фрагменти у процесі. Зазвичай, усе вирішують властивості реального світу: Common Lisp занадто великий (у будь-якому сенсі), а світ Scheme фрагментується між багатьма несумісними версіями, тоді як Python має єдину безплатну компактну реалізацію.

Таким чином, для розробки програмного продукту було обрано мову програмування Python.

Python – це високорівнева мова програмування загального призначення. Синтаксис ядра Python є мінімалістичним. У той же час стандартна бібліотека включає у себе великий об’єм корисних функцій.

Python підтримує декілька парадигм програмування, у тому числі структурне, об’єктно-орієнтоване, функціональне, імперативне та аспектно-орієнтоване програмування. Основні архітектурні особливості – динамічна

типізація, автоматичне управління пам’яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень та високорівневі структури даних.

Еталонною реалізацією Python є інтерпретатор CPython, що підтримує більшість активно використовуваних платформ. Python – мова програмування, що активно розвивається, нові версії (з додаванням / зміною мовних властивостей) виходять приблизно раз у два з половиною роки. Внаслідок цього та деяких інших причин на Python відсутній стандарт ANSI, ISO або інші офіційні стандарти, їх роль виконує CPython.

4.5 Опис розробленого програмного продукту

Для проведення імітаційного моделювання балансування навантаження розподіленої системи з вхідним мультифрактальним потоком було створено програмний продукт, написаний на мові Python. Даний програмний продукт дозволяє проводити імітаційне моделювання роботи системи балансування навантаження за допомогою різних алгоритмів балансування, використовуючи запропонований динамічний метод балансування навантаження.

На вхід системи поступає згенерований мультифрактальний трафік, описаний у розділі 3. Запити, що поступають із зовнішньої мережі, створюють адитивний мультифрактальний трафік та направляються на балансувальник, який у свою чергу регулює потік завдань за допомогою обраної політики балансування, що віддаються серверам. Також у системі присутній Secondary Load Balancer, що забезпечує відмовостійкість системи, відновлюючи балансувальник, якщо той не витримав навантаження. Використання даної структури дозволяє розподілити навантаження поміж серверами через взаємодію компонентів програми один з одним.

У ході експериментів було виявлено, що дані слабо змінюються при збільшенні кількості кластер них вузлів та кількості серверів у них. Тому для еко-

номії часу генерації трафіка та розрахунків було обрано два кластера з кількістю серверів у кожному з них рівним шести. В силу того, що кластери зазвичай є неоднорідними і у них стоять сервері різної продуктивності, було обрано наступні параметри серверів у кожному з кластерів:

$$\mu_{1n}\left(CPU_{1n}, RAM_{1n}, Net_{1n}\right), \mu_{2n}\left(CPU_{2n}, RAM_{2n}, Net_{2n}\right), n = \overline{1,6},$$

де $CPU_{1,2n} = 400, RAM_{1,2n} = 450, Net_{1,2n} = 300$, при $n = 1,3,5$;

$CPU_{1,2n} = 300, RAM_{1,2n} = 350, Net_{1,2n} = 250$, при $n = 2,4$;

$CPU_{16} = 500, RAM_{16} = 550, Net_{16} = 350$;

$CPU_{26} = 600, RAM_{26} = 650, Net_{26} = 400$.

У ході імітаційного моделювання було проведено експерименти на основі роботи чотирьох основних алгоритмів балансування навантаження: Least connection, Round Robin Weight, Connection mechanism та Compare Balance. В алгоритмі Round Robin Weight кожному вузлу присвоюється ваговий коефіцієнт у відповідності до його продуктивності та потужності. Це допомагає розподіляти навантаження гнучкіше, тому що вузли з великою вагою обробляють більше запитів. Алгоритм Least connection враховує завантаження мережі та кількість активних з'єднань з кожним сервером. Алгоритм балансування Connection mechanism базується на механізмі найменшої кількості з'єднань кожного сервера та динамічній оцінці навантаження. Балансувальник навантаження записує кількість з'єднань кожного серверу. Алгоритм Compare Balance використовується для досягнення рівноважного стану та управління незбалансованим навантаженням системи. У цьому алгоритмі балансувальник випадковим чином обирає сервери та порівнює їх навантаження, після чого відправляє заявку на обробку сервера з меншим завантаженням.

Для аналізу роботи алгоритмів та методів балансування було проведено чисельні дослідження роботи системи балансування при різноманітних значен-

них параметрів мультифрактального трафіку: діапазон значень узагальненого показника Херста $1{,}5 \le \Delta h \le 6$, значення параметра Херста $0{,}6 \le H \le 0{,}9$, інтенсивність потоку заявок $0{,}5 \le \lambda \le 1$.

4.6 Результати імітаційного моделювання

На рис. 4.3–4.4 показано зміну дисбалансу процесорів, пам'яті та пропускної здатності у випадку 2-х кластерів, а також комплексне значення дисбалансу навантаження кожного кластеру (див. формулу (3.7), розділ 3) при роботі алгоритму балансування Compare Balance. Параметри a, b, c, які означають вагові коефіцієнти для процесора, пам'яті та пропускної здатності мережі, було обрано рівнозначними.

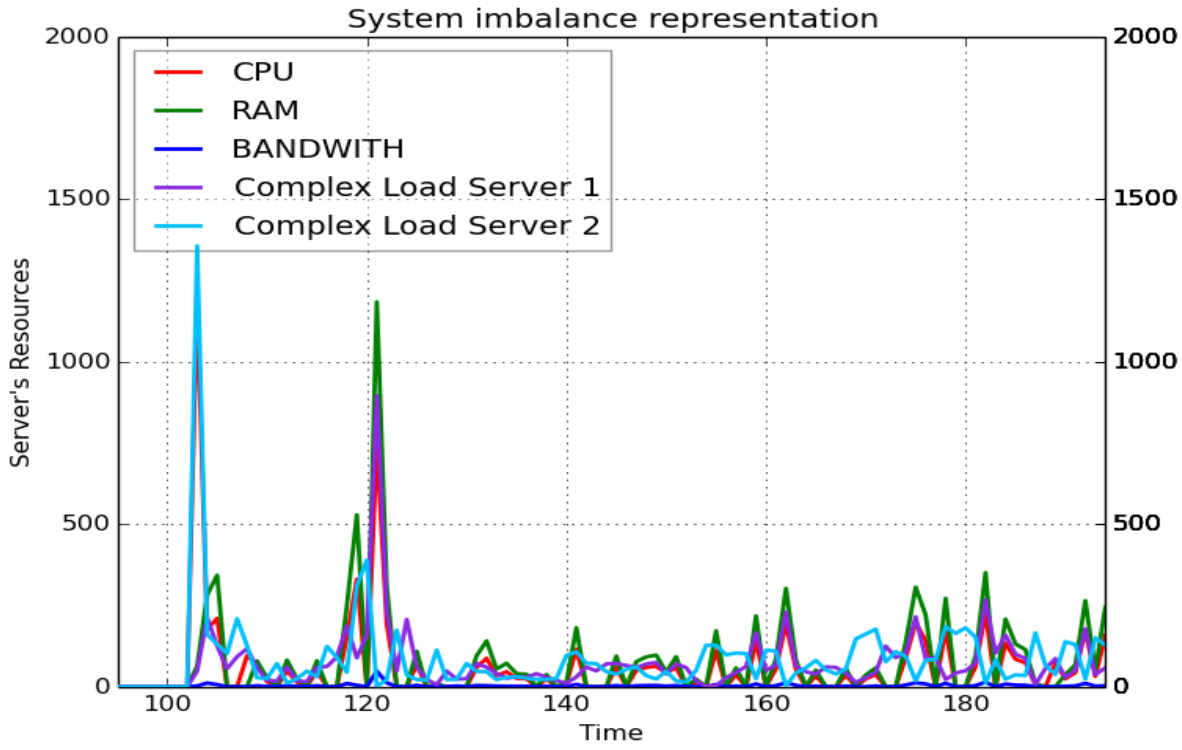


Рисунок 4.3 – Дисбаланс системи при параметрах трафіка $H = 0.6$ і $\Delta h = 1.5$

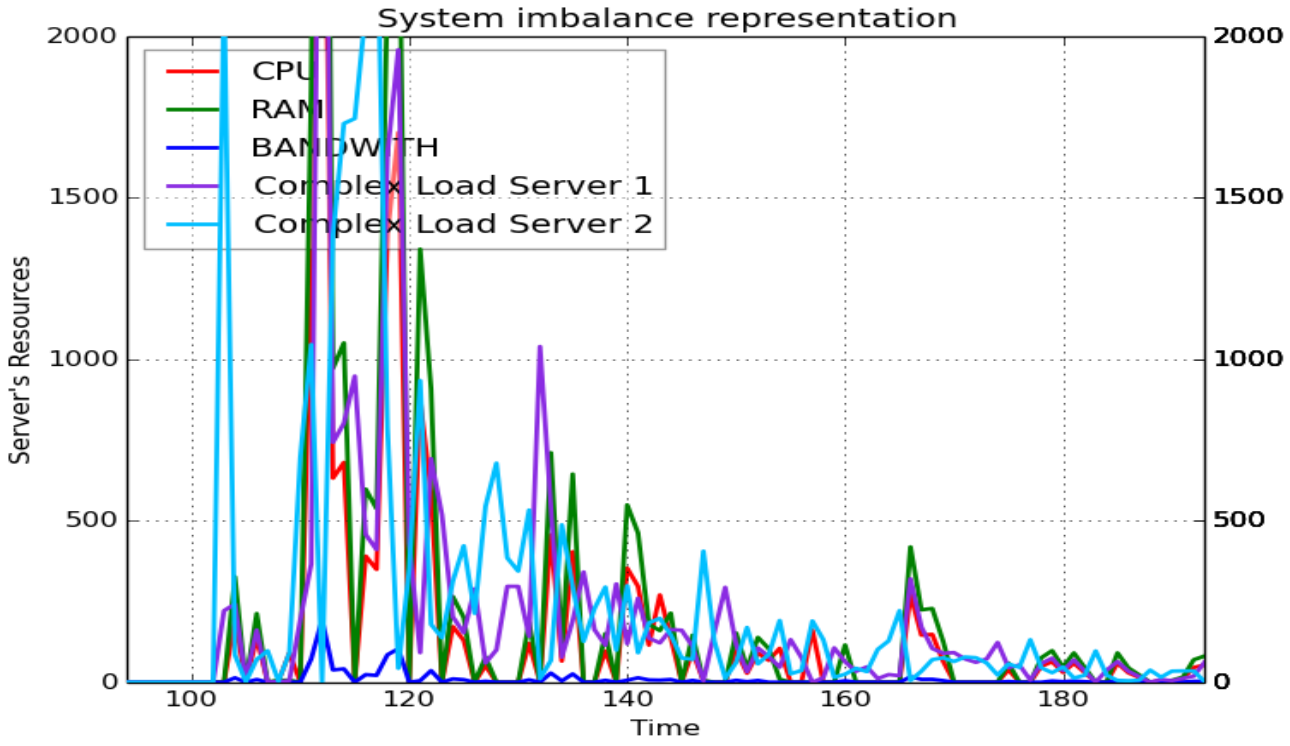


Рисунок 4.4 – Дисбаланс системи при параметрах трафіка $H = 0.8$ і $\Delta h = 1.5$

У першому випадку (див. рис. 4.3) на балансувальник надходить згенерований трафік з параметром $H = 0.6$ та діапазоном узагальненого показника Херста $\Delta h = 1.5$. Рис. 4.4 демонструє дисбаланс системи для трафіку з більшою довгостроковою залежністю ($H = 0.8$) і тією ж неоднорідністю, що і у першому випадку.

З рисунку 4.5 видно, що перші 230 секунд роботи система знаходиться у нестійкому стані та ресурси серверів використовуються неефективно, починаючи з 230 секунди роботи, дисбаланс системи починає зменшуватися, зменшується і середнє значення використання ресурсів. А починаючи з 400 секунди роботи, система приходить до стабільного стану та значення дисбалансу практично не змінюється.

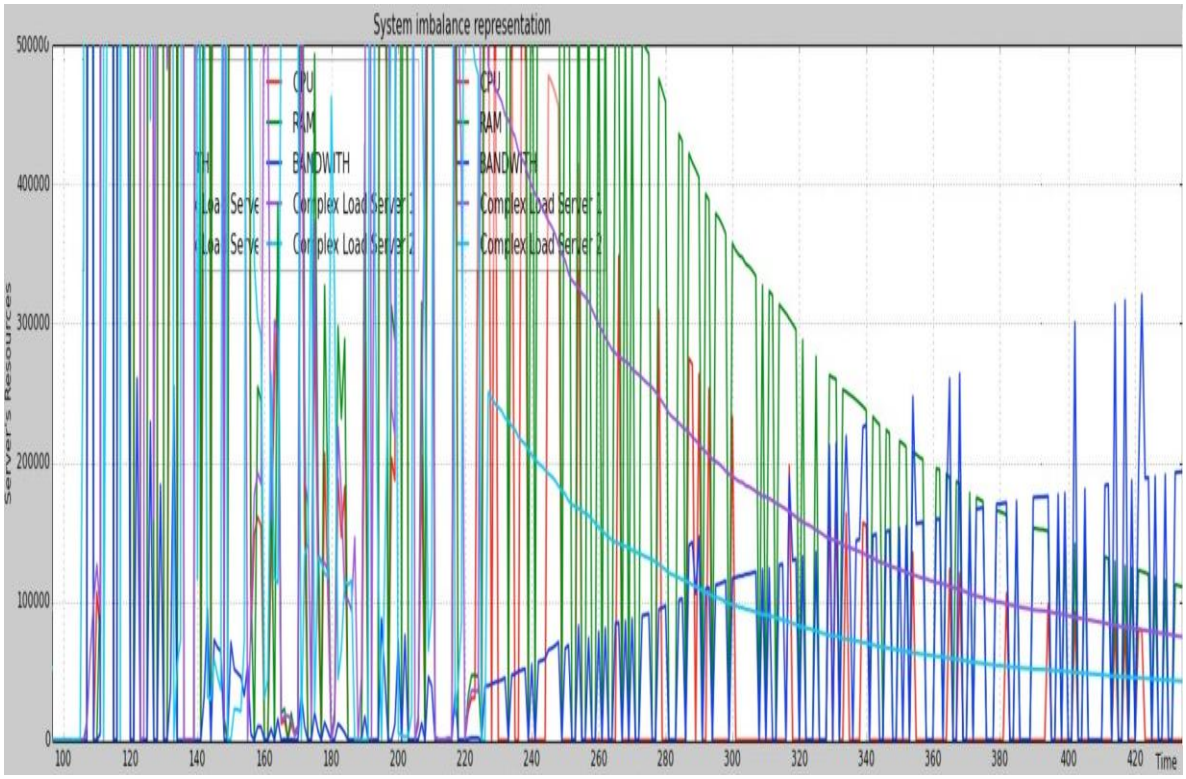


Рисунок 4.5 – Дисбаланс системи при параметрах трафіка H=0.7 і Δh=2

На рис. 4.6 надано результати моделювання роботи балансувальника у випадку великих значень як параметра Херста $H = 0.9$, так і діапазону узагальненого показника Херста $\Delta h = 2.5$.

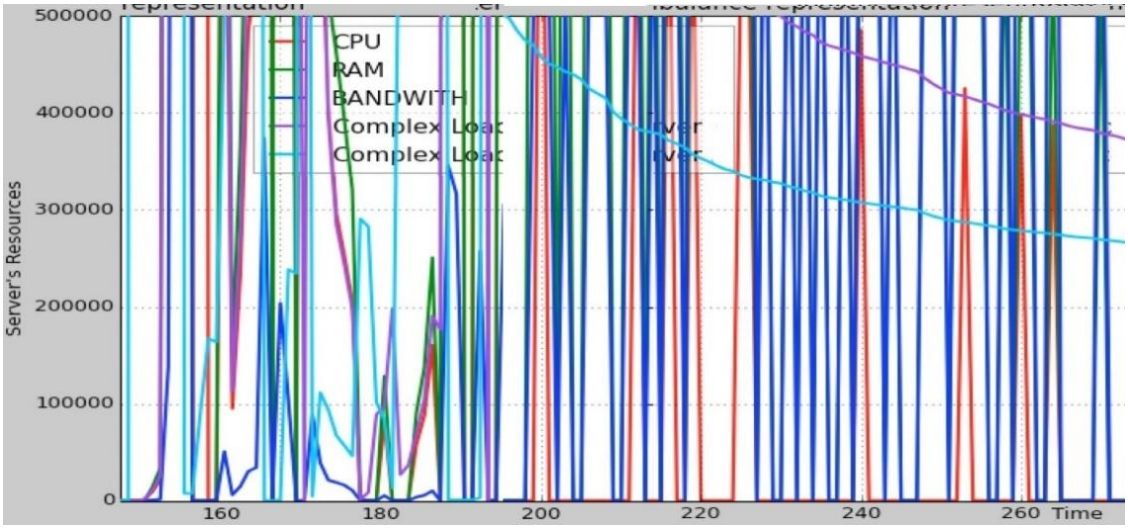


Рисунок 4.6 – Дисбаланс системи при параметрах трафіка $H = 0.9$ і $\Delta h = 2.5$

На рисунку 4.7 зображений графік затримки запиту у системі при роботі алгоритму Round Robin Weight для часового проміжку у 100 секунд.

Кількість втрачених пакетів при балансуванні складає $X_{LB} = 1,64\%$.

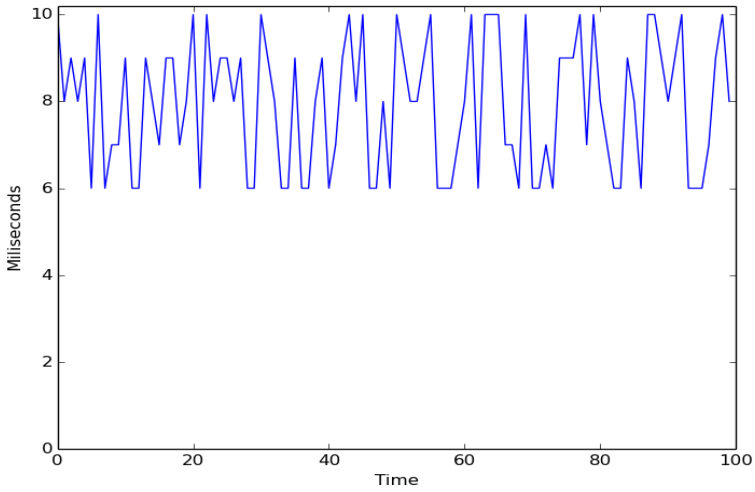


Рисунок 4.7 – Графік затримки запиту в системі при роботі алгоритму балансування Round Robin Weight

На рисунку 4.8 зображений графік затримки запиту у системі при роботі алгоритму Least connection для часового проміжку у 100 секунд.

Кількість втрачених пакетів при балансуванні складає $X_{LB} = 1,23\%$.

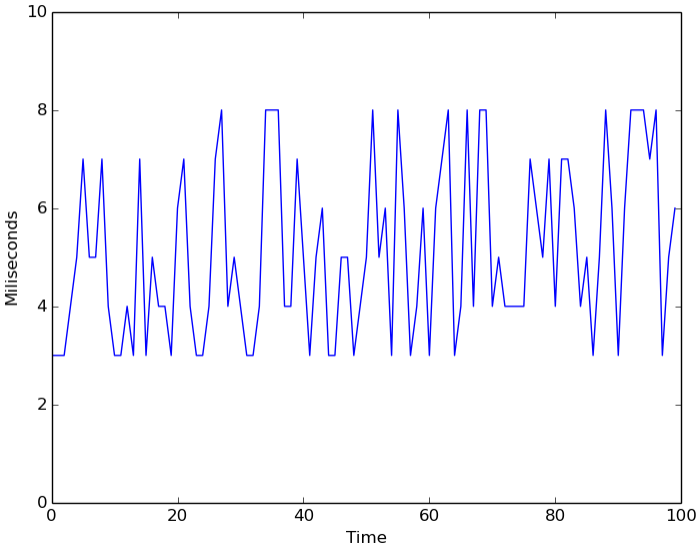


Рисунок 4.8 – Графік затримки запиту в системі при роботі алгоритму балансування Least connection

На рисунку 4.9 зображений графік затримки запиту в системі при роботі алгоритму Connection mechanism для часового проміжку в 100 секунд.

Кількість втрачених пакетів при балансуванні складає $X_{LB} = 1{,}02\%$.

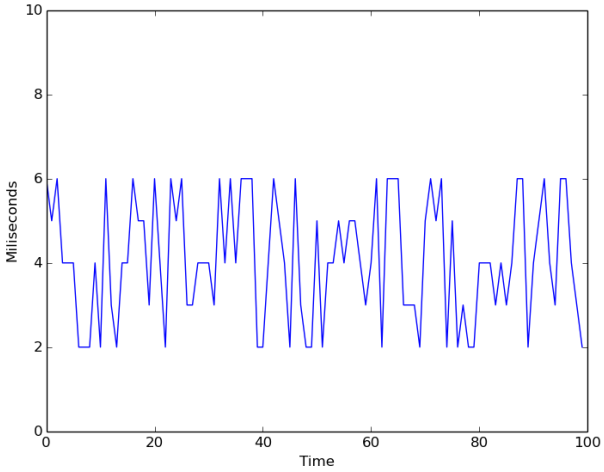


Рисунок 4.9 – Графік затримки запиту в системі при роботі алгоритму балансування Connection mechanism

На рисунку 4.10 зображений графік затримки запиту в системі при роботі алгоритму Compare Balance для часового проміжку в 100 секунд.

Кількість втрачених пакетів при балансуванні скдадає $X_{LB} = 0{,}98\%$.

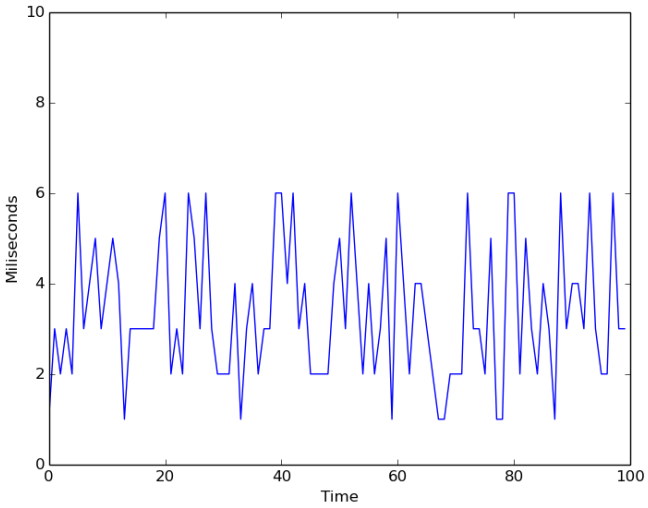


Рисунок 4.10 – Графік затримки запиту в системі при роботі алгоритму балансування Connection mechanism

В таблиці 4.1 надано порівняльні характеристики роботи алгоритмів балансування: кількість втрачених даних та середній час очікування заявок у розподіленій системі.

Таблиця 4.1 – Порівняльні характеристики алгоритмів балансування

Параметри	Round Robin Weight	Least connection	Connection mechanism	Compare Balance
Втрачені дані, %	1,64	1,23	1,02	0,98
Середній час очікування, мс	8	6	4	4,1

Розглянемо завантаження кожного кластеру при роботі алгоритму балансування Round Robin Weight.

На рисунках 4.11 и 4.12 зображені графіки завантаження ресурсів кластерів «1» та «2» відповідно, на яких видно, що трафік, передаючись по черзі на кожний кластер, у результаті привів до переповнення використання ресурсів кластера «2». Із цього слідує, що алгоритм Round Robin Weight не підходить для оптимального балансування мультифрактального трафіка.

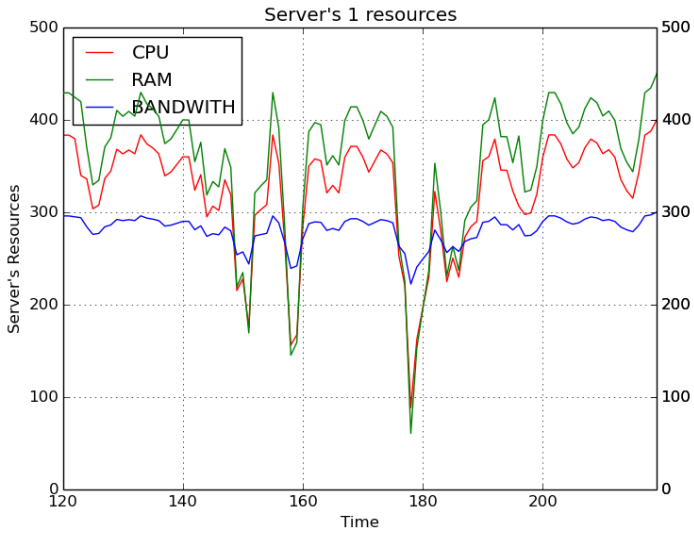


Рисунок 4.11 – Графік завантаженості ресурсів кластера «1» при роботі алгоритму балансування Round Robin Weight

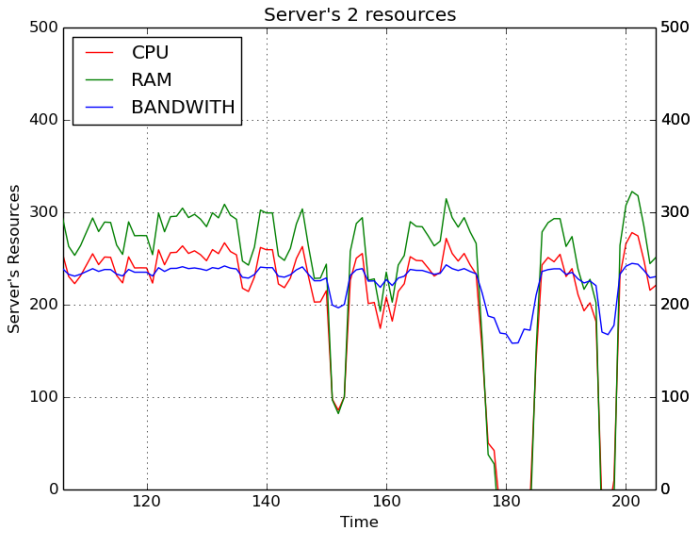


Рисунок 4.12 – Графік завантаженості ресурсів кластера «2» при роботі алгоритму балансування Round Robin Weight

Розглянемо завантаження кожного сервера при роботі алгоритму Least connection.

На рисунках 4.13 и 4.14 зображено графіки завантаження ресурсів кластерів «1» и «2» відповідно, на яких видно, що трафік, передаючись по черзі на сервери з меншою кількістю активних з’єднань, що отримують більше запитів, у результаті привів до перевантаження у першому кластері.

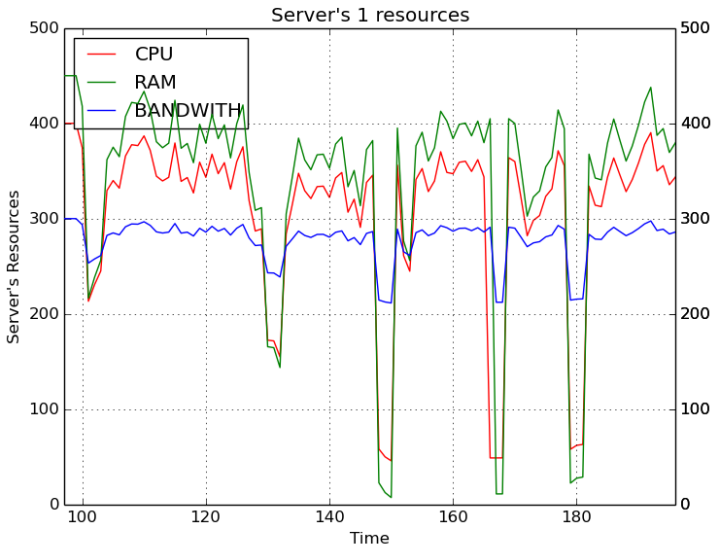


Рисунок 4.13 – Графік завантаженості ресурсів кластера «1» при роботі алгоритму Least connection

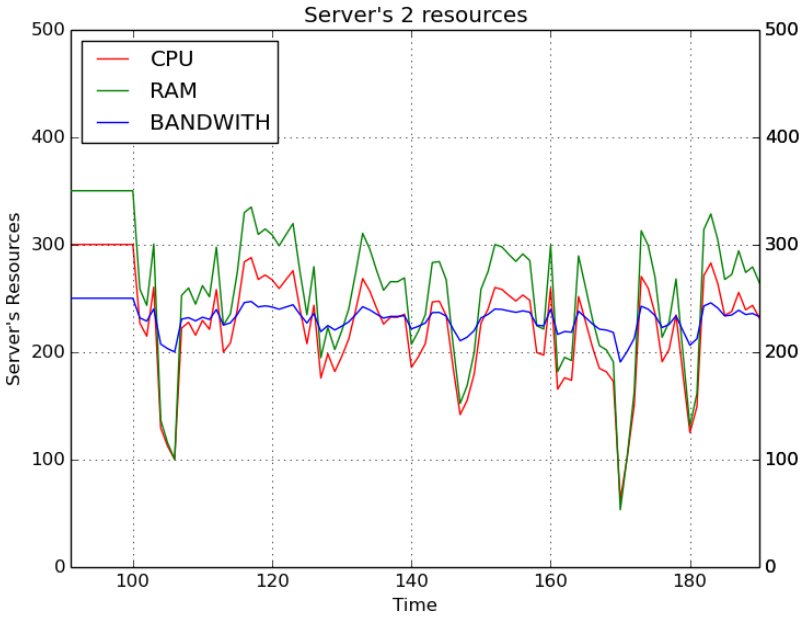


Рисунок 4.14 – Графік завантаженості ресурсів кластера «2»
при роботі алгоритму Least connection

Розглянемо завантаження кожного сервера при роботі алгоритму Connection mechanism. На рисунках 4.15 и 4.16 зображені графіки завантаження ресурсів кластерів «1» и «2» відповідно, на яких видно, що алгоритм привів до більшої завантаженості кластера «1».

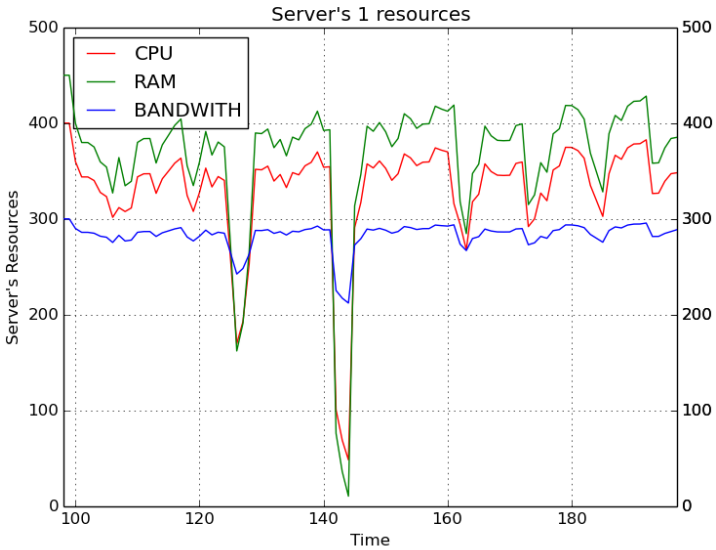


Рисунок 4.15 – Графік завантаженості ресурсів кластера «1»
при роботі алгоритму Connection mechanism

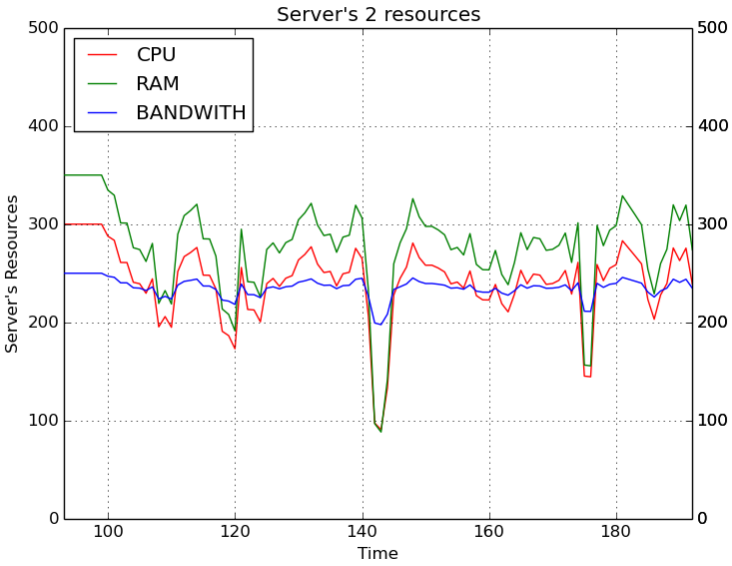


Рисунок 4.16 – Графік завантаженості ресурсів кластера «2»
при роботі алгоритму Connection mechanism

Розглянемо завантаження кожного кластера при роботі алгоритма Compare Balance.

На рисунках 4.17 и 4.18 зображені графіки завантаження ресурсів серверів «1» и «2» відповідно, на яких видно, що трафік, передаючись на сервери з урахуванням завантаженості кожного сервера, у результаті привів до найбільш збалансованого навантаження на кожному кластері.

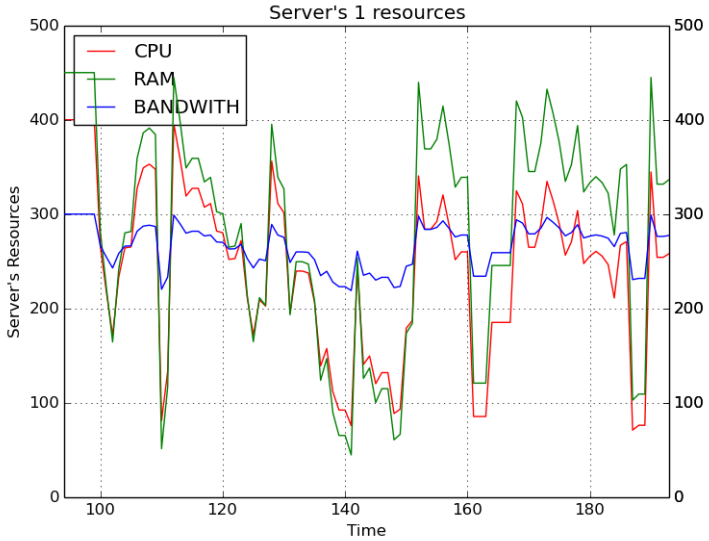


Рисунок 4.17 – Графік завантаженості ресурсів кластера «1»
при роботі алгоритму Compare Balance

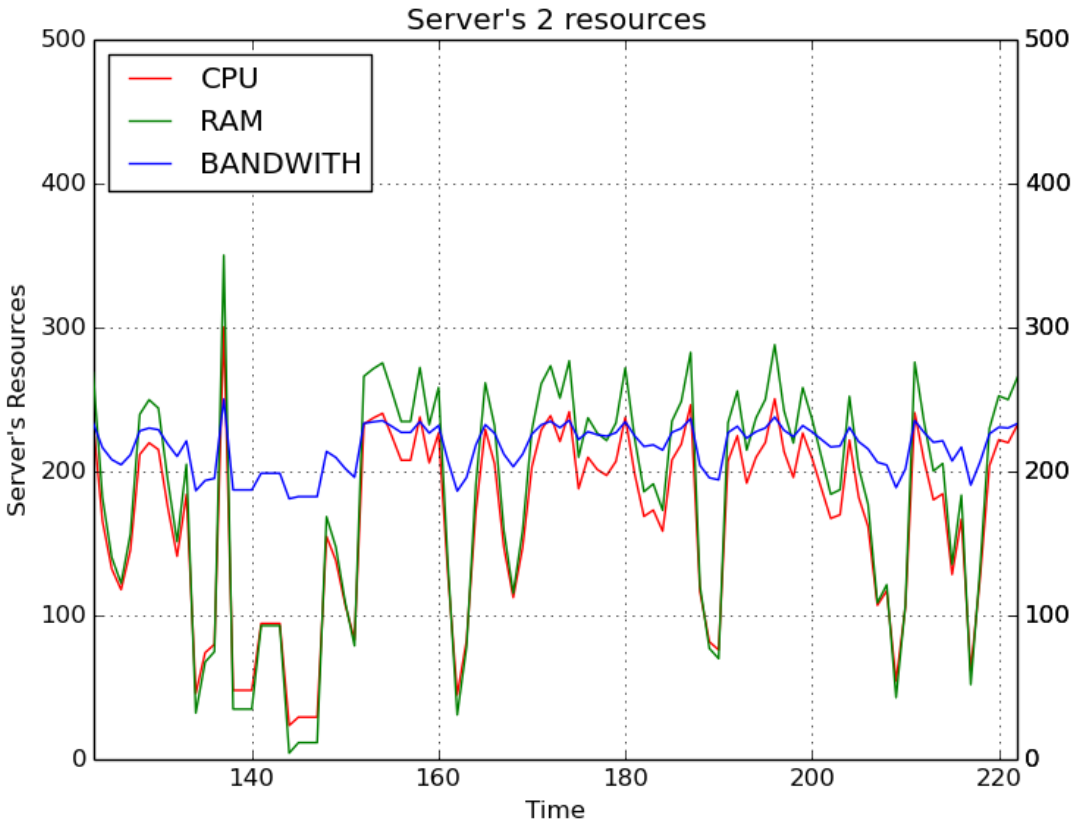


Рисунок 4.18 – Графік завантаженості ресурсів кластера «2»
при роботі алгоритму Compare Balance

При проведенні експериментів було виявлено, що найкращий результат показують динамічні алгоритми Connection mechanism і Compare Balance. При роботі алгоритму Compare Balance кількість втрачених даних є найменшою, а середній час очікування більше усього на 0,1 мс, відповідно, він забезпечує найкращу якість обслуговування для чутливого до втрат трафіка.

Дослідження показали, що дисбаланс системи суттєво залежить від мультифрактальних характеристик трафіка, як це показано в таблиці 4.2. При невеликих значеннях H та невеликій неоднорідності система балансування приходить у рівноважний стан, а значення дисбалансу прагне до нуля. При збільшенні показника Херста з часом дисбаланс системи не затухає, а система балансування не приходить до рівноважного стану. При більших значеннях показника Херста та великій неоднорідності система балансування знаходиться у нестій-

кому стані та значення дисбалансу змінюється у декілька разів, що приводить до максимального завантаження ресурсів.

Таблиця 4.2 – Зміна дисбалансу системи в залежності від параметрів мультифрактальності

Параметри мультифрактальності	Час встановлення рівноваги, сек.	Значення дисбалансу системи
H=0,6, Δh=1,5	110	0.2
H=0,6, Δh=2	130	0.2
H=0,6, Δh=4	200	0.5
H=0,6, Δh=6	310	0.58
H=0,7, Δh=1,5	110	0.3
H=0,7, Δh=2	140	0.33
H=0,7, Δh=4	220	0.55
H=0,7, Δh=6	320	0.65
H=0,8, Δh=1,5	160	0.34
H=0,8, Δh=2	180	0.4
H=0,8, Δh=4	240	0.63
H=0,8, Δh=6	400	0.7
H=0,9, Δh=1,5	170	0.4
H=0,9, Δh=2	180	0.45
H=0,9, Δh=4	240	0.7
H=0,9, Δh=6	>500	≈1

У таблиці 4.3 представлені дані використання ресурсів системи при роботі стандартного методу балансування і запропонованого динамічного методу з урахуванням фрактальних властивостей трафіку і дисбалансу ресурсів системи.

Таблиця 4.3 – Розподіл ресурсів системи

Ресурси **(**H=0,8; Δh=2**)**	Стандартний метод	Розроблений метод
$\overline{CPU}$	300	190
$\overline{RAM}$	320	170
$\overline{Net}$	280	210

Як видно з таблиці 4.3, при роботі запропонованого динамічного методу балансування використовується значно менше ресурсів системи при одному і тому ж вхідному потоці.

У таблиці 4.4 представлені значення параметрів реальної і модельної мережі до проведення експериментів і після. Як видно з таблиці, при використанні динамічного методу балансування навантаження параметри якості обслуговування мережі набагато покращилися і кількість втрачених даних зменшилася до 1,8%.

Таблиця 4.4 – Середні оцінки параметрів якості обслуговування мережі оцінки

Методи / Оцінки	Дисбаланс системи	Середнє значення втрат,%	Середній час очікування, мс
Без методів підвищення QoS	0,7	3,6	9,7
Стандартний метод балансування навантаження	0,65	2,5	7,9
Динамічний метод балансування навантаження	0,52	1,8	4,6

Як видно з таблиці 4.4 використання розробленого динамічного методу балансування навантаження значно знижує кількість втрачених даних і середній час очікування заявок в системі.

Результати експериментів принесли наступний ефект: підвищення ступеня використання ресурсів розподіленої системи, за рахунок балансування мультифрактального навантаження на менш завантажені сервери, зменшення втрат даних до 1,8% і середнього часу очікування до 4,6 мс.

4.7 Висновки з розділу 4

1. У роботі запропонований динамічний метод розподілу навантаження з урахуванням оцінювання завантаження вузлів розподіленої системи, який також враховує мультифрактальні властивості трафіку. Метод балансування навантаження, що пропонується, завдяки аналізу та обліку дисбалансу й мультифрактальних властивостей вхідного потоку забезпечує статистично рівномірний розподіл навантаження на серверах, високі показники продуктивності та пропускної здатності, а також зниження часу відгуку та кількості втрачених даних.

2. У створеному програмному продукті, що написаний на мові Python, з використанням генератора мультифрактального трафіка проведено імітаційне моделювання розроблених методів балансування навантаження у комп'ютерних мережах, за допомогою різноманітних алгоритмів балансування, яке показало можливість оптимізації управління трафіком та динамічного розподілу мережевих ресурсів.

3. У роботі проведено імітаційне моделювання запропонованого метода з використанням наступних алгоритмів балансування: Least connection, Round Robin Weight, Connection mechanism і Compare Balance. Результати моделюван-

ня показали, що мультифрактальні характеристики трафіка суттєво впливають на дисбаланс системи. При невеликих значеннях показника Херста та невеликій неоднорідності трафіка значення дисбалансу прагне до нуля і система балансування приходить до рівноважного стану. При великих значеннях показника Херста і неоднорідності система балансування постійно знаходиться у нестійкому стані, що приводить до максимального завантаження ресурсів. Використання запропонованого метода при балансуванні навантаження, з урахуванням інформації про стан серверів та всієї системи, дозволяє балансувальнику виділити сервер, що здатний найкращим чином зробити обробку мультифрактального потоку задач, що надійшов.

4. У корпоративній інформаційно-комуникативній мережі компанії «Ипра софт» було досліджено параметри якості обслуговування мережі (пропускна здатність, завантаженість каналів, кількість втрачених даних, затримки при передачі) та діапазон параметрів мережевого трафіку, що передається. Тестування показало, що трафік є мультифрактальним. Згідно з отриманими даними у розробленому програмному продукті була побудована модельна мережа з аналогічними параметрами, куди було підключено модуль генератора мультифрактального трафіку.

5. В модельній мережі було реалізовано метод динамічного балансування навантаження з урахуванням фрактальних властивостей трафіка та дисбалансу для рівномірного використання ресурсів мережі для трафіку різноманітних класів. Імітаційне моделювання показало збільшення продуктивності серверів та мережі загалом.

6. Результати проведених чисельних експериментів були реалізовані у реальній корпоративній мережі та принесли наступний ефект: збільшення ступеню використання каналів передачі даних за рахунок перенаправлення найбільш критичних інформаційних потоків на менш завантажені альтернативні канали, покращення якості обслуговування, зменшення втрат даних.

7. Результати досліджень впроваджені та використовуються у компанії «Ипра софт», що підтверджується відповідним актом впровадження (див. Додаток А).

8 Результати розділу було опубліковано у роботах [40, 55, 127, 128, 130, 133, 135-137, 146]

Список використаних джерел у даному розділі наведено у повному списку використаних джерел під номерами: [24,32,33,82,89,92,105,108,122,138,150,158].

ВИСНОВКИ

Дисертаційна робота присвячена модифікації методів балансування навантаження інформаційних потоків в розподілених комп’ютерних системах, які враховують фрактальні властивості трафіка і дозволяють забезпечити високий рівень якості обслуговування. Аналіз отриманих в дисертаційній роботі наукових і практичних результатів дозволяє зробити наступні висновки:

1. Проведено огляд і аналіз сучасного стану теорії фрактального мережевого трафіку, методів управління потоками даних і ресурсами комп'ютерних мереж і впливу фрактальних властивостей трафіку на якість обслуговування в мережі.

2. Розглянуто основні методи і моделі забезпечення якості обслуговування в комп'ютерних системах з точки зору застосування їх при балансуванні фрактального трафіка. Проведено огляд основних існуючих алгоритмів і методів балансування навантаження розподілених систем. Проведено аналіз методів балансування навантаження на різних рівнях мережевої моделі, вказані достоїнства і недоліки кожного методу.

3. Проведено класифікацію найбільш часто використовуваних алгоритмів балансування навантаження розподілених систем за різними типами. На основі проведеного аналізу вказані сфера застосування, необхідні вимоги роботи, визначено недоліки та питання, які потребують вирішення для кожного типу алгоритмів. Для кожного алгоритму вказані показники ефективності, основні особливості та сфера застосування, визначені переваги і недоліки. Таким чином можна обрати конкретний тип алгоритму балансування навантаження, виходячи із специфіки конкретного виконуваного завдання або проекту та з цілей, які планується досягти.

4. Представлені результати чисельного дослідження самоподібних властивостей адитивних потоків даних. Було показано, що при розгляді самоподібних властивостей сумарного потоку необхідно брати до уваги став-

лення коефіцієнтів варіації підсумовуваних потоків. Значення показника Херста сумарного загального потоку визначається максимальним значенням показника Херста потоків, що сумуються, і ставленням коефіцієнта варіації потоку з максимальним показником Херста до інших потоків.

5. Представлені результати чисельного дослідження зміни характеристик мультифрактального потоку при підсумовуванні з потоком, який не має мультифрактальних властивостей. Результати дослідження показали, що фрактальні характеристики мультифрактального потоку зберігаються в залежності від величини відносин сигнал/шум. Зі збільшенням відносини сигнал/шум узагальнений показник Херста сумарного потоку прагне до показника початкового мультифрактального потоку в діапазоні позитивних значень параметра. Якщо адитивний потік не має самоподібних властивостей, мультифрактальні характеристики зберігаються при меншому співвідношенні сигнал/шум.

6. Отримала подальший розвиток математична модель системи балансування навантаження, в якій балансувальник навантаження описується за допомогою системи масового обслуговування. Стани серверів описуються обсягом вільних ресурсів ЦПУ і обсягом вільної оперативної пам'яті. Всі значення параметрів моделі мають залежність від часу. Така модель дозволяє описувати поведінку розподіленої мережі в часі для різних класів обслуговування вхідного трафіку, при заданих обмеженнях на час очікування пакета в черзі і кількість втрачених пакетів.

7. Запропоновано метод розрахунку дисбалансу системи на основі оцінки завантаження вузлів розподіленої системи. В якості оцінки завантаження ресурсів вузлів запропоновані характеристики завантаження процесора, пам'яті і пропускної здатності каналу. Розраховується середнє завантаження процесора, пам'яті і пропускної здатності каналу на основі завантаження, яка виміряна системою обліку або моніторингом операційної системи Запропонований метод дозволяє проводити розрахунок завантаження для потоків різних класів обслуговування, як для кожного сервера окремо, так і для всієї системи. Введено комплексне значення дисбалансу навантаження сервера, що враховує вагові

коефіцієнти для процесора, пам'яті і пропускної здатності мережі. Введені вагові коефіцієнти дозволяють визначити значимість кожної характеристики сервера по відношенню однієї до одної. Таким чином, даний метод дозволяє обчислити дисбаланс всіх серверів системи і ефективність використання ресурсів системи.

8. Розроблено програмне забезпечення для виконання імітаційного моделювання роботи балансувальника розподіленої системи. Методами імітаційного моделювання виконано порівняння показників якості обслуговування мережі для найбільш затребуваних алгоритмів балансування навантаження. На підставі результатів досліджень запропоновано рекомендації щодо вибору алгоритмів балансування навантаження в комп'ютерних мережах з фрактальним трафіком в залежності від мережевих параметрів і характеристик.

9. На основі розробленої математичної моделі запропоновано динамічний метод розподілу навантаження, який враховує мультифрактальні властивості трафіку і задані обмеження на час очікування і кількість втрачених пакетів. Пропонований метод балансування навантаження завдяки аналізу та обліку мультифрактальнх властивостей вхідного потоку забезпечує статистично рівномірний розподіл навантаження на серверах, високі показники продуктивності і пропускної здатності, а також зниження часу відгуку і кількості втрачених даних.

10. Результати проведених чисельних експериментів були реалізовані в корпоративній мережі компанії «Ипра софт» і забезпечили наступний ефект: підвищення ступеня використання каналів передачі даних за рахунок перенаправлення найбільш критичних інформаційних потоків на менш завантажені альтернативні канали, підвищення якості обслуговування на 13%, зменшення втрат даних до 1,8%.

11. Розроблені моделі і методи впроваджені в компанії «Ипра софт», де вони використовуються для попередження перевантаження вузлів мережі за рахунок перерозподілу потоків даних. Також результати роботи впроваджені в

навчальний процес в Харківському національному університеті радіоелектроніки, що підтверджено відповідними актами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Abry P. Self-Similar Network Traffic and Performance Evaluation / P. Abry, P. Flandrin, M. S. Taqqu, D. Veitch. – New-York: John Wiley & Sons, 2000. – P. 39–88.

2. Abry P. The multiscale nature of network traffic: discovery analysis and modeling / P. Abry, R. Baraniuk, P. Flandrin // IEEE Signal Processing Magazine. – 2002. – № 4 (2). – P. 5–18.

3. Acharya H.S. The Impact of self-similarity Network traffic on quality of services (QoS) of Telecommunication Network / H.S. Acharya, S.R. Dutta, R. Bhoi // International Journal of IT Engineering and Applied Sciences Research (IJIEASR). – 2013. – Vol. 2. – pp. 54–60.

4. Angrish R. Efficient String Sorting Algorithms: Cache-aware and Cache-Oblivious / R. Angrish, D. Garg // International Journal of Soft Computing and Engineering (IJSCE). – Vol 1(2). – 2011. – P.12–16.

5. Barreto P. A Traffic Characterization Procedure for Multimedia Applications in Converged Networks / P. Barreto, P. de Carvalho, J. Soares, H. Abdalla // Júnior Proc. of the 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems: 13th IEEE International Symposium, 26-29 September 2005: Abstract Book. – Washington, USA, 2005. – P. 153–160.

6. Cao P. Implementation and Performance of Integrated Application-Controlled File Caching, Prefetching, and Disk Scheduling / P. Cao, E.W. Felten, A.R. Karlin, K. Li // ACM Transactions on Computer Systems. – Vol. 14(4). – 1996. – P. 311–343.

7. Cardellini Valeria. Dynamic Load Balancing on Web-server Systems / Valeria Cardellini, Michele Colajanni, Philip S. Yu // IEEE Internet Computing. – Vol. 3, No. 3. – 1999. – P. 28–39.

8. Cardellini V. A performance study of distributed architectures for the quality of web services / V. Cardellini, E. Casalicchio, M. Colajanni // Proceedings of the 34th Conference on System Sciences: 34th Conference, 6 January 2001: Abstract Book. – Hawaii, 2001. – Vol. 10. – pp. 213-217.

9. Carvalho de P.H.P. Analysis of the influence of self-similar traffic in the performance of real time applications / P.H P. de Carvalho, H. Abdalla Jr., A.M. Soares, P. Solís Barreto, P. Tarchetti // Department of Electrical Engineering, University of Brasilia. – 2005. – P. 480–485.

10. Carvalho, de, P.H.P. An Experimental Testbed for Evaluation Topics in Converged Networks / De Carvalho P.H.P., H. Abdalla JR., A.M. Soares, P. Solís. Barreto, P. Tarchetti, R. Lambert, G. Amvame-nze // Departamento de Engenharia Elétrica, Universidade de Brasília. Brazil. – 2005. – P. 503–509.

11. Casalicchio E. A client aware dispatching algorithm for web clusters providing multiple services / E. Casalicchio, M.Colajanni // Proceeding of the 10th International Conference on WWW: 10th International Conference, 1-5 May, 2001: Abstract Book. – Hong Kong, 2001. – pp. 535–544.

12. Casellas R. Packet Based Load Sharing Schemes in MPLS networks / Ramon Casellas, Jean Louis Rougier, Daniel Kohan // Proceedings of the 3rd international IFIP/ACM Latin American conference on Networking: 3rd international IFIP/ACM Latin American conference, April 8-10, 2002: Abstract Book. – Colmar, France. – P. 18–28.

13. Chen H. User-priority guided min-min scheduling algorithm for load balancing in cloud computing / H. Chen, F. Wang, N. Helian, G. Akanmu // National Conference Parallel Computing Technologies: National Conference, February 21-23, 2013: Abstract Book. – Bangalore, India, 2013. – P. 1–8.

14. Cheng-Zhong Xu. Iterative Methods for Dynamic Load Balancing in Multicomputers / Cheng-Zhong Xu. – Bibliolabs, LLC, 2017. – 68 p.

15. Clegg R.G. A practical guide to measuring the hurst parameter / R.G. Clegg // International Journal of Simulation. Systems, Science & Technology. – 2006. – Vol. 7. – № 2. – P. 3–14.

16. Crouse M.S. Network Traffic Modeling using a Multifractal Wavelet Model / M.S. Crouse, R.H. Riedi, V.J. Ribeiro, R.G. Baraniuk // 5-th International Symposium on Digital Signal Processing for Communication Systems DSPCS'99: 5-th International Symposium, April 1-3, 1999: Abstract Book. – Perth, 1999. – P. 609-618.

17. Crovella M. Heavy-tailed probability distribution in World Wide Web. A practical guide to heavy tails: statistical techniques and application / M. Crovella, M. Taqqu, A. Bestavros. – Boston: Department of Computer Science and Department of Mathematics, 1998. – 256 p.

18. Czarkowski M. Traffic Type Influence on Performance of OSPF QoS Routing / M. Czarkowski, S. Kaczmarek, M. Wolff // Journal of telecommunication and information technology. – 2013. – Vol. 3. – pp. 19–28.

19. Czarkowski Michał. Influence of Self-Similar Traffic Type on Performance of QoS Routing Algorithms / Michał Czarkowski, Sylwester Kaczmarek, Maciej Wolff // INTL Journal of electronics and telecommunications. – 2016. – Vol. 62, No. 1. – pp. 81–87.

20. Dhinesh Babu L.D. Honey bee behavior inspired load balancing of tasks in cloud computing environments / Dhinesh Babu L.D., P. Venkata Krishna // Applied Soft Computing. – 2013. – Vol 13(5). – P. 2292–2303.

21. Donghyuk Han. Self-Similar Traffic End-to-End Delay Minimization Multipath Routing Algorithm / Han Donghyuk, Chung Jong-Moon // IEEE Communications Letters. – 2014. –vol. 18. – pp. 2121–2124.

22. Elzeki O. Improved max-min algorithm in cloud computing / O. Elzeki, M. Reshad, M. Elsoud // International Journal of Computer Applications. – 2012 – Vol. 50(12). – P. 22–27.

23. Erl Thomas. Cloud Computing: Concepts, Technology & Architecture / Thomas Erl, Ricardo Puttini, Zaigham Mahmood. – Prentice Hall, Ed.1st, 2013. – 528 p.

24. Erl Thomas. Cloud Computing Design Patterns / Thomas Erl, Robert Cope, Amin Naserpour. – Prentice Hall, Ed.1st, 2015. – 592 p.

25. Forney Brian. Storage-Aware Caching: Revisiting Caching for Heterogeneous Storage Systems / Brian Forney, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau // Conference on file and storage technologies, Janiuary 28-30, 2002. – Monterey, California, USA. – P. 61–74.

26. Ghanbari Shamsollah. A Priority based Job Scheduling Algorithm in Cloud Computing / Shamsollah Ghanbari, Mohamed Othman // International Conference on Advances Science and Contemporary Engineering (ICASCE): International Conference, October 24-25, 2012: Abstract Book. – Jacarta, Indonesia, 2012. – V. 50. – P. 778–785.

27. Ghuge Kalyani. A Survey of Various Load Balancing Techniques and Enhanced Load Balancing Approach in Cloud Computing / Kalyani Ghuge, Minaxi Doorwar // International Journal of Emerging Technology and Advanced Engineering. –214. – Volume 4(10). – P. 410–414.

28. Grama A. Introduction to Parallel Computing, Second Edition / A. Grama, A. Gupta, G. Karypis, V. Kumar. – USA: Addison Wesley, 2003. – P. 159-212.

29. Gupta Rohit. A Survey of Proposed Job Scheduling Algorithms in Cloud Computing Environment / Rohit Gupta, Tushar Champaneria // International Journal of Advanced Research in Computer Science and Software Engineering. – 2013. Vol.3(11). – P. 782–790.

30. Hong Y.S. DNS-based load-balancing in distributed web-server systems / Y.S. Hong, J.H. No, S.Y. Kim // Proceeding, in: Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (WCCIA 2006): Fourth IEEE Workshop, April 27-28, 2006: Abstract Book. – Gyeongju, South Korea, 2006. – P. 251–254.

31. Hu Y. An optimal migration algorithm for dynamic load balancing / Y. Hu, R. Blake, D. Emerson // Concurrency: Practice and Experience. – 1998. – V. 10(6). – P. 467–483.

32. Hui Li, Workload characterization, modeling and prediction in Grid computing / Li Hui. – Thesis Universiteit Leiden, 2008. – 141 p.

33. Ignatenko E.G. The algorithm of adaptive load balancing in cluster systems / E.G. Ignatenko, V.I. Bessarab, V.V. Turupalov // Modeling and information technologies. – 2010. – № 58. – pp. 142–150. – (Kyiv: IPME G.E. Puhova NAN of Ukraine).

34. Ignatenko O. Game Theoretic Analysis of Multi-Processor Schedulers: Matrix Multiplication Example / Proceedings of the 13th International Conference on ICT in Education, Research and Industrial Applications. Integration, Harmonization and Knowledge Transfer, Kyiv, Ukraine, May 15-18, 2017. Pp.88-95.

35. Ignatenko O. “Modelling of Conflict Controlled Networks.” Proceedings of the NATO RTO Modelling and Simulation Group Symposium held in Brussels, Belgium on 15 and 16 October 2009, pp.17-1-17-10.

36. Ivanisenko Igor. Load Balancing in Cloud Services Considering the Self-Similar Properties of Incoming Flows / Igor Ivanisenko, Ludmila Kirichenko, Tamara Radivilova // Modern Problems of Radio Engineering, Telecommunications and Computer Science: XII th International Conference, February 25 – March 1, 2014: Abstract Book. – Lviv-Slavske, Ukraine. – P. 571–572.

37. Ivanisenko Igor. Methods and Algorithms of load balancing / Igor Ivanisenko // Information technologies and knowledge. – 2015. – Vol. 9, N. 4. – P. 340–375.

38. Ivanisenko Igor. Investigation of Self-Similar Properties of Additive Data Traffic / Igor Ivanisenko, Ludmila Kirichenko, Tamara Radivilova // Computer Science and Information Technologies (CSIT 2015): X th International Scientific and Technical Conference, 14–17 September 2015: Abstract Book. – Lviv, Ukraine. – pp. 169–172.

39. Ivanisenko Igor. Survey of Major Load Balancing Algorithms in Distributed System / Igor Ivanisenko, Tamara Radivilova // Інформаційні технології у інноваційному бізнесі (ITIB 2015): II Міжнар. наук.-практ. конф., 7–9 жовтня 2015: матер. конф. – Харків, Україна. – С. 89–92.

40. Ivanisenko Igor. The multifractal load balancing method / Igor Ivanisenko, Tamara Radivilova // Problems of Infocommunications Science and Technology:

Second Intern. Scien. Pract. Conf., October 13–15 2015: Abstract Book. – Kharkiv, Ukraine. – pp. 122–123.

41. Ivanisenko Igor. Using cloud storage services in decision support system / Igor Ivanisenko, Yu Kobytska // Проблеми автоматизації: Третя міжнар. наук.-техн. конф., 12–13 лист. 2015: тези доп. – Черкаси-Баку-Бельсько-Бяла-Полтава, Украина. – С. 29.

42. Jeongy, H.-D. J. A Comparative Study of Generators of Synthetic Self-Similar Teletrafic / H.-D. J. Jeongy, D. McNickle, K. Pawlikowski. – Department of Computer Science and Management, University of Canterbury, 1998. – 84-98.

43. Jiao Yang. Design and Implementation of Load Balancing of Distributed-system-based Web Server / Jiao Yang, Wang Wei // Electronic Commerce and security. – 2010. – P. 337–342.

44. Kameda Hisao. Optimal Load Balancing in Distributed Computer Systems / Hisao Kameda, Li Lie, Kim Chonggun, Zhang Yongbing. – Springer, Verlag London Limited.- London, 1997. – P. 238.

45. Kantelhardt J.W. Multifractal detrended fluctuation analysis of non-stationary time series / J.W. Kantelhardt, S.A. Zschiegner, A. Bunde, S. Havlin, E. Koscielny-Bunde, H.E. Stanley // Physica A. – 2002. – № 316. – P. 87–114.

46. Kantelhardt J. Fractal and Multifractal Time Series /J.W. Kantelhardt [Электронный ресурс]: 2008.– Режим доступа: http://arxiv.org/abs/0804.0747.

47. Kantelhardt J.W. Fractal and multifractal time series / J.W. Kantelhardt // Mathematics of complexity and dynamical systems. – 2012. – №3. – P. 463–487.

48. Kanungo Priyesh. Scheduling in Distributed Computing Environment Using Dynamic Load Balancing / Priyesh Kanungo. – Anchor Academic Publishing, 2016. – 147 p.

49. Kashyap Dharmesh. A Survey Of Various Load Balancing Algorithms In Cloud Computing / Dharmesh Kashyap, Viradiya Jaydeep // International Journal Of Scientific & Technology Research. – 2014. – Vol. 3(11). – pp. 115–119.

50. Katyal Mayanka. A Comparative Study of Load Balancing Algorithms in Cloud Computing Environment / Mayanka Katyal, Mishra Atul // International Journal of Distributed and Cloud Computing. – 2013. – Vol. 1(2). – 2013. – pp. 6–14.

51. Kaur Rajwinder. Load Balancing in Cloud Computing / Rajwinder Kaur, Luthra Pawan // Association of Computer Electronics and Electrical Engineers. – 2014. – P. 374–381.

52. Keshav S. . An Engineering Approach to Computer Networking / Keshav S. – Addison-Wesley, Reading, MA, 1997. – pp. 215–217.

53. Kirichenko L. Modeling telecommunications traffic using the stochastic multifractal cascade process / L. Kirichenko, T. Radivilova, E. Kayali; [ed. K. Markov, V. Velychko, O. Voloshin] // Problems of Computer Intellectualization. – Kiev–Sofia: ITHEA. – 2012. – pp. 55–63.

54. Kirichenko L. Mathematical simulation of self-similar network traffic with aimed parameters / L. Kirichenko, T. Radivilova, Saif Abed // Anale. Seria Informatică. – 2013. – Vol. XI fasc. 1. – pp. 17–22.

55. Kirichenko Lyudmila. Dynamic load balancing algorithm of distributed systems / Lyudmila Kirichenko, Igor Ivanisenko, Tamara Radivilova // Modern Problems of Rradio Engineering, Telecommunications and Computer Science: XIIIth Intern. Conf. TCSET'2016, February 23–26, 2016: Abstract Book. – Lviv-Slavsko, Ukraine. – pp. 515–518.

56. Kirichenko Lyudmila. Investigation of multifractal properties of additive data stream / Lyudmila Kirichenko, Igor Ivanisenko, Tamara Radivilova // 2016 IEEE First International Conference on data mining and processing: First International Conference, August 23-27, 2016: Abstract Book. – Lviv, Ukrain, 2016. – pp. 305–308.

57. Kirichenko Lyudmila. Calculation of distributed system imbalance / Lyudmila Kirichenko, Igor Ivanisenko, Tamara Radivilova // Problems of Infocommunications. Science and Technology (PICS&T-2016): Third International Scientific-Practical Conference, October 4 – 6, 2016: Abstract Book. – Kharkiv, Ukraine, 2016. – pp. 156-159.

58. Kopparapu Chandra. Load balancing servers, firewalls, and caches / Chandra Kopparapu. – Published by John Wiley & Sons, Inc., 2002. – 208 p.

59. Koteswaramma Rudra. Client-Side Load Balancing and Resource Monitoring in Cloud / Koteswaramma Rudra // International Journal of Engineering Research and Applications (IJERA). – 2012. – Vol. 2(6). – pp. 167–171.

60. Kurose J.F. Computer Networking: A Top-Down Approach / J.F. Kurose, K.W. Ross, Pearson, 2012. – P. 864 p. – (6th Edition) .

61. Kyryk Maryan. The model of evaluation quality and timing parameters service device in multiservice network / Maryan Kyryk, Volodymyr Yanyshyn // Experience of Designing and Application of CAD Systems in Microelectronics (CADSM): scientifical conference, 2013: Abstract Book.

62. Kyu-Seek, Sohn. A Distributed LSP Scheme to Reduce Spare Bandwidth Demand in MPLS Networks / Kyu-Seek Sohn, Seung Yeob Nam, Dan Keun Sung // Communications, IEEE. – 2006. – Vol. 54, Issue:7. – pp. 1277–1288.

63. Laviol Vitalij. Приборы с балансировкой нагрузки в системах сетевого мониторинга или «что такое Network Packet Broker». – 2014. – URL: http://m.habrahabr.ru/company/metrotek/blog/259633/.

64. Leland W. E. On the self-similarnature of ethernet traffic / W.E. Leland, M.S. Taqqu, W. Willinger, D.V. Wilson // IEEE/ACM Transactions of Networking. – 1994. – № 2(1). – P. 1–15.

65. Ling Zhuo. Document Replication and Distribution Algorithms for Load Balancing in Geographically Distributed Web Server Systems / Zhuo Ling. – Bibliolabs, LLC, 2017. – 72 p.

66. Liu Jing. Job Scheduling Model for Cloud Computing Based on Multi-Objective Genetic Algorithm / Jing Liu, Luo Xing-Guo, Zhang Xing-Ming, Zhang Fan, Li Bai-Nan // IJCSI International Journal of Computer Science. – 2013. – V.10(1), № 3. – pp. 134–139.

67. Lгopez, V. A Bayesian decision theory approach for thetechno-economic analysis of an all-optical router / Vгıctor Lгopez, Josгe Alberto Hernгandez, Javier Aracil, Juan P. Fernгandez Palacios and г Oscar Gonzгalez de Dios // Computer

Networks: The International Journal of Computer and Telecommunications Networking . – 2008. – Vol. 52, Issue 10. – pp. 1916–1926. (Inc. New York, NY, USA).

68. Mehta H. Decentralized content aware load balancing algorithm for distributed computing environments / H. Mehta, P. Kanungo, M. Chandwani // ICWET '11 Proceedings of the International Conference & Workshop on Emerging Trends in Technology: International Conference, February 25-26, 2011: Abstract Book. – N-Y, USA, 2011. – pp. 370–375.

69. Mendonca M. A Survey of software-defined networking: past, present, and future of programmable networks / M. Mendonca, B.A.A. Nunes, X.-N. Nguyen, K.Obraczka, T. Turletti // Communications Surveys & Tutorials, IEEE. – 2013. – Vol. 16(3). – pp. 1617–1634.

70. Meyer R.A. User Manual. Release 1.1. / Richard A. Meyer, Rajive Bagrodia Parsec. – UCLA Parallel Computing Laboratory. – 1998. – URL: pcl.cs.ucla.edu/projects/parsec.

71. Mishra Ratan. Ant colony Optimization: A Solution of Load balancing in Cloud / Mishra Ratan, Jaiswal Anant // International Journal of Web & Semantic Technology (IJWesT). – 2012. – Vol. 3, No. 2. – pp. 335–338.

72. Natario R. Load Balancing / Rui Natario. – 2011. - URL: http://networksandservers.blogspot.com/2011/03/load-balancing-iv.html.

73. Norros I. On the use of fractional Brownian motion in the theory of connectionless networks / I. Norros // IEEE Journal on Selected Areas in Communications. – 1995. – № 13(6). – pp. 953–962.

74. Park Kun I.. QoS in Packet Networks / I. Park Kun. – Springer Science & Business Media, 2006. – 243 p.

75. Pavan Illa Kumar. A Generalized Framework for Building Scalable Load Balancing Architectures in the Cloud / Illa Pavan Kumar, Subrahmanyam Kodukula // International Journal of Computer Science and Information Technologies. – 2012. – Vol. 3(1). – pp. 3015–3021.

76. Performance Tradeoffs in Static and Dynamic Load Balancing Strategies / NASA, March, 1986. – 28 p.

77. Nameer Qasim. Aggregated Self-Similar Traffic Parameters Determination Methods for EPS network planning Scholars / Qasim Nameer // Journal of Engineering and Technology. – 2014. – Vol. 2(5A). – pp. 727–732.

78. Raghava N.S. Comparative Study on Load Balancing Techniques in Cloud Computing / N. S. Raghava, Deepti Singh // Open journal of mobile computing and cloud computing. – 2014. – Vol. 1, No.1. – 2014. – P. 18–25.

79. Rajwinder Kaur. Load Balancing in Cloud Computing / Kaur Rajwinder, Luthra Pawan // Association of Computer Electronics and Electrical Engineers. – 2014. – P. 374–381.

80. Randles Martin. A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing / Martin Randles, David Lamb, A. Taleb-Bendiab // IEEE 24th International Conference on Advanced Information Networking and Applications Workshops: 24th International Conference, April 20-23, 2010: Abstract Book. – Perth, Australia, 2010. – pp. 551–556.

81. Ray Soumya. Execution analysis of load balancing algorithms in cloud computing environment / Soumya Ray, Ajanta De Sarkar // International Journal on Cloud Computing: Services and Architecture (IJCCSA). – 2012. – Vol. 2, No. 5. – 2012. – pp. 2657–2664.

82. Red Hat Enterprise Linux. Обзор планирования распределения нагрузки [Електронний ресурс] / Red Hat Enterprise Linux – Електронні дані. – [Red Hat Enterprise Linux, 2015]. - Режим доступу: https://access.redhat.com/documentation/ru-RU/Red_Hat_Enterprise_Linux/6/html/Virtual_Server_Administration/s1-lvs-scheduling-VSA.html (дата звернення 13.06.2016 р.). – Назва з екрана.

83. Riedi, R. A Multifractal Wavelet Model with Application to Network Traffic / R. Riedi, M.S. Crouse // IEEE Transactions on information theory. – 1999. – Vol. 45, № 3. – pp. 992–1018.

84. Riedi R.H. Multifractal processes / R.H.Riedi, P. Doukhan, G.Oppenheim [Taqqu M.S. (Eds.)] // Long Range Dependence: Theory and Applications: Birkhuser, 2002. – P. 625–715.

85. Roth Gregor. Server load balancing architectures, Part 1: Transport-level load balancing. – 2008. - URL: http://www.javaworld.com/article/2077921/architecture-scalability/server-load-balancing-architectures--part-1--transport-level-load-balancing.html.

86. Ryu, B.K. Fractal network traffic: from understanding to implications / B.K. Ryu // IEEE/ACM Transactions on Networking. – 2001. – № 9. – pp. 634– 649.

87. Sheluhin O.I. Similar processes in telecommunications / O.I. Sheluhin, S.M. Smolskiy, A.V. Osin. – John Wiley & Sons Ltd, England, 2007. – 310 p.

88. Singhal Priyank. Load Balancing Algorithm over a Distributed Cloud Network / Priyank Singhal, Shah Sumiran // 3rd IEEE International Conference on Machine Learning and Computing: 3rd IEEE International Conference, дата 2011: Abstract Book. – Singapore, 2011. – P. 37–42.

89. Smith I.M. Programming the Finite Element Method / I.M. Smith, D.V. Griffiths, L. Margetts. – John Wiley & Sons, 2013. – 688 p.

90. Some New Findings on the Self-Similarity Property in Communications Networks and on Statistical End-to-End Delay Guarantee: Technical Report [JNG05-01] / Department of Computer Science, Hong Kong Baptist University. – Hong Kong, 2001. – 14 p.

91. Sran Nayandeep. Comparative Analysis of Existing Load Balancing Techniques in Cloud Computing / Nayandeep Sran, Kaur Navdeep // International Journal of Engineering Science Invention. – 2013. – vol. 2. – pp. 60–68.

92. Tian Wenhong. Optimized Cloud Resource Management and Scheduling: Theories and Practices / Wenhong Tian, Zhao Yong. – Morgan Kaufman, 2014. – P. 284.

93. Tsybakov B. On self-similar traffic in ATM queues: definition, overflow probability bound, and cell delay distribution / B. Tsybakov, N.D. Georganas // IEEE/ACM Trans. on Networking. – 1997. – Vol. 5, No 3. – pp. 397–408.

94. Tuncer D. Towards decentralized and adaptive network resource management / D. Tuncer, M. Charalambides, G. Pavlou, N. Wang // Network and Service Management (CNSM), IEEE. – 2011. № 11– pp. 296–301.

95. Turnbull Malcolm. Load Balancing Methods / Malcolm Turnbull. – 2015. – URL: http://www.loadbalancer.org/blog/load-balancing-methods

96. Vargas-Rosales C. Routing with Wavelet-Based Self-Similarity Estimation / Cesar Vargas-Rosales, Luis J. Manzanero // Computación y Sistemas. – 2004. – vol. 8, n. 2. – pp. 119–131. – (Monterrey, Mexico).

97. Veitch, D. Multifractality in TCP/IP Traffic: the Case Against / D. Veitch, N. Hohn, P. Abry // Computer Networks. –2005. – Vol. 48(3). – P. 293–313.

98. Vlaeminck K. Design and implementation of an application server load balancing architecture supporting the end-to-end provisioning of value-added services / K. Vlaeminck, S. Van Hoecke, F. De Turck, B. Dhoedt, P. Demeester // Telecommunications Network Strategy and Planning Symposium: Symposium, June 13-16, 2004: Abstract Book. – Vienna, Austria, 2004. – pp. 345–350.

99. Wei Ni. A routing algorithm for Network-on-Chip with self-similar traffic / Ni Wei, Liu Zhenwei // ASICON, 2015 IEEE 11th International Conference on ASIC: IEEE 11th International Conference, 3-6 Nov. 2015: Abstract Book. – Chengdu, China, 2015. – pp. 1-4.

100. Wenhong Tian, Yong Zhao, Optimized Cloud Resource Management and Scheduling: Theories and Practices / Tian Wenhong, Zhao Yong. – Morgan Kaufman, 1 st ed., 2014. – P. 284.

101. Willinger Walter Self-Similarity through high-variability: statistical analysis of ethernet LAN traffic at the source level / Willinger Walter, Taqqu Murad S., Sherman Robert [etc.] // IEEE/ACM Transactions on Networking. – 1997. – № 5(1). – pp. 43–62.

102. Wowza Media Systems. Wowza Dynamic Load Balancing AddOn / Wowza Media Systems. – Wowza Streaming Engine, 2015. – 35 p.

103. 104. XiPeng Xiao. Technical, Commercial and Regulatory Challenges of QoS: An Internet Service Model Perspective / Xiao XiPeng. – Morgan Kaufmann. – 2008. – 296 p.

104. Yucesan Enver. Distributed Web-based Experiments for optimization / Enver Yucesan, Yah Chuyn Luo, Chun-Hung Chen [etc.] // Simulation Practice and Theory. – Vol. 9. – 2001. – pp. 73–90.

105. Zaborowski V.S. Simulation modeling of telematics systems / V.S. Zaborowski, A.S. Il'yashenko, V.A. Mulyuha // Proc., St. Petersburg: Publishing House of of the SPbSPU. – 2013. – 58 p.

106. Zheng H. Design and implementation of load balancing in web server cluster system / H. Zheng, L. Zhou, J. Wu // Journal of Nanjing University of Aeronautics & Astronautics. – 2006. – Vol. 38, No. 3. – pp. 156-162.

107. Zhenyu Na. Research on Aggregation and Propagation of Self-Similar Traffic in Satellite Network / Na Zhenyu, Liu Yi, Cui Yang [etc.] // International Journal of Hybrid Information Technology. – 2015. – Vol. 8, No. 1. – P. 325–338.

108. Zhihao Shang Design and implementation of server cluster dynamic load balancing based on OpenFlow / Shang Zhihao, Chen Wenbo, Ma Qiang [etc.] // Awareness Science and Technology and Ubi-Media Computing (iCAST-UMEDIA). – 2013. – № 3. – pp. 691–697.

109. Агеев Д.В. Методика определения параметров потоков на разных участках мультисервисной телекоммуникационной сети с учетом эффекта самоподобия / Д.В. Агеев, А.А. Игнатенко, А.Н. Копылев // Проблеми телекомунікацій. – 2011. – № 3 (5). – С. 18–37.

110. Бажин Алексей. Принципы балансировки Компании Mail.ru. / Бажин А – 2010. – URL: http://profyclub.ru/docs/21.

111. Бершадский А.М. Исследование стратегий балансировки нагрузки в системах распределенной обработки данных / А.М. Бершадский, Л.С. Курилов, А.Г. Финогеев // Известия высших учебных заведений. – 2009. – № 4 (12). –

С. 38–48. – (Поволжский регион. Технические науки. Информатика, вычислительная техника).

112. Буров А.А. Исследование влияния методов маршрутизации на качество обслуживания в мультисервисных сетях связи, функционирующих в экстремальных условиях: автореф. дис. на соискание учен. степени канд. техн. наук: специальность 05.13.12 «Системы, сети и устройства телекоммуникаций» / А.А. Буров. – Новосибирск, 2009. – 114 с.

113. Вегешна Ш. Качество обслуживания в сетях IP / Ш. Вегешна [пер. с англ.] – М.: Вильямс, 2003. – 386 с.

114. Воеводин В.В. Параллельные вычисления / В.В. Воеводин. – СПб: БХВ-Петербург, 2003. – 512 с.

115. Гайнулин А.Г. Управление ресурсами в беспроводных сетях с переменной топологией: автореф. дис. на соискание учен. степени канд. техн. наук: специальность 05.13.18 «Математическое моделирование, численные методы и комплексы программ» / А.Г. Гайнулин. – Нижний Новгород, 2009. – 117 с.

116. Галкин А.М. Исследование вероятностно-временных характеристик и протоколов построения маршрутов в сетях Metro Ethernet: автореф. дис. на соискание учен. степени канд. техн. наук: специальность 05.13.13 «Телекоммуникационные системы и компьютерные сети» / А.М. Галкин. – Санкт-Петербург, 2008. – 125 с.

117. Горбачов В.О. Суб’єктно-об’єктна модель доступу з використанням апаратних закладок до комп'ютерної інформації / В.О. Горбачов, В.В. Степаненко, І.М. Іванісенко // Радиоэлектроника и информатика. – 2006. – Том 6, №3. – С. 47–50.

118. Горбачев В.А. Классификация и формальные модели аппаратных закладных устройств / В.О. Горбачев, И.Н. Иванисенко // Прикладна радіоелектроніка та інформатика. – 2007. – Том 6, № 2. – С. 306–310.

119. Городецкий А.Я. Информатика. Фрактальные процессы в компьютерных сетях : уч. пособие / А.Я. Городецкий, В.С. Заборовский. – СПб.

: СПбГТУ, 2000. – 102 с.

120. Гуревич Григорий. Crescendo Networks – эволюция в мире WEB балансировки / Григорий Гуревич.– 2010. – URL: http://profyclub.ru/docs/99.

121. Дейнеко Ж.В. Об одном методе моделирования самоподобного стохастического процесса / Ж.В. Дейнеко, А.А. Замула, Л.О. Кириченко // Вісник Харківського національного університету ім. В.Н. Каразіна.. – 2010. – № 890. – Вип. 13. – С. 53–63. – (Сер. Математичне моделювання. Інформаційні технології. Автоматизовані системи управління).

122. Дорожкин С.К. Методы оценки загрузки вычислительных узлов распределенной вычислительной системы / С.К. Дорожкин // Научно-технический вестник информационных технологий, механики и оптики. – 2004. – Вып. 14. – С. 140–144.

123. Иванисенко И.Н. Проблемы моделирования дискретно-событийных систем // Вісник Харківського Університету. – Серія: Актуальні проблеми сучасної науки в дослідженнях молодих вчених м. Харкова. – 2002. – № 551. – С. 195–199.

124. Иванисенко И.Н. О проблеме решения задач оптимального распределения ресурсов в вычислительной системе / И.Н. Иванисенко // Проблемы информатики и моделирования: Восьмая междунар. науч.-техн. конф., 26-28 нояб.2008: матер. конф. – Харьков, Украина, 2008. – С. 53.

125. Иванисенко И.Н. Модель оптимального использования ресурсов IP-сети путем оптимизации трафика / И.Н. Иванисенко // Проблемы информатики и моделирования: Девятая междунар. науч.-техн. конф., 26-28 нояб. 2009: матер. конф. – Харьков, Украина, 2009. – С. 44.

126. Иванисенко И.Н. Методы решения задачи балансировки вычислительной нагрузки в сети распределенных вычислений / И.Н. Иванисенко // Информационные технологии в навигации и управлении: состояние и перспективы развития: Первая научно-техн. конф., 5–6 июля 2010: матер. конф. – Киев, Украина, 2010. – С. 26.

127. Иванисенко И.Н. Обзор и анализ программного обеспечения для

решения задачи балансировки нагрузки компьютерной сети / И.Н. Иванисенко // Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління: Перша наук.-техн. конф., 13–14 груд. 2010: матер. конф. – Київ, Україна, 2010. – С. 79.

128. Иванисенко И.Н. Имитационное моделирование облачных сервисов с учетом самоподобных свойств входящих потоков. Информатика, математическое моделирование, экономика / И.Н. Иванисенко, Ю.А. Кобицкая // Сб. научных статей по итогам Четвертой Междунар. науч.-практ. конф. – 23–25 апр. 2014. – Россия, Смоленск, – Том 1. – С. 79–83.

129. Иванисенко И.Н. Балансировка нагрузки в облачных сервисах с учетом самоподобных свойств входящих потоков / И.Н. Иванисенко // Радиоэлектроника и молодежь в XXI веке: Междунар. молодеж. форум, 14–16 апр. 2014: матер. форума. – Харків, Україна. – Том 5. – С. 74.

130. Иванисенко И.Н. Методы балансировки с учетом мультифрактальных свойств загрузки / И.Н. Иванисенко, Л.О. Кириченко, Т.А. Радивилова // Information content and processing. – 2015. – Vol. 2 (4). – P. 345–368.

131. Иванисенко И.Н. Анализ методов балансировки нагрузки в распределённых системах / И.Н. Иванисенко, Т.А. Радивилова // Сучасні напрямки розвитку інформаційно-комунікаційних технологій та засобів управління: П’ята міжнар. наук.-техн. конф., 23–25 квіт. 2015: матер. конф. – Полтава-Баку-Кіровоград-Харків, Україна. – С. 20–21.

132. Иванисенко И.Н. Исследование зашумленных мультифрактальных рядов / И.Н. Иванисенко, Л.О. Кириченко, А.Ю. Хабачева // Информационные системы и технологии: Междунар. науч.-техн. конф., 21–27 сент. 2015: матер. конф. – Харьков, Украина. – С. 62–63.

133. Иванисенко И.Н. Об одном методе распределения нагрузки с учетом мультифрактальных свойств трафика / И.Н. Иванисенко, Л.О. Кириченко, Т.А. Радивилова // Системні технології. – 2016. – Вип. 3(104). – С. 125–135.

134. Иванисенко И.Н. Анализ дисбаланса распределенной системы при самоподобной нагрузке / И.Н. Иванисенко, Л.О. Кириченко, Т.А. Радивилова //

Вісник Херсонського національного технічного університету. – 2016. – Вип. 3(58). – С. 224–231.

135. Иванисенко И.Н. Динамическая балансировка фрактального трафика / И.Н. Иванисенко, Л.О. Кириченко, Т.А. Радивилова // Інформаційні технології в металургії та машинобудуванні (ІТММ - 2016): Міжнар. наук.-техн. конф., 29–31 бер. 2016: матер. конф. – Дніпропетровськ, Україна. – С. 58.

136. Іванісенко І.М. Балансування навантаження з урахуванням рівня дисбалансу системи / І.М. Іванісенко // Сучасні напрямки розвитку інформаційно-комунікаційних технологій та засобів управління: П’ята міжнар. наук.-техн. конф., 21–22 квіт. 2016: матер. конф. – Полтава-Баку-Кіровоград-Харків, Україна. – С. 32.

137. Іванісенко І.М. OPEN SOURCE продукти балансування навантаження / І.М. Іванісенко // Free and Open Source Software: VII Всеукр. наук.-практ. конф., 24–27 лист. 2015: матер. конф. – Харків, Україна. – С. 85.

138. Игнатенко Е.И. Адаптивный алгоритм мониторинга загруженности сети кластера в системе балансировки нагрузки / Е.И.Игнатенко, В.И.Бессараб, И.В.Дегтяренко // Наукові праці ДонНТУ. – 2011. – Вип. 21(183). – С. 95–102.

139. Кириченко Л.О. Анализ методов повышения QOs в сетях MPLS с учетом самоподобия трафика / Л.О. Кириченко, Э. Кайали, Т.А. Радивилова // Системні технології. – 2011. – Вип. 3. – С. 52–59.

140. Кириченко Л.О. Аналіз стану розподіленої системи при самоподібному навантаженні / Л.О. Кириченко, И.Н. Иванисенко, Т.А. Радивилова // Інформаційні управляючі системи та технології: Міжнар. наук.-практ. конф., 20-22 вер. 2016: матер. конф. – Одеса, Україна. – С. 115–117.

141. Кириченко Л.О. Модели и методы оценивания параметров самоподобных и мультифрактальных стохастических процессов: дис. на соискание учёной степени доктора технических наук: 01.05.02 / Кириченко Людмила Олеговна. – Харьков, 2012. – 408 с.

142. Комиссаров А.М. Адаптивная маршрутизация в сетях передачи данных с учетом самоподобия трафика: автореф. дис. на соискание уч. степени

канд. техн. наук: спец. 05.13.12 «Системы, сети и устройства телекоммуникаций» / А.М. Комиссаров. – Уфа, 2011. – 19 с.

143. Мандельброт Б. Фрактальная геометрия природы / Б. Мандельброт. – М. : Ин-т компьютерных исследований, 2002. – 656 с.

144. Олемской А.И. Мультифрактальный анализ временных рядов / А.И. Олемской, В.Н. Борисюк, И.А. Шуда // Вісник СумДУ. – 2008. – №2. – С. 70–81. – (Серія «Фізика, математика, механіка»).

145. Петров В.В. Структура телетрафика и алгоритм обеспечения качества обслуживания при влиянии эффекта самоподобия: дис. на соискание уч. степени канд. техн. наук: 05.12.13 / Петров Виталий Валерьевич. – М., 2004. – 175 с.

146. Радивилова Т.А. Алгоритм балансировки самоподобной нагрузки / Т.А. Радивилова, И.Н. Иванисенко, Л.О. Кириченко // Современные информационные и электронные технологии: XVII междунар. науч.-практ. конф., 23–27 мая 2016: труды конф. – Одесса, Украина – С. 115–117.

147. Радивилова Т.А. Динамический метод оценки загрузки узлов распределенной системы [Електронний ресурс] / И.Н. Иванисенко, Т.А. Радивилова // Проблеми телекомунікацій. – 2016. – № 1(18). – С. 42–51. – Режим доступу до 12.11.2016.

148. Игорь Савчук. Балансировка нагрузки сервера по методу SLB. – 2012. – URL: http://blogerator.ru/page/high-load-balansirovka-nagruzki-servera-po-metodu-sticky-load-balancing.

149. Столлингс В. Современные компьютерные сети / В. Столлингс // СПб.: Питер, 2003. – 783 с.

150. Тарасов В.Н. Математические модели облачного вычислительного центра обработки данных с использованием Openflow / В.Н. Тарасов, П.Н. Полежаев, А.Е. Шухман // Вестник ОГУ. – 2012. – № 9 (145). – С. 150–155.

151. Телекоммуникационные стандарты: Recommendation ITU-T Q.3925 Traffic flow types for testing quality of service parameters on model networks: ITU 2012.

152. Урьев Г.А. Исследование фрактальных свойств потоков трафика реального времени и оценка их влияния на характекристики обслуживания телекоммуникационных сетей: автореф. дис. на соискание уч. степени канд. техн. наук : спец. 05.12.13 «Системы, сети и устройства телекоммуникаций» / Г.А. Урьев. – М., 2007. – 166 с.

153. Цыбаков, Б.С. Модель телетрафика на основе самоподобного случайного процесса / Б.С. Цыбаков // Радиотехника. – 1999. – № 5. – С. 24–31.

154. Чжоу Тао. Системы балансировки нагрузки Web-серверов [Электронный ресурс] / Тао Чжоу // Windows IT Pro. – 2000. – № 03. –Режим доступа: http://www.osp.ru/win2000/2000/03/174228/.

155. Шелухин О.И. Фрактальные процессы в телекоммуникациях: моногр. / О.И. Шелухин, А.М. Тенякшев, А.В. Осин. – М.: Радиотехника, 2003. – 480 с.

156. Шелухин О.И. Причины самоподобия телетрафика и методы оценки показателя Хёрста / О.И. Шелухин // Электротехнические комплексы и информационные системы. – 2007. – № 1. – С. 7–10.

157. Шелухин О. И. Самоподобие и фракталы. Телекоммуникационные приложения / О. И. Шелухин, А. В. Осин, С. М. Смольский. – М.: Физматлит, 2008. – 368 с.

158. Шелухин О. И. Мультифракталы. Инфокоммуникационные приложения / О. И. Шелухин. – М.: Горячая линия – Телеком, 2011. – 576 с.

ДОДАТОК А

Акти впровадження результатів дисертаційної роботи

«ЗАТВЕРДЖУЮ»
Генеральний директор ТОВ «Ипра-софт»

_____ В.Б. Романчук

«11» 01 2017 р.

АКТ

про використання результатів науково-дослідної роботи

«Методи балансування навантаження у розподілених системах з урахуванням самоподібних властивостей вхідних потоків»

Цим актом підтверджується, що розроблений старшим викладачем кафедри електронних та обчислювальних машин Харківського національного університету радіоелектроніки Іванісенком Ігорем Миколайовичем під керівництвом доктора технічних наук, професора Кіріченко Людмили Олегівни програмний продукт «Self-similarity load balancer», в рамках виконання науково-дослідної роботи на тему «Методи балансування навантаження у розподілених системах з урахуванням самоподібних властивостей вхідних потоків», використовується для балансування навантаження розподіленої комп'ютерної мережі ТОВ «Ипра-софт».

Програмний продукт «Self-similarity load balancer» виконано у повній відповідності то технічного завдання з оцінки рівня дисбалансу розподіленої системи та балансування трафіку, який має самоподібні властивості на підставі програмного продукту, який розроблено в комп'ютерній мережі ТОВ «Ипра-софт», було проведено ряд робіт, пов'язаних з моніторингом, аналізом та прогнозуванням перевантажень в мережі підприємства для забезпечення якості обслуговування, оптимального використання ресурсів мережі та поліпшення якості обслуговування. Програмний продукт «Self-similarity load balancer» забезпечує наступне:

– моніторинг вхідного навантаження із розрахунком параметрів самоподібності та мультифрактальності, які впливають на функціонування мережі;

– попередження перевантаження вузлів розподіленої системи;

– визначення найменш навантажених ресурсів системи, на які можна перерозподілити вхідні завдання;

– підвищення якості обслуговування мережі: зменшення затримок і втрат пакетів.

Проведені експериментальні дослідження розробленого програмного продукту для балансування навантаження розподіленої комп'ютерної мережі показали на його високу ефективність. Технічний ефект від впровадження програмного, інформаційного та методичного інструментаріїв для вирішення задачі балансування навантаження складається у можливості оцінки рівня дисбалансу у розподіленій системі та, після застосування програмного продукту «Self-similarity load balancer» на 17% підвищилася якість обслуговування, а кількість втрачених даних зменшилася до 1,92%.

Даний акт не є підставою для фінансових розрахунків.

Технічний директор О.Ф. Божок

«ЗАТВЕРДЖУЮ»
в.о. проректора з науково-
методичної роботи Харківського
національного університету
радіоелектроніки
_____ Рубан І.В.
«16» _____ 2017 р.

АКТ
про використання в навчальному процесі результатів дисертаційної роботи на тему: «Методи балансування навантаження у розподілених системах з урахуванням самоподібних властивостей вхідних потоків» аспіранта кафедри прикладної математики Харківського національного університету радіоелектроніки Іванісенко Ігора Миколайовича

Комісія у складі:
Голови: завідувача кафедри електронних обчислювальних машин д.т.н., проф. Міхаля О.П.
Членів комісії: начальника навчального відділу ХНУРЕ к.т.н., доц. Свид І.В., доц. кафедри електронних обчислювальних машин к.т.н., доц. Коваленка А.А.
встановила, що результати наукових досліджень реалізовано в навчальному процесі Харківського національного університету радіоелектроніки на кафедрі електроних обчислювальних машин (протокол засідання кафедри ЕОМ №7 від 06.01.2017 р.).

Розглянувши матеріали роботи та організації навчального процесу на кафедрі ЕОМ, комісія відзначає, що при проведенні лекційних занять та лабораторних робіт з курсів «Комп'ютерні мережі» та «Корпоративні комп'ютерні мережі», використані наступні результати дисертаційної роботи:
- модель системи балансування самоподібного навантаження, яка враховує мультифрактальні властивості мережевого трафіку;
- метод балансування, який враховує мультифрактальні властивості мережевого трафіку і розрахунок дисбалансу ресурсів на основі комплексного вимірювання загального рівня дисбалансу системи, та враховує пріоритети мережевих ресурсів.

Завідувач кафедри електронних обчислювальних машин О.П. Міхаль

Начальник навчального відділу І.В. Свид

Доцент кафедри електронних обчислювальних машин А.А. Коваленко

ДОДАТОК Б
СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

в яких опубліковані основні наукові результати дисертації:

1. Иванисенко И.Н. Проблемы моделирования дискретно-событийных систем // Вісник Харківського Університету. – Серія: Актуальні проблеми сучасної науки в дослідженнях молодих вчених м. Харкова. – 2002. – № 551. – С. 195–199.

2. Горбачов В.О. Суб'єктно-об'єктна модель доступу з використанням апаратних закладок до комп'ютерної інформації / В.О. Горбачов, В.В. Степаненко, І.М. Іванісенко // Радиоэлектроника и інформатика. – 2006. – Том 6, №3. – С. 47–50.

3. Горбачев В.А. Классификация и формальные модели аппаратных закладных устройств // В.А. Горбачов, И.Н. Иванисенко // Прикладна радіоелектроніка та інформатика. – 2007. – Том 6, № 2. – С. 306–310.

4. Иванисенко И.Н. Имитационное моделирование облачных сервисов с учетом самоподобных свойств входящих потоков. Информатика, математическое моделирование, экономика / И.Н. Иванисенко, Ю.А. Кобицкая // Сб. научных статей по итогам Четвертой Междунар. науч.-практ. конф. – 23–25 апр. 2014. – Россия, Смоленск, – Том 1. – С. 79–83.

5. Иванисенко И.Н. Методы балансировки с учетом мультифрактальных свойств загрузки / И.Н. Иванисенко, Л.О. Кириченко, Т.А. Радивилова // Information content and processing. – 2015. – Vol. 2 (4). – P. 345–368.

6. Ivanisenko Igor. Methods and Algorithms of load balancing / Igor Ivanisenko // Information technologies and knowledge. – 2015. – Vol. 9, N. 4. – P. 340–375.

7. Иванисенко И.Н. Об одном методе распределения нагрузки с учетом мультифрактальных свойств трафика / И.Н. Иванисенко, Л.О. Кириченко, Т.А. Радивилова // Системні технології. – 2016. – Вип. 3(104). – С. 125–135. (Входить до міжнар. наукометричних баз Index Copernicus).

8. Иванисенко И.Н. Анализ дисбаланса распределенной системы при самоподобной нагрузке / И.Н. Иванисенко, Л.О. Кириченко, Т.А. Радивилова // Вісник Херсонського національного технічного університету. – 2016. – Вип. 3(58). – С. 224–231. (Входить до міжнар. наукометричних баз: РИНЦ (eLibrary), Google Scholar).

9. Радивилова Т.А. Динамический метод оценки загрузки узлов распределенной системы [Електронний ресурс] / И.Н. Иванисенко, Т.А. Радивилова // Проблеми телекомунікацій. – 2016. – № 1(18). – С. 42–51. – Режим доступу до журн.:http://pt.journal.kh.ua/2016/1/1/161_radivilova_utilization.pdf .

які засвідчують апробацію матеріалів дисертації:

10. Иванисенко И.Н. Модель оптимального использования ресурсов IP-сети путем оптимизации трафика / И.Н. Иванисенко // Проблемы информатики и моделирования: Девятая междунар. науч.-техн. конф., 26-28 нояб. 2009: матер. конф. – Харьков, Украина, 2009. – С. 44.

11. Иванисенко И.Н. Методы решения задачи балансировки вычислительной нагрузки в сети распределенных вычислений / И.Н. Иванисенко // Информационные технологии в навигации и управлении: состояние и перспективы развития: Первая научно-техн. конф., 5–6 июля 2010: матер. конф. – Киев, Украина, 2010. – С. 26.

12. Иванисенко И.Н. Обзор и анализ программного обеспечения для решения задачи балансировки нагрузки компьютерной сети / И.Н. Иванисенко // Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління: Перша наук.-техн. конф., 13–14 груд. 2010: матер. конф. – Київ, Україна, 2010. – С. 79.

13. Ivanisenko Igor. Load Balancing in Cloud Services Considering the Self-Similar Properties of Incoming Flows / Igor Ivanisenko, Ludmila Kirichenko, Tamara Radivilova // Modern Problems of Radio Engineering, Telecommunications and Computer Science: XII th International Conference, February 25 – March 1, 2014:

Abstract Book. – Lviv-Slavske, Ukraine. – P. 571–572.

14. Иванисенко И.Н. Балансировка нагрузки в облачных сервисах с учетом самоподобных свойств входящих потоков / И.Н. Иванисенко // Радиоэлектроника и молодежь в XXI веке: Междунар. молодеж. форум, 14–16 апр. 2014: матер. форума. – Харків, Україна. – Том 5. – С. 74.

15. Ivanisenko Igor. Investigation of Self-Similar Properties of Additive Data Traffic / Igor Ivanisenko, Ludmila Kirichenko, Tamara Radivilova // Computer Science and Information Technologies (CSIT 2015): X th International Scientific and Technical Conference, 14–17 September 2015: Abstract Book. – Lviv, Ukraine. – P. 169–172. (Входить до міжнар. наукометричної бази Scopus).

16. Иванисенко И.Н. Анализ методов балансировки нагрузки в распределённых системах / И.Н. Иванисенко, Т.А. Радивилова // Сучасні напрямки розвитку інформаційно-комунікаційних технологій та засобів управління: П’ята міжнар. наук.-техн. конф., 23–25 квіт. 2015: матер. конф. – Полтава-Баку-Кіровоград-Харків, Україна. – С. 20–21.

17. Ivanisenko Igor. Survey of Major Load Balancing Algorithms in Distributed System / Igor Ivanisenko, Tamara Radivilova // Інформаційні технології у інноваційному бізнесі (ITIB 2015): II Міжнар. наук.-практ. конф., 7–9 жовтня 2015: матер. конф. – Харків, Україна. – С. 89–92. (Входить до міжнар. наукометричної бази Scopus).

18. Ivanisenko Igor. The multifractal load balancing method / Igor Ivanisenko, Tamara Radivilova // Problems of Infocommunications Science and Technology: Second Intern. Scien. Pract. Conf., October 13–15 2015: Abstract Book. – Kharkiv, Ukraine. – P. 122–123. (Входить до міжнар. наукометричної бази Scopus).

19. Иванисенко И.Н. Исследование зашумленных мультифрактальных рядов / И.Н. Иванисенко, Л.О. Кириченко, А.Ю. Хабачева // Информационные системы и технологии: Междунар. науч.-техн. конф., 21–27 сент. 2015: матер. конф. – Харьков, Украина. – С. 62–63.

20. Іванісенко І.М. OPEN SOURCE продукти балансування навантаження / І.М. Іванісенко // Free and Open Source Software: VII Всеукр. наук.-практ.

конф., 24–27 лист. 2015: матер. конф. – Харків, Україна. – С 85.

21. Иванисенко И.Н. Динамическая балансировка фрактального трафика / И.Н. Иванисенко, Л.О. Кириченко, Т.А. Радивилова // Інформаційні технології в металургії та машинобудуванні (ІТММ - 2016): Міжнар. наук.-техн. конф., 29–31 бер. 2016: матер. конф. – Дніпропетровськ, Україна. – С. 58.

22. Kirichenko Lyudmila. Dynamic load balancing algorithm of distributed systems / Lyudmila Kirichenko, Igor Ivanisenko, Tamara Radivilova // Modern Problems of Rradio Engineering, Telecommunications and Computer Science: XIIIth Intern. Conf. TCSET'2016, February 23–26, 2016: Abstract Book. – Lviv-Slavsko, Ukraine. – P. 515–518. (Входить до міжнар. наукометричної бази Scopus).

23. Іванісенко І.М. Балансування навантаження з урахуванням рівня дисбалансу системи / І.М. Іванісенко // Сучасні напрямки розвитку інформаційно-комунікаційних технологій та засобів управління: П'ята міжнар. наук.-техн. конф., 21–22 квіт. 2016: матер. конф. – Полтава-Баку-Кіровоград-Харків, Україна. – С. 32.

24. Радивилова Т.А. Алгоритм балансировки самоподобной нагрузки / Т.А. Радивилова, И.Н. Иванисенко, Л.О. Кириченко // Современные информационные и электронные технологии: XVII междунар. науч.-практ. конф., 23–27 мая 2016: труды конф. – Одесса, Украина – С. 115–117.

25. Kirichenko Lyudmila. Investigation of multifractal properties of additive data stream / Lyudmila Kirichenko, Igor Ivanisenko, Tamara Radivilova // 2016 IEEE First International Conference on Data Mining and Processing: First Intern. Conf., 23–27 August 2016: Abstract Book. – Lviv, Ukraine. – P. 305–308. (Входить до міжнар. наукометричної бази Scopus).

26. Кириченко Л.О. Аналіз стану розподіленої системи при самоподібному навантаженні / Л.О. Кириченко, И.Н. Иванисенко, Т.А. Радивилова // Інформаційні управляючі системи та технології: Міжнар. наук.-практ. конф., 20-22 вер. 2016: матер. конф. – Одеса, Україна. – С. 115–117.

27. Kirichenko Lyudmila. Calculation of distributed system imbalance /

Lyudmila Kirichenko, Igor Ivanisenko, Tamara Radivilova // Problems of Infocommunications. Science and Technology (PICS&T-2016): Third Intern. Scien.-Pract. Conf., October 4–6 2016: Abstract Book. – Kharkiv, Ukraine. – P. 156–159 (Входить до міжнар. наукометричної бази Scopus).

які додатково відображають наукові результати дисертації:

28. Иванисенко И.Н. О проблеме решения задач оптимального распределения ресурсов в вычислительной системе / И.Н. Иванисенко // Проблемы информатики и моделирования: Восьмая междунар. науч.-техн. конф., 26-28 нояб.2008: матер. конф. – Харьков, Украина, 2008. – С. 53.

29. Ivanisenko Igor. Using cloud storage services in decision support system / Igor Ivanisenko, Yu Kobytska // Проблеми автоматизації: Третя міжнар. наук.-техн. конф., 12–13 лист. 2015: тези доп. – Черкаси-Баку-Бельсько-Бяла-Полтава, Украина. – С. 29.

ДОДАТОК В

ВІДОМОСТІ ПРО АПРОБАЦІЮ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЇ

1. 8-й та 9-й міжнародній науково-технічній конференції «Проблеми інформатики і моделювання» (Харків, Україна, 2008, 2009) – очна участь.
2. 1-й та 2-й науково-технічній конференції «Інформаційні технології в навігації і управлінні: стан та перспективи розвитку» (Київ, Україна, 2010, 2011) – очна участь.
3. 1-й, 6-й та 7-й науково-технічній конференції «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління» (Київ, Україна, 2010; Полтава, Україна, 2015; Харків, Україна, 2016) – заочна участь.
4. 18-му Міжнародному молодіжному форумі «Радіоелектроніка і молодь в XXI столітті» (Харків, Україна, 2014) – очна участь.
5. Xth International Scientific and Technical Conference «Computer science and information technologies» (Lviv, Ukraine, 2015) – очна участь.
6. II Міжнародній науково-практичній конференції «Інформаційні технології у інноваційному бізнесі (ITIB 2015)», (Харків, Україна, 2015) – очна участь.
7. Second and third International Scientific Practical Conference "Problems of Infocommunications Science and Technology" (Kharkiv, Ukraine, 2015, 2016) – очна участь.
8. Міжнародній науково-технічній конференції "Інформаційні системи і технології" (Харків, Україна, 2015) – очна участь.
9. 3-й міжнародній науково-технічній конференції «Проблеми автоматизації», (Черкаси, Україна, 2015) – очна участь.
10. 7-й всеукраїнській науково-практичній конференції «Free and Open Source Software» (Харків, Україна, 2015) – очна участь.
11. Міжнародній науково-технічній конференції «Інформаційні технології в металургії та машинобудуванні (ITMM - 2016)» (Дніпропетровськ, Україна, 2016) – очна участь.

12. XIIIth International Conference TCSET'2016 «Modern problems of radio engineering, telecommunications, and computer science» (Lviv-Slavsko, 2016) – очна участь.

13. 17-й міжнародній науково-практичній конференції «Сучасні інформаційні та електронні технології» (Одеса, Україна, 2016) – очна участь.

14. 2016 IEEE First International Conference on data mining and processing. (Львів, Україна, 2016) – очна участь.

15. Міжнародній науково-практичній конференції «Інформаційні управляючі системи та технології» (Одеса, Україна, 2016) – очна участь.