УДК 004.422
UDC 004.422

DOI: 10.33744/0365-8171-2022-112-227-231

ЩОДО БАЛАНСУВАННЯ НАВАНТАЖЕННЯ В ДОДАТКАХ З ВИКОРИСТАННЯМ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ КОНТЕЙНЕРИЗАЦІЇ

CONCERNING LOAD BALANCING IN APPLICATIONS USING CONTAINERING INFORMATION TECHNOLOGY

Акімов Дмитро Дмитрович, *Національний транспортний університет, аспірант кафедри інформаційних систем і технологій,* *e-mail: std.akimov@gmail.com*

https://orcid.org/0000-0003-2440-1864

Миронов Денис Олександрович, *Національний транспортний університет, аспірант кафедри інформаційних систем і технологій,* *e-mail: mironov.link@gmail.com*

https://orcid.org/0000-0001-6020-0707

Руських Юрій Олегович*, Національний транспортний університет, аспірант кафедри інформаційних систем і технологій, e-mail:* mr.ruskih@gmail.com, тел. +380934337930

https://orcid.org/0000-0001-6718-1203

Сисоєв Ілля Костянтинович, *Національний транспортний університет, аспірант кафедри інформаційних систем і технологій, e-mail: i.sisoev.w@gmail.com,*

https:// orcid.org/0000-0003-4823-9179

Анотація. В статті розглянуто адаптивний алгоритм балансування навантаження для додатків з використанням технології контейнеризації. Наведено теоретичні приклади реалізації такого алгоритму на основі багаторівневої системи. Приведено теоретичний опис роботи алгоритму на різних рівнях системи. Функціонування додатку розгорнутого за допомогою технології контейнеризації потребує обов'язкового використання синхронізатору, який в свою чергу повинен бути наділений оптимальним алгоритмом балансування для досягнення максимального використання доступних ресурсів. При цьому слід враховувати особливість запитів які притаманні конкретному додатку, так і їх неоднорідність в часі, для цього пропонується ввести паралельну систему для обробки и статичного аналізу вхідних запитів. В статті наведені критерії які при досягненні яких таки алгоритм можна буде використовувати замість вже існуючих.

Ключові слова: контейнеризація, балансування, адаптивний алгоритм, контейнеризація, обчислювальний вузол.

Вступ. Балансування навантаження використовується для оптимізації використання паралельних обчислювальних систем для виконання паралельних обчислень, що забезпечує рівномірне навантаження на обчислювальних вузлів [1]. Якщо виникає нове завдання, програмне забезпечення,

що реалізує баланс, має вирішити, який обчислювальний блок повинен здійснювати обчислення, які пов'язані з цим новим завданням. Також балансування навантаження прогнозує притягнення частини обчислень із найбільш завантажених обчислювальних вузлів на менш навантажені.

Контейнерна програма — це набір ідентичних контейнерів, які містять екземпляри програми. Примірники розподілені між різними обчислювальними вузлами та працюють паралельно. Запити розподіляються між обчислювальними вузлами, а навантаження обчислюваних вузлів є урівноваженими [2].

Втім, при виконанні паралельних додатків можуть з'являтися суперечки між стабільним поділом об'єктів на обчислювальних вузлах і неквапливою швидкістю обміну даними між цими об'єктами. Деякі обчислювальні вузли можуть простоювати, а інші можуть бути перевантажені через поганий зв'язок між ними, не беручи до уваги те, що вартість зв'язку збалансованої системи може бути високою [3]. Тому метод балансування слід вибирати так, щоб обчислювальні вузли були поступово завантажені і мала оптимальну швидкість обміну даними між ними.

Реалізація такої паралельної комп'ютерної системи вимагає розробки алгоритмів синхронізації об'єктів, що працюють на різних вузлах комп'ютерної системи. Натомість ефективність реалізації алгоритму синхронізації залежить від балансування навантаження на вузли комп'ютерної системи.

Ось декілька причин, з-за яких виникає дисбаланс навантаження:

- неоднорідність структури паралельних додатків, тобто різні процеси запитів вимагають різної обчислювальної потужності; неоднорідна
- структура обчислювального комплексу, тобто різні вузли мають різну продуктивність;
- неоднорідність структури взаємодії між вузлами комп'ютерної мережі;

Тобто характеристики зв'язку між вузлами можуть мати різні ознаки пропускної здатності.

На сьогоднішній день не існує загального способу вирішення дисбалансу навантаження.

Методи балансування поділяються на **статичні** та **динамічні**.

Перед початком виконання паралельної програми виконується **статичне балансування**. Логічний розподіл процесу між процесорами використовує попередньо виконані навички (так звана наука, заснована на історичних даних). Проте, попереднє розміщення логічних процесів на вузлах комп'ютерної мережі може не дати очікуваних результатів [4].

Це обумовлюється тим, що:

- обчислювальна сфера, в якій здійснюється виконання програми, може змінюватися, інакше кажучи, один чи декілька обчислювальних вузлів можуть вийти з ладу;
- обчислювальний осередок, в якому виконується паралельний додаток, може бути завантажений іншими обчисленнями;
- навантаження на обчислювальні вузли є непередбаченим (аномальна активність користувачів, DDOs атаки та ін.);

Динамічне балансування є перерозподіл завантаженості на обчислювальні вузли під час реалізації програми. Для такого балансування застосовують програмне забезпечення, яке визначає:

- завантаженість обчислювальних вузлів;
- пропускну здатність ліній зв'язку серед вузлів;
- періодичність вхідних запитів;
- ресурсомісткість вхідних запитів;
- завантаженість та кількість обчислювальних вузлів.

Динамічні методи застосовують як правило тоді, коли час, потрібний на балансування, значно менше часу на виконання самого завдання [5]. Енергійне завдання балансування зазвичай містить не виключно розподіл завантаженості по обчислювальним вузлам, а й, з огляду на особливості алгоритму балансування, альтернативу оптимального числа обчислювальних вузлів. Балансування навантаження може здійснюватися програмно чи апаратно, централізовано або децентралізовано [6].

Постановка задачи. Алгоритмом динамічного балансування визначається, до якого обчислювального блоку надсилати запити під час роботи системи. Такий підхід дозволяє реагувати на зміни стану програми. Однак динамічне балансування вимагає додаткового часу для збору статистики про стан програми, аналізу даних і прийняття рішень. Балансування навантаження – це процес, який використовується для розподілу завдань між обчислювальними вузлами.

Балансування навантаження та перенесення використовуються для підвищення продуктивності паралельних систем. Через різнорідність обчислювальних середовищ алгоритм може добре працювати в одній паралельній системі, але не так добре, як мав би працювати в іншій.

У поточній тенденції розробки розподілених додатків зростають навички розгортання за допомогою хмарних середовищ, таких як (Google Cloud, Azure тощо) і контейнерних технологій. Таким чином можна збалансувати такі показники дисбалансу навантаження, як неоднорідність складної структури та неоднорідність взаємодії між вузлами. Використання технології контейнеризації дає змогу розгортати обчислювальні вузли як незалежні контейнери з однаковими обчислювальними можливостями початкової конфігурації для досягнення гомогенізації середовищ обчислювальних запитів. Усі контейнери належать до вузла синхронізатора (балансувальника навантаження), а синхронізатор реалізує алгоритм розподілу навантаження на підлеглих вузлах. При використанні контейнеризації, крім розподілу навантаження, синхронізатор може, у свою чергу, залучати нові обчислювальні вузли або вимикати невикористовувані вузли, щоб заощадити гроші. Відповідно до моделі ціноутворення постачальника хмарного середовища кожен обчислювальний блок, розгорнутий у хмарі, має певну вартість. Додатковим стимулом для максимально ефективного застосовування наявних ресурсів стає економія коштів.

Саме тому, для ефективного балансування завантаженості треба щоб алгоритм якомога більше задовольняв двом критеріям:

- Максимально ефективне застосування наявних обчислювальних вузлів.
- Оптимальний алгоритм доповнення чи вимикання цих вузлів.

Під ефективністю праці вузлів вважається їх використання на 90-95 відсотків. Тобто на нього повинно бути відправлено стільки операцій, скільки він зможе ефективно виконати не витрачаючи зайвого часу.

Оптимальний алгоритм для додавання вузлів означає додавання синхронізатором нових вузлів лише в тому випадку, якщо наявні вузли більше не можуть виконувати операції знову ж таки без втрати її часу.

Для досягнення зазначених критеріїв пропонується розробити алгоритм адаптивного балансування, який прийматиме рішення на основі комплексного показника ресурсної потужності вузла, тобто величини, що характеризує N операцій, які може виконати вузол. Певний тип не має втрат в експлуатації. Для цього визначається ресурсоємність вузлів RPS (requests per second - запитів в секунду). Також визначається RPS для кожного типу запиту.

Тип запиту визначається його URI, який є унікальним ідентифікатором для запиту та гарантує, що за правильної архітектури програми кожен запит до того самого URI використовуватиме приблизно однакову кількість ресурсів сервера. Точність формули RPS для кожного запиту безпосередньо впливатиме на продуктивність синхронізатора під час додавання або видалення обчислювальних вузлів. Алгоритм розподілу запитів від одного обчислювального вузла до іншого дуже складний.

Однак введення деяких припущень може зменшити складність цього алгоритму.

- розподілені системи поєднують кілька різних типів даних (гетерогенні);
- користувач може взаємодіяти з машиною в будь-який час;
- у повністю автоматичному режимі кількість обчислювальних вузлів у мережі контролюється лише синхронізатором без втручання користувача;
- можливі зміни комп'ютерних мереж;
- затримки на обмін даними між вузлами залишаються незмінними;

Опис багаторівневої алгоритмічної моделі. Для реалізації алгоритму пропонується реалізувати трирівневу систему прийняття рішень і збору даних.

На першому рівні пропонується дозволити синхронізатору збирати та обробляти статистику навантаження від самих вузлів, тим самим активуючи паралельний процес, який прогнозує вхідне навантаження на основі вхідного навантаження. Крім того, синхронізатор повинен оцінити вхідні запити, щоб визначити їх місткість на основі статистики, щоб вирішити, чи надсилати запит до наявного обчислювального вузла, якщо потужність вузла достатня для задоволення запиту, або до новоствореного обчислювального вузла, через базовий вхід. отримані від системи прогнозування.

На другому рівні пропонується створити систему прогнозування навантаження, яку використовує синхронізатор, надсилаючи йому паралельні вхідні запити та дані з обчислювальних

вузлів про використану потужність. В життєвому циклі додатка завантаженість в часі в основному не однорідна, а його піки неодноразово збігаються с піками активності користувачів чи завчасно запланованими піковими активностями, система прогнозування цих навантажень дасть нам перспективу не створювати додаткові обчислювальні вузли в реальному часі, а привносити їх за декілька хвилин до того як запасний обчислювальний осередок нам знадобиться.

На третьому рівні, коли створюється новий контейнер, рекомендується розгорнути додатковий проксі та обчислювальний вузол, який збиратиме дані про вхідні запити та ресурси, витрачені на їх обробку у фоновому режимі. Для того, щоб точно судити про пропускну здатність запиту, ці низи необхідно відправити в синхронізатор, набір цих низів рекомендується передавати на сторону обчислювального вузла, щоб зменшити навантаження на синхронізатор і скоротити час прийняття рішення. збір на фоновому рівні намагайтеся не займати обчислювальний вузол величезні ресурси.

Висновок. В результаті впровадження такого алгоритму очікується реалізація певних переваг, що дозволяють використовувати таку систему замість чинних методів і алгоритмів.

По-перше, ресурси, які використовуються для прийняття рішень, не повинні перевищувати ті, які необхідні для задоволення цієї вимоги.

По-друге, доступність програми має бути на рівні 99,99%, без зниження продуктивності під час додавання або завершення роботи нових обчислювальних вузлів.

По третє, існування незалучених вузлів та вузлів з частковою завантаженістю менше повинна фігурувати мінімізована.

Перелік посилань

1. HIGH AVAILABILITY FOR CLUSTER OF STATEFUL SERVICES BEHIND LOAD BALANCER IN PRIVATE/public cloud / Kent Leung, Louis Zhijun Liu, Jeremy Felix, Andrew Ossipov, Mohamed Mahmoud. – IEEE. – 2019, June 19. – 7 p.
2. A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms / Klaithem Al Nuaimi, Nader Mohamed, Mariam Al Nuaimi,Jameela AlJaroodi. – IEEE. – 2012, December 3. – 5 p.
3. Karl Matthias, Sean P. Kane. Docker: Up & Running. – O'Reilly Media, Inc. – 2015, June 11. – 232 p.
4. Alan A. A. Donovan, Brian W. Kernighan, The Go Programming Language. – Addi lWesley Professional. – 2015, November 16. – 400 p.
5. Tony Bourke, Server Load Balancing. – 1st Edition O'Reilly Media. – 2011, August 11. – 208 p.
6. Айрапетян Б.Г. РІЗНОВИДИ РОЗПОДІЛЕНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ. АРХІТЕКТУРНІ ОСОБЛИВОСТІ. ПЕРЕВАГИ ТА НЕДОЛІКИ // Системи обробки інформації, 2013, № 6. – С.199-203.

CONCERNING LOAD BALANCING IN APPLICATIONS USING CONTAINERING INFORMATION TECHNOLOGY

Akimov Dmytro D., National Transport University, postgraduate Student of the Department of Information Systems and Technologies, phone: +380672363635, e-mail: std.akimov@gmail.com, ORCID: https://orcid.org/0000-0003-2440-1864
Myronov Denys O., National Transport University, postgraduate Student of the Department of Information Systems and Technologies, phone: +380634198501, e-mail: mironov.link@gmail.com, ORCID: https://orcid.org/0000-0001-6020-0707
Ruskikh Yurii O., National Transport University, postgraduate Student of the Department of Information Systems and Technologies, phone: +380934337930, e-mail: mr.ruskih@gmail.com, ORCID: https://orcid.org/0000-0001-6718-1203
Sysoev Ilya K., National Transport University, postgraduate Student of the Department of Information Systems and Technologies, phone: +380958650637, e-mail: i.sisoev.w@gmail.com, ORCID: https://orcid.org/0000-0003-4823-9179

Abstract. The article discusses an adaptive load balancing algorithm for applications using containerization technology. Theoretical examples of the implementation of such an algorithm based on a multilevel system are given. A theoretical description of the operation of the algorithm at different levels of the system is given. The operation of an application deployed using containerization technology requires the mandatory use of a synchronizer, which must be endowed with an optimal balancing algorithm to maximize the use of available

resources. In this case, one should consider the peculiarity of requests that are inherent in a particular application, and their heterogeneity in time; for this, it is proposed to introduce a parallel system for processing and static analysis of incoming requests. The article provides criteria that, when some of the algorithms are achieved, can be used instead of the existing ones.
Key words: containerization, balancing, adaptive algorithm, containerization, computing node.

References

1. HIGH AVAILABILITY FOR CLUSTER OF STATEFUL SERVICES BEHIND LOAD BALANCER IN PRIVATE/PUBLIC CLOUD / Kent Leung, Louis Zhijun Liu, Jeremy Felix, Andrew Ossipov, Mohamed Mahmoud. – IEEE. – 2019, June 19. – 7 p.

2. A Survey of Load Balancing in Cloud Computing: Challenges and Algorithms / Klaithem Al Nuaimi, Nader Mohamed, Mariam Al Nuaimi, Jameela AlJaroodi. – IEEE. – 2012, December 3. – 5 p.

3. Karl Matthias, Sean P. Kane. Docker: Up&Running. – O'Reilly Media, Inc. – 2015, June 11. – 232 p.

4. Alan A.A. Donovan, Brian W. Kernighan. The Go Programming Language. – Addi mWesley Professional. – 2015, November 16. – 400 p.

5. Tony Bourke. Server Load Balancing. – 1st Edition O'Reilly Media. – 2011, August 11. – 208 p.

6. B.G. Hayrapetyan. VARIETIES OF DISTRIBUTED COMPUTING SYSTEMS. ARCHITECTURAL FEATURES. ADVANTAGES AND DISADVANTAGES // Information processing systems. – 2013, No. 6. – pp.199-203.